THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

LabVIEW & NI myRIO

Design Real Systems, Fast

Mostafa Salama

Vehicle and Robotics Engineering Laboratory - VREL

University of Alabama at Birmingham - UAB

NATIONAL INSTRUMENTS™

VEHICLE & ROBOTICS ENGINEERING LABORATORY

THE UNIVERSITY OF
ALABAMA AT BIRMINGHAM

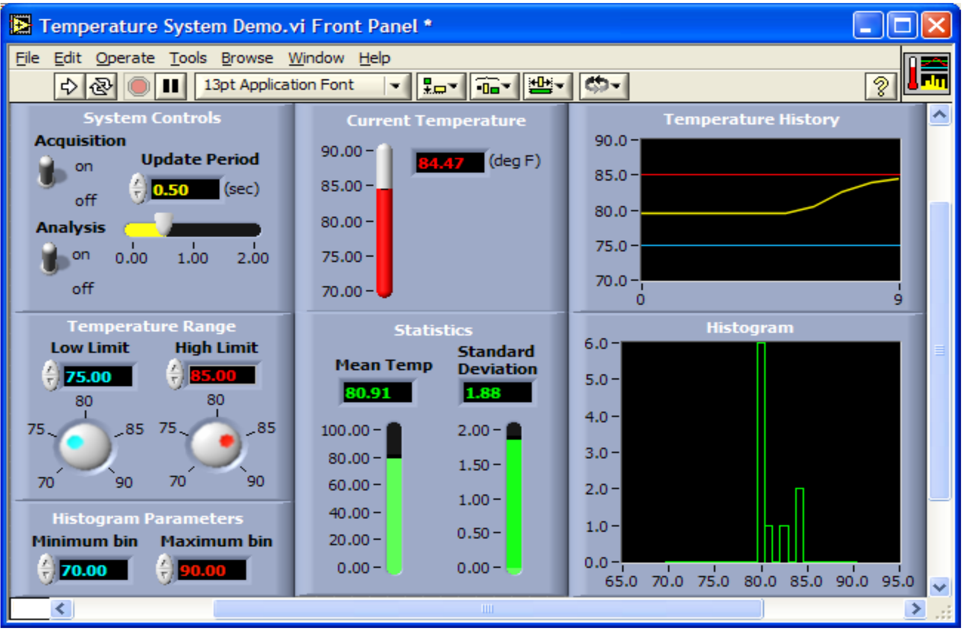
Knowledge that will change your world

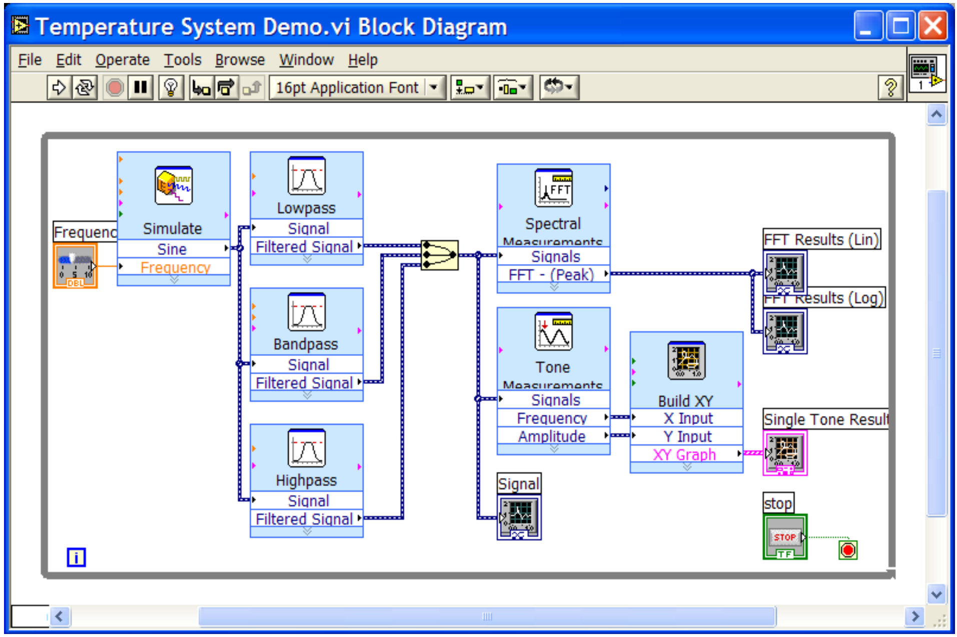
Part 1

LabVIEW

What is LabVIEW?

Laboratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench





- Compiled graphical development environment
- Development time reduction of four to ten times
- Tools to acquire, analyze, and present your data

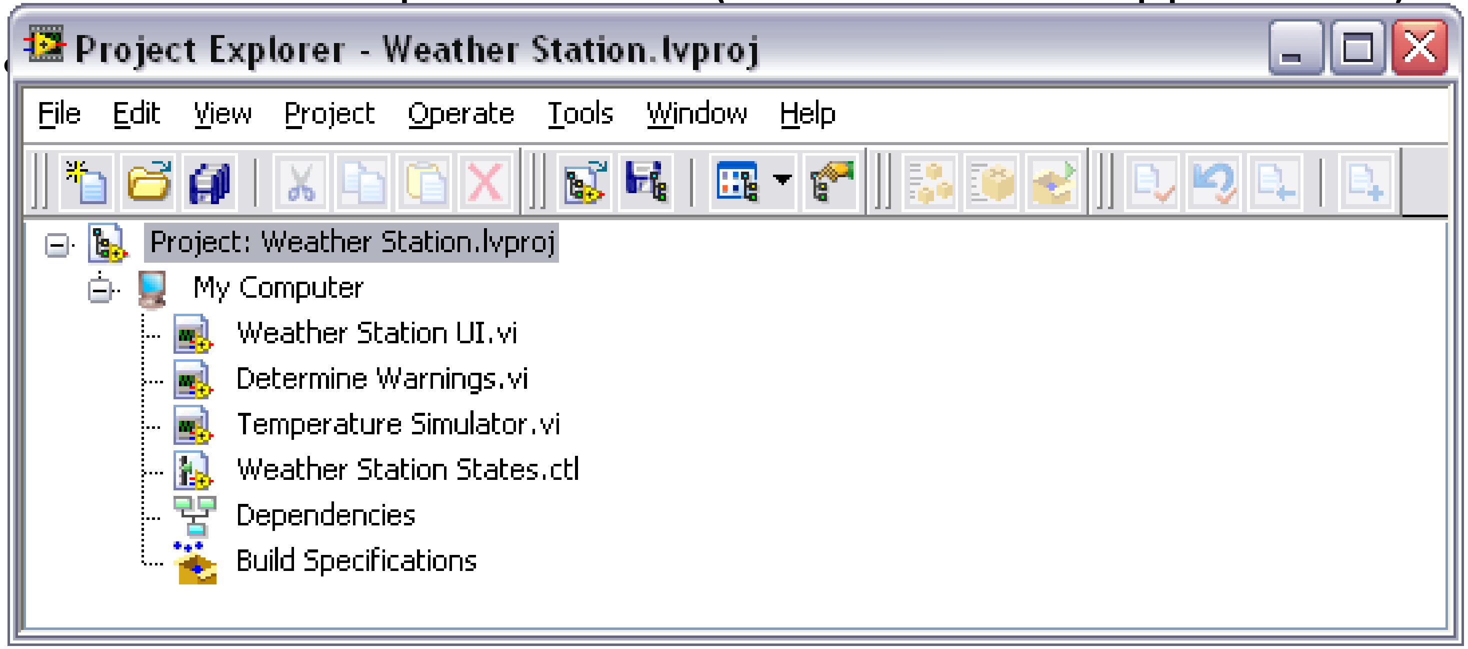
NATIONAL INSTRUMENTS™

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

VEHICLE & ROBOTICS ENGINEERING LABORATORY

Project Explorer

- Use LabVIEW Projects to:
  - Group LabVIEW files and non-LabVIEW files
  - Create build specifications (i.e. stand-alone applications)



What is a Virtual Instrument (VI)?

Answer: a LabVIEW program

1. Front Panel

User interface (UI)

- Controls = inputs
- Indicators = outputs

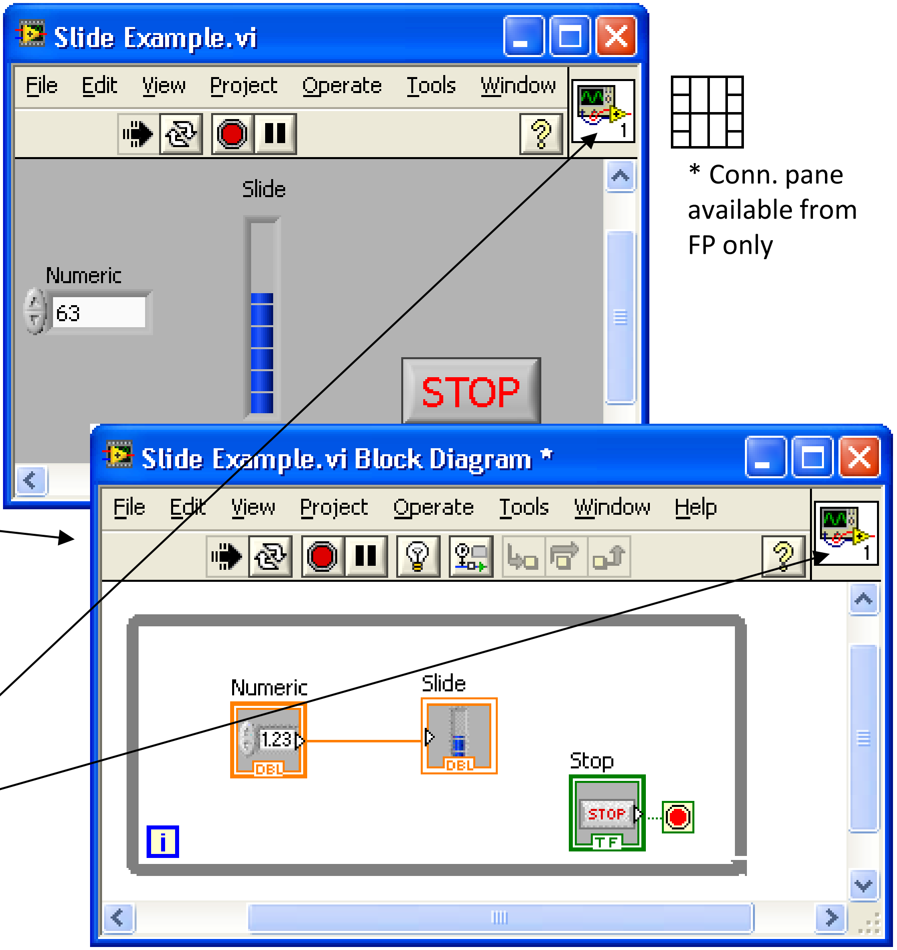
2. Block Diagram

Graphical source code

- Data travels on wires from control terminals through functions to indicator terminals
- Blocks execute by data flow

3. Icon/Connector Pane

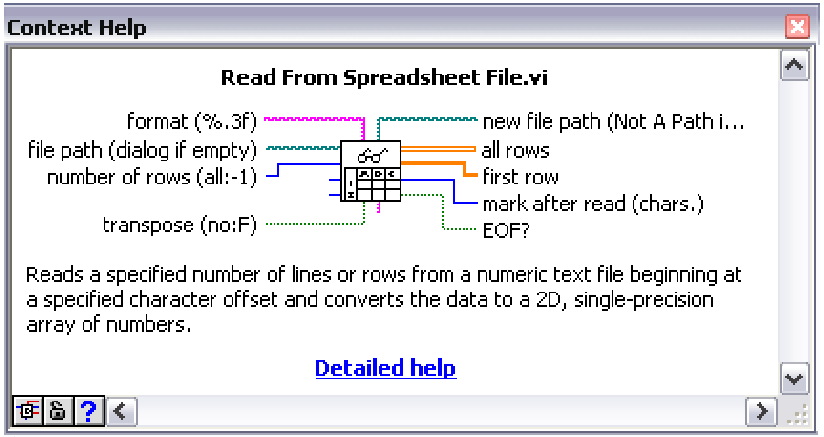
- Graphical representation of a VI
- Means of connecting VIs (subVIs)


Slide Example.vi
File Edit View Project Operate Tools Window
Slide
Numeric
63
STOP
Slide Example.vi Block Diagram *
File Edit View Project Operate Tools Window Help
Numeric
1.23
DBL
Slide
DBL
Stop
STOP
TF
i

* Conn. pane available from FP only

Front Panel Toolbar

It is best not to use the **Abort** button because you run the risk of not closing references or cleaning up memory correctly

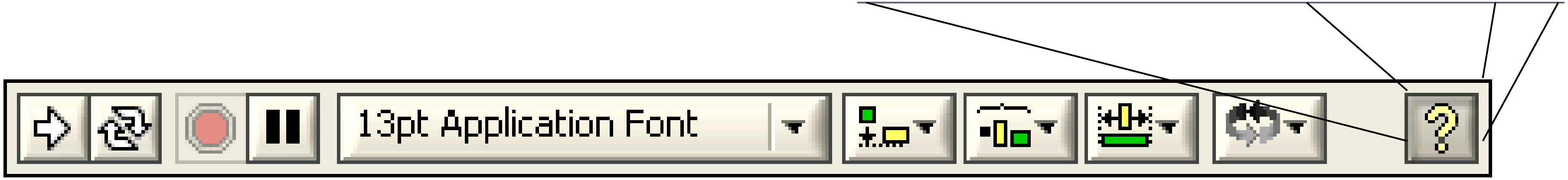


Context Help

Read From Spreadsheet File.vi

Reads a specified number of lines or rows from a numeric text file beginning at a specified character offset and converts the data to a 2D, single-precision array of numbers.

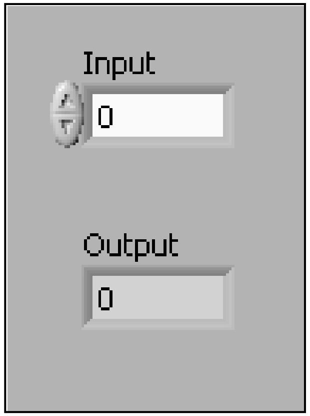
Detailed help



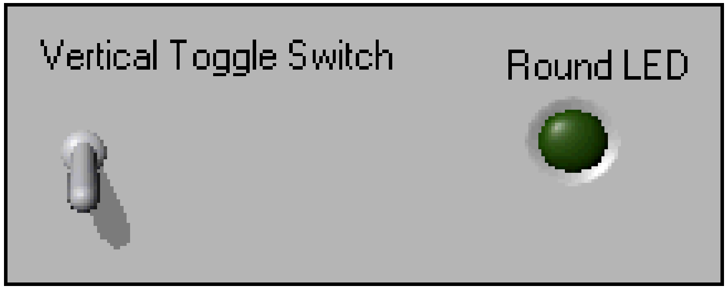
- Run
- Run Continuously
- Abort
- Pause
- Text Settings
- Context Help

Front Panel Controls and Indicators

Numeric

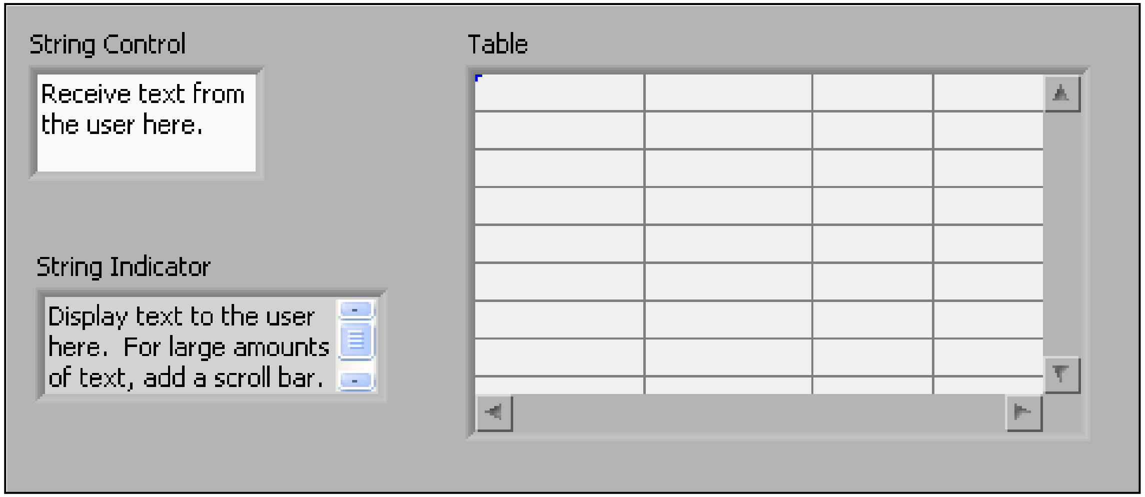


Boolean



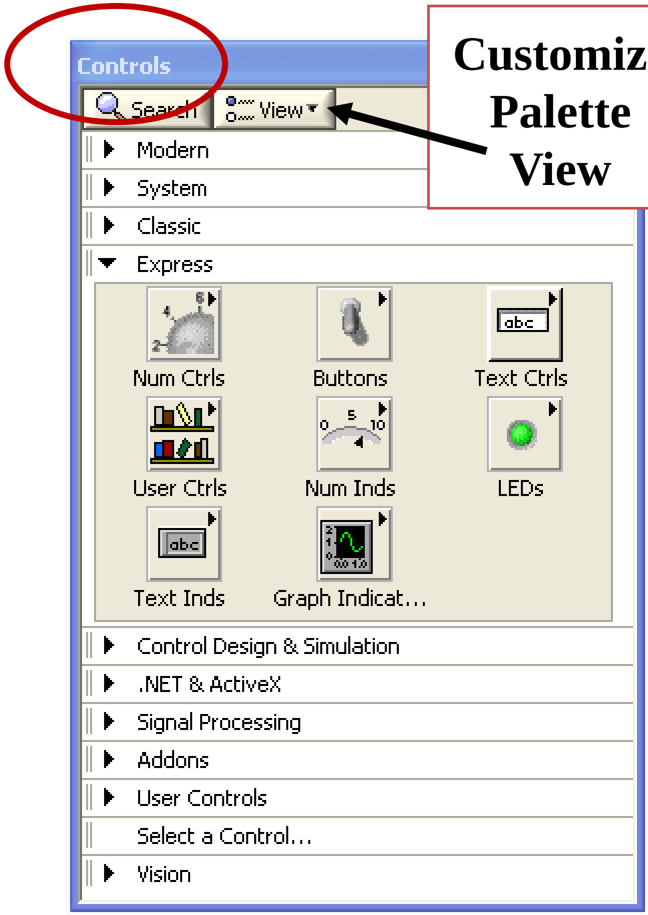
Right click!

