

Getting Results with CompactRIO™ and LabVIEW™

This tutorial demonstrates how to develop a CompactRIO application in LabVIEW. The application uses a CompactRIO R Series Expansion system with LabVIEW to make a thermostat. While developing this simple application, you will learn the concepts and techniques necessary to develop your CompactRIO application.

Getting Started

This section lists the software and hardware used in the tutorial. This section also lists documents you may find helpful while completing the tutorial.

Software

- LabVIEW 7.1 or later
- LabVIEW FPGA Module 1.1 or later
- NI-RIO 1.1 or later for R Series Devices
- *LabVIEW FPGA Release Notes*—for software installation instructions

The LabVIEW Real-Time Module is not required for this tutorial.

Hardware

Even if you do not have the hardware used in this tutorial, you can still follow the steps to learn concepts for using CompactRIO with LabVIEW.

- PCI or PXI R Series device
- CompactRIO R Series Expansion chassis
- cRIO-9211 thermocouple input module
- cRIO-9474 digital output module
- Getting started guide for your R Series device—for R Series device installation instructions

- *CompactRIO R Series Expansion System Installation Instructions*—for information about installing the CompactRIO system and modules
- *cRIO-9211 Operating Instructions*—for information about connecting thermocouples to the cRIO-9211
- *cRIO-9472/9474 Operating Instructions*—for information about connecting devices to the cRIO-9474

Overview of the Application

The CompactRIO system reads the temperature of a thermocouple. If the current temperature is above the target temperature range, the system turns on a fan. If the data is below the target temperature range, the system turns on a heater. After completing this tutorial, you will be able to view the current temperature and set the target temperature range you want the thermostat to maintain.



Note You do not need an actual fan and heater to complete this tutorial.

For this tutorial, you will create an FPGA VI and a host VI. The FPGA VI is the VI you download to the FPGA device which, in this application, is an R Series device. You use the FPGA VI to read and write to the I/O channels of the CompactRIO modules. The FPGA VI also compares the actual temperature to the target temperature range and turns on either the fan or the heater.

The host VI communicates with the FPGA VI. You can run the host VI on a Real-Time (RT) target, such as a PXI RT controller, or on a Windows PC. You use the host VI to convert data read by and written by the FPGA VI into meaningful engineering units. You also use the host VI to enter the target temperature range and to view the actual temperature and the state of the fan and heater.

Hardware Configuration

Connect a CompactRIO R Series Expansion chassis to an installed R Series device. Install a cRIO-9211 in Slot 1 of the CompactRIO R Series Expansion chassis and a cRIO-9474 in Slot 2 of the chassis. Connect a thermocouple to channel 0 of the cRIO-9211. Refer to the *cRIO-9211 Operating Instructions*, available by selecting **Start»Program Files»National Instruments»CompactRIO»Search the CompactRIO Bookshelf**, for information about connecting thermocouples to the cRIO-9211.

Use channel 0 on the thermocouple input module to read the temperature.
Use channel 0 on the digital output module to control the fan and channel 1 to control the heater.

You are ready to develop the FPGA VI for your CompactRIO thermostat.

Developing the FPGA VI

Use the FPGA VI to read from and write to the I/O lines on the CompactRIO modules. To develop and use an FPGA VI with CompactRIO, you must create a LabVIEW Embedded Project (LEP) file, configure the CompactRIO system in LabVIEW, develop the FPGA VI, and build and download the VI to the FPGA device.

Complete the following sections to develop the FPGA VI for the CompactRIO thermostat.

Creating a LabVIEW Embedded Project File

FPGA VIs must be associated with an LEP file. The LEP file is a collection of VIs that make up your application. Use the Embedded Project Manager to organize VIs, configure settings for the VIs, configure CompactRIO devices, and configure channel aliases.

1. Launch LabVIEW.
2. Select the FPGA device that is connected to the CompactRIO R Series Expansion chassis from the **Execution Target** pull-down menu in the **LabVIEW** window.

The FPGA device appears in the **Execution Target** pull-down menu as **FPGA Device (*device*) RIOy::INSTR**, where *device* is the model of the device. If the FPGA device is not in the **Execution Target** pull-down menu, it may not be configured correctly. Refer to the *Accessing and Configuring FPGA Devices* section of the *LabVIEW FPGA Module Release Notes* for more information about configuring the R Series device.

If you are not using hardware with this tutorial, select **FPGA Device (*device*) <compile only>** from the pull-down menu.



Note Do not select **FPGA Emulator (*device*) RIOy::INSTR** when using the R Series device with CompactRIO. CompactRIO does not support emulation mode.

