LabVIEW™ RT

User Manual

Worldwide Technical Support and Product Information

`www.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 (Ontario) 905 785 0085, Canada (Québec) 514 694 8521,
China 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 (D.F.) 5 280 7625,
Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466, Norway 32 27 73 00, Poland 48 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 2377 1200, 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

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

LabVIEW™, National Instruments™, ni.com™, NI-DAQ™, and PXI™ are trademarks of National Instruments Corporation.

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

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

Figures

# Activities

About This Manual

This manual contains introductory and installation information on LabVIEW RT software and describes how to use the LabVIEW RT software.
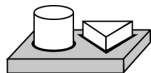
Conventions

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.
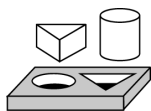
This icon denotes the beginning of an activity.

This icon denotes the end of an activity.

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.

`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 RT Release Notes*
- *LabVIEW QuickStart Guide*
- *LabVIEW User Manual*
- *G Programming Reference Manual*
- *LabVIEW Function and VI Reference Manual*
- *LabVIEW Online Reference*, available by selecting **Help»Online Reference**
- *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. National Instruments created LabVIEW RT to address 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 generate deterministic applications using graphical programming. These 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 these 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 all code in this environment 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. However, 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 LabVIEW RT. You also can design your applications so you can retrieve any data that was collected on RT Series hardware while the host PC was not in communication with the LabVIEW RT application.

Plug-In RT Series DAQ Devices

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

Refer to the *RT Series DAQ Device User Manual* for more information about the RT Series DAQ boards.

Networked RT Series Devices

The current networked RT Series device is a PXI Controller. The networked RT Series device communicates to a host PC or PXI system through an ethernet connection. You can use the host PC or the 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 user manual for more information about the networked RT devices.

Architecture of LabVIEW RT

Figure 1-1 and Figure 1-2 show the components of LabVIEW RT. In this architecture, LabVIEW RT applications run on RT Series hardware.
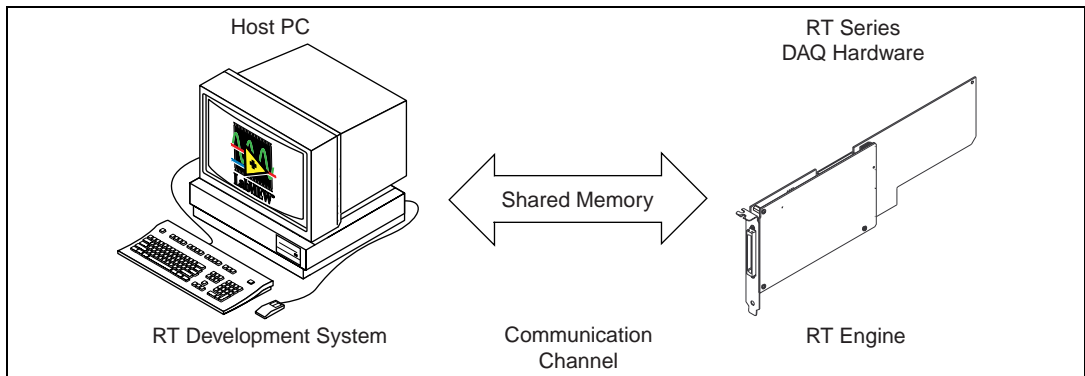
Figure 1-1. Components of LabVIEW RT with Plug-In RT Series DAQ Devices
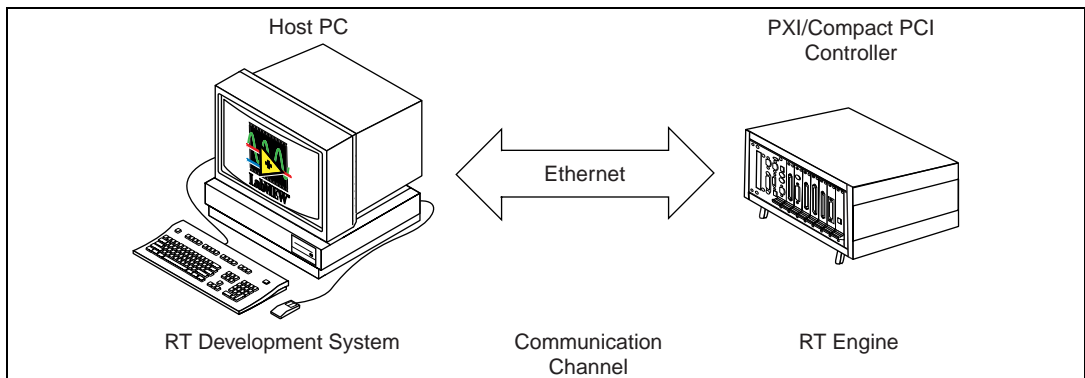

