NATIONAL INSTRUMENTS™
LabVIEW™

Real-Time User Manual

April 2001 Edition
Part Number 322154C-01

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information

ni.com

National Instruments Corporate Headquarters

11500 North Mopac Expressway Austin, Texas 78759-3504 USA Tel: 512 794 0100

Worldwide Offices

Australia 03 9879 5166, Austria 0662 45 79 90 0, Belgium 02 757 00 20, Brazil 011 284 5011,
Canada (Calgary) 403 274 9391, Canada (Ottawa) 613 233 5949, Canada (Québec) 514 694 8521,
China (Shanghai) 021 6555 7838, China (ShenZhen) 0755 3904939, 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 91805275406, Israel 03 6120092, Italy 02 413091, Japan 03 5472 2970,
Korea 02 596 7456, Mexico 5 280 7625, Netherlands 0348 433466, New Zealand 09 914 0488,
Norway 32 27 73 00, Poland 0 22 528 94 06, Portugal 351 1 726 9011, Singapore 2265886, 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 Resources* appendix. To comment on the documentation, send e-mail 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

Trademarks

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.

Contents

Chapter 4
Real-Time Programming

Appendix A
Technical Support Resources

Glossary

Index

About This Manual

This manual contains introductory and installation information about LabVIEW Real-Time (RT) software and describes how to use the LabVIEW RT software.

Conventions

The following conventions appear 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 icon denotes a tip, which alerts you to advisory information.

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

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.

`monospace italic` Italic text in this font denotes text that is a placeholder for a word or value that you must supply.

Related Documentation

The following documents contain information that you might find helpful as you read this manual:

- your target-specific RT Series hardware user manual
- *LabVIEW Real-Time Release Notes*
- *LabVIEW Real-Time Help,* available by selecting **Help»LabVIEW Real-Time Help**
- *Getting Started with LabVIEW*
- *LabVIEW User Manual*
- *LabVIEW Measurements Manual*
- *LabVIEW Help*, available by selecting **Help»Contents and Index**
- *LabVIEW Application Builder Release Notes*

1 Introduction

Most LabVIEW applications run on general-purpose operating systems like Microsoft Windows. Some LabVIEW applications require deterministic real-time performance that non-real-time operating systems like Windows cannot guarantee. LabVIEW Real-Time (RT) addresses the need for deterministic real-time performance using LabVIEW.

LabVIEW RT combines the ease of use of LabVIEW with the power of real-time systems so you can create deterministic applications using graphical programming. Real-time applications run on RT Series hardware, either plug-in data acquisition devices or networked RT Series devices. Embedded LabVIEW RT applications do not have a user interface, so they must have a host PC to generate the user interface. You create embedded, real-time LabVIEW RT applications from a host development system. The LabVIEW RT Development System runs on Windows, just like LabVIEW. You can develop code in the LabVIEW RT Development System and download real-time code to run embedded applications on a hardware target.

General-purpose operating systems can crash or hang, which causes programs to quit running. Because embedded LabVIEW RT applications run on a separate hardware platform, they do not stop executing if the host PC operating system crashes. If a crash occurs on the host PC operating system, the user interface is lost, and any other communication between an embedded LabVIEW RT application and the host PC ceases. Embedded LabVIEW RT applications continue to run, even after the host PC operating system crashes.

You can reboot the host PC without disrupting embedded LabVIEW RT applications. After you reboot the host PC, you can reestablish communication between the host PC and the embedded LabVIEW RT application. You also can design your applications to retrieve any data that was collected on RT Series hardware while the host PC was not in communication with the embedded LabVIEW RT application.

Plug-In RT Series DAQ Devices

RT Series Data Acquisition (DAQ) devices combine data acquisition with the processing power of a computer. You can install the devices in a host PC or PXI system. The devices include a processor on which to embed LabVIEW RT applications. Part of the onboard memory (RAM) on the RT Series device is accessible to both the host PC and the RT Series device. The host PC and the RT Series DAQ device communicate by using the shared memory.

Refer to the *Real-Time Series DAQ Device User Manual* for more information about the RT Series DAQ devices.

Networked RT Series Devices

The networked RT Series devices communicate to a host PC or PXI system through an ethernet connection. You can use the host PC or PXI system to embed LabVIEW RT applications on the networked RT Series devices. The host PC communicates with the networked RT Series device through the network connection.

Refer to the appropriate RT Series hardware user manual for more information about the networked RT devices.

Architecture of LabVIEW RT

LabVIEW RT consists of the following three components: LabVIEW, the RT Development System, and the RT Engine, shown in Figure 1-1. LabVIEW and the RT Development System are different modes of the same executable, `labview.exe`. LabVIEW RT gives the LabVIEW executable the ability to target different processors for execution. `labview.exe` is in the LabVIEW mode when targeted to the Windows PC it is installed on. When in this mode, `labview.exe` acts as normal LabVIEW for Windows. When you target any RT Series hardware, then `labview.exe` goes into RT Development System mode. Stated another way, the RT Development System and normal LabVIEW can not run on the same Windows machine at the same time because they are the same executable, just different modes of the same executable. Conversely, the RT Engine is a different executable, `emblview.exe`, that runs on specific RT Series hardware with a real-time operating system (RTOS).

