LabWindows/CVI

Evaluation Guide

Worldwide Technical Support and Product Information

`ni.com`

National Instruments Corporate Headquarters

11500 North Mopac Expressway   Austin, Texas 78759-3504   USA   Tel: 512 683 0100

Worldwide Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 3262 3599,
Canada (Calgary) 403 274 9391, Canada (Montreal) 514 288 5722, Canada (Ottawa) 613 233 5949,
Canada (Québec) 514 694 8521, Canada (Toronto) 905 785 0085, China (Shanghai) 021 6555 7838,
China (ShenZhen) 0755 3904939, Czech Republic 02 2423 5774, Denmark 45 76 26 00, Finland 09 725 725 11,
France 01 48 14 24 24, Germany 089 741 31 30, Greece 30 1 42 96 427, Hong Kong 2645 3186,
India 91 80 535 5406, Israel 03 6393737, Italy 02 413091, Japan 03 5472 2970, Korea 02 3451 3400,
Malaysia 603 9596711, Mexico 001 800 010 0793, Netherlands 0348 433466, New Zealand 09 914 0488,
Norway 32 27 73 00, Poland 0 22 528 94 06, Portugal 351 210 311 210, Russia 095 238 7139,
Singapore 2265886, Slovenia 386 3 425 4200, South Africa 11 805 8197, Spain 91 640 0085,
Sweden 08 587 895 00, Switzerland 056 200 51 51, Taiwan 02 2528 7227, United Kingdom 01635 523545

For further support information, see the *Technical Support and Professional Services* appendix. To comment on
the documentation, send email to `techpubs@ni.com`.

Important Information

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 document 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

CodeBuilder™, CVI™, DataSocket™, FlexMotion™, IMAQ™, IVI™, LabVIEW™, Measurement Studio™, National Instruments™, NI™, NI Developer Suite™, ni.com™, NI-488.2™, NI-DAQ™, NI-IMAQ™, and TestStand™ are trademarks of National Instruments Corporation.

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

Patents

For patents covering National Instruments products, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your CD, or `ni.com/patents`.

WARNING REGARDING USE OF NATIONAL INSTRUMENTS PRODUCTS

(1) NATIONAL INSTRUMENTS PRODUCTS ARE NOT DESIGNED WITH COMPONENTS AND TESTING FOR A LEVEL OF RELIABILITY SUITABLE FOR USE IN OR IN CONNECTION WITH SURGICAL IMPLANTS OR AS CRITICAL COMPONENTS IN ANY LIFE SUPPORT SYSTEMS WHOSE FAILURE TO PERFORM CAN REASONABLY BE EXPECTED TO CAUSE SIGNIFICANT INJURY TO A HUMAN.

(2) IN ANY APPLICATION, INCLUDING THE ABOVE, RELIABILITY OF OPERATION OF THE SOFTWARE PRODUCTS CAN BE IMPAIRED BY ADVERSE FACTORS, INCLUDING BUT NOT LIMITED TO FLUCTUATIONS IN ELECTRICAL POWER SUPPLY, COMPUTER HARDWARE MALFUNCTIONS, COMPUTER OPERATING SYSTEM SOFTWARE FITNESS, FITNESS OF COMPILERS AND DEVELOPMENT SOFTWARE USED TO DEVELOP AN APPLICATION, INSTALLATION ERRORS, SOFTWARE AND HARDWARE COMPATIBILITY PROBLEMS, MALFUNCTIONS OR FAILURES OF ELECTRONIC MONITORING OR CONTROL DEVICES, TRANSIENT FAILURES OF ELECTRONIC SYSTEMS (HARDWARE AND/OR SOFTWARE), UNANTICIPATED USES OR MISUSES, OR ERRORS ON THE PART OF THE USER OR APPLICATIONS DESIGNER (ADVERSE FACTORS SUCH AS THESE ARE HEREAFTER COLLECTIVELY TERMED "SYSTEM FAILURES"). ANY APPLICATION WHERE A SYSTEM FAILURE WOULD CREATE A RISK OF HARM TO PROPERTY OR PERSONS (INCLUDING THE RISK OF BODILY INJURY AND DEATH) SHOULD NOT BE RELIANT SOLELY UPON ONE FORM OF ELECTRONIC SYSTEM DUE TO THE RISK OF SYSTEM FAILURE. TO AVOID DAMAGE, INJURY, OR DEATH, THE USER OR APPLICATION DESIGNER MUST TAKE REASONABLY PRUDENT STEPS TO PROTECT AGAINST SYSTEM FAILURES, INCLUDING BUT NOT LIMITED TO BACK-UP OR SHUT DOWN MECHANISMS. BECAUSE EACH END-USER SYSTEM IS CUSTOMIZED AND DIFFERS FROM NATIONAL INSTRUMENTS' TESTING PLATFORMS AND BECAUSE A USER OR APPLICATION DESIGNER MAY USE NATIONAL INSTRUMENTS PRODUCTS IN COMBINATION WITH OTHER PRODUCTS IN A MANNER NOT EVALUATED OR CONTEMPLATED BY NATIONAL INSTRUMENTS, THE USER OR APPLICATION DESIGNER IS ULTIMATELY RESPONSIBLE FOR VERIFYING AND VALIDATING THE SUITABILITY OF NATIONAL INSTRUMENTS PRODUCTS WHENEVER NATIONAL INSTRUMENTS PRODUCTS ARE INCORPORATED IN A SYSTEM OR APPLICATION, INCLUDING, WITHOUT LIMITATION, THE APPROPRIATE DESIGN, PROCESS AND SAFETY LEVEL OF SUCH SYSTEM OR APPLICATION.

Conventions

The following conventions are used in this manual:

» The » symbol leads you through nested menu items and dialog box options to a final action. The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and select **Options** from the last dialog box. This symbol also guides you through the LabWindows/CVI function panel libraries.

This icon denotes an activity that you can complete to practice the concepts presented in that section.

This icon denotes a tip, which alerts you to advisory information.

This icon denotes a note, which alerts you to important information.

