# Contents

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

## Lesson 1
### Introduction to Real-Time
- Exercise 1-1 Project Specification Document ................................................................................................................ 1-1

## Lesson 2
### Configuring Your Hardware
- Exercise 2-1 Configure Hardware ........................................................................................................................................ 2-1
- Exercise 2-2 Targeting Real-Time Hardware ...................................................................................................................... 2-10

## Lesson 3
### Real-Time Architecture: Design
- Exercise 3-1 Priority Levels .................................................................................................................................................. 3-1
- Exercise 3-2 Inter-Task Communication Using Shared Variables ...................................................................................... 3-5
- Exercise 3-3 Course Project: Requirements Document .................................................................................................. 3-19

## Lesson 4
### Timing Applications and Acquiring Data
- Exercise 4-1 Software Timing .............................................................................................................................................. 4-1
- Exercise 4-2 Course Project: Deterministic Loop ................................................................................................................ 4-5

© National Instruments Corporation
Contents

Lesson 5  
Communication  
Exercise 5-1  Course Project: Creating the Target VI ................................................................. 5-1  
Exercise 5-2  Course Project: Creating the Project Host VI ............................................................... 5-17  
Exercise 5-3  Network Streams ......................................................................................................... 5-25

Lesson 6  
Verifying Your Application  
Exercise 6-1  Distributed System Manager ................................................................................ 6-1

Lesson 7  
Deploying Your Application  
Exercise 7-1  Deploy an Application .......................................................................................... 7-1

Lesson A  
Course Slides
Thank you for purchasing the *LabVIEW Real-Time 1* course kit. This course manual and the accompanying software are used in the 2-day, hands-on *LabVIEW Real-Time 1* course.

You can apply the full purchase price of this course kit toward the corresponding course registration fee if you register within 90 days of purchasing the kit. Visit [ni.com/training](http://ni.com/training) to register for a course and to access course schedules, syllabi, and training center location information.

**Note** For course manual updates and corrections, refer to [ni.com/info](http://ni.com/info) and enter the Info Code lvrt1.

### A. NI Certification

The *LabVIEW Real-Time 1* course is part of a series of courses designed to build your proficiency with *LabVIEW* and help you prepare for exams to become an NI Certified LabVIEW Developer and NI Certified LabVIEW Architect. The following illustration shows the courses that are part of the *LabVIEW* training series. Refer to [ni.com/training](http://ni.com/training) for more information about NI Certification.

*Core courses are strongly recommended to realize maximum productivity gains when using LabVIEW.*
B. Course Description

The LabVIEW Real-Time 1 course teaches you to use LabVIEW Real-Time to develop a deterministic and reliable application. Most LabVIEW applications run on a general-purpose operating system (OS) like Windows, Linux, Solaris, or Mac OS. Some applications require deterministic real-time performance that general-purpose operating systems cannot guarantee. The LabVIEW Real-Time Module extends the capabilities of LabVIEW to address the need for deterministic real-time performance.

This course assumes you have a level of experience with LabVIEW equivalent to completing the material in the LabVIEW Core 1 course. In addition, you should be familiar with the Windows operating system and computer components such as the mouse, keyboard, connection ports and plug-in slots, and have experience writing algorithms in the form of flowcharts or block diagrams. The course and exercise manuals are divided into lessons, described as follows.

In the course manual, each lesson consists of the following:

• An introduction that describes the purpose of the lesson and what you will learn
• A description of the topics in the lesson
• A summary quiz that tests and reinforces important concepts and skills taught in the lesson

In the exercise manual, each lesson consists of the following:

• A set of exercises to reinforce topics

C. What You Need to Get Started

Before you use this course manual, make sure you have the following items:

- LabVIEW Full Development System version 2010 or later
- LabVIEW Real-Time Module version 2010 or later
- Temperature Chamber including a 12 volt fan, lamp, and a J-type thermocouple
- cRIO-9074 integrated chassis and controller with a cRIO-9211 thermocouple module and a cRIO-9474 digital output module

- *LabVIEW Real-Time 1 Exercises*

- *LabVIEW Real-Time 1* CD, which contains the following files:

<table>
<thead>
<tr>
<th>Filename</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exercises</td>
<td>A folder containing all files needed to complete the exercises</td>
</tr>
<tr>
<td>Solutions</td>
<td>A folder containing the solutions to each exercise</td>
</tr>
<tr>
<td>LabVIEW-RT.pdf</td>
<td><em>LabVIEW Real-Time 1 Course Manual</em></td>
</tr>
</tbody>
</table>

**D. Installing the Course Software**

Insert the course CD and follow the onscreen instructions to install the software.

Exercise files are located in the `<Exercises>\LabVIEW Real-Time 2\` folder, where `<Exercises>` represents the path to the Exercises folder on the root directory of your computer.
E. Course Goals

This course presents the following topics:

- Concepts of real-time and determinism
- Configuring and communicating with real-time hardware
- Understanding memory usage, multithreading, priorities, and shared resource in the LabVIEW Real-Time Module
- Communicating between a host computer and RT target over the network
- Developing a deterministic, reliable application

This course does not present any of the following topics:

- Information and concepts covered in *LabVIEW Core 1* course
- Control, PID, and/or Fuzzy Logic theory
- Analog-to-digital (A/D) theory
- Operation of GPIB, RS-232, Motion, CAN, or VISA
- Every built-in LabVIEW object, function, or library VI; refer to the *LabVIEW Help* for more information about LabVIEW features not described in this course
- Development of a complete application for any student in the class; refer to the NI Example Finder, available by selecting Help»Find Examples for example VIs you can use and incorporate into VIs you create
F. Course Conventions

The following conventions are used in this course manual:

»

The » symbol leads you through nested menu items and dialog box options to a final action. The sequence `Options » Settings » General` directs you to pull down the `Options` menu, select the `Settings` item, and select `General` from the last dialog box.

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

**bold**

Bold text denotes items that you must select or click in the software, such as menu items and dialog box options. Bold text also denotes parameter names, controls and buttons on the front panel, dialog boxes, sections of dialog boxes, menu names, and palette names.

**italic**

Italic text denotes variables, emphasis, a cross-reference, or an introduction to a key concept. Italic text 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 enter from the keyboard, sections of code, programming examples, and syntax examples. This font also is used for the proper names of disk drives, paths, directories, programs, subprograms, subroutines, device names, functions, operations, variables, filenames, and extensions.

**monospace bold**

Bold text in this font denotes the messages and responses that the computer automatically prints to the screen. This font also emphasizes lines of code that are different from the other examples.

**monospace italic**

Italic text in this font denotes text that is a placeholder for a word or value that you must supply.
Introduction to Real-Time

Exercise 1-1

Project Specification Document

Goal

Read and discuss the Project Specification document and determine if a real-time system is necessary for this application.

When beginning a project for a customer, you may receive a project specification document. This is a general document, and you must meet further with the customer to clarify the project needs to produce a requirements document. In a later exercise, you will receive a more detailed requirements document for the course project.

In this course, the term customer refers to the person requesting the project. A customer can be a client, coworker, manager, and so on.

In this exercise, read the Project Specification document and determine if and why a real-time system is necessary. Look for needs that set a real-time system apart from other systems.

Overview

This application controls the temperature of a temperature chamber to match a user-specified setpoint by outputting a range of intensities to a lamp inside the temperature chamber. The user can simulate disturbance to the temperature chamber by turning a fan on and off inside the temperature chamber. The temperature in the temperature chamber ranges from room temperature to 40 °C.
User Interface

The user interface allows the user to select a setpoint for the temperature in the temperature chamber. The user can also set the PID gains for the control algorithm controlling the temperature. A graph shows actual temperature versus time and setpoint versus time. The user can toggle a button to turn a fan on and off inside the temperature chamber to simulate disturbance. The user can also stop the application running on the RT target from the user interface.

Data Logged

Data logged includes temperature, time stamp, setpoint, lamp intensity, and fan status. All data are logged every time the temperature is read.

Specifications

The temperature should be measured, recorded, and used to calculate output to the lamp. Also, an output should be applied within a set response rate (loop cycle). This system needs a response rate at 10 Hz with a maximum of 1 Hz deviation.

The system should start automatically on power up. External systems should be able to view and utilize current data through a network connection. The system must respond to the Abort command as soon as possible.

End of Specification Document

Discussion

1. Is a real-time system appropriate for this project?

2. If so, what are some of the project needs that require a real-time system?

When considering the project needs, look for requirements that specify a need for real-time response. Real-time response is the ability to respond reliably and without fail to an event, or to perform an operation, within a given time period.

End of Exercise 1-1
Configuring Your Hardware

Exercise 2-1 Configure Hardware

Goal

Configure the CompactRIO target system and the I/O hardware using MAX.

Throughout the course, you use the CompactRIO (cRIO) system as your RT target.

For more information about configuring real-time targets, refer to the MAX Remote Systems Help. Select Help»Help Topics»Remote Systems in MAX.

Implementation

IP Address
1. If you are attending an instructor-led course and your computer is connected to the Internet, disconnect your Ethernet cable.
2. Verify that your network is set to obtain an IP address automatically.
   - Click Start. Select Control Panel»Network Connections.
   - Right-click Local Area Connection and select Properties.
   - Select Internet Protocol (TCP/IP) and click Properties.

If the Obtain an IP address automatically option is selected, your network uses DHCP or AutoIP. Otherwise, your network uses static IP addresses.

- If you are attending an instructor-led course, verify that the Obtain an IP address automatically option is selected.
Lesson 2  Configuring Your Hardware

☐ Click **OK**.

☐ Close the Local Area Connection Properties dialog box.

3. Connect the cRIO-9074 to the network. Your system may already be connected.

☐ If you are attending an instructor-led course, connect port 1 of the RT target to the computer using a cross-over cable.

☐ If the system uses DHCP, connect the RT target to the nearest hub using an Ethernet cable.

⚠️ **Note**  If you connect an RT target directly to your computer using a cross-over cable, the Local Area Connection icon in the Windows system tray will display a limited or no connectivity status because the computer is unable to access the internet.
Set Up Hardware
1. Refer to Figure 2-1 as you set up your CompactRIO system.

Figure 2-1. System Wiring Diagram for *LabVIEW Real-Time 1* Course
cRIO Configuration
Complete the following steps to wire the system.

1. Connect a power cable to the power supply attached to the side of the temperature chamber.

2. Connect the second power supply to the cRIO-9074 controller.
   - Connect the red wire to a V1 input.
   - Connect the black wire to a C input.

3. Connect a power cable to the power supply now attached to the CompactRIO RT target.

4. Plug in the power cables.

5. Watch the LEDs carefully while the cRIO-9074 controller runs a power-on self test (POST). During the POST, the Power and Status LEDs turn on. The Status LED turns off, indicating that the POST is complete.

6. Wait for the system to boot and confirm that the Power LED remains on after booting. The Status LED might blink continuously and slowly if the unit has not yet been configured. You will see this blinking behavior again later after reformatting the disk on the cRIO target and putting the cRIO into an unconfigured state.

7. Connect the plugs attached to the cRIO modules to the appropriate input on the temperature chamber.
Reformatting the Controller
Complete the following steps to reformat the disk on the CompactRIO target.

1. Put the CompactRIO target in safe mode. To reformat a CompactRIO target, the target must be in safe mode.
   - Set the SAFE MODE switch on the CompactRIO target to the ON position.
   - Set the IP RESET switch on the CompactRIO target to the ON position.
   - Press and release the reset button on the CompactRIO target.
     The reset button is a small button on the controller that can be depressed with a small object, such as a pen.
   - Wait for the POST to complete. After the POST completes, the STATUS LED will repeatedly flash three times to indicate that the CompactRIO target is in safe mode.

2. Launch MAX from the desktop or select Start»All Programs»National Instruments»Measurement & Automation.

3. Verify that MAX detects the CompactRIO target.
   - Expand the Remote Systems tree.
   - Verify that the CompactRIO target appears in the Remote Systems tree.
   - If no device is listed, refresh the list by clicking Refresh or pressing <F5>.

4. Reformat the disk on the CompactRIO target.
   - In the Remote Systems tree, right-click the CompactRIO target in Remote Systems and select Format Disk.
   - Click Format in the Format Disk dialog.
   - A dialog box will appear when the disk on the CompactRIO has been formatted successfully. Click OK to reboot the target.
Configuring the Controller

1. Verify that MAX detects the CompactRIO target.
   - Expand the **Remote Systems** tree.
   - Verify that the CompactRIO target appears in the Remote Systems tree.
   - If no device is listed, refresh the list by clicking **Refresh** or pressing <F5>.

2. Configure the CompactRIO target to use a Link Local IP address.
   - Select the CompactRIO target and view the **Network Settings** tab.
   - Verify that Configure IPv4 Address is set to **DHCP or Link Local**.
   - Select the **System Settings** tab.
   - Verify that the **Halt on IP Failure** checkbox is disabled.
   - Click the **Save** button unless it is dimmed.
   - Set the SAFE MODE switch on the CompactRIO target back to the OFF position.
   - Set the IP RESET switch on the CompactRIO target back to the OFF position.
   - Press the reset button on the CompactRIO target.
   - Wait for the POST to complete.

**Note** When the CompactRIO is configured to use a DHCP or link local IP address and the Halt on IP Failure checkbox is disabled, the CompactRIO will use a link local address if it does not find a DHCP server. If you are attending an instructor-led course, you connect directly to the CompactRIO to your computer using a cross-over cable, so the CompactRIO will not find a DHCP server and will use a link local IP address instead.
3. View the IP address of your CompactRIO target.
   - In MAX, select View»Refresh.
   - Expand Remote Systems and select the CompactRIO target.
   - View the Network Settings tab.
   - Record the IP address and Subnet mask values for the CompactRIO RT target for future reference.
     - IPv4 Address:_________________________________
     - Subnet mask:________________________________

   **Note** When an RT target is configured to use a DHCP or link local IP address, the RT target may not always have the same IP address after rebooting. In later exercises, if you cannot find your RT target at the IP address recorded above, check the current IP address of the RT target in the Network Settings tab in MAX and use the current IP address instead.

4. Name the CompactRIO target.
   - View the System Settings tab and enter cRIO-9074 in the Name field of the General Settings section.
   - Click Save to apply the new name.

5. Verify that the correct software is installed on your system.
   - Expand the CompactRIO target under Remote Systems.
   - Right-click Software in the configuration tree under cRIO-9074 and select Add/Remove Software.
   - If a warning dialog appears, click OK.
Select LabVIEW Real-Time 10.0»NI-RIO 3.5.0 with Scan Engine support and click Next, as shown in Figure 2-2.

- Click Next.
- Enable the following add-on selections:
  - NI Application Web Server
  - NI System Configuration network support
  - NI Web-based Configuration and Monitoring
- Click Next.
Lesson 2  Configuring Your Hardware

- Review the installation selections and click **Next** to begin the installation.
- After the software is installed, click **Finish** to exit the LabVIEW Real-Time Software Wizard.

6. Examine the software installed on the CompactRIO expanding **Remote Systems»cRIO-9074» Software** in MAX.

7. Examine the software installed on the CompactRIO by displaying the software and folder hierarchy in a browser with FTP.
   - Open Windows Explorer.
   - Enter `ftp://<IP address>/` where `<IP address>` is the IP address of the CompactRIO target.
   - Examine the folder contents and the dates modified.
   - Close Windows Explorer.

8. Identify the chassis used in the target.
   - In MAX, expand the **Devices and Interfaces** tree under cRIO-9074.
   - Verify that **RIO0** is displayed.
   - Select **RIO0** to view the information in the **General** tab.

**End of Exercise 2-1**
Exercise 2-2  

Targeting Real-Time Hardware

Goal

Create a LabVIEW project and add an RT target for exercises in the course.

In this exercise, you learn how to run a VI on the Real-Time series hardware.

Implementation

In this exercise, you build the project shown in Figure 2-3.
1. Create a project.
   - Open LabVIEW from the desktop or select Start » All Programs » National Instruments » LabVIEW 2010.
   - Select Empty Project from the Getting Started window.
   - Select File » Save Project and save the project as Basic RT Setup.lvproj in the <Exercises> \ LabVIEW Real-Time 1 \ Basic RT Setup directory. The name of the project root changes from Untitled Project 1 to Basic RT Setup.lvproj.
   - Notice that the project contains the root and the My Computer target.

2. Create the CompactRIO RT target.
   - Right-click Project: Basic RT Setup.lvproj in Project Explorer and select New » Targets and Devices to display the Add Targets and Devices dialog box.
   - Be sure to right-click the Project: Basic RT Setup.lvproj entry and not the My Computer entry. You can add new targets by right-clicking My Computer. However, targets under My Computer must be internal targets on the computer, such as PCI Real-Time devices.
   - Expand Real-Time CompactRIO.
   - Select the cRIO-9074 target.
   - Click OK. Wait while LabVIEW detects the C Series modules.
   - In the Select Programming Mode dialog box, select Scan Interface and click Continue.

3. Verify proper configuration of the project.
   - Expand the cRIO-9074 entry in the Project Explorer window.
   - Expand the Chassis (cRIO-9074) entry.
Expand the modules and verify that the following I/O variables exist.

- Mod1 (Slot 1, NI 9211)
  - AI0
  - AI1
  - AI2
  - AI3

- Mod2 (Slot 2, NI 9474)
  - DO0
  - DO1
  - DO2
  - DO3
  - DO4
  - DO5
  - DO6
  - DO7

4. Save the project.

5. Rename and configure the thermocouple module.
   - Right-click Mod1 and select Properties.
   - Change the Name of the module to Thermocouple Mod.

   **Note** Each channel of the NI 9211 could be configured for a different type of thermocouple. This project uses a J-Type thermocouple.
   - Click OK.
   - Right-click Mod2 and select Properties.
   - In the Module Configuration category, change the Name of the module to PWM Mod.
   - Select the Specialty Digital Configuration category.
   - Set Specialty Mode to Pulse-Width Modulation.
   - Select PWM0 in the Channels listbox.
   - Set Frequency (Period) to 1 kHz (1,000 µs).
   - Select PWM1 in the Channels listbox.
   - Set Frequency (Period) to 1 kHz (1,000 µs).
   - Click OK.

7. Rename the I/O variables.
   - Right-click AI0 of Thermocouple Mod and select Rename. Change the name of the I/O variable to Temperature.
   - Right-click the PWM0 of PWM Mod and select Rename. Change the name of the I/O variable to Lamp Intensity.
   - Right-click the PWM1 of PWM Mod and select Rename. Change the name of the I/O variable to Fan Intensity.

8. Save the project.
   Use this Basic RT Setup project in the course whenever you need to execute a simple VI on your CompactRIO RT target. For more complex programs, you create a unique project.
9. In Project Explorer window, right-click the CompactRIO RT target and select **Connect** to ensure that LabVIEW can connect to your target.

