Version 16.01

Hands-On Tasters

Please do not remove this manual.

You will be sent an email which enables you to download the presentations and manual. Thank you.

uk.ni.com
ireland.ni.com

NATIONAL INSTRUMENTS

NATIONAL INSTRUMENTS

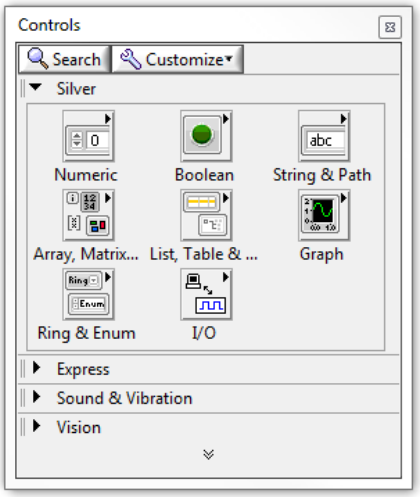
Introduction to LabVIEW

Exercise 1: Create a Simple LabVIEW VI

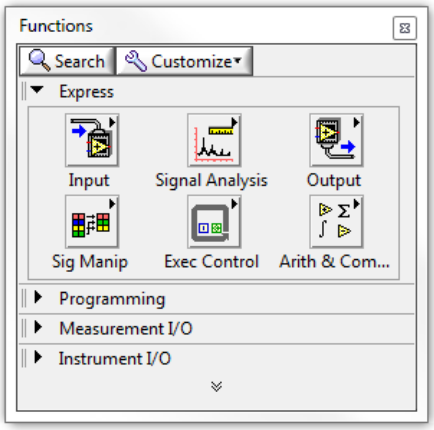
In this exercise you will create a simple piece of LabVIEW code - a VI - that simulates an analogue signal and plots it on a waveform graph. The VI will test the input values against a user-specified limit and light an LED if the input value exceeds that limit.

Below are pictures identifying each of the palettes found in LabVIEW to assist you as you complete these exercises.

Note: LabVIEW has a built-in Automatic Tool Selection feature that changes the behaviour of the cursor depending on what type of object or which part of an object you are currently pointing to.



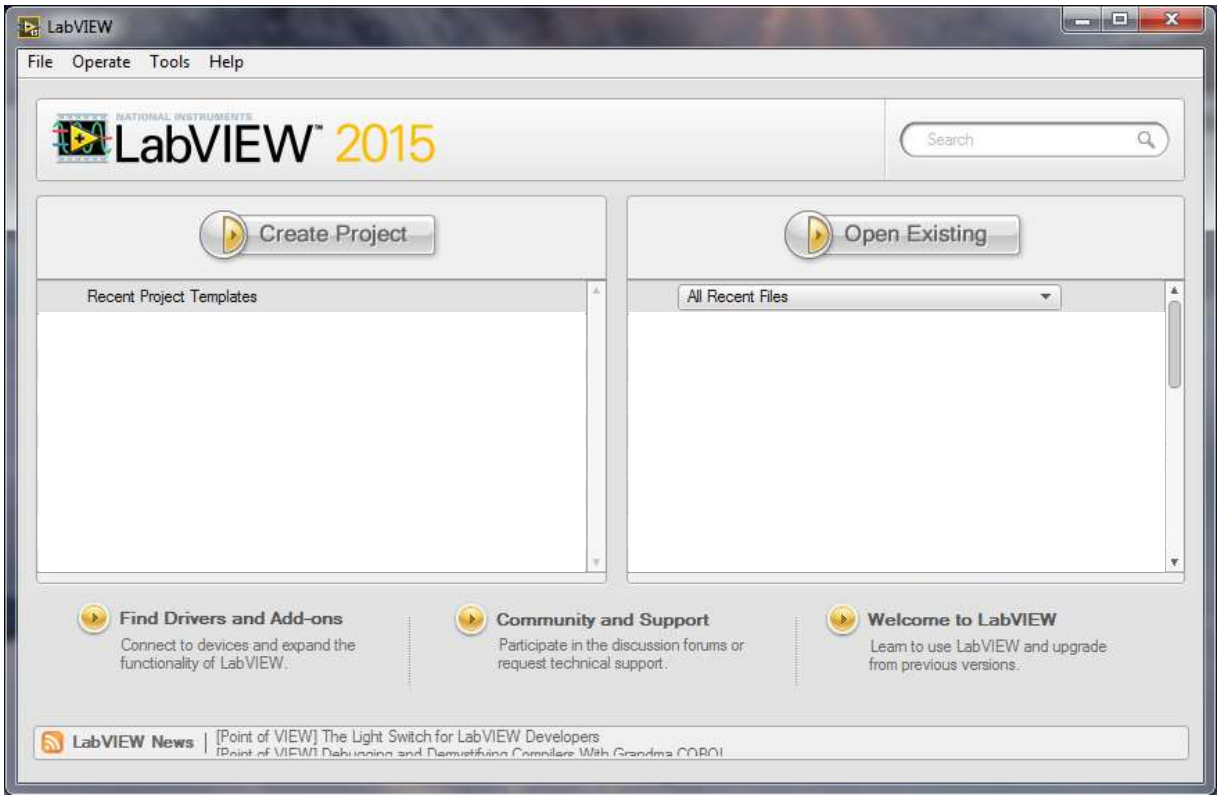
Controls Palette



Functions Palette

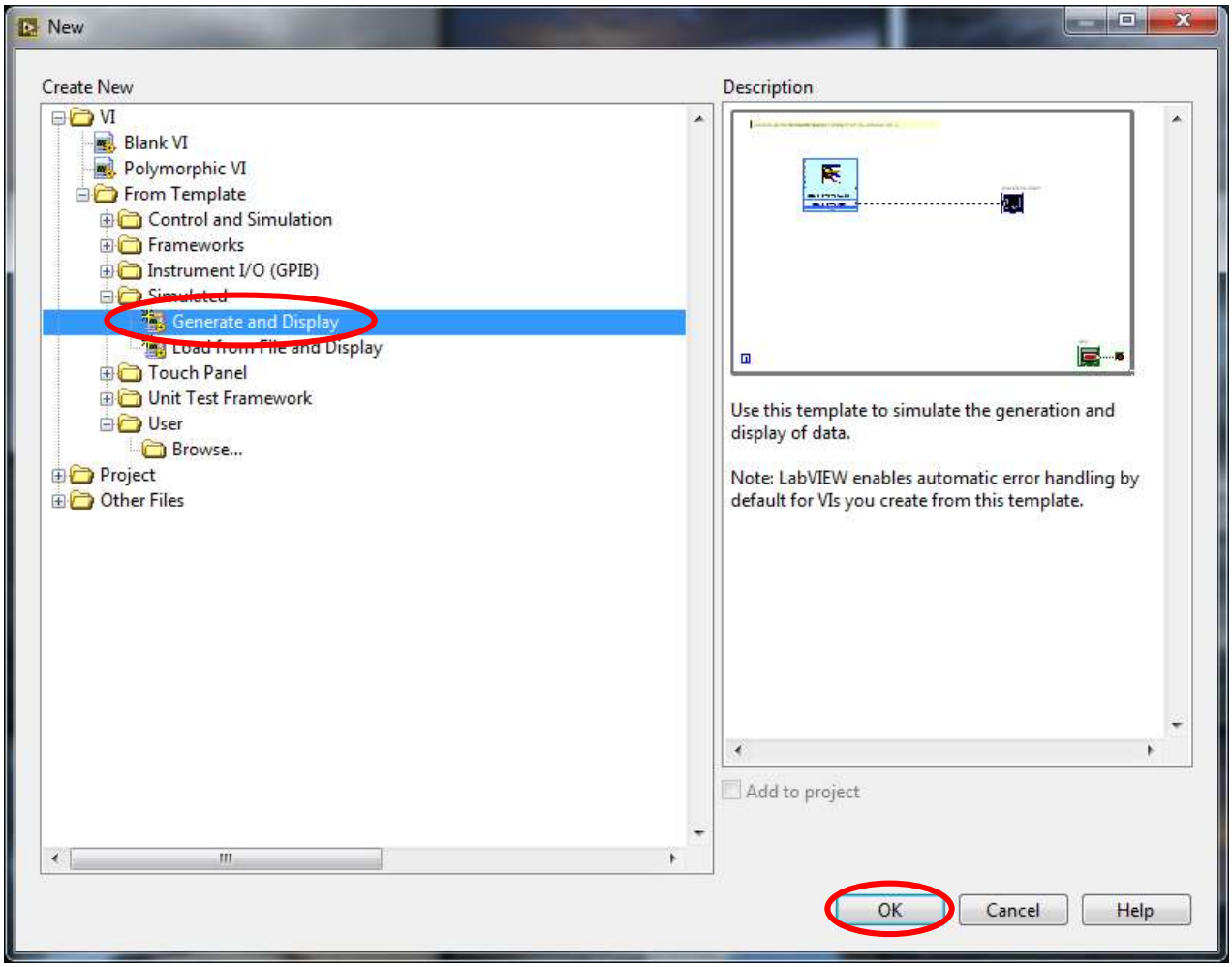
1. Navigate to **Start** **» All Programs» National Instruments» LabVIEW 20xx**, and select **LabVIEW**.

2. Once you launch LabVIEW, a splash screen like the following appears. Click **File » New…**



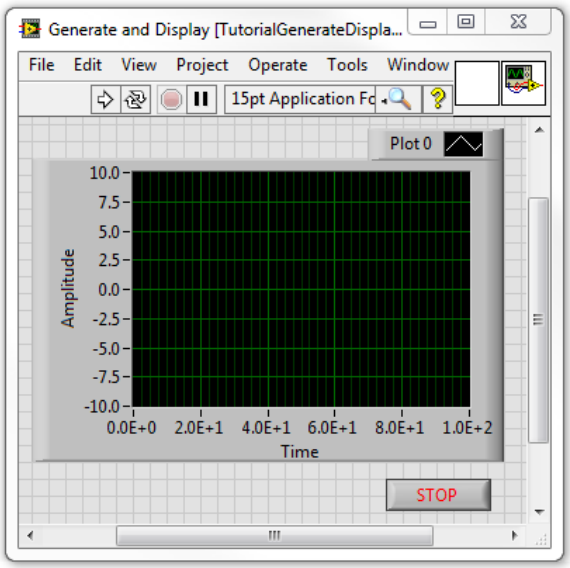
3. In the new window, expand **From Template**. Notice the different categories that appear, corresponding to the types of tasks which you can choose from. You can select **Blank VI** to start from scratch, and there are also Template VIs to use as a starting point for building your application. **Projects** and **Other Files** are more advanced components and will not be described in detail. To get more information on any of the listings in the New Dialog Box, click the **Help** button in the lower right corner of the window.

4. Navigate to and Select **VI» From Template» Simulated » Generate and Display** and click **OK**.

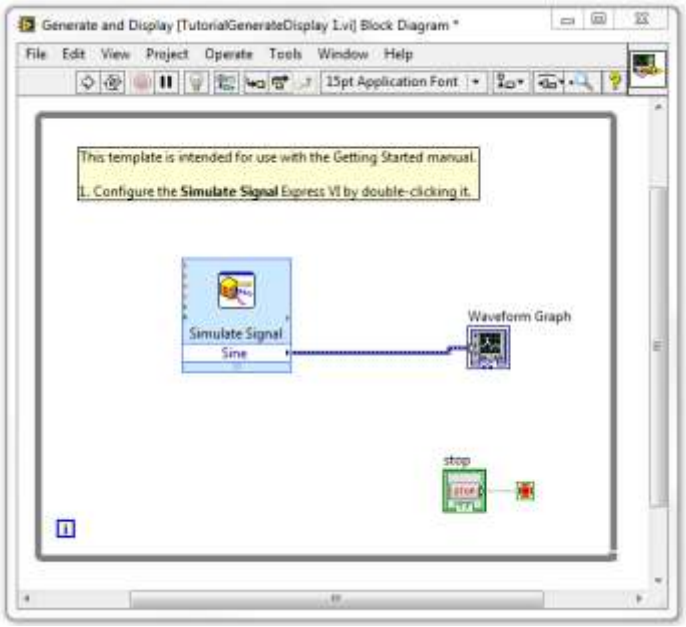


Two windows appear. The grey window is the **Front Panel**, and the white one is the **Block Diagram**. The Front Panel contains the parts of your VI used for the User Interface and presenting information, whereas the Block Diagram contains the code that controls the functionality of the VI. You can toggle between the two windows by selecting **Window» Show Block Diagram** or **Window» Show Front Panel**. You can also switch between the windows by pressing <**Ctrl+E**> on your keyboard.

5. Examine the Front Panel and Block Diagram of this Template VI. The Front Panel contains a Waveform Chart and a **STOP** button as shown in the following figure.



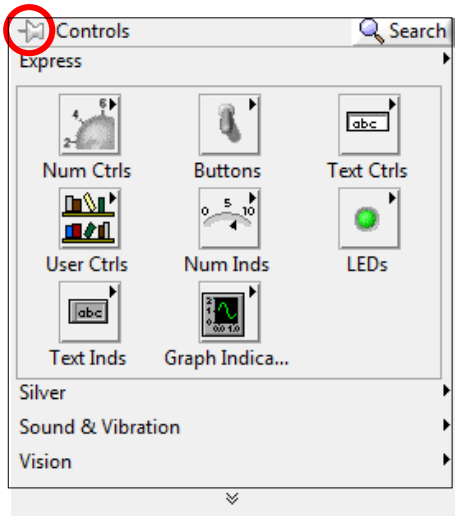
The Block Diagram contains a Simulate Signal VI, which is currently configured to simulate a sine wave and plot it to the chart.



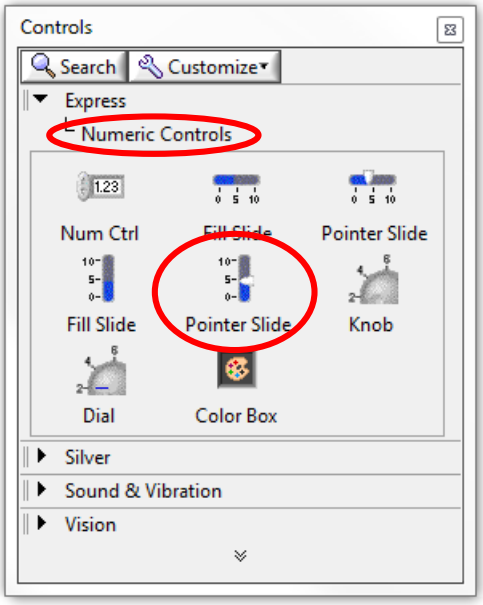
6. Switch back to the Front Panel by pressing <**Ctrl+E**>. Since the **Run** button (the white arrow at the top left corner) is solid, you can run this VI as it is. Click the **Run** button and examine the operation of the VI. When you are finished, click the **STOP** button **on the Front Panel** to stop running the VI.

   Note: As you will see later in the exercise, when the **Run** button in the upper left corner of both the Front Panel and the Block Diagram changes from a solid white arrow to a broken grey arrow, this new icon indicates that the VI is currently not executable. Pressing it will reveal the programmatic errors that are keeping it from executing.

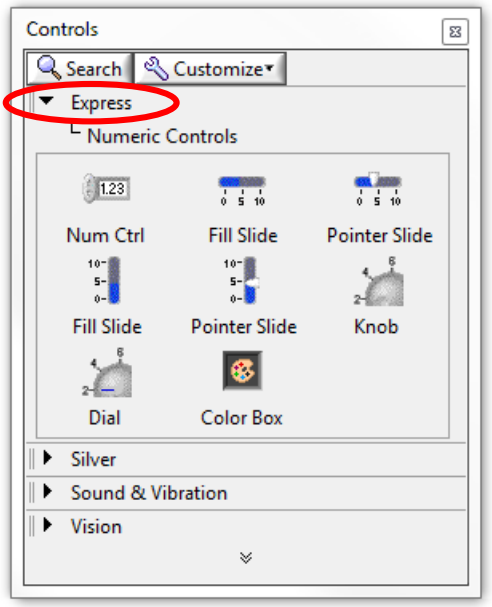
7. Now we can add some functionality to this basic VI. We will modify the VI to flash an alarm whenever the signal value is above a certain level. Open the **Controls** palette (if it is not open already) by **right-clicking** the **Front Panel** window. A small pin icon in the upper left corner of this palette appears. Click this pin to force the palette to remain on your screen.



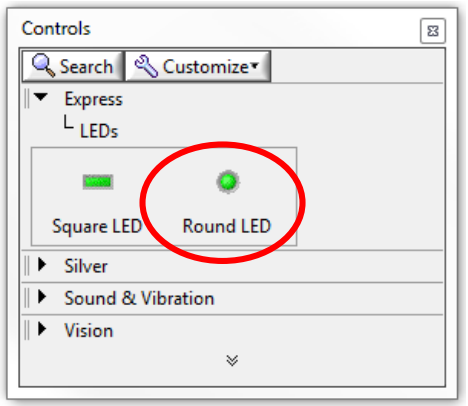
8. Navigate to the **Express** palette and click on the **Numeric Controls** sub-palette. Select a **Vertical Pointer Slide** to be placed on the Front Panel. To do this, click the **Vertical Pointer Slide** and drag it onto the Front Panel.



9. Click the Express menu item on the Controls palette to return to the Express Controls palette.



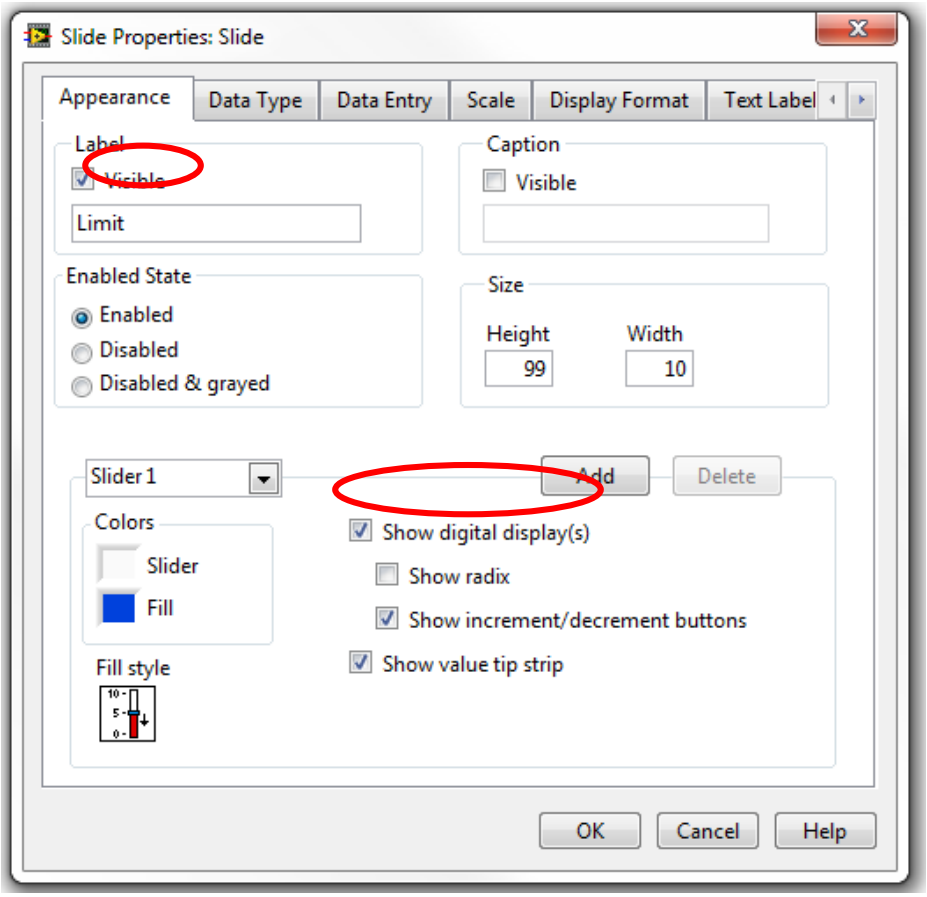
10. Click the **LEDs** sub-palette, and place a **Round LED** on the Front Panel.



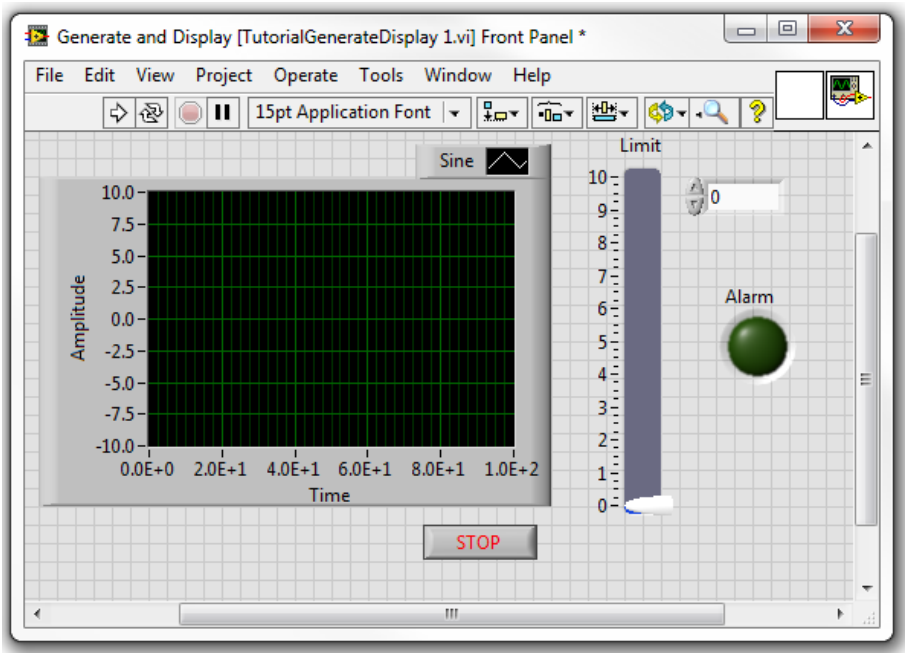
11. Right-click the **Vertical Pointer Slide** and select **Properties**. A property page will appear. Examine the different properties that you can modify. Make the following changes on the **Appearance** tab and click **OK** to apply the changes.

    Label: Limit

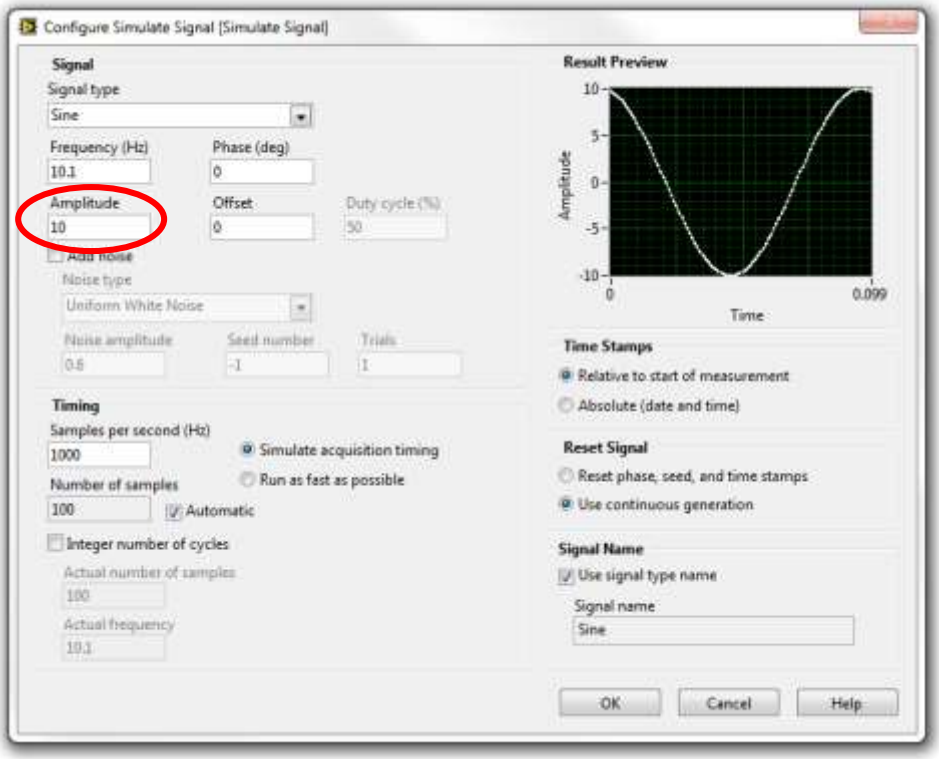
    Slider 1: Check Show digital display(s)



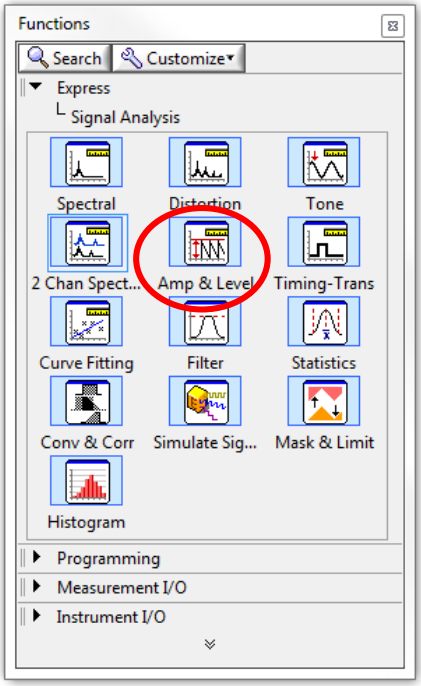
12. Right-click the Round LED labelled **Boolean** and select **Properties**. Examine the different properties that can be modified. On the **Appearance** tab, change the label from Boolean to **Alarm**. Click **OK** to apply your change. Move the objects on the Front Panel so it resembles the following.



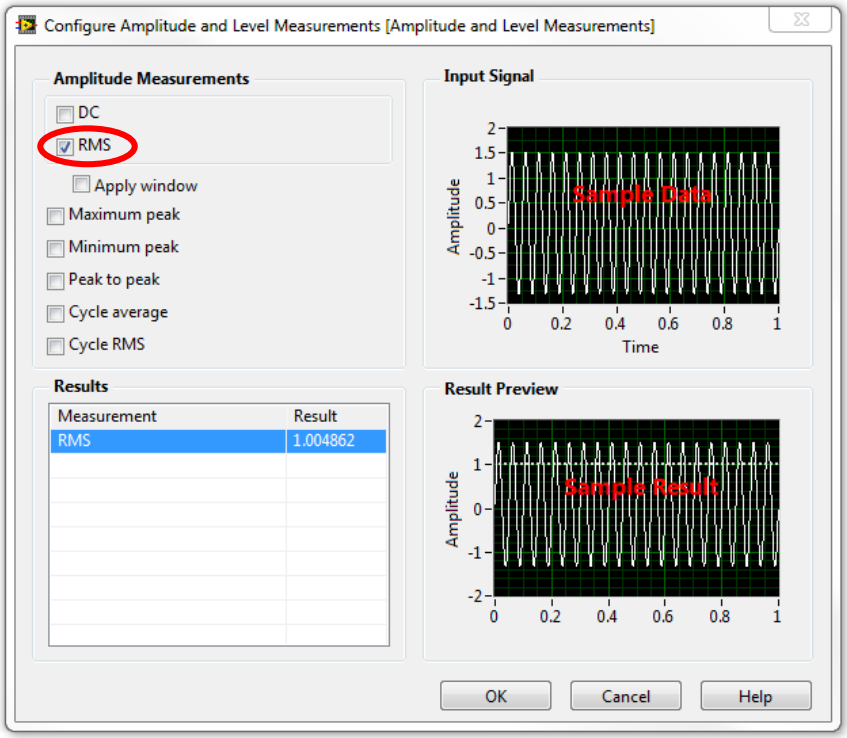
13. Switch to the Block Diagram by pressing <**Ctrl+E**>. Double-click the **Simulate Signal** Express VI to bring up its properties window. Examine the different properties you can modify. Change the **Amplitude** of the signal to **10**. Click **OK** to apply this change and to close the properties window.



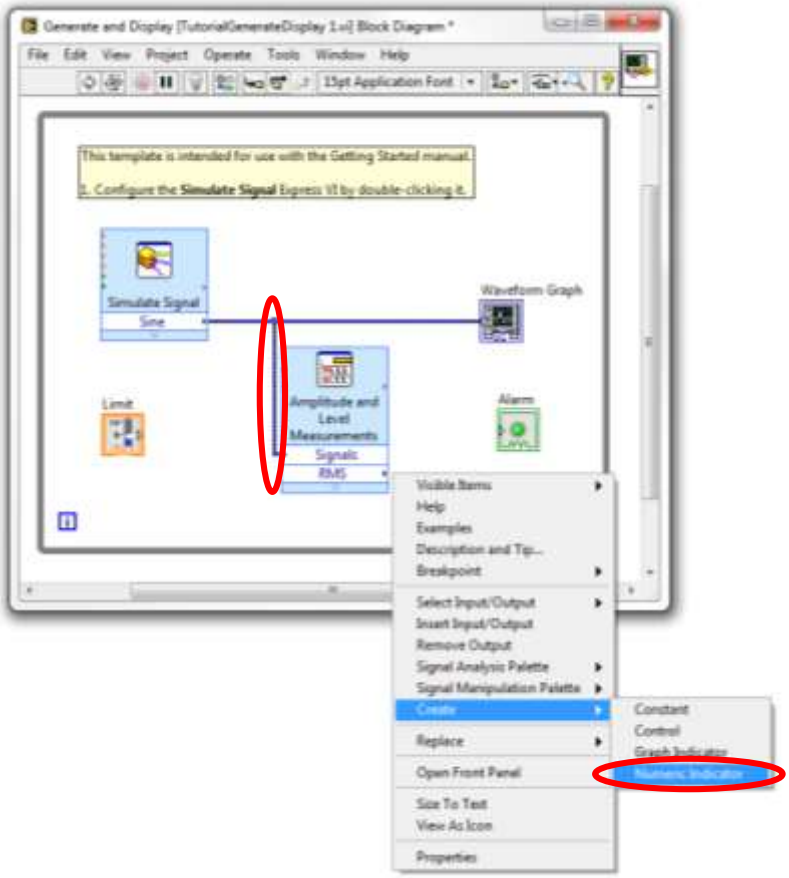
14. Bring up the **Functions** palette by **right-clicking** on the **Block Diagram**. Select **Express» Signal Analysis** and place the **Amp & Level** Express VI on the Block Diagram by dragging and dropping as before.



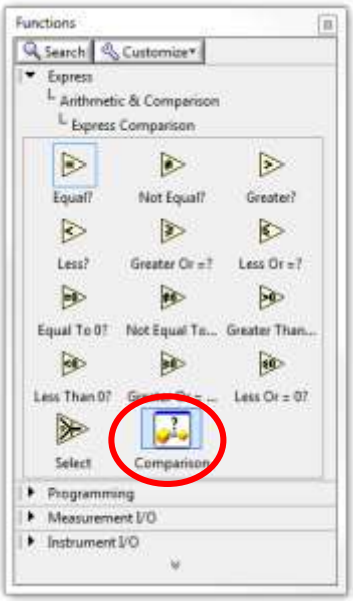
15. When you place an Express VI on the Block Diagram a dialog box appears so that you can configure the function as per your needs. For this function select **RMS** as shown below.



16. Wire the **output** of the **Simulate Signal** VI to the **Signals** input on the **Amplitude and Level Measurements** VI. Right-click on the **RMS** output and select **Create» Numeric Indicator** from the list.



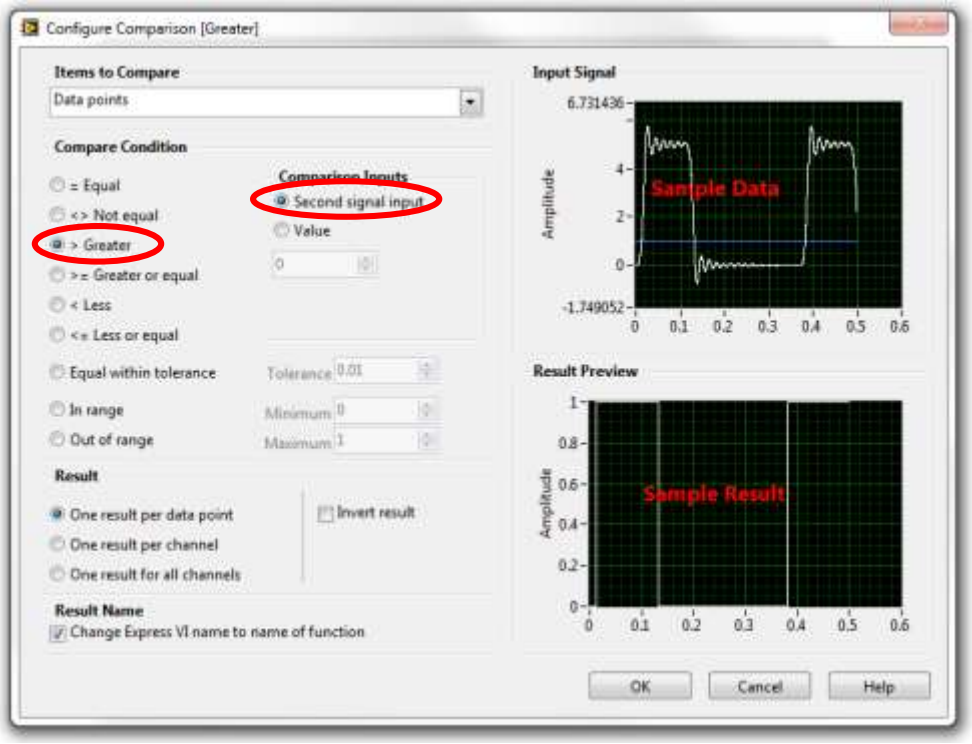
17. Bring up the functions palette by right-clicking the Block Diagram. Select **Arithmetic & Comparison» Comparison** and place the **Comparison** Express VI on the diagram



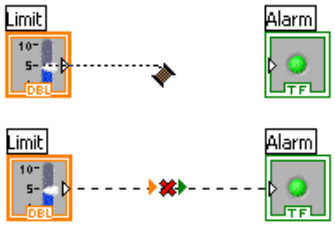
When you place the Comparison Express VI on the Block Diagram, a dialog box appears that lets you configure what type of comparison you will be doing. Make the following selections, then click **OK** to apply these changes and to close the dialog box.

Compare Condition: Greater

Comparison Inputs: Second signal input



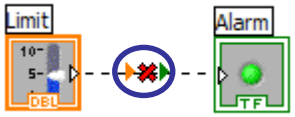
18. You can connect Controls, Functions, and Indicators on the Block Diagram by pointing to an object and clicking on its terminal when the cursor changes to a spool of wire. You can then move the cursor to the terminal of the object you want to connect it to and click again. Following this, connect the **Limit** control to the **Alarm** indicator.



Note: The **Run** button in the upper left corner of both the Front Panel and the Block Diagram has changed from a solid white arrow, to a broken grey arrow. This new icon indicates that the VI is currently not executable. If you click the **Run** button when it is solid and white, it runs the VI. Clicking it when it is broken and grey brings up a dialog box that will help you debug the VI.

19. Click the **Run** button now. The resulting dialog box shows that, in this case, the error results from connecting terminals of two different types. Since the **Limit** control is a Numeric type and the **Alarm** indicator is a Boolean type, we cannot wire these two terminals together. Highlight the error by clicking it, and then click **Show Error**. LabVIEW will highlight the location of the error.

20. Notice that the wire between **Limit** and **Alarm** is dashed and a red ✕ is displayed on it.

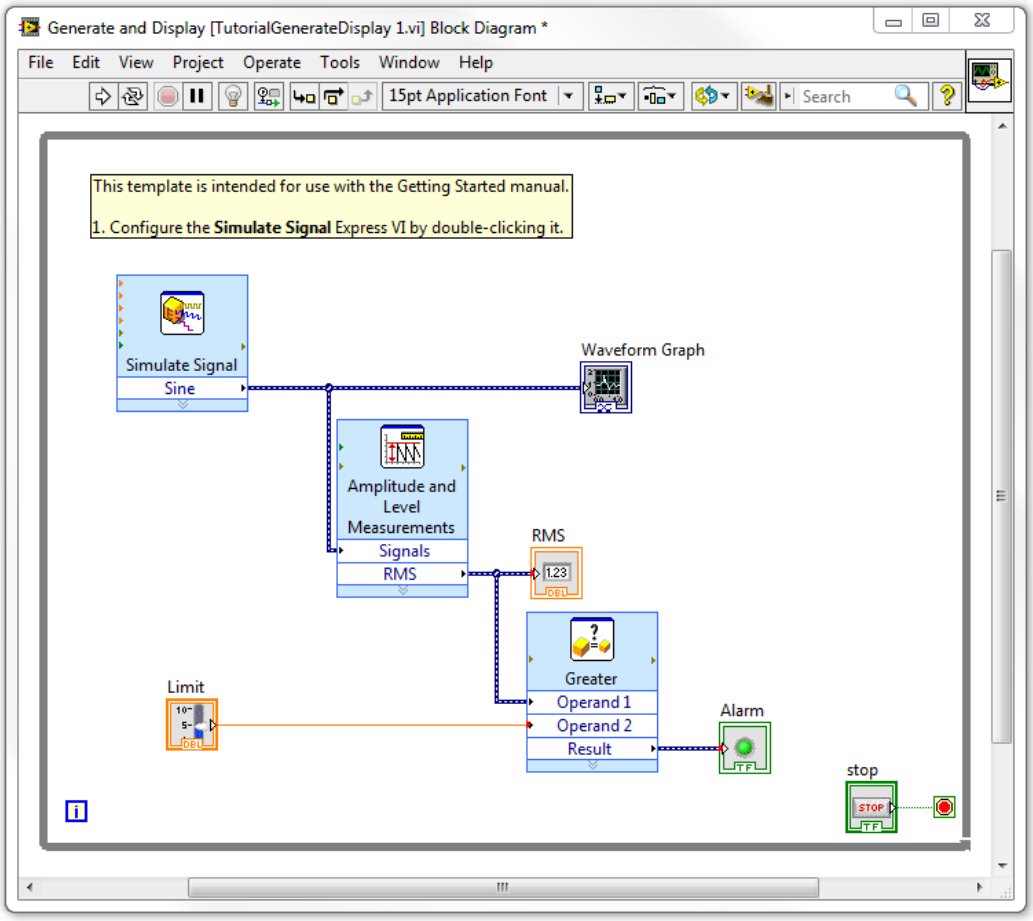


To delete this broken wire, press <**Ctrl+B**>. This keyboard shortcut removes **all** broken wires from the Block Diagram.

21. Make your Block Diagram resemble the following image by completing these steps:

    a. Wire the **Limit** control to the **Operand 2** input of the **Comparison** function.
    b. Connect the wire between the **Amplitude and level Measurements** block and the **RMS Indicator** to the **Operand 1** input of the **Comparison** block.
    c. Wire the **Result** output of the **Comparison** block to the **Alarm** indicator.

Your Block Diagram should now resemble the following image.



