Lab 03: Edge Detection– Tutorial

Step 1: Start LabVIEW(LV) Robotics 2009, and then create a new robotics project. The project explorer window will then pop up. Save this project as Lab3EdgeDetection. Once the project has been created it will automatically build and open the Roaming VI from lab 1. Minimize this for now; however, we will be altering elements of this code later.

Step 2: From the project explorer window create a new virtual interface Right click “FPGA target” on the project tree and select “new” and then VI. Two windows will open: the block diagram and the front panel. For the purpose of robotics we will rarely ever use the front panel as the labs in this course require autonomous operation.

Step 3: This is an FPGA VI where we will set up the new sensors required for this project. By going on ni.com and entering “9631” in the search bar, the first item on the list will be the sbRIO webpage for the robot. By going to resources, clicking on manuals, and then selecting user manual, you will be able to bring up the pdf file that contains the pin layout for the sbRIO. Depending on which pins you connect the sensor to will change what ports you will select for your FPGA VI.

Step 4: Copy all of the content from the VI titled “Starter kit FPGA VI.” Paste the content into your FPGA VI and then add two while loops. Inside these while loops you will right click the stop sign in the corner and create a constant to have an infinite loop.

Step 5: Next, drag the ports that you will be using into the while loops. Afterwards, right click on the second output of each port (not the FPGA I/O output), select create, and then select indicator.

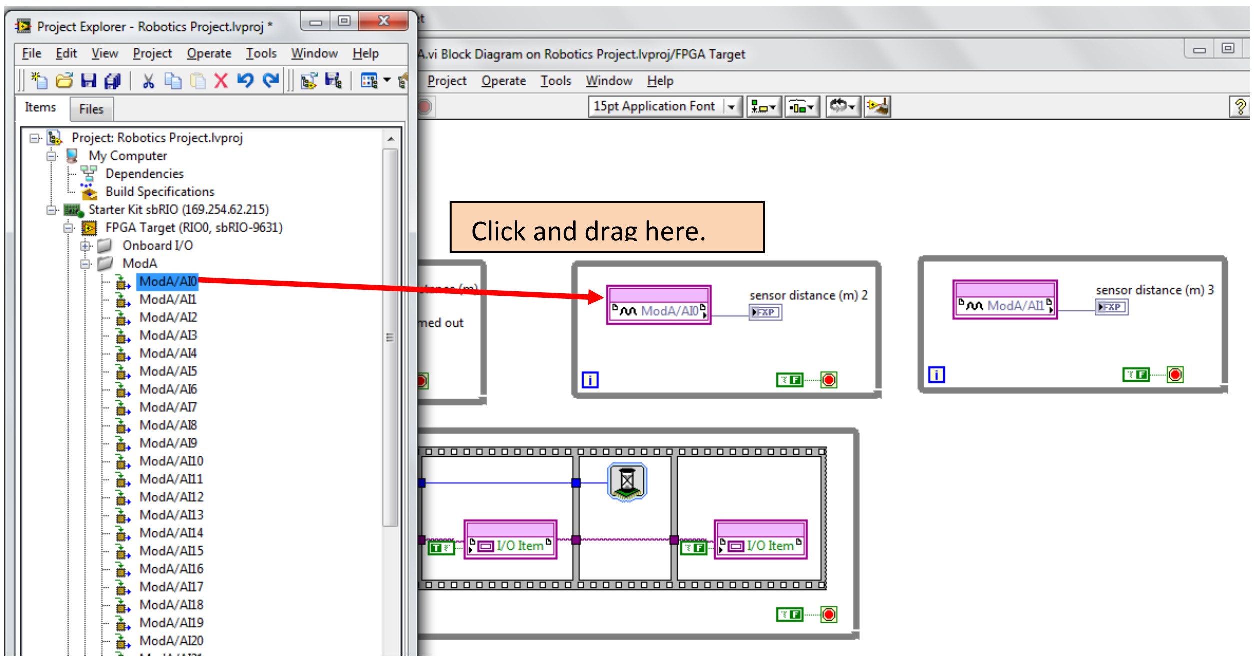

Fig. 3-1

Step 6: Save the VI as "StairSensorFPGA" and compile it. We are now ready to move on to making a subVI of the roaming.vi.

Step 7: Right click "starter kit sbRIO (some ip address)", select new, and then click VI to create a new VI that will be our subVI version of the roaming.vi. Copy all content from roaming.vi and paste it into this new blank vi. Save the vi as "roamingsubVI."

Step 8: To create a subVI, the connector pane must be the same for all other subVIs used in the main code. Also, connections must be assigned to either control elements or indicators in the code. First we must start off by creating the controls and indicators for this VI. Delete the "Open FPGA reference" vi along with the constant input to it. Press "ctrl b" to remove all unconnected wires.

Step 9: Right click the input to the first "read/write control" vi that was directly connected to the "Open FPGA reference" vi. Select create control for both inputs. It should end up looking like the figure below. The figure also shows how to create a control after right clicking the input.