   - Enable the **Close on successful completion** checkbox on the Deployment Progress dialog box, as shown in Figure 2-4.

![Deployment Progress Dialog Box](image)

**Figure 2-4.** Deploy Dialog Box

10. When deployment completes, click the **Close** button to exit the Deploy dialog box.

11. The small green light next to the target icon should now be lit, indicating that the target is connected.

12. Deploy the I/O variables to the CompactRIO RT target.

   - Right-click the Thermocouple Mod in the Project Explorer and select **Deploy**.
   - Right-click the PWM Mod in the Project Explorer and select **Deploy**.
Test the I/O Variables
The NI Distributed System Manager provides a central location for monitoring systems on the network and managing published data. The system manager offers test panels for CompactRIO modules running in scan mode. As soon as your system is available on the network, you have access to real-time and historical-trend I/O values, so you can quickly verify your connections and signal integrity.

1. Open the Distributed System Manager by selecting Start » All Programs » National Instruments » Distributed System Manager 2010.

2. Verify that the Fan Intensity I/O variable you configured is operating as specified.
   - In the NI Distributed System Manager, expand the Network Items tree and select the IP address that you specified for the CompactRIO RT target in Exercise 2-1.
   - On the Scan Engine tab, click Enable Forcing. Verify that the Forcing Enabled LED turns green.
   - Expand the tree next to your IP address and PWM Mod. Select the Fan Intensity I/O variable.
   - Enter 50 as the New Value for the I/O variable.
   - Click Force. The fan should turn on with 50% intensity.

   **Note** Clicking Force will force the variable to assume New Value until you unforce the variable, reboot the target, or force the variable to assume another New Value.

   - Verify that the Current Value changes to 50. Notice that the label of Current Value now indicates that the value has been forced.
   - Enter 0 as the New Value for the I/O variable.
   - Click Force. The fan should turn off.
   - Click Unforce to return control of the Fan Intensity I/O variable to the NI Scan Engine.
3. Verify that the Lamp Intensity I/O variable you configured is operating as specified.
   - Select the **Lamp Intensity** I/O variable.
   - Enter 50 as the **New Value** for the I/O variable.
   - Click **Force**. The lamp should turn on with 50% intensity.
   - Verify that the **Current Value** changes to 50. Notice that the label of **Current Value** now indicates that the value has been forced.
   - Enter 0 as the **New Value** for the I/O variable.
   - Click **Force**. The lamp should turn off.

4. Click **Unforce** to return control of the **Lamp Intensity** I/O variable to the NI Scan Engine.

5. Verify that the Temperature I/O variable you configured is operating as specified.
   - Expand the **Thermocouple Mod** tree and select **Temperature**. Does the **Current Value** make sense? The current value represents the temperature in Celsius between 20 and 40 degrees for room temperature.

6. Close the NI Distributed System Manager.

**Run a VI**

1. In the Basic RT Setup Project Explorer, right-click your CompactRIO RT target in the Project Explorer and select **Add File**.
   - Select **Blinking Light.vi** located in the `<Exercises>\LabVIEW Real-Time 1\Basic RT Setup` directory and click **Add File**.

2. Save the project.

3. Double-click the VI to open the front panel.

4. Run the VI with the default front panel settings.

   Notice that the light on the temperature chamber turns on and off at the blink rate.
Note Throughout this course, when you run a VI on the RT target, you may be prompted to save other VIs. In all instances, save VIs when prompted.

6. Click the **Stop** button on the VI to stop the VI.

7. Close the VI.

**End of Exercise 2-2**
Real-Time Architecture: Design

Exercise 3-1  Priority Levels

Goal
Adjust the priority levels to understand the effects of priority levels and timing on scheduling a VI.

Scenario
In this exercise, you learn the effects of changing the priority and timing of a VI on a real-time system.

Implementation
1. Open the Basic RT Setup project from the <Exercises>\LabVIEW Real-Time 1\Basic RT Setup directory. You created this project in Exercise 2-2.
2. Observe the effects of priority and timing using Timed Loops.
   - Add the Process VI from the <Exercises>\LabVIEW Real-Time 1\Basic RT Setup directory to your RT target in the Project Explorer.
   - Add the Priority Trouble - Timed Loop VI from the <Exercises>\LabVIEW Real-Time 1\Basic RT Setup directory to your RT target in the Project Explorer.
   - Double-click the Priority Trouble - Timed Loop VI to open it.
   - Examine the block diagram.
     - Double-click the Input Node of the Timed Loop to open the Configure Timed Loop window.
     - Notice that the Maintain original phase checkbox is disabled. This means if an iteration of the Timed Loop starts late, the Timed Loop will execute the iteration immediately instead of maintaining the original phase.
– Click OK to close the Configure Timed Loop window.
– Notice that you can monitor the Finished Late and Iteration Duration information of the previous iteration in a Timed Loop.
– Notice the Process subVI. The process duration (ms) control configures how long this subVI executes.
– Notice the Period (ms) control. This configures the period of the Timed Loop.

❑ On the front panel, set the controls to the following values:
  – Period (ms): 10
  – Process duration (ms): 2

❑ Run the VI.

The chart displays a sine wave.

❑ Slowly decrement the Period (ms) control. Notice that the Iteration Duration indicator updates.

❑ After the Period (ms) control is set to 2 or less, a dialog box warning, Waiting for Real-Time target to respond, appears. Wait for the dialog box to disappear.

❑ A dialog box stating, Warning: Connection to the Real-Time target has been lost, appears. Click OK.

With each step, the chart display speeds up. However, when the code inside this Timed Loop takes longer to execute than the Timed Loop period, the front panel no longer updates.

The reason for this behavior is that the Real-Time engine on the target can no longer communicate with the Real-Time development system front panel on the host computer.

Because timed loops execute at a priority between high and time-critical priority in the LabVIEW execution system, the user interface thread that updates the front panel cannot use any processor time. As a result, the front panel and VI appear locked. However, the code running on the Real-Time Engine still runs normally.
While the value of the Period (ms) control is greater than the value of the Process duration (ms) control, the timed loop sleeps for the difference between those millisecond values. During this sleep, the Real-Time Engine schedules non-time-critical tasks like updating the user interface or sending information over TCP/IP.

- Close the VI.
- Press the reset button on the CompactRIO RT target with a small object, such as a pen.

Because the Priority Trouble - Timed Loop VI is still running on the target, you must reset the target to stop the VI.

3. Observe the effects of priority and timing using VI Priority.

- Add the Priority Trouble - VI Priority VI from the <Exercises>\LabVIEW Real-Time 1\ Basic RT Setup directory to your RT target in the Project Explorer.
- Double-click the Priority Trouble - VI Priority VI to open it.
- Select File»VI Properties.
- Select Execution from the Category 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 Real-Time engine.

- Click OK to close the VI Properties dialog box.
- Set the msec to wait control to 10.
- Run the VI.

The chart displays a sine wave.

- Slowly decrement the msec to wait control.
- After reaching 0, a dialog box warning, Waiting for Real-Time target to respond appears. Wait for the dialog box to disappear.
A dialog box stating, **Warning: Connection to the Real-Time target has been lost.** appears. Click **OK**.

With each step, the chart display speeds up. However, when the **msec to wait** control reaches 0, the front panel no longer updates.

The reason for this behavior is that the Real-Time engine on the target can no longer communicate with the Real-Time development system front panel on the host computer. Because the time-critical loop has the highest priority, the user interface thread that updates the front panel cannot use any processor time. As a result, the front panel and VI appear locked. However, the code running on the Real-Time Engine still runs normally.

While the **msec to wait** control is greater than 0, the loop in this time-critical VI sleeps for that many milliseconds. During this sleep, the Real-Time Engine schedules non-time-critical priority tasks like updating the user interface or sending information over TCP/IP.

Press the reset button on the CompactRIO RT target with a small object, such as a pen.

Because the Priority Trouble - VI Priority VI is still running on the target, you must reset the target to stop the VI.

Save all VIs and close the project.

**End of Exercise 3-1**
Exercise 3-2  

Inter-Task Communication Using Shared Variables

Goal

Use single-process shared variables with RT FIFO enabled for inter-task communication.

Scenario

In this exercise, you learn how to use single-process shared variables with the RT FIFO option enabled for inter-task communication.

You will use single-process shared variables to share data between two Timed Loops running on an RT target. 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. Single-process shared variables provide a communication method that is easy to use and deterministic when you enable the Real-Time FIFO.

Implementation

1. Open the Inter-Task Communication project.
   - In LabVIEW, select File » Open Project.
   - Select \Exercises\LabVIEW Real-Time 1\Inter-Task Communication\Inter-Task Communication.lvproj.

2. Configure the CompactRIO target.
   - Right-click the RT target in the Project Explorer and select Properties.
   - In the General category, set the IP address to the IP address of your cRIO-9074.
   - Click OK.
Shared Variables
1. Create the library that will contain the single-process shared variables for inter-task communication.
   ❑ Right-click the RT target in the Project Explorer and select New»Library.

   Note You must always create a shared variable inside a project library. Creating a shared variable outside a project library automatically creates a new library.

   ❑ Right-click the new library and select Save»Save. Save the library as Communication Shared Variables.lvlib in the <Exercises>\LabVIEW Real-Time 1\Inter-Task Communication directory.

2. Create the Waveform Type single-process shared variable for inter-task communication.
   ❑ Right-click the Communication Shared Variables library and select New»Variable. The Shared Variable dialog box appears.
   ❑ Set the following options in the Variable category of the Shared Variable Properties dialog box.
     – Name: Waveform Type
     – Variable Type: Single-Process
     – Data Type: UInt8
   ❑ Set the following options in the RT FIFO category of the Shared Variable Properties dialog box.
     – Enable RT FIFO: Checked
     – FIFO Type: Single Element
   ❑ Click OK.
3. Create the Amplitude single-process shared variable for inter-task communication.
   - Right-click the library and select New » Variable. The Shared Variable dialog box appears.
   - Set the following options in the Variable category of the Shared Variable Properties dialog box.
     - Name: Amplitude
     - Variable Type: Single-Process
     - Data Type: Double
   - Set the following options in the RT FIFO category of the Shared Variable Properties dialog box.
     - Enable RT FIFO: Checked
     - FIFO Type: Single Element
   - Click OK.

4. Create the Result Data single-process shared variable for inter-task communication.
   - Right-click the library and select New » Variable. The Shared Variable dialog box appears.
   - Set the following options in the Variable category of the Shared Variable Properties dialog box.
     - Name: Result Data
     - Variable Type: Single-Process
     - Data Type: Double
5. Create the Stop single-process shared variable for inter-task communication.

- Right-click the library and select New»Variable. The Shared Variable dialog box appears.
- Set the following options in the Variable category of the Shared Variable Properties dialog box.
  - Name: Stop
  - Variable Type: Single-Process
  - Data Type: Boolean
- Set the following options in the RT FIFO category of the Shared Variable Properties dialog box.
  - Enable RT FIFO: Checked
  - FIFO Type: Single Element
- Click OK.

6. Right-click the Communication Shared Variables.lvlib and select Save»Save All (this Library).
Deterministic Loop
In this deterministic application, you will separate deterministic tasks from non-deterministic tasks and place them in a Timed Loop with the highest priority in an RT target VI to ensure the deterministic tasks receive enough processor resources. This Timed Loop will be the deterministic Timed Loop in this application and will perform the deterministic task of generating waveform data.

1. Open the Single-process Shared Variable Method VI from the Project Explorer.

2. Add and configure the deterministic Timed Loop.
   - Place a Timed Loop structure on the block diagram.
   - Double-click the Input Node of the Timed Loop and configure the Timed Loop with the following options:
     - Period: 10 ms
     - Priority: 500
     - Structure Name: DL
     - Leave all other options at their default settings.
   - Click OK.

3. On the block diagram, configure the Timed Loop Input Node.
   - Resize the Input Node of the Timed Loop to show four inputs.
   - Click the first input of the Input Node and select Structure Name.
   - Click the second input of the Input Node and select Source Name.
   - Click the third input of the Input Node and select Period.
   - Click the fourth input of the Input Node and select Priority.
4. To place the shared variables inside the deterministic loop, complete the following steps:
   - In the Project Explorer window, click the Waveform Type shared variable and then press and hold <Ctrl>. With <Ctrl> still held, click the other three shared variables.
   - With the four variables selected, drag them inside the Timed Loop on the block diagram.
   - Right-click the Result Data shared variable and select Access Mode→Write.

5. Create a shift register to pass error data to following loop iterations.
   - Right-click the right border of the Timed Loop and select Add Shift Register.
   - Wire the error out output of the Stop shared variable to the input of the shift register on the right border of the Timed Loop.

6. Create the deterministic loop as shown in Figure 3-1 using the following items:

![Deterministic Loop of Single-Process Shared Variable Method VI](image)
Waveform Generator VI—Place the Waveform Generator VI on the block diagram inside the Timed Loop. This VI is located in the Project Files virtual folder in the Project Explorer.

- **Unbundle By Name**
  - Set to the `code` element.
  The Unbundle By Name extracts the code numeric data from the error cluster. 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 code `-2221` and overwrites the oldest value in the FIFO with the new value. The oldest value is permanently lost.

- **Case structure**
  - Wire the output of the Unbundle By Name function to the case selector terminal for the Case structure.
  - Change the selector label for the 1 case to `-2221`.
  - Place a True constant and a Clear Errors VI in the case.
  - Place a False constant in the 0, Default case.
  - Wire both cases as shown in Figure 3-2 and Figure 3-3.

![Figure 3-2. Overflow Case in Deterministic Loop of Single-Process Shared Variable Method VI](image)
Unbundle By Name—Extracts the status Boolean data from the error cluster. If an error has occurred or the Stop shared variable outputs a true value, then stop the Timed Loop.

Or function

Non-deterministic Loop
In this deterministic application, you will separate non-deterministic tasks from deterministic tasks and place them in a Timed Loop in an RT target VI that has a lower priority than the deterministic Timed Loop to ensure the non-deterministic tasks do not preempt the deterministic tasks. The Timed Loop containing the non-deterministic task of plotting waveform data will be the non-deterministic Timed Loop in this application.

1. Add and configure the non-deterministic Timed Loop.
   - Place a Timed Loop structure on the block diagram above the deterministic Timed Loop.
   - Double-click the Input Node of the Timed Loop and configure the Timed Loop with the following options:
     - Period: 100 ms
     - Priority: 100
     - Structure Name: NL
2. On the block diagram, configure the Timed Loop Input Node.
   - Resize the Input Node of the Timed Loop to show four inputs.
   - Click the first input of the Input Node and select Structure Name.
   - Click the second input of the Input Node and select Source Name.
   - Click the third input of the Input Node and select Period.
   - Click the fourth input of the Input Node and select Priority.

3. Place the four shared variables inside the non-deterministic Timed Loop.
   - In the Project Explorer window, click the Waveform Type shared variable and then press and hold Ctrl. With Ctrl still held, click the other three shared variables.
   - Right-click the Waveform Type shared variable and select Access Mode»Write.
   - Right-click the Amplitude shared variable and select Access Mode»Write.
   - Right-click the Stop shared variable and select Access Mode»Write.

4. Create a shift register to pass error data to following loop iterations.
   - Right-click the right border of the Timed Loop and select Add Shift Register.
   - Wire the error out output of the Stop shared variable to the input of the shift register on the right border of the Timed Loop.
5. Create the non-deterministic loop as shown in Figure 3-4 using the following items:

- **To Unsigned Byte Integer function**—This function converts the Waveform Type control enum data type into the unsigned 8-bit integer data type of the Waveform Type shared variable.

- **While Loop**—Place a While Loop inside the non-deterministic Timed Loop. This While Loop will read from the Result Data shared variable until the Result Data shared variable RT FIFO is empty.
  - Right-click the border of the While Loop and select **Add Shift Register**.

- **Unbundle By Name**
  - Set to the **Code** element.
  - Extracts the code numeric data from the error cluster. 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 code -2220 and returns a default value for the data item.

*Figure 3-4. Non-deterministic Loop of Single-Process Shared Variable Method VI*
Case structure

- Wire the output of the Unbundle By Name function to the case selector terminal for the Case structure.
- Change the selector label for the 1 case to -2220 and place a True constant and a Clear Errors VI in the case.
- Place the Result Data indicator and a False constant in the 0, Default case.
- Wire the -2220 and 0, Default cases as shown in Figure 3-5 and Figure 3-6, respectively.

![Figure 3-5. Underflow Case in Non-deterministic Loop of Single-Process Shared Variable Method VI](image)

![Figure 3-6. Default Case in Non-deterministic Loop of Single-Process Shared Variable Method VI](image)

Unbundle By Name—Extracts the status Boolean data from the error cluster. If an error has occurred or the Stop button has been pressed, then stop the Timed Loop.

Or function
**Initialization**

Initialize the values of all the shared variables to known values before the deterministic and non-deterministic Timed Loops begin executing.

1. Initialize the value of a shared variable at the start of the VI.
   - In the Project Explorer window, select the Waveform Type shared variable and drag it to the left of the Timed Loops.
   - Right-click the Waveform Type shared variable and select Access Mode»Write.
   - Right-click the input terminal of the shared variable and select Create»Constant.
   - Repeat Step 1 for the Amplitude, Result Data, and Stop shared variables.

2. Wire the block diagram as shown in Figure 3-7.

![Figure 3-7. Initialization of Single-Process Shared Variable Method VI](image)
Shutdown

After the deterministic and non-deterministic Timed Loops stop executing, merge and handle the errors.

1. Merge and handle the errors as shown in Figure 3-8 using the following items:

- Merge Errors function
  - Right-click the error out output and select Create » Indicator.

2. In the Project Explorer, select File » Save All (this Project).
Testing
1. Run the VI. Notice that the graph displays the data generated by the deterministic Timed Loop.

2. Stop the VI.

3. Change the RT FIFO size of the Result Data shared variable.
   - On the Project Explorer, right-click the Result Data shared variable and select Properties.
   - Select the RT FIFO category of the Shared Variable Properties dialog box.
   - Set number of elements to 5.
   - Click OK.

4. Click Save. Run the VI.
   Notice that the Overflow indicator shows that data in the RT FIFO of the Result Data shared variable is lost because the RT FIFO size is not large enough. As a general rule, the RT FIFO size should be the period of the reader loop, divided by the period of the writer loop, multiplied by 1.1. The calculation for this particular is (100 ms /10 ms) × 1.1 which equals 11. Therefore, a safe buffer size for the Results Data RT FIFO is 11.

5. Stop the VI.

6. Change the RT FIFO size of the Result Data shared variable to 11.

7. Close the VI when finished. Save the VI if prompted.

End of Exercise 3-2
Exercise 3-3

Course Project: Requirements Document

Goal

In this exercise, you analyze the requirements document for the project and examine a flowchart based on these requirements.