This icon denotes a caution, which advises you of precautions to take to avoid injury, data loss, or a system crash.

bold Bold text denotes items that you must select or click on in the software, such as menu items and dialog box options. Bold text also denotes parameter names.

italic Italic text denotes variables, emphasis, a cross reference, or an introduction to a key concept. This font also denotes text that is a placeholder for a word or value that you must supply.

monospace Text in this font denotes text or characters that you should enter from the keyboard, sections of code, programming examples, and syntax examples. This font is also used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames and extensions, and code excerpts.

Contents

Chapter 4
Where to Go from Here

Appendix A
Technical Support and Professional Services

1

National Instruments, Virtual Instrumentation, and LabWindows/CVI

National Instruments is committed to providing hardware and software for engineers, scientists, and systems integrators who build, maintain, and improve test, measurement, and automation applications. NI provides high performance, seamless integration, and rapid application development of virtual instruments at a lower cost than traditional measurement instruments.

This chapter discusses the NI platform and provides an introduction to virtual instrumentation and NI LabWindows/CVI.

NI Platform

NI pioneered the field of virtual instrumentation, which combines advanced hardware, powerful software, and off-the-shelf personal computers, to revolutionize the test and measurement field. This revolution has become the NI platform.

For traditional C-language programmers who develop virtual instrumentation, LabWindows/CVI has become the most popular software development environment. With LabWindows/CVI, you can take advantage of the functionality targeted at test, measurement, and automation applications for faster application development. LabWindows/CVI combines the power of ANSI C and the flexibility of Microsoft Windows with easy-to-use tools for building virtual instrumentation systems.

You can use the unique interactive code generation utilities in LabWindows/CVI to unleash the power of ANSI C and speed development in the following areas:

- Creating and controlling graphical user interfaces.
- Controlling instruments and plug-in data acquisition hardware.
- Developing and debugging ANSI C programs for instrumentation.

Virtual Instrumentation

A virtual instrument is a combination of hardware and software in a PC with the functionality of a classic stand-alone instrument. With a virtual instrument, you acquire and control data much like you do with traditional laboratory instruments. The power of NI virtual instrumentation comes with interactive data analysis, presentation, and distribution of your applications.

Creating virtual instruments is a three-step process when you use NI hardware with LabWindows/CVI.

Step 1—Acquire

Acquire raw data through a hardware interface. LabWindows/CVI provides hardware connections through an extensive list of interfaces—PCI, PXI, GPIB, VXI, serial, and even Ethernet. Whether you collect data through a plug-in board or through a benchtop instrument, NI has the interface to collect the correct signals and measurements.

Step 2—Analyze

Analyze raw data to make it meaningful to you and your users. Rather than collecting a list of raw signals and later analyzing them with a separate analysis application, use in-place analysis for fast and meaningful results. LabWindows/CVI offers the Advanced Analysis Library for measurement analysis.

Step 3—Present

Present results in an intuitive way and when applicable, allow users to interact with the instrumentation system through a graphical user interface (GUI). A GUI is the interactive user interface of a virtual instrument. Use three-dimensional, instrument-style controls and indicators—buttons, knobs, thermometers, tanks, LEDs, slides, numerical displays, strip charts, graphs, and tables—to build front panels that represent traditional test and measurement instruments.

LabWindows/CVI

LabWindows/CVI is a complete ANSI C development environment and compiler with built-in libraries for acquisition, analysis, and presentation. LabWindows/CVI provides a simplified drag-and-drop User Interface Editor and automated code generation tools that you can use to interactively test code before adding it to your project. With LabWindows/CVI, you can create localized user interfaces, work with ActiveX components, and create multithreaded applications. LabWindows/CVI delivers unique code generation utilities and the ability to create and control GUIs, control instruments and plug-in data acquisition hardware from your PC, and develop and debug ANSI C programs.

Measurement Libraries and Components

You can interact with the LabWindows/CVI analysis, I/O, presentation, and other measurement-specific libraries and components through function panel libraries. The following figure shows the User Interface function panel library.
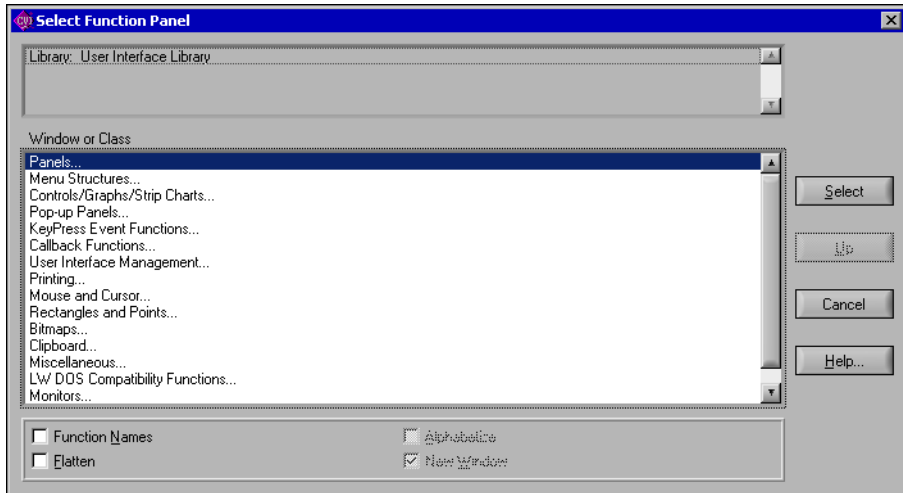
Function panels are graphical representations of LabWindows/CVI functions and their parameters. You can use function panels to interactively test your function calls and paste them into your program. The following figure shows the function panel for the `LoadPanel` function.
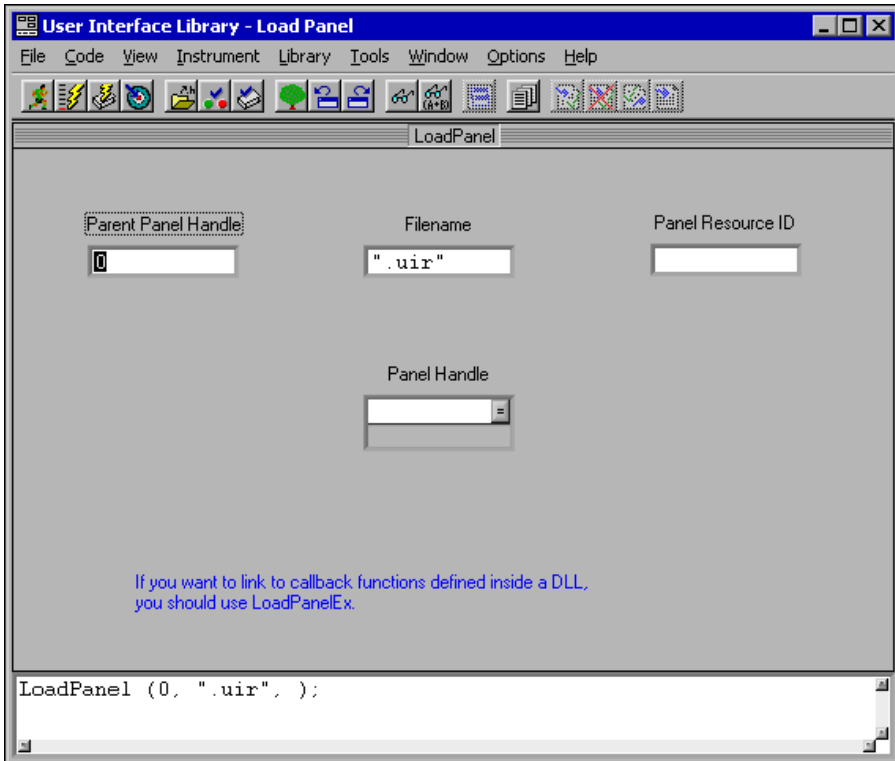
Data Acquisition and Instrument Control

