Getting Started with the LabVIEW™ Real-Time Module

This document provides steps to build a simple deterministic application and references to programming examples and documentation for more information as you build the application. Refer to the *LabVIEW Real-Time Module Release Notes* for installation and configuration instructions. Refer to the *LabVIEW Real-Time Module User Manual* for information about programming deterministic applications using LabVIEW.

Contents

Launching LabVIEW and Exploring Documentation

The LabVIEW Real-Time Module includes a comprehensive documentation set designed to help you create deterministic applications. The *LabVIEW Real-Time Module Bookshelf*, available by selecting **Help»Search the Real-Time Module Bookshelf** from LabVIEW, is a PDF that contains links to and descriptions of the available Real-Time Module documentation.

1. Launch LabVIEW to open the **LabVIEW** dialog box shown in Figure 1.
2. Open and explore the *LabVIEW Real-Time Module Bookshelf*.

April 2004
323525B-01

NATIONAL INSTRUMENTS™

Figure 1. The **LabVIEW** Dialog Box

The *LabVIEW Help* contains VI reference information for all Real-Time VIs.

3. Select **Help»VI, Function, & How-To Help** to open and explore the *LabVIEW Help*.

The RT Module allows you to select the host computer or an RT target as an execution target to run VIs built in LabVIEW. The **LabVIEW** dialog box indicates the current execution target. The default execution target is the development computer, or host computer. Refer to Chapter 2, *Connecting to RT Targets*, of the *LabVIEW Real-Time Module User Manual* for information about selecting execution targets.

When you select an RT target—an RT Series plug-in device or networked RT Series device—as the execution target, the LabVIEW development environment continues to run on the host computer but VIs run on the RT target. Refer to Chapter 1, *Introduction to the LabVIEW Real-Time Module*, of the *LabVIEW Real-Time Module User Manual* for information about RT targets.

Creating and Running a Deterministic Application

This section guides you through creating a deterministic PID control application. You need to configure an NI RT Series PXI controller, NI PCI-7041 plug-in device, or an RT Series FieldPoint module as an RT target to complete this exercise. Refer to the *Measurement & Automation Explorer Help* for information about configuring a remote system as an RT target.

The application receives an analog input, processes the data using PID control VIs, and outputs the resulting control data. Refer to the *LabVIEW PID Control Toolset User Manual* for information about the PID control algorithm and the PID Control Toolset VIs. Refer to the *LabVIEW Real-Time Module User Manual* for information about Real-Time Module features.

Creating the Control Loop

Complete the following steps to create a control loop for the control VI.

1. Start LabVIEW and select the RT target from the **Execution Target** pull-down list. Refer to the *LabVIEW Real-Time Module User Manual* for more information about connecting to RT targets.
2. Open a blank VI and display the block diagram.
3. Place a While Loop on the block diagram. Refer to the *LabVIEW User Manual* for information about using While Loops.
4. Right-click the conditional terminal and select **Create Control** from the shortcut menu to create and wire a **Stop** button control to the conditional terminal.

The While Loop runs all code within its boundaries until the conditional terminal receives a TRUE value. Use the conditional loop as the control loop in the next section. When you press the **Stop** button on the front panel, the button passes a TRUE value to the conditional terminal.

Adding Analog Input/Output and Data Processing to the Control Loop

Complete the following steps to add analog input/output and a PID control VI to the control loop.

1. Place the DAQmx Read VI in the control loop on the block diagram.

   (FieldPoint) Place the FP Read VI.

2. Right-click the **task/channel in** input of the DAQmx Read VI and select **Create»Control** from the shortcut menu to create a control for the **task/channel in** input on the front panel.

   (FieldPoint) Create a control for the **FieldPoint IO Point In** input of the FP Read VI.

3. Select Browse from the pull-down list of the **task/channel in** control on the front panel to open the Select name(s) dialog box. Click the **Create New** button to open the DAQ Assistant and create a new NI-DAQmx task. Refer to the *NI-DAQmx Help* for information about using the DAQ Assistant to create an NI-DAQmx task. Select the NI-DAQmx task that you create from the pull-down list of the **task/channel in** control on the front panel.

   (FieldPoint) Select the device in the **FieldPoint IO Point In** control. Refer to the *Measurement & Automation Explorer Help for FieldPoint* for information about configuring a device.