String



Receive text from the user here.

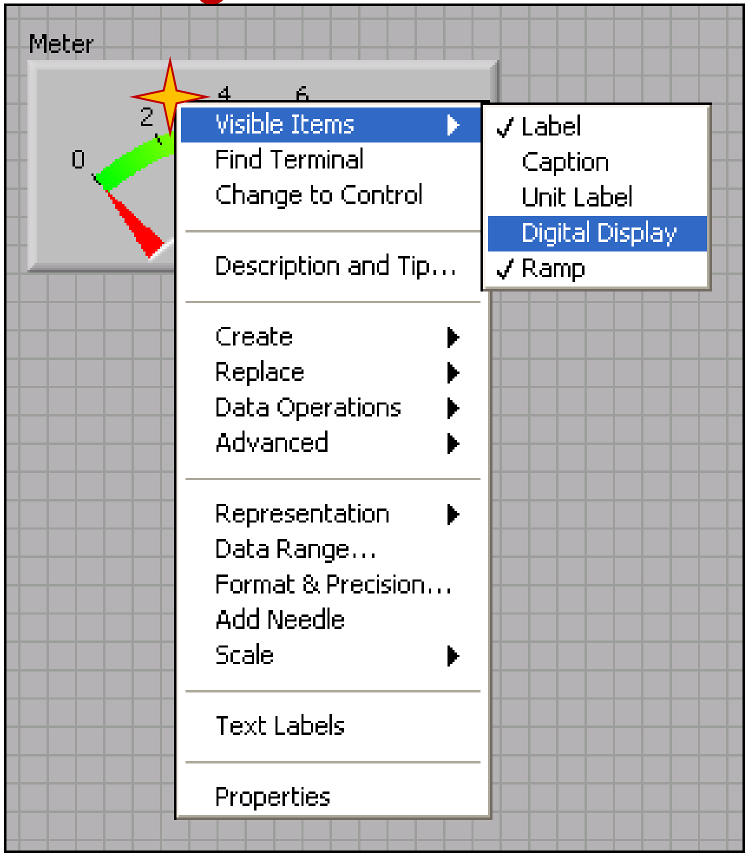
Display text to the user here. For large amounts of text, add a scroll bar.

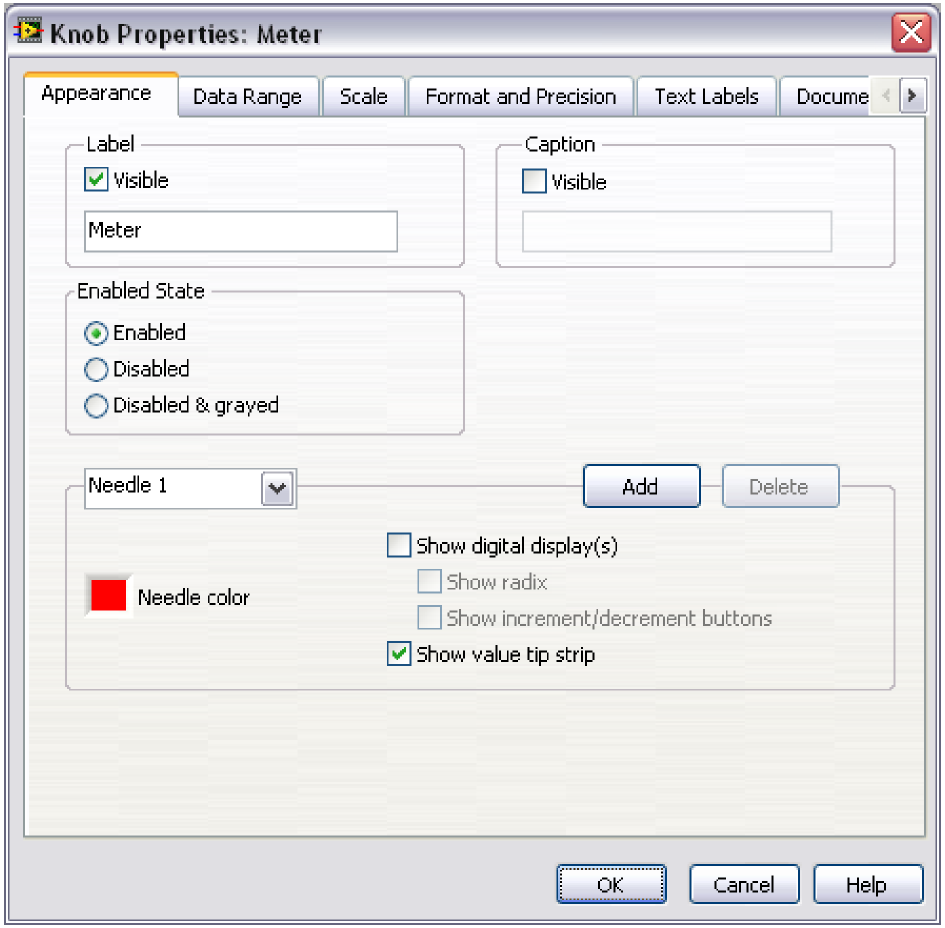


Customize Palette View

Shortcut Menus and Properties Dialog

Right Click!





Label
☑ Visible
Meter

Caption
☐ Visible

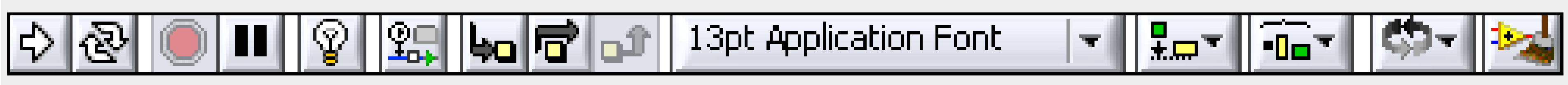
Enabled State
● Enabled
○ Disabled
○ Disabled & grayed

Needle 1

🟥 Needle color

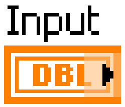
☐ Show digital display(s)
☐ Show radix
☐ Show increment/decrement buttons
☑ Show value tip strip

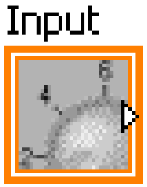
Block Diagram Toolbar

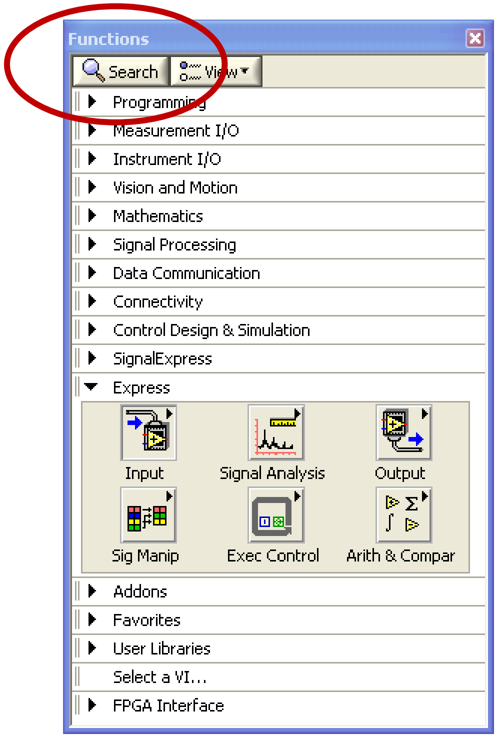


- Run
- Run Continuously
- Abort
- Pause
- Highlight Execution
- Retain Wire Values
- Text Settings
- Clean Up Block Diagram

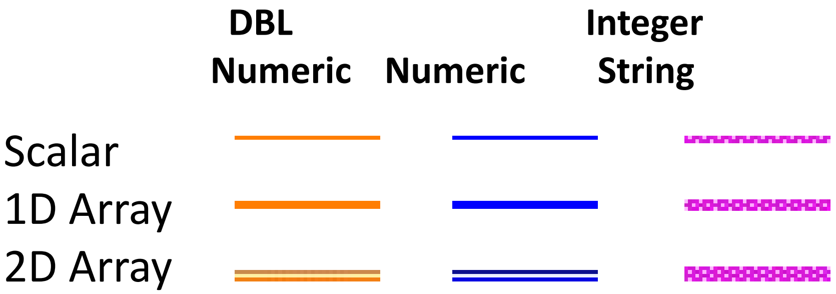
Block Diagram

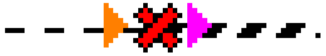






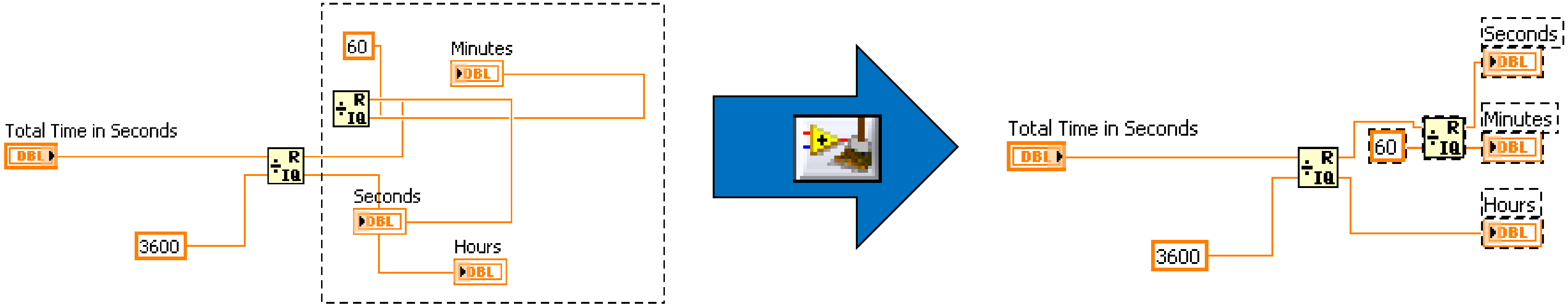
- Terminals
  - Block Diagram appearance of front panel objects
  - Entry & exit ports that exchange information between the front panel and block diagram
  - Analogous to parameters and constants in text- based programming languages
- Wires
  - Transfer data between block diagram objects
  - Wires are different colors, styles, and thicknesses, depending on data type
  - A broken wire appears as a dashed black line with a red X in the middle



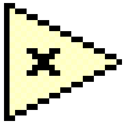


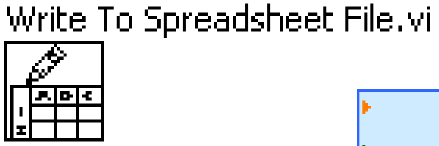
Block Diagram: Wiring Tips

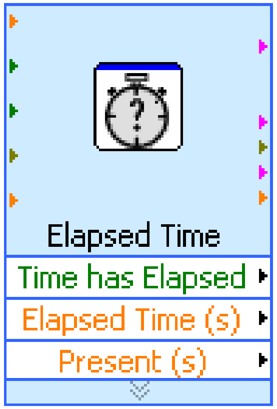
- Press <Ctrl>-B to delete all broken wires
- Right-click and select **Clean Up Wire** to reroute the wire
- Use the Clean Up Diagram tool to reroute multiple wires and objects to improve readability
  - Select a section of your block diagram
  - Click the Clean Up Diagram button on the block diagram toolbar (or <Ctrl>-U)



Block Diagram







- Nodes
  - Objects on the block diagram that have inputs and/or outputs and perform operations when a VI runs
  - Analogous to statements, operators, functions, and subroutines in text-based programming languages

Functions

- Fundamental operating elements of LabVIEW
- Do not have front panels or block diagrams, but do have connector panes
- Double-clicking a function only selects the function – does not open it like a VI
- Has a pale yellow background on its icon

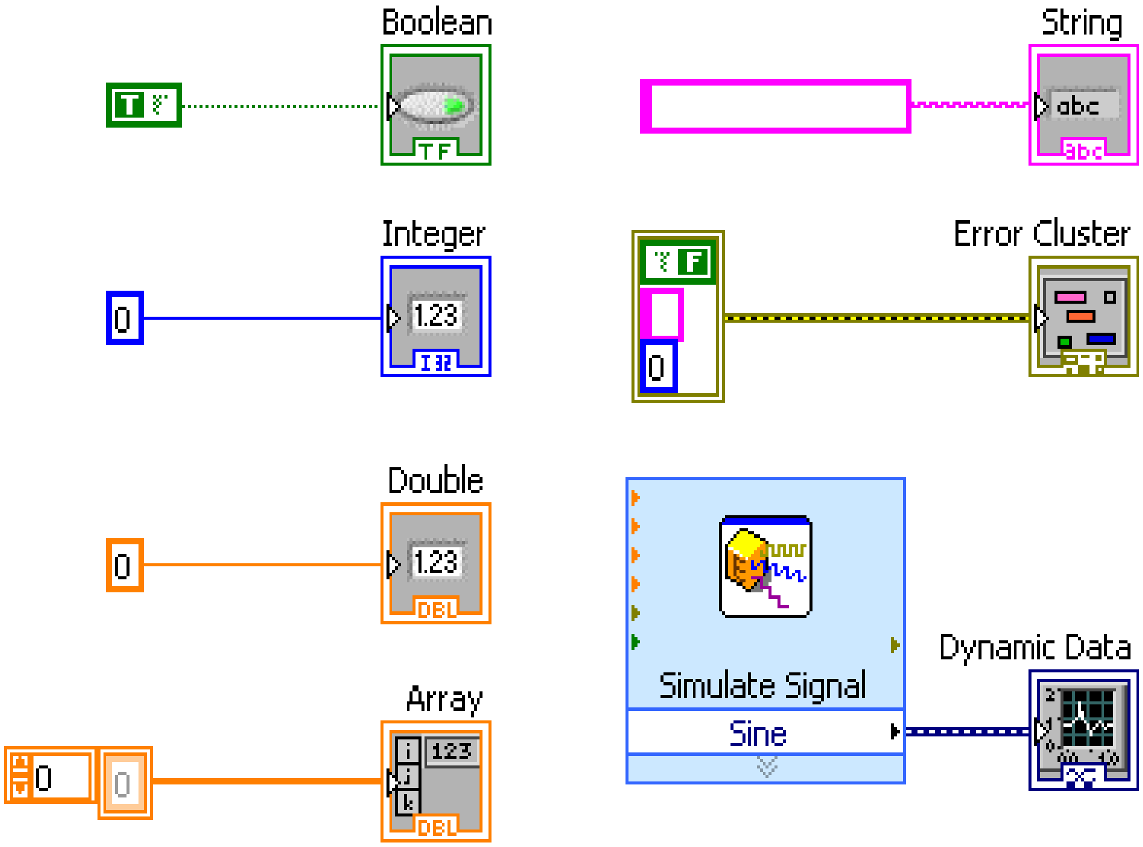
subVIs

- VI that you build to use inside another VI
- Any VI has potential to become a subVI
- Double-clicking a subVI will open it (exception: Express Vis- config. window opens)
- Icon represents subVI in main VI

Structures

- While loops, for loops, event structures
- More discussion later

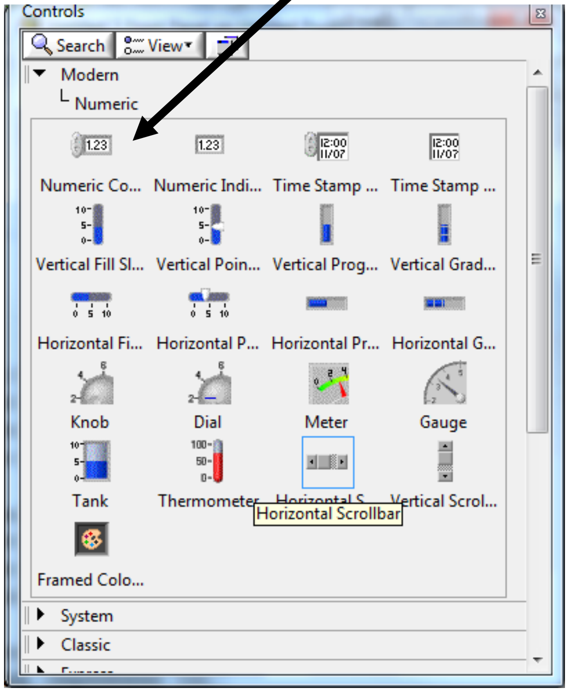
Common Data Types Found in LabVIEW

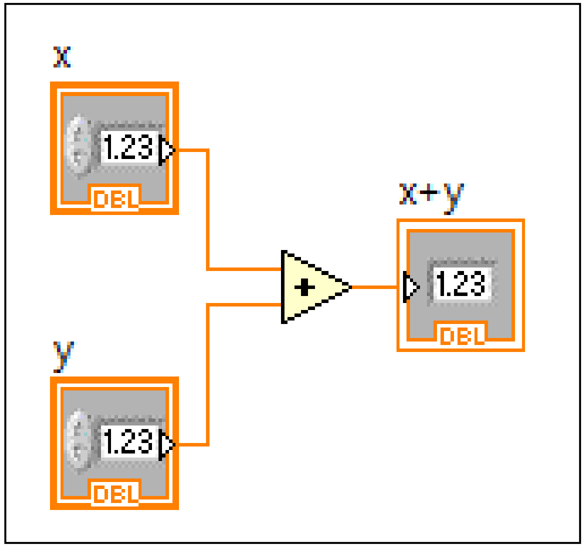


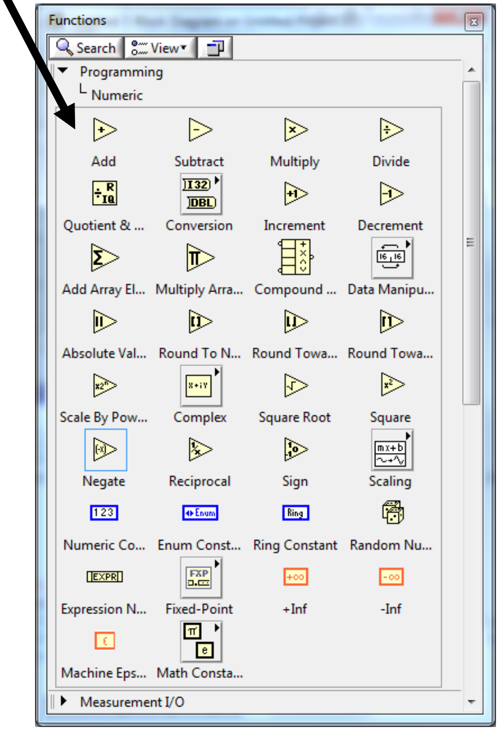
Numeric Controls and Functions

- (Front Panel) From the **Controls»Modern»Numeric** subpalette, select the **Numeric Control** icon.