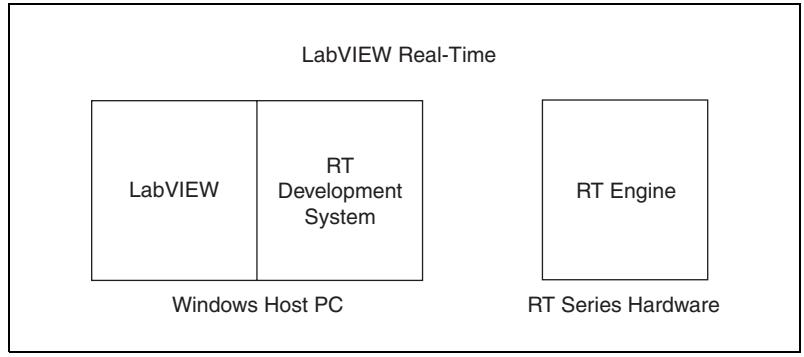

Figure 1-1. Components of LabVIEW RT with Plug-In RT Series DAQ Devices

Refer to Chapter 3, *Software Overview*, for more information about launching the RT Development System and the RT Engine.

Both the RT Development System and LabVIEW can communicate with the RT Engine. In the RT Development System mode, all VIs are executed by the RT Engine on the RT Series hardware to which the RT Development System is targeted. In other words, the RT Development System and the RT Engine are executing the same VI. The RT Development System displays the front panel of the VI while the RT Engine executes the block diagram code. The RT Development System provides the only user interface to VIs executed by the RT Engine because the RT series hardware does not load the front panel into memory.

All communication is performed through shared memory when using RT Series DAQ devices, shown in Figure 1-2, and ethernet when using networked RT Series hardware, shown in Figure 1-3. LabVIEW communicates with the RT Engine through specific programmatic control such as TCP/IP, VI Server, and, in the case of RT Series DAQ devices, shared memory reads and writes. When communicating in this way LabVIEW is executing a different VI than that being executed by the RT Engine.

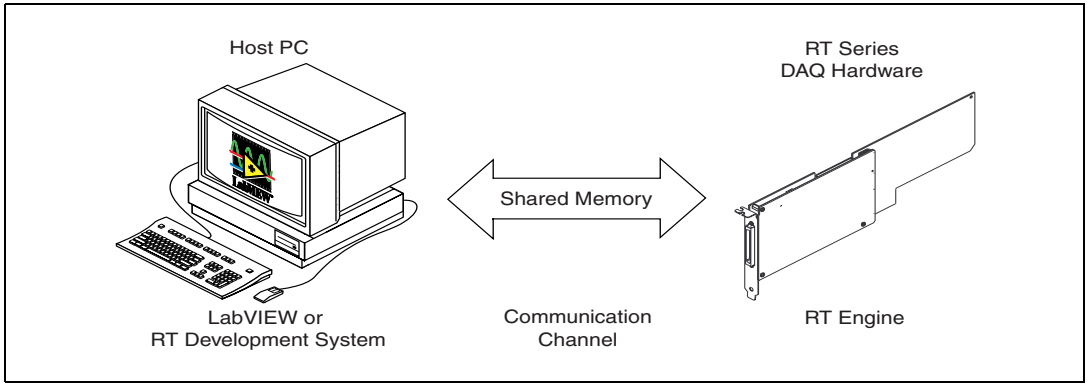

Figure 1-2. Components of LabVIEW RT with RT Series DAQ Devices

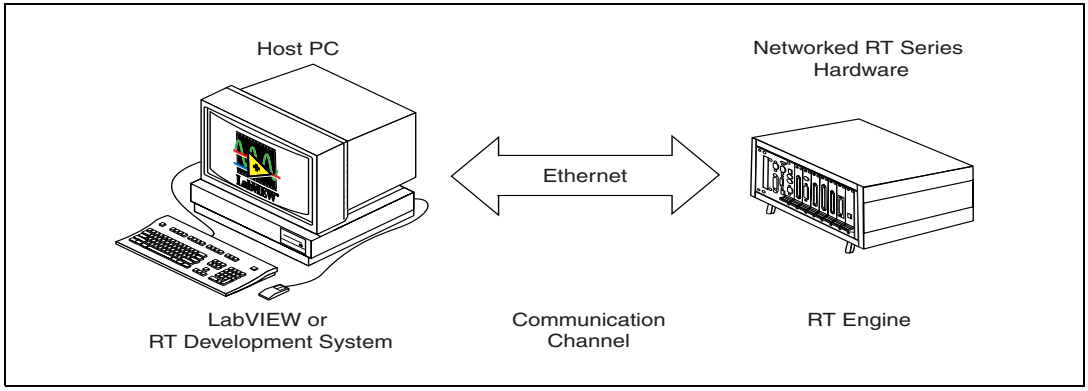

Figure 1-3. Components of LabVIEW RT with Networked RT Series Devices

Another important distinction between the three LabVIEW RT components is that LabVIEW can run independently from the other components whereas the RT Development System is dependent on a corresponding RT Engine running on an RT Series device.

RT Development System

The RT Development System is a Windows application that runs on the host PC. When you boot the RT Series target platform for the first time, it has no VIs to run. You can use the RT Development System to download VIs to the target platform.

After you download and run the VIs, you can close the RT Development System. The RT Engine continues to run the embedded VIs. You can keep the RT Development System open to show and operate the front

panel of the embedded VI. When VIs run on the target platform, the RT Development System exchanges messages with the RT Engine to update the controls and indicators on the front panel. These communications are built into the RT Development System and embedded software, so they occur automatically.

Note When you open the front panel of an embedded LabVIEW RT VI in the RT Development System, real-time performance of the VI cannot be ensured. Refer to Chapter 4, *Real-Time Programming*, for more information about real-time programming.

You can use the RT Development System to debug embedded LabVIEW RT VIs while the embedded LabVIEW RT VIs run on the target platform. You can use LabVIEW debugging tools, such as probes, breakpoints, and single stepping.

