NI Single-Board RIO Hands-On Session

Welcome to the NI Single-Board RIO Hands-On Session. This document contains step-by-step instructions for getting started with LabVIEW embedded software and hardware, specifically the LabVIEW Real-Time and LabVIEW FPGA development modules and the NI Single-Board RIO hardware platform.

Table of Contents

Before You Begin – LabVIEW and Your Embedded Hardware Basics .................................................... 3
EXERCISE 1 – Open and Run a FPGA Application ................................................................................... 5
EXERCISE 2 – Create an FPGA Application .............................................................................................. 9
EXERCISE 3 – Creating a Real-Time Application ................................................................................... 21
Explore Additional Examples ................................................................................................................ 27
Changing the IP Address in the LabVIEW Project.................................................................................. 28
Tutorial Overview

In this tutorial, you will complete four exercises that demonstrate how to use LabVIEW to create Real-Time Processor (RT) and field-programmable gate array (FPGA) applications:

- **Exercise 1** – Open and run a precompiled FPGA application and review the documentation of the source code to understand how the FPGA application works.
- **Exercise 2** – Create an FPGA application on your own. While this application is compiling, complete Exercise 3.
- **Exercise 3** – Extend the FPGA application you created in Exercise 2 to interface with an application executing on the reconfigurable I/O (RIO) device RT.

After completing these exercises, explore a variety of FPGA as well as RT and FPGA applications at ni.com/embeddedeval. These examples will demonstrate how to implement common tasks and functions used in embedded applications using LabVIEW.
Before You Begin – LabVIEW and Your Embedded Hardware Basics

If you are new to LabVIEW, this section will help you learn more about the LabVIEW environment and graphical programming language.

The hardware device included in your kit is based on the NI Single-Board RIO reconfigurable I/O platform. The device consists of two processing devices: an RT that you can program with the LabVIEW Real-Time Module and a FPGA that you can program with the LabVIEW FPGA Module. These units are connected via a PCI bus and the LabVIEW development environment includes many interfaces for communicating between them. Moving forward, this hardware device will be referred to as a “RIO device.”

A LabVIEW application is called a “VI”, or virtual instrument, and is composed of two primary elements, a front panel and a block diagram, which you can program using the Functions palette. See below for a definition of these terms:

- **Front panel** – The front panel is what you use to create a LabVIEW user interface (UI). For embedded applications, such as FPGA applications, you use the front panel to define sockets/registers that are exposed to other elements of your system (such as the RT) with read/write access.
  - **Note:** If you close the front panel, it will also close the block diagram, so be sure to minimize it instead if you wish to explore the block diagram.

- **Block diagram** – The block diagram is where you program LabVIEW applications using a combination of graphical and textual notations. To program in the block diagram, right-click anywhere on the diagram (blank white window) to bring up the Functions palette.

- **Functions palette** – The Functions palette contains components for creating FPGA, RT, and host interface applications. To do this, place the components on the block diagram and wire them together. When learning LabVIEW, many users find it useful to adjust the configuration of their Functions palette so icons and text for each function are displayed. To adjust your environment, use the menu bar of any LabVIEW project or application and select **Tools»Options»Controls/Functions Palettes** and change the format to “Category (Icons and Text).” See image below.
**Using the functions palette** – In this tutorial, blue, bold text denotes an item to select from the Functions palette. To access the Function palette, right-click anywhere on the LabVIEW block diagram and then left-click on the specified object. Left-click a second time to place or drop the object onto the block diagram.

For example, you would follow these steps to place **Programming»Numeric»Add**.

1. Right-click to access the Functions palette
2. Select the “Numeric” icon (Numeric palette will automatically appear)
3. Select the “Add” function and left-click
4. Move the add icon to the block diagram and left-click to place
EXERCISE 1: Open and Run a Dynamic RT-FPGA Application

Exercise 1 highlights the capability of the RIO to dynamically load bit files to the FPGA. The RT (Real-Time processor) allows the user to select between two application bit files to load on to the FPGA, which then communicate their results back to the RT for display.