Analysis

1. Read the following requirements document.

   Note The abbreviations NL and DL are used throughout this course. NL stands for non-deterministic loop and DL stands for deterministic loop.

Start of Requirements Document

This application controls a temperature chamber. The temperature in the chamber is achieved through the control of a lamp. A fan simulates disturbance to the temperature chamber. The entire control setup is shown in Figure 3-9.

Figure 3-9. Overview of Control Setup
Target temperature ranges are confined from room temperature to 40 °C. The user inputs a setpoint.

The control algorithm used for the system is PID. The system has already been modeled as a first-order system and the PID values determined. Use the following PID values:

- P: 15
- I: 0.5
- D: 0.0

This system will be implemented using CompactRIO hardware and needs a response rate of 10 Hz with a maximum jitter of 1 Hz.

**Implementation – CompactRIO**

- cRIO-9074 integrated chassis and controller
- cRIO-9211 thermocouple module
- cRIO-9474 digital output module
- Temperature chamber: 12 volt lamp, J-type thermocouple, and a 12 volt fan to simulate disturbance
Host VI

The user interface is implemented on a Windows XP computer. The user interface communicates with a data acquisition and control program running on a real-time operating system on the target hardware.

Set the setpoint and PID gains through the user interface. Although the PID values have already been determined, the user should be able to adjust these values from the user interface. Figure 3-10 shows an example user interface with the default values set.

You can click the **Apply Disturbance** button at any time to turn on the fan. You can also click the **Stop Target** button to stop the hardware. The target VI must respond to the **Stop Target** and **Apply Disturbance** buttons as soon as possible.

Information displayed to the user includes the current Lamp Intensity output to the lamp as a percentage of maximum power, the current Fan Status as on or off, timestamp of last temperature value read, and a chart displaying the actual temperature and current target setpoint. The user interface also displays the current RT Target Status, which is **Stopped** or **Running**.
The following is a summary of the major tasks of the host VI:

- Transfer Setpoint and PID Gains to the RT target using network communication
- Send Stop Target command through network communication as needed
- Send Disturbance command through network communication as needed
- Receive and chart data from target through network communication

Figure 3-10. Example User Interface
**Target VI**

When the target VI begins running, the VI continually reads the temperature of the chamber, receives setpoints and PID gains from the host VI, and controls the lamp output to change the temperature of the chamber to match the setpoint. The VI must also continually monitor for the Stop Target command and the Disturbance command while controlling the temperature of the chamber and implement them as soon as possible.

The main three sections of the target VI are Initialize, Ongoing Tasks, and Shutdown.

In the Initialize section, the target VI initializes resources and variables for this application. In the Shutdown section, the target VI closes resources and safely shuts down the application.

In the Ongoing Tasks section, the target VI executes the deterministic task of controlling the temperature chamber. The order of execution in the deterministic is defined in the following flowchart:

![Deterministic Loop Flowchart](image)

*Figure 3-11. Deterministic Loop Flowchart*
In the Ongoing Tasks section, the target VI also executes the non-deterministic task of logging the acquired data to a file on the RT target. This data must be accurately time-stamped. The system must log the data in a tab-delimited ASCII file. Data is logged for every temperature read. Logging of data must not affect the system determinism in any form. Table 3-1 shows an example of the expected log file.

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>Setpoint (°C)</th>
<th>Temp (°C)</th>
<th>Lamp Intensity (%)</th>
<th>Fan Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:12:35</td>
<td>28</td>
<td>25.00</td>
<td>80.00</td>
<td>OFF</td>
</tr>
<tr>
<td>10:12:36</td>
<td>28</td>
<td>26.01</td>
<td>75.00</td>
<td>ON</td>
</tr>
</tbody>
</table>

The target VI also executes the non-deterministic task of communicating over the network with the host VI. The target VI sends the latest values of the timestamp, setpoint, temperature, lamp intensity, and fan status to the host VI through network communication. The target VI also receives the setpoint, PID gains, and disturbance command from the host VI through network communication.

The following is a summary of the major tasks of the target VI:

- Control temperature chamber as specified
- Monitor for Stop Target and Disturbance commands
- Implement Stop Target and Disturbance commands
- Transfer and receive data to and from the host through network communication
- Log data to file on the RT target

End of Requirements Document
Discussion Questions

Answer the following discussion questions:

1. What data does the host VI need to transfer to the target?

2. What data does the target VI need to transfer to the host?

3. In addition to transferring data to the host VI, the target VI must also log the data to file. Because data logging is not deterministic, it must be handled in the non-deterministic loop. What is a valid method for transferring data from the deterministic loop to the non-deterministic loop for the log file?
Discussion Question Answers

1. What data does the host VI need to transfer to the target?
   Answer: Setpoint, PID Gains, Stop Target command, Disturbance command

2. What data does the target VI need to transfer to the host?
   Answer: Timestamp, Setpoint, Temperature, Lamp Intensity, Fan Status

3. In addition to transferring data to the host VI, the target VI must also log the data to file. Because data logging is not deterministic, it must be handled in the non-deterministic loop. What is a valid method for transferring data from the deterministic loop to the non-deterministic loop for the log file?
   Answer: Single-process shared variables with the RT FIFO option enabled

End of Exercise 3-3
Timing Applications and Acquiring Data

Exercise 4-1

Software Timing

Goal

Modify pre-existing code to implement software timing using the millisecond and microsecond timer of the RTOS.

Scenario

In this exercise, you implement software timing for an existing piece of code. The purpose of the existing VI is to turn the lamp on and off on the temperature chamber within a set number of seconds.

Implementation

Millisecond Timer

1. Open the Basic RT Setup project from the `<Exercises>\LabVIEW Real Time\Basic RT Setup\` directory. You created this project in Exercise 2-2.

2. Add the `Sleep - Millisecond Timing.vi` from the `<Exercises>\LabVIEW Real-Time 1\Basic RT Setup` directory to your RT target in the Project Explorer.

3. Double-click the `Sleep - Millisecond Timing VI` to open it.
4. Modify the block diagram of the VI to use the millisecond timer of the RTOS, as shown in Figure 4-1.

![Block Diagram](image)

- Double-click the Input Node of the Timed Loop to view its configuration. Verify that under the Loop Timing Source section, the Source Type is set to **1 kHz Clock**. This tells the Timed Loop to use software timing by accessing the operating system millisecond clock on the target as its timing source. Using the 1 kHz clock, the Timed Loop can execute an iteration once every 1 millisecond. Click **OK**.
- Right-click the dt input of the Timed Loop Input Node and select **Create->Control**. Rename the control as **Period (ms)**.
- Place three Lamp Intensity cRIO I/O nodes on the block diagram to control the intensity of the lamp in the temperature chamber. Select the **cRIO-9074»Chassis»PWM Mod»Lamp Intensity(PWM0)** item in the Project Explorer and drag it onto the block diagram. Wire the constants to the I/O nodes as shown in Figure 4-1.
- Complete the wiring as shown in Figure 4-1.

5. Save the VI and, if prompted, any related VIs.
Testing
1. Return to the front panel.

2. Set the Period (ms) to a value that allows you to visually inspect whether the millisecond timing is executing. For example, a value such as 500 ms would turn the lamp on and off every 0.5 seconds.

3. Run the VI.

4. Stop and close the VI when you are finished.

Microsecond Timer
1. Save a copy of the Sleep - Millisecond Timing.vi as Sleep - Microsecond Timing.vi in the <Exercises>\LabVIEW Real-Time 1\Basic RT Setup directory.

2. Modify the front panel window of the VI.

   - Rename the Period (ms) control as Period (us) as shown in Figure 4-2. You will modify this VI to use microsecond units instead of millisecond units.

![Figure 4-2. Sleep - Microsecond Timing VI Front Panel](image-url)
3. Modify the block diagram of the VI to use the microsecond timer as shown in Figure 4-3.

   Double-click the input Node of the Timed Loop to view its configuration. Under the Loop Timing Source section, set the Source Type to 1 MHz Clock. This tells the Timed Loop to use the microsecond timer on the processor on the target as its timing source. Using the 1 MHz clock, the Timed Loop can execute an iteration once every 1 microsecond. Click OK.

![Figure 4-3. Sleep - Microsecond Timing VI Block Diagram](image)

4. Save the VI and, if prompted, any related VIs.

**Testing**

1. Return to the front panel.

2. Set the **Period (us)** to a value that allows you to visually inspect whether the microsecond timing is executing. For example, a value such as 500,000 us would turn the lamp on and off every 0.5 seconds.

3. Run the VI.

4. Stop and close the VI when you are finished.

**End of Exercise 4-1**
Exercise 4-2

Course Project: Deterministic Loop

Goal

Create the deterministic loop portion of the course project.

Scenario

In this exercise, you build the deterministic loop portion of your course project. You may want to refer to the flowchart from Exercise 3-3 as you complete each step. The Implementation section of this exercise is divided into two sections:

- Create Deterministic Loop in Chamber Control subVI
- Configure Chamber Control subVI in RT Target VI

The deterministic loop in the Chamber Control subVI must complete the following tasks:

1. Read PID gains from host computer
2. Read setpoint from host computer
3. Read temperature of temperature chamber
4. Get timestamp of temperature reading
5. Determine proportional, integral, derivative (PID) output
6. Read disturbance status from host computer
7. Output fan intensity
8. Output lamp intensity
9. Write data to RT FIFO
10. Read Stop RT status from host computer

Complete each of these tasks in sequence, using error wires and frames of a Timed Loop to impose the order of operations on each task. Doing so ensures that there are no parallel processes in the deterministic loop.
Implementation

Create Deterministic Loop in Chamber Control subVI
In this section, you create the deterministic loop that controls the temperature of the temperature chamber for this application. The deterministic loop is contained within a subVI. This helps make the main VI more readable.

1. Open <Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller\Temperature Chamber Controller.lvproj. This project contains a target and a number of modules, subVIs, and variables for implementing the project.

2. Configure the CompactRIO target.
   - Right-click the RT CompactRIO target in the Project Explorer and select Properties.
   - In the General category, set the IP Address to the IP address of your cRIO-9074.

3. In the Project Explorer, expand the RT target and double-click the RT Target VI to open it.
   This VI contains Initialization, Ongoing Tasks, and Shutdown sections. The deterministic and non-deterministic tasks both take place in the Ongoing Tasks section. In this exercise, you only implement the deterministic Chamber Control task in the Ongoing Tasks section. You will implement the non-deterministic Logging and Network Communication task in a later exercise.

4. On the block diagram, double-click the Chamber Control subVI to open it.
   This subVI contains a deterministic loop and follows the deterministic loop flowchart.

5. Examine the block diagram of the Chamber Control subVI.
   - Notice that the period and priority of the Timed Loop is configured by the Period and Priority numeric controls. These two controls have been mapped to terminals on the connector panel, so you can set the period and priority inputs of this subVI from the main VI.

   Note Because this Timed Loop will be the highest priority task, it will not relinquish processor resources until it completes all tasks it contains. You must be sure to configure enough sleep time in this Timed Loop to allow lower priority tasks and Timed Loops to execute.
6. Configure the Timed Loop.
   - Click the Left Data Node element and set it to Period.
   - Right-click the border of the Timed Loop and select Add Frame After. Repeat this until the Timed Loop contains five frames.
     You will use these frames to help define sequential execution and dataflow for functions that do not have error inputs and outputs.
   - Right-click the top border of each of the frames except the first frame and deselect Show Left Data Node and Show Right Data Node.
   - Wire the error in control to the left border of the Timed Loop. Right-click the tunnel on the Timed Loop and select Replace with Shift Register.

7. Modify the Chamber Control subVI block diagram to control the temperature chamber temperature deterministically, as shown in Figures 4-4 and 4-5, using the following items in the corresponding frames of the Timed Loop:
Figure 4-4. Chamber Control VI Block Diagram - First 3 Frames

Figure 4-5. Chamber Control VI Block Diagram - Last 2 Frames
Lesson 4  Timing Applications and Acquiring Data

- To Double Precision Float function
- Divide function
  - Right-click the y input and select Create » Constant. Set the constant to 1000. This converts the units of the Period output of the Timed Loop Left Data Node from milliseconds to seconds.
- Equal to 0? function
- Network-published shared variables with the RT FIFO enabled—Use this type of shared variable to deterministically read PID gains, setpoint, and stop values from the host computer. You will learn more about this type of shared variable in Lesson 5.
  - PID Gains shared variable—Drag a copy of this shared variable from cRIO-9074 » Project Variables.lvlib » Host to Target » PID Gains in the Project Explorer to the block diagram.
  - Setpoint shared variable—Drag a copy of this shared variable from cRIO-9074 » Project Variables.lvlib » Host to Target » Setpoint in the Project Explorer to the block diagram.
  - Stop RT shared variable—Drag two copies of this shared variable from cRIO-9074 » Project Variables.lvlib » Host to Target » Stop RT in the Project Explorer to the block diagram.
- Temperature-cRIO I/O variable—Use this I/O variable to read current temperature values of the temperature chamber from the NI 9211.
  - Drag a copy of the Temperature-cRIO I/O Node from cRIO-9074 » Chassis » Thermocouple Mod » Temperature-cRIO(AI0) in the Project Explorer to the block diagram.
  - Right-click this I/O variable and select Timestamp » Show.
- PID VI—This VI uses PID to control the temperature of the temperature chamber by outputting the correct power to the lamp based on the current temperature, period, setpoint, and PID gains.
  - Right-click the output range input and select Create » Constant. Set the output high element of the constant to 100. Set the output low element of the constant to 0.
  - Wire the Setpoint shared variable to the Setpoint input of this VI.
- Wire the current temperature value from the Temperature-cRIO I/O node to the **process variable** input of this VI.
- Wire the PID Gains shared variable to the **PID gains** input of this VI.
- Wire the output of the Divide function to the **dt (s)** input of this VI.
- Wire the output of the Equal to Zero function to the **reinitialize?** input of this VI. This VI should be initialized only on the first iteration of the loop. This is true when the iteration terminal is equal to zero.

- **Fan shared variable**—Use this shared variable to read disturbance Boolean data from the host computer.
  - Drag a copy of this shared variable from *cRIO-9074*»**Projects**»**Variables**.lvlib»**Host to Target»**Fan* in the Project Explorer to the block diagram.
  - Right-click the shared variable and select **Access Mode»**Change to Read.

- **Select function**—Use this function to turn the temperature chamber fan on or off based on the value of the Fan shared variable.
  - Place a numeric constant on the block diagram. Right-click the constant and select **Representation»**DBL. Set the constant to **100**.
  - Wire the constant to the **t** input of the Select function.
  - Right-click the **f** input of the Select function and select **Create»**Constant. Set the constant to **0**.
  - Wire the output of the Fan shared variable to the **s** input of the Select function.

- **Fan-cRIO I/O variable**—Drag a copy of this I/O node from *cRIO-9074*»**Chassis»**PWM Mod»**Fan-cRIO* in the Project Explorer to the block diagram. A value of **0** will turn the fan off, and a value of **100** will output full power to the fan.

- **Lamp-cRIO I/O variable**—Drag a copy of this I/O node from *cRIO-9074*»**Chassis»**PWM Mod»**Lamp-cRIO* in the Project Explorer to the block diagram. This I/O node accepts values between **0** and **100**. The higher the value, the brighter the lamp.
Lesson 4  Timing Applications and Acquiring Data

❑ Target Data.ctl

  – Drag a copy of this shared variable from cRIO-9074»Project Files»Controls»Target Data.ctl in the Project Explorer to the block diagram.

  – Right-click the border of this Target Data.ctl cluster constant and select View Cluster as Icon to take up less block diagram space.

❑ Bundle By Name function

  – Wire the Target Data.ctl cluster constant to the input cluster input of this function.

  – Resize this function to show five elements.

  – Set the elements of this function in the following order from top to bottom: Timestamp, Setpoint, Temperature, Fan, Lamp Intensity.

  – Wire the timestamp output of the Temperature-cRIO I/O variable to the Timestamp element of this function.

  – Wire the output of the Setpoint shared variable to the Setpoint element of this function.

  – Wire the output of the Temperature-cRIO I/O variable to the Temperature element of this function.

  – Wire the output of the Fan shared variable to the Fan element of this function.

  – Wire the output of the PID VI to the Lamp Intensity element of this function.

❑ Target Data - RT single-process shared variable with a multi-element RT FIFO enabled—Use this shared variable to transfer every timestamp, setpoint, temperature, fan, and lamp intensity value from the deterministic loop in this exercise to the non-deterministic loop that you will create in the next exercise.

  – Double-click the cRIO-9074»Project Variables.lvlib»Intertask»Target Data - RT shared variable in the Project Explorer window to open the Shared Variable Properties window.

  – Notice that this shared variable has a cluster data type that matches Target Data.ctl and the output of the Bundle By Name function.
Select the RT FIFO category in the Shared Variable Properties window and notice that a multi-element RT FIFO is enabled. The RT FIFO stores 11 elements.

Use single-process shared variables with the RT FIFO option enabled to transfer data between the deterministic and non-deterministic loops on an RT target.

Click OK to close the Shared Variable Properties window.

Drag a copy of this shared variable from cRIO-9074\Project Variables.lvlib\Intertask\Target Data - RT in the Project Explorer to the block diagram.

- Unbundle By Name function
- Or function
- Flat Sequence structure
  - Place a True constant in this structure.
- Complete the wiring as shown in Figures 4-4 and 4-5.

8. Compare the flowchart you reviewed in Exercise 3-3 with your completed block diagram. Do they match? Are you accomplishing the tasks as designed?

9. Notice how the frames of the Timed Loop, error wires, and data flow control the order of operations in this deterministic loop.

10. Save and close the Chamber Control VI.

**Configure Chamber Control subVI in RT Target VI**

In this section, you will call and configure the Chamber Control subVI from the RT Target VI, which is the top-level VI for this application.

1. Open the RT Target VI.

2. Notice that the Initialization and Shutdown subVIs have already been created for you. You will learn more about these subVIs in a later exercise.
3. Modify the RT Target VI block diagram to configure the period and priority of the Timed Loop inside the Chamber Control subVI.
   - Right-click the Period input of the Chamber Control subVI and select **Create»Constant**. Set the constant to 100.
   - Right-click the Priority input of the Chamber Control subVI and select **Create»Constant**. Set the constant to 500.

