LabWindows®/CVI

User Manual

July 1996 Edition

Part Number 320681C-01

© Copyright 1994, 1996 National Instruments Corporation.
All rights reserved.

Internet Support

GPIB: gpib.support@natinst.com
DAQ: daq.support@natinst.com
VXI: vxi.support@natinst.com
LabVIEW: lv.support@natinst.com
LabWindows: lw.support@natinst.com
HiQ: hiq.support@natinst.com
VISA: visa.support@natinst.com
Lookout: lookout.support@natinst.com
FTP Site: ftp.natinst.com
Web Address: www.natinst.com

Bulletin Board Support

BBS United States: (512) 794-5422 or (800) 327-3077
BBS United Kingdom: 01635 551422
BBS France: 1 48 65 15 59

FaxBack Support

(512) 418-1111

Telephone Support (U.S.)

Tel: (512) 795-8248
Fax: (512) 794-5678

International Offices

Australia 03 9 879 9422, Austria 0662 45 79 90 0, Belgium 02 757 00 20,
Canada (Ontario) 519 622 9310, Canada (Québec) 514 694 8521, Denmark 45 76 26 00,
Finland 90 527 2321, France 1 48 14 24 24, Germany 089 741 31 30, Hong Kong 2645 3186,
Italy 02 413091, Japan 03 5472 2970, Korea 02 596 7456, Mexico 95 800 010 0793,
Netherlands 0348 433466, Norway 32 84 84 00, Singapore 2265886, Spain 91 640 0085,
Sweden 08 730 49 70, Switzerland 056 200 51 51, Taiwan 02 377 1200, U.K. 01635 523545

National Instruments Corporate Headquarters

6504 Bridge Point Parkway Austin, TX 78730-5039 Tel: (512) 794-0100

Warranty

The media on which you receive National Instruments software are warranted not to fail to execute programming instructions, due to defects in materials and workmanship, for a period of 90 days from date of shipment, as evidenced by receipts or other documentation. National Instruments will, at its option, repair or replace software media that do not execute programming instructions if National Instruments receives notice of such defects during the warranty period. National Instruments does not warrant that the operation of the software shall be uninterrupted or error free.

A Return Material Authorization (RMA) number must be obtained from the factory and clearly marked on the outside of the package before any equipment will be accepted for warranty work. National Instruments will pay the shipping costs of returning to the owner parts which are covered by warranty.

National Instruments believes that the information in this manual is accurate. The document has been carefully reviewed for technical accuracy. In the event that technical or typographical errors exist, National Instruments reserves the right to make changes to subsequent editions of this document without prior notice to holders of this edition. The reader should consult National Instruments if errors are suspected. In no event shall National Instruments be liable for any damages arising out of or related to this document or the information contained in it.

EXCEPT AS SPECIFIED HEREIN, NATIONAL INSTRUMENTS MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AND SPECIFICALLY DISCLAIMS ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. CUSTOMER'S RIGHT TO RECOVER DAMAGES CAUSED BY FAULT OR NEGLIGENCE ON THE PART OF NATIONAL INSTRUMENTS SHALL BE LIMITED TO THE AMOUNT THERETOFORE PAID BY THE CUSTOMER. NATIONAL INSTRUMENTS WILL NOT BE LIABLE FOR DAMAGES RESULTING FROM LOSS OF DATA, PROFITS, USE OF PRODUCTS, OR INCIDENTAL OR CONSEQUENTIAL DAMAGES, EVEN IF ADVISED OF THE POSSIBILITY THEREOF. This limitation of the liability of National Instruments will apply regardless of the form of action, whether in contract or tort, including negligence. Any action against National Instruments must be brought within one year after the cause of action accrues. National Instruments shall not be liable for any delay in performance due to causes beyond its reasonable control. The warranty provided herein does not cover damages, defects, malfunctions, or service failures caused by owner's failure to follow the National Instruments installation, operation, or maintenance instructions; owner's modification of the product; owner's abuse, misuse, or negligent acts; and power failure or surges, fire, flood, accident, actions of third parties, or other events outside reasonable control.

Copyright

Under the copyright laws, this publication may not be reproduced or transmitted in any form, electronic or mechanical, including photocopying, recording, storing in an information retrieval system, or translating, in whole or in part, without the prior written consent of National Instruments Corporation.

Trademarks

NI-DAQ®, NI-488.2™, and NI-488.2M™ are trademarks of National Instruments Corporation.

Product and company names listed are trademarks or trade names of their respective companies.

WARNING REGARDING MEDICAL AND CLINICAL USE OF NATIONAL INSTRUMENTS PRODUCTS

National Instruments products are not designed with components and testing intended to ensure a level of reliability suitable for use in treatment and diagnosis of humans. Applications of National Instruments products involving medical or clinical treatment can create a potential for accidental injury caused by product failure, or by errors on the part of the user or application designer. Any use or application of National Instruments products for or involving medical or clinical treatment must be performed by properly trained and qualified medical personnel, and all traditional medical safeguards, equipment, and procedures that are appropriate in the particular situation to prevent serious injury or death should always continue to be used when National Instruments products are being used. National Instruments products are NOT intended to be a substitute for any form of established process, procedure, or equipment used to monitor or safeguard human health and safety in medical or clinical treatment.

Contents

About This Manual.....xix
Organization of This Manual xix
Conventions Used in This Manual.....xx
The LabWindows/CVI Documentation Set xxi
Standard Documentation Set.....xxi
Related Documentation.....xxii
Customer Communication xxii

Chapter 1
Configuring LabWindows/CVI 1-1
LabWindows/CVI Startup Options 1-1
How to Set the Configuration Options.....1-2
Windows 95/NT 1-2
Windows 3.1.....1-2
UNIX.....1-3
Option Descriptions.....1-3
activate (UNIX Only).....1-3
Directory Options.....1-3
cfgdir (Windows 3.1 and Unix Only).....1-3
Using cfgdir in Windows 3.1 1-4
Using cfgdir under UNIX.....1-4
cvidir.....1-4
resdir [obsolete].....1-4
tmpdir 1-5
Using tmpdir in Windows 1-5
Using tmpdir under UNIX.....1-5
Font Options.....1-5
DialogFontName (Windows Only) 1-5
DialogFontSize (Windows Only).....1-5
DialogFontBold (Windows Only).....1-5
dialogFont (UNIX Only).....1-6
editorFont (UNIX Only).....1-6
menuFont (UNIX Only) 1-6
appFont (UNIX Only) 1-6
Debug Options.....1-6
DisplayCVIDebugVxDMissingMessage (Windows 3.1 Only) 1-6
CatchProtectionFaults (Windows 3.1 Only) 1-6
LoadCVIDebugVxD (Windows 3.1 only) 1-7

Chapter 2
LabWindows/CVI Overview 2-1
Components of LabWindows/CVI 2-1
Standard Libraries 2-2
User Interface Library 2-2
Data Acquisition Library and Easy I/O for DAQ Library 2-3
VISA Library 2-3
Instrument Library 2-3
LabWindows/CVI Environment 2-3
How to Create Applications with LabWindows/CVI 2-5
Creating A User Interface 2-6
Creating Standalone Programs and DLLs 2-6

Chapter 3
Project Window 3-1
Project Window Overview 3-1
File Menu 3-3
New 3-4
Open 3-5
Save 3-5
Save As 3-6
Save All 3-6
Auto Save Project 3-6
Print 3-6
Most Recently Closed Files 3-6
Exit LabWindows/CVI 3-7
Edit Menu 3-7
Add Files to Project 3-7
Exclude File from Build / Include File in Build 3-8
Remove File 3-8
Move Item Up 3-8
Move Item Down 3-9
View Menu 3-9
Show Full Path Names 3-9
Show Full Dates 3-9
Sort By Date 3-9
Sort By Name 3-9
Sort By Pathname 3-10
Sort By File Extension 3-10
No Sorting 3-10
Build Menu 3-10
Compile File 3-11
Build Project 3-11
Link Project 3-11
Update Program Files from Disk 3-11

Mark File for Compilation 3-11
Mark All for Compilation 3-12
Target (Windows 95/NT Only) 3-12
Instrument Driver Support Only 3-12
Create Standalone Executable... 3-14
Create Dynamic Link Library (Windows 95/NT Only) 3-15
Create Static Library (Windows 95/NT Only) 3-18
External Compiler Support (Windows 95/NT only) 3-19
Create Distribution Kit (Windows 3.1 and Windows 95/NT Only) 3-21
Build Information Section 3-22
File Groups Section 3-23
Main Section 3-25
Advanced Distribution Kit Options 3-25
Run Menu 3-27
Run Project 3-27
Run-Time Error Reporting 3-28
Continue 3-28
Terminate Execution 3-28
Break at First Statement 3-28
Breakpoints 3-28
Execute 3-28
Using Instrument Drivers 3-29
Instrument Driver Files 3-29
Loading/Unloading Instrument Drivers 3-31
Precedence Rules for Loading the Instrument Driver Program File 3-31
Loading an Instrument without an Instrument Program 3-32
Modules Containing Non-Instrument Functions 3-32
Modifying an Instrument Driver 3-33
Instrument Menu 3-33
Load 3-34
File Format Conversion 3-34
Unload 3-34
Edit 3-34
Accessing Function Panels from the Instruments Menu 3-36
Library Menu 3-38
User Interface 3-38
Analysis 3-38
Data Acquisition... (Windows Only) 3-39
Easy I/O for DAQ... (Windows Only) 3-39
VXI... 3-39
GPIB/GPIB 488.2... 3-39
RS-232 3-40
VISA 3-40
TCP 3-40
X Property... (UNIX Only) 3-40
DDE... (Windows only) 3-40

Formatting and I/O 3-40
Utility 3-40
ANSI C 3-41
User Libraries 3-41
Dummy .fp Files for Support Libraries 3-41
System Libraries 3-41
Window Menu 3-42
Cascade Windows 3-42
Tile Windows 3-42
Minimize All (Windows 95 only) 3-42
CloseAll 3-43
Project 3-43
Build Errors 3-43
Runtime Errors 3-43
Variables 3-43
Watch 3-43
Array/String Displays 3-44
User Interface 3-44
Function Panel 3-44
Function Tree 3-44
Help Editor 3-44
Interactive Execution 3-45
Standard Input/Output 3-45
Open Source Files 3-45
Options Menu 3-45
Compiler Options 3-46
Compiler Defines 3-48
Include Paths 3-49
Instrument Directories 3-50
Run Options 3-50
Environment 3-52
Keyboard Options... (SPARCstation Only) 3-52
Library Options 3-53
User Libraries 3-53
Dummy .fp Files for Support Libraries 3-54
National Instruments Libraries 3-54
Project Move Options 3-55
Font 3-55
Colors 3-55

Chapter 4
Source, Interactive Execution, and Standard Input/Output Windows 4-1
Source Windows 4-1
Toolbars in LabWindows/CVI 4-1
Modifying Your Toolbars 4-1

Positioning Buttons and Separators on the Toolbar 4-2
Adding and Positioning Buttons 4-2
Adding and Positioning Separators 4-2
Other Positioning Controls 4-3
Notification of External Modification (Windows Only) 4-3
Context Menus 4-3
Interactive Execution Window 4-3
Standard Input/Output Window 4-5
Using Subwindows 4-5
Selecting Text in the Source and Interactive Execution Windows 4-5
File Menu 4-7
New 4-7
Open 4-7
Open Quoted Text 4-7
Save 4-8
Save As 4-8
Save Copy As... 4-8
Close 4-8
Save All 4-8
Add File to Project 4-8
Read Only 4-8
Print... 4-9
Exit LabWindows/CVI 4-9
Edit Menu 4-9
Undo and Redo 4-10
Cut and Copy 4-10
Paste 4-10
Delete 4-11
Select All 4-11
Clear Window 4-11
Toggle Exclusion 4-11
Resolve All Excluded Lines 4-11
Insert Construct 4-11
Balance 4-12
Diff 4-12
Go To Definition 4-13
Find... 4-13
Replace... 4-16
Next File 4-17
View Menu 4-17
Line Numbers 4-18
Line Icons 4-18
Toolbar 4-18
Line 4-19
Beginning/End of Selection 4-19
Toggle Tag 4-19

Next Tag....4-19
Previous Tag....4-19
Tag Scope....4-19
Clear Tags....4-19
Function Panel History....4-20
Function Panel Tree....4-20
Recall Function Panel....4-20
Invoking the Recall Function Panel Command....4-20
Recalling a Function Panel from a Function Name Only....4-20
Multiple Panels for One Function....4-21
Multiple Functions in One Function Panel Window....4-21
Syntax Requirements for the Recall Function Panel Command....4-21
Find Function Panel....4-21
Find UI Object....4-22
Build Menu....4-22
Compile File....4-23
Build Project....4-23
Link Project....4-23
Mark File for Compilation....4-23
Clear Interactive Declarations....4-23
Insert Include Statements....4-24
Add Missing Includes....4-24
Generate Prototypes....4-24
Next Build Error....4-24
Previous Build Error....4-24
Build Errors in Next File....4-24
Run Menu....4-25
Introduction to Breakpoints and Watch Variables/Expressions....4-25
The Breakpoint State....4-26
Setting and Clearing Breakpoints....4-26
Conditional Breakpoints....4-27
Watch Variables/Expressions....4-27
Run Project / Run Interactive Statement....4-27
Running in a Source Window....4-27
Running in the Interactive Execution Window....4-27
Run-time Error Reporting....4-28
Continue....4-28
Go to Cursor....4-28
Step Over....4-28
Step Into....4-29
Finish Function....4-29
Terminate Execution....4-29
Close Libraries....4-29
Break at First Statement....4-29
Toggle Breakpoint....4-29
Breakpoints....4-30

Activate Panels When Resuming 4-31
Stack Trace... 4-32
Up Call Stack 4-32
Down Call Stack....4-32
Variable Value....4-32
Expression Value....4-32
Dynamic Memory 4-32
Instrument Menu 4-33
Library Menu....4-33
Window Menu....4-33
Options Menu....4-33
Editor Preferences... 4-34
Undo 4-34
Paste 4-34
Tabs 4-35
Line Terminator....4-35
Toolbar 4-35
Bracket Styles....4-35
Font 4-35
Colors 4-36
Syntax Coloring 4-36
User Defined Tokens for Coloring....4-36
Translate DOS LW Program... 4-36
Generate DLL Import Source (Windows 95/NT Only) 4-37
Generate DLL Import Library (Windows 95/NT Only) 4-37
Generate DLL Glue Code... (Windows 3.1 Only)....4-38
Generate DLL Glue Object... (Windows 3.1 Only) 4-38
Generate Visual Basic Include... (Windows Only) 4-38
Create Object File....4-38
Help Menu....4-39

Chapter 5
Using Function Panels 5-1
Accessing Function Panels....5-1
Multiple Function Panels in a Window....5-2
Generated Code Box 5-3
Toolbars in LabWindows/CVI....5-3
Function Panel Controls....5-3
Specifying a Return Value Control Parameter....5-4
Specifying an Input Control Parameter....5-4
Specifying a Numeric Control Parameter 5-5
Specifying a Slide Control Parameter 5-5
Specifying a Binary Control Parameter 5-5
Specifying a Ring Control Parameter....5-5
Specifying an Output Control Parameter 5-6

Using a Global Control 5-6
Common Control Function Panel.....5-6
Convenient Viewing of Function Panel Variables.....5-6
File Menu 5-7
New 5-7
Open 5-7
Close.....5-7
Save All.....5-7
Add .FP File to Project.....5-7
Add Program File to Project.....5-8
Exit LabWindows/CVI.....5-8
Code Menu 5-8
Run Function Panel 5-8
Declare Variable.....5-9
Clear Interactive Declarations 5-10
Close Libraries 5-10
Select UI Constant.....5-10
Selecting Constants from .uir Files 5-11
Selecting attribute constants from userint.h 5-12
Selecting value constants from userint.h.....5-13
Select Variable 5-14
What Is Included in the List Box 5-15
Data Type Compatibility.....5-15
Sorting of List Box Entries 5-16
Insert Function Call.....5-16
Set Target File 5-17
Variable Value.....5-17
Expression Value.....5-17
View Menu.....5-17
Toolbar 5-18
Error 5-18
Include File.....5-18
Current Tree 5-18
Function Panel History.....5-18
Find Function Panel 5-18
Previous Function Panel.....5-19
Next Function Panel Window 5-19
Previous Function Panel Window 5-19
Next Function Panel Window 5-19
First Function Panel Window.....5-19
Last Function Panel Window 5-19
Instrument Menu 5-20
Library Menu.....5-20
Window Menu.....5-20
Options Menu.....5-20
Default Control.....5-20

Default All 5-21
Toolbar 5-21
Exclude Function 5-21
Toggle Control Style 5-21
Change Format 5-21
Edit Function Panel Window 5-21
Help Menu 5-22
Control 5-22
Function 5-22

Chapter 6
Variable Display and Watch Windows 6-1
Variable Display Window 6-1
Watch Window 6-2
File Menu 6-4
New 6-5
Open 6-5
Output 6-5
Hide 6-5
Save All 6-5
Exit LabWindows/CVI 6-5
Edit Menu for the Variable Display Window 6-5
Edit Value 6-6
Find 6-6
Next Scope 6-8
Previous Scope 6-8
Edit Menu for the Watch Window 6-8
Edit Value 6-8
Add Watch Expression 6-8
Edit Watch Expression 6-8
Delete Watch Point 6-9
Find 6-9
View Menu 6-9
Expand Variable 6-9
Close Variable 6-10
Follow Pointer Chain 6-10
Retrace Pointer Chain 6-11
Go To Execution Position 6-12
Go To Definition 6-12
Array Display 6-12
String Display 6-12
Format Menu 6-13
The Run Menu 6-13
Window Menu 6-14
Options Menu 6-14

Variable Size... 6-14
Interpret As... 6-15
Estimate Number of Elements.....6-15
Add Watch Expression... 6-15

Chapter 7
Array and String Display Windows 7-1
Array Display Window 7-1
Multi-Dimensional Arrays 7-2
String Display Window 7-3
Multi-Dimensional String Array 7-3
File Menu 7-4
New 7-4
Open 7-4
Output.....7-5
Input (Array Display Only) 7-5
Close.....7-5
Save All 7-5
Exit LabWindows/CVI.....7-5
Edit Menu for the Array Display Window 7-5
Edit Value.....7-5
Find... 7-6
Goto... 7-7
Edit Menu for the String Display Window 7-7
Edit Character... 7-7
Edit Mode 7-7
Overwrite.....7-7
Find 7-8
Goto 7-8
Format Menu 7-8
Run Menu 7-9
Window Menu 7-9
Options Menu 7-9
Reset Indices 7-9
Display Entire Buffer (*String Display Only*).....7-9

Appendix A
Source Window Keyboard Commands A-1

Appendix B
Customer Communication B-1

Glossary G-1

Index I-1

Figures

Figure 1-1. Registry for Windows 95 1-2

Figure 2-1. The Project Window....2-5

Figure 3-1. The Project Window....3-2
Figure 3-2. The File Menu....3-3
Figure 3-3. The New Command....3-4
Figure 3-4. The Open Command....3-5
Figure 3-5. The Edit Menu....3-7
Figure 3-6. The Add File to Project Command....3-7
Figure 3-7. The View Menu....3-9
Figure 3-8. The Build Menu....3-10
Figure 3-9. The Create Standalone Executable Dialog Box....3-14
Figure 3-10. The Create Dynamic Link Library Dialog Box....3-16
Figure 3-11. The Create Static Library Dialog Box....3-18
Figure 3-12. External Compiler Support Dialog Box....3-19
Figure 3-13. The Create Distribution Kit Dialog Box....3-22
Figure 3-14. Advanced Distribution Kit Options dialog box....3-26
Figure 3-15. The Run Menu....3-27
Figure 3-16. The Instrument Menu....3-29
Figure 3-17. The Instrument Menu with Two Instruments Loaded....3-33
Figure 3-18. The Edit Instrument Dialog Box....3-35
Figure 3-19. The Instrument Driver Dialog Box....3-35
Figure 3-20. Select Function Panel Dialog Box....3-36
Figure 3-21. A Help Dialog Box....3-37
Figure 3-22. The Library Menu....3-38
Figure 3-23. The Window Menu....3-42
Figure 3-24. The Options Menu....3-46
Figure 3-25. The Library Options Dialog Box....3-53

Figure 4-1. The Customize Toolbar Dialog Box for the Source Window....4-2
Figure 4-2. Selecting Text Using Character Select Mode....4-6
Figure 4-3. Selecting Text Using Line Select Mode....4-6
Figure 4-4. Selecting Text Using Column Select Mode....4-6
Figure 4-5. The File Menu....4-7
Figure 4-6. The Edit Menu....4-9
Figure 4-7. The Diff Submenu....4-12
Figure 4-8. The Find Dialog Box....4-13
Figure 4-9. The Find Button Bar....4-16
Figure 4-10. The Replace Button Bar....4-16
Figure 4-11. The View Menu....4-18
Figure 4-12. The Build Menu....4-22
Figure 4-13. The Run Menu....4-25
Figure 4-14. The Breakpoints Dialog Box....4-30

Figure 4-15. The Edit Breakpoint Dialog Box....4-30
Figure 4-16. The Options Menu....4-33
Figure 4-17. Editor Preferences....4-34
Figure 4-18. The Help Menu....4-39
Figure 4-19. Keyboard Help....4-40

Figure 5-1. Instrument Driver Function Panel Window....5-2
Figure 5-2. Function Panel Controls....5-3
Figure 5-3. The File Menu....5-7
Figure 5-4. The Code Menu....5-8
Figure 5-5. The Declare Variable Dialog Box....5-9
Figure 5-6. The Select UIR Constant Dialog Box....5-11
Figure 5-7. The Select Attribute Constant Dialog Box....5-12
Figure 5-8. The Select Attribute Value Dialog Box....5-13
Figure 5-9. The Select Variable or Expression Dialog Box....5-14
Figure 5-10. The View Menu....5-17
Figure 5-11. The Options Menu....5-20
Figure 5-12. The Help Menu....5-22

Figure 6-1. The Variable Display Window....6-1
Figure 6-2. The Watch Window....6-3
Figure 6-3. The Add/Edit Watch Expression Dialog Box....6-3
Figure 6-4. The File Menu....6-4
Figure 6-5. The Edit Menu in the Variable Display Window....6-6
Figure 6-6. The Find Dialog Box in the Variable Display Window....6-6
Figure 6-7. The Find Button Bar....6-7
Figure 6-8. The Edit Menu in the Watch Window....6-8
Figure 6-9. The View Menu....6-9
Figure 6-10. A Closed Array in the Variable Display Window....6-10
Figure 6-11. An Expanded Array in the Variable Display Window....6-10
Figure 6-12. A Parent Structure Pointer in a Chain....6-11
Figure 6-13. A Child Structure Pointer in a Chain....6-11
Figure 6-14. The Format Menu....6-13
Figure 6-15. The Options Menu....6-14

Figure 7-1. The Array Display for a Double-Precision Array....7-1
Figure 7-2. The Array Display for a Three-Dimensional Array....7-2
Figure 7-3. The Reset Indices Dialog Box for a 3-Dimensional Array....7-2
Figure 7-4. The String Display for a String Variable....7-3
Figure 7-5. The File Menu....7-4
Figure 7-6. The Edit Menu for the Array Display Window....7-5
Figure 7-7. The Find Dialog Box in the Array and String Display Windows....7-6
Figure 7-8. The Find Button Bar....7-6
Figure 7-9. The Edit Menu for the String Display Window....7-7
Figure 7-10. The Format Menu....7-8

Figure 7-11. The Format Menu for a Real Array in the Array Display Window 7-8
Figure 7-12. The Options Menu.....7-9

Figure A-1. Keyboard Commands.....A-1

Tables

Table 1-1. LabWindows/CVI Startup Options 1-1
Table 1-2. Subdirectories Needed by LabWindows/CVI. 1-4

Table 3-1. Platforms Where Utility Functions Need Low-Level Support Driver 3-23
Table 3-2. VXIplug&play Framework Subdirectories 3-31
Table 3-3. Libraries in the Bin Directory of LabWindows/CVI.....3-54

Table 4-1. Regular Expression Characters 4-14
Table 4-2. Keyboard Commands for Implementing Find 4-16
Table 4-3. Keyboard Commands for Implementing Replace 4-17

About This Manual

The *LabWindows/CVI User Manual* is a reference manual that contains detailed descriptions of LabWindows/CVI features and functionality. Begin by reading Chapter 1, *Configuring LabWindows/CVI*, and Chapter 2, *LabWindows/CVI Overview*, because subsequent chapters build upon the information in the first two chapters. To use this manual effectively, you should be familiar with *Getting Started with LabWindows/CVI*, DOS, and Windows fundamentals.

Organization of This Manual

The *LabWindows/CVI User Manual* is organized as follows:

- Chapter 1, *Configuring LabWindows/CVI*, describes special options that override some of the configuration defaults established during the LabWindows/CVI installation or through the configuration dialog boxes within the environment.

- Chapter 2, *LabWindows/CVI Overview*, describes the components of LabWindows/CVI, including the LabWindows/CVI environment, and how to create applications with LabWindows/CVI.

- Chapter 3, *Project Window*, describes the LabWindows/CVI Project window, which controls specific tasks related to organizing and executing application programs.

- Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*, describes the LabWindows/CVI Source, Interactive Execution, and Standard Input/Output windows. Each of these windows supports specific tasks related to developing and executing programs.

- Chapter 5, *Using Function Panels*, describes how to use LabWindows/CVI function panels to generate code for calling functions in any of the LabWindows/CVI libraries.

- Chapter 6, *Variable Display and Watch Windows*, describes the Variable Display and Watch Windows. You use these windows to inspect and modify the values of program variables.

- Chapter 7, *Array and String Display Windows*, describes the Array and String Display windows. Use these windows to inspect and modify the contents of a single array or string during a breakpoint.

- Appendix A, *Source Window Keyboard Commands*, can help you quickly identify common Source window keyboard commands that are not in the menus.
- Appendix B, *Customer Communication*, contains a form to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation.
- The *Glossary* contains an alphabetical list of terms used in this manual and a description of each.
- The *Index* contains an alphabetical list of key terms and topics used in this manual, including the page where each one can be found.

Conventions Used in This Manual

The following conventions are used in this manual:

bold	Bold text denotes a parameter, menu item, return value, function panel item, or dialog box button or option.
italic	Italic text denotes emphasis, a cross reference, or an introduction to a key concept.
bold italic	Bold italic text denotes a note, caution, or warning.
`monospace`	Text in this font denotes text or characters that you should literally enter from the keyboard. Sections of code, programming examples, and syntax examples also appear in this font. This font also is used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, variables, filenames, and extensions, and for statements and comments taken from program code.
`italic monospace`	Italic text in this font denotes that you must supply the appropriate words or values in the place of these items.
< >	Angle brackets enclose the name of a key. A hyphen between two or more key names enclosed in angle brackets denotes that you should simultaneously press the named keys–for example, <Ctrl-Alt-Delete>.

»	The **»** symbol leads you through nested menu items and dialog box options to a final action. The sequence **File » Page Setup » Options » Substitute Fonts** directs you to pull down the **File** menu, select the **Page Setup** item, select **Options**, and finally select the **Substitute Fonts** option from the last dialog box.
paths	Paths in this manual are denoted using backslashes (\) to separate drive names, directories, and files, as in `drivename\dir1name\dir2name\myfile`

Acronyms, abbreviations, metric prefixes, mnemonics, and symbols, and terms are listed in the *Glossary*.

The LabWindows/CVI Documentation Set

Standard Documentation Set

You should begin by reading *Getting Started with LabWindows/CVI*, which gives you a hands-on introduction to LabWindows/CVI. This manual shows you how to develop applications in LabWindows/CVI.

The *LabWindows/CVI User Manual* is a reference manual that describes the features and functionality of LabWindows/CVI.

The *LabWindows/CVI Standard Libraries Reference Manual* describes the LabWindows/CVI standard libraries–the Analysis Library, the Easy I/O for DAQ Library, the Formatting and I/O Library, the GPIB/GPIB-488.2 Library, the RS-232 Library, and the Utility Library. The *LabWindows/CVI Standard Libraries Reference Manual* assumes that you are familiar with the material presented in *Getting Started with LabWindows/CVI* and the *LabWindows/CVI User Manual*.

The *LabWindows/CVI User Interface Reference Manual* describes how to create custom user interfaces with the LabWindows/CVI User Interface Library. This manual assumes that you are familiar with the material presented in *Getting Started with LabWindows/CVI*, and the *LabWindows/CVI User Manual*.

The *LabWindows/CVI Instrument Driver Developers Guide* describes how to create instrument drivers for the LabWindows/CVI Instrument Library. This manual assumes that you are familiar with the material presented in *Getting Started with LabWindows/CVI*, and the *LabWindows/CVI User Manual*.

The *LabWindows/CVI Programmer Reference Manual* contains information to help you develop programs in LabWindows/CVI. This manual assumes that you are familiar with DOS, Windows fundamentals, and with the material presented in *Getting Started with LabWindows/CVI* and the *LabWindows/CVI User Manual*.

Related Documentation

The *LabWindows/CVI Advanced Analysis Library Reference Manual* describes a library of advanced analysis functions. This manual is distributed with the optional LabWindows/CVI Advanced Analysis Library software package.

The *NI-488.2 Function Reference Manual for DOS/Windows* and the *NI-488.2M Software Reference Manual* describe functions you can use to program National Instruments GPIB interfaces. These manuals are distributed with National Instruments GPIB interface products.

The *NI-DAQ User Manual for PC Compatibles* and the *NI-DAQ Function Reference Manual for PC Compatibles* describe functions you can use to program National Instruments data acquisition boards. These manuals are distributed with National Instruments data acquisition boards.

The *NI-VXI Software Reference Manual for C* describes functions you can use to program National Instruments VXI controllers. This manual is distributed with National Instruments VXI controllers for LabWindows/CVI VXI Development System users.

Customer Communication

National Instruments wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help you if you have problems with them. To make it easy for you to contact us, this manual contains comment and technical support forms for you to complete. These forms are in Appendix B, *Customer Communication*, at the end of this manual.

Chapter 1
Configuring LabWindows/CVI

This chapter describes special options that override some of the configuration defaults established during the LabWindows/CVI installation or through the configuration dialog boxes within the environment. These options inform LabWindows/CVI where to find system files, where to place temporary files, and set the amount of memory for LabWindows/CVI to use. You might not need to set any of these options.

Getting Started with LabWindows/CVI contains installation instructions for LabWindows/CVI, as well as a hands-on tutorial. You should be familiar with the material in *Getting Started with LabWindows/CVI* before reading this manual.

LabWindows/CVI Startup Options

You can append certain options to the `cvi` command line, separating various parameters by spaces. The valid startup options appear in Table 1.1.

Table 1-1. LabWindows/CVI Startup Options

Option	**Purpose**
`<filename>`	LabWindows/CVI automatically loads the file at startup. The file can be any of the types available under the **File** » **Open** command in LabWindows/CVI.
`-run`	This option automatically invokes the **Run Project** command from the **Run** menu of LabWindows/CVI.
`-run_then_exit`	This option automatically invokes the Run Project command from the **Run** menu and then automatically invokes the **Exit LabWindows/CVI** command from the **File** menu when the project is terminated. This option also suppresses the LabWindows/CVI startup screen and Project window.

How to Set the Configuration Options

Windows 95/NT

Windows 95 and NT configuration options are set in the Registry under the following key.

`HKEY_LOCAL_MACHINE\Software\National Instruments\CVI`

There is a configuration string value associated with each option, as in the following figure.

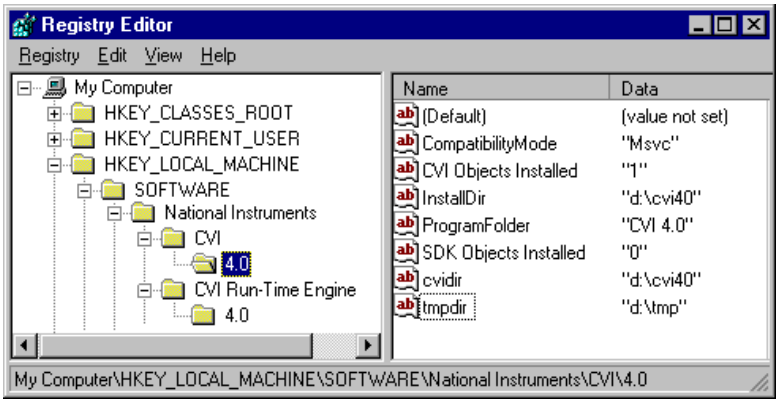

Figure 1-1. Registry for Windows 95

You need not include an unused configuration string in the Registry.

You must specify an absolute path name, including a drive letter, for configuration strings that take a directory name.

Windows 3.1

Windows 3.1 configuration options are set in the `win.ini` file. There is a configuration string associated with each option, as in the following example.

```
[cvi]
cvidir=
cfgdir=c:\localdir
tmpdir=c:\mytmpdir
CatchProtectionFaults=yes
LoadCVIDebugVxD=yes
```

You need not include an unused configuration string in `win.ini`. Leaving the space to the right of the equals sign (`=`) empty is equivalent to not including the configuration string.

You must specify an absolute path name, including a drive letter, for configuration strings that take a directory name.

UNIX

Configuration options under UNIX are set in the `.Xdefaults` file. The options do not take effect until you restart your X server, or until you use the `xrdb` command to load your `.Xdefaults` file.

There is a configuration string associated with each option, as in the following example.

```
cvi.cvidir:
cvi.cfgdir:/home/myhomedir/localdir
cvi.tmpdir:/home/myhomedir/mytmpdir
```

You need not include an unused configuration string in the `.Xdefaults` file. Leaving the space to the right of the colon (`:`) empty is equivalent to not including the configuration string.

You must specify an absolute path name for configuration strings that take a directory name.

Option Descriptions

You have the following options when configuring LabWindows/CVI.

activate (UNIX Only)

When this option is set to `True` (`cvi.activate: True`) the input focus is set to a window when the user brings it to the front.

The default value is `False`. Use this option only when `ClickToType` is in effect.

Directory Options

cfgdir (Windows 3.1 and Unix Only)

`cfgdir` sets the location for the LabWindows/CVI configuration file (`cvi.ini` in Windows 3.1, `.cvi.ini` under UNIX), which contains the final settings from the previous session of LabWindows/CVI.

In Windows 95 and NT, `cvi.ini` is not used. Instead, the current state of the options that you can set within the environment are stored in the Registry under the following key.

`HKEY_CURRENT_USER\Software\National Instruments\CVI`

Using cfgdir in Windows 3.1

If you do not specify a directory, LabWindows/CVI assumes that the installation directory (which can be specified with the `cvidir` option) contains `cvi.ini`.

If you are running LabWindows/CVI across a network, you must set `cfgdir` to one of your local directories.

Using cfgdir under UNIX

If you do not specify a directory, LabWindows/CVI assumes that your home directory contains the configuration file `.cvi.ini`.

cvidir

You should set the `cvidir` option only if the LabWindows/CVI executable (`cvi.exe` in Windows, `cvi` under UNIX) resides in a directory other than the installation directory. The `cvidir` option specifies the directory that contains the subdirectories needed by LabWindows/CVI, as shown in Table 1-2.

Table 1-2. Subdirectories Needed by LabWindows/CVI.

Name	**Contents**
`bin`	Resource files (`cvi.rsc`, `cvimsgs.txt`), National Instruments Function Panels (`.lfp` files), National Instruments Libraries (`.obj` and `.lib` in Windows, `.o` and `.a` under UNIX).
`font`	Font description files.
`include`	C header files for National Instruments libraries.
`sdk`	(Windows 95 and NT only) Windows SDK.
`hyperhelp`	(UNIX only) HyperHelp viewer and associated files.

If you do not specify a directory, LabWindows/CVI assumes that these directories are in the directory containing the executable file (`cvi.exe` in Windows, `cvi` under UNIX).

resdir [obsolete]

The `resdir` option formerly used by LabWindows/CVI for Windows 3.1 is obsolete. If you need to place the LabWindows/CVI executable in a directory other than the installation directory, use the `cvidir` option.

tmpdir

`tmpdir` sets the location for temporary files.

Using tmpdir in Windows

If you do not specify a directory, LabWindows/CVI uses the value of the environment variable `TMP`. If the value of `TMP` is not defined or is invalid, LabWindows/CVI uses the value of the environment variable `TEMP`. If the value of `TEMP` is not defined or is invalid, LabWindows/CVI uses the directory containing `cvi.exe`.

If you are running LabWindows/CVI across a network, you must set `tmpdir` to one of your local directories.

Using tmpdir under UNIX

If you do not specify a directory, LabWindows/CVI uses the value of the environment variable `TMPDIR`. If `TMPDIR` is not defined or is invalid, LabWindows/CVI uses `/tmp` as the location for temporary files.

Font Options

DialogFontName (Windows Only)

You can specify the font to use in the LabWindows/CVI dialog boxes and the built-in pop-up panels as in the following example.

```
DialogFontName=Courier
```

DialogFontSize (Windows Only)

You can specify the font size to use in the LabWindows/CVI dialog boxes and the built-in popup panels as in the following example.

```
DialogFontSize=30
```

DialogFontBold (Windows Only)

You can specify whether the font used in the LabWindows/CVI dialog boxes and the built-in pop-up panels is bold, as in the following example.

```
DialogFontBold=Yes
```

dialogFont (UNIX Only)

This option specifies the font NIDialog, which LabWindows/CVI uses in dialog boxes and built-in pop-up panels. The default is `cvi.dialogFont`: `adobe-helvetica.`

editorFont (UNIX Only)

This option specifies the font NIEditor, which LabWindows/CVI uses in the Source window. This font must have a fixed width. The default is `cvi.editorFont`: `adobe-courier.`

menuFont (UNIX Only)

This option, formerly called systemFont, specifies the font NIMenu, which LabWindows/CVI uses in menus. The default is `cvi.menuFont`: `adobe-helvetica`.

appFont (UNIX Only)

This option specifies the font NIApp, which LabWindows/CVI uses in the Project window. The default is `cvi.appFont: adobe-helvetica`.

Debug Options

DisplayCVIDebugVxDMissingMessage (Windows 3.1 Only)

By default, if `LoadCVIDebugVxD` is set to yes or is not in the [`cvi`] section of `win.ini` and you do not have the following line in the [`386Enh`] section of `system.ini`, you will see a `missing cvidebug.386` message when you start LabWindows/CVI.

```
device=CVIDebug.386
```

However, if the following line is present in the [`cvi`] section of `win.ini`, the message will not be displayed.

```
DisplayCVIDebugVxDMissingMessage=no
```

When a value is not specified in the `win.ini` file, the default value is `yes`.

CatchProtectionFaults (Windows 3.1 Only)

`CatchProtectionFaults` determines whether LabWindows/CVI traps General Protection Faults while LabWindows/CVI is running. Set this value to `no` if you do not want LabWindows/CVI to catch protection faults. If this value is set to `no`, using the Variable Display may cause crashes in LabWindows/CVI. If this entry is not present, it defaults to `yes`.

When a value is not specified in the `win.ini` file, the default is `yes`.

LoadCVIDebugVxD (Windows 3.1 only)

`LoadCVIDebugVxD` determines whether LabWindows/CVI looks for `cvidebug.386`. If this entry is set to `no`, LabWindows/CVI does not catch General Protection Faults and the <Ctrl-Alt-SysRq> key combination does not interrupt a running program. If this entry is not present, it defaults to `yes`.

Chapter 2
LabWindows/CVI Overview

This chapter describes the components of LabWindows/CVI, including the LabWindows/CVI environment, and how to create applications with LabWindows/CVI.

Components of LabWindows/CVI

LabWindows/CVI is a programming environment for developing instrument control, automated test, and data acquisition applications in ANSI C. LabWindows/CVI has the following components.

- Standard libraries and interactive function panels for:
  - GPIB
  - RS-232
  - VISA
  - Data acquisition (distributed with National Instruments PC-based data acquisition boards)
  - Data analysis
  - Transport Control Protocol (TCP)
  - Using X Client Properties for interprocess communication
  - Windows Dynamic Data Exchange (DDE) communication
  - File I/O
  - Data formatting
  - ANSI C
- A graphical user interface editor, CodeBuilder, and library for building, displaying, and controlling a graphical user interface.
- A set of instrument drivers containing high-level functions and interactive function panels for controlling specific instruments.
- A development environment with windows to manage projects and source code with complete editing, debugging, and user protection features.

Two additional libraries, the VXI Library and the Advanced Analysis Library, are available for LabWindows/CVI. These libraries are optional packages that you can order from National Instruments.

Standard Libraries

The standard LabWindows/CVI libraries are as follows.

- User Interface Library
- Analysis Library
- GPIB-488/488.2 Library
- RS-232 Library
- Easy I/O for DAQ Library
- VISA Library
- TCP Library
- X Property Library (UNIX Only)
- DDE Library (Windows Only)
- Formatting and I/O Library
- Utility Library
- ANSI C Library

The functions that make up these libraries can be executed in the LabWindows/CVI environment. You can find descriptions of these library functions in the following manuals.

- *LabWindows/CVI Standard Libraries Reference Manual*
- *LabWindows/CVI User Interface Reference Manual*
- *NI-488.2 Software Reference Manual*
- *NI-VISA User Manual* (available upon request)
- *NI-VISA Programmer Reference Manual* (available upon request)

User Interface Library

You can use the User Interface Library in conjunction with the User Interface Editor in the LabWindows/CVI environment. In the User Interface Editor, you can create command bars,

pull-down menus, dialog boxes, controls, graphs, and strip charts. You then can save these objects to a User Interface Resource (`.uir`) file. The functions in the User Interface Library allow you to load these objects from the `.uir` file, display them, receive user input from them, and display program data and results on them. The User Interface Library also has functions for programmatic creation of a Graphical User Interface. The User Interface Library and Editor are described in detail in the *LabWindows/CVI User Interface Reference Manual*.

Data Acquisition Library and Easy I/O for DAQ Library

The Data Acquisition Library, which comes with National Instruments data acquisition boards, contains high-level functions for controlling National Instruments plug-in data acquisition boards. The library functions are described in the *NI-DAQ Function Reference Manual for PC Compatibles*, which is also distributed with National Instruments data acquisition boards.

The Easy I/O for DAQ Library contains functions which make writing simple DAQ programs easier than if you use the Data Acquisition Library. Although the function panels for the Easy I/O for DAQ Library come with LabWindows/CVI, the library requires the NI-DAQ DLL, which comes with your National Instruments data acquisition board. The library functions are described in Chapter 10 of the *LabWindows/CVI Standard Libraries Reference Manual*.

VISA Library

The VISA (Virtual Instrument Software Architecture) Library gives VXI and GPIB software developers, particularly instrument driver developers, a single interface library for controlling VXI, GPIB, RS-232, and other types of instruments. The functions are described in the *NI-VISA Programmer Reference Manual*, which is available upon request.

Instrument Library

The Instrument Library is a set of instrument drivers, each containing high-level C functions for controlling a specific GPIB, RS-232, or VXI instrument. The low-level steps needed to control the instrument and read data are encapsulated in the high-level functions. Instrument drivers are loaded into the environment and used in the same way as the other LabWindows/CVI libraries.

LabWindows/CVI Environment

The LabWindows/CVI environment makes it easy for you to create and test applications that use the LabWindows/CVI libraries. The environment is a combination editor, compiler, and debugger with extensive run-time checking. A special feature called a *function panel* makes the task of developing programs much easier. Using a function panel, you can execute a LabWindows/CVI library function interactively, and generate code that calls the function. Function panels also contain online help information for the functions and function parameters.

You can build, execute, test, and debug the source code for your application in the LabWindows/CVI environment.

The LabWindows/CVI environment also has a User Interface Editor for creating a graphical user interface for your application programs. The user interface is controlled by functions in the User Interface Library.

You can also use the LabWindows/CVI environment to create instrument drivers.

The LabWindows/CVI environment has the following different windows, each with its own menu bar.

- The *Project* window, which appears when you start LabWindows/CVI. This window is used to open, edit, build, run, and save application project (`.prj`) files. A project file is a list of files used by your application. Certain files are required to be in the list while others are optional. The project window is described in detail in Chapter 3, *Project Window*.

- *Source* windows, used to create, edit, run, debug, and save source code. This window includes an optional toolbar to give you quick access to commands you use frequently. Source windows are described in detail in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*.

- *Function Panel* windows, used to interactively execute library functions and insert code into Source windows. This window includes an optional toolbar to give you quick access to commands you use frequently. Function Panel windows are described in Chapter 5, *Using Function Panel Windows*.

- The *Interactive Execution* window, used to execute selected portions of code. You do not need to have a complete program in the Interactive Execution window, as is the case in a Source window. For instance, you can execute variable declarations and assignment statements in C without declaring a main function. This window is described in detail in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*.

- *Variable Display*, *Array Display*, *String Display*, and *Watch* windows, used for debugging programs. These windows are described in Chapter 6, *Variable Display and Watch Windows*, and Chapter 7, *Array and String Display Windows*.

- *User Interface Editor* windows, used to build graphics-mode command bars, pull-down menus, dialog boxes, controls, graphs, and strip charts and save them to User Interface Resource (`.uir`) files. The User Interface Editor is described in the *LabWindows/CVI User Interface Reference Manual*.

- *Function Tree Editor* windows, used to build the tree structure of function panel files. Function Tree Editor windows are described in the *LabWindows/CVI Instrument Driver Developers Guide*.

- *Function Panel Editor* windows, used to build function panels. This window includes an optional toolbar to give you quick access to commands you use frequently. Function Panel Editor windows are described in the *LabWindows/CVI Instrument Driver Developers Guide*.

- *Function Tree Help Editor and Function Panel Help Editor* windows, used to add online help to function panels. These windows are described in the *LabWindows/CVI Instrument Driver Developers Guide*.

- The *Standard Input/Output* window, used for printing text messages and receiving user input from the keyboard. This window is described in detail in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*.

You develop applications in the LabWindows/CVI environment using the ANSI C programming language. See the *LabWindows/CVI Programmer Reference Manual* for a discussion of compiler, loadable compiled module, and information on developing applications for multiple platforms.

How to Create Applications with LabWindows/CVI

You use LabWindows/CVI as a text editor in which you enter your entire program. You can greatly simplify application development by using function panels to execute LabWindows/CVI functions and automatically insert the code into your program. Function panels contain complete online help. See Chapter 5, *Using Function Panels*, for more details.

The Project window contains all the component files of your application. The simplest case would be one source file as shown in Figure 2-1.

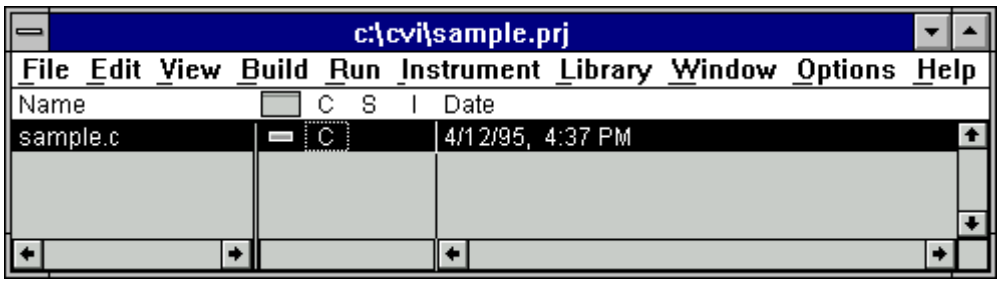

Figure 2-1. The Project Window

A typical project, however, contains multiple code modules and a User Interface Resource file. Code modules may be listed as source files or compiled files. Source files must be recompiled each time the project is opened thereby increasing initial project start-up time. However, source files can be debugged and can use run-time error checking.

Compiled files, such as library or object files, must have been previously compiled with LabWindows/CVI or a compatible external compiler. (See the *LabWindows/CVI Programmer*

Reference Manual for more information on compatible external compilers.) Compiled files are not recompiled each time the project is opened, reducing initial project start-up time. Additionally, compiled files consume less memory and run faster than source files. However, they cannot be debugged and they do not have run-time error checking.

You can strike a balance between initial project start-up time, execution speed, memory consumption, and the ability to debug code modules by varying the types of code modules listed in your project.

Creating A User Interface

You can create user interface objects (panels, controls, menus) using the User Interface Editor window. You can save these objects in a `.uir` file. You can load, display and modify these objects in your program using the functions in the User Interface Library. You can also specify callback functions which are called when events occur on these objects.

The LabWindows/CVI CodeBuilder automatically generates complete C code that compiles and runs based on a user interface (`.uir`) file you are creating or editing. By choosing certain options presented to you in the **Code** menu, you can produce *skeleton code*. Skeleton code is syntactically and programmatically correct code that compiles and runs before you have typed a single line of code. With the Code Builder feature, you save the time of typing in standard code included in every program, eliminate syntax and typing errors, and maintain an organized source code file with a consistent programming style. For more information, refer to the *CodeBuilder Overview* section in Chapter 2 of the *LabWindows/CVI User Interface Reference Manual*.

Creating Standalone Programs and DLLs

With the LabWindows/CVI Run-Time System, you can create standalone executables. Chapter 7, *Creating and Distributing Standalone Executables and DLLs*, in the *LabWindows/CVI Programmer Reference Manual* describes the system.

Chapter 3
Project Window

This chapter describes the LabWindows/CVI Project window, which controls specific tasks related to organizing and executing application programs.

Project Window Overview

You use the Project window to open, edit, build, run, and save application project (`.prj`) files. A project file is a list of files your application uses. Certain files must be in the list, while others are optional. If you had a project loaded the last time you used LabWindows/CVI, that project appears in the Project window when you start LabWindows/CVI again.

Unless they are used as instrument driver program files or they are loaded dynamically using `LoadExternalModule`, the following files are required in your project file list.

- Source files used by your application program, ending with the `.c` extension.
- Object files used by your application program, ending with `.obj` under Microsoft Windows or `.o` under UNIX.
- Library files used by your application program, ending with `.lib` under Microsoft Windows or `.a` under UNIX. DLL import libraries for Windows 95 and NT are included in this category.
- Dynamic-link library files used by your application program (under Microsoft Windows 3.1 only), ending with the `.dll` extension.

The following files are optional in your project file list.

- Header files (`.h`) used by your application program. Listing `.h` files makes it easy to open them for viewing or editing, and ensures that the co-mpiler will find them.
- User interface resource files (`.uir`) used by your application program. Listing `.uir` files makes it easy to open them for viewing or editing, and ensures that the compiler will find them.
- Instrument driver function panel files (`.fp`). Listing `.fp` files lets LabWindows/CVI automatically load instruments when the project is opened.
- Instrument driver program files. Listing these files overrides the loading precedence for instrument driver program files. See the *Using Instrument Drivers* and *The Instrument Menu* sections in this chapter for information about instrument driver program files.

A sample Project window is shown in Figure 3-1.