1. Open “\1 - Open & Run Dynamic Example\ Example 1 Open & Run Dynamic Example.lvproj”
2. Change the IP Address in the project to match the IP address of your evaluation board (see the CHANGING THE IP ADDRESS section of this document for directions)
3. Expand the RIO device by clicking on the “+” symbol next to your RIO device IP address. This level shows all of the resources available on your embedded evaluation hardware and contains the code, libraries, and supporting documentation that will execute on the device.
4. Expand the FPGA resource on your RIO device by clicking on the “+” symbol next to the FPGA Target for your RIO device. This level shows all of the resources available specifically on the FPGA of your RIO device and contains the code, libraries, and supporting documentation for code that will execute on the FPGA.
5. Double-click on Main.vi to open the RT application

6. Select Window»Show Block Diagram from the menu bar to display to source code for the host application.
7. Click on the white run arrow to download and run the application on the Real-Time processor.

8. A pop-up dialog box will allow you to select between the different applications to deploy to the FPGA
9. Select the Simple Counter application from the list and click Enter Selection. The RT will now deploy this bitfile to the FPGA.
10. Once the application is deployed, you can interact with it by turning the encoder knob on your RIO evaluation board and observe the incrementing count on the LED bank. The indicator on the RT front panel will also update with the corresponding count value.

11. Click the Stop button to bring up the FPGA Application selector dialog box.

12. Select the Two Edge Count and click the Enter Selection button to load the FPGA Application.

13. Once the application is deployed, you can interact with it by asserting Push Button 1 (PB1) and then Push Button 2 (PB2) on your RIO evaluation board and observe the delay between the button presses on the main VI.

14. Click the Stop button to bring up the FPGA Application selector dialog box again.

15. Press the Exit button on the dialog box to exit the RT application.

**Review the FPGA Application Architectures**

16. In the Project Explorer, expand the FPGA Target and double-click the Simple Counter VI to open the application.

17. Select *Window* » *Show Block Diagram* from the menu bar to display the source code for the host application.
18. Review the documentation below to understand how the Simple Counter application code implements its functionality:

The Simple Counter FPGA application reads a digital inputs from channel A and B of the Quadrature Encoder and depending on whether A leads B or B leads A increment or decrement the count. The boolean representation of the count is then displayed on the 8 LEDs. Because this is an 8-bit counter, the count will roll over after 255 events have been detected.

19. Close the Simple Counter Example VI.

20. Double click the second FPGA Application, the Two Edge Example VI to open the application.

21. Select Window » Show Block Diagram from the menu bar to display to source code for the FPGA application.

22. Review the documentation below to understand how the Two Edge application code implements its functionality:
The Two Edge FPGA application calculates the time lapse between when push button 1 (PB1) and 2 (PB 2) are asserted. To complete this process, the IP Block uses a feedback node to store the previous value of PB1 and PB2 and trigger the corresponding case structure when it sees a rising edge of one of the push buttons.

If a rising edge is detected on PB1, then the current value of tick count is stored in a feedback node. If a rising edge is detected on PB2, then the stored tick count from PB1 assertion is subtracted from the current tick count.

23. Close the IP Block as well as the Two Edge Count VI.
24. Close this project selecting File ➤ Exit in the Project Explorer window. Don’t save the changes that you made to the project.
EXERCISE 2 – Create a FPGA Application

In Exercise 1, you opened a RT application that controlled the deployment of multiple precompiled FPGA applications that ran on your RIO device. In this exercise, you will build a complete FPGA application, synthesize it, and deploy it to the FPGA on your RIO device.

You will begin by opening a pre-created blank project from the exercise folder and ensuring you have a connection to your RIO device.

1. Open “\2 – Create RT + FPGA Application\Example 2 (LED Control).lvproj.”

2. Change the IP address in the project to match the IP address of your evaluation board.

3. Expand the RIO device by clicking on the “+” symbol next to your RIO device IP address. This level shows all the resources available on your embedded evaluation hardware and contains the code, libraries, and supporting documentation that will execute on the device.

4. Expand the FPGA resource on your RIO device by clicking on the “+” symbol next to the FPGA target for your RIO device. This level shows all the resources available specifically on the FPGA of
your RIO device and contains the code, libraries, and supporting documentation for code that will execute on the FPGA.

