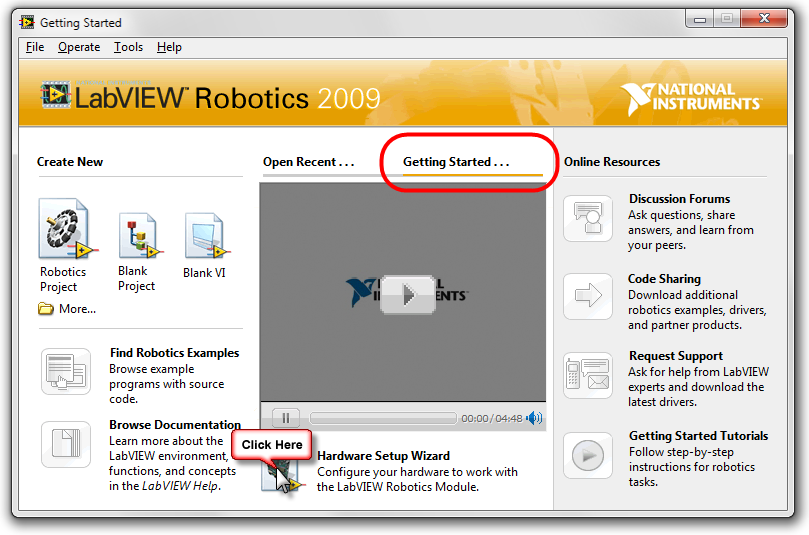
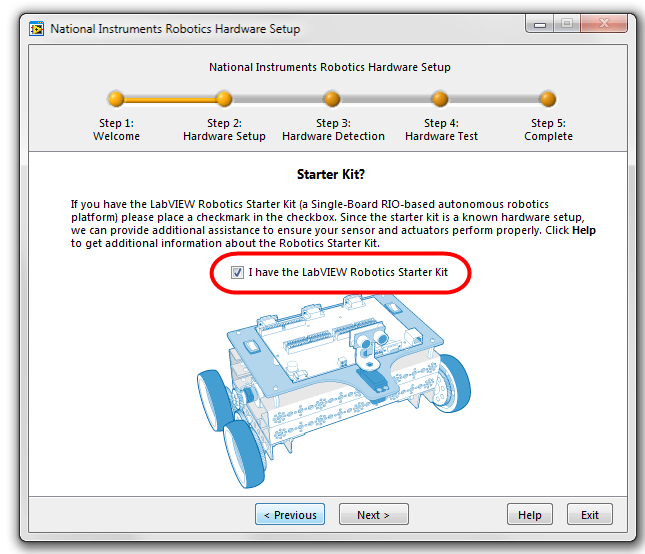
Exercise 1 | Test and Configure the Robotics Starter Kit

**NOTE:** Before setting up your robot you may need to verify the IP settings of the laptop. Navigate to end of this manual for the IP CONFIGURATION page and follow its instructions.

1. Launch LabVIEW Robotics by clicking Start Menu >>All Programs>>National Instruments LabVIEW Robotics 2009.
2. Launch the **Hardware Setup Wizard** by clicking the icon. If you don’t see the embedded video and the Hardware Setup Wizard, verify that you are on the “**Getting Started**” tab.

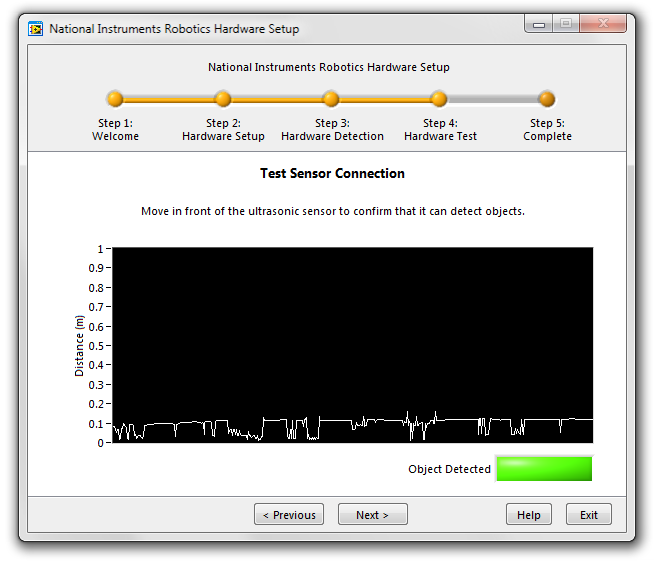


1. Follow the instructions on the screen to configure and test the hardware. Be sure to check the box to indicate that we are using the LabVIEW Robotics Starter Kit.

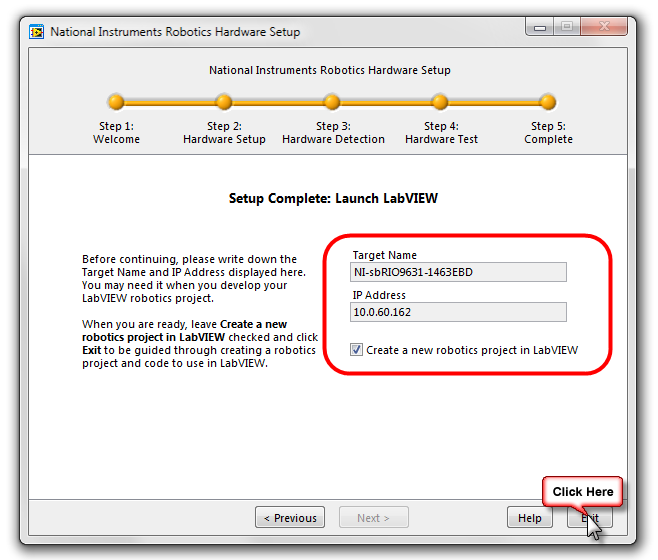


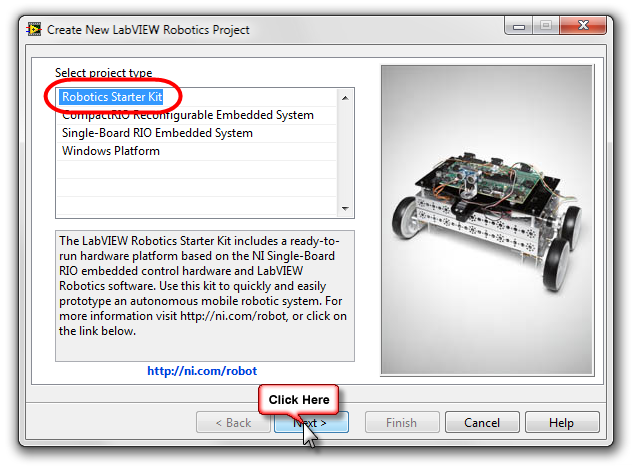
(After detecting the sbRIO device, you may encounter a step that requires you to setup networking for the robot. If so, then you will need to change the IP settings on the laptop and restart the hardware setup. Refer to IP Configuration page located at the back of this manual.)

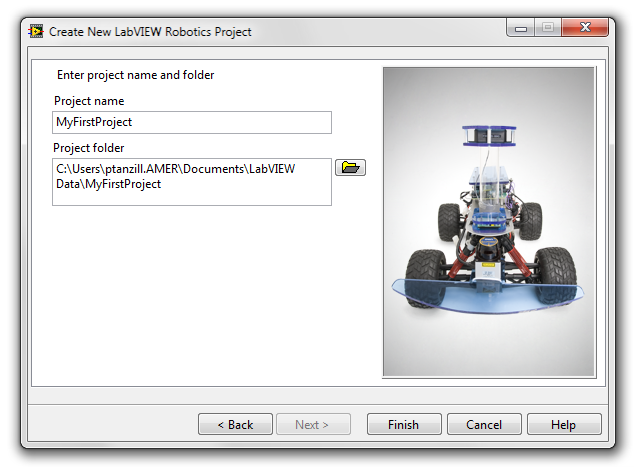
1. Test and calibrate the ultrasonic sensor orientation and electrical connection.