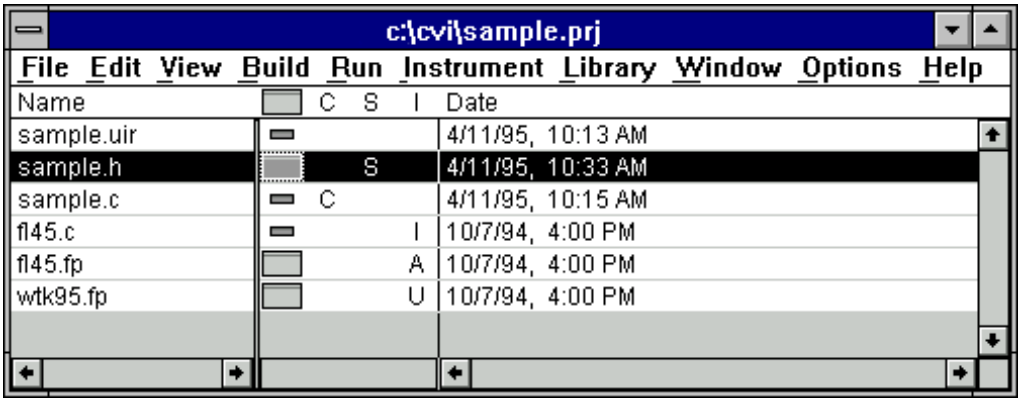

Figure 3-1. The Project Window

You can open `.c`, `.h`, and `.uir` files in the project list by double-clicking directly on the file name. Double-clicking on a `.fp` file brings up the Function Panel Selection dialog box for the instrument driver.

There are five icons used in the Project window. The icons are described below.

The file is currently closed. Double-clicking on this icon opens the file. If the file is a `.fp` file, double clicking brings up the Function Tree Editor.

The file is currently open. Double-clicking on this icon closes the file. If the file is a `.fp` file, double clicking hides the Function Tree Editor window, but does not unload the `.fp` file.

C The file has not yet been compiled, or has been modified since it was last compiled, or it was manually marked for compilation. Double-clicking on this icon compiles the file.

S The file has been modified since it was last saved. Double-clicking on this icon saves the file.

I The file is associated with a loaded instrument driver.

A This icon (A) indicates that the `.fp` file is loaded into the **Instruments** menu and signifies the following: `Attached` to (or `Associated` with) a program file.

U This icon (U) indicates that the `.fp` file is loaded into the **Instruments** menu and signifies the following: `Unattached` to any program file. When you double-click on this icon, LabWindows/CVI tries to attach a program file.

If there is no icon in the `I` column next to a `.fp` file, the `.fp` file is not loaded into memory. When you double-click on this icon, LabWindows/CVI tries to load the `.fp` file into memory and attach the instrument driver program file.

File Menu

This section contains a detailed description of the Project window **File** menu, shown in Figure 3-2.

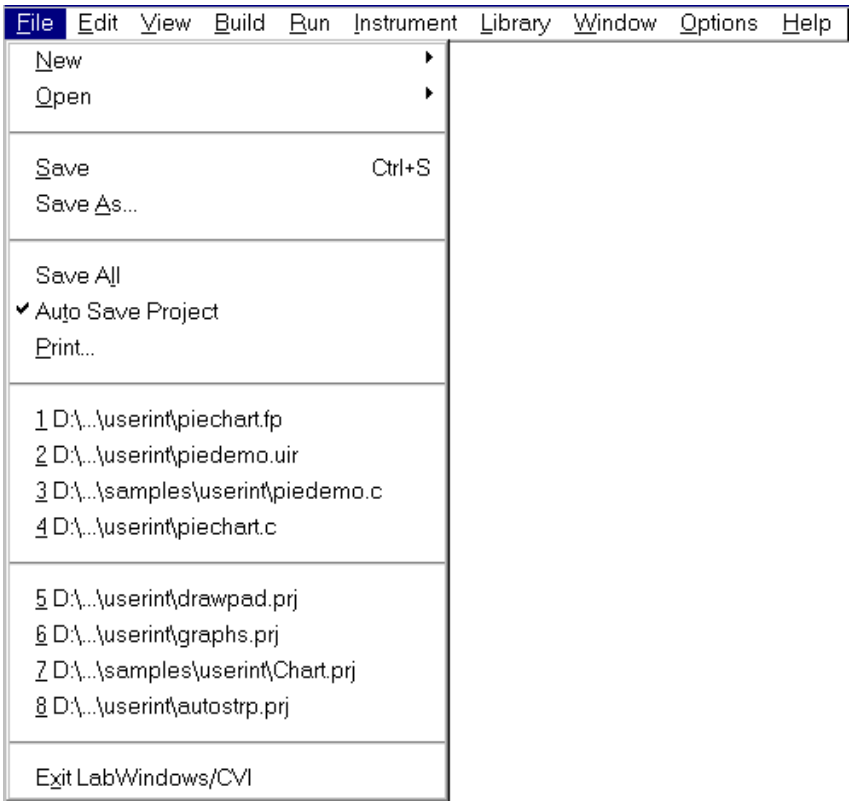

Figure 3-2. The File Menu

New

The **New** command has a submenu as shown in Figure 3-3.

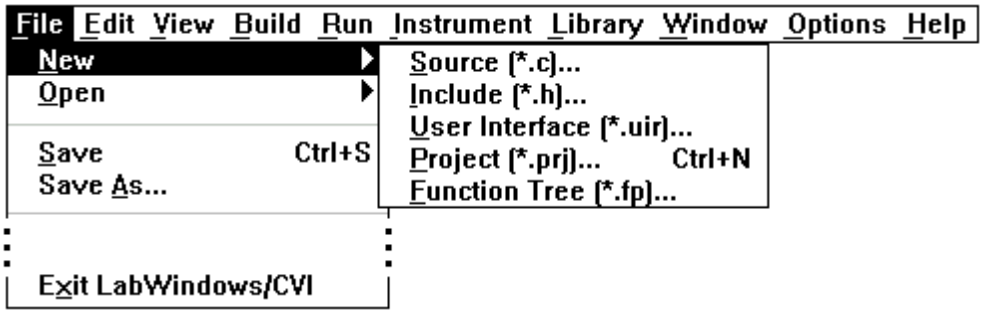

Figure 3-3. The New Command

The **New** command is used to open various types of new empty windows.

If you choose **Source** or **Include**, a new Source window appears in which you can create a new `.c` or `.h` file.

If you choose **User Interface**, a new User Interface Editor window appears in which you can create a new `.uir` file. See the *LabWindows/CVI User Interface Reference Manual* for more information about User Interface Editor windows.

If you choose **Project**, a dialog box appears with a message asking if you want to unload the current project, because only one project can be loaded at a time. If you select **Yes**, a new Project window appears. You will be prompted to save any modified files in the old project. You will also be prompted to save project options. These options are described in *The Options Menu* section later in this chapter.

If you choose **Function Tree**, a new Function Tree Editor window appears in which you can create a new `.fp` file. See the *LabWindows/CVI Instrument Driver Developers Guide* for details about Function Tree Editor windows.

Open

The **Open** command has a submenu as shown in Figure 3-4.

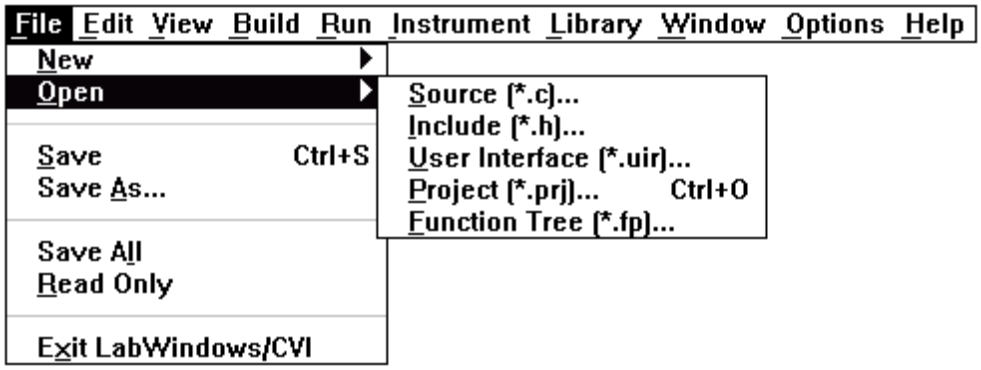

Figure 3-4. The Open Command

The **Open** command is used to open various types of user specified files. When you select **Open**, a dialog box appears, prompting you for a filename to be loaded into a new window. One feature of this dialog box is the **Directories** ring, which is in the upper-right corner of the dialog box under Windows 3.1, NT, and UNIX, and at the top of the dialog box under Windows 95. When activated, this ring displays a list of directories from which you have previously opened files.

If you choose **Source** or **Include**, a Source window appears with your specified `.c` or `.h` file.

If you choose **User Interface**, a User Interface Editor window appears with your specified `.uir` file.

If you choose **Project**, a Project window appears with your specified `.prj` file. You will be prompted to save any modified files in the old project.

If you choose **Function Tree**, a new Function Tree Editor window appears with your specified `.fp` file. You cannot use this command to load instrument modules. Instrument modules are loaded from the **Instrument** menu.

Save

Use the **Save** command to write the project (`.prj`) file to disk. If you want a different extension to be appended, type it in after the filename. If you do not want any extension to be appended, enter a period after the filename.

Save As...

Use the **Save As** command to write the contents of the Project window to disk using a new name you specify and change the name of the Project window to your new name. If you want an extension other than `.prj`, type it in after the filename. If you do not want any extension to be appended, enter a period after the filename.

Save All

The Save All command saves all open files to disk.

Auto Save Project

Your project files are saved automatically when the **Auto Save Project** command is enabled. When a project is loaded, the **Auto Save Project** command is initially enabled unless the project file is read-only on disk. If the command is enabled, the LabWindows/CVI automatically saves the project file whenever there is significant new or modified information to save in the project. If the command is disabled, the project file is saved only in the following cases.

- The **Save**, **Save As**, or **Save All** command is executed from the **File** menu.
- You unload the project or exit LabWindows/CVI. (You are prompted to save the file in this case).

Notice that if the **Auto Save Project** command is disabled, the project file is not saved when you start running a program, even if the **Save changes before running** option in the Run Options dialog box is set to **Always** or **Ask**.

Print

The **Print** command brings up a selectable list of all of the files in the project that are printable. You can checkmark the files you want to print.

Most Recently Closed Files

Two lists appear in the **File** menu, for your reference.

- a list of the four most recently closed files (other than project files)
- a list of the four most recently closed project files

Exit LabWindows/CVI

Use the **Exit LabWindows/CVI** command to close the current LabWindows/CVI session. If any open files have been modified since the last save, or if any windows contain unnamed files, you will be prompted to save them.

Edit Menu

This section contains a detailed description of the Project window **Edit** menu, shown in Figure 3-5.

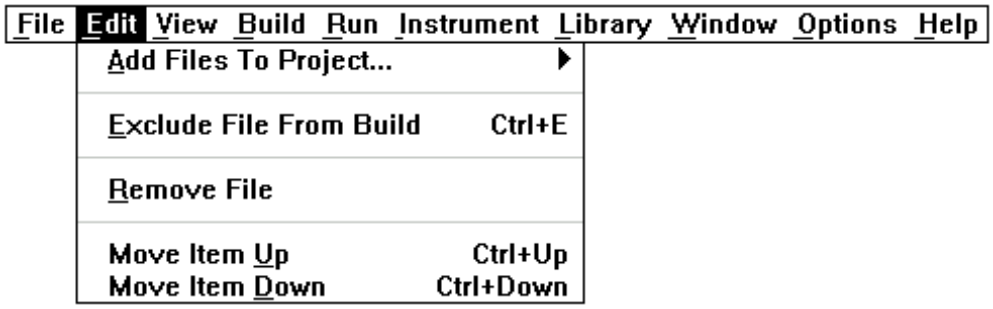

Figure 3-5. The Edit Menu

Add Files to Project...

The **Add Files to Project** command has a submenu as shown in Figure 3-6.

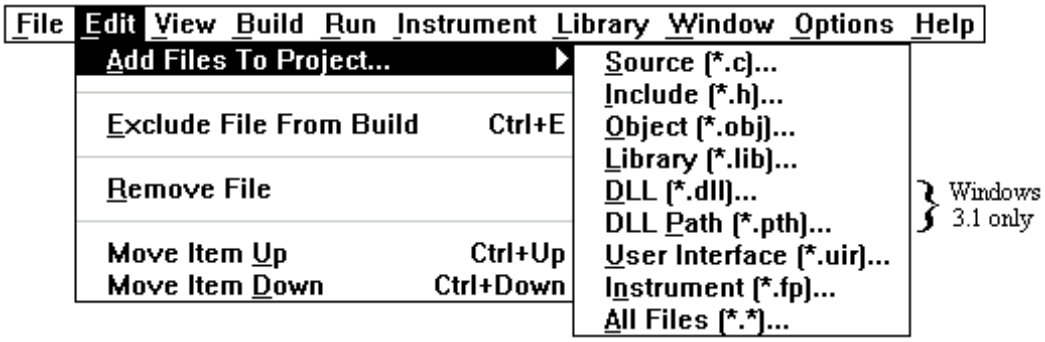

Figure 3-6. The Add File to Project Command

The **Add Files to Project** item allows you to add any type of file to the project list. Choosing any one of the file types listed in the tiered menu invokes a dialog box for selecting a file.

Use the **Source**, **Object**, **Library**, and **DLL** items to add code modules to your project. See Chapter 2, *Using Loadable Compiled Modules*, in the *LabWindows/CVI Programmer Reference Manual* for information about using object, library, and DLL files in LabWindows/CVI.

Use the **Include** item to add header files to your project. It is a good idea to have header files listed in your project because this makes access to you header files much easier.

Under Windows 3.1, use the **DLL Path** item to add a DLL path (`.pth`) file to your project. DLL path files are required when you want to load a DLL using the search method of the Windows `LoadLibrary` function. See Chapter 4, *Windows 3.1 Compiler/Linker Issues*, in the *LabWindows/CVI Programmer Reference Manual* for more details.

Under Windows 95 and NT, each DLL must be accompanied by an import library (`.lib`) file. If you want to use a DLL in your project, you must list the import library rather than the DLL. DLL and DLL path (`.pth`) files cannot be added to the project under Windows 95 or NT. If you load a project that was created in Windows 3.1 and that contains `.dll` or `.pth` files, a warning message appears, and the files are excluded.

For more detailed information on using DLLs in LabWindows/CVI for Windows 95 and NT, see the *Loading DLLs in LabWindows/CVI* section in Chapter 3, *Windows 95 and NT Compiler/Linker Issues*, in the *LabWindows/CVI Programmer Reference Manual*.

Use the **User Interface** item to add `.uir` files to your project. You should list `.uir` files in your project to make access to the file easier.

Use the **Function Tree** command to add instrument drivers to your project. Instrument drivers that are loaded via the project remain in memory while the project is open.

Exclude File from Build / Include File in Build

The **Exclude File from Build** command excludes the highlighted file from the build. This command does not apply to `.fp` files or `.uir` files because they do not affect the build. Excluded files appear in a different color in the Build window. They cannot be compiled or linked into the project. When an excluded file is highlighted, the command toggles to **Include File in Build** so that the file can be included in the build again.

Remove File

Use the **Remove File** command to remove the highlighted file from the project list.

Move Item Up

Use the **Move Item Up** command to move the highlighted file up one line in the project list. To activate this menu item, select **No Sorting** from the **View** menu.

Move Item Down

Use the **Move Item Down** command to move the highlighted file down one line in the project list. To activate this menu item, select **No Sorting** from the **View** menu.

View Menu

This section explains how to use the commands in the Project window **View** menu, shown in Figure 3-7.

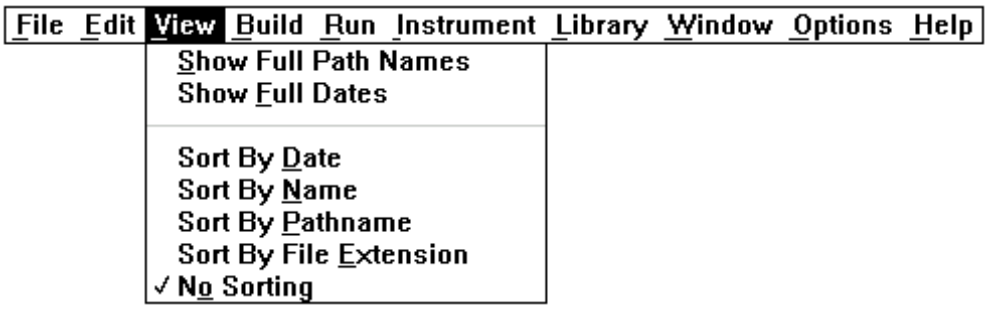

Figure 3-7. The View Menu

Show Full Path Names

Use this command to toggle between displaying the project list with full path names or simple base file names.

Show Full Dates

Use this command to toggle between displaying the project list with short file dates, such as `06/15/93`, and full file dates, such as `Tue, Jun, 15, 1993`.

Sort By Date

If you sort by date, the project list is displayed in chronological order.

Sort By Name

If you sort by name, the project list is displayed in alphabetical order by file name.

Sort By Pathname

If you sort by pathname, the project list is displayed in alphabetical order by directory pathname.

Sort By File Extension

If you sort by file extension, the project list is displayed in alphabetical order by file extension.

No Sorting

If you choose **No Sorting**, you can list your project files in any order by using the **Move Item Up** and **Move Item Down** items in the **Edit** menu.

Build Menu

This section explains how to use the commands in the Project window **Build** menu, as shown in Figure 3-8.

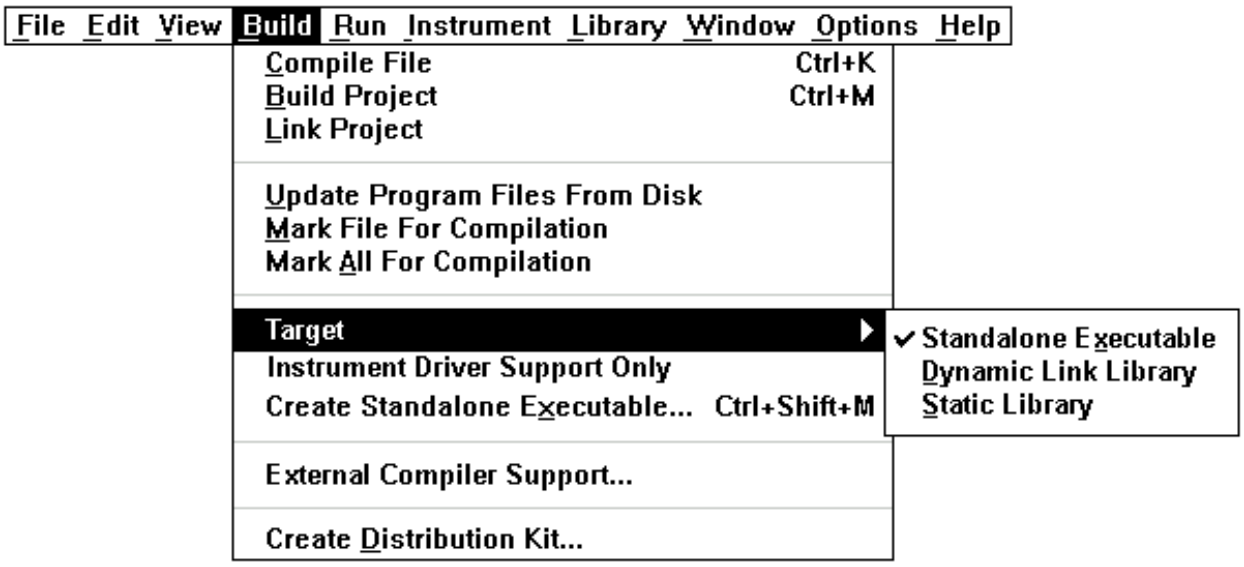

Figure 3-8. The Build Menu

Commands in the **Build** menu are used for compiling files, building and linking projects, marking files for compilation, and creating application files.

Compile File

You must compile your source code before executing your project. Use the **Compile File** command to compile the selected source file into a LabWindows/CVI object module. If any build errors are encountered, the process terminates and the Build Errors window appears with a list of errors.

See the descriptions of **Compiler Options** and **Compiler Defines** in the *Options Menu* section of this chapter for a discussion of compiler options and defines.

Build Project

The **Build Project** command compiles all source files listed in the project that are marked for compilation and links the compiled files. See the description of the **Mark File for Compilation** command in this section for information on marking files for compilation.

Link Project

The **Link Project** command links the compiled files from the project. It does not compile any files.

Update Program Files from Disk

This command determines if any `.c`, `.h`, `.obj` (Windows), `.o` (UNIX), `.lib` (Windows), or `.a` (UNIX) files either in your project or associated with a loaded instrument have been modified outside of LabWindows/CVI. It does so by comparing the recorded dates of the files with the actual dates of the files on disk. If any file in the project (other than `.uir` or `.fp` files, which cannot be modified outside LabWindows/CVI) is found on disk with a different date, the following rules apply.

- If the file is not open, or if the file is open but has not been modified, the file on disk is reloaded into the project.
- If the file is open and has been modified, you can choose to overwrite the disk file with the file window contents, or load the disk file into a new window and close the current window.

Mark File for Compilation

When a source file is marked for compilation, a `'C'` appears next to the filename in the Project window. LabWindows/CVI recompiles marked files the next time the project is built. When you modify a source file, LabWindows/CVI automatically marks the file for compilation. You can force LabWindows/CVI to compile a source file on the next build with the **Mark File for Compilation** command.

Mark All for Compilation

Use the **Mark All Files for Compilation** command to force LabWindows/CVI to recompile all source files in the project the next time the project is built.

Target (Windows 95/NT Only)

The **Target** item brings up a submenu in which you select the target type for your project. The target type determines what type of file is created when you execute the command that appears below **Target** in the **Build** menu. The **Create** command that appears below **Target** in the **Build** menu changes name depending on the target type selected. The target types that you can select are:

- Standalone Executable
- Dynamic Link Library
- Static Library

When anything other than **Standalone Executable** is selected, the **Run Project** command in the **Run** menu is dimmed.

Instrument Driver Support Only

Note: ***This command is* not *available under Windows 3.1.***

If the **Instrument Driver Support Only** command is enabled, your project does not link to the entire set of LabWindows/CVI libraries. Instead, it links to a smaller set of functions. Standalone executables and DLLs created when the **Instrument Driver Support Only** command is enabled do not use the CVI Run-Time Engine DLL, `cvirte.dll`. Rather, they use `instrsup.dll`, which is much smaller. (On Solaris 1 and 2, it is called `libinstrsup.so`. On HP-UX, it is called `libinstrsup.sl`.)

This command is particularly useful for creating instrument driver DLLs. It allows the DLLs to be used by other applications without having to load the large CVI Run-Time Engine DLL.

`instrsup.dll` contains functions from the following libraries.

- Formatting and I/O Library
- RS-232 Library
- Utility Library (selected functions only -- see below)
- ANSI C

If you are using a standalone compiler and want to use `instrsup.dll`, include `cvi\extlib\instrsup.lib` in your external compiler project instead of `cvirt.lib` and `cvisupp.lib`. Remember that when you use an external compiler, you link to that compiler's ANSI C library.

Your project can also link to the following libraries.

- Analysis or Advanced Analysis Library
- GPIB Library
- VXI Library
- VISA Library
- Easy I/O For DAQ Library
- Data Acquisition Library

If you are using a standalone compiler under Windows 95 or NT and want to use any of these libraries, refer to the *Using the LabWindows/CVI Libraries in External Compilers* section in *Chapter 3, Windows 95 and NT Compiler/Linker Issues* of the *Programmer Reference Manual*.

If you use the **Create Distribution Kit** command on a project which is linked for instrument driver support only, `instrsup.dll` is automatically included in the distribution kit. Also, the option to distribute the full CVI Run-Time Engine is disabled by default.

Utility Library Functions

The following Utility Library functions are in `instrsup.dll`.

```
Beep
DateStr
Delay
SyncWait
Timer
TimeStr
RoundRealToNearestInteger
TruncateRealNumber
```

No Standard Input/Output

`instrsup.dll` does not support the Standard Input/Output Window. Functions such as `FmtOut` or `ScanIn` return errors.

Multi-Threaded Safe

All of the functions in `instrsup.dll` are multi-threaded safe.

Create Standalone Executable...

Use the **Create Standalone Executable** command to create an executable version of the current project. Before creating an executable version of your project, read Chapter 7, *Creating and Distributing Standalone Executable Programs*, in the *LabWindows CVI Programmer Reference Manual*. When you select the **Create Standalone Executable** command, the Create Application File dialog box appears as shown in the following figure.

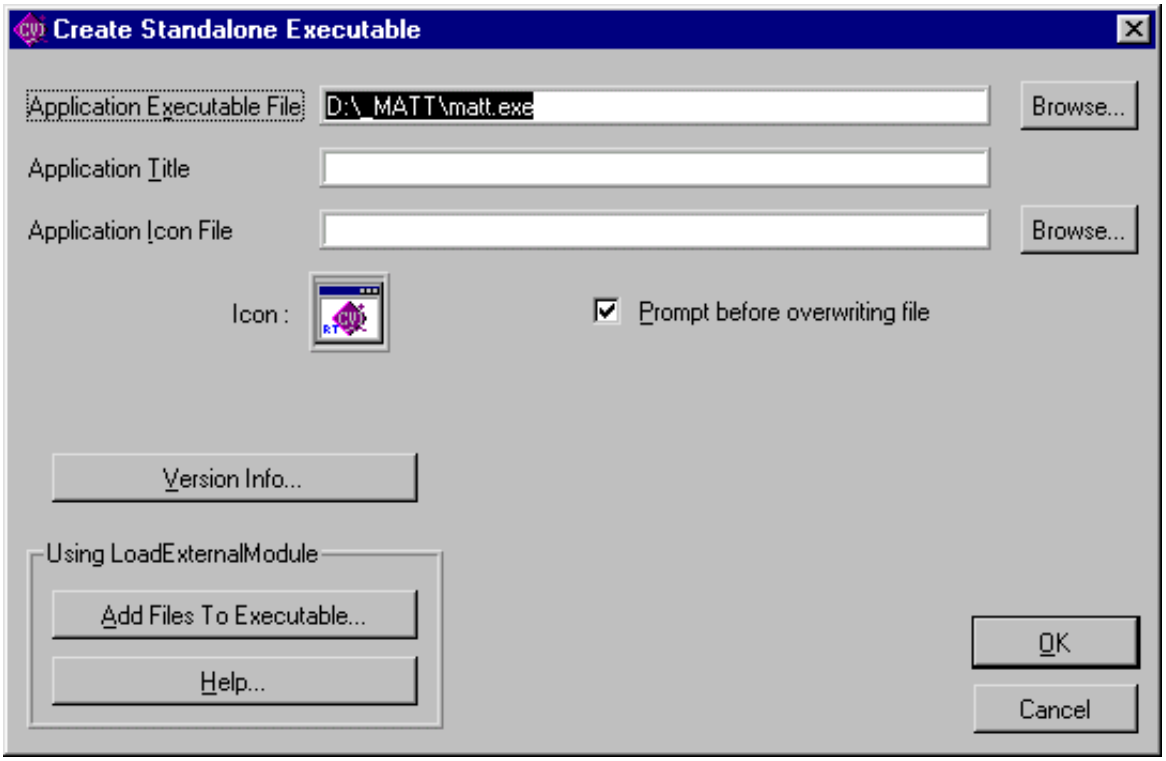

Figure 3-9. The Create Standalone Executable Dialog Box

- **Application Executable File**—The name of the executable file associated with your program. You can use the **Browse** button to select an existing filename.
- **Application Title**—A descriptive title for your program. This title appears below the icon of the executable program in the Program Manager and it is registered with the operating system when a user starts the application.
- **Application Icon File (Windows only)**—A file containing a descriptive graphical icon for your program. You can double-click on the icon in the Program Manager to start your executable. You can use the **Browse** button to select an existing icon file.
- **Icon (Windows only)**—The graphical representation of the Application Icon File. You can double-click on this control to browse for an icon file on disk. The sample program, `samples\apps\iconedit.exe`, is included so that you can create your own icon files.

- **Prompt before overwriting executable file**—A checkbox where you can set the program to prompt you before it overwrites an Application Executable File that has the same name as the one you are saving.

- **Using LoadExternalModule**

  – **Add Files to Executable**—This button lets you select additional module files which you want to be linked into the Application Executable File. These are modules which are not directly referenced by your project files but which are referenced by modules you load at run-time by calling `LoadExternalModule`.

  – **Help**—This button describes the use of `LoadExternalModule` in an executable and the **Add Files to Executable** button.

- **OK**—This button accepts the current inputs and attempts to create the Application Executable File.

- **Version Info**—When you click on this button the Version Info dialog box appears, where you can enter version information for the executable file. The information is saved in the executable in the form of a standard Windows version resource. The information can be obtained from the executable by using the Windows SDK functions `GetFileVersionInfo` and `GetFileVersionInfoSize`.

  In the Version Info dialog box, the entries for **File Version** and **Product Version** must be in the form,

  `n,n,n,n`

  where `n` is a number from 0 to 255

- **Cancel**—This button cancels the operation and removes the dialog box.

Create Dynamic Link Library (Windows 95/NT Only)

Use the **Create Dynamic Link Library** command to create a dynamic link library (`.dll`) file from the current project. A DLL import library (`.lib`) file is also created. When you select the command, the Create Dynamic Link Library dialog box appears as shown in the following figure.

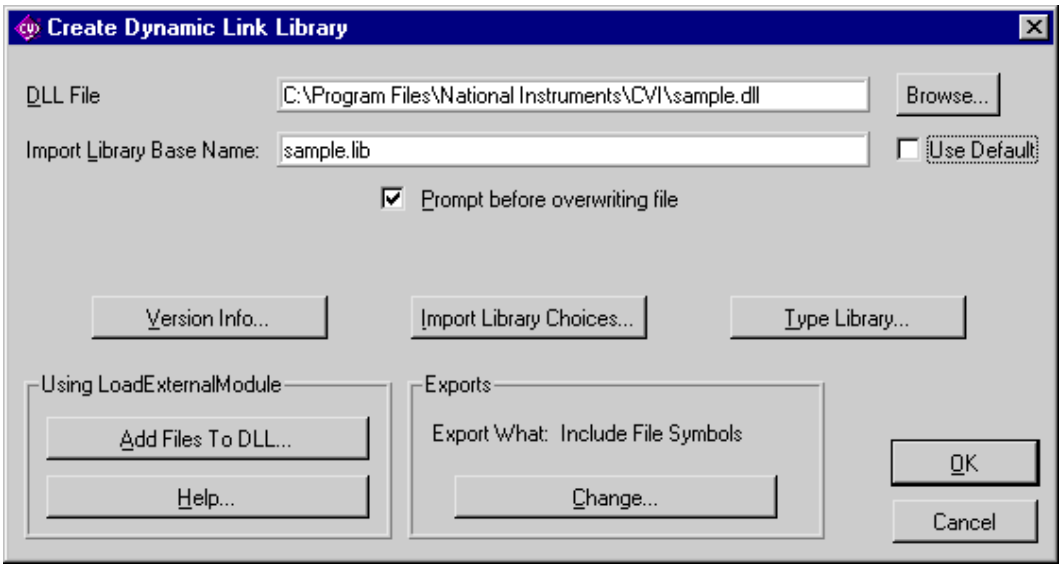

Figure 3-10. The Create Dynamic Link Library Dialog Box

- **DLL File**—The name of the DLL file to be created. You can use the **Browse** button to select an existing filename or enter a new one.
- **Import Library Base Name** — Normally the name of the import library is the same as the name of the DLL, except that the extension is `.lib`. There may be some cases, however, where want to use a different name. For example, you may want to append "`_32`" to the name of your DLL to distinguish it as a 32-bit DLL, but not append it to the import library name. This is, in fact, the convention used for VXI*plug&play* instrument driver DLLs. If you want to enter a different name for the import library, deselect the **Use Default** checkbox. Enter a name without any directory names.
- **Prompt before overwriting file**—A checkbox where you can choose to be prompted before the program overwrites a DLL file that has the same name as the one you are creating.
- **Version Info**—When you click on this button the Version Info dialog box appears, where you can enter version information for the DLL. The version information is saved in the DLL in the form of a standard Windows version resource. The information can be obtained from the DLL by using the Windows SDK functions `GetFileVersionInfo` and `GetFileVersionInfoSize`.

  In the Version Info dialog box, **File Version** and **Product Version** must be in the form,

  `n,n,n,n`

  where `n` is a number from 0 to 255
- **Import Library Choices**—This button lets you choose whether to create a DLL import library for each of the compatible external compilers, or to create one only for the current compatible compiler. See the *Compatibility with External Compilers* section in Chapter 3 *Windows 95 and NT Compiler/Linker Issues*, of the *LabWindows/CVI Programmer*

Reference Manual. It also lets you choose whether to create the import libraries in the VXI*plug&play* subdirectories instead of the directory of the DLL.

If you choose to use the DLL directory and to create an import library for each compiler, the files are created in subdirectories named `MSVC`, `BORLAND`, `WATCOM`, and `SYMANTEC`. The library for the current compatible compiler is also created in the directory of the DLL.

If you choose to create an import library only for the current compiler, the file is created in the directory of the DLL.

If you choose to use the VXI*plug&play* directories and to create an import library for each compiler, the files are created in the subdirectories `MSC`, `BC`, `WC`, and `SC` under the VXI*plug&play* LIB directory. If you choose to create an import library only for the current compiler, the file is created in the appropriate one of those subdirectories.

- **Type Library**—This button lets you choose whether to add a Type Library resource to your DLL. You can also choose to include links in the Type Library resource to a Windows help file. The Type Library resource is generated from a function panel (`.fp`) file. You must specify the name of the `.fp` file. A Windows help file can be generated from the `.fp` file by using the **Generate Windows Help** command in the **Options** menu of the .Function Tree Editor window and then using the Windows Help Compiler.

  This feature is useful if you intend your DLL to be used from Visual Basic. For more information, see the section *Automatic Inclusion of Type Library Resource for Visual Basic* in Chapter 3, *Windows 95 and NT Compiler/Linker Issues*, of the *LabWindows/CVI Programmer Reference Manual*.

- **Using LoadExternalModule**
  - **Add Files to DLL**—This button lets you select additional module files which you want to be linked into the DLL. These are modules which are not directly referenced by your project files but which are referenced by modules you load at run-time by calling `LoadExternalModule`.
  - **Help**—This button describes the use of `LoadExternalModule` in a DLL and the **Add Files to DLL** button.

- **Exports**
  - **Export What**—This indicates your current choice of method for determining which symbols in the DLL are exported to the users of the DLL. The **Change** button is used to change your choice.
  - **Change**—This button lets you select the method to use for determining which symbols in the DLL are exported to the users of the DLL. The choices are the following.

    - **Include File Symbols**—You must name one or more include files that declare symbols defined globally in the DLL. The declared symbols are the ones exported. You can select from a list of include files in the project.

    - **Symbols Marked for Export**—All symbols in the DLL defined with qualifier `__declspec(dllexport)` or `export` are exported.

- **OK**—This button accepts the current inputs and attempts to create the DLL.
- **Cancel**—This button cancels the operation and removes the dialog box.

Note: ***When you use the* Symbols Marked for Export *option and have included in your project an object or library file defining exported symbols, LabWindows/CVI cannot correctly create the import libraries for each of the four compatible compiler. This problem does not arise if you are using only source code files in your DLL project.***

For more information on creating DLLs, see the *Preparing Source Code for Use in a DLL* section in Chapter 3, *Windows 95 and NT Compiler/Linker Issues*, of the *LabWindows/CVI Programmer Reference Manual*..

Create Static Library (Windows 95/NT Only)

Use this **Create Static Library** command to create a static library (`.lib`) file from the current project. When you select the command, the Create Static Library dialog box appears as shown in the following figure.

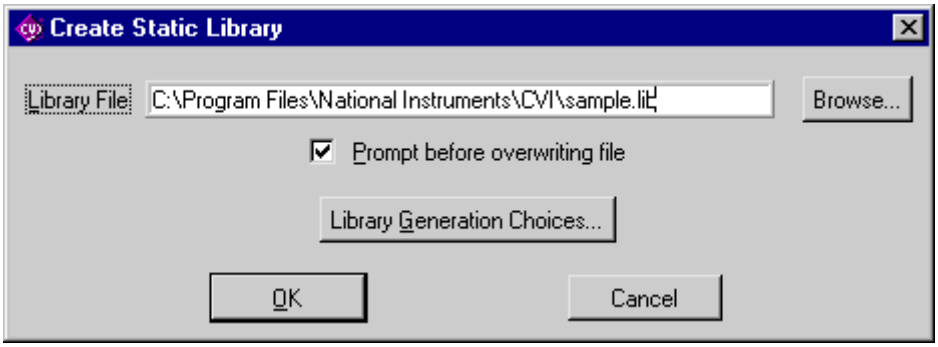

Figure 3-11. The Create Static Library Dialog Box

- **Library File—**The name of the library file to be created. You can use the **Browse** button to select an existing filename or name a new one.
- **Prompt before overwriting file**—A checkbox where you can choose to be prompted before the program overwrites a library file that has the same name as the one you are creating.
- **Library Generation Choices**—This button lets you choose whether to create a static library for each of the compatible external compilers, or to create one only for the current compatible compiler. See the *Compatibility with External Compilers* section in Chapter 3, *Windows 95 and NT Compiler/Linker Issues*, of the *LabWindows/CVI Programmer Reference Manual*. If you want to create a static library for each compiler, you must not include any object or library files in your project, because such files are specific to particular compiler.
- If you choose to create a static library for each compiler, the files are created in subdirectories named `MSVC`, `BORLAND`, `WATCOM`, and `SYMANTEC`. The library for the current compatible compiler is also created in the parent directory.

- **OK**—This button accepts the current inputs and attempts to create the library file.
- **Cancel**—This button cancels the operation and removes the dialog box.

Note: ***If you include a*** `.lib` ***file in a static library project, all object modules from the*** `.lib` ***are included in the static library. (This differs from creating an executable or DLL, in which only the*** `.lib` ***modules referenced by other modules in the project are included in the target.) In addition, LabWindows/CVI reports an error if you attempt to build a static library when you have a DLL import library in your project.***

External Compiler Support (Windows 95/NT only)

Use the **External Compiler Support** command to help you build your executable or DLL in one of the four compatible external compilers. For detailed information on this topic, see the *Creating Executables and DLLs in External Compilers for Use with the LabWindows/CVI Libraries* section in Chapter 3, *Windows 95 and NT Compiler/Linker Issues*, of the *LabWindows/CVI Programmer Reference Manual.*

When you execute the command, the External Compiler Support dialog box appears, as shown in the following figure.

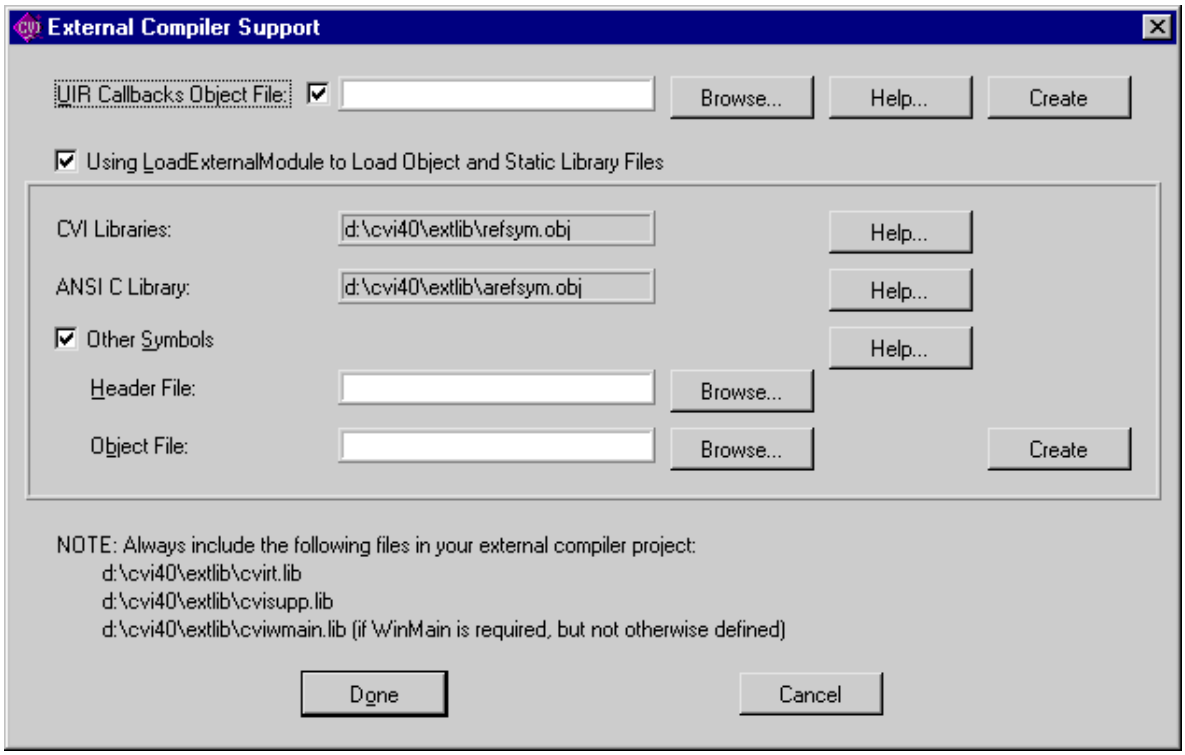

Figure 3-12. External Compiler Support Dialog Box

- **UIR Callbacks Object File**—This option creates an object file for you to link into your executable or DLL. The object file contains a list of the callback functions specified in the User Interface Resource (`.uir`) files in your project. When you load a panel or menu bar from the `.uir` file, the User Interface Library uses the list to link the objects in the panel or menu bar to their callback functions in your executable or DLL. If you specify callback function names in your `.uir` file(s), checkmark the checkbox, enter the name of the object file to be created, and click on the **Create** button. In the future, whenever you save modifications to any of the `.uir` files in the project, LabWindows/CVI automatically updates the object file.

  You must call the `InitCVIRTE` function at the beginning of your `main`, `WinMain`, or `DLLmain` function so that LabWindows/CVI run-time libraries can initialize the list of names from the object file. If you are creating a DLL and any of your callback functions are defined in, but not exported by, the DLL, you must call `LoadPanelEx` or `LoadMenuBarEx` (rather than `LoadPanel` or `LoadMenuBar`) from the DLL.

- **Using LoadExternalModule to Load Object and Static Library Files**—This option enables the section of the dialog box that you use when creating an executable or DLL that calls the Utility Library `LoadExternalModule` function to load object or static library files.

  Note: ***You do not need this option if you use*** `LoadExternalModule` ***to load only DLLs (which are loaded via DLL import libraries).***

  Unlike DLLs, in which all of the external references are resolved at link time, objects and static libraries can contain unresolved external references. When you use `LoadExternalModule` to load an object or static library file, these references are resolved using symbols in your executable or DLL, or in previously loaded external modules. Consequently, the names of the symbols in your executable or DLL that are needed to resolve these references must be available to the `LoadExternalModule` function.

- **CVI Libraries**—This provides information needed when your run-time modules reference symbols in any of the following LabWindows/CVI libraries:
  - User Interface
  - RS-232
  - DDE
  - TCP
  - Formatting and I/O
  - Utility

  If you use one of these libraries, include in your external compiler project the object file displayed in this indicator.

- **ANSI C Library**—This provides information needed when your run-time modules reference symbols in the ANSI C library. Include in your external compiler project the object file displayed in this indicator.

- **Other Symbols**—Checkmark this option if your run-time modules reference symbols other than those covered by the previous two options. Such symbols include functions or variables that are defined globally in your executable or DLL and to which your object or static library run-time modules expect to link. This option creates a file for you to link into your executable (or DLL).
  - **Header File**—Insert the name of an include file that contains complete declarations of all of the symbols needed to resolve references from run-time modules.
  - **Object File**—Enter the name of the object file that is to be created. Click on the **Create** button to create the file. You must include this file in your external compiler project.

The bottom of the External Compiler Support dialog box contains a list of library files that you need to include in your external compiler project. The files are the following.

```
cvi\extlib\cvirt.lib
cvi\extlib\cvisupp.lib
cvi\extlib\cviwmain.lib
```

`cvimain.lib` is needed only when the external compiler requires `WinMain` to be defined, you have not defined it in your project, and it is not defined in any of the libraries automatically linked by the external compiler. In general `WinMain` is not required for console applications. It is required for GUI applications but is sometimes included automatically when you use a GUI application "wizard".

Create Distribution Kit (Windows 3.1 and Windows 95/NT Only)

The **Create Distribution Kit** option is available only in LabWindows/CVI for Windows. For information on distributing executables under UNIX, see the *Distributing Standalone Executables Under UNIX* section in Chapter 5 of the *LabWindows/CVI Programmer Reference Manual*.

Use the **Create Distribution Kit** command to make a set of disks from which you can install your executable program on a target machine. **Create Distribution Kit** automatically includes all the files necessary to run your executable program on a target computer except for DLLs for National Instruments hardware and files that are loaded using `LoadExternalModule`.

DLLs for National Instruments hardware should not be part of your distribution kit. End-users can install the DLLs for their hardware from the distribution disks that National Instruments supplies to those users.

If you are loading files using `LoadExternalModule`, you must include these files manually using the **Add**/**Edit Group** features of the **Create Distribution Kit** command. Refer to the *Rules for Loading Files Using LoadExternalModule* section in Chapter 7 of the *LabWindows/CVI Programmer Reference Manual*.

When you select the **Create Distribution Kit** command, the Create Distribution Kit dialog box appears as shown in the following figure.

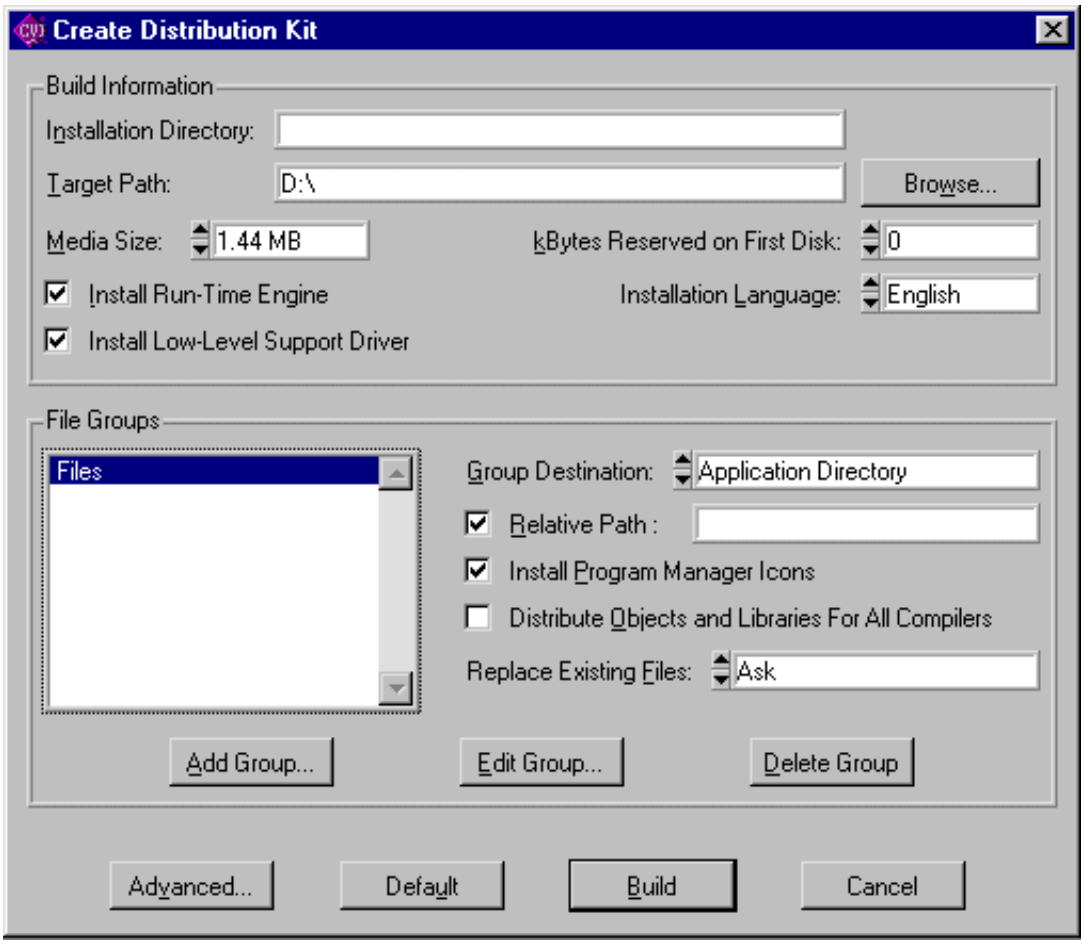

Figure 3-13. The Create Distribution Kit Dialog Box

Build Information Section

- **Installation Directory**—The default directory that appears in the installation of the end-user.
- **Target Path**—The path into which you want to build your distribution kit.

  Note: ***When you define a target path to a floppy disk, you must specify the root directory.***

- **Browse…**—This button lets you browse for a target path on disk.
- **Install Run-Time Engine**—The checkbox for including the LabWindows/CVI Run-time Engine and associated files in your distribution kit. If you know that the Run-time Engine is already installed on the target machine or if you want to copy and distribute the Run-time Engine separately, you do not need to include the Run-time Engine files in your distribution kit.
- **Install Low-Level Support Driver**—This option appears in Windows 95 and NT. It lets you choose whether to install the LabWindows/CVI low-level support driver on the end-user's

computer. The following Utility Library functions require the LabWindows/CVI low-level driver to be loaded at startup.

Table 3-1. Platforms Where Utility Functions Need Low-Level Support Driver

Function	**Platforms where low-level support driver is needed**
`inp`	Windows NT
`inpw`	Windows NT
`outp`	Windows NT
`outpw`	Windows NT
`ReadFromPhysicalMemory`	Windows 95 and NT
`ReadFromPhysicalMemoryEx`	Windows 95 and NT
`WriteToPhysicalMemory`	Windows 95 and NT
`WriteToPhysicalMemoryEx`	Windows 95 and NT
`DisableInterrupts`	Windows 95
`EnableInterrupts`	Windows 95
`DisableTaskSwitching`	Windows 95

- **Media Size**—The media size of your distribution diskettes. When you choose a floppy drive as your target path, LabWindows/CVI determines the media size automatically and makes this control inactive.
- **kBytes Reserved on First Disk**—Allows you to reserve space on the first disk of your distribution kit for extra files.
- **Installation Language**—The language that is used for text during the installation.

File Groups Section

- **File Groups**—The list box where you can separate the files in your distribution kit into groups. You must assign a destination directory to each group. The installation program creates the directories on the target machine and places each of the file groups in its assigned directory. Each of the options to right of the list box can be set to different values for each file group.
- **Group Destination**—This ring control sets the root destination directory for the selected group.

- **Relative Path**—This control lets you assign a relative path, based on the root destination directory, in which to install the selected group.