5. Right-click on the FPGA Target and select **New»VI**.
**Application Overview**

This application reads the analog value of a potentiometer whose range is between 0 and 4 VDC and displays its relative value on a bank of eight LEDs. In this design, the potentiometer input will be scaled to be between 0 and 8 with a resolution of 1. The scaled range will be mapped to the bank of 8 LEDs. The diagram below shows the FPGA code you will create in this exercise. The step-by-step documentation following will guide you through the creation this FPGA application in LabVIEW.

![Diagram of FPGA code](image)

**Read Analog Value from Potentiometer** – RIO devices offer a variety of interfaces including AIO, DIO, digital communications protocols, specialty I/O, and memory. You can use the FPGA I/O Node to implement read/write operations to these interfaces.

6. Ensure you start from the block diagram (Window»Show Block Diagram or Ctrl + E) and right-click to access the Functions palette.

7. Select Programming»FPGA I/O»I/O Node in the Functions palette and place the FPGA I/O Node on the left side of the block diagram.
8. Configure the FPGA I/O Node to read from the potentiometer analog input channel.
   a. Left-click on “I/O Item” on the FPGA I/O Node and select Eval Acc IO»AI»POT.

   ![Diagram of FPGA I/O Node and potentiometer]

   **Scale Input for LED Display** – The potentiometer returns a value between 0 and 4 VDC in increments of 4.7684E-7 VDC. In this section, you will scale that input to be between 0 and 8 in increments of 1 to match the “range” of the LED bank.

9. Select Programming»Numeric»Multiply from the Functions palette and place the Multiply block to the right of the FPGA I/O block you configured in the previous section.

10. Connect the output of the potentiometer read to the ‘x’ input of the multiply function.
    a. Place your mouse pointer over the black arrow on the right side of the FPGA I/O block and left-click.

    ![Diagram of connection from FPGA I/O to Multiply block]

    b. Place your mouse pointer over the top-left corner of the Multiply block and left-click.

11. Create a constant for the Y input of the multiply function.
    a. Place your mouse pointer over the bottom-left corner of the Multiply block (the ‘y’ terminal) and right-click to select Create»Constant.
b. Double left-click in the constant box and type “2” to replace the default constant value of “0.” (Note – the trailing 0’s are the default representation for fixed-point numerics and can be adjusted using the properties dialog by right-clicking on the constant)

12. Select Programming»Numeric»Round To Nearest from the Functions palette and place the Round To Nearest block to the right of the Multiply block.

13. Connect the output of the Multiply block to the input of the Round To Nearest block.
   a. Place your mouse pointer over the right corner of the Multiply block (‘x*y’ terminal) and left-click.
   b. Place your mouse pointer over the left side of the Round To Nearest block (‘number’ terminal) and left-click.

**Update LED Bank Register** – To write to the bank of eight LEDs on the hardware, create a register using an array of Boolean values. Depending on the scaled value of the potentiometer, one of the eight elements will be asserted, illuminating the corresponding LED. In this section, you will create the code that will update the appropriate element based on the scaled potentiometer value.

14. Select Programming»Array»Array Constant from the Functions palette and place the Array Constant to the right and above the Round To Nearest block.
15. Select Programming»Boolean»False Constant from the Functions palette and place the False Constant inside the Array Constant.

16. Right-click on the “0” in the Array Constant and select “Set Dimension Size” to specify the size of the array.
   a. Change the size from ‘variable’ to ‘fixed’.
   b. Set the number of elements to eight.
   c. Resize the Array Constant to show all eight elements by left-clicking and holding on the bottom edge of the right-half of the Array Constant and dragging it down to expose 8 false constants.

17. Select Programming»Array»Replace Array Subset and place the Replace Array Subset block to the right of the Round To Nearest block.

18. Connect the Array Constant output (on the right) to the array input terminal (on the left) of the Replace Array Subset block.
   a. Place your mouse pointer over the right side of the Array Constant and left-click.
   b. Place your mouse pointer over the top-left side of the Replace Array Subset block (‘array’ terminal) and left-click.
19. Connect the Nearest Integer Value output of the Round To Nearest block to the index input of the Replace Array Subset block.
   a. Place your mouse pointer over the right corner of the Round To Nearest block (‘nearest integer value’ terminal) and left-click.
   b. Place your mouse pointer over the middle-left side of the Replace Array Subset block (‘index’ terminal) and left-click.