(Block Diagram) From the **Functions»Programming»Numeric** subpalette, select the **Add** icon.

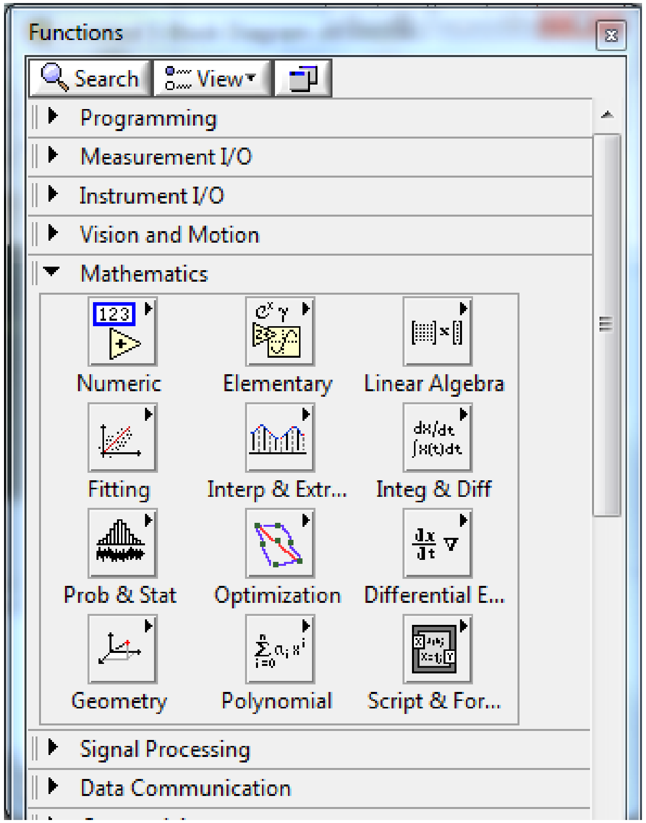


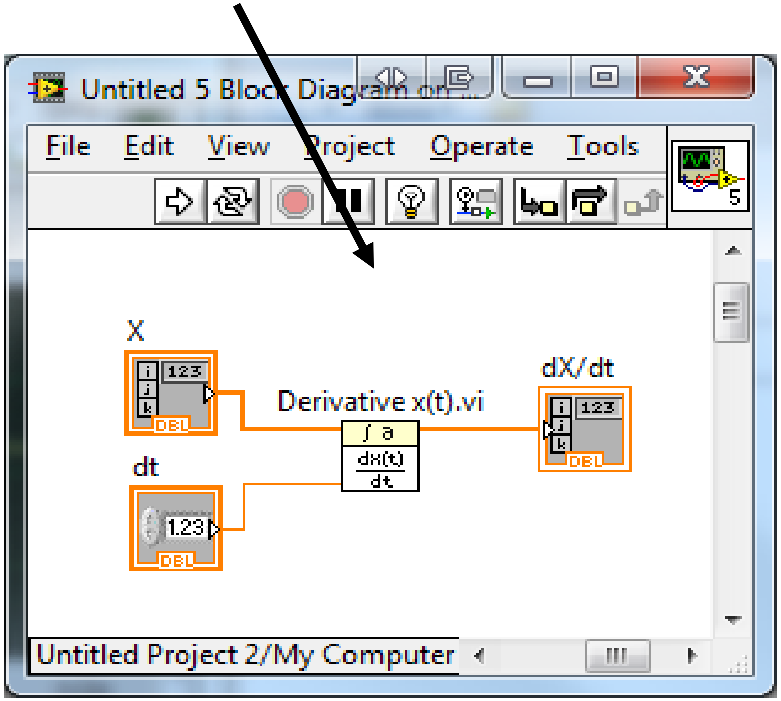




Mathematical Operations

- (Block Diagram) From the **Functions»Mathematics»Integration and Differentiation** subpalette, select the **Derivative x(t).vi**

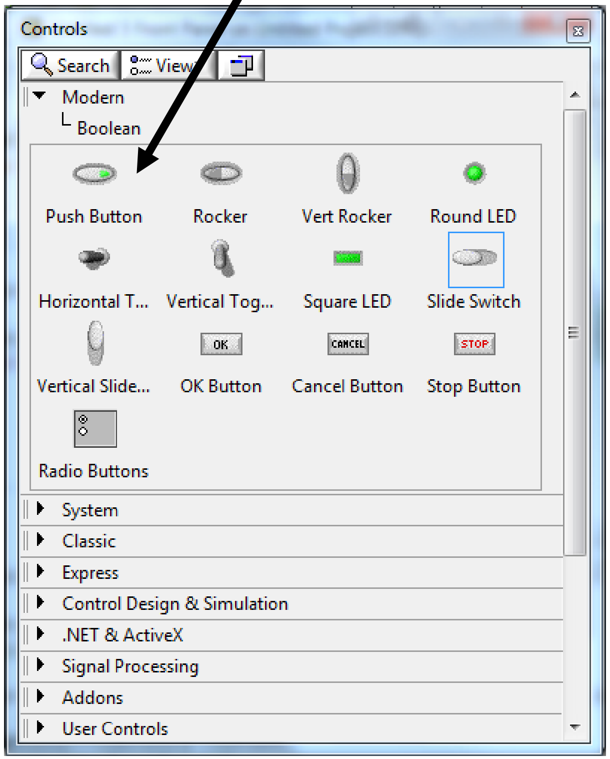


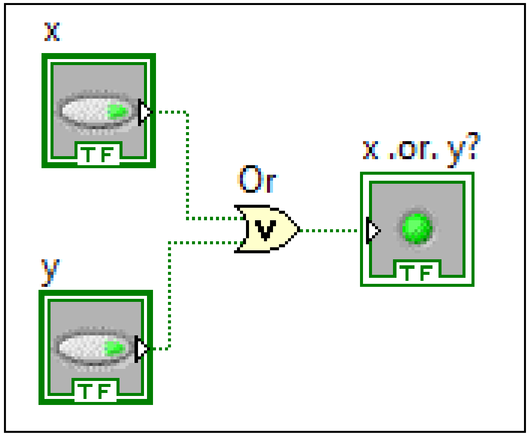


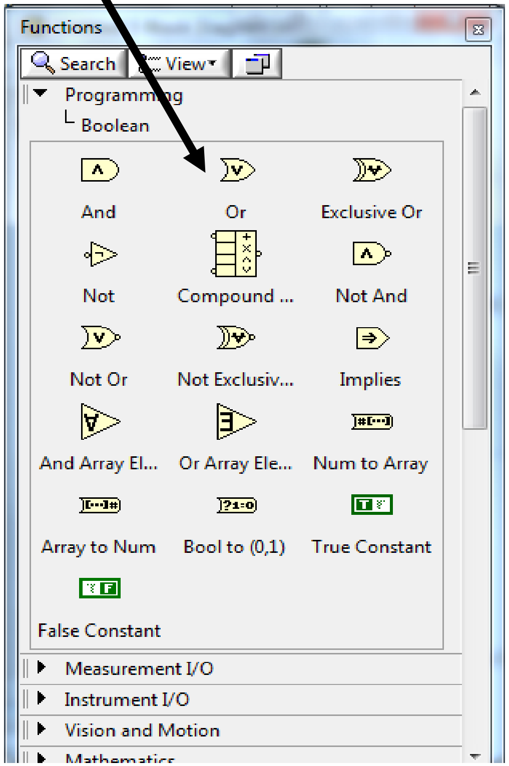
Boolean Controls and Functions

- (Front Panel) From the **Controls»Modern»Boolean** subpalette, select the **Push Button** icon.

(Block diagram) From the **Function»Programming»Boolean** subpalette, select the **OR** icon.

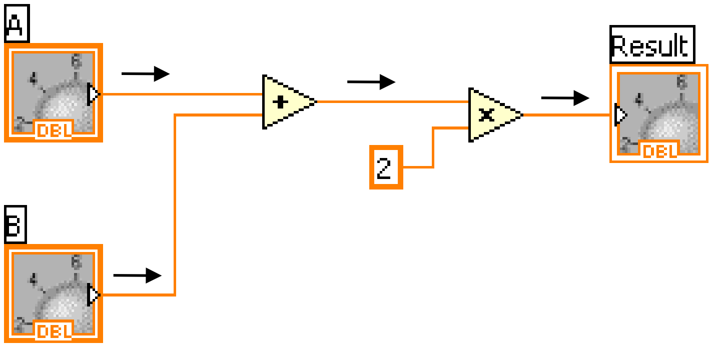


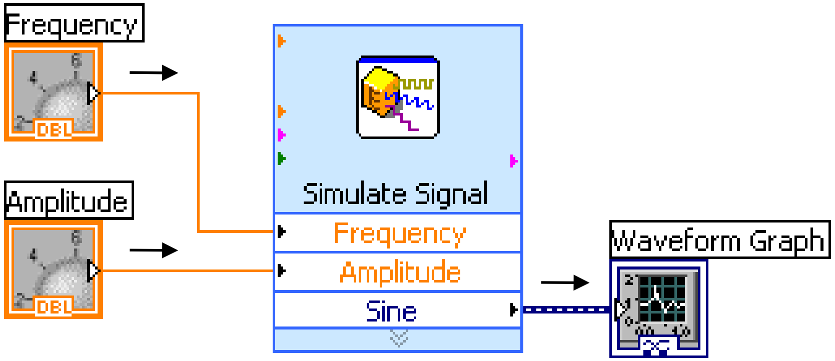




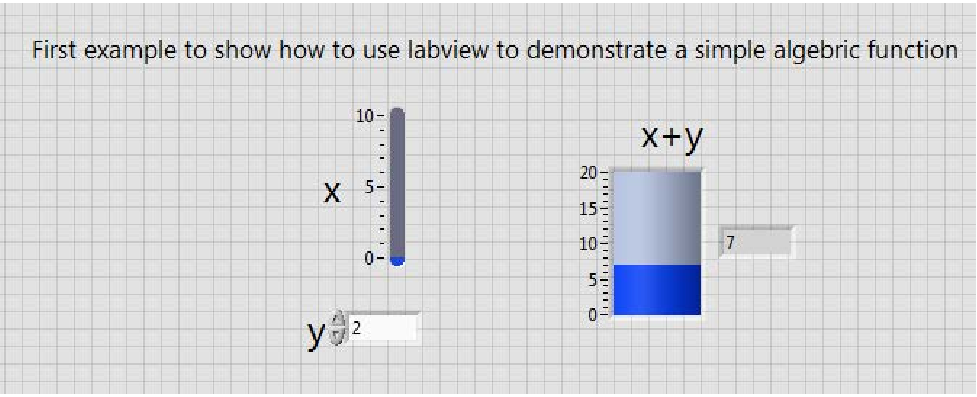
Data Flow

- Block diagram execution is dependent on the flow of data
- Block diagram does NOT execute left to right
- Node executes when data is available to ALL input terminals
- Nodes supply data to all output terminals when done
- If the computer running this code had multiple processors, these two pieces of code could run independently without additional coding

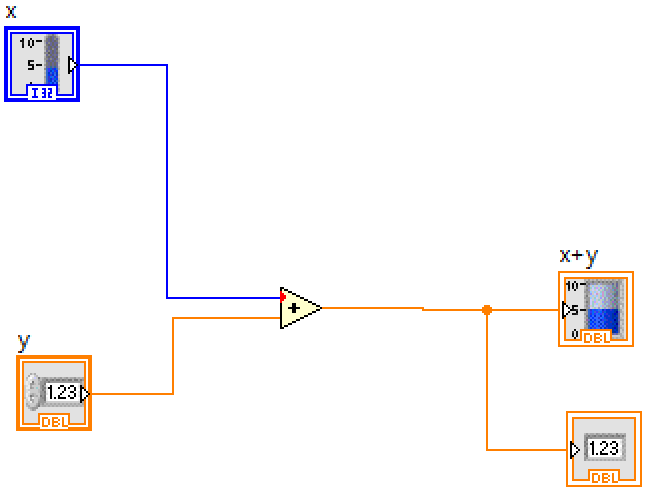




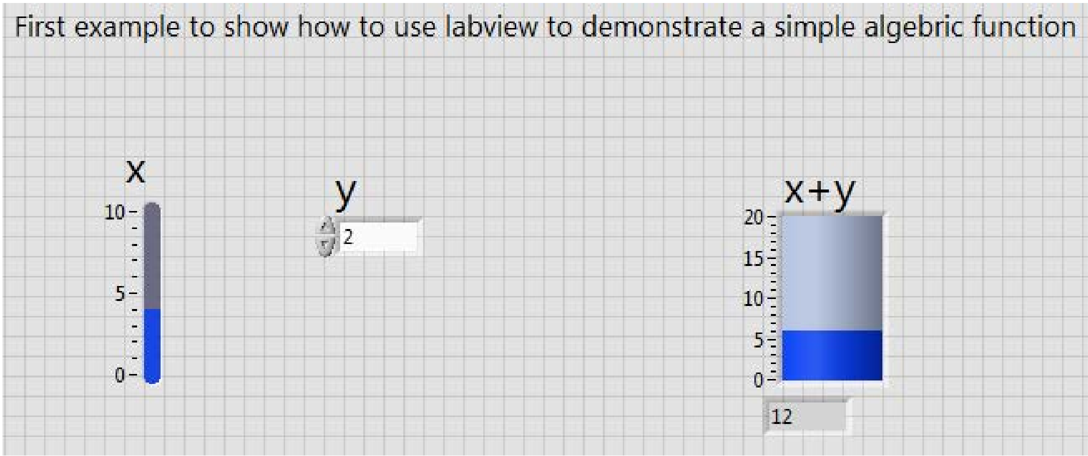
Tutorial 1: Simple Algebraic Equation



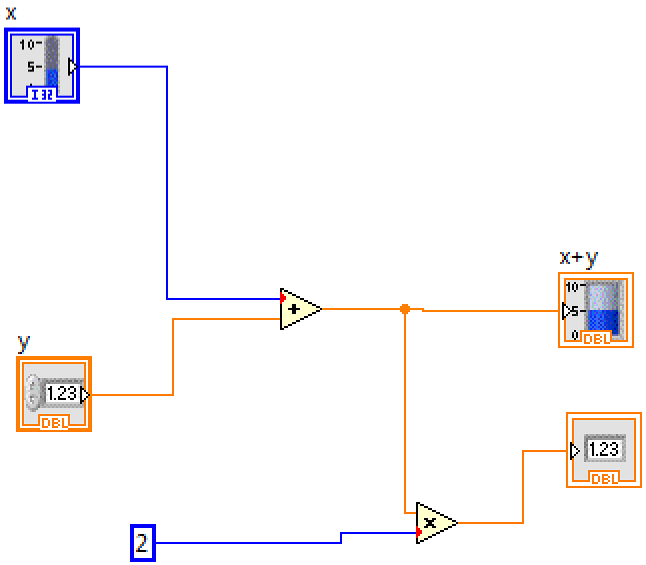
First example to show how to use labview to demonstrate a simple algebric function