3. LabVIEW automatically launches the **Embedded Project Manager** window. In the **Embedded Project Manager** window, select **File»New**. The **New** dialog box appears.

4. On the **Projects** tab, enter `crio thermostat` in the **Project name** text box.
5. Click the Browse button. The **Select project directory** dialog box appears.
6. Browse to the directory where you want to save the LEP file. Click the **Select Cur Dir** button. The directory path appears in the **Project location** listbox.

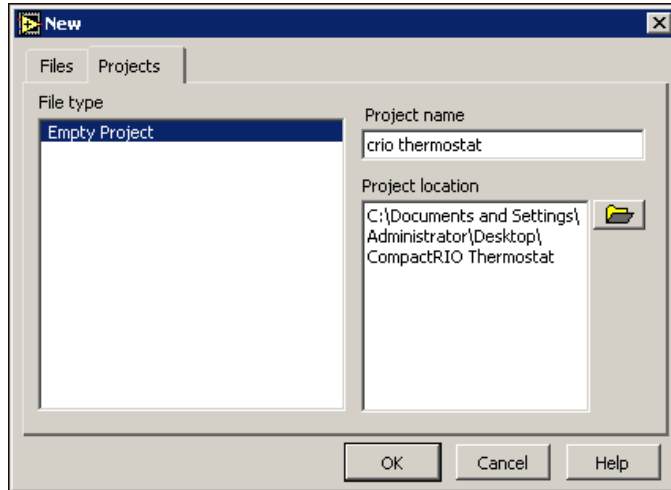


Figure 1. Creating a New Project

7. Click the **OK** button.

Configuring a CompactRIO R Series Expansion System in Software

1. In the **Embedded Project Manager** window, select **Hardware»Configure Target for CompactRIO**. The **Configure Target for CompactRIO** dialog box appears.
2. Select the connector that is connected to the CompactRIO R Series Expansion chassis.
3. Place a checkmark in the **Use Connector with CompactRIO** checkbox.
4. Select **1: <empty>**, which is Slot 1 of the CompactRIO chassis. The **Module Configuration** tab appears.
5. Select **cRIO-9211** from the **Module Type** pull-down menu. Enter `thermocouple` input in the **Module Name** text box.
6. In the left pane, select Slot 2 of the CompactRIO chassis.

7. On the **Module Configuration** tab, select **cRIO-9474** from the **Module Type** pull-down menu. Enter `digital output` in the **Module Name** text box.

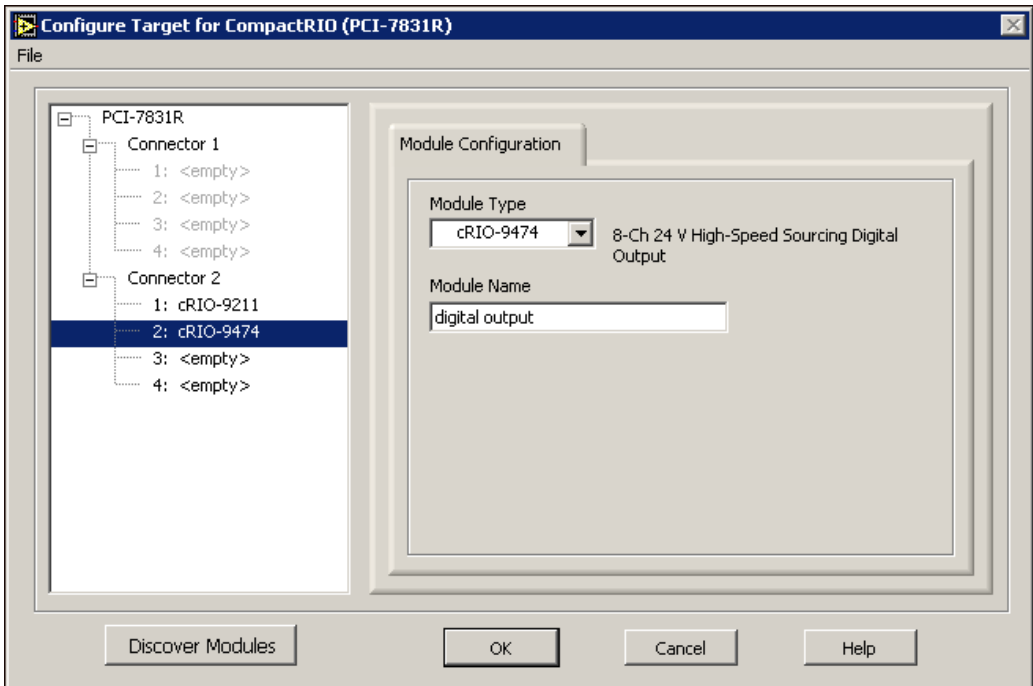


Figure 2. Configuring the CompactRIO System



Tip You can export CompactRIO configuration settings to a *RIO file*. A RIO file is a CompactRIO configuration file. You can import configurations from the RIO file into other LEP files. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help** in the **Embedded Project Manager** window, for more information about importing and exporting CompactRIO configuration files.