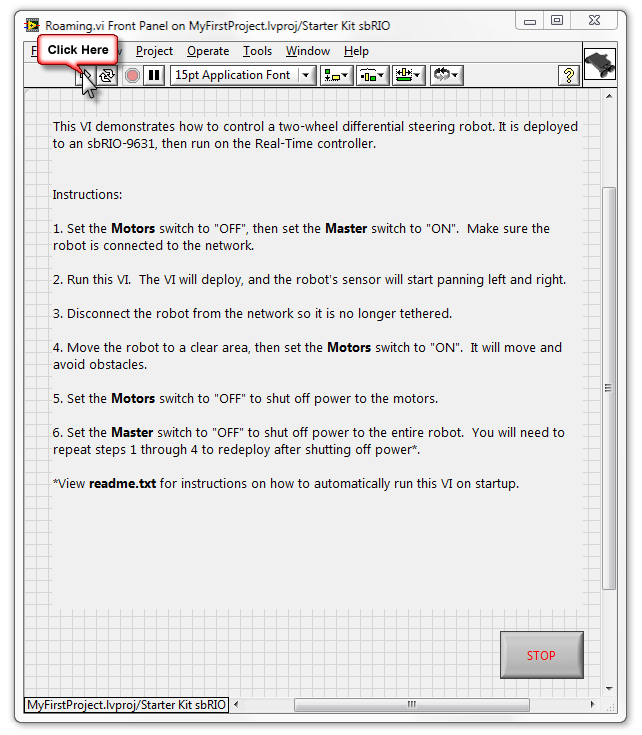
1. When the setup is complete and the hardware is configured, make sure that you have selected to **Create a robotics project in LabVIEW** and click the Exit button. This will launch the LabVIEW Project Wizard.



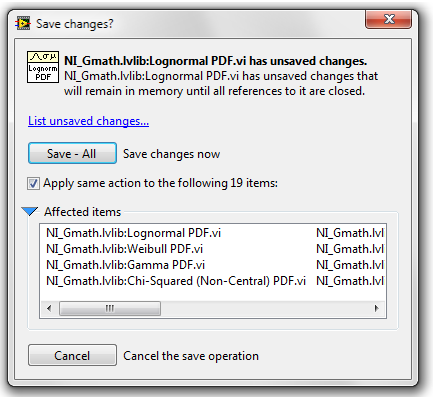
1. Choose the **Robotics Starter Kit** as the project type and click **Next**.
2. Name the project “**MyFirstProject**” and save the file to the **Customer HandsOn Work** folder located on your desktop. You will save everything you do in today’s exercise in this folder.



1. The LabVIEW project will be generated with a working example. Before we get into coding, we will run this program to see the robot in action. To do so, just click the **Run arrow** at the top of the open application. This will download the application to the Single-Board RIO (sbRIO) on the robotics starter kit and start the execution. For safety, make sure that you follow the instructions on the front panel carefully.

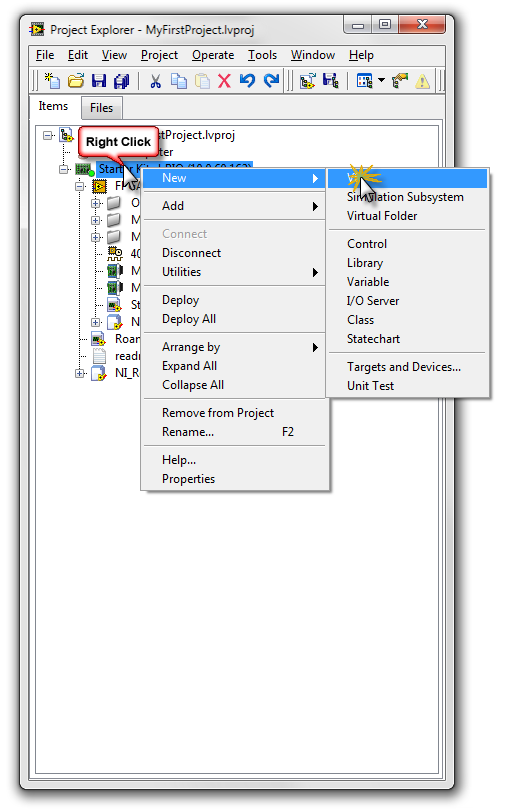


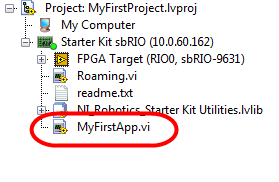
Note: You may come across a dialog like this, choose Save All…



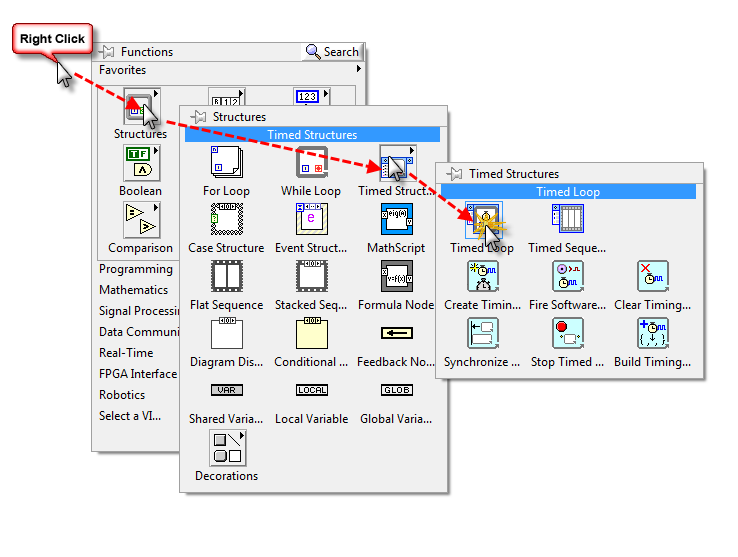
**CONGRATULATIONS! YOU HAVE FINISHED THIS EXERCISE.**

Exercise 2 | Introduction to the LabVIEW Development Environment

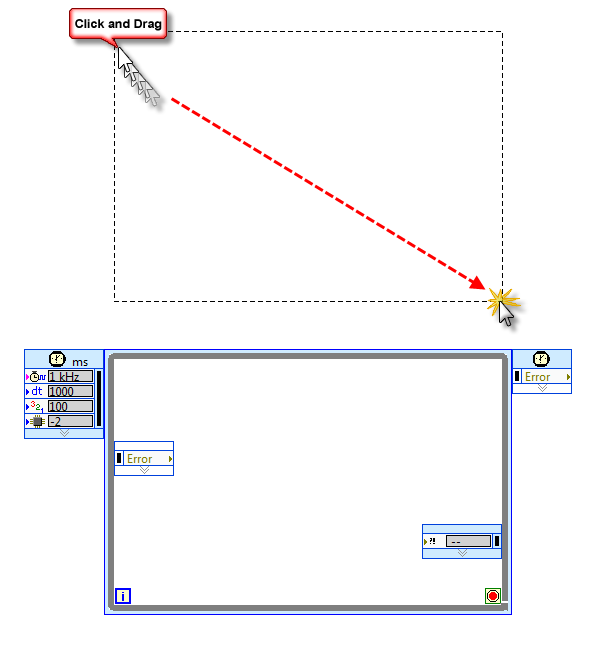
1. Now you will begin your first application. To start, navigate to the LabVIEW project you created in the first exercise. **Right Click** on the **Starter Kit** inside the project, and choose **New>>VI**. This will create a new LabVIEW application, or VI. 
2. Save the newly created VI as **MyFirstApp**. When you are finished, a new element should show up in the project tree.