4. Save the RT Target VI.

**Test**

1. Run the RT Target VI.
2. Select **Tools»Distributed System Manager** to open the Distributed System Manager.
3. View the current temperature of the temperature chamber.
   - In the Distributed System Manager, select the **Network Items»[x.x.x.x]»Thermocouple Mod»Temperature-cRIO** item, where [x.x.x.x] represents the IP address of your RT target, to display the current temperature.
   - Write the current value of the temperature:
     Current Temperature: ___________________ degrees (Celsius)
4. Test the Chamber Control subVI by using the Distributed System Manager to assign values to the Setpoint shared variable.
   - In the Distributed System Manager, select the **Network Items»[x.x.x.x]»Project Variables»Setpoint** item, where [x.x.x.x] represents the IP address of your RT target.
   - Set New Value to a value higher than the Current Temperature recorded in the previous step and click **Set**.
     Observe the temperature chamber, and confirm that the lamp turns on.
View the **Network Items\>`x.x.x</`\>`Thermocouple Mod</`\>`Temperature-cRIO** item, where \`x.x.x` represents the IP address of your RT target, to view the current temperature. After a while, the temperature should match the setpoint value you entered.

The deterministic Chamber Control loop outputs varying lamp intensities to control the temperature to match the setpoint.

5. Test the Chamber Control subVI by using the Distributed System Manager to assign values to the Fan shared variable.
   - In the Distributed System Manager, select the **Network Items\>`x.x.x</`\>`Project Variables</`\>`Fan** item, where \`x.x.x` represents the IP address of your RT target.
   - Set New Value to True and click Set.

   Observe the temperature chamber, and confirm that the fan turns on.

6. Stop the Chamber Control subVI and RT Target VI by using the Distributed System Manager to assign values to the Stop RT shared variable.
   - In the Distributed System Manager, select the **Network Items\>`x.x.x</`\>`Project Variables</`\>`Stop RT** item, where \`x.x.x` represents the IP address of your RT target.
   - Set New Value to True and click Set.

   Observe the temperature chamber, and confirm that the lamp and fan both turn off. This is done by the code contained in the Shutdown subVI.

   Notice that the RT Target VI has also stopped.

Note In a later exercise, you use a VI on the host computer to programmatically write to and read from these shared variables and I/O variables instead of using the Distributed System Manager.

**End of Exercise 4-2**
Communication

Exercise 5-1  
Course Project: Creating the Target VI

Goal

Create the target VI, which includes the non-deterministic loop and the deterministic loop.

Scenario

In this exercise, you create the rest of the target VI and non-deterministic loop for the project. This includes programming the non-deterministic tasks and using the deterministic loop created in Exercise 4-2.

Implementation

The RT Target VI contains three distinct sections: Initialization, Ongoing Tasks, and Shutdown. In Exercise 4-2, you created the deterministic loop in the Ongoing Tasks section of the RT Target VI.

In this exercise, you develop the non-deterministic loop in the Ongoing Tasks section. This loop executes the non-deterministic tasks of logging date to file and communicating with the host computer over the network.

You also modify the code in the Initialization and Shutdown sections of the RT Target VI.
Exploring the Project
In this section you will explore different items in the project.

1. **Open** Temperature Chamber Controller.lvproj from the <Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller directory.

2. Explore the File Module subVI provided to you. This VI is located in the Project Files folder of your project.

   The File Module subVI has Create, Write, and Close operations. This module logs the data into an ASCII file on the target.

3. Explore the Project Variables library in the Project Explorer.

   This library contains single-process shared variables with the RT FIFO option enabled and network-published shared variables with the RT FIFO option enabled for deterministic reading and writing while transferring data in the deterministic loop. The library also contains network-published shared variables with the RT FIFO option disabled for non-deterministic reading and writing while transferring data between the non-deterministic loop on the target VI and the host VI.

4. Examine the configuration of the Stop RT shared variable.
   
   - Right-click Stop RT located at cRIO-9074\Project Variables.lvlib\Host to Target\Stop RT in the Project Explorer and select Properties to view the configuration for this shared variable.
   
   - Notice that the Variable Type is set to Network-Published.

   Select the RT FIFO category and notice that the RT FIFO is enabled.

   By enabling the RT FIFO option for a network-published shared variable, you can share its data across a network without affecting the determinism in the VIs. This is necessary for the Stop RT network-published shared variable because it is also being read in the deterministic loop.

   Network-published shared variables with the RT FIFO option enabled automatically generate an invisible communication loop. In this application, the host VI will transfer the Stop RT value from the host to the communication loop, and the communication loop transfers the Stop RT value to the deterministic loop and non-deterministic loop using an RT FIFO.

   - Click OK to close the Shared Variable Properties window.
5. Examine the Target Data - network-published shared variable

- In the Project Explorer, right-click Target Data - network located at cRIO-9074»Project Variables.lvlib»Target to Host»Target Data - network and select Properties to view the configuration for this shared variable.
- Notice that the Variable Type is set to Network-Published.
- Select the RT FIFO category and notice that the RT FIFO is not enabled.

In this application, we use the Target Data - RT single-process shared variable with the RT FIFO option enabled to transfer the target data from the deterministic loop to the non-deterministic loop. We then use the Target Data - network network-published shared variable without the RT FIFO option enabled to transfer the target data from the non-deterministic loop to the host computer.

- Select the Network category and notice that the Use Buffering checkbox is not enabled.

In this application, the host computer only need to display the latest value of this shared variable.

- Click OK to close the Shared Variable Properties window.

Create Non-deterministic Loop in Logging and Network Communication subVI

In this section, you create a non-deterministic loop to log data to file and send the target data over the network to the host computer.

1. Open the Logging and Network Communication VI from the Project Explorer by double-clicking cRIO-9074»Project Files»Logging and Network Communication.vi.

2. Examine the VI priority of the Logging and Network Communication VI.

- Select File»VI Properties.
- Set Category to Execution.
- Notice that the Priority is set to normal priority. Because this application uses the Timed Loop priority scheme, all VIs should be set to normal priority to prevent undesired behavior.
- Click OK to close the VI Properties window.
3. Modify the block diagram to create the non-deterministic loop, as shown in Figure 5-1, using the following items:

- **While Loop**
  - Place a While Loop inside the While Loop on the block diagram.
  - You will use this While Loop to quickly read elements of the Target Data - RT shared variable until its RT FIFO is empty.
  - Right-click the border of the inner While Loop and select **Add Shift Register**.
Note Notice that the non-deterministic loop is a While Loop in a normal priority VI. Because the deterministic loop is a Timed Loop, the non-deterministic While Loop will only execute its code when the deterministic Timed Loop is sleeping. If this application required more loops with different levels of priority, you could add additional Timed Loops set to a lower priority than the deterministic Timed Loop.

- Target Data - RT single-process shared variable with multi-element RT FIFO enabled—Use this shared variable in the non-deterministic loop to read target data written by the deterministic loop. In Exercise 4-2, you wrote the timestamp, setpoint, temperature, fan status and lamp intensity values to this shared variable in the deterministic Timed Loop.
  - Drag a copy of this shared variable from cRIO-9074\Project Variables.lwlib\Intertask\Target Data - RT in the Project Explorer to the block diagram.
  - Right-click this shared variable and select **Timeout»Show**.
  - Right-click the ms timeout input and select **Create»Constant**. Set the constant to 0.

  This configures this shared variable to immediately return the next available element in the RT FIFO. If the RT FIFO is empty, this shared variable immediately returns a true value on its timed out? output and does not return an error.

- Case structure
  - Wire the error out output of Target Data - RT shared variable to the Case selector input of this structure.
  - In the **Error** case, place down a False constant and wire the case as shown in Figure 5-2.
Case structure

- Place a Case structure inside the No Error case of the Case structure on the block diagram.
- Wire the timed out? output of the Target Data - RT shared variable to the Case Selector input of this structure.
- In the True case, place a True constant and wire the case as shown in Figure 5-3.
Target Data - network-published shared variable with RT FIFO disabled—Use this shared variable to transfer target data from the non-deterministic loop over the network to the host computer.

- Drag a copy of this shared variable from `cRIO-9074\Project Variables.lvlib\Target to Host\Target Data - network` in the Project Explorer to the False case of the Case structure in the block diagram, as shown in Figure 5-1.

- Wire the output of the Target Data - RT shared variable to the input of the Target Data - network shared variable.

File Module subVI

- Drag a copy of this subVI from `cRIO-9074\Project Files\File Module` in the Project Explorer to the False case of the Case structure in the block diagram, as shown in Figure 5-1.

- Right-click the Operation input and select `Create»Constant`. Set the constant to Write.

- Wire the output of the Target Data - RT shared variable to the Target Data input of this subVI.

False constant—Place this constant in the False case of the Case structure inside the No Error case of the previous Case structure, as shown in Figure 5-1.

Stop RT network-published shared variable with a single-element RT FIFO enabled

- Drag a copy of this shared variable from `cRIO-9074\Project Variables.lvlib\Host to Target\Stop RT` in the Project Explorer to the block diagram.

Unbundle By Name function

Or function
Lesson 5  Communication

- Compound Arithmetic function
  - Resize the function to show three elements.
  - Right-click the function and select Change Mode»OR.
- Wait Until Next ms Multiple function—This function allows the outer While Loop to yield execution after each time the inner While Loop finishes reading all available elements of the target data RT FIFO.
  - Place this function in the outer While Loop.
  - Wire the Period control to the input of this function.
- Complete the wiring of the block diagram as shown in Figure 5-1.

4. Save and close the Logging and Network Communication VI.

**Configure Chamber Control subVI in RT Target VI**

In this section, you will call and configure the Chamber Control subVI from the RT Target VI, which is the top-level VI for this application.

1. Open the RT Target VI from the Project Explorer.
2. Modify the Ongoing Tasks section of the RT Target VI block diagram to execute the non-deterministic loop inside the Logging and Network Communication subVI simultaneously with the deterministic loop inside the Chamber Control subVI, as shown in Figure 5-4, using the following items.

- **Logging and Network Communication subVI**
  - Drag a copy of this subVI from `cRIO-9074»Project Files»Logging and Network Communication.vi` in the Project Explorer to the block diagram.
  - Right-click the Period input of the Logging and Network Communication subVI and select **Create»Constant**. Set the constant to 500.

- **Merge Errors function**

- **Complete the wiring for the block diagram as shown in Figure 5-4.**

3. Save the RT Target VI.
Initialization
In this section, you will examine and modify the Initialize subVI. This subVI initializes the values for the resources and variables used in the application.

1. Open the Initialize subVI from the Project Explorer by double-clicking cRIO-9074\Project Files\Initialize.vi.

2. Examine the block diagram.

- Notice the code that initializes the network-published shared variables (PID Gains, Setpoint, Fan, Stop RT, and Target Data - network) to safe default values.

- Notice the code that uses a While Loop to ensure that the initialized values of the network-published shared variables have propagated through the Shared Variable Engine before continuing.

  If the initialized values of the network-published shared variables do not propagate through the Shared Variable Engine within 10 seconds, the timed out? indicator of the Initialize subVI will return a boolean value of True.

- Notice the code that ensures the RT FIFO of the Target Data - RT single-process shared variable is empty before continuing by writing an element to the RT FIFO and reading from the RT FIFO until an underflow error is detected.
3. Modify the block diagram of the Initialize subVI to create a log file and turn on the USER1 LED on the cRIO-9074, as shown in Figure 5-5, using the following items:

![Figure 5-5. Initialize VI Block Diagram](image)

- File Module subVI—Use this subVI to create the log file.
  - Drag a copy of this subVI from cRIO-9074\Projects\File Module in the Project Explorer to the block diagram.
  - Right-click the Operation input and select Create\Constant. Set the constant to Create.
  - Right-click the file path input and select Create\Constant. Set the constant to C:\target_data.txt.

- RT LEDs VI—This VI will turn on the USER1 LED on the cRIO-9074 to alert the user that the RT target has been completed all initialization tasks and is now running the main code.
  - Right-click the LED Num input and select Create\Constant. Set the constant to 0.
  - Right-click the State input and select Create\Constant. Set the constant to 1.

- Complete the wiring for the block diagram as shown in Figure 5-5.

4. Save and close the Initialize VI.
Shutdown
In this section, you will examine and modify the Shutdown subVI. This subVI safely shuts down the application.

1. Open the Shutdown subVI from the Project Explorer by double-clicking cRIO-9074»Project Files» Shutdown.vi.

2. Examine the Shutdown VI block diagram.
   - Notice that this VI turns the lamp and fan off by setting the lamp and fan intensities to 0. Shutdown code often involves setting hardware to safe states.
   - Notice that this VI sets the Stop RT shared variable to true. This will notify the host computer that the RT Target VI has shut down and is no longer running.
   - Notice that all the error in inputs of the I/O variables and shared variable are unwired. This ensures that each I/O variable and shared variable will execute even if a previous error has occurred in the application.

3. Modify the block diagram of the Shutdown subVI to close the log file and turn off the USER1 LED on the cRIO-9074, as shown in Figure 5-6, using the following items:
Figure 5-6. Shutdown VI Block Diagram

- Merge Errors function
  - Resize this function to show two additional elements.

- RT LEDs VI—This VI will turn off the USER1 LED on the cRIO-9074 to alert the user that the RT target has exited the main code and has shut down.
  - Right-click the LED Num input and select Create»Constant. Set the constant to 0.
  - Right-click the State input and select Create»Constant. Set the constant to 0.

- File Module subVI—Use this subVI to close the log file properly.
  - Drag a copy of this subVI from cRIO-9074»Project Files»File Module in the Project Explorer to the block diagram.
  - Right-click the Operation input and select Create»Constant. Set the constant to Close.

- Complete the wiring for the block diagram as shown in Figure 5-6.

4. Save and close the Shutdown VI.
Test

1. Notice that the USER1 LED on the cRIO-9074 is currently unlit.
2. Run the RT Target VI.
3. Verify that the USER1 LED on the cRIO-9074 is now lit.
4. Select Tools»Distributed System Manager to open the Distributed System Manager.
5. View the current temperature of the temperature chamber.
   - In the Distributed System Manager, select the Network Items » [x.x.x.x] » Thermocouple Mod » Temperature-cRIO item, where [x.x.x.x] represents the IP address of your RT target, to display the current temperature.
   - Write the current value of the temperature:
     
     Current Temperature: _____________________ degrees (Celsius)

6. Test the RT Target VI by using the Distributed System Manager to assign values to the Setpoint shared variable.
   - In the Distributed System Manager, select the Network Items » [x.x.x.x] » Project Variables » Setpoint item, where [x.x.x.x] represents the IP address of your RT target.
   - Set New Value to a value higher than the Current Temperature recorded in the previous step and click Set.
   - Observe the temperature chamber, and confirm that the lamp turns on.
   - View the Network Items » [x.x.x.x] » Thermocouple Mod » Temperature-cRIO item, where [x.x.x.x] represents the IP address of your RT target, to view the current temperature. After a while, the temperature should match the setpoint value you entered.

The deterministic Chamber Control loop outputs varying lamp intensities to control the temperature to match the setpoint.
7. Test the RT Target VI by using the Distributed System Manager to assign values to the Fan shared variable.

- In the Distributed System Manager, select the Network Items»[x.x.x.x]»Project Variables»Fan item, where /x.x.x.x/ represents the IP address of your RT target.
- Set New Value to True and click Set.

Observe the temperature chamber, and confirm that the fan turns on.

8. Stop the RT Target VI by using the Distributed System Manager to assign values to the Stop RT shared variable.

- In the Distributed System Manager, select the Network Items»[x.x.x.x]»Project Variables»Stop RT item, where /x.x.x.x/ represents the IP address of your RT target.
- Set New Value to True and click Set.

Observe the temperature chamber, and confirm that the lamp and fan both turn off. This is done by the code contained in the Shutdown subVI.

Notice that the RT Target VI has also stopped.

9. Verify that the USER1 LED on the cRIO-9074 is now unlit.

10. Check the log file output.

- Open a Internet Explorer web browser.
- Navigate to ftp://{x.x.x.x}/ to view the contents of the RT target hard drive.

Note [x.x.x.x] represents the IP address of your RT target.

- Select View»Open FTP Site in Windows Explorer to view the contents of the RT target hard drive in Windows Explorer.
- Double-click the target_data.txt file to open the log file.
Lesson 5  Communication

- View the contents of the tab-delimited log file. You should see data logged.
- Close the log file.
- Delete target_data.txt in Windows Explorer.
- Close Windows Explorer.
- Close the Internet Explorer web browser.

Note  In a later exercise, you use a VI on the host computer to programmatically write to and read from these shared variables and I/O variables instead of using the Distributed System Manager.

End of Exercise 5-1
Lesson 5  Communication

Exercise 5-2  Course Project: Creating the Project Host VI

Goal

Create the host VI for the project from the provided flowchart.

Scenario

In this exercise, you create the Project Host VI which will run on your host computer. This VI will be the user interface for the temperature chamber application. This VI will use network-published shared variables to communicate with the RT Target VI running on the RT target. You created the RT Target VI in Exercise 4-2 and Exercise 5-1.

Implementation

1. Open Temperature Chamber Controller.lvproj from the <Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller directory.

2. Add the Project Host VI to the project.
   - In the Project Explorer, right-click My Computer and select Add»File.
   - Select the Project Host.vi located in the <Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller directory. Click Add File.

   The host VI does not run on the RT target. It runs on the host computer to provide a user interface. The My Computer target is the host computer in your project.

3. Examine the configuration of the Fan shared variable.
   - Right-click Fan located at cRIO-9074\Project Variables.lvlib\Host to Target\Fan in the Project Explorer and select Properties to view the configuration for this shared variable.
   - Notice that the Variable Type is set to Network-Published.
❑ Select the RT FIFO category and notice that the RT FIFO is enabled.

By enabling the RT FIFO option for a network-published shared variable, you can share its data across a network without affecting the determinism in the VIs. This is necessary for the Fan network-published shared variable because it is also being read in the deterministic loop.

Network-published shared variables with the RT FIFO option enabled automatically generate an invisible communication loop. In this application, the Project Host VI will transfer the Fan, PID Gains, Setpoint, and Stop RT values from the host computer to this communication loop, and the communication loop transfers these values to the deterministic loop using RT FIFOs.

❑ Notice in the RT FIFO category that the FIFO Type is set to Single Element. Notice in the Network category that the Use Buffering checkbox is disabled. This application only needs to know the latest value of this shared variable.

❑ Click OK to close the Shared Variable Properties window.

4. Notice that the PID Gains, Setpoint, and Stop RT shared variables are also configured as network-published shared variables with a single-element RT FIFO enabled.
5. Open the Project Host VI by double-clicking it in the Project Explorer.

![Host User Interface](image)

**Figure 5-7.** Host User Interface

6. The user interface shown in Figure 5-7 has already been completed for you. From this user interface, the user will set the Setpoint and PID Gains. The user can click the Apply Disturbance button at any time during the temperature ramp to turn on the fan. The user can also click the Stop Target button at any time to stop the target VI from running.

   The current setpoint and temperature will appear as plots on the Temperature Chamber chart during the temperature ramp. The Lamp Intensity, Fan Status, and Current Timestamp indicators display their current values during the temperature ramp.