RT Engine

The RT Engine on the target platform runs the LabVIEW RT VIs you downloaded to the target platform using the RT Development System. The RT Engine can provide deterministic real-time performance for the following reasons:

- The RT Engine runs on a real-time operating system, which ensures that the scheduler and other operating system services adhere to real-time operation.
- The RT Engine does not run on the host PC but on the RT Series hardware. Unnecessary applications or device drivers, such as video drivers, do not run on the RT Series hardware. The absence of extraneous software and drivers means that a third-party application or driver does not impede LabVIEW RT applications.
- The RT Engine is tuned for real-time performance.
- The RT Series hardware uses no virtual memory. By not using virtual memory a major source of unpredictability in deterministic systems is eliminated.

System and Local Time on the RT Engine

When running on Windows, certain LabVIEW date/time functions adjust time values based on the time zone information from the operating system. Some examples of time zone adjustments from the operating system can be seen when using the **Seconds to Date/Time** function or when a chart axis marker has been formatted to show date and time. Due to limitations of the real-time operating system on RT Series devices, the RT Engine does not support time zone information. As a result, system time and local time are

treated the same in the RT Engine. Functions such as **Seconds to Date/Time** do not adjust time values based on the local time zone. Therefore, date/time functions produce different results in the RT Engine than when used in LabVIEW on the host PC.

2

Installation

This chapter explains how to install LabVIEW Real-Time (RT).

Installing the Software

Complete the following steps to install LabVIEW RT on the host PC or PXI system.

1. If you are not installing LabVIEW RT on Windows NT, proceed to step 2. If you are installing LabVIEW RT on Windows NT, log on to Windows NT as an administrator or as a user with administrator privileges.
2. Insert the LabVIEW RT CD into your CD drive. The LabVIEW RT installation program runs automatically.
3. Click **Install LabVIEW RT**. Follow the instructions that appear on your screen.

When installing LabVIEW RT for use with a networked RT Series device, you must complete the additional steps to configure and install software on the networked RT Series device. Refer to the networked RT Series device user manual for more information on configuring your device.

3

Software Overview

This chapter describes how to operate and program in LabVIEW Real-Time (RT).

Operating LabVIEW RT

This section describes the basic operating procedures for LabVIEW Real-Time (RT), such as launching LabVIEW RT, downloading VIs, debugging LabVIEW RT VIs, and creating stand-alone applications.

Launching LabVIEW RT

When you first launch LabVIEW RT, it defaults to the host PC as the platform target. Targeting the host PC makes LabVIEW RT behave like LabVIEW for Windows. Select **Operate»Switch Execution Target** to open the **Select Target Platform** dialog box, shown in Figure 3-1.

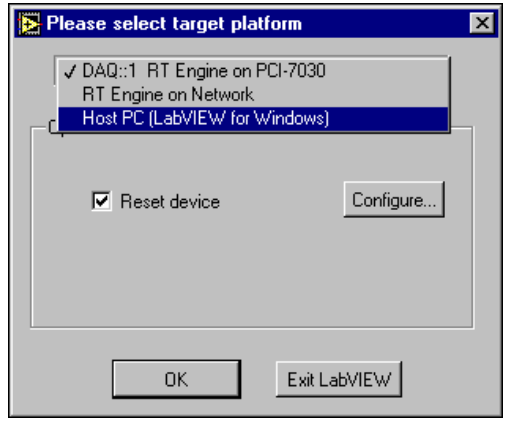

Figure 3-1. Select Target Platform Dialog Box

Use the **Select Target Platform** dialog box to target LabVIEW RT to an RT Series device. Using the pull-down menu, select where you want to run VIs. When you select a target platform, any VI you subsequently run is

downloaded to that target platform, where it can be executed. Click **OK** to launch the RT Development System.

The list of possible targets changes based on the number of RT Series DAQ devices configured on your system. If you do not configure any RT Series DAQ devices, only the **RT Engine on Network** and **Host PC** options appear in the **Select Target Platform** dialog box.

When you select an RT Engine target, various options specific to that target appear. Refer to the corresponding RT Series hardware user manual for more information about the options for a specific RT Engine target.

Complete the following steps to change your preferences so the **Select Target Platform** dialog box appears when you launch LabVIEW RT.

1. Select **Tools»Options**. The **Options** dialog box appears.
2. Select **Miscellaneous** from the pull-down menu.
3. Click the **Prompt for Target Execution Engine** checkbox.
4. Click **OK**.

With the exception of the **Prompt for Target Execution Engine** option, all LabVIEW RT options are the same as normal LabVIEW for Windows options.

Downloading VIs

When you select an RT Engine target in the **Select Target Platform** dialog box, the RT Development System establishes a connection with that RT Engine target platform. When you open VIs and click the **Run** button, the RT Development System downloads the VI and its associated subVIs to the target platform. The RT Engine on the target platform then runs the VI.

Note When you make changes to a VI, such as when you edit the VI or when you convert the VI from a different version of LabVIEW, you must save the VI on the host PC before you can download and run it on the RT Engine.

You can download LabVIEW RT VIs without running them by selecting **Operate»Download Application**. Download LabVIEW RT VIs without running them if you want to use the LabVIEW VI Server to programmatically run these VIs later.

To see which VIs have been downloaded to your target platform, select **Browse»Show VI Hierarchy** from the RT Development System. The VI hierarchy appears with a thumb tack in the upper left corner of each VI.

When the thumb tack is in the vertical position, as shown on the left, the VI has been downloaded.

When the thumb tack is in the horizontal position, as shown on the left, you need to download the VI to run it.

