Worldwide Technical Support and Product Information
ni.com

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

National Instruments Corporate Headquarters
11500 North Mopac Expressway  Austin, Texas 78759-3504  USA  Tel: 512 683 0100
To comment on National Instruments documentation, refer to the National Instruments Web site at ni.com/info and enter the Info Code feedback.
Contents

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

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

Lesson 3
Accessing Your I/O
Exercise 3-1 Targeting Real-Time Hardware ............................................................................................................................................3-1

Lesson 4
Real-Time Architecture: Design
Exercise 4-1 Priority Levels.............................................................................................................................................................................4-1
Exercise 4-2 Timing While Loops and Timed Loops.......................................................................................................................................4-4
Lesson 5
Inter-Process Communication
- Exercise 5-1: Inter-Process Communication Using Shared Variables
- Exercise 5-2: Course Project: Requirements Document
- Exercise 5-3: Course Project: Deterministic Loop

Lesson 6
Communicating Between the RT Target and Host
- Exercise 6-1: Course Project: Creating the Target VI
- Exercise 6-2: Course Project: Creating the Project Host VI
- Exercise 6-3: Network Streams

Lesson 7
Verifying Your Application
- Exercise 7-1: Distributed System Manager

Lesson 8
Introduction to Deployment
- Exercise 8-1: Deploy a Startup RT Application

Appendix A
Course Slides
Inter-Process Communication

Exercise 5-1  Inter-Process Communication Using Shared Variables

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

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

You will use single-process shared variables to share data between a deterministic Timed Loop and non-deterministic normal priority While Loop 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-Process Communication project.
   - In LabVIEW, select File»Open Project.
   - Select <Exercises>\LabVIEW Real-Time 1\Inter-Process Communication\Inter-Process 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-process 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-Process Communication directory.

2. Create the Waveform Type single-process shared variable for inter-process communication.

   □ Right-click the Communication Shared Variables library and select New»Variable.

   □ 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-process 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
4. Create the Result Data single-process shared variable for inter-process 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
   - Set the following options in the RT FIFO category of the Shared Variable Properties dialog box.
     - Enable RT FIFO: Checked
     - FIFO Type: Multi-element
     - Number of elements: 10
   - Click OK.

5. Create the Stop single-process shared variable for inter-process communication.
   - Right-click the library and select New»Variable.
   - 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 processes from non-deterministic processes and place the deterministic processes in a Timed Loop with the highest priority in an RT target VI to ensure the deterministic processes receive enough processor resources. This Timed Loop will be the deterministic Timed Loop in this application and will perform the deterministic process 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.
   - Verify the Input Node of the Timed Loop shows 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. This sets the data type of the shift register to an error cluster.
6. Create the deterministic loop as shown in Figure 5-1 using the following items:

**Figure 5-1. Deterministic Loop of Single-Process Shared Variable Method VI**

- **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 5-2 and Figure 5-3.

**Figure 5-2. Overflow Case in Deterministic Loop of Single-Process Shared Variable Method VI**

- 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 processes from deterministic processes and place the non-deterministic processes in a While Loop in a normal priority RT target VI. This While Loop executes at a lower priority than the deterministic Timed Loop to ensure the non-deterministic processes do not preempt the deterministic processes. The While Loop will execute the non-deterministic process of plotting waveform data in this application.

1. Add and configure the non-deterministic While Loop.
   - Place a While Loop on the block diagram above the deterministic Timed Loop.
2. Place the four shared variables inside the non-deterministic While 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 and drag them to the block diagram.
   - 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.

3. Create a shift register to pass error data to following loop iterations.
   - Right-click the right border of the While 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 While Loop.

4. Create the non-deterministic While Loop as shown in Figure 5-4 using the following items:

   Figure 5-4. Non-deterministic Loop of Single-Process Shared Variable Method VI

   - Wait Until Next ms Multiple function—This function determines the period of the outer While Loop
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 another While Loop inside the first While Loop. The inner 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 inner 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.

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 5-5 and Figure 5-6, respectively.

Figure 5-5. Underflow Case in Non-deterministic Loop of Single-Process Shared Variable Method VI
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 Timed Loop and non-deterministic While Loop 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 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 5-7.

**Figure 5-7. Initialization of Single-Process Shared Variable Method VI**
Shutdown
After the deterministic Timed Loop and non-deterministic While Loop stop executing, merge and handle the errors.

1. Merge and handle the errors as shown in Figure 5-8 using the following items:

   **Figure 5-8.** Shutdown of Single-Process Shared Variable Method VI

   - 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 window, 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 \((100 \text{ ms} / 10 \text{ ms}) \times 1.1\) 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.
Lesson 5  Inter-Process Communication

Exercise 5-2  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 5-9.

Figure 5-9. 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 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 5-10 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 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
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 VI also maintains a Finished Late Count that keeps track of the number of times that the deterministic loop finished late.

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 loop is defined in the following flowchart:

Figure 5-11. 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 5-1 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, fan status, and finished late count 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
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, Finished Late Count

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 5-2
Exercise 5-3  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. Refer to the flowchart from Exercise 5-2 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 Main VI

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

- Read PID gains from the host computer
- Read setpoint from the host computer
- Read temperature of the temperature chamber
- Get timestamp of temperature reading
- Determine proportional, integral, derivative (PID) output
- Read disturbance status from the host computer
- Output fan intensity
- Output lamp intensity
- Write data to RT FIFO
- Read Stop RT status from host computer
- Update the finished late count if the previous iteration of the deterministic loop finished late.
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 window, expand the RT target and double click the RT Main 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 of the Timed Loop is configured by the Period numeric control. This control has been mapped to a terminal on the connector pane, so you can set the period input 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.
   - Left Data Node of the Timed Loop—Resize this node to show two items. Set the top item to `Period` and the bottom item to `Finished Late? [i-1]`.
   - 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 Figure 5-12, using the following items:

**Figure 5-12. Chamber Control VI Block Diagram**

- **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. Use this type of shared variable to deterministically write the value of the finished late count. You will learn more about this type of shared variable in Lesson 6.
  - **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.

– Finished Late Count variable—Drag a copy of this shared variable from cRIO-9074»Project Variables.lvlib»Target to Host»Finished Late Count 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 in the Project Explorer to the block diagram.

– Right-click this I/O variable and select Timestamp»Show.

PID VI—Drag and drop this VI from the Functions palette. 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»Project Variables.lvlib»Host to Target»Fan in the Project Explorer to the block diagram.

– Right-click the shared variable and select Access Mode»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.

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 space on the block diagram.

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 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»Interprocess»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»Interprocess»Target Data - RT in the Project Explorer to the block diagram.

□ Unbundle By Name function

□ Or function

□ Case structure—Use a Case structure to increment the finished late count each time the Timed Loop finishes late. The finished late count will be sent to the host VI.

- In the True case, place an Increment function and the Finished Late Count network-published shared variable.

- In the False case, wire the input tunnel to the output tunnel.

□ Flat Sequence structure

- Place a True constant and the Stop RT network-published shared variable in this structure.

□ Complete the wiring as shown in Figures 5-12.

8. Compare the flowchart you reviewed in Exercise 5-2 with your completed block diagram. Do they match? Are you accomplishing the tasks as designed?

9. Save and close the Chamber Control VI.
Configure Chamber Control subVI in RT Main VI
In this section, you will call and configure the Chamber Control subVI from the RT Main VI, which is the top-level VI for this application.

1. Open the RT Main 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 Main 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.

4. Save the RT Main VI.

Test
1. Run the RT Main 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.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.
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.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.

6. Stop the Chamber Control subVI and RT Main 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 Main 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 5-3**