- **Install Program Manager Icons**—This checkbox lets you choose whether to create a Windows program group containing icons for files in the selected file group. The installation program can install the embedded icons for `.exe` files and the default icons for `.pif`, `.com`, `.txt`, `.wri`, `.bat`, and `.hlp` files.

- **Distribute Objects and Libraries for All Compilers—**This checkbox appears in LabWindows/CVI for Windows 95 and NT to help you distribute object files, static libraries, and DLL import libraries for all of the compatible external compilers. When selected, this option affects all of the `.obj` and `.lib` files listed in the selected file group. Four versions of each file are included in the distribution kit. These versions are expected to be in subdirectories under the specified location of each file. The subdirectories must be named `MSVC`, `BORLAND`, `WATCOM`, and `SYMANTEC`. For example, if you specify the following file,

  ```
  c:\myapp\distr\big.lib
  ```

  in a file group and the **Distribute Objects and Libraries for All Compilers** checkbox is checked, then when the distribution kit is created, you must have the following files on your disk:

  ```
  c:\myapp\distr\msvc\big.lib
  c:\myapp\distr\borland\big.lib
  c:\myapp\distr\watcom\big.lib
  c:\myapp\distr\symantec\big.lib
  ```

  When installing the software, the end-user is prompted to choose one of the compatible external compilers. Only the files for the chosen compiler are actually installed on the end-user's computer.

  You might want to use this feature if you are distributing modules for use with the LabWindows/CVI development environment or external compilers. If you are distributing a turnkey application, you do not need this feature.

Note: ***Do* not *use this feature for distributing DLL import libraries for VXIplug&play instrument drivers. When installing a VXIplug&play instrument driver, you need to install two import libraries, one compatible with Visual C/C++, the other with Borland C/C++. The two import libraries must be installed in the*** `MSC` ***and*** `BC` ***subdirectories under the VXIplug&play*** `LIB` ***directory. LabWindows/CVI sets this up automatically for you if you use the* Create DLL Project *command in the Function Tree Editor Window with the* VXIplug&play Style *command enabled.***

- **Replace Existing Files**—This control lets you configure the installation program to always replace, never replace, replace if newer, or ask before replacing existing files.

  The **Check Version** option applies to files with a Windows version resource (in other words, DLLs and executables). The file on the distribution kit and the existing file are checked for a

version resource. If each has a version resource, the existing file is replaced if the version number of the file in the distribution kit is newer than the version number in the existing file. If the file in the distribution kit has a version resource but the existing file does not, the existing file is replaced. When the file in the distribution kit does not have a version resource, the existing file is replaced if the date of the file in the distribution kit is newer.

- **Add Group…**—This button lets you add a new group to your distribution kit.
- **Edit Group…**—This button lets you edit the selected group.
- **Delete Group**—This button lets you delete the selected group from your distribution kit.

Main Section

- **Restore Defaults—**Resets all controls in the Create Distribution Kit dialog box to their default values. When you are creating a new distribution kit, you can click on **Restore Defaults** to undo changes you have made to the controls in the dialog box.

  Note: ***When you are using the Create Distribution Kit dialog box to modify an existing distribution kit,*** **Restore Defaults** ***replaces your existing file grouping and settings with default values. If you click on*** **Restore Defaults** ***in error, click on*** **Cancel** ***to prevent this change to your distribution kit.***

- **Build**—This button lets you build your distribution kit.
- **Cancel**—This button lets you cancel the **Create Distribution Kit** operation.
- **Advanced**—When you click on this button, the Advanced Distribution Kit Options dialog box appears.

Advanced Distribution Kit Options

Clicking on the **Advanced** button in the Create Distribution Kit dialog box brings up the Advanced Distribution Kit Options dialog box.

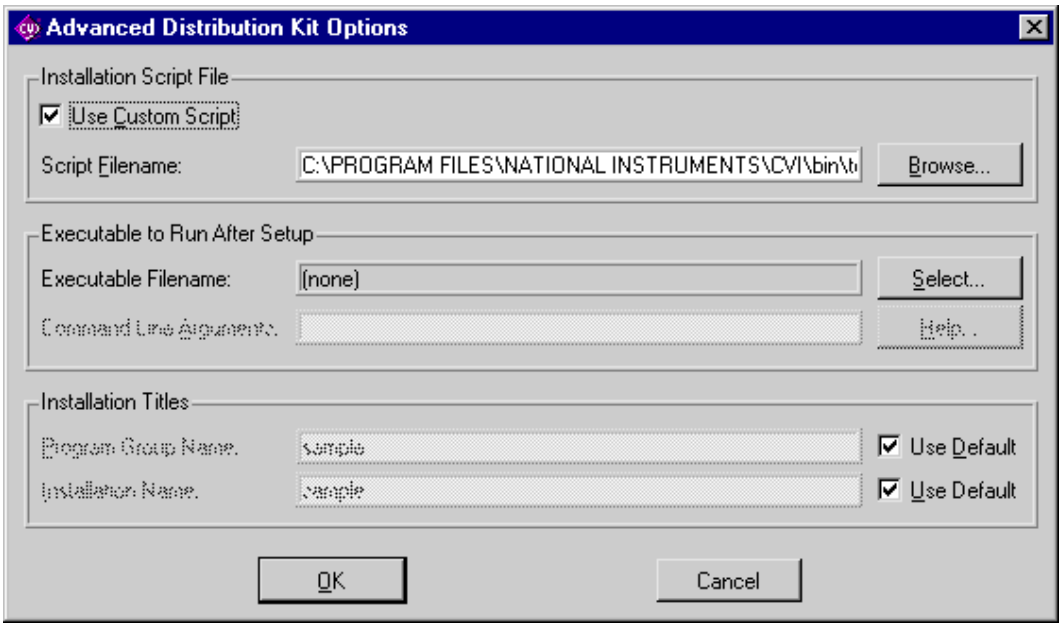

Figure 3-14. Advanced Distribution Kit Options dialog box

Installation Script File Section

- **Use Custom Script**—When this checkbox is selected, you can enter the name of a customized installation script file for your distribution kit. The default installation script file is `cvi\bin\template.inf`.

  The installation script for distributing VXI*plug&play* instrument drivers is `cvi\bin\vxipnp.inf`. The instructions for using the VXI*plug&play* installation script are in the file `cvi\bin\vxipnp.doc`. If you use the **Create DLL Project** command in the Function Tree Editor Window with the **VXIplug&play Style** command enabled, project settings are automatically generated so that the **Create Distribution Kit** command uses the VXI*plug&play* installation script and distributes all of the files required of a VXI*plug&play* installation.

- **Script Filename**—The pathname of the customized installation script file. You can use the **Browse** button to select an existing filename.

Executable to Run After Setup

- **Executable Filename**—The name of an executable file to run after the end-user installation is complete. Use the **Select** button to select a file that has already been added to one of the file groups.

- **Command Line Arguments**—The command line arguments to pass to the executable run after the installation is complete. Use the Help button to view detailed information on special macros you can use in this control.

Installation Titles

- **Program Group Name**—The name of the program group created during the installation. If the **Use Default** button is checked, the following priority is used to determine the program group name.
  - If the project target is an executable, the application title entered in the Create Standalone Executable dialog box is used, if it is non-empty.
  - Otherwise, the base file name of the target file (executable, DLL, or static library) is used, if it has been created.
  - Otherwise, the base file name of the project is used.
- **Installation Name** -- The installation window title and the text displayed in upper part of the installation window. If the **Use Default** button is checked, the name is set using the same priority as for the **Program Group Name**.

Run Menu

This section explains how to use the commands in the Project window **Run** menu, as shown in Figure 3-15.

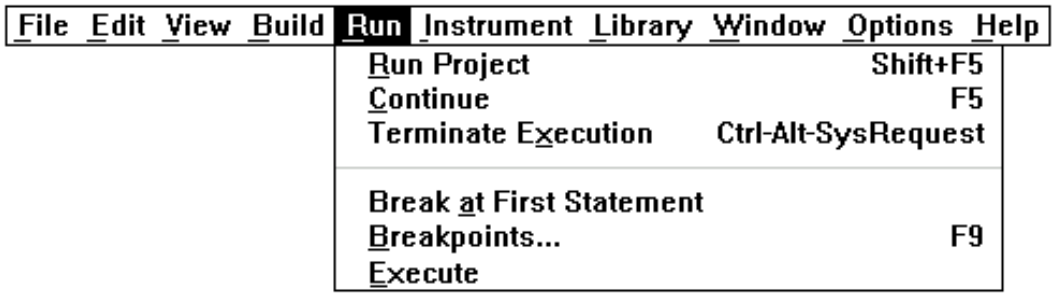

Figure 3-15. The Run Menu

You can use commands in the **Run** menu to run your program and assign breakpoints. For more information about breakpoints, see the *Introduction to Breakpoints and Watch Variables/Expressions* section in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows.*

Run Project

The **Run Project** command compiles any source files in the project that are marked for compilation, links them together, and runs the project. If any build errors are encountered, the process terminates and the Build Errors window appears with a list of errors.

Run-Time Error Reporting

During the execution of a program, LabWindows/CVI can report various run-time errors. One example of a run-time error is a call to a LabWindows/CVI library function in which an array or string is not large enough to hold the output data.

When such errors occur, a dialog box appears identifying the type of error and the location in the program where the error occurred. The error is then listed in the Runtime Error window.

LabWindows/CVI then suspends the program so you can inspect the values of variables in the Variable Display. To terminate a program that has been suspended because of a run-time error, select the **Terminate Execution** command or press <Ctrl-Alt-SysRq> under Windows 3.1 or <Control-F12> under Windows 95/NT and UNIX.

Continue

Use the **Continue** command to resume program execution when running in breakpoint mode.

Terminate Execution

Use the **Terminate Execution** command to terminate a program that is suspended in breakpoint mode.

Break at First Statement

Selecting **Break at First Statement** forces LabWindows/CVI to break a program on the first executable statement. When activated, this command has a checkmark beside it in the menu.

Breakpoints...

The **Breakpoints** command brings up the Breakpoints dialog box containing a list of the breakpoints in the project. You may add, delete, or edit project breakpoints from this dialog box. For a complete description of this command, see *The Run Menu* section of Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows.*

Execute

The **Execute** command launches the executable associated with the project. The executable associated with the project is the executable file that you last created from the Create Standalone Executable dialog box. If no executable file has been yet created or the current target type for the project is DLL or Static Library, then this command is dimmed.

Using Instrument Drivers

This section presents a general overview of instrument drivers. For detailed information on creating instrument drivers, see the *LabWindows/CVI Instrument Driver Developers Guide.*

Load and unload instrument drivers using the commands in the **Instrument** menu, shown in Figure 3-16.

Figure 3-16. The Instrument Menu

An *instrument driver* is a set of high-level functions with graphical function panels that make programming easier. It encapsulates many low-level operations, such as data formatting and communication with GPIB, RS-232, and VXI, into intuitive, high-level functions. An instrument driver usually controls a physical instrument, but it can also be a software utility.

Instrument driver programs have an associated include file that declares the high-level functions you can call, the global variables you can access, and the defined constants you can use.

Instrument Driver Files

A LabWindows/CVI instrument driver typically consists of three files. Each file has the same base filename, which is an abbreviation of the actual instrument name. The three instrument driver files must reside in the same directory on your disk.

- The function panels are in a file with the extension `.fp`. See Chapter 5, *Using Function Panels*, for a detailed description of function panels.
- The function, variable, and defined constant declarations are in an include file with the extension `.h`.
- The instrument driver program can be in one of several different types of files.
  - A source file with the extension `.c`.
  - An object file containing one or more compiled C modules with the extension `.obj` under Windows , or `.o` under UNIX, or a library file with the extension `.lib` under Windows or `.a` under UNIX.. The compilation must be done by LabWindows/CVI or a compatible external compiler. See the *LabWindows/CVI Programmer Reference Manual* for more information on compatible external compilers.

- A dynamic-link library (`.dll`, Windows 3.1 only). Under Windows 95 and NT, each DLL must be accompanied by an import library (`.lib`) file. If you want to use a DLL instrument driver, the import library is the instrument driver program file.

- A path (`.pth`) file used to specify a `.dll` that is not in the same directory as the `.fp` file (Windows 3.1 only).

For example, the instrument module files for a Fluke 8840A multimeter are `fl8840a.fp`, `fl8840a.c`, and `fl8840a.h`.

You can load an instrument driver into the LabWindows/CVI interactive program whether the instrument program is in the form of a `.c`, `.obj` (`.o`), or `.lib` (`.a`) file. The presence of the `.h` file is essential, because it must be included in your program in order to reference global variables and constants in the instrument driver. Furthermore, the `.h` file prototypes the instrument driver functions that can be called externally.

VXI*plug&play* Instrument Driver Files

When you install a VXI*plug&play* instrument driver, the include (`.h`) file is not placed in the same directory as the `.fp` file. It is placed in the `include` subdirectory under the VXI*plug&play* directory. LabWindows/CVI can find the include files in the VXI*plug&play* include directory.

When you install VXI*plug&play* instrument driver, the source (.c) file is placed in the same directory as the `.fp` file. Under Windows, a dynamic-link library (`.dll`) and two import libraries (`.lib`), one compatible with Visual C/C++ and the other with Borland C/C++, are also distributed. These files are not placed in the same directory as the `.fp` file. The import libraries are placed in the `MSC` and `BC` subdirectories in the VXI*plug&play* `LIB` directory. The directory in which the DLL is placed is listed in the `PATH` environment variable so that the DLL can be found using the standard Windows DLL search algorithm.

Under Windows 95 and NT, if the `.fp` file is under the VXI*plug&play* framework directory and your current compatible compiler is Visual C/C++ or Borland C/C++, LabWindows/CVI can find the appropriate import library. (If your current compatible compiler is WATCOM C/++ or Symantec C/C++, LabWindows/CVI looks for the import library in the `WC` or `SC` subdirectory in the VXI*plug&play* `LIB` directory. However, import libraries for these two compilers are not normally distributed with VXI*plug&play* instrument drivers.) If LabWindows/CVI finds the import library, it gives it precedence over the `.c` file as the program file for the instrument driver.

Under Windows 3.1, the import library is 16-bit and thus cannot be used by LabWindows/CVI. LabWindows/CVI uses the program file found in the directory of the `.fp` file. If LabWindows/CVI cannot find a program file in the directory of the .fp file, it searches for a DLL using the standard Windows search algorithm. (It performs this search whether or not the `.fp` file is under the VXI*plug&play* framework directory.)

Loading/Unloading Instrument Drivers

You can load and unload instrument drivers manually using the **Instrument** menu. Instrument drivers loaded through the **Instrument** menu do not need to be listed in the project, and may be loaded/unloaded at any time.

You can incorporate instrument drivers into the project using the **Add to Project** command in the **File** menu of a Function Panel window or the Function Tree Editor window, or in the **Edit** menu of the Project window. The `.fp` file represents the instrument driver in the project list. If the `.fp` file is in the project list when the project is opened, the instrument driver is loaded automatically, and it is removed when the project is unloaded.

Precedence Rules for Loading the Instrument Driver Program File

Instrument drivers are loaded into LabWindows/CVI by loading the `.fp` file. This triggers LabWindows/CVI to load the instrument driver program file. In some cases, you might have an instrument driver program file in more than one format. For instance, you might have `fl8840a.obj` and `fl8840a.c` in the same directory. This can occur if you have obtained the source code for the instrument driver and compiled it. LabWindows/CVI chooses which file to load according to the following rules.

- If there is an instrument driver program file in the project, it is associated with the `.fp`. There can be at most one unexcluded program file with the same base name as the `.fp` file in the project list. Thus, `x.obj` (`x.o` under UNIX) and `x.c` both cannot be in the project list at the same time unless one or both are excluded.

- If you are working in Windows 95 or NT, and the following conditions apply,

  – the `.fp` file is under the VXI*plug&play* framework directory, and

  – there is a `.lib` file in the appropriate VXI*plug&play* framework subdirectory listed in the table below,

  then the `.lib` file is associated with the `.fp` file. The corresponding subdirectories appear in the following table.

Table 3-2. VXIplug&play Framework Subdirectories

Compatible Compiler	Corresponding VXIplug&play Framework Subdirectory Containing the .lib File
Visual C/C++	`LIB\MSC`
Borland C/C++	`LIB\BC`
WATCOM C/C++	`LIB\WC`
Symantec C/C++	`LIB\SC`

- Otherwise, if there is an instrument driver program file on disk in the same directory as the `.fp` file, it is loaded with the following precedence.
  - `.lib` under Windows or `.a` under UNIX
  - `.obj` under Windows or `.o` under UNIX
  - `.pth` (Windows 3.1 only)
  - `.dll` (Windows 3.1 only)
  - `.c`

  Note: ***In Windows 3.1, if the*** `.pth` ***file supplies a simple path to the*** `.dll` ***file, the*** `.dll` ***file is loaded using the search method specified in the documentation for the Windows SDK*** `LoadLibrary` ***function.***

- If you are working in Windows 3.1 and no program file has been found, LabWindows/CVI searches for a DLL with the same base name as the `.fp` file using the standard Windows search algorithm. (The algorithm is defined in the documentation for the Windows SDK `LoadLibrary` function.)

Loading an Instrument without an Instrument Program

You can load an `.fp` file for an instrument driver even if there is no program file for the instrument driver in the same directory as the `.fp` file. In this case, no program will be associated with the `.fp` file, but the `.fp` file appears in the **Instrument** menu.

This is useful if you want to use `.fp` files for documenting functions in your project. Because no program is associated with the `.fp` file, you cannot execute any function panels, but you can insert code into Source windows.

If you try to execute an instrument driver function panel when no program is associated with the instrument, LabWindows/CVI reports a run-time error. It is possible to associate a `.c` file with a `.fp` file after it is loaded. See the description of the **Edit** command for more information.

Modules Containing Non-Instrument Functions

Although the LabWindows/CVI instrument driver mechanism is primarily for modules that control instruments, it can be used by any module containing a set of high-level functions.

For instance, suppose you write a set of specialized analysis functions. If you develop function panels and a `.h` file for the module, you can then load the module from the **Instrument** menu and call the functions from the function panels, a Source window, or the Interactive Execution window.

Modifying an Instrument Driver

You may want to modify an instrument driver that you received from National Instruments or elsewhere. If you want to modify the instrument driver program file, you must have the `.c` file for the instrument driver.

Before modifying an instrument driver, familiarize yourself with the *LabWindows/CVI Instrument Driver Developers Guide.*

There are four parts of an instrument driver that you can modify:

- You can modify the function tree by selecting the `.fp` file using the **File » Open » Function Tree (.fp)** option in any window, or by selecting **Instrument** » **Edit** from any window containing the **Instrument** menu.
- You can modify the function panels by selecting **Option » Edit Function Panel Window** from a Function Panel window or **Edit » Edit Function Panel Window** from a Function Tree window.
- You can modify the instrument driver program file by selecting **Instrument » Edit** from any window containing the **Instrument** menu.
- You can modify the instrument driver include file by selecting the `.h` file using the **File » Open » Include (*.h)** option from any window.

Instrument Menu

The **Instrument** menu is a *dynamic* menu. It contains a list of the loaded instrument drivers as well as commands to load, unload, and edit instruments. When you load an instrument, its name is added to the list. When you unload an instrument, its name is removed from the list. When you select an instrument name in the **Instrument** menu, you can access its function panels.

For example, when two instruments have been loaded, the **Instrument** menu appears as shown in Figure 3-17.

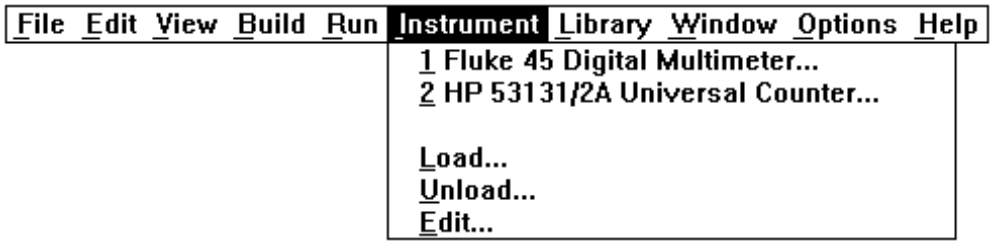

Figure 3-17. The Instrument Menu with Two Instruments Loaded

Load...

When you select the **Load** command, a dialog box appears that is similar to the dialog box displayed by the **Open** command in the **File** menu. In the Instrument Load dialog box, the filename `*.fp` appears in the text box. Instruments are always loaded through the `.fp` filename. You cannot load an instrument driver unless there is an `.fp` file for it.

When you specify an `.fp` file to load, LabWindows/CVI also looks for a program file with the same base filename in the same directory. If it finds one, it loads the instrument driver program along with the function panels.

For VXI*plug&play* instrument drivers, the program file may be in a different directory. See the *Priority of Loading the Instrument Driver File* section earlier in this chapter.

File Format Conversion

If the `.fp` file you are loading was created using LabWindows for DOS, a message appears indicating that the `.fp` file is being converted to the LabWindows/CVI format. You can use the dialog box that appears next to save the converted `.fp` file to disk.

Unload...

When you select the **Unload** command, a dialog box appears that contains a scrollable list of all the instruments you loaded with the **Load** menu. From this dialog box you may select one or more instrument drivers to unload.

Edit...

You can use the **Edit** command to edit an instrument driver program in a Source window or an instrument driver function tree in a Function Tree Editor window. When the **Edit** command is selected, the Edit Instrument dialog box shown in Figure 3-18 is invoked.

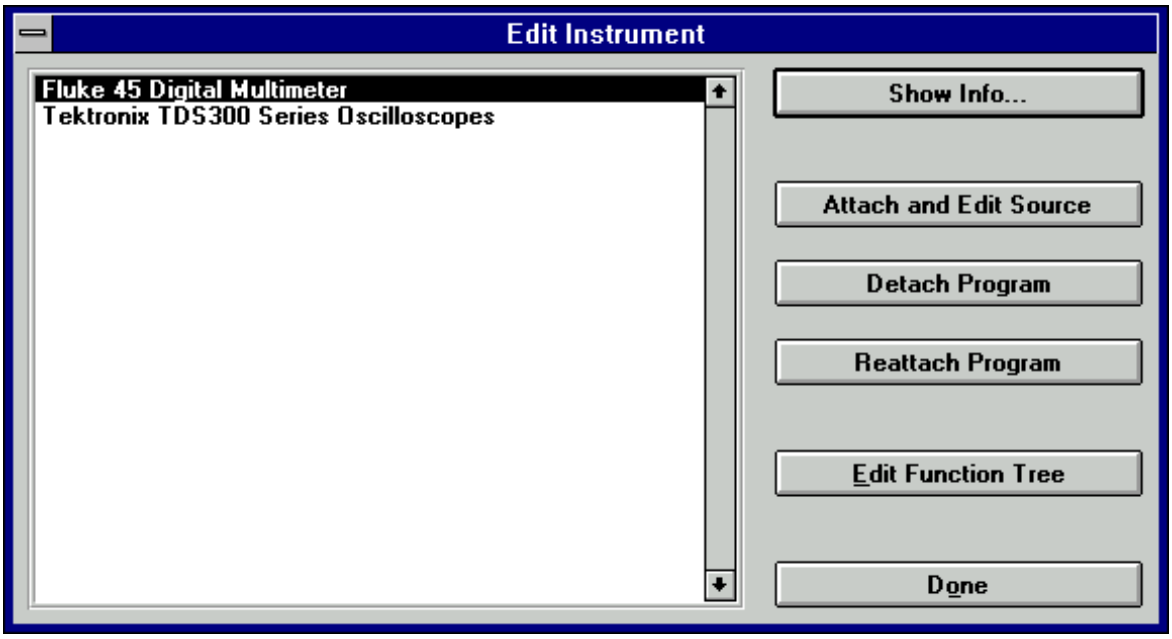

Figure 3-18. The Edit Instrument Dialog Box

The dialog box displays instrument drivers loaded as part of the project or through the **Instrument** menu. The commands in the Edit Instrument dialog box are as follows.

- **Show Info...**—Brings up the read-only Instrument Driver Information dialog box as shown in Figure 3-19.

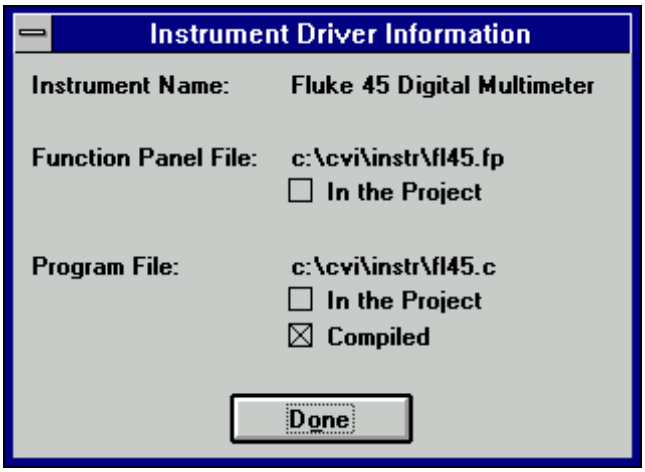

Figure 3-19. The Instrument Driver Dialog Box

- **Attach and Edit Source**—If a `.c` file with the same base name as the selected `.fp` file exists in the same directory as the `.fp` file, LabWindows/CVI loads, compiles, and attaches the `.c` file to the `.fp` file. The `.c` file is displayed in a new Source window.

- **Detach Program**—The instrument driver program file for the instrument driver is detached from the `.fp` file.
- **Reattach Program**—The current (if any) instrument driver program file is detached. A program file is then reloaded using the precedence rules outlined in the *Precedence Rules for Loading the Instrument Driver Program File* section earlier in this chapter.
- **Edit Function Tree**—A Function Tree Editor window appears with the function tree for the selected `.fp` file displayed.

Accessing Function Panels from the Instruments Menu

When you select an instrument name in the **Instrument** menu, the Select Function Panel dialog box appears. An example of an instrument driver dialog box is shown in Figure 3-20.

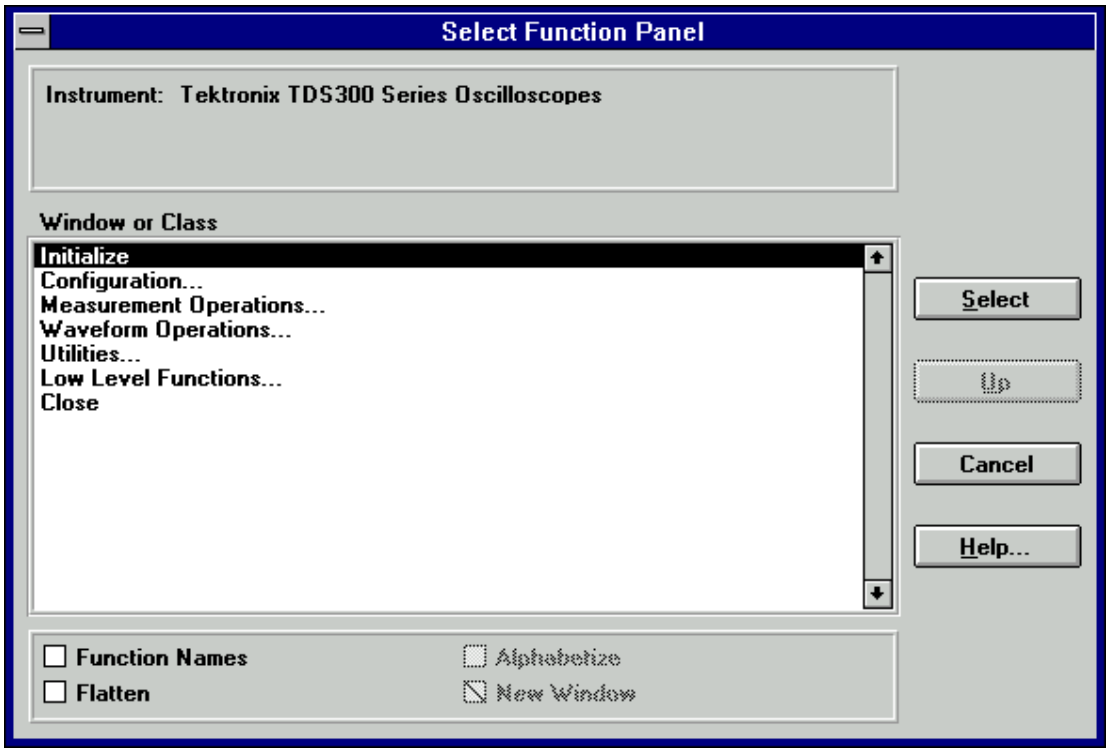

Figure 3-20. Select Function Panel Dialog Box

The Select Function Panel dialog box shows the function panels available in the driver you selected. Class names appear in the dialog box followed by ellipses (...). The ellipses indicate that more functions or classes of functions exist below that class name. If you select **Flatten**, class names are replaced with a list of all function panels at or below the current level.

Function Names shows the function names associated with each function panel. While in this mode, **Alphabetize** redisplays the function list in alphabetical order.

If **New Window** is selected, the next selected function panel appears in a new window; otherwise, it overwrites the current Function Panel window. This overrides the **Use only one function panel window** selection in the **Environment** command of the **Options** menu.

Select the desired class name to view the functions within a class. A class can contain other classes as well as functions. An instrument driver can contain up to four levels of classes and functions. Each time you select a class name, the function list is updated. Select the **Up** command button to return to the previous level.

When you select a function from a dialog box, that function panel appears. See Chapter 5, *Using Function Panels*, for more information about function panels.

To close the dialog box, select **Cancel**.

The dialog box contains a **Help** command button. You can get help information about the functions and classes listed in the dialog box. Select **Help** or press <F1> to display the Help dialog box shown in Figure 3-21.

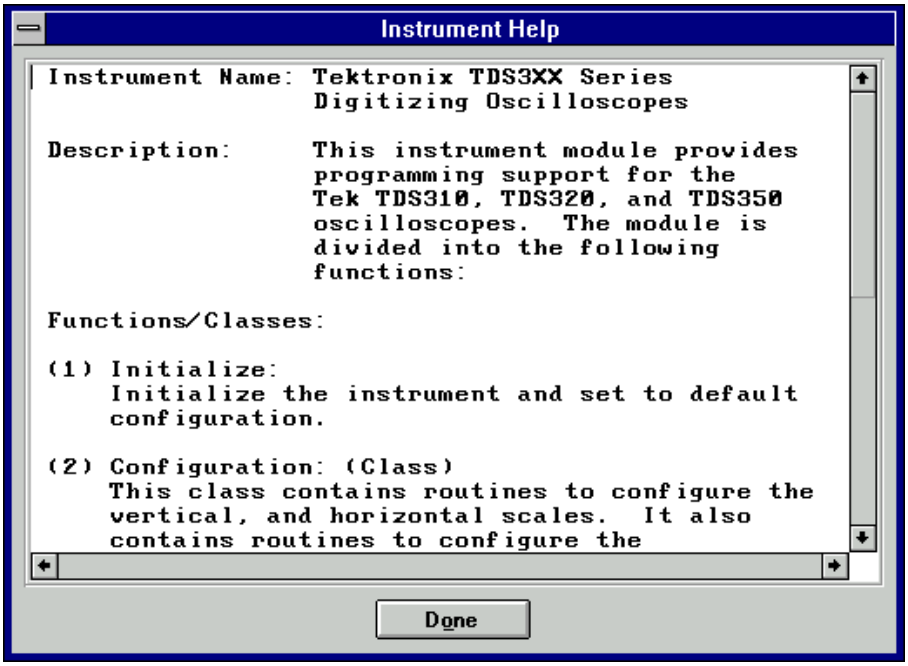

Figure 3-21. A Help Dialog Box

To close the Help dialog box, click on the **Done** command button, or press the <Enter> or <Esc> key.

Library Menu

This section explains how to use the commands in the **Library** menu for the Project window.

Use the **Library** menu commands to access function panels for the LabWindows/CVI libraries.

Library function panels are used to interactively run library functions and insert these function calls into any open Source window.

Figure 3-22 shows the **Library** menu.

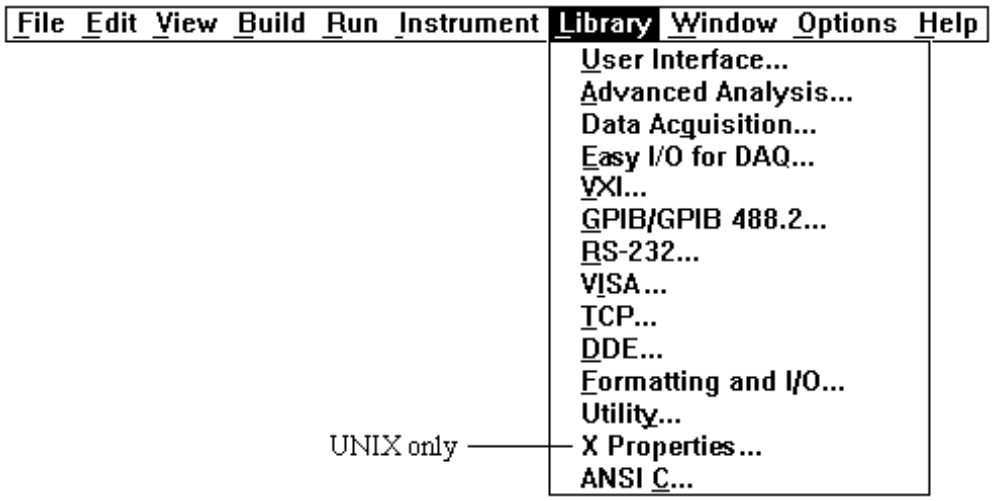

Figure 3-22. The Library Menu

When you select a library name in the **Library** menu, you can access the library function panels in the same way you access instrument driver function panels when you select instrument names in the **Instrument** menu. For more information, see the *Accessing Function Panels* section of Chapter 5, *Using Function Panels.*

User Interface...

The User Interface Library is a set of functions for controlling a graphical user interface. The user interface is composed of a collection of objects such as menu bars, panels, controls, and graphs. The user interface may be constructed programmatically or with the User Interface Editor. See the *LabWindows/CVI User Interface Reference Manual* for more information on creating and controlling a user interface.

Analysis...

The Analysis Library includes functions for one-dimensional (1D) and two-dimensional (2D) array manipulation, complex operations, matrix operations, and statistics. See Chapter 3, *The*

Analysis Library, in the *LabWindows/CVI Standard Libraries Reference Manual* for analysis function descriptions.

The Advanced Analysis Library also includes functions for signal generation, signal processing, and curve fitting. See the *LabWindows/CVI Advanced Analysis Library Reference Manual* for advanced analysis function descriptions.

Data Acquisition... (Windows Only)

Use the Data Acquisition Library to control your National Instruments data acquisition boards. These include the AT and PC series boards for the IBM PC AT and compatibles that use the AT bus, and the EISA-A2000 for EISA bus computers. See the *NI-DAQ User Manual for PC Compatibles* and the *NI-DAQ Function Reference Manual for PC Compatibles* included with your data acquisition hardware, for a software overview and function descriptions.

Easy I/O for DAQ... (Windows Only)

The Easy I/O for DAQ Library contains functions which make writing simple DAQ programs easier than if you use the Data Acquisition Library. For detailed descriptions of the Easy I/O for DAQ functions, see Chapter 10, in the *LabWindows/CVI Standard Libraries Reference Manual*.

VXI...

Use the VXI Library to control your VXI instruments from a National Instruments embedded VXI controller or a PC equipped with a MXI controller. The VXI Library includes functions for Commander and Servant Word Serial Protocol, low-level VXIbus access, local resource access, VXI signals, interrupts, triggers, system interrupt handlers, and system configuration. See the *NI-VXI Software Reference Manual for C*, included with your LabWindows/CVI VXI Development system, for a software overview and function descriptions.

GPIB/GPIB 488.2...

Use the GPIB Library to create an interface to GPIB instruments using the NI-488/488.2 protocol. You must have a National Instruments GPIB 488.2 controller board for the IBM PS/2 line of personal computers, the IBM PC AT or compatible, or the Sun SPARCstation to use the GPIB Library. See the *NI-488.2 Software Reference Manual* (or *NI-488.2M Software Reference Manual*), included with your GPIB 488.2 controller board, for a software overview and function descriptions. See Chapter 4**,** *The GPIB and GPIB-488.2 Libraries*, in the *LabWindows/CVI Standard Libraries Reference Manual* for a description of the Device Manager routines, which ensure against device name conflicts when using instrument drivers.

RS-232...

Use the RS-232 Library to control the RS-232 serial ports on the PC and the SPARCstation. See Chapter 5, *The RS-232 Library*, in the *LabWindows/CVI Standard Libraries Reference Manual* for function descriptions.

VISA...

The VISA Library gives VXI and GPIB software developers, particularly instrument driver developers, a single interface library for controlling VXI, GPIB, and RS-232 instruments. The functions are described in the *NI-VISA Programmer Reference Manual*, which is available upon request.

TCP...

Use the Transmission Control Protocol (TCP) Library to control the TCP network cards on the PC and the SPARCstation. See Chapter 7, *The TCP Library*, in the *LabWindows/CVI Standard Libraries Reference Manual* for function descriptions.

X Property... (UNIX Only)

Use the X Client Property Library to communicate among X clients by reading and writing properties to and from X Windows. See Chapter 9, *X Properties*, in the *LabWindows/CVI Standard Libraries Reference Manual* for function descriptions.

DDE... (Windows only)

Use the Dynamic Data Exchange (DDE) Library to create an interface with other Windows applications using the DDE standard. See Chapter 6, *The DDE Library*, in the *LabWindows/CVI Standard Libraries Reference Manual* for function descriptions.

Formatting and I/O...

Use the Formatting and I/O Library functions to input and output data to files and manipulate the format of data in a program. See Chapter 2**,** *The Formatting and I/O Library*, in the *LabWindows/CVI Standard Libraries Reference Manual* for function descriptions.

Utility...

The Utility Library contains functions that do not fit into any of the other LabWindows/CVI libraries. You can use these functions for file manipulation and other miscellaneous tasks. See

Chapter 8**,** *The Utility Library*, in the *LabWindows/CVI Standard Libraries Reference Manual* for function descriptions.

ANSI C...

The ANSI C Library contains the ANSI C functions available in LabWindows/CVI. See Chapter 1, *The ANSI C Library*, in the *LabWindows/CVI Standard Libraries Reference Manual* for more information.

User Libraries

You may install your own libraries into the **Library** menu. A user library has the same form as an instrument driver. Anything that can be loaded with the **Instruments** menu can be loaded as a user library, provided the program is in compiled form. See the *Instrument Driver Files* section earlier in this chapter for more information on loading files with the **Instrument** menu. The main difference between modules loaded as instrument drivers and those loaded as user libraries is that instrument drivers can be unloaded with the **Unload** command in the **Instrument** menu, but user libraries cannot be unloaded. Also, because user libraries must be in compiled form, they cannot be edited while they are loaded as libraries. See the *LabWindows/CVI Instrument Library Developers Guide* for detailed information about writing an instrument driver.

You install user libraries with the **Library Options** command in the **Options** menu. The next time you enter LabWindows/CVI, the libraries load automatically and appear at the bottom of the **Library** menu.

Dummy .fp Files for Support Libraries

You can develop a user library module to provide support functions for instrument drivers or any modules in your project. In such a case, function panels may not be necessary for the library module. If the `.fp` file for the library module contains no classes or functions, the name does *not* appear in the **Library** menu when the library is loaded.

System Libraries

If you are running under Microsoft Windows, some low-level system functions are included. The prototypes for the functions are in `lowlvlio.h`. No function panels are available.

Under UNIX, the Host System Library `/lib/libc.a` is included. Host System Library functions should be called primarily from compiled modules because there are no header files, function panels, or user protection available for this library. For more information on using UNIX system functions, see the *Calling the Sun C Library from Source Code* section in Chapter 1 of the *LabWindows/CVI Programmer Reference Manual.*

Window Menu

Commands in the **Window** menu, shown in Figure 3-23, are used for bringing any open window to the front for viewing or editing.

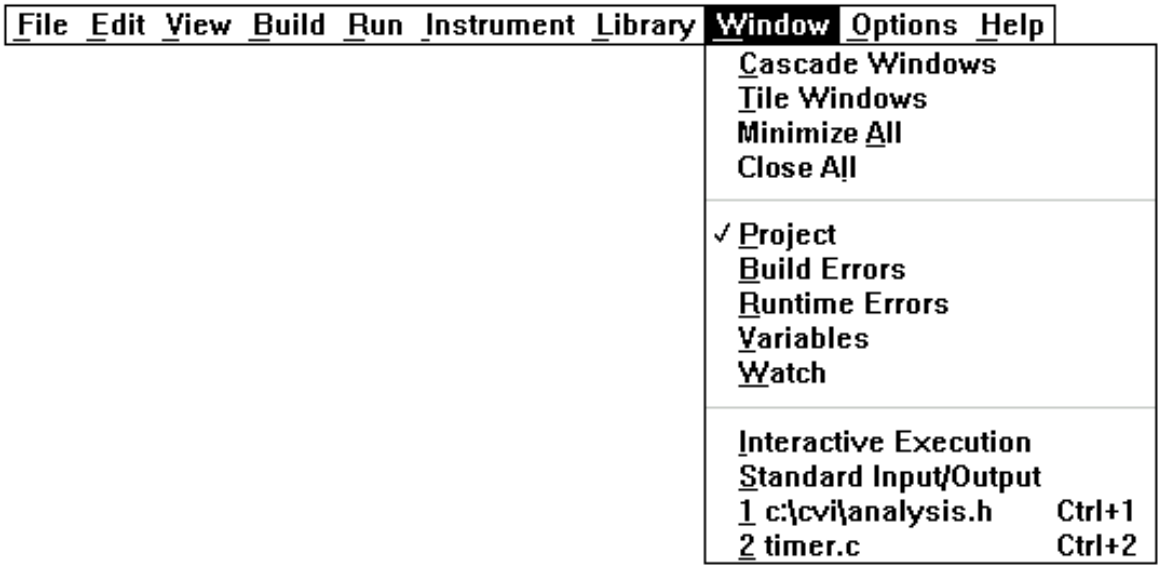

Figure 3-23. The Window Menu

Cascade Windows

Use this command to arrange all open windows so that each title bar is visible.

Tile Windows

Use this command to arrange all open windows in smaller sizes to fit next to each other.

Minimize All (Windows 95 only)

The Minimize All command hides all of the LabWindows/CVI windows, including the Project window and any User Interface Library panels displayed by a program you are running in LabWindows/CVI. You can restore the windows by clicking on LabWindows/CVI in the Windows 95 task bar.

CloseAll

The **Close All** command closes all of the LabWindows/CVI windows, excluding the Project window and any User Interface Library panels displayed by a program you are running in LabWindows/CVI. It works on all platforms.

Project

Use this command to bring the Project window to the front.

Build Errors

If you attempt to build a project and the project has build errors, such as syntax or link errors, the Build Errors window contains a list of the errors. The **Build Errors** command in the **Windows** menu brings the Build Errors window to the front for viewing.

Runtime Errors

If you attempt to run a project and the project has runtime errors, such as over indexing an array, the Runtime Errors window contains a list of the errors. The **Runtime Errors** command in the **Windows** menu brings the Runtime Errors window to the front for viewing.

Variables

Use this command to bring the Variable Display window to the front. The Variable Display window shows the contents of all variables currently defined in LabWindows/CVI. You can access the Array Display and String Display windows from the Variable Display window.

The Variable Display window is useful for debugging programs. The Variable Display window is updated at each breakpoint, and you can modify the variables while in a breakpoint state.

See Chapter 6, *The Variable Display and Watch Windows*, for more information about the Variable Display window.

Watch

The **Watch** command brings the Watch window to the front. The Watch window shows a user-defined set of variables and expressions. You can access the Array Display and String Display windows from the Watch window.

The Watch window is useful for debugging programs. Watch variables and expressions will update at each breakpoint unless they are set to update continuously.

See Chapter 6, *The Variable Display and Watch Windows*, for more information about the Watch window.

Array/String Displays

If there are any Array/String Display windows active, they will appear in the **View** menu. Selecting one of these windows from the **View** menu brings it to the front for viewing and editing. Array/String Display windows are described in Chapter 7, *Array Display and String Display Windows*.

User Interface

All open User Interface Resource (`.uir`) files are dynamically shown in the **Window** menu. If the file is in the project, only the file name is shown. If the file is not in the project, the full path name is shown. Selecting a `.uir` file from this menu brings the file to the front in a User Interface Editor window. See the *LabWindows/CVI User Interface Reference Manual* for a complete description of User Interface Editor windows.

Function Panel

All open Function Panel windows are dynamically shown in the **Window** menu. Selecting a Function Panel window from this menu will bring that window to the front. See Chapter 5, *Using Function Panels*, for a complete description of Function Panel windows.

Function Tree

All open Function Tree files are dynamically shown in the **Window** menu. If the file is in the project, only the file name is shown. If the file is not in the project, the full path name is shown. Selecting an `.fp` file from this menu will bring that Function Tree window to the front. See Chapter 2, *The Function Tree Editor*, in the *LabWindows/CVI Instrument Driver Developers Guide* for a complete description of Function Tree windows.

Help Editor

All open Help Editor windows are dynamically shown in the **Window** menu. Selecting a Help Editor window from this menu will bring that Help Editor window to the front. See Chapter 4, *Adding Help Information*, in the *LabWindows/CVI Instrument Driver Developers Guide* for more information about the Help Editor.

Interactive Execution

Use the **Interactive Execution** command to bring the Interactive Execution window to the front. You can execute selected portions of code with the Interactive Execution window. You do not need to have a complete program in the Interactive Execution window, as is the case in a Source window. For example, you can execute variable declarations and assignment statements in C without declaring a main function.

This window is described in detail in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*.

Standard Input/Output

Use the **Standard Input/Output** command to bring the Standard Input/Output window to the front. You can print text messages and receive user input from the keyboard using the Standard Input/Output window.

This window is described in detail in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*.

Open Source Files

The **Window** menu lists any open source files at the bottom. If the file is in the project, only the file name is shown. If the file is not in the project, the full path name is shown. Selecting a source file from this menu brings the source file to the front in a Source window.

Source windows are described in detail in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*.

Options Menu

Commands in the **Options** menu are used for setting up preferences in the LabWindows/CVI environment, and executing various utilities.

This section explains how to use the commands in the Project Window **Options** menu, shown in Figure 3-24.

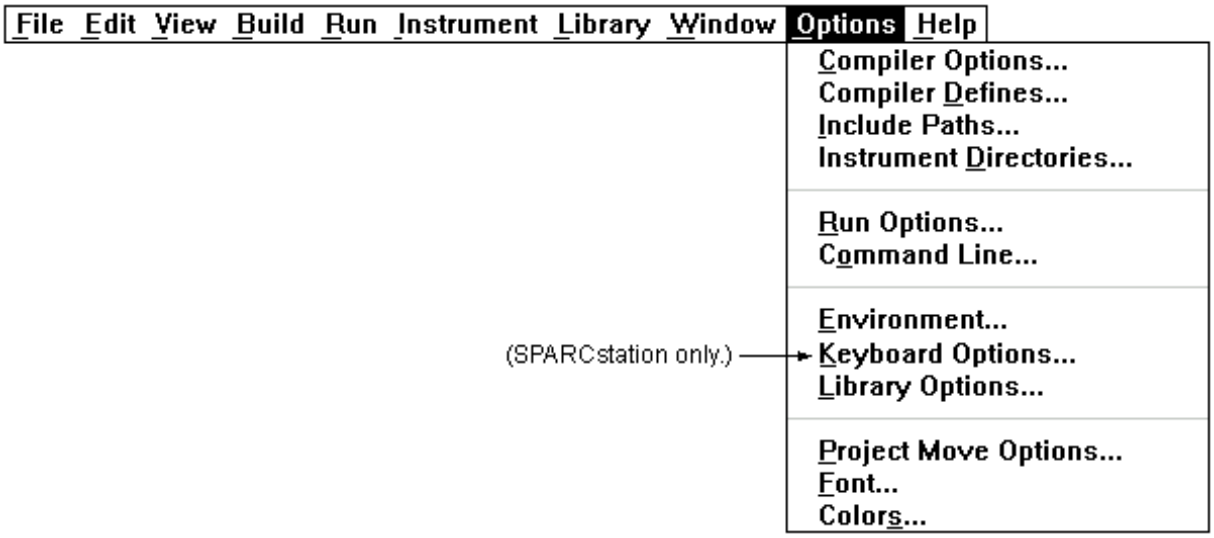

Figure 3-24. The Options Menu

Compiler Options...

This command invokes a dialog box which allows you to set the following LabWindows/CVI compiler options.

- For Windows 95 and NT, the compatible compiler is indicated in the **Compiler Options** dialog box. For more information on external compiler compatibility, see Chapter 3, *Windows 95 and NT Compiler/Linker Issues*, in the *LabWindows/CVI Programmer Reference Manual*.

- For Windows 95 and NT, the default calling convention can be changed in the **Compiler Options** dialog box, unless the compatible compiler is WATCOM. For the other compilers, the default calling convention is normally `cdecl` but can be changed to `stdcall`. For WATCOM, it is always the stack-based calling convention. For more information on external compiler compatibility, see Chapter 3, *Window 95 and NT Compiler/Linker Issues*, in the *LabWindows/CVI Programmer Reference Manual*.

  Note: ***It is recommended that you do* not *use*** `stdcall` ***as the default calling convention if you plan to create an object file or static library for each of the four external compilers.***

- **Maximum number of compile errors** sets an upper limit on the number of compiler errors to be listed in the Build Errors window.

- **Require function prototypes** requires all function references to be preceded by a full prototype declaration. A full prototype is one that includes the function return type as well as the types of each parameter. If a function has no parameters, a full prototype must have the `void` keyword to indicate this case. A *new style* function definition (one in which parameters are declared directly in the parameter list) serves as a prototype.

Missing prototype errors can occur at the following places.

– Typedefs such as `typedef void FUNTYPE()`.

– Function pointer declarations such as `void (*fp)()` whether used as a global, local, parameter, array element or structure member.

– *Old style* function definitions (one in which parameters are declared outside of the parameter list) that are not preceded by a full prototype.

– Function call expressions such as `(*fp)()`, where `fp` does not have a full prototype.

Caution: ***Be sure you enable the* Require Function Prototypes *option. If disabled, some of the runtime error checking will also be disabled.***

- **Require return values for non-void functions** generates compile warnings for non-void functions (except `main`) that do not end with a `return` statement returning a value. LabWindows/CVI reports a runtime error when a non-void function executes without returning a value.

  For example, the following code produces a compile-time warning and will produce a run-time error when `(g == 6)` is not true.

```
int fn (void)
{
   if (g == 6)
      return 20;
   g = 7;
}
```

- **Enable signed/unsigned pointer mismatch warning** generates a compiler warning for pointer assignments in which the left side and right side are not both `signed` or `unsigned` expressions. According to the ANSI C standard, these assignments should be errors because they involve incompatible types. In practice, however, assigning a pointer to `unsigned` *type*, the value of a pointer to `signed` *type*, or vice versa, causes no real problems.