8. Click the **OK** button.
9. Select **File»Save Project** in the **Embedded Project Manager** window. LabVIEW saves the configuration settings in the LEP file.

Configuring Channel Aliases

You can configure aliases for the CompactRIO module channels. Configuring aliases for the channels makes your VIs portable so that you can easily use the same VIs with different CompactRIO systems.

1. In the **Embedded Project Manager** window, select **Hardware»Alias Manager**. The **Alias Manager** dialog box appears.
2. Select channel 0 of the thermocouple input module, **thermocouple input/TC 0**.
3. Click the **Add/Edit Alias** button. The **Enter Alias Name** dialog box appears.
4. Enter `actual temperature` in the **Alias Name** text box. Click the **OK** button.
5. Repeat steps 2 to 4 for the following I/O resources:

I/O Resource	Alias Name
digital output/DO 0	fan control
digital output/DO 1	heater control

6. To view only resources with aliases assigned to them, enable the **View by Alias** button.

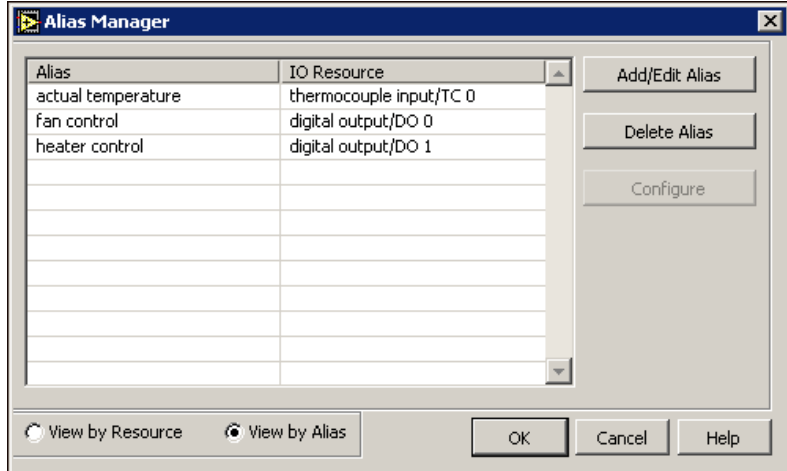


Figure 3. Confirming Channel Aliases

7. Click the **OK** button.
8. To view the aliases for the LEP file, select the **Aliases** tab in the **Embedded Project Manager** window.
9. Select **File»Save Project**.



Tip The aliases you configure are saved with the LEP file. You can import aliases to and from other LEP files. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for more information about importing and exporting aliases.

Developing the FPGA VI

1. In the **Embedded Project Manager** window, select **Project»Add to Project»New**. The **New** dialog box appears.
2. Select **Blank VI** in the left pane. Enter `crio thermostat (fpga)` in the **Filename** text box. Verify that the **Add to active project** checkbox is selected.

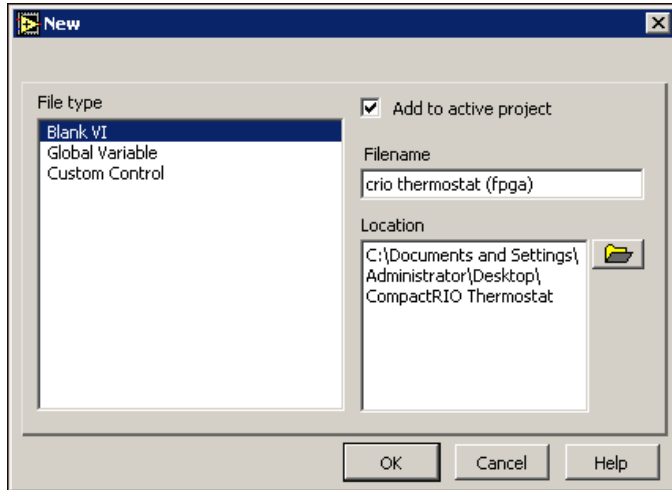


Figure 4. Creating a New VI

3. Click the **OK** button. The blank VI appears.

Reading the Temperature

1. Place a While Loop on the block diagram. Right-click the conditional terminal and select **Create»Control** from the shortcut menu.
2. Place an Analog Input function in the While Loop. The Analog Input function is on the FPGA Device I/O palette.