1. A LabVIEW VI consists of two elements – a Block Diagram and a Front Panel. Choose **Window>>Tile Left and Right** so that you can see both of these elements side by side. Begin by programming on the Block Diagram which has a white background and will occupy the right half of the screen.
2. Start by placing a **Timed Loop** on the block diagram from the LabVIEW functions palette. To bring up the functions palette, right-click anywhere on the block diagram. To drill down into the palettes, hover over the palette that you wish to expand. In this case, we are looking for a timed loop, which is in the **Structures>>Timed Structures palette**. The diagram below shows the path to get to the Timed Loop.



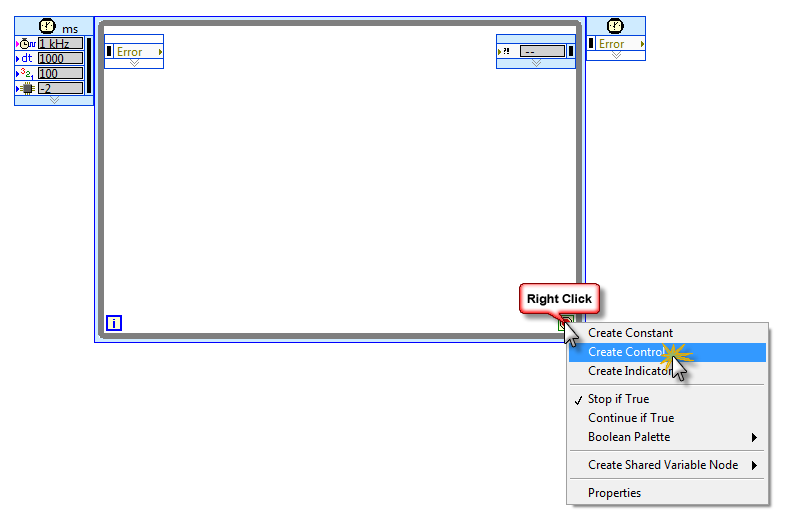
1. To place the loop on the diagram, click and drag just like you are selecting a region. When you release the mouse button, a Timed Loop will be drawn.



1. Next, you will need to add a stop button to the application so that you can stop execution. This will be done by creating a front panel control, and logically connecting the value of that control to the termination condition of the loop. The loop will continue to execute until the condition is true. Similar text pseudo code would look like this:

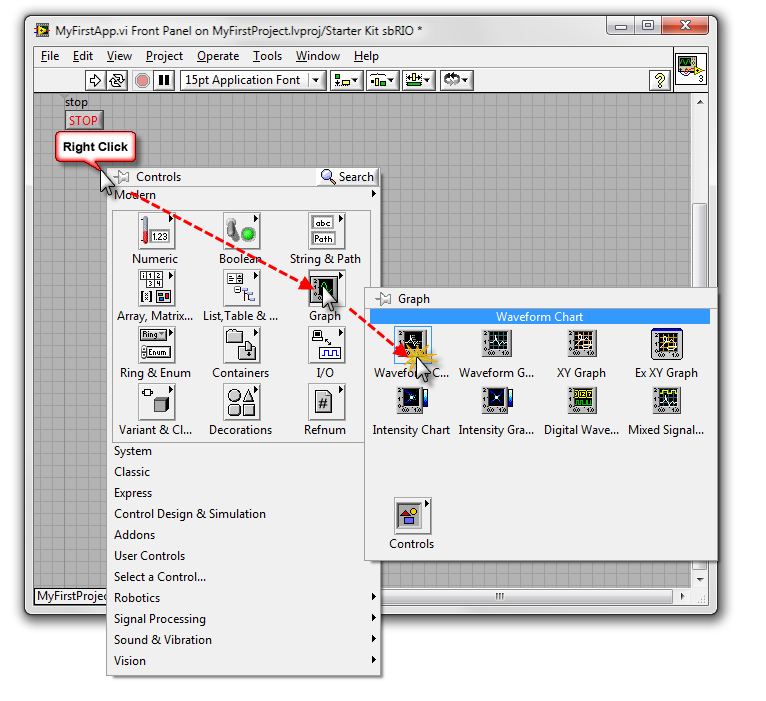
while (STOP is not pressed)  
{  
 //do what’s in the loop  
}

To accomplish this in LabVIEW, right click the termination condition inside the timed loop at the bottom right corner (shaped like a “Stop“ sign), and choose **Create>>Control**.