20. Create a True Constant of the element input terminal of the Replace Array Subset block.
   a. Place your mouse pointer over the bottom-left side of the Replace Array Subset block (‘new element/subarray’ terminal), right-click, and select Create»Constant.
   b. Left-click on the False Constant (F) to change it to a True Constant (T).

21. Select Programming»Array»Index Array from the Functions palette and place the Index Array block to the right of the Replace Array Subset block.

22. Connect the output array terminal of the Replace ArraySubset block to the array terminal of the Index Array block.
   a. Place your mouse pointer over the right side of the Replace Array Subset block (‘output array’ terminal) and left-click.
   b. Place your mouse pointer over the middle-left side of the Index Array block and left-click.

23. Expand the Index Array block to show eight output terminals.
   a. Place your mouse pointer over the bottom edge of the Index Array block and click-and-hold the left mouse button to drag the icon down until eight output terminals are exposed on the right side of the Index Array block.

Write to LED Bank – Now that you have created an LED bank register, you need to map that register to the LED bank so changes to the register are written to the LED bank.

21. Select Programming»Array»Index Array from the Functions palette and place the Index Array block to the right of the Replace Array Subset block.

22. Connect the output array terminal of the Replace Array Subset block to the array terminal of the Index Array block.
   a. Place your mouse pointer over the right side of the Replace Array Subset block (‘output array’ terminal) and left-click.
   b. Place your mouse pointer over the middle-left side of the Index Array block and left-click.

23. Expand the Index Array block to show eight output terminals.
   a. Place your mouse pointer over the bottom edge of the Index Array block and click-and-hold the left mouse button to drag the icon down until eight output terminals are exposed on the right side of the Index Array block.
24. Select Programming»FPGA I/O»I/O Node from the Functions palette and place the FPGA I/O Node to the right of the Index Array block.

25. Configure the FPGA I/O Node to write to the LED bank.
   a. Left-click on I/O item and select Eval Acc IO»LED»LED 0.
   b. Right-click on LED 0 and select “Change to Write.”
   c. Place your mouse pointer over the bottom edge of the LED 0 block and click-and-hold the left mouse button to drag the LED nodes until ‘LED 0’ through ‘LED 7’ are exposed.

26. Connect the Index Array output terminals to the FPGA I/O Node input terminals.
   a. Place your mouse pointer over the top-right output node and left-click.
   b. Place your mouse pointer over the left edge of LED 0 and left-click.
   c. Repeat for LED 1 through LED 7. (Note – do not worry about overlapping wires, use the Clean-up Diagram function to automatically clean-up your block diagram – see below or use the shortcut Ctrl+U)
Create Socket/Register for RT Code – Although it is not necessary for this exercise, in Exercise 3 you will extend this exercise to interface with code running on the RIO device RT. To avoid the need to recompile the FPGA application, add a socket now so that the RT application you will create in Exercise 3 can read the current value of the POT.

27. Place your mouse pointer over the wire leaving the FPGA I/O Node that is reading from the potentiometer, right-click, and select “Create Indicator”. After using the Diagram Clean-up tool (Ctrl+U) again, you should have a block diagram that looks similar to the one below:

Specify Execution Timing – If the FPGA application you just created was compiled and run at this point, it would execute only once. You can specify continuous execution of your application by using a looping structure. There are several looping structures available in the LabVIEW FPGA Module. For this exercise, you will use a While Loop. This loop structure specifies that your circuit executes continuously. You can slow the execution speed down using a block from the Timing palette or speed it up by changing the default clock for your FPGA target.
28. Place a While Loop around the complete circuit.
   a. Select Programming Structures While Loop from the Functions palette.
   b. Move your mouse pointer to the top-left side of your code and left-click.
   c. Move your mouse pointer to the bottom-right side of your code so the entire block of code is contained in the dashed box and left-click.
   **Note** – Another method to create the While Loop is to left-click-and-hold in step 23b (instead of releasing the mouse button) and drag to the other corner of the While Loop before releasing the mouse button.
   d. Place your mouse pointer over the red circle in the bottom-right of the While Loop, right-click, and select “Create » Constant.” **Note:** The red circle is the stop condition for the While Loop; wiring a False Constant to this circle specifies that the code will run indefinitely.
Compile and Run FPGA Application — Your FPGA application is complete. Save your project and begin synthesizing your FPGA application so it can execute on the FPGA.