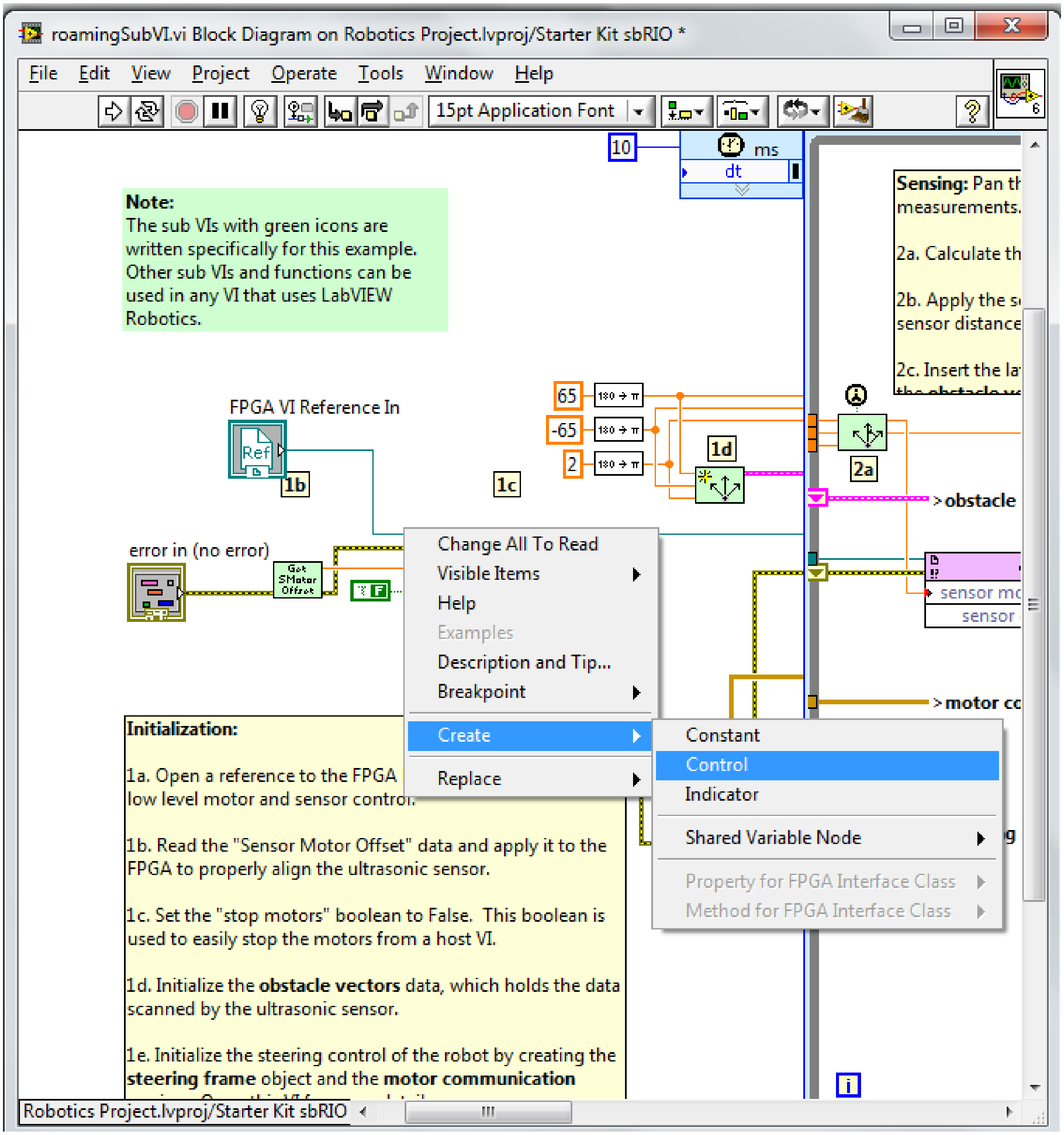

Fig. 3-2

Warning: If the control is not created this way, then there is a chance that an error will occur when trying to use the subVI in the main VI that we will make later. When connecting the FPGA reference to the input to the subVI the error will say that it cannot connect to the subVI because the wires are of two different data types when they are not. If the error still occurs then go back to the subVI and try to create the controls and set up the connector pane again.

Step 10: Now we will make the indicators for the outputs of this subVI. We will do the same as we did for the inputs and delete the “close FPGA reference” vi. Then press “ctrl b” to get rid of unconnected wires and right click the output of the last “read/write control” vi and create an indicator for the FPGA reference out. If it is in the loop, then drag the indicator outside of the loop. For the error out we still want to use the “close motor connections” subVI, so right click the output of that subVI and create an indicator for that output. The figure below shows how it should look like. Ignore the changes to the halt condition for this step.