Figure 1-2. Components of LabVIEW RT with Networked RT Series Devices

LabVIEW RT consists of two components, the *RT Development System* and the *RT Engine*, which communicate with each other through the built-in *communication channel*. The communication channel for RT Series DAQ devices is the shared memory, as shown in Figure 1-1. The communication channel for networked RT Series devices is the ethernet connection. The RT Development System, running on the host PC, initializes the RT Series device with the RT Engine, downloads VIs to the RT Series device, and provides a user interface on the host PC for debugging LabVIEW RT applications. The RT Engine runs on the target RT Series device and executes the LabVIEW RT VIs in real time.

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

LabVIEW RT Development System

The LabVIEW 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.

- The RT Engine is tuned for real-time performance, and bottlenecks are reduced or eliminated.

- The RT Series hardware uses no virtual memory and thereby removes a major source of unpredictability in deterministic systems.

System Time/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. This can be seen when using the **Seconds to Date/Time** function or when a chart axis marker is 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 as being 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, these 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 RT.

Software Installation

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

⚠️ **Caution** LabVIEW RT is available in English only. If you upgrade from a French, German, or Japanese version of LabVIEW to LabVIEW RT, your version of LabVIEW will be in English.

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.

📝 **Note** LabVIEW updaters that upgrade LabVIEW to a newer version do not work with LabVIEW RT. Use only a LabVIEW RT updater to upgrade your version of LabVIEW RT.

4. When prompted during the LabVIEW RT installation, select to install NI-DAQ.

5. Follow the installation instructions that appear on your screen for NI-DAQ.

When installing LabVIEW RT for use with an RT Series DAQ device, you do not need to follow additional steps. The RT Engine automatically downloads to the RT Series DAQ device when you target and reset the device. Refer to the *RT Series DAQ Device User Manual* for more information about installing your RT Series DAQ device.

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

3

Software Overview

This chapter describes how to operate and program in LabVIEW RT. This chapter also includes three activities to help you get started with LabVIEW RT.

Operating LabVIEW RT

This section describes the basic operating procedures for LabVIEW RT, such as launching LabVIEW RT, downloading VIs, debugging LabVIEW RT VIs, and creating stand-alone executables.

Launching LabVIEW RT

When you first launch LabVIEW RT, the RT Development System is targeted to the host PC. This makes LabVIEW RT behave like normal 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

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

1. Select **Edit»Preferences** to access the **Preferences** dialog box.

2. Select **Miscellaneous** from the pull-down menu.

3. Click the **Prompt for Target Execution Engine** checkbox.

4. Click **OK**.

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

You can use the **Select Target Platform** dialog box to target LabVIEW RT to an RT Series device.

Use the pull-down menu to 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. Selecting **Host PC (LabVIEW for Windows)** makes LabVIEW RT behave like normal LabVIEW for Windows on the host PC.

Note   The list of possible targets changes based on the number of RT Series DAQ boards configured on your system. If you do not configure any RT Series DAQ boards, 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 information on the options for a specific RT Engine target.