LabWindows/CVI provides a high-level interface for acquiring your data—whether you use a GPIB or serial instrument, a plug-in data acquisition board, PXI hardware, NI computer-based instruments, or an image acquisition device.

Data Analysis

LabWindows/CVI delivers the analysis functions you need to convert raw data into meaningful information. You can employ a variety of signal processing and data analysis tools—such as curve fitting, spectral analysis, statistics, and visualization—for research and development, engineering and validation, or manufacturing and service applications. Because you can perform your analysis inside your application, you do not need another application for post-process analysis.

Presentation

LabWindows/CVI delivers ready-to-use, measurement-specific user interface components with which you can quickly create professional measurement applications that mimic stand-alone instruments, such as scopes and temperature loggers. Use three-dimensional controls to create revolutionary application interfaces.

Networking

Whether you pass data between applications or over the Internet, LabWindows/CVI makes sharing measurement data easy. With NI DataSocket, a programming tool designed for publishing and subscribing to live data in measurement and automation applications, you can publish or retrieve data, control remote applications, or distribute your system with one or more client applications on a network without worrying about data formats and network protocols.

Connect to data sources via the following interfaces from applications you develop with LabWindows/CVI:

- DSTP (DataSocket Server Transfer Protocol)
- OPC
- HTTP
- FTP
- File

More Features to Meet Your Needs

NI is committed to improving and extending the features of LabWindows/CVI. LabWindows/CVI 6.0 incorporates new tools and features capable of improving virtual instrumentation development and execution.

- ActiveX Control Container—Use ActiveX controls in the User Interface Editor and program them in your LabWindows/CVI programs.
- ActiveX Server Creation—LabWindows/CVI 6.0 includes a wizard that you can use to create and edit ActiveX server projects.
- New Palette of User Interface Controls—The User Interface Library includes a new set of controls with three-dimensional styles. In addition, the existing controls have been modified to more closely resemble standard Windows controls.
- Source Code Browser—The Source Code Browser is a cross-reference tool that lists all files, functions, variables, data types, and macros in a program. You can use the browser to identify how different parts of a program interact with each other.

- Enhanced User Protection—LabWindows/CVI 6.0 includes additional user protection in the form of new compile-time and run-time errors and warnings:

  - Uninitialized local variables—Generates compiler warnings and run-time errors when LabWindows/CVI detects the use of variables that do not have assigned values.

  - Unreferenced identifiers—Generates compiler warnings for identifiers that are not referenced in your program.

  - Assignments in conditional expressions—Checks for possible errors in conditional expressions that make the conditional expression an assignment.

Related Software Packages

LabWindows/CVI offers additional packages for targeted applications. Visit the Product Catalog at ni.com for more information about the following software packages:

NI Developer Suite

For one package that includes all of the tools you need to develop an automated test application, consider NI Developer Suite. NI Developer Suite is a software subscription program that packages our most popular software and includes free updates for a year. The NI Developer Suite Professional Test Edition includes LabWindows/CVI, LabVIEW, and Measurement Studio for developing test routines and TestStand for managing test execution, sequencing, collecting data, and generating reports. The test edition also includes a comprehensive set of LabVIEW, Measurement Studio, and LabWindows/CVI add-on tools for Internet connectivity, database communication, signal processing, and code distribution.

TestStand

TestStand is a ready-to-run test executive for organizing, controlling, and executing your automated prototype, validation, or manufacturing test systems. TestStand is completely customizable, so you can modify and enhance it to match your specific needs. Built on a high-speed, multithreaded, parallel execution engine, TestStand delivers high performance to meet your rigorous test throughput requirements.

Vision and Image Processing Software

NI Vision and Image Processing Software includes IMAQ Vision, a library of vision functions, and IMAQ Vision Builder, an interactive environment for prototyping vision applications. Use vision and image processing software to build machine vision and scientific imaging applications.

IVI Driver Toolset

Instrument drivers, which are an important component in test systems today, perform the actual communication and control of the instrument hardware in your system. Instrument drivers provide a high-level, easy-to-use programming model. This programming model gives you complete access to the complex measurement capabilities of instruments via an intuitive API, thus delivering modular, off-the-shelf components ready to use in your test programs.

Tip   Visit the Instrument Driver Network at `ni.com/idnet` to find additional instrument drivers. Browse through instrument drivers by instrument type, manufacturer, or development language. Or, search for a specific instrument driver using the specifications most important to you—from a specific model name or manufacturer, to IVI drivers or drivers supported by NI.

LabWindows/CVI Enterprise Connectivity Toolset

Enterprise connectivity tools help you track research and development, production, test, and servicing of products. The Enterprise Connectivity Toolset provides integrated tools for SQL database operations, SPC quality control, and Internet-enabling technology to give you the access and networking you need for business operations.