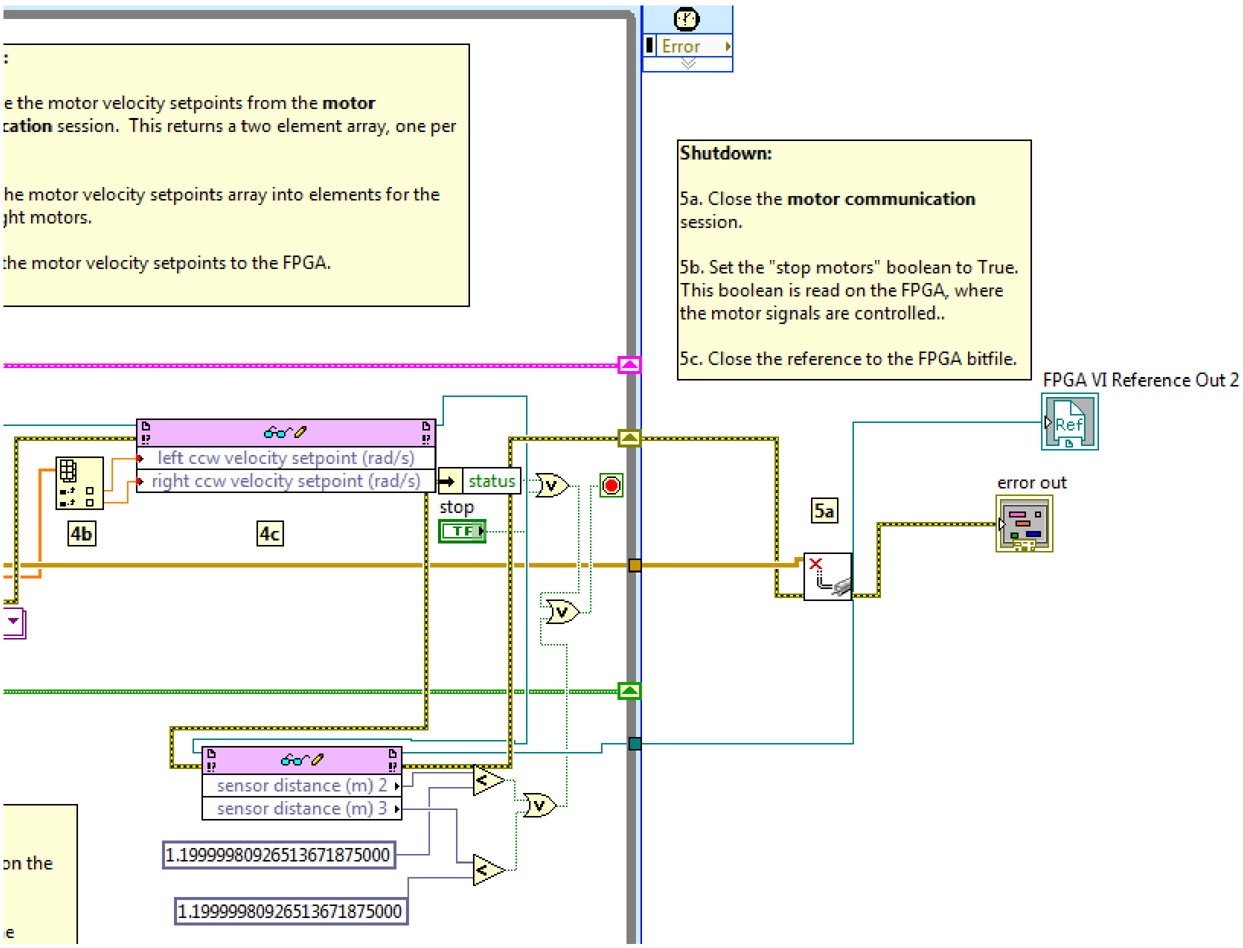

Fig. 3-3

Step 11: Fig. 3-3 shows the changes to the stop condition for the loop that we will make. Add another “read/write control” vi to the loop and connect it from the previous “read/write control” vi. You may have to change your wire connections to the indicators but in the end it should look like the Fig. 3-3. Set the new “read/write control” vi to read from the infrared sensors. This may require you to change the FPGA reference seen by the VI to access the new sensor distances. Go to the front panel and double click on the FPGA VI reference in control we made in step 9. Choose the FPGA bit file like you normally would when setting the “open FPGA reference” vi.

Step 12: Now that we can read the distance values (remember that this value is actually the raw AC voltage), we can use the graph in the datasheet for the infrared sensor to see what AC voltage corresponds to what distance. Also, testing the sensor and observing what AC voltage the voltage reading needs to stay above is also good. For our implementation of this code we found that as long as the AC voltage stays above 1.2V then the robot has not detected an edge. Any lower and that means that there has been an increase in the distance seen.