- **Enable unreachable code warning** generates a compiler warning for statements that are not reachable in a function.

- **Track include file dependencies** keeps the project up to date by tracking the dependencies between source files and include files. Whenever a file is modified, LabWindows/CVI marks for compilation all project source files that include the modified file.

- **Prompt for include file paths** sets LabWindows/CVI to prompt you to make a manual search for any header files listed in the `#include` lines that the compiler cannot find. Once found, you may automatically insert the appropriate path into the Include Paths list for the project.

- **Stop on first file with errors** sets the LabWindows/CVI compiler to terminate compilation after one file is found to have errors. This way you can fix build errors in your project one file at a time.

- **Show Build Error Window for Warnings** sets the LabWindows/CVI compiler to bring up the Build Error window when warnings occur, even if there are no errors. If it is deactivated, warnings can occur without being brought to your attention.

- **Display status dialog during build** displays a status box during the build, showing the name of the file being compiled, the number of errors and warnings encountered, and a percent complete value. Disabling this feature speeds up compiling.

Compiler Defines...

The LabWindows/CVI compiler accepts compiler defines through the **Compiler Defines** command in the **Build** menu of the Project window.

Compiler defines have the following syntax.

```
/Dx or /Dx=y
```

The variable `x` is a valid C identifier. `x` can be used by the `#if` and `#ifdef` preprocessor directives for conditional compilation or as a predefined macro in your source code. If `y` contains embedded blanks, it must be surrounded by double quotes.

LabWindows/CVI predefines macros to help you write platform dependent code.

- `_CVI_` is defined to be 1 in version 3.0, 301 in version 3.0.1, and 310 in version 3.1.

- `_NI_mswin_` is defined if compiling with Windows 3.x, Windows 95, or Windows NT.

- `_NI_mswin16_` is defined if compiling with Windows 3.x.

- `_NI_mswin32_` will be defined in future supported versions for Windows 95 and Windows NT.

- `_NI_i386_` is defined if compiling on a PC.

- `_NI_unix_` is defined if compiling under UNIX.

- `_NI_sparc_` is defined if compiling under UNIX. The value of `_NI_sparc_` is 1 if you are running under Solaris 1.x and 2 if you are running under Solaris 2.x.

- `_CVI_DEBUG_` is defined if compiling with the debugging level set to **Standard** or **Extended**. The value of the macro is 1.

The following predefined macros are defined for Windows 95 and NT.

- `_CVI_EXE_` is defined if the project target type is Standalone Executable
- `_CVI_DLL_` is defined if target type is Dynamic Link Library
- `_CVI_LIB_` is defined if target type is Static Library
- `__DEFALIGN` is defined to the default structure alignment (8 for Microsoft and Symantec, 1 for Borland and WATCOM)
- `_NI_VC_` is defined to 220 if in Microsoft Visual C/C++ compatibility mode
- `_NI_SC_` is defined to 720 if in Symantec C/C++ compatibility mode
- `_NI_BC_` is defined to 451 if in Borland C/C++ mode
- `_NI_WC_` is defined to 1050 if in WATCOM C/C++ mode
- `_WINDOWS` is defined
- `WIN32` is defined
- `_WIN32` is defined
- `__WIN32__` is defined
- `__NT__` is defined
- `_M_IX86` is defined to 400
- `_NI_mswin32_` is defined
- `__FLAT__` is defined to 1

For Windows 95 and NT, the default compiler defines string now contains,

```
/DWIN32_MEAN_AND_LEAN
```

to reduce the time and memory taken by compiling Windows SDK include files. For more information, see Chapter 3, *Windows 95 and NT Compiler/Linker Issues*, in the *LabWindows/CVI Programmer Reference Manual*.

For all platforms, the **Compiler Defines** dialog box now contains a list of the macros predefined by LabWindows/CVI. This list includes the name and value of each predefined macro.

Include Paths...

The **Include Paths** command invokes a dialog box in which you can list paths that the compiler uses when searching for header files with simple path names. The Include Paths list is stored in

the project file. The **Include Paths** dialog box has two lists of paths in which to search for include files. The top list is saved with the project file. The bottom list is saved from one LabWindows/CVI session to another on the same machine, regardless of project.

Instrument Directories...

The **Instrument Directories** command invokes a dialog box in which you can list directories that LabWindows/CVI uses to search for instruments that are referenced by other instruments. When loading an instrument `.fp` file that references other instrument `.fp` files, LabWindows/CVI tries to find the referenced instrument `.fp` files and load them. It searches for each `.fp` file in the following directories and in the following order.

1. The directory of the referencing `.fp` file.

2. The directories listed in the Instrument Directories dialog box.

3. The `instr` directory in the directory where LabWindows/CVI is installed.

The Instrument Directories list is maintained in the `cvi.ini` (`.cvi.ini` under UNIX) file associated with your installation of LabWindows/CVI. Thus, it applies to all of your CVI sessions, regardless of which project you have loaded.

To find out more about referencing one instrument from another, refer to *.FP Auto-Load List* in the *Edit* section of Chapter 3, *The Function Tree Editor Menu Bar* in the *LabWindows/CVI Instrument Driver Developers Guide*.

The Instrument Directories are also used when you load a project file that has been moved since it was last saved. If an `.fp` file listed in the project cannot be found using either its original pathname in the project or its location relative to the project, LabWindows/CVI searches the Instrument Directories and the `cvi/instr` directory for an `.fp` file with the same base name.

Run Options...

The **Run Options** command invokes a dialog box you use to set the following options.

- **Maximum stack size (bytes)**—The stack is used for passing function parameters and storing automatic local variables. By setting a stack limit, LabWindows/CVI can catch runaway recursive functions and report a stack overflow. See the *Stack Size* section in Chapter 1, *LabWindows/CVI Compiler*, in the *LabWindows/CVI Programmer Referene Manual*, for platform-specific limitations on the stack size.

- **Debugging level**—There are three debugging levels for the source modules in your application.

  – **None**—Source modules execute faster without debugging, but you sacrifice the ability to set breakpoints, or to use the Variable Display window. Also, there is no user protection to check for using bad pointers, over indexing arrays, invalid array sizes, and so on.

  – **Standard**—In this mode, you can set breakpoints, use the Variable Display window, and you have user protection.

  – **Extended**—This mode has the same benefits as **Standard** mode, with added user protection that validates every attempt to free dynamically allocated memory by verifying that the address passed is actually the beginning of an allocated block.

- **Save Changes before running**—You can configure LabWindows/CVI to *never* save modified files, to *always* save modified files, or to *ask* whether to save modified files before running.

- **Break on library errors**—When this checkbox is activated, a breakpoint occurs when any function call to a LabWindows/CVI library results in an error.

- **Hide Windows**—When this checkbox is activated, all LabWindows/CVI system windows are hidden until execution terminates or a breakpoint occurs.

- **Check Disk Dates Before Each Run**—When you activate the **Check Disk Dates Before Each Run** option, LabWindows/CVI automatically compares, before each execution of the project, the disk dates of all program files used to build the project with the dates CVI last loaded or saved them. The program notifies you of discrepancies and gives options to resolve them. This option causes a small delay before each project execution, but is useful when you are modifying program files (such as DLLs) in applications other than CVI. You can invoke the same action manually by selecting the **Update Program Files From Disk** command in the **Build** menu.

- **Unload DLLs After Each Run (Windows 95/NT Only)**—When this checkbox is activated, DLLs are unloaded after each execution of a user program in the development environment.

- **Reload DLLs On Each Run (Windows 3.1 Only)** —When activated, this checkbox has the following effects.

  – If you are loading DLLs directly, the program regenerates the DLL glue code and reloads the DLLs each time your project runs.

  – If you are loading DLL glue code instead of loading DLLs directly, the program reloads the glue code and the DLLs each time your project runs.

Environment...

The **Environment** command invokes a dialog box you use to set the following conditions.

- **Sleep policy when not running program**—Each time LabWindows/CVI checks for an event from the operating system, it may put itself in the background (into *sleep* mode) for a specified period of time. While LabWindows/CVI is in sleep mode other applications have more processor time. However, LabWindows/CVI may run slower. You can specify how much LabWindows/CVI sleeps when a user program is not running. (When a user program is running, sleeping is controlled by the `SetSleepPolicy` function in the User Interface Library.) You have the following sleep policy choices.

  – Do not sleep (the default setting)

  – Sleep some (sleep a short period of time)

  – Sleep more (sleep a longer period of time)

  The setting that is optimal for you depends on the operating system you are using and the other applications you are running. If you think you may need to make an adjustment, try the different settings and observe the resulting behavior.

- **Maximum number of lines in Standard Input/Output Window**—You can specify a ceiling on the number of lines maintained in the Standard Input/Output window.

- **Bring Standard Input/Output Window to front whenever modified**—Activate this checkbox for the Standard Input/Output window to appear whenever functions such as `printf` or `FmtOut` append text to it.

- **Use only one function panel window**—Activate this checkbox to overwrite the current function panel window each time a new function panel window is selected.

- **Use host system's standard I/O (UNIX Only)** —Activate this checkbox to use the host Standard I/O instead of the LabWindows/CVI Standard Input/Output window.

- **Warp mouse over dialog boxes (UNIX Only)** —Activate this checkbox to automatically move the mouse over newly displayed dialog boxes.

Keyboard Options... (SPARCstation Only)

The **Keyboard Options** command invokes a dialog box you use to select <Control> or <Meta> as your modifier key for accessing menus.

Library Options...

You use the **Library Options** command to specify user libraries that are loaded automatically when you start LabWindows/CVI. You can also specify which of the optional National Instruments libraries are loaded on start up using the **Library Options** command. The **Library Option** command invokes the dialog box shown in Figure 3-25.

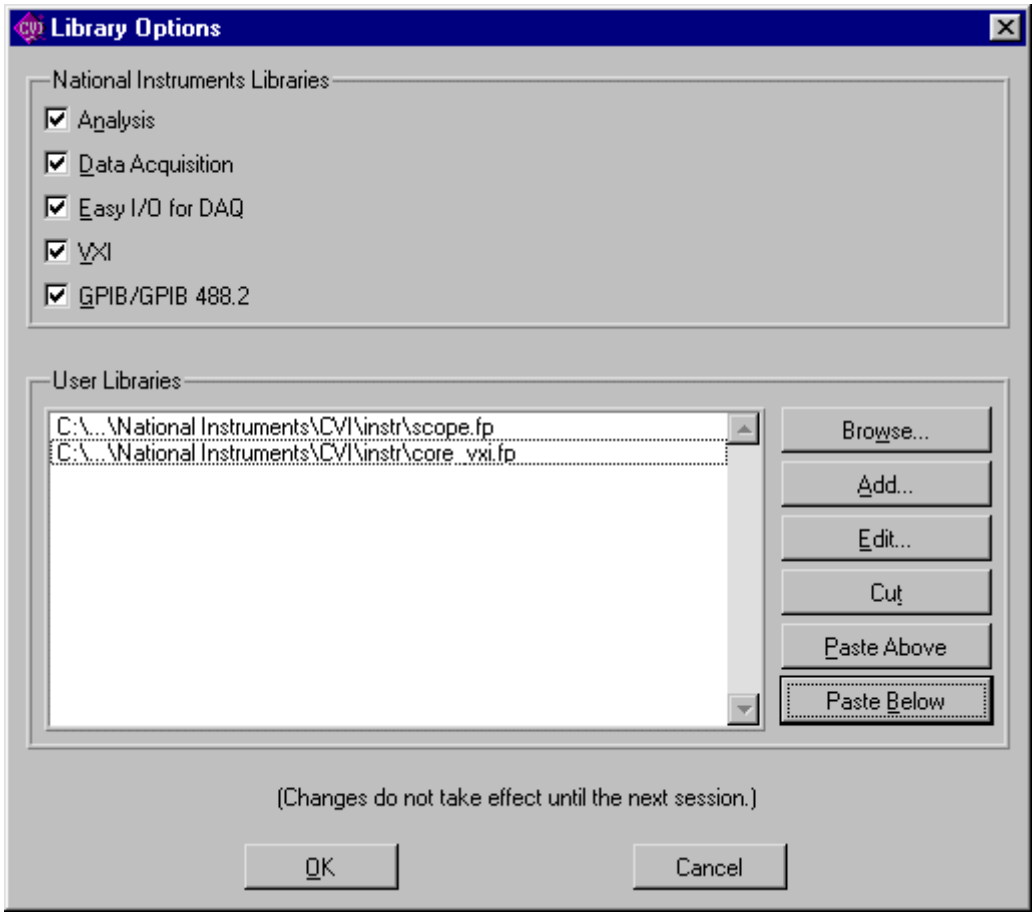

Figure 3-25. The Library Options Dialog Box

This dialog box contains two different areas: National Instruments Libraries and User Libraries.

User Libraries

The User Libraries list contains the pathnames of user libraries loaded automatically when you start LabWindows/CVI. The entries must be the pathnames of the function panel (`.fp`) files for the library modules. Because a user library has the same form as an instrument driver, anything that can be loaded from the **Instrument** menu can also be loaded as a user library, provided it is in compiled form. See the *Instrument Driver Files* discussion in the *Using Instrument Drivers* section for more information about loaded `.fp` files.

The main difference between modules loaded as instrument drivers and those loaded as user libraries is that while instrument drivers can be unloaded using the **Unload** command in the **Instrument** menu, user libraries cannot. Also, because user libraries must be in compiled form, they cannot be edited while they are loaded as libraries. See the *LabWindows/CVI Instrument Driver Developers Guide* for information about writing an instrument driver.

User libraries are installed using the **Library Options** command in the **Options** menu. Once installed, the libraries are loaded automatically and appear at the bottom of the **Library** menu each time you launch LabWindows/CVI.

Dummy .fp Files for Support Libraries

You can develop a user library module to provide support functions for instrument drivers or any modules in your project. In such a case, function panels may not be necessary for the library module. If the `.fp` file for the library module contains no classes or functions, the name does *not* appear in the **Library** menu when the library is loaded.

National Instruments Libraries

There are five optional National Instruments libraries—Advanced Analysis, Data Acquisition, Easy I/O for DAQ, VXI, and GPIB/GPIB 488.2. In the National Instruments section, click on the selection box to the left of the library name(s) that you want LabWindows/CVI to load when you start up. Use the National Instruments Libraries section of the **Library Options** dialog box to specify whether to load these libraries into memory when LabWindows/CVI is started. A checkmark next to the library name indicates that you want the library to be loaded. Changes do not take effect until the next time you start LabWindows/CVI.

If a library is not loaded, you cannot call any of the functions in that library, and you cannot access any of its function panels.

If LabWindows/CVI fails to load a requested library, it is probably because LabWindows/CVI cannot find the appropriate files. The following files belong in the `bin` directory in the LabWindows/CVI installation directory.

Table 3-3. Libraries in the Bin Directory of LabWindows/CVI

Library	Windows 95/NT	Windows 3.1	UNIX
Analysis or Advanced Analysis	`analysis.lib`	`analysis.lib`	`analysis.a`
Data Acquisition	`dataacq.lib`	`dataacq.obj`	(not available)
Easy I/O for DAQ	`easyio.lib`	`easyio.lib`	(not available)
VXI	`nivxi.lib`	`nivxi.lib`	`nivxi.a`
GPIB/GPIB 488.2	`gpib.lib`	`gpib.obj`	`gpib.o`

Also, you must install the Windows or UNIX drivers that came with your Data Acquisition, VXI, and GPIB/GPIB 488.2 hardware to use the optional libraries.

Project Move Options

This command invokes a dialog box that you use to set the following options. These options take effect when the `.prj` file is opened from a different directory than that from which it was saved.

- **Fixup Pathnames when Project is Moved**—When selected, this option makes LabWindows/CVI update the pathnames in the project, the project include paths, the pathnames in the distribution kit, and the pathnames of the standalone executable and icon files when a user opens the project from a directory other the one in which the project was saved. It is assumed that all of these pathnames have been moved to the same position relative to the `.prj` file. Pathnames may also be modified if the CVI directory or the VXI*plug&play* directory has changed since the project was last saved. If `.fp` files in the project cannot be found, LabWindows/CVI searches for them in the directories listed in the Instrument Directories dialog box and in the `cvi/instr` directory. If LabWindows/CVI cannot find one or more files in the project or project include paths, a dialog box appears prompting you to manually confirm those pathnames.

  If you disable this option, LabWindows/CVI does not update pathnames when the project is moved, and you will have to re-create the `.prj` file and associated pathnames manually.

The **Always Prompt Before Fixing Pathnames** option no longer exists.

Font

Use the **Font** command to select the font and font size for the text in the Project window.

Colors...

Use the **Colors** menu item to select colors for the Project window, Source windows, Interactive Execution window, Standard I/O window, Watch window, Variable Display, String Display, and Array Display. (Dialog boxes, function panels, and the User Interface Editor window are not affected).

The **Colors** menu item brings up a dialog box containing a list box. Each line in the list box describes the purpose of the color and shows its current color state. To change the color, click and hold on the color control at the bottom left of the dialog box. A color pop-up palette appears. While holding the mouse button down, move the pointer over the desired color and then release. The color change takes effect immediately in all CVI windows currently visible.

To change all of the colors to their default state, click on **Default** . The color changes are reflected immediately in all CVI windows currently visible.

If you want to accept these changes, press the **OK** button. If you want to revert to the state before the dialog box appeared, click on **Cancel**.

You can also access the **Colors** command in the Source window, the Interactive Execution window, and the Standard I/O window.

The Color dialog box also contains eight color types for syntax coloring. Refer to the *Syntax Coloring* section in Chapter 4*, Source, Interactive Execution and Standard Input/Output Windows*.

A new checkbox, **Use System Colors**, has been added for Windows 95. When enabled, this option removes from the list box several color types associated with the Project window, Source window, and scroll bars. Colors are automatically assigned to these types based on the system colors defined in the Appearance tab in the Windows 95 Display Properties dialog box.

Chapter 4
Source, Interactive Execution, and Standard Input/Output Windows

This chapter describes the LabWindows/CVI Source, Interactive Execution, and Standard Input/Output windows. Each of these windows supports specific tasks related to developing and executing programs.

Source Windows

Source windows display the source code for the program you are developing. The windows behave like standard text editors. You can type text directly into a Source window or load text from an ASCII file. You can insert code from LabWindows/CVI function panels directly into Source windows. You can save a program from a Source window as an ASCII file. Source windows can contain up to one million lines with up to 254 characters in each line. A tab is considered to be one character.

When you run a program in a Source window, the program must be complete and obey the syntax rules of ANSI C. See *The Build Menu* and *The Run Menu* sections in this chapter for more information on running programs.

Toolbars in LabWindows/CVI

The LabWindows/CVI toolbar appears within function panels, in the function panel editor window, and in source windows. Using the toolbar gives you quick access to common commands, such as File Open and File Save. You can configure the toolbar to meet your needs and you can also choose not to display it.

To find out what a toolbar button does, use Toolbar Help to display the name of a toolbar button. Position the mouse cursor over that button, and either hold the cursor there for a short period of time, or click on the toolbar button with the right mouse button.

Modifying Your Toolbars

To modify a toolbar, choose **Options** » **Toolbars...** to display the Customize Toolbar dialog box.

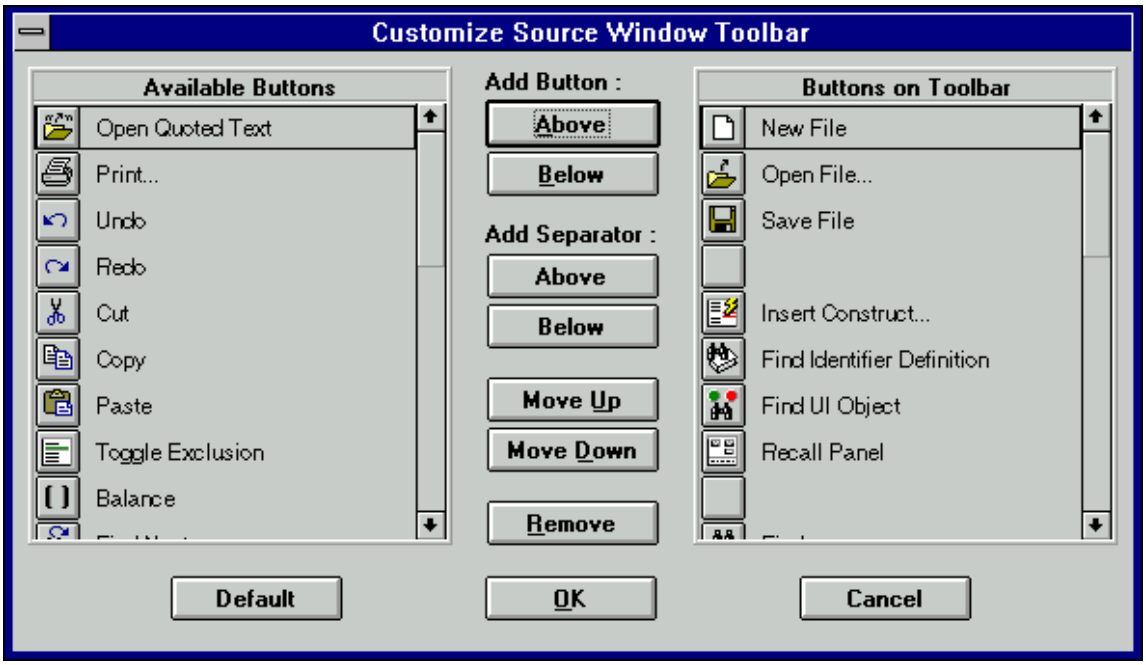

Figure 4-1. The Customize Toolbar Dialog Box for the Source Window

The list box on the left of the Customize Toolbar Dialog Box contains names and icons of toolbar buttons that do not currently appear in the toolbar. The list box on the right contains the names and icons of toolbar buttons that currently appear in the toolbar.

Positioning Buttons and Separators on the Toolbar

The Customize Toolbar dialog box gives you several ways to configure your toolbar.

Adding and Positioning Buttons

Use the Add Button controls to add and position new buttons on the toolbar. First, select the button you want to add to the toolbar from the list box on the left. In the list box on the right, select the button you want to place the new button next to**.** The **Above** button positions the item you are adding above the button that is selected in the list box on the right. When you click on **OK**, the new item will appear to the left of the other button in the toolbar. The **Below** button positions the item you are adding below the other button that you selected in the list on the right. When you click on **OK**, the new item will appear to the right of the other button in the toolbar.

Adding and Positioning Separators

Use the Add Separator controls to add and position separators on the toolbar. Select a button in the list box on the right. The **Above** button adds a separator above the button that is selected in the list box on the right. When you click on **OK**, a small gap (the separator) appears in the

toolbar, to the left of the selected button. The **Below** button adds a separator below the button that you selected in the list box on the right. When you click on **OK**, a small gap (the separator) appears in the toolbar, to the right of the selected button.

Other Positioning Controls

You can select any item in the list box on the right and use the **Remove** control to remove it from the toolbar. If you remove a button, it moves to the list box on the left. If you remove a separator, it disappears from the list. Your modifications take effect on the toolbar when you click on **OK**.

Click on **Default** to restore the default toolbar configuration for LabWindows/CVI.

You can position any item displayed on the toolbar by selecting it in the list box and clicking on **Move Up** or **Move Down**. Your modifications take effect on the toolbar when you click on **OK**.

Notification of External Modification (Windows Only)

If a file in a source window has been modified externally because it was last loaded or saved by LabWindows/CVI, then when you switch back to LabWindows/CVI from another Windows application, a dialog box appears. You are given the option of updating the source window from the file on disk, overwriting the file on disk with the contents of the source window, or doing nothing.

Context Menus

You can bring up a context menu in the Source window by pressing the right mouse button. The context menu contains a selection from the most commonly used menu commands from the Source window menu bar. The selection of commands is different depending on whether the mouse is over the text editing area or over the line number or line icon area.

Interactive Execution Window

You can execute selected portions of code in the Interactive Execution window (IEW). Unlike the Source window, you do not need to have a complete program in the IEW. For instance, you can execute C variable declarations and assignment statements without declaring a main function.

Use the IEW to test portions of code before including them in your main program. You can also use the IEW to execute selected functions that have been defined in a loaded instrument or in a file in the project, if the project has been linked. The IEW can access functions and data declared

as global in a Source window, but a Source window has no access to the functions and data declared in the IEW.

When you execute a function from a function panel, LabWindows/CVI inserts the function call into the IEW, where it executes. In this way, the IEW keeps a record of the functions you have executed from function panels.

When LabWindows/CVI copies a function call from a function panel to the IEW for execution, it inserts the code after all of the pre-existing lines. LabWindows/CVI also copies the header file associated with the function to the IEW if it is not already included. When you execute a function call from a function panel, all previous lines in the IEW are automatically excluded. An *excluded* line is displayed in a different color, and is ignored by the LabWindows/CVI compiler. See the *Toggle Exclusion* section of this chapter for more information about excluded lines. Executing code from the IEW automatically excludes all declarations after they are executed. This is why you should avoid placing executable statements on the same line as declarations in the IEW. Auto-exclusion also occurs when you type a line of code beneath a line that has just been executed. You can manually exclude and include lines with the **Edit » Toggle Exclusion** command.

LabWindows/CVI does not remove a declaration after it has been executed in the IEW, even when the line declaring it is auto-excluded. Declarations in the IEW remain there until you execute the **Clear Interactive Declarations** command from the **Build** menu or the **Clear Window** command from the **Edit** menu.

Rules for executing code in the IEW:

- When executing code from the IEW, data declarations must precede any program statements. Function declarations are also required unless you have disabled **Require Function Prototypes** with the **Compiler Preferences** option in the **Options** menu.

- Function definitions are not allowed in the IEW. The following statements are treated the same:

  `extern int fn (void);` and `int fn (void);`

  The following statements are treated as errors:

  ```
  static int fn (void);
  static int fn () {}
  ```

- LabWindows/CVI treats all global data declarations in the IEW as if they have been declared as static, unless they are preceded by the `extern` keyword. If preceded by the `extern` keyword, the global declaration must exist in a loaded instrument or in a file in the project.

  The following data declaration is invalid in the IEW:

  ```
  extern int x=6;
  ```

Standard Input/Output Window

While a program is executing, it often prints messages and data to the screen. LabWindows/CVI can display text-based messages and data in the Standard Input/Output window. LabWindows/CVI can also accept user input through the Standard Input/Output window. This window can contain up to one million lines and 254 characters per line of screen input and output. You can, however, place a ceiling on the number of lines using the **Environment** command in the **Options** menu. You can save the contents of the window in a file.

You can clear the Standard Input/Output window using the **Clear Window** command in the **Edit** menu or the `Cls` function in the Utility library.

Using Subwindows

The Source, Interactive Execution, and Standard Input/Output windows support subwindows so that you can have two scrollable editing areas for the same file. To create a subwindow from any of these windows, drag the thin line beneath the menu bar to a lower position in the window using the mouse. You can then switch between the subwindows by pressing <F6> or by clicking in a subwindow with the mouse.

Selecting Text in the Source and Interactive Execution Windows

Certain LabWindows/CVI commands require that you select the block of text to which the next command applies. In LabWindows/CVI, you can select a range of characters, a range of lines, or a range of columns. The selected text is highlighted in reverse video.

To select text with the keyboard, hold down the <Shift> key as you move the cursor over the text you want to select. You can use the <Shift> key in combination with any of the keyboard commands for moving the cursor or scrolling the screen.

To select text with the mouse, click on the first character you want to select and drag the mouse over the remaining characters. To select a word, double-click on the word. To select a line, triple-click on the line. If you make a mistake while selecting text, click the mouse or press <Esc> to cancel the selection.

There are three modes for selecting text depending on the state of the graphical icon at the bottom of the window, as illustrated in Figures 4-2 through 4-4.

Character Select mode highlights characters from the position that you begin selecting text through the position that you stop selecting text. Between these endpoints, full lines are highlighted.

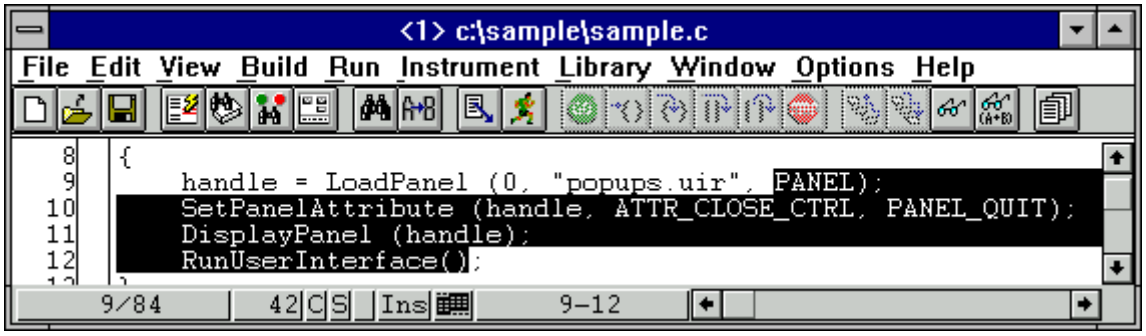

Figure 4-2. Selecting Text Using Character Select Mode

Line Select Mode highlights full lines of text from the beginning of the line where you begin selecting text through the end of the line where you stop selecting text.

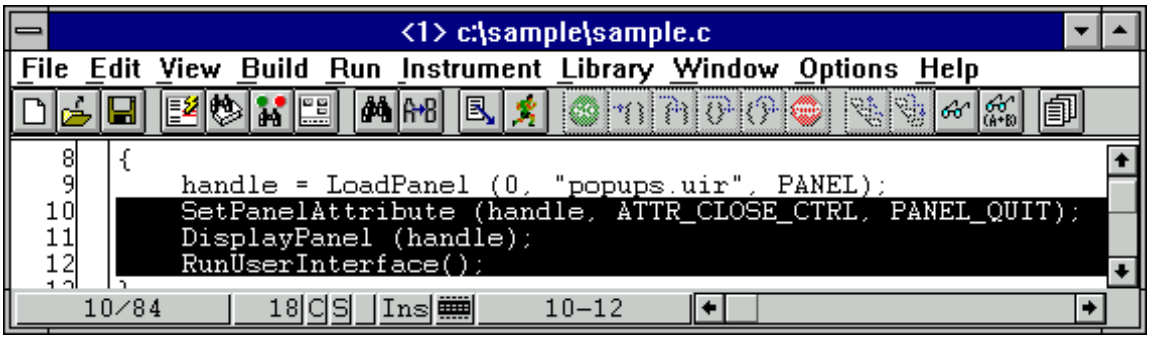

Figure 4-3. Selecting Text Using Line Select Mode

Column Select Mode highlights a rectangular block of text from the position that you begin selecting text through the position that you stop selecting text.

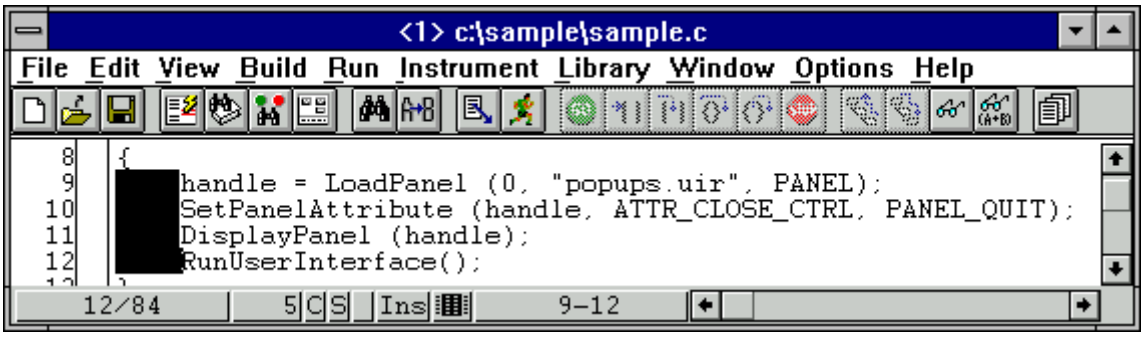

Figure 4-4. Selecting Text Using Column Select Mode

You can cycle through these three modes by pressing <Ctrl-Insert> on the keyboard or by clicking the mouse on the graphical icon at the bottom of the window.

File Menu

This section contains a detailed description of the **File** menu for Source, Interactive Execution, and Standard Input/Output windows, as shown in Figure 4-5.

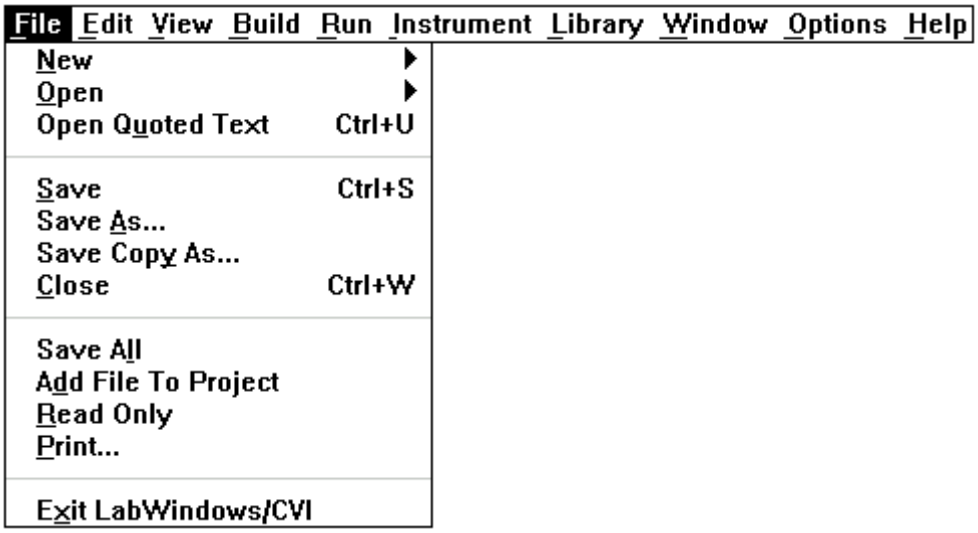

Figure 4-5. The File Menu

New

The **New** command operates the same as in the Project window. For a description of this command, see the discussion of the **New** command in *The File Menu* section of *Chapter 3, Project Window*.

Open

The **Open** command operates the same as in the Project window. For a description of this command, see the discussion of the **Open** command in *The File Menu* section of *Chapter 3, Project Window*.

Open Quoted Text

The **Open Quoted Text** command is used to open `.c`, `.h`, `.fp`, and `.uir` files that appear by name in the active window. If you select **Open Quoted Text** when the text cursor in the active window is on a line containing a filename within quotes or angle brackets, that file will be opened in the proper type of window.

Save

The **Save** command writes the contents of the active window to disk. If you want a different extension to be appended, type it in after the filename. If you do not want any extension to be appended, enter a period after the filename.

Save As...

The **Save As** command writes the contents of the active window to disk using a new name you specify and changes the name of the active window to the user specified name.

Save Copy As...

The **Save Copy As** command writes the contents of the active window to disk using a name you specify *without* changing the name of the active window.

Close

The **Close** command closes the active window. If the contents of the window have been modified since the last save, you will be prompted to save the file to disk.

Note: ***The* Close *command is replaced with the* Hide *command in the Interactive Execution and Standard Input/Output windows. The* Hide *command visually closes the Interactive Execution and Standard Input/Output windows, but retains their contents in memory.***

Save All

The **Save All** command saves all open files under LabWindows/CVI to disk.

Add File to Project

The **Add File to Project** command adds the file in the current window to the project list.

Read Only

The **Read Only** command suppresses the text editing capabilities in the current window. When a file is initially opened, the **Read Only** command is disabled unless the file is read-only on disk.

Print...

The **Print** command prints the window contents to a printer or a file.

Exit LabWindows/CVI

The **Exit LabWindows/CVI** command closes the current LabWindows/CVI session. If any open files have been modified since the last save, or if any windows contain unnamed files, LabWindows/CVI prompts you to save them to disk.

Edit Menu

Use the commands in the **Edit** menu to edit text in Source windows and the IEW. You can copy and save text from the Standard Input/Output window, but you cannot edit it.

Note: ***The procedures for moving the cursor, scrolling, and selecting text are described in*** *Selecting Text in Source and Interactive Execution Windows* ***section earlier in this chapter.***

Figure 4-6 shows the Edit menu.

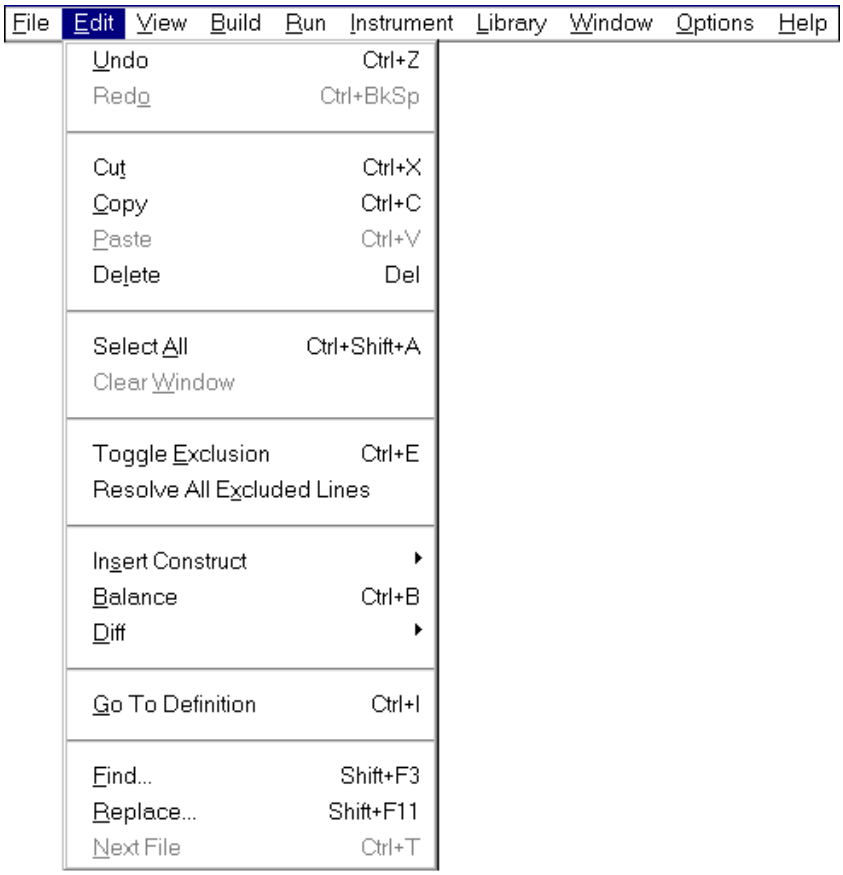

Figure 4-6. The Edit Menu

Note: **Undo** ***and*** **Redo** ***are disabled until you make an edit. The*** **Cut** ***and*** **Copy** ***commands are disabled until you select text, and*** **Paste** ***is disabled until text is placed in the Clipboard. If you select an edit command while it is disabled, nothing happens.***

Undo and Redo

The **Undo** command reverses your last edit action. LabWindows/CVI stores editing actions in a stack so that sequential **Undo** commands reverse a history of your edit actions. You can set the size of this stack using the **Options » Editor Preferences** command. The maximum capacity of this stack is 1,000 operations.

The **Redo** command reverses your last **Undo** command. You can use **Redo** if you go too far in using the **Undo** command to reverse your edit actions. The **Redo** command is enabled only when the previous action was the **Undo** command. Any action, even moving the cursor, disables the **Redo** command.

Cut and Copy

To copy or cut text to the Windows Clipboard, select the text you want to place in the Clipboard and then select **Cut** or **Copy** from the **Edit** menu. LabWindows/CVI places the selected text in the Clipboard. If you used the **Copy** command, the text remains in the window. If you used the **Cut** command, the text is deleted from the window.

Note: ***The*** **Cut** ***command is disabled in the Standard Input/Output window.***

Paste

The **Paste** command inserts text from the Clipboard. If you **Paste** in character-select mode, the characters appear at the cursor on the current line. If you **Paste** in line-select mode, the new lines appear above the current line. If you **Paste** in column-select mode, the new block of characters is inserted at the cursor on the current line. If text is selected when you execute the Paste command, the selected text is replaced by the contents of the Clipboard.

You can **Paste** the same information from the Clipboard as many times as you like. Text remains in the Clipboard until you use **Cut** or **Copy** again, or until another application overwrites the Clipboard. The **New** and **Open** commands do not erase the Clipboard.

To insert text from the Clipboard, move the cursor to the place you want the text inserted and select **Paste** from the **Edit** menu.

Note: ***The*** **Paste** ***command is disabled in the Standard Input/Output window.***

Delete

The **Delete** command deletes highlighted text without placing the text in the Clipboard.

Select All

The **Select All** command selects all of the text in the source window, and positions the keyboard cursor at the end of the file.

Clear Window

Use the **Clear Window** command in the Interactive Execution and Standard Input/Output windows to clear the contents of these windows. The **Clear Window** command also clears any variables declared in the IEW.

Note: ***The* Clear Window *command is disabled in Source windows.***

Toggle Exclusion

You can specify portions of code, called *excluded code*, to be ignored during compilation and execution. LabWindows/CVI displays excluded code in a different color than included code.

The **Toggle Exclusion** command marks lines in Source windows and the IEW as excluded or included code. This command acts both on single and multiple line selections.

You can exclude lines automatically when working in the IEW. See *The Interactive Execution Window* section for more information on automatically excluding lines. Use the **Toggle Exclusion** command if you want to include these lines.

Resolve All Excluded Lines

The **Resolve All Excluded Lines** command interactively highlights the next excluded line or set of consecutive excluded lines and allows you to unexclude, comment out, conditionally compile out, delete, or skip the code.

Insert Construct

The **Insert Construct** command has a submenu of various C programming constructs. Use the command to insert a construct into your source window at the current keyboard cursor position.

For most of these menu items, a dialog box appears asking you to fill in portions of the construct. You may press <Enter> or click on **OK** without filling in the controls.

When you insert the construct into your program, the keyboard cursor moves to the first location in the construct where you need to enter text.

You can set the location of the curly brackets in the construct using the **Options » Bracket Styles** command.

Balance

Use the **Balance** command to find pairs of opening and closing curly braces, brackets, and parentheses. If the cursor is within (or near) a set of any of these symbols when the **Balance** command is selected, all characters between them are highlighted. This command is useful when a large number of these symbols are nested inside each other and you need to find a missing opening or closing symbol.

Diff

Use the **Diff** command for comparing two source files to detect any differences. The **Diff** command has a submenu as shown in Figure 4-7.

Figure 4-7. The Diff Submenu

Use **Diff With** from a Source window to select any other open source file and compare it against the current source file.

Once the two files to be compared are established, use **Synchronize at Top** to display both files starting at the top.

Select **Find Next Difference** to display the next point in the files where there is a difference.

Highlight a section from one of the files and select **Synchronize Selections** from that window to find a matching section in the other file.

Use **Match Criteria** to establish the number of lines that must match to mark the end of differing sections in a file.

Select **Ignore White Space** to compare files while ignoring spaces, tabs, or other text control characters that are not code specific.

Use **Recompare Selections Ignoring White Space** once a difference has been found to determine if the only difference in the selections involves white space characters.

Go To Definition

When you place the text cursor on a C identifier and select **Go To Definition**, LabWindows/CVI highlights the definition of the identifier. If the definition is not available (for example, a LabWindows/CVI library function definition), LabWindows/CVI highlights the declaration of the identifier. This command does *not* work with defined constants.

Find...

Use the **Find** command to locate particular text in your program. When you select the **Find** command, the **Find** dialog box opens, as shown in Figure 4-8.

Figure 4-8. The Find Dialog Box

Enter the text you want to find in the **Find What** text box. If you have text selected on a single line when you execute the **Find** command, the selected text appears in the **Find What** box. Otherwise, the text you last searched for appears in the text box. You can access a history of selections for the **Find What** box by clicking the mouse on the arrow to the right of the **Find What** box, or using the up or down arrow keys.

The **Case Sensitive** option finds only the instances of the specified text that match exactly. For example, if `CHR` is the specified text, the **Case Sensitive** option finds `CHR` but not `Chr`.

The **Whole Word** option finds the specified text only when it is surrounded by spaces, punctuation marks, or other characters not considered parts of a word. LabWindows/CVI treats the characters `A` through `Z`, `a` through `z`, `0` through `9`, and underscore (_) as parts of a word.

If you select **Regular Expression**, LabWindows/CVI treats certain characters in the **Find What** box as regular expression characters instead of a literal characters. The regular expression characters are described in Table 4-1.

Table 4-1. Regular Expression Characters

Purpose	Character	Description	Example
Wildcard Matching	`.` (period)	match 1 character	`a.t` matches `act` and `apt`, but not `abort`
Matching Zero or More Occurrences	`*` (asterisk)	match 0 or more occurrences of preceding character or expression	`0*1` matches `1`, `01`, `001`, etc. `a.*` matches `act`, `apt`, and `abort`
	`+` (plus sign)	match 1 or more occurrences of preceding character or expression	`0+1` matches `01`, `001`, `0001`, ...
Matching Either/Or	`?` (question mark)	match 0 or 1 occurrences of preceding character or expression	`0?1` matches `1`, `01`, but not `001`

	`|` (pipe)	matches either the preceding or following character or expression	`a|b` matches every occurrence of `a` or `b`
			`abor|ut` matches every occurrence of `abort` or `about`
			`{if}|{else}` matches every occurrence of `if` or `else`

Matching the Beginning or Ending of a Line	`^` (caret)	matches the beginning of a line	`^int` matches any line that begins with `int`
	`$` (dollar sign)	matches the end of a line	`end$` matches any line that ends with `end`
Grouping Expressions	`{}` (curly braces)	groups characters or expressions for searches	`{if}|{else}` matches every occurrence of `if` or `else`

(continues)

Table 4-1. Regular Expression Characters (Continued)

Purpose	Character	Description	Example
Matching a Set	`[]` (brackets)	matches any one character or range listed within the brackets	`[a-z]` matches every occurrence of lowercase letters `[abc]` matches every occurrence of `a`, `b`, or `c`
	`~` (tilde)	when appearing immediately after the left bracket, negates the contents of the set	`[~a-z]` matches everything except lowercase letters `[a-z~A-Z]` matches all letters and the '~' character
Special Characters	`\t` (backslash t)	matches any tab character	`\t3` matches every occurrence of a tab character followed by a `3`
	`\x` (backslash x)	matches any character specified in hex	`\x2a` matches every occurrence of the `'*'` character
	`\` (backslash)	used if any of the above characters themselves are to be included in the search	`\-\?` matches every occurrence of `'-'` followed by '?'

Use the **Multiple Files** option to include open source files and source files from the project in the search.

Use **Selected Text Only** to search only within the region of highlighted text when the highlighted text extends beyond one line. This option is automatically enabled when you bring up the Find dialog box and multiple lines of text are selected in the Source window.

Use **Wrap** to continue searching from the beginning of the file once the end of the file has been reached.

Use the **Button Bar** option to enable or disable the built-in dialog box for interactive searching as shown in Figure 4-9. **Find Prev** and **Find Next** operate the same as in the main Find dialog box. **Stop** terminates the search leaving the keyboard cursor at the current position. **Return** terminates the search leaving the keyboard cursor at the original positions where you began the search. Use the **Keyboard Help** command in the **Options** menu for a list of the search hot-keys.

Figure 4-9. The Find Button Bar

Find Prev searches for the closest previous occurrence of the specified text.

Find Next searches for the next occurrence of the specified text.

The **Cancel** command button cancels the **Find** command.
You may bypass the **Find** dialog box using the keyboard commands shown in Table 4-2.

Table 4-2. Keyboard Commands for Implementing Find

Function	Key Combination
Find again (up)	Ctrl+F3
Find again (down)	F3
Use selected text as search string	Ctrl+Shift+F3

Replace...

The **Replace** command operates the same as the **Find** command except that you can replace the search string with another string. As the search is performed, a button bar appears as shown in Figure 4-10.

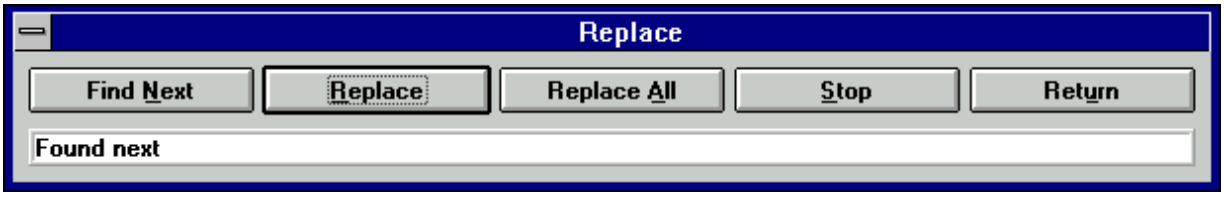

Figure 4-10. The Replace Button Bar

Find Next skips to the next occurrence of the search string without making a change.

Replace executes the replacement.

Replace All finds and replaces all occurrences of the specified text without asking for confirmation.

Stop terminates the search, leaving the keyboard cursor at the current position.

Return terminates the search leaving the keyboard cursor at the position where you initiated the search.

You may bypass the Replace dialog box using the keyboard commands shown in Table 4-3.

Table 4-3. Keyboard Commands for Implementing Replace

Function	Key Combination
Find again (up)	Ctrl+F3
Find again (down)	F3
Use selected text as search string	Ctrl+Shift+F3
Replace selected text	Ctrl+F11
Replace selected text and find again	F11
Use selected text as replace string	Ctrl+Shift+F11

Next File

If you have selected **Multiple Files** from either the **Find** or **Replace** dialog box, you can move to the next file in the search list using this command.

View Menu

Use commands in the **View** menu for displaying line numbers and tags on source code, stepping through build errors, and manipulating function panels that pertain to your editing session.

Figure 4-11 shows the **View** menu.