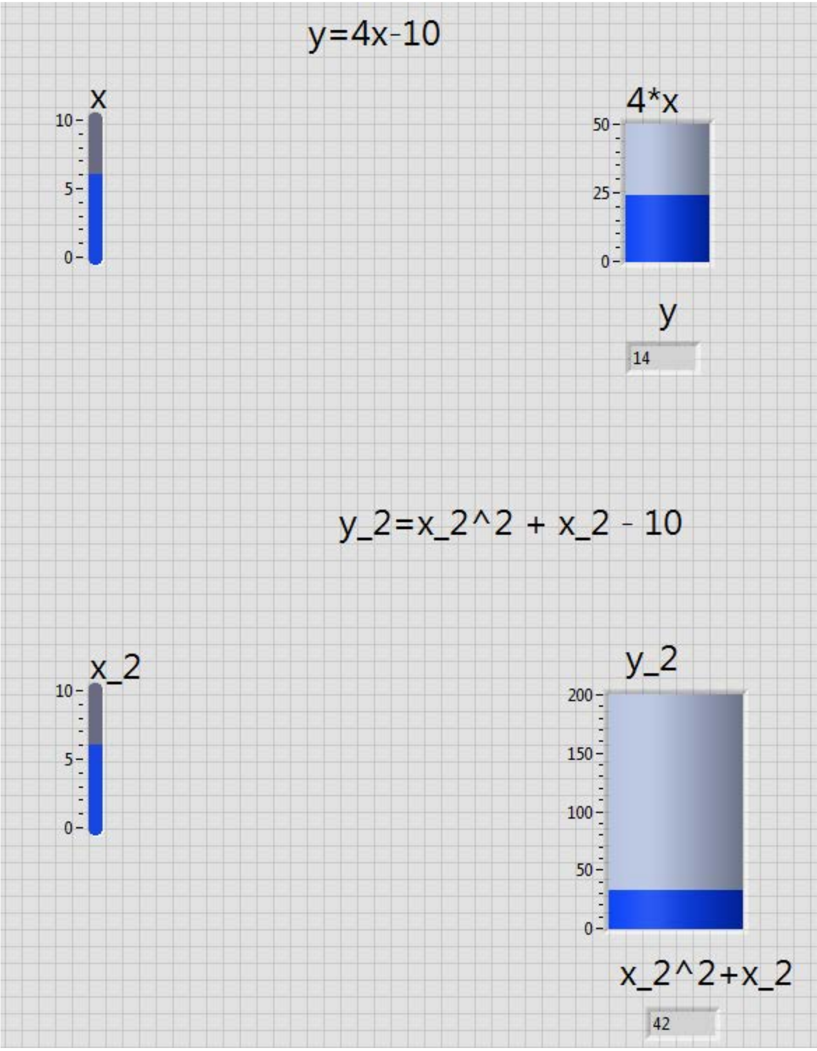
Tutorial 2: Simple Algebraic Equation

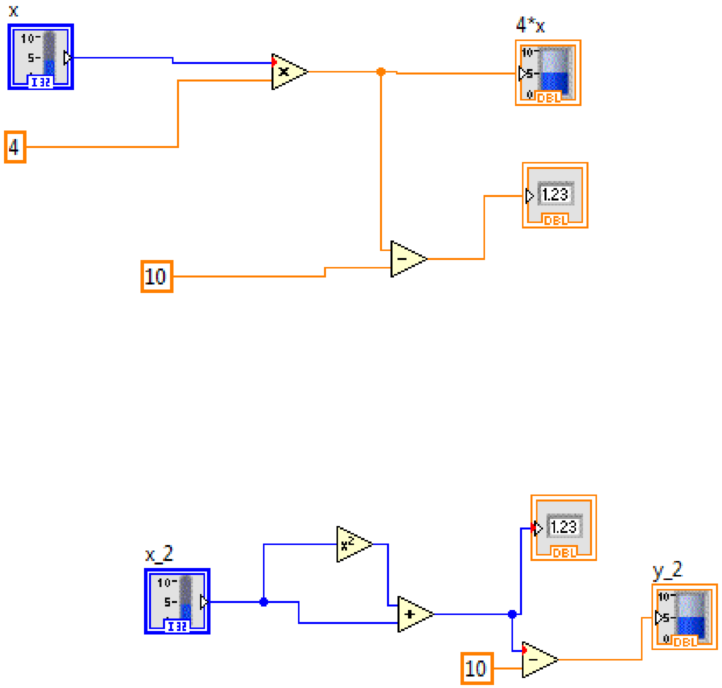


First example to show how to use labview to demonstrate a simple algebric function

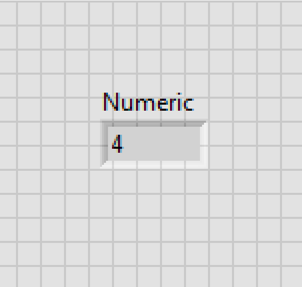


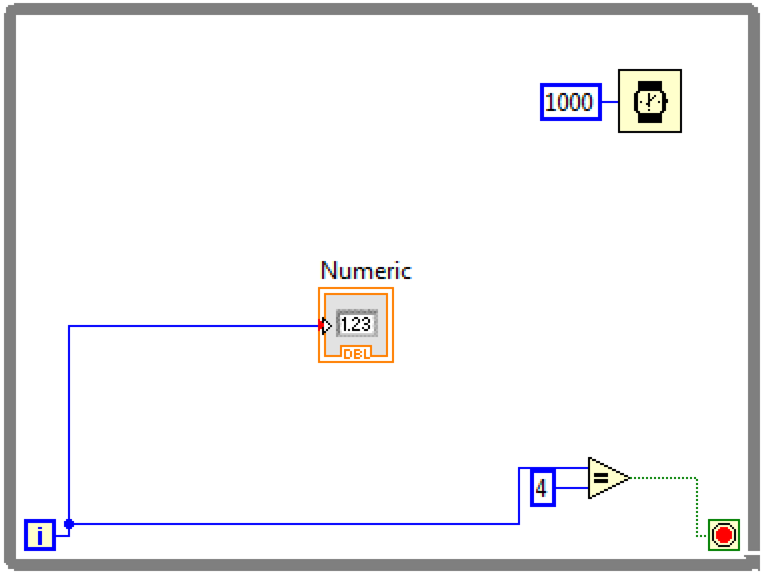
Tutorial 3: Parallel Computations



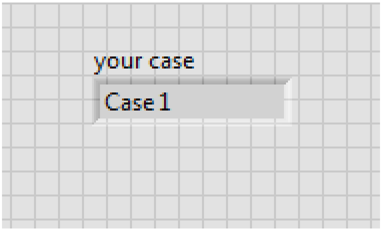


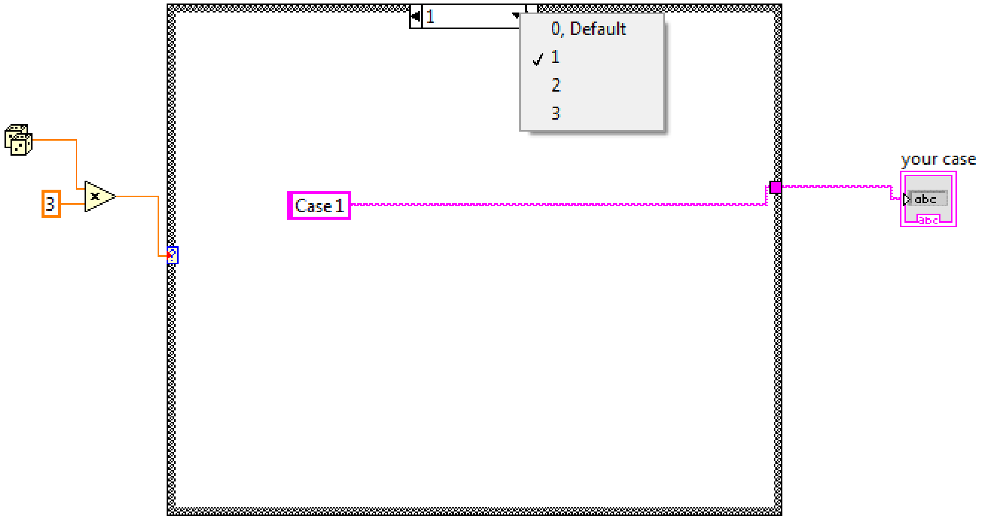
Tutorial 4: Loops



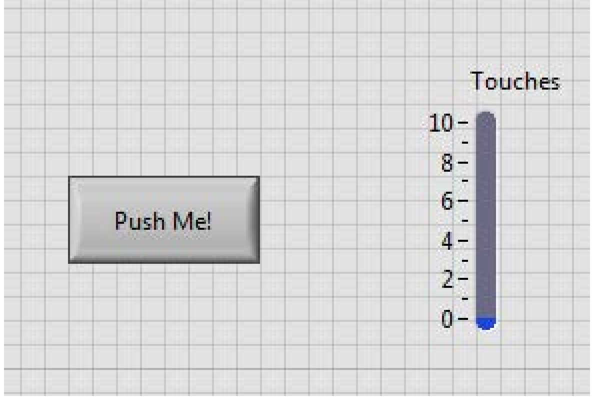


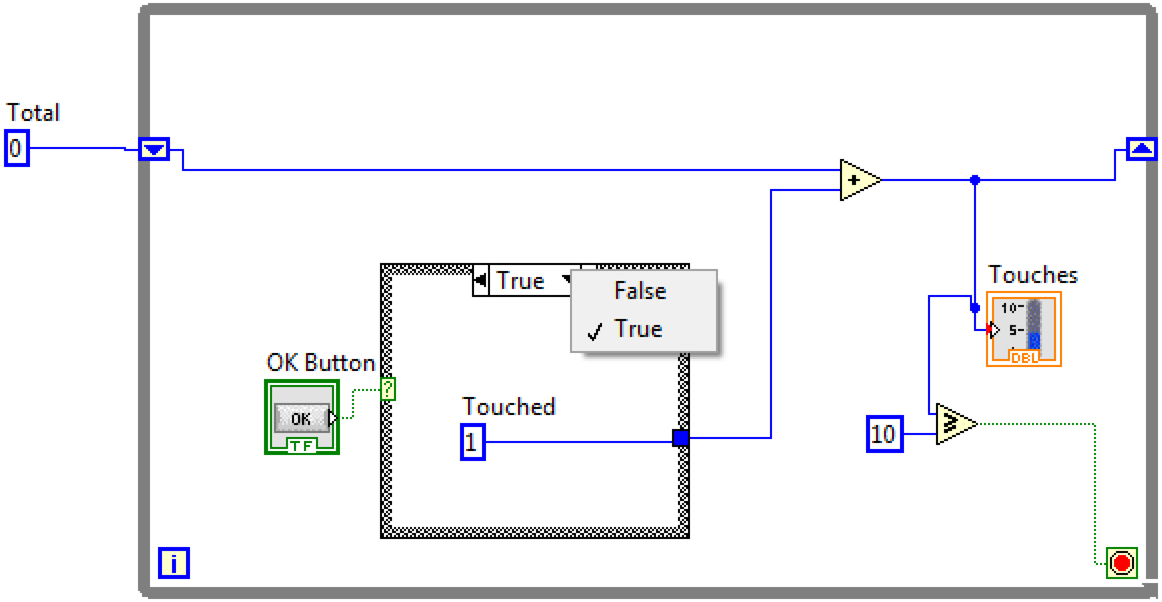
Tutorial 5: Case Statements



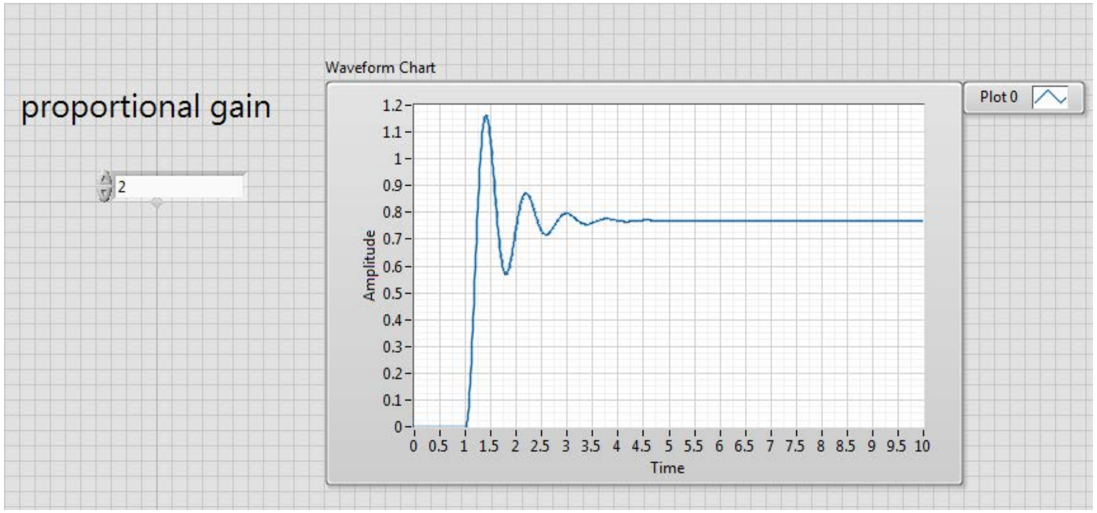


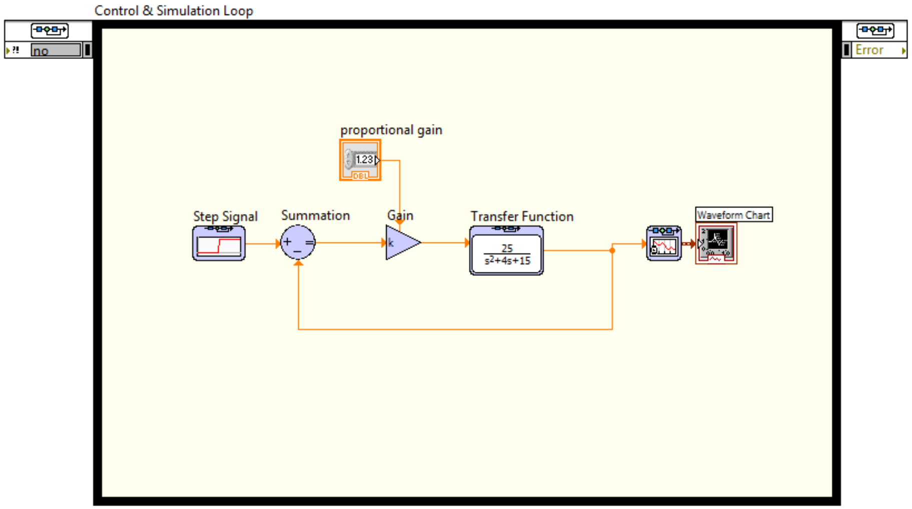
Tutorial 6: Push button Counter



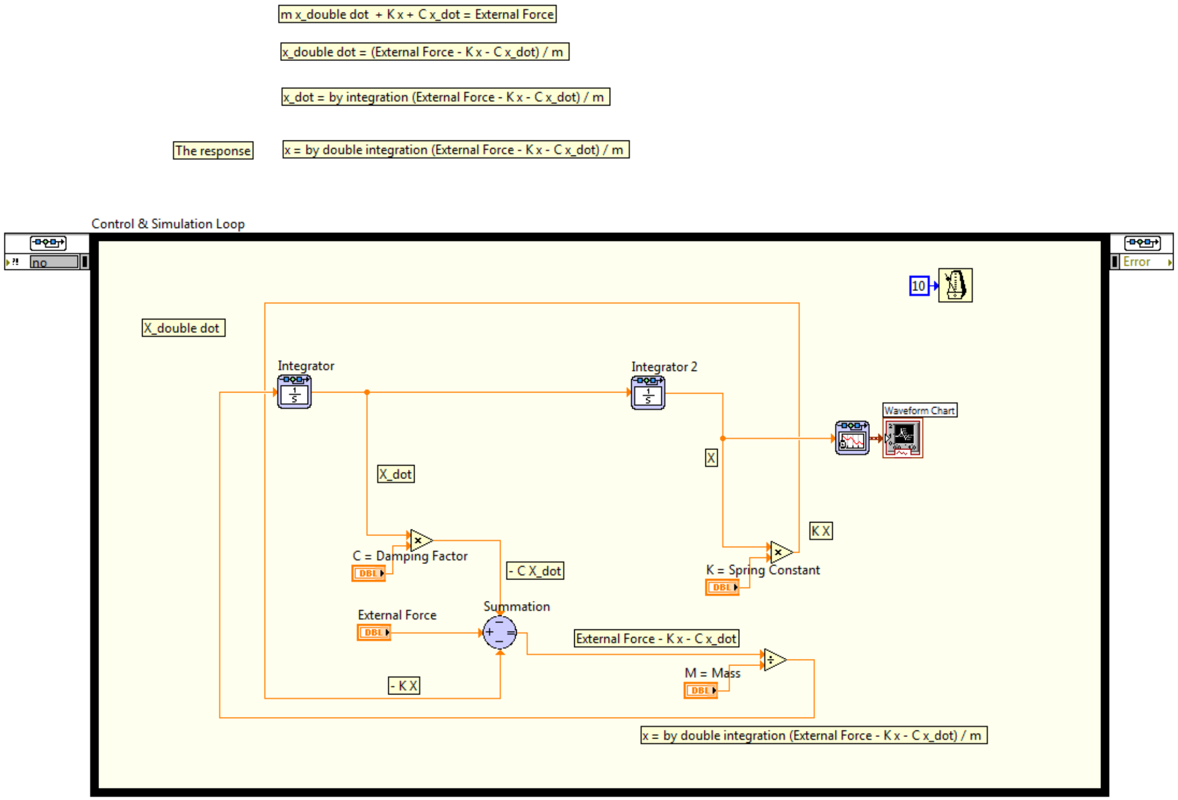


Tutorial 7: Control & Simulation Module





Tutorial 8: Differential Equations



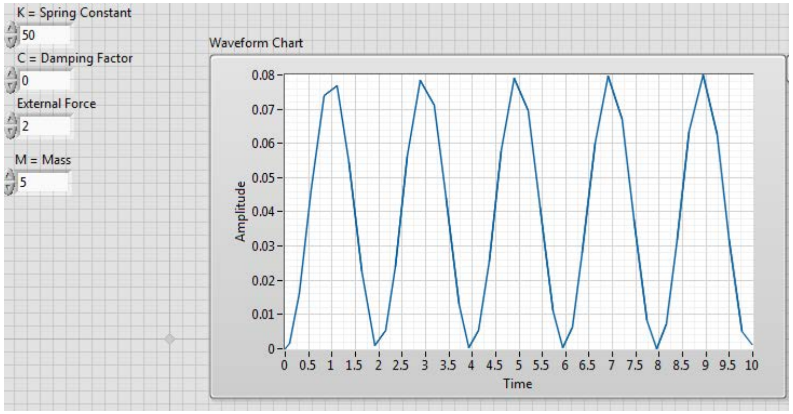
m x_double dot + K x + C x_dot = External Force

x_double dot = (External Force - K x - C x_dot) / m

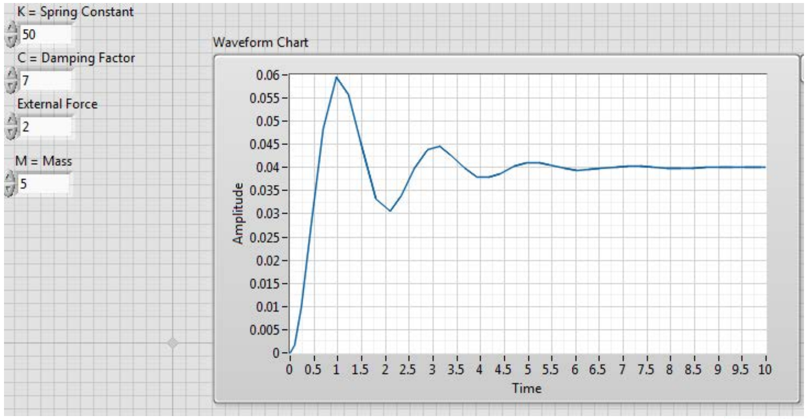
x_dot = by integration (External Force - K x - C x_dot) / m