Step 13: Thus, to analyze both sides we must compare this value of 1.2V to what is seen by the sensor. So place two less than symbols on the block diagram and wire both readings to one of the less than symbols. For the other input of each less than symbol, create a constant input and set it to 1.2V. An OR symbol must then be placed on the block diagram. Wiring the outputs of both less than symbols to the OR symbol to complete our comparison to check whether one of the sensors has detected a greater distance. Lastly, place another OR symbol and connect the output of the OR symbol for the distance readings and the other OR symbol with the inputs coming from the constant and status. Wire the output of this OR symbol to the stop condition for the loop.

Step 14: To finish this subVI, we will now make the connections to the connector pane. To do this, go to the front panel and right click on the icon in the right hand corner. Click on show connector pane.

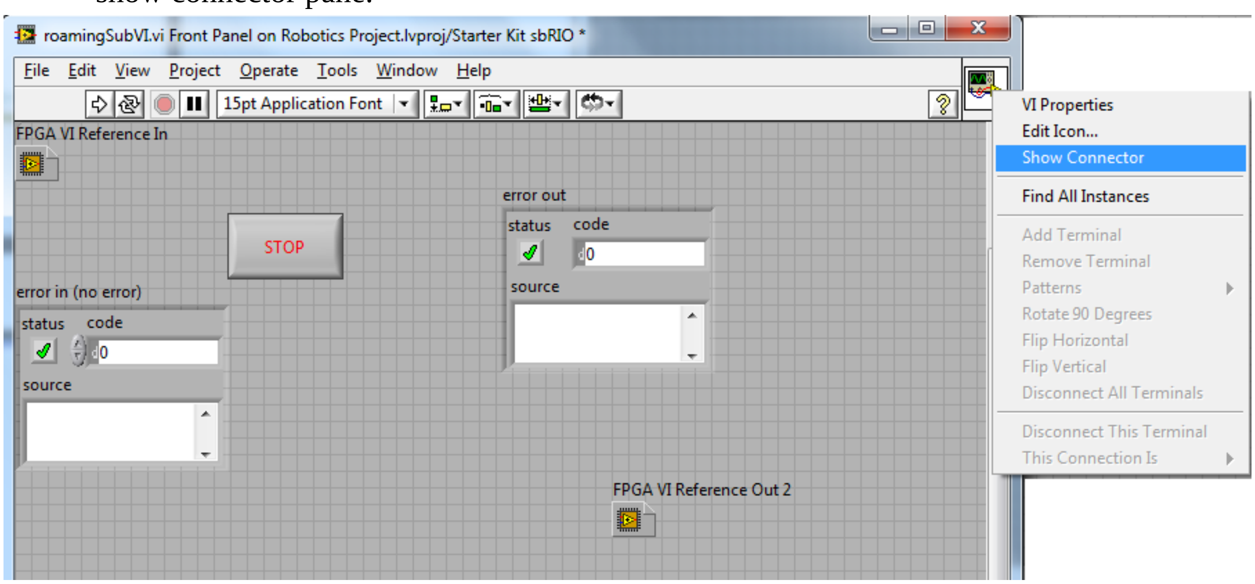

Fig. 3-4

Step 15: The left side of the connector pane along with the center are all blocks for inputs to the VI and the right side are for the outputs of the VI. To assign inputs, click one of the blocks on the left side and then select one of the controls that we set up before. Once that is done for each input we do the same for each output but click on blocks on the right side of the connector pane. The figure below shows the last input being connected to the connector pane with the outputs already connected.