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. From this point on, 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. Do this if you want to use the LabVIEW
VI Server to run these VIs later, programmatically.

To see which VIs have been downloaded to your target platform, select
Project»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 VI Server section later in this chapter, and
Chapter 21, *VI Server*, of the *G Programming Reference Manual*, for more
information about the VI Server.

Debugging LabVIEW RT VIs

You debug LabVIEW RT VIs the same way you debug any VI. All the
existing LabVIEW debugging tools, except the **Call List Window**, are
available with LabVIEW RT. Refer to Chapter 5, *Debugging,* in the
LabVIEW QuickStart Guide, and the *How Do You Debug a VI?* section in
Chapter 2, *Creating VIs*, of the *LabVIEW User Manual*, for more
information about the LabVIEW debugging features.

Creating Stand-Alone Executables

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 following
two options. 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 **Project»Build Application** to launch the Application Builder, shown in Figure 3-2. The **App 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 **App 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 **App 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.

Using the **Quit RT host application after downloading** option in the application you build 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.

In addition to building applications, the Application Builder can create an installer for your application. Select the **Installer** tab in the **Build Application** dialog box to access installer options. In the Advanced Installer settings, you can add additional installations such as the LabVIEW Run-Time Engine. When using the Application Builder in LabVIEW RT, you can use another additional installation option for the LabVIEW Real Time Engine. Add this installation if you want 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 board.

Because LabVIEW RT 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 board 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 deploying 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 might want to do more than just display data from embedded LabVIEW RT VIs. The RT Development System can display the front panel of your embedded LabVIEW RT application, but it cannot execute any additional code.

The following sections describe how to get RT Engine information, exit the RT Development System, connect to the RT Engine with VIs already running, and exit the RT Development System.

Getting RT Engine Information

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 on Windows is currently connected, select **Operate»RT Engine Info**. 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**. Doing this 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.
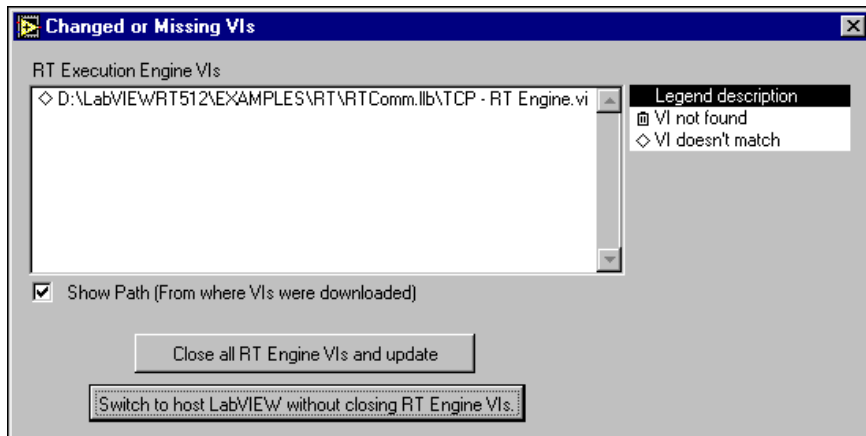
Figure 3-3. Changed or Missing VIs Dialog Box

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 (they do not 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 affecting 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 VI later on by using the VI Server from a LabVIEW application running on the host PC, or a host LabVIEW application.

When you quit the RT Development System by selecting **File»Exit**, a confirmation dialog box appears that prompts you to confirm that you want to shut down the RT Engine before exiting. If you select **OK**, the RT Development System closes all the VIs running on the target platform, unloads the VIs from memory, and shuts down the RT Engine.
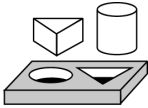
Activity 3-1. Use the RT Development System as a User Interface

Your objective is to run a VI on the target platform using the RT Development System as your user interface.