The following toolkits are included in the LabWindows/CVI Enterprise Connectivity Toolset:

• LabWindows/CVI Internet Developers Toolkit

• LabWindows/CVI SQL Toolkit

• LabWindows/CVI SPC Toolkit

• LabWindows/CVI Signal Processing Toolset

   The Signal Processing Toolset provides four toolkits for digital filter design, joint time-frequency analysis, wavelet and filter bank design, and super-resolution, model-based spectral analysis.

   The following toolkits are included in the Signal Processing Toolset:

   – Digital Filter Design Toolkit—Use for signal conditioning, control systems, and digital signal processing.

   – Joint Time-Frequency Analysis Toolkit—Use to simultaneously examine the time and frequency domain representations of a signal.

   – Wavelet & Filter Bank Toolkit—Use for wavelet design and implementation.

   – Super-Resolution Spectral Analysis—Use the stand-alone application that provides a model-based alternative to the FFT method of spectral analysis.

PID Control Toolset

The PID Control Toolset adds sophisticated control algorithms to your LabWindows/CVI applications. Implementing the standard PID algorithm, this toolset includes an autotuning wizard that assists you in optimizing the PID parameters to your specific system. Use the PID toolset with any kind of physical input and output options, including DAQ devices and PLCs.

2

Getting Started with LabWindows/CVI

This chapter introduces the LabWindows/CVI Evaluation Package and provides system requirements and installation information.

About this Evaluation Package

Evaluate LabWindows/CVI with real-world test and measurement applications. The LabWindows/CVI Evaluation Package contains the LabWindows/CVI Full Development System with the following restrictions:

- 30-day expiration
- 10-minute run-time timeout
- No external compiler support
- Disabled Create Distribution Kit feature
- Disabled Create Object File feature
- Disabled generation of import libraries from header files
- No IVI Wizard support (disabled Create IVI Instrument Driver feature)
- No Instrument Driver Only support
- No LabVIEW Real-Time support

The Full Development System includes the instrument control, user interface, advanced analysis, digital signal processing, and networking tools you need to start building test and measurement programs. With LabWindows/CVI, you can acquire data with NI data acquisition (DAQ) boards; control serial, GPIB, and VXI instruments and controllers; analyze the data you acquired from a device; develop custom user interfaces to present data; and share live data between different applications over the Internet.

⚠ **Caution** To prevent the loss of work, save all program files created with the LabWindows/CVI Evaluation Package and open them later with a licensed version of LabWindows/CVI.

Minimum System Requirements

To run LabWindows/CVI, your computer must have the following:

• 133 MHz Pentium microprocessor

• Windows 2000/Me/9*x* or Windows NT, Service Pack 3 or later

• VGA resolution or higher video adapter with a minimum 800 × 600 resolution; 16-bit color for user interface controls

• 32 MB of memory for Windows NT/9*x*; 64 MB of memory for Windows 2000/Me

• 175 MB of free disk space

Installation Instructions

Unless you specify another location during installation, the LabWindows/CVI installation program copies files to `c:\Program Files\National Instruments\ MeasurementStudio\` after you complete the following steps:

1. Insert the CD in the CD drive. If the CD startup screen does not appear, use Windows Explorer to run `SETUP.EXE`.

2. On installation startup, the National Instruments Measurement Studio 6.0 screen appears. Select **Install Measurement Studio**. Click **Next**. Follow the instructions on the screen.

3. To install only LabWindows/CVI, on the Select Features screen, click the component that you do not want to install and select **Entire feature will be unavailable**. Click **Next**.

✎ **Note** If you decide later that you want to install additional features or components, run the installer again, click the feature or component you want to install, and select **Entire feature will be installed on local hard drive**.

4. Continue to follow the instructions on the screen.

5. Install driver software if you plan to use LabWindows/CVI with NI hardware.

6. Install NI hardware. Refer to the hardware installation guide for installation information.

7. Configure the NI hardware with NI Measurement & Automation Explorer (MAX).

Exploring LabWindows/CVI

If you are already programming with C, LabWindows/CVI complements existing efforts and streamlines future development. Because LabWindows/CVI is built on an open software architecture, you can reuse existing programs within the LabWindows/CVI environment, and you can incorporate standard ANSI C code, object files, and DLLs in applications.

As you develop applications in LabWindows/CVI, you work within three main windows: the Project window for `*.prj` files, the Source window for `*.c` files, and the User Interface Editor window for `*.uir` files.

You can complete this activity in 5 minutes.

1. Launch LabWindows/CVI. When you open LabWindows/CVI, you see an empty Project window. The project file contains all of the files you need in final compiled projects.

2. Explore the three main windows in LabWindows/CVI. To open the Source window, select **File»New»Source (*.c)**. To open the User Interface Editor window, select **File»New»User Interface (*.uir)**.

For information about using the LabWindows/CVI environment, refer to *Using LabWindows/CVI* from the Contents of the *LabWindows/CVI Help*.

Context-Sensitive Help

NI designed LabWindows/CVI to deliver the appropriate information when you need it. In addition to traditional help files, you can find context-sensitive help throughout LabWindows/CVI. Look for and take advantage of **Help** buttons in all dialog boxes and F1 connectivity in the Source window, User Interface Editor, and menus in all windows.

*   **Help file**—Select **Help»Contents** in LabWindows/CVI to access the help file.

*   **Menus, dialog boxes, and Source window keywords**—Press <F1> on menus in any window, dialog boxes, or function names in the Source window. You also can right-click menu items and select **Menu Help** to learn more about the options and features available in that menu.

*   **Function panels**—Right-click the panel or any parameter for help about that function or parameter.

*   **User Interface Editor window**—Right-click any control and select **Control Help** to open help for GUI controls.

3

Creating a LabWindows/CVI Project

This chapter guides you through the process of creating a project that retrieves data from a simulated instrument. You will learn how to create a user interface, generate code, and add an instrument driver to your project.

Creating a New LabWindows/CVI Project

You can complete this activity in 2 minutes.

To create a Project file, perform the following steps:

1. Select **File»New»Project (*.prj)**.

2. If prompted, click **Yes** to unload the current project, and select **Create Project in New Workspace** in the New Project Options dialog box.