7. Modify the block diagram to communicate with the RT Target VI over the network, as shown in Figure 5-8, using the following items:
Figure 5-8. Project Host VI Block Diagram

- Stop RT network-published shared variable with single-element RT FIFO enabled
  - Drag two copies of this shared variable from cRIO-9074»Project Variables.lvlib»Host to Target»Stop RT in the Project Explorer to the block diagram.

- Select function
  - Place a string constant on the block diagram and wire it to the t input of this function. Set the constant to Stopped.
  - Right-click the f input of this function and select Create»Constant. Set the constant to Running.

- Fan network-published shared variable with single-element RT FIFO enabled
  - Drag a copy of this shared variable from cRIO-9074»Project Variables.lvlib»Host to Target»Fan in the Project Explorer to the block diagram.
❑ PID Gains network-published shared variable with single-element RT FIFO enabled
  – Drag a copy of this shared variable from cRIO-9074»Project Variables.lvlib»Host to Target»PID Gains in the Project Explorer to the block diagram.

❑ Setpoint network-published shared variable with single-element RT FIFO enabled
  – Drag a copy of this shared variable from cRIO-9074»Project Variables.lvlib»Host to Target»Setpoint in the Project Explorer to the block diagram.

❑ Target Data - network network-published shared variable with RT FIFO and network buffer disabled—The Project Host VI receives data from the RT Target VI with this shared variable.
  – Drag a copy of this shared variable from cRIO-9074»Project Variables.lvlib»Target to Host»Target Data - network in the Project Explorer to the block diagram.

❑ Unbundle by Name function
  – Resize this function to show five elements.

❑ Format Date/Time String function—Use this function to display the timestamp as a string containing the date and time.

❑ Bundle function—Use this function to bundles the setpoint and current temperature data into a cluster. The Temp Chart indicator displays both the setpoint and current temperature data on a multi-plot chart.

❑ Case structure—this structure will set the Stop RT shared variable to true if the user presses the Stop Target control.
  – Wire the output of the Stop Target control to the Case Selector input of this structure.
  – In the True case, place down a Stop RT shared variable and a Merge Errors function. Wire the True case as shown in Figure 5-8.
  – In the False case, wire the error input tunnel to the error output tunnel.
☐ Wait Until Next ms Multiple function—Use this function in the While Loop to provide time to the processor.
  – Right-click the input and select Create Constant. Set the constant to 100.

8. Save the VI.

Test

1. Open the RT Target VI and run it. This runs the VI that you built in Exercises 4-2 and 5-1 on the RT Target.

2. Verify that the USER1 LED on the cRIO-9074 is lit.

3. On the Project Host VI, verify that the controls are set to the following values:
   - Setpoint: 25
   - PID Gains: proportional gain: 15.000
   - PID Gains: integral time: 0.050
   - PID Gains: derivative time: 0.000

  Note The PID Gains parameters adjust the control response of the lamp intensity to the setpoint and current temperature.

4. Run the Project Host VI.

   You should see the current setpoint and temperature plots appearing on the Temperature Chamber chart, and you should see the Lamp Intensity, Fan Status, and Current Timestamp indicators displaying their current values during the temperature ramp.

5. Verify that the RT Target Status string indicator displays Running.

6. Set the Setpoint control to a value higher than the current temperature

   You should see the lamp intensity increase in response.
7. Press **Apply Disturbance** to toggle the fan on and off.
   
   Observe the temperature chamber and verify that the fan toggles on and off. Verify that the Fan Status boolean indicator also toggles on and off.

8. Press **Stop Target**.

9. On the front panel of the RT Target VI, notice that the run arrow indicates that the RT Target VI is no longer running. Also notice that both the lamp and fan are turned off by the Shutdown subVI in the RT Target VI.

10. On the Project Host VI, verify that the RT Target Status string indicator displays **Stopped**.

11. Verify that the USER1 LED on the cRIO-9074 is now unlit.

12. Press **Stop User Interface** to stop the Project Host VI.

13. Check the log file output.

   - Open a Internet Explorer web browser.
   - Navigate to `ftp://[x.x.x.x]/` to view the contents of the RT target hard drive.

   ![Note](image) 
   
   `[x.x.x.x]` represents the IP address of your RT target.
   
   - Select **View»Open FTP Site in Windows Explorer** to view the contents of the RT target hard drive in Windows Explorer.
   - Select **View»Refresh**.
   - Copy `target_data.txt` to the `<Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller` directory.
   - Delete the `target_data.txt` file in Windows Explorer.
   - Close Windows Explorer.
   - Close the Internet Explorer web browser.
14. View the log file in Excel.
   - Open Excel.
   - Drag `<Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller\target_data.txt` into the empty Excel worksheet.
   - Resize columns to view data.
   - Close Excel without saving.

15. Save and close the RT Target VI and Project Host VI when finished.

**End of Exercise 5-2**
Exercise 5-3

Network Streams

Goal

Create an application that streams buffered data from the RT target to the host computer using the Network Streams functions.

Scenario

Your RT target is generating waveform data, and you must transfer every piece of generated data to your host computer for further processing. Because you do not want to lose any data, you will need a buffered method of network communication.

In this exercise, you will use Network Streams functions to transfer data from the NS Writer VI running on the RT target to the NS Reader VI running on the host computer. The NS Writer VI will generate the data on the RT target in a deterministic loop. Because the Network Streams functions are non-deterministic, the NS Writer VI will use Network Streams functions in a non-deterministic loop.

The NS Reader VI on the host computer and the deterministic Timed Loop in the NS Writer VI on the RT target has already been completed for you. Complete the non-deterministic Timed Loop that uses the Network Streams VIs to stream data from the RT target to the host computer.

Implementation

Writer VI - Deterministic Loop

1. Open Network Streams.lvproj in the <Exercises>\LabVIEW Real-Time 1\Network Streams directory.

2. Configure the CompactRIO target.
   - Right-click the RT CompactRIO target in the Project Explorer and select Properties.
   - In the General category, set the IP Address to the IP address of your cRIO-9074.
   - Click OK.
3. Examine the shared variables contained within the library on the RT target.

☐ Notice that the Waveform Type and Stop Generation are network-published shared variables with a single-element RT FIFO enabled. These shared variables can be written to and read from both the RT target and host computer.

☐ Notice that the Result Data is a single process shared variable with a multi-element RT FIFO enabled. This shared variable can be used to share buffered values between multiple loops on the RT target.

4. Open the NS Writer VI under the RT target.

5. Examine the block diagram of the NS Writer VI.

☐ Notice that the shared variables are being initialized before the deterministic Timed Loop begins.

☐ Notice that the deterministic Timed Loop is generating waveform points and writing each point to the Result Data shared variable.
**NS Writer VI - Non-deterministic Loop**

In this application, you want to send buffered values from the RT target to the host computer. You will implement this functionality using the Network Streams functions. Because the Network Streams functions are not deterministic, you should only use Network Streams functions in a non-deterministic loop.

1. Modify the block diagram to send buffered values from the NS Writer VI on the RT target to the NS Reader VI on the host computer, as shown in Figure 5-9, using the following items:

   - **Create Network Stream Writer Endpoint**—This function creates the writer endpoint of the network stream. This function will connect to the Create Network Stream Reader Endpoint in the NS Reader VI to establish a connection and create the network stream.

     - Right-click the writer name input and select `Create»Constant`. Set the constant to `MyWriter`.
     - Right-click the write buffer size input and select `Create»Constant`. Set the constant to `10000`. This sets the size of the writer buffer to 10,000 elements.
     - If left unwired, the timeout in ms input will have a default value of –1, which configures this function to wait indefinitely for a connection.

   ![Figure 5-9. NS Writer VI Block Diagram](image)
 Numeric constant
  - Right-click the numeric constant and select **Representation»DBL**.
  - Wire the numeric constant to the data type input of the Create Network Stream Writer Endpoint function. This specifies a double-precision numeric data type for the elements you want to write to the stream.

 Result Data single process shared variable with multi-element RT FIFO enabled

 Two Unbundle By Name functions
  - Set the leftmost Unbundle by Name function to **Code** and set the rightmost to **Status**.

 Case structure
  - Wire the output of the Unbundle By Name function to the case selector terminal for the Case structure.
  - Change the selector label for the 1 case to **-2220**. This case executes when an underflow error occurs.
  - Place a Clear Errors VI and the Stop Generation shared variable in the **-2220** case.
  - Wire the **-2220** case as shown in Figure 5-10.

![Figure 5-10. -2220 case of Case Structure](image)
– Switch to the 0, Default case of the Case structure. Place a Write Single Element to Stream function and a False Boolean constant in this case.

– Wire the 0, Default case as shown in Figure 5-9.

❑ Or function—Notice that the non-deterministic Timed Loop will exit if the Stop Generation shared variable is true and the Result Data shared variable RT FIFO is empty. The Timed Loop will also exit if an error occurs.

❑ Flush Stream—Configure this function to send all data from the Network Stream writer endpoint to the Network Stream reader endpoint and wait indefinitely until all elements have been read by the Network Stream reader endpoint.

– Right-click the wait condition input and select Create » Constant. Set the constant to All Elements Read From Stream.

– Right-click the timeout in ms input and select Create » Constant. Set the constant to -1.

❑ Destroy Stream Endpoint

**NS Reader VI**
The reader VI on the host computer has already been completed for you in this exercise.

1. Open the NS Reader VI under the My Computer target.

2. Examine the block diagram.

  ❑ Notice the Create Network Stream Reader Endpoint function will establish a connection to the writer endpoint specified by the writer url control.

  ❑ Notice a value of -1 is wired to the # elements input of the Read Multiple Elements from Stream function. This configures this function to read all available elements from the stream.
Notice the code in the False case of the Case structure will only execute when the Read Multiple Elements from Stream function does not time out and no error has occurred.

The code in the false case updates the number of elements that have a value greater than or equal to the user-specified upper limit.

Notice how the stop Boolean control can tell the NS Writer VI on the RT target to stop writing data to the network stream.

Test

1. Run the NS Writer VI. The writer VI is now running on the RT target and the Create Network Stream Writer Endpoint function is waiting indefinitely for a connection from a reader endpoint.

2. If a conflict resolution dialog appears, click Apply.

3. On the front panel of the NS Reader VI, set the writer url string control to //x.x.x.x/MyWriter where x.x.x.x is the IP address of your RT target.

4. If a conflict resolution dialog appears, click Apply.

5. Run the NS Reader VI. The writer and reader endpoints should now establish a connection, and you should see the waveform chart on the NS Reader VI update with values streamed from the NS Writer VI.

6. Click the Stop button on the NS Reader VI.

   This should cause the NS Writer VI on the RT target to stop writing data to the network stream and flush the remaining elements to the reader endpoint. When the reader endpoint has finished reading those remaining flushed elements, the Flush Elements function on the NS Writer VI will resume and execute the Destroy Stream Endpoint function. Destroying the network stream on the NS Writer VI will cause an error on the NS Reader VI, which causes the NS Reader VI to exit the While Loop and destroy the stream on the reader endpoint.

7. Save and close all VIs when finished.

End of Exercise 5-3
Notes
Verifying Your Application

Exercise 6-1

Distributed System Manager

Goal

Use the Distributed System Manager to verify the CPU and memory usage, monitor VI states, and log alerts on an RT target.

Scenario

In this exercise, you will use the Distributed System Manager to verify the CPU and memory usage, monitor VI states, and log alerts on the RT target while running the Temperature Chamber Controller application.

Implementation

1. Open <Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller\Temperature Chamber Controller.lvproj.

2. In the Project Explorer, select Tools>Distributed System Manager. This opens the NI Distributed System Manager window.

3. Under Network Items folder, select the network item that corresponds with the IP address of your RT Target.
4. Examine the CPU/Memory tab.

- In the Auto View section, select the CPU/Memory tab. You can use this tab to monitor the CPU and memory usage on your RT target.

- Write the CPU and memory statistics in the following blanks then click **OK**.
  
  Total CPU: ___________________
  
  Total memory: ___________________
  
  Available memory: ___________________

- Open and run the RT Target VI from the Project Explorer.

- While observing the CPU/Memory tab in the Distributed System Manager, use the host VI to communicate to the RT target, set the Setpoint, and apply disturbance. What happened to the memory and CPU usage?

- Click the Stop Target button on the host VI to stop the RT Target VI.

- Click the Stop User Interface button on the host VI to stop the Project Host VI.

5. Examine the VI States tab.

- Enable VI Server. In order to use the VI States tab, you must enable VI Server on your RT target.
  
  - Right-click the RT target in the Project Explorer and select **Properties**.
  
  - Select the **VI Server** category and enable the TCP/IP checkbox.
  
  - In the Machine Access section in the VI Server category, verify * is in the Machine access list and has a green checkmark next to it. This allows all machines to access the RT target through VI Server.
  
  - Click **OK** to exit the dialog box.
  
  - Right-click the RT target in the Project Explorer and select **Deploy** to load the configuration changes to the target.
In the Distributed System Manager, select the **VI States** tab and click **Start Monitoring**.

Examine the Status column.

Select the RT Target VI in the VI Name column.

Click **Start VI** to start the RT Target VI on the RT target.

Notice the updated values in the Status column.

Select the RT Target VI in the VI Name column and click **Stop VI**.

Click **Stop Monitoring**.

6. Examine the Alerts tab.

Select the Alerts tab.

Enable the **Log Alert when memory usage is above** checkbox.

Notice that the Recent Alerts list will populate when the RT target memory usage is above the user-specified amount and will stop populating when the RT target memory usage is below the user-specified amount.

Disable the **Log Alert when memory usage is above** checkbox.

Enable the **Log Alert when CPU usage** is above checkbox.

Notice that the Recent Alerts list will populate when the RT target CPU usage is above the user-specified amount and will stop populating when the RT target CPU usage is below the user-specified amount.

Disable the **Log Alert when CPU usage** is above checkbox.

Enable the **Log Alert when VI changes state** checkbox.

Open the RT Target VI from the Project Explorer and run it.

Notice that the Recent Alerts list populated with the updated states of VIs on the RT target.
☐ Click the Abort button on the RT Target VI to stop the VI.

☐ Disable the Log Alert when VI changes state checkbox.

☐ Click Save in the Alerts tab.

☐ Save the log file as <Exercises>\LabVIEW Real-Time 1\DSM alerts.txt.

☐ In Windows Explorer, navigate to and open <Exercises>\LabVIEW Real-Time 1\DSM alerts.txt to view the log file.

7. Close all open VIs.

8. Close Temperature Chamber Controller.lvproj.

9. Close the Distributed System Manager.

**End of Exercise 6-1**
Deploying Your Application

Exercise 7-1

Deploy an Application

Goal

Prepare the project for deployment and build a stand-alone real-time application that you can set to run automatically when you power on the RT target.

Scenario

You have completed code development for a temperature chamber controller application. Now, you may want to deploy and embed a stand-alone version of the application on the RT target and launch it automatically when the target boots.

Normally, you need to prepare your application for deployment by reviewing your code for unsupported functions, such as functions that modify front panel objects and functions that use technologies specific to other operating systems. For example, front panel Property Nodes do not work because they modify front panel objects, and ActiveX VIs will not work because ActiveX is a technology specific to the Windows OS. However, the temperature chamber controller application that you have built does not have any unsupported functions, so you can begin the steps for deployment immediately.

Implementation

1. Reboot the CompactRIO target by pressing the reset button.
2. Open<br>\<Exercises>\LabVIEW Real-Time 1\Temperature Chamber Controller\Temperature Chamber Controller.lvproj.<br>
3. Close all VIs before building an application. You cannot build an application from a VI that is in memory.
4. Build a stand-alone real-time application.
   - In the Project Explorer, right-click **Build Specifications** under the RT target and select **New» Real-Time Application**.
   - In the **Build specification name** textbox, enter **Temperature Chamber Controller**.
   - For **Category**, select **Source Files**.
   - Select **RT Target VI** and click the uppermost right arrow to add it to the **Startup VIs** list.

![Figure 7-1. Build Specifications Source Files Configuration](image)

---

**Lesson 7  Deploying Your Application**

LabVIEW Real-Time 1 Exercises  
7-2  
ni.com
Lesson 7  Deploying Your Application

- Examine the settings in the other Categories but do not make any changes.
- Click OK.
- Right-click the **Temperature Chamber Controller** build specification in the Project Explorer and select **Build**. Click **Done**.

5. Set the stand-alone real-time application to run automatically each time you power on the RT target.
- Right-click the **Temperature Chamber Controller** build specification in the Project Explorer and select **Run as startup**.

**Note**  The Run as startup shortcut menu item sets the application as the startup application, deploys the application to the target, and prompts you to reboot the RT target.

After the application is deployed, each time the RT target is powered on, the application starts. You can disable the VI from starting automatically by right-clicking the build specification or by deploying any other RT project to the target.
- When prompted to reboot the RT target, click **Yes**.
  
  Rebooting takes approximately 1 minute. After rebooting the hardware, the application you built launches.

6. Test the stand-alone real-time application.
- Right-click the RT Target in the Project Explorer and select **Disable Autodeploy Variables**.

**Note**  By default, when you run a VI, LabVIEW automatically deploys all shared variables, including I/O variables, that the VI references. Automatic variable deployment is convenient during VI development because it allows you to run a VI simply by pressing the Run button, without separately deploying or redeploying variables that the VI references. However, in some cases it is useful to disable automatic variable deployment.

If you attempt to run a host VI that references a variable hosted on an RT target that is running a startup VI, LabVIEW returns a deployment conflict unless you disable automatic variable deployment. If automatic variable deployment is enabled when you attempt to run the VI, LabVIEW attempts to deploy all the variables that the VI references and returns a conflict when one or more of those variables is hosted on a device that is running a startup VI. In this case, disabling automatic variable deployment on the RT target hosting the variables allows you to run the host VI without encountering a deployment conflict.
Lesson 7  Deploying Your Application

- Open the Project Host VI of the project.
- Verify that the USER1 LED on the cRIO-9074 is lit before you continue.

  Once the USER1 LED is lit, the RT Target VI is ready to communicate with the Project Host VI.
- Run the Project Host VI. Test the application. You should see the same activity as before.
- When done testing, click the **Stop Target** button on the Project Host VI to stop the stand-alone real-time application.
- Verify that the USER1 LED on the cRIO-9074 is now unlit. Verify that the RT Target Status indicator on the Project Host VI displays **Stopped**.
- Click the **Stop User Interface** button on the Project Host VI to stop the VI.

7. Unset the application as the startup application so it no longer starts automatically each time the RT target is powered on.
- Right-click the RT target in the Project Explorer and select **Connect**.
- Right-click the **Temperature Chamber Controller** build specification and choose **Unset as startup**.
- Right-click the **Temperature Chamber Controller** build specification and select **Deploy** to disable the automatic startup of the application.

