Worldwide Technical Support and Product Information
ni.com

Worldwide Offices
Visit ni.com/niglobal to access the branch office Web sites, which provide up-to-date contact information, support phone numbers, email addresses, and current events.

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

For further support information, refer to the Additional Information and Resources appendix. To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the Info Code feedback.
Contents

Student Guide
A. NI Certification.....................................................................................................vii
B. Course Description ...............................................................................................viii
C. What You Need to Get Started.............................................................................viii
D. Installing the Course Software..............................................................................ix
E. Course Goals.........................................................................................................ix
F. Course Conventions..............................................................................................x

Lesson 1
Introduction to Real-Time Systems
A. What is a Real-Time System?...............................................................................1-2
B. Real-Time System Components ...........................................................................1-5

Lesson 2
Configuring Your Hardware
A. Hardware Setup and Installation...........................................................................2-2
B. Configuring Network Settings ..............................................................................2-2
C. Installing Software on Target ...............................................................................2-5
D. Configuring Target I/O .........................................................................................2-6

Lesson 3
Accessing Your I/O
A. Connecting to a Target in LabVIEW.................................................................3-2
B. Accessing Your I/O—PXI/PXIe, Stand-Alone cDAQ, Desktop RT Targets........3-4
C. Accessing Your I/O—cRIO and sbRIO..............................................................3-5

Lesson 4
Real-Time Architecture: Design
A. Host and Target Application Architecture............................................................4-2
B. Multithreading ......................................................................................................4-3
C. Yielding Execution in Deterministic Loops .........................................................4-15
D. Timing Loops in LabVIEW Real-Time .................................................................4-16
E. Improving Speed and Determinism ......................................................................4-19

Lesson 5
Inter-Process Communication
A. Sharing Data Locally on an RT Target.................................................................5-2
B. Sharing Data Between Deterministic and Non-Deterministic Loops...................5-2
C. Sharing Data Between Non-Deterministic Loops ...............................................5-6
Lesson 6
Communicating Between RT Target and Host
   A. Using Front Panel Communication ..............................................................6-2
   B. Using Programmatic Network Communication ...........................................6-2

Lesson 7
Verifying Your Application
   A. Verifying Correct Application Behavior ....................................................7-2
   B. Verifying Performance and Memory Usage ...............................................7-3

Lesson 8
Introduction to Deployment
   A. Preparing a Stand-Alone RT Application ...................................................8-2
   B. Creating a Build Specification ...................................................................8-4
   C. Communicating with Deployed Applications ...........................................8-6

Appendix A
Additional Information and Resources
Inter-Process Communication

In this lesson, you learn how to share data between multiple processes on your RT target, between deterministic and non-deterministic processes, and between multiple non-deterministic processes. You also learn how to share latest data and buffered data.

Topics

A. Sharing Data Locally on an RT Target
B. Sharing Data Between Deterministic and Non-Deterministic Loops
C. Sharing Data Between Non-Deterministic Loops
A. Sharing Data Locally on an RT Target

There are many different inter-process communication methods that you can use to share data locally between loops and VIs on an RT target.

When choosing an appropriate inter-process communication method, you should consider the following questions:

- Are you communicating between a deterministic loop and non-deterministic loop? Or are you communicating between non-deterministic loops?
- Do you need to transfer every value or only the latest value?

<table>
<thead>
<tr>
<th></th>
<th>Deterministic loop and Non-deterministic loop</th>
<th>Non-deterministic loop and Non-deterministic loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest value</td>
<td>• Single-process shared variable with RT FIFO enabled (single element)</td>
<td>• Single-process shared variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Local variable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functional global variable</td>
</tr>
<tr>
<td>Every value</td>
<td>• Single-process shared variable with RT FIFO enabled (multi-element)</td>
<td>• Queues</td>
</tr>
<tr>
<td></td>
<td>• RT FIFO functions</td>
<td></td>
</tr>
</tbody>
</table>

B. Sharing Data Between Deterministic and Non-Deterministic Loops

After dividing tasks in an application into a deterministic Timed Loop and normal priority While Loops, you might need to communicate between the loops on a block diagram or between the different VIs on the RT target. You can use the following techniques to send and receive data between VIs or loops in an application:

- Single-process shared variables with the Real-Time FIFO enabled
- Functional global variables
- RT FIFO (first in, first out buffer) functions

This course discusses the single-process shared variables with Real-Time FIFO enabled method. Refer to the LabVIEW Real-Time 2 course for information on the other techniques.
Single-Process Shared Variables with the RT FIFO Enabled

You can use single-process shared variables with the RT FIFO enabled to transfer data from deterministic loop to other loops or VIs without affecting the determinism of the deterministic loop.