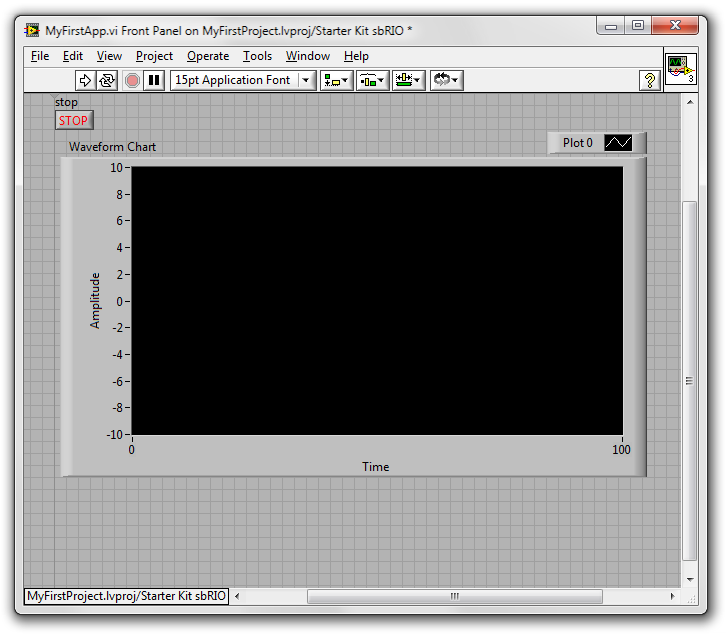


Helpful hint: When in doubt, try right-clicking

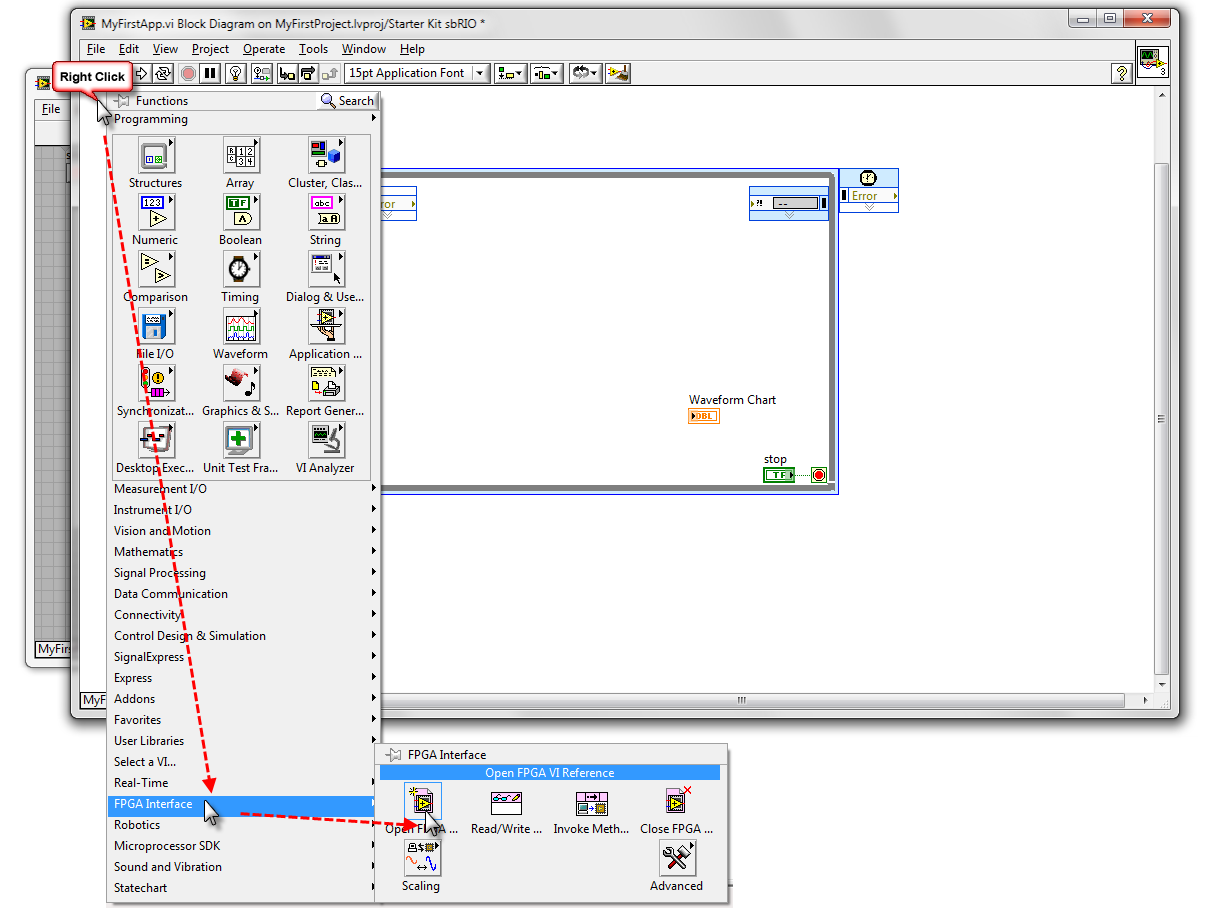
1. You will notice that a Stop button has appeared on the LabVIEW front panel that corresponds with the terminal on the block diagram. You can create front panel objects from the block diagram, as we just did, or you can use the controls palette. Add a waveform chart to the front panel under the Stop button by right-clicking the front panel (the window on the left with the gray background). Select **Graph>>Waveform chart.**



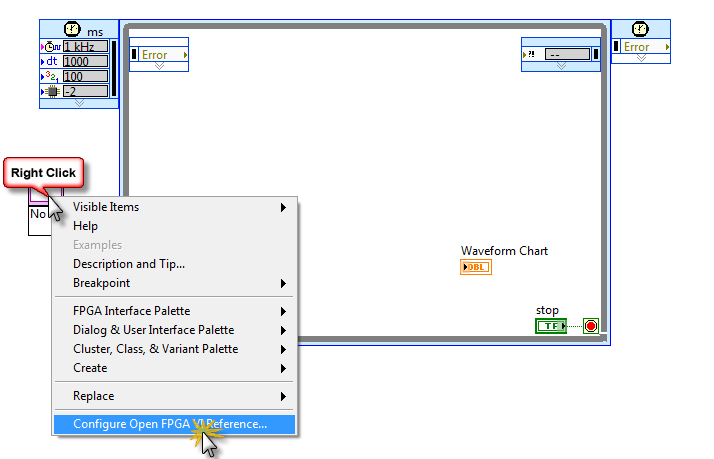
1. When you place the chart on your front panel you can resize it to any dimension. You may also notice that a corresponding terminal was added to the block diagram – you’ll use this later.

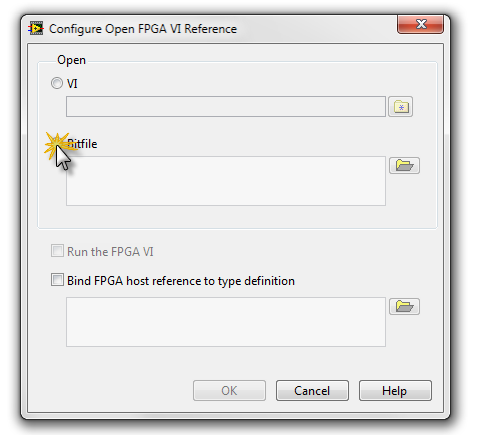


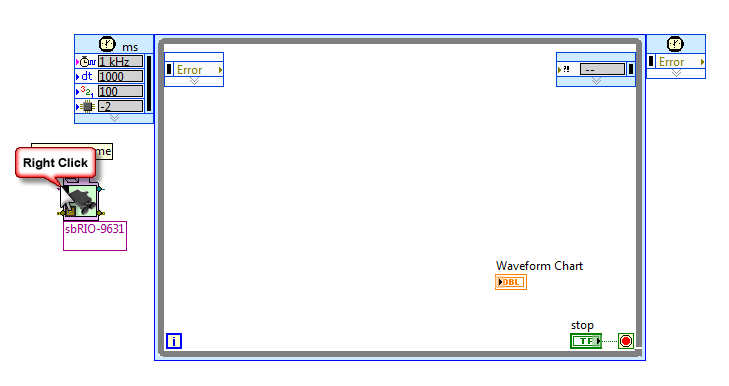
1. Next, switch back to the block diagram and add some code to communicate to the ultrasonic distance sensor. The driver for this device has already been implemented for you, and it will be loaded onto the FPGA. Later, you will learn how you can develop our own FPGA bitfile, but for now, use the one that was generated by the wizard. To begin, place an Open FPGA Reference on the diagram. This will open the communication channel between your processor and your FPGA on the sbRIO. **Right click** the block diagram and select **FPGA Interface>>Open FPGA** **Reference**. Place the VI on the diagram to the left of the timed loop.



1. Right click the Open FPGA Reference VI, and choose **Configure Open FPGA Reference**…



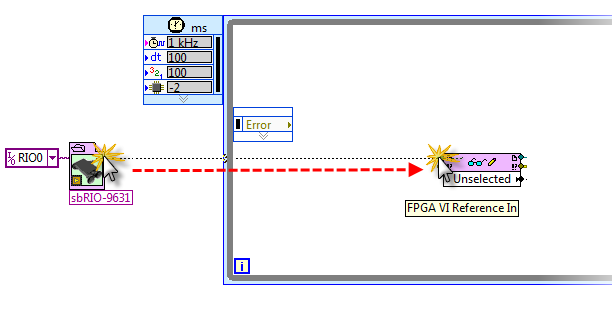
1. Select Bitfile and navigate to the bitfile in the ..\FirstProject\Starter Kit Roaming\FPGA Bitfiles\Starter Kit Roamin~F0\_FPGA Target\_Starter Kit FPGA V~B4.lvbitx. Then **Click OK.**
2. Right click the resource name input in the upper left hand corner of the Open FPGA Reference VI and choose **Create>>Constant**. This will create a reference constant so that you can indicate which FPGA you will be working with.