4. Place the PID VI in the control loop on the block diagram. You also can create custom control VIs using LabVIEW.

5. Wire the **data** output of the DAQmx Read VI to the **process variable** input of the PID VI.

   (FieldPoint) Wire the **values** output of the FP Read VI to the **process variable** input of the PID VI.

6. Create controls for the **PID gains** and **setpoint** inputs of the PID VI.

7. Place the DAQmx Write VI in the control loop on the block diagram.

   (FieldPoint) Place the FP Write VI.

8. Wire the **output** output of the PID VI to the **data** input of the DAQmx Write VI.

   (FieldPoint) Wire the **output** output of the PID VI to the **values** input of the FP Write VI.

9. Create a control for the **task/channel in** input of the DAQmx Write VI and select an NI-DAQmx task. Refer to step 3 for information about creating an NI-DAQmx task using the DAQ Assistant.

   (FieldPoint) Create a control for the **FieldPoint IO Point In** input of the FP Write VI.

The block diagram for an NI-DAQmx system should look similar to Figure 2.

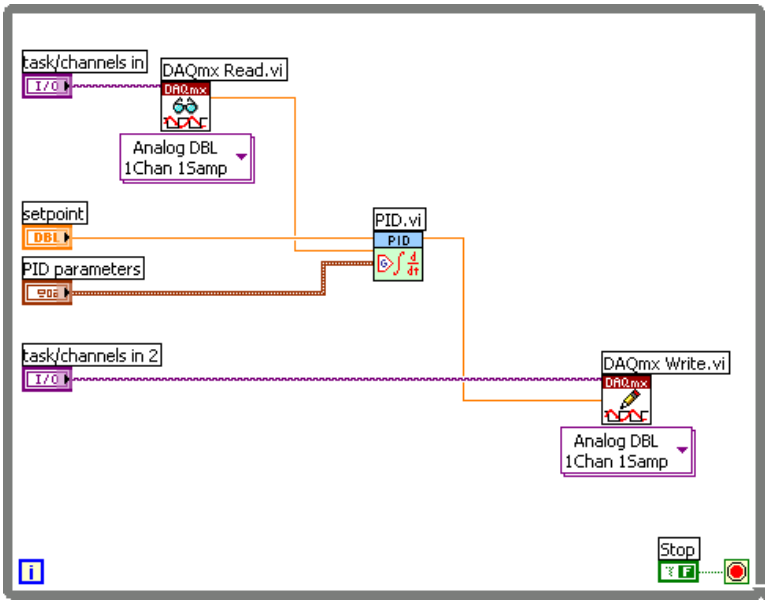

Figure 2. NI-DAQmx Control Loop

The block diagram for a FieldPoint I/O system should look similar to Figure 3.

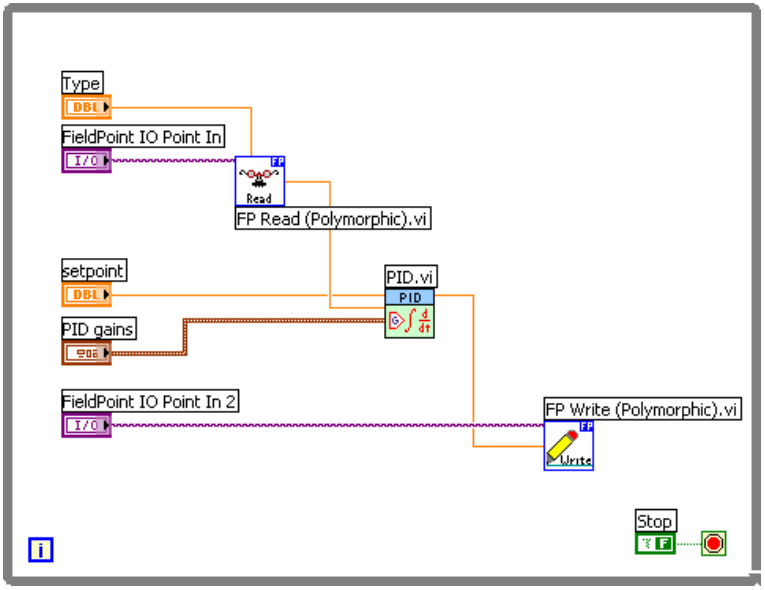

Figure 3. FieldPoint Control Loop

The VI reads analog data and sends the data to the PID VI. The PID VI processes the data using a PID control algorithm. The VI returns the results to the device. Refer to the *LabVIEW PID Control Toolset User Manual* for information about using PID VIs.