3. Save the project as `AcquireWaveform.prj`.

Creating a Graphical User Interface

LabWindows/CVI provides a collection of highly configurable, instrument-style user interface controls and indicators so users of your application can interact with and monitor the measurement or control system. Choose from the following controls to create your user interface:

*   knobs, meters, gauges, dials
*   binary switches, LEDs
*   slides, tanks, thermometers
*   real-time 2D graphs
*   tables
*   ActiveX controls
*   timers

The User Interface Editor

The User Interface Editor is an interactive drag-and-drop environment for the design of GUIs. You can select controls (buttons, toggles, slides, graphs, LEDs, and more) from the **Create** menu and position them on your GUI. You then can use dialog boxes to set control attributes, such as labels, colors, and hot key connections. In the following section, you build a GUI that acquires and displays a waveform.

🕐   **You can complete the following activity in 10 minutes.**

The following figure shows the completed GUI.
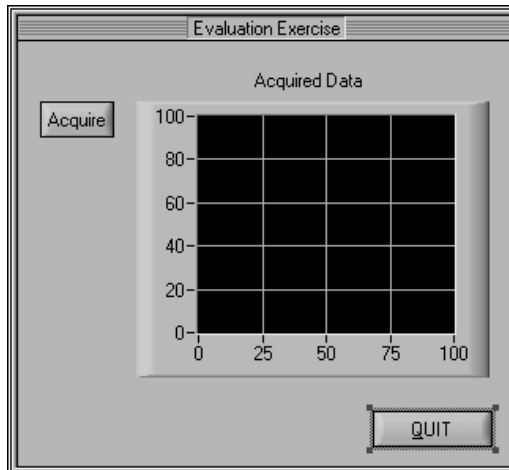
## Creating a New User Interface Resource (.uir) File

1. Select **File»New»User Interface (.uir)**. The User Interface Editor opens an untitled file with an empty panel.

2. Double-click the panel. Enter `Evaluation Exercise` as the Panel Title and click **OK**.

3. Save the `.uir` as `AcquireWaveform.uir`.

✏️   **Note**   When you save a `.uir` file, LabWindows/CVI automatically creates a header (`.h`) file with the same name for the user interface. This header file is stored in the same directory as the `.uir` file.

Adding Command Buttons

1. Select **Create»Command Button**»**Square Command Button** from the Lab Style Controls. A button labeled **OK** appears on the panel. Position the button in the upper left corner of the panel.

2. To edit the button attributes, double-click the button. In the Edit Command Button dialog box, enter ACQUIRE as the Constant Name for the command button.

3. Enter AcquireData in the Callback Function field. The callback function is called when a user clicks the button.

4. Enter Acquire as the label for the command button.

5. Make sure the Edit Command Button dialog box matches the one shown in the following figure and then click **OK**.
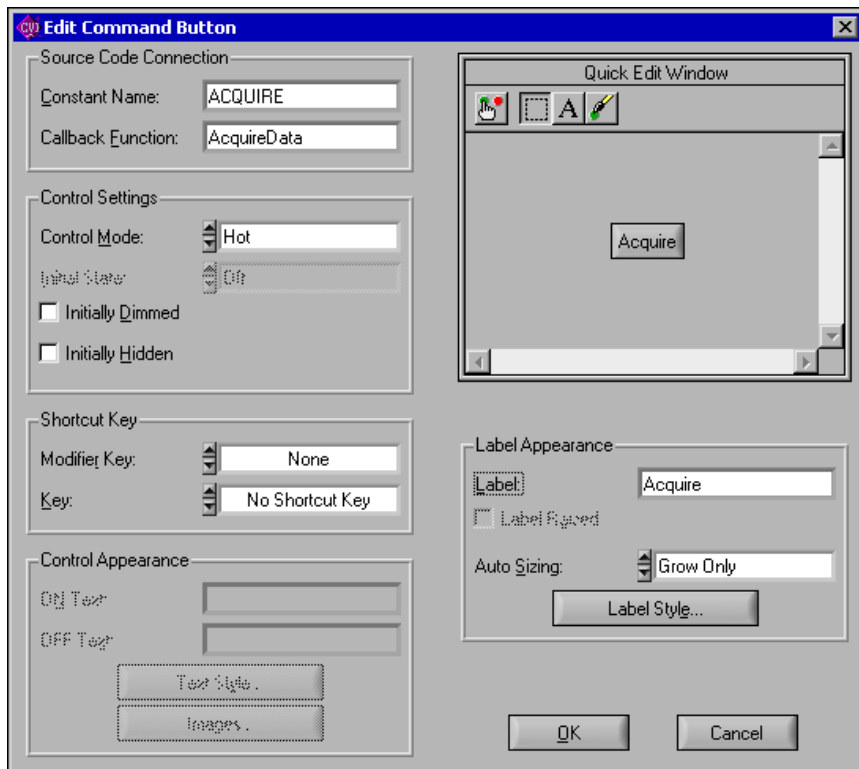
6. Select **Create»Custom Controls»Quit Button**. Custom controls are controls that you can configure and save between LabWindows/CVI sessions.

7. Position the **QUIT** button in the lower right corner of the panel.

Adding a Graph Control

1.   Select **Create»Graph»Graph** from the Lab Style Controls.

2.   Position the graph between the command buttons.

3.   Double-click the graph control to open the Edit Graph dialog box.

4.   Type WAVEFORM as the Constant Name.

Note   Because this graph does not require an action from the user interface, you do not need to assign a callback function to this control. Callback functions are necessary only when the operation of the control initiates an action or acts as an input.

5.   Type Acquired Data in the Label field.

6.   Click **OK** to close the dialog box.

7.   Save AcquireWaveform.uir.

Generating Your Program Code with CodeBuilder

LabWindows/CVI offers a unique code generation tool called CodeBuilder. CodeBuilder uses information that you specify in control and event dialog boxes to create a code skeleton. The code skeleton includes the main function, event callbacks, and an application shutdown callback.

LabWindows/CVI provides event-driven programming through callback functions. You can associate a specific callback function with a specific user interface control in the control's Edit dialog box, as you did in the previous section. As you set the properties for the control, think about what events you want the control to recognize—a single mouse click, double-click, or keypress—and specify the callback function that executes when those events occur.