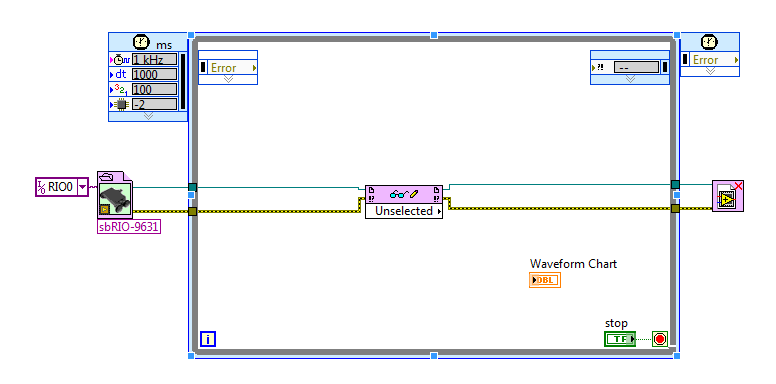
1. In this case, we only have one FPGA on the board, so we will **select RIO0**.



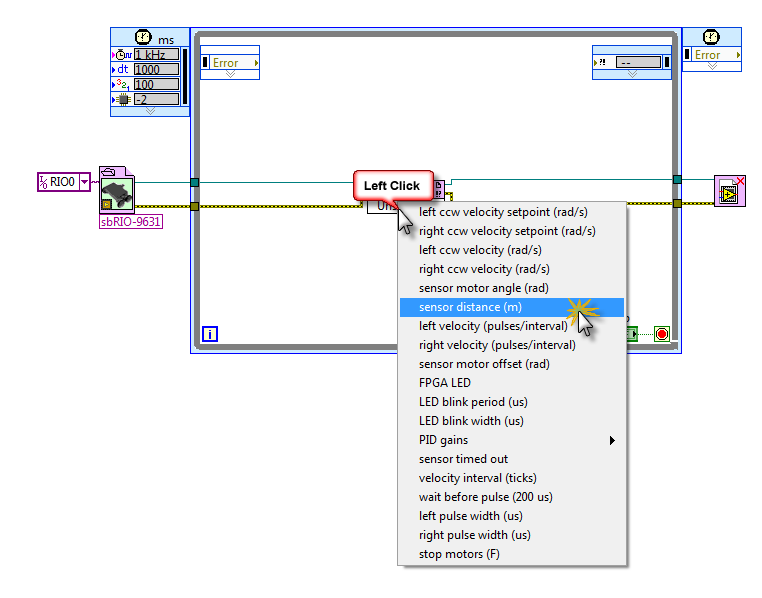
1. From the same FPGA interface palette, place an **FPGA Read/Write VI inside the loop**.
2. Now connect the FPGA VI Reference **Out terminal** on the Open FPGA Reference VI to the FPGA VI Reference **In terminal on the FPGA Read/Write VI**. To connect these two with a wire, click the two terminals that you wish to connect.



1. From the same palette, place a **Close FPGA VI Reference** on the diagram to the right of the loop. Wire the diagram as shown below:

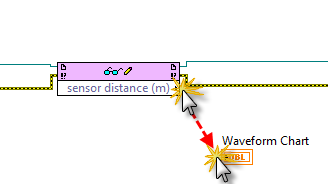


1. Click the Unselected node on the Read/Write block. Now that the block is connected to a bitfile, you will be able to select the particular value that you want to read. For now, **select sensor distance (m)** to read the sensor value.

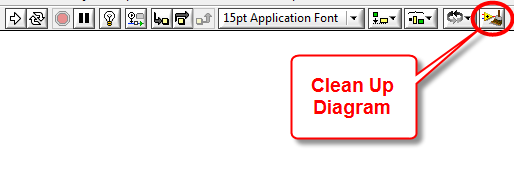


Helpful hint: To remove broken wires, press CTRL +B

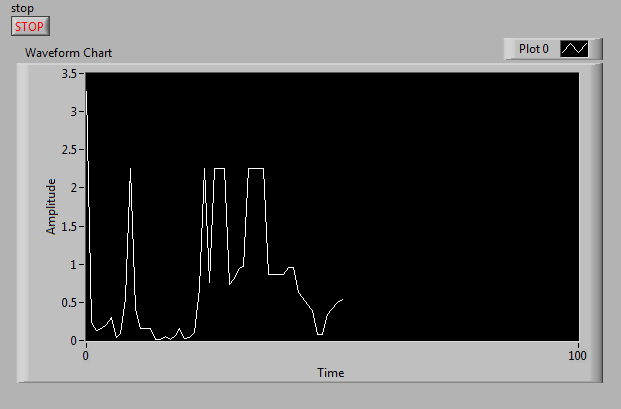
1. Wire the sensor distance output of the Read/Write VI to the waveform chart terminal.



1. To clean up your diagram at any time, click the Block Diagram Clean-Up button or type CTRL+U.



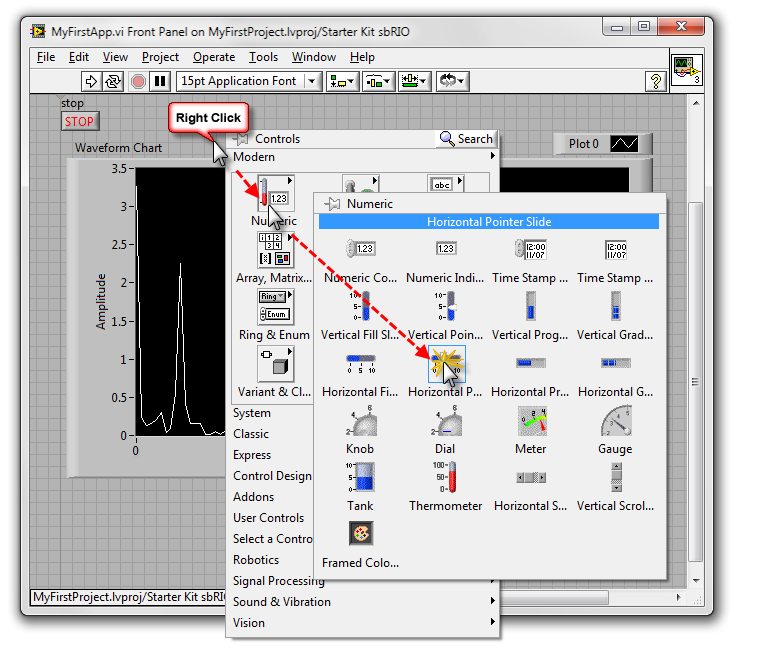
1. Save the VI and click the **Run** arrow. The application will download and run, and the front panel will begin to update. Move your hand back and forth in front of the ultrasonic sensor and watch the corresponding value change on the chart. Press the Stop button when you finish, but do not close the VI, you will use it in the next exercise.