(ETS) Refer to the `examples\Real-Time\ETS\RT Control.llb` for examples of control VIs.

Setting Priority of the VI to Time Critical

To ensure determinism, you must assign time-critical priority to the control loop VI. A VI set to time-critical priority receives enough processor resources to complete time-critical tasks.

1. Right-click the connector pane icon at the top right corner of the VI and select **VI Properties** from the shortcut menu to open the **VI Properties** dialog box.
2. Select **Execution** from the **Category** pull-down menu.
3. Select **time critical priority (highest)** from the **Priority** pull-down menu.
4. Click the **OK** button to set the priority.

Refer to Chapter 3, *Building Deterministic Applications*, of the *LabVIEW Real-Time Module User Manual* for information about dividing and prioritizing application tasks. Refer to the Priority Trouble VI in the `examples\Real-Time\RT Tutorial.llb` for an example of priority settings.

Adding Timing to the Control Loop

You must add a timing mechanism that pauses, or sleeps, the control VI to ensure that other VIs in the application have enough processor resources to complete. A time-critical VI that does not have any sleep time monopolizes the embedded CPU resources. Refer to Chapter 3, *Building Deterministic Applications*, of the *LabVIEW Real-Time Module User Manual* for information about programming deterministic applications.

1. Place the Wait Until Next ms Multiple VI in the control loop on the block diagram.
2. Right-click the **millisecond multiple** input of the Wait Until Next ms Multiple VI and select **Create»Constant** from the shortcut menu.
3. Enter `10` into the **millisecond multiple** constant. The timing mechanism assures you that the control loop code runs and then sleeps until the time equals 10 ms from the start of the iteration. Other VIs in the application can run during the sleep time.
4. Select **File»Save As** and save the VI as `First PID Control.vi`.

The block diagram should look similar to Figure 4.

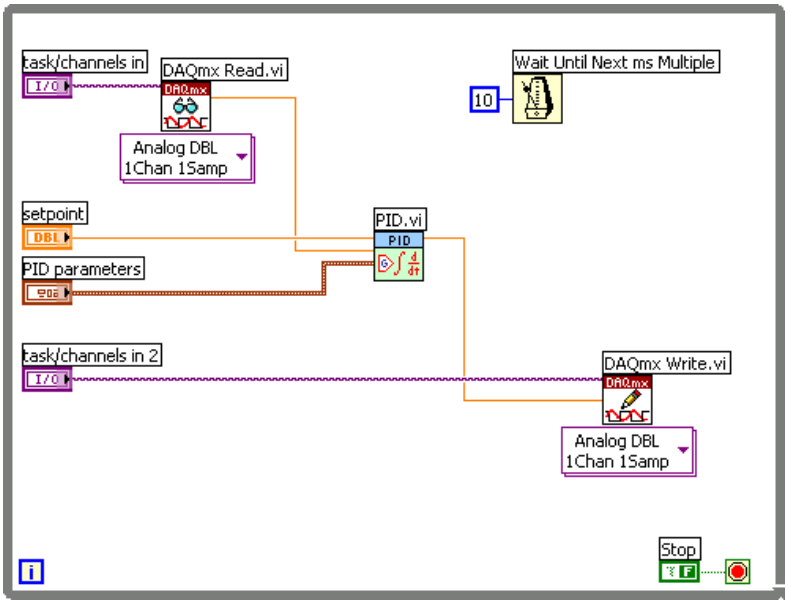

Figure 4. NI-DAQmx Control Loop with Timing

Refer to Chapter 4, *Timing Applications and Acquiring Data*, of the *LabVIEW Real-Time Module User Manual* for information about the different methods for timing control loops. Refer to the `examples\Real-Time\RT Timing.llb` for examples of different timing methods.

Sending Data Out of the Time-Critical VI

The First PID Control VI can run on the RT target at time-critical priority, but you might need to monitor the status of the VI as it runs on the target. Also, you might need to update the values of control loop parameters. Use the Real-Time FIFO VIs to send data between the First PID Control VI and a normal priority communication VI that also runs on the RT target. The communication VI on the RT target can receive data from the First PID Control VI using Real-Time FIFOs and then use network communication methods to update the host computer VI. Refer to the *LabVIEW Real-Time Module User Manual* for information about using the Real-Time FIFO VIs.

Complete the following steps to control the **Stop** button from the host computer.