22. Switch to the Front Panel by pressing <**Ctrl+E**>.

23. Save the VI in the **C:\Users\Student\Desktop\Hands On Taster** folder by using the File menu and name it **Exercise1.vi**

    Note: Be sure to save this VI, as you will be using it later in the seminar.

24. **Run** the VI. While the VI is running you can change the **Limit** value. Also notice that when a data point received from the **Simulate Signal VI** is greater than the **Limit** value, the **Alarm** indicator lights up.

    While the VI is still running, switch to the Block Diagram by pressing <**Ctrl+E**>. Enable **Highlight Execution** by clicking on the light bulb on the tool bar. This will allow you to see the flow of data through your program.

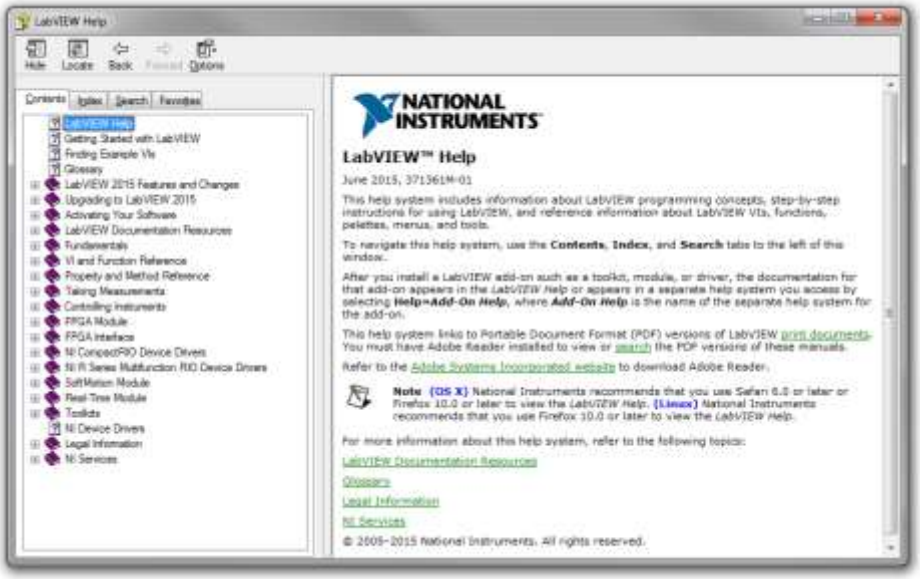
25. When you are finished, stop the VI by clicking the **STOP** button on the Front Panel.

- End of Exercise -

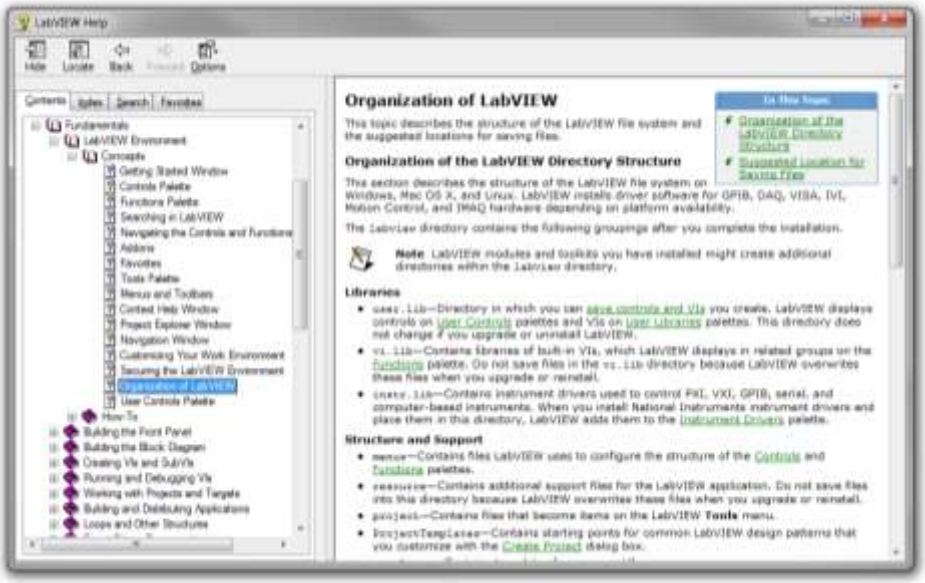
Exercise 2 (Optional): The LabVIEW Help System

The LabVIEW help system is a great place to learn about LabVIEW and to go when you have questions. This exercise will introduce you to the rich source of information that is available for you to take advantage of.

1. Go back to the VI you just created, and press <F1> on the keyboard to start the help system



2. Expand **Fundamentals» LabVIEW Environment** and explore the information available here, click around and get a feel for how it is organised.



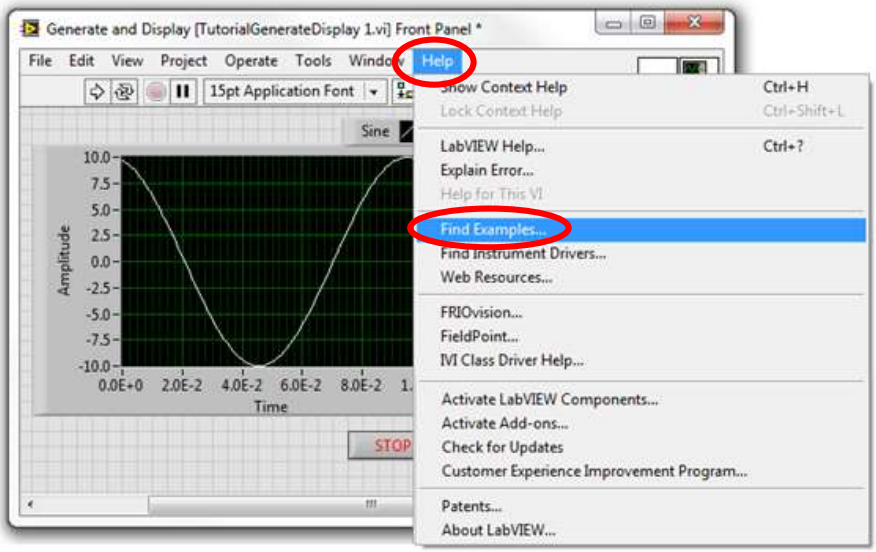
3. Take a few minutes to explore other topics in the help system.

4. Click on the **Search** tab and try searching for analysis functions with features you might need in your work application.

Exercise 3 (Optional): LabVIEW Example Finder

As you learned earlier, LabVIEW has a comprehensive help system. LabVIEW also includes an extensive set of examples that in many cases are the building blocks for your applications. This exercise will introduce you to what examples are available.

1. Go back to the VI you created, and from the **Help** menu select **Find Examples**.



2. **Switch** to the **Search** tab, type in “express” and hit Enter. From the results, select and double-click on **Express VI - Filter.vi**.
3. **Run** the VI and explore the Block Diagram. This VI shows some results from using the Filter Express VI.
4. Take a few minutes to look around at the other example programs available, search on terms that you are familiar with, switch back to the **Browse** tab and navigate through the different example categories, and try out as many as there is time for.

- End of Exercise -

LabVIEW DAQ

Introduction to LabVIEW and Data Acquisition

Exercise 1a: Data Acquisition with NI LabVIEW

Objective

Learn about data acquisition in LabVIEW

Goals

When you have completed this exercise, you will:

- Know how to configure a data acquisition device
- Know how to use the DAQ Assistant and take measurements with NI LabVIEW

Hardware set-up

1. Make sure that the NI CompactDAQ chassis is powered on.
2. Connect the chassis to the PC/laptop using the USB cable.
3. The NI-DAQmx driver set that is installed will automatically detect the chassis and bring up the following window. If this exercise has been attempted on this PC/laptop before, then the dialog may not appear. If the window does not appear, open **Measurement & Automation Explorer** (**MAX**) via **Start » All Programs » NI MAX**.

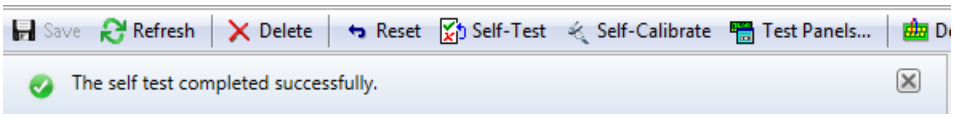
4. Click on **Configure and Test This Device** to open **Measurement & Automation Explorer** (**MAX**).

   Note: MAX is a configuration utility for all National Instruments hardware.

Configure Hardware

5. Once **MAX** is launched, you will find that the **Devices and Interfaces** section under **My System** shows all the National Instruments devices installed and configured on your station. By default, the NI CompactDAQ chassis shows up with the name “cDAQ1”.
6. This section of MAX also shows the installed modules as well as empty slots in the CompactDAQ chassis.
7. **Right-click on the chassis** and click on **Self-Test**.

The device passes the self test, which means it is configured properly and ready to be used in your LabVIEW application.



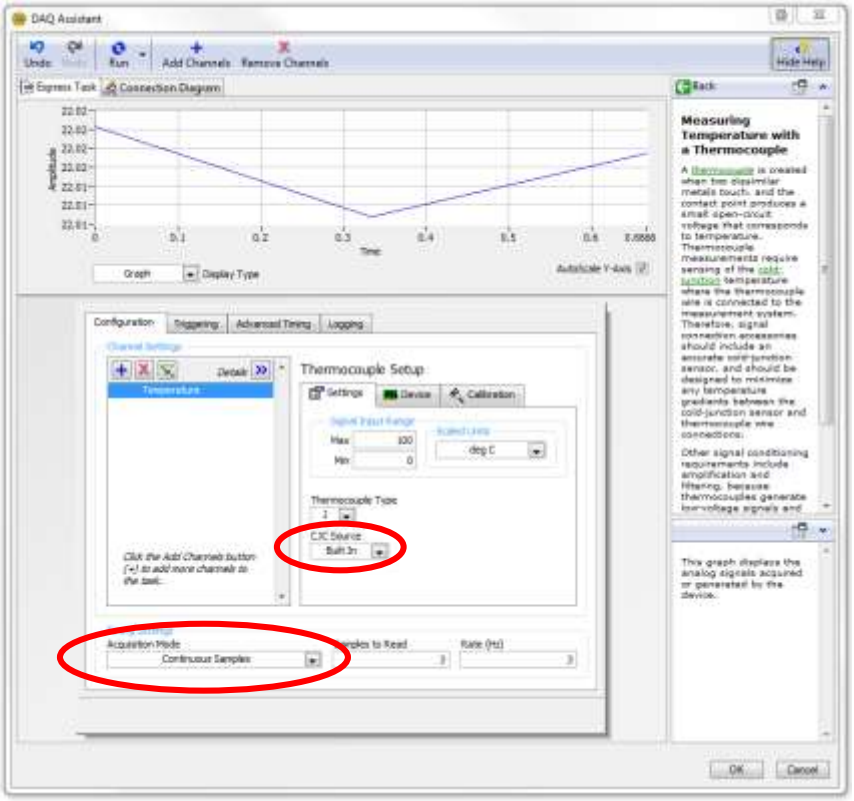
LabVIEW

8. Launch **LabVIEW** and press <**Ctrl+N**> to create a blank VI.
9. Press <**Ctrl+T**> to tile Front Panel and Block Diagram windows.
10. Pull up the **Functions Palette** by right-clicking on the white space of the LabVIEW Block Diagram window.
11. Move your mouse over the **Express» Input** palette, and click the **DAQ Assistant** Express VI. Click again on the white space of the LabVIEW Block Diagram to place down the DAQ Assistant.

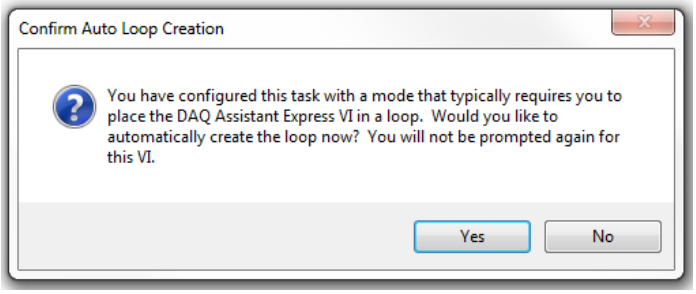
12. The **Create New Express Task…** window appears.

13. To configure a temperature measurement application with a thermocouple, click on **Acquire Signals» Analog Input» Temperature» Thermocouple**. Click the **+** sign next to the **cDAQ1Mod1 (NI 9211)** entry, highlight channel **ai0**, and click **Finish**. This adds a physical channel to your measurement task.

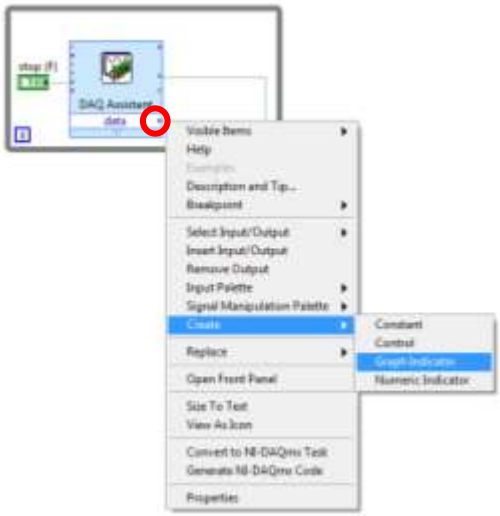
14. Change the **CJC source** to **Built In** if it is not already set, and the **Acquisition Mode** to **Continuous**. Click **Run**. You will see the temperature readings from the thermocouple in the test panel window.



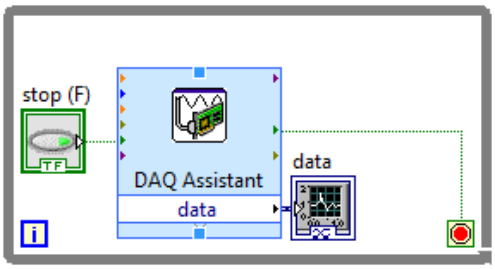
15. Click **OK** to close the Express block configuration window and return to the LabVIEW Block Diagram.
16. Notice that LabVIEW automatically creates the code for you for this measurement task. Click **Yes** to automatically create a While Loop.



17. Right-click the **data** terminal output of the DAQ Assistant Express VI (the blue output arrow on the right side) and select **Create» Graph Indicator**.

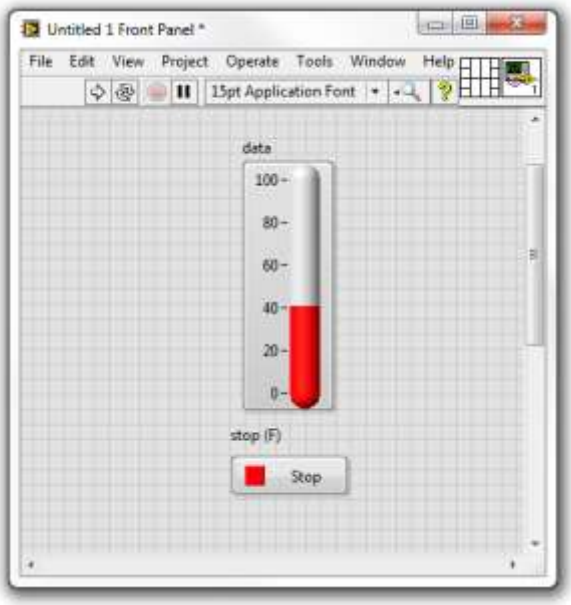


18. Notice that a graph indicator is placed on the Front Panel.
19. Your Block Diagram should now look like the figure below. The while loop automatically adds a stop button to your Front Panel that allows you to stop the execution of your code.

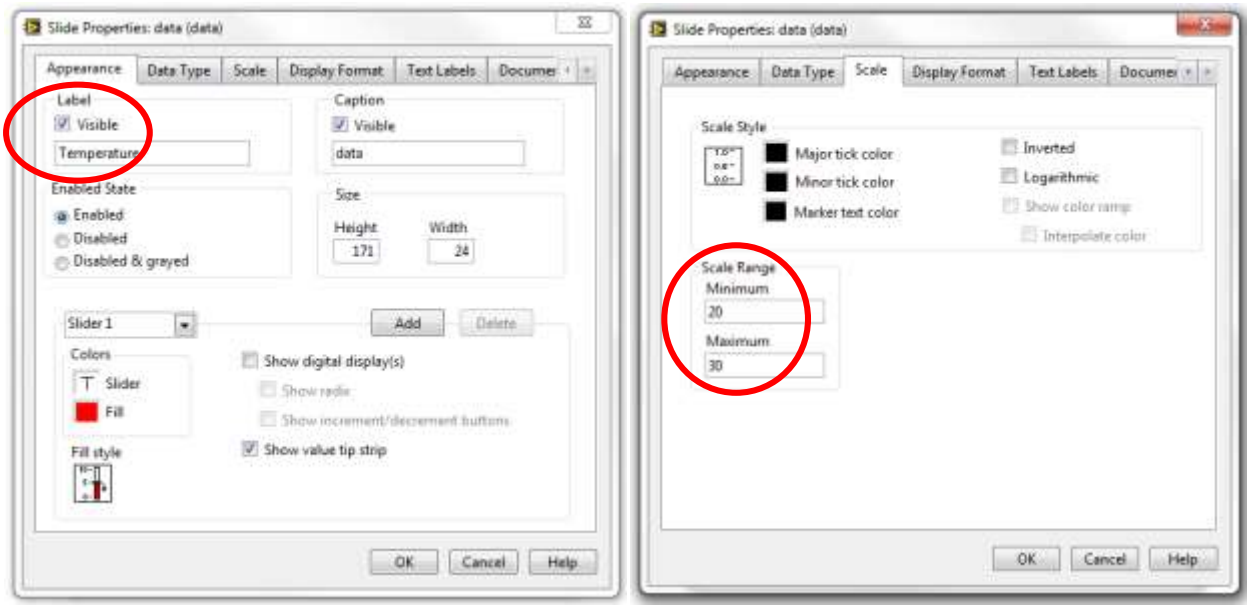


20. Rather than displaying our data in a graph indicator, we want to display our temperature readings in a thermometer indicator. To do this, go to the Front Panel by pressing <**Ctrl+E**>, right-click on the graph indicator and select **Replace**. The Controls palette will appear. Click on the two downwards pointing arrows to display more pallets, Select **Silver» Numeric» Thermometer**. The thermometer indicator should now appear instead of the graph indicator.

21. Also, right-click on the **stop (F)** button and select **Replace.** The Controls palette will appear. Select **Buttons » Stop Button**.

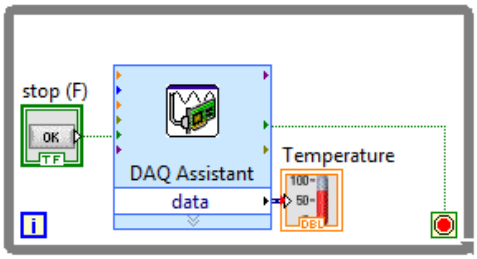


22. Modify the thermometer indicator by right-clicking it and selecting **Properties**. On the Appearance Tab, change the **Label** to **Temperature**. On the Scale tab, change the **Minimum** to **20** and the **Maximum** to **30**.

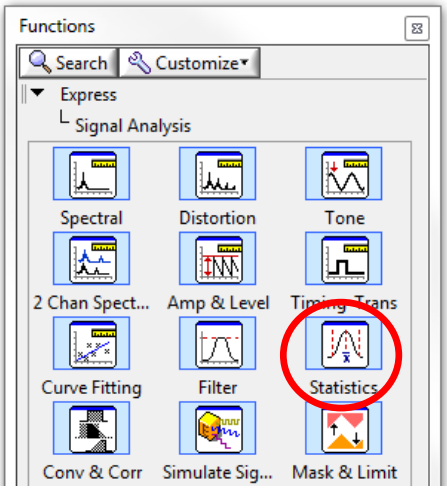


Click **OK** when you are finished.

23. Switch to the Block Diagram and expand the While Loop. Your Block Diagram should now resemble the following illustration.



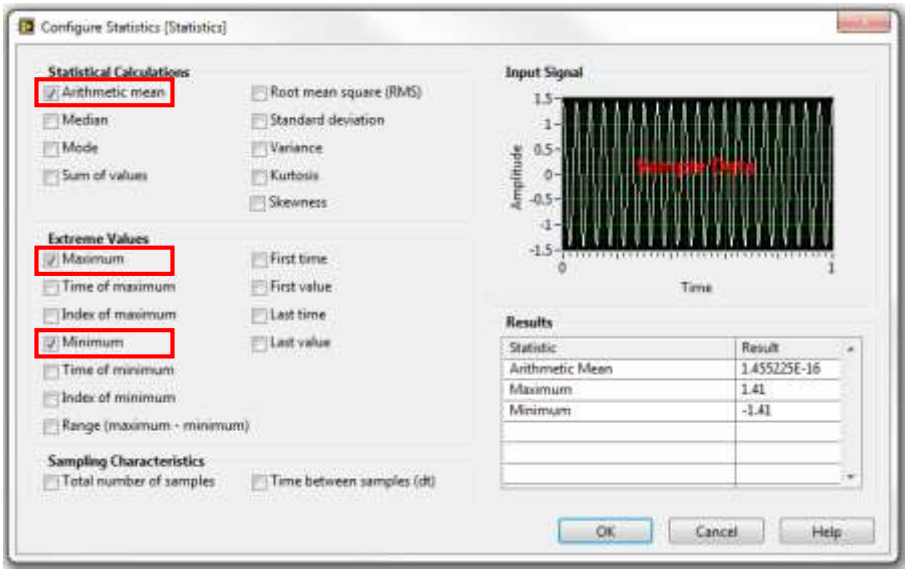
24. To perform analysis on your data, select the **Express» Signal Analysis» Statistics** Express VI and place it on your Block Diagram.



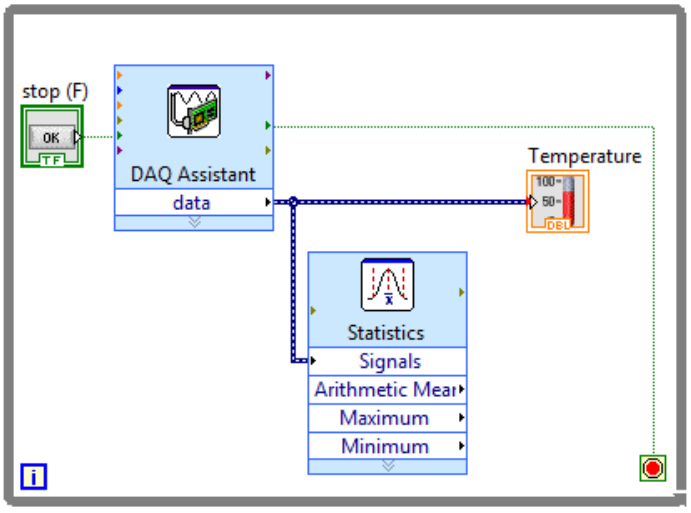
25. A properties window will appear. Make the following selections and click **OK**.

    Statistical Calculations: Arithmetic Mean

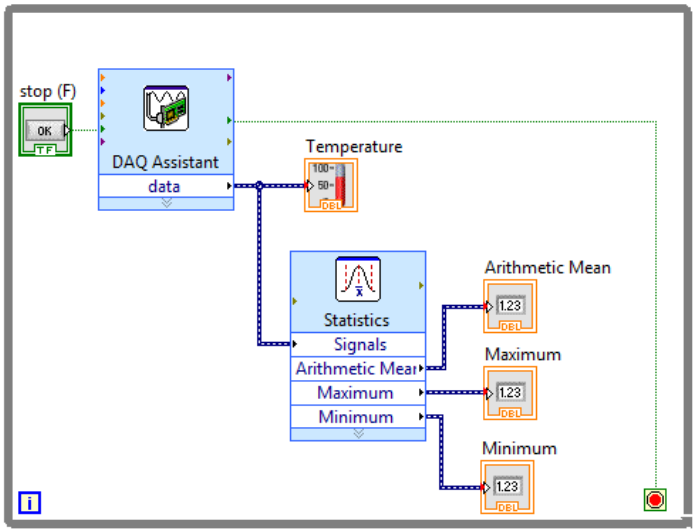
    Extreme Values: Maximum, Minimum



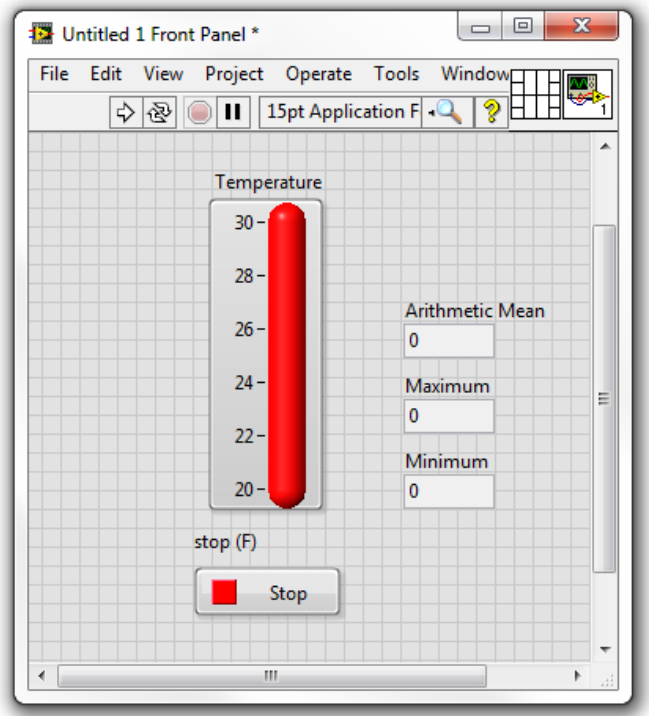
26. Connect the data output of the DAQ Assistant VI to the Signals input of the Statistics VI.



27. **Right-click** the **Arithmetic Mean** output of the **Statistics** VI and select **Create» Numeric Indicator**. This will create an indicator on the Front Panel to display the mean. Repeat this for both the Maximum and Minimum outputs of the Statistics VI. Your Block Diagram should now resemble the following image.



28. Switch to the Front Panel and rearrange your controls and indicators to resemble the following User Interface.



29. Save the VI in the **C:\Users\Student\Desktop\Hands On Taster** folder by using the File menu and name it **Exercise1a.vi**
30. **Run** the VI. Hold the thermocouple between your fingers to raise the temperature. Notice the change in temperature on the thermometer. If you are not seeing enough of a temperature fluctuation, stop the VI and decrease the range on your thermometer indicator.
31. Click the **STOP** button on the Front Panel when you are finished.
32. Close the VI.

- End of Exercise -

Exercise 1b: File I/O in LabVIEW

Objective

How to log data to a file using LabVIEW.

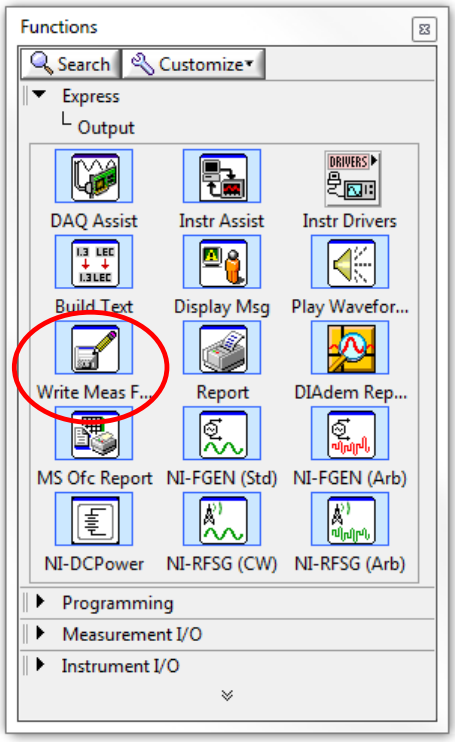
Goals

When you have completed this exercise, you will:

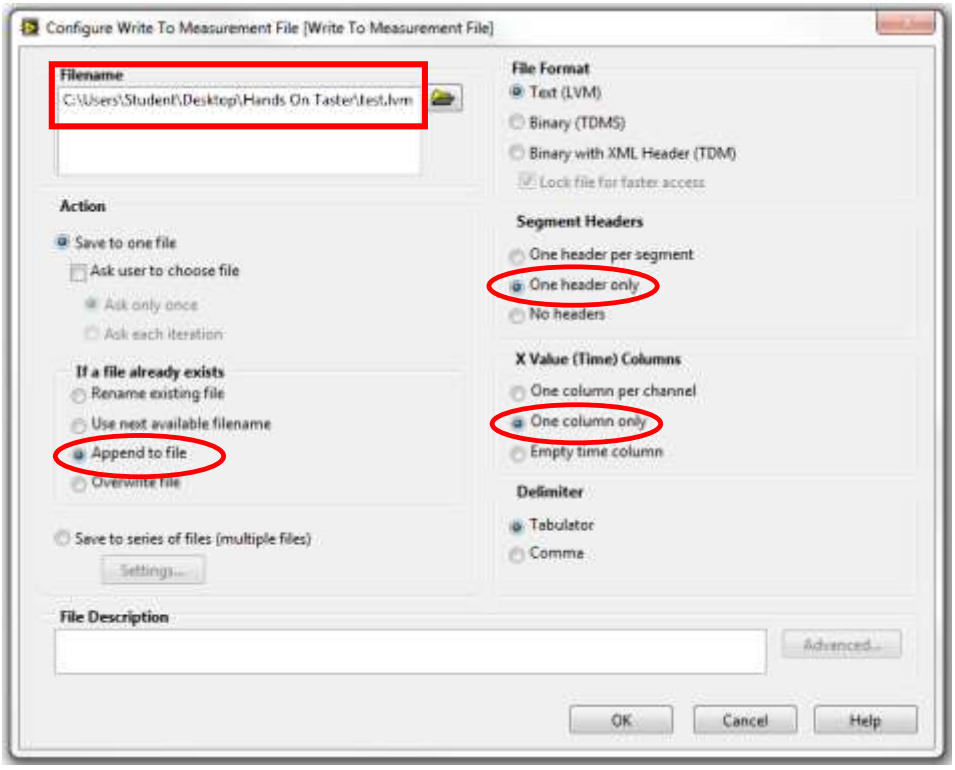
- Know how to use the Write to Measurement File Express VI and how to log data to a file using LabVIEW.

Implementation

1. Open the VI from Exercise 1a.
2. Right-click on the Block Diagram and select **Express» Output» Write to Measurement File** and place it inside the While Loop.



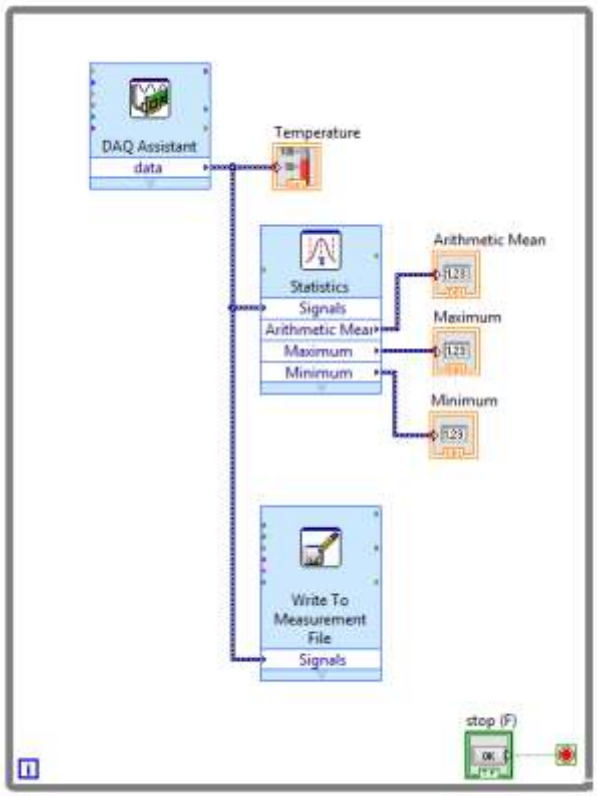
3. A configuration window will appear. Set the following parameters and click **OK.**



4. Wire the output of the **DAQ Assistant Express VI** to the input of the **Write to Measurement File Express VI**.

5. **Delete the wire** connecting the DAQ Assistant to the conditional stop terminal of the while loop. **Delete the wire** connecting the Stop button to the DAQ assistant. **Wire** the **stop button** to the conditional stop terminal of the while loop.

6. Your Block Diagram should now resemble the following.



7. Save the VI in the **C:\Users\Student\Desktop\Hands-on Tasters** folder by using the File menu and name it **Exercise1b.vi**
8. **Run** the VI momentarily and then press the **STOP** button on the **Front Panel**. Do not use Abort.
9. Your file will be created in the folder specified (**C:\Users\Student\Desktop\Hands-on Tasters**).
10. Open the file (**test.lvm**) using Microsoft Office Excel or Notepad. Review the header and temperature data saved in the file.
11. Close the data file and the LabVIEW VI.

- End of Exercise -

Exercise 1c (Optional): Automatic Code Generation in LabVIEW

Objective

Create NI-DAQmx code from the DAQ Assistant.

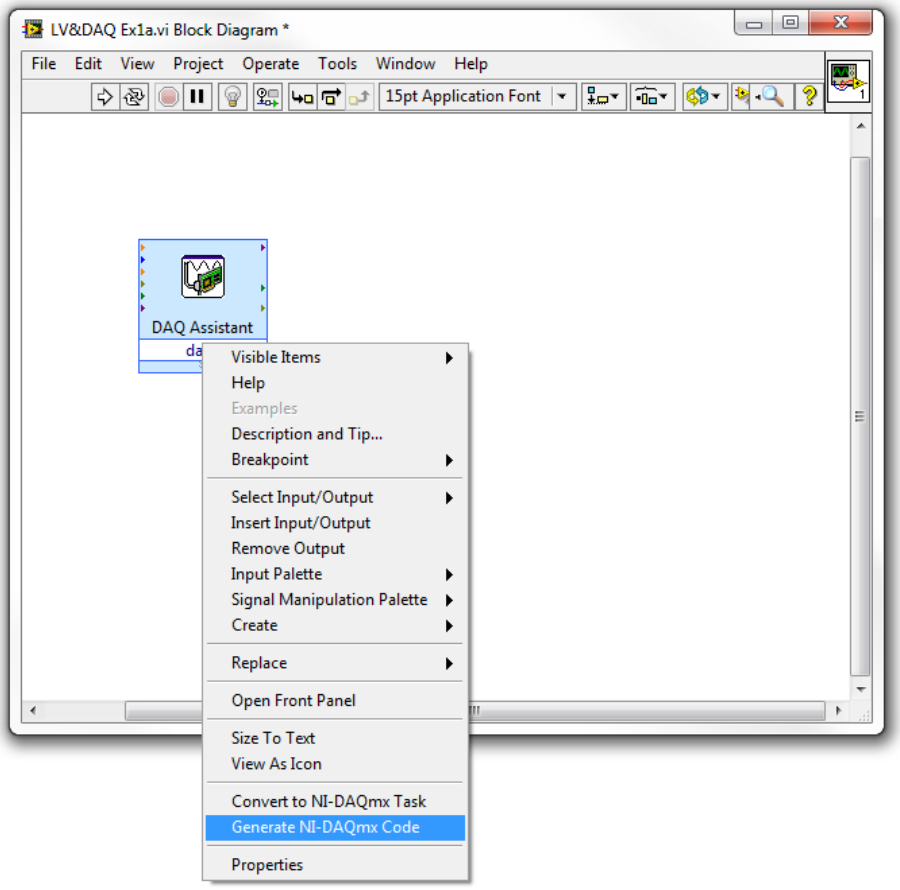
Goals

When you have completed this exercise, you will:

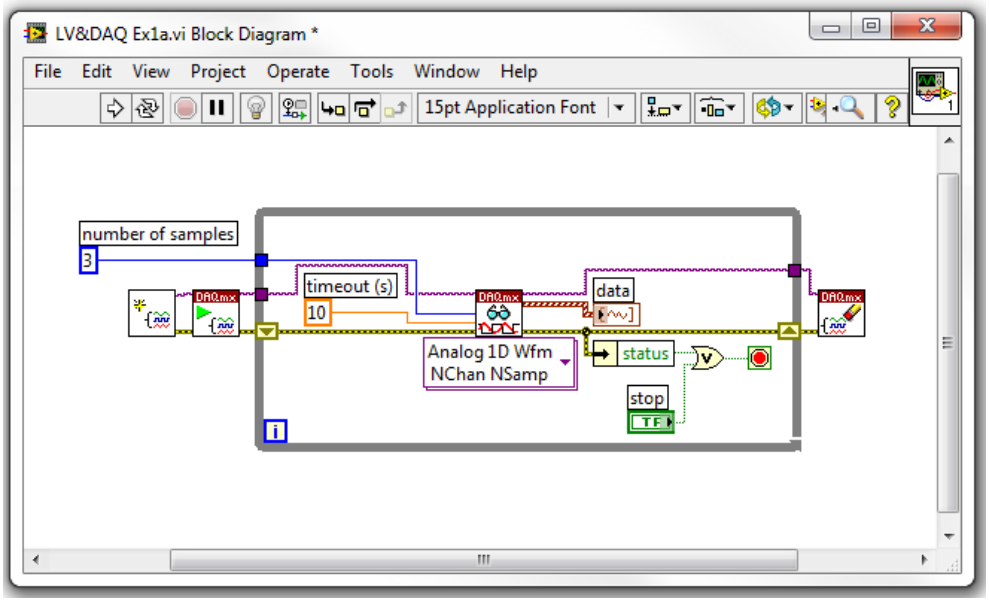
- Know how to generate code automatically from the DAQ Assistant.
- Have experience using some of the NI-DAQmx VIs.

Implementation

1. Open the VI from Exercise 1a.
2. Delete all the functions and terminals from the Block Diagram **except** the DAQ Assistant. Delete all the wires as well, or use <**Ctrl+B**> to clear them.
3. Right-click on the DAQ Assistant and select **Generate NI-DAQmx Code.**

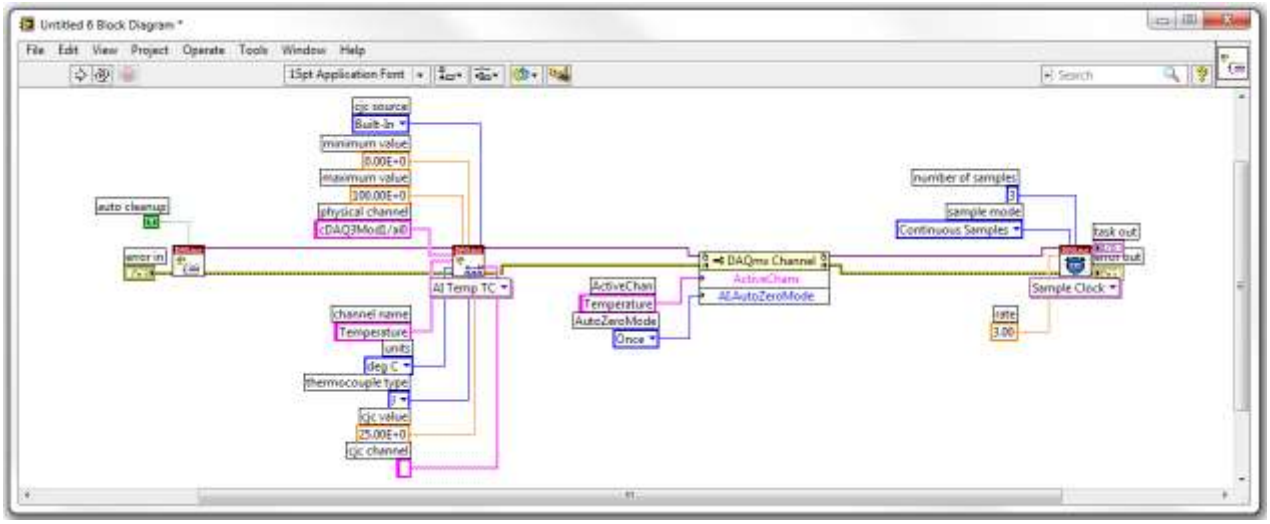


4. NI-DAQmx will now generate LabVIEW code using the NI-DAQmx API. Your Block Diagram should now resemble this following picture.