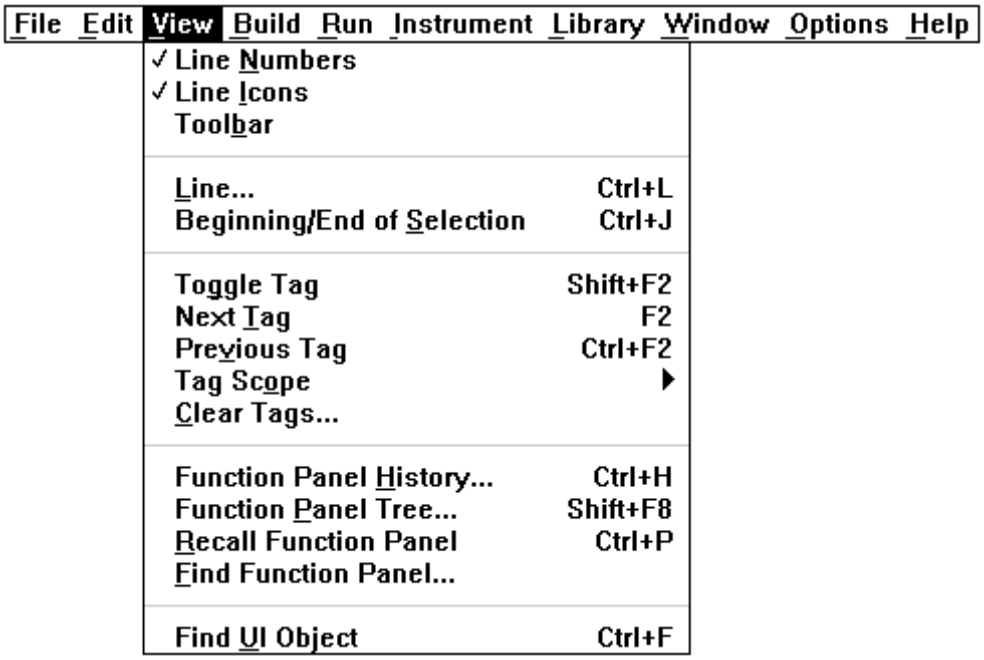

Figure 4-11. The View Menu

Line Numbers

The **Line Numbers** command controls the presence of line numbers in a window. A checkmark appears next to the **Line Numbers** item in the **View** menu when the line number display is activated.

Line Icons

The **Line Icons** command controls the presence of line icons in a window. You can use line icons to give a visual indication for lines that are marked for breakpoint, and lines that are tagged. Breakpoints are discussed in *The Run Menu* section, and tagged lines are discussed later in this section. A checkmark appears next to the **Line Icons** item in the **View** menu when line icons are displayed.

Note: ***Line icons are saved in the project file when the window contents are saved. Editing source files outside of LabWindows/CVI, however, may invalidate the associated line icons.***

Toolbar

Use the **Toolbar** command to toggle the visibility of the Source window toolbar.

Line...

The **Line** command moves the cursor to the line that you specify. When you select the **Line** command, a dialog box appears in which you enter the number of the line where you want to position the cursor.

If you specify a line number greater than the total number of lines in the program, the cursor moves to the last line of the program.

Beginning/End of Selection

The **Beginning/End of Selection** command toggles the window between the beginning and the end of a highlighted block of text. This is useful when you want to verify a selected block of text that is larger than the Source window.

Toggle Tag

The **Toggle Tag** command toggles the tag associated with the active line. Tags are used to mark lines of code that need to be revisited quickly. See **Next Tag** and **Previous Tag** commands for more information.

Next Tag

Use the **Next Tag** command to go to the next tagged line. Selecting **Next Tag** repeatedly takes you to all tagged lines in the windows specified by the **Tag Scope** command.

Previous Tag

The **Previous Tag** command is used to go to the previous tagged line. Selecting **Previous Tag** repeatedly takes you to all tagged lines in the windows specified by the **Tag Scope** command.

Tag Scope

Use the **Tag Scope** command to set which files are searched with **Next Tag** and **Previous Tag**. You can set the scope to the current window, all open windows, or all files.

Clear Tags...

Use the **Clear Tags** command to selectively remove existing tags.

Function Panel History...

The **Panel History** command displays a scrollable list of the function panels you have used during the current LabWindows/CVI session. You can display function panels from the list as new windows or you can overwrite the current Function Panel window.

Function Panel Tree...

The **Current Tree** command displays the Select Function Panel dialog box for the most recently used function panel, making it easy for you to return to the location of the current panel in the function tree.

Recall Function Panel

When you are editing a function call in a Source window or the IEW, you may want to display the function panel corresponding to the call. You can do this with the **Recall Function Panel** command. The **Recall Function Panel** command not only finds and displays the panel, but also sets the panel controls so that they contain the parameter values appearing in the function call. After modifying one or more controls, you can then replace the original call with the modified call.

Invoking the Recall Function Panel Command

Before you invoke the **Recall Function Panel** command, you must indicate the function panel you want to recall. The simplest method is to place the cursor on a line that contains a function call or a portion of a function call. You can also select, or highlight, a range of lines that contains one or more function calls. You can also select part of a line, provided that part contains a function call.

If a line contains multiple function calls or one function call embedded within another, you can resolve the ambiguity by placing the cursor on or immediately after the function name.

When you have indicated the function call, select the **Recall Function Panel** command from the **View** menu. The function panel for that function appears, and the controls contain the parameter values from the call.

Recalling a Function Panel from a Function Name Only

You can recall a panel from a function name without specifying any of the parameters. If you place the keyboard cursor on or immediately after a function name, **Recall Function Panel** recognizes the function name even if it is not followed by a parameter list. Thus, you can simply type a function name into the Source window and execute **Recall Function Panel**.

You can also use the **Find Function Panel** command to bring up a function panel from a function name or a portion of a function name. See the **Find Function Panel** section, which follows this section.

Multiple Panels for One Function

If the selected function appears in more than one function panel window, LabWindows/CVI displays a list of panels. Select one by highlighting the panel name and pressing <Enter>, or by double-clicking on the panel name.

Multiple Functions in One Function Panel Window

If the selected function matches a function panel window containing multiple function panels, LabWindows/CVI attempts to match the panel to function calls on the lines surrounding the selected call. After the panel appears, you can check how many lines were matched to the function panel window by looking at the Source window. The matched lines are highlighted.

If you selected multiple lines before executing the **Recall Function Panel** command, all function calls in the selected lines must appear in one Function Panel window, and the order in which the window generates the calls must be identical to the order in which they appear in the selected lines. Otherwise, an error message appears.

Syntax Requirements for the Recall Function Panel Command

You do not have to compile the window you are working in before you invoke the **Recall Function Panel** command. In fact, the function call you select need not even be syntactically valid. The only requirement is that the name of the function must be spelled correctly. If the function name is not spelled correctly, LabWindows/CVI displays an error message indicating that the panel could not be found.

Find Function Panel...

When you select the **Find Function Panel** command, a dialog box appears in which you can enter the name of a function. You can enter just a substring, and Find Function Panel finds all functions that contain that substring anywhere in their names. For instance, if you enter

```
ctrl
```

and click on **OK**, a dialog box appears with a list of functions including `NewCtrl`, `SetCtrlVal`, `GetCtrlVal`, and so on.

You can use a regular expression as your search string. See Table 4-1, *Regular Expression Characters*, for a list of regular expression characters.

If there is a function panel for the function, LabWindows/CVI displays the panel. If there are two or more Function Panel windows for the function, LabWindows/CVI displays a list of the Function Panels.

The shortcut key for **Find Function Panel** is <Ctrl-Shift-P>.

Find UI Object

The **Find UI Object** command is used to move directly from a Source window to a User Interface Editor window. To use it, place the cursor on the constant name or callback function name of the User Interface panel, control, or menu object you want to view. Then select the **Find UI Object** command from the **View** menu. LabWindows/CVI searches each `.uir` file that is currently open or in the project for UI objects with a matching constant name or callback function name. If it finds an object, the User Interface window containing the object comes to the foreground.

If the matching object is a panel, the panel's title bar briefly flashes and the panel becomes active. If the object is a control, the control is selected. If **Find UI Object** finds a menu object or more than one matching object, a dialog box containing the list of matches appears. In this dialog box you can view information about each of the objects or select one to edit.

Build Menu

Use the commands in the **Build** menu for compiling files, and building and linking projects. Figure 4-12 shows the **Build** menu.

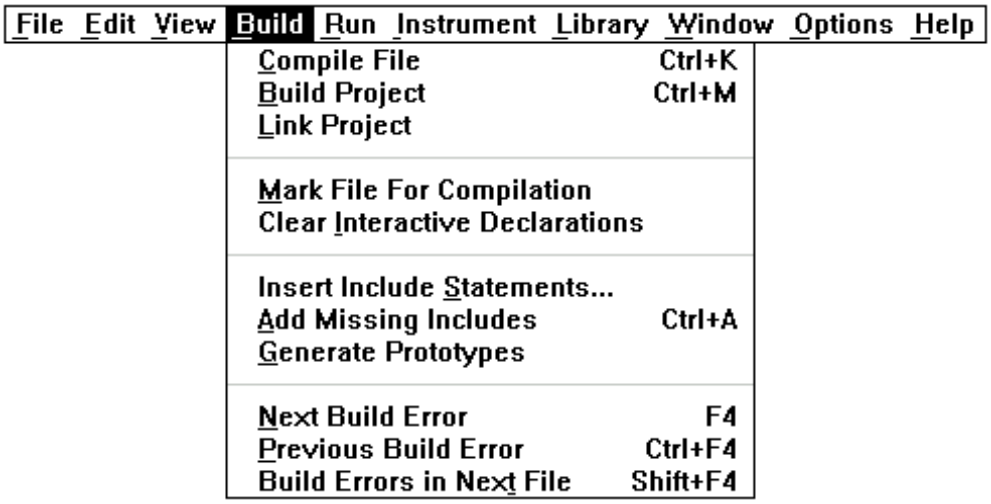

Figure 4-12. The Build Menu

Compile File

You must compile your source code before executing it in a Source window or the IEW. The **Compile File** command adds the file to the project if necessary, checks it for syntax errors, and compiles it. After LabWindows/CVI completes compilation, a Build Errors box appears if the file has any build errors.

When you want to call a function that is defined in a Source window from another Source window, the IEW, or from a function panel, you must first execute the **Compile File** command in the Source window where the function is defined. If you subsequently modify the function, you must recompile the Source window before calling the function again.

Note: ***The* Compile File *command is not available in the Standard Input/Output window.***

See the *Compiler Options* discussion in *The Options Menu* section of *Chapter 3, Project Window*, for a discussion of compiler options.

Build Project

Use the **Build Project** command to compile all source files listed in the project that are marked for compilation and to link the compiled files.

Link Project

Use the **Link Project** command to link the compiled files from the project. **Link Project** does not invoke any compilation.

Mark File for Compilation

When you mark a source file for compilation, a `C` appears next to the filename in the Project window and it is recompiled the next time the project is built. LabWindows/CVI automatically marks source files for compilation when the file has been modified since it was last compiled. However, you can manually force a source file to be compiled on the next build by selecting the **Mark File for Compilation** command.

Clear Interactive Declarations

Variables declared in the IEW remain in effect until you explicitly remove them. This lets you use these variables in succeeding executions of the IEW. It also enables different function panels to access the same variables.

When you delete the entire contents of the IEW with the **Clear Window** command in the **Edit** menu, the variables are removed. If you want to remove the variables without deleting the contents of the IEW, use the **Clear Interactive Declarations** command.

Insert Include Statements

Insert Include Statements invokes a dialog box you can use to select one or more header files to be included at the top of the program.

Add Missing Includes

If, when you last attempted to compile the source file, the compiler reported that function prototypes were missing, **Add Missing Includes** can find include (`.h`) files that contain some or all of the missing prototypes. It inserts `#include` statements for these files into your source file at the current cursor position. LabWindows/CVI adds `#include` statements only for libraries or instrument drivers that appear in the **Instrument** or **Library** menu.

Generate Prototypes

After you compile a source file, you can use **Generate Prototypes** to generate a file containing declarations for global and static functions and external declarations for global variables into a new Source window. You can copy these declarations into your source and header files.

Next Build Error

After you have compiled a file or built your project, use the **Next Build Error** command to step to your next build error. LabWindows/CVI highlights source code errors as you step through the errors.

Previous Build Error

After you have compiled a file or built your project, use the **Previous Build Error** command to step to your previous build error. LabWindows/CVI highlights source code errors as you step through the errors.

Build Errors in Next File

After you have built your project, use the **Build Errors in Next File** command to step to your next file with build errors. LabWindows/CVI highlights source code errors as you step through the errors.

Run Menu

Use the commands in the **Run** menu to run and debug your program. Figure 4-13 shows the **Run** menu.

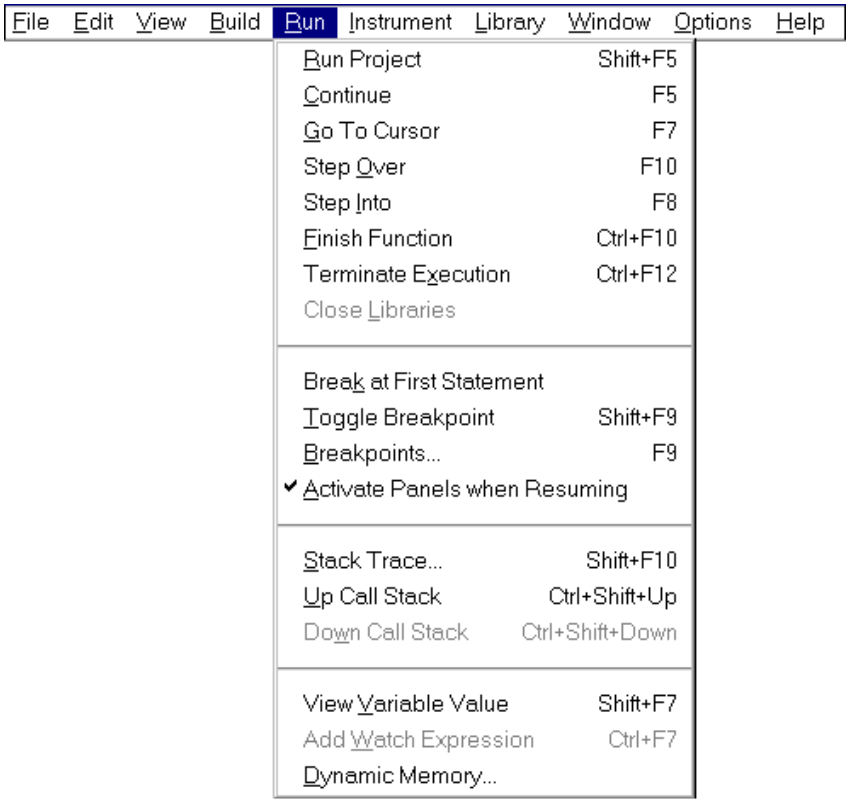

Figure 4-13. The Run Menu

Introduction to Breakpoints and Watch Variables/Expressions

It is important to understand the concepts of breakpoints and watch variables/expressions before learning about the commands in the **Run** menu.

You can pause the execution of a program without aborting it altogether by marking breakpoints in your code. You can use these breakpoints to interrupt program execution for debugging. Breakpoints can be either conditional or unconditional. Breakpoints apply to specific lines of code, but LabWindows/CVI maintains them separately from your source file. If you modify your source code outside of the LabWindows/CVI environment, you may invalidate breakpoint position information. The only exception to this is the `Breakpoint` function, which is an actual line of code stored in your source code file.

You can also use watch variables/expressions for debugging. With watch variables/expressions, you can specify that LabWindows/CVI suspend execution conditionally without regard to a specific line of code.

Note: ***Breakpoints and watch variables/expressions apply only to source code modules. You cannot set breakpoints in include files.***

The Breakpoint State

When a program reaches a breakpoint, LabWindows/CVI positions the keyboard cursor at the program statement to be executed, and outlines the statement. You cannot edit the source code in the window while the breakpoint is in effect. However, you can use many other features of the LabWindows/CVI environment. For instance, you can look at other windows, change the state of breakpoints, and modify the value of variables in the Variable, Array, and String Displays. Also, if you are at a breakpoint in a Source window, you can execute code in the IEW or a function panel.

To resume the execution after a breakpoint, you have several options under the **Run** menu which control the next breakpoint. You can restart the project at a breakpoint by selecting **Run Project**. To halt the execution of a program at a breakpoint, select **Terminate Execution** or press <Ctrl-Alt-SysRq> under Windows 3.1 or <Ctrl-F12> under Windows 95, Windows NT, and UNIX.

Setting and Clearing Breakpoints

There are several ways for you to set and clear breakpoints:

- If you activated **Line Icons** in the **View** menu, you can click the mouse in the line icon area next to a line of code to set or clear a breakpoint on that line.
- Move the cursor to the line of code where you want to set or clear a breakpoint and select **Toggle Breakpoint** in the **Run** menu (<Shift-F9>).
- Select **Breakpoints** from the **Run** menu to edit all breakpoints in the project and the IEW. You can also use the **Breakpoints** command to set *conditional* breakpoints. See the *Conditional Breakpoints* section for information about conditional breakpoints.
- Select **Breakpoint at First Statement** from the **Run** menu to breakpoint on the first executable statement in the project or the IEW.
- You can set code breakpoints in your source code using the `Breakpoint` function.
- You can enter a breakpoint state manually if the program checks for user input. For example, if the program makes calls to `RunUserInterface` or `scanf`, pressing <Ctrl-Alt-SysRq> under Windows 3.1 or <Ctrl-F12> under Windows 95, Windows NT, and UNIX causes a breakpoint state.

Conditional Breakpoints

Set conditional breakpoints with the **Breakpoints** command in the **Run** menu. When you assign a conditional breakpoint to a line in your program, LabWindows/CVI evaluates a user supplied expression, such as `x==100` or `y<0`, before executing the line. If the expression is true, program execution is suspended.

If you assigned these expressions to line 23 in your program, `x` and `y` would have to be defined before line 23.

Watch Variables/Expressions

You can also use watch variables/expressions to suspend program execution conditionally. Watch variables/expressions do not apply to specific lines of code, however. LabWindows/CVI evaluates them between every statement in your source code instead. See Chapter 6, *Variable Display and Watch Windows*, for more information about watch variables/expressions.

Run Project / Run Interactive Statement

Running in a Source Window

The **Run Project** command compiles any source files in the project that are marked for compilation, links them together, and runs the project. See the **Mark File for Compilation** command in *The Build Menu* section earlier in this chapter for information on marking files for compilation. If any build errors are encountered, LabWindows/CVI terminates the process and the Build Errors window appears with the list of errors.

Running in the Interactive Execution Window

Select **Run Interactive Statements** in the IEW to execute code in that window. You do not need to have a complete program in the IEW. For instance, you can execute variable declarations and assignment statements in C without declaring a main function.

You can use the IEW to test portions of code before including them in your main program. You can also use the IEW to execute selected functions you have defined in a loaded instrument or in a file in the project if the project has been linked. The IEW can access functions and data declared as global in a Source window, but a Source window cannot access the functions and data declared in the IEW.

See *The Interactive Execution Window* section for the rules governing code execution in the IEW.

LabWindows/CVI does not disturb asynchronous I/O, RS-232 ports, opened files, and User Interface Library resources used in the IEW at the beginning or end of execution in the IEW.

LabWindows/CVI terminates, closes, or deletes these program elements only when one of the following events occurs:

- You select **Clear Interactive Declarations** from the **Build** menu.
- You link a project.
- You run a project.

Run-time Error Reporting

LabWindows/CVI reports various run-time errors during the execution of a program. For instance, a call to a LabWindows/CVI library function with an array or string too small to hold the output data is one example of a run-time error.

When such errors occur, a dialog box appears identifying the type of error and the location in the file where the error occurred. LabWindows/CVI then displays the error in the Runtime Error window.

LabWindows/CVI suspends the program so you can inspect the values of variables in the Variable Display window. To terminate a program that has been suspended because of a run-time error, select the **Terminate Execution** command or <Ctrl-Alt-SysRq> under Windows 3.1 or <Ctrl-F12> under Windows 95, Windows NT, and UNIX.

Continue

Use the **Continue** command to resume program execution when in a breakpoint state.

Go to Cursor

When the program is in a breakpoint state, you can move the keyboard cursor to a line in the program and select **Go to Cursor**. Program execution then continues until it reaches that line, where it will enter another breakpoint state.

Step Over

Use **Step Over** to execute an outlined statement while in a breakpoint state. If the program is breakpointed on a function call statement, **Step Over** executes the entire function and then breakpoints on the statement following the function call. If a breakpoint is encountered within the function call, **Step Over** pauses at the breakpoint.

Step Into

The **Step Into** command is similar to the **Step Over** command except that, when the program suspends operation at a function call marked as a breakpoint, **Step Into** enters the function and suspends at the function's first statement. **Step Into** can enter a function only if it is a user defined function in a source file in the project. Otherwise, **Step Into** executes the entire function and suspends execution on the statement following the function call.

Finish Function

The **Finish Function** command resumes execution through the end of the current function and breakpoints on the next statement after the current function.

Terminate Execution

The **Terminate Execution** command terminates a program that is suspended at a breakpoint. The shortcut key for terminating execution of a suspended program or suspending a running program is <Ctrl-F12> under Windows 95, Windows NT, and on UNIX. Under Windows 3.1 the shortcut key is <Ctrl-Alt-SysRq>

Close Libraries

The **Close Libraries** command closes the LabWindows/CVI libraries you have accessed through function panels and the IEW. Use this command if you are working from function panels or the IEW and you want to close the LabWindows/CVI libraries without clearing your interactive variables.

Note: ***LabWindows/CVI automatically closes the libraries before and after you run a project.***

Break at First Statement

Break at First Statement is a run mode that breakpoints on the first executable statement in your source code. When activated, LabWindows/CVI puts a checkmark beside this command in the menu.

Toggle Breakpoint

The **Toggle Breakpoint** command toggles the state of the breakpoint on the current line.

Breakpoints...

The **Breakpoints** command brings up the Breakpoints dialog box containing a list of the breakpoints in the project as illustrated in Figure 4-14. You can also bring up this dialog box by clicking with the right mouse button in the line icons column.

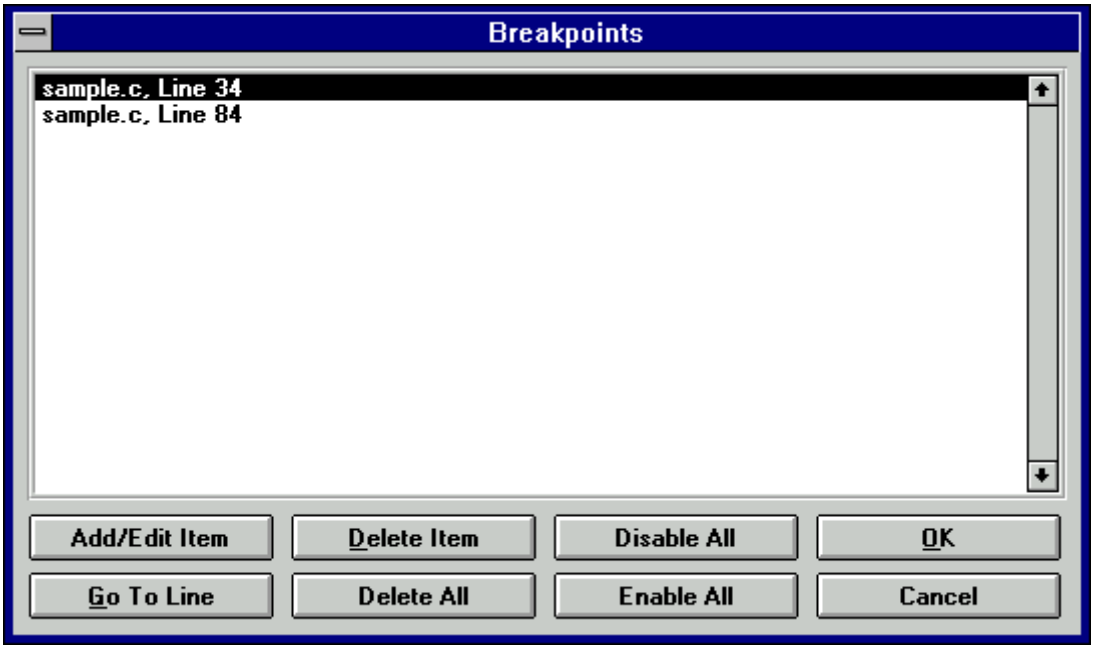

Figure 4-14. The Breakpoints Dialog Box

Use the **Add/Edit Item** command button to edit a single breakpoint with the Edit Breakpoint dialog box, as shown in Figure 4-15.

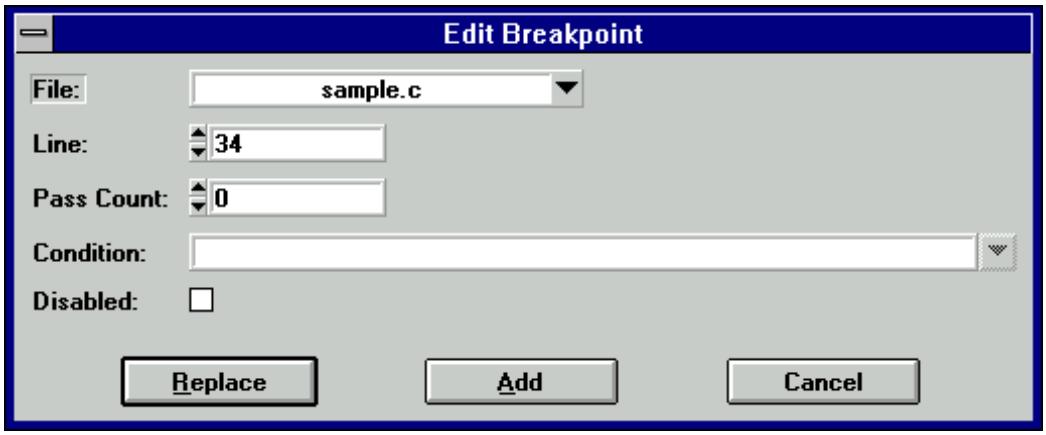

Figure 4-15. The Edit Breakpoint Dialog Box

- **File** is the source file containing the breakpoint you want to edit.

- **Line** is the line containing the breakpoint you want to edit.
- **Pass Count** is the number of times that **Line** will execute before the breakpoint.
- **Condition** is an optional expression that is evaluated before LabWindows/CVI executes **Line**. If the condition is true at that time, LabWindows/CVI institutes a breakpoint; otherwise, execution continues. See the *Conditional Breakpoints* section for examples of conditional expressions.
- The **Disabled** checkbox specifies a breakpoint that LabWindows/CVI will ignore. The attributes of the breakpoint are retained so that you can later enable the same breakpoint. The breakpoint icon in the Source window changes color to indicate that it is disabled.
- After you set all the breakpoint attributes in the dialog box, you can **Replace** the breakpoint on **Line** with the new attributes, or you can **Add** the new attributes to the breakpoint on **Line**, or you can **Cancel** the operation.

The **Go to Line** command button takes you to the source code location of the selected breakpoint.

The **Delete Item** command button deletes the breakpoint you selected.

The **Disable All** command button forces LabWindows/CVI to ignore all the breakpoints. The attributes of the breakpoints are retained so that you may later enable them. The breakpoint icons in the Source window change color to indicate that they are disabled.

The **Enable All** command button activates all the breakpoints. The breakpoint icons in the Source window change color to indicate that they are activated.

The **OK** command button accepts the current breakpoint attributes, and the **Cancel** command button cancels the current operation.

Activate Panels When Resuming

You can use **Activate Panels when Resuming** to choose whether the user interface panels in your programs are reactivated every time you resume execution during debugging. By default, this option is enabled. Activating the panels whenever you resume guarantees that the activation state of every panel is identical to what it would be if you were not debugging. In general, however, this is not important, and activating panels each time you resume can be time consuming.

If you disable this option, your panels are activated when your program causes events to be processed or explicitly displays, activates, hides, or discards panels.

This option is saved from one LabWindows/CVI session to another, not in the project file.

Stack Trace...

You can use the **Stack Trace** command only when in a breakpoint state. **Stack Trace** brings up a dialog box listing the currently active functions in the program, displaying the most recently called function at the top through the originating function at the bottom. If you highlight a function in the list and select **Display**, a Source window appears with the file containing that function. LabWindows/CVI highlights the last statement that was executed in that function.

Up Call Stack

You can use the **Up Call Stack** command only when in a breakpoint state. **Up Call Stack** moves up one level in the function call stack.

Down Call Stack

You can use the **Down Call Stack** command only when in a breakpoint state. **Down Call Stack** moves down one level in the function call stack.

Variable Value

Variable Value is a convenient way to view the contents of arrays, structures, and global variables that appear in source code. Highlight the variable that you want to see and select **Variable Value**. Depending on the type of the variable, the Variable, Array, or String Display appears with your selected variable highlighted.

Expression Value

Expression Value is a convenient way to view the value of an expression that appears in source code. Highlight the expression that you want to see and select **Expression Value**. The Watch window appears with your selected expression highlighted.

Dynamic Memory...

The **Dynamic Memory** command brings up a dialog box displaying the dynamic memory area. This area contains data that has been dynamically allocated by your program using `malloc`, and any variables you have declared in the IEW. Memory may be viewed in hexadecimal (byte, word, long), integer (byte, word, long), single-precision floating-point, double-precision floating-point, or ASCII representation.

Instrument Menu

The **Instrument** menu for Source, Interactive Execution, and Standard Input/Output windows works the same as the **Instrument** menu in the Project window. See *Chapter 3, Project Window*, for information on the **Instrument** menu.

Library Menu

The **Library** menu for Source, Interactive Execution, and Standard Input/Output windows works the same as the **Library** menu in the Project window. See *Chapter 3, Project Window*, for information on the **Library** menu.

Window Menu

The **Window** menu in the Source, Interactive Execution, and Standard Input/Output windows works the same as the **Window** menu in the Project window. See *Chapter 3, Project Window*, for information on the **Window** menu.

Options Menu

Use the commands in the **Options** menu to set up preferences in the LabWindows/CVI environment, and execute various LabWindows/CVI utilities.

Figure 4-16 shows the **Options** menu.

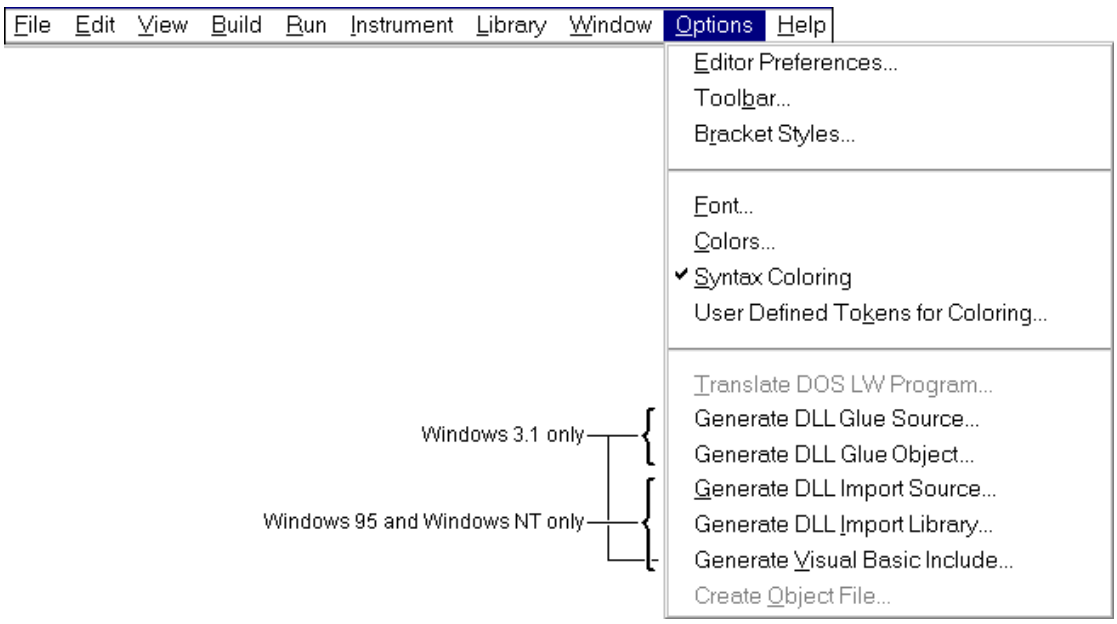

Figure 4-16. The Options Menu

Editor Preferences...

The **Editor Preferences** command invokes the dialog box shown in Figure 4-17 which you can use to set up Source window editor preferences.

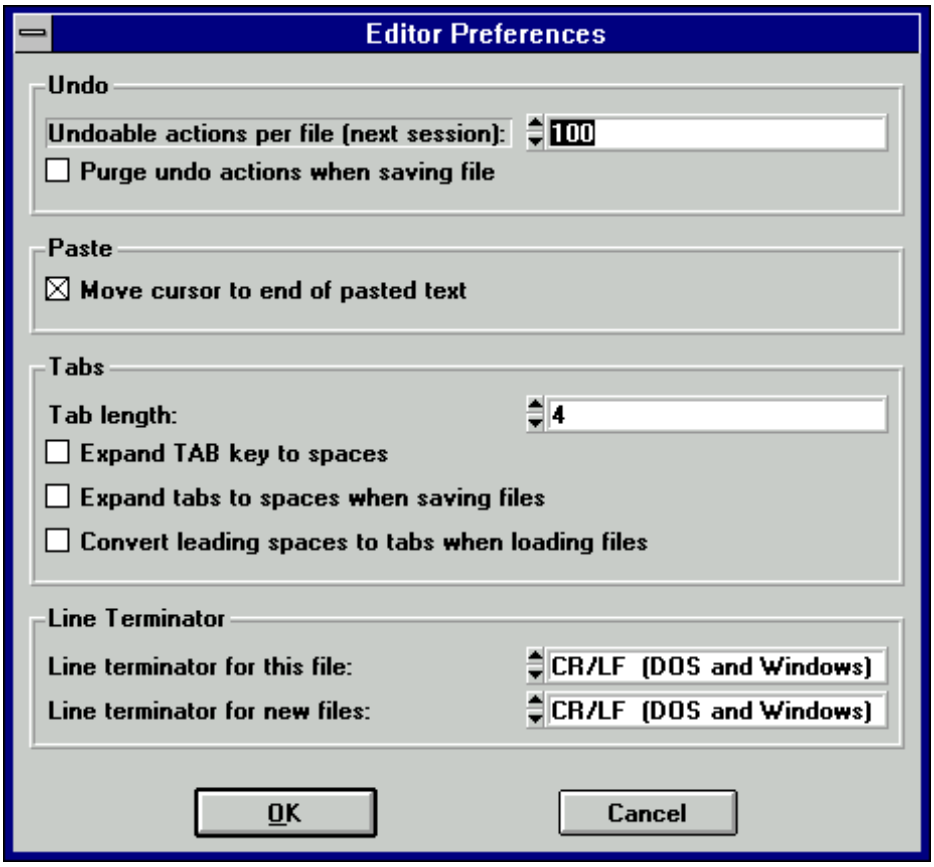

Figure 4-17. Editor Preferences

Undo

Use the **Undo** option of the **Editor Preferences** dialog box to set the number of actions per file that you can undo. Check **Purge undo actions when saving file** to clear the accumulated list of editing actions each time you save a file.

Paste

Use the **Paste** option of the **Editor Preferences** dialog box to set where LabWindows/CVI places the text cursor after completing a **Paste** operation. Check the **Move cursor to the end of pasted text** option to put the cursor at the end of the pasted text. Leave this option unchecked to put the cursor at the beginning of the pasted text.

Tabs

Use the **Tabs** option of the **Editor Preferences** dialog box to set the tab length. Use the checkboxes for LabWindows to convert tab characters into spaces when saving files and convert leading spaces to tab characters when loading files. These options are convenient if you are using another editor or a printer that does not support tab characters.

Line Terminator

LabWindows/CVI can read source files with any of the commonly used line-termination sequences. It remembers what line-termination sequence was found in each file and uses the same sequence when saving each file. If you want to change that sequence because you want to load the file into another editor, use the **Line Terminator** option as follows:

- If you want to load your text file into DOS/Windows editors, select CR/LF termination.
- If you want to load your text file into a UNIX editor, select LF termination.
- If you want to load your text file into a Macintosh editor, select CR termination.

Toolbar...

Use the Toolbar command to select which icons appear in the Source window toolbar.

Bracket Styles...

The **Bracket Styles** command allows you to set the location of curly brackets when the following commands generate them in your program.

- The **Insert Construct** command in the **View** menu of the source window.
- The **Generate** command in the **Code** menu of the User Interface Editor window.

You can specify two bracket styles: one for functions, and another for statements (for example, `if` or `switch` statements).

Font...

Use the **Font** command to select the font and font size for text in Source windows and Variable displays.

Colors...

Refer to *Chapter 3, Project Window, The Options Menu*, for a description of the **Colors** command.

Syntax Coloring

When you enable the **Syntax Coloring** option, LabWindows/CVI color codes the various types of tokens in your source and include files. The following are the different types of tokens that can be color coded.

- C keywords
- identifiers
- comments
- integers
- real numbers
- strings
- preprocessor directives
- user-defined tokens

You can set the color for a token type via the **Color** command in the **Options** menu.

You can create the list of user-defined tokens via the **User Defined Tokens for Coloring** command in the **Options** menu.

User Defined Tokens for Coloring

You can use the **User Defined Tokens for Coloring** command to define tokens that can be displayed in a unique color when the **Syntax Coloring** option is enabled. You use the **Colors** command to set the color. Each token must be in the form of a valid C identifier. You can cause a token to be saved in your project file or saved from one CVI session to another without regard to which project is loaded.

Translate DOS LW Program...

Use the **Translate DOS LW Program** command to convert a source file written in LabWindows for DOS so that it will run in LabWindows/CVI. See Chapter 12, *Converting LabWindows for DOS Applications* in the *Getting Started with LabWindows/CVI* manual for details about converting `.c`, `.uir`, `.lbw`, and `.obj` files from LabWindows for DOS for use in LabWindows/CVI.

Generate DLL Import Source (Windows 95/NT Only)

This command generates source code that can be used to create a DLL import library. In general, you do not need to use this command. For most cases, you can generate a DLL import library directly using the **Generate DLL Import Library** command. Use this command only when you must do special processing in the DLL import library. LabWindows/CVI never requires such special processing.

The **Generate DLL Import Source** command is enabled only when you have an include file in the Source window. The include file should contain declarations of all of the functions you want to access from the DLL. When you execute the command a file dialog box appears. Enter the pathname of the DLL.

The command generates the import library source into a new Source window. You can modify the code, including making calls to functions in other source files. Create a new project containing the source file and any other files it references. Select **Static Library** from the submenu attached to the **Target** command in the **Build** menu of the Project window. Execute the **Create Static Library** command.

Note: ***You cannot export variables from a DLL using the import library source code generated by this command. When you want to export a variable, create functions to get and set its value or create a function to return a pointer to the variable.***

Note: ***When you edit the source code generated by this command, you cannot use the*** `__import` ***qualifier in the function declarations in the DLL include file.***

Note: ***The import source code does not operate in the same way as a normal DLL import library. When you link a normal DLL import library into an executable, the operating system attempts to load the DLL as soon as the program starts. The import source generated by LabWindows/CVI is written so that the DLL is not loaded until the first function call into it is made.***

Generate DLL Import Library (Windows 95/NT Only)

This command generates a DLL import library. The command is enabled only when you have an include file in the Source window. The include file should contain declarations of all of the functions and global variables you want to access from the DLL. When you execute the command, you have the option to generate an import library for each of the compatible external compilers rather than just for the current compatible compiler. A file dialog box then appears. Enter the pathname of the DLL.

The command generates a `.lib` file with the same base name as the include file. If you choose to create an import library for each compiler, the files are created in subdirectories named `MSVC`, `BORLAND`, `WATCOM`, and `SYMANTEC`. The library for the current compatible compiler is also created in the directory of the DLL.

Generate DLL Glue Source... (Windows 3.1 Only)

If you are using a Windows `.dll` file, LabWindows/CVI requires that *glue code* be present in a `.lib` or `.obj` file that accompanies the `.dll` file. The glue code is necessary for LabWindows/CVI, which is a 32-bit application, to access Windows 16-bit `.dll` files.

LabWindows/CVI automatically generates and compiles glue code when it loads the `.dll` file based in the contents of the accompanying `.h` file. Sometimes the automatically generated glue code is sufficient. In other cases you must generate, modify, and compile the glue source code yourself.

See the *16-bit Windows DLLs* section of Chapter 2, *Using Loadable Compiled Modules*, in the *LabWindows/CVI Programmer Reference Manual* for details about generating DLL glue source code.

The Generate DLL Glue Source command is enabled only in windows that contain include files. The include file should contain declarations for all user-callable functions on the DLL. The command generates glue source code into a new window.

Generate DLL Glue Object... (Windows 3.1 Only)

If you do not need to modify the DLL glue code that is automatically generated by LabWindows/CVI, then you can place your `.dll` file directly in the project and LabWindows/CVI generates the glue code when the DLL is loaded. However, you can use the **Generate DLL Glue Object** command to create a compiled version of the DLL glue code. The DLL loads faster when the DLL glue object module is placed in the project instead of the DLL.

The **Generate DLL Glue Object** command generates the glue object module from a window containing an include file. The file should contain declarations for all user-callable functions in the `.dll`.

Generate Visual Basic Include... (Windows Only)

This command generates a Visual Basic include file from the `.h` file of an instrument driver. Use this command if you are porting an instrument driver to a DLL for use in Visual Basic.

Create Object File

You can use the **Create Object File** command to compile the contents of a Source window into an object file. Compiled files consume less memory and run faster than source files. They are especially useful for instrument driver programs because they load faster. Compiled files cannot be debugged, however, and they do not have run-time error checking.

For Windows 95 and NT, the **Create Object File** command has been modified so that it gives you the option of creating an object file for each of the compatible external compilers rather than just for the current compatible compiler. If you chose to create an object file for each compiler, the files are created in subdirectories named `MSVC`, `BORLAND`, `WATCOM`, and `SYMANTEC`. The object file for the current compatible compiler is also created in the parent directory.

You can compile your file using a third-party compiler supported by LabWindows/CVI. See the *LabWindows/CVI Programmer Reference Manual* for more information on compatible external compilers. These compiled files are smaller and execute faster than object files created by LabWindows/CVI. The **Create Object File** command is primarily intended for those who do not have access to such a compiler.

Help Menu

The **Help** menu provides information about LabWindows/CVI. The **Help** menu is shown in Figure 4-18.

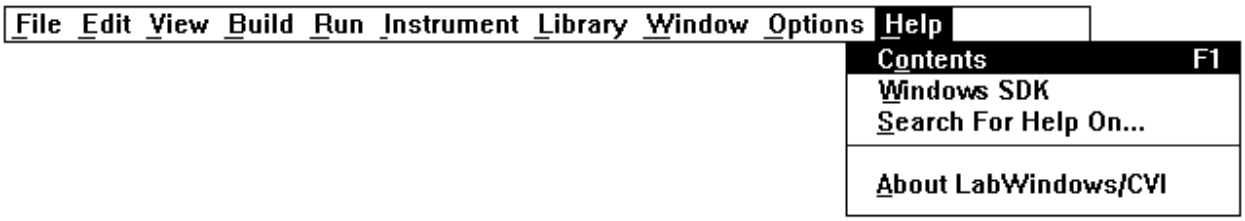

Figure 4-18. The Help Menu

The **Contents** command invokes the online help for LabWindows/CVI.

The **Search for Help On** command allows you to search for keywords used within the online help.

The **About LabWindows/CVI** displays a read-only dialog box with information about your LabWindows/CVI session.

The **Keyboard Help** command invokes a scrollable list of keyboard shortcut keys. These shortcut keys are shown in Figure 4-19.

Find again (up)	Ctrl+F3
Find again (down)	F3
Use selected text as search string	Ctrl+Shift+F3
Replace selected text	Ctrl+F11
Replace selected text and find again	F11
Use selected text as replace string	Ctrl+Shift+F11
Windowing:	
Next window	Ctrl+F6
Previous window	Ctrl+Shift+F6
Switch sub-windows	F6
Editing:	
Change text selection mode	Ctrl+Ins
Toggle insert/overwrite mode	Ins
Delete to end of line	Ctrl+D
Backspace to beginning of word	Ctrl+Shift+BkSp
Cut line to clipboard	Ctrl+Y
Insert a new line above	Shift+Enter
Insert a new line below	Ctrl+Enter
Select text	Shift+<cursor movement key>
Remove text selection	Esc
Cursor Movement:	
Up 1 line	Up
Down 1 line	Down
Left 1 column	Left
Right 1 column	Right
Scroll up 1 line	Ctrl+Up
Scroll down 1 line	Ctrl+Down
Left 1 word	Ctrl+Left
Right 1 word	Ctrl+Right
Top of window	Ctrl+PgUp
Bottom of window	Ctrl+PgDown
Beginning of line	Home
End of line	End
Move up 1 page	PgUp
Move down 1 page	PgDown
Top of file	Ctrl+Home
Bottom of file	Ctrl+End

Figure 4-19. Keyboard Help

Chapter 5
Using Function Panels

This chapter describes how to use LabWindows/CVI function panels to generate code for calling functions in any of the LabWindows/CVI libraries.

A function panel is an interface to the functions in the LabWindows/CVI libraries and instrument drivers. You can use function panels to help generate and test function calls within LabWindows/CVI.

A Function Panel window generates one or more function calls, with the function parameters determined by the state of the function panel controls. LabWindows/CVI can execute these functions immediately in the Interactive Execution window (IEW). When you execute a function panel, LabWindows/CVI copies the generated code to the IEW and executes it. The first time you execute a function panel for an instrument driver or library, LabWindows/CVI creates and executes an `#include` statement for the header file associated with the instrument driver or library. Other sections of this chapter discuss the relationship between a function panel and the IEW in more detail.

Normally, you use function panels to call into instrument drivers that appear in the **Instrument** menu and libraries that appear in the **Library** menu. (See the *Using Instrument Drivers* section in Chapter 3, *Project Window*, for detailed information on the relation between instrument drivers and function panels.) You can also use function panels to call functions in the project, as long as the functions are declared in the IEW. Thus, you can create function panels for functions that you call frequently, even if you do not have them in a separate file. See the *LabWindows/CVI Instrument Driver Developer Guide* for detailed information about creating function panels.

Accessing Function Panels

You can access a function panel for a library from the **Library** menu or for an instrument driver from the **Instrument** menu. After selecting an instrument or library name, you choose a panel by making selections from the Select Function Panel dialog box.

Functions are grouped in a multilevel structure called a *function tree*. This structure groups functions into various *classes* according to the operation they perform to make finding individual functions easier. When the Select Function Panel dialog box contains class names, you can select a class name to view the next level of the function tree, until you reach a list of Function Panel windows.

In certain cases, it is convenient to access library or instrument module function panels in a linear fashion, that is, by moving through the list of functions without using the tree structure. The

Select Function Panel dialog box has a **Flatten** checkbox which replaces the function class hierarchy with a list of all function panels at or below the current level. Once you have selected a function panel, four function panel commands—**Previous Panel**, **Next Panel**, **First Panel**, and **Last Panel**—give you access to function panels in this linear manner. See *The View Menu* section for more detailed information about using these commands.

You can access function panels in other ways, as well. For instance, you may want to return to a panel you recently used, or recall a panel from the text of a function call in a Source window. The commands that give you access to panels in these and other ways are in the **View** menu of the Source window. A similar set of commands exist in the **View** menu of the Function Panel window. See *The View Menu* section in this chapter, and *The View Menu* section in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*, for more information on using these commands.

Figure 5-1 shows a Function Panel window for an instrument driver function.

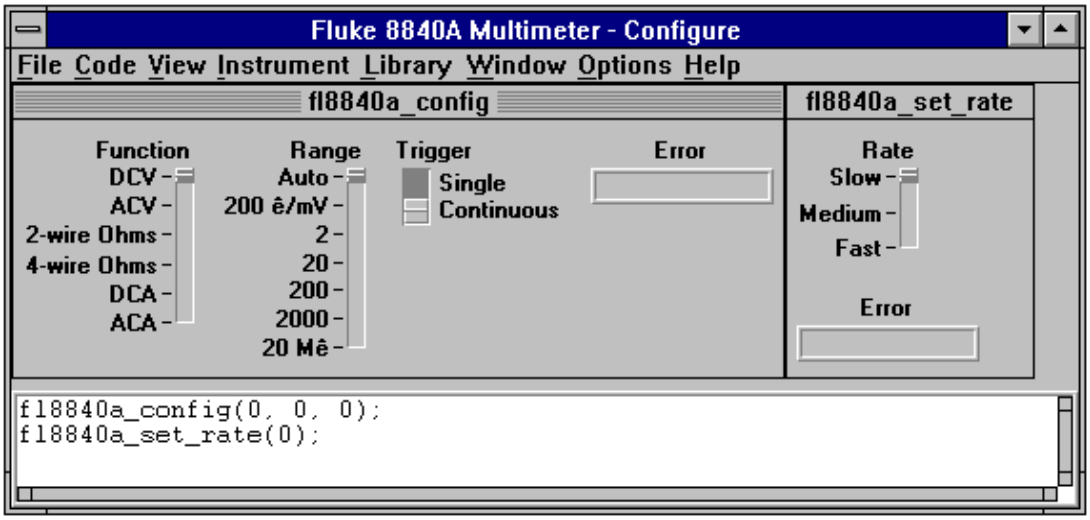

Figure 5-1. Instrument Driver Function Panel Window

The Function Panel window illustrated in Figure 5-1 is the *Fluke 8840A Multimeter - Configure*, containing two function panels corresponding to the two functions, `fl8840a_config` and `fl8840a_set_rate`. You can use the controls on the function panels to specify parameters for the functions. The Generated Code Box at the bottom of the window displays the function calls generated by these function panels.

Multiple Function Panels in a Window

The Function Panel window can contain more than one function panel as Figure 5-1 demonstrates. Each function panel corresponds to one function, with the controls on that function panel manipulating the parameters to that function call. Individual functions may be disabled by selecting **Function Call Disabled** from the **Options** menu. Disabled function calls

do not appear in the Generated Code Box and so you cannot execute or insert them into a Source window.

Generated Code Box

The Generated Code box at the bottom of the Function Panel window displays the code produced by the function panels when you manipulate the panel controls. The Generated Code box displays up to 3 lines of code at a time and is scrollable.

Toolbars in LabWindows/CVI

The LabWindows/CVI toolbar appears within function panels, in the function panel editor window, and in source windows. It gives you quick access to common commands, such as File Open and File Save. You can configure the toolbar to meet your needs and you can also choose not to display it. See the *Toolbars in LabWindows/CVI* section in Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows* in this manual for a full description of toolbar use and configuration.

Function Panel Controls

Function panel controls specify parameters in a function call. There are eight types of function panel controls, as illustrated in Figure 5-2.