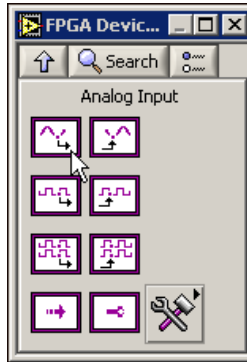


Figure 5. FPGA Device I/O Palette

3. Right-click the Analog Input function and select **Error Terminals** from the shortcut menu.



Tip NI recommends that you use error terminals in FPGA VIs. However, if an FPGA VI is too large for the FPGA, you can disable error terminals to decrease the size of the VI.

4. Double-click the function. The **Configure Analog Input** dialog box appears.
5. Select **actual temperature** from the **Alias** pull-down menu. Thermocouple input channel 0 is automatically selected in the **Terminal** pull-down menu.

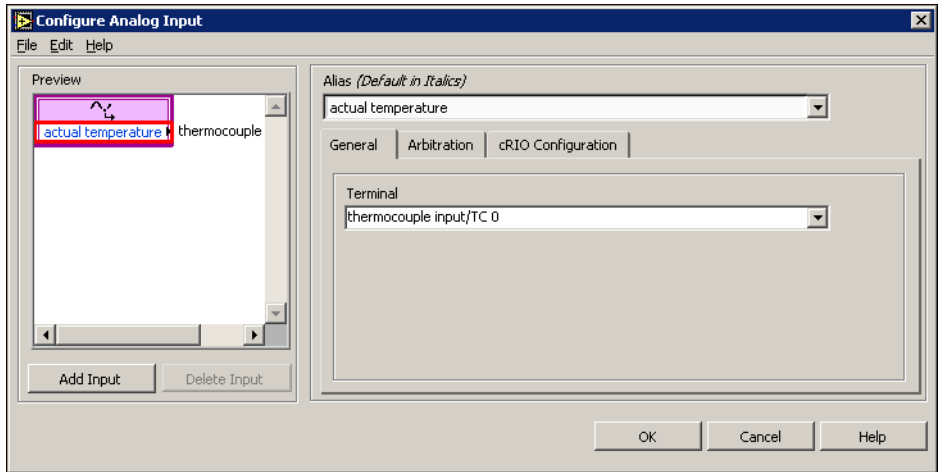


Figure 6. Configuring the Analog Input Function

6. To view the settings for the module you selected, click the **cRIO Configuration** tab.
7. Click the **OK** button.
8. Right-click the **actual temperature** terminal of the Analog Input function and select **Create»Indicator** from the shortcut menu. Label the indicator `actual temperature`.

Reading the CJC and the Autozero Channels

To ensure accurate measurements, you must read the cold-junction compensation (CJC) and autozero channels for the CompactRIO thermocouple module. These channels are internal-only channels on the cRIO-9211. Read these channels the same way you read the temperature in the *Reading the Temperature* section. You can use the same Analog Input function that you used to read the temperature.

1. Double-click the function. The **Configure Analog Input** dialog box appears.
2. Click the **Add Input** button and select **thermocouple input/Autozero** from the **Terminal** pull-down menu.
3. Click the **Add Input** button and select **thermocouple input/CJC** from the **Terminal** pull-down menu.

4. Click the **OK** button.
5. On the block diagram, right-click the **thermocouple input/Autozero** output and select **Create»Indicator** from the shortcut menu. Create an indicator for the CJC output as well. Label the indicators `autozero` and `CJC`, respectively.

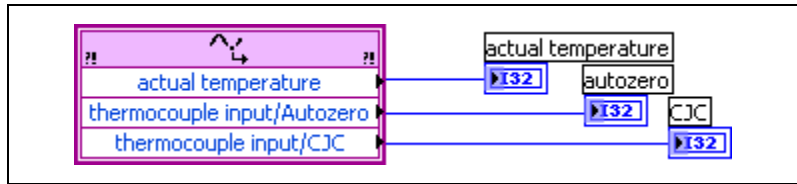


Figure 7. Reading the Autozero and CJC Channels

You will use the autozero and CJC data in the host VI to convert the data read from the thermocouple input channel from binary values to temperature.

Controlling the Temperature

To control the temperature, the FPGA VI must compare a target temperature range to the actual temperature and then decide whether to turn on the fan or the heater.