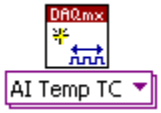
5. Double-click on the configuration VI (shown below) that was created for you on your Block Diagram.

6. The Block Diagram of the configuration VI should appear as shown below. This is an example of how you can use the DAQmx VIs if you need to create customised DAQ code that includes features beyond those offered by the DAQ Assistant.



In this Block Diagram, you see three SubVIs:

DAQmx Create Task.vi



DAQmx Create Channel.vi

DAQmx Timing.vi

To learn about each VI, hover your mouse over each one and press <**Ctrl+H**>. This will bring up the **Context Help** which explains the parameters and functionalities of each VI.

7. **Close** the VI and do not save any changes.

- End of Exercise -

Exercise 2: Controlling Program Execution

If you are up for challenge, try to build the following LabVIEW application without using the instructions; if you feel you need more guidance, skip down to the Step-by-step section for a complete set of instructions.

Challenge

Using a While Loop, Case Structure, toggle switch and the Simulate Signal VI, create a simple application that charts a sine or triangle wave depending on the toggle switch position.

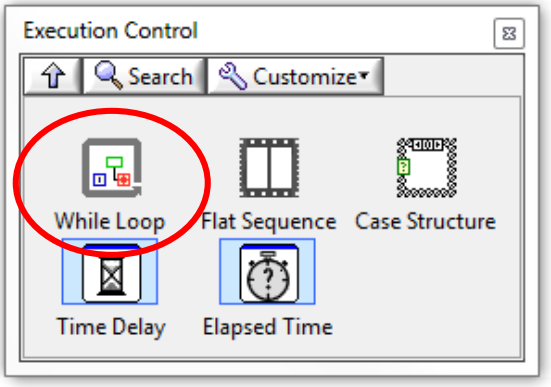
When you are done, save your VI as **Exercise2-Program Execution.vi** in **C:\Seminars\Hands-on Tasters**. You will use it in the next exercise.

If you are *really* up for a challenge, add logic to the VI so that it will stop either when you press the stop button or when the loop iterations have exceeded 10,000.

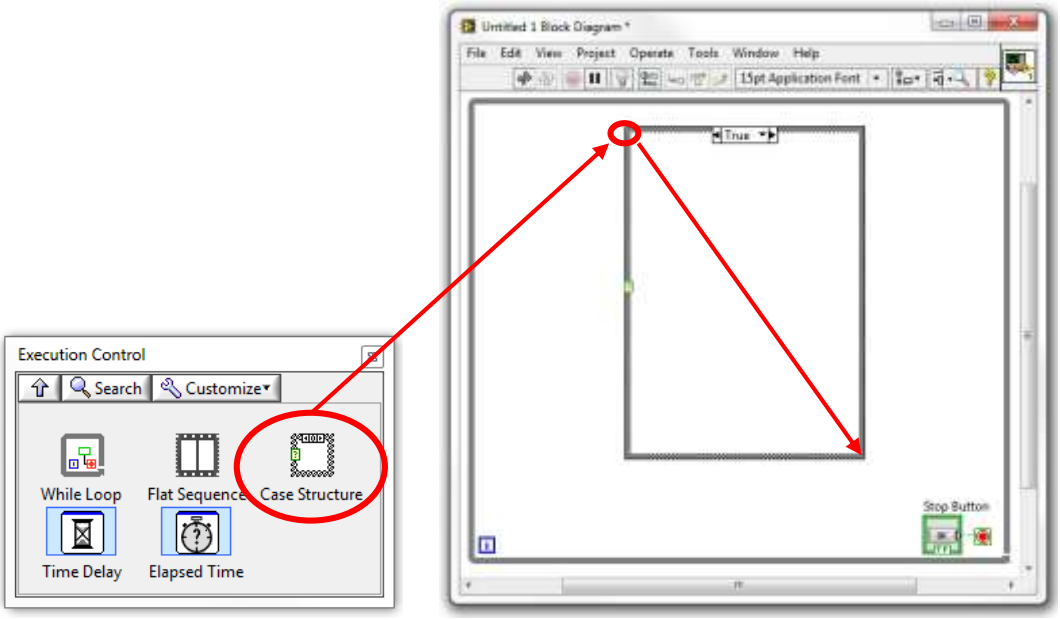
Step By Step Instructions

In this exercise, you will create a LabVIEW VI that will output a triangle or sine wave to a Front Panel graph depending on the state of a toggle switch. You will use a **Case Structure** to handle the logic of which signal is output and a While Loop to keep the application running until a Stop button is pressed.

1. Let's start by placing a While Loop on the Block Diagram. Draw it large enough to accommodate the rest of the code you will shortly be placing inside it. You can locate the While Loop on the **Express» Execution Control** palette as shown below.

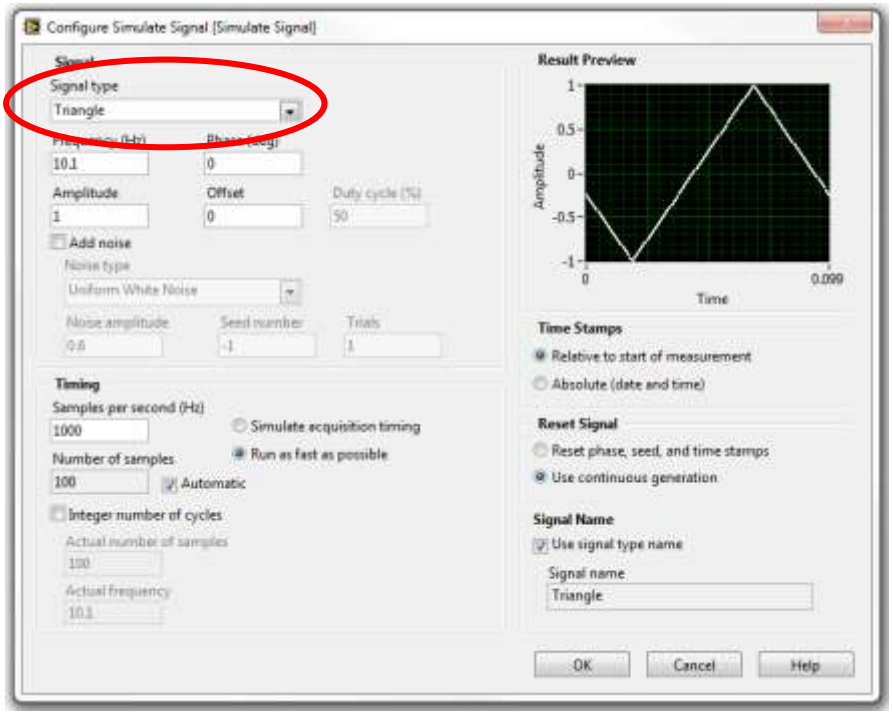


2. Next select the **Case Structure** from the Execution Control palette and place it **inside** the While Loop as shown below.

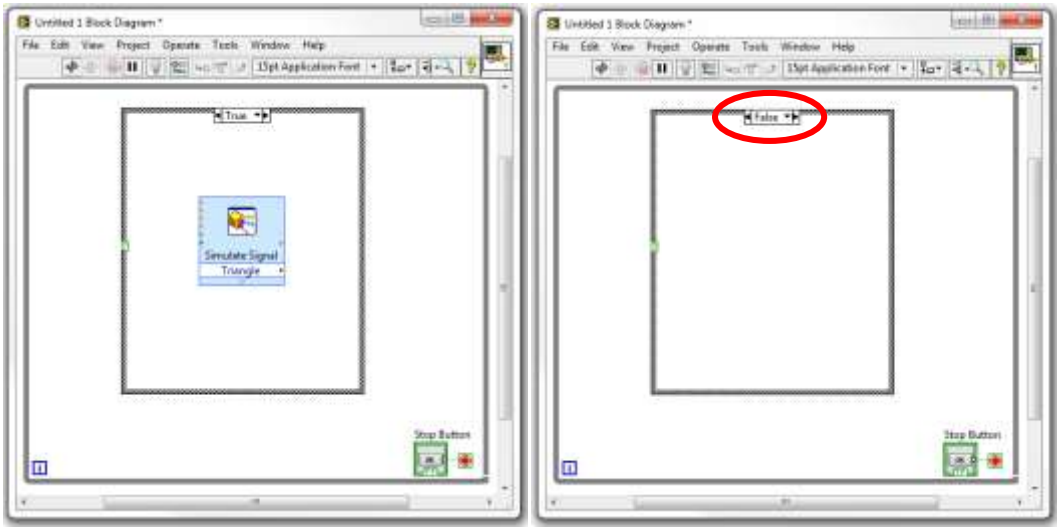


3. Next we will add two Simulate Signal VIs to each case in the Case Structure. Remember, one will output a **triangle** wave, the other a **sine** wave. From the **Express» Input** palette, select **Simulate Signal Express VI**, and place it inside the **Case Structure**.

4. Configure the first **Simulate Signal VI** to produce a triangle wave by selecting the Signal type as shown below.

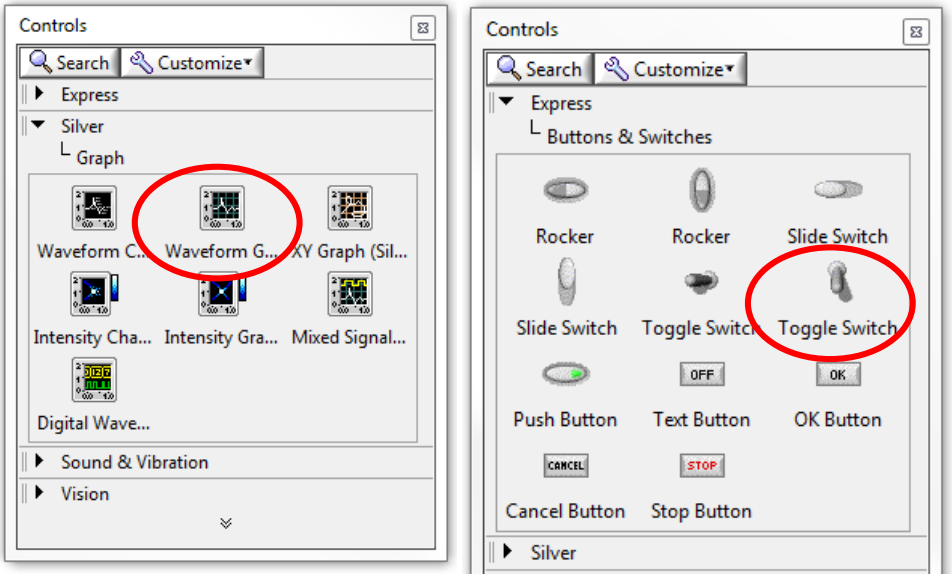


5. Your Block Diagram should now look like the following. Next, switch the Case Structure from True to False, as shown below.

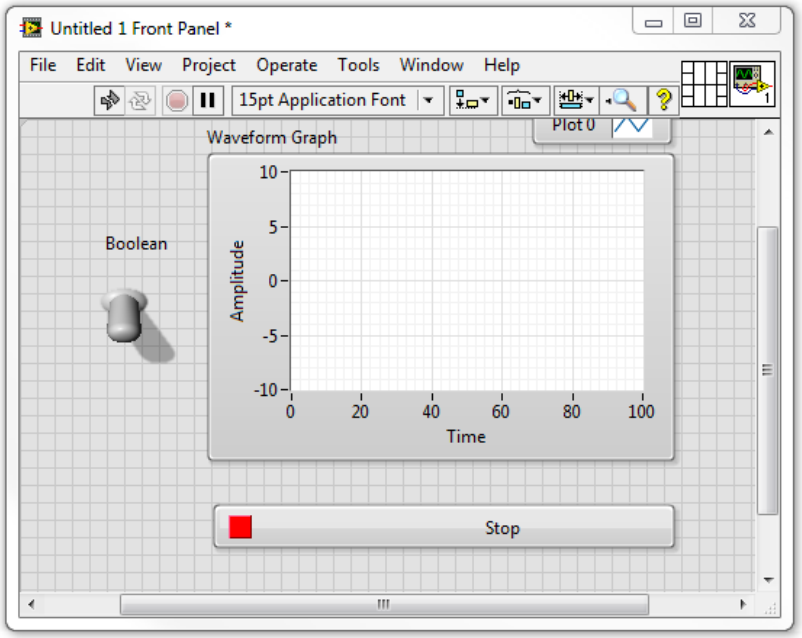


6. Repeat steps 3 and 4, except this time leave the default settings for the **Simulate Signal** VI so that it will generate a **sine wave**.

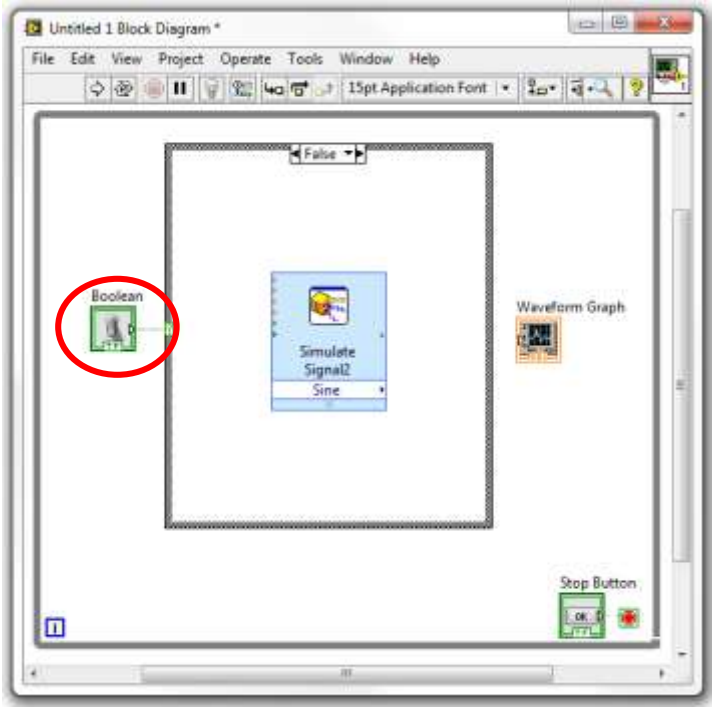
7. Switch to your Front Panel and place a Waveform Graph from the **Graph** palette as shown below. Also add a vertical toggle switch from the **Buttons & Switches** palette to the Front Panel.



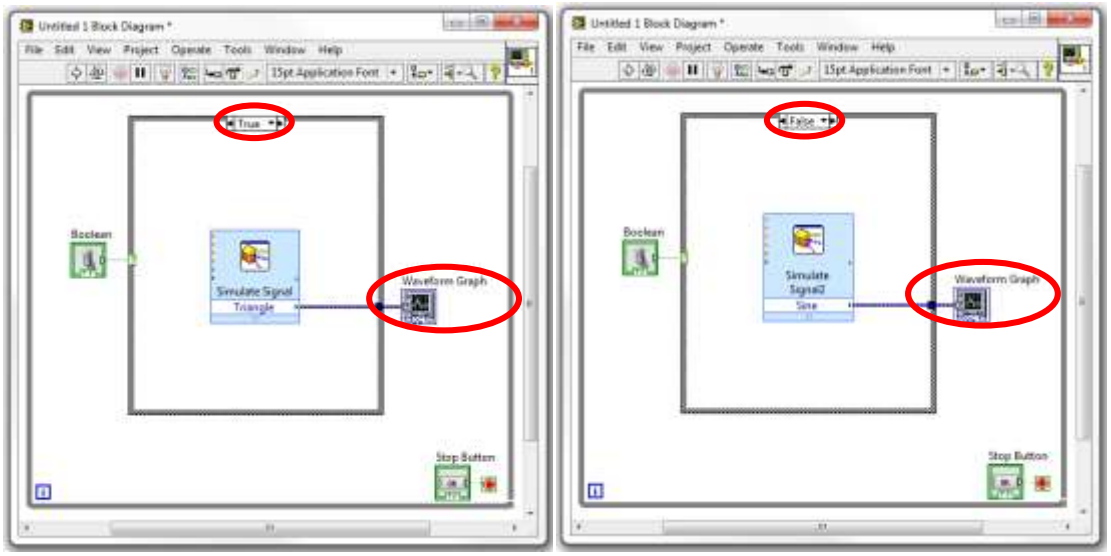
8. Your Front Panel should resemble the following.



9. Switch back to the Block Diagram and wire the toggle switch to the input of the Case Structure as shown below.



10. Wire the outputs from the **Simulate Signal VIs** to the graph. Note that you will have to wire through the wall of Case Structure and don't forget to wire through both cases, True and False.

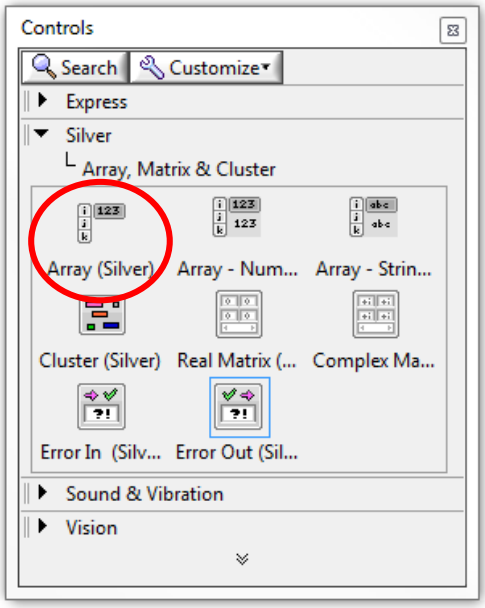


11. You are ready to **Run** your VI and test it by switching the toggle switch back and forth. You should see the graph switch between a sine and a triangle wave.

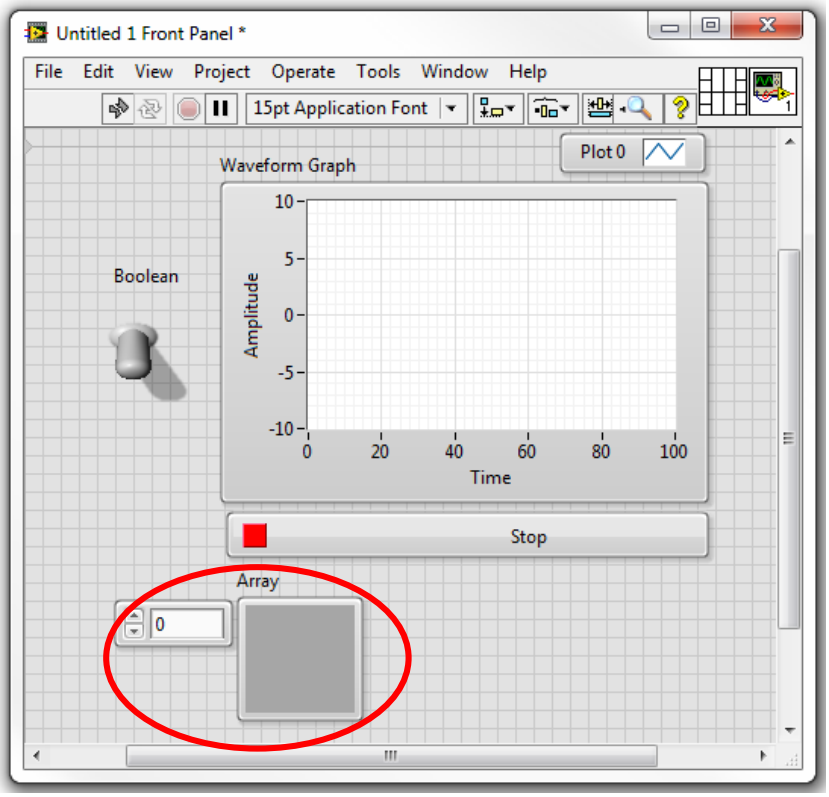
Adding Digital Output to Your Project

Often when you are creating control applications, controlling digital lines is necessary to interface with pumps, valves, lamps, etc. In the next part of this exercise we will extend our application to include control of the digital output lines.

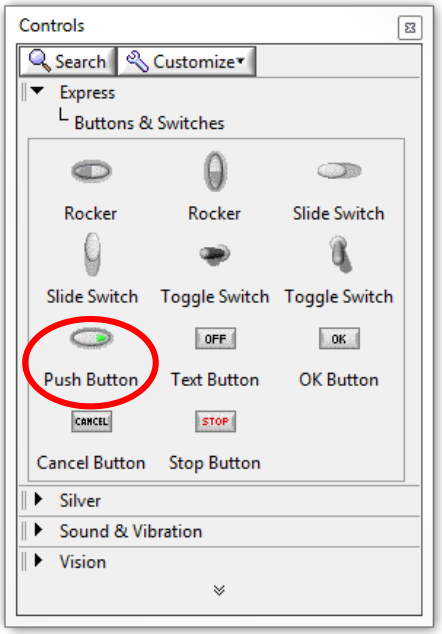
12. Right-click on your Front Panel and select an empty array container as shown below.



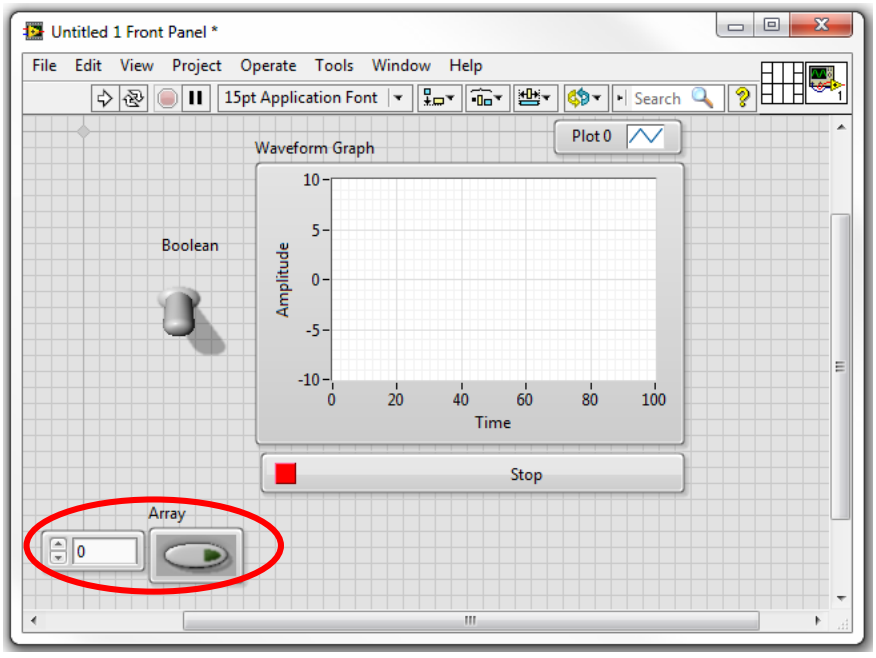
13. Place the array container at the bottom of the VI, as shown below.



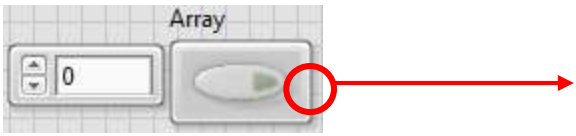
14. You can create an array of controls by simply inserting them into the array. Right-click on the Front Panel and select a push button from the Boolean palette and drop it into the empty array.



15. It should look like the following.



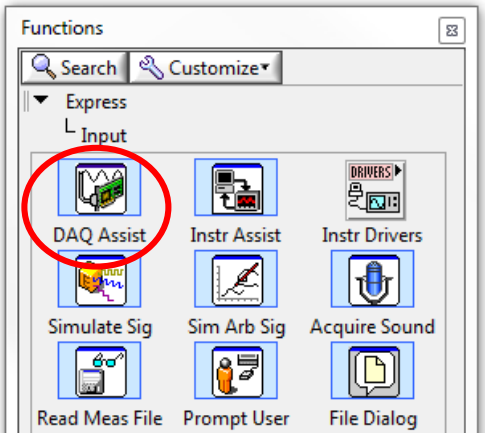
16. Next we will increase the size of the array of Booleans so that it contains eight elements. To do this, click on the right edge of the array container and drag it to the right until eight push-buttons are visible.



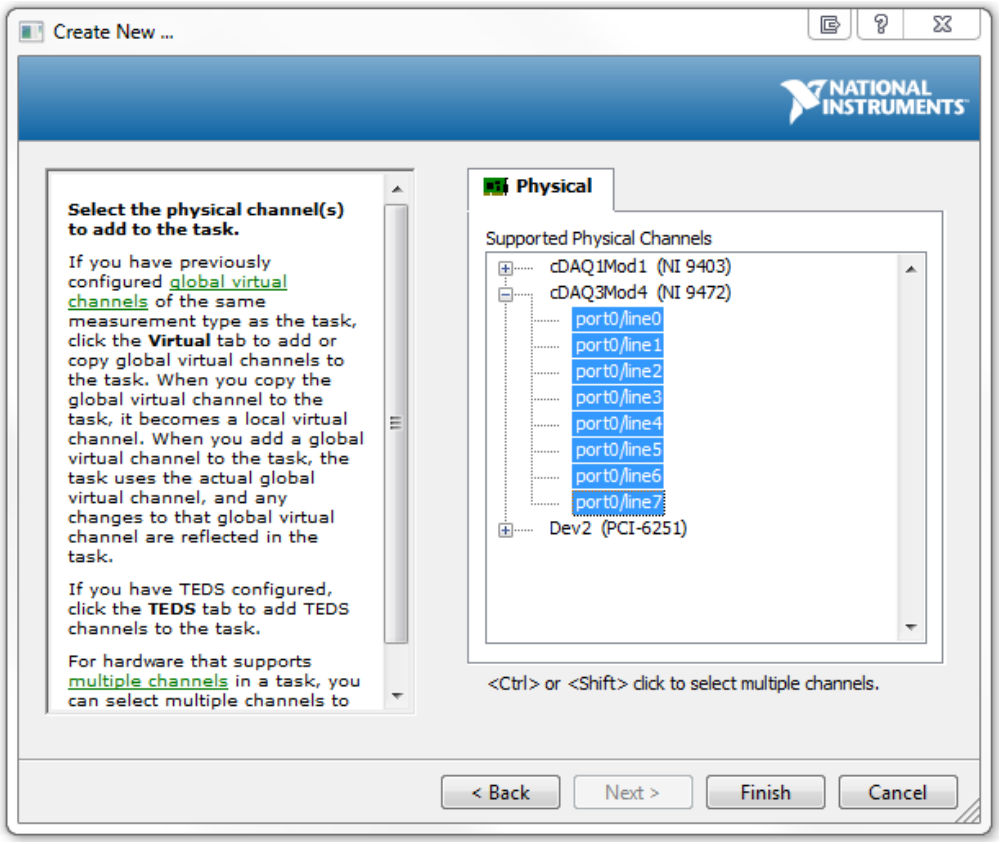
17. **VERY IMPORTANT STEP** - Click on the eighth switch to initialise the array size.

18. Next configure the hardware to recognise these switches and control one digital output per switch. Start by switching to the Block Diagram using <**Ctrl+E**>.

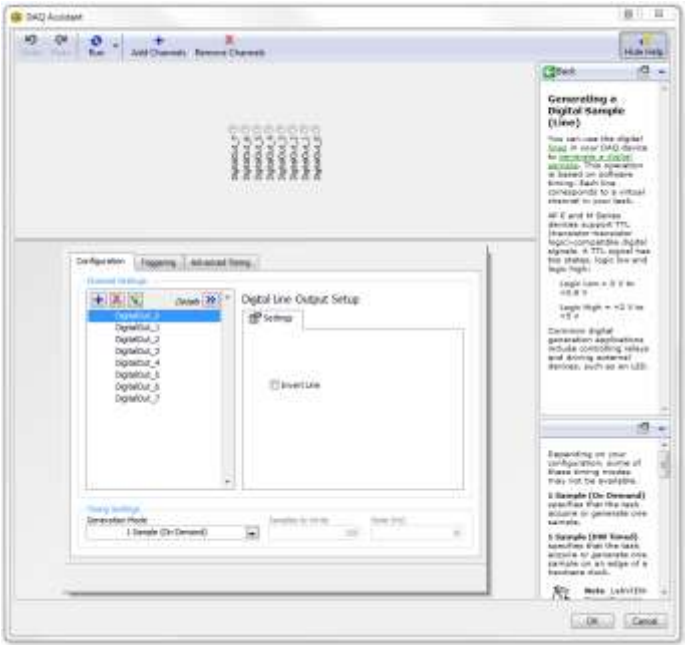
19. Next select the DAQ Assistant from the **Express» Output** palette.



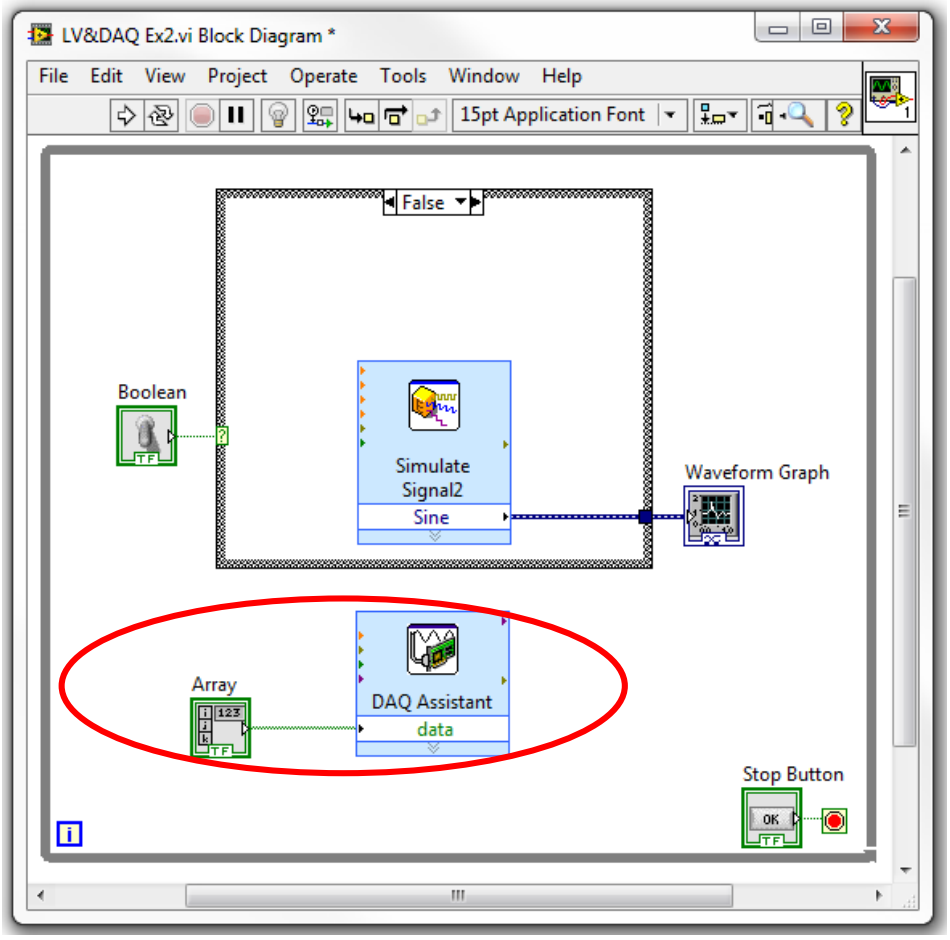
20. In the DAQ Assistant dialog box that pops up, select **Generate Signals» Digital Output» Line Output**.
21. Select all the digital output lines by clicking on the first then holding the shift key down and clicking on the last one. Then click **Finish**.



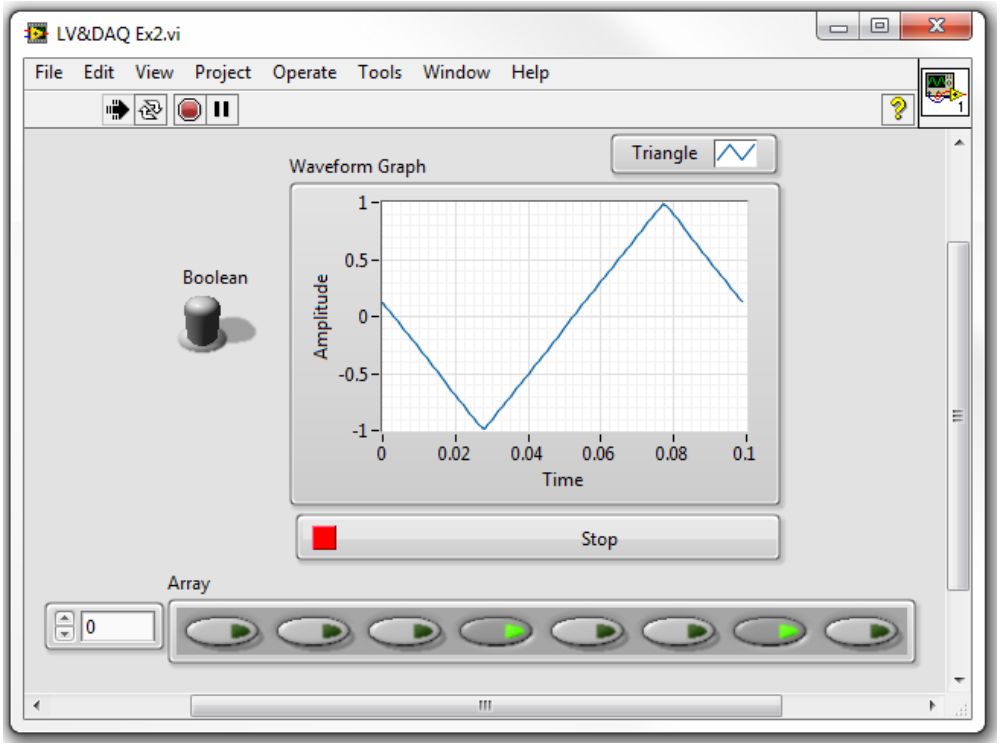
22. Click **OK** at the following dialog.



23. Finally wire the output of the array to the data input on the DAQ Assistant, as shown below.



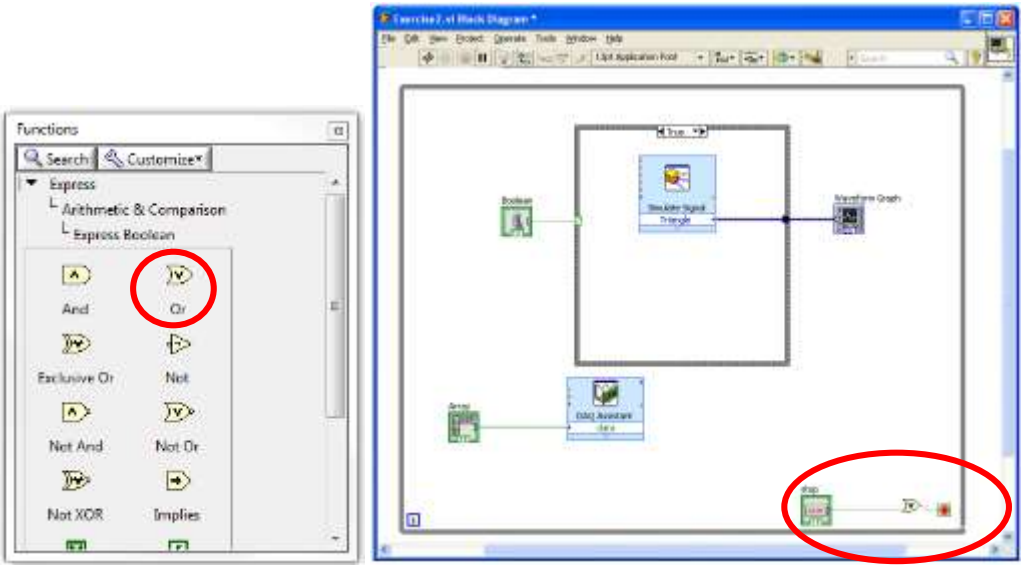
24. Switch to the Front Panel and **Run** the VI. Use the push buttons to activate the digital outputs on the NI 9472 digital module and watch the LEDs change as you select different push buttons.



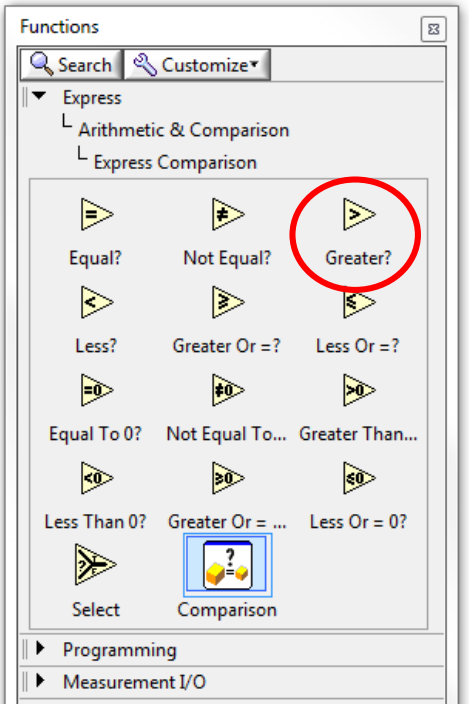
Optional Steps

25. Add logic to the VI so it will stop either when you press the **STOP** button or when the loop iterations have exceeded 10,000.