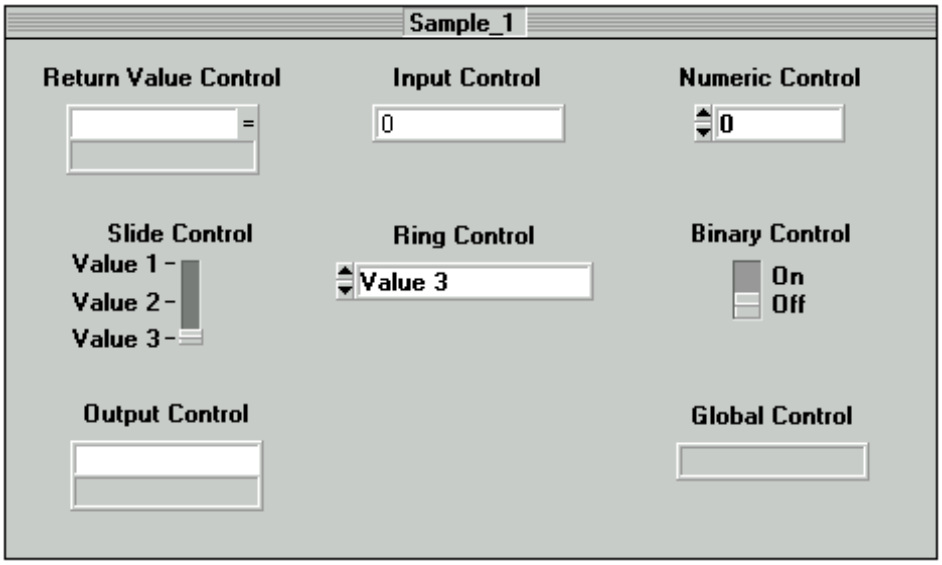

Figure 5-2. Function Panel Controls

When a function panel is displayed, input from the keyboard or mouse affects the control that is currently selected or highlighted. Pressing the <Tab> key selects the next control. Pressing <Shift-Tab> selects the previous control. To select a control with the mouse, click on the control. Pressing <Page Up> or <Page Down> moves the input focus across multiple function panels in one window. Pressing <Ctrl-Page Up> and <Ctrl-Page Down> moves from one function panel window to the next.

The way you specify parameter values differs for each type of control. The following sections contain instructions for specifying parameters for each type of control.

Specifying a Return Value Control Parameter

A *return value control* displays a value returned from a function as a return value rather than as a formal parameter.

For scalar return values, you can leave the control blank. LabWindows/CVI generates a temporary variable when you run the function panel.

If you type a variable name into a return control, the variable must be defined statically in the IEW or defined elsewhere and declared as `extern` in the IEW before executing the function. You can use the **Declare Variable** command from the **Code** menu to define the variables in the IEW. The type of value entered must agree with the data type of the control. To determine the data type of the control, press <F1> or click the right mouse button on the control to view the Help window. After executing the function, the return value control displays the value for the variable beneath the variable name.

Specifying an Input Control Parameter

An *input control* accepts a value typed in from the keyboard. An input control can have a default value associated with it. This value appears in the control when the panel is first displayed.

To specify a parameter for an input control, select the control and type in either a variable name, a numeric value, or a valid expression. Before executing a function panel window, any names typed into input controls must be defined statically in the IEW or defined elsewhere and declared as `extern` in the IEW. You can use the **Declare Variable** command from the **Code** menu to define variables in the IEW for use in the function panels. The type of value entered (constant, expression, simple variable, or array) must agree with the data type of the control. To determine the data type of the control, press <F1> or click the right mouse button on the control to view the Help window.

Specifying a Numeric Control Parameter

A *numeric* control behaves like an input control except that it accepts numeric values only.

If you want to type a variable name into a numeric control, use the **Toggle Control Style** command in the **Options** menu.

Specifying a Slide Control Parameter

With a *slide control* you select one item from a list of options. The position of the *slider*, the cross-bar on the slide control, determines the value LabWindows/CVI places in the function call.

To move the slider with the keyboard, press the up or down arrow key when the control is selected. As you move the slider, the corresponding argument in the function call in the Generated Code box changes. The <Home> and <End> keys move you to the top and bottom of the slide control, respectively. To move the slider with the mouse, click on the slider and drag it up and down, or just click on the desired position.

If you want to type a variable name into a slide control, use the **Toggle Control Style** command in the **Options** menu.

Specifying a Binary Control Parameter

The *binary control* is a limited version of the slide control that has only two positions.

To select the position of the binary control, press the up or down arrow key, or the <Home> or <End> key. To change the binary control with the mouse, click on the desired position.

If you want to type a variable name into a binary control, use the **Toggle Control Style** command in the **Options** menu.

Specifying a Ring Control Parameter

The *ring control* represents a range of values much like the slide control. A ring control displays only a single item from a list, instead of displaying the whole list at once as the slide control does. Each item has a different value associated with it that is placed in the function call.

To select an item from a ring control with the keyboard, use the up and down arrow keys to scroll through the list. Pressing the space bar displays the entire list of items for the selected ring control. To select an item from a ring control with the mouse, you can click on the up or down arrow of the ring control until the value you want appears, or you can click on the display field of the control and select the value you want directly from the list that appears.

If you want to type a variable name into a ring control, use the **Toggle Control Style** command in the **Options** menu.

Specifying an Output Control Parameter

The *output control* displays a value determined by the function you execute.

An output control parameter must be an array name or the address of a scalar or structure. For non-array parameters, you can leave an output control blank. LabWindows/CVI generates a temporary variable when you run the function panel. If the output control requires an array, or if you type a variable name into the output control, the variable must be defined statically in the IEW or defined elsewhere and declared as `extern` in the IEW before executing the function. You can use the **Declare Variable** command from the **Code** menu to define a variable in the IEW.

To specify a parameter for an output control, select the control and type in the desired variable name.

To view the value at an output control parameter after the function is executed, double-click on the lower half of the output control to open the variable display.

Using a Global Control

A *global control* displays the contents of global variables in a library function. You can use global controls to monitor global variables in a function that are not specifically returned as results by the function. These are read-only controls. You cannot alter the content, and the controls do not contribute a parameter to the generated code.

Common Control Function Panel

A Function Panel window can contain a special function panel called a *Common Control* function panel. The *n* controls on a Common Control function panel specify the first *n* parameters of all functions in the Function Panel window.

Convenient Viewing of Function Panel Variables

Select **Variable Value** or **Expression Value** from the **Code** menu for a convenient way to view the contents of arrays, structures, and global variables that exist in function panel controls. Depending on the type of the variable or expression, one of the Variable Display windows or the Watch window will appear with the variable or expression highlighted.

File Menu

This section contains a detailed description of the **File** menu for Function Panel windows as shown in Figure 5-3.

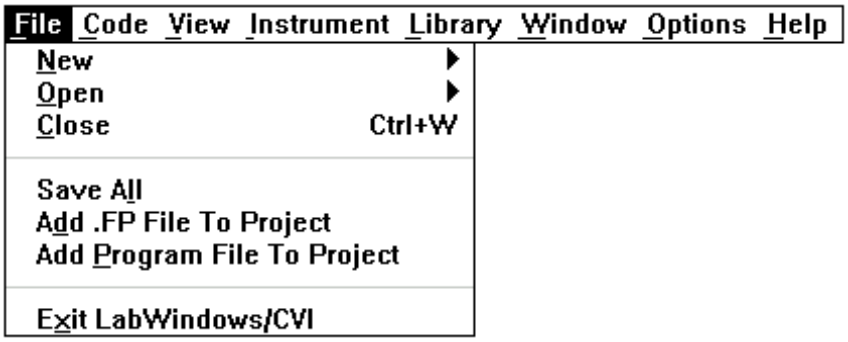

Figure 5-3. The File Menu

New

The **New** command operates the same as the **New** command in the Project window. See the **File** menu section of Chapter 3, *Project Window*, for more information on the **New** command.

Open

The **Open** command operates the same as the **Open** command in the Project window. See the **File** menu section of *Chapter 3, Project Window*, for more information on the **Open** command.

Close

The **Close** command closes the active Function Panel window.

Save All

The **Save All** command saves all open files to disk.

Add .FP File to Project

The **Add .FP File to Project** command adds the `.fp` file in the current Function Panel window to the project list.

Add Program File to Project

The **Add Program File to Project** command adds the instrument driver program file associated with the instrument driver or library of the current Function Panel window to the project list.

Exit LabWindows/CVI

The **Exit LabWindows/CVI** command closes the current LabWindows/CVI session. If you have modified any open files since the last save, or if any windows contain unnamed files, LabWindows/CVI prompts you to save them to disk.

Code Menu

This section contains a detailed description of the **Code** menu for Function Panel windows as shown in Figure 5-4.

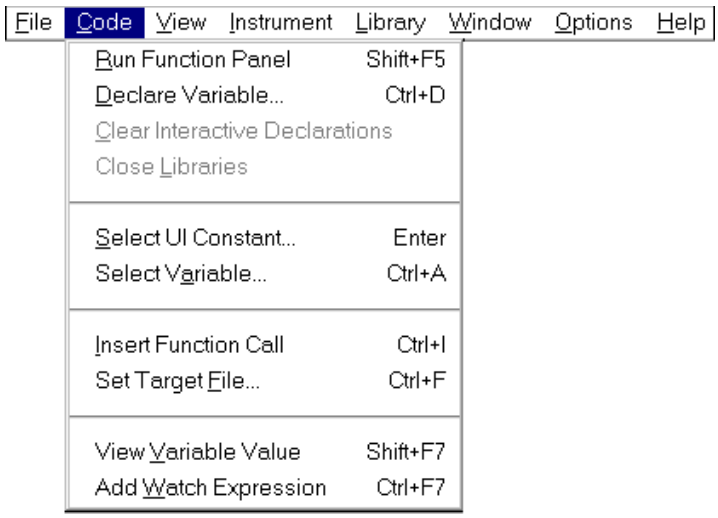

Figure 5-4. The Code Menu

Run Function Panel

Selecting the **Run Function Panel** command executes the code in the Generated Code Box. When you select **Run Function Panel**, the following actions take place.

- LabWindows/CVI automatically inserts the header file for the library or instrument driver into the Interactive Execution window if it is not already there.
- LabWindows/CVI generates temporary variables for blank scalar output controls.

- LabWindows/CVI copies the generated function(s) to the Interactive Execution window.
- LabWindows/CVI executes the code. While executing, the word *Running* appears in the upper left corner of the function panel menu bar.
- If outputs, such as values returned from the function, are to be displayed on the panel, LabWindows/CVI updates the controls associated with outputs to display the current value.

Declare Variable...

Use **Declare Variable** to declare a variable to be placed in the currently active control on the function panel. When you select **Declare Variable**, a dialog box like the one shown in Figure 5-5 appears on the screen.

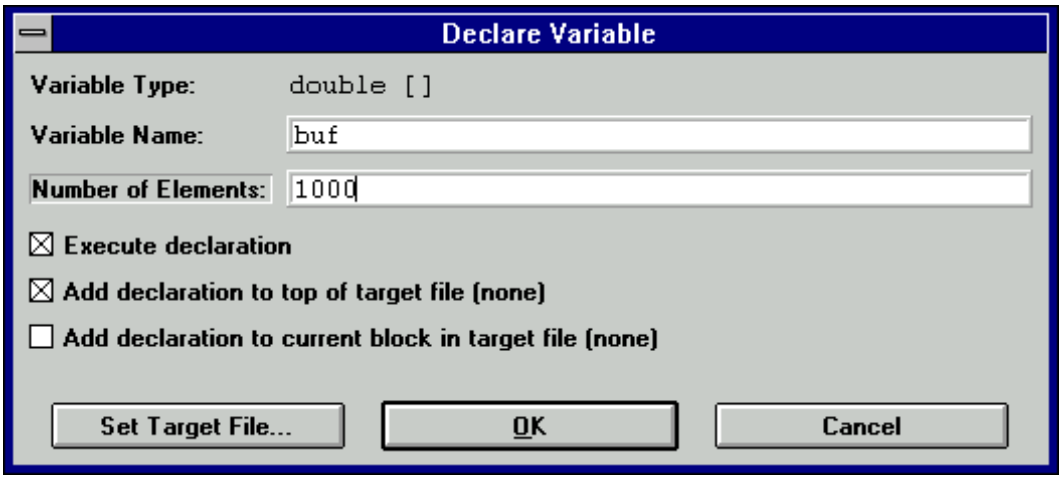

Figure 5-5. The Declare Variable Dialog Box

To declare a variable with the Declare Variable dialog box, enter the name of the variable you want to declare in the **Variable Name** text box. LabWindows/CVI automatically prefixes scalar output variables with an ampersand (&) when they are declared.

The **Variable Type** message indicates the data type associated with the currently active control on the panel. Some panel controls can be used with more than one data type. In such cases, a ring selector enables you to select the data type.

The **Number of Elements** box appears when the currently active control is for an array or a string. Enter the number of elements.

Select the action options you want. When you select **Execute declaration**, LabWindows/CVI executes the variable declaration immediately in the IEW. When you select **Add declaration to top of target file (`filename`)**, LabWindows/CVI inserts a copy of the declaration at the top of the specified target which you set using the **Set Target File** command. When you select **Add declaration to current block in target file (`filename`)**, LabWindows/CVI inserts a copy of

the declaration at the beginning of the code block in which the keyboard cursor is currently placed in the target file. A code block is delimited by curly braces.

Execute the **OK** command button to declare the variable according to the options you have selected.

Click on the **Cancel** button to cancel the operation and remove the Declare Variable dialog box from the screen.

When you use the **Declare Variable** command, the variable is always declared using the static storage class.

In addition to generating the variable declaration, the **Declare Variable** command also places the variable name in the currently active control. The previous contents of the control are overwritten.

If the currently active control already contains a syntactically correct variable name, it appears in the **Variable Name** text box when the Declare Variable dialog box first appears.

Clear Interactive Declarations

Variables declared in the IEW remain in effect until you explicitly remove them. This lets you use these same variables in succeeding executions of the IEW. It also enables different function panels to access the same variables.

The **Clear Interactive Declarations** command removes the variables without deleting the contents of the Interactive Execution window.

Close Libraries

The **Close Libraries** command closes the LabWindows/CVI libraries you have accessed through function panels and the IEW. This command is useful if you are working from function panels or the IEW and you want to close the LabWindows/CVI libraries without clearing your interactive variables.

Note: ***LabWindows/CVI automatically closes the libraries before and after you run a project.***

Select UI Constant...

The **Select UI Constant** command can help you use the function panels for the User Interface Library. The command lets you select from the list of constant names associated with the object in your `.uir` files. It also helps you select attribute and value constant names from the `userint.h` file.

Selecting Constants from .uir Files

When you specify a parameter for an input control that can accept a panel resource ID, control ID, menu bar resource ID, menu ID, or menu item ID, use **Select UI Constant** to bring up the Select UIR Constant dialog box, as shown in Figure 5-6.

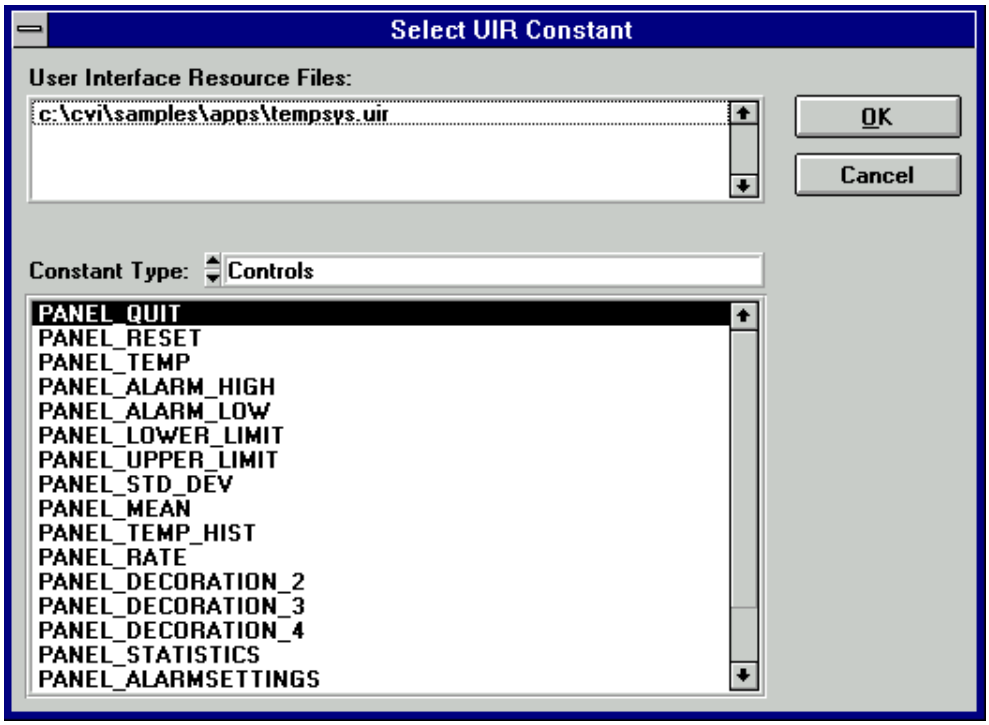

Figure 5-6. The Select UIR Constant Dialog Box.

The list box at the top of the dialog box lists all of the `.uir` files open or in the project. Only constants from the currently selected `.uir` file are shown in the list box at the bottom. Click on a file to select it.

The Constant Type ring control allows you to select which category of constant name is shown.

When you press the **OK** button, the currently selected constant name is copied into the function panel control.

Note: ***If you attempt to use* Select UI Constant *on the Panel Handle and Menu Bar Handle controls that appear on most User Interface Library function panels, an error message appears. These controls take the values returned from*** `LoadPanel` ***and*** `LoadMenuBar`***, so an attempt to select*** `.uir` ***constants will fail.***

You can use **Select UI Constant** in user-defined panels. That way, the command is available to function panels for user libraries that are built on top of the User Interface Library.

Selecting attribute constants from userint.h

The **Select UI Constant** command has special behavior on the Attribute ring controls in panels for functions such as `GetCtrlAttribute`, `SetCtrlAttribute`, `GetPanelAttribute`, and `SetPanelAttribute`.

When you execute the command on an attribute ring control, the Select Attribute Constant dialog box appears, as shown in Figure 5-7.

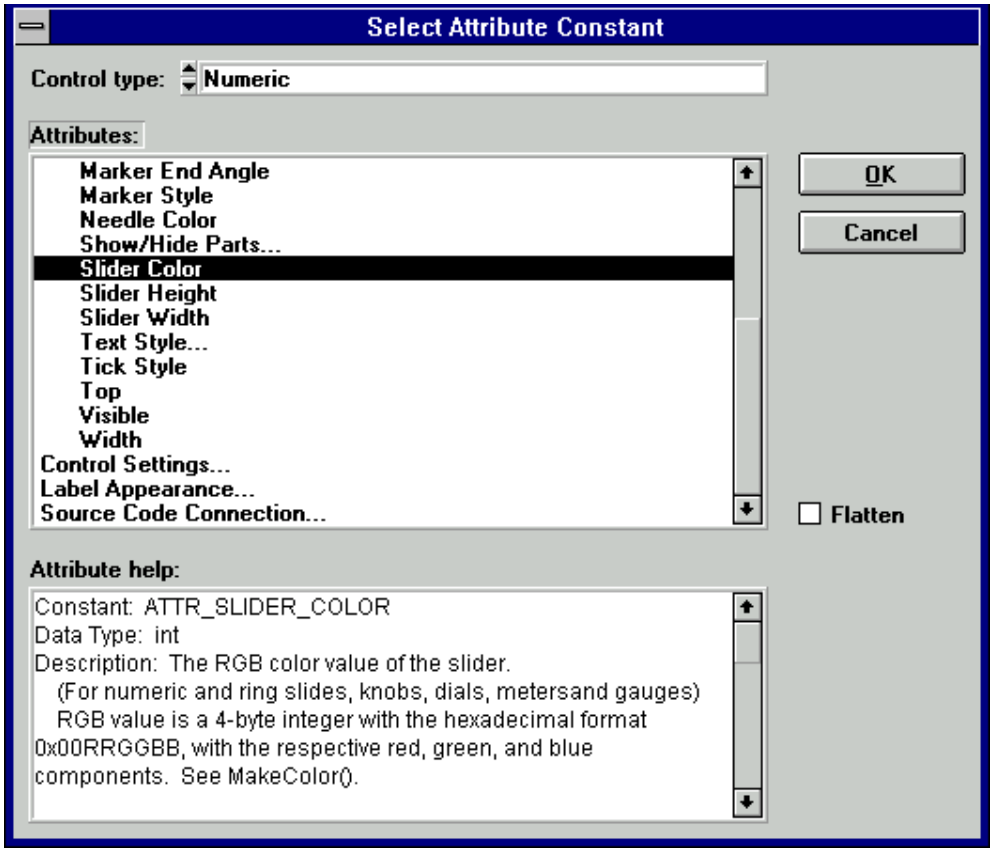

Figure 5-7. The Select Attribute Constant Dialog Box

The **Control Type** ring allows you to restrict the list of attributes to those applicable to a particular control type.

The **Attributes** list box displays the attributes valid for the selected control type. The attributes are organized under classes. (Classes are denoted by a trailing ellipsis, "**. . .**"). To see a list of all of the attributes without the classes and in alphabetical order, select the **Flatten** control.

The **Attribute help** text box displays help information for the currently selected attribute.

Double-click on an attribute, or press **OK**, to change the function panel ring control to that attribute.

Notice that when you attempt to operate the Attribute ring control in the function panel as a normal ring control, exactly the same dialog box appears in place of the pop-up menu that normally appears on a ring control.

Selecting value constants from userint.h

The **Select UI Constant** command has special behavior on the Attribute Value input and output controls in panels for functions such as `GetCtrlAttribute`, `SetCtrlAttribute`, `GetPanelAttribute`, and `SetPanelAttribute`.

The operation of **Select UI Constant** on an Attribute Value control depends on the attribute currently selected in the Attribute ring control on the same function panel.

If the Attribute ring control is set to an attribute for which there is no small, discrete set of values, a dialog box appears repeating the help information for the attribute. If, on the other hand, there is a small, discrete set of values, the Select Attribute Value dialog box appears, as in Figure 5-8.

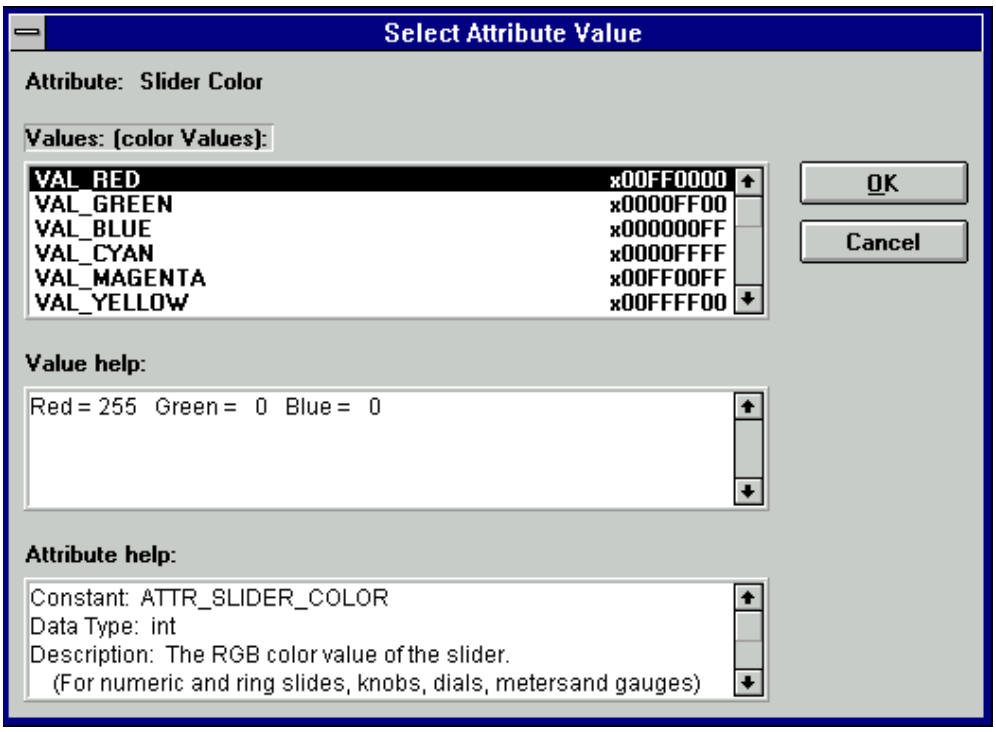

Figure 5-8. The Select Attribute Value Dialog Box

When the value shown in the Value list box is a constant name, the actual value appears on the right-hand side of the list box.

If the Attribute Value control is an input control (for example, on `SetCtrlAttribute` or `SetPanelAttribute`), double-click on an entry in the Values list, or press **OK**, to copy it into the Attribute Value control on the function panel.

If the Attribute Value control is an output control (for example, on `GetCtrlAttribute` or `GetPanelAttribute`), and a value appears in the bottom half of the control (because you have executed the function panel), CVI will select, when possible, the value in the Values list that corresponds to the value shown in the bottom half of the output control. The list box entry that contains that value is marked with an arrow symbol to the left.

Select Variable

The **Select Variable** command gives you a list of previously used variables or expressions having data types that are compatible with the currently active function panel control. The command is enabled only when the currently active function panel control is one that accepts text entry. When you select a variable or expression from the list, it is copied into the function panel control. The Select Variable command can significantly reduce the amount of keyboard entry needed when using function panels.

When you execute the **Select Variable** command, the Select Variable or Expression dialog box appears.

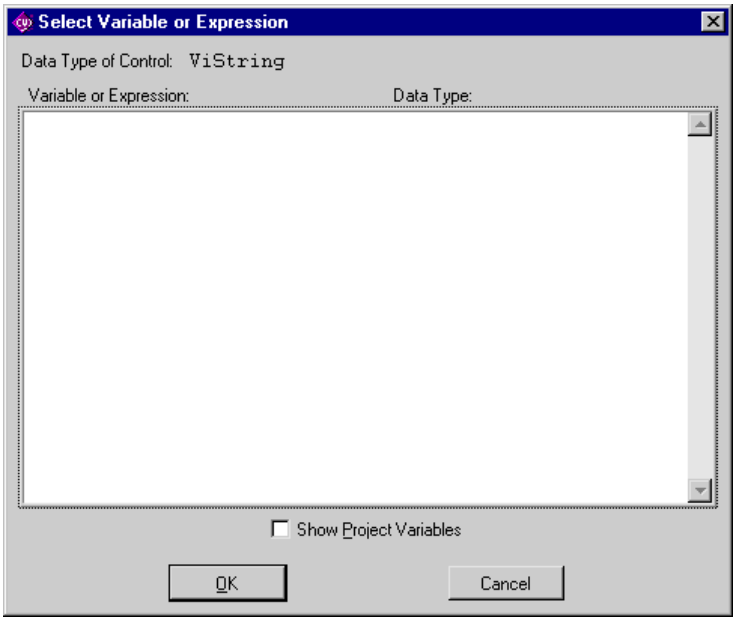

Figure 5-9. The Select Variable or Expression Dialog Box

- **Data Type of Control**—Indicates the data type of the currently active function panel control.
- **Variable or Expression**—This list box column contains the variables and expressions that have data types compatible with the data type of the control.
- **Data Type**—This list box column indicates the data type of each variable and expression.
- **Show Project Variables**—This option adds to the list box global variables (both static and non-static) defined in project files that have been successfully compiled.

- **OK**—This button dismisses the dialog box and copies the variable or expression into the function panel control. It may add a leading ampersand (`&`) when the function panel control is an output control. It may add one or more leading asterisks (`*`) or a trailing array indexation (`[0]`) when needed to correctly match the data type of the control.
- **Cancel**—This button cancels the operation.

What Is Included in the List Box

The following items are considered for inclusion in the list box.

- Variables declared in the Interactive Execution window.
- Variables declared using the **Declare Variable** command in a function panel.
- Variables or expressions used in function panels that are executed.
- Variables or expressions used the function panels from which code is inserted into a source window.
- User interface panel handle variables added to a Source window by CodeBuilder.
- Variables declared as global or static global in a project file that has been successfully compiled, but only if the **Show Project Variables** option has been enabled in the dialog box.

Some or all of these items are cleared from memory when you unload the current project or execute the **Clear Interactive Declarations** command in the **Build** menu.

Data Type Compatibility

Compatibility between data types is a more complex issue than might be expected. In the end, heuristics must be used. The heuristics differ based on whether the variable is known to the compiler.

Variables known to the compiler include variables declared in the Interactive Window, and variables declared in project files that have been successfully compiled. For such variables, the following are the major factors in determining whether the variable is type-compatible with a function panel control.

- Data types declared with the `typedef` keyword are reduced to their most intrinsic type, as long as the `typedef` is known to the compiler. For example, assume the following declarations have been processed by the compiler.

```
typedef int   typeA;
typedef int   typeB;
typedef typeB typeC;
```

Then a variable of type `typeA` is an exact match for a function panel control having type `typeC`.

- All numeric types are considered compatible with each other, except that floating point variables or expressions are not considered compatible with integer function panel controls.
- Types that have the same base type but differ in levels of indirection are considered to be compatible. For example, the following are all compatible:

```
int
int *
int **
int [];
```

An expression or a variable name not known to the compiler must match exactly to the function panel control's data type to be included in the list box. (An example of a variable name not known to the compiler is one used in a function panel from which code has been inserted into a Source window.)

Note: ***An expression or variable name not known to the compiler can be associated with multiple data types. For instance, you might use the same variable name in an*** `int` ***control and a*** `double` ***control. If the variable is not known to the compiler, LabWindows/CVI has no way of knowing the true data type of the variable name. Thus, you might see the variable name associated with different data types.***

Sorting of List Box Entries

The entries in the list box are first sorted by data type. The most compatible data types are shown first. (Exception: Some function panel controls are declared with "meta" data types, such as `numeric array`, `any array`, or `any type`. Such controls are equally compatible with a wide range of data types. In this case, the order of data types does not indicate differing degrees of compatibility.)

Within each data type, the entries are sorted alphabetically by the variable/expression text.

Insert Function Call

The **Insert Function Call** command copies the generated code to the selected window at the current location of the keyboard cursor. You can copy code to any open Source window or to the IEW. You determine the destination window with the **Set Target File** command in the **Code** menu.

If the destination window contains selected text, LabWindows/CVI displays a dialog box giving you the option of replacing the selected text or inserting the generated code after the selected text. See the discussion of the **Recall Function Panel** command in *The View Menu* section of

Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows* for more information.

Set Target File...

Use the **Set Target File** command to set the destination file for the **Insert Function Call** command. **Set Target File** has a submenu you can use to select from any open Source windows or the IEW. A checkmark denotes the currently selected target file.

Variable Value

Variable Value is a convenient way to view the contents of arrays, structures, and global variables that appear in a function panel. Highlight the variable that you want to see and select **Variable Value**. Depending on the type of the variable, the Variable, Array, or String Display appears with the variable highlighted.

Expression Value

Expression Value is a convenient way to view the value of an expression that appears in a function panel. Highlight the expression you want to see and select **Expression Value**. The Watch window appears with the expression highlighted.

View Menu

This section contains a detailed description of the **View** menu for Function Panel windows as shown in Figure 5-10.

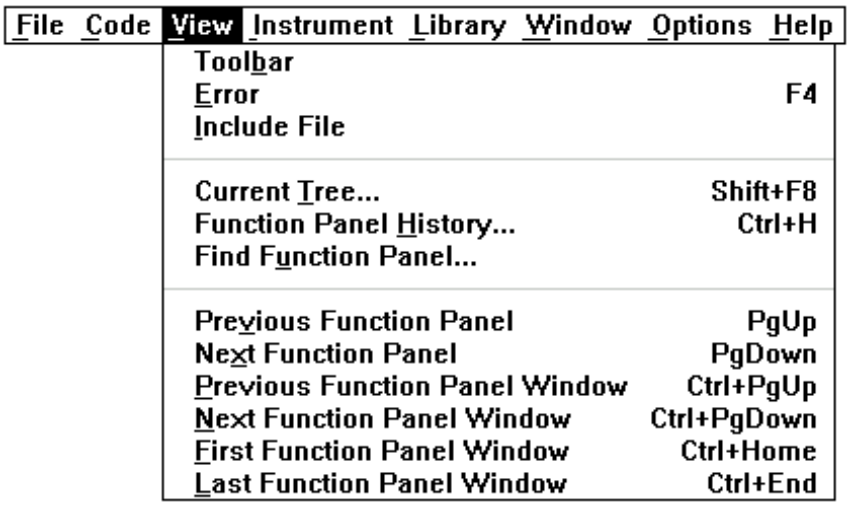

Figure 5-10. The View Menu

Toolbar

Use the **Toolbar** command to toggle the visibility in the Function Panel window toolbar.

Error

If an error occurs while a function panel is running, you can use the **Error** command to toggle between the error and the code in the Generated Code Box.

Include File

The **Include File** command displays the include file associated with the library or instrument driver in a Source window. The include file contains all of the function prototypes for the library or instrument driver.

Current Tree...

The **Current Tree** command displays the Select Function Panel dialog box for the most recently used function panel, making it easy for you to return to the location of the current panel in the function tree.

Function Panel History...

The **Function Panel History** command displays a scrollable list of the function panels you have used during the current LabWindows/CVI session. You can display function panels from the list as new windows or you can overwrite the current Function Panel window.

Find Function Panel...

When you select the **Find Function Panel** command, a dialog box appears in which you can enter the name of a function. You can enter just a substring, and Find Function Panel finds all functions that contain that substring anywhere in their names. For instance, if you enter

```
ctrl
```

and click on **OK**, a dialog box appears with a list of functions including `NewCtrl`, `SetCtrlVal`, `GetCtrlVal`, and so on.

You can use a regular expression as your search string. See Table 4-1, *Regular Expression Characters*, for a list of regular expression characters.

If there is a function panel for the function, LabWindows/CVI displays the panel. If there are two or more Function Panel windows for the function, LabWindows/CVI displays a list of the Function Panels.

The shortcut key for **Find Function Panel** is <Ctrl-Shift-P>.

Previous Function Panel

The **Previous Function Panel** command displays the previous Function Panel in the current Function Panel window.

Next Function Panel Window

The **Next Function Panel** command displays the next Function Panel in the current Function Panel window.

Previous Function Panel Window

The **Previous Function Panel Window** command calls up the Function Panel window that precedes the current Function Panel window in the same Function Panel Tree.

Next Function Panel Window

The **Next Function Panel Window** command calls up the Function Panel window that follows the current Function Panel window in the same Function Panel Tree.

The rotation order for the Function Panel Tree is circular. If the first Function Panel window in the tree is visible on the screen, selecting **Previous Function Panel Window** displays the last Function Panel window in the tree. If the last Function Panel window in the tree is visible, selecting **Next Function Panel Window** displays the first Function Panel window in the tree.

First Function Panel Window

The **First Function Panel Window** command displays the first Function Panel window in the Function Tree.

Last Function Panel Window

The **Last Function Panel Window** command displays the last Function Panel window in the Function Tree.

Instrument Menu

The **Instrument** menu for Function Panel windows behaves the same as the **Instrument** menu in the Project window. See *The Instrument Menu* and the *Using Instrument Drivers* sections in *Chapter 3, Project Window*, for command descriptions.

Library Menu

The **Library** menu for Function Panel windows behaves the same as the **Library** menu in the Project window. See *The Library Menu* section in *Chapter 3, Project Window*, for command descriptions.

Window Menu

The **Window** menu for Function Panel windows behaves the same as the **Window** menu in the Project window. See *The Window Menu* section in *Chapter 3, Project Window*, for command descriptions.

Options Menu

This section contains a detailed description of the **Options** menu for Function Panel windows as shown in Figure 5-11.

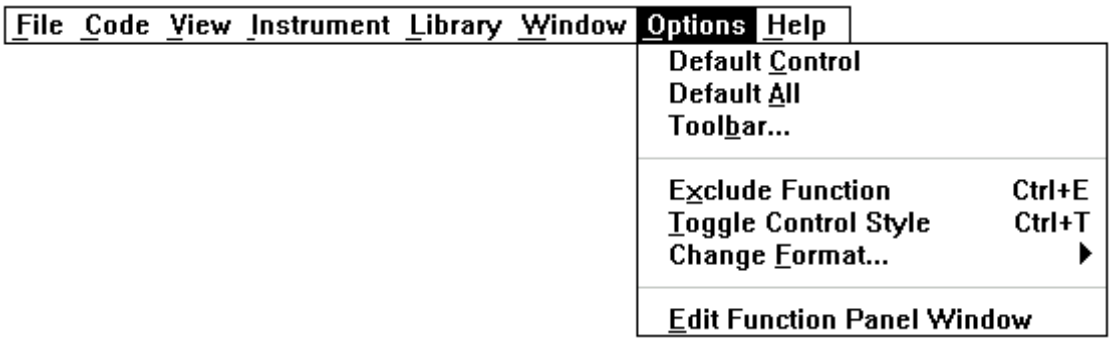

Figure 5-11. The Options Menu

Default Control

Default Control resets a control to its default value.

Default All

Default All resets all the controls on the current Function Panel window to their default values.

Toolbar...

Use the **Toolbar** command to select which icons appear in the Function Panel window toolbar.

Exclude Function

The **Exclude Function** command disables the current function panel, so that the function call does not appear in the Generated Code Box and is not in effect when you use the **Run** or **Insert** commands from the **Code** menu.

Toggle Control Style

Slide, binary, and ring controls insert a number into a function call in the Generated Code box. The value of this number depends on the option chosen on the control. You can override the configured values of these controls by using the **Toggle Control Style** command.

Toggle Control Style replaces a slide, binary, or ring control with an input control. You can use this input control to enter a variable name, a constant, or an expression. This entry appears in the Generated Code Box in the same position as the parameter produced by the original control.

The variable name or constant that you enter must match the type specified for the control, such as short, long, single-precision, double-precision, string, and so on. Otherwise, a syntax error occurs when you execute the function.

Change Format...

Change Format lets you change the numeric format for scalar controls. The list of formats depends on the data type associated with the control.

You can display short and long data types in decimal, hexadecimal, octal, or ASCII form. You can display real numbers in floating-point or scientific format.

Edit Function Panel Window

Edit Function Panel Window puts the Function Panel window in edit mode. See Chapter 3, *The Function Panel Editor*, in the *LabWindows/CVI Instrument Driver Developers Guide* for information about editing instrument function panels.

Note: ***You cannot edit the function panels of the LabWindows/CVI libraries or user libraries.***

Help Menu

The **Help** menu provides information about the current function panel and its controls. The **Help** menu is shown in Figure 5-12.

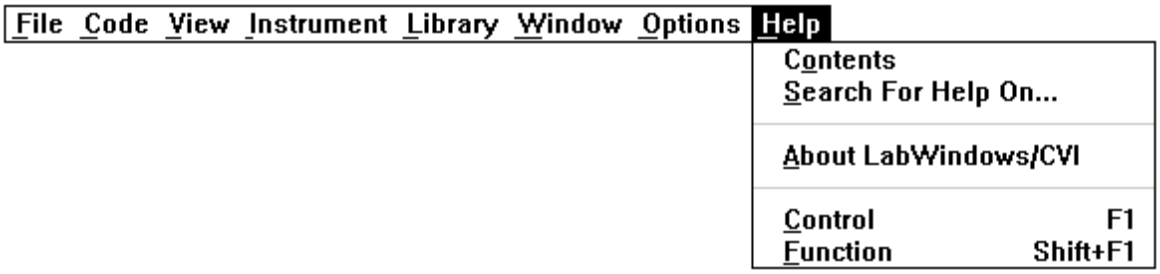

Figure 5-12. The Help Menu

Control

The **Control** command displays help information about the currently highlighted control. To select control help with the mouse, click the right mouse button anywhere on the desired control.

Function

The **Function** command displays general information about the function generated by the current Function Panel. To select function help with the mouse, click the right mouse button on the Function Panel.

Chapter 6
Variable Display and Watch Windows

This chapter describes the Variable Display and Watch Windows. You use these windows to inspect and modify the values of program variables. You can invoke these windows either when no program is running or when a program is suspended at a breakpoint.

The Variable Display window shows the names and types of all variables, including arrays and strings. The current values of numeric scalars, values and contents of pointers, and string contents are also shown in the Variable Display window.

Note: ***When strings are displayed in ASCII format, there is no visual distinction between a space (ASCII 32) and a NUL byte (ASCII 0). You can see the difference by displaying the string in decimal format.***

Variable Display Window

To view the Variable Display window, select the **Window** » **Variables** option in the active LabWindows/CVI window. You can also invoke the Array Display window for the currently highlighted variable from a Source or Function Panel window with the **Run » Variable Value** command in the Source window, or the **Code** » **Variable Value** Command in the Function panel window. An example of the Variable Display window is shown in Figure 6-1.

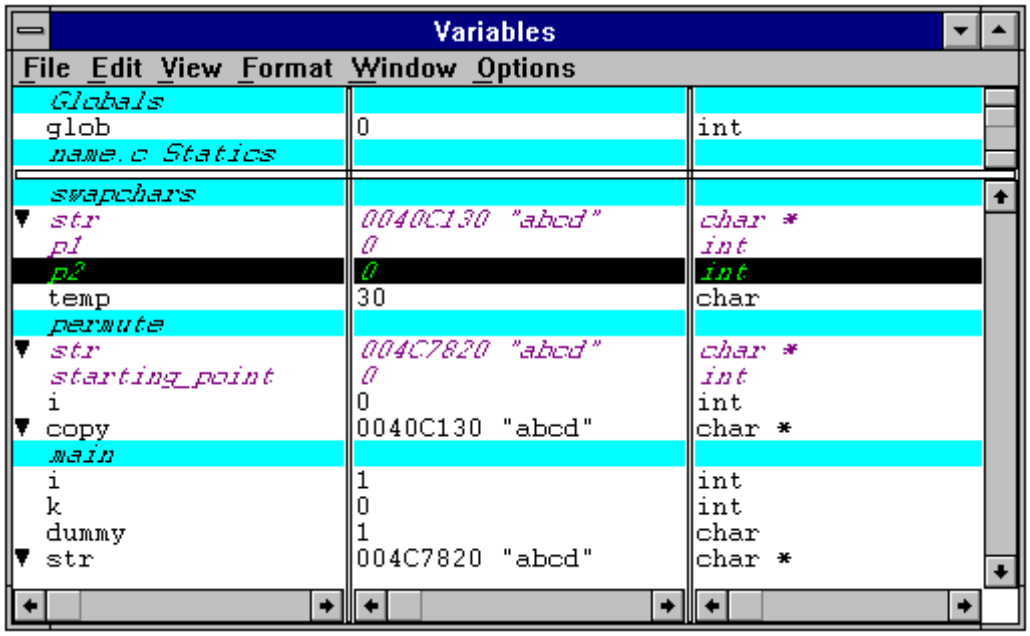

Figure 6-1. The Variable Display Window

The Variable Display window shows all currently defined variables in LabWindows/CVI. Variables in this window are updated at each breakpoint. The vertical bars separate the window into three scrollable fields: name, value, and variable type. You can change the width of the fields by dragging the vertical bars with the mouse. The window is also divided into two horizontal sections; the Global subwindow and the Function subwindow.

The Global subwindow displays the following variables

- Project globals which include all global variables not declared as `static`
- Interactive Execution window variables, declared in the Interactive Execution window
- Global variables declared as `static`

The Function subwindow displays currently active function parameters and local variables from active functions. The variable list for each function appears in a different section. For any given function, the variable display lists formal parameters first, followed by local variables. Formal parameters are displayed in italics.

There are several icons that appear to the left of certain variables as shown below:

The variable on this line is the starting pointer to a block of defined data such as an array, string, or structure. Clicking on this icon or selecting **Expand Variable** from the **View** menu expands the variable so that you can see each element or member. See the discussion of **Expand Variable** in *The View Menu* section later in this chapter.

The variable on this line is the starting pointer to a block of defined data that is currently being viewed in expanded form. Clicking on this icon or selecting **Close Variable** from the **View** menu closes the variable so that you see only the starting pointer. See the discussion of **Close Variable** in *The View Menu* section later in this chapter.

The variable on this line is a member of a structure which is a parent pointer to another structure of the same type. Clicking on this icon or selecting **Follow Pointer Chain** from the **View** menu replaces the current structure with the child structure that the pointer references. See the discussion of **Follow Pointer Chain** in *The View Menu* section later in this chapter.

The variable on this line is a child structure in a chain whose parent structure pointer is not displayed. Clicking on this icon or selecting **Retrace Pointer Chain** from the **View** menu replaces the current structure with its parent. See the discussion of **Retrace Pointer Chain** in *The View Menu* section later in this chapter.

Watch Window

The Watch window is similar in nature to the Variable Display window, except that you can select your own set of variables *and expressions* to view in the Watch window. By default,

variables and expressions in the Watch window are updated at each breakpoint, but you can also set them to update continuously and cause a breakpoint when their values change. To activate the Watch window, select **Window » Watch** in the active LabWindows/CVI window. An example of the Watch window is shown in Figure 6-2.

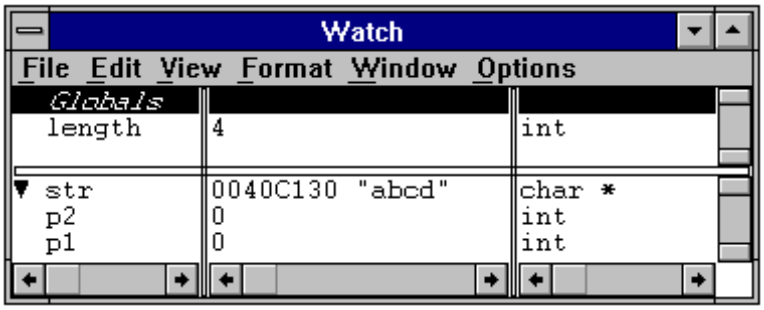

Figure 6-2. The Watch Window

Select Watch window variables and expressions from the Variable Display window using the **Options » Add Watch Expression** command. The **Watch** command invokes the dialog box shown in Figure 6-3.

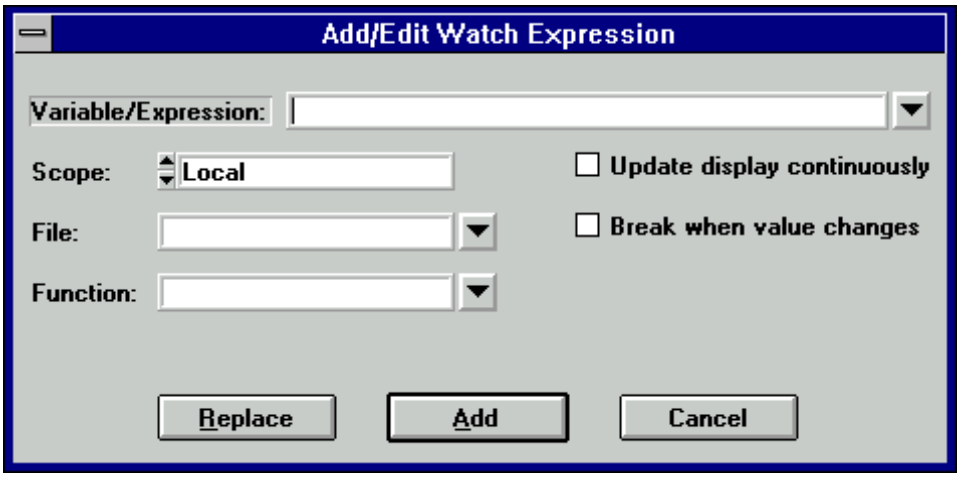

Figure 6-3. The Add/Edit Watch Expression Dialog Box

The controls in the Add/Edit Watch Expression dialog box are used as follows:

- **Variable/Expression** contains the variable or expression to be placed into the Watch window.
- **Scope** corresponds to whether the variable or expression variables are global to the project, global to a file, local to a function, or global to the Interactive Execution window.
- **File** is the name of the file where the variable or expression variables are defined if they are global to a file or local to a function.

- **Function** is the name of the function where the variable or expression variables are defined if they are local to a function.
- **Update display continuously** causes the variable or expression to be evaluated and updated on the Watch window between each statement in your program while the program is running.
- **Break when value changes** suspends the program when the value of the variable or expression changes.
- **Replace** replaces the existing attributes of the current variable or expression of the same name in the Watch window with the current attributes of the dialog box. **Replace** is only available when the dialog box is invoked from the Watch window.
- **Add** inserts the variable or expression into the Watch window.
- **Cancel** aborts the operation.

You can add Watch expressions to the Watch window directly from a Source window or a Function Panel window. To add a watch expression from a Source window, highlight the expression and select **Run » Expression Value**. To add a watch expression from a Function Panel window, highlight the expression and select **Code » Add Watch Expression**.

File Menu

This section contains a detailed description of the **File** menu for the Variable Display and Watch windows.

The **File** menu is shown in Figure 6-4.

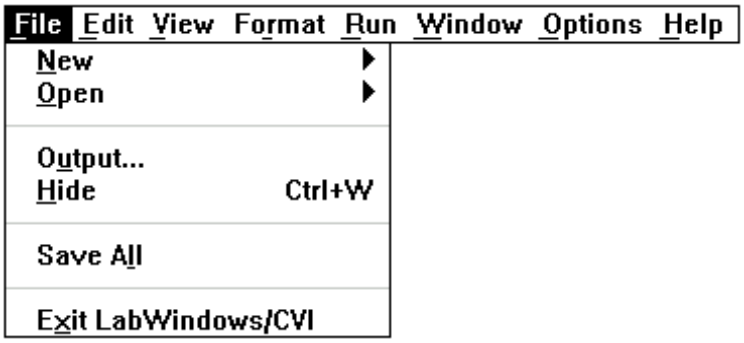

Figure 6-4. The File Menu

New

The **New** command operates the same as in the Project window. For a description of this command, see the discussion of the **File » New** command in *Chapter 3, Project Window*.

Open

The **Open** command operates the same as in the Project window. For a description of this command, see the discussion of the **File » Open** command in *Chapter 3, Project Window*.

Output...

The **Output** command writes the contents of the window to an ASCII file on disk. When you select **Output**, a dialog box appears prompting you to specify the name of the file.

Hide

The **Hide** command visually closes a window while retaining the contents in memory.

Save All

The **Save All** command saves all open files to disk.

Exit LabWindows/CVI

The **Exit LabWindows/CVI** command closes the current LabWindows/CVI session. If any open files have been modified since the last save, or if any windows contain unnamed files, you are prompted to save them to disk.

Edit Menu for the Variable Display Window

This section contains a detailed description of the **Edit** menu for the Variable Display and Watch windows.

The **Edit** menu for the Variable Display window is shown in Figure 6-5.

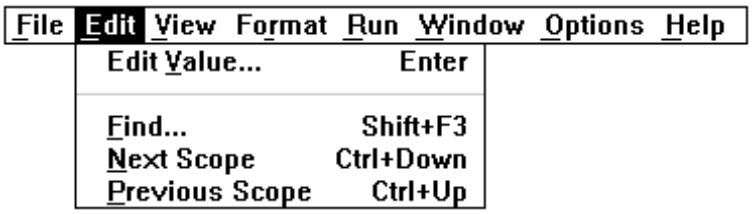

Figure 6-5. The Edit Menu in the Variable Display Window

Edit Value...

You can change the value of a variable with the **Edit Value** command. You can invoke the **Edit Value** command with the mouse by double-clicking on the variable name. When the dialog box appears, type in the new value.

The value that you enter in the Edit dialog box depends upon the type and display format of the variable, as shown by the following items.

- Edit integers and longs in the format in which they are displayed.
- Edit reals in either scientific or floating-point format, regardless of the display format.
- Edit individual array elements if the array has been expanded using the **Expand Variable** command.
- Edit individual bytes of strings if the string has been expanded using the **Expand Variable** command. The bytes appear in the integer format specified in the **Format** menu unless the **ASCII** command is activated from the same menu.

Find...