Because the VI Server does not download VIs, you must manually download VIs you need to call for use with the VI Server. Refer to the *Distributed Computing with the VI Server* section of this chapter, and Chapter 16, *Programmatically Controlling VIs*, of the *LabVIEW User Manual*, for more information about the VI Server.

Debugging LabVIEW RT VIs

You debug LabVIEW RT VIs in the RT Development System the same way you debug any VI. All the existing LabVIEW debugging tools, except the Call List Window, are available in the RT Development System. Refer to Chapter 4, *Debugging,* of the *Getting Started with LabVIEW* manual, and the *Debugging Techniques* section in Chapter 6, *Running and Debugging VIs*, of the *LabVIEW User Manual*, for more information about the LabVIEW debugging features.

Creating Stand-Alone Applications

Applications you build using the LabVIEW RT Professional Development System or the Application Builder can target VIs to RT Series devices or the host PC. Use the LabVIEW RT Application Builder in the same manner as the LabVIEW Application Builder, with the exception of the **Show LabVIEW RT Target selection when Launched** and the **Quit LabVIEW RT host application after downloading** options explained later in this section. Refer to your *LabVIEW Application Builder Release Notes* for more information about the LabVIEW Application Builder.

The use of the Application Builder varies depending on the platform to which you target LabVIEW RT. The information in this manual applies to using the Application Builder when targeted to the host PC. Refer to your RT Series hardware user manual for more information about using the Application Builder when targeted to the RT Series hardware.

Select **Tools»Build Application or Shared Library (DLL)** to launch the Application Builder, shown in Figure 3-2. The **Application Settings** tab contains two options in addition to the normal LabVIEW Application Builder, **Show RT target selection dialog when launched** and **Quit RT host application after downloading**.

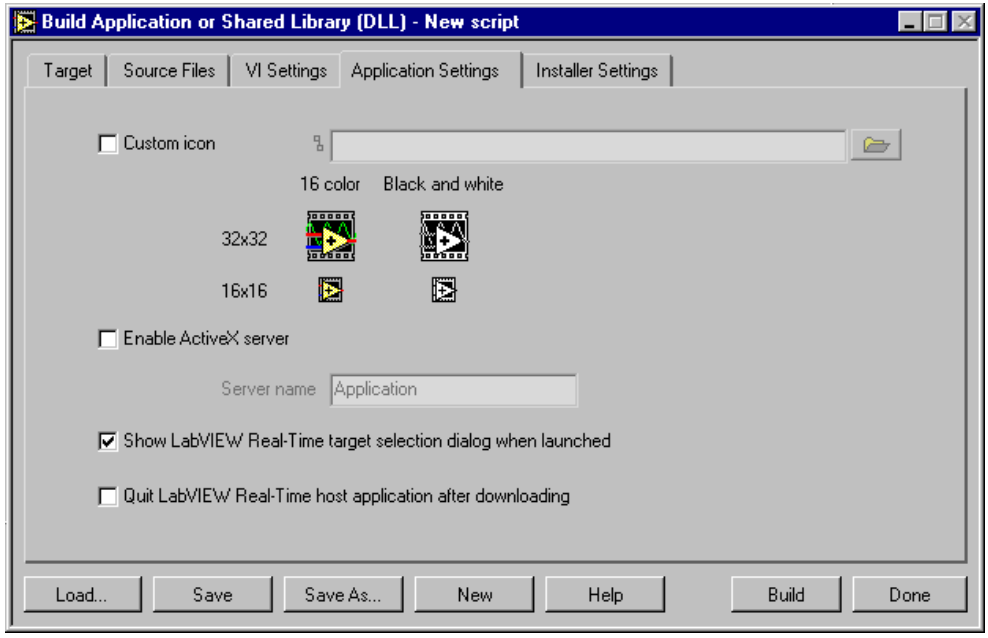

Figure 3-2. Build Application Dialog Box

When you run the LabVIEW RT application you build with the Application Builder, the **Select Target Platform** dialog box appears. You can select any target platform in your system on which to run your application. If you do not want to select the target platform each time you run the application, clear the **Show RT target selection dialog when launched** checkbox in the **Application Settings** tab of the Application Builder. This makes the application run automatically on the host PC.

If you design your application to run on the RT Engine, you might want the host PC to disconnect from the RT Engine after you download and run the VIs. Select the **Quit RT host application after downloading** checkbox in the **Application Settings** tab of the Application Builder to have the host disconnect after launching. You can use command line arguments with the applications you build to specify the target platform. Refer to the appropriate RT Series hardware user manual for more information about command line arguments.

Selecting the **Quit RT host application after downloading** option in the Application Builder is equivalent to selecting **File»Exit without closing RT Engine VIs** in the RT Development System. Refer to the *Exiting the RT Development System* section later in this chapter for more information about exiting the RT system.

In addition to building applications, the Application Builder can create an installer for your application. Select the **Installer** tab in the **Build Application or Shared Library (DLL)** dialog box to access installer options. In the **Advanced Installer Settings**, select **Install LabVIEW Run-Time Engine** to add the LabVIEW RT Run-Time Engine installer. The LabVIEW RT Run-Time Engine installer includes support for RT Series devices. You can use the LabVIEW RT Run-Time Engine to target the application to RT Series hardware from the machine on which the application is installed.

Some RT Series devices have media storage capability, such as the hard drive on a PXI controller. For these devices you can embed applications you build directly onto the RT Series device. Refer to your RT Series hardware user manual for additional information on embedded applications.