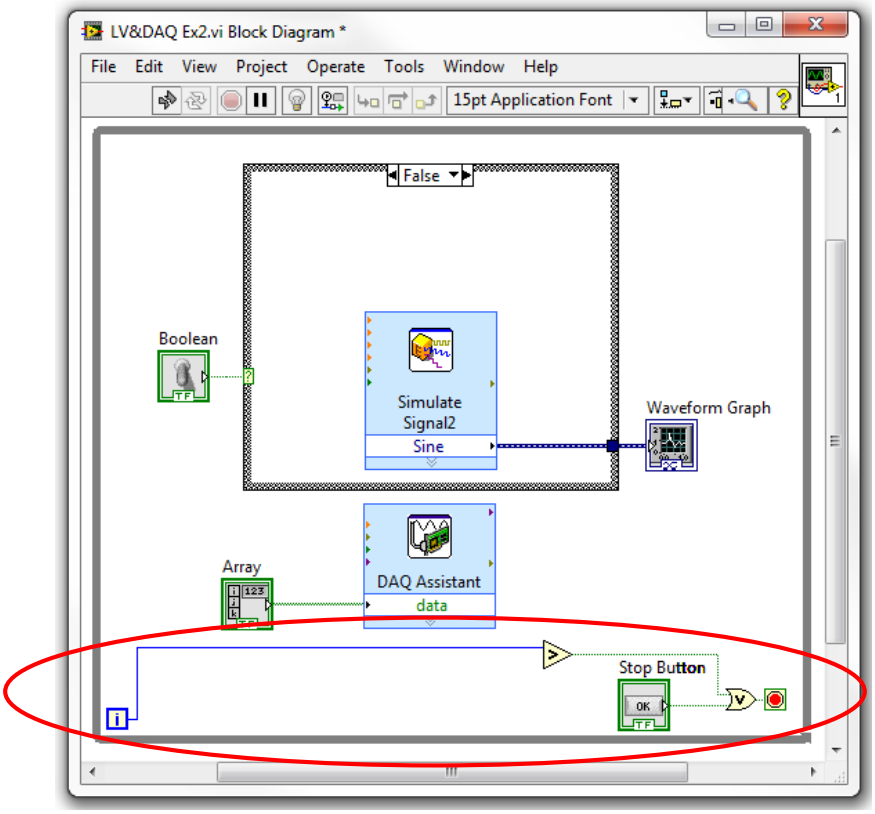
26. Do this by inserting an **Or** gate from the **Boolean** palette before the **STOP** button, as shown below.



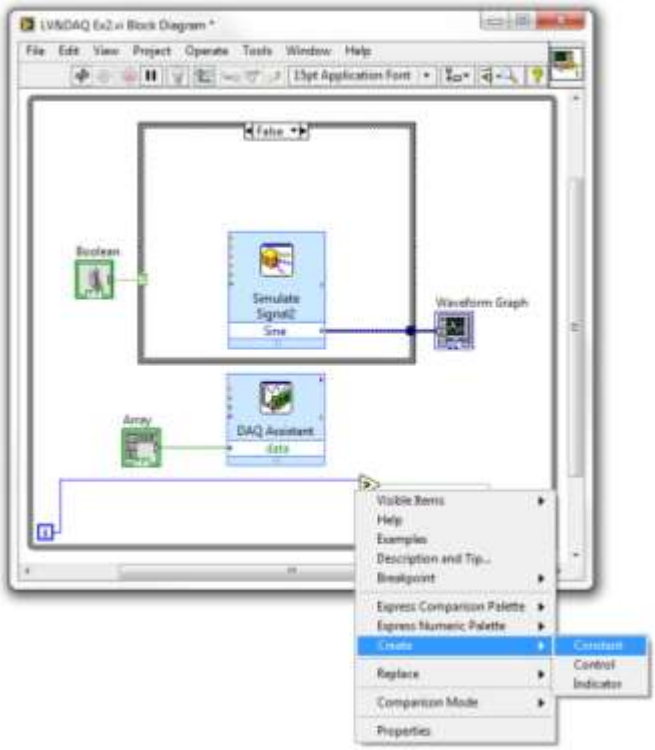
27. Next select the **Greater?** function from the **Comparisons** palette.



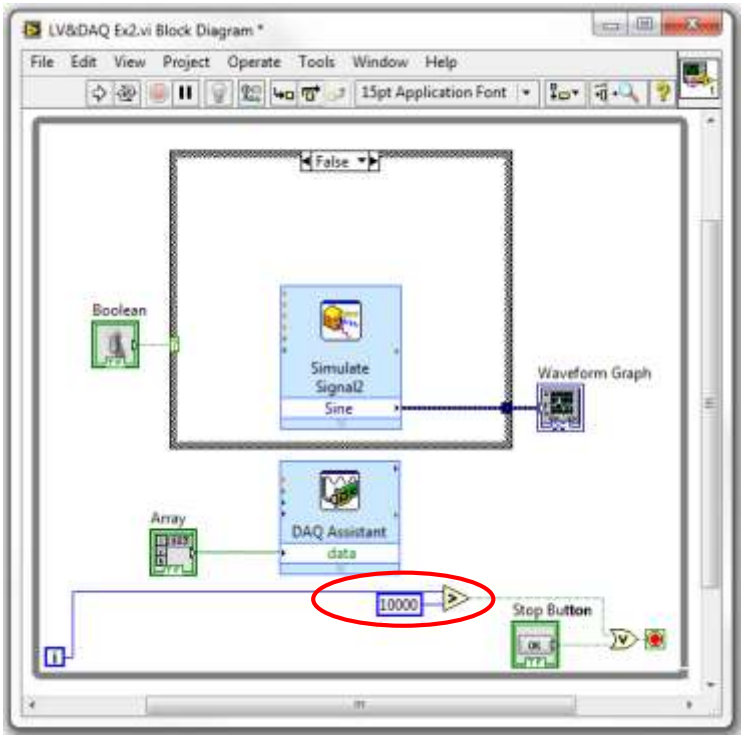
28. Wire as follows.



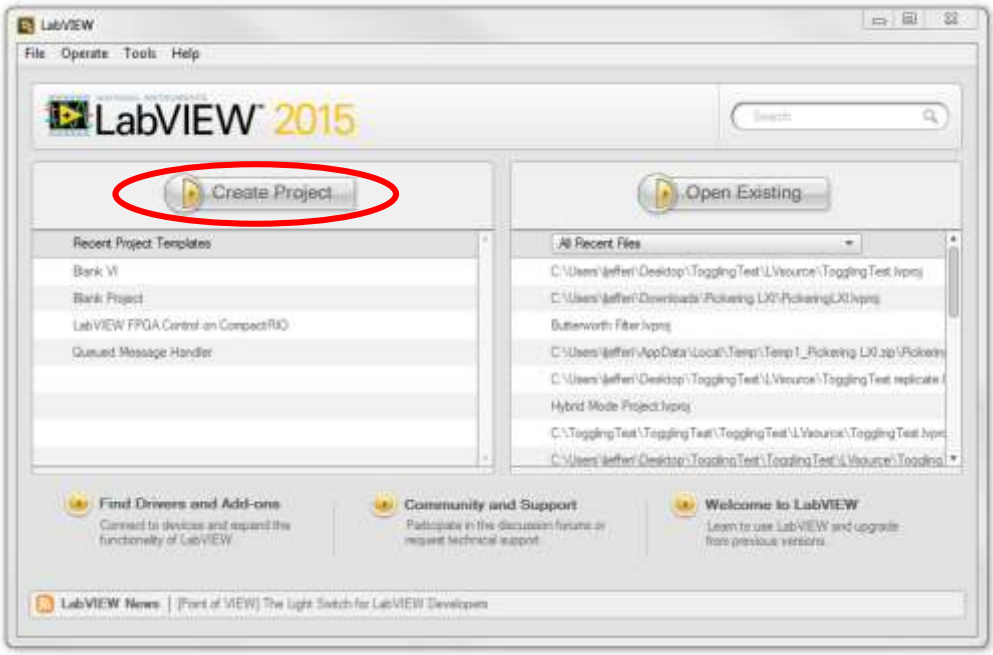
29. Right-click on the open terminal of the **Greater?** function and select **Create» Constant**.



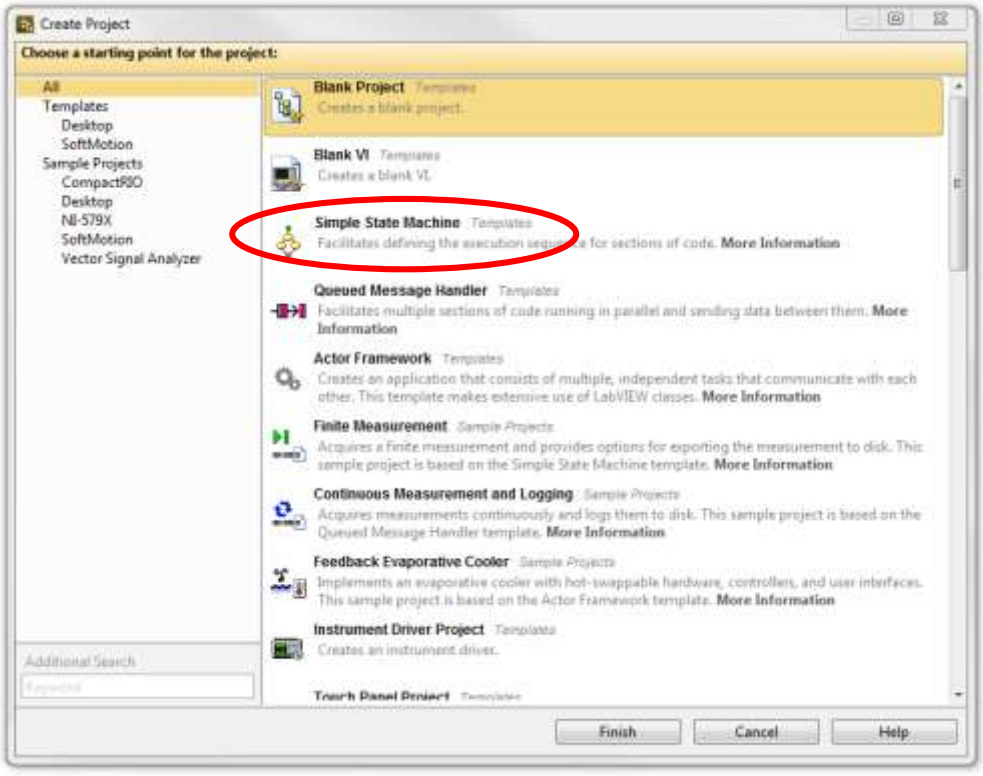
30. Change the constant to 10000 as shown below and **Run** the VI.



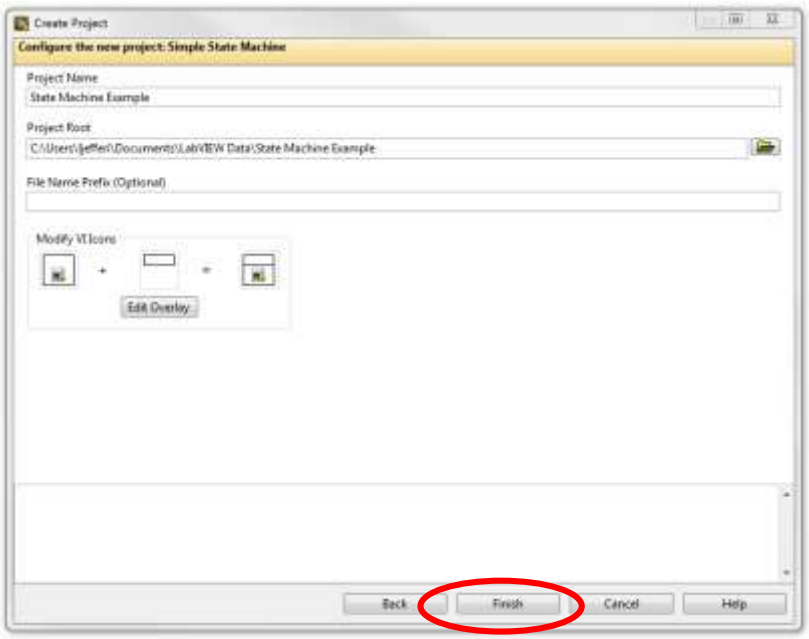
31. Finally, open and examine the **State Diagram template** that ships with LabVIEW. Go back to the splash screen then select **Create Project.**

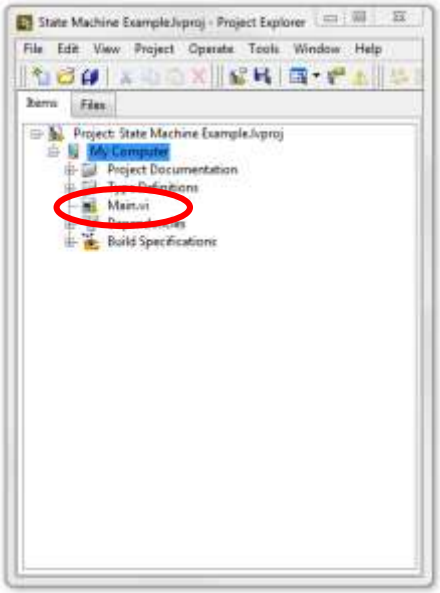


32. Select **Standard State Machine** from the list box as shown below.



33. Create the project and open the main VI. Examine the comments included in the template. There are several templates such as this one to help you get started with your LabVIEW applications. Take a few minutes to familiarise yourself with what is available.





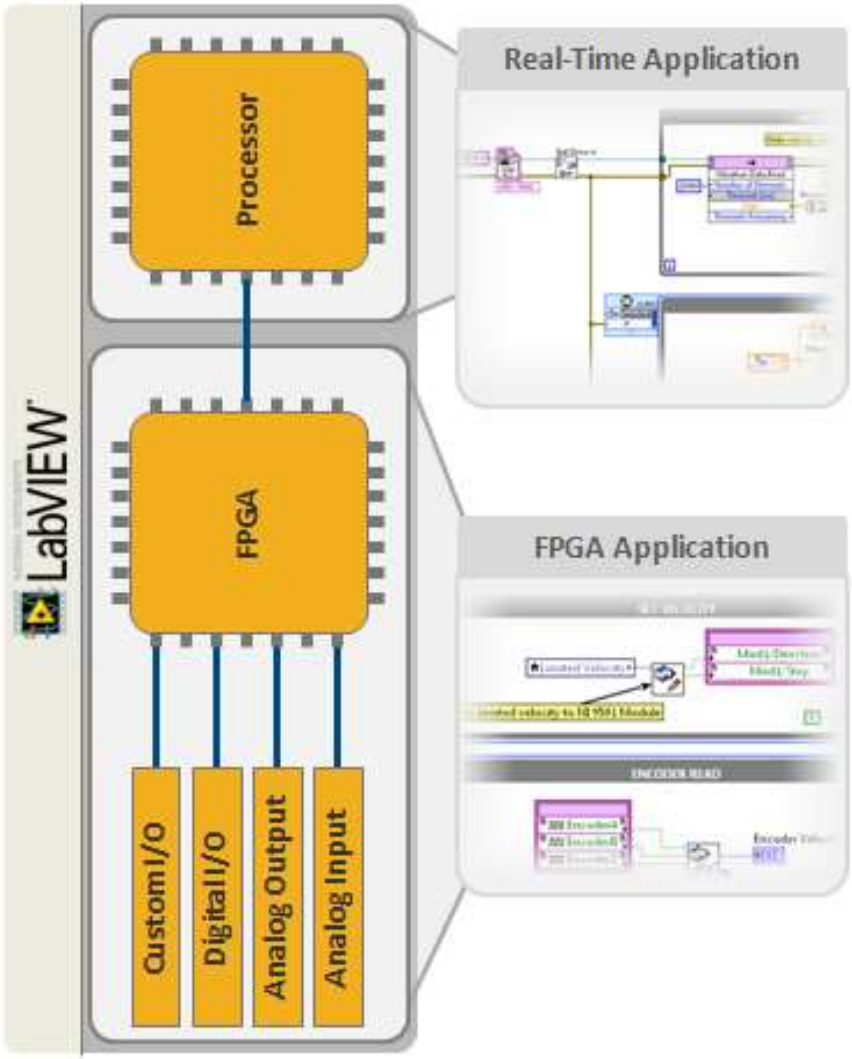
- End of Exercise -

NATIONAL INSTRUMENTS

LabVIEW Real-Time

Developing Reliable and Deterministic Applications

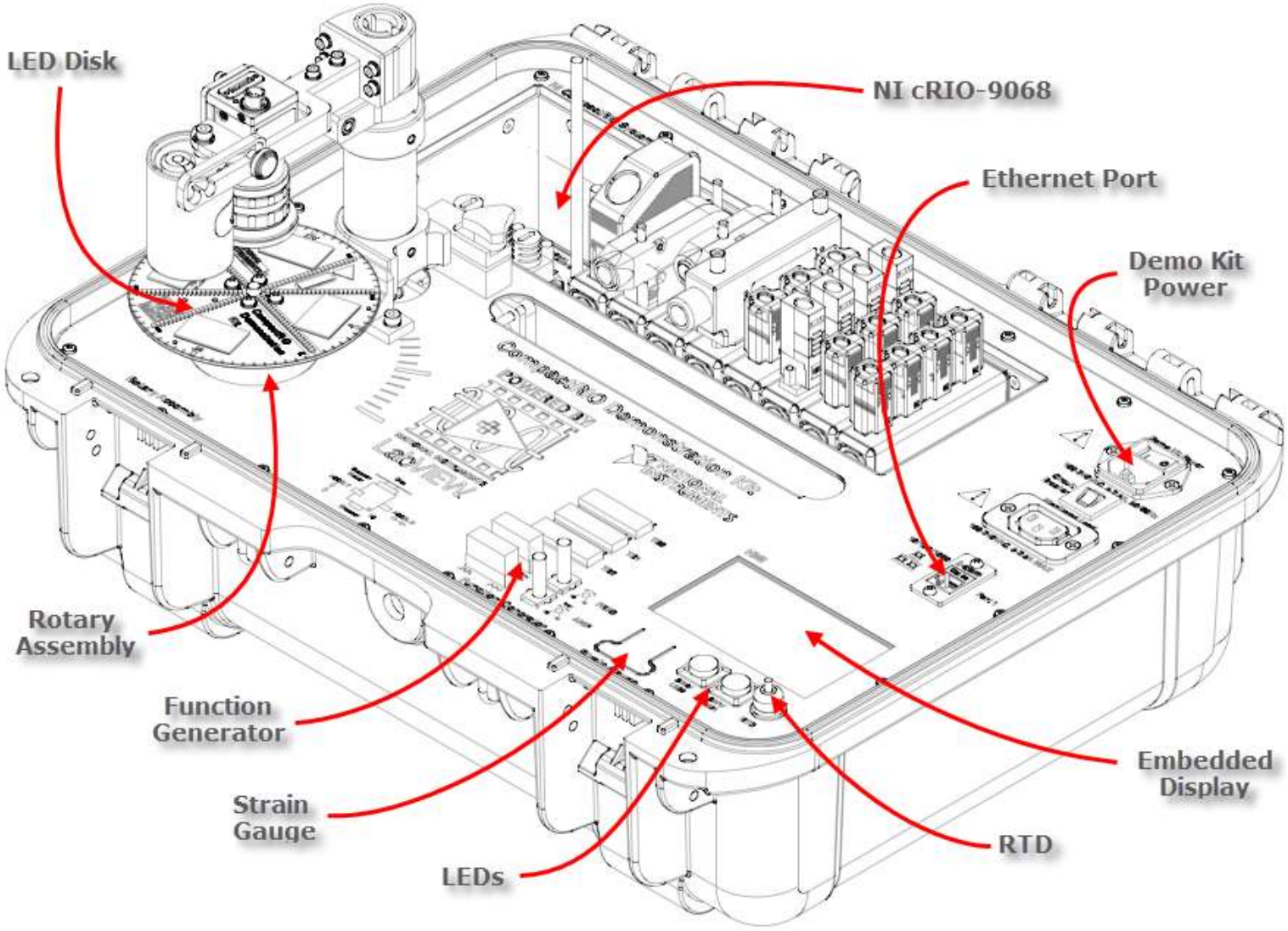
THE LabVIEW RIO ARCHITECTURE



- Real-time OS
- Application software
- Networking and peripheral I/O drives
- DMA, interrupt, and bus control drivers

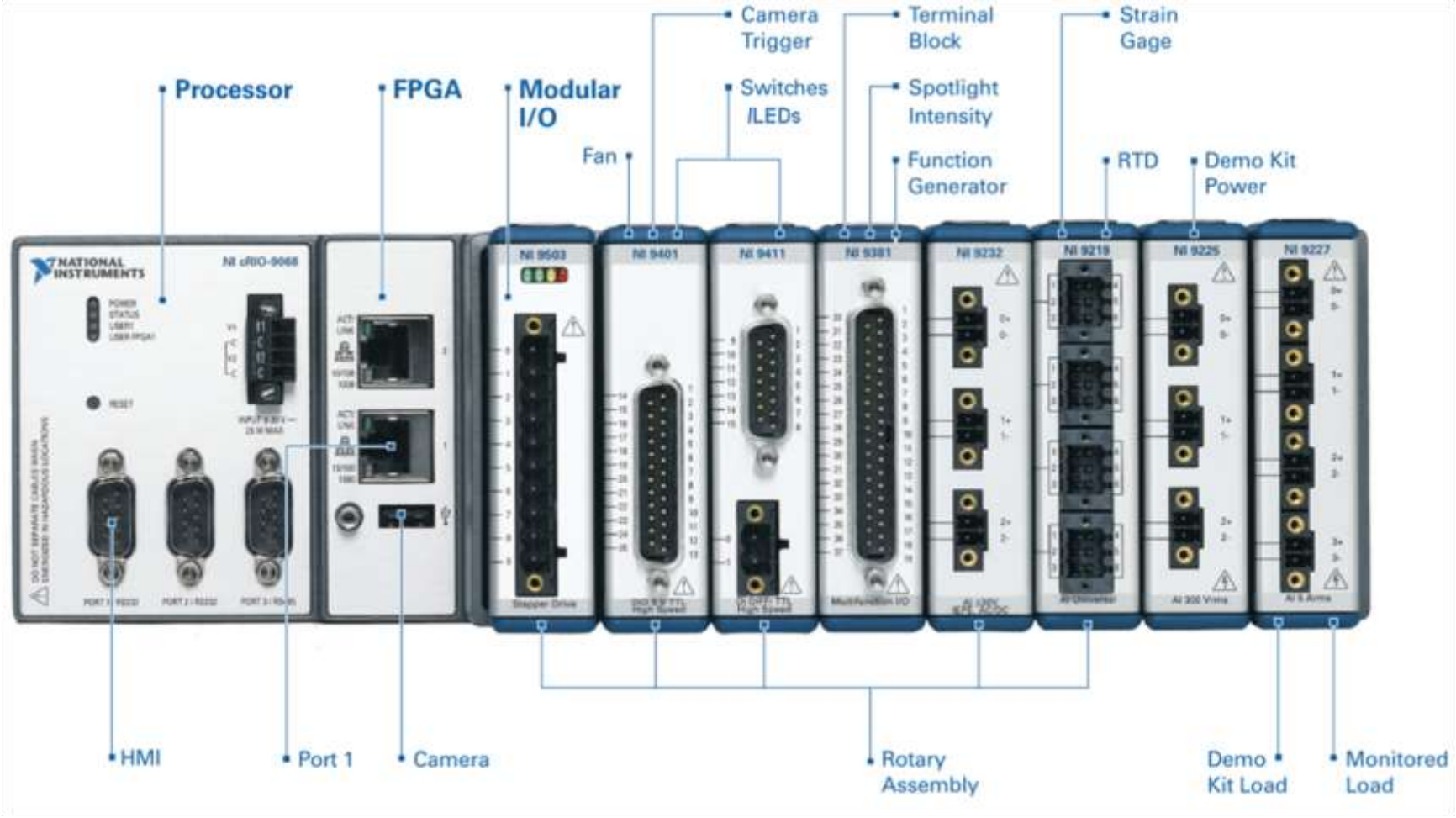
- Application IP
- Control IP
- Digital Signal Processing IP
- Specialized I/O drivers and interface
- DMA controller

NI CompactRIO DEMONSTRATION KIT



NI CompactRIO DEMONSTRATION KIT CONNECTION GUIDE

Built on the NI LabVIEW RIO Architecture

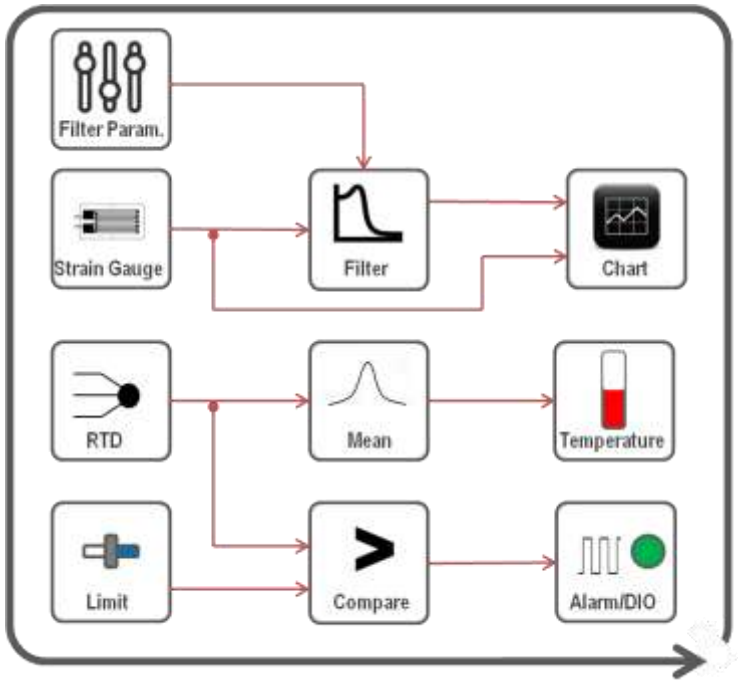


EXERCISE: TEMPERATURE AND STRAIN MONITORING

Goals

- Use ***LabVIEW*** system design software and the ***CompactRIO*** system configured in ***Scan Mode*** to interface with the sensors located in the ***CompactRIO Demonstration Kit***
- Use the ***LabVIEW Signal Processing*** library to obtain meaningful data from the sensors and display that information on a user interface

Part A—APPLICATION DESCRIPTION



In this exercise, use ***LabVIEW*** and the ***CompactRIO system*** configured in ***Scan Mode*** to acquire data from an ***RTD*** and a ***strain gauge***. Use the ***Signal Processing*** library to filter the strain gauge signal and obtain the mean of the RTD signal. Implement a ***limit-based alarm*** system for the RTD signal. Show the results of the analysis in a user interface.

CompactRIO System

NI Scan Mode

This feature gives you easy access to signals wired to measurement modules connected to the ***FPGA*** included in the ***CompactRIO system***.

In this exercise, use the measurement modules inserted in ***slots 2*** *and* ***6*** of the ***CompactRIO system***.

Inputs

Slot 6: **NI 9219**—*24-Bit Universal Analog Input (4 Diff, 100 S/s/ch)*
Analog Input 0 (AI0)→RTD (3-Wire Pt100-TCR3851)
Analog Input 2 (AI2)→Strain Gauge (Quarter Bridge/350 Ω)

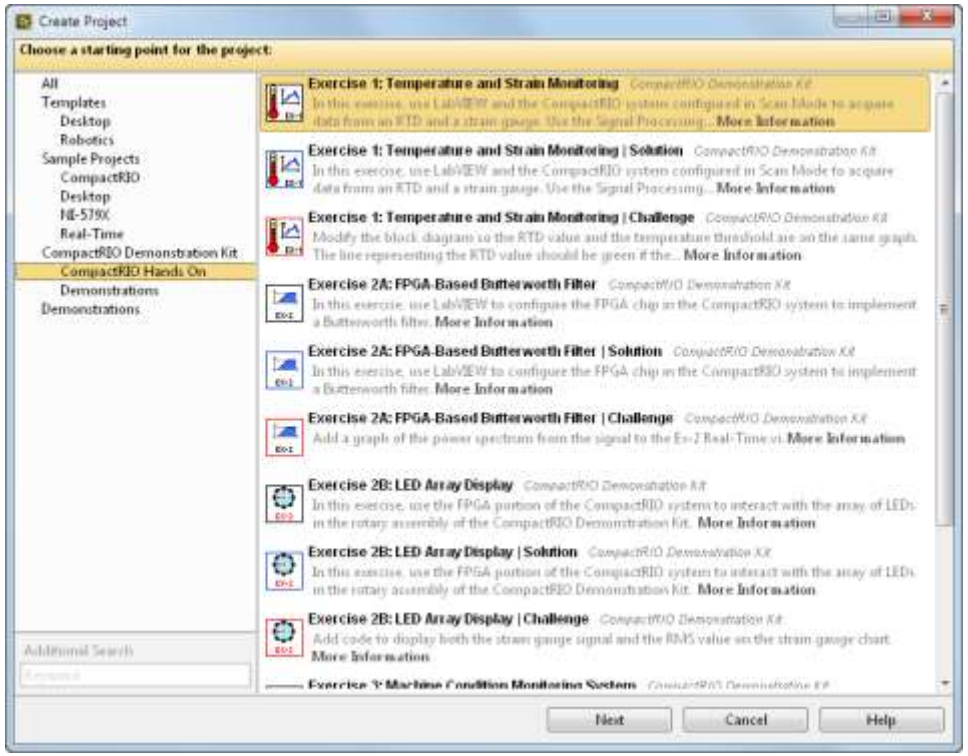
Outputs

Slot 2: NI 9401—*8 Ch, 5 V/TTL High-Speed Bidirectional Digital I/O Module*
Digital I/O 5 (DIO5)→LED1

Part B—CODE IMPLEMENTATION

1. Create an Instance of the Exercise 1: Temperature and Strain Monitoring Sample Project

Launch LabVIEW and create a new LabVIEW project from the Exercise 1: Temperature and Strain Monitoring sample project.



DETAILED INSTRUCTIONS

In this exercise, use an existing ***LabVIEW sample project*** as the starting point of the application.

- ***Launch LabVIEW***
  Launch LabVIEW by navigating to ***Start » All Programs » National Instruments » LabVIEW <Year>» LabVIEW.***

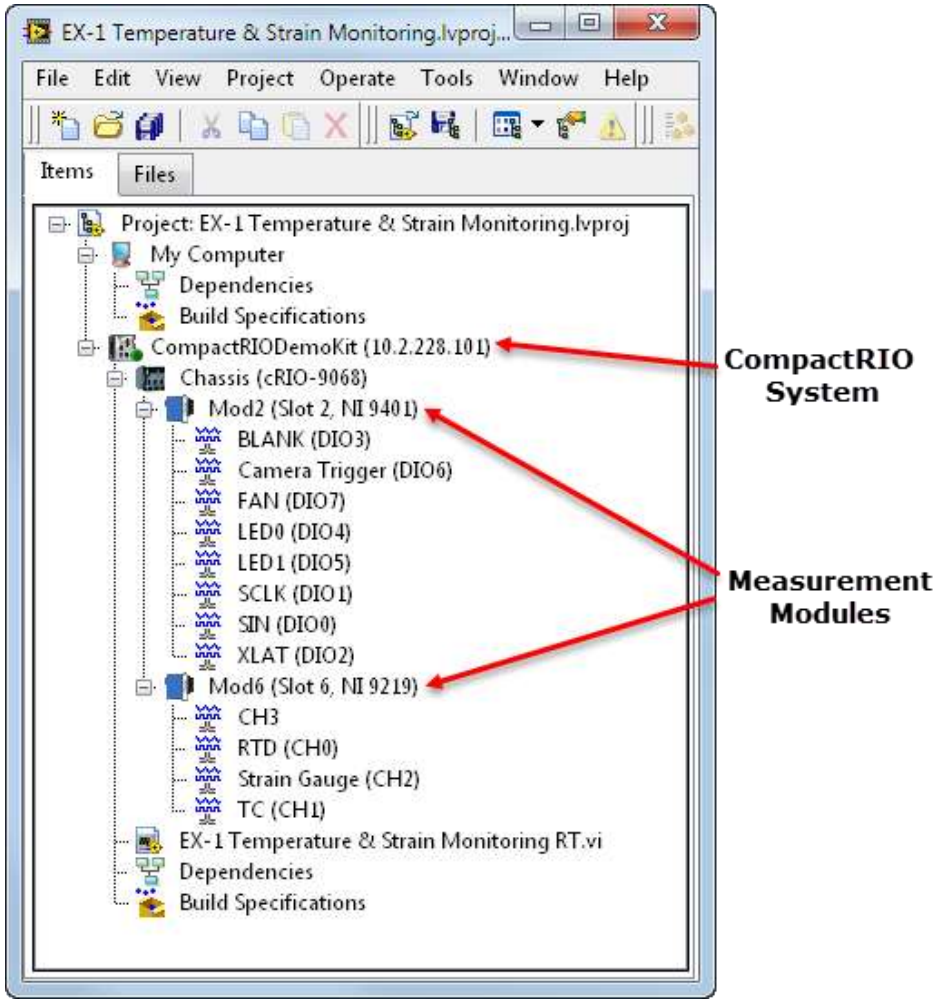
- ***Create an Instance of the Existing LabVIEW Project***
  Click on the ***Create Project*** button on the right side of the main screen of ***LabVIEW*** and create an instance of the **Exercise 1: Temperature and Strain Monitoring** sample project located at:

CompactRIO Demonstration Kit>> CompactRIO Hands On

Click **Next** to customize the **Project Name** and **Project Root** if desired. Finally, click **Finish** to create the project.

2. Explore the LabVIEW project.

Explore the project and expand the hierarchy on the project to display all measurement modules and channels to be used in the exercise.



DETAILED INSTRUCTIONS

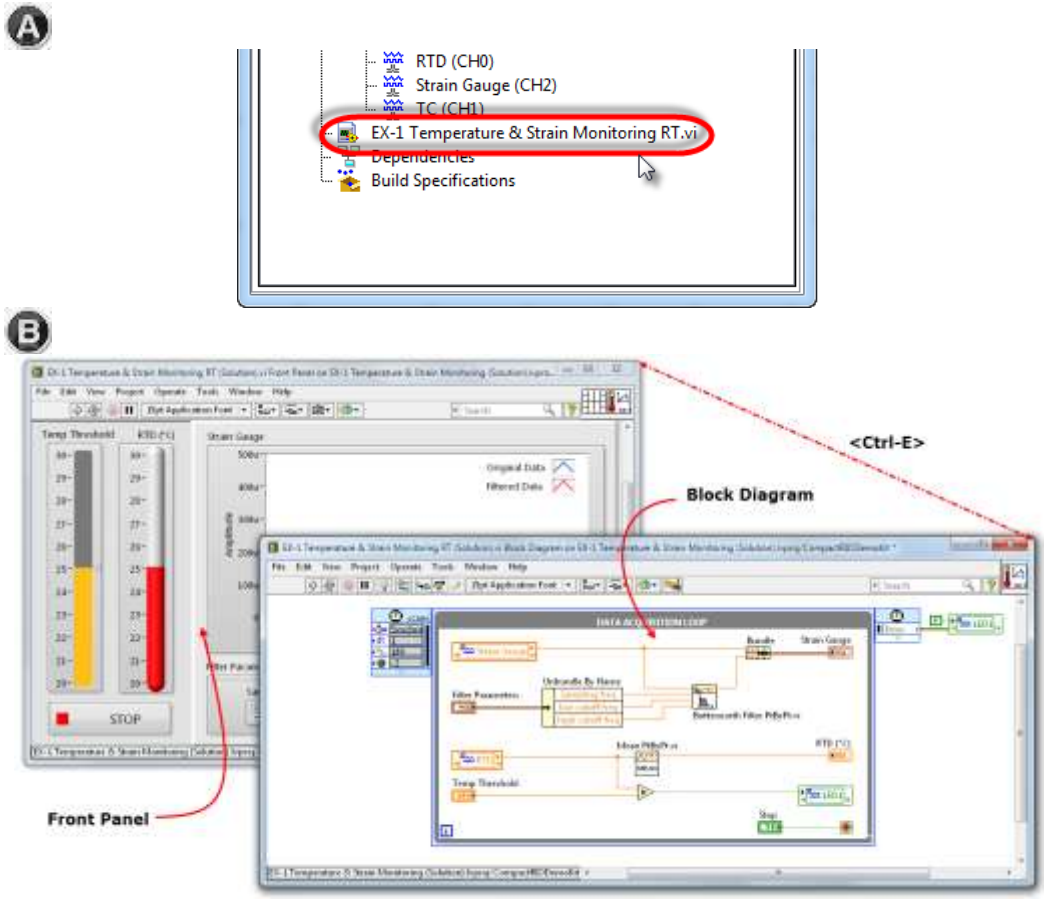
- ***Expand of the Exercise 1: Temperature and Strain Monitoring sample project***
  This project contains the ***CompactRIO system*** and ***measurement modules*** for this exercise.

  To expand the hierarchy and reveal the contents of the project, click the "**+**" boxes next to each part of the CompactRIO system and measurement modules listed.

  The inputs and outputs for this exercise have already been named and are listed under their corresponding modules.

3. Open the EX-1 Temperature & Strain Monitoring RT.vi file.

This file contains the user interface and the block diagram with the code that you need to complete the temperature control exercise.



DETAILED INSTRUCTIONS

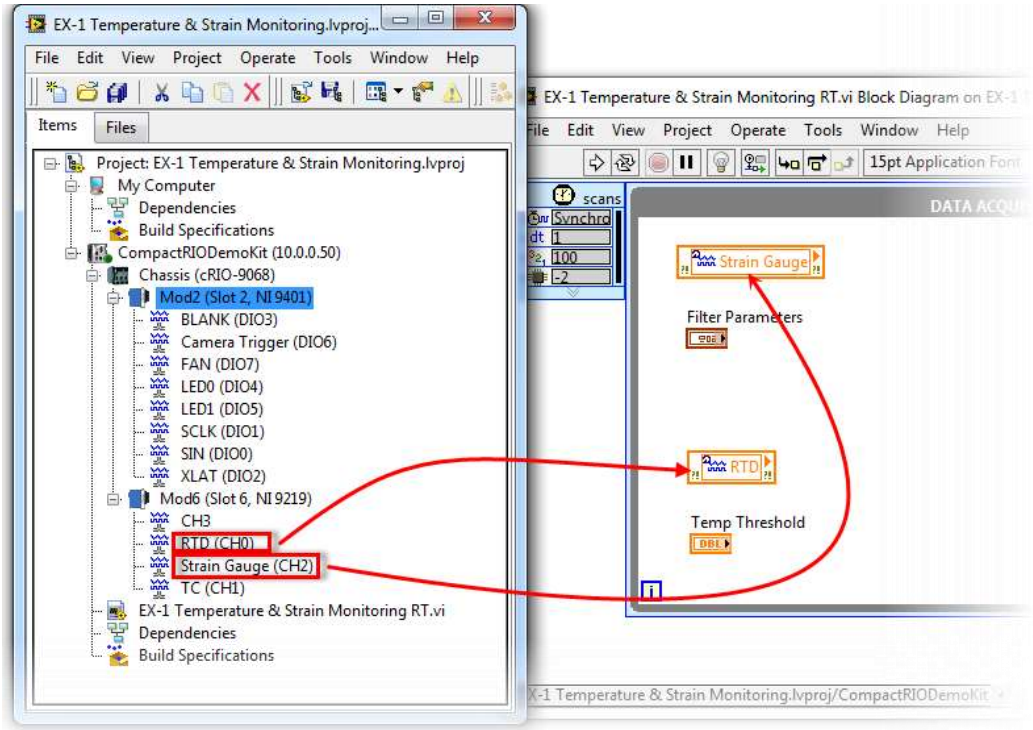
- ***Open the EX-1 Temperature & Strain Monitoring RT.vi File***
  Double-click on the ***EX-1 Temperature & Strain Monitoring RT.vi*** file located at the bottom of the LabVIEW Project Explorer window as shown in ***Figure A***.