The **Find** command invokes the dialog box shown in Figure 6-6.

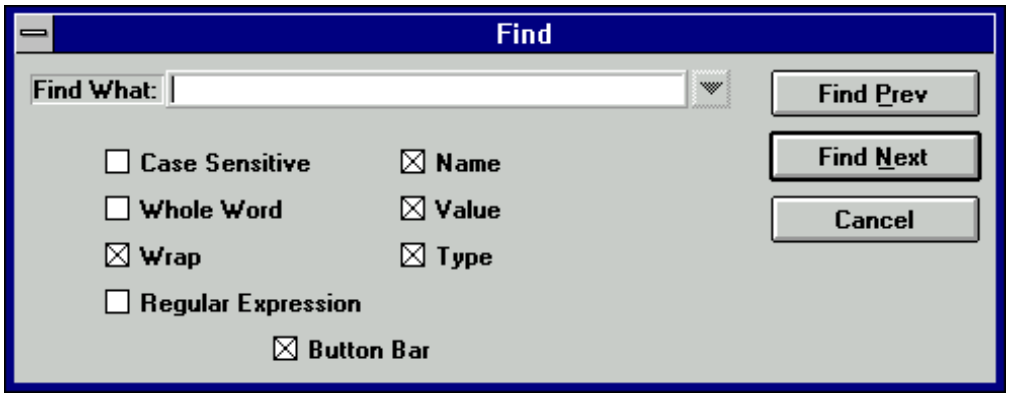

Figure 6-6. The Find Dialog Box in the Variable Display Window

- The **Case Sensitive** option finds only the instances of the specified text that match exactly. For example, if CHR is the specified text, the **Case Sensitive** option finds CHR but not Chr.
- The **Whole Word** option finds the specified text only when it is surrounded by spaces, punctuation marks, or other characters not considered parts of a word. The characters A through Z, a through z, 0 through 9, and underscore (_) are considered parts of a word.
- The **Wrap** option specifies to continue searching from the beginning of the window once the end of the window has been reached.
- If you select the **Regular Expression** option, certain characters in the **Find What** box are treated as regular expression characters instead of literal characters. The regular expression characters are described in Table 4-1 of Chapter 4, *Source Interactive Execution and Standard Input/Output Windows*.
- Activate the **Name** option to include the variable name field of the Variable Display/Watch window in the search.
- Activate the **Value** option to include the value field of the Variable Display/Watch window in the search.
- Activate the **Type** option to include the variable type field of the Variable Display/Watch window in the search.
- Use the **Button Bar** option to enable or disable the built-in dialog box for interactive searching as shown in Figure 6-7.

Figure 6-7. The Find Button Bar

Find Prev and **Find Next** search for the closest previous or next occurrence of the specified text. **Stop** terminates the search, leaving the highlight on the current line. **Return** terminates the search, moving the highlight to where it was when the search was initiated.

The search hot-keys remain active even if the Button Bar is disabled. Use the **Keyboard Help** command in the **Options** menu of a Source window for a list of the search hot-keys.

Next Scope

In the function subwindow, **Next Scope** highlights the function that called the current function. In the global subwindow, **Next Scope** highlights the next module. This command is not available in the Watch window.

Previous Scope

In the function subwindow, **Previous Scope** highlights the function that was called directly by the current function. In the global subwindow, **Previous Scope** highlights the previous module. This command is not available in the Watch window.

Edit Menu for the Watch Window

The **Edit** menu for the Watch window is shown in Figure 6-8.

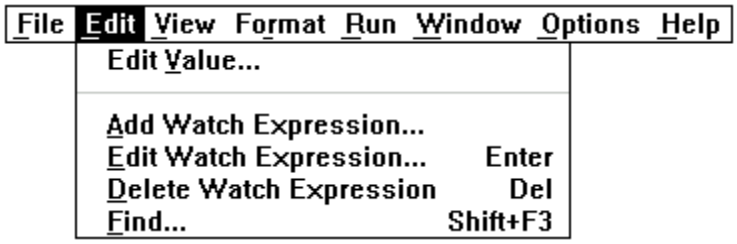

Figure 6-8. The Edit Menu in the Watch Window

Edit Value...

The **Edit Value** command operates the same as in the Variable Display window.

Add Watch Expression...

Add Watch Expression invokes the Add/Edit Watch Expression dialog box. This dialog box is explained in *The Watch Window* section earlier in this chapter.

Edit Watch Expression...

Edit Watch Expression invokes the Add/Edit dialog box for the selected watch expression.

Delete Watch Point

Delete Watch Point removes the selected watch variable/expression from the Watch window. This command is not available in the Variable Display window.

Find...

The **Find** command operates the same as in the Variable Display window.

View Menu

This section contains a detailed description of the **View** menu for the Variable Display and Watch windows.

The **View** menu is shown in Figure 6-9.

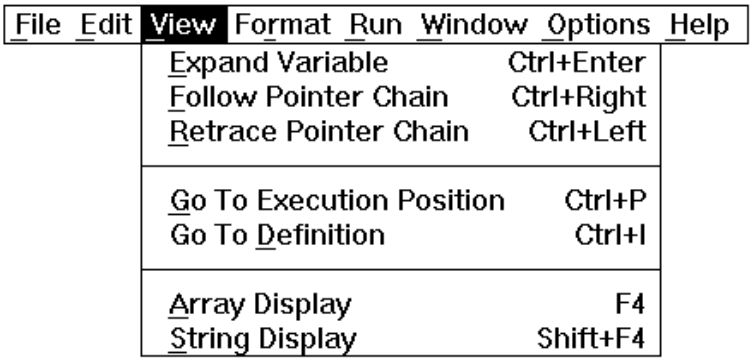

Figure 6-9. The View Menu

To use these commands, select a particular array or string by clicking on it with the mouse or using the UP and DOWN arrow keys, then access the command from the **View** menu.

Expand Variable

The Variable Display and Watch windows can display arrays, strings, and structures in closed form or expanded form. In closed form, you see only the name and address of the aggregate variable next to the icon as shown in Figure 6-10.

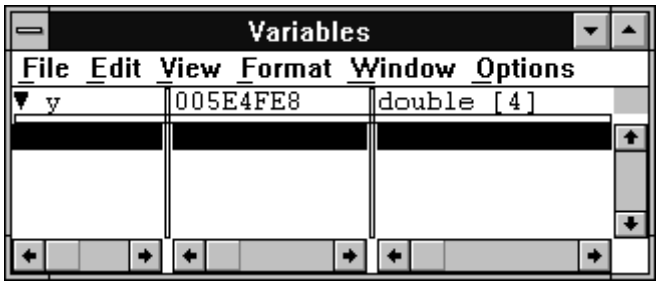

Figure 6-10. A Closed Array in the Variable Display Window

In expanded form, the icon changes to ● and you see the individual elements and their values as shown in Figure 6-11.

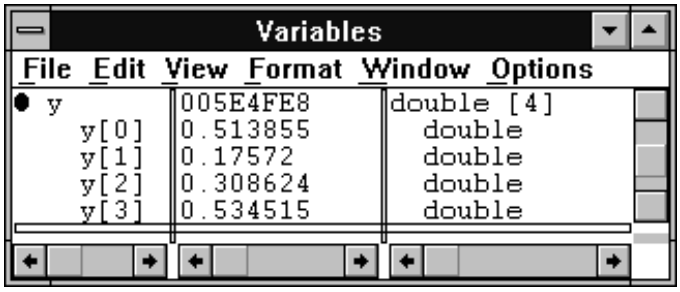

Figure 6-11. An Expanded Array in the Variable Display Window

The **Expand Variable** command expands a currently closed aggregate variable so you can see its contents. Clicking on the ▼ icon has the same effect as selecting **Expand Variable**.

Close Variable

See **Expand Variable** for a discussion of expanded and closed variables.

The **Close Variable** command closes the currently expanded aggregate variable so you can see its name and starting address. Clicking on the ● icon has the same effect as selecting **Close Variable**.

Follow Pointer Chain

Use **Follow Pointer Chain** to examine complex pointer linked structures such as linked lists and trees. If a pointer is a member of a structure and points to a structure of the same type, **Follow Pointer Chain** replaces the current structure with the child structure that the pointer references. For example, in Figure 6-12, `hquework->begin->next` is a member of the structure `hquework->begin` and points to another structure type of `Item`.

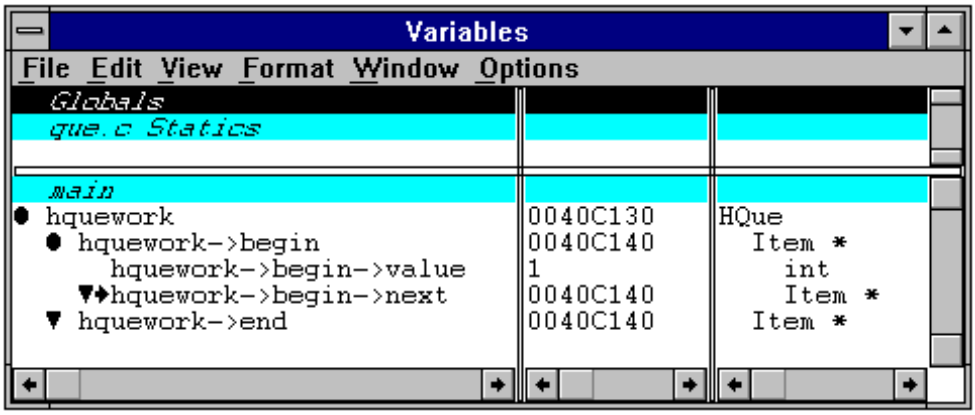

Figure 6-12. A Parent Structure Pointer in a Chain

Clicking on the ➡ icon or selecting **Follow Pointer Chain** replaces the current structure with the child structure that the pointer references, as shown in Figure 6-13.

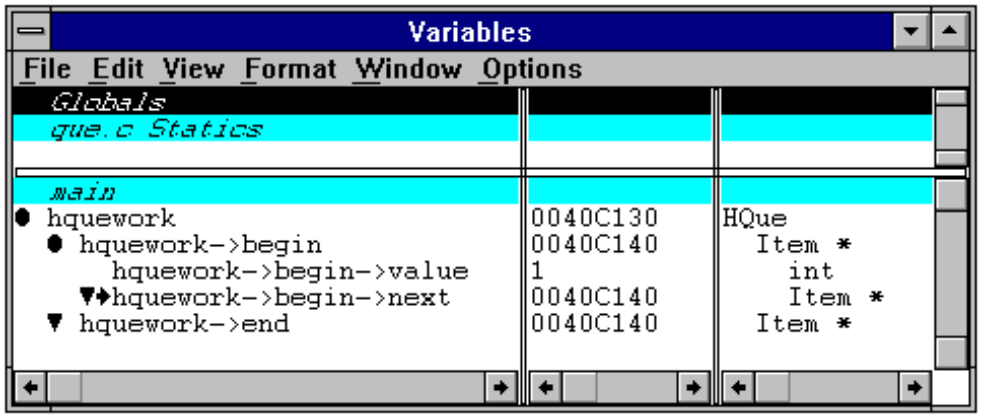

Figure 6-13. A Child Structure Pointer in a Chain

Retrace Pointer Chain

Retrace Pointer Chain replaces the current structure with its parent. Notice the presence of the ⬅ icon in Figure 6-13 after selecting **Follow Pointer Chain**. This indicates that the structure `hquework->begin->next` is a child structure in a chain. Clicking on the ⬅ icon or selecting **Retrace Pointer Chain** causes the variable display to revert back to Figure 6-12.

Note: Retrace Pointer Chain *is only valid when the current structure was obtained with* Follow Pointer Chain.

Go To Execution Position

The **Go To Execution Position** command is available only when the currently highlighted item is a function name or the name of a formal parameter or local variable. The command brings up the Source window containing the call to the function in which execution is suspended, and highlights the function call. This command is valid only in the Variable Display.

You can also execute the **Go To Execution Position** by double-clicking on the function name in the Variable Display window.

Go To Definition

The **Go To Definition** command brings up the Source window containing the definition of the currently selected function or variable, and highlights the definition. This command is valid only in the Variable Display.

Array Display

The **Array Display** command invokes the Array Display window for the currently highlighted array. You can also double-click on an array to invoke the Array Display window. See Chapter 7, *The Array and String Display Windows* for more information.

String Display

The **String Display** command invokes the String Display window for the currently highlighted string. To invoke the String Display window, you can either double click on a string or use the key strokes <Shift-F4>. See Chapter 7, *The Array and String Display Windows* for more information.

Format Menu

This section contains a detailed description of the **Format** menu for the Variable Display and Watch windows. The **Format** menu is shown in Figure 6-14.

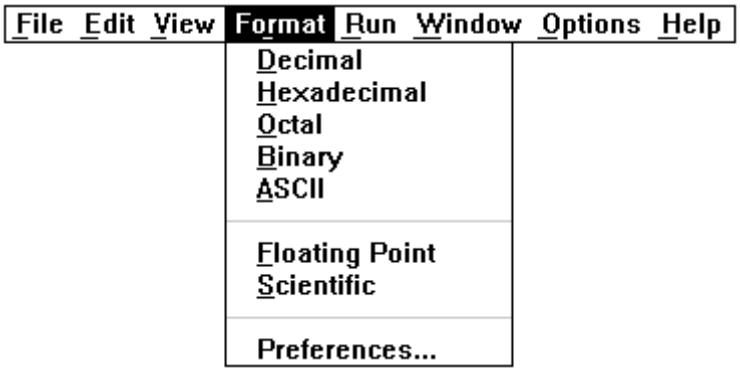

Figure 6-14. The Format Menu

You can set the format used to display numbers in the Variable Display window with the commands in the **Format** menu. The first five items in the menu specify the available formats for displaying individual integers in the Variable Display window. You can display integers in decimal, hexadecimal, octal, binary, or ASCII format. The next two items in the **Format** menu specify the formats available for displaying individual real numbers. Real numbers are displayed in either floating-point or scientific notation. The last item, **Preferences**, sets the default formats for all integers and all real numbers.

The Run Menu

The **Run** menu contains the following subset of the commands that appear in the **Run** menu of the Source window:

Run Project
Continue
Step Over
Step Into
Finish Function
Terminate Execution
Break at First Statement
Breakpoints

Refer to Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows, The Run Menu* for descriptions of each of these commands.

Window Menu

The **Window** menu in the Variable Display and Watch windows operate the same as in the Project window. Refer to *The Window Menu* section in *Chapter 3, Project Window*, for command descriptions.

Options Menu

This section contains a detailed description of the **Options** menu for the Variable Display and Watch windows.

The **Options** menu is shown in Figure 6-15.

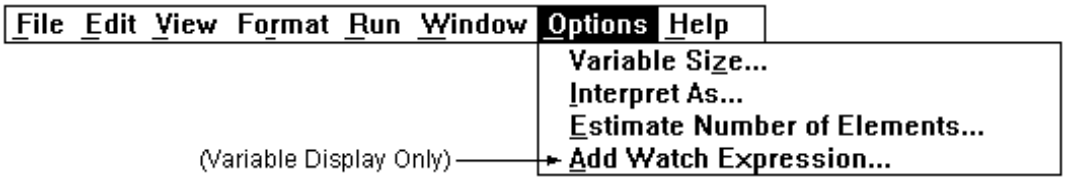

Figure 6-15. The Options Menu

To use these commands, select a particular variable by clicking on it with the mouse or using the UP and DOWN arrow keys. Then access the command from the **Options** menu.

Variable Size...

The **Variable Size** command displays the number of bytes consumed by the variable. If you declare the variable as a buffer, the variable size is considered to be the total size of the buffer. If you declare the variable as a pointer, the **Variable Size** displays the number of bytes consumed by the pointer itself as well as the number of bytes that the individual pointer references. For example, if your code contains the following declaration:

```
static double y_array [4];
```

Variable Size displays a variable size of 32 bytes for `y_array`.

Assume your code defines `dblPtr` as follows:

```
static double *dblPtr;
dblPtr = malloc (2 * sizeof(double));
```

Variable Size displays a variable size of 4 bytes for `dblPtr` pointing to 16 bytes (2 elements).

Interpret As...

Interpret As displays a variable as if it were another type. Selecting a type from the Available Types dialog box displays the variable as the new type.

If **Interpret As** does not offer the exact type you want, you can use a watch expression.

Estimate Number of Elements...

The Variable Display normally cannot expand variables for which LabWindows/CVI does not have user protection information. You can use this command to estimate the number of elements for a variable that you currently cannot expand in the Variable Display. Once you have estimated the number of elements for the variable, you will be able to view the elements in the Variable Display. See the *Limitations of User Protection* section of Chapter 1, *LabWindows/CVI Compiler*, *LabWindows/CVI Programmer Reference Manual* for more information about variable types that do not have user protection.

Add Watch Expression...

Add Watch Expression invokes the Get Watch Expression dialog box. This dialog box is explained in *The Watch Window* section earlier in this chapter.

Chapter 7
Array and String Display Windows

This chapter describes the Array and String Display windows. Use these windows to inspect and modify the contents of a single array or string during a breakpoint.

Note: ***When strings are displayed in ASCII format, there is no visual distinction between a space (ASCII 32) and a NUL byte (ASCII 0). You can see the difference by displaying the string in decimal format.***

Array Display Window

You can use the Array Display window to view and edit the contents of an array or string.

From the Variable Display window, use the **View » Array Display** command to invoke the Array Display window for the currently highlighted array. You can also double-click on an array to invoke the Array Display window.

Use the **Variable Value** command in the **Run** menu of a Source window or the **Code** menu of a Function Panel window to invoke the Array Display window for the currently highlighted array.

The Array Display window for a single-dimensional array is shown in Figure 7-1.

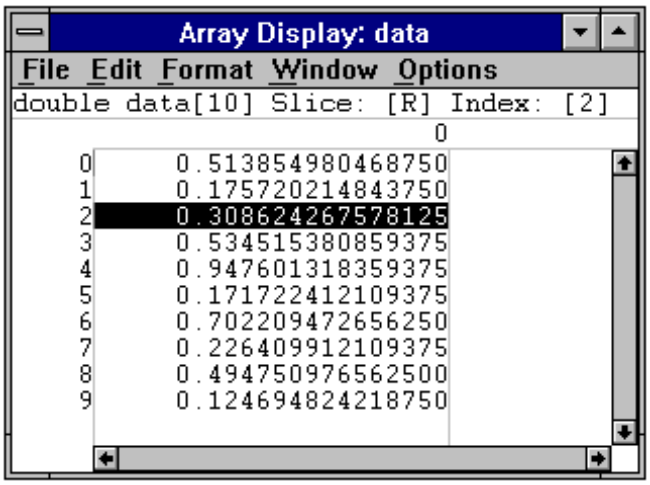

Figure 7-1. The Array Display for a Double-Precision Array

Slice indicates the dimension that is displayed. You can display a single-dimensional array by row [R] or column [C] using the **Options » Reset Indices** command.

Index indicates the element that is currently selected.

Multi-Dimensional Arrays

For an array with two or more dimensions, you can specify two dimensions as the rows and columns of the display. You can also specify constant values used to fix the other dimensions. Use the **Options » Reset Indices** command to specify which plane of the array to display. Figure 7-2 shows the Array Display for a three-dimensional array.

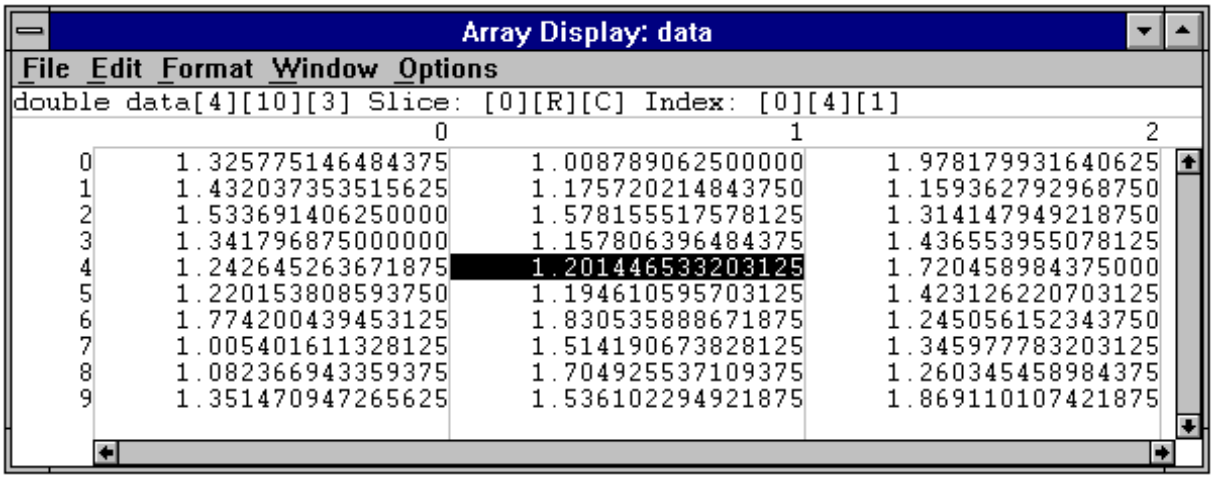

Figure 7-2. The Array Display for a Three-Dimensional Array

The Array Display window shows a 2-dimensional view. By default, the next-to-last dimension is displayed as rows, the last dimension is displayed as columns, and the indices of the other dimensions are held constant at 0. Use the **Reset Indices** command from the **Options** menu to specify the dimensions you want to display as rows and columns, and set the other dimensions to constant values. When you select **Reset Indices** for a three-dimensional array, the Reset Indices dialog box appears, as shown in Figure 7-3.

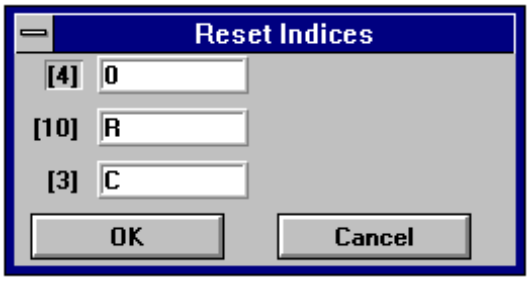

Figure 7-3. The Reset Indices Dialog Box for a 3-Dimensional Array

The dialog box shows the size and display index for each array dimension. The letter *R* indicates the dimension displayed as rows, and the letter *C* indicates the dimension displayed as columns. The indices for the remaining dimensions, those dimensions not specified as either row or column, are held constant at the specified value. For the three-dimensional array shown in Figure 7-2, there is one remaining dimension.

If you enter an invalid character, such as a non-alphanumeric character, or any alphabetic character besides *R*, *r*, *C*, or *c*, an error message appears. Likewise, if you enter an index out of the range of a dimension, an error message appears. Press <Enter> to remove the error message. If you want to close the Reset Indices dialog box without changing the indices, select **Cancel**.

String Display Window

You can use the String Display window to view and edit the contents of a string variable or string array.

From the Variable Display window, use the **String Display** command in the **View** menu to invoke the String Display window for the currently highlighted string. Double-click on a string to invoke the String Display window.

From a Source or Function Panel window, use the **Variable Value** command in the **Run** or **Code** menu to invoke the String Display window for the currently highlighted string.

The String Display for a string variable is shown in Figure 7-4.

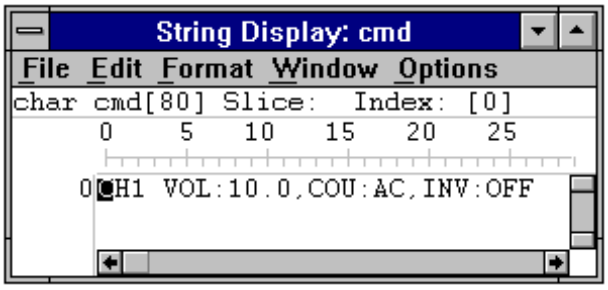

Figure 7-4. The String Display for a String Variable

Multi-Dimensional String Array

Use the **Reset Indices** command to specify which index of a multi-dimensional string array to use as rows in the String Display window. **Reset Indices** is disabled if you are viewing a single string variable. For a string array of two or more dimensions, you can specify which index to use for the rows of the display. The other dimensions are held constant at indices that you specify. When you select **Reset Indices**, the Reset Indices dialog box appears.

The dialog box shows the size and display index for each array dimension. The letter *R* indicates the dimension displayed as rows. The indices for the remaining dimensions are held constant at the specified values.

If you enter an invalid character, or any alphabetic character besides *R* or *r*, or an invalid index, a dialog box appears to indicate the error.

File Menu

This section contains a detailed description of the **File** menu for the Array and String Display windows.

The **File** menu is shown in Figure 7-5.

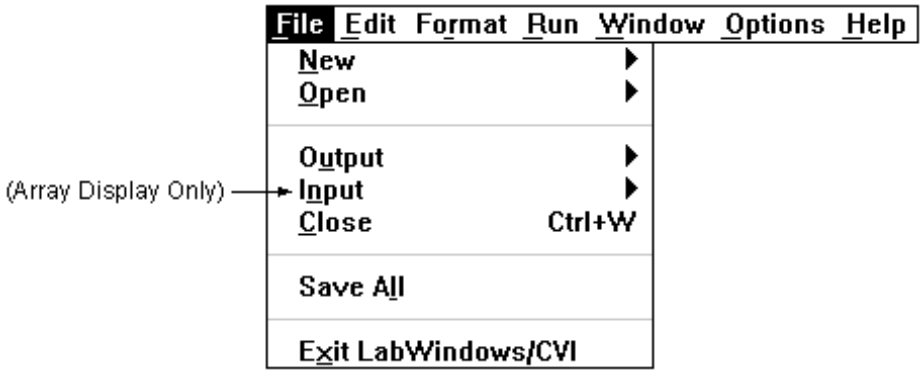

Figure 7-5. The File Menu

New

The **New** command operates the same as in the Project window. For a description of this command, see the discussion of the **New** command in the *File Menu* section of *Chapter 3, Project Window*.

Open

The **Open** command operates the same as in the Project window. For a description of this command, see the discussion of the **Open** command in the *File Menu* section of *Chapter 3, Project Window*.

Output

The **Output** command writes the contents of the window to an ASCII or binary data file on disk. When you select **Output**, a dialog box appears to prompt you to specify the name of the file.

Input (Array Display Only)

Use the **Input** command to select an ASCII or binary data file on disk to replace the currently viewed array in memory.

Close

The **Close** command closes the window.

Save All

The **Save All** command saves all open files to disk.

Exit LabWindows/CVI

The **Exit LabWindows/CVI** command closes the current LabWindows/CVI session. If any open files have been modified since the last save, or if any windows contain unnamed files, you are prompted to save them to disk.

Edit Menu for the Array Display Window

The **Edit** menu for the Array Display window is shown in Figure 7-6.

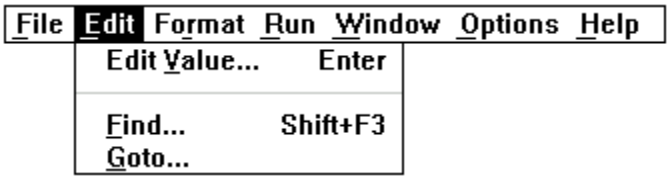

Figure 7-6. The Edit Menu for the Array Display Window

Edit Value...

The **Edit Value** command in the Array Display window invokes a dialog box that you can use to change the value of the selected array element.

Find...

The **Find** command invokes the dialog box in Figure 7-7.

Figure 7-7. The Find Dialog Box in the Array and String Display Windows

- The **Case Sensitive** option finds only the instances of the specified text that match exactly. For example, if CHR is the specified text, the **Case Sensitive** option finds CHR but not Chr.
- The **Whole Word** option finds the specified text only when it is surrounded by spaces, punctuation marks, or other characters not considered parts of a word. The characters *A* through *Z*, *a* through *z*, *0* through *9*, and underscore (_) are considered parts of a word.
- Use the **Wrap** option to continue searching from the beginning of the file once the end of the file has been reached.
- If you select the **Regular Expression** option, certain characters in the **Find What** box are treated as regular expression characters instead of literal characters. The regular expression characters are described in Table 4-1 of Chapter 4, *Source, Interactive Execution, and Standard Input/Output Windows*.
- The **Button Bar** option is used to enable or disable the built-in dialog box for interactive searching as shown in Figure 7-8.

Figure 7-8. The Find Button Bar

Find Prev and **Find Next** search for the closest previous or next occurrence of the specified text. **Stop** terminates the search, leaving the keyboard cursor at the current position. **Return**

terminates the search, leaving the keyboard cursor at the original position where the search was initiated.

The search hot-keys remain active even if the Button Bar is disabled. Use the **Keyboard Help** command in the **Options** menu of a Source window for a list of the search hot-keys.

Goto...

The **Goto** command moves the highlight to a particular location in the current string or array plane. When you execute the **Goto** command, a dialog box appears where you can enter the row and column number of the desired location. For a single string, only the column is specified.

Edit Menu for the String Display Window

The **Edit** menu for the String Display window is shown in Figure 7-9.

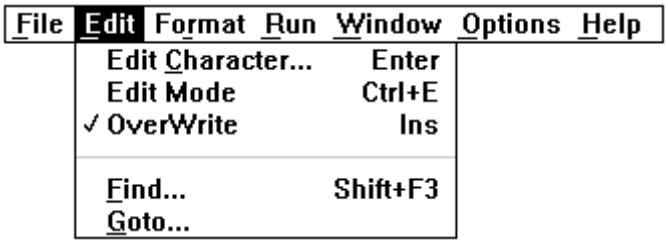

Figure 7-9. The Edit Menu for the String Display Window

Edit Character...

Use the **Edit Character** command in the String Display window to change one character at a time.

Edit Mode

The **Edit Mode** command places the String Display window in edit mode so you can directly edit the string from the keyboard. This mode is valid only when the ASCII display format is selected from the **Format** menu. Alternatively, you can edit one character at a time using the **Options » Edit Character** command.

Overwrite

When the **Edit Mode** command is activated in the String Display window, use the **Overwrite** command to toggle between the overwrite and insert modes of editing.

Find...

The **Find** command operates the same as in the Array Display window.

Goto...

The **Goto** command operates the same as in the Array Display window.

Format Menu

This section contains a detailed description of the **Format** menu for the Array and String Display windows.

The **Format** menu is shown in Figure 7-10.

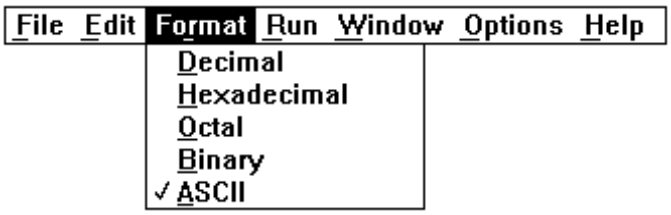

Figure 7-10. The Format Menu

However, if a real array is displayed in the Array Display window, the **Format** menu appears as shown in Figure 7-11.

Figure 7-11. The Format Menu for a Real Array in the Array Display Window

You can set the format used to display numbers in the Array and String Display windows with the commands in the **Format** menu. You can display integers in decimal, hexadecimal, octal, binary, or ASCII format. Real arrays can be displayed in either floating-point or scientific notation.

Run Menu

The **Run** menu contains a subset of the commands that appear in the **Run** menu of the Source window.

Run Project
Continue
Step Over
Step Into
Finish Function
Terminate Execution
Break at First Statement
Breakpoints

Window Menu

The **Window** menu in the Array and String Display windows operates the same as in the Project window. Refer to *The Window Menu* section in *Chapter 3, Project Window*, for command descriptions.

Options Menu

This section contains a detailed description of the **Options** menu for the Array and String Display windows.

The **Options** menu is shown in Figure 7-12.

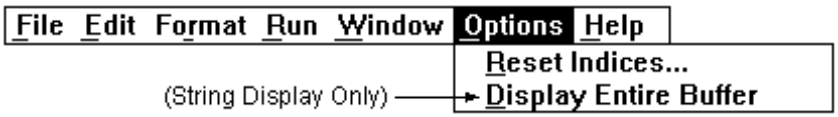

Figure 7-12. The Options Menu

Reset Indices...

Use **Reset Indices** in the Array Display Window to set which array dimension is displayed as rows and which array dimension is displayed as columns.

Use **Reset Indices** in the String Display window to set which string dimension is displayed as rows.

Display Entire Buffer (*String Display Only*)

By default, the String Display window displays only the characters preceding the first ASCII `NUL`. To see characters beyond the `NUL`, select **Options » Display Entire Buffer**.

Appendix A
Source Window Keyboard Commands

The following figure can help you quickly identify common Source window keyboard commands that are not in the menus.

Find again (up)	Ctrl+F3
Find again (down)	F3
Use selected text as search string	Ctrl+Shift+F3
Replace selected text	Ctrl+F11
Replace selected text and find again	F11
Use selected text as replace string	Ctrl+Shift+F11
Windowing:	
Next window	Ctrl+F6
Previous window	Ctrl+Shift+F6
Switch sub-windows	F6
Editing:	
Change text selection mode	Ctrl+Ins
Toggle insert/overwrite mode	Ins
Delete to end of line	Ctrl+D
Backspace to beginning of word	Ctrl+Shift+BkSp
Cut line to clipboard	Ctrl+Y
Insert a new line above	Shift+Enter
Insert a new line below	Ctrl+Enter
Select text	Shift+<cursor movement key>
Remove text selection	Esc
Cursor Movement:	
Up 1 line	Up
Down 1 line	Down
Left 1 column	Left
Right 1 column	Right
Scroll up 1 line	Ctrl+Up
Scroll down 1 line	Ctrl+Down
Left 1 word	Ctrl+Left
Right 1 word	Ctrl+Right
Top of window	Ctrl+PgUp
Bottom of window	Ctrl+PgDown
Beginning of line	Home
End of line	End
Move up 1 page	PgUp
Move down 1 page	PgDown
Top of file	Ctrl+Home
Bottom of file	Ctrl+End

Figure A-1. Keyboard Commands

Appendix B
Customer Communication

For your convenience, this appendix contains forms to help you gather the information necessary to help us solve technical problems you might have as well as a form you can use to comment on the product documentation. Filling out a copy of the *Technical Support Form* before contacting National Instruments helps us help you better and faster.

National Instruments provides comprehensive technical assistance around the world. In the U.S. and Canada, applications engineers are available Monday through Friday from 8:00 a.m. to 6:00 p.m. (central time). In other countries, contact the nearest branch office. You may fax questions to us at any time.

Electronic Services

Bulletin Board Support

National Instruments has BBS and FTP sites dedicated for 24-hour support with a collection of files and documents to answer most common customer questions. From these sites, you can also download the latest instrument drivers, updates, and example programs. For recorded instructions on how to use the bulletin board and FTP services and for BBS automated information, call (512) 795-6990. You can access these services at:

- United States: (512) 794-5422 or (800) 327-3077
  Up to 14,400 baud, 8 data bits, 1 stop bit, no parity
- United Kingdom: 01635 551422
  Up to 9,600 baud, 8 data bits, 1 stop bit, no parity
- France: 1 48 65 15 59
  Up to 9,600 baud, 8 data bits, 1 stop bit, no parity

FaxBack Support

FaxBack is a 24-hour information retrieval system containing a library of documents on a wide range of technical information. You can access FaxBack from a touch-tone telephone at the following number: (512) 418-1111.

FTP Support

To access our FTP site, log on to our Internet host, `ftp.natinst.com`, as anonymous and use your Internet address, such as `joesmith@anywhere.com`, as your password. The support files and documents are located in the /`support` directories.

E-Mail Support (currently U.S. only)

You can submit technical support questions to the appropriate applications engineering team through e-mail at the Internet addresses listed below. Remember to include your name, address, and phone number so we can contact you with solutions and suggestions.

GPIB:	`gpib.support@natinst.com`
DAQ:	`daq.support@natinst.com`
VXI:	`vxi.support@natinst.com`
LabVIEW:	`lv.support@natinst.com`
LabWindows:	`lw.support@natinst.com`
HiQ:	`hiq.support@natinst.com`
Lookout:	`lookout.support@natinst.com`
VISA:	`visa.support@natinst.com`

Fax and Telephone Support

National Instruments has branch offices all over the world. Use the list below to find the technical support number for your country. If there is no National Instruments office in your country, contact the source from which you purchased your software to obtain support.

	Telephone	Fax
Australia	03 9 879 9422	03 9 879 9179
Austria	0662 45 79 90 0	0662 45 79 90 19
Belgium	02 757 00 20	02 757 03 11
Canada (Ontario)	519 622 9310	
Canada (Quebec)	514 694 8521	514 694 4399
Denmark	45 76 26 00	45 76 26 02
Finland	90 527 2321	90 502 2930
France	1 48 14 24 24	1 48 14 24 14
Germany	089 741 31 30	089 714 60 35
Hong Kong	2645 3186	2686 8505
Italy	02 413091	02 41309215
Japan	03 5472 2970	03 5472 2977
Korea	02 596 7456	02 596 7455
Mexico	95 800 010 0793	5 520 3282
Netherlands	0348 433466	0348 430673
Norway	32 84 84 00	32 84 86 00
Singapore	2265886	2265887
Spain	91 640 0085	91 640 0533
Sweden	08 730 49 70	08 730 43 70
Switzerland	056 200 51 51	056 200 51 55
Taiwan	02 377 1200	02 737 4644
U.K.	01635 523545	01635 523154

Technical Support Form

Photocopy this form and update it each time you make changes to your software or hardware, and use the completed copy of this form as a reference for your current configuration. Completing this form accurately before contacting National Instruments for technical support helps our applications engineers answer your questions more efficiently.

If you are using any National Instruments hardware or software products related to this problem, include the configuration forms from their user manuals. Include additional pages if necessary.

Name _____

Company _____

Address _____

_____

Fax (_____)_____ Phone (_____)_____

Computer brand _____ Model _____ Processor _____

Operating system: Windows 3.1, Windows for Workgroups 3.11, Windows NT 3.1, Windows NT 3.5, Windows 95, other (include version number)_____

Clock Speed _____MHz RAM _____MB Display adapter _____

Mouse ____yes ____no Other adapters installed _____

Hard disk capacity _____MB Brand _____

Instruments used _____

National Instruments hardware product model _____ Revision _____

Configuration _____

National Instruments software product _____ Version _____

Configuration _____

The problem is _____

_____

_____

_____

List any error messages _____

_____

_____

_____

The following steps will reproduce the problem _____

_____

_____

_____

Hardware and Software Configuration Form

Record the settings and revisions of your hardware and software on the line to the right of each item. Complete a new copy of this form each time you revise your software or hardware configuration, and use this form as a reference for your current configuration. When you complete this form accurately before contacting National Instruments for technical support, our applications engineers can answer your questions more efficiently.

National Instruments Products

Data Acquisition Hardware Revision _____

Interrupt Level of Hardware _____

DMA Channels of Hardware _____

Base I/O Address of Hardware _____

NI-DAQ, LabVIEW, or
LabWindows Version _____

Other Products

Computer Make and Model _____

Microprocessor _____

Clock Frequency _____

Type of Video Board Installed _____

Operating System _____

Operating System Version _____

Operating System Mode _____

Programming Language _____

Programming Language Version _____

Other Boards in System _____

Base I/O Address of Other Boards _____

DMA Channels of Other Boards _____

Interrupt Level of Other Boards _____

Documentation Comment Form

National Instruments encourages you to comment on the documentation supplied with our products. This information helps us provide quality products to meet your needs.

Title: **LabWindows®/CVI User Manual**

Edition Date: **July 1996**

Part Number: **320681C-01**

Please comment on the completeness, clarity, and organization of the manual.

_____

_____

_____

_____

_____

_____

_____

_____

If you find errors in the manual, please record the page numbers and describe the errors.

_____

_____

_____

_____

_____

_____

_____

_____

Thank you for your help.

Name _____

Title _____

Company _____

Address _____

_____

Fax (_____)_____ Phone (_____)_____

Mail to: Technical Publications
National Instruments Corporation
6504 Bridge Point Parkway
Austin, TX 78730-5039

Fax to: Technical Publications
National Instruments Corporation
(512) 794-5678

Glossary

Prefix	Meaning	Value
m-	milli-	10^{-3}
µ-	micro-	10^{-6}
n-	nano-	10^{-9}

A

active window — The window affected by user input at a given moment. The title of an active window is highlighted.

Array Display — A mechanism for viewing and editing numeric arrays.

auto-exclusion — A mechanism that prevents pre-existing lines from executing in the Interactive Execution Window.

B

binary control — A function panel control that resembles a physical on/off switch and can produce one of two values depending upon the position of the switch.

bps — bits per second

breakpoint — An interruption in the execution of a program.

`Breakpoint` — A function that interrupts the execution of a program.

Breakpoint command — A specific command that interrupts the execution of a program.

C

check box	A dialog box item that allows you to toggle between two possible options.
clipboard	A temporary storage area LabWindows/CVI uses to hold text that is cut, copied, or deleted from a work area.
CodeBuilder	The LabWindows/CVI feature that creates code based on a `.uir` file to connect your GUI to the rest of your program. This code can be compiled and run as soon as it is created.
command button	A dialog box item that, when selected, executes a command associated with the dialog box.
common control	A control on a Common Control Function Panel that specifies a parameter in all functions associated with a Function Panel window.
compiler define	A command line argument passed to the compiler that defines an identifier as a macro to the preprocessor.
control	An input and output device that appears on a function panel for specifying function parameters and displaying function results.
cursor	The flashing rectangle that shows where you can enter text on the screen. If you have a mouse installed, there is also a mouse cursor.
cursor location indicator	An element of the LabWindows/CVI screen that specifies the row and column position of the cursor in the window.

D

default command	The action that takes place when ENTER is pressed and no command is specifically selected. Default command buttons are indicated in dialog boxes with an outline.
dialog box	A prompt mechanism in which you specify additional information needed to complete a command.

E

entry mode entry indicator	An element of the LabWindows/CVI screen that indicates the current text mode as either insert or overwrite.
excluded code	Code that is ignored during compilation and execution. Excluded lines of code are displayed in a different color than included lines of code.

F

`.fp` file	A file containing information about the function tree and function panels of an instrument module.
full-screen mode	A screen display mode in which one window occupies the entire screen.
function panel	A screen-oriented user interface to the LabWindows/CVI libraries in which you can interactively execute library functions and generate code for inclusion in a program.
Function Panel Editor window	The window in which you build a function panel. It is described in the *LabWindows Instrument Driver Developer's Guide*.
Function Panel window	The window that contains function panels.
function tree	The hierarchical structure in which the functions in a library or an instrument driver are grouped. The function tree simplifies access to a library or instrument driver by presenting functions organized according to the operation they perform, as opposed to a single linear listing of all available functions.
Function Tree Editor window	The window in which you build the skeleton of a function panel file. It is described in the *LabWindows Instrument Driver Developer's Guide*.

G

Generated Code box	A text box located at the bottom of the function panel window that displays the code produced by the manipulation of function panel controls.
global control	A function panel control that displays the contents of global variables in a library function. Global controls allow you to monitor global variables in a function that are not specifically returned as results by the function. These are read-only controls that cannot be altered by the user, and do not contribute a parameter to the generated code.

H

hex	hexadecimal
highlight	The way in which input focus is displayed on a LabWindows/CVI screen; to move the input focus onto an item.

I

immediate action menu	A menu that has no menu items associated with it and takes effect immediately when selected. An immediate action command is suffixed with an exclamation point (!).
in	inches
input control	A function panel control that accepts a value typed in from the keyboard. An input control can have a default value associated with it. This value appears in the control when the panel is first displayed.
input focus	Displayed on the screen as a highlight on an item, signifying that the item is active. User input affects the item in the dialog box that has the input focus.
instrument driver	A set of high-level functions for controlling an instrument. It encapsulates many low-level operations, such as data formatting and GPIB, RS-232, and VXI communication, into intuitive, high-level functions. An instrument driver can pertain to one particular instrument or to a group of related instruments. An instrument driver consists of a program and a set of function panels. The program contains the code for the high-level functions. Associated with the instrument program is an include file that declares the high-level functions you can call, the global variables you can access, and the defined constants you can use.
Instrument Library	A LabWindows/CVI library that contains instrument control functions.
Interactive Execution window	A LabWindows/CVI work area in which sections of code may be executed without creating an entire program.

L

list box	A dialog box item that displays a list of possible choices.

M

MB	megabytes of memory
menu	An area accessible from a menu bar that displays selectable menu items.

N

new style (function definition)	A function definition in which parameters are declared directly in the parameter list

O

old style (function definition)	A function definition in which parameters are declared outside of the parameter list
output control	A function panel control that displays a value determined by the function you execute. An output control parameter must be a string, an array, or a reference parameter of type integer, long, single-precision, or double-precision.

P

primary control	A function panel control that specifies parameters in the function panel's primary function.
primary function	The function that performs the main task associated with a function panel. A function panel has only one primary function, but can have many secondary functions.
Project window	A window containing a list of files used by your application.
prompt command	A command that requires additional information before it can be executed; a prompt command appears on a pull-down menu suffixed with three ellipses (…).

R

return value control	A function panel control that displays a value returned from a function as a return value rather than as a formal parameter.

ring control	A function panel control that represents a range of values much like the slide control, but displays only a single item in a list, rather than displaying the whole list at once as the slide control does. Each item has a different value associated with it. This value is placed in the function call.

S

s	seconds
scroll bars	Areas along the bottom and right sides of a window that show your relative position in the file. Scroll bars can be used with a mouse to move about in the window.
scrollable text box	A dialog box item that displays text in a scrollable display.
select	To choose the item that the next executed action will affect by moving the input focus (highlight) to a particular item or area.
shortcut key commands	A combination of keystrokes that provide a means of executing a command without accessing a menu in the menu bar.
slide control	A function panel control that resembles a physical slide switch. A slide control is a means for selecting one item from a list of options; it inserts a value in a function call that depends upon the position of the cross-bar on the switch.
slider	The cross-bar on the slide control which determines the value placed in the function call.
Source window	A LabWindows/CVI work area in which programs are edited and executed.
split-screen mode	A screen display mode in which two windows occupy the screen at once; each window occupies one-half of the display.
Standard Input/ Output window	A LabWindows/CVI work area in which textual output to and input from the user take place.
standard libraries	The LabWindows/CVI User Interface, Analysis, Data Formatting and I/O, GPIB, GPIB-488.2, DDE, TCP RS-232, Utility, and C system libraries.
String Display window	A window for viewing and editing string variables and arrays.

T

text box	A dialog box item in which text is entered from the keyboard to complete a command.

U

User Interface Editor window	The window in which you build pull-down menus, dialog boxes, panels, and controls and save them to a User Interface Resource (`.UIR`) file. It is described in the *LabWindows/CVI User Interface Reference Manual*.

V

Variable Display	A window that shows the values of the variables that are currently window defined.

W

Watch window	A window that shows the values of user-selectable variables and expressions that are currently defined.
window	A working area that supports specific tasks related to developing and executing programs.