1. Start LabVIEW RT. Select **Operate»Switch Execution Target** to access the **Select Target Platform** dialog box.

2. From the **Select Target Platform** dialog box, select your target platform. Click **OK**.

3. Open the Tank Simulation VI located in `Examples\Apps\ tankmntr\Tank Simulation.vi`. Click the **Run** button.

    The RT Development System downloads the Tank Simulation VI to the RT Engine on the target platform, which runs the VI. Experiment with the control values to control the Tank Simulation VI.

4. Exit LabVIEW RT by selecting **File»Exit Without Closing RT Engine VIs**.

    This exits the RT Development System but leaves the embedded Tank Simulation VI running on the target platform.

5. Start LabVIEW RT again. From the **Select Target Platform** dialog box, select the same target platform. Clear the **Reset** checkbox if you target an RT Series DAQ device. Click **OK**.

    Because you chose to exit without closing the embedded VIs, LabVIEW RT establishes communication with the target platform, opens the Tank Simulation VI in the RT Development System on the host PC, and provides the RT Development System with the most recently updated data from the embedded LabVIEW RT VI still running on the target platform.

6. Exit LabVIEW RT by selecting **File»Exit**. Do not save changes.

    This closes LabVIEW RT and stops and closes the Tank Simulation VI on the target platform.
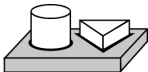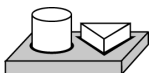
End of Activity 3-1.
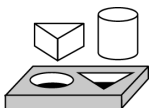
Activity 3-2. Interaction of Embedded VI with Local Copy

Your objective is to use the Changed or Missing VIs dialog box.

1. Start LabVIEW RT and select your target platform from the **Select Target Platform** dialog box.

2. Select **File»Open** and open the TCP - RT Engine VI from `Examples\RT\RTComm.llb`.

3. Select **File»Save As** and save the VI to your desktop.

4. Download the VI by selecting **Operate»Download Application**. Do not run the VI.

5. Select **File»Exit Without Closing RT Engine VIs**.

6. Delete the VI you just saved to the desktop.

7. Launch LabVIEW RT and select the appropriate target platform from the **Select Target Platform** dialog box. Clear the **Reset** checkbox if you target an RT Series DAQ device. Because the RT Development System cannot find the local copy of the VI, it prompts you to locate it. Click **Cancel**. The **Changed or Missing VIs** dialog box appears.

    When the RT Development System detects that there is an embedded VI on the target platform, it attempts to open the front panel from the local copy of the VI on the host PC hard drive. If the RT Development System detects a change in version or cannot locate the host copy of the VI, the **Changed or MIssing VI** dialog box appears. When the **Changed or Missing VI** dialog box appears you have the option to close all the embedded VIs and update them with the copy on the host PC or to exit the RT Development System and not update the embedded VIs.

8. Click the **Close all RT Engine VIs and Update** button in the **Changed or Missing VIs** dialog box.

9. Exit LabVIEW RT by selecting **File»Exit**. This closes LabVIEW RT.

End of Activity 3-2.

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. Selecting this option makes LabVIEW RT behave like regular LabVIEW for Windows on the host PC.

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 drawbacks. 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 Chapter 48, *TCP VIs*, of the *Function and VI Reference Manual* 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 where x is the device number of the processor board that runs the LabVIEW RT code.

Distributed Computing with VI Server

You use the VI Server to monitor and control VIs and other executables 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 21, *VI Server,* of the *G Programming Reference Manual* for more information about the VI Server.

A drawback 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 hardware 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 hardware.

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 drawback 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 your *RT Series DAQ Device User Manual* for more information about the use of shared memory.
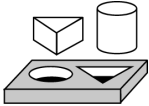
Activity 3-3. Using a Host LabVIEW Application as User Interface

Your objective is to use a separate host application as the user interface to your embedded application. This example uses a server VI and a client VI. The server VI runs on the target platform and communicates with the client VI that runs on the host PC. The client VI provides the user interface.