You can specify default events for both the panel and controls on the panel. To enable events, select **Code»Preferences»Default Panel Events** and **Default Control Events**. When you generate your code, CodeBuilder creates a case statement for each event you enable—you just fill in the code to handle that event.

At this point, you have created a GUI with the user interface objects to acquire a waveform, display the waveform, and terminate program execution. Now, use CodeBuilder to create skeleton source code.

You can complete this activity in 5 minutes.

1.   Specify the type of events to which your program will respond. Open AcquireWaveform.uir if it is not already open. Select **Code»Preferences»Default Control Events**.

2. In the Control Callback Events dialog box, select the events that a control can generate. In this project, the controls respond to one type of event: a commit event (left-click or <Enter>). Verify that only EVENT_COMMIT is selected and then click **OK**.

💡 **Tip**   Click the **Help** button in the dialog boxes to learn more about default events and generating code. For more information about events, refer to *Events Overview* from the Index of the *LabWindows/CVI Help*.

3. Select **Code»Generate»All Code** to open the Generate All Code dialog box.

4. Select **Add To Current Project** in the Target Files section.

5. In general, you must set which panels you want to display at program startup. Because there is only one panel in this example, verify that PANEL is selected as the panel to load and display at startup and that the Panel Variable Name is panelHandle.

6. Select the QuitCallback function as the QuitUserInterface callback. LabWindows/CVI inserts the QuitUserInterface function into the QuitCallback callback.

7. Click **OK**. CodeBuilder builds the source code for your program. A new Source window, AcquireWaveform.c, appears containing the new source code.

Reviewing the Source Code

The main Function

The main function is the first component you need when you build any application. The main function for this project consists of the following lines of code:

```
int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1;  /* out of memory */
    if ((panelHandle = LoadPanel (0, "AcquireWaveform.uir", PANEL)) <
        0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;

}
```

• The LoadPanel function loads the panel from the .uir file into memory.

• The DisplayPanel function displays the panel on the screen.

• The RunUserInterface function prepares LabWindows/CVI to send events from the user interface to the C program you are developing.

The AcquireData Function

The AcquireData function automatically executes when a user clicks the **Acquire** button. Currently, the AcquireData function contains no functions to execute. In the following sections, you will add data acquisition functions within the AcquireData function.

```
int CVICALLBACK AcquireData (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{
   switch (event) {
      case EVENT_COMMIT:

         break;
   }
   return 0;
}
```

The QuitCallback Function

The QuitCallback function automatically executes when a user clicks the **QUIT** button. This function disables the user interface from sending event information to the callback function and stops execution of the program.

```
int CVICALLBACK QuitCallback (int panel, int control, int event,
      void *callbackData, int eventData1, int eventData2)
{
   switch (event) {
      case EVENT_COMMIT:
         QuitUserInterface (0);
         break;
   }
   return 0;
}
```

Running the Project

To run the project, select **Run»Debug AcquireWaveform_dbg.exe**. Currently, only the **QUIT** button responds to events.
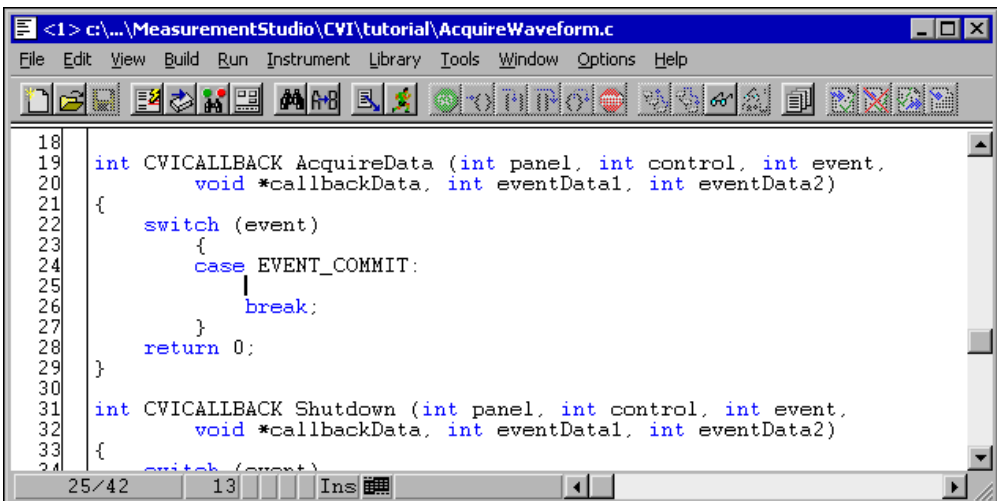
Adding an Instrument Driver to the Project

An instrument driver is a set of functions that controls an instrument or a group of related instruments. The high-level functions in an instrument driver incorporate many low-level operations, including GPIB, VXI, or RS-232 read and write operations, data conversion, and scaling. The sample module you will use communicates with a simulated instrument and shows you how to use an instrument driver to acquire a waveform from an oscilloscope.

To be complete, your program must read an array of simulated data from an instrument driver and plot the array on the graph. You must modify the AcquireData function in the AcquireWaveform.c source file as described in the following steps. Then, when a user clicks the **Acquire** button, the program reads the data from the instrument and plots the data on the graph.