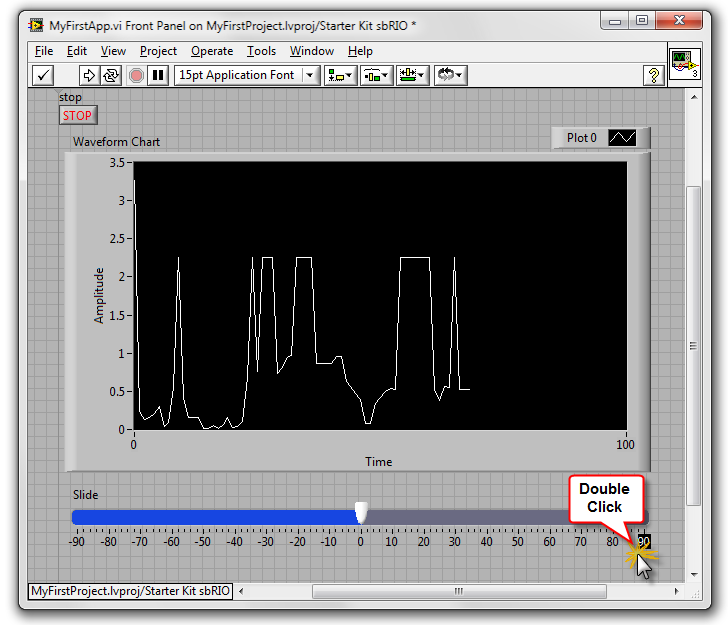
**CONGRATULATIONS! YOU HAVE FINISHED THIS EXERCISE.**

Exercise 3 – Math and Debugging in LabVIEW

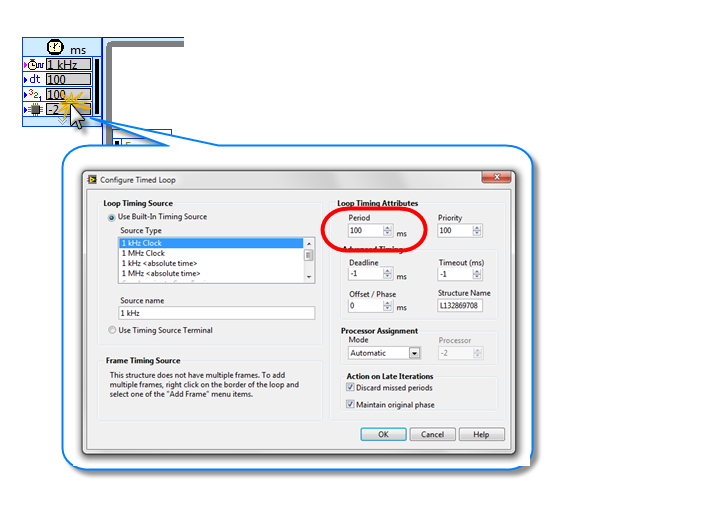
1. Now, you will add a control to the front panel that sets the position of the servo motor that orients the ultrasonic sensor – in effect, you will make the robot turn its head. To begin, place a new control on the front panel that will control the orientation. Place a Horizontal Slider control by right-clicking the front panel and choosing **Numeric>>Horizontal Pointer Slider**.



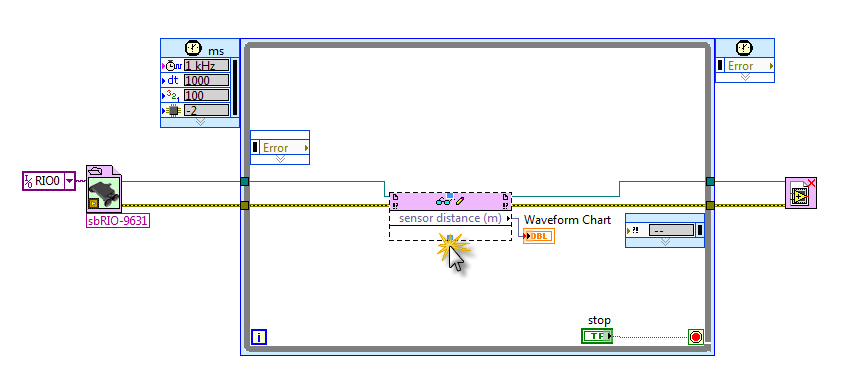
1. The range of motion of the motor is approximately ±90 degrees from center. Therefore, you want the range of your slider to be 90 to -90. To change the range, double click the extreme values. You can resize the slider to the desired length.



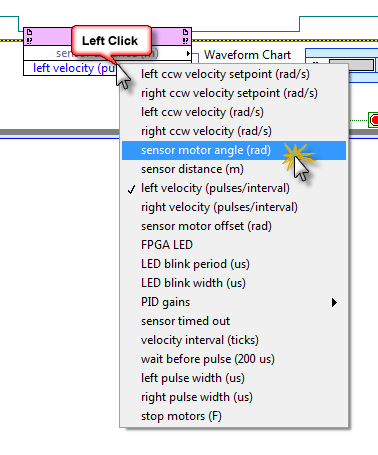
1. Now switch to the block diagram. Let’s make the loop run a little faster to have a more responsive UI. To do so, double-click the upper left hand node on the Timed Loop. Change the period of execution from 1000ms to 100ms.



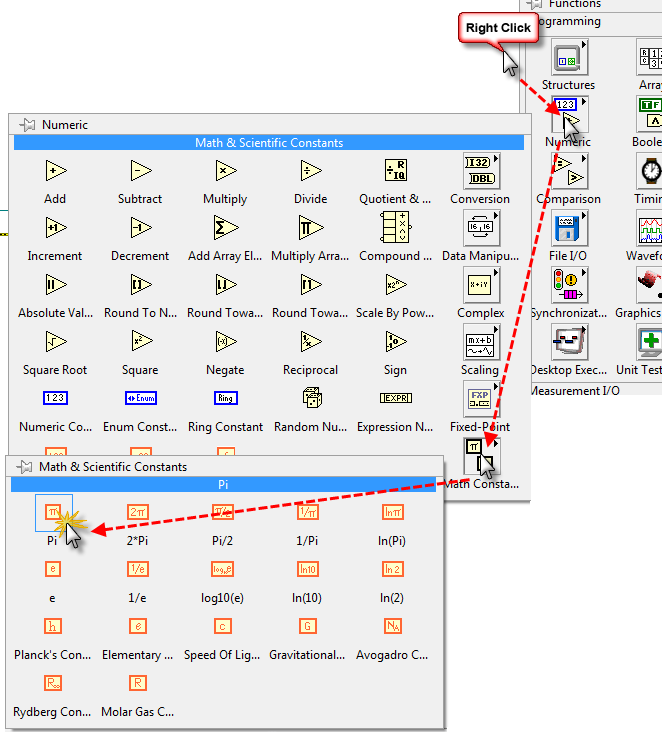
1. To write to the position controller, you will need to expand the FPGA Read/Write VI to have one more element. To do so, highlight the node by left clicking and then resize it from the bottom.



1. Select the sensor motor angle(rad). This will create and input so that we can pass the angle of the motor to the FPGA bitfile that implements the driver. We will see this bitfile in Exercise 5.



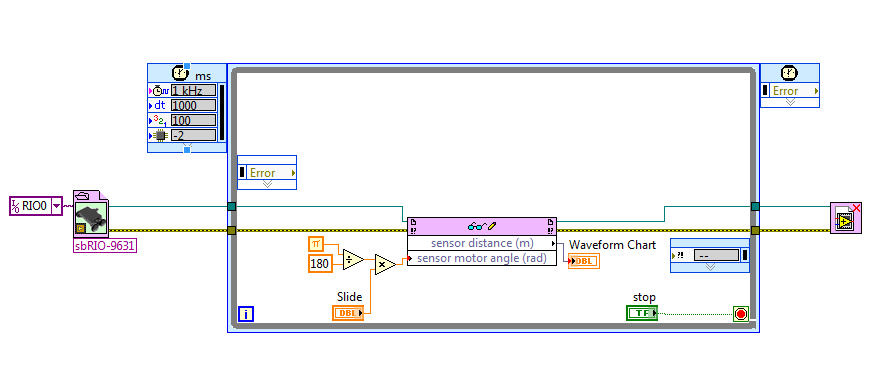
1. You may have noticed that the value of the angle accepted by the driver is measured in radians, but we have created a contol that is measured in degrees. To convert between the two, we need to apply this simple calculation:  
     
     
   To begin, place a Pi constant on the block diagram by navigating to function pallete then **Numeric>>Math Constants>>Pi**