The response

x = by double integration (External Force - K x - C x_dot) / m



K = Spring Constant: 50
C = Damping Factor: 0
External Force: 2
M = Mass: 5



K = Spring Constant: 50
C = Damping Factor: 7
External Force: 2
M = Mass: 5

NATIONAL INSTRUMENTS™

UAB THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

VREL VEHICLE & ROBOTICS ENGINEERING LABORATORY

THE UNIVERSITY OF
ALABAMA AT BIRMINGHAM
Knowledge that will change your world

Part 2

NI myRIO

The Problem

- Today's tools do not let engineering students accomplish real projects within one semester

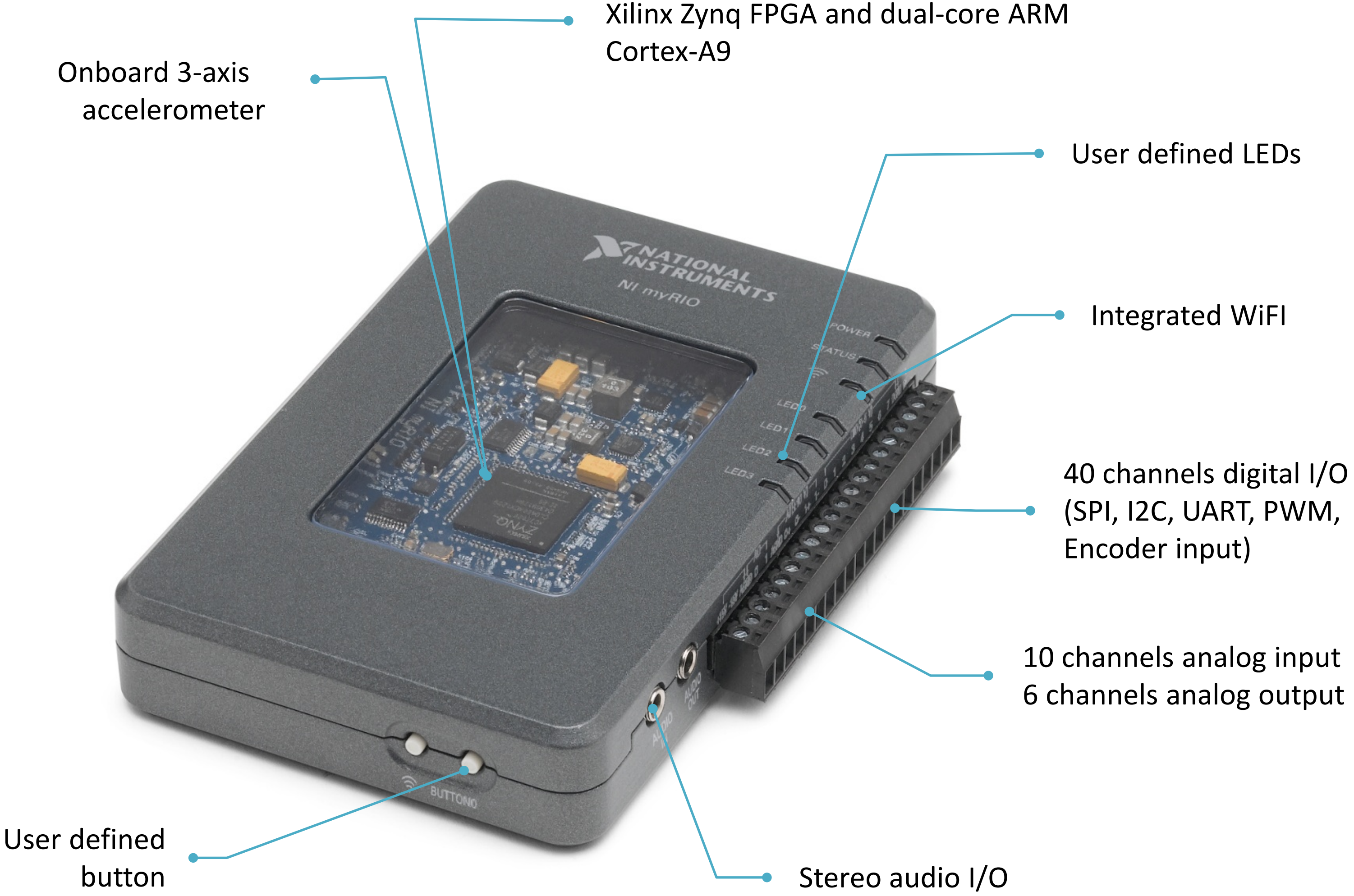
The Solution

- Students need a powerful hardware / software solution that allows them to get up to speed quickly

What is NI myRIO

- An embedded hardware / software device designed specifically to help students design real, complex engineering systems more quickly and affordably than ever before.

NI myRIO



Xilinx Zynq FPGA and dual-core ARM Cortex-A9

Onboard 3-axis accelerometer

User defined LEDs

Integrated WiFI

40 channels digital I/O (SPI, I2C, UART, PWM, Encoder input)

10 channels analog input
6 channels analog output

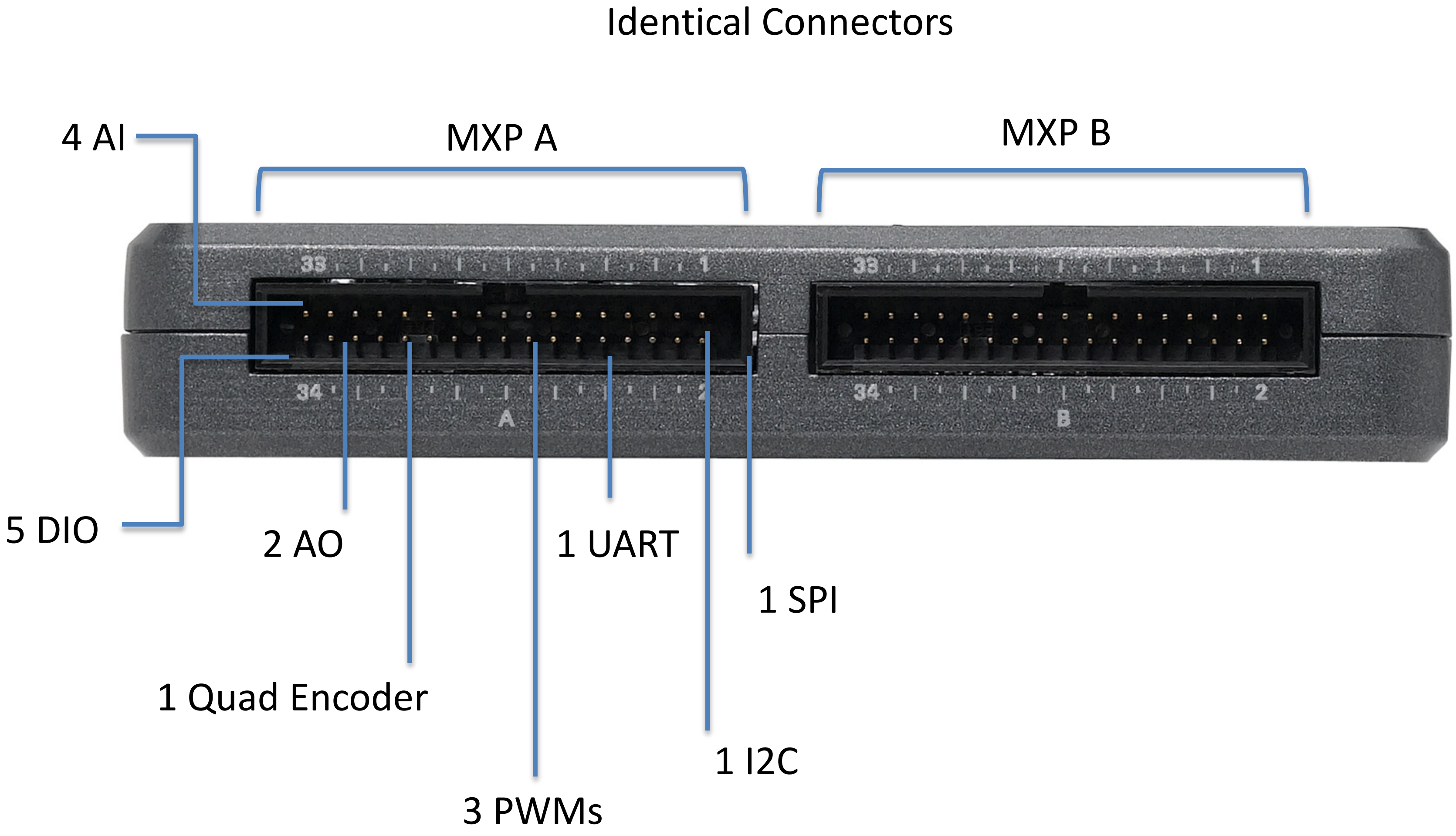
User defined button

Stereo audio I/O

Additional Features

- Fully programmable FPGA through LabVIEW FPGA
- Dual-Core ARM Cortex-A9 processor
- Expandable ecosystem of sensors and actuators
- Ready to use projects and courseware
- Deploy code to real-time processor and FPGA via USB or WiFi
- Minutes to first measurement
- Processor programmable in C/C++

NI myRIO Expansion Port (MXP)



Identical Connectors

MXP A

MXP B

4 AI

5 DIO

2 AO

1 Quad Encoder

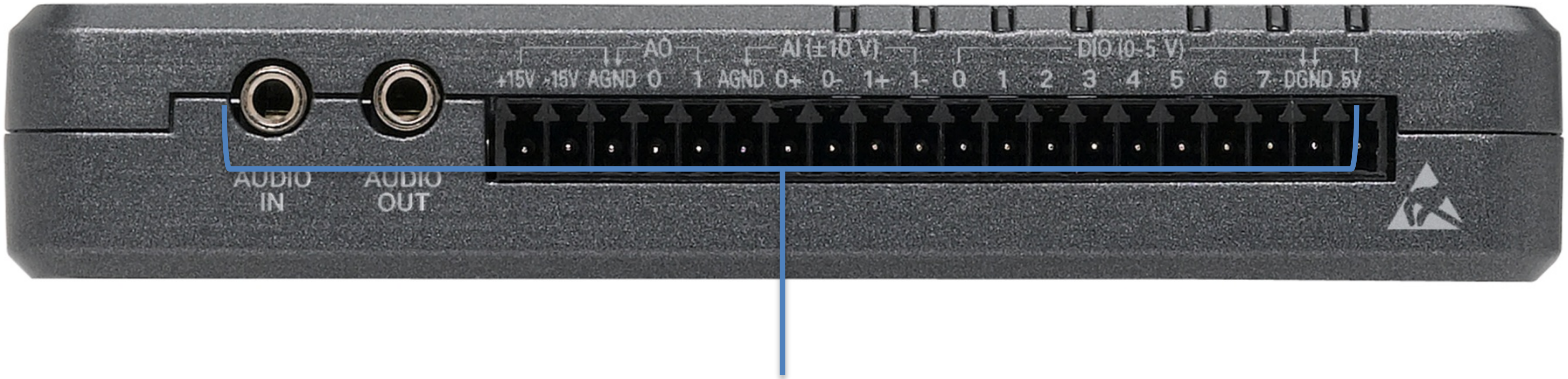
3 PWMs

1 UART

1 I2C

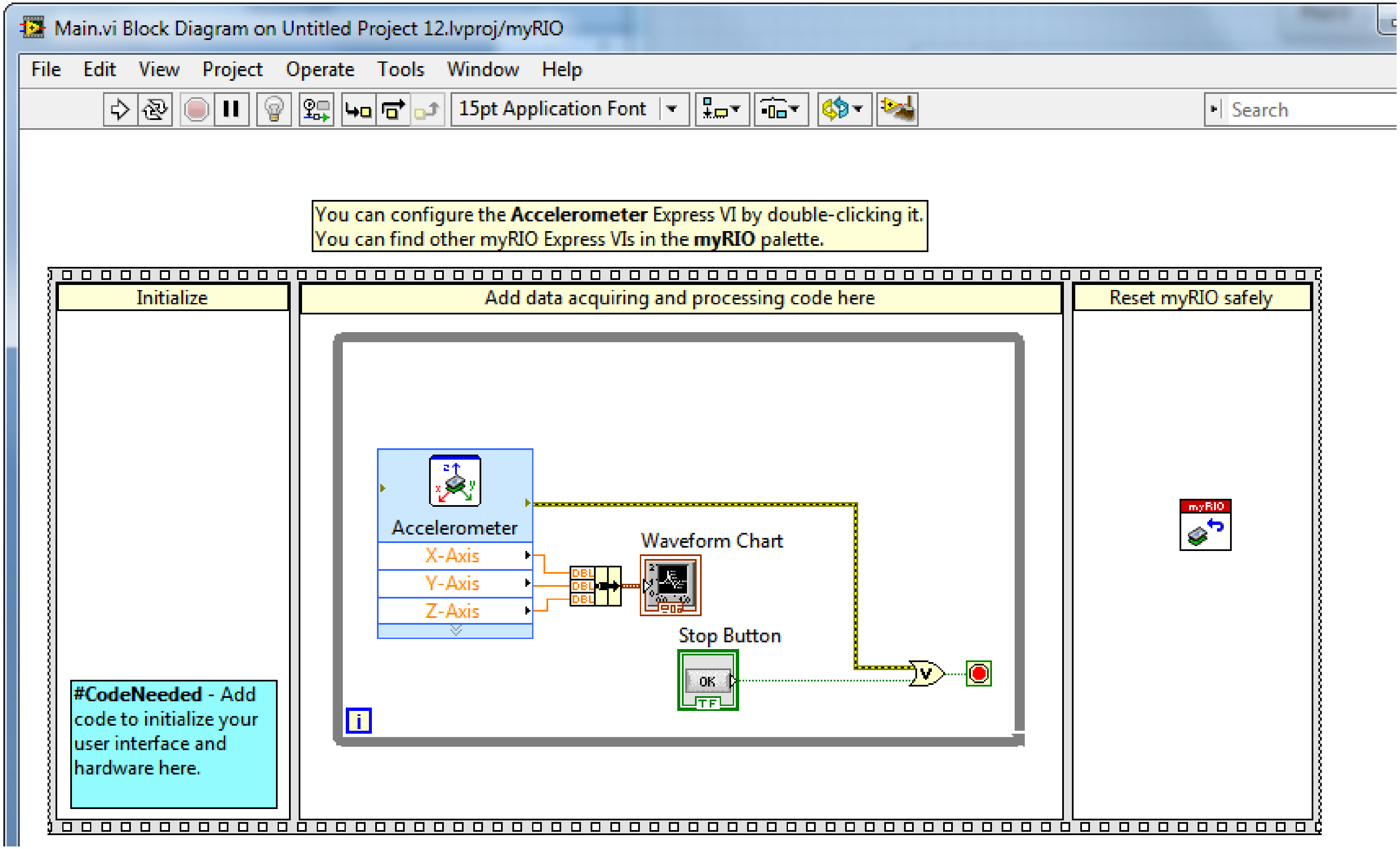
1 SPI

miniSystems Port (MSP)



Identical to NI myDAQ

RT Template VI



You can configure the **Accelerometer** Express VI by double-clicking it.
You can find other myRIO Express VIs in the **myRIO** palette.

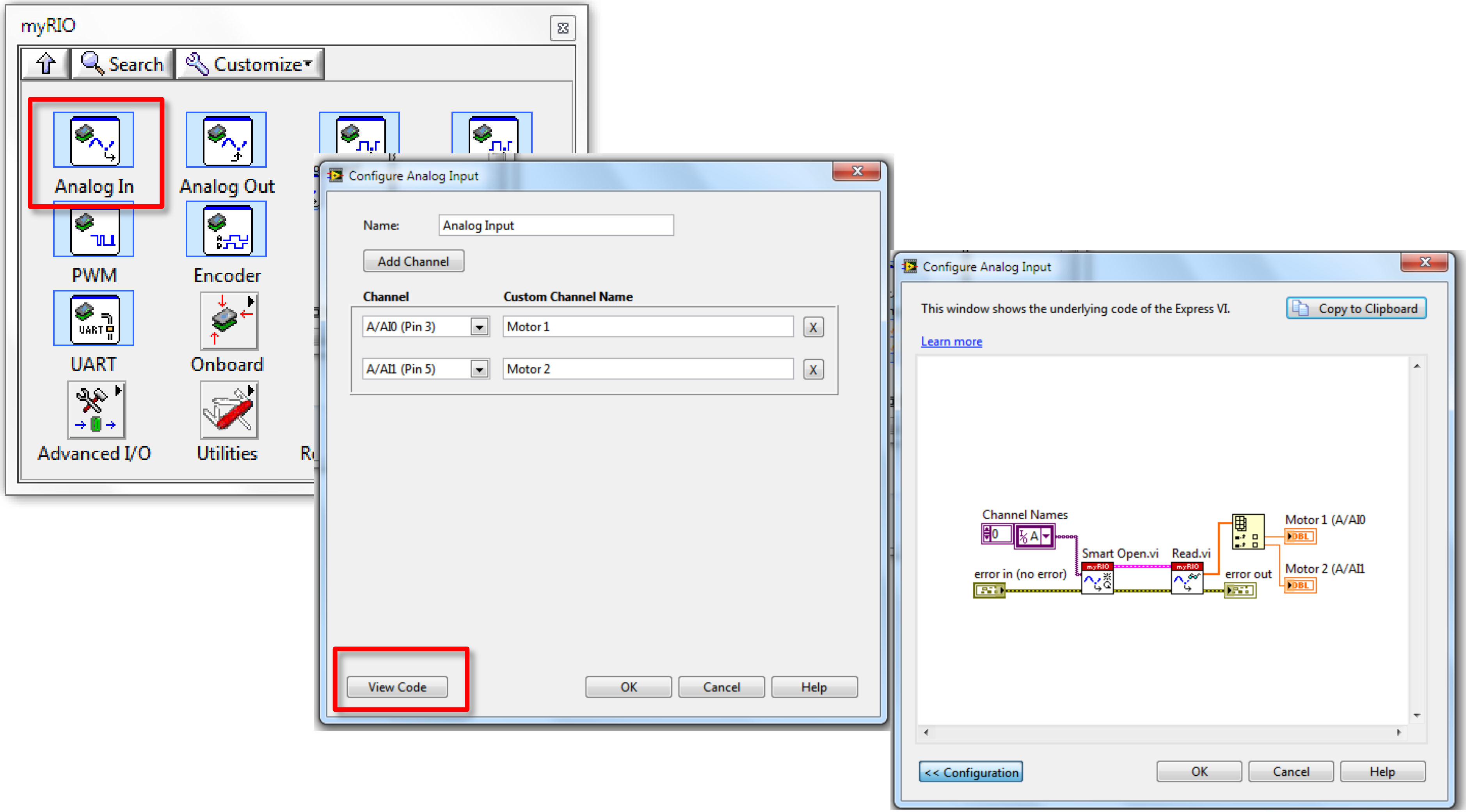
#CodeNeeded - Add code to initialize your user interface and hardware here.