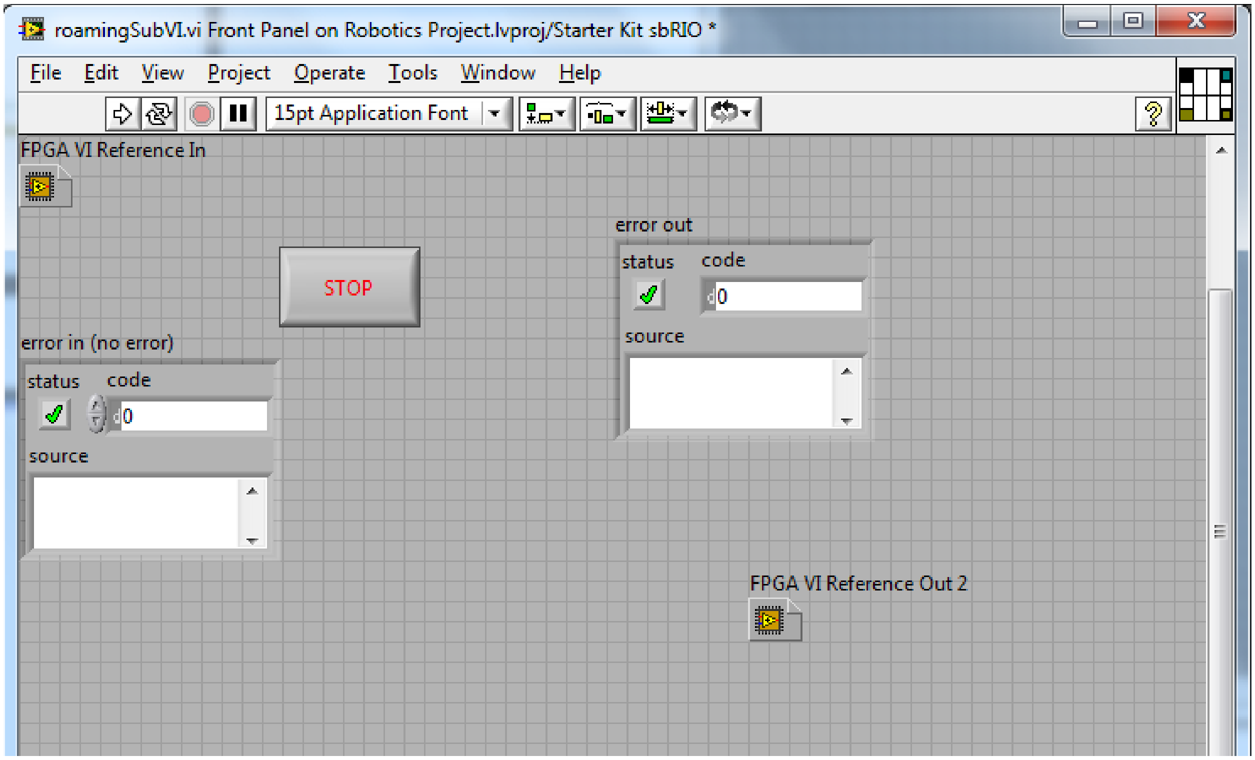

Fig. 3-5

Step 16: Create a new main VI and title it "RoamingWithStairSensor." Place an "Open FPGA reference" vi and create a constant "RIO0" for its input. Set the bit file to be from the stairSensorFPGA vi. Place a flat sequence into the loop. To find it right click on the block diagram, select structures, and then select flat sequence.