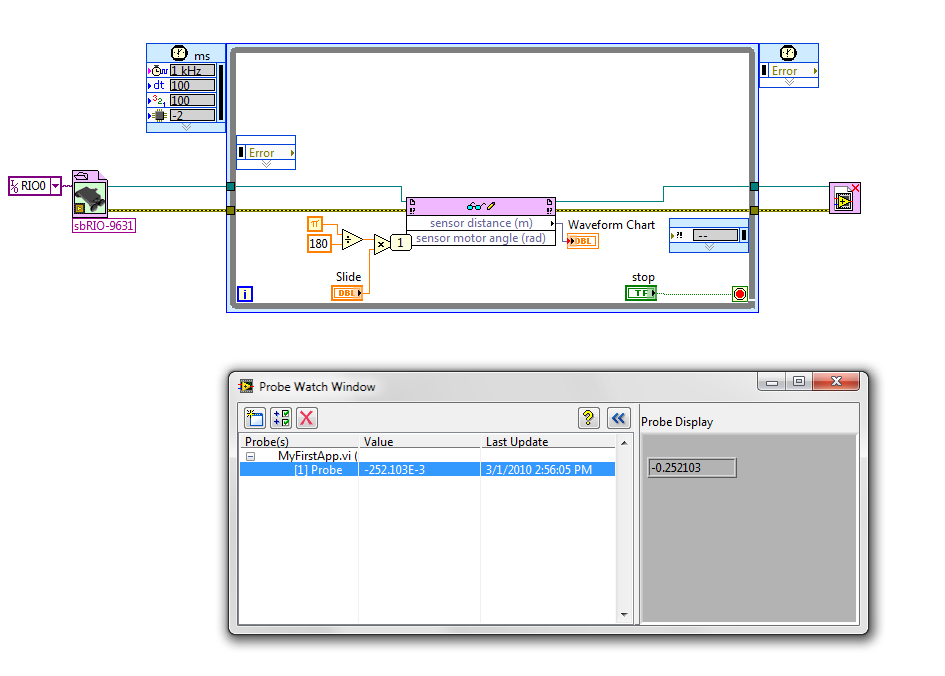
1. You can find a **Divide** and a **Multiply** block on the **Numeric palette**. Place them on the diagram and connect them as shown below. To create the constant value of 180, right click the input to the multiply Block and choose **Create>>Constant**.

Helpful hint: To make more room on the diagram, hold CTRL and click and drag with your mouse.

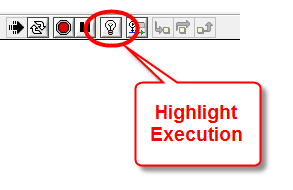
When you are finished, and used the block diagram clean-up by pressing CTRL+U, your block diagram should look like this:



1. Run the application by clicking the **Run arrow**. You will be able to move the slider and see the sensor orientation change. LabVIEW also includes a number of run-time debugging capabilities like breakpoints, single stepping, probing, and many more. You can use probes to find the value on any wire while the application is running. Click one or more of the wires, and you will see the probe watch window updating with live values.



1. To gain a high-level understanding of the dataflow of the application as a whole, you can use the **highlight execution** feature in LabVIEW. This will slow down the application and illustrate the flow of data down the wires. To turn on highlight execution, click the **light bulb** in the **tools bar**.

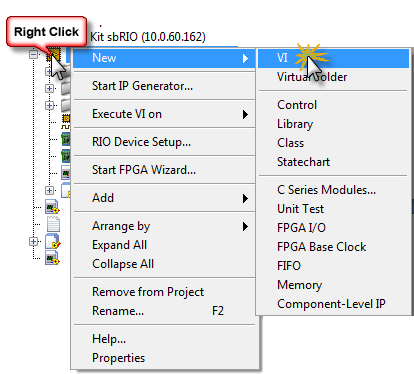


**CONGRATULATIONS! YOU HAVE FINISHED THIS EXERCISE.**

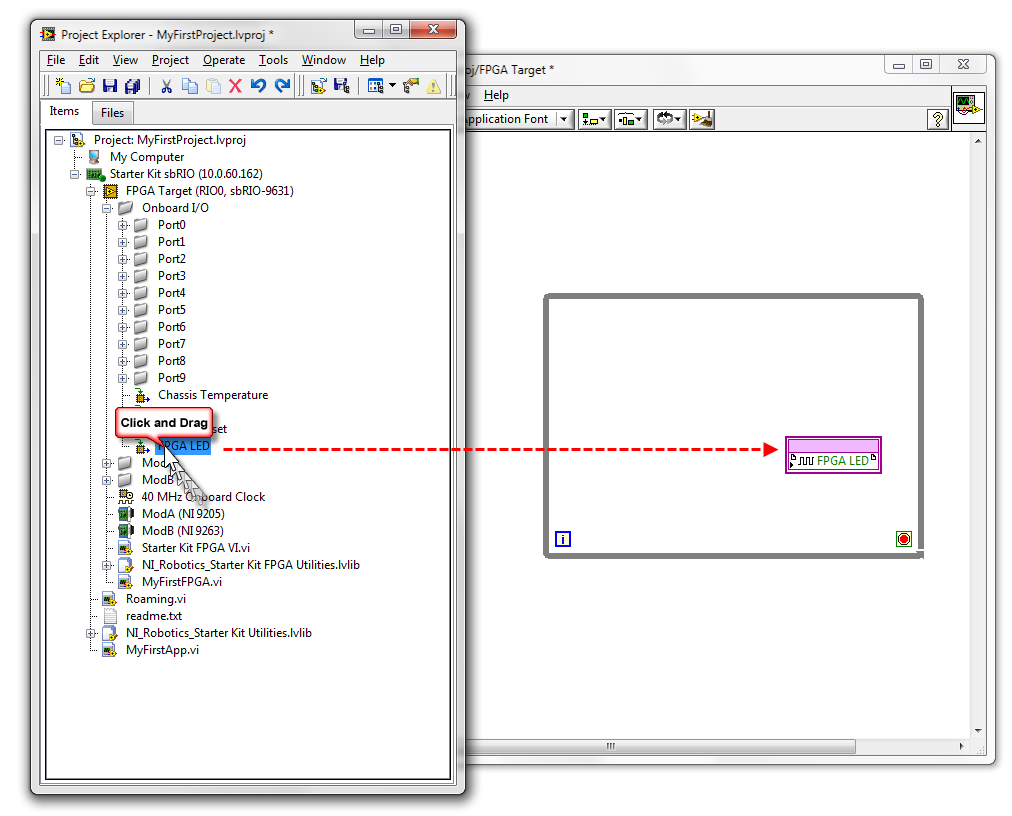
Exercise 4 – Your First LabVIEW FPGA Application

In this exercise, you will create a simple blinking LED application for the FPGA, where you will learn to access FPGA I/O and learn how to work with shift registers.

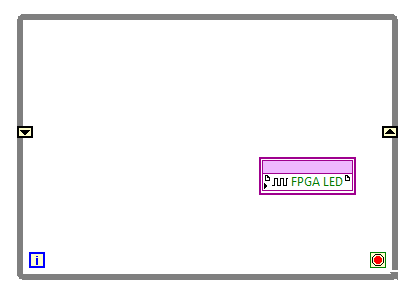
1. To begin, right click the **FPGA target** in the project and select **New>>VI**. Save the VI as **MyFirstFPGA.vi.** Verify that the new VI shows up under the FPGA target in the project hierarchy – this will ensure that the VI will run on the FPGA instead of the processor on the sbRIO.