NATIONAL INSTRUMENTS™

UAB THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

VRE LI
VEHICLE & ROBOTICS ENGINEERING LABORATORY

Express VIs to Advanced VIs Open Source

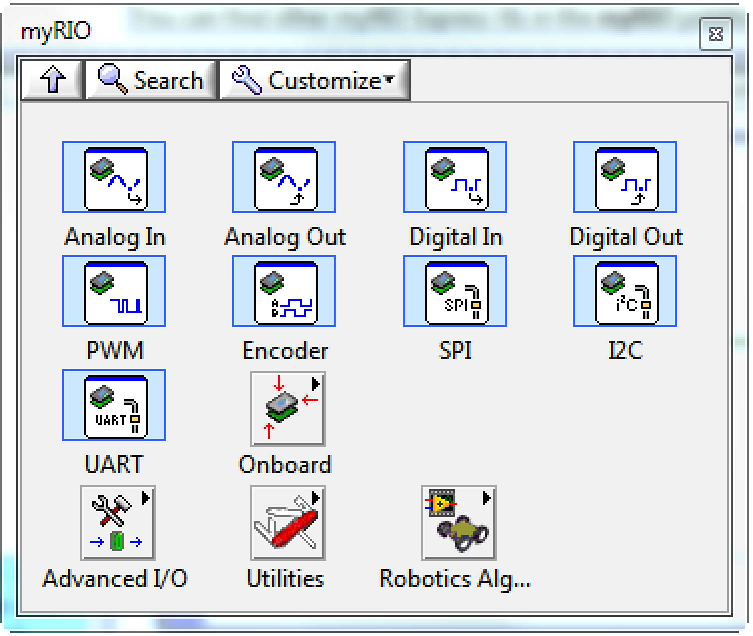


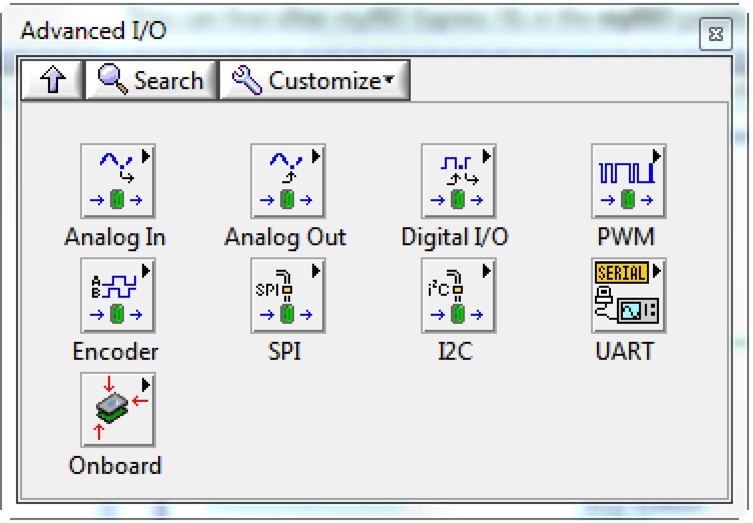
Name: Analog Input

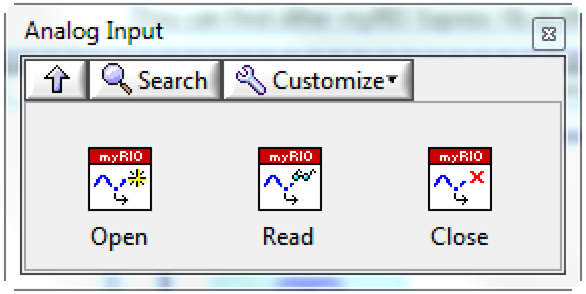
Channel	Custom Channel Name
A/AI0 (Pin 3)	Motor 1
A/AI1 (Pin 5)	Motor 2

This window shows the underlying code of the Express VI.

NI myRIO Palette





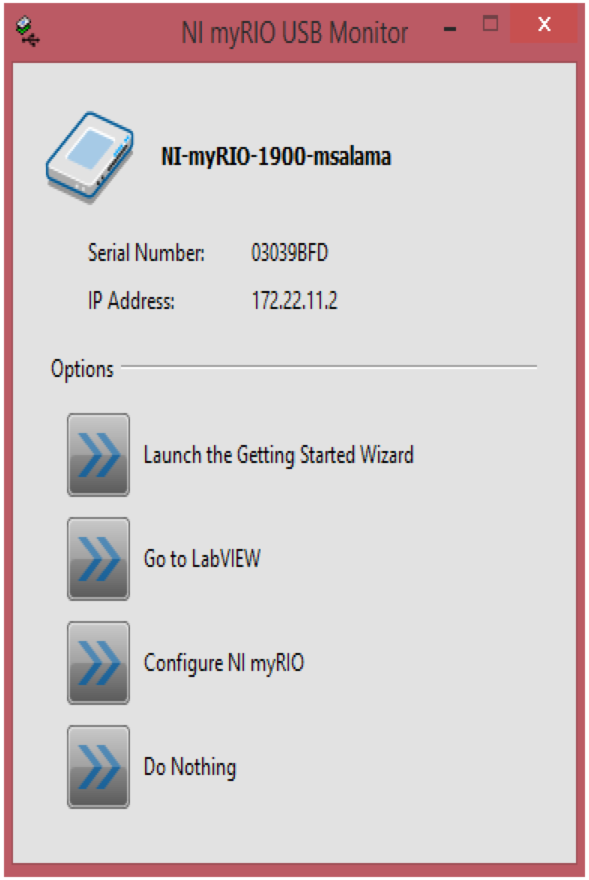


NATIONAL INSTRUMENTS

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

VEHICLE & ROBOTICS ENGINEERING LABORATORY

Connecting myRIO to your Computer



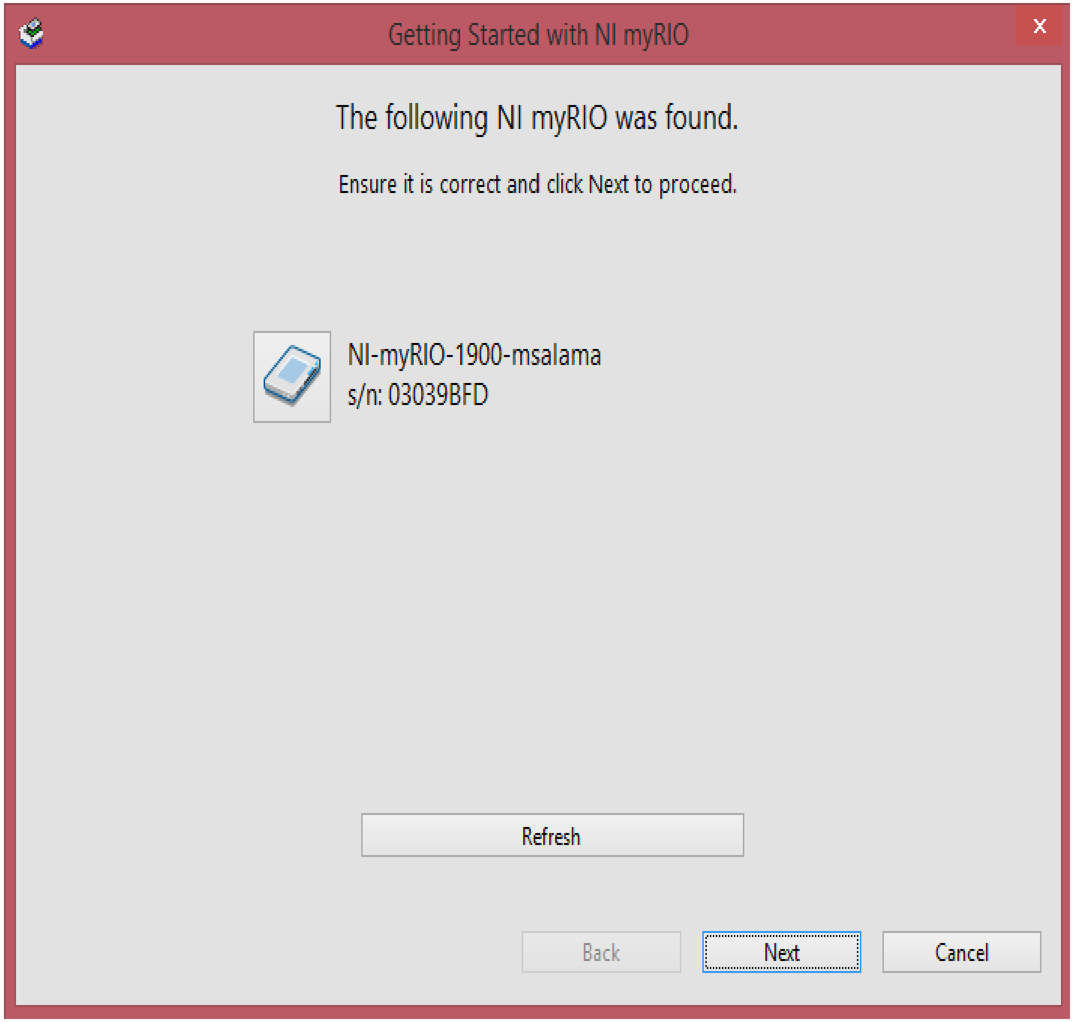
NI myRIO USB Monitor

NI-myRIO-1900-msalama

Serial Number: 03039BFD

IP Address: 172.22.11.2

Options

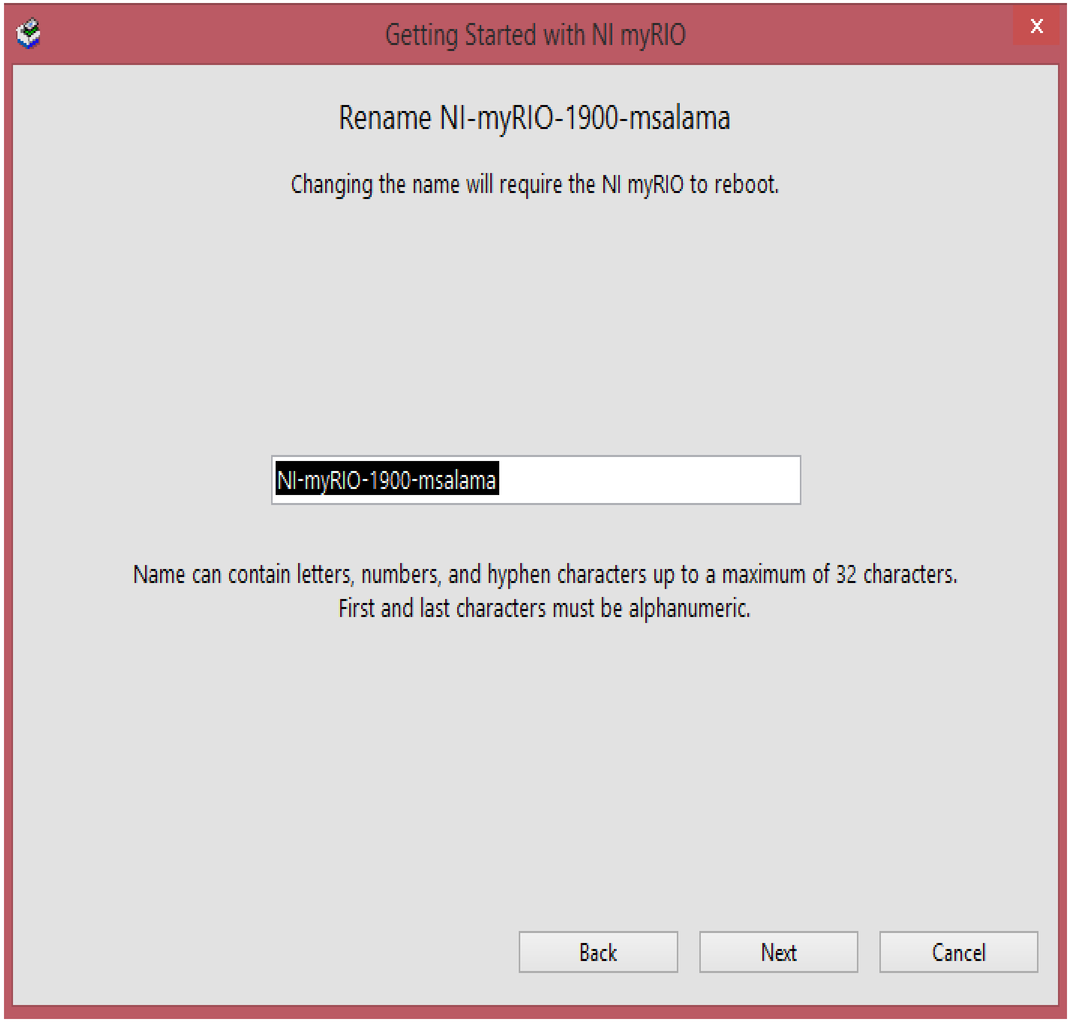


Getting Started with NI myRIO

The following NI myRIO was found.

Ensure it is correct and click Next to proceed.

NI-myRIO-1900-msalama
s/n: 03039BFD

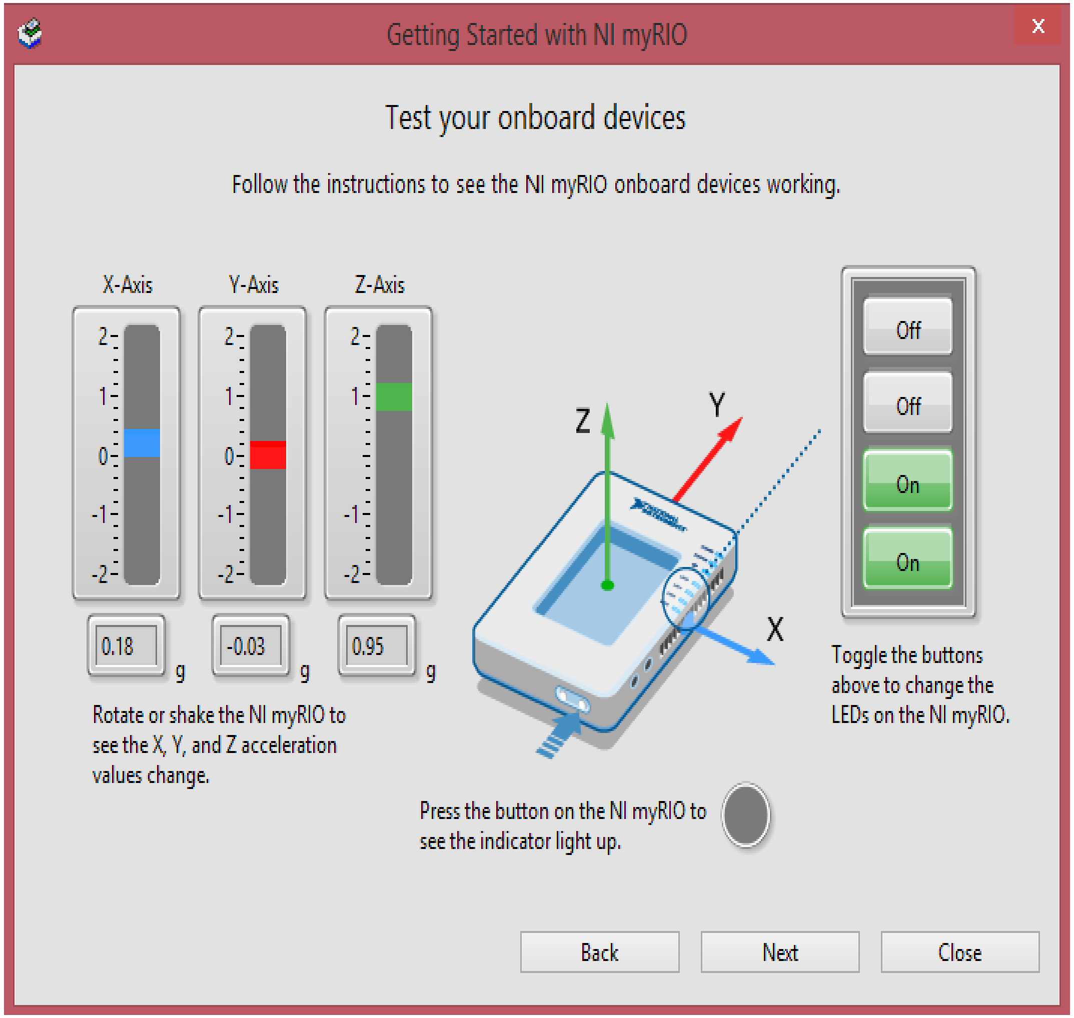


Rename NI-myRIO-1900-msalama

Changing the name will require the NI myRIO to reboot.