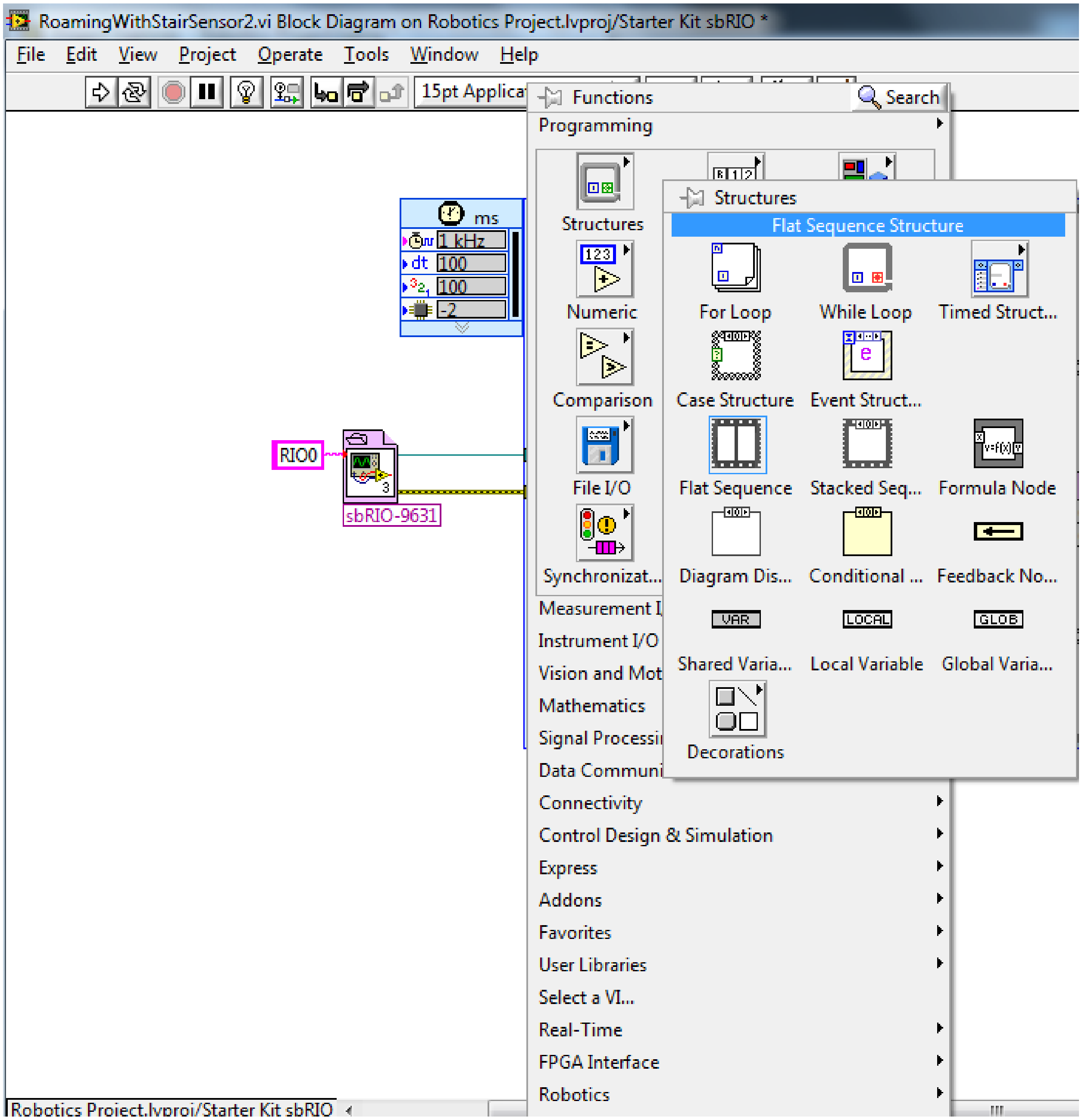

Fig. 3-6

Step 17: In the first block of the flat sequence, we will place the subVI we created earlier. To do this, right click inside the block of the flat sequence, select “Select a VI…” and then browse your files until you find your roamingSubVI.vi. Fig. 3-6 shows how to select a subVI also. Connect the FPGA reference and error wires from the “Open FPGA reference” vi to the subVI. Hopefully no error will occur but if one does occur refer back to the warning mentioned earlier when creating the subVI.

Step 18: The subVI will only exit whenever a greater distance is seen from the infrared sensors. We will now program what will happen after the subVI exits. Add a frame to the flat sequence structure by right clicking the edge of the block and selecting add frame after.