1. Start LabVIEW RT. Select **Operate»Switch Execution Target** to access the **Select Target Platform** dialog box.

2. From the **Select Target Platform** dialog box, select your RT Series device. Click **OK**.

3. Open the Simple Data Server VI, found in `Examples\Comm\tcpex` and click the **Run** button.

   The RT Development System automatically downloads the Simple Data Server VI to the target platform, which runs the VI.

4. Select **Operate»Switch Execution Target**. This disconnects the RT Development System from the target platform but leaves the TCP/IP Server VI running on the RT Engine.

5. In the **Select Target Platform** dialog box, select **Host PC (LabVIEW for Windows)** to use LabVIEW on Windows to run VIs rather than the RT Engine on the RT Series device. Click **OK**.

6. Open the Simple Data Client VI, found in `Examples\Comm\tcpex`.

7. On the Simple Data Client VI front panel, you must change the **Address** control from `localhost` to the address of the RT Series device. For RT Series DAQ devices, use `DAQ::`x, where x is the device number of the desired RT Series DAQ device on your system. For networked RT Series devices, use the IP address of your networked RT Series device.

Tip   When you access an RT Series DAQ device, the device number you use depends on what type of operation you are attempting. When you are communicating from the host PC to an embedded VI, use the device number for the processor board. When you are performing DAQ operations, use the device number for the DAQ daughter board. When you are accessing shared memory from an embedded VI on the RT Engine, use device number = 0.

8. Click the **Run** button.

   Because you selected the Host PC rather than the RT Engine in the **Select Target Platform** dialog box, the Simple Data Client VI runs on the host PC. It receives and displays data through TCP/IP from the Simple Data Server VI running on the RT Series device.

9. Select **File»Exit** to exit LabVIEW RT. Do not save your changes.

   This closes LabVIEW RT and stops and closes the server and client VIs on the target platform.

End of Activity 3-3.

4
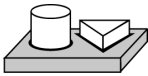
Real-Time Programming

This chapter describes how to program real-time VIs in LabVIEW 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 out of the target platform. These tips can help you write high-performance, deterministic programs needed in time-critical loops. This chapter also includes an activity to help you learn priorities.

Real-Time Performance of VIs

LabVIEW RT 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. Failure to use good priority settings can result in VIs that use all the processor resources and starve other lower priority tasks. These low priority tasks and threads, like communication, or the user interface thread, may not even be allowed to run.

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

1. Right-click on the VI icon on the front panel and select **VI Setup**.

2. Select **Execution Options** from the pull-down menu.

3. Select **time critical priority (highest)** in the priority ring control.

Real-Time Features of the LabVIEW RT Environment

Because running a VI at higher priority than events in the communication channel means a VI gets more processor time and resources than the communication channel, setting a VI to run at time-critical priority can have unexpected consequences. 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 the VI Server or TCP/IP communications run as separate threads on the target embedded processor. Therefore, a VI running at time-critical priority may prevent these threads from running. Other VIs running on the RT target platform may also be affected.

The Wait Until Next ms Multiple and the Wait (ms) functions, found in the **Function»Time and Dialog** palette, cause the execution thread of the VI to "sleep," or pause, 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 VI instead of the Wait (ms) VI, since it can be used to set a known loop time. For example, placing the Wait Until Next ms Multiple VI 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, assuming that the loop can complete in the specified amount of time. The processor uses any time left over to run other tasks. If there is no extra time left over, no other threads run. In this way, 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. This can prevent you from interacting with the VI front panel and prevent you from stopping the VI using the **Abort** button. There is no problem with the VI or LabVIEW RT if this occurs. This expected behavior signifies that the target platform is too busy to communicate with the RT Development System while the time-critical VI is 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. From this dialog box, click the **Abort** button 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 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. Even 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 sluggish or nonresponsive.