1. Place two Numeric Controls on the front panel. Label the controls `low target temperature` and `high target temperature`. The `low target temperature` control is for the lower limit of the target temperature range and the `high target temperature` control is for the upper limit of the target temperature range.
2. Place a `Less?` function and a `Greater?` function on the block diagram.
3. Wire the **actual temperature** output of the Analog Input function to the **x** inputs of the `Less?` and `Greater?` functions.
4. Wire the **low target temperature** control to the **y** input of the `Less?` function. Write the **high target temperature** control to the **y** input of the `Greater?` function.
5. Place a Digital Output function in the While Loop.
6. Right-click the Digital Output function and select **Error Terminals** from the shortcut menu.
7. Double-click the Digital Output function. The **Configure Digital Output** dialog box appears.
8. Select **fan control** from the **Alias** pull-down menu.
9. Click the **Add Output** button.
10. Select **heater control** from the **Alias** pull-down menu.
11. Click the **OK** button.

12. Wire the **x<y?** output of the Less? function to the **heater control** input of the Digital Output function. Wire the **x>y?** output of the Greater? function to the **fan control** input of the Digital Output function.
13. Wire the **Error Out** output of the Analog Input function to the **Error In** input of the Digital Output function.

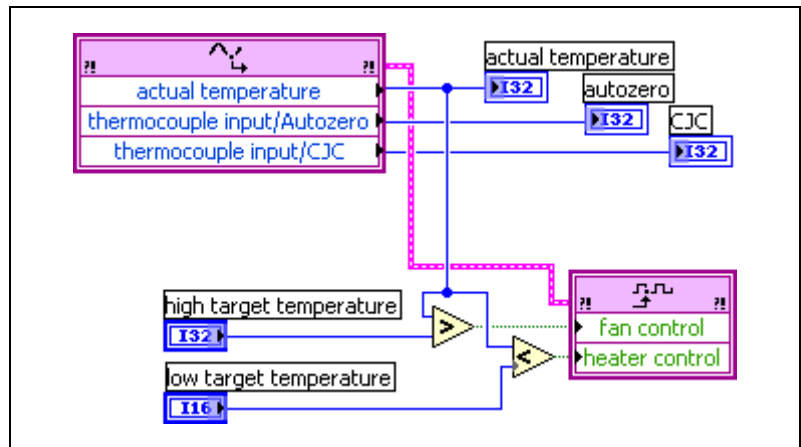


Figure 8. Controlling the Temperature

Reading the State of the Fan and the Heater

You can use the Digital Input function to read the state of CompactRIO digital output channels. The LEDs on the digital output module turn on and off to indicate the state of the channels. It is useful to confirm the state of the outputs in software as well.

1. Place the Digital Input function in the While Loop.
2. Right-click the Digital Input function and select **Error Terminals** from the shortcut menu.
3. Double-click the Digital Input function. The **Configure Digital Input** dialog box appears.
4. Select **fan control** from the **Alias** pull-down menu.
5. Click the **Add Input** button.
6. Select **heater control** from the **Alias** pull-down menu.
7. Click the **OK** button.
8. Create indicators for the **fan control** and **heater control** outputs of the Digital Input function. Label the indicators `fan on` and `heater on`, respectively.

9. Wire the **Error Out** output of the Digital Output function to the **Error In** input of the Digital Input function.
10. Right-click the **Error Out** output of the Digital Input function and select **Create»Indicator** from the shortcut menu.



Tip In this application, the Digital Input function executes after the Analog Input and Digital Output functions, so you can wire the error terminals between functions and only have one error indicator after the last function executes. However, if you have an application in which the I/O functions must execute simultaneously, create an **Error Out** indicator for each I/O function.