- ***Display the Block Diagram***
  On the menu bar, click on ***Window » Show Block Diagram*** or press ***<Ctrl-E>*** to switch between the ***front panel*** and the ***block diagram*** as shown in ***Figure B***.

  Complete the code by inserting the inputs and outputs from the measurement modules into the ***While Loop*** labeled ***Data Acquisition***. Using a ***Butterworth Filter*** function and a ***Mean function***, monitor the ***RTD*** and the ***Strain Gauge***.

4. Add the inputs of the monitoring application to the block diagram.

Place the RTD and Strain Gauge I/O channels onto the block diagram.



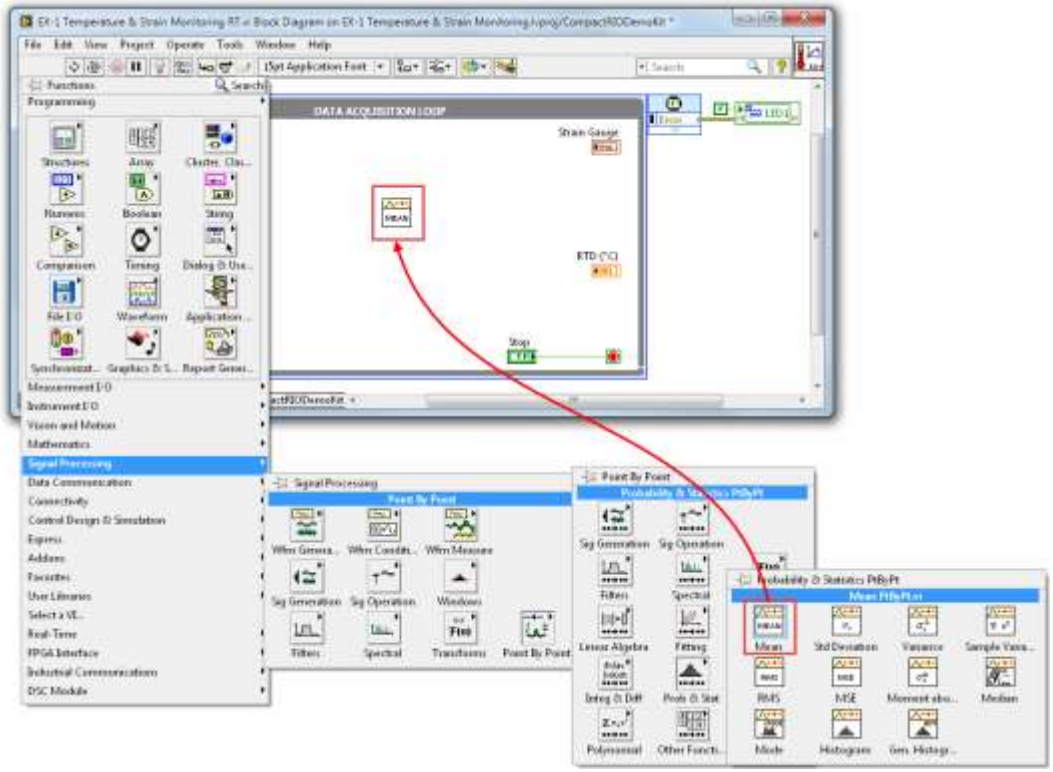
DETAILED INSTRUCTIONS

- ***Add the RTD I/O Node to the Block Diagram***
  From the LabVIEW Project Explorer window, drag and drop the channel labeled ***RTD (CH0)*** under ***Mod6 (Slot6, NI 9219)*** onto the block diagram inside the ***Data Acquisition loop***.

- ***Add the Strain Gauge I/O Node to the Block Diagram***
  From the LabVIEW Project Explorer window, drag and drop the channel labeled ***Strain Gauge (CH2)*** under ***Mod 6 (Slot6, NI 9219)*** onto the block diagram inside the ***Data Acquisition loop***.

5. Implement the code for calculating the mean of RTD measurements.

Insert the Mean Point-by-Point.vi function onto the block diagram.



DETAILED INSTRUCTIONS

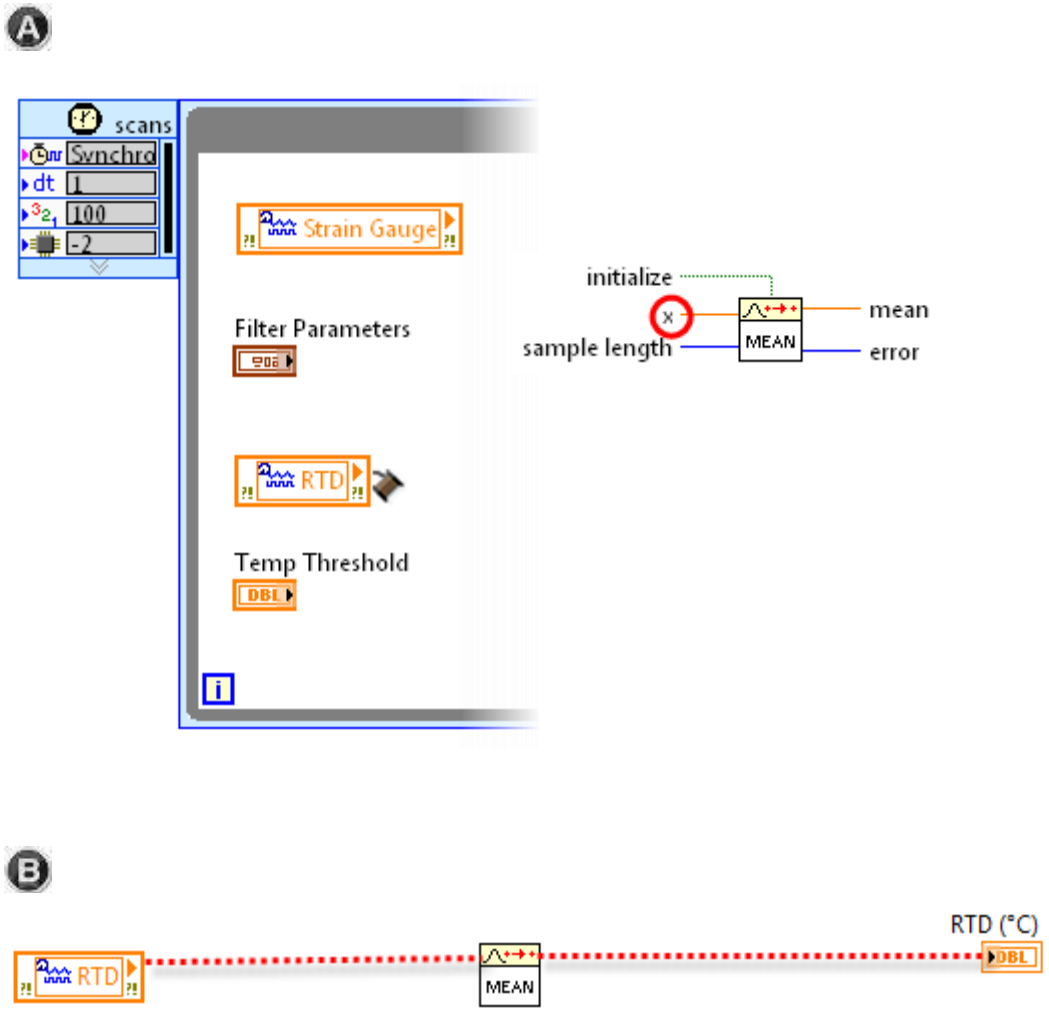
- ***Insert the Mean Point-by-Point.vi Function on the Block Diagram***
  Place the ***Mean Point-by-Point.vi*** inside the **Data Acquisition loop** by right-clicking on the block diagram to open the ***Functions Palette***.

  Navigate to this function by clicking to ***Signal Processing » Point by Point » Prob & Stat » Mean PtByPt.vi***.

  Point-by-point analysis is a method of continuous data analysis in which analysis occurs for each data point, point by point. In point-by-point analysis, the input-analysis-output process takes place continuously in ***real time***.

6. Wire the inputs and outputs of the MeanPtByPt.vi function.

Wire the RTD I/O terminal to the "x" input terminal of the MeanPtByPt.vi function and wire the output of the MeanPtByPt.vi function to the RTD (°C) indicator.



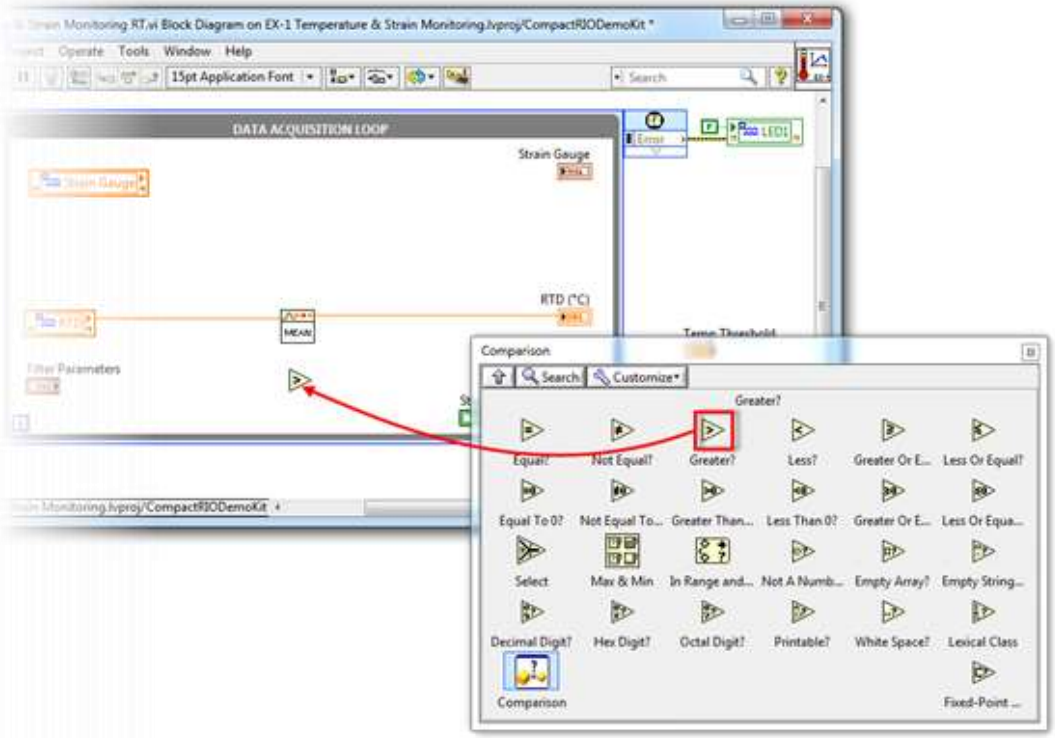
DETAILED INSTRUCTIONS

- ***Wire the RTD I/O Terminal to the "x" Terminal of the Mean PtByPt. vi***
  Move the cursor over the triangle in the upper-left corner of the ***RTD I/O terminal***. The cursor should become a spool; this indicates the cursor is in wiring mode. See ***Figure A***. Click on the terminal and move the cursor to the ***"x"*** input terminal of the ***Mean PtByPt.vi*** and click. A wire now connects the ***RTD I/O terminal*** to the ***Mean PtByPt*** function.

- ***Wire the Output of the Mean PtByPt.vi to the RTD (°C) Indicator***
  Create a wire that connects the Mean output terminal of the ***Mean PtByPt.vi*** function to the ***RTD (°C)*** indicator. The inputs and outputs should look like ***Figure B***.

7. Implement a temperature threshold for the system.

Add logic to the code to determine when the temperature is greater than a user-defined threshold.



DETAILED INSTRUCTIONS

- ***Add the Greater? Function to the Block Diagram***
  Place the ***Greater?*** function on the block diagram by opening the ***Functions Palette*** and navigating to ***Comparison » Greater?***.

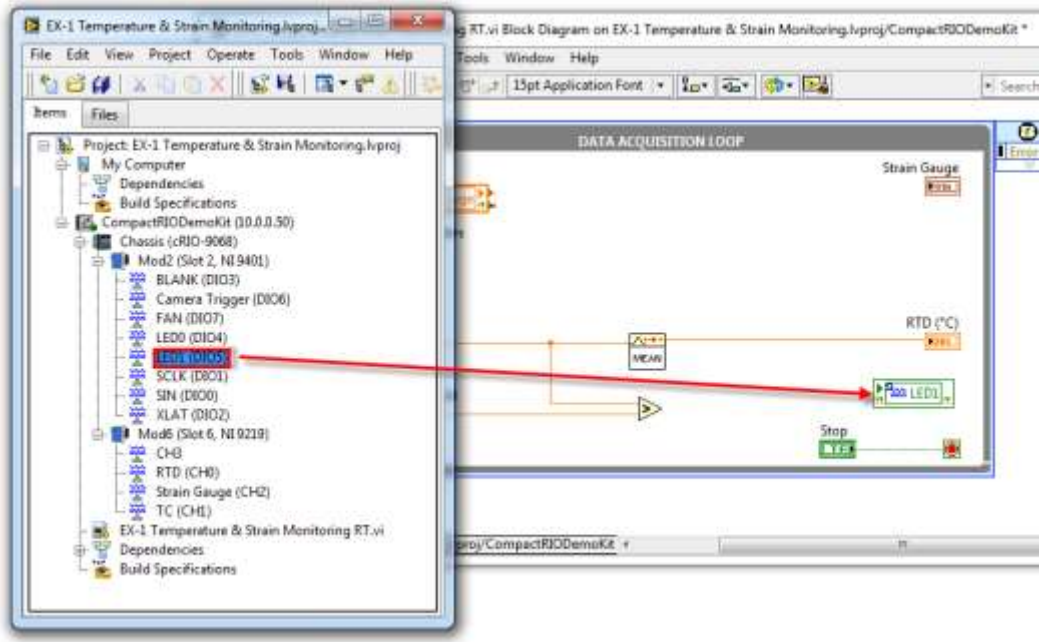
  This function helps you implement the ***limit-based alarm system***. You can now compare the current RTD value against a user-defined limit to fire up an alarm.

8. Complete wiring the temperature threshold.

Wire the output terminals of the RTD I/O terminal and the Temperature Threshold control to the Greater? Function. Also create an instance of the LED indicator that illuminates when the RTD I/O output is greater than the temperature threshold.

A

B



DETAILED INSTRUCTIONS

- ***Wire Inputs to the Greater? Function***
  Wire the output of the ***RTD I/O terminal*** to the first input terminal of the ***Greater?*** function by creating a branch from the wire connecting the ***RTD I/O terminal*** and the ***Mean PtByPt.vi***. Click on the wire with the cursor in wiring mode and move the cursor to the first input terminal of the ***Greater?*** function.

  Wire the output of the ***Temp Threshold control*** to the second input terminal of the ***Greater?*** function as shown in ***Figure A***.

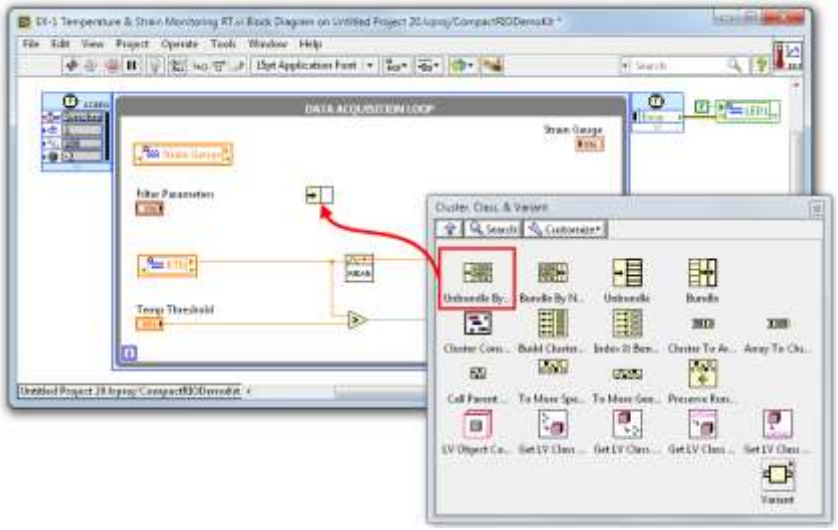
- ***Create an Instance of the LED Indicator***
  Create an instance of the ***LED*** indicator by dragging and dropping the ***LED1 (DIO5)*** from under ***Mod 2 (Slot2, NI 9401)*** in the LabVIEW Project Explorer into the ***Data Acquisition loop*** as shown in ***Figure B.***

  Wire the output of the ***Greater?*** function to the input terminal of the ***LED1 I/O terminal***.

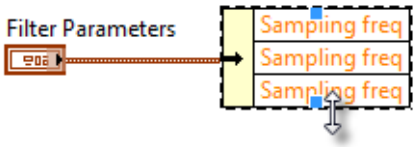
9. Unbundle the values of the filter parameters cluster.

Unbundle the values of the filter parameters cluster to access the individual frequency values within the cluster.

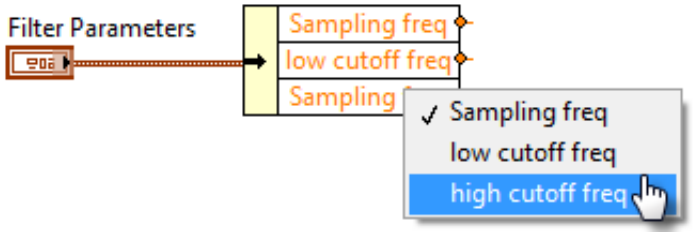
A



B



C



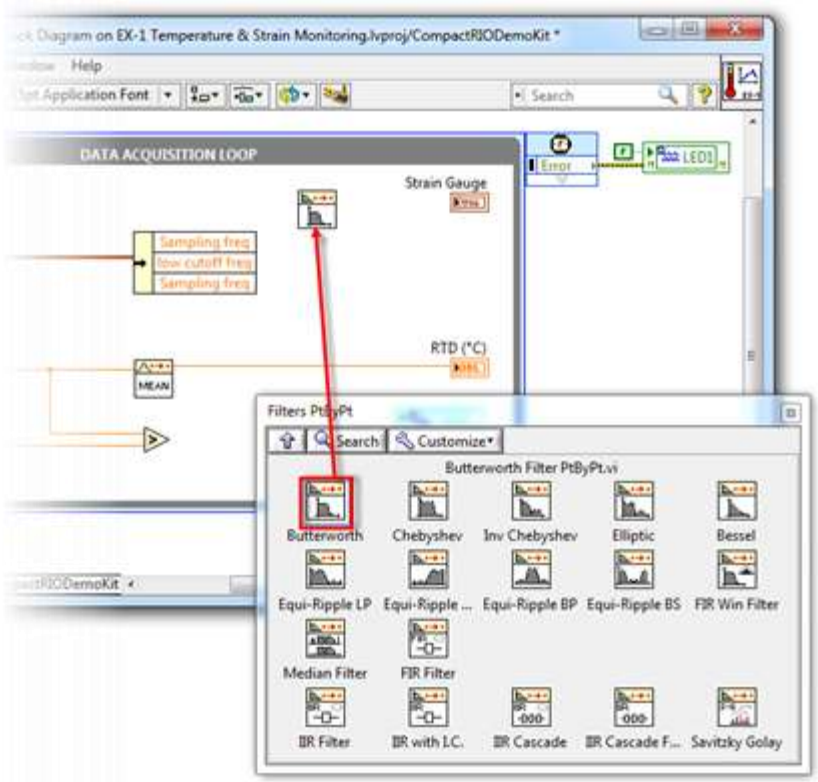
DETAILED INSTRUCTIONS

- ***Use the Unbundle By Name Function to Access the Frequency Values***
  Place an ***Unbundle By Name*** function onto the block diagram by opening the ***Functions Palette*** and navigating to ***Cluster, Class, & Variant » Unbundle By Name*** as shown in ***Figure A***.

  Wire the output from the ***Filter Parameters*** control to the input terminal of the ***Unbundle By Name*** function as shown in ***Figure B***. Drag the bottom resizing tool on the ***Unbundle By Name*** function down to reveal the three boxes labeled ***"Sampling freq"*** as shown in ***Figure B***. Select the frequency that each box represents as shown in ***Figure C.***

10. Implement a Butterworth filter to filter the strain gauge value.

Insert the Butterworth Filter PtByPt.vi onto the block diagram to provide filtering for the strain gauge.



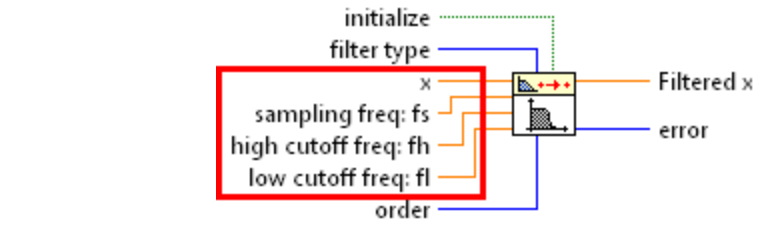
DETAILED INSTRUCTIONS

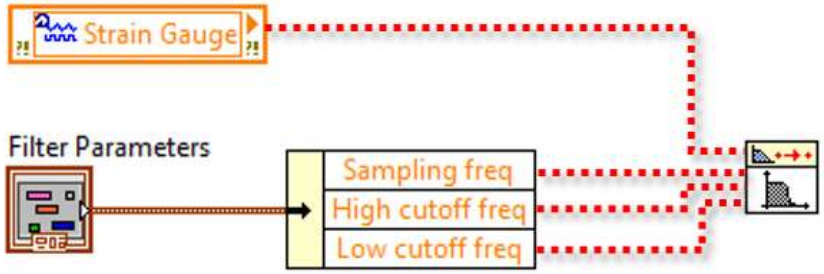
- ***Insert the Butterworth Filter PtByPt.vi Onto the Block Diagram***
  Place the ***Butterworth Filter PtByPt.vi*** onto the block diagram by opening the ***Functions Palette*** and navigating to ***Signal Processing » Point By Point » Filters » Butterworth Filter PtByPt.vi***.

  Wire the output of the ***Strain Gauge I/O*** terminal to the first input terminal, or the **"x"** terminal, of the ***Butterworth Filter*** function.

11. Wire the filter parameters and the strain gauge I/O terminal to the Butterworth Filter PtByPt function.

Bundle the raw Strain Gauge Value and the output of the Butterworth Filter PtByPt function together and display on the same graph on the front panel.





DETAILED INSTRUCTIONS

- ***Wire the Outputs of the Unbundle By Name Function and the Strain Gauge I/O Terminal to the Butterworth Filter PtByPt Function***

  Wire the output of the ***Strain Gauge I/O*** terminal to the first input terminal, or the **"x"** terminal, of the ***Butterworth Filter*** function.

  Wire the ***Sampling freq terminal*** from the ***Unbundle By Name*** function to the second input terminal of the ***Butterworth Filter*** function.

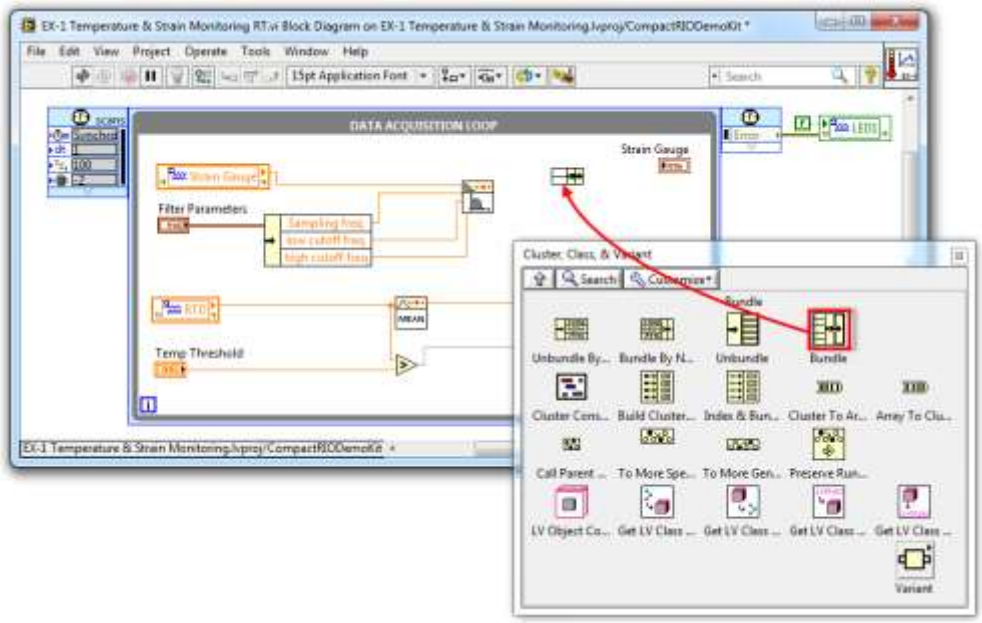
  Wire the low ***cutoff freq terminal*** from the ***Unbundle By Name*** function to the third input terminal of the ***Butterworth Filter*** function.

  Finally, wire the ***high cutoff freq terminal*** of the ***Unbundle By Name*** function to the last input terminal of the ***Butterworth Filter*** function.

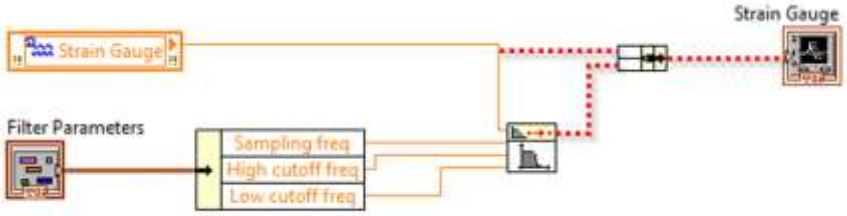
12. Display the raw strain gauge value and the filtered strain gauge value on the front panel.

Bundle the raw Strain Gauge Value and the output of the Butterworth Filter PtByPt function together and display on the same graph on the front panel.

A



B



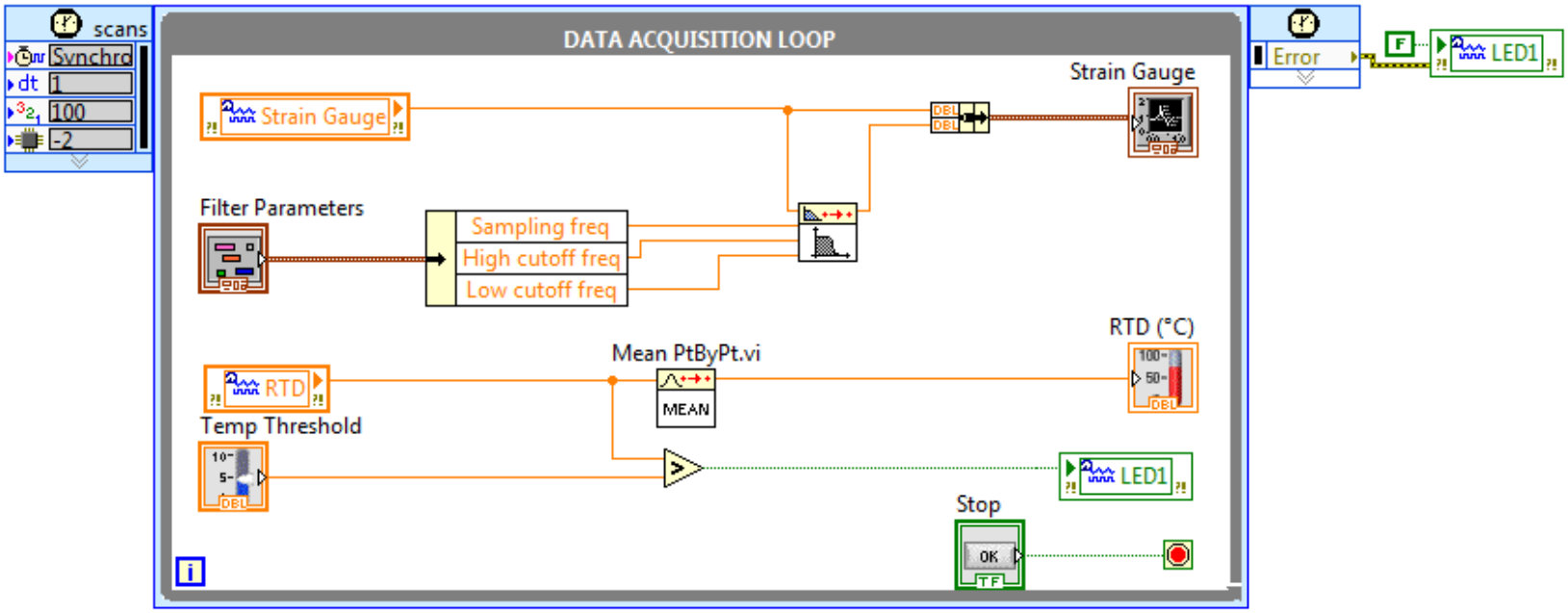
DETAILED INSTRUCTIONS

- ***Bundle the Raw Strain Gauge Value and the Filter Stain Gauge Value***
  Place a ***Bundle*** function on the block diagram by opening the ***Functions Palette*** and navigating to ***Cluster, Class, & Variant » Bundle***. Wire the output from the ***Strain Gauge I/O terminal*** to the first input terminal of the ***Bundle*** function. Wire the output of the ***Butterworth Filter PtByPt.vi*** to the second input terminal of the ***Bundle*** function as shown in ***Figure A.***

- ***Display the Strain Gauge Values on the Same Graph***
  Wire the output of the ***Bundle*** function to the ***Strain Gauge cluster indicator*** to display the values on the same graph. The wiring should be similar to the wiring in ***Figure B***.

13. The completed block diagram should look like the image below.

Be sure to save your work by pressing <Ctrl-S> or clicking File » Save.

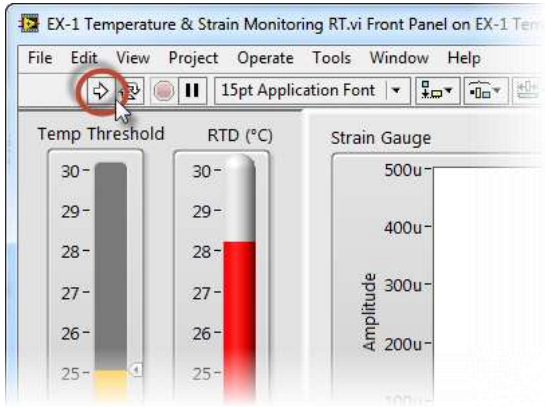


Part C—RUN THE APPLICATION

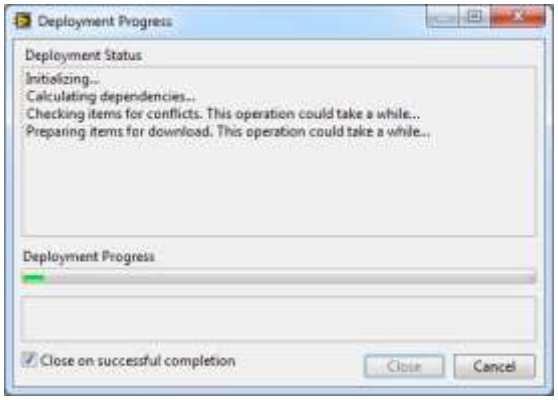
14. Run the application.

Run the EX-1 Temperature & Strain Monitoring RT.vi by clicking the run arrow to deploy the code to the CompactRIO system.

A



B



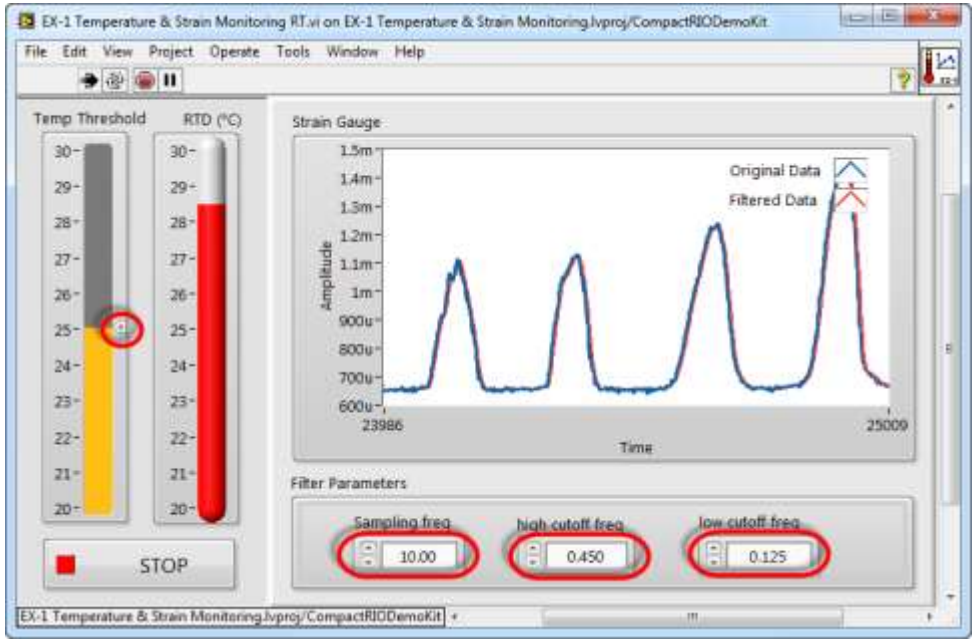
DETAILED INSTRUCTIONS

- ***Run the VI***
  Switch back to the front panel **<Ctrl-E>** and run the program by clicking on the **arrow button** at the top of the front panel as shown in ***Figure A***. Save the VI when prompted.

- ***Deploy the VI***
  Running the VI deploys the code through the ***Ethernet cable*** to the real-time controller of the ***CompactRIO system*** as shown in ***Figure B***. This initializes an interactive execution of the code in which the code is executed on the CompactRIO system while you monitor and control inputs and outputs from the user interface on the development machine.

15. Modify the Temp Threshold and Filter Parameters values to interact with the system.

Change the values of the Temp Threshold and Filter Parameters on the front panel to interact with the application.



DETAILED INSTRUCTIONS

- ***Change the Value of the Temp Threshold on the Front Panel***
  Slide the ***Temp Threshold control*** to change the value of the temperature threshold for the system.

- ***Change the Values of the Filter Parameters on the Front Panel***
  Adjust the values of the different ***Filter Parameter values*** to see how the filtering of the strain gauge measurement changes.

16. Stop the application.

Stop the application of the code by pressing the Stop button on the front panel of the EX-1 Temperature & Strain Monitoring RT.vi.

Part D—CHALLENGE

17. Modify the code to display temperature readings on a graph.

Modify the block diagram so the RTD value and the temperature threshold are on the same graph. The line representing the RTD value should be green if the value is below the temperature threshold and red if the value is above the temperature threshold.

<END OF EXERCISE 1: TEMPERATURE AND STRAIN MONITORING>

LabVIEW FPGA

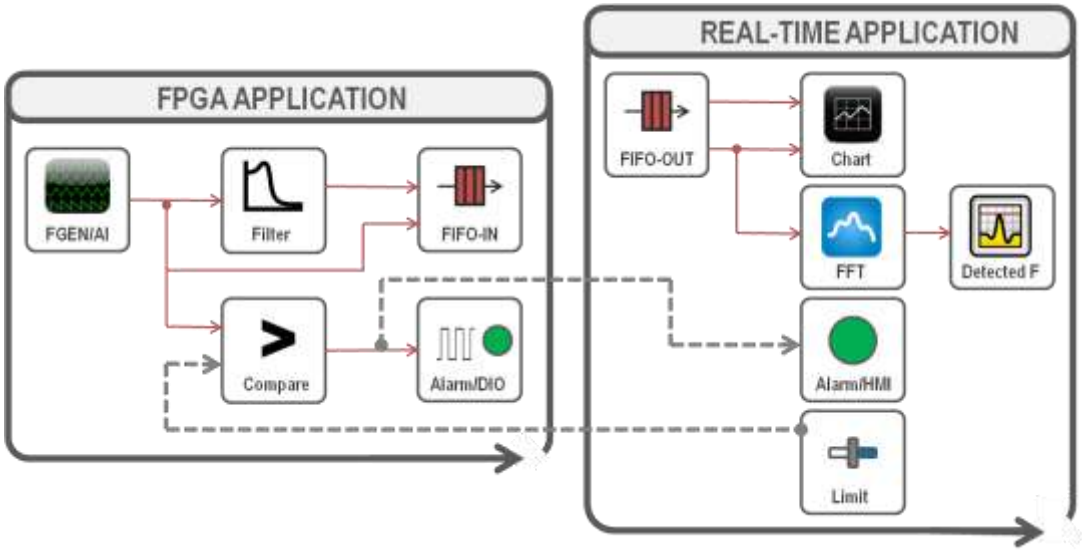
High-Speed Control and Greater Determinism

EXERCISE: FPGA-BASED BUTTERWORTH FILTER

Goals

- Use the ***FPGA*** portion of the ***CompactRIO system*** to implement a ***Butterworth filter*** for the signal coming from the ***function generator*** of the CompactRIO Demonstration Kit
- Use the ***simulation*** capabilities of ***LabVIEW FPGA*** to validate the design before compiling
- Use the ***real-time processor*** of the ***CompactRIO system*** to interface with the ***FPGA*** and generate a ***user interface*** to visualize the function generator and filtered signals as well as the calculated frequency

Part A—APPLICATION DESCRIPTION



In this exercise, use ***LabVIEW*** to configure the ***FPGA*** chip in the ***CompactRIO system*** to implement a ***Butterworth filter***.

As shown in the diagram on the left, the FPGA portion of the architecture is used to acquire a signal ranging from ***0 V to 5V*** at ***0 kHz to 1 kHz*** coming from the ***function generator*** in the ***CompactRIO Demonstration Kit.*** Filter the signal in this section of the application and send it via a ***DMA FIFO*** to the ***real-time processor*** for additional processing and visualization. Additionally, implement an ***alarming system*** based on a limit value coming from the user interface implemented in the real-time processor.

CompactRIO System

Field-Programmable Gate Array (FPGA)

An FPGA is a reprogrammable silicon chip. In contrast to programming the processors in your PC, programming an FPGA "physically" rewires the chip itself to implement your functionality rather than run a software application.

Real-Time Operating System (RTOS)

An RTOS is specially design to manage hardware resources and run applications with very precise timing and a high degree of reliability.

In this exercise, use measurement modules in ***slots 2 and 4*** of the **CompactRIO *system***.