Use single-process shared variables to share data between two locations in a block diagram or between VIs running on an RT target. Right-click an RT target in the Project Explorer window and select **New»Variable** from the shortcut menu to open the Shared Variable Properties dialog box, which you can use to create a single-process shared variable.

The Real-Time Module adds real-time FIFO capability to the shared variable. By enabling the real-time FIFO of a shared variable, you can share data without affecting the determinism of VIs running on an RT target. From the Real-Time FIFO page of the Shared Variable Properties dialog box, place a checkmark in the **Enable Real-Time FIFO** checkbox to enable the real-time FIFO of a shared variable.

Single-process shared variables provide a communication method that is easy to use and deterministic when you enable the Real-Time FIFO.

How Are Shared Variables Used?

Shared variables are designed to facilitate communication in LabVIEW. You can configure shared variables to perform many tasks:

- Transfer non-deterministic data between loops or VIs on a single target. In this capacity, a shared variable functions much like a global variable. This type of shared variable is called a **single-process shared variable**.

- Transfer data from a non-deterministic loop to a host. This type of shared variable is a **network-published shared variable**. Refer to Lesson 6, *Communicating Between RT Target and Host*, for more information on communicating between a target and host.

- Transfer non-deterministic data between hosts or between a host and other computers. Shared variables implement a publisher/subscriber model that allows non-real-time computers to communicate across a network. This type of shared variable is called a **network-published shared variable**.

- Transfer deterministic data between Real-Time VIs or loops (Real-Time FIFO). Shared variables can implement a Real-Time FIFO to transfer data deterministically between loops on an RT target. This type of variable is usually a **single-process shared variable with the Real-Time FIFO option enabled**.

- Transfer deterministic data between targets. With a dedicated network connection, shared variables can deterministically transfer data between two or more real-time targets over a network. This type of variable is called a **Time-Triggered shared variable**.

For more information on the Time-Triggered shared variable, refer to the *LabVIEW Help*. 
Creating Shared Variables

To create a shared variable, right-click a target or library in the Project Explorer window and select **New Variable**. All variables must exist inside a library. If you create a new variable outside a library, LabVIEW automatically creates a library. When you create a shared variable, LabVIEW displays a Shared Variable Properties dialog box in which you configure the type of shared variable and any other options such as buffering and Real-Time FIFO.

To use a shared variable on the block diagram, drag the shared variable from the Project Explorer window to the block diagram. Shared variable references on the block diagram work much like local or global variables. You can right-click a shared variable and select **Change to Read** or **Change to Write** to change the direction of the variable. Shared variables contain additional terminals along with the data. Each shared variable has error in and error out terminals, and shared variables set to read can return a timestamp indicating when the data was written. To add a timestamp output to a single-process shared variable, you must first place a checkmark in the **Enable timestamp** checkbox on the Variable page of the Shared Variable Properties dialog box, and then right-click the Shared Variable node and select **Show Timestamp**.

**Shared Variable with the Real-Time FIFO Enabled**

When you enable the Real-Time FIFO option on the Shared Variable Properties page, LabVIEW uses Real-Time FIFOs to transfer the data that is written to and read from the shared variable. You can configure the FIFO to be single or multi-element and define the size of the FIFO. When you enable the Real-Time FIFO option, a small icon appears on references to the variable to indicate that it uses Real-Time FIFOs.

**Single Element FIFO**

A single-element FIFO shares the most recent data value. The shared variable overwrites the data value when it receives a new data value. Use this option when you need only the most recent value. Configure the size of the array elements or the size of the waveform for the FIFO buffer if you select an array or waveform data type.

**Multi-Element FIFO**

A multi-element FIFO buffers the values shared by the shared variable. You must specify the number of elements in the FIFO buffer.

**Note**

For both single-element and multi-element FIFOs, if the variable contains array or waveform data, you must configure the size of the FIFO elements equal to the size of the data you want to share.

**Programming Shared Variable FIFOs—Initialization**

Shared variable FIFOs are created the first time a variable is read from or written to. This results in a slight delay. Therefore, either initialize the variable by reading from or writing to it before your main loop or allow for a delay in the first iteration of your loop as the FIFO is created.
Programming Shared Variable FIFOs—Overflow

Multi-element RT FIFOs have a fixed memory size and a fixed number of elements, which you configure in the Shared Variable Properties dialog box. Therefore, multi-element shared variable RT FIFOs introduce the possibility for overflow and underflow errors.