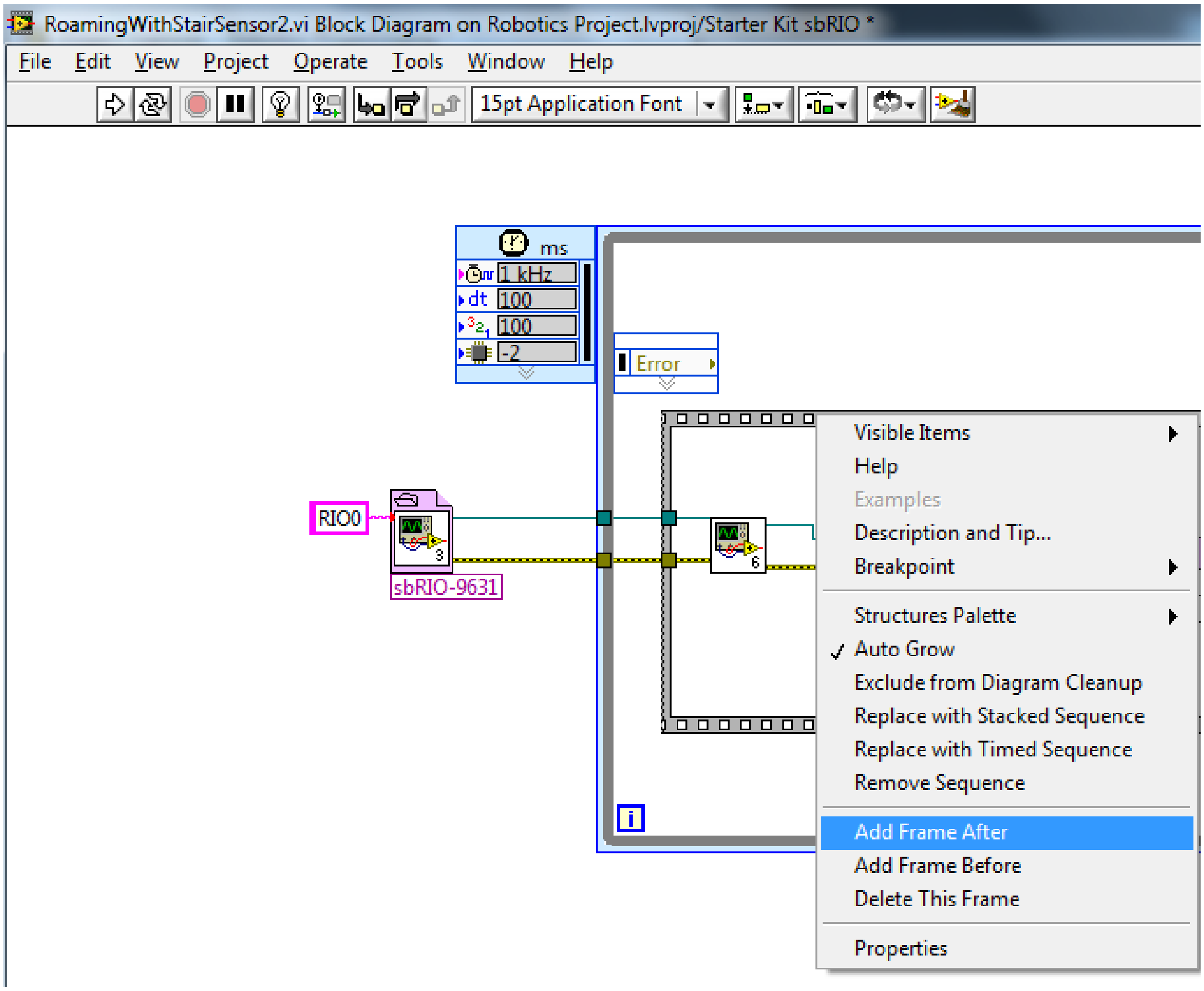

Fig. 3-7

Step 19: First the robot will back away for two seconds. To accomplish this, we will need to set the left velocity setpoint to a negative value and the right velocity setpoint to a positive value. For this solution we set the velocity setpoints to -5 rad/s and 5 rad/s. We then place a wait statement, which can be obtained by right clicking, selecting timing, and then clicking on "wait (ms)." Create a constant for its input and set it to 2000 to set it to wait for 2 seconds. This wait statement can be placed in the same block where you are changing the velocity setpoints. For this solution we have placed every part of the procedure in separate blocks.

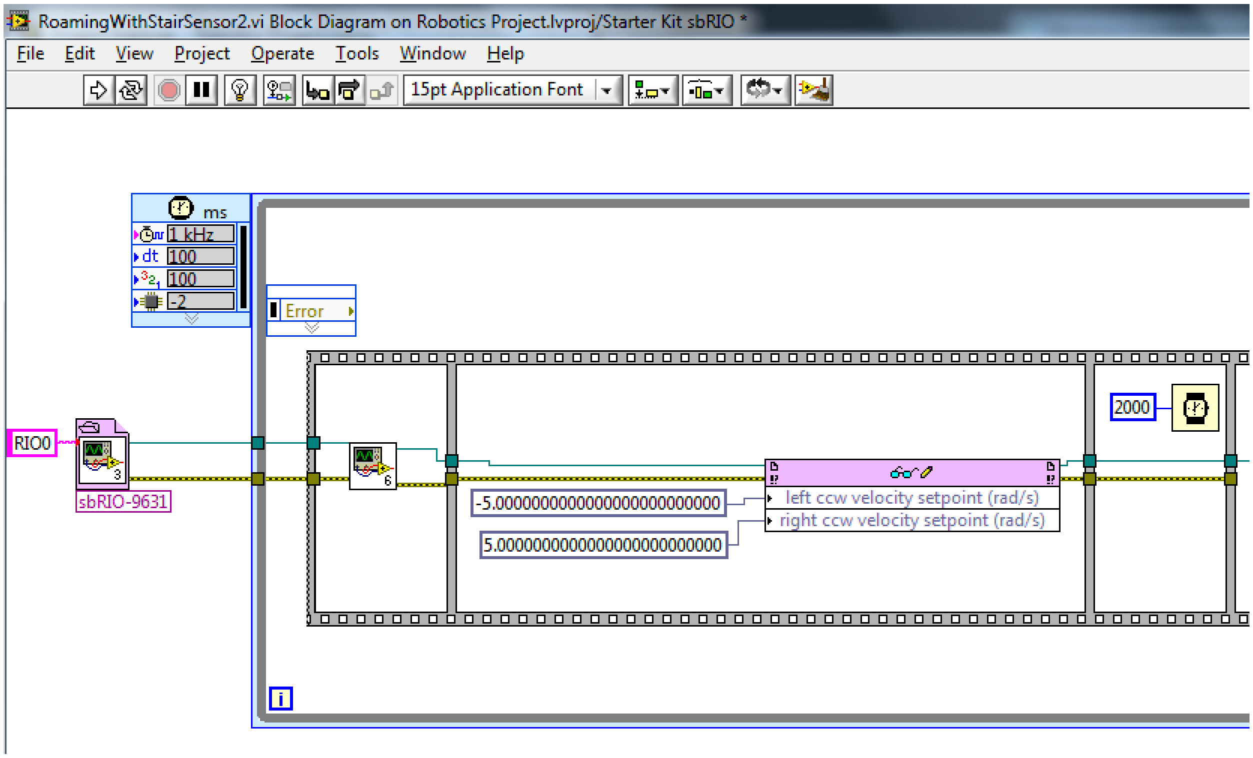

Fig. 3-8

Step 20: Lastly we will have the robot turn either 90 degrees or more. To do this, we will set the velocity setpoints to positive values and place another wait statement to have the robot turn at that velocity for that amount of time. For this solution we have chosen to set both velocity setpoints to 5 rad/s. The time that we have set to make it turn to the right is 3000 ms or 3 seconds.