Inputs

Slot 4: NI 9381—*0 V to 5 V AI/AO Module With 4 LVTTL DIO Lines*
Mod4/AI7→Function Generator
Sampling Period→20 kS/s
Resolution→12-bit

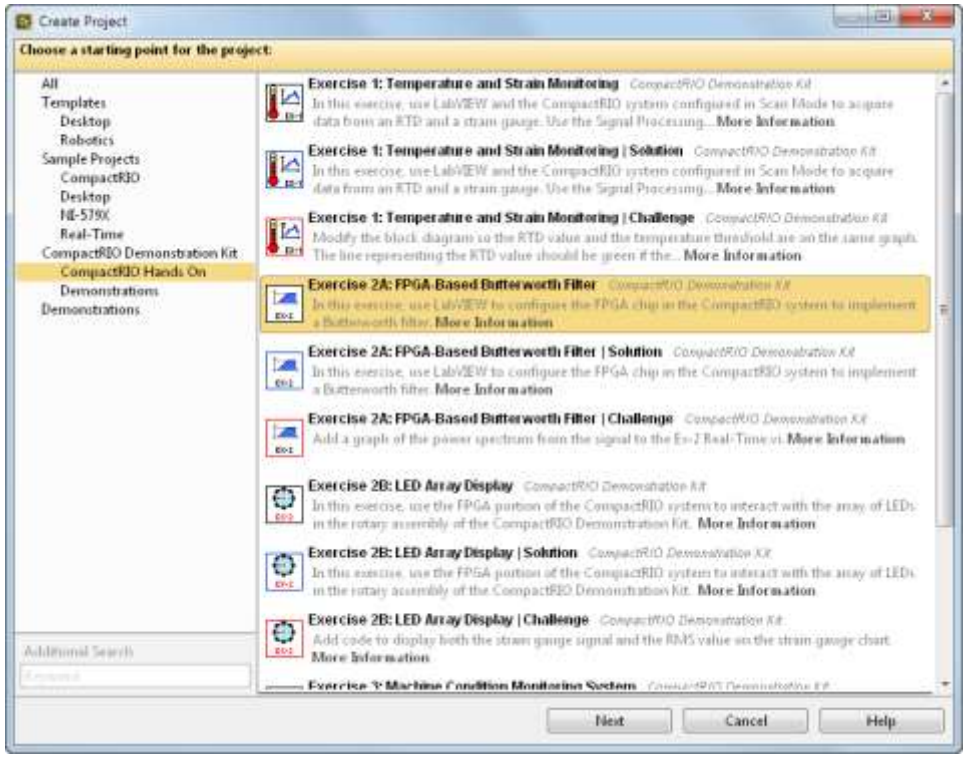
Outputs

Slot 2: NI 9401—*8 Ch, 5 V/TTL High-Speed Bidirectional Digital I/O Module*
Digital I/O 4 (DIO4)→LED0

Part B—CODE IMPLEMENTATION

1. Create an Instance of the Exercise 2A: FPGA-Based Butterworth Filter Sample Project

Launch LabVIEW and create a new LabVIEW project from the Exercise 2A: FPGA-Based Butterworth Filter sample project.



DETAILED INSTRUCTIONS

In this exercise, use an existing ***LabVIEW sample project*** as the starting point of the application.

- ***Launch LabVIEW***
  Launch LabVIEW by navigating to ***Start » All Programs » National Instruments » LabVIEW <Year>» LabVIEW.***

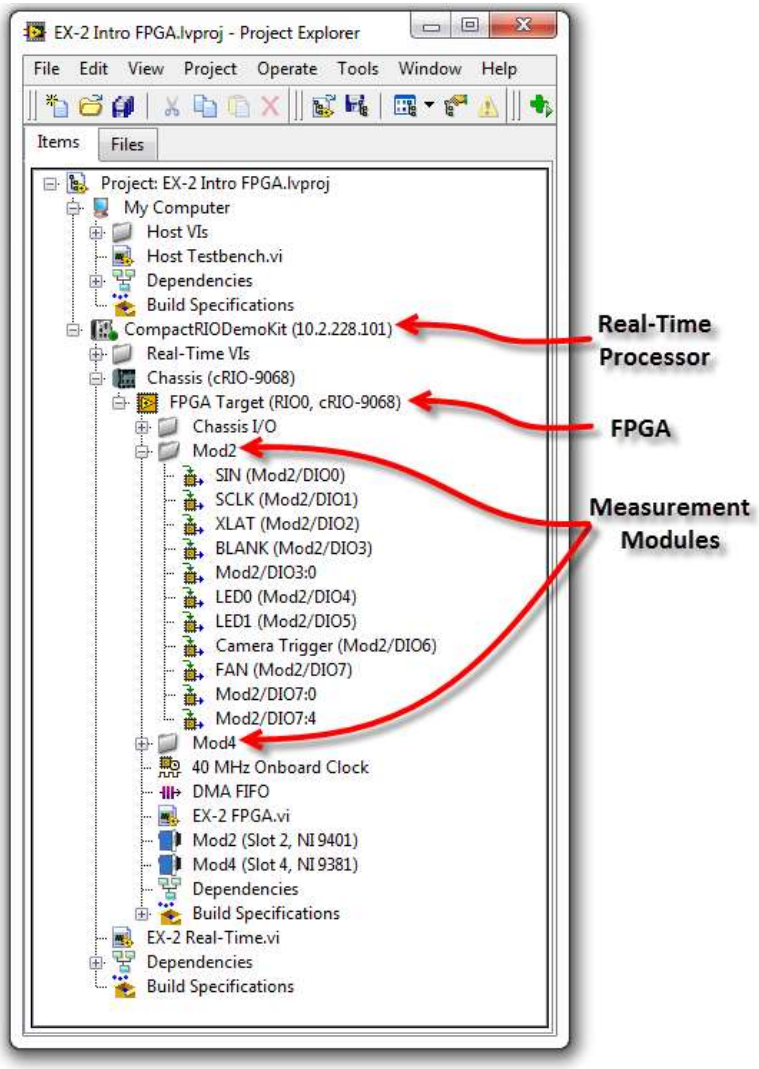
- ***Create an Instance of the Existing LabVIEW Project***
  Click on the ***Create Project*** button on the right side of the main screen of ***LabVIEW*** and create an instance of the **Exercise 2A: FPGA-Based Butterworth Filter** sample project located at:

CompactRIO Demonstration Kit>> CompactRIO Hands On

Click **Next** to customize the **Project Name** and **Project Root** if desired. Finally, click **Finish** to create the project.

2. Explore the project and reveal its components.

Expand each hierarchy in the LabVIEW Project Explorer to reveal the real-time processor and the FPGA of the CompactRIO system.



DETAILED INSTRUCTIONS

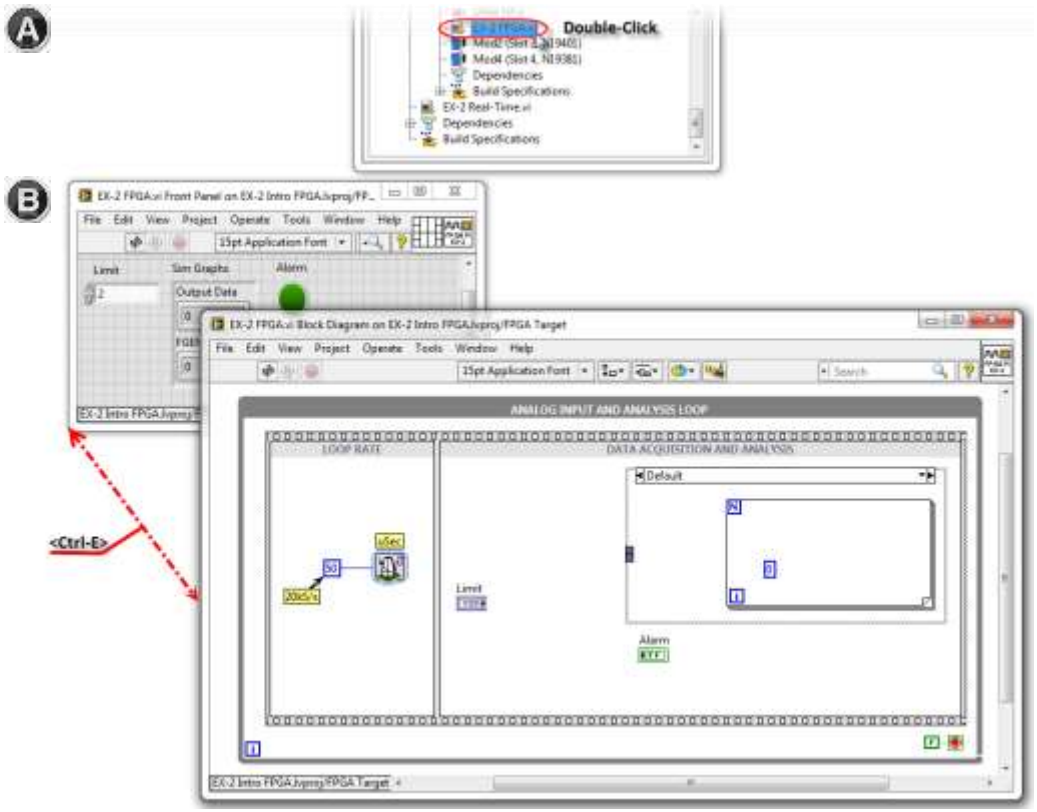
- ***Expand the Hierarchies*** **Exercise 2A: FPGA-Based Butterworth Filter sample project**
  This project already contains the **CompactRIO** system exposing the **FPGA** and the measurement modules that you are going to use.

  Notice that the measurement modules are located under the FPGA target in the hierarchy.

  To reveal the components of the project, expand the hierarchies in the project tree by clicking on the "**+**" boxes.

3. Open the Ex-2 FPGA.vi and show its block diagram.

The Ex-2 FPGA.vi is located under the FPGA target in the hierarchy. This VI already contains timing code to use for this exercise.



DETAILED INSTRUCTIONS

- ***Open the Ex-2 FPGA.vi***
  Double-click on the file named **Ex-2 FPGA.vi** located under the FPGA target in the hierarchy as shown in ***Figure A***.

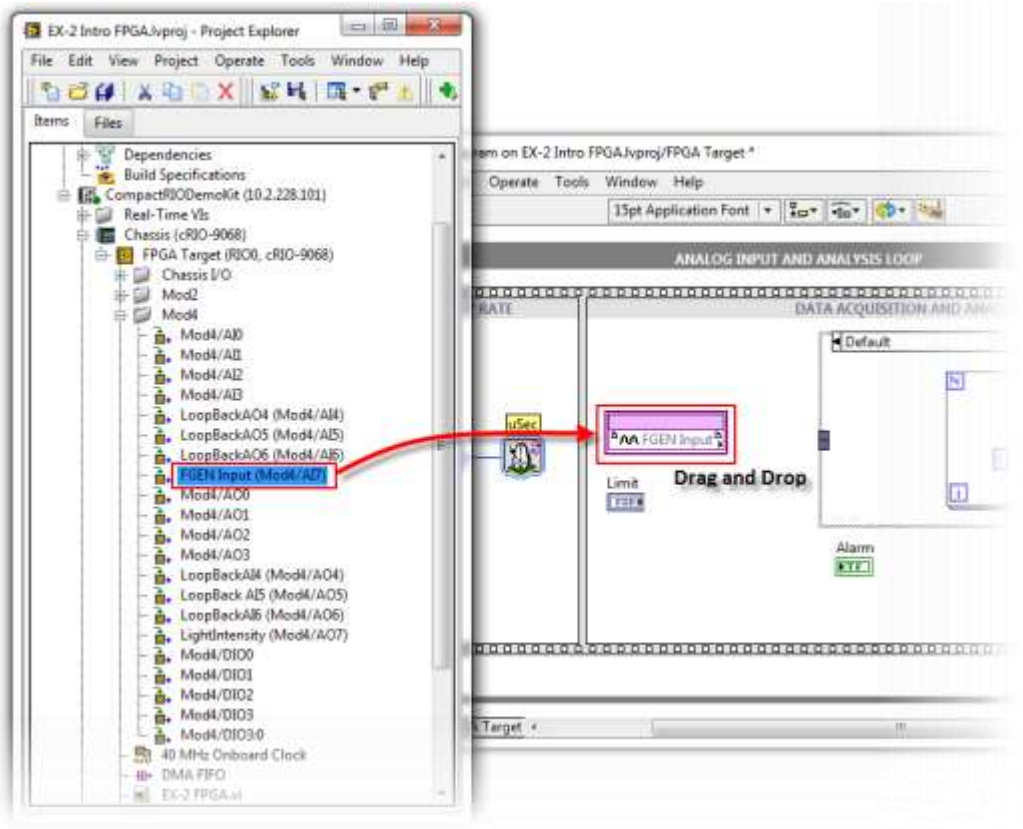
- ***Show the Block Diagram***
  Press ***<Ctrl-E>*** or click on ***Window » Show Block Diagram*** to display or toggle between the block diagram and the front panel as shown in ***Figure B***.

  The block diagram contains a ***While Loop*** with a ***sequence structure*** embedded to enforce loop timing.

  Program this VI to acquire input from a ***function generator*** and write the analog waveform data to memory that can be accessed by the ***real-time operating system*** on the ***CompactRIO***.

4. Add the Function Generator Channel to the block diagram.

Drag and drop the Function Generator Input node of the NI 9381 C Series measurement module (Mod4) into the sequence structure.

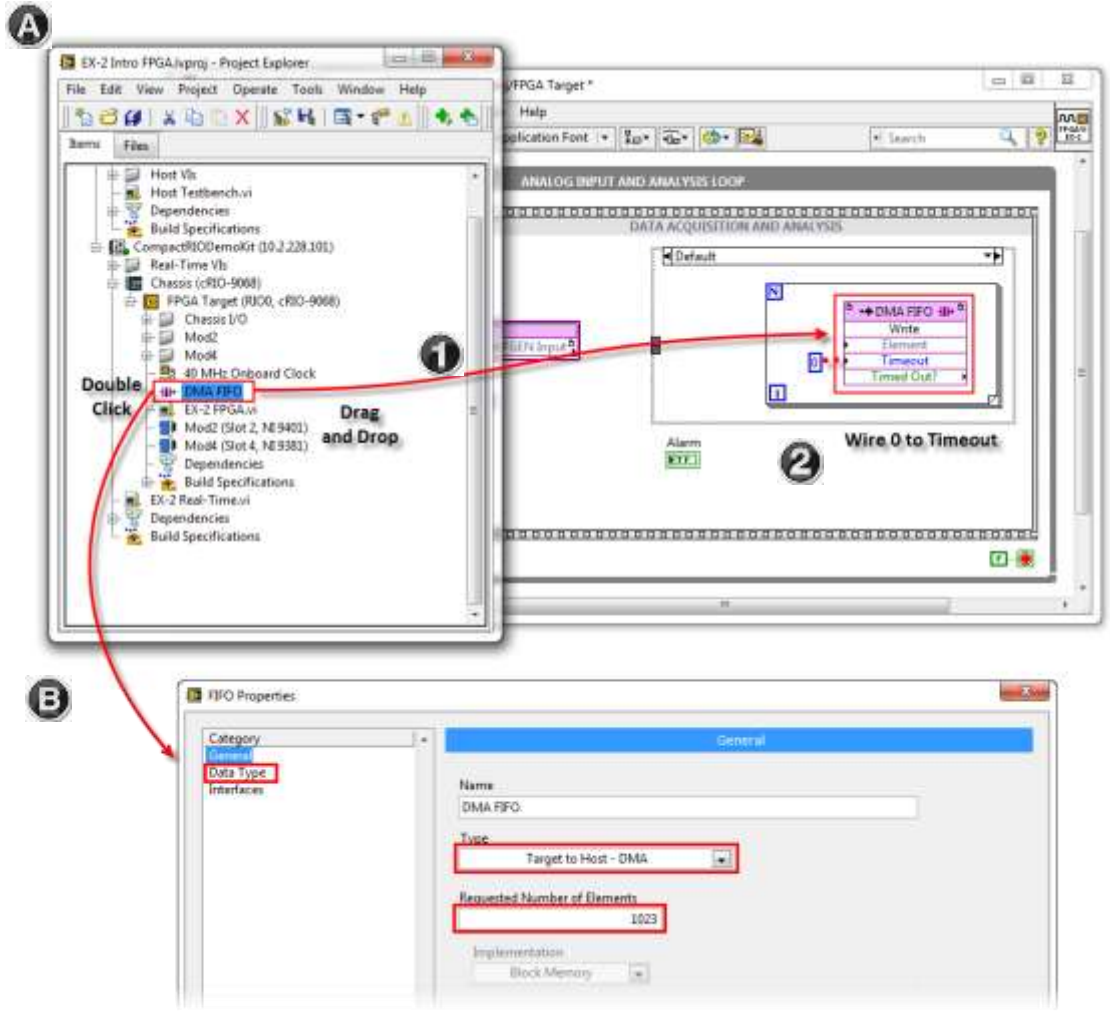


DETAILED INSTRUCTIONS

- ***Drag and Drop the FGEN Input (Mod4/AI7) Into the Sequence Structure***
  Locate the ***FGEN Input (Mod4/AI7)*** terminal located in the LabVIEW project under the **Mod4** measurement module of the FPGA target in the LabVIEW Project Explorer. Drag and drop the terminal onto the block diagram and place it inside the ***Data Acquisition and Analysis*** section of the sequence structure.

5. Add the communication channel between the FPGA and the real-time processor.

With the DMA FIFO, you can transfer data from the FPGA to the real-time processor**.** Drag and drop the DMA FIFO from the LabVIEW Project Explorer into the For Loop.



DETAILED INSTRUCTIONS

- ***Drag and Drop the DMA FIFO Onto the Block Diagram***
  Drag and drop the ***DMA FIFO*** node onto the block diagram and place it inside the ***For Loop*** as shown in ***Figure A1***.

- ***Wire the Constant With Value 0 to the Timeout Terminal of the DMA FIFO as Shown in Figure A2***
  A constant with value 0 indicates that the method cannot wait for space to be available in the FIFO before it attempts to write data to the FIFO.

  The ***DMA FIFO*** has been preconfigured as follows:

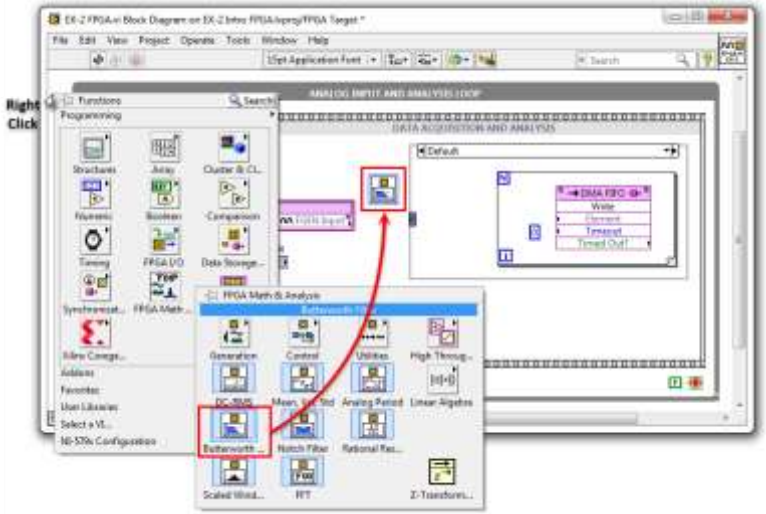
  - Target to Host-DMA
  - 1023 Elements
  - Fixed-Point Data Type

  Double-click on the ***DMA FIFO*** to open its configuration window as shown in ***Figure B***.

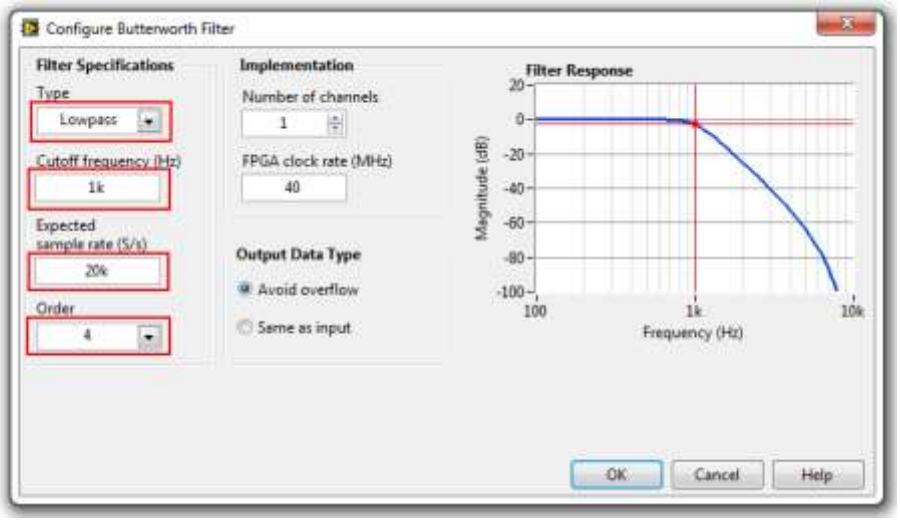
6. Filter the data from the Function Generator Input.

Filter the data from the Function Generator Input by sending it through a Butterworth filter configured to be a lowpass filter.

A



B



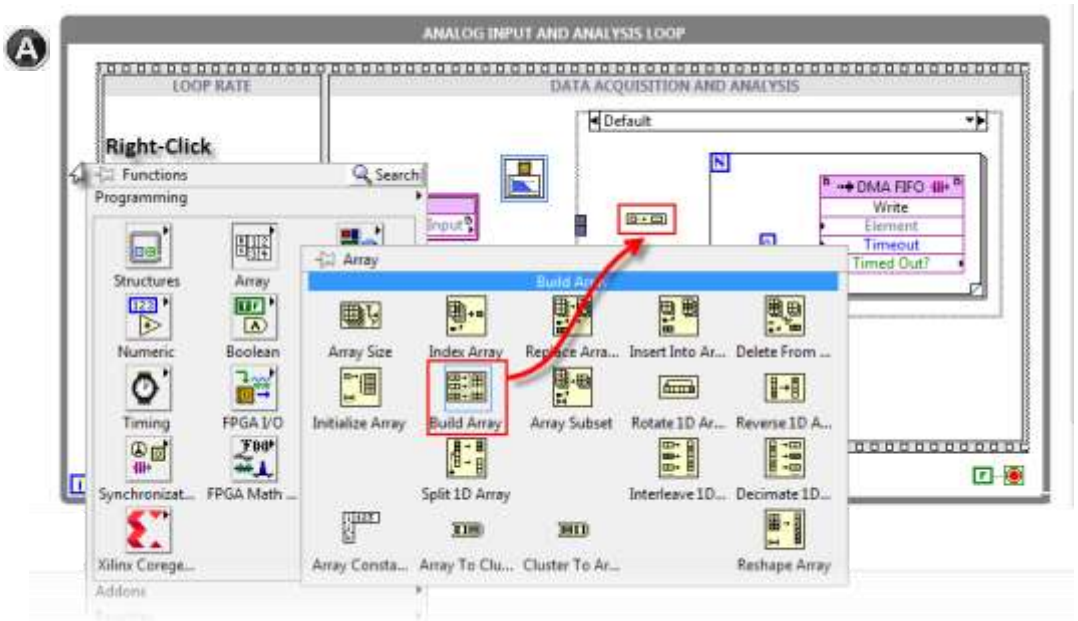
DETAILED INSTRUCTIONS

- ***Add a Butterworth Filter Function to the Block Diagram***
  Right-click on the block diagram to open the ***Functions Palette***. Navigate to ***Programming » FPGA Math & Analysis***. Drag and drop the ***Butterworth Filter*** function onto the ***block diagram*** as shown in ***Figure A***. The configuration window automatically displays. Configure the function as shown in ***Figure B*** using the following parameters:

  Type: Lowpass
  Cutoff frequency (Hz): 1 k
  Expected Sample Rate (S/s): 20k
  Order: 4

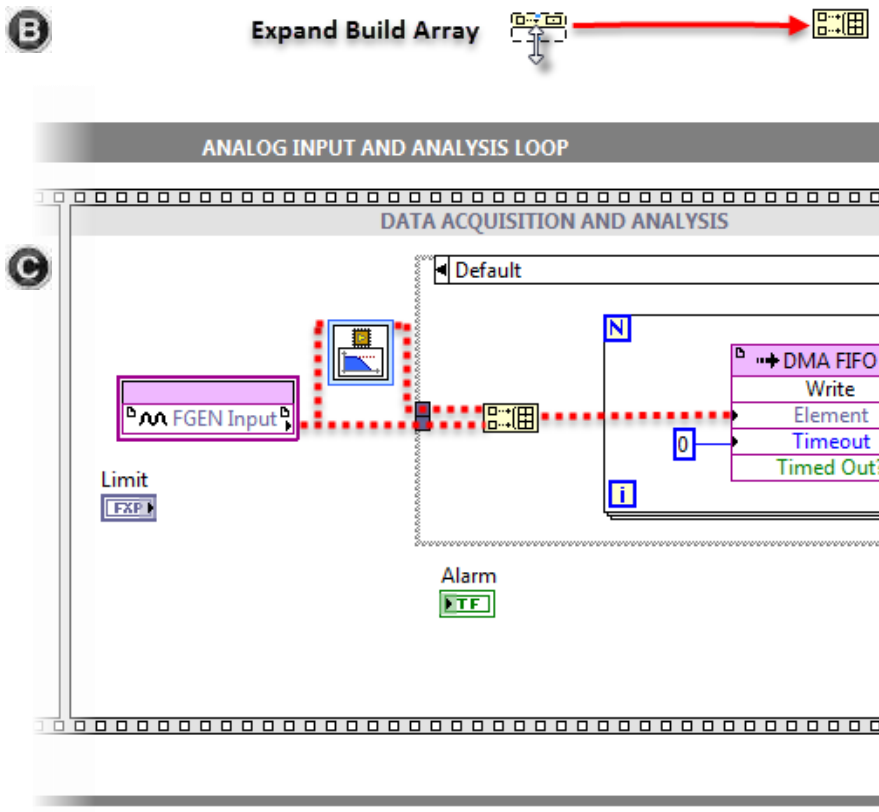
7. Send the data and filtered data to the DMA FIFO.

Send signal data and filtered data from the FPGA FGEN Input to the DMA FIFO.



DETAILED INSTRUCTIONS

- ***Add a Build Array Function to the Block Diagram***
  Right-click on the block diagram to open the ***Functions Palette***. Navigate to ***Programming » Array***. Drag and drop the ***Build Array function*** onto the block diagram as shown in ***Figure A***.

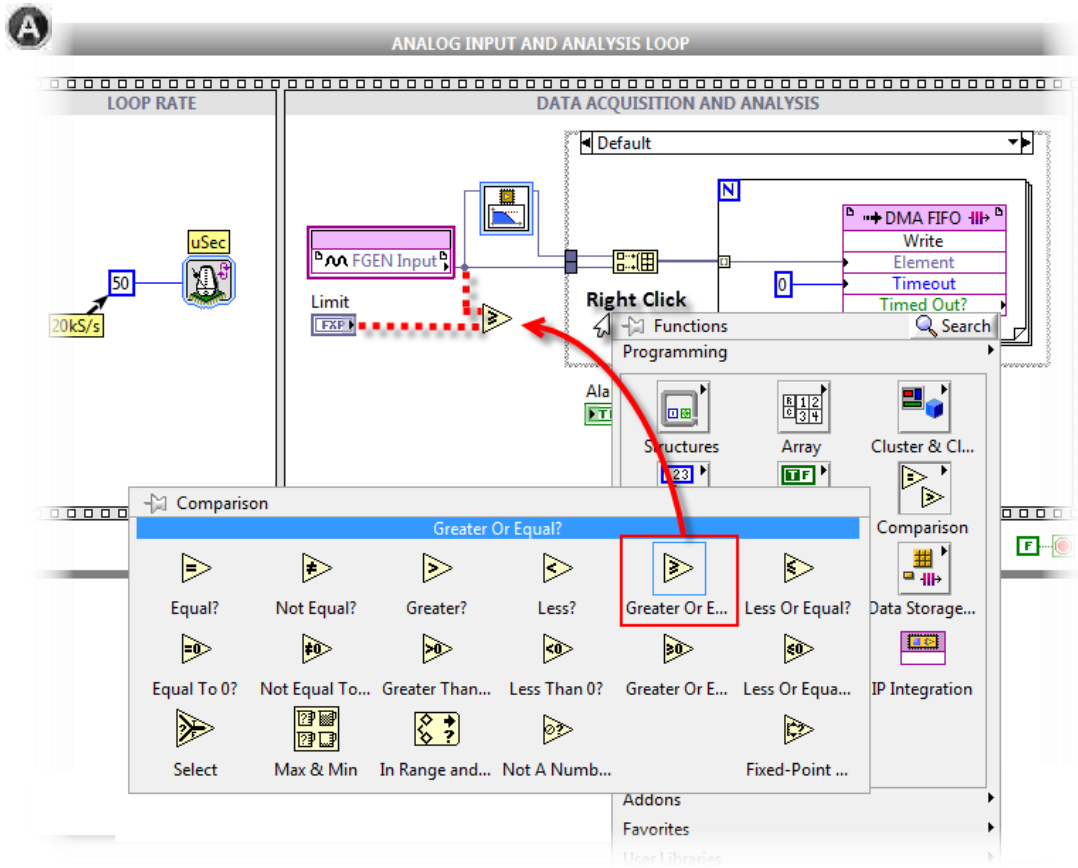


- ***Expand the Build Array Function to Show Two Input Terminals***
  Move the cursor over the ***Build Array function*** to show the blue points from which you can expand the node. Expand the node to include an additional terminal as shown in ***Figure B***.

- ***Wire the Filtered and Unfiltered Signals Into the Build Array Function and Wire the Output Into the DMA FIFO Element Node as Shown in Figure C***
  Connect the ***FGEN Input*** node to the ***Build Array function*** and ***Butterworth Filter*** input. Connect the output of the ***Butterworth Filter*** to the ***Build Array function***. Wire the array into the ***DMA FIFO Element node***.

8. Create an alarm if the data exceeds a preset limit.

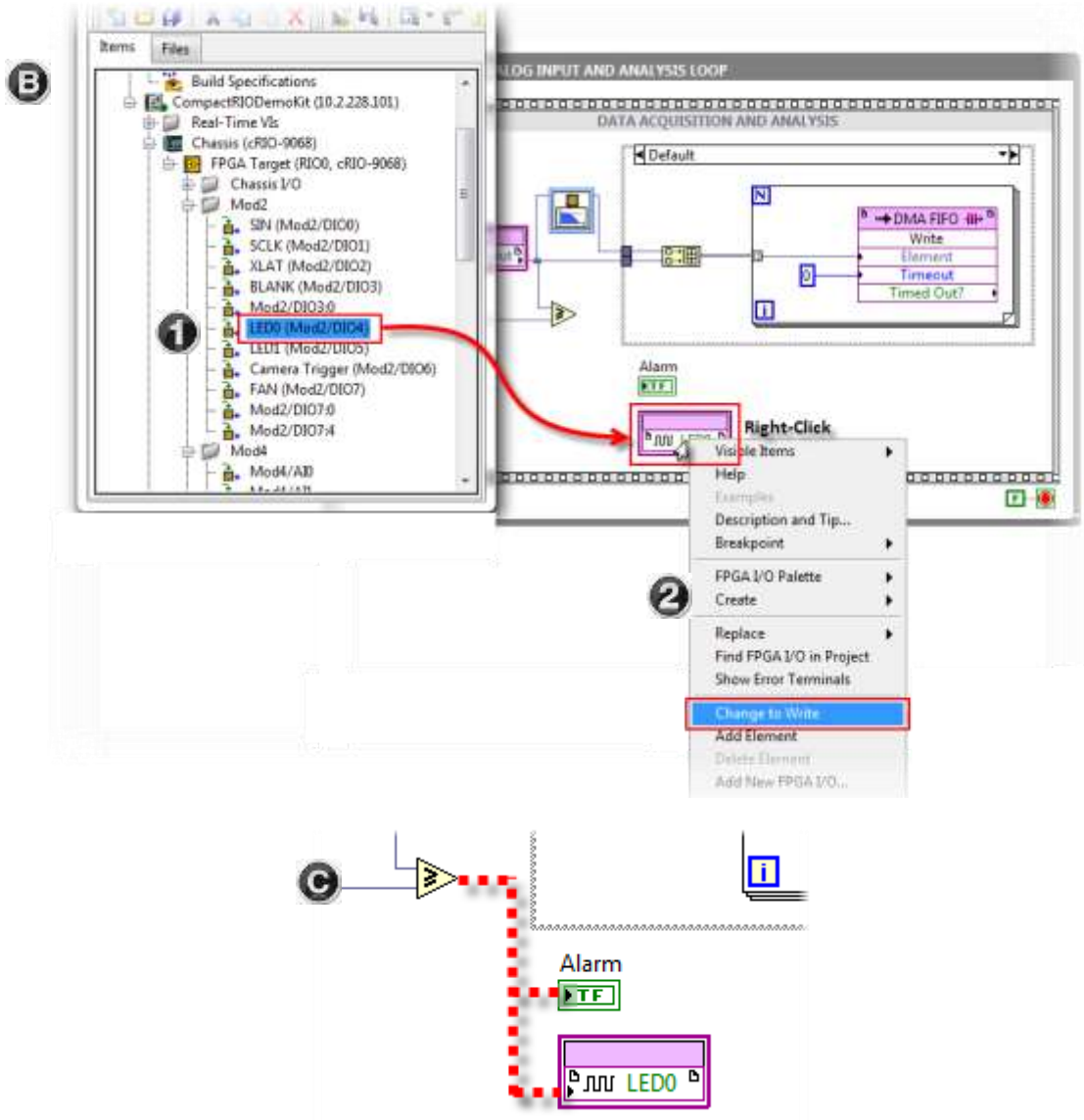
Configure an LED alarm on the user interface and CompactRIO to light up if the signal exceeds a user-set limit.



DETAILED INSTRUCTIONS

- ***Add a Greater Or Equal? Function to the Block Diagram***
  Right-click on the block diagram to open the ***Functions Palette***. Navigate to ***Programming » Comparison***. Drag and drop the ***Greater Or Equal? function*** onto the ***block diagram*** as shown in ***Figure A***.

- ***Wire the FGEN Input Signal and the Limit Control Into the Greater Or Equal? Function as Shown in Figure A***
  Create a new branch from the wire produced in the previous step to connect the ***FGEN Input*** terminal to the ***Greater Or Equal? function***. Connect the output of the ***Limit Control*** to the ***Greater Or Equal? function***.



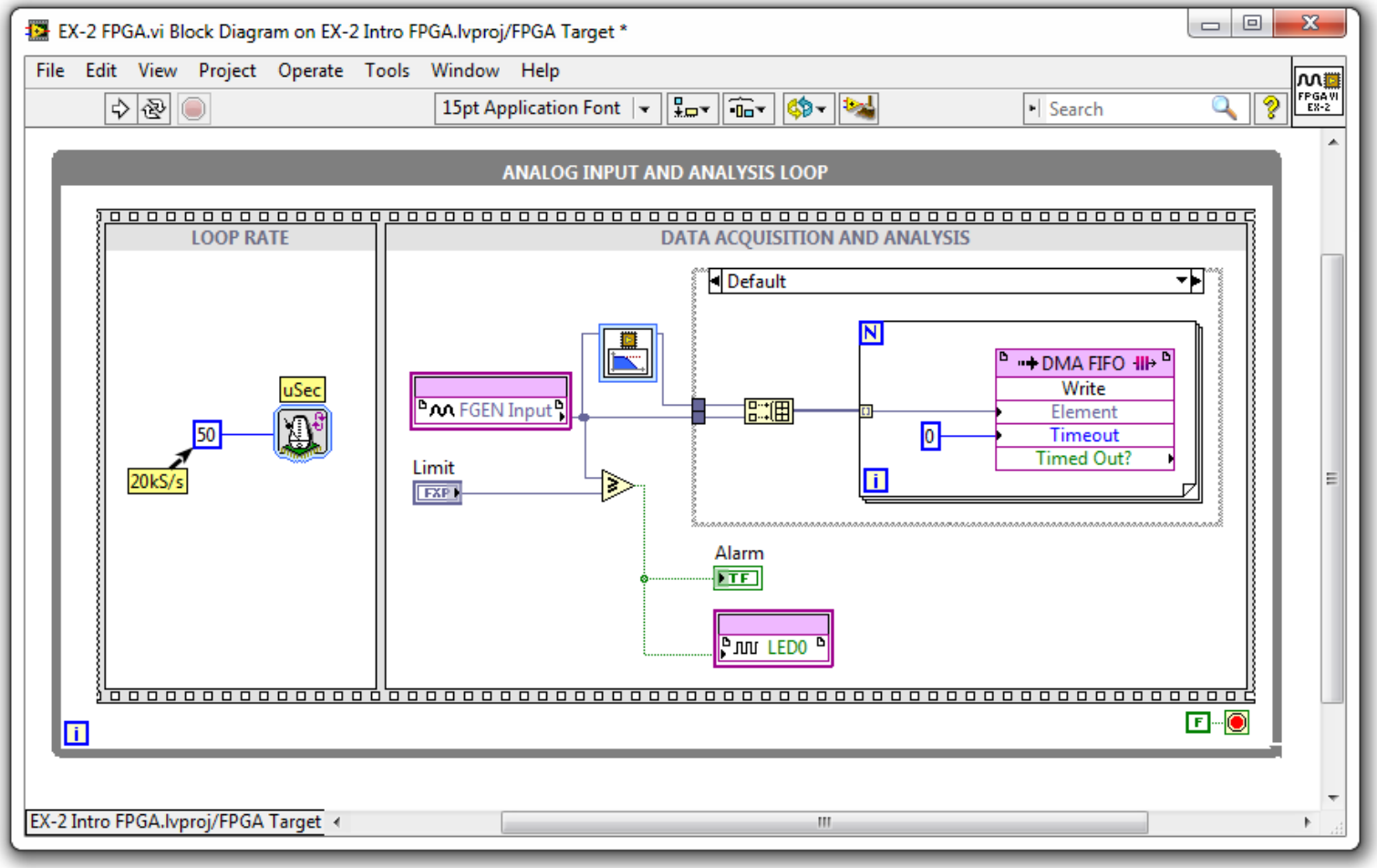
- ***Drag and Drop an LED0 Node Into the Sequence Structure as Shown in Figure B1***
  Drag and drop the ***LED0*** terminal located in the LabVIEW Project Explorer under the Mod2 measurement module of the FPGA target. Place it beneath the ***Alarm*** indicator.

- ***Change the LED0 Input to Write as Shown in Figure B2***
  Right-click on the LED0 text within the node and select **Change to Write**.

- ***Wire the Greater Or Equal? Function Output to the Alarm Indicator and the LED0 Node as Shown in Figure C***
  Wire the output from the ***Greater Or Equal? function*** to the ***Alarm*** indicator. Create a new branch from the wire to connect to the ***LED0*** input.