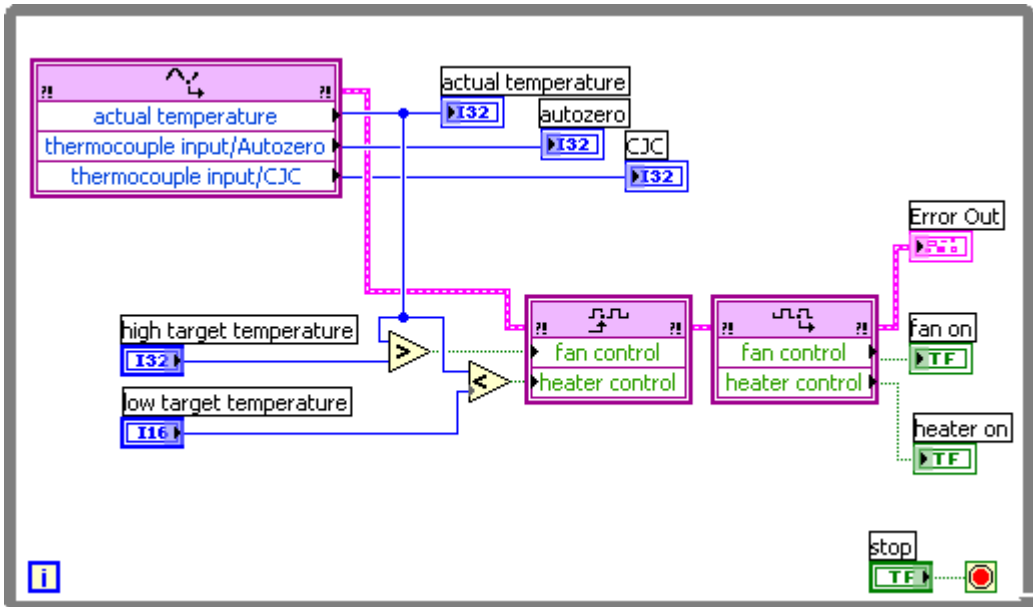


Figure 9. CompactRIO Thermostat FPGA VI

You have finished developing the FPGA VI. You are ready to build and download the VI to the FPGA device.

Building and Downloading the FPGA VI



1. In the **Embedded Project Manager** window, click the **Build** button.
2. In the **FPGA Project Builder** dialog box, confirm that the **VI Path** is correct. Make sure that **40 MHz** is selected in the **Design Clock Rate** pull-down menu.
3. Place a checkmark in the **Auto Run VI** checkbox.
4. Click the **Build** button.



5. In the **Warning: Beginning compile for FPGA** dialog box, click the **OK** button.
6. When LabVIEW returns the **Successful Compile Report** dialog box, click the **OK** button.
7. In the **Embedded Project Manager** window, click the Download button. Click the Run button when the VI has finished downloading. If you are not using hardware with the tutorial, you cannot download and run the VI.

The VI is now running on the FPGA of your device. Enter different target temperature ranges on the front panel of the VI. If the fan turns on, LED 0 on the cRIO-9474 and the **fan on** indicator on the front panel of the VI turn on. If the heater turns on, LED 1 on the cRIO-9474 and the **heater on** indicator on the front panel of the VI turn on.

Downloading the VI to Flash Memory

Now that the FPGA VI is compiled, you can download it to flash memory on an FPGA device and configure it to automatically load to the FPGA when the device powers on. You do not need LabVIEW or the LabVIEW FPGA Module to download and configure the load settings for the VI. All you need is the LabVIEW Run-Time Engine for 7.1 or later. This makes FPGA VIs easy to distribute after they have been developed and compiled. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for more information about downloading an FPGA VI to flash memory.

You can close the VI and the **Embedded Project Manager** window. If you have an application in which you want the CompactRIO thermostat to maintain the same target temperature range, your development is complete. However, in this tutorial you are going to develop a host VI so you can monitor the temperature and change the target temperature range after the FPGA VI is running on the FPGA device.

Developing the Host VI for CompactRIO

Use the host VI to communicate with the FPGA VI and to convert the data from the FPGA VI from binary values to meaningful engineering units. You can run the host VI on an RT target or on a Windows PC. Complete the following steps to develop the host VI for the CompactRIO thermostat.

1. In the **LabVIEW** window, select **LabVIEW for Windows** from the **Execution Target** pull-down menu.
2. Select **File»New VI**.

3. Place the Open FPGA VI Reference function on the block diagram.
Use this function to open a reference to the VI you downloaded to the FPGA device. Every host VI must use the Open FPGA VI Reference function.
4. Right-click the function and select **Select Target VI** from the shortcut menu. The **Select Target VI** dialog box appears.
5. Browse to the **crio thermostat (fpga) VI** and click the **OK** button.
6. Right-click the Open FPGA VI Reference function and select **FPGA»device** where *device* is the FPGA device you are using.
7. Place a While Loop on the block diagram. Right-click the conditional terminal and select **Create»Control** from the shortcut menu.
8. Place the Close FPGA VI Reference function outside of the While Loop. Every host VI must use the Close FPGA VI Reference function.
9. Place the Read/Write Control function inside the While Loop.
10. Wire the **HW Exec Ref Out** output of the Open FPGA VI Reference function to the **HW Exec Ref** input of the Read/Write Control function.