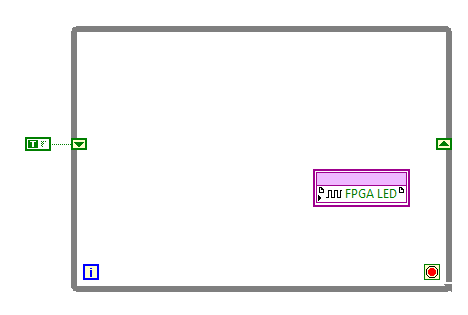
1. Start by drawing an empty **while loop** on the block diagram. You can find a while loop in the **Structures palette.**
2. To add I/O to the diagram, expand the Onboard I/O folder under the FPGA. This is a list of all of the available analog and digital I/O available on the board. Here, you will simply use the FPGA LED output node. To add the node to the diagram, **drag and drop** from the project tree to the block diagram.



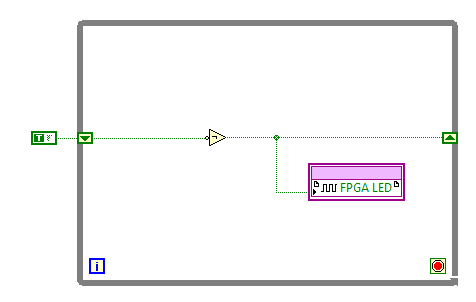
1. In this example, we will use a shift register to maintain the state of the LED from one iteration of the loop to the next. To add a shift register to the loop, **right click the loop** and choose **Add Shift Register.**



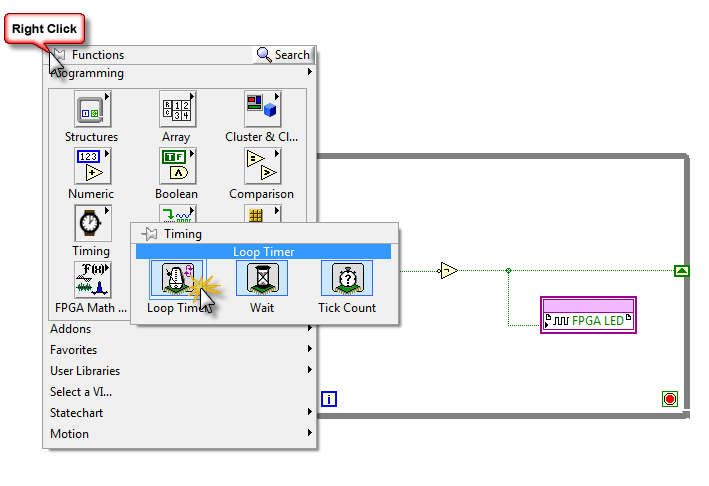
1. You need to give the shift register an initial value for the first iteration of the loop. For this, wire a **True constant to the terminal**. You can find the Boolean constants in the **Boolean palette**. Notice that the function palettes have changed to reflect the subset of functions that are available on the FPGA.



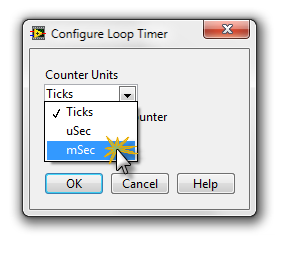
1. Place a Not block on the diagram from the Boolean palette.
2. Wire the diagram as shown.



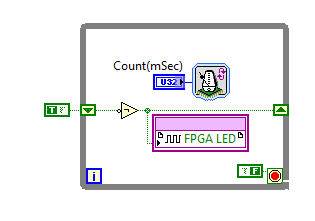
1. The FPGA is capable of controlling execution on sub-microsecond scale. To change the timing of a loop, you need to include a loop timer. Place a **Loop Timer** in the loop from the **Timing palette.**



1. The default clock rate of the FPGA on the sbRIO is 40MHz, meaning that a “tick” takes 25ns to execute. This is obviously overkill for blinking an LED, so instead, you should choose **mSec** from the timer units dropdown. Then Press **OK**.

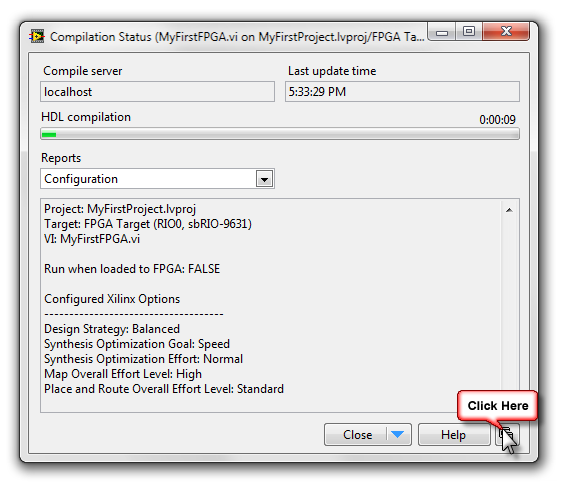


1. **Right-click** the input to the Loop Timer VI and choose **Create>>Control**. This will create a control for the blink rate of the LED.
2. **Right click** the termination condition (stop sign)of the While Loop and **create a False constant** as shown below. This will configure the loop to run forever. The completed diagram is shown below.



1. Now you’re ready to **Save and Compile** the application. This will take several minutes. Click the **Run** button to start the compile. To minimize the compile window, click in the lower right hand window.

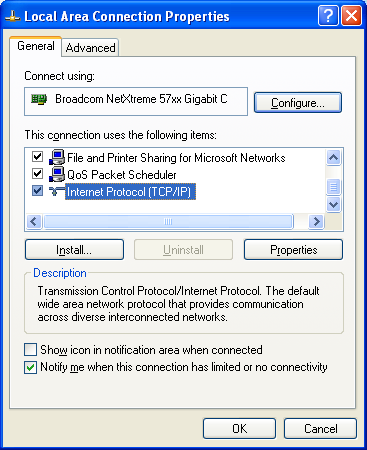
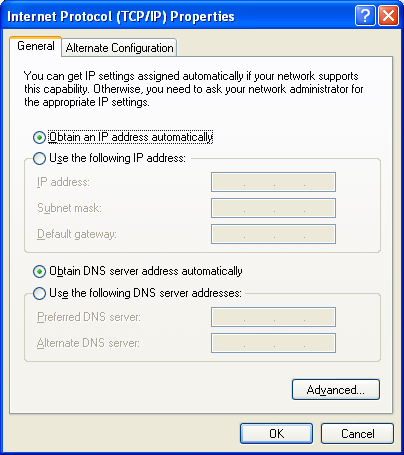
Helpful hint: FPGA Compiles can take a long time. To minimize the number of compiles, start by configuring the FPGA to execute on the Host Computer with Simulated I/O.



1. Once the compile is complete, the application will begin executing. Set the rate of blinking on the front panel to **1000**, and the FPGA LED will change state every second. Change it to 100, and the FPGA LED will change state 10 times faster.

**CONGRATULATIONS! YOU HAVE FINISHED THIS EXERCISE.**

**IP CONFIGURATION (for laptops not the Robot)**

1. Navigate to the Desktop and **right-click** on **My Network Places** and choose **Properties**.
2. **Double-Click** on **Local Area Connection.** You should see the following Properties window. 
3. Choose **Internet Protocol (TCP/IP)** and then choose **Properties**. Your screen should look like the following, if not make the necessary changes. 
4. Click on the **Alternate Configuration** tab. It should look like the screen below, If not, then click on Automatic Private IP. **Click OK**. Then **Click OK again** on the Local Area properties window. This should get the IP settings to where you need to be in order to get through the Robot setup.