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

One way to improve performance of your VIs is to run the VIs on the target platform without the RT Development System. In other words, disconnect LabVIEW RT from the RT Engine selecting **Operate»Switch Execution Target** or **File»Exit without closing RT Engine VIs**. The VI then needs to yield only as much time as is required for its TCP/IP or VI Server communication, which can be substantially less time than required for the full RT Development System to communicate.
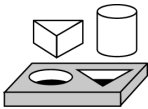
Activity 4-1.   Learning Priorities

Your objective is to observe the effects of changing the priority setting of a VI.

1. Launch LabVIEW RT and select the appropriate RT Series device from the **Select Target Platform** dialog box. Close any VIs already running on the target platform.

2. For RT Series DAQ devices, select the **Reset** checkbox to reboot the hardware.

3. Click the **OK** button.

4. Select **File»Open** and open the Priority Trouble VI from `Examples\RT\RTTutorial.llb`.

5. Right-click on the VI icon on the front panel and select **VI Setup**.

6. Select **Execution Options** from the pull-down menu. Notice that the priority is set to **time critical priority (highest)**. This makes the VI run in the highest priority thread on the RT Engine.

7. On the front panel, set the **msec to wait** control to greater than 0, 10 is the default. Run the VI. The chart displays a sine wave.

8. Slowly decrement the **msec to wait** control. With each step, the chart display speeds up. However, when the **msec to wait** control reaches zero, the front panel no longer updates.

    The reason for this behavior is that the RT Engine can no longer communicate with the RT Development System front panel. Because the time-critical loop has the highest priority, the user interface thread that updates front panel objects cannot use any processor time. As a result, the front panel and VI appears locked. However, the block diagram is still running.

    While **msec to wait** control is greater than 0, the time-critical loop will "sleep" for that many milliseconds. During this sleep, you can run non-time-critical tasks like updating user interfaces.

9. Click the **Abort** button. After a short period of time, the RT Development System recognizes that it has lost communication with the RT Series device, and displays a warning message. Click **OK** to close the RT Development System.

    You must reset the RT target platform in the **Select Target Platform** dialog box before the RT Development System can communicate with the target platform again.

10. Exit LabVIEW RT by selecting **File»Exit**. Do not save any changes.
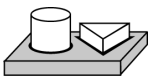
End of Activity 4-1.

Performance of LabVIEW RT VIs

There are a number of programming paradigms you can use to greatly increase the performance of any LabVIEW program, but these take on special importance with LabVIEW RT, because performance and deterministic real-time requirements often go hand in hand. In addition, because priorities and communication play a major role in LabVIEW RT, understanding them is crucial to any high-performance programming.

Refer to Chapter 28, *Performance Issues*, of the *G Programming Reference Manual* for more information about performance.

Writing Efficient Loops

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

Improving DAQ Configuration

The most time-intensive part of any DAQ operation on the RT Engine is the configuration. If possible, try to use the intermediate or advanced DAQ VIs to move configuration outside of your main loop. If you do 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 that 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 does not affect performance if the VI is running at time-critical priority, but it hurts the performance of that communication, making the RT Development System seem sluggish 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 boards, 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 dramatically, because only one read or write is necessary per loop iteration rather than several. For an example of this technique, 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 makes your VIs run 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

Avoid array copying because this 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 in to 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 yield 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\RTTutorial.llb` directory, to accurately measure the time it takes your VI to execute.

Refer to Chapter 28, *Performance Issues*, in the *G Programming Reference Manual* for more information about performance issues using LabVIEW.

A

Technical Support Resources

This appendix describes the comprehensive resources available to you in the Technical Support section of the National Instruments Web site and provides technical support telephone numbers for you to use if you have trouble connecting to our Web site or if you do not have internet access.

NI Web Support

To provide you with immediate answers and solutions 24 hours a day, 365 days a year, National Instruments maintains extensive online technical support resources. They are available to you at no cost, are updated daily, and can be found in the Technical Support section of our Web site at `www.ni.com/support`

Online Problem-Solving and Diagnostic Resources

- **KnowledgeBase**—A searchable database containing thousands of frequently asked questions (FAQs) and their corresponding answers or solutions, including special sections devoted to our newest products. The database is updated daily in response to new customer experiences and feedback.