29. Save your FPGA application if you have not already done so.
   a. Select File\Save from the menu bar and save the FPGA application as “.\2 - Create FPGA Personality\FPGA Personality\Display POT Value.vi.”

30. Compile and run your application.
   a. Select the white run arrow on your application and LabVIEW will automatically start the compilation process, displaying the “Generation Intermediate Files” dialog.

At this point, VHDL code is being generated from the FPGA application you designed using LabVIEW FPGA. Once generated, the VHDL will be compiled using the Xilinx toolchain. As mentioned previously, this step was already done for you in Exercise 1.
While this compile is taking place (estimated 10 minutes), continue on to Exercise 3. In order to do this, you will need to first disconnect from the LabVIEW FPGA Compile Server so that you can continue to work in LabVIEW:

a. Click on the LabVIEW FPGA VI (application) window you created in exercise 2 (Display POT Value.vi) to bring the Compile Status Window to the front (or select this window from your task bar).

b. Click on the “Disconnect” button

c. Minimize, but do not close, all of the LabVIEW windows that are currently open:
   - Display POT Value.vi Front Panel
   - Display POT Value.vi Block Diagram
   - LabVIEW FPGA Compile Server

You will reconnect to the LabVIEW FPGA Compile Server after you are finished with Exercise 3.
EXERCISE 3 – Creating a Real-Time Application

In Exercise 2, you created a FPGA personality that reads the value of the potentiometer channel and displays the relative value on the LED bank. In this exercise, you will add a RT application that interfaces this data back to your RT and display it.

1. In the Example 2 Project Explorer, right-click on your RIO Device and select **New»VI**

2. Ensure you start from the block diagram (Window»Show Block Diagram or Ctrl + E) and right-click to access the Functions palette.

Create Link to FPGA Resource – the following steps will specify which FPGA personality is to be loaded to the FPGA on your RIO device when the RT application is run.

3. Select **FPGA Interface»Open FPGA VI Reference** from the Functions palette and place it on the left side of your diagram.

4. Right-click on the Open FPGA VI Reference object and select ‘Configure Open FPGA VI Reference’

5. Select the ‘VI’ option and choose ‘Display POT Value.vi’ to specify that the FPGA Personality you created in Exercise 2 is to be loaded and run on the FPGA when this function is called.
Read Potentiometer Value and Display – The following steps will read the value of the potentiometer that is acquired by your FPGA personality (‘Display POT Value.vi’).

6. Select FPGA Interface»Read/Write Control from the Functions palette and place it on the right of the ‘Open FPGA VI Reference’ function you just placed.

7. Connect the ‘FPGA Reference’ and ‘error’ terminals of the two FPGA objects you just placed as shown below.

8. Left-click on ‘Unselected’ in the Read/Write Control and select ‘POT’ to specify that this function is to read from the POT register of your FPGA personality.

9. Select Programming»Numeric»Conversion »To Double Precision Float from the Functions palette and place it on the right of the ‘Read/Write Control’ function you just placed. This function will convert the fixed-point data received from your FPGA personality to a double precision floating point data type.
10. Wire the ‘POT’ output of the ‘Read/Write Control’ function to the ‘number’ (left) input terminal of the ‘To Double Precision Float’ function you just placed.

![Diagram showing wiring connection]

11. Switch to the front panel (*Window*›*Show Block Diagram* or Ctrl + E)

12. Select *Modern*›*Graph*›*Waveform Chart* from the Controls palette and place it on the front panel.

![Waveform Chart diagram]

13. Double left click on the waveform chart to locate it in the block diagram.

14. Move the waveform chart icon in the block diagram near the ‘To Double Precision Float’ function and wire them together as shown below:

![Diagram showing wiring connection]

15. Select *FPGA Interface*›*Close FPGA VI Reference* from the Functions palette and place it on the right of the ‘Read/Write Control’ function that you created earlier in this exercise.