Loading the Instrument Driver

1. Select **Edit»Add Files To Project»Instrument (*.fp)** in the Project window.

2. Select the tutorial\scope.fp file. Click **Add** and then click **OK** to add the driver to the project.

3. In the Project window, open AcquireWaveform.c.

4. Place your cursor on the blank line just after the EVENT_COMMIT: statement in the AcquireData function, as shown in the following figure.
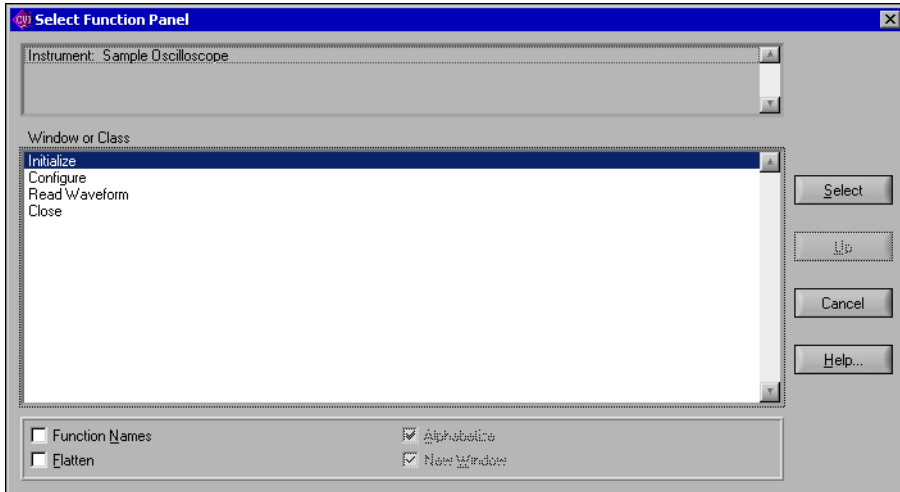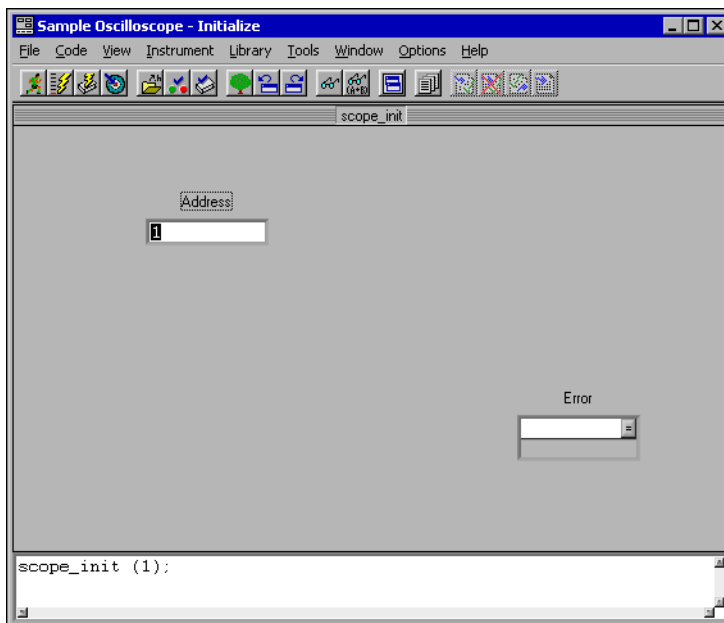
Initializing the Instrument

1. Select **Instrument»Sample Oscilloscope**. The sample oscilloscope driver contains four functions for communicating with a scope, as shown in the following figure.

2. Double-click the Initialize function. The scope initialization function panel appears, as shown in the following figure.

Function panels have complete help for the function. Right-click the function panel to see the help for the `Initialize` function, and right-click any parameter for parameter help. Function panels also generate code. When you enter values in the function panel controls, LabWindows/CVI builds the function call at the bottom of the screen.

3. Enter the value `1` in the Address control.

4. Enter `err` in the Error control.

5. Declare the `err` variable. Select **Code»Declare Variable**.

6. In the Declare Variable dialog box, select the **Execute declaration in Interactive Window** and **Add declaration to top of target file** options. Click **OK**.

7. Select **Code»Run Function Panel**.

    If no errors are detected during execution, the value in the Error control is `0`.

8. Select **Code»Insert Function Call** to copy the generated code to the Source window.

9. Close the function panel.

The function call to initialize the instrument driver now appears in your source code, below the `EVENT_COMMIT` code, as follows:

```
err = scope_init (1);
```

Reading Data from the Instrument

Perhaps the most important function of an instrument driver is to read data from an instrument and convert the raw data into a format your program can use. For example, a digital oscilloscope returns a waveform as a string of comma-separated ASCII numbers. The instrument driver parses the string, scales the data to volts, and places the data into an array in memory.

1. Select **Instrument»Sample Oscilloscope»Read Waveform**.

2. In the Read Waveform function panel, set the Channel control to 2. Channel 1 is sine wave data and Channel 2 is random data.

3. Enter `datapoints` in the Waveform control.

4. Select **Code»Declare Variable** to declare the `datapoints` variable in memory. In the Declare Variable dialog box, enter `100` in the Number of Elements control.

5. Verify that the **Execute declaration in Interactive Window** and **Add declaration box to top of target file** options are selected.

6. Click **OK** to declare the `datapoints` array.

7. Enter `delta_t` in the Sample Period control.

8. Select **Code»Declare Variable**. Verify that the **Execute declaration in Interactive Window** and **Add declaration box to top of target file** options are selected and click **OK**.

9. Enter x_zero in the X Zero control.

10. Select **Code»Declare Variable**. Verify that the **Execute declaration in Interactive Window** and **Add declaration to top of target file** options are selected and click **OK**.

11. Enter err in the Error control.

12. Select **File»Save All**. Select **Code»Run Function Panel** to execute the function panel. If the Initialize function executes correctly, the Error control indicates 0. After the function executes, a row of boxes in the Waveform control signifies that the data is in the waveform array.

13. Optional—You can double-click the row of boxes to display two windows of data returned by the function call: one showing variable values and the other showing the waveform data collected in the array. If you choose to review these windows, close both of them before continuing.

14. Select **Code»Insert Function Call** to copy the generated code to the Source window.

15. Close the function panel. The following line of code now exists within the AcquireData callback function:

```
err = scope_read_waveform (2, datapoints, &delta_t, &x_zero);
```

Displaying the Waveform on a Graph Control

At this point, you have completed the steps necessary to acquire data. Now you need to plot this data using the Plot Y function, which can plot the acquired data array to the graph control on the user interface.

1. To open the Plot Y function panel, type PlotY in the blank line below the read waveform function call and press <Ctrl-P>.

2. Complete the function panel controls to match the following settings:

    Panel Handle: panelHandle
    Control ID: PANEL_WAVEFORM
    Y Array: datapoints
    Number of Points: 100

    Leave the default settings for the remaining controls.

3. From the PlotY function panel, select **Code»Insert Function Call** to paste the PlotY function call into your source code. A message box appears with a warning that text already exists on the line where LabWindows/CVI will insert the function call. Click **Replace**.

4. Close the function panel. The changes you made to the AcquireData callback function match the source code shown in the following section.