- **Troubleshooting Wizards**—Step-by-step guides lead you through common problems and answer questions about our entire product line. Wizards include screen shots that illustrate the steps being described and provide detailed information ranging from simple getting started instructions to advanced topics.

- **Product Manuals**—A comprehensive, searchable library of the latest editions of National Instruments hardware and software product manuals.

- **Hardware Reference Database**—A searchable database containing brief hardware descriptions, mechanical drawings, and helpful images of jumper settings and connector pinouts.

- **Application Notes**—A library with more than 100 short papers addressing specific topics such as creating and calling DLLs, developing your own instrument driver software, and porting applications between platforms and operating systems.

Software-Related Resources

- **Instrument Driver Network**—A library with hundreds of instrument drivers for control of standalone instruments via GPIB, VXI, or serial interfaces. You also can submit a request for a particular instrument driver if it does not already appear in the library.

- **Example Programs Database**—A database with numerous, non-shipping example programs for National Instruments programming environments. You can use them to complement the example programs that are already included with National Instruments products.

- **Software Library**—A library with updates and patches to application software, links to the latest versions of driver software for National Instruments hardware products, and utility routines.

Worldwide Support

National Instruments has offices located around the globe. Many branch offices maintain a Web site to provide information on local services. You can access these Web sites from www.ni.com/worldwide

If you have trouble connecting to our Web site, please contact your local National Instruments office or the source from which you purchased your National Instruments product(s) to obtain support.

For telephone support in the United States, dial 512 795 8248. For telephone support outside the United States, contact your local branch office:

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 (Ontario) 905 785 0085, Canada (Québec) 514 694 8521, China 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 (D.F.) 5 280 7625, Mexico (Monterrey) 8 357 7695, Netherlands 0348 433466, Norway 32 27 73 00, Poland 48 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 2377 1200, United Kingdom 01635 523545

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

A

Application Builder, for stand-alone
   executables, 3-3 to 3-5
architecture of LabVIEW RT, 1-2 to 1-5

B

Build Application dialog box, 3-4

C

communicating using host LabVIEW
   applications, 3-11 to 3-14
     shared memory (RT Series DAQ devices
      only), 3-12 to 3-13
     TCP/IP, 3-11
     user interface (activity 3-3), 3-13 to 3-14
     VI Server, 3-11 to 3-12
communication channels, 1-3
computer platform. *See* target platform.
conventions used in manual, *vii-viii*
creating stand-alone executables, 3-3 to 3-5

D

DAQ devices
     in LabVIEW RT architecture (figure), 1-3
     overview, 1-2
     shared memory (RT Series DAQ devices
      only), 3-12 to 3-13
date/time functions on RT Engine, 1-5
debugging LabVIEW RT VIs, 3-3
diagnostic resources, online, A-1
distributed computing with
   VI Server, 3-11 to 3-12

documentation
     conventions used in manual, *vii-viii*
     related documentation, *viii*
downloading VIs, 3-2 to 3-3

E

executables, stand-alone, creating, 3-3 to 3-5
execution target. *See* target platform.

H

host LabVIEW applications. *See* communicating
   using host LabVIEW applications.

I

installation of LabVIEW RT software, 2-1 to 2-2

L

LabVIEW applications. *See* communicating
   using host LabVIEW applications.
LabVIEW RT. *See also* programming.
     architecture, 1-2 to 1-5
     components
      with networked RT series devices
       (figure), 1-3
      with plug-in RT series DAQ devices
       (figure), 1-3
     creating stand-alone executables, 3-3 to 3-5
     debugging LabVIEW RT VIs, 3-3
     downloading VIs, 3-2 to 3-3
     installation, 2-1 to 2-2
     launching, 3-1 to 3-2
     networked RT series devices, 1-2
     overview, 1-1
     plug-in RT series DAQ devices, 1-2

M

N

O

P

W