1. Open the First PID Control VI in LabVIEW.
2. Remove the **Stop** button control wired to the conditional terminal of the control loop on the block diagram.
3. Place the RTFIFOCreate VI to the left of the control loop on the block diagram.
4. Right-click the **name** input of the RTFIFOCreate VI and select **Create»Constant** from the shortcut menu. Enter `Stop Control` into the **name** constant.
5. Wire a numeric constant to the **type** input of the RTFIFOCreate VI. The **type** input represents the type of data that you want the RT FIFO to contain.
6. Place the RTFIFORead VI in the control loop on the block diagram.
7. Right-click the While Loop and select **Add Shift Register** from the shortcut menu. Use While Loop shift registers to pass data from one iteration to the next. Refer to the *LabVIEW User Manual* for information about using shift registers.
8. Wire the **rt fifo** output of the RTFIFOCreate VI to the left terminal of the shift register and then wire the terminal to the **rt fifo** input of RTFIFORead VI.
9. Add another shift register and repeat step 8 to wire the **error out** output of the RTFIFOCreate VI to the **error in** input of the RTFIFORead VI.
10. Wire the **rt fifo** and **error out** outputs of the RTFIFORead VI to the right terminal of the shift registers.
11. Place the Equal? function in the control loop to the left of the conditional terminal. Wire the **element out** output of the RTFIFORead VI to the **x** input of the Equal? function. Right-click the **y** input of the Equal? function and select **Create»Constant** from the shortcut menu. Enter `1` into the constant.
12. Wire the **x = y?** output of the Equal? function to the conditional terminal of the control loop.
13. Select **File»Save As** and save the VI as `Second PID Control.vi`.

The block diagram should look similar to Figure 5.

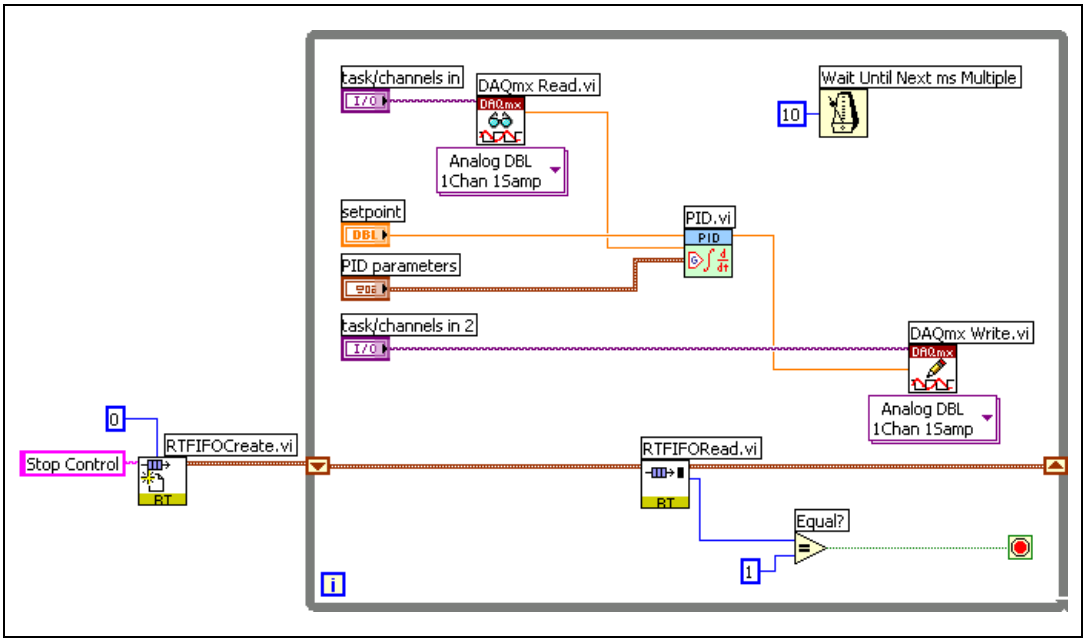

Figure 5. DAQ Control Loop with RT FIFOs

Downloading and Running the Application

You have created the time-critical control loop that runs on the RT target. You can use the `RT Engine - Normal Priority Loop (TCP Comm).vi` and the `Host Machine (TCP Comm).vi` examples located in the `examples\Real-Time\RT Tutorial.llb` for the normal priority and host computer VI.

Complete the following steps to download the time-critical VI and the network communication VI and test the functionality.