  When the conflict message, **This application is being deployed, but it is not currently set to automatically launch when the target reboots.** appears, click **OK**.

8. Save and close the project and VIs.

**End of Exercise 7-1**
Course Slides

This appendix contains the Course Slides.

Topics

A. Introduction to Real-Time
B. Configuring Your Hardware
C. Real-Time Architecture: Design
D. Timing Applications and Acquiring Data
E. Communication
F. Verifying Your Application
G. Deploying Your Application
Lesson 1
Identifying Real-Time Application Requirements & Design

TOPICS
A. Analyzing Your Real-Time Application
B. RT Target Considerations
C. Host Considerations

A. Analyzing Your RT Application

Identify Tasks
• Before coding, think through the high-level design of your application by defining tasks and actions your application must perform
• Examples:
  - Control
  - Log
  - Transfer data to and from user interface
Identify Performance Requirements

- Determinism – separate deterministic and non-deterministic tasks
- Response time – if response time needs to be fast and/or reliable, the task may need to be deterministic

Identify Performance Requirements

- CPU Efficiency
- Throughput – determine how much data needs to be transferred within a given amount of time

Identify Task Timing

- What causes the task to execute?
- What causes the task to sleep?
- How frequently does the task execute?
Identify Task Timing
Determine timing behavior of each task
• Periodic task – executes at regular intervals
• Non-periodic task – executes when triggered by a non-periodic event

Identify Task Timing
Examples of Periodic Tasks
• Execute PID algorithm to control temperature chamber at a minimum rate of 1 kHz
• Transfer data between the user interface and RT target at a minimum rate of 10 Hz
• Log temperature chamber data at least once per minute

Identify Task Timing
Examples of Non-Periodic Tasks
• Execute a profile of multiple temperature setpoints when user presses Start Profile button on user interface
• Execute safe shutdown procedure if user presses Stop button on user interface or if temperature exceeds maximum threshold
Identify Data Relationships Between Tasks

- List all data that needs to be transferred
- List the writer(s) and reader(s) of each piece of data
- List how each piece of data should theoretically be transferred
  - Only need latest value
  - Need every value

Identify Reliability Issues

- How long will the application run?
  - Minutes, days, weeks, indefinitely
- How accessible is the application?
  - Easy access in laboratory
  - Remote or hazardous location
- Application operators do not know how to program code
- Possible effects on application
  - Implement redundant components
  - Create diagnostic tool for operators
  - Monitor memory and disk space in application

Additional Considerations

- Appropriate time to wait for new data
- Throughput
- Response time and latency
- Ease of implementation
Identify Reliability Issues

What failures could occur and how should the application handle them?

• Hardware failure examples
  – Network cable is disconnected
  – Hardware breaks down

• Software failure examples
  – Invalid inputs (i.e., out of range value from operator)
  – Insufficient memory

Identify Reliability Issues

• If the application needs to shut down, what is a safe shutdown procedure?
  – Set all I/O to safe states (i.e., turn heater off)
  – Log timestamp and reason for shutdown to file
  – Properly close all open references and connections
  – Display dialog on user interface
  – Trigger other hardware components to safely shutdown

Identify Reliability Issues

• If applicable, perform a more detailed reliability analysis on the application, such as Failure Mode and Effects Analysis (FMEA)
  • Perform risk analysis on application
    – Prioritize which risks and potential failures to address
    – Actively acknowledge which risks are acceptable
B. RT Target Considerations

- Performance
  - Able to meet requirements such as loop rate?
- Size of hard disk
  - Enough disk space for files such as log files?
- Size of hardware
  - Does application have physical space limitations?
- Ruggedness
  - Is the target located in a harsh environment?
- Symmetric multiprocessing (SMP)
  - Does application require multiple simultaneous deterministic loops?

RT Target Considerations

- Do you need multiple RT targets?
  - Necessary physical locations of RT target
  - Need more resources (performance, I/O, etc.)
  - Implement redundant RT hardware

Real-Time Targets

Refer to ni.com for detailed descriptions and specifications of all available RT targets

- I/O count
- Dimensions
- Ruggedness
- Loop speed
C. Host Considerations

What type of interface does the application need?
• Diagnostic
• Analysis system
• Configuration tool
• Operator interface

Host Considerations

Diagnostic interface
• Check CPU and memory usage
• Check for error state

Example diagnostic interfaces
• External LEDs flash to indicate system status
• Distributed System Manager
• Host VI displaying diagnostics

Host Considerations

Analysis system interface
• Receives data from target
• Analyzes and displays data

Example analysis system interface
• Host computer periodically receives, processes, logs data to file, and activates alarm if necessary
Host Considerations

Configuration tool interface
• Apply new configuration to application
• Restart RT target

Example configuration tool interfaces
• MAX
• Host VI with configuration and restart functionality
• Host VI to replicate and backup an RT target disk image

Host Considerations

Operator interface
• Allows for user interaction

Example operator interfaces
• Host VI that allows operator to start and stop a variety of tasks
• Web client that allows operator to control and monitor an application

Host Considerations

Where does the interface need to be?
• Different components can be on the same host computer or different host computers
Examples:
• Embedded vehicle system – use diagnostic and configuration interfaces on a single host computer
• Data logger system – use configuration interface on a host computer and implement analysis interface on multiple host computers on the network
• Machine control system – use operator interface on a host computer to control and monitor the application
Exercise 1-1: Configure RT Target

Skip this exercise if you have already configured the RT target during the LabVIEW Real-Time 1 course exercises.

Configure the RT target for this course.

GOAL

Summary – Quiz

1. Which of the following should you start identifying about an application before you start coding?
   a. Tasks
   b. Performance Requirements
   c. Task Timing
   d. Data Transfer Relationships
   e. Reliability Issues

Summary – Quiz Answers

1. Which of the following should you start identifying about an application before you start coding?
   a. Tasks
   b. Performance Requirements
   c. Task Timing
   d. Data Transfer Relationships
   e. Reliability Issues
Summary – Quiz

2. Which of the following are types of interfaces that can be found on a host computer?
   a. Diagnostic
   b. Analysis system
   c. Deterministic
   d. Configuration tool
   e. Operator interface

Summary – Quiz Answers

2. Which of the following are types of interfaces that can be found on a host computer?
   a. Diagnostic
   b. Analysis system
   c. Deterministic
   d. Configuration tool
   e. Operator interface
Lesson 2
Configuring Your Hardware

TOPICS
A. Hardware Setup and Installation
B. Configuring Network Settings
C. Installing Software on Target
D. Configuring Target I/O
E. Connecting to Target in LabVIEW

Configuring Your Hardware

- Setup RT Hardware and Host PC
- Configure Network Settings
- Install Software on Target
- Configure Target I/O
- Connect to Target in LabVIEW

ni.com/training
A. Hardware Setup and Installation

- Refer to the documentation for your target hardware for hardware setup and installation instructions.
- Target hardware arrives pre-installed with operating system.
- Classroom systems have been pre-installed and set up.

Host PC Setup and Installation

- Install LabVIEW and LabVIEW Real-Time Module on the host computer.
- Install necessary drivers for your hardware on the host computer. Refer to the documentation for your target hardware for driver installation instructions.

Configuring Your Hardware

- Configure Network Settings.
- Install Software on Target.
- Configure Target I/O.
B. Configuring Network Settings

Use one of the following connection methods:

- Connect target and host computer to the same local area network:
  - Use standard ethernet cables
  - Common scenario
- Connect target directly to host computer
  - Use an ethernet crossover cable
  - Many targets also work with standard ethernet cables

Detect the Remote Target

- Use the Measurement & Automation Explorer (MAX) to detect and configure your RT target
- Host computer must be on the same subnet to configure RT target

Configure Network Settings

- **IP Address** – the unique address of a device
  - a set of four one- to three-digit numbers in the range 0–255
  - dotted decimal notation.
  - example IP address: 224.102.13.24
- **Subnet Mask** – a code that helps the network device determine whether another device is on the same network.
  - 255.255.255.0 is the most common subnet mask
  - 255.255.0.0 is common for direct connections
- **Gateway** (Optional) – address of a gateway server (a connection between two networks)
- **DNS Address** (Optional) – address of a device that stores DNS host names and translates them into IP addresses
Host-Target Network Setup

- Host and Target need an IP Address
- To configure remote system
  - Host and Target need to be on the same subnet
  - Might need to set an exception in your firewall for MAX and LabVIEW

Assign IP Address to Target

Assign an IP address to your target using one of these methods:
- Automatically obtain an IP address
  - DHCP
  - Link Local
- Manually specify an IP address
  - Static

Automatically Obtain an IP Address – DHCP

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DHCP (Dynamic Host Configuration Protocol)</td>
<td>A network that has a DHCP server may automatically assign an IP to the target each time the target boots up. Assigned IP may differ from previously assigned IP. Common network protocol for administering IP addresses.</td>
</tr>
</tbody>
</table>

Use case
- Local Area Network (LAN) with DHCP server
  - Host computer automatically obtains IP address from DHCP server
  - RT target automatically obtains IP address from DHCP server
Automatically Obtain an IP Address – Link Local

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
</table>
| Link-local | Link-local addresses are network addresses intended for use in a local network only. RT target will connect to the network with a link-local address if DHCP fails.*
|          | Link-local address range: 169.254.x.x |

* This feature is available with some controllers and/or versions of Real-Time.

Automatically Obtain an IP Address – Link-Local Address

Use cases for link-local address
- Direct connection
  - Host computer is assigned or defaults to a link-local address
  - RT target connects with a link-local address
- LAN without DHCP server (or DHCP server is down)
  - Host computer connects to network with link-local address
  - RT target connects to network with link-local address

Automatically Obtain an IP Address

Select DHCP or Link Local configuration for RT Target
Automatically Obtain an IP Address – Caveats

Caveats of using DHCP or Link Local
- Automatically obtains an IP address on each reboot
  - New address may not be the same
  - To avoid this, use a static IP
- Typical DHCP servers allow you to reserve specific IP addresses for static IP addresses

Manually Specify an IP Address – Static IP

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static IP</td>
<td>• Manually specify an IP address</td>
</tr>
<tr>
<td></td>
<td>• IP address remains the same each time the target boots up</td>
</tr>
</tbody>
</table>

Use case
- **LAN**
  - Host computer can still use DHCP or link local address
  - Need to manually specify IP address of target
- **Direct connection**
  - Host computer can have link local address or static IP address
  - Manually specify IP address of target

Static IP – LAN

- **LAN use case**
  - Host computer connects to LAN with a DHCP-assigned IP address or link local address
  - Manually assign an IP address to RT target within the same subnet as the host computer

<table>
<thead>
<tr>
<th>Host computer example IP</th>
<th>RT target example IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example IP 1</td>
<td>Example IP 2</td>
</tr>
<tr>
<td>IP Address: 10.0.5.1</td>
<td>IP Address: 10.0.5.2</td>
</tr>
<tr>
<td>Subnet mask: 255.255.255.0</td>
<td>Subnet mask: 255.255.255.0</td>
</tr>
<tr>
<td>Example IP 3</td>
<td>Example IP 4</td>
</tr>
<tr>
<td>IP Address: 169.254.62.100</td>
<td>IP Address: 169.254.221.123</td>
</tr>
<tr>
<td>Subnet mask: 255.255.0.0</td>
<td>Subnet mask: 255.255.0.0</td>
</tr>
</tbody>
</table>
Static IP – Direct Connection with Link Local Address

- Direct connection with link local address use case
  - Host computer defaults to a link local address (no DHCP server)
  - Manually assign an IP address to RT target within the same link local subnet as the host computer

<table>
<thead>
<tr>
<th>Host computer example</th>
<th>RT target example</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address: 169.254.62.100</td>
<td>IP Address: 169.254.221.123</td>
</tr>
<tr>
<td>Subnet mask: 255.255.0.0</td>
<td>Subnet mask: 255.255.0.0</td>
</tr>
</tbody>
</table>

Static IP – Direct Connection without Link Local Address

- Requires both host computer and RT target to use static IP addresses
  - Set host computer to use static IP address
    - Control Panel > Network Connections
    - Local Area Connection > Internet Protocol > Properties

<table>
<thead>
<tr>
<th>Host computer example</th>
<th>RT target example</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Address: 192.168.0.1</td>
<td>IP Address: 192.168.0.2</td>
</tr>
<tr>
<td>Subnet mask: 255.255.255.0</td>
<td>Subnet mask: 255.255.255.0</td>
</tr>
</tbody>
</table>
Manually Specify an IP Address – Static IP

Select Static IP Address Configuration for RT Target

- Host computer and RT target must be in the same subnet

If a Gateway and DNS are not available, leave as 0.0.0.0

Configuring Your Hardware

C. Installing Software on Target
Exercise 2-1: Configure Hardware

Configure the cRIO target systems and the I/O hardware using MAX.

**GOAL**

**DISCUSSION**

- Is your RT target using a static IP address, link local IP address, or an IP address received from a DHCP server?

Confusing Your Hardware

**Setup RT Hardware and Host PC**

**Configure Network Settings**

**Install Software on Target**

**Configure Target I/O**

Connect to Target in LabVIEW
D. Configuring Target I/O

- Use MAX to configure the remote I/O hardware if necessary
  - RT PXI Systems require user to identify controller and chassis
  - Make sure hardware detected in MAX matches actual hardware
- Refer to product-specific documentation for assistance

PXI Configuration in MAX

Demonstrate the configuration of a PXI system in MAX.

Configuring Your Hardware

© National Instruments Corporation

Not For Distribution
E. Connecting to the Target in LabVIEW

For LabVIEW to connect to the RT target you must configure the target in a project.
1. Create a project
2. Create a real-time target within the project
3. Connect to the target
4. Add VIs to the target

LabVIEW Projects

- Projects manage files and targets
- Projects allow you to build executables or use source control
- Targets represent systems that can run VIs
  - Computer
  - Real-time systems
  - FPGA systems
  - Mobile devices

Adding Folders to a Project

- Virtual folder
  - Organizes project items and does not represent files on disk
- Auto-populating folder
  - Adds a directory on disk to the project
  - LabVIEW continuously monitors and updates the folder according to changes made in the project and on disk
Project Libraries

- Group a set of VIs, controls, and variables
- Libraries create a namespace
- Items can be public/private
- All shared variables must be in a library

Creating a Project

- Select File»New Project to open the Project Explorer window
- Project starts with the My Computer Target
- Add VIs that run on the host computer to the My Computer target

Adding a Real-Time Target

- Right-click the project in the Project Explorer window and selecting New»Targets and Devices
- Select the type of RT target
- Select a device
Connecting to a Target
Ways to connect to a target:
• Right-click the RT target and select Connect
• Run a VI on the target

Adding VIs to a Target
• Any VIs added to the target become targeted to that system
  - Right-click to create new VIs or add existing VIs
  - Drag VIs from Windows Explorer or other project folders
• Targeted VIs automatically deploy to the system when they run

Running VIs on a Target
• When you run an RT VI, the compiled code for the VI downloads to the RT target
• The compiled code on the RT target and the VI on the host PC exchange data through front panel communication
Closing a Front Panel Connection Without Closing VIs

- Selecting File » Exit while a VI runs results in the dialog box at right:
- Right-click the target and select Disconnect
  - RT VIs continue to run on the target, but no longer exchange front panel communication or debugging information with the host
  - Reconnecting to the target automatically opens and re-establishes a connection with any running VIs

Exercise 2-2: Targeting Real-Time Hardware

Exercise 2-2: Targeting Real-Time Hardware

Create a LabVIEW project and add an RT target for exercises in the course.

GOAL

DISCUSSION

If you unplug the Ethernet cable while the Blinking Light VI is running, will the Blinking Light VI continue to run on the RT target?
Summary – Quiz
1. Which of following are methods for connecting a target and host computer?
   a. Connect target and host computer to the same local area network
   b. Connect target directly to host computer using an Ethernet crossover cable
   c. Both a & b

Summary – Quiz Answer
1. Which of following are methods for connecting a target and host computer?
   a. Connect target and host computer to the same local area network
   b. Connect target directly to host computer using an Ethernet crossover cable
   c. Both a & b

Summary – Quiz
2. True or False? If your target is configured to obtain an IP address automatically using a DHCP or link local address, the target will have the same IP address every time it boots up.
2. True or False? If your target is configured to obtain an IP address automatically using a DHCP or link local address, the target will have the same IP address every time it boots up.

False

Automatically obtaining an IP address using a DHCP or link local address does not guarantee that the target will reboot with the same IP address each time.

3. For LabVIEW to connect to and run VIs on the RT target, you must create a ________.
   a. DHCP server
   b. Local Area Network
   c. LabVIEW Project

3. For LabVIEW to connect to and run VIs on the RT target, you must create a ________.
   a. DHCP server
   b. Local Area Network
   c. LabVIEW Project
Lesson 3
Real-Time Architecture: Design

TOPICS
A. Host and Target Application Architecture
B. Multithreading
C. Yielding Execution in Deterministic Loops
D. Improving Speed and Determinism
E. Sharing Data Locally on RT Target

A. Host and Target Application Architecture

Host Application
- Runs on the host computer
- Handles non-deterministic tasks
  - Communicates with the target application—user interface
  - Parameters and data retrieval
  - Data logging
  - Data analysis
  - Data broadcast to other systems

Target Application
- Non-deterministic Loop
- Data Storage

User Interface
- Communicates with host application

Data Storage
- Stores data

B. Multithreading

C. Yielding Execution in Deterministic Loops
Target Application

• Higher priority processes preempt lower priority processes
• Processes that must be deterministic are time-critical processes—set all other processes to a lower priority
• Use multithreading to set the priority of a process

B. Multithreading

Multithreading extends the following multitasking capabilities into applications:
• Subdivide specific operations within an application into individual threads
• Divide processing time among threads
• Enable assigning priorities
**Advantage of Multithreading**

Differentiates between deterministic and non-deterministic tasks

- **Deterministic task examples**
  - Control loop
  - Safety monitoring

- **Non-deterministic task examples**
  - Network Communication
  - Data logging

Real-time performance requires an operating system that gives scheduling priority to deterministic tasks

---

**Real-Time Multithreading Analogy**

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Highest Priority (one per core)</th>
<th>Normal Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Receptionist (Operating System)</td>
<td>![Ambulance]</td>
<td>![One Mechanic (Processor)]</td>
</tr>
</tbody>
</table>

---

**Scheduling Threads**

- **Round Robin Scheduling**
  - Threads of equal priority receive equal shares of processor time. It might take several turns for a thread to complete

- **Preemptive Scheduling**
  - Higher priority thread immediately pauses execution of all lower priority threads
Target Application Design

- Separate deterministic tasks from all other tasks
- Place deterministic tasks in the highest priority section of the target application

Class Exercise – Choose Priority Level

Potential Solution

Deterministic Tasks

Non-deterministic Tasks

UDP
TCP
Buffered DAQ
User Interface
signal analysis
Serial
CAN
DAQ
Host VI
Setting Priorities

Two methods to separate deterministic and non-deterministic tasks

VI Priorities

Timed Loop Priorities

Setting Priorities – VI Priority Method

Separate deterministic tasks from non-deterministic tasks into VIs set to different priorities

• Normal priority is the default priority
• Change the priority of a VI in the VI Properties dialog box
• SubVIs will inherit the priority of a higher-priority caller VI
Setting Priorities – VI Priority Method

Time-critical priority
• Preempts all other priorities
• VI does not relinquish processor resources until it yields to lower priority tasks or completes
• Set only one VI to time-critical priority

Setting Priorities – Timed Loop Method

Separate deterministic tasks from non-deterministic tasks using Timed Loops

Timed Loop Priority Level

Timed Loop
• Executes above high priority but below time critical priority
• Preempts all tasks except VIs set to time-critical priority
• Only use in VIs set to normal priority to avoid priority inversions

Timed Loop Execution Priority
Setting Priorities – Timed Loop Method

Separate deterministic tasks from non-deterministic tasks using Timed Loops set to different Timed Loop priorities.