Programming LabVIEW RT

A LabVIEW RT application runs on the RT Series hardware target and controls or monitors external events through the target hardware I/O, such as analog and digital I/O channels of a DAQ device.

Because the RT Engine runs on hardware platforms that do not have all the components of a PC, the RT applications you build lack some LabVIEW features when you target the application to the RT Engine. For example, the RT Series DAQ device does not have a disk drive, so it does not support file I/O. Refer to your RT Series hardware user manual for more information about support of specific LabVIEW functions.

If you attempt to download and run a VI on your target platform that has any of the unsupported functionality, the VI still executes. Unsupported functions do not work and return standard LabVIEW error codes.

You can provide a user interface to LabVIEW RT VIs in the following two ways:

- Using the RT Development System.
- Writing LabVIEW applications that run on the host PC and communicate with the embedded application.

Using the RT Development System

Using the RT Development System is the most straightforward way to provide a user interface for embedded LabVIEW RT applications. You launch the RT Development System when you select the RT Series target device. You use the RT Development System to show the controls and indicators of the embedded LabVIEW RT VIs running on the RT Series target platform. You can interact with the VIs by changing the value of the controls, and you can see the updates to the indicators.

The RT Development System is most useful during program development and debugging. It might not be useful for running embedded real-time applications for the following reasons.

First, exchanging messages between the RT Development System and the RT Engine runs at a lower priority than a real-time algorithm. If a high-priority thread, or task, uses all the system resources of the target platform, no resources are available for lower-priority tasks, which can cause the user interface to be nonresponsive. Refer to Chapter 4, *Real-Time Programming*, for more information about priorities.

Second, in many applications you want the host PC to do more than just provide a user interface for embedded LabVIEW RT VIs. While targeted to an RT Engine the RT Development System can display the front panel of your embedded LabVIEW RT application, but it cannot execute any VIs on the host.

The following sections describe how to acquire RT Engine information, switch LabVIEW RT execution targets, connect to the RT Engine with VIs already running, and exit the RT Development System.

Acquiring RT Engine Information

Select **Operate»RT Engine Info** to view the hardware device number for RT Series DAQ devices or the IP address for networked RT Series devices to which the RT Development System is currently connected. The front panel also displays the device number or IP address in the lower left corner.

Switching LabVIEW RT Execution Targets

You can change the target platform that the RT Development System connects to by selecting **Operate»Switch Execution Target**. Changing the target platform disconnects the RT Development System from the current RT Engine or host PC and opens the **Select Target Platform** dialog box. Changing the target platform is useful when you develop embedded VIs and host PC VIs in parallel.

When the RT Development System is targeted to an RT Engine and you select **Operate»Switch Execution Target**, the RT Development System disconnects the RT Engine as if you had selected **File»Exit Without Closing RT Engine VIs**. VIs running in the target platform continue running, and VIs downloaded but not running remain loaded in the memory of the RT Engine. Refer to the *Exiting the RT Development System* section later in this chapter for more information.

Connecting to the RT Engine with VIs Already Running

If you restart LabVIEW RT and establish a connection to the target platform while embedded LabVIEW RT VIs are still running, the RT Development System detects VIs running on the target platform. The RT Development System attempts to open a copy of the embedded VIs on the host PC to show the front panel of the embedded VIs.

Note If you select the **Reset** checkbox in the **Select Target Platform** dialog box for an RT Series DAQ device, the RT Development System aborts and unloads from memory any running VIs. Therefore, the host PC does not display the front panel of these aborted VIs.

If the copies of the embedded VIs on the host PC have been moved or modified since you downloaded them to the target platform, the RT Development System displays the **Changed or Missing VIs** dialog box, shown in Figure 3-3.

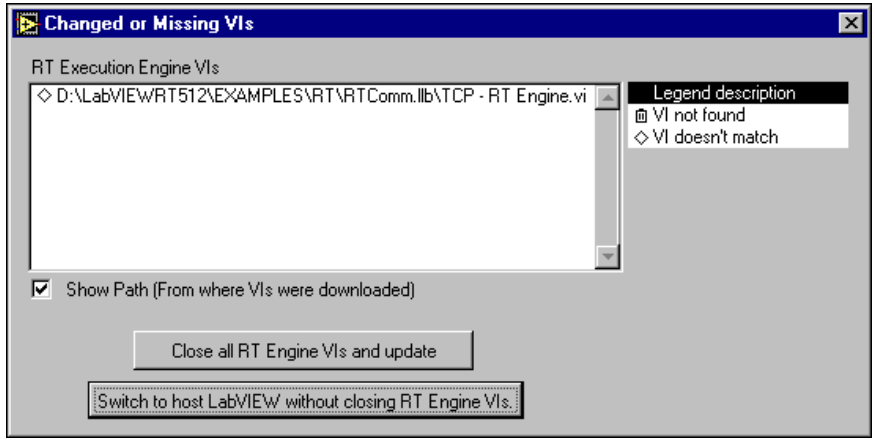

Figure 3-3. Changed or Missing VIs Dialog Box

The **Changed or Missing VIs** dialog box shows the name of the VIs and indicates that the RT Development System cannot find the VIs or that the VIs have been modified and no longer match the embedded VIs running on the target platform. From the **Changed or Missing VIs** dialog box, you can close all the VIs running in the target platform and update them all with the latest version of each VI, or you can leave the embedded VIs running and quit the RT Development System.

Exiting the RT Development System

You can exit the RT Development System without closing the embedded LabVIEW RT VIs running in the target platform by selecting **File»Exit Without Closing RT Engine VIs**. This closes the RT Development System on the host PC. The VIs running on the target platform continue running. VIs downloaded but not running remain loaded in the memory of the target platform. You can call and execute the embedded VIs later by using the VI Server from a LabVIEW application running on the host PC or a host LabVIEW application.