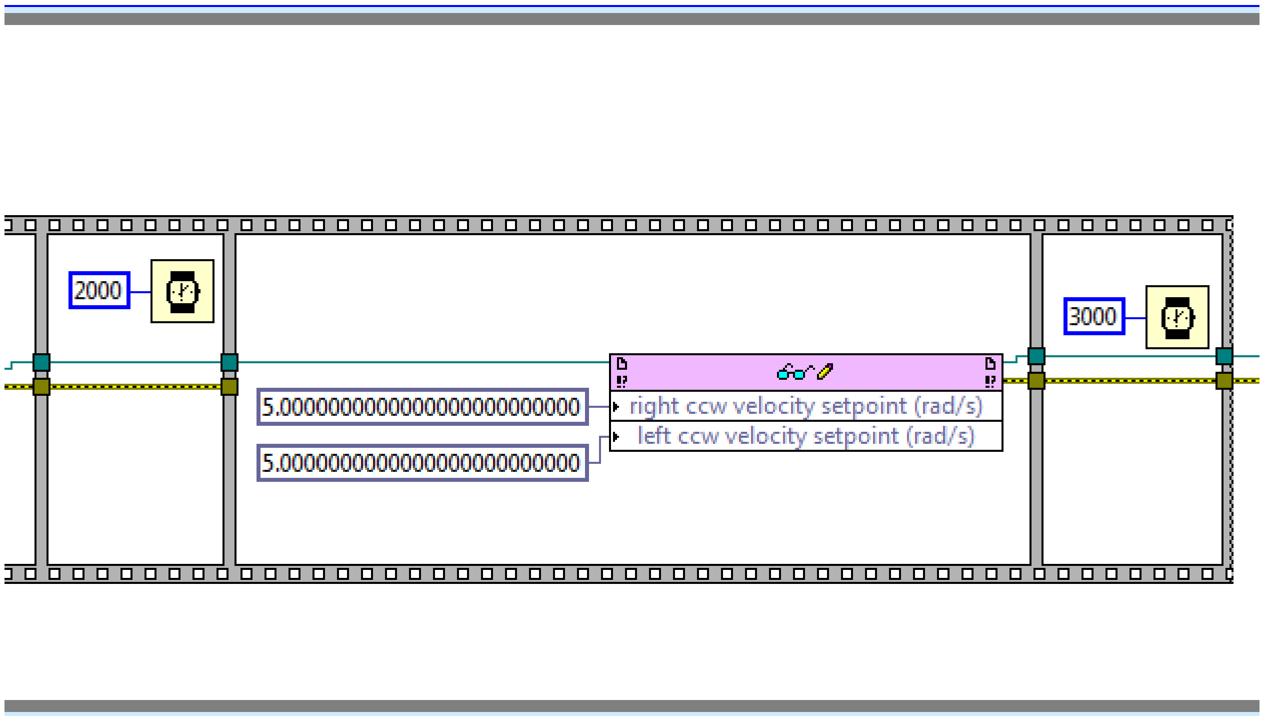

Fig. 3-9

Step 21: Lastly we just need to close the FPGA reference. Thus, place a "close FPGA reference" vi outside of the loop and wire the FPGA reference and error out wires to that vi. This completes the code for this lab.

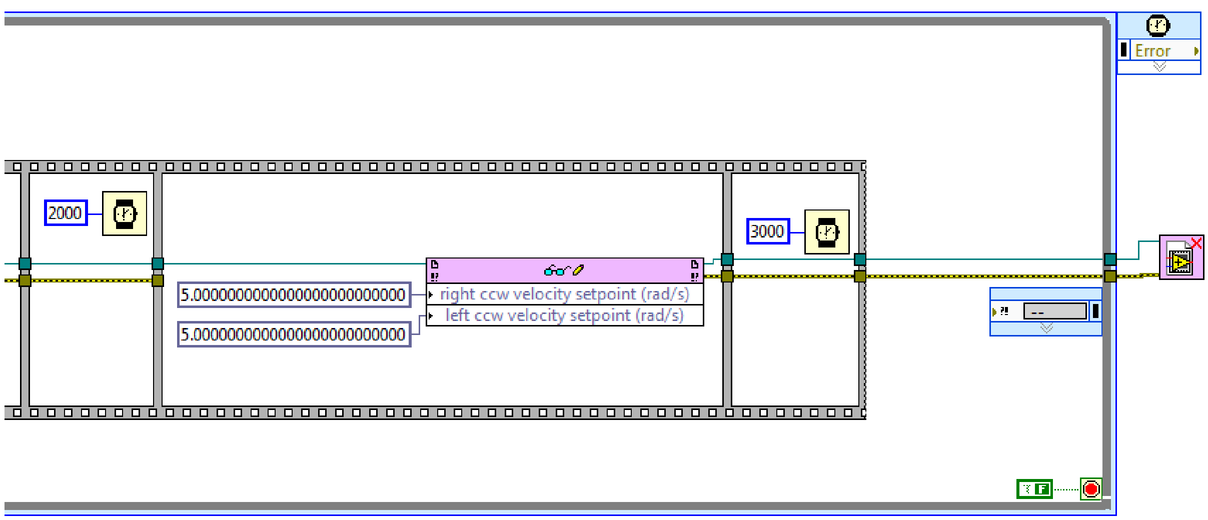

Fig. 3-10