Setting Priorities – Timed Loop Method

Timed Loop Priority
- Can specify the priority level of a Timed Loop relative to other Timed Loops

Deterministic Timed Loop
- Priority configured higher than all other Timed Loops
- Preempts all other Timed Loops
- Set only one Timed Loop per CPU to the highest priority

Timed Loop – Configuration
**Timed Loop – Setting Priorities**

- Use to write applications with multiple tasks that can preempt each other within the same VI
- Higher priority value—higher priority relative to other Timed Loops on the block diagram.
- Priority values can range between 1 and 65,535
- All Timed Loops execute below time-critical priority but above high priority in relation to ordinary VIs

**Timed Loop – Timing Source**

- Default source: 1 kHz clock of the operating system
  - Maximum execution: once every 1 ms
- Other timing sources
  - 1 MHz clock available on some real-time hardware
  - Events such as digital change detection
  - Hardware clock on a DAQ device

**Timed Loop – Period and Offset**

- Period—Length of time between loop executions
- Offset—Length of time the Timed Loop waits to execute
- Timing source determines the time unit of the period and the offset
Timed Loop – Naming Timed Loops

- Unique identifier for each Timed Loop
- Use the name provided by LabVIEW or use a custom name
- Refer to the loop programmatically using this name

Timed Loop – Assigning Processors

- By default, LabVIEW determines the processor or core where each thread executes
  - Automatic assignment makes efficient use of multiple processors
- A Timed Loop allows you to explicitly assign a processor or core
  - All code in the loop executes on the specified processor
  - Can ensure that time-critical and non-critical code do not share a processor
  - Can reduce the number of thread swaps on processors

Timed Loop – Modes

Determines how to handle late iterations:

- Maintain original phase
  - If enabled, Timed Loop will return to the original configured phase
  - If disabled, Timed Loop will adjust to run at the same period but not on the original configured phase
Timed Loop – Modes

Determines how to handle late iterations:
- Discard missed periods
  - If enabled, Timed Loop will discard any data generated during missed iterations
  - If disabled, Timed Loop will process any data generated during missed iterations

Timed Loop – Monitoring and Debugging

Functionality

Previous Iteration Timing » Finished Late terminal
- Possible uses
  - Display running count of how many loop iterations did not complete before the expected deadline
  - If a critical loop finishes late, send application into a recovery or safe shutdown mode

Previous Iteration Timing » Iteration Duration terminal
- Possible uses
  - Monitor a running average of loop iteration durations
  - If loop finishes late, you can use the iteration duration to calculate exactly how late
Timed Loop – Advanced Functionality

- Changing input node values dynamically
- Aborting specific Timed Loops
- Synchronizing multiple Timed Loop starts

Timed Loop – Changing Input Node Values Dynamically

- Possible scenario
  - Loop period should normally be 400ms
  - Need a higher loop period of 600ms while the application warms up during the first 100 loop iterations

Timed Loop – Aborting Execution

- Use the Stop Timed Structure VI to abort execution programmatically
- In this example, clicking Abort? Stops Loop 1 and Loop 2
Timed Loop – Synchronizing Timed Loop Starts

- Use Synchronize Timed Structure Starts VI with Offset input to ensure which loop starts first.

VI Priority and Timed Loop Advantages

**Timed Loop Method**
- More readable code
- Up to 65,535 different priorities
- Processor assignment
- Loop execution feedback
- Timing control using nodes
- Dynamic priorities and timing

**VI Priority Method**
- Less overhead
- Shorter warm-up time
- Not every task preempts basic system tasks
VI Priority and Timed Loop

Caution
• To prevent undesired behavior, use only one of the priority methods in your application
• If your application uses Timed Loops
  – Keep all VIs at normal priority
  – All Timed Loops run between high and time-critical priority
  – Can use While Loops to run normal priority tasks

C. Yielding Execution in Deterministic Loops

<table>
<thead>
<tr>
<th>Loops</th>
<th>Deterministic</th>
<th>Normal</th>
<th>Normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleeping</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>Executing</td>
<td>/ / / / / / /</td>
<td>/ / / /</td>
<td>/ / / /</td>
</tr>
<tr>
<td>Waiting</td>
<td>/ / / / / / /</td>
<td>/ / / /</td>
<td>/ / / /</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Loops</th>
<th>No sleep in deterministic loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Executing</td>
</tr>
<tr>
<td>Normal</td>
<td>Waiting</td>
</tr>
<tr>
<td>Normal</td>
<td>/</td>
</tr>
</tbody>
</table>

Starvation

<table>
<thead>
<tr>
<th>Loops</th>
<th>No sleep causes starvation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Executing</td>
</tr>
<tr>
<td>Above Normal</td>
<td>Waiting</td>
</tr>
<tr>
<td>Normal</td>
<td>/</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Loops</th>
<th>Sleep added – may still starve some loops</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>Executing</td>
</tr>
<tr>
<td>Above Normal</td>
<td>Waiting</td>
</tr>
<tr>
<td>Normal</td>
<td>/</td>
</tr>
</tbody>
</table>
Providing Sleep

The act of pausing the execution of a VI or thread

<table>
<thead>
<tr>
<th>Software-timed Sleep</th>
<th>Hardware-timed Sleep</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use the RTOS clocks to control the rate of a software loop.</td>
<td>Use external clocks to control the rate of a software loop.</td>
</tr>
</tbody>
</table>

Sleeping and Deterministic Loops

The following points are unique to the LabVIEW Real-Time Module:

- Avoid parallelism inside a time-critical priority VI or deterministic Timed Loop, because the code executes serially on the processor.
- A Wait VI or Wait Until Next Multiple VI in a time-critical priority VI or deterministic Timed Loop will execute serially with the rest of the code in the thread even if it is placed in parallel.
- All VIs set to time-critical VI priority execute in the time-critical priority thread.
- Each Timed Loop executes in its own thread.

Exercise 3-1: Priority Levels

Adjust the priority levels to understand the effects of priority levels on scheduling a VI.

GOAL
Exercise 3-1: Priority Levels

In this exercise, could you have rebooted the RT target by clicking the Restart button in MAX instead of pressing the reset button on the cRIO-9074?

D. Improving Speed and Determinism

• Choose appropriate hardware
• Use only one deterministic Timed Loop per CPU
• Avoid shared resources
• Disable non-essential options
• Use low-level functions to increase execution speed

Avoid Shared Resources

• A shared resource in the LabVIEW Real-Time Module is anything that can be used by only one process at a time
• LabVIEW RT shared resources include the following:
  - Global variables
  - The LabVIEW memory manager
  - Single-threaded DLLs
  - Networking code (TCP/IP, UDP, VI Server)*
  - Non-reentrant subVIs
  - Semaphore VIs
  - File I/O
* inherently non-deterministic

© National Instruments Corporation
Avoid Shared Resources Example

Before a process can begin using a shared resource, it must obtain a mutual exclusion object (mutex).

Process 1
- Running
- Waiting

Process 2
- Shared Resource
- Running
- Waiting

After Process 1 finishes, Process 2 can proceed.

Shared Resources – Priorities

Time-Critical Priority

Priority inversion:
- Normal priority VI blocks the higher priority VI with a mutex around the shared resource

Priority inheritance:
- Normal priority VI inherits higher priority to release mutex

Shared Resources – SubVIs

If unrelated parallel processes call the same VI, configure the VI for reentrant execution to allow multiple instances of the VI to be called simultaneously.

Reentrant VIs do not act like global variables.
Shared Resources – Memory Management

- LabVIEW typically manages memory automatically
  - You do not need to explicitly allocate or deallocate memory
  - Memory management is easy, but harder to control
- The LabVIEW memory manager is a shared resource
  - Control memory allocations to avoid shared resource conflicts
    with the memory manager
  - Statically allocate memory before time-critical process begins

Preallocate Arrays

- Avoid allocating arrays within a time-critical loop
- Preallocate array size to the largest expected array size
Setting VI Properties
To reduce memory requirements and increase performance of VIs, disable nonessential options
- Disable Allow debugging
- Disable Auto handle menus at launch

Use Low-Level Functions to Increase Execution Speed
- Express VIs
  - Intended to increase LabVIEW ease of use and improve productivity
  - May require additional performance overhead and perform optional tasks during execution
- Use low-level functions to increase execution speed with finer control

More Optimizations
- The LabVIEW Real-Time 2 course covers more LabVIEW Real-Time specific optimizations
- The LabVIEW Performance course covers more general LabVIEW performance and memory optimizations
E. Sharing Data Locally on RT Target

Target Program

Non-Deterministic Loop  Deterministic Loop

Inter-task Communication

Non-Deterministic Loop  Deterministic Loop

Use case  Description  Examples

Latest value  Only need to share the latest value  • Check if another loop has stopped  • Monitoring current I/O values

Every value  Need to share every value written using multi-element buffer. Cannot lose any values  • Send data to non-deterministic file logging loop  • Send data to non-critical data processing and analysis loop

Sharing Data Locally on RT Target

Target Program

Non-Deterministic Loop  Deterministic Loop

Inter-task Communication Methods

• Single-process Shared Variables with RT FIFO enabled
• RT FIFO functions

Single-Process Shared Variables with RT FIFO Enabled Method

Similar to a local or global variable on the block diagram with the following exceptions:

• Shared variables have error terminals
• Shared variables configured for read can return a timestamp
Creating Single-Process Shared Variables with RT FIFO Enabled

1. Right-click project or library and select New Variable.
2. In the Variable category of the Shared Variable Properties dialog box, set Variable Type to Single Process.
3. Select the Real-Time FIFO category, and configure options such as buffering and Real-Time FIFO.
Configure Single Process Shared Variable with RT FIFO Enabled

Use case

- You only need the latest value
  - Set FIFO Type to Single-element

- You need every value
  - Set FIFO Type to Multi-element
  - Set the number of elements in the FIFO

Single Process Shared Variable RT FIFOs – Initialization

The FIFO is created the first time a variable is read or written
- Set FIFO Type to Single-element
- Write a meaningful value to the variable before the main loop to create and initialize the FIFO
- Or allow for extra jitter in the first iteration of a loop that uses variables

Single Process Shared Variable RT FIFOs – Overflow

- Occurs when writing to an already full multi-element FIFO
- Overwrites the oldest unread data with the new data
- Error -2221 indicates an overflow
Single Process Shared Variable RT FIFOs – Underflow

- Occurs when reading from an empty multi-element FIFO
- Returns a default data value
- Error -2220 indicates an underflow

Single Process Shared Variable RT FIFOs – Multiple Readers and Writers

- Multiple readers and writers block other operations of the same type
- In deterministic loops, avoid writing/reading to single process shared variables that can be blocked by another writer/reader

Single Process Shared Variable RT FIFOs – Multiple Readers and Writers

- Multiple readers of a multi-element FIFO each remove elements, preventing either reader from getting all the data
- Multiple readers of an empty multi-element FIFO read last value that each read from the buffer or the default value for the data type of the variable if they have not read from the variable before
Exercise 3-2: Inter-Task Communication Using Shared Variables

**GOAL**

Use single-process shared variables with RT FIFO enabled for inter-task communication.

**DISCUSSION**

After the Single-Process Shared Variable Method VI begins running, can you adjust the RT FIFO size of the Result Data shared variable?

---

Static versus Dynamic Configuration of RT FIFO

- Single process shared variables are created statically through a configuration box
  - Name and data type
  - Number of elements in RT FIFO
Static versus Dynamic Configuration of RT FIFO

Use RT FIFO functions instead of shared variables if you need programmatic control of the RT FIFO
- Programmatically create and delete RT FIFO
- Programmatically set the number of elements in RT FIFO
- Set timeouts and timeout behavior
- Read the number of elements remaining in RT FIFO

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

Exercise 3-3: Course Project: Requirements Document

In this exercise, you analyze the requirements document for the project and examine a flowchart based on these requirements.
Summary – Quiz

1. Which of following are methods for improving speed and determinism in a deterministic loop?
   a. Avoid File I/O functions
   b. Cast all data to the proper data type
   c. Use the Build Array function to dynamically build arrays
   d. Disable non-essential VI options

Summary – Quiz Answers

1. Which of following are methods for improving speed and determinism in a deterministic loop?
   a. Avoid File I/O functions
   b. Cast all data to the proper data type
   c. Use the Build Array function to dynamically build arrays
   d. Disable non-essential VI options

2. True or False? You should only use one deterministic Timed Loop per CPU in a deterministic application.
2. True or False? You should only use one deterministic Timed Loop per CPU in a deterministic application.

True

3. True or False?
It is good practice to use Timed Loops within VIs set to background, above normal, high, or time-critical priority.

False. To avoid priority inversions, National Instruments recommends using timed structures only in VIs set to normal priority.
Lesson 4
Timing Applications and Acquiring Data

TOPICS
A. Timing Control Loops
B. Software Timing
C. Hardware Timing
D. Event Response

A. Timing Control Loops
• Provide sleep so lower priority tasks can execute
• Reduce application jitter

Timing Control Loops (continued)

Software Timing
• Timing tied to the ms or μs timer of the RTOS
• Loop timing achieved with timing functions or Timed Loop
• Timing VIs and functions mask software jitter within loops but introduce a minor amount of their own jitter

Hardware Timing
• Timing tied to an external clock (i.e. NI-DAQmx hardware)
• Clock is independent from OS timing
• Hardware jitter depends on accuracy of the clock
B. Software Timing

Three methods of implementing software timing:
• Insert standard Wait function or Wait Express VI in loop
• Insert standard Wait Until Next Multiple function in loop
• Replace looping mechanism with a Timed Loop

Software Timing – Wait
• Causes VI to sleep for a specified time
• Do not use in parallel inside deterministic loops
• Code execution time can vary, therefore total loop execution time varies

Software Timing – Wait Until Next Multiple
• Thread sleeps until the operating system ms timer equals a multiple of the Count (mSec) input
• First loop iteration is indeterminate
Software Timing – Wait Until Next Multiple (continued)

• Add a Wait before the loop to initialize the timer
• First iteration of loop time is determinate with this method

<table>
<thead>
<tr>
<th>OS ms timer</th>
<th>Code execution</th>
<th>Code execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>100</td>
<td>112</td>
</tr>
<tr>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>300</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Software Timing – Wait Until Next Multiple (continued)

\[ T_{sw} \text{ (worst case execution time)} < \Delta T \text{ (ms multiple + jitter)} \]

Software Timing – µs Timing

• Choose µs timing for the Wait Express VI and Wait Until Next Multiple Express VI to achieve greater loop rate resolution
• Enables loop rates of 1 MHz, 500 kHz, ~333 kHz, 250 kHz, 200 kHz, ~167 kHz, and so on
• Worst case code execution time must still be less than \( \Delta T \)
• Use same programming architecture as with ms timing
Software Timing – Timed Loop

• Automatically imposes sleep as needed to achieve the specified loop rate
• Configure the highest priority Timed Loop with a period large enough to perform the deterministic task and have idle time to allow lower priority loops to execute

Exercise 4-1: Software Timing

Modify code to use the RTOS ms and μs timer for timing.

GOAL
Exercise 4-1: Software Timing

If you replaced the Timed Loop with a While Loop, how would you modify the VI to offer the same ms and μs timing functionality?

C. Hardware Timing

Hardware timing methods:
• Use NI data acquisition hardware to time real-time applications with external clocks (i.e. NI-DAQmx)
• Refer to NI driver documentation for information about functions that you can use to sleep and wait for driver events

Hardware Timing – DAQmx

NI Example Finder
Hardware Input and Output»DAQmx»Control»General»PID Control-Single Channel.vi

Real-Time Module Control Architecture • Not Available on cFP
D. Event Response – Monitoring for Events

Use the point-by-point VIs to monitor for the following events:
- Triggering datalogging
- Triggering an alarm
- Performing a calculation

Event Response – Digital Change Detection

- Common event response application
- Must have a digital I/O device that supports change detection
Exercise 4-2: Course Project: Deterministic Loop

GOAL
Create the deterministic portion of the project.

DISCUSSION
How can you ensure that the Timed Loop in the Chamber Control subVI will not be preempted by other tasks that you will add to the application?

Summary – Quiz
1. Which of the following are benefits of using timing in a control loop?
   a. Provide sleep so lower priority tasks can execute
   b. Reduce application jitter
   c. Both a & b
Summary – Quiz Answer

1. Which of the following are benefits of using timing in a control loop?
   a. Provide sleep so lower priority tasks can execute
   b. Reduce application jitter
   c. Both a & b

Summary – Quiz

2. True or False? It is good programming practice to use wait functions in parallel with code inside a deterministic loop.