Select **File»Exit to** quit the RT Development System. A confirmation dialog box appears that asks if you want to shut down the RT Engine before exiting. If you click the **OK** button, the RT Development System closes all the VIs running on the target platform, unloads the VIs from memory, and shuts down the RT Engine.

Communicating Using Host LabVIEW Applications

Instead of using the RT Development System to communicate with embedded RT Engine VIs, you can write a host LabVIEW application to communicate. A host LabVIEW application can control the behavior of the RT Engine applications programmatically and save and analyze data an RT Engine VI produces. A host LabVIEW application also provides a user interface for communicating with the embedded RT Engine VI.

You can develop host LabVIEW applications with LabVIEW RT by selecting **Host PC (LabVIEW for Windows)** in the **Select Target Platform** dialog box.

You can use high-level software protocols such as TCP/IP and the VI Server to communicate between host LabVIEW applications and RT Engine VIs. Each protocol has its advantages and disadvantages. You can choose any one of the following methods based on your communication needs:

- TCP/IP
- VI Server
- Shared Memory (RT Series DAQ devices only)

TCP/IP

TCP/IP is an industry-standard protocol for communication over networks. Host LabVIEW VIs can communicate with RT Engine VIs using the LabVIEW TCP/IP VIs. Refer to the *LabVIEW Help,* available by selecting **Help»Contents and Index**, for more information about the TCP/IP VIs.

LabVIEW RT extends the capability of the existing TCP/IP VIs so you can use these VIs to communicate to networked RT Series devices and across shared memory to RT Series DAQ devices.

To use TCP/IP, you must supply the network address of your RT Series hardware. To communicate with an RT Series DAQ device, use `DAQ::`*`x`* as the network address, where *x* is the device number of the processor board that runs the LabVIEW RT code.

Distributed Computing with the VI Server

You use the VI Server to monitor and control VIs and other LabVIEW applications on a remote computer across a network. Using VI Server technology, a host LabVIEW application views the target platform as another computer on which it can invoke VIs. As the host LabVIEW application calls LabVIEW RT VIs through the VI Server, the host LabVIEW application can pass parameters in and out of the embedded LabVIEW RT VIs.

One advantage to using the VI Server for communication is that it allows you to access the functionality of TCP/IP while working within the framework of LabVIEW. Refer to Chapter 16, *Programmatically Controlling VIs,* of the *LabVIEW User Manual* for more information about the VI Server.

A disadvantage of the VI Server is that the host LabVIEW application can only call LabVIEW RT VIs that are already downloaded onto the RT Series hardware. The host LabVIEW application cannot dynamically download LabVIEW RT VIs to the target platform. The only way to download VIs to the RT Series hardware is through the RT Development System.

Shared Memory (RT Series DAQ Devices Only)

Shared memory is the physical medium in which the host PC and RT Series DAQ device communicate.

In operating systems like Windows, two processes or applications can communicate with each other using the shared memory mechanism the operating system provides. Similarly, RT Engine applications and host LabVIEW applications can communicate using shared memory VIs to read and write to the shared memory locations on the RT Series DAQ device.

Shared memory VIs have very low overhead and are ideal for time-critical, real-time applications. However, the size of the shared memory is limited to 1 kB. If you need to transfer several megabytes of data, you must break up the data into smaller portions and then transfer them. In doing so, you must make sure that data in the shared memory is not overwritten before it is read. The TCP/IP VIs are more convenient for transferring large amounts of data.

There are advantages to using TCP/IP VIs or the VI Server for communication. TCP/IP VIs and the VI Server manage flow control, so they are more convenient for bulk transfer. These methods automatically separate the transfer into smaller sizes. TCP/IP VIs and the VI Server are

also more portable and medium independent. By comparison, shared memory access is limited to communicating over shared memory.

The disadvantage of TCP/IP and the VI Server is that these methods have higher overhead than shared memory access and might not be suitable for some fast real-time applications.

Refer to the *Real-Time Series DAQ Device User Manual* for more information about the use of shared memory.

4

Real-Time Programming

This chapter describes how to program real-time VIs in LabVIEW Real-Time (RT) using the scheduling priority of a VI and the methods of communication between the target platform and host LabVIEW VIs. This chapter also outlines tips you can use to get the best possible real-time performance with your target platform. These tips can help you write high-performance, deterministic programs needed in time-critical loops.

Real-Time Performance of VIs

The RT Engine runs VIs in a deterministic manner, which means that VIs run with the same time characteristics each time you run the VI. For example, control loops execute at a consistent loop rate every iteration of the loop. To achieve optimal real-time performance, you need to understand the following programming concepts.

Time-Critical Priority

In a real-time operating system like the one used to power the embedded RT Engine, events are prioritized and higher priority events are executed over lower priority events. Therefore, VI priorities and threads become very important in developing a real-time application. Using priority settings incorrectly can result in VIs that use all of the processor resources, displacing other lower priority tasks. Low priority tasks and threads, like communication, or the user interface thread, may not receive enough resources to run.

Complete the following steps to set the priority of a VI.

1. Right-click the icon and connector pane on the front panel and select **VI Properties** from the shortcut menu.
2. Select **Execution** from the **Category** pull-down menu.
3. Select **time critical priority (highest)** in the **Priority** ring control pull-down menu.