5. Save AcquireWaveform.c.

Reviewing the Program

Make sure that your completed source file matches the following code block:

```
#include <cvirte.h>
#include <userint.h>
#include "AcquireWaveform.h"
static double x_zero;
static double delta_t;
static double datapoints[100];
static int err;

static int panelHandle;

int main (int argc, char *argv[])

{

if (InitCVIRTE (0, argv, 0) == 0)
        return -1;/* out of memory */
    if ((panelHandle = LoadPanel (0, "AcquireWaveform.uir", PANEL)) <
        0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;

}

int CVICALLBACK AcquireData (int panel, int control, int event,
        void *callbackData, int eventData1, int eventData2)

{
    switch (event) {
        case EVENT_COMMIT:
            err = scope_init (1);
            err = scope_read_waveform (2, datapoints, &delta_t,
                &x_zero);
            PlotY (panelHandle, PANEL_WAVEFORM, datapoints, 100,
                VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                VAL_SOLID, 1, VAL_RED);

            break;
    }
    return 0;
}
```

```
int CVICALLBACK QuitCallback (int panel, int control, int event,
     void *callbackData, int eventData1, int eventData2)
{
   switch (event) {
      case EVENT_COMMIT:
         QuitUserInterface (0);
         break;
   }
   return 0;
}
```

Running the Completed Project

You now have a completed project, saved as AcquireWaveform.prj. Select **Run»Debug AcquireWaveform_dbg.exe** to execute the code. During the compile process, LabWindows/CVI recognizes that your program is missing the scope.h header file statement and offers to insert it into your source code. Click **Yes** to add this include file to the program.

Note   When you click **Acquire** more than once, the data from previous acquisitions remains on display with the new data plot. You can modify the AcquireData callback to include the DeleteGraphPlot function as shown in the following code excerpt. DeleteGraphPlot deletes all plots from the graph.

Note   You can find the DeleteGraphPlot function panel in the User Interface Library.

```
int CVICALLBACK AcquireData (int panel, int control, int event,
   void *callbackData, int eventData1, int eventData2)
{
   switch (event) {
      case EVENT_COMMIT:
         DeleteGraphPlot (panelHandle, PANEL_WAVEFORM, -1,
                    VAL_DELAYED_DRAW);
         err = scope_init (1);
         err = scope_read_waveform (2, datapoints,
                    &delta_t, &x_zero);
         PlotY (panelHandle, PANEL_WAVEFORM, datapoints, 100,
                    VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE,
                    VAL_SOLID, 1, VAL_RED);
         break;
   }
   return 0;
}
```

Now that you have completed a LabWindows/CVI project, look through the rest of the libraries for functionality not covered in this manual. Also, explore the samples included in this evaluation version of LabWindows/CVI. You can find a number of samples in the `samples` folder.

4

Where to Go from Here

LabWindows/CVI is yet another long-standing commitment by NI to provide tools that simplify the development of instrumentation, data acquisition, and control systems. When you choose LabWindows/CVI, you join thousands of scientists and engineers who take advantage of the power of this versatile tool for quick and easy test and measurement development.

Visit the LabWindows/CVI Web site at `ni.com/cvi` and NI Developer Zone at `ni.com/zone` for the most up-to-date information.

Tutorials and Related Documentation

Visit the Measurement Studio Library, available from **Start»Programs» National Instruments»Measurement Studio»Help**. The Library lists all currently available LabWindows/CVI documentation. The Library also describes and links to LabWindows/CVI manuals, application notes, and white papers in PDF format. Read the *Getting Started with LabWindows/CVI* manual for a complete tutorial.

Note Use Adobe Acrobat Reader 3.0 or later to view any LabWindows/CVI PDF file. You can download a free version of the Reader from `www.acrobat.com` or install a copy from your CD.

Programmer and Reference Information

The *LabWindows/CVI Help* contains complete reference information. *LabWindows/CVI Help* contains the following sections:

- *Using LabWindows/CVI* describes each window and dialog box in the LabWindows/CVI.
- *Library Reference* includes reference for functions you can call in LabWindows/CVI.
- *Programmer Reference* provides information to help you develop applications.
- *Example Programs* contains LabWindows/CVI samples you can launch and run from the help topic.
- *Tools Library* lists instrument drivers included with LabWindows/CVI.

To launch the help file, select **Help»Contents** from all LabWindows/CVI windows.

Also, review the Quick Reference card provided with this evaluation package. The card lists the LabWindows/CVI libraries.

LabWindows/CVI Examples

LabWindows/CVI installs example programs. You also can visit NI Developer Zone at `ni.com/zone` for new examples.

Tip Examples are an excellent way to get started. Look for an example similar to what you need and modify it according to your specifications.

Customer Education

For additional training, NI offers hands-on LabWindows/CVI courses to help you quickly master the environment and develop successful applications. Visit `ni.com/custed` for information on courses, training, technical workshops, and more.

A

Technical Support and Professional Services

Visit the following sections of the National Instruments Web site at `ni.com` for technical support and professional services:

- **Support**—Online technical support resources include the following:

    - **Self-Help Resources**—For immediate answers and solutions, visit our extensive library of technical support resources available in English, Japanese, and Spanish at `ni.com/support`. These resources are available for most products at no cost to registered users and include software drivers and updates, a KnowledgeBase, product manuals, step-by-step troubleshooting wizards, hardware schematics and conformity documentation, example code, tutorials and application notes, instrument drivers, discussion forums, a measurement glossary, and so on.

    - **Assisted Support Options**—Contact NI engineers and other measurement and automation professionals by visiting `ni.com/ask`. Our online system helps you define your question and connects you to the experts by phone, discussion forum, or email.

- **Training**—Visit `ni.com/custed` for self-paced tutorials, videos, and interactive CDs. You also can register for instructor-led, hands-on courses at locations around the world.

- **System Integration**—If you have time constraints, limited in-house technical resources, or other project challenges, NI Alliance Program members can help. To learn more, call your local NI office or visit `ni.com/alliance`.

If you searched `ni.com` and could not find the answers you need, contact your local office or NI corporate headquarters. Phone numbers for our worldwide offices are listed at the front of this manual. You also can visit the Worldwide Offices section of `ni.com/niglobal` to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.