An overflow error occurs when a shared variable reference attempts to write to an RT FIFO that is already full. When an overflow occurs, the shared variable returns error -2221 and overwrites the oldest value in the FIFO with the new value. The oldest value is permanently lost.

Programming Shared Variable FIFOs—Underflow

An underflow error occurs when a shared variable reference attempts to read an empty RT FIFO. When an underflow occurs, the shared variable returns error -2220 and returns a default value for the data item. This is different from error -2222, which only applies if a variable has never been written to. Also, error -2220 applies only to multi-element FIFOs, whereas error -2222 applies to all shared variables.

Programming Shared Variable FIFOs—Multiple Readers and Writers

LabVIEW creates a single, real-time FIFO for each single-process shared variable even if the shared variable has multiple writers or readers. To ensure data integrity, multiple writers block each other as do multiple readers. Only a single reader and a single writer can access a shared variable at the same time. However, a reader does not block a writer, and a writer does not block a reader. If a single variable has multiple readers and writers, the readers and writers alternate in accessing the variable like any other resource. Variable references waiting on another reader or writer are blocked and do not continue block diagram execution.

Because variables can block with multiple readers or writers, when using variables in a deterministic loop, ensure that a variable read by a deterministic loop cannot be read by another loop and that a variable written by a deterministic loop cannot be written to by another loop. Failure to follow these rules can cause the deterministic loop to block and execute non-deterministically.

By enabling the real-time FIFO, you can select between two slightly different types of FIFO-enabled variables: the single-element and the multi-element buffer. One distinction between these two types of buffers is that the single-element FIFO does not report warnings on overflow or underflow conditions. A second distinction is the value that LabVIEW returns when multiple readers read an empty buffer. Multiple readers of the single-element FIFO receive the same value, and the single-element FIFO returns the same value until a writer writes to that variable again. Multiple readers of an empty multi-element FIFO each get the last value that they read from the buffer or the default value for the data type of the variable if they have not read from the variable before.

If an application requires that each reader get every data point written to a multi-element FIFO shared variable, use a separate shared variable for each reader.
RT FIFO Functions

If your application requires programmatic control of the RT FIFO, use RT FIFO functions instead of shared variables for inter-task communication. For example, you can use RT FIFO functions to:

- Programatically create and delete RT FIFOs
- Programatically set the number of elements in RT FIFOs
- Set timeouts and timeout behavior
- Read the number of elements remaining in RT FIFOs

RT FIFO functions and other inter-task communication methods are covered in detail in the LabVIEW Real-Time 2 course.

C. Sharing Data Between Non-Deterministic Loops

When sharing data between non-deterministic loops, the method you use does not need to be deterministic.

To share latest data between non-deterministic loops, you can use the following methods:

- Local variables—Share data between non-deterministic loops.
- Single Process Shared Variable with RT FIFO disabled—Share data between non-deterministic loops and VIs.
- Functional global variables—Share data between non-deterministic loops and VIs. Refer to the LabVIEW Core 2 course for more information.

To share buffered data between non-deterministic loops, you can use the Queue functions. The Queue API consists of a full-featured set of functions you can use to programmatically create, read from, write to, and get the status of a multiple-element FIFO. Unlike shared variables with the RT FIFO option enabled and RT FIFO functions, queues are compatible with data types of variable size such as strings and variants.

For more information about queues. Refer to LabVIEW Core 2 course or LabVIEW Help.
Self-Review: Quiz

1. You need to transfer every value acquired in a deterministic loop to a non-deterministic loop without affecting the determinism of the loops. Which of the following methods should you use?
   a. Local variable
   b. Single Process Shared Variable with RT FIFO disabled
   c. Single Process Shared Variable with RT FIFO enabled (single element)
   d. Single Process Shared Variable with RT FIFO enabled (multi-element)
Self-Review: Quiz Answers

1. You need to transfer every value acquired in a deterministic loop to a non-deterministic loop without affecting the determinism of the loops. Which of the following methods should you use?
   a. Local variable
   b. Single Process Shared Variable with RT FIFO disabled
   c. Single Process Shared Variable with RT FIFO enabled (single element)
   d. Single Process Shared Variable with RT FIFO enabled (multi-element)