Reading and Converting the Actual Temperature

1. Right-click the **Unselected** output of the Read/Write Control function and select **actual temperature** from the shortcut menu. This terminal corresponds to the actual temperature indicator you created in the FPGA VI.

The actual temperature output returns binary data. Convert CompactRIO data to engineering units in the host VI as opposed to in the FPGA VI.
2. Select **Select a VI** from the **Functions** palette. The **Choose VI to Open** dialog box appears.
3. Browse to `LabVIEW 7.1\examples\FPGA\CompactRIO\cRIO-9211` and select **cRIO-9211 Support Files**. The **File Dialog** dialog box appears.
4. Select **Convert to Temperature (cRIO-9211).vi** and click the **OK** button.



Tip Every analog CompactRIO module has an example VI you can use as a subVI to convert the data into meaningful units. You can find these VIs in the `LabVIEW 7.1\examples\FPGA\CompactRIO` directory.

5. Wire the **actual temperature** output of the Read/Write Control function to the **Thermocouple Channel** input of the Convert to Temperature (cRIO-9211) VI.

6. Right-click the Read/Write Control function and select **Add Control** from the shortcut menu.
7. Right-click the **Unselected** output and select **CJC** from the shortcut menu. Wire the **CJC** output of the Read/Write Control function to the **CJC Channel** input of the Convert to Temperature (cRIO-9211) VI.
8. Add the autozero output to the Read/Write Control function the same way you added the actual temperature and CJC outputs. Wire the **autozero** output of the Read/Write Control function to the **Autozero Channel** input of the Convert to Temperature (cRIO-9211) VI.
9. Create constants for the **Temperature Units** and **Thermocouple Type** inputs of the Convert to Temperature (cRIO-9211) VI.
10. Right-click the **Linearized Temperature** output of the Convert to Temperature (cRIO-9211) VI and select **Create>Indicator** from the shortcut menu. Rename the indicator `actual temperature`.

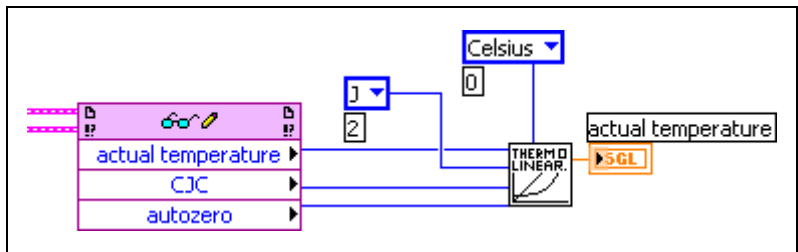


Figure 10. Reading and Converting the Actual Temperature

You can now read the temperature. Next, you need to write the target temperature range to the CompactRIO thermostat.

Converting and Writing the Target Temperature Range

You must write binary values to CompactRIO output modules. If you want to enter the value to output in engineering units, you must convert the value to a binary value before the host VI writes the target temperature range to the FPGA VI.

1. Select **Select a VI** from the Functions palette. The **Choose VI to Open** dialog box appears.
2. Browse to `LabVIEW 7.1\examples\FPGA\CompactRIO\cRIO-9211` and select **cRIO-9211 Support Files**. The **File Dialog** dialog box appears.
3. Select **Convert to Binary (cRIO-9211).vi** and click the **OK** button.
4. Place another **Convert to Binary (cRIO-9211) VI** on the block diagram.

5. Right-click the **Linearized Temperature** input of one of the Convert to Binary (cRIO-9211) VIs and select **Create»Control** from the shortcut menu. Label the control `low target temperature`. Create a **Linearized Temperature** control for the other Convert to Binary (cRIO-9211) VI and label it `high target temperature`.
6. Wire the **CJC** and the **autozero** outputs of the Read/Write Control function to the **CJC Channel** and **Autozero Channel** inputs of the Convert to Binary (cRIO-9211) VIs.
7. Wire the **Thermocouple Type** and the **Temperature Channel** constants to their respective inputs of the Convert to Binary (cRIO-9211) VIs.
8. Place another Read/Write Control function. Wire the **HW Exec Ref Out** output of the *first* Read/Write Control function to the **HW Exec Ref** input of the *second* Read/Write Control function.
9. Right-click the **Unselected** output and select **low target temperature** from the shortcut menu. Wire the **Thermocouple Channel (Binary)** output of the *first* Convert to Binary (cRIO-9211) VI to the **low target temperature** input of the Read/Write Control function.
10. Right-click the Read/Write Control function and select **Add Control** from the shortcut menu. Right-click the **Unselected** output and select **high target temperature** from the shortcut menu. Wire the **Thermocouple Channel (Binary)** output of the *second* Convert to Binary (cRIO-9211) VI to the **high target temperature** input of the Read/Write Control function.
11. Wire the **Hardware Exec Ref Out** output of the second Read/Write Control function to the **HW Exec Ref** input of the Close FPGA VI Reference function.
12. Wire the **error in** and **error out** clusters. Refer to the *LabVIEW Help*, available by selecting **Help»VI, Function, & How-To Help**, for more information about using error clusters.