NI-myRIO-1900-msalama

Name can contain letters, numbers, and hyphen characters up to a maximum of 32 characters.
First and last characters must be alphanumeric.



Test your onboard devices

Follow the instructions to see the NI myRIO onboard devices working.

0.18 g

-0.03 g

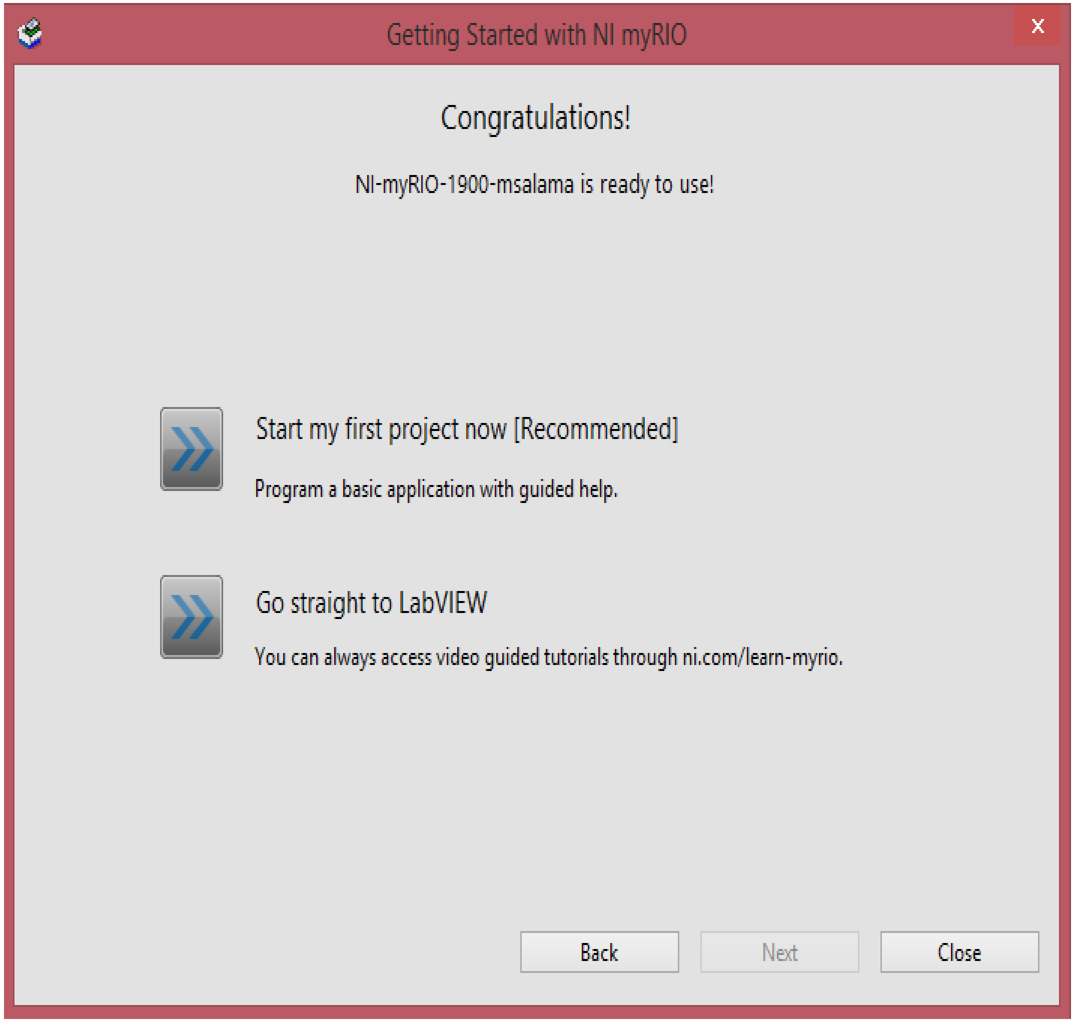
0.95 g

Rotate or shake the NI myRIO to see the X, Y, and Z acceleration values change.

Toggle the buttons above to change the LEDs on the NI myRIO.

Press the button on the NI myRIO to see the indicator light up.

Start my first project now [Recommended]



Getting Started with NI myRIO

Congratulations!

NI-myRIO-1900-msalama is ready to use!

Start my first project now [Recommended]

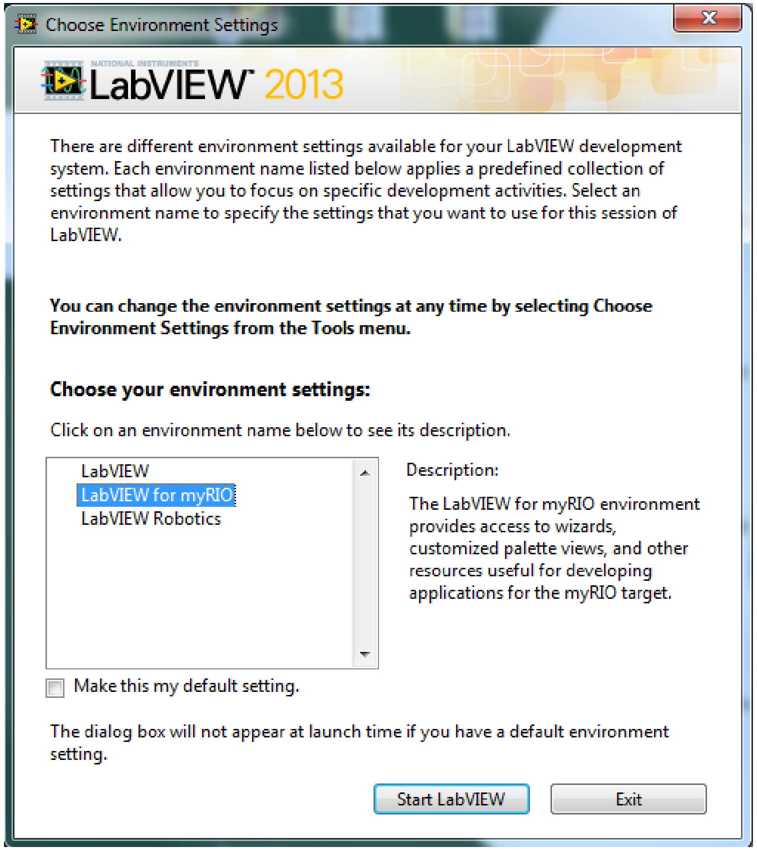
Program a basic application with guided help.

Go straight to LabVIEW

You can always access video guided tutorials through ni.com/learn-myrio.

Choose Environment Settings

- Choose LabVIEW for myRIO.
- Then click **Start LabVIEW**.



Choose Environment Settings
NATIONAL INSTRUMENTS LabVIEW 2013

There are different environment settings available for your LabVIEW development system. Each environment name listed below applies a predefined collection of settings that allow you to focus on specific development activities. Select an environment name to specify the settings that you want to use for this session of LabVIEW.

You can change the environment settings at any time by selecting Choose Environment Settings from the Tools menu.

Choose your environment settings:

Click on an environment name below to see its description.

- LabVIEW
- LabVIEW for myRIO
- LabVIEW Robotics

Description:

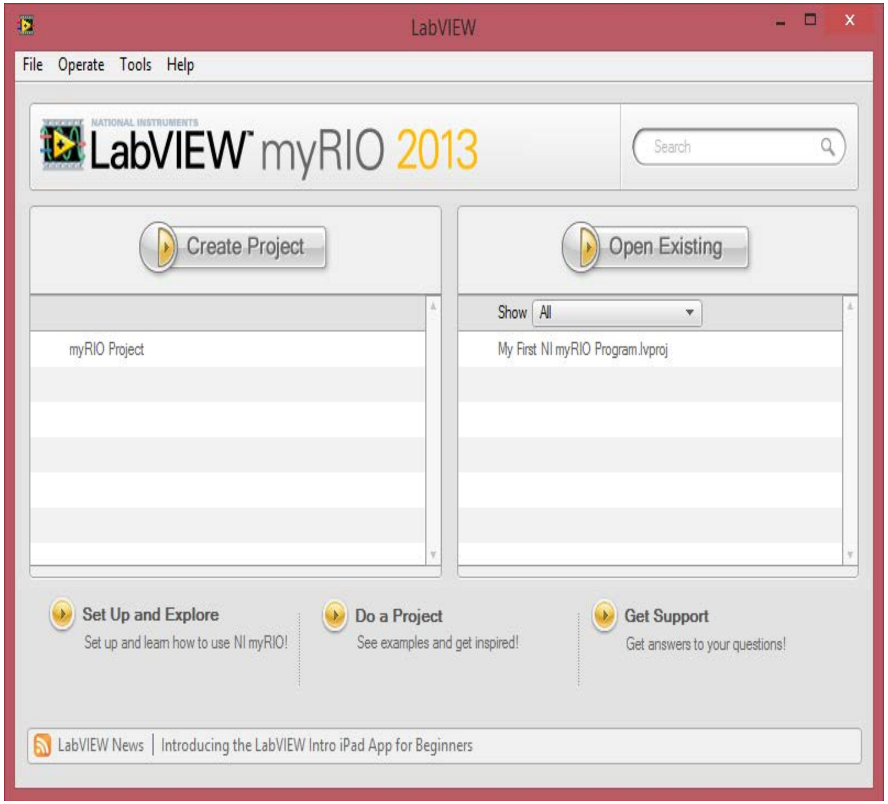
The LabVIEW for myRIO environment provides access to wizards, customized palette views, and other resources useful for developing applications for the myRIO target.

☐ Make this my default setting.

The dialog box will not appear at launch time if you have a default environment setting.

Part 1: Creating a myRIO Project

- Click **Create Project** from the **Getting Started** window of LabVIEW



Set up and learn how to use NI myRIO!

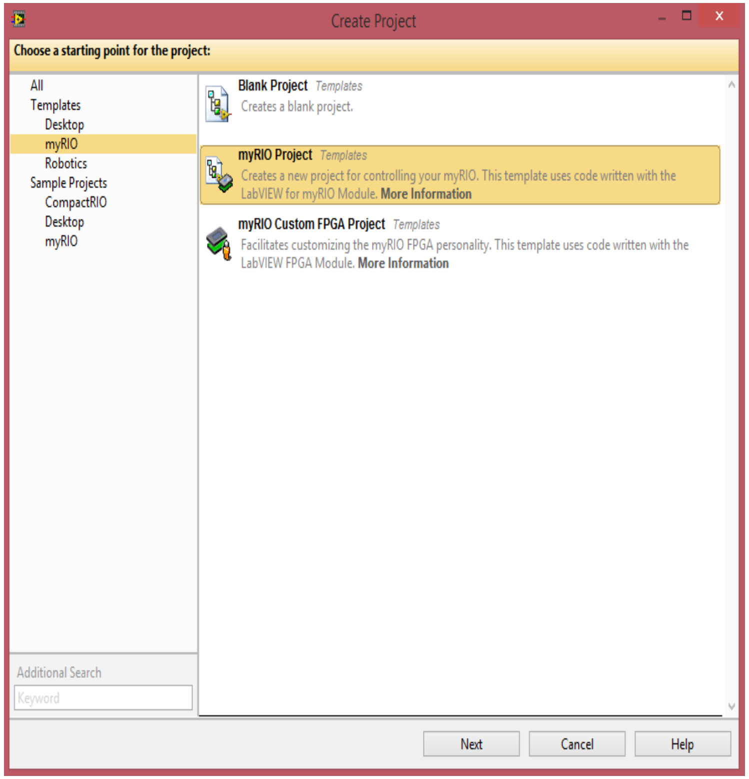
See examples and get inspired!

Get answers to your questions!

Introducing the LabVIEW Intro iPad App for Beginners

Part 1: Cont’d

- Select **Templates>>myRIO** and then select the **myRIO Project** template in the **Project List**.



Choose a starting point for the project:

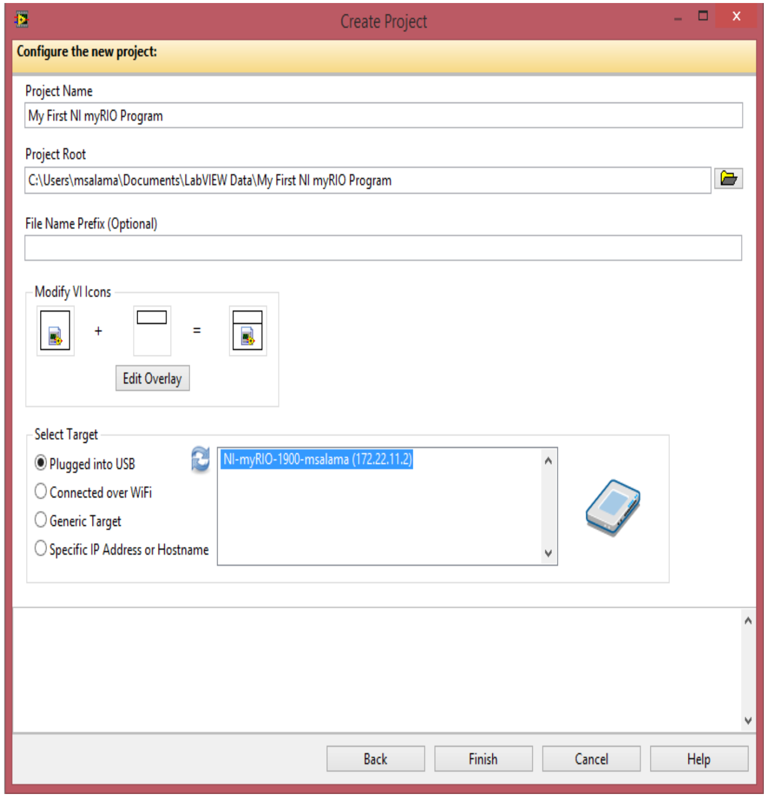
Blank Project *Templates*
Creates a blank project.

myRIO Project *Templates*
Creates a new project for controlling your myRIO. This template uses code written with the LabVIEW for myRIO Module. **More Information**

myRIO Custom FPGA Project *Templates*
Facilitates customizing the myRIO FPGA personality. This template uses code written with the LabVIEW FPGA Module. **More Information**

Part 1: Cont’d

- Enter My First NI myRIO Program under **Project Name**.
- Under **Project Root**, specify the directory into which you want to save the project. labVIEW will place all the project files and Vis in this directory.
- In **Select Target**, select the NI myRIO that you connected to your computer.
- Click **Finish**.



Create Project

Configure the new project:

Project Name: My First NI myRIO Program

Project Root: C:\Users\msalama\Documents\LabVIEW Data\My First NI myRIO Program

File Name Prefix (Optional):

Modify VI Icons

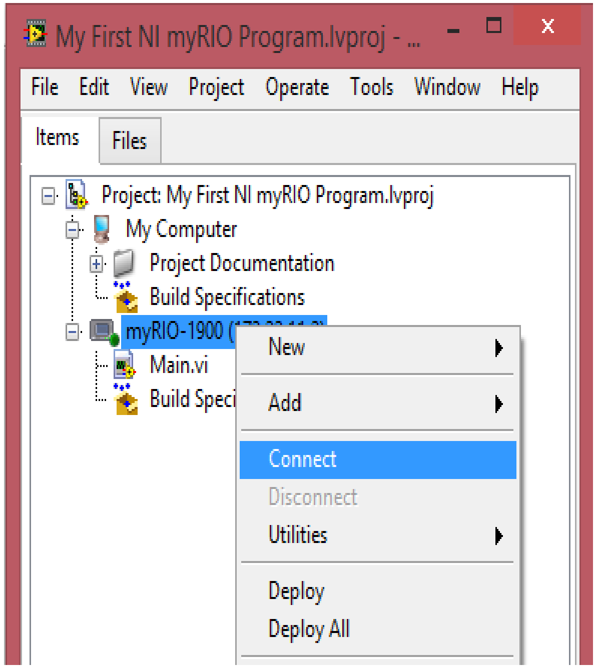
Select Target
- ● Plugged into USB
- ○ Connected over WiFi
- ○ Generic Target
- ○ Specific IP Address or Hostname

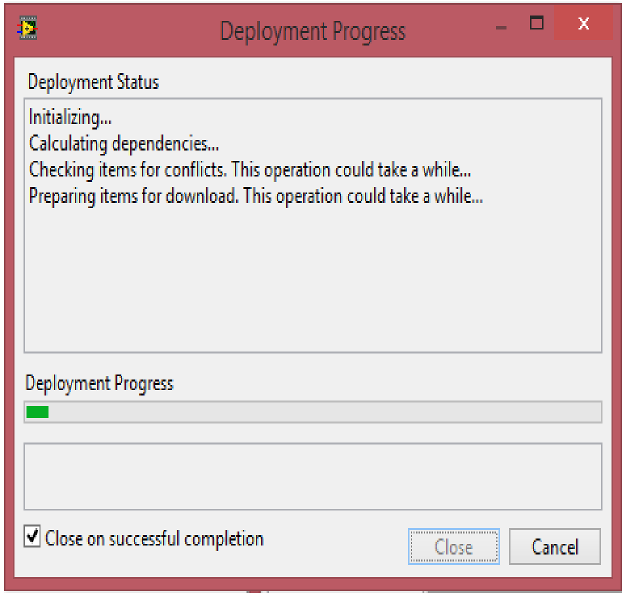
NI-myRIO-1900-msalama (172.22.11.2)

Part 1: Cont’d

- Explore the **Project Explorer** window. If you want to learn more details about this myRIO project, open myRIO Project Documentation.html under **Project Documentation**.