16. Connect the FPGA Reference and Error terminals of the ‘Read/Write Control’ and ‘Close FPGA VI Reference’ functions as shown below.

![Diagram showing wiring connection]
17. Place a While Loop around the ‘Read/Write Control’ function and the POT Value indicator
   a. Select Programming»Structures»While Loop from the Functions palette.
   b. Move your mouse pointer to the top-left side of your ‘Read/Write control’ and left-click.
   c. Move your mouse pointer to the bottom-right side of your POT Value indicator so the
      entire block of code is contained in the dashed box and left-click.
18. Place your mouse pointer over the red circle in the bottom-right of the While loop, right-click,
    and select Create Control to create a stop button for your RT application.
19. Select Programming»Timing»Wait (ms) from the Functions palette and place inside the while
    loop.
20. Right-click on the Milliseconds to Wait input of the Wait VI and select Create»Constant.
21. Double left click in the constant and enter a value of 100.

22. Save your RT application if you have not already done so.
    a. Select File»Save from the menu bar and save the RT application as “\2 - Create FPGA
       Personality\Read POT Value.vi.”

23. In order to now run your FPGA application:
    a. Right-click on your FPGA application (Display POT Value.vi) in the Example 2 Project Explorer
       and select “Reconnect to Compilation”.
24. Close the Successful Compile Report window by selecting OK.

25. Close the LabVIEW FPGA Compile Server window.

26. Select **File»Save All** in the **Project Explorer** menu bar.

27. Open your FPGA application (**Display POT Value.vi**) and click on the white run arrow to download and deploy your application to the RIO device.
28. Turn the potentiometer knob of your evaluation board once your FPGA application downloads to the RIO device and observe the behavior of the LED bank.

29. While your FPGA Application is still running, open your RT application (*Read POT Value.vi*) and click on the white run arrow to download and deploy your application to the Real-Time processor.

30. Turn the potentiometer knob of your evaluation board once your RT application downloads to the device and observe the behavior of the POT Value indicator on your front panel.

31. Stop your RT application by clicking the Stop button on your Read POT Value VI.

32. Stop your FPGA application by left-clicking on the Abort Execution button.

33. Close this project selecting **File»Exit** in the Project Explorer window.

**Congratulations!** You have successfully created and deployed a LabVIEW RT and FPGA Application to your NI Single-Board RIO device.
Next Steps - Explore Additional Examples

Now that you are familiar with how to create and run FPGA and RT-FPGA applications using LabVIEW, you have several options to learn more about using LabVIEW for embedded applications.

1. Explore a variety documented examples that demonstrate how to implement common tasks and functions used in embedded applications using LabVIEW. These examples are ready-to-run on your RIO device include:
   - Watch dogs
   - Additional counter implementations
   - Pulse Width Modulation
   - FPGA-based Fast Fourier Transform (FFT)
   - FPGA-based filtering
   - Quadrature encoder position measurement
   - And many more RT and FPGA implementations

Navigate to ni.com/embeddedeval using your web browser to view a complete list of examples ready-to-run on your RIO device.

2. Select Help»Find Examples to open the LabVIEW Example Finder. This utility makes it easy to search 100’s of LabVIEW examples to learn more about how to implement FPGA and RT-FPGA applications in LabVIEW.
Changing the IP Address in the LabVIEW Project

Your RIO device is identified by its IP address. For each exercise, confirm that the IP address in the project matches the IP address of your RIO device. The setup utility should have prompted you to write down the RIO device IP address, but you also can locate the device through the following steps. **If you already know your IP address, skip to Step 3.**

1. Right-click on the RIO device and select properties to change the IP address.
   a. Determine the IP address of your RIO device by opening the Measurement & Automation Explorer with the icon on your desktop or by selecting `Start»Programs»National Instruments»Measurement & Automation.`
   b. Click on the “+” symbol next to Remote Systems.
2. Click on the device that appears below to locate the IP address.
3. Return to your project, right-click on your RIO device, and select properties to update the IP address.

---

![Image of Measurement & Automation Explorer](image.png)

![Image of LabVIEW project properties](image.png)