   False
Summary – Quiz

3. Which of the following provides finer resolution?
   a. \( \mu s \) timing
   b. ms timing

Summary – Quiz Answer

3. Which of the following provides finer resolution?
   a. \( \mu s \) timing
   b. ms timing

4. Which of the following methods use hardware timing?
   a. Timed Loop linked to a \( \mu s \) clock
   b. DAQmx VIs connected to an external clock
   c. Wait Express VI with \( \mu s \) resolution
   d. Timed Loop linked to a ms clock
Summary – Quiz Answers

4. Which of the following methods use hardware timing?
   a. Timed Loop linked to a μs clock
   b. DAQmx VIs to connect to external clock
   c. Wait Express VI with μs resolution
   d. Timed Loop linked to a ms clock
Lesson 5
Communication

TOPICS
A. Front Panel Communication
B. Network Communication
C. Network Communication Programming

A. Front Panel Communication

Host Computer
Displays front panel of RT Target VI

User Interface Communication

RT Target
Executes the block diagram logic of RT Target VI

Front Panel Communication

Use case
• Debug VIs
• Quickly monitor VIs running on RT target during development

Caveats
• Sections of code that contain front panel controls and indicators can introduce jitter
• Increased overhead
B. Network Communication

Host Computer
Executes Host VI which communicates with the RT Target VI

RT Target
Executes RT Target VI which communicates with the Host VI

C. Network Communication Programming

<table>
<thead>
<tr>
<th>Use case</th>
<th>Examples</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Latest value</td>
<td>Host computer displays the most recent I/O values of RT target</td>
<td>Network-published Shared Variables</td>
</tr>
<tr>
<td>Buffered values</td>
<td>Transfer data to host computer for file logging</td>
<td>Network Streams</td>
</tr>
<tr>
<td></td>
<td>Data must not be lost during transfer</td>
<td></td>
</tr>
<tr>
<td>Other protocols</td>
<td>Transfer data to LabVIEW and &quot;non-LabVIEW&quot; applications</td>
<td>Standard protocols such as TCP, UDP, serial, etc</td>
</tr>
</tbody>
</table>

Latest Values – Network-Published Shared Variables

Use cases
- Host computer displays the most recent I/O values of the RT target
- RT target monitors the status of the Stop button on the host computer
Network-published shared variables can communicate with host:
- From deterministic loop— with RT FIFOs
- From non-deterministic loop— without RT FIFOs

For latest value use case, verify that network buffering is disabled.

Initialize all shared variables at the start of your application:
- Write a meaningful value to the variable
- Wait for initialized value to propagate through the network and back
Location for Network-Published Shared Variables

Network-published shared variables can be located on the host PC or the target. Both locations have advantages:

<table>
<thead>
<tr>
<th>Target Advantages</th>
<th>Host PC Advantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability due to the</td>
<td>Less memory and CPU usage on RT target</td>
</tr>
<tr>
<td>stability of the RT target</td>
<td></td>
</tr>
<tr>
<td>LabVIEW DSC features available</td>
<td></td>
</tr>
<tr>
<td>Other computers reading the data do not use resources on target</td>
<td></td>
</tr>
</tbody>
</table>

Deploying Network-Published Shared Variables

- Network-published shared variables deploy and publish when a VI that references a variable runs
- All variables in a library deploy at the same time
- Network-published shared variables never automatically undeploy
- Deployed network-published shared variables remain on a system even after rebooting

Undeploying Shared Variables

Undeploy variables
- Programmatically, using the Library>Undeploy Library method of the Application VI Server class
- Manually, using the Project Explorer
- Manually, using the Distributed System Manager
Exercise 5-1: Course Project: Creating the Target VI

Create the target application that includes the normal priority loop and calls the time-critical Timed Loop.

**GOAL**

**DISCUSSION**

What are the benefits of having Initialization and Shutdown sections in your RT target VI?
Exercise 5-2: Course Project: Creating the Project Host VI

Create the host VI for the temperature chamber controller application.

GOAL

Exercise 5-2: Course Project: Creating the Project Host VI

Will the waveform chart on the Project Host VI display every single temperature point acquired on the RT target?

DISCUSSION

Buffered Values – Network Streams Functions

Use case
- No data can be lost during transfer between RT target and host computer
- Transfer data from RT target to host computer for logging data to file
- Transfer data from RT target to host computer for data processing and analysis that requires more memory than the RT target has available
Buffered Values – Network Streams Functions

Network Streams Characteristics

• Lossless
  – Each network stream writes to and reads from FIFOs
• Unidirectional
  – Each network stream transfers data in only one direction
• One-to-one
  – Each network stream consists of one writer and one reader

Network Streams – Specifying Endpoints

• Each network stream must have one writer endpoint and one reader endpoint
• Must assign each endpoint an endpoint name
• Must specify the data type of the elements in the network stream
  – Data type of writer and reader must match

Network Streams – Establishing Connection Between Endpoints

• Endpoint URL
  – /host_name/endpoint_name
• Establish a connection between two endpoints to create a network stream
  – Specify the URL of a remote endpoint with the URL input on either one of the Create Network Stream Endpoint functions
Network Streams – Programming Flow

Computer 1
(i.e. RT target)
Create Writer Endpoint
Write Element(s)
Flush Stream
Destroy Stream

Computer 2
(i.e. host computer)
Create Reader Endpoint
Read Element(s)
Destroy Stream

Network Streams – Configure Buffer Sizes
- **Writer buffer size** – size of writer buffer in number of elements
- **Reader buffer size** – size of reader buffer in number of elements

Network Streams – Writing Data
- **Write Single Element to Stream function**
  - Wire single element to `data` terminal
  - Set timeout and monitor timeout status
  - Default value is -1, which means there is no time limit on how long this function has to write the data to stream
Network Streams – Writing Data

• Write Multiple Elements to Stream function
  - Wire an array of elements to data in terminal

Network Streams – Reading Data

- Read Single Element from Stream
  • Read a single value from the data out terminal
  • Set timeout and monitor timeout status
  - Default timeout value is -1, which means the function waits for data indefinitely

- Read Multiple Elements from Stream function
  • Read an array of values from the data out terminal
  • # elements determines how many elements to read
  - Function returns an error if # elements > read buffer size of the Create Network Stream Reader Endpoint function
Network Stream Endpoint Property Node

- Access properties of a network stream
- Available Elements for Reading
- Available Elements for Writing
- Buffer Size
- Connected
- Data Type
- Name
- And more...

Network Streams – Flush Stream

Flush Stream function

- Transfers all data from writer endpoint to reader endpoint before data flow resumes
- Call this function from the writer endpoint only

Network Streams – Flush Stream

Wait condition input

- All Elements Read from Stream (default)
  - Data flow resumes when all data has been transferred to the reader AND the reader has finished reading the data
- All Elements Available for Reading
  - Data flow resumes when all data has been transferred to the reader
- Timeout determines how long this function waits for the wait condition to complete
Network Streams – Destroy Stream

Destroy Stream Endpoint function

- Must destroy both the reader and writer endpoints to completely destroy a stream and free up the memory allocated to that stream.
- Use the Flush Stream function first on the writer endpoint to ensure you do not lose any data.

Network Streams – Non-deterministic

- Network Streams functions are non-deterministic
- Do not use inside deterministic loops

Computer 1 (i.e. RT target)

Computer 2 (i.e. host computer)
Exercise 5-3: Network Streams

GOAL
Create an application that streams buffered data from the RT target to the host computer using the Network Streams functions

DISCUSSION
• Is it appropriate to put Network Streams functions in a loop that needs to execute deterministically?
• How would you modify the VIs in this application if you also needed to stream data from the host to the target?

Standard Protocols
Use case
• Communicate with hardware and software that does not support LabVIEW
  – May already have a completed application that uses a standard protocol
• Standard protocol may be easiest solution
  – Easily implement a broadcast model using UDP protocol
**Standard Protocols**

Example standard protocols
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Serial

**Standard Protocols – TCP Communication**

- TCP—Transmission Control Protocol
- Industry standard protocol for network communication
- Non-deterministic—Do not use inside a deterministic loop
- LabVIEW TCP VIs and functions:

**Standard Protocols – UDP Communication**

- UDP—User Datagram Protocol
- Not connection-based—Little overhead; cannot verify receipt of data
- Non-deterministic—Do not use inside a deterministic loop
- LabVIEW UDP VI and functions:
### Network Communication Comparison

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Common Use</th>
<th>Speed</th>
<th>Deterministic Read/Write</th>
<th>Deterministic Data Transfer</th>
<th>Advantages</th>
<th>Caveats</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network-Published Shared Variable</td>
<td>Latest value, host interface</td>
<td>Fast</td>
<td>Yes</td>
<td>No</td>
<td>Ease of programming</td>
<td>LabVIEW only</td>
</tr>
<tr>
<td>Network Streams</td>
<td>Data streaming</td>
<td>Fastest</td>
<td>No</td>
<td>No</td>
<td>Built-in functions</td>
<td>LabVIEW only</td>
</tr>
<tr>
<td>TCP</td>
<td>Data streaming</td>
<td>Fastest</td>
<td>No</td>
<td>No</td>
<td>High transfer rates, standard protocol</td>
<td>String data only</td>
</tr>
<tr>
<td>UDP</td>
<td>Broadcast latest values</td>
<td>Fastest</td>
<td>No</td>
<td>No</td>
<td>High transfer rates, standard protocol</td>
<td>String data, Lossy</td>
</tr>
</tbody>
</table>

### Summary – Matching Quiz

1. TCP functions  
   a. Transfer buffered values over the network

2. Network Streams functions  
   b. Easy method to transfer latest values over the network

3. Network-published shared variables  
   c. Commonly used standard protocol that is fast

### Summary – Matching Quiz Answers

1. TCP functions  
   a. Transfer buffered values over the network

2. Network Streams functions  
   b. Easy method to transfer latest values over the network

3. Network-published shared variables  
   c. Commonly used standard protocol that is fast
### Summary – Quiz

2. True or False? When using front panel communication, sections of code that contain front panel controls and indicators execute deterministically.

### Summary – Quiz Answer

2. False. When using front panel communication, sections of code that contain front panel controls and indicators are non-deterministic because LabVIEW must switch to the user interface thread, which is non-deterministic, to complete the task.

### Summary – Quiz

3. True or False? You should never use Network Streams, TCP, or UDP functions inside a deterministic loop.
Summary – Quiz Answer

3. True or False? You should never use Network Streams, TCP, or UDP functions inside a deterministic loop.

   True. Network Streams, TCP, and UDP functions are non-deterministic, so you should never put them in a time-critical loop. Put them in a lower priority loop instead.

Summary – Quiz

4. Which of the following should you use to communicate from the deterministic loop directly to the host VI?
   a. Single-process shared variable with RT FIFO disabled
   b. Single-process shared variable with RT FIFO enabled
   c. Network-published shared variable with RT FIFO disabled
   d. Network-published shared variable with RT FIFO enabled

Summary – Quiz Answer

4. Which of the following should you use to communicate from the deterministic loop directly to the host VI?
   a. Single-process shared variable with RT FIFO disabled
   b. Single-process shared variable with RT FIFO enabled
   c. Network-published shared variable with RT FIFO disabled
   d. Network-published shared variable with RT FIFO enabled
Lesson 6
Verifying Your Application

TOPICS
A. Verifying Correct Application Behavior
B. Verifying Performance and Memory Usage

Lesson

A. Verifying Correct Application Behavior
• Debug real-time VIs the same as normal VIs
• Use any debugging technique, except the Call Chain Ring and debugging reentrant VIs, while targeting the RT system
• Enable Allow debugging in VI Properties dialog box

Standard Debugging Techniques
Finding Errors
Click Broken Run
A window showing the error appears

Error Handling
Debug and manage errors in VIs

Execution Highlighting
Click Execution Highlighting
Data flow is animated using bubbles
Values are displayed on wires
Standard Debugging Techniques (continued)

Probe
Right-click a wire and select **Probe**, which shows data as it flows through the wire segment.
Select **Retain Wire Values** before running to probe wires that have already executed.

Breakpoint
Right-click a wire and select **Set Breakpoint**, which pauses execution at the breakpoint.

Conditional Probe
Combination of a breakpoint and a probe; right-click a wire and select **Custom Probe**.

Standard Debugging Techniques (continued)

Use Step Into, Step Over, and Step Out buttons for single stepping.
- **Click Step Into** to enable single stepping.
- After single stepping has begun, the button steps into nodes.

- **Click Step Over** to enable single stepping or to step over nodes.
- **Click Step Out** to step out of nodes.

B. Verifying Performance and Memory Usage

- Profile Performance and Memory window
- Distributed System Manager
- RT Utility VIs
Profile Performance and Memory Tool

- Select Tools » Profile » Performance and Memory
- Analyze the execution time and memory usage of an application
- Identify specific subVIs that need to be optimized
- Perform time profiling separately from memory profiling

Distributed System Manager

Displays RT target resources in addition to shared variable and I/O variable data

- View system memory and CPU usage
  - CPU usages of different priority levels
- View and control VI states
  - Running, Idle, Stopped
- Configure Alerts
  - Includes logging at configured ranges

© National Instruments Corporation
Distributed System Manager

Distributed System Manager – Memory and CPU Usage
• Determine if memory leaks occur over time
• Shows how much time a deterministic loop is leaving for other loops to run
  – Time-critical priority
  – Timed structures

Distributed System Manager – VI States
• Use VI States tab to start, stop, and monitor VIs on the RT target
• Uses VI Server to communicate to RT target
  – Must configure RT target to allow VI Server access on a TCP/IP port
  – Configure DSM to connect to corresponding port on RT target
Distributed System Manager – VI States

Configure VI Server access to VI States
1. Select Properties for the RT target in the Project Explorer
2. Configure VI Server to use TCP/IP
3. Add the host IP address to the Machine Access list
4. Right-click RT target in the Project Explorer and select Deploy

Distributed System Manager – VI States

Update Interval
- How often the Distributed System manager checks for information
- If target is too busy to provide this information, then no information is reported

Distributed System Manager – Alerts

- Saves system data when any property is out of its configured range
- Lets the user run the utility for long periods of time without having to constantly monitor it
- Can save data to disk
Exercise 6-1: Distributed System Manager

GOAL

Use the Distributed System Manager to verify the CPU and memory usage, monitor VI states, and log alerts on an RT target.

DISCUSSION

How can you monitor CPU and memory usage and log alerts in an RT Target VI?
Real-Time Execution Trace Toolkit
• Analyze and benchmark thread and VI execution
• Optimize performance by identifying memory allocation, sleep spans, mutexes and thread swaps
• Send trace sessions to host computer or save to local file
• Print trace sessions for documentation and code reviews
• Covered in the LabVIEW Real-Time 2 course

Summary – Quiz
1. True or False? You must enable the Allow debugging option in VI Properties in order to use the Execution Highlighting and Single-stepping tools on a deterministic VI.

Summary – Quiz Answer
1. True
Summary – Quiz
Match the tool with what it can report:
1. Distributed System Manager
2. Performance and Memory Profiler
3. Probes
4. Execution Highlighting
A. CPU Usage
B. Memory Usage
C. Timing
D. Application Behavior

Summary – Quiz Answers
Match the tool with what it can report:
1. Distributed System Manager (A, B)
2. Performance and Memory Profiler (B, C)
3. Probes (D)
4. Execution Highlighting (D)
Lesson 7
Deploying Your Application

TOPICS
A. Introduction to Deployment
B. Creating a Build Specification
C. Communicating with Deployed Applications
D. System Replication

A. Introduction to Deployment
- Creating a stand-alone application (executable) of a LabVIEW Real-Time Module application using the LabVIEW Application Builder
- Benefits of deploying your RT Application:
  - Embed executable in non-volatile memory on the target
  - Launch executable automatically when target boots

Preparing Your Application for Deployment
Review the code for unsupported functions
- Functions that modify front panel objects
- Functions that use technologies specific to other operating systems
Avoid Modifying Front Panel Objects

- Front panel Property Nodes and control references
- Dialog functions
- VI Server front panel functions

Do Not Use OS-Specific Technologies

Examples:
- ActiveX VIs
- .NET VIs
- Windows Registry Access VIs
- TestStand VIs (ActiveX-based)
- Report Generation Toolkit VIs
- Cursor VIs
- Call Library Nodes that access an operating system API other than ETS or VxWorks
- Graphics and Sound VIs
- Database Connectivity Toolset
- XML DOM Parser and G-Web Server for CGI Support

Deploying a Stand-Alone RT Application

1. Configure a real-time application build specification
2. Build the real-time application
3. Run the real-time application as a startup application
B. Creating a Build Specification

Configuring Settings – Information
Set names and destinations

Configuring Settings – Source Files
Set startup VI(s) and include dynamically referenced files
Configuring Settings – Preview
Preview files and destinations

Build the Stand-Alone RT Application

Automatic Start on Target
Run as startup:
• Sets application as the startup application
• Deploys the application to the target
• Prompts you to reboot the RT target
Unsetting Startup Executables

Unset the application as the startup application
• Right-click the build specification and select Unset as startup
• Right-click the RT target and select Deploy

Exercise 7-1: Deploy Project

Prepare the project for deployment and create an executable.

GOAL

C. Communicating with Deployed Applications

• Create a VI on the host computer to communicate with the stand-alone RT application as shown in Lesson 5, Communication
• Use the Debug Application tool to connect to the stand-alone RT application for debugging
**Debugging Executables**

1. Enable the debugging option in the **Advanced** category when configuring the real-time build specification.

2. Select **Operate** → **Debug Application or Shared Library** on host computer.

3. Enter the IP Address of the target and click **Refresh**.

4. Select the application to debug and click **Connect**.

5. Debug the application normally.

**D. System Replication**

Replicate one Real-Time target into a copy of itself

- **Use cases**
  - Store a backup image of your RT target
  - Deploy an image to multiple identical RT targets
  - Replicate one RT target onto multiple identical RT targets

- **Caveat**
  - Images can only be deployed to identical models
System Replication – USB Method

- Use RT Desktop PC Utility USB Drive to create an RT disk image from an RT target and apply it to an identical model
  - Make sure USB drive has enough space available to store RT disk image
  - This method only works with PXI and Desktop PC RT targets
    - Target must have USB port
    - Must connect a monitor and keyboard to target

System Replication – USB Method

Create a RT Desktop PC Utility USB Drive

- In NI MAX, select Tools»Real-Time Disk Utilities»Create Desktop PC Utility USB Drive

Create RT target disk image from RT target

- Connect the USB drive to RT target
  - Return RT target and navigate menu to save a backup disk image on the USB drive

Apply RT target disk image to identical model

- Connect the USB drive to RT target
  - Reboot RT target and apply disk image from the USB drive onto the RT system

System Replication – Programmatic method

- Use RT Utilities VIs on host computer
  - Create custom utilities for performing replication and backup operations
System Replication – Programmatic method

• RT Create Disk VI
  – Use on Windows host computer
  – Copies the contents of the primary hard drive on the RT target
to the local zip file you specify

System Replication – Programmatic method

• RT Apply Target Disk Image VI
  – Replaces the entire contents of the primary hard drive on the
RT target with the files in the target disk image you specify
  – RT target must have the same model code as the target used
to create the disk image
  • Use RT Get Target Information VI to find the model code of
an RT target

Summary – Quiz

1. True or False? Front panel property nodes are supported
   in a stand-alone real-time application.
Summary – Quiz Answer

1. True or False? Front panel property nodes are supported in a stand-alone real-time application.
   False

Summary – Quiz

2. True or False? If you are replicating a disk image onto an RT target, the RT target does not need to be the same model as the target used to create the disk image.
   False