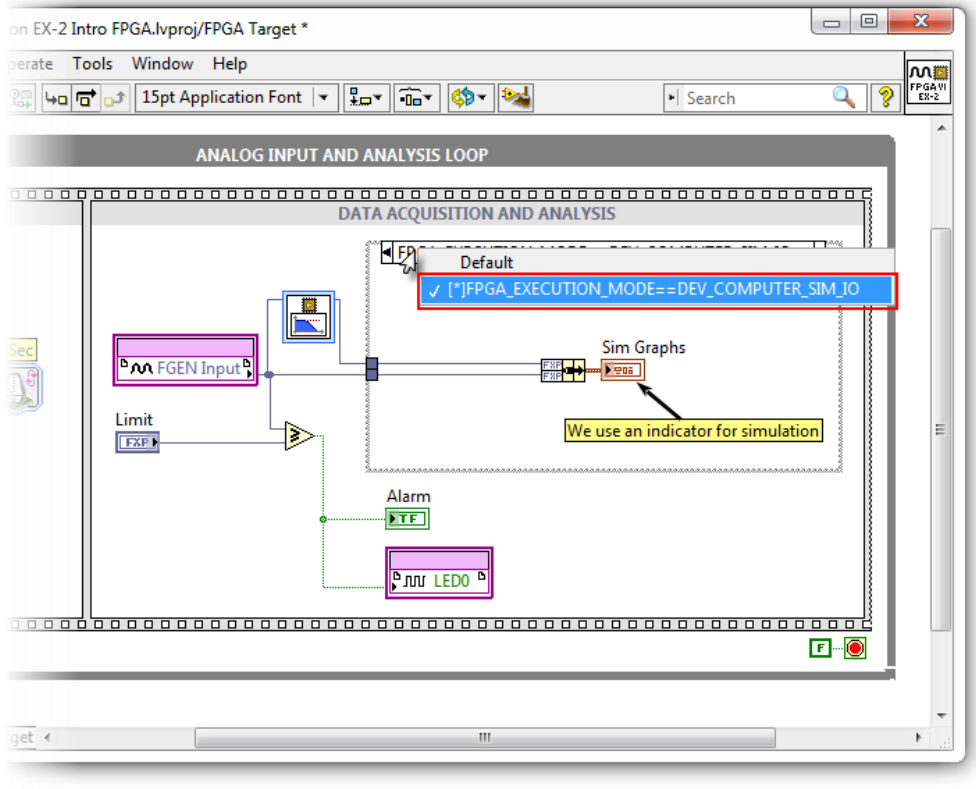
9. The completed block diagram for Ex-2 FPGA.vi should look like the image below.

Be sure to save your work by pressing <Ctrl-S> or clicking File » Save.



10. Observe the simulated case in the Conditional Disable structure.

Create a Host Testbench Application in Part C to test your FPGA code in a simulated mode without having to compile it. Observe the code that enables you to do this within the Ex-2 FPGA.vi.



DETAILED INSTRUCTIONS

- ***View the Simulated Case Within the Conditional Disable Structure***
  Click on the ***Default*** text on the menu bar of the ***Conditional Disable structure*****.** From the pull down menu, select:

FPGA_EXECUTION_MODE==DEV_COMPUTER_SIM_IO

- ***Observe the Graph Indicator Used for Simulation***
  Rather than passing the data onto a memory buffer (***DMA FIFO***), view the simulated data on a graph. Bundle the filtered and unfiltered data together and display it on a graph called ***Sim Graph***.

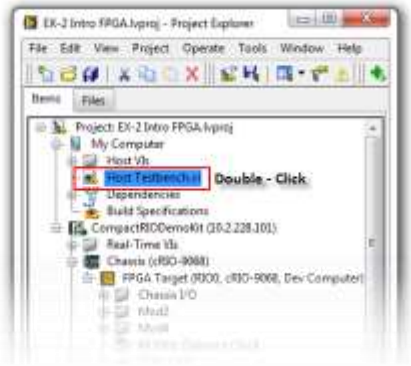
  This code runs when the ***FPGA Target*** is set to run in ***simulation mode*** on the development computer rather than on the actual FPGA hardware. You will configure the simulation mode when you run the ***Host Testbench*** application in ***Part D***. This section is grayed out while the FPGA target is not set to run in simulation mode.

Part C—HOST TESTBENCH APPLICATION (FPGA SIMULATION)

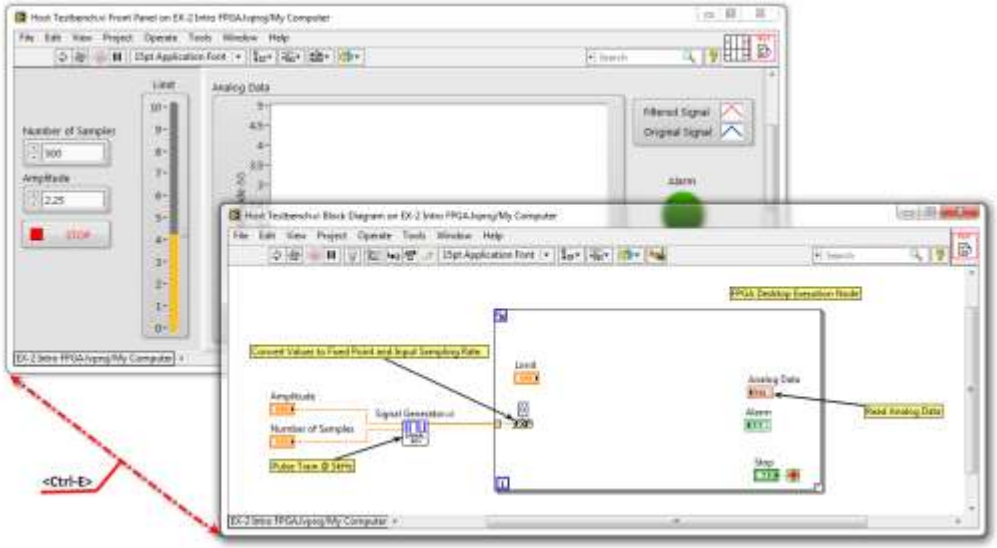
11. Open the Host Testbench.vi and show its block diagram.

This VI is located under My Computer in the LabVIEW Project Explorer, and it already contains some code that you will use for this application.

A



B



DETAILED INSTRUCTIONS

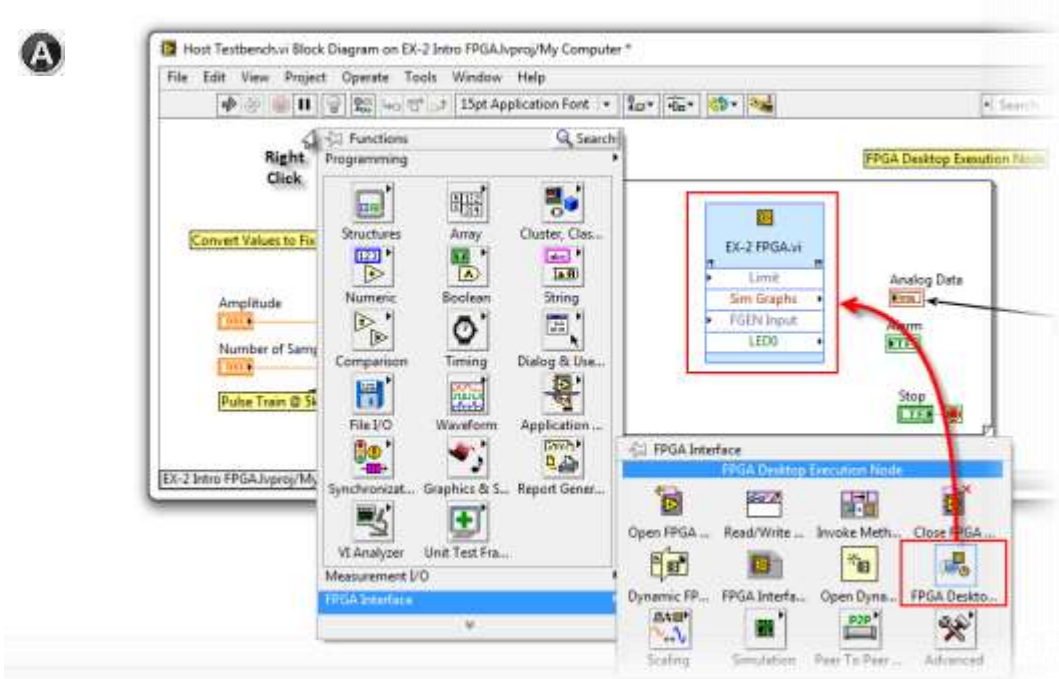
- ***Open the Host Testbench.vi***
  Double-click on the file named ***Host Testbench.vi*** located under My Computer in the LabVIEW Project Explorer as shown in ***Figure A***.

- ***Show the Block Diagram***
  Press ***<Ctrl-E>*** or click on ***Window » Show Block Diagram*** to open or toggle between the block diagram and the front panel as shown in ***Figure B***.

  The VI contains a conditional For Loop, signal generator, and some controls and indicators for the user interface.

12. Modify Host Testbench.vi to simulate the Ex-2 FPGA.vi on the development computer.

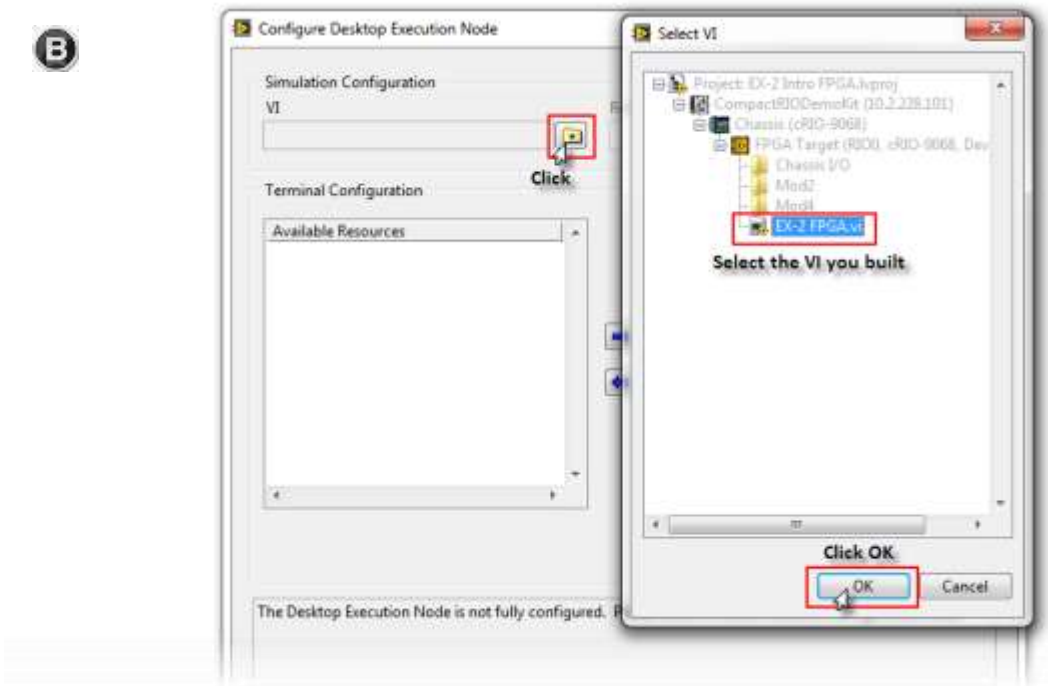
Use an FPGA Desktop Execution Node to run the FPGA VI from a host computer.

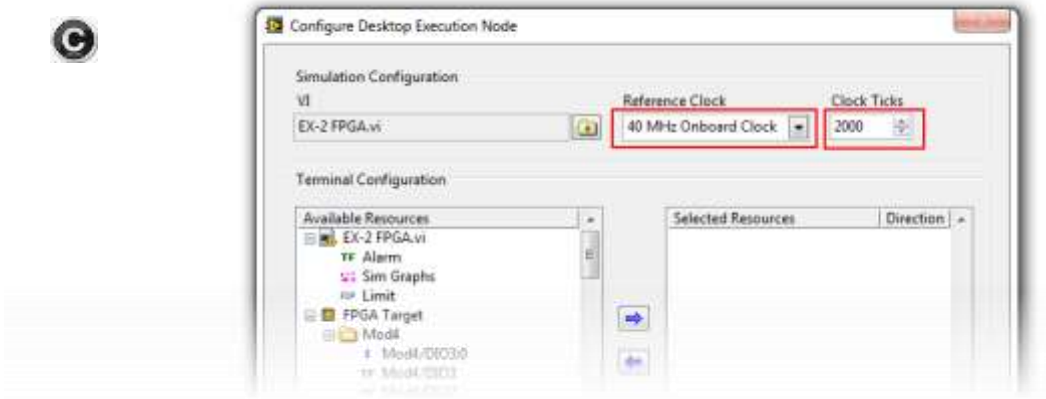


DETAILED INSTRUCTIONS

- ***Add an FPGA Desktop Execution Node to the Block Diagram as Shown in Figure A***
  Right-click on the block diagram to open the ***Functions Palette***. Navigate to **FPGA Interface** and drag and drop an ***FPGA Desktop Execution Node*** onto the block diagram.

  Once you place this function on the block diagram, the configuration window displays automatically. Configure the function as shown in ***Figure B.***





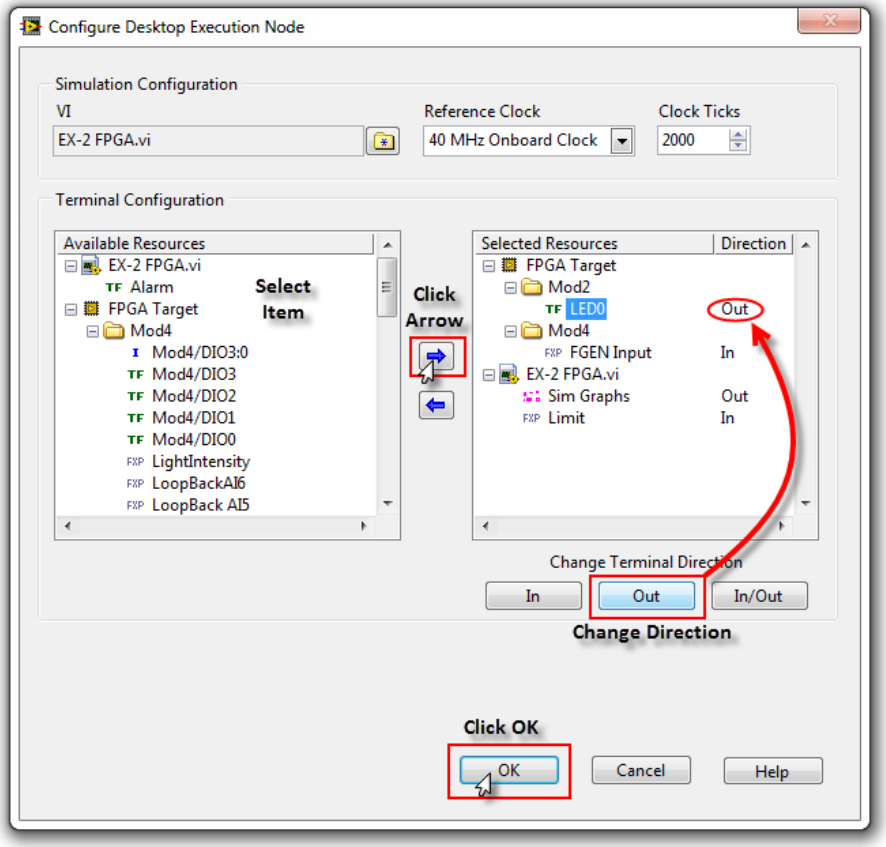
DETAILED INSTRUCTIONS

- ***Configure FPGA Desktop Execution Node to Run Ex-2 FPGA.vi as Shown in Figure B***
  In the configuration window, click the file icon next to the VI category. Select ***Ex-2 FPGA.vi*** from the menu and click ***OK***.

- ***Configure the Timing Settings for the FPGA Desktop Execution Node as Shown in Figure C***
  Configure the timing settings as shown in ***Figure C*** using the following parameters:

  Reference Clock: 40 MHz Onboard Clock
  Clock Ticks: 2000

  To achieve the expected sample rate of ***20 kS/s***, there must be ***2000 clock ticks*** of the 40 MHz onboard clock.



DETAILED INSTRUCTIONS

- ***Configure the Terminals for the FPGA Desktop Execution Node as Shown in Figure D***

  Select the following four items in the **Available Resources** table and click the **Arrow** to send them to **Selected Resources** one at a time:

  Ex-2 FPGA.vi:
  1. Limit
  2. Sim Graphs

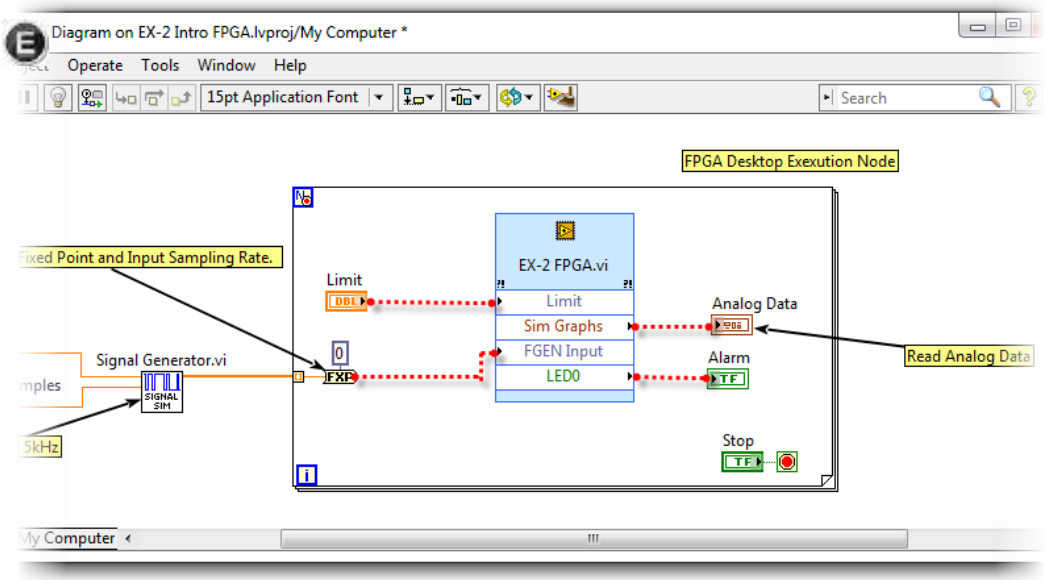
  FPGA Target\Mod4:
  3. FGEN Input

  FPGA Target\Mod2:
  4. LED0

- ***Change the Direction of the LED0 Terminal FPGA Desktop Execution Node as Shown in Figure D***

  Select ***LED0*** in the **Selected Resources** table. Click **Out** to change the terminal direction. Use the default directions for the other three terminals.

- ***Save Your Changes by Clicking OK***



DETAILED INSTRUCTIONS

- ***Wire the FPGA Desktop Execution Node According to Figure E***

  Connect the ***Limit*** control to the **Limit** terminal.

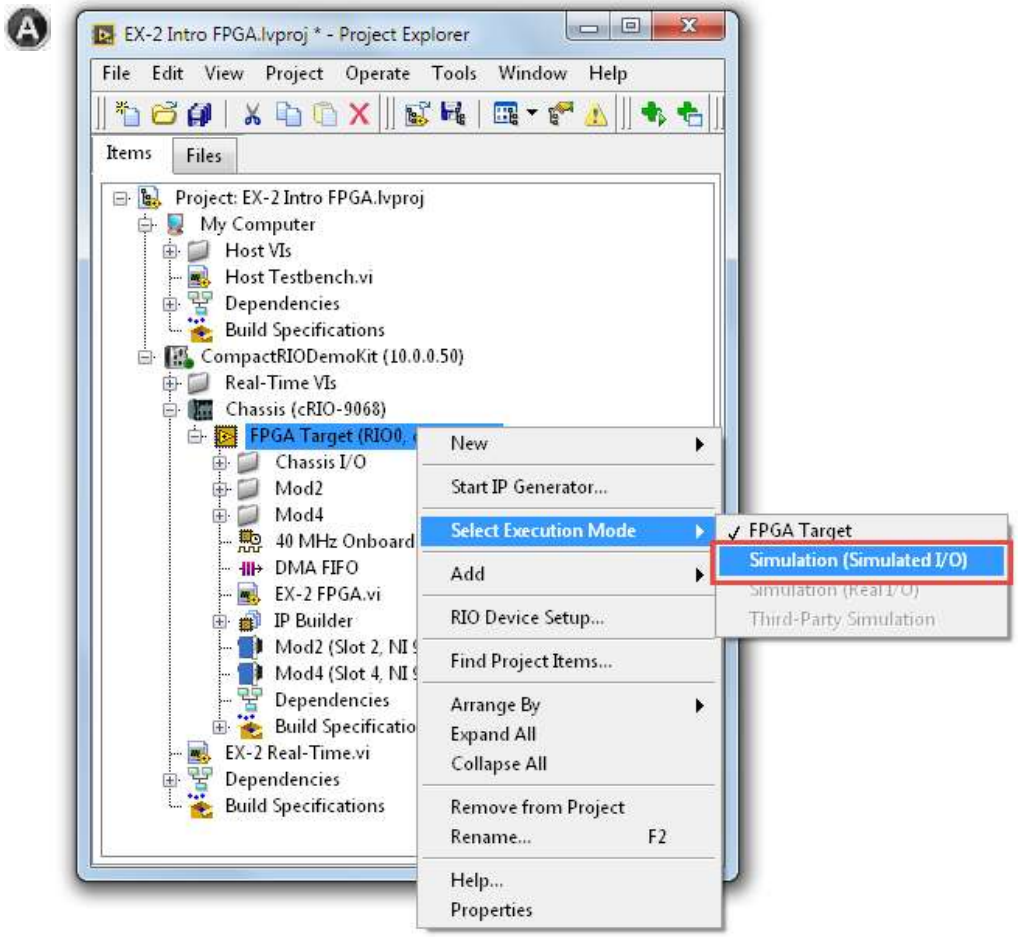
  Connect the **Signal Generator.vi** output that has been converted to the fixed point data type to the **FGEN Input** terminal.

  Connect the **Sim Graphs** terminal to the ***Analog Data*** indicator.

  Connect the **LED0** terminal to the ***Alarm*** indicator.

13. Set the FPGA target to run in simulated mode.

Set the FPGA target to run in simulated mode on the development computer rather than on the FPGA target.

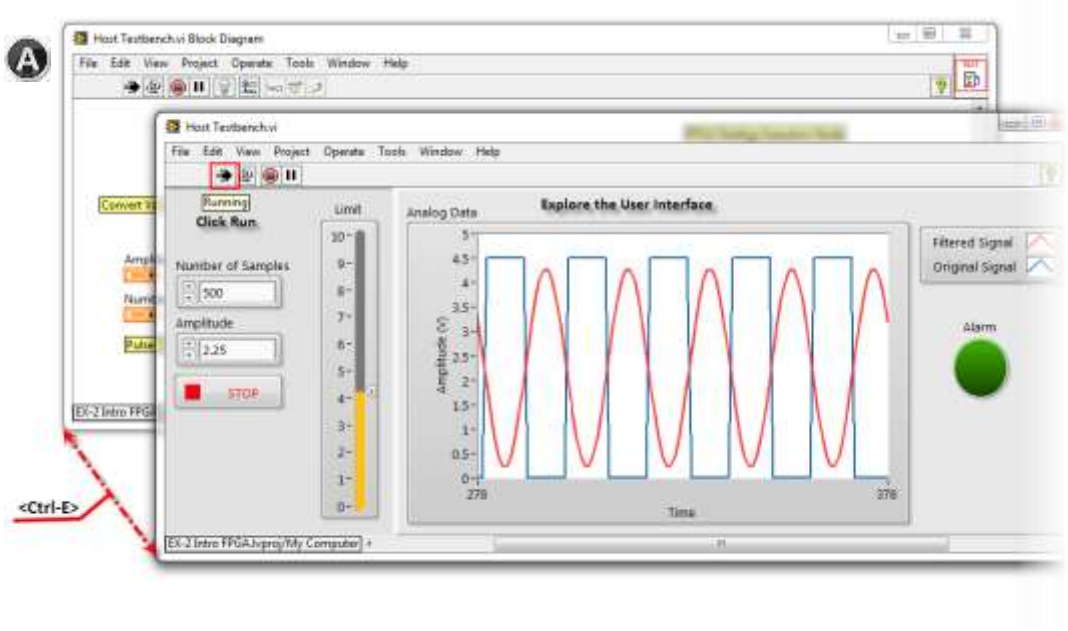


DETAILED INSTRUCTIONS

- ***Configure the FPGA Target to Run in Simulated Mode as Shown in Figure A***
  Right-click on the **FPGA Target** in the LabVIEW Project Explorer. In the **Execute VI on** category, choose **Simulation**.

14. Verify the FPGA code by running the host testbench application.

Run the **Host Testbench.vi** to verify that the FPGA code behaves correctly before compiling the FPGA code.



DETAILED INSTRUCTIONS

- ***Run Host Testbench.vi as Shown in Figure A***
  Switch back to the front panel of **Host Testbench.vi** by pressing ***<Ctrl-E>*** and run the program by clicking on the arrow button at the top of the front panel.

  Adjust the **Number of Samples** to control how long the **signal generator** runs.

  Adjust the **Amplitude** to change the amplitude of the **signal generator**.

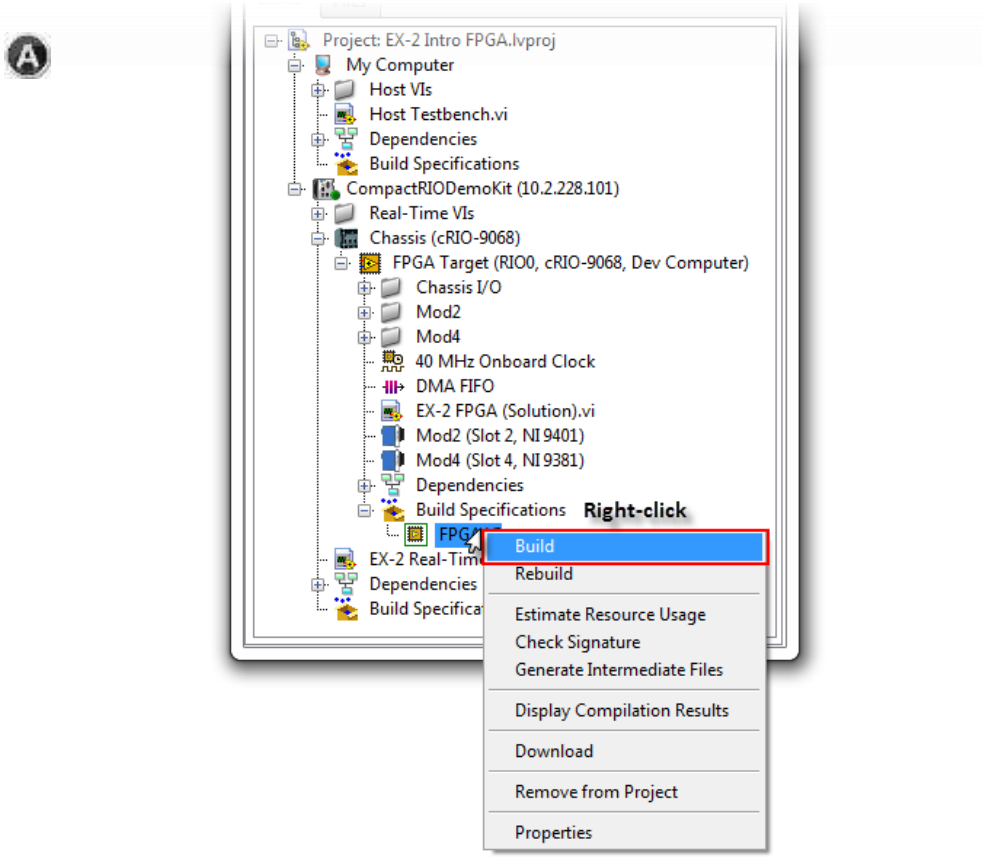
  Adjust the limit that triggers the **Alarm** LED by moving the slide control up or down.

  Using the **FPGA Desktop Execution Node**, you could verify that the FPGA programming executes as expected prior to compiling the FPGA code. This can help save development time because compiling FPGA code is a slow process.

Part D—FPGA COMPILATION PROCESS

15. Save and compile the EX-2 FPGA.vi.

The compilation process of the EX-2 FPGA.vi takes about 30 minutes. While the code is compiling, continue working on the real-time portion of the application in **Part E**.

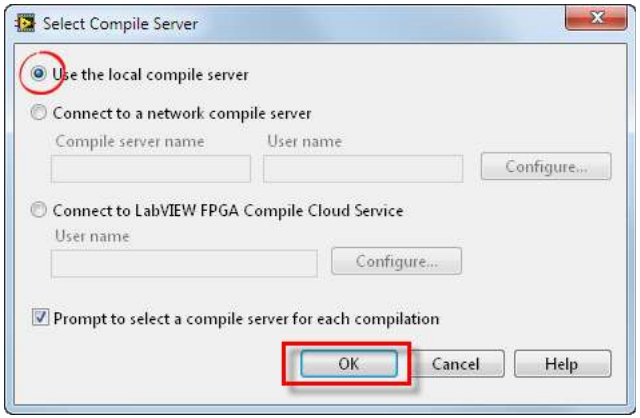


DETAILED INSTRUCTIONS

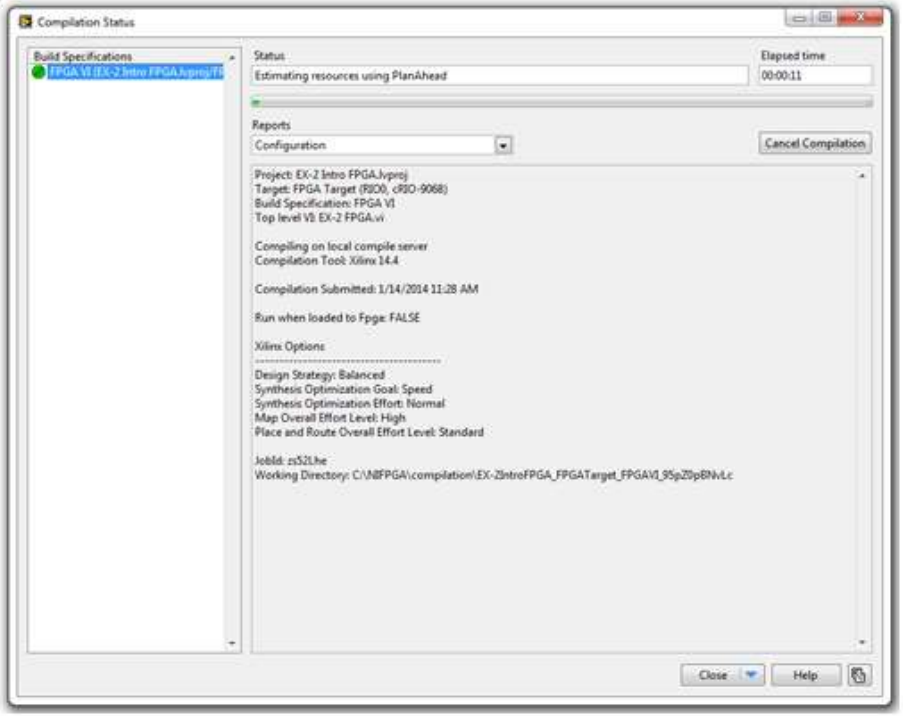
- ***Save the EX-2 FPGA.vi***
  Press ***<Ctrl-S>*** or go to ***File » Save.***

- ***Compile the EX-2 FPGA.vi***
  Navigate to the ***Build Specifications*** category under the FPGA hierarchy. Expand this category and right-click on the existing ***EX-2 FPGA*** and select ***Build*** as shown in ***Figure A***.

B



C



DETAILED INSTRUCTIONS

LabVIEW prompts you to select the ***compile server***. Select the ***local compile server*** and click ***OK*** as shown in ***Figure B***. This will start the compilation process.

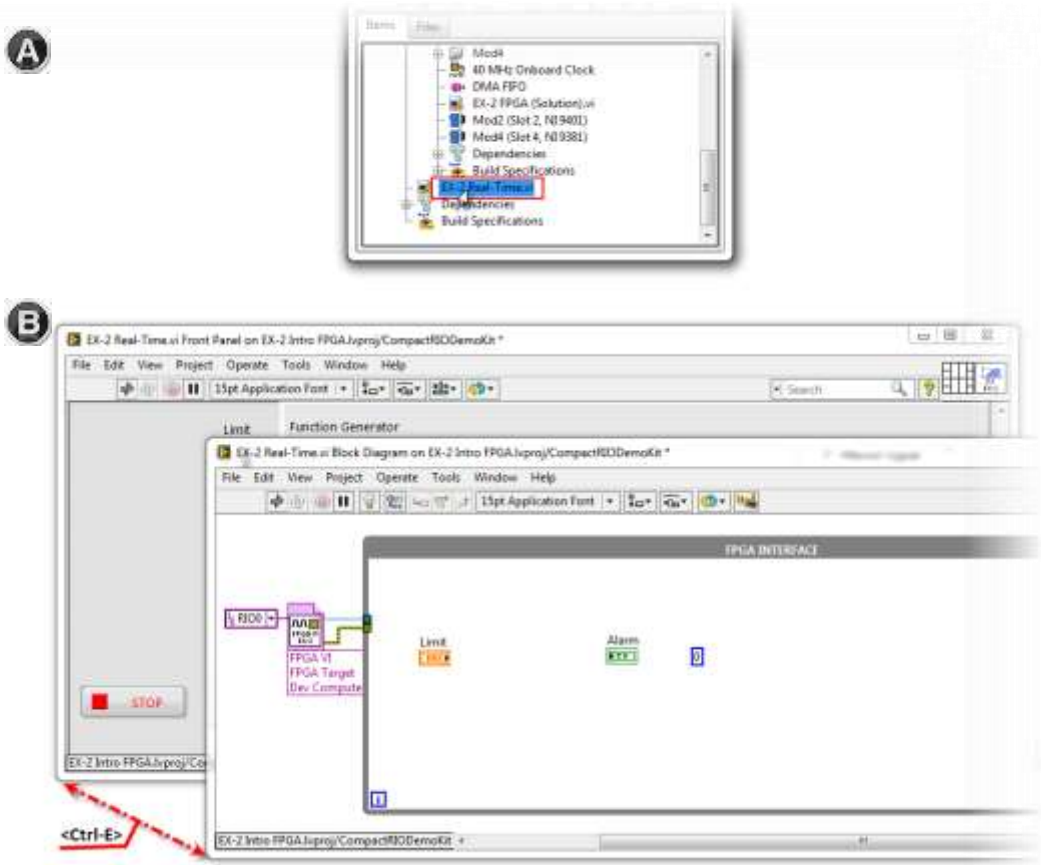
During the compilation process, ***LabVIEW*** generates intermediate HDL files that are later processed by the ***Xilinx Compiler,*** which outputs a bitfile containing the placing and routing information of the FPGA design.

The compilation process for this VI takes **about 15 minutes**. Once you have reached the Compilation Status window, as shown in ***Figure C***, ***PROCEED TO PART E*** to start developing the real-time portion of the application.

Part E—REAL-TIME CODE IMPLEMENTATION

16. Open the Ex-2 Real-Time.vi and show its block diagram.

This VI is located under the CompactRIO hierarchy, and it already contains some code that you will use for this application.



DETAILED INSTRUCTIONS

- ***Open the Ex-2 Real-Time.vi***
  Double-click on the file named **Ex-2 Real-Time.vi** located under CompactRIO in the LabVIEW Project Explorer as shown in ***Figure A***.

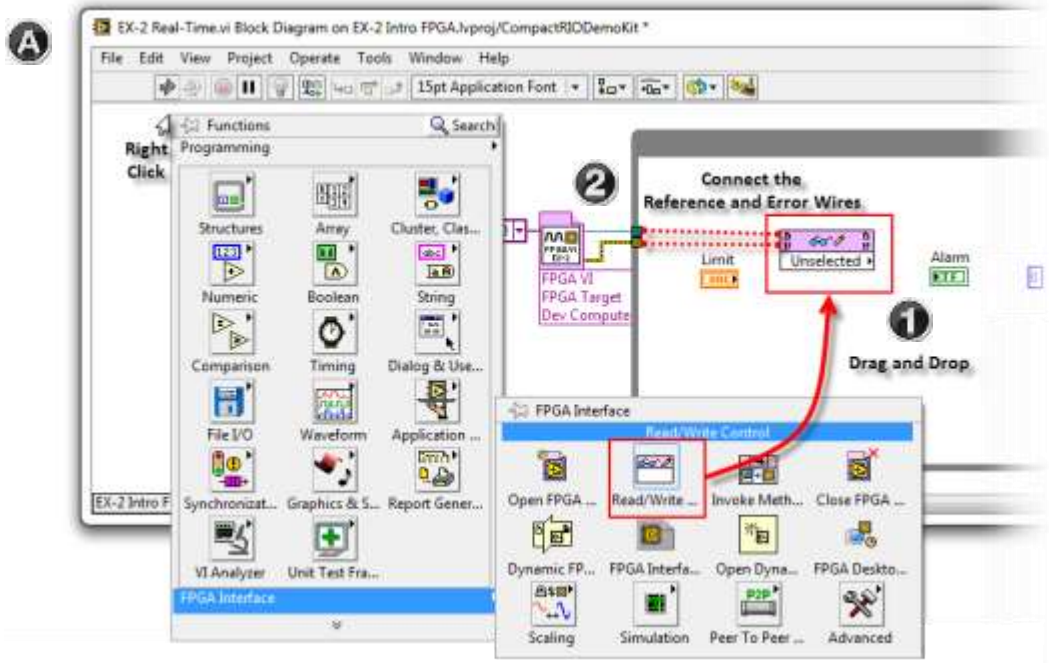
- ***Show the Block Diagram***
  Press ***<Ctrl-E>*** or click on ***Window » Show Block Diagram*** to open or toggle between the block diagram and the front panel as shown in ***Figure B***.

  The block diagram contains a While Loop, FPGA target references, and some controls and indicators to interact with the user interface.

  This VI runs on the real-time OS. It reads from the DMA FIFO and displays the data.

17. Set and view limit information between the real-time OS and the FPGA.

Send and receive data between the real-time OS and the FPGA to configure the limit that triggers the LED alarm to light up.