Real-Time Features of the LabVIEW RT Environment

Setting a VI to run at time-critical priority can have unexpected consequences. Running a VI at higher priority than events in the communication thread means a VI gets more processor time and resources than the communication thread. Although the RT Engine is a multithreaded environment, running a VI at time-critical priority can cause it to monopolize the target embedded processor, making other threads unable to run.

The front panel updates on the host RT Development System and VI Server or TCP/IP communications run as separate threads on the target embedded processor. Therefore, a VI running at time-critical priority might prevent these threads from running. Other VIs running on the RT target platform also might be affected.

The Wait Until Next ms Multiple and the Wait (ms) functions located on the **Functions»Time and Dialog** palette cause the execution thread of the VI to sleep for a period of time so other threads, such as the communication thread, have a chance to run. In general, use the Wait Until Next ms Multiple function instead of the Wait (ms) function, becuase it can be used to set a known loop time. For example, placing the Wait Until Next ms Multiple function in a While loop inside a VI with a constant of `5` wired to the loop makes the VI attempt to execute the loop once every 5 ms, or with a loop rate of 200 Hz. If the VI is running at time-critical priority, such a loop rate is guaranteed, but the loop must complete in the specified amount of time. The processor uses any time left over to run other tasks. If no extra time remains, no other threads run. This means that even a VI that waits can monopolize too much processor time to yield and let other threads run.

Running a VI at Time-Critical Priority in the RT Development System

If you run a time-critical priority VI that does not wait, it monopolizes the processor on the target platform. Running VIs that do not wait can prevent you from interacting with the VI front panel and prevent you from stopping the VI using the **Abort** button. Although not a problem with the VI or LabVIEW RT, this expected behavior means that the target platform is too busy to communicate with the RT Development System while the time-critical VI is running. Because the time-critical loop has the highest priority and does not wait, the user interface thread that updates front panel objects cannot use any processor time. As a result, the front panel of the VI appears locked even though the VI is still running. The RT Development System displays a dialog box that informs you communication between the RT Development System and the RT Engine has been lost. Click **Abort** to close the RT Development System. You must reset the RT Engine before

communication can be re-established. To avoid this behavior during development and testing of your VI, set the VI to normal priority. After you complete development, you can set the priority to the appropriate level.

Running a VI that waits at time-critical priority is often acceptable during development, because the RT Development System can use the time while the time-critical thread is sleeping to perform communication. However, if the VI is computation intensive and is unable to complete in the time allotted to it, no wait occurs, and the RT Development System is unable to communicate with the target platform. If the loop completes but leaves only a small fraction of the time available on the processor to the RT Development System communication threads, interaction with that VI can seem slow or nonresponsive.

Running a VI at Time-Critical Priority without the RT Development System

You can improve the performance of your RT Engine VIs by running them on the target platform without the RT Development System. Disconnect the RT Development System from the RT Engine by selecting **Operate»Switch Execution Target** or **File»Exit without closing RT Engine VIs**. The time critical VIs on the target platform then need to yield only as much time as is required for its TCP/IP or VI Server communication, which can be less time than required for the full RT Development System to communicate.

Performance of LabVIEW RT VIs

You can use many different programming paradigms to increase the performance of any LabVIEW program. These paradigms have a special importance with LabVIEW RT, because performance and deterministic real-time requirements are often related. In addition, because priorities and communication are of major importance in LabVIEW RT, understanding them is critical to high-performance programming.

Writing Efficient Loops

This section describes various ways you can write more efficient loops.

Improving DAQ Configuration

The most time-intensive part of a loop in any DAQ operation on the RT Engine is the configuration. If possible, use the intermediate or advanced DAQ VIs to move configuration outside of your main loop. If you use the Basic DAQ VIs, wire the loop iteration number to the VI so configuration happens only once. In addition, if you place a control or indicator within a loop and the VI runs in the RT Development System, the VI tries to communicate between the RT Engine and the RT Development System running on the host PC. This communication does not affect performance if the VI is running at time-critical priority, but it decreases the performance of that communication, making the RT Development System seem slow or nonresponsive.

Removing Redundant Operations

Keep redundant or unnecessary computations out of the loop where you need performance. For example, when you use the RT Series DAQ devices, LabVIEW RT provides a set of VIs that use peek and poke to pass an array of data between the host application and an RT Series DAQ device by passing only a single element with each iteration. This distribution of the cost to transfer an array often improves performance, because only one read or write is necessary per loop iteration rather than several. For an example of this technique of array data passing, run the One Channel PID VI, available from `examples\rt\RT Control.llb`.

Setting the Priorities of SubVIs

One of the priority options available for subVIs is subroutine. A subVI at subroutine priority requires less overhead than a normal subVI and therefore runs smoother and faster. After you test your subVI, set its priority to **subroutine**. Remember that your highest level VI cannot have subroutine priority. Set your highest level VI priority to **time critical**.

Avoiding Array Copying

Array copying requires a linear, rather than a constant, amount of time. One way to avoid array copying is to pass initialized arrays of the expected size into a control loop rather than create arrays using the Build Array VI within the loop.

Test and experiment several times to determine exactly how long each iteration of the loop takes. If possible, try to pause the maximum amount of time in each Wait function to enable the communication with the RT target platform to be as responsive as possible.

Tip Use the Benchmark Shell VI located in the `examples\rt\RT Tutorial.llb` directory to accurately measure the time it takes your VI to execute.

A

Technical Support Resources

Web Support

National Instruments Web support is your first stop for help in solving installation, configuration, and application problems and questions. Online problem-solving and diagnostic resources include frequently asked questions, knowledge bases, product-specific troubleshooting wizards, manuals, drivers, software updates, and more. Web support is available through the Technical Support section of `ni.com`

NI Developer Zone

The NI Developer Zone at `ni.com/zone` is the essential resource for building measurement and automation systems. At the NI Developer Zone, you can easily access the latest example programs, system configurators, tutorials, technical news, as well as a community of developers ready to share their own techniques.