Reading the State of the Fan and the Heater

1. Right-click either Read/Write Control function and select **Add Control** from the shortcut menu.
2. Right-click the **Unselected** output and select **fan on** from the shortcut menu. Right-click the **fan on** output and select **Create»Indicator** from the shortcut menu.
3. Right-click the Read/Write Control function and select **Add Control** from the shortcut menu. Create an output and an indicator for **heater on**.

Reading Errors from the FPGA VI

1. Right-click either Read/Write Control function and select **Add Control** from the shortcut menu.
2. Right-click the **Unselected** output and select **Error Out»All Elements** from the shortcut menu.
3. Right-click the **Error Out** output of the Read/Write Control function and select **Create»Indicator** from the shortcut menu.
4. Run the VI.

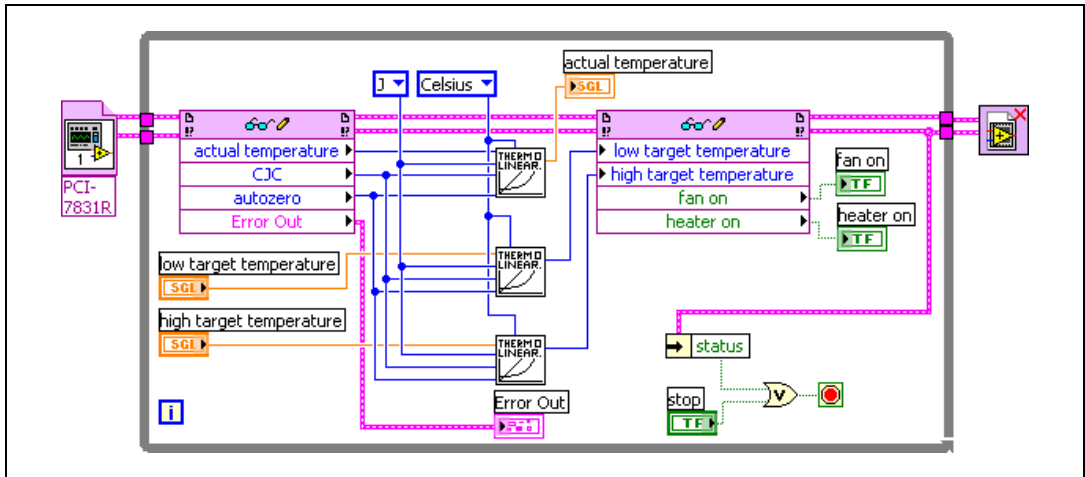


Figure 11. CompactRIO Thermostat Host VI

You have finished developing your CompactRIO thermostat. Enter different target temperature ranges and see how the cRIO-9474 LED and the indicators on the front panel react. While the FPGA VI handles the I/O and the logic, you can use the host VI to view the current temperature and set the target temperature range.

What You Have Learned

This tutorial taught the following key points about developing a CompactRIO application:

- A typical CompactRIO application consists of a LabVIEW Embedded Project (LEP) file, an FPGA VI, and a host VI. The FPGA VI runs on the FPGA of an FPGA device such as an R Series device. The host VI can run on an RT target or on a Windows PC.
 - Use the Embedded Project Manager to organize VIs, configure settings for the VIs, configure the CompactRIO devices, and configure the channel aliases.
 - Use the FPGA VI to read and write to the CompactRIO I/O channels and to implement logical operations in the FPGA.
 - Use the host VI to communicate with the FPGA VI.
- Use error terminals throughout a CompactRIO application. However, if the FPGA VI does not fit in the FPGA, disabling error checking decreases the size of the FPGA VI.
- CompactRIO modules communicate with FPGA VIs in binary values. You must write binary data to CompactRIO output modules. Convert CompactRIO data in the host VI as opposed to in the FPGA VI. Doing so helps prevent your FPGA VI from consuming too many FPGA resources.
- You can use the autoload and auto run features to deploy a stand-alone control system. You can download the compiled FPGA VI to flash memory on the R Series device. You can configure the VI to load and run when the device powers on.