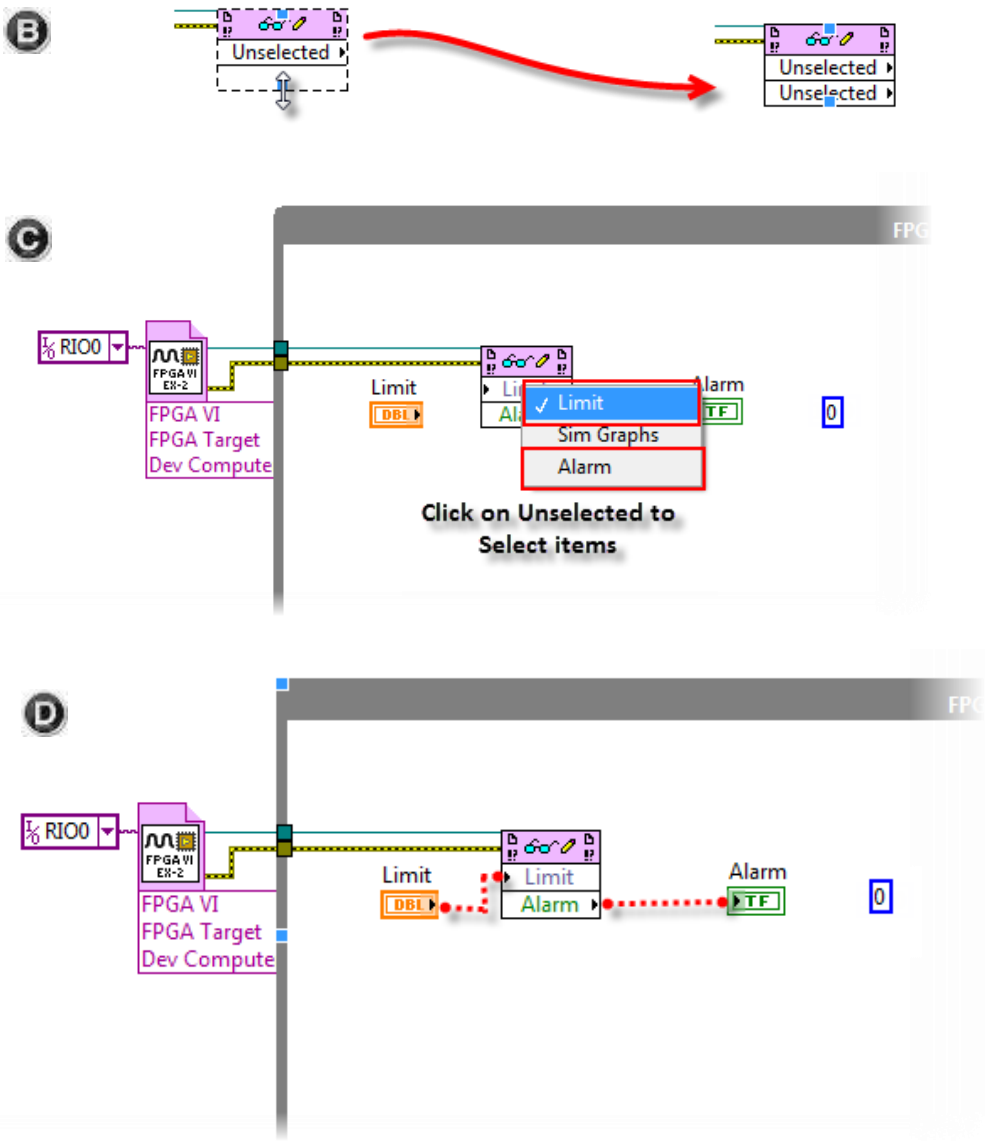
DETAILED INSTRUCTIONS

- ***Add a Read/Write Control Node as Shown in Figure A1***
  Right-click on the block diagram to open the ***Functions Palette***. Navigate to **FPGA Interface** and drag and drop a ***Read/Write Control Node*** onto the block diagram.

- ***Configure the Read/Write Control Node as Shown in Figure A2***
  Continue the ***Reference*** (aqua) and ***Error*** (yellow) wires coming from the ***Open FPGA VI Reference*** function and connect them to the ***Read/Write Control***.

  Connecting the FPGA reference to the ***Read/Write Control*** populates the item menu with items specific to your FPGA VI.

DETAILED INSTRUCTIONS



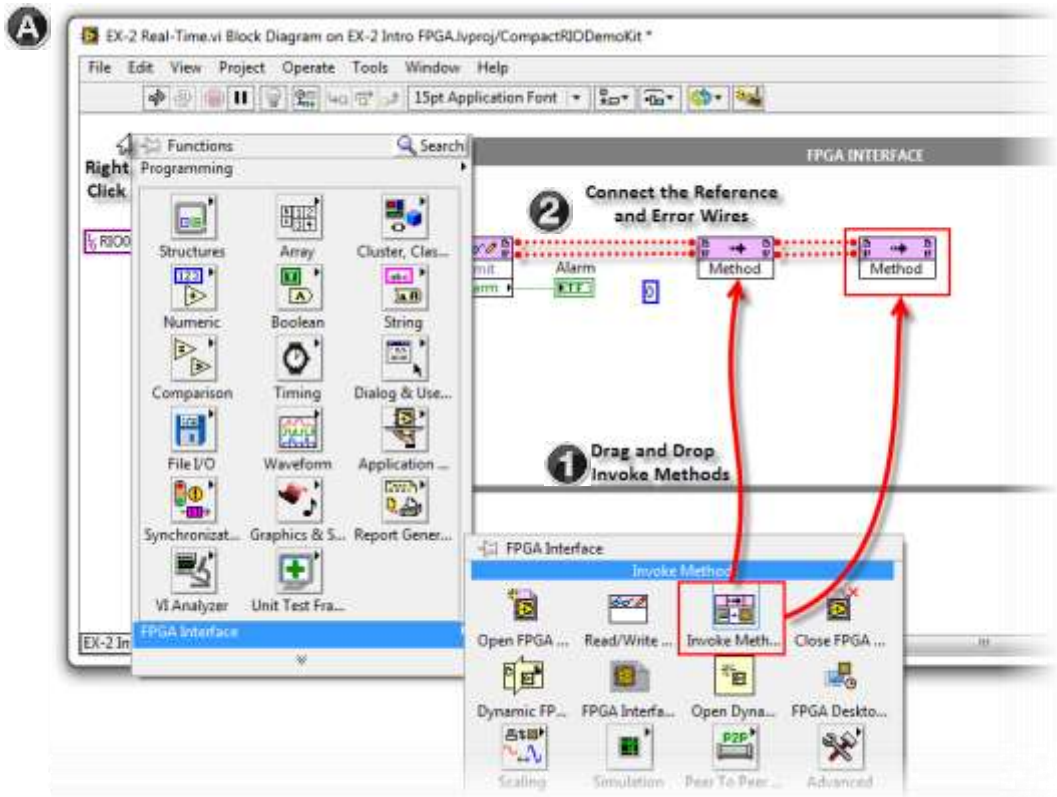
- ***Expand the Read/Write Control to Display Two Items as Shown in Figure B***
  Move the cursor over the ***Read/Write Control*** to show the blue points from which you can expand the node. Expand the node to show one additional terminal.

- ***Select the Read/Write Control Items as Shown in Figure C***
  Click on the **Unselected** text to show menu items. Select **Limit** for the top item and **Alarm** for the bottom item.

- ***Connect the Limit Control and the Alarm Indicator to the Read/Write Control Items as Shown in Figure D***

18. Read the elements from the DMA FIFO.

Place two Invoke Method functions from the FPGA Interface Palette and configure them to read data from the DMA FIFO.



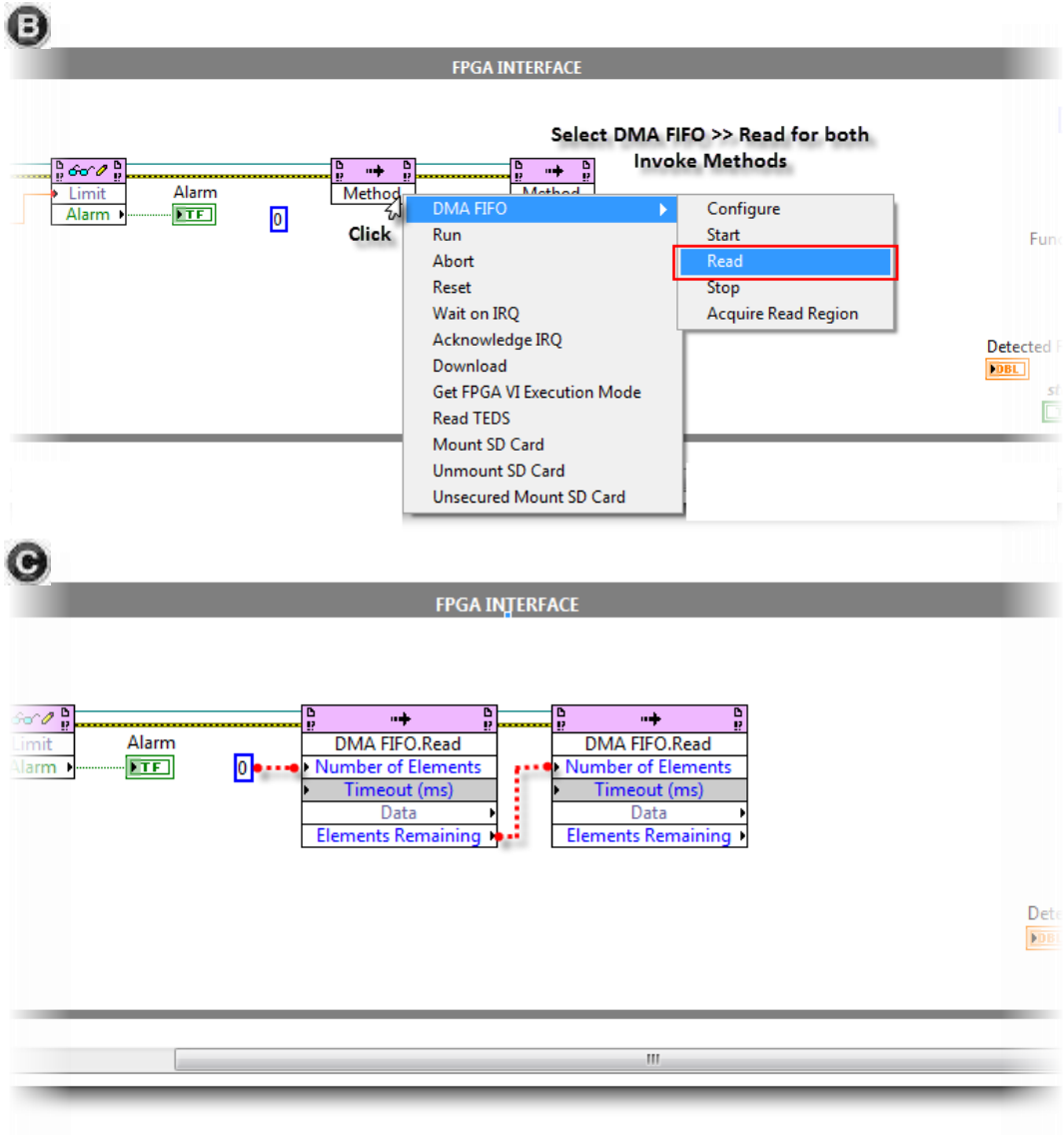
DETAILED INSTRUCTIONS

- ***Add Two Invoke Method Functions as Shown in Figure A1***
  Right-click on the block diagram to open the ***Functions Palette***. Navigate to **FPGA Interface** and drag and drop two ***Invoke Method functions*** onto the block diagram.

  This node invokes a method or action from a host VI on the FPGA VI. In this exercise, use it to access the Function Generator Input data.

- ***Connect the Invoke Method as Shown in Figure A2***
  Continue the ***Reference*** (aqua) and ***Error*** (yellow) wires coming from the ***Read/Write Control*** and connect them to the ***Invoke Method functions***.

  Connecting the FPGA reference to the ***Invoke Method*** populates the item menu with items specific to your FPGA VI.



DETAILED INSTRUCTIONS

- ***Select the Invoke Method Items as Shown in Figure B***
  Click on the **Method** text to show menu items. Under **DMA FIFO**, select **Read.**

- ***Configure the DMA FIFO Read Methods as Shown in Figure C***
  Connect the numeric constant of **0** to the **Number of Elements** input on the first ***Invoke Node.*** Connect the **Elements Remaining** output from the first ***Invoke Node*** to the **Number of Elements** input on the second ***Invoke Node***.

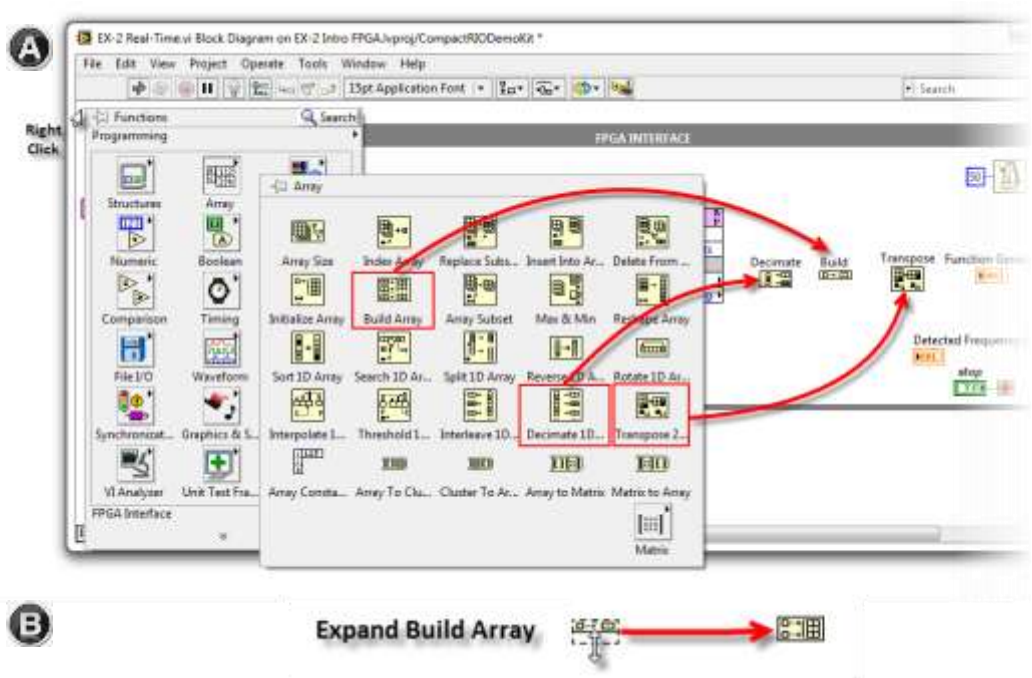
  The dual-FIFO Read is designed to prevent FIFO Overflow.

  The first Read assesses the number of remaining elements. The second read actually reads the elements.

  Using this setup, you can read all remaining elements on the FIFO and not worry about memory overflow.

19. Show the function generator data from the FPGA to the user.

Sort filtered and unfiltered data into two separate arrays. Build them into a 2D array and plot them for the user.



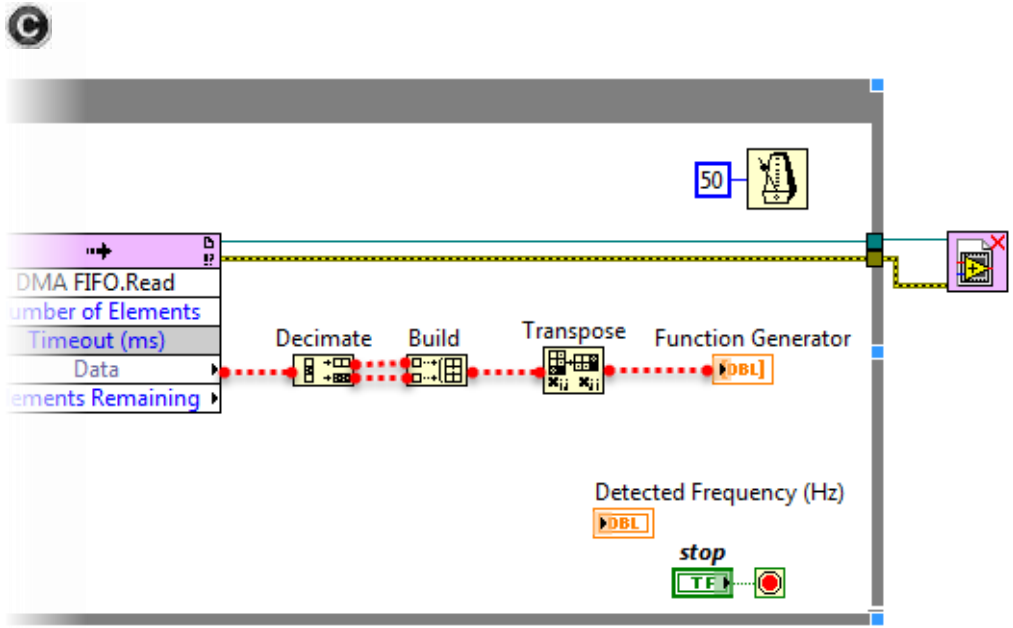
DETAILED INSTRUCTIONS

- ***Create a 2D array of data to send into the Function Generator Chart***
  Right-click on the block diagram to bring up the functions palette. Navigate to the Programming>>Arrays subpalette. Drag and drop the following three functions onto the block diagram as shown in ***Figure A***:

  1. **Decimate 1D Array** – Sorts the elements of the array into two separate arrays
  2. **Build Array** – Builds a 2D array from two 1D arrays
  3. **Transpose 2D Array**

- ***Expand the Build Array Function to show two Input Terminals***
  Move the cursor over the **Build Array** function to show the blue points from which you can expand the node. Expand the node to show the additional terminal as shown in ***Figure B***.

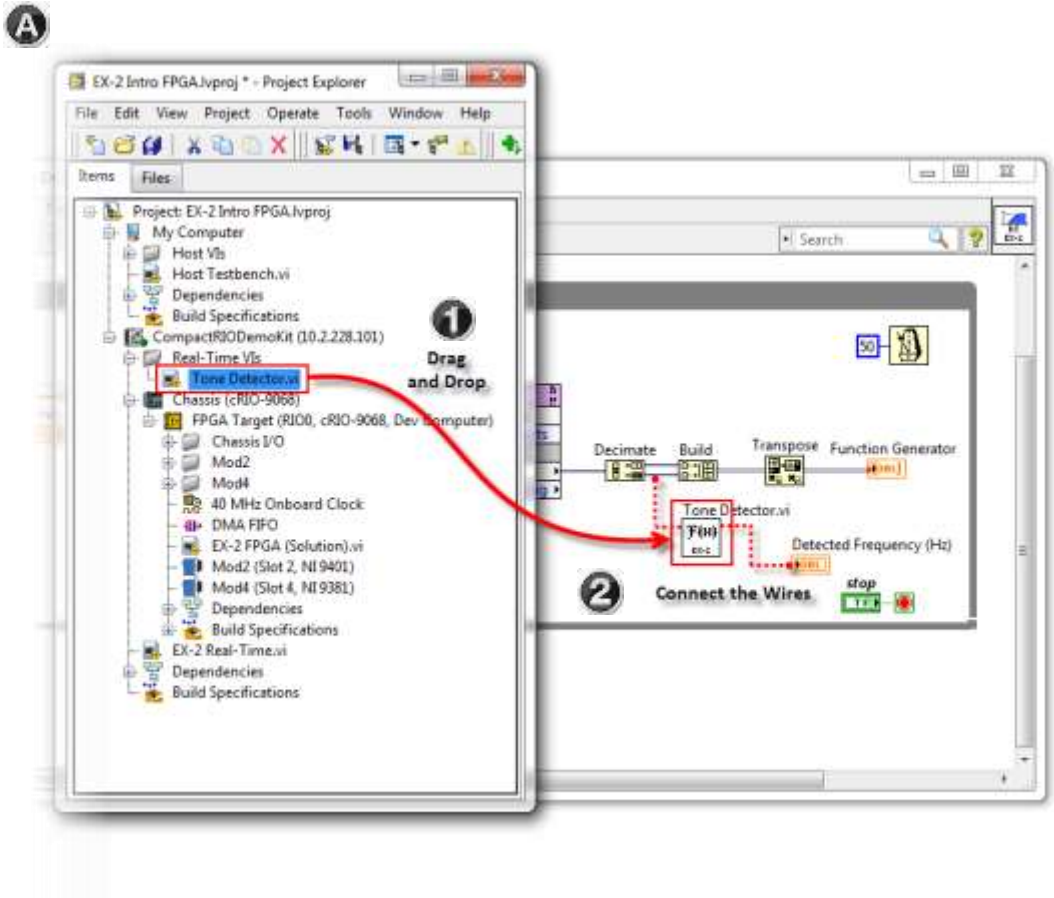
DETAILED INSTRUCTIONS



- ***Connect the Functions From the Data Output to the Function Generator Chart as Shown in Figure C***
  Connect the ***Data*** output to the ***Decimate Array*** function. Connect the two outputs from the ***Decimate Array*** function to the ***Build Array*** function. Connect the ***Build Array*** output to the ***Transpose Array*** input. Connect the ***Transpose Array*** output to the ***Function Generator Chart*** input.

20. Analyze the frequency of the signal.

Use the prebuilt Tone Detector.vi to filter out a frequency from the function generator signal.



DETAILED INSTRUCTIONS

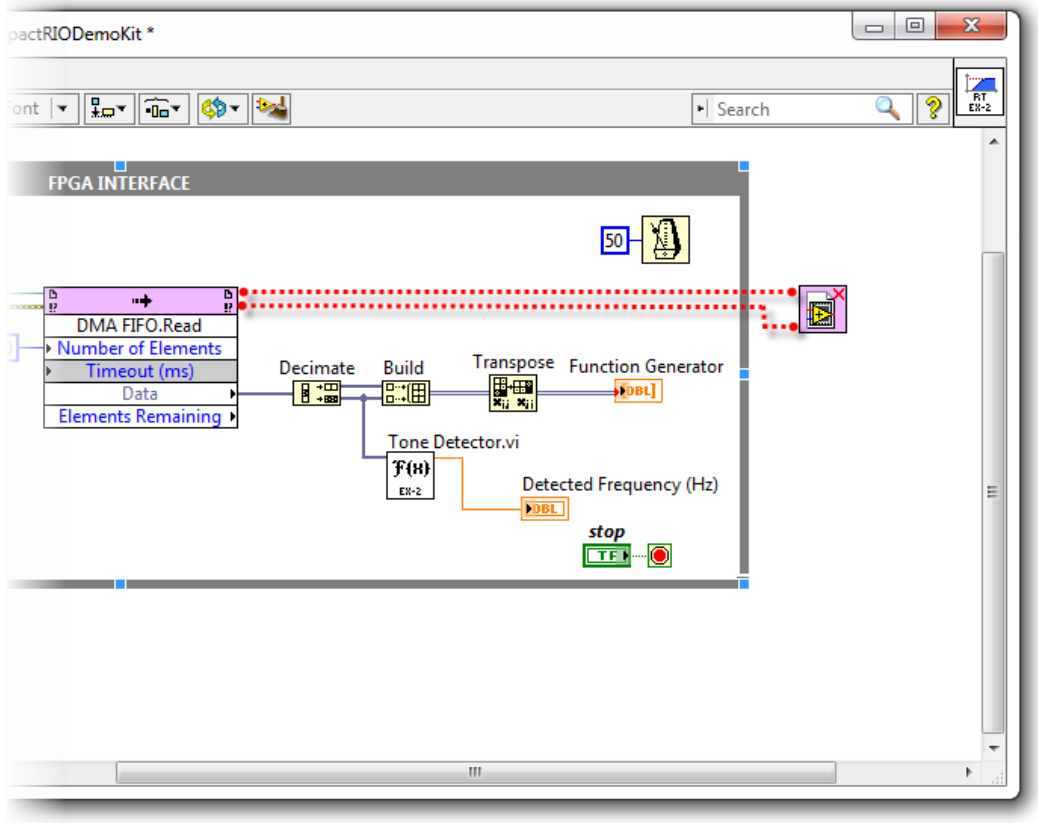
- ***Drag and Drop the Tone Detector.vi Onto the Block Diagram as Shown in Figure A***
  Drag and drop the ***Tone Detector.vi*** in the LabVIEW project under the Real-Time VIs folder in the CompactRIO target.

  This VI was prebuilt to accept an input signal and output the frequency.

- ***Connect the Unfiltered Signal to the Tone Detector.vi and Display the Frequency to the User as Shown in Figure B***
  Connect the bottom array (the unfiltered signal) to the **Tone Detector.vi**. Connect the output frequency to the **Detected Frequency (Hz)** indicator to display the frequency to the user.

21. Close the FPGA target reference.

Close the FPGA target reference at the end of the program outside the While Loop.

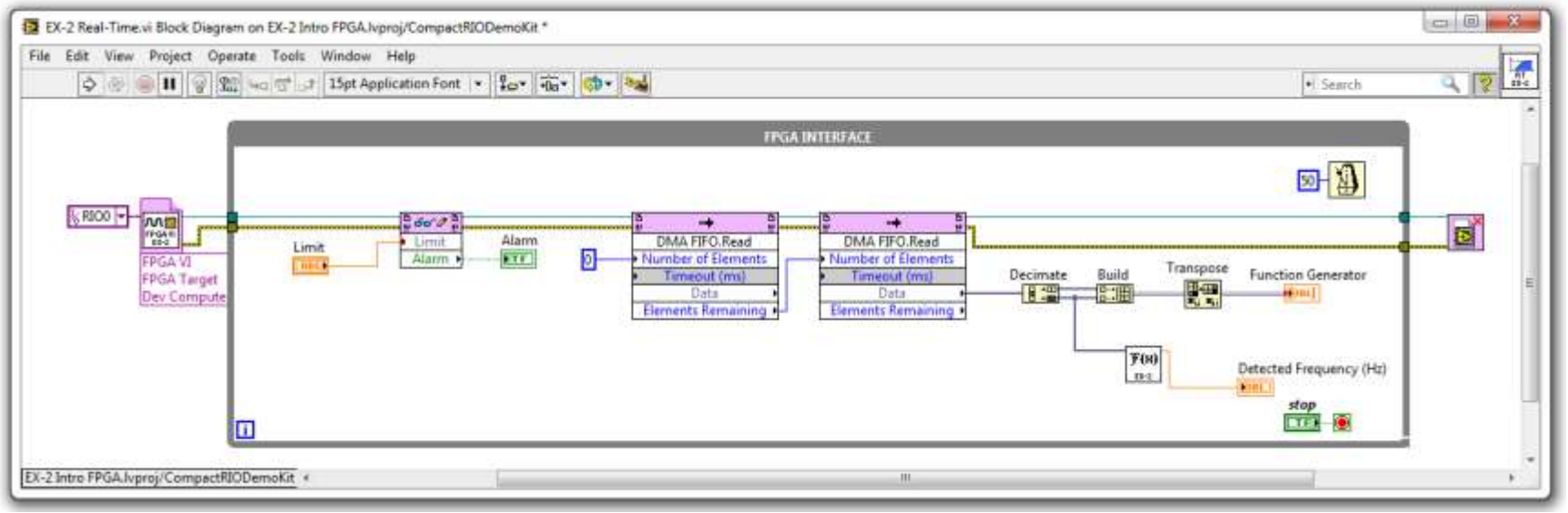


DETAILED INSTRUCTIONS

- ***Connect the FPGA VI Reference From the Invoke Method to the Close FPGA VI Reference***
  Continue the ***Reference*** (aqua) and ***Error*** (yellow) wires coming from the ***Invoke Method*** and connect them to the ***Close FPGA VI Reference***.

22. The completed block diagram for Ex-2 Real-Time.vi should look like the image below.

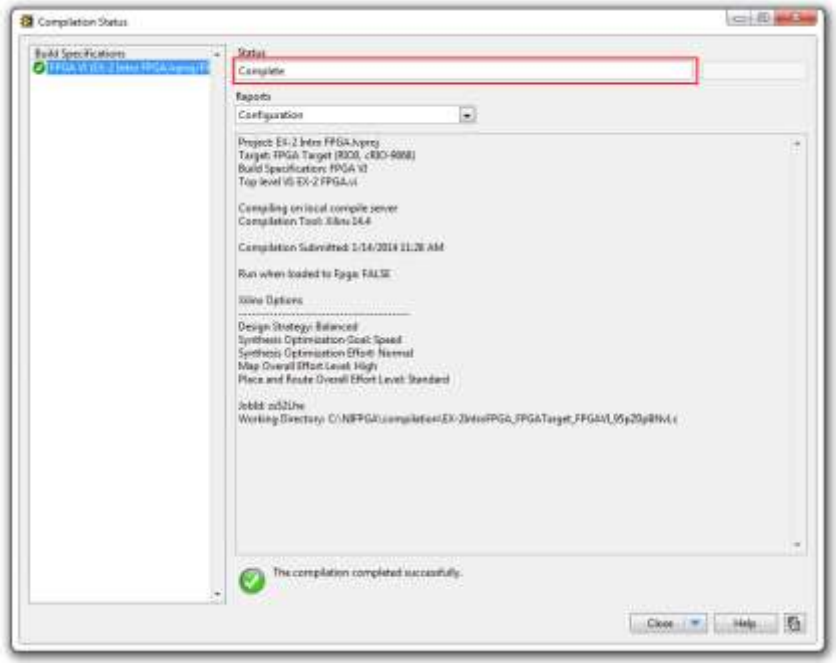
Be sure to save your work by pressing <Ctrl-S> or clicking File » Save.



Part F—RUN THE APPLICATION

23. Verify that the compilation process is complete.

At this point, the compilation of Ex-2 FPGA.vi should be complete. Run Ex-2 Real-Time.vi and view function generator signals on the plot. Interact with the limits and watch the alarm trigger to verify the correct behavior of the application.



DETAILED INSTRUCTIONS

- ***Compilation Process Complete***
  Verify that the compilation process has completed successfully. The ***Compilation Status window*** shows a summary of the performance and resource use of the FPGA as shown in
  Figure A. If the code has not finished compiling, wait for the compilation to complete.

  NOTE: If you closed the ***Compilation Status window***, you can reopen it by right-clicking on the ***FPGA Main specification*** and selecting ***Display Compilation Result***.

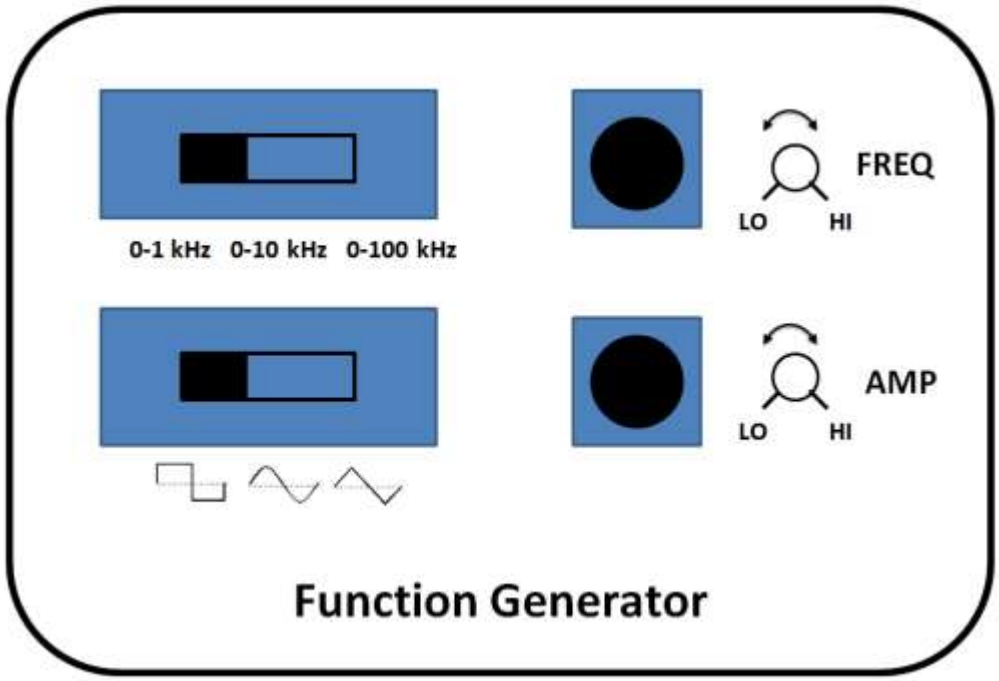
  Device Utilization indicates the percentage of FPGA elements that the FPGA application uses.

  Timing is a summary of the FPGA clocks as estimated during the mapping of the FPGA VI.

  Compilation time depends on the size of the VI, processor speed, and amount of memory in the computer on which you are compiling.

24. Set up the function generator on the CompactRIO demo box.

Set the amplitude and frequency of the function generator with settings similar to your simulation settings for comparison purposes.



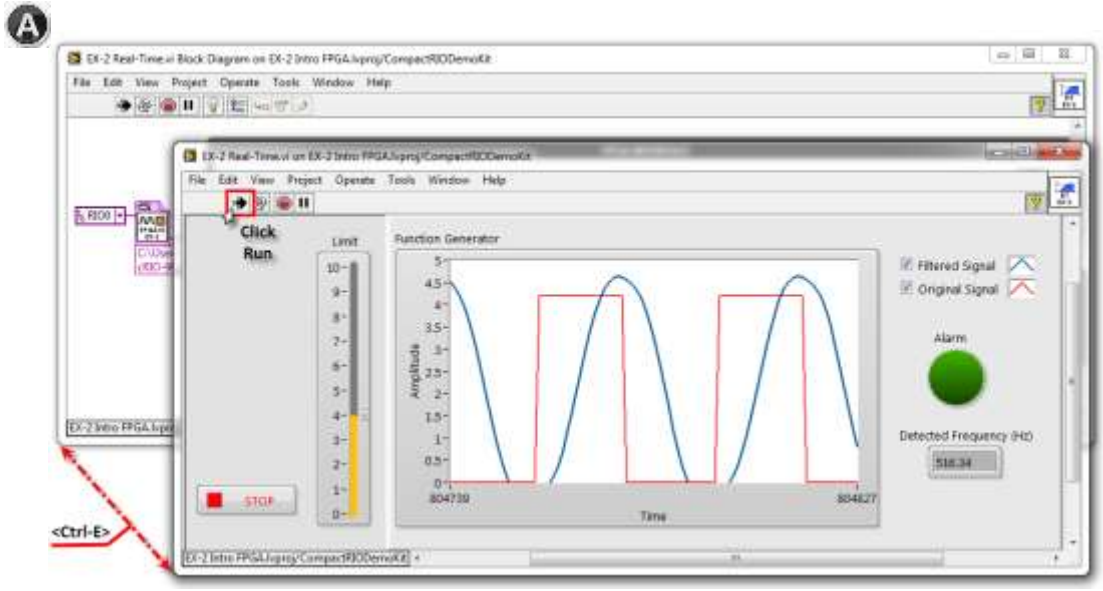
DETAILED INSTRUCTIONS

- ***Configure the Function Generator on the CompactRIO Demo Box***
  Set the ***Frequency*** to 0 kHz–1 kHz and the ***WaveType*** to Square.

  This configuration produces a waveform like the one you simulated in the ***Host Testbench.vi***.

25. Run the Ex-2 Real-Time.vi on the CompactRIO target.

Run Ex-2 Real-Time.vi and view the function generator reading on the plot.



DETAILED INSTRUCTIONS

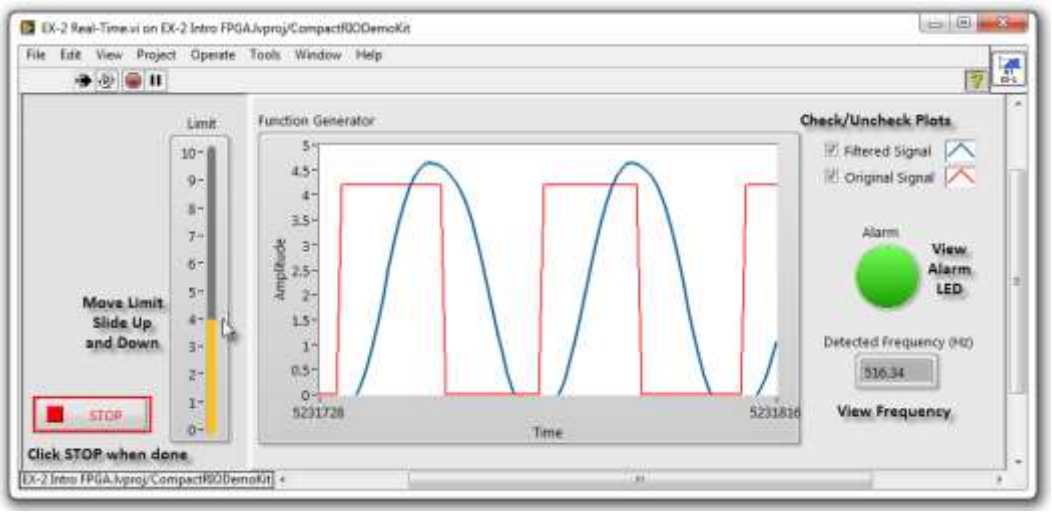
- ***Run Ex-2 Real-Time.vi***
  Switch back to the Front Panel by pressing **<Ctrl-E>** and run the program by pressing the arrow button at the top of the front panel as shown in ***Figure A.*** Save the VI when prompted to do so.

  NOTE: A warning window may appear indicating that the chassis is configured in **Scan Engine Mode** (Exercise 1). Close the warning window, go to the project tree, right-click on the chassis, and select **Deploy**.

  Click the **Apply** button on any subsequent warnings during this process. This changes the configuration of the chassis to **FPGA mode**.

26. Interact with the front panel of Ex-2 Real-Time.vi.

Adjust the waveform types, frequency, and amplitude and watch the response on the front panel display. Interact with the limits and watch the alarm trigger to verify the correct behavior of the application.



DETAILED INSTRUCTIONS

- ***Adjust the Waveform Type, Frequency, and Amplitude on CompactRIO Demo Box***
  Use knobs and switches on CompactRIO Demo Box to adjust waveform, frequency, and amplitude. Watch the signal on the user interface.

  NOTE: Due to your sampling rates, you cannot accurately read signals above 1 kHz. Do not adjust the frequency switch beyond 0 kHz to 1 kHz.

- ***Interact With the Front Panel***
  Adjust the limit control, select which plots to display, and view alarm and frequency data. Press the Stop button to stop the program.

Part G—CHALLENGE

27. Plot the power spectrum.

Add a graph of the power spectrum from the signal to the **Ex-2 Real-Time.vi**.

<END OF EXERCISE 2A: FPGA-BASED BUTTERWORTH FILTER>

ADDITIONAL RESOURCES

BUILD YOUR OWN EMBEDDED SYSTEM WORKSHOP

In the NI Build Your Own Embedded System hands-on workshop, focus on extending your NI LabVIEW skills into FPGA-based embedded design using NI reconfigurable I/O (RIO) hardware.

Purchase the LabVIEW RIO Evaluation kit through the registration process and attend the workshop to receive the following:

- 90-day evaluation of LabVIEW and the LabVIEW FPGA and LabVIEW Real-Time modules
- Board-level NI RIO evaluation hardware device and daughterboard for easy I/O interfacing
- Introduction to the LabVIEW RIO architecture with a qualified NI instructor
- Hands-on experience building your first FPGA-based RIO embedded system with the evaluation kit

To find an event in your area, visit ***ni.com/byoes***.

LabVIEW RIO EVALUATION KIT

NATIONAL INSTRUMENTS LabVIEW™

Using the evaluation kit, develop an embedded system with the LabVIEW RIO architecture. LabVIEW system design software helps you program NI RIO hardware, which includes a real-time processor, FPGA, and I/O. NI CompactRIO hardware uses this same architecture for prototyping through deployment with a flexible array of configuration, expansion, and NI C Series module I/O options.

The kit includes an extended evaluation of the LabVIEW FPGA and LabVIEW Real-Time modules; an NI RIO evaluation device; a daughterboard for easy I/O interfacing; a step-by-step tutorial; and numerous fully documented, ready-to-run examples of common embedded tasks implemented in LabVIEW.

To learn more, visit ***ni.com/rioeval***.

NI LabVIEW Real-Time and NI LabVIEW FPGA RECOMMENDED RESOURCES AND TRAINING OPTIONS

*A CLD or higher is required before attempting the CLED exam