Index

Numbers/Symbols

CVI macro, 3-48
_CVI_DEBUG macro, 3-48
_CVI_DLL_ macro, 3-49
_CVI_EXE_ macro, 3-49
_CVI_LIB_ macro, 3-49
__DEFALIGN macro, 3-49
__FLAT__ macro, 3-49
_M_IX86 macro, 3-49
_NI_BC macro, 3-49
_NI_i386_ macro, 3-48
_NI_mswin_ macro, 3-48
_NI_mswin16_ macro, 3-48
_NI_mswin32_ macro, 3-48, 3-49
_NI_SC macro, 3-49
_NI_sparc_ macro, 3-48
_NI_unix_ macro, 3-48
_NI_VC macro, 3-49
_NI_WC macro, 3-49
__NT__ macro, 3-49
_WIN32 macro, 3-49
__WIN32__ macro, 3-49
_WINDOWS macro, 3-49

A

About LabWindows/CVI command, Help menu, 4-39
activate configuration option, 1-3
Activate Panels When Resuming option, Run menu, 4-31
Add/Edit Watch Expression dialog box, 6-3 to 6-4
Add File to Project command, File menu, 4-8
Add .FP File to Project command, File menu, 5-7
Add Missing Includes command, Build menu, 4-24
Add Program File to Project command, File menu, 5-8
Add Watch Expression command
    Edit menu, 6-8
    Options menu, 6-3, 6-15
Alphabetize command, Select Function Panel dialog box, 3-36
Always prompt before fixing pathnames option, 3-55
Analysis command, Library menu, 3-38 to 3-39
Analysis Library, 3-38
ANSI C command, Library menu, 3-41
ANSI C Library, 3-41
appFont option, 1-6
applications, creating, 2-5 to 2-6
Array Display command, View menu, 6-12, 7-1
Array Display window
    displayed in View menu, 3-44
    Edit menu, 7-5 to 7-7
    File menu, 7-4 to 7-5
    Format menu, 7-8
    invoking, 7-1
    multi-dimensional arrays
        illustration, 7-2
        Reset Indices dialog box, 7-2 to 7-3
        specifying dimensions, 7-2
    Options menu, 7-9
    purpose and use, 2-4, 7-1
    Run menu, 7-9
    single-dimensional array (figure), 7-1
    Window menu, 7-9
Attach and Edit Source command, Edit Instrument dialog box, 3-35
attribute constants, selecting, 5-12 to 5-13
Auto Save Project command, File menu, 3-6

B

Balance command, Edit menu, 4-12
Beginning/End of Selection command, View menu, 4-19
bin directory (table), 1-4
binary control parameters, specifying, 5-5
Bracket Styles command, Options menu, 4-35
brackets
  finding pairs of, 4-12
  setting location for, 4-35
Break at First Statement command, Run menu
  Project window, 3-28
  Source, Interactive Execution, and Standard Input/Output windows, 4-26, 4-29
Break on Library Errors option, 3-51
Breakpoint function, 4-25
breakpoints. *See also* watch variables/expressions.
  applicable only in source code modules (note), 4-26
  breakpoint state, 4-26
  conditional, 4-27
  Edit Breakpoint dialog box, 4-30 to 4-31
  purpose and use, 4-25 to 4-26
  resuming execution, 4-26
  setting and clearing, 4-26
Breakpoints command, Run menu, 3-28, 4-26, 4-30 to 4-31
Breakpoints dialog box. *See also* Edit Breakpoint dialog box.
  Add/Edit Item button, 4-30
  Delete Item button, 4-31
  Disable All button, 4-31
  Enable All button, 4-31
  Go to Line button, 4-31
  illustration, 4-30
  OK button, 4-31
Bring Standard Input/Output window to front option, 3-52
Build Error window, 3-48
build errors
  Build Errors in Next File command, 4-24
  Next Build Error command, 4-24
  Previous Build Error command, 4-24
Build Errors command, Window menu, 3-43
Build Errors in Next File command, View menu, 4-24
Build menu
  Project window
    Build Project command, 3-11
    Compile File command, 3-11
    Create Distribution Kit command, 3-21 to 3-27
    Create Dynamic Link Library command, 3-15 to 3-18
    Create Standalone Executable command, 3-14 to 3-15
    Create Static Library command, 3-18 to 3-19
    External Compiler Support command, 3-19 to 3-20
    illustration, 3-10
    Instrument Driver Support Only command, 3-12 to 3-13
    Link Project command, 3-11
    Mark All for Compilation command, 3-12
    Mark File for Compilation command, 3-11
    Target command, 3-12
    Update Program Files from Disk command, 3-11
  Source, Interactive Execution, and Standard Input/Output windows
    Add Missing Includes command, 4-24
    Build Project command, 4-23
    Clear Interactive Declarations command, 4-4, 4-23 to 4-24
    Compile File command, 4-23
    Generate Prototypes command, 4-24
    illustration, 4-22
    Insert Include Statements command, 4-24
    Link Project command, 4-23

Mark File for Compilation command, 4-23
Next Build Error command, 4-24
Previous Build Error command, 4-24
Build Project command, Build menu
Project window, 3-11
Source, Interactive Execution, and Standard Input/Output windows, 4-23
bulletin board support, B-1
Button Bar option, Find command
Array and String Display windows, 7-6 to 7-7
Source, Interactive Execution, and Standard Input/Output windows, 4-15 to 4-16
Variable Display window, 6-7
buttons, adding and positioning on toolbar, 4-2

C

Cascade Windows command, Window menu, 3-42
Case Sensitive option, Find command
Array and String Display windows, 7-6
Source, Interactive Execution, and Standard Input/Output windows, 4-13
Variable Display window, 6-7
CatchProtectionFaults option, 1-6
cfgdir configuration option, 1-3 to 1-4
Change Format command, Options menu, 5-21
Character Select mode, 4-6
Check Disk Dates Before Each Run option, 3-51
child structure, 6-2
child structure pointer in chain (figure), 6-11
Clear Interactive Declarations command
Build menu, 4-4, 4-23 to 4-24
Code menu, 5-10
Clear Tags command, View menu, 4-19
Clear Window command, Edit menu, 4-4, 4-11
Close All command, Window menu, 3-43
Close command, File menu
Array and String Display windows, 7-5
Function Panel windows, 5-7
Close Libraries command
Code menu, 5-10
Run menu, 4-29
Close Variable command, View menu, 6-2, 6-10
Closed Array, Variable Display window (figure), 6-10
code. *See* source files.
Code menu, Function Panel windows
Clear Interactive Declarations command, 5-10
Close Libraries command, 5-10
Declare Variable command, 5-4, 5-6, 5-9 to 5-10
Expression Value command, 5-6, 5-17
illustration, 5-8
Insert Function Call command, 5-16 to 5-17
Run Function Panel command, 5-8 to 5-9
Select UI Constant command, 5-10 to 5-14
Select Variable command, 5-14 to 5-16
Set Target File command, 5-17
Variable Value command, 5-6, 5-17, 6-1, 7-3
code modules
adding to projects, 3-8
listing in Project window, 2-5
color coding tokens in source and include files, 4-36
Colors command, Options menu
Project window, 3-55 to 3-56
Source, Interactive Execution, and Standard Input/Output windows, 4-36
Column Select mode, 4-6
common control function panel, 5-6
comparing source files. *See* Diff command, Edit menu.
compile errors, maximum number of, 3-46

Compile File command, Build menu
Project window, 3-11
Source, Interactive Execution, and Standard Input/Output windows, 4-23
compiled files, 2-5
compiler defines
predefined macros, 3-48 to 3-49
syntax, 3-48
compiler options
Display status dialog during build, 3-48
Enable signed/unsigned pointer mismatch warning, 3-47
Enable unreachable code warning, 3-47
Maximum number of compile errors, 3-46
Prompt for include file paths, 3-47
Require function prototypes, 3-46 to 3-47, 4-4
Require return values for non-void functions, 3-47
Show Build Error window for warnings, 3-48
Stop on first file with errors, 3-48
Compiler Options command, Options menu, 3-46
compiler support, external. *See* External Compiler Support dialog box.
compiling files. *See* Build menu.
conditional breakpoints, 4-27
configuration options
cfgdir, 1-3 to 1-4
cvidir, 1-4
resdir, 1-4
tmpdir, 1-5
UNIX, 1-3
Windows 3.1, 1-2
Windows 95/NT, 1-2
constants, user interface. *See* user interface constants, selecting.
Contents command, Help menu, 4-39
context menus, Source window, 4-43
Continue command, Run menu
Project window, 3-28
Source, Interactive Execution, and Standard Input/Output windows, 4-28
Control command, Help menu, 5-22
<Control> key (SPARCstation), 3-52
Copy command, Edit menu, 4-10
Create Distribution Kit command, Build menu, 3-21 to 3-27
Create Distribution Kit dialog box, 3-22 to 3-27
Build Information section, 3-22 to 3-23
File Groups section, 3-23 to 3-25
illustration, 3-22
Installation Script File section, 3-26 to 3-27
Main section, 3-25
Create Dynamic Link Library command, Build menu, 3-15
Create Dynamic Link Library dialog box, 3-16 to 3-18
Cancel button, 3-18
DLL file field, 3-16
Exports
Change button, 3-17
Export What field, 3-17
Include File Symbols option, 3-17
Symbols Marked for Export option, 3-17
Import Library Base Name field, 3-16
Import Library Choices button, 3-16 to 3-17
OK button, 3-18
Prompt before overwriting file checkbox, 3-16
Type Library button, 3-17
Using LoadExternalModule
Add Files to DLL button, 3-17
Help button, 3-17
Create Object File command, Options menu, 4-38 to 4-39
Create Standalone Executable command, Build menu, 3-14 to 3-15
Create Standalone Executable dialog box, 3-14 to 3-15
Application Executable File field, 3-14
Application Icon File field, 3-14
Application Title field, 3-14
Cancel button, 3-15
Icon control, 3-14
OK button, 3-15

Prompt before overriding executable file checkbox, 3-15
Using LoadExternalModule item, 3-15
Version Info button, 3-15
Create Static Library command, Build menu, 3-18 to 3-19
Create Static Library dialog box, 3-18 to 3-19
Cancel button, 3-19
Library File field, 3-18
Library Generatin Choices button, 3-18
OK button, 3-19
Prompt before overwriting file checkbox, 3-18
creating
applications, 2-5 to 2-6
object files, 4-38 to 4-39
standalone executables. *See* standalone executables, creating and distributing.
user interface, 2-6
Windows DLLs. *See* Microsoft Windows DLLs.
curly braces
finding pairs of, 4-12
setting location for, 4-35
Current Tree command, View menu, 5-18
customer communication, *xxii*, B-1 to B-2
customizing
bracket styles, 4-35
colors, 3-55 to 3-56
fonts, 1-5 to 1-6, 3-55, 4-35
toolbars, 4-2 to 4-3
Cut command, Edit menu, 4-10
CVI macro, 3-48
_CVI_DEBUG macro, 3-48
_CVI_DLL_ macro, 3-49
_CVI_EXE_ macro, 3-49
_CVI_LIB_ macro, 3-49
cvidir configuration option, 1-4

D

Data Acquisition command, Library menu, 3-39
Data Acquisition Library
definition, 3-39
purpose and use, 2-3
data type compatibility for function panel variables, 5-15 to 5-16
dates
Check Disk Dates Before Each Run option, 3-51
displaying files in chronological order, 3-9
displaying for project list files, 3-9
DDE command, Library menu, 3-40
DDE Library, 3-40
debug options
CatchProtectionFaults, 1-6
DisplayCVIDebugVxDMissingMessage, 1-6
LoadCVIDebugVxD, 1-7
debugging levels
Extended, 3-51
None, 3-51
Standard, 3-51
Declare Variable command, Code menu, 5-9 to 5-10
specifying input control parameter, 5-4
specifying output control parameter, 5-6
Declare Variable dialog box, 5-9 to 5-10
Add declaration to current block in target file checkbox, 5-9
Add declaration to top of target file checkbox, 5-9
Cancel button, 5-10
Execute declaration checkbox, 5-9
illustration, 5-9
Number of Elements box, 5-9
OK button, 5-10
Variable Name text box, 5-9
Variable Type message, 5-9
__DEFALIGN macro, 3-49
Default All command, Options menu, 5-21
Default Control command, Options menu, 5-20
Delete command, Edit menu, 4-11

Delete Watch Point command, Edit menu, 6-9
Detach Program command, Edit Instrument dialog box, 3-36
dialogFont option, 1-6
DialogFontBold option, 1-5
DialogFontName option, 1-5
DialogFontSize option, 1-5
Diff command, Edit menu, 4-12 to 4-13
    Diff With, 4-12
    Find Next Difference, 4-12
    Ignore White Space, 4-12
    Match Criteria, 4-12
    Recompare Selections Ignoring White Space, 4-13
    Synchronize at Top, 4-12
    Synchronize Selections, 4-12
Display Entire Buffer command, Options menu, 7-9
Display status dialog during build option, 3-48
DisplayCVIDebugVxDMissingMessage option, 1-6
distributing standalone executables. *See* standalone executables, creating and distributing.
Distribution Kit. *See* Create Distribution Kit dialog box.
DLLs. *See* Microsoft Windows DLLs.
documentation
    conventions used in manual, *xx-xxi*
    LabWindows/CVI documentation set, *xxi-xxii*
    organization of manual, *xix-xx*
Down Call Stack command, Run menu, 4-32
Dynamic Memory command, Run menu, 4-32
dynamic-link library files, required in project file list, 3-1

E

Easy I/O for DAQ command, Library menu, 3-39
Easy I/O for DAQ Library, 3-39
Edit Breakpoint dialog box, 4-30 to 4-31
Edit Character command, Edit menu, 7-7
Edit command, Instrument menu, 3-34. *See also* Edit Instrument dialog box.
Edit Function Panel Window command, 5-21
Edit Function Tree command, Edit Instrument dialog box, 3-36
Edit Instrument dialog box
    Attach and Edit Source command, 3-35
    Detach Program command, 3-36
    Edit Function Tree command, 3-36
    illustration, 3-35
    Reattach Program command, 3-36
    Show Info command, 3-35
Edit menu
    Array Display window
        Edit Value command, 7-5
        Find command, 7-6 to 7-7
        Goto command, 7-7
        illustration, 7-5
    Project window, 3-7 to 3-9
        Add Files to Project command, 3-7 to 3-8
        Exclude File from Build command, 3-8
        illustration, 3-7
        Include File in Build command, 3-8
        Move Item Down command, 3-9
        Move Item Up command, 3-8
        Remove File command, 3-8
    Source, Interactive Execution, and Standard Input/Output windows
        Balance command, 4-12
        Clear Window command, 4-4, 4-11
        Copy command, 4-10
        Cut command, 4-10
        Delete command, 4-11
        Diff command, 4-12 to 4-13
        disabled commands (note), 4-10
        Find command, 4-13 to 4-16

Go To Definition command, 4-13
illustration, 4-9
Insert Construct command, 4-11 to 4-12
Next File command, 4-17
Paste command, 4-10
Redo command, 4-10
Replace command, 4-16 to 4-17
Resolve All Excluded Lines command, 4-11
Select All command, 4-11
Toggle Exclusion command, 4-4, 4-11
Undo command, 4-10
String Display window
Edit Character command, 7-7
Edit Mode command, 7-7
Find command, 7-8
Goto command, 7-8
Overwrite command, 7-7
Variable Display window
Edit Value command, 6-6
Find command, 6-6 to 6-7
illustration, 6-6
Next Scope command, 6-8
Previous Scope command, 6-8
Watch window
Add Watch Expression command, 6-8
Delete Watch Point command, 6-9
Edit Value command, 6-8
Edit Watch Expression command, 6-8
Find command, 6-9
illustration, 6-8
Edit Mode command, Edit menu, 7-7
Edit Value command, Edit menu
Array Display window, 7-5
Variable Display window, 6-6
Watch window, 6-8
Edit Watch Expression command, Edit menu, 6-8
Editor Preferences command, Options menu
dialog box, 4-34 to 4-35
Line Terminator option, 4-35
Paste option, 4-34
Tabs option, 4-35
Undo option, 4-10, 4-34
editorFont option, 1-6
electronic support services, B-1 to B-2
e-mail support, B-2
Enable signed/unsigned pointer mismatch warning option, 3-47
Enable unreachable code warning option, 3-47
End of Selection command, View menu, 4-19
Environment command, Options menu, 3-37, 3-52
environment options
Bring Standard Input/Output window to front whenever modified, 3-52
Maximum number of lines in Standard Input/Output window, 3-52
Sleep policy when not running program, 3-52
Use host's system standard I/O, 3-52
Use only one function panel window, 3-37, 3-52
Warp mouse over dialog boxes, 3-52
Error command, View menu, 5-18
errors
Break on library errors option, 3-51
build errors, 4-24
Display status dialog during build option, 3-48
Maximum number of compile errors option, 3-46
run-time error reporting, 3-28, 4-28
Show Build Error window for warnings option, 3-48
Stop on first file with errors option, 3-48
terminating compilation for file errors, 3-48
Estimate Number of Elements command, Options menu, 6-15
Exclude File from Build command, Edit menu, 3-8
Exclude Function command, Options menu, 5-21
excluding lines of code, 4-4, 4-11

executables, creating and distributing. *See* standalone executables, creating and distributing.
Execute command, Run menu, 3-28
Exit LabWindows/CVI command, File menu
  Array and String Display windows, 7-5
  Function Panel windows, 5-8
  Project window, 3-7
  Source, Interactive Execution, and Standard Input/Output windows, 4-9
  Variable Display and Watch windows, 6-5
Expand Variable command, View menu, 6-2, 6-9 to 6-10
Expanded Array, Variable Display window (figure), 6-10
Expression Value command
  Code menu, 5-6, 5-17
  Run menu, 4-32
expressions. *See* watch variables/expressions.
expressions, regular (table), 4-14 to 4-15
External Compiler Support command, 3-19
External Compiler Support dialog box, 3-19 to 3-21
  ANSI C Library field, 3-20
  CVI Libraries field, 3-20
  illustration, 3-19
  Other Symbols checkmark, 3-21
    Header File field, 3-21
    Object File field, 3-21
  UIR Callbacks Object File option, 3-20
  Using LoadExternalModule to Load Object and Static Library Files optimization, 3-20

F

fax and telephone support, B-2
FaxBack support, B-1
file extensions, displaying project files in order of, 3-10
File menu
  Array and String Display windows
    Close command, 7-5
    Exit LabWindows/CVI command, 7-5
    illustration, 7-4
    Input command, 7-5
    New command, 7-4
    Open command, 7-4
    Output command, 7-5
    Save All command, 7-5
  Function Panel windows
    Add .FP File to Project command, 5-7
    Add Program File to Project command, 5-8
    Close command, 5-7
    Exit LabWindows/CVI command, 5-8
    illustration, 5-7
    New command, 5-7
    Open command, 5-7
    Save All command, 5-7
  Project window, 3-3 to 3-7
    Auto Save Project command, 3-6
    Exit LabWindows/CVI command, 3-7
    illustration, 3-3
    most recently closed files list, 3-6
    New command, 3-4
    Open command, 3-5
    Print command, 3-6
    Save command, 3-5
    Save All command, 3-6
    Save As command, 3-6
  Source, Interactive Execution, and Standard Input/Output windows
    Add File to Project command, 4-8
    Close command, 4-8

Exit LabWindows/CVI command, 4-9
Hide command (note), 4-8
illustration, 4-7
New command, 4-7
Open command, 4-7
Open Quoted Text command, 4-7
Print command, 4-9
Read Only command, 4-8
Save command, 4-8
Save All command, 4-8
Save As command, 4-8
Save Copy As command, 4-8
Variable Display and Watch windows
Exit LabWindows/CVI command, 6-5
Hide command, 6-5
illustration, 6-4
New command, 6-5
Open command, 6-5
Output command, 6-5
Save All command, 6-5
<filename> startup option (table), 1-1
files. *See also* project files.
adding to project list, 3-7 to 3-8
Check Disk Dates Before Each Run option, 3-51
format conversion when loading, 3-34
instrument driver files, 3-29 to 3-30
Find command, Edit menu
Array Display window, 7-6 to 7-7
Source, Interactive Execution, and Standard Input/Output windows, 4-13 to 4-16
String Display window, 7-8
Variable Display window, 6-6 to 6-7
Watch window, 6-9
Find dialog box
Array Display window, 7-6 to 7-7
Source, Interactive Execution, and Standard Input/Output windows, 4-13 to 4-16
Variable Display window, 6-6 to 6-7
Find Function Panel command, View menu
Function Panel windows, 5-18 to 5-19
Source, Interactive Execution, and Standard Input/Output windows, 4-21 to 4-22
Find Next option, Find command
Array Display window, 7-6
Source, Interactive Execution, and Standard Input/Output windows, 4-16
Variable Display and Watch Windows, 6-7
Find Prev option, Find command
Array Display window, 7-6
Source, Interactive Execution, and Standard Input/Output windows, 4-16
Variable Display and Watch Windows, 6-7
Find UI Objects command, View menu, 4-22
Find What text box option, Find command, 4-13
Finish Function command, Run menu, 4-29
First Function Panel Window command, View menu, 5-19
First Panel command, View menu, 5-2
Fixup pathnames when project is moved option, 3-55
__FLAT__ macro, 3-49
Flatten option, Select Function Panel dialog box, 3-36, 5-2
Follow Pointer Chain command, View menu, 6-2, 6-10 to 6-11
Font command, Options menu
Project window, 3-55
Source, Interactive Execution, and Standard Input/Output windows, 4-35
font directory (table), 1-4
font options
appFont, 1-6
dialogFont, 1-6
DialogFontBold, 1-5
DialogFontName, 1-5
DialogFontSize, 1-5
editorFont, 1-6
menuFont, 1-6

format conversion of files during loading, 3-34
Format menu
Array and String Display windows, 7-8
Variable Display and Watch windows, 6-13
Formatting and I/O command, Library menu, 3-40
Formatting and I/O Library, 3-40
.fp files. *See* instrument driver function panel (.fp) files.
FTP support, B-2
function classes, 5-1
Function command, Help menu, 5-22
function panel, recalling. *See* Recall Function Panel command, View menu.
function panel controls
binary control parameter, 5-5
common control panel, 5-6
global control, 5-6
illustration, 5-4
input control parameter, 5-4
numeric control parameter, 5-5
output control parameter, 5-6
overriding with Toggle Control Style command, 5-21
retoring default value, 5-20
return value control parameter, 5-4
ring control parameter, 5-5 to 5-6
slide control parameter, 5-5
types of controls (figure), 5-3
viewing arrays, structures, and variables, 5-6
Function Panel Editor window, 2-5
Function Panel Help Editor window, 2-5
Function Panel History command, View menu
Function Panel windows, 5-18
Source, Interactive Execution, and Standard Input/Output windows, 4-20
Function Panel Tree command, View menu, 4-20
Function Panel windows
Code menu, 5-8 to 5-17
displayed in Window menu, 3-44
File menu, 5-7 to 5-8
Generated Code Box, 5-3
Help menu, 5-22
illustration, 5-2
Instrument menu, 5-20
Library menu, 5-20
multiple function panels per window, 5-2 to 5-3
Options menu, 5-20 to 5-21
purpose and use, 2-4
View menu, 5-2, 5-17 to 5-19
Window menu, 5-20
function panels. *See also* function panel controls.
accessing, 5-1 to 5-2
from Instrument menu, 3-36 to 3-37
definition, 2-3, 5-1
executing in Interactive Execution window, 5-1
finding functions, 4-21 to 4-22
multiple function panels per window, 5-2 to 5-3
selecting, 5-1 to 5-2
function prototypes, enabling, 3-46 to 3-47
Function subwindow, Variable Display window, 6-2
Function Tree/Help Editor window, 2-5
Function Tree command, 3-8
Function Tree Editor window
opening
with New command, 3-4
with Open command, 3-5
purpose and use, 2-4
function trees
definition, 5-1
files displayed in Window menu, 3-44

G

Generate DLL Glue Object command, Options menu, 4-38
Generate DLL Glue Source command, Options menu, 4-38
Generate DLL Import Library command, Options menu, 4-37

Generate DLL Import Source command, Options menu, 4-37
Generate Prototypes command, Build menu, 4-24
Generate Visual Basic Include command, Options menu, 4-38
Generated Code Box, 5-3
global control, 5-6
Global subwindow, Variable Display window, 6-2
glue code. *See* Microsoft Windows DLLs.
glue object, generating, 4-38
Go to Cursor command, Run menu, 4-28
Go To Definition command
  Edit menu, 4-13
  View menu, 6-12
Go To Execution Position command, View menu, 6-12
Goto command, Edit menu
  Array Display window, 7-7
  String Display window, 7-8
GPIB Library, 3-39
GPIB/GPIB 488.2 command, Library menu, 3-39

H

header files
  including, 3-8
  optional in project file list, 3-1
Help dialog box
  for functions and classes, 3-37
  illustration, 3-37
Help Editor windows, displayed in Window menu, 3-44
Help menu
  Function Panel windows
    Control command, 5-22
    Function command, 5-22
    illustration, 5-22
  Source, Interactive Execution, and Standard Input/Output windows, 4-39 to 4-40
  About LabWindows/CVI command, 4-39
  Contents command, 4-39
  Keyboard Help command, 4-39 to 4-40
  Search for Help On command, 4-39
Hide command, File menu
  Interactive Execution and Standard Input/Output windows (note), 4-8
  Variable Display and Watch windows, 6-5
Hide Windows option, 3-51
host Standard I/O, selecting, 3-52
hyperhelp directory (table), 1-4

I

icons
  associated with variables, 6-2
  Project window, 3-2 to 3-3
IEW. *See* Interactive Execution window.
include directory (table), 1-4
Include File command, View menu, 5-18
Include File in Build command, Edit menu, 3-8
include files
  adding missing files, 4-24
  generating for Visual Basic, 4-38
  prompting for path, 3-47
  tracking dependencies, 3-47
Include option, Add Files to Project command, 3-8
Include Paths command, Options menu, 3-49 to 3-50
Input command, File menu, 7-5
input control parameters, specifying, 5-4
Insert Construct command, Edit menu, 4-11 to 4-12
Insert Function Call command, Code menu, 5-16 to 5-17
Insert Include Statements command, Build menu, 4-24
instrsup.dll
  functions from selected libraries, 3-12
  linking project to selected libraries, 3-13

multi-threaded safe functions, 3-13
Standard Input/Output Window not supported, 3-13
Utility Library functions in, 3-13
Instrument Directories command, Options menu, 3-50
instrument driver function panel (.fp) files
adding to project list, 5-7
dummy .fp files for support libraries, 3-41, 3-54
purpose and use, 3-1
Instrument Driver Support Only command, Build menu, 3-12 to 3-13
instrument drivers
definition, 3-29
files for instrument drivers, 3-29 to 3-30
loading/unloading, 3-31 to 3-32
instruments without instrument program, 3-32
precedence rules, 3-31 to 3-32
modifying, 3-33
modules containing non-instrument functions, 3-32
programming, 3-1
VXI*plug&play* instrument driver files, 3-30
Instrument Library, 2-3
Instrument menu
accessing function panels, 5-1
Function Panel windows, 5-20
illustration, 3-29
Project window
accessing function panels, 3-36 to 3-37
Edit command, 3-34
illustration, 3-33
Load command, 3-34
loading user libraries, 3-41
Unload command, 3-34
Source, Interactive Execution, and Standard Input/Output windows, 4-33
Interactive Execution command, Window menu, 3-45
Interactive Execution window, 4-3 to 4-4
Build menu, 4-22 to 4-24
Edit menu, 4-9 to 4-17
excluding lines, 4-11
executing code, 4-4
rules for, 4-4
executing function panels, 5-1
File menu, 4-7 to 4-9
Instrument menu, 4-33
Library menu, 4-33
Options menu, 4-33 to 4-39
purpose and use, 2-4
rules for executing code, 4-4
Run menu, 4-25 to 4-32
selecting text, 4-5 to 4-7
subwindows, 4-5
View menu, 4-17 to 4-22
Window menu, 4-33
Interpret As command, Options menu, 6-15

K

keyboard commands
bypassing Find dialog box, 4-16
bypassing Replace dialog box, 4-16
Source window (figure), A-1
Keyboard Help command, Options menu, 4-39 to 4-40
Keyboard Options command, Options menu, 3-52

L

LabWindows/CVI. *See also* specific windows.
components, 2-1 to 2-5
Data Acquisition Library, 2-3
Instrument Library, 2-3
LabWindows/CVI environment, 2-3 to 2-5
standard libraries, 2-2
User Interface Library, 2-2 to 2-3
VISA Library, 2-3
creating applications, 2-5 to 2-6
environment, 2-3 to 2-5
Last Function Panel Window command, View menu, 5-19

Last Panel command, View menu, 5-2
libraries. *See also* specific library names; standard libraries; user libraries.
files required in project file list, 3-1
selected libraries in instrsup.dll, 3-12
Library menu
accessing function panels, 5-1
Function Panel windows, 5-20
Project window
Analysis command, 3-38 to 3-39
ANSI C command, 3-41
Data Acquisition command, 3-39
DDE command, 3-40
Easy I/O for DAQ command, 3-39
Formatting and I/O command, 3-40
GPIB/GPIB 488.2 command, 3-39
illustration, 3-38
installing user libraries, 3-41
RS-232 command, 3-40
TCP command, 3-40
User Interface command, 3-38
Utility command, 3-40 to 3-41
VISA command, 3-40
VXI command, 3-39
X Property command, 3-40
Source, Interactive Execution, and Standard Input/Output windows, 4-33
Library option, Add Files to Project command, 3-8
Library Options command, Options menu, 3-41, 3-53 to 3-54
Library Options dialog box
illustration, 3-53
National Instrument Libraries, 3-54 to 3-55
User Libraries list, 3-53 to 3-54
Line command, View menu, 4-19
Line Icons command, View menu, 4-18, 4-26
Line Numbers command, View menu, 4-18
Line Select mode, 4-6
Line Terminator option, Editor Preferences command, 4-35
lines of code
excluding, 4-4, 4-11
Maximum number of lines in Standard Input/Output window option, 3-52
Link Project command, Build menu
Project window, 3-11
Source, Interactive Execution, and Standard Input/Output windows, 4-23
Load command, Instrument menu, 3-34
LoadCVIDebugVxD option, 1-7
loading/unloading instrument drivers, 3-31 to 3-32
instruments without instrument program, 3-32
precedence rules, 3-31 to 3-32

M

macros, for writing platform-dependent code, 3-48
manual. *See* documentation.
Mark All for Compilation command, Build menu, 3-12
Mark File for Compilation command, Build menu
Project window, 3-11
Source, Interactive Execution, and Standard Input/Output windows, 4-23
Maximum number of compile errors option, 3-46
Maximum number of lines in Standard Input/Output window option, 3-52
maximum stack size, setting, 3-50
menuFont option, 1-6
<Meta> key (SPARCstation), 3-52
Microsoft Visual Basic, generating include file for, 4-38
Microsoft Windows
configuration options
cfgdir (Windows 3.1), 1-3 to 1-4
cvidir, 1-4
setting
Windows 3.1, 1-2
Windows 95/NT, 1-2
tmpdir, 1-5

debug options (Windows 3.1)
CatchProtectionFaults option, 1-6
DisplayCVIDebugVxDMissingMessage, 1-6
LoadCVIDebugVxD, 1-7
DLLs. *See* Microsoft Windows DLLs.
font options
DialogFontBold, 1-5
DialogFontName, 1-5
DialogFontSize, 1-5
installing system libraries, 3-41
Microsoft Windows DLLs. *See also* Create Dynamic Link Library dialog box.
DLL import library, generating, 4-37
DLL option, Add Files to Project command, 3-8
DLL path option, Add Files to Project command, 3-8
glue code, generating, 4-38
instrusup.dll, 3-12 to 3-13
Reload DLLs On Each Run option, 3-51
source code for creating DLL import library, generating, 4-37
Unload DLLs After Each Run option, 3-51
Minimize All command, Window menu, 3-42
_M_IX86 macro, 3-49
mouse, moving over new dialog boxes, 3-52
Move Item Down command, Edit menu, 3-9
Move Item Up command, Edit menu, 3-8
multi-dimensional arrays
illustration, 7-2
Reset Indices dialog box, 7-2 to 7-3
specifying dimensions, 7-2
multi-dimensional strings, 7-3 to 7-4
Multiple Files option, Find command, 4-15

N

Name option, Find command, 6-7
National Instruments libraries. *See* specific libraries; standard libraries.
New command, File menu
Array and String Display windows, 7-4
Function Panel windows, 5-7
Project window, 3-4
Source, Interactive Execution, and Standard Input/Output windows, 4-7
Variable Display and Watch windows, 6-5
New Window option, Select Function Panel dialog box, 3-37
Next Build Error command, View menu, 4-24
Next File command, Edit menu, 4-17
Next Function Panel command, View menu, 5-19
Next Function Panel Window command, View menu, 5-19
Next Panel command, View menu, 5-2
Next Scope command, Edit menu, 6-8
Next Tag command, View menu, 4-19
_NI_BC macro, 3-49
_NI_i386_ macro, 3-48
_NI_mswin_ macro, 3-48
_NI_mswin16_ macro, 3-48
_NI_mswin32_ macro, 3-49
_NI_SC macro, 3-49
_NI_sparc_ macro, 3-48
_NI_VC macro, 3-49
_NI_WC macro, 3-49
No Sorting command, View menu, 3-10
non-void functions, requiring return values for, 3-47
__NT__ macro, 3-49
NUL byte, difference from space character (note), 6-1, 7-1
numeric control parameters, specifying, 5-5

O

object files
creating, 4-38 to 4-39
required in project file list, 3-1
Object option, Add Files to Project command, 3-8
one-dimensional array, displaying in Array Display window (figure), 7-1
Open command, File menu
Array and String Display windows, 7-4
Function Panel windows, 5-7
Project window, 3-5
Source, Interactive Execution, and Standard Input/Output windows, 4-7
Variable Display and Watch windows, 6-5
Open Quoted Text command, File menu, 4-7
Options menu
Array and String Display windows
Display Entire Buffer command, 7-9
illustration, 7-9
Reset Indices command, 7-2, 7-3, 7-9
Function Panel windows
Change Format command, 5-21
Default All command, 5-21
Default Control command, 5-20
Edit Function Panel Window command, 5-21
Exclude Function command, 5-21
illustration, 5-20
Toggle Control Style command, 5-21
Toolbar command, 5-21
Project window
Colors command, 3-55 to 3-56
Compiler Options command, 3-46
Environment command, 3-37, 3-52
Font command, 3-55
illustration, 3-46
Include Paths command, 3-49 to 3-50
Instrument Directories command, 3-50
Keyboard Options command, 3-52
Library Options command, 3-41, 3-53 to 3-55
Project Move Options command, 3-55
Run Options command, 3-50 to 3-51
Source, Interactive Execution, and Standard Input/Output windows
Bracket Styles command, 4-35
Colors command, 4-36
Create Object File command, 4-38 to 4-39
Editor Preferences command, 4-10, 4-34 to 4-35
Font command, 4-35
Generate DLL Glue Object command, 4-38
Generate DLL Glue Source command, 4-38
Generate DLL Import Library command, 4-37
Generate DLL Import Source command, 4-37
Generate Visual Basic Include command, 4-38
illustration, 4-33
Keyboard Help command, 4-39 to 4-40
Syntax Coloring option, 4-36
Toolbar command, 4-35
Translate DOS LW program command, 4-36
User Defined Tokens for Coloring command, 4-36
Variable Display and Watch windows
Add Watch Expression command, 6-3, 6-15
Estimate Number of Elements command, 6-15
illustration, 6-14
Interpret As command, 6-15
Variable Size command, 6-14

Output command, File menu
Array and String Display windows, 7-5
Variable Display and Watch windows, 6-5
output control parameters, specifying, 5-6
Overwrite command, Edit menu, 7-7

P

parent structure, 6-2
parent structure pointer in chain (figure), 6-11
parentheses, finding pairs of, 4-12
Paste command, Edit menu, 4-10
Paste option, Editor Preferences command, 4-34
path options
Always prompt before fixing pathnames, 3-55
Fixup pathnames when project is moved, 3-55
Prompt for include file paths, 3-47
pathnames
displaying project files by full pathname, 3-10
sorting project files by pathname, 3-10
paths for compiler, listing, 3-49 to 3-50
pointer mismatch warning, enabling, 3-47
predefined macros, for writing platform-dependent code, 3-48
Previous Build Error command, View menu, 4-24
Previous Function Panel command, View menu, 5-19
Previous Function Panel Window command, View menu, 5-19
Previous Panel command, View menu, 5-2
Previous Scope command, Edit menu, 6-8
Previous Tag command, View menu, 4-19
Print command, File menu
Project window, 3-6
Source, Interactive Execution, and Standard Input/Output windows, 4-9
Project command, Window menu, 3-43
project files
optional files, 3-1
options for displaying, 3-9 to 3-10
required files, 3-1
saving automatically, 3-6
Project Move Options command, Options menu
Always prompt before fixing pathnames, 3-55
description, 3-55
Fixup pathnames when project is moved, 3-55
Project window
Build menu, 3-10 to 3-27
Edit menu, 3-7 to 3-9
File menu, 3-3 to 3-7
icons, 3-2 to 3-3
illustration, 2-5, 3-2
Instrument menu, 3-33 to 3-37
Library menu, 3-38 to 3-41
opening
with New command, 3-4
with Open command, 3-5
optional files, 3-1
Options menu, 3-45 to 3-56
overview, 3-1 to 3-3
purpose and use, 2-4
required files, 3-1
Run menu, 3-27 to 3-28
View menu, 3-9 to 3-10
Window menu, 3-42 to 3-45

R

Read Only command, File menu, 4-8
Reattach Program command, Edit Instrument dialog box, 3-36
Recall Function Panel command, View menu
invoking, 4-20
multiple functions in one function panel window, 4-21
multiple panels for one function, 4-21
purpose, 4-20

recalling from function name only, 4-20 to 4-21
syntax requirements, 4-21
Redo command, Edit menu, 4-10
regular expression characters (table), 4-14 to 4-15
Regular Expression option, Find command
Array and String Display windows, 7-6
Source, Interactive Execution, and Standard Input/Output windows, 4-13
Reload DLLs On Each Run option, 3-51
Remove File command, Edit menu, 3-8
Replace command, Edit menu
button bar, 4-16 to 4-17
Find Next button, 4-16
keyboard commands for bypassing dialog box (table), 4-17
Replace button, 4-16
Replace All button, 4-17
Return button, 4-17
Stop button, 4-17
Require function prototypes option, 3-46 to 3-47, 4-4
Require return values for non-void functions option, 3-47
resdir configuration option, 1-4
Reset Indices command, Options menu
description, 7-9
displaying single-dimensional arrays, 7-2
specifying index for string array, 7-3
specifying plane and dimensions for multi-dimensional arrays, 7-2
Reset Indices dialog box, 7-2 to 7-3
Resolve All Excluded Lines command, Edit menu, 4-11
Retrace Pointer Chain command, View menu, 6-11
return value controls, 5-4
return values, requiring for non-void functions, 3-47
ring control parameters, specifying, 5-5 to 5-6
RS-232 command, Library menu, 3-40
RS-232 Library, 3-40
Run Function Panel command, Code menu, 5-8 to 5-9
Run Interactive Statements command, Run menu, 4-27 to 4-28
Run menu
Array and String Display windows, 7-9
Project window
Break at First Statement command, 3-28
Breakpoints command, 3-28
Continue command, 3-28
Execute command, 3-28
illustration, 3-27
Run Project command, 3-27
Terminate Execution command, 3-28
Source, Interactive Execution, and Standard Input/Output windows
Activate Panels When Resuming option, 4-31
Break at First Statement command, 4-26, 4-29
Breakpoints command, 4-26, 4-30 to 4-31
Close Libraries command, 4-29
Continue command, 4-28
Down Call Stack command, 4-32
Dynamic Memory command, 4-32
Expression Value command, 4-32
Finish Function command, 4-29
Go to Cursor command, 4-28
illustration, 4-25
Run Interactive Statements command, 4-27 to 4-28
Run Project command, 4-27 to 4-28
Stack Trace command, 4-32
Step Into command, 4-29
Step Over command, 4-28
Terminate Execution command, 4-29
Toggle Breakpoint command, 4-26, 4-29
Up Call Stack command, 4-32
Variable Value command, 4-32, 6-1, 7-3
Variable Display and Watch windows, 6-13

Run Options command, Options menu, 3-50 to 3-51
Break on library errors option, 3-51
Check disk dates before each run, 3-51
Debugging level, 3-51
Hide windows, 3-51
Maximum stack size (bytes), 3-50
Reload DLLs on each run, 3-51
Save changes before running, 3-51
Unload DLLs after each run, 3-51
Run Project command, Run menu
Project window, 3-27
Source, Interactive Execution, and Standard Input/Output windows, 4-27 to 4-28
-run startup option (table), 1-1
-run_then_exit startup option (table), 1-1
run-time error reporting
Project window, 3-28
Source, Interactive Execution, and Standard Input/Output windows, 4-28
Runtime Errors command, Window menu, 3-43

S

Save All command, File menu
Array and String Display windows, 7-5
Function Panel windows, 5-7
Project window, 3-6
Source, Interactive Execution, and Standard Input/Output windows, 4-8
Variable Display and Watch windows, 6-5
Save As command, File menu
Project window, 3-6
Source, Interactive Execution, and Standard Input/Output windows, 4-8
Save changes before running option, 3-51
Save command, File menu
Project window, 3-5
Source, Interactive Execution, and Standard Input/Output windows, 4-8
Save Copy As command, File menu, 4-8
sdk directory (table), 1-4
Search for Help On command, Help menu, 4-39
Select All command, Edit menu, 4-11
Select Attribute Constant Dialog Box, 5-12 to 5-13
Select Attribute Values Dialog Box, 5-13
Select Function Panel dialog box, 3-36 to 3-37
Alphabetize command, 3-36
Flatten checkbox, 3-36, 5-2
Function Names command, 3-36
Help button, 3-37
illustration, 3-36
New Window command, 3-37
Select UI Constant command, Code menu, 5-10 to 5-14
attribute constants from userint.h, 5-12 to 5-13
constants from .uir file, 5-11
value constants from userint.h, 5-13 to 5-14
Select UIR Constant Dialog Box, 5-11
Select Variable command, Code menu, 5-14 to 5-16
Select Variable or Expression Dialog Box, 5-14 to 5-15
data type compatibility, 5-15 to 5-16
Data Type list box, 5-14
Data Type of Control display, 5-14
illustration, 5-14
items included in list box, 5-15
Show Project Variables option, 5-14
sorting of list box entries, 5-16
Variable or Expession list box, 5-14
Selected Text Only option, Find command, 4-15
separators, adding and positioning on toolbar, 4-2 to 4-3
Set Target File command, Code menu, 5-17
Show Build Error window for warnings option, 3-48
Show Full Dates command, View menu, 3-9
Show Full Path Names command, View menu, 3-9
Show Info command, Edit Instrument dialog box, 3-35

signed/unsigned pointer mismatch warning, enabling, 3-47
single-dimensional array, displaying in Array Display window (figure), 7-1
skeleton code, 2-6
Sleep policy when not running program option, 3-52
slide controls
  definition, 5-5
  specifying parameters, 5-5
Sort By Date command, View menu, 3-9
Sort By File Extension command, View menu, 3-10
Sort By Name command, View menu, 3-9
Sort By Pathname command, View menu, 3-10
source files
  debugging, 2-5
  listed in Window menu, 3-45
  required in project file list, 3-1
Source option, Add Files to Project command, 3-8
Source window
  Build menu, 4-22 to 4-24
  context menus, 4-43
  Edit menu, 4-9 to 4-17
  File menu, 4-7 to 4-9
  Instrument menu, 4-33
  keyboard commands (figure), A-1
  Library menu, 4-33
  notification of external modification, 4-43
  opening
    with New command, 3-4
    with Open command, 3-5
  Options menu, 4-33 to 4-39
  purpose and use, 2-4, 4-1
  Run menu, 4-25 to 4-32
  selecting text, 4-5 to 4-7
  subwindows, 4-4
  View menu, 4-17 to 4-22
  Window menu, 4-33
space character, difference from NUL byte (note), 6-1, 7-1
stack size, setting, 3-50
Stack Trace command, Run menu, 4-32
  Down Call Stack, 4-32
  Up Call Stack, 4-32
standalone executables, creating and distributing
  Create Distribution Kit command, 3-21 to 3-27
  Create Standalone Executable dialog box, 3-14 to 3-15
Standard Input/Output command, Window menu, 3-45
Standard Input/Output window
  bringing to front whenever modified, 3-52
  Build menu, 4-22 to 4-24
  clearing, 4-5
  Edit menu, 4-9 to 4-17
  File menu, 4-7 to 4-9
  Instrument menu, 4-33
  Library menu, 4-33
  Options menu, 4-33 to 4-39
  purpose and use, 2-5, 4-5
  Run menu, 4-25 to 4-32
  selecting text, 4-5 to 4-7
  specifying maximum number of lines for, 3-52, 4-5
  subwindows, 4-5
  View menu, 4-17 to 4-22
  Window menu, 4-33
standard libraries. *See also* specific libraries.
  list of libraries, 2-2
  specifying optional libraries to be loaded, 3-53 to 3-54
startup options for LabWindows/CVI (table), 1-1
static library, creating, 3-18 to 3-19
status dialog, displaying, 3-48
Step Into command, Run menu, 4-29
Step Over command, Run menu, 4-28
Stop on first file with errors option, 3-48
String Display command, View menu, 6-12, 7-3
String Display window
  displayed in View menu, 3-44
  Edit menu, 7-7 to 7-8
  File menu, 7-4 to 7-5

Format menu, 7-8
illustration, 7-3
multi-dimensional strings, 7-3 to 7-4
Options menu, 7-9
purpose and use, 2-4, 7-3
Run menu, 7-9
Window menu, 7-9
structures
child structure, 6-2
child structure pointer in chain (figure), 6-11
Follow Pointer Chain command, 6-10 to 6-11
parent pointer to structure, 6-2
parent structure pointer in chain (figure), 6-11
pointer-linked structures, 6-10 to 6-11
replacing, 6-10 to 6-11
Retrace Pointer Chain command, 6-11
subwindows, in Source, Interactive Execution, and Standard Input/Output windows, 4-5
Syntax Coloring option, Options menu, 4-36
system libraries, installing
Microsoft Windows, 3-41
UNIX, 3-41

T

Tabs option, Editor Preferences command, 4-35
Tag Scope command, View menu, 4-19
tagged lines
Clear Tags command, 4-19
Next Tag command, 4-19
Previous Tag command, 4-19
Tag Scope command, 4-19
Toggle Tag command, 4-19
Target command, Build menu, 3-12
TCP command, Library menu, 3-40
TCP Library, 3-40
technical support, B-1 to B-2
Terminate Execution command, Run menu
Project window, 3-28
Source, Interactive Execution, and Standard Input/Output windows, 4-29
text, selecting
Character Select mode, 4-6
Column Select mode, 4-6
Line Select mode, 4-6
Tile Windows command, Window menu, 3-42
tmpdir configuration option, 1-5
Toggle Breakpoint command, Run menu, 4-26, 4-29
Toggle Control Style command, Options menu, 5-21
Toggle Exclusion command, Edit menu, 4-4, 4-11
Toggle Tag command, View menu, 4-19
tokens
Syntax Coloring option, Options menu, 4-36
User Defined Tokens for Coloring command, Options menu, 4-36
Toolbar command
Options menu
Function Panel windows, 5-18
Source, Interactive Execution, and Standard Input/Output windows, 4-35
View menu
Function Panel windows, 5-18
Source, Interactive Execution, and Standard Input/Output windows, 4-18
toolbars, 4-1 to 4-3
Customize Toolbar Dialog Box (illustration), 4-2
displaying names of button or icons, 4-1
function panels, 5-3
modifying, 4-2 to 4-3
positioning buttons and separators, 4-2 to 4-3
removing items, 4-3

Track include file dependencies option, 3-47
Translate LW DOS program command, Options menu, 4-36
Type option, Find command, 6-7

U

.uir files. *See* user interface resource (.uir) files.
Undo command, Edit menu, 4-10
Undo option, Editor Preferences command, 4-10, 4-34
UNIX operating system
  configuration options
    activate, 1-3
    cfgdir, 1-3 to 1-4
    cvidir, 1-4
    setting, 1-3
    tmpdir, 1-5
  font options
    appFont, 1-6
    dialogFont, 1-6
    editorFont, 1-6
    menuFont, 1-6
  installing system libraries, 3-41
Unload command, Instrument menu, 3-34, 3-41
Unload DLLs after each run, 3-51
unloading instrument drivers, 3-31 to 3-32
unreachable code warning, enabling, 3-47
Up Call Stack command, Run menu, 4-32
Update Program Files from Disk command, Build menu, 3-11
Use host's system standard I/O option, Environment command, 3-52
Use only one Function Panel option, Environment command, 3-52
Use only one Function Panel Window option, Environment command, 3-37
User Defined Tokens for Coloring command, Options menu, 4-36
User Interface command, Library menu, 3-38
user interface constants, selecting, 5-10 to 5-14
  attribute constants, 5-12 to 5-13
  from .uir files, 5-11
  value constants, 5-13 to 5-14
User Interface Editor window
  moving to using Find UI Object command, 4-22
  opening
    with New command, 3-4
    with Open command, 3-5
  purpose and use, 2-4
User Interface Library
  definition, 3-38
  purpose and use, 2-2 to 2-3
user interface objects, finding, 4-22
User Interface option, Add Files to Project command, 3-8
user interface resource (.uir) files
  displayed in Window menu, 3-44
  optional for project file list, 3-1
user libraries. *See also* libraries.
  dummy .fp files for support libraries, 3-41, 3-54
  installing into Library menu, 3-41
  instrument drivers vs., 3-54
  specifying in Library Options dialog box, 3-53 to 3-54
Utility command, Library menu, 3-40 to 3-41
Utility Library
  definition, 3-40
  functions in instrsup.dll, 3-13

V

value constants, selecting, 5-13 to 5-14
Value option, Find command, 6-7
Variable Display window
  Edit menu, 6-5 to 6-8
  File menu, 6-4 to 6-5
  Format menu, 6-13
  Function subwindow, 6-2
  Global subwindow, 6-2
  icons associated with variables, 6-2

illustration, 6-1
Options menu, 6-14 to 6-15
purpose and use, 2-4, 6-1 to 6-2, 6-2
Run menu, 6-13
View menu, 6-9 to 6-12
viewing, 6-1
Window menu, 6-14
Variable Size command, Options menu, 6-14
Variable Value command
Code menu, 5-6, 5-17, 6-1, 7-3
Run menu, 4-32, 6-1, 7-3
variables, selecting. *See* Select Variable or Expression Dialog Box.
Variables command, Window menu
Project window, 3-43
Variable Display window, 6-1
View menu
Function Panel windows
Current Tree command, 5-18
Error command, 5-18
Find Function Panel command, 5-18 to 5-19
First Function Panel Window command, 5-19
First Panel command, 5-2
Function Panel History command, 5-18
illustration, 5-17
Include File command, 5-18
Last Function Panel Window command, 5-19
Last Panel command, 5-2
Next Function Panel command, 5-19
Next Function Panel Window command, 5-19
Next Panel command, 5-2
Previous Function Panel command, 5-19
Previous Function Panel Window command, 5-19
Previous Panel command, 5-2
Toolbar command, 5-18
Project window, 3-9 to 3-10
illustration, 3-9
No Sorting command, 3-10
Show Full Dates command, 3-9
Show Full Path Names command, 3-9
Sort By Date command, 3-9
Sort By File Extension command, 3-10
Sort By Name command, 3-9
Sort By Pathname command, 3-10
Source, Interactive Execution, and Standard Input/Output windows
Beginning/End of Selection command, 4-19
Build Errors in Next File command, 4-24
Clear Tags command, 4-19
Find Function Panel command, 4-21 to 4-22
Find UI Object command, 4-22
Function Panel History command, 4-20
Function Panel Tree command, 4-20
illustration, 4-18
Line command, 4-19
Line Icons command, 4-18, 4-26
Line Numbers command, 4-18
Next Tag command, 4-19
Previous Tag command, 4-19
Recall Function Panel command, 4-20 to 4-21
Tag Scope command, 4-19
Toggle Tag command, 4-19
Toolbar command, 4-18
Variable Display and Watch windows
Array Display command, 6-12, 7-1
Close Variable command, 6-2, 6-10
Expand Variable command, 6-2, 6-9 to 6-10
Follow Pointer Chain command, 6-2, 6-10 to 6-11
Go To Definition command, 6-12
Go To Execution Position command, 6-12
illustration, 6-9
Retrace Pointer Chain command, 6-11
String Display command, 6-12, 7-3

VISA command, 3-40
Visual Basic, generating include file for, 4-38
VXI command, Library menu, 3-39
VXI Library, 3-39
VXI*plug&play* instrument driver files, 3-30

W

Warp mouse over dialog boxes option, 3-52
Watch command, Window menu
  Project window, 3-43
  Watch window, 6-3
watch variables/expressions
  Add/Edit Watch Expression dialog box, 6-3 to 6-4
  applicable only in source code modules (note), 4-26
  purpose and use, 4-25 to 4-26
  selecting, 6-3 to 6-4
  suspending program execution conditionally, 4-27
Watch window
  activating, 6-3
  Add/Edit Watch Expression dialog box, 6-3 to 6-4
  Edit menu, 6-8 to 6-9
  File menu, 6-4 to 6-5
  Format menu, 6-13
  illustration, 6-3
  purpose and use, 2-4, 6-1, 6-2 to 6-3
  selecting variables and expressions, 6-3 to 6-4
  View menu, 6-9 to 6-12
  Window menu, 6-14
Whole Word option, Find command
  Array and String Display windows, 7-6
  Source, Interactive Execution, and Standard Input/Output windows, 4-13
  Variable Display window, 6-7
WIN32 macro, 3-49
_WIN32 macro, 3-49
__WIN32__ macro, 3-49
Window menu
  Array/String Display windows, 3-44, 7-9
  Function Panel windows, 3-44, 5-20
  Help Editor windows, 3-44
  Project window
    Build Errors command, 3-43
    Cascade Windows command, 3-42
    Close All command, 3-43
    Function Tree files, 3-44
    illustration, 3-42
    Interactive Execution command, 3-45
    Minimize All command, 3-42
    open source files, 3-45
    Project command, 3-43
    Runtime Errors command, 3-43
    Standard Input/Output command, 3-45
    Tile Windows command, 3-42
    User Interface Resource files, 3-44
    Variables command, 3-43
    Watch command, 3-43
  Source, Interactive Execution, and Standard Input/Output windows, 4-33
  Variable Display and Watch windows, 6-1, 6-3, 6-14
Windows. *See* Microsoft Windows.
windows, hiding, 3-51
Windows DLLS. *See* Microsoft Windows DLLs.
_WINDOWS macro, 3-49
Wrap option, Find command
  Array and String Display windows, 7-6
  Source, Interactive Execution, and Standard Input/Output windows, 4-15
  Variable Display window, 6-7

X

X Client Property Library, 3-40
X Property command, 3-40
.Xdefaults file, setting configuration options, 1-3