Customer Education

National Instruments provides a number of alternatives to satisfy your training needs, from self-paced tutorials, videos, and interactive CDs to instructor-led hands-on courses at locations around the world. Visit the Customer Education section of `ni.com` for online course schedules, syllabi, training centers, and class registration.

System Integration

If you have time constraints, limited in-house technical resources, or other dilemmas, you may prefer to employ consulting or system integration services. You can rely on the expertise available through our worldwide network of Alliance Program members. To find out more about our Alliance system integration solutions, visit the System Integration section of `ni.com`

Worldwide Support

National Instruments has offices located around the world to help address your support needs. You can access our branch office Web sites from the Worldwide Offices section of `ni.com`. Branch office Web sites provide up-to-date contact information, support phone numbers, e-mail addresses, and current events.

If you have searched the technical support resources on our Web site and still cannot find the answers you need, contact your local office or National Instruments corporate. Phone numbers for our worldwide offices are listed at the front of this manual.

Glossary

Prefix	Meanings	Value
m-	milli-	10^{-3}
k-	kilo-	10^{3}
M-	mega-	10^{6}
G-	giga-	10^{9}

A

address — character code that identifies a specific location (or series of locations) in memory

D

DAQ — data acquisition—(1) collecting and measuring electrical signals from sensors, transducers, and test probes or fixtures and inputting them to a computer for processing; (2) collecting and measuring the same kinds of electrical signals with A/D and/or DIO boards plugged into a computer, and possibly generating control signals with D/A and/or DIO boards in the same computer

default setting — a default parameter value recorded in the driver. In many cases, the default input of a control is a certain value (often 0) that means *use the current default setting*. For example, the default input for a parameter may be *do not change current setting*, and the default setting may be *no AMUX-64T boards*. If you do change the value of such a parameter, the new value becomes the new setting. You can set default settings for some parameters in the configuration utility or manually using switches located on the device.

device	a plug-in data acquisition board, card, or pad that can contain multiple channels and conversion devices. Plug-in boards, PCMCIA cards, and devices such as the DAQPad-1200, which connects to your computer parallel port, are all examples of DAQ devices. SCXI modules are distinct from devices, with the exception of the SCXI-1200, which is a hybrid.
drivers	software that controls a specific hardware device such as a DAQ board or a GPIB interface board

H

h	hour
hardware	the physical components of a computer system, such as the circuit boards, plug-in boards, chassis, enclosures, peripherals, and cables
host PC	the computer on which the LabVIEW RT Development System is used to develop and target VIs to RT Series hardware
Hz	hertz—the number of scans read or updates written per second

I

I/O	input/output—the transfer of data to/from a computer system involving communications channels, operator interface devices, and/or data acquisition and control interfaces

K

k	kilo—the standard metric prefix for 1,000, or 10^3, used with units of measure such as volts, hertz, and meters
K	kilo—the prefix for 1,024, or 2^{10}, used with B in quantifying data or computer memory
kbytes/s	a unit for data transfer that means 1,000 or 10^3 bytes/s

L

LabVIEW	laboratory virtual instrument engineering workbench
library	a file containing compiled object modules, each comprised of one of more functions, that can be linked to other object modules that make use of these functions. NIDAQMSC.LIB is a library that contains NI-DAQ functions. The NI-DAQ function set is broken down into object modules so that only the object modules that are relevant to your application are linked in, while those object modules that are not relevant are not linked.

M

m	meters
M	(1) Mega, the standard metric prefix for 1 million or 10^6, when used with units of measure such as volts and hertz; (2) mega, the prefix for 1,048,576, or 2^{20}, when used with B to quantify data or computer memory
MIO	multifunction I/O

N

NI-DAQ	National Instruments driver software for DAQ hardware

O

operating system	base-level software that controls a computer, runs programs, interacts with users, and communicates with installed hardware or peripheral devices

P

PCI	Peripheral Component Interconnect—a high-performance expansion bus architecture originally developed by Intel to replace ISA and EISA. It is achieving widespread acceptance as a standard for PCs and work-stations; it offers a theoretical maximum transfer rate of 132 Mbytes/s.

PID control	a three-term control mechanism combining proportional, integral, and derivative control actions. Also see proportional control, integral control, and derivative control.
protocol	the exact sequence of bits, characters, and control codes used to transfer data between computers and peripherals through a communications channel, such as the GPIB bus

R

RAM	random-access memory
real time	a property of an event or system in which data is processed as it is acquired instead of being accumulated and processed at a later time

S

s	seconds
S	samples
shared memory	memory that can be sequentially accessed by more than one controller or processor but not simultaneously accessed. Also known as dual-mode memory.
soft reboot	restarting a computer without cycling the power, usually through the operating system
synchronous	(1) hardware—a property of an event that is synchronized to a reference clock; (2) software—a property of a function that begins an operation and returns only when the operation is complete

U

update	the output equivalent of a scan. One or more analog or digital output samples. Typically, the number of output samples in an update is equal to the number of channels in the output group. For example, one pulse from the update clock produces one update which sends one new sample to every analog output channel in the group.

V

V	volts
VI	virtual instrument—(1) a combination of hardware and/or software elements, typically used with a PC, that has the functionality of a classic stand-alone instrument; (2) a LabVIEW software module (VI), which consists of a front panel user interface and a block diagram program

Index

M

N

O

P

R

S

T

V

W