1. Without closing the Second PID Control VI, open the RT Engine - Normal Priority Loop (TCP Comm) VI and view the block diagram.
2. In the upper-right corner of the Second PID Control VI front panel there is a connector pane image of the VI. Click and drag the image to an area outside the While Loop on the block diagram of the RT Engine - Normal Priority Loop (TCP Comm) VI to use the Second PID Control VI as a subVI in the normal priority communication VI.

3. From the RT Engine - Normal Priority Loop (TCP Comm) VI block diagram, select **Browse»Show VI Hierarchy** to open the **Hierarchy** window. Notice that the thumbtack for RT Engine - Normal Priority Loop (TCP Comm) VI is in the horizontal position, shown at left, indicating that you have not downloaded the VI to the RT target.

4. Close the **Hierarchy** window.
5. Select **Operate»Download Application** to download the VI. LabVIEW prompts you to save the VI. Save the VI as `Top Level Application.vi`.

   The block diagram should look similar to Figure 6.

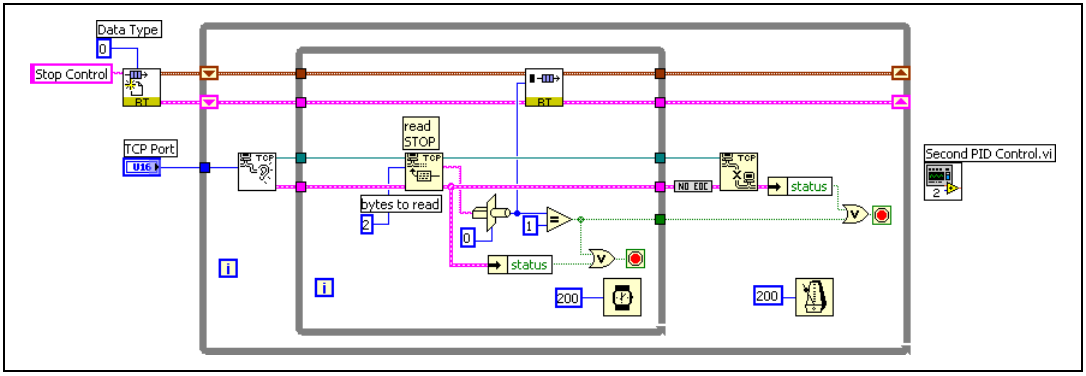

Figure 6. Communication VI

6. Select **View»Redraw** in the **Hierarchy** window to refresh the screen. Notice in the **Hierarchy** window that now the thumbtack for RT Engine - Normal Priority Loop (TCP Comm) VI is in the vertical position, shown at left, which indicates that you have downloaded the VI to the RT target.
7. Click the **Run** button. LabVIEW runs the VI on the RT target.
8. Select **File»Exit without closing RT Engine VIs** to close LabVIEW on the host computer while continuing to run the VIs on the RT target.
9. Start LabVIEW again and open the Host Machine (TCP Comm) VI. The Host Machine (TCP Comm) VI publishes and controls the **Stop** button in the time-critical VI on the RT target. Click the **Stop** button.
10. Select **Operate»Switch Execution Target** and select the RT target to reconnect to the target and verify if the RT Engine - Normal Priority Loop (TCP Comm) VI stopped running.
11. Exit LabVIEW by selecting **File»Exit** and, when prompted, close all VIs running on the RT target.

Optimizing the Control Application

Now that you have created a basic application, you can optimize the control application. Complete the following additional tasks to experiment with deterministic programming techniques.

- Optimize the time-critical VI further by moving all front panel control and indicator references from the time-critical loop VI to the host computer VI using the Real-Time FIFOs VIs and network communication. You can use the RT Communication Wizard to move all front panel controls and indicators out of a time-critical VI. Refer to Chapter 3, *Building Deterministic Applications*, of the *LabVIEW Real-Time Module User Manual* for information about using the RT Communication Wizard.
- Optimize the data acquisition. Other data acquisition VIs offer better performance for real-time applications. You can use hardware timing to validate that the control loop is behaving deterministically. Refer to the *NI-DAQmx Help* for information about timing control loops using the DAQmx - Data Acquisition VIs. Refer to the *LabVIEW Real-Time Module User Manual* for information about optimizing VIs for deterministic performance.
- Use different control algorithms. The PID Control Toolset, which contains many different control algorithms, installs with the Real-Time Module. In this exercise, you used the PID VI, but you can use the other control VIs available in the toolset.