- You have successfully created a myRIO project. Proceed to the next tutorial to learn how you can create applications based on the myRIO project.




Deployment Progress

Deployment Status

Initializing...
Calculating dependencies...
Checking items for conflicts. This operation could take a while...
Preparing items for download. This operation could take a while...

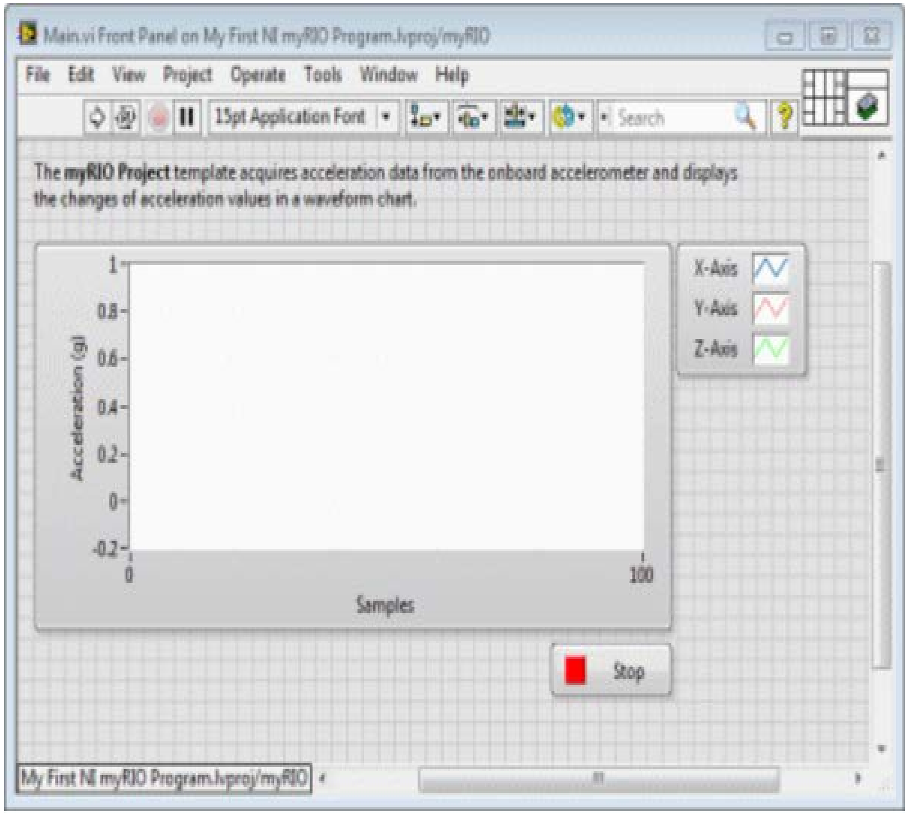
Deployment Progress

☑ Close on successful completion

Part 2: Testing the Accelerometer

- NI myRIO contains an onboard accelerometer that can be used for general orientation and acceleration measurements.

- Open `Main.vi` from the **Project Explorer** window of your myRIO project. By default, LabVIEW opens the front panel of `Main.vi`. The front panel is the user interface of a VI.



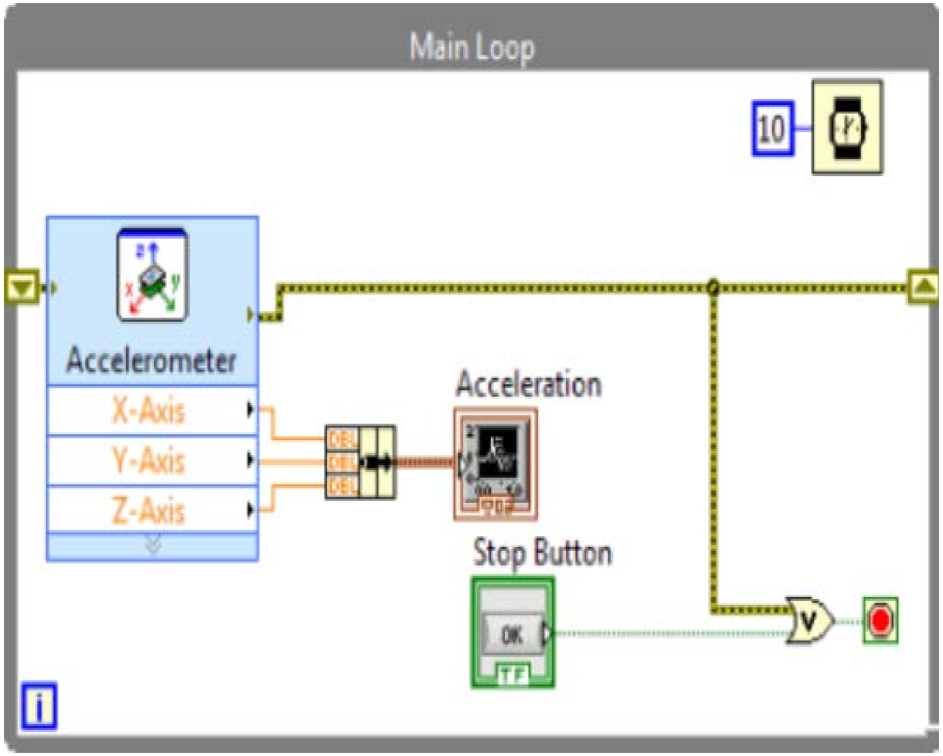
Main.vi Front Panel on My First NI myRIO Program.lvproj/myRIO
File Edit View Project Operate Tools Window Help
15pt Application Font
Search

The **myRIO Project** template acquires acceleration data from the onboard accelerometer and displays the changes of acceleration values in a waveform chart.

Acceleration (g)
1
0.8
0.6
0.4
0.2
0
-0.2
0
100
Samples
X-Axis
Y-Axis
Z-Axis
Stop
My First NI myRIO Program.lvproj/myRIO

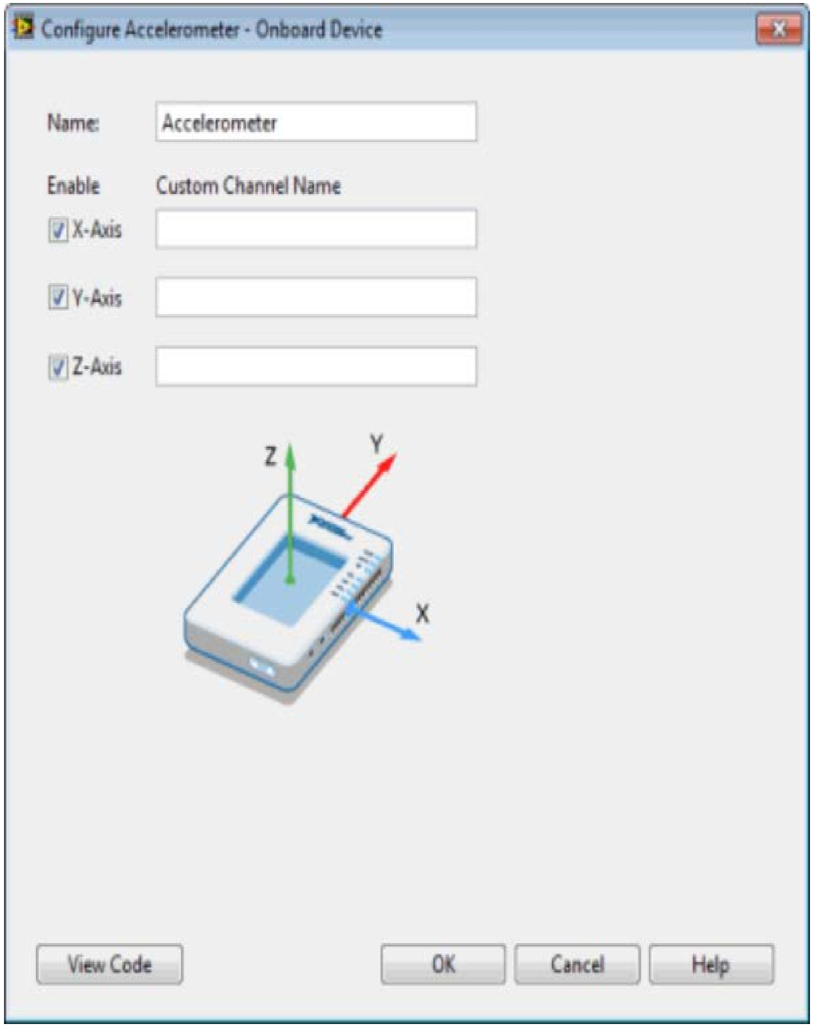
Part 2: Cont’d

- Press <Ctrl-E> to switch to the block diagram. The block diagram contains the graphical code of a VI. This VI uses the Accelerometer Express VI to read acceleration values from the onboard accelerometer and uses the waveform chart indicator to display the acceleration values



Part 2: Cont'd

- Double-click the Accelerometer Express VI to display the **Configure Accelerometer** dialog box.
- Press <Ctrl-H> to display the **Context Help** window. You can move the cursor over options in the dialog box and learn basic information about the options from the **Context Help** window. Most objects in LabVIEW display context help information.
- Click **OK** to apply your configuration.



Configure Accelerometer - Onboard Device

Name: Accelerometer

Enable | Custom Channel Name

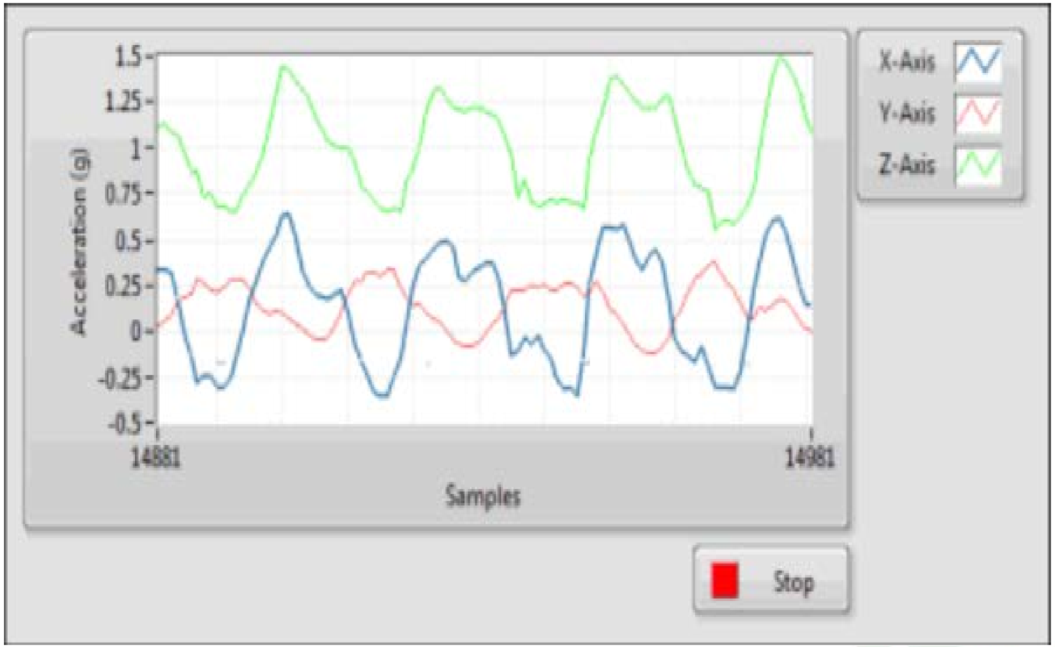
☑ X-Axis

☑ Y-Axis

☑ Z-Axis

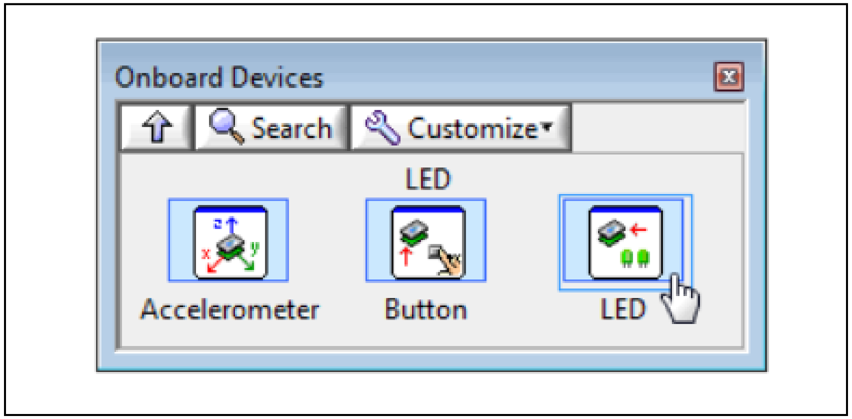
Part 2: Cont’d

- Press <Ctrl-E> to switch to the front panel and click **Run**. Rorate or shake your NI myRIO to see the changes of the X, Y, and Z acceleration values on the waveform chart.
- Click **Stop**.
- Click **File>>Save**.



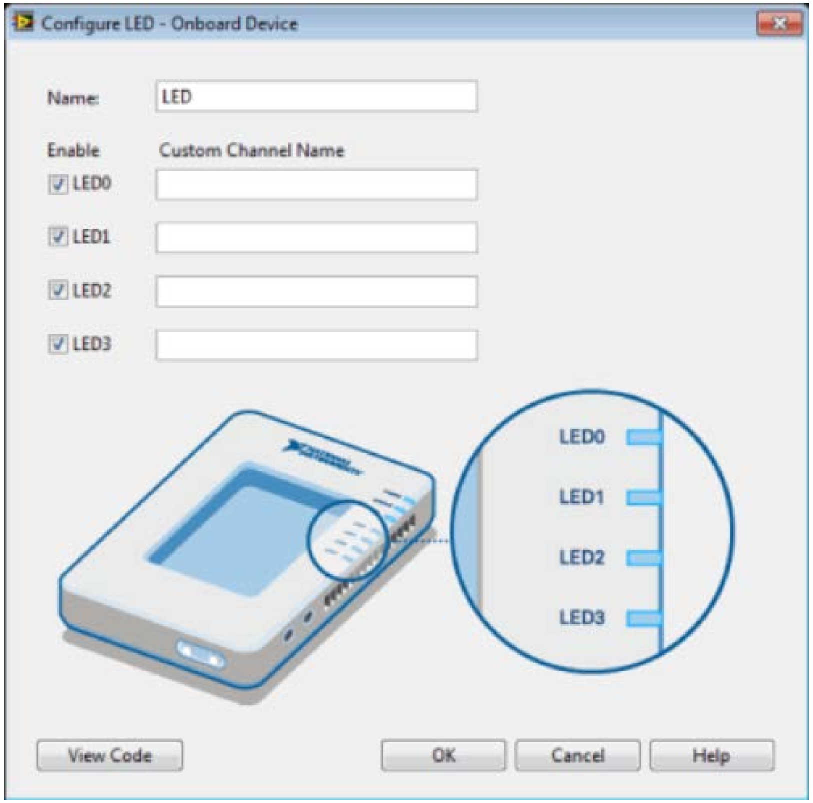
Part 3: Testing the LEDs

- In the block diagram window of Main.vi, click **View>> Functions Palette**. You can find all the myRIO VIs and LabVIEW functionality on the Functions palette.
- Select **Functions>>myRIO>>Onboard Devices** to locate the LED Express VI.
- Click the LED Express VI and add it to the While Loop in Main.vi.



Part 3: Cont’d

- In the **Configure LED** dialog box, select the LEDs you want to test and click **OK**. in this tutorial, we will test all four onboard LEDs.



Configure LED - Onboard Device

Name: LED

Enable Custom Channel Name

☑ LED0

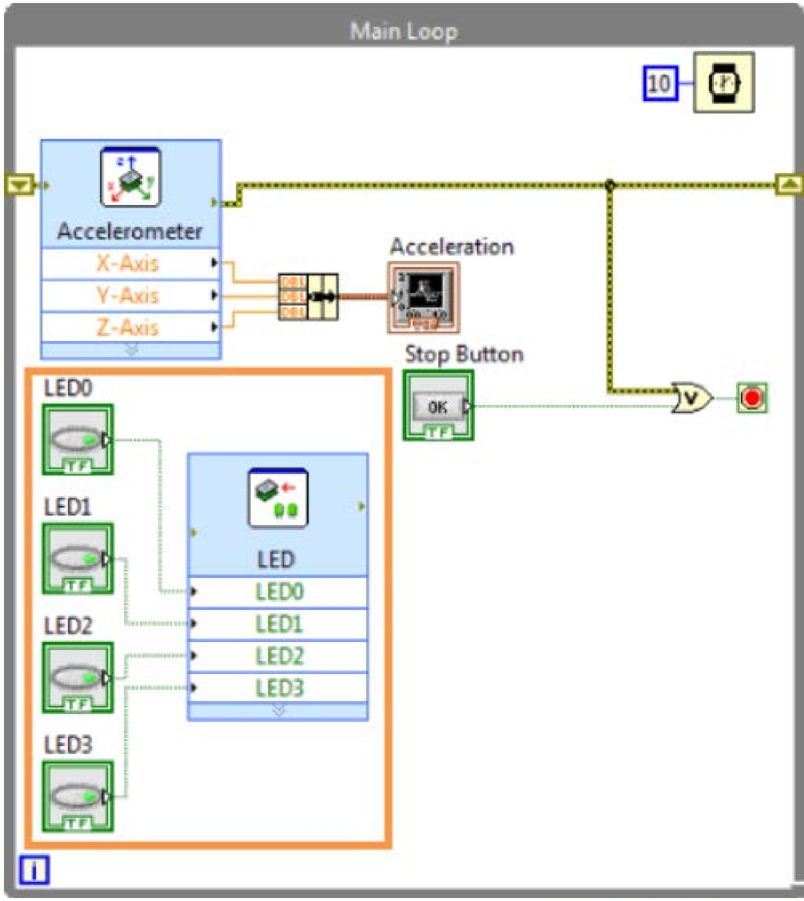
☑ LED1

☑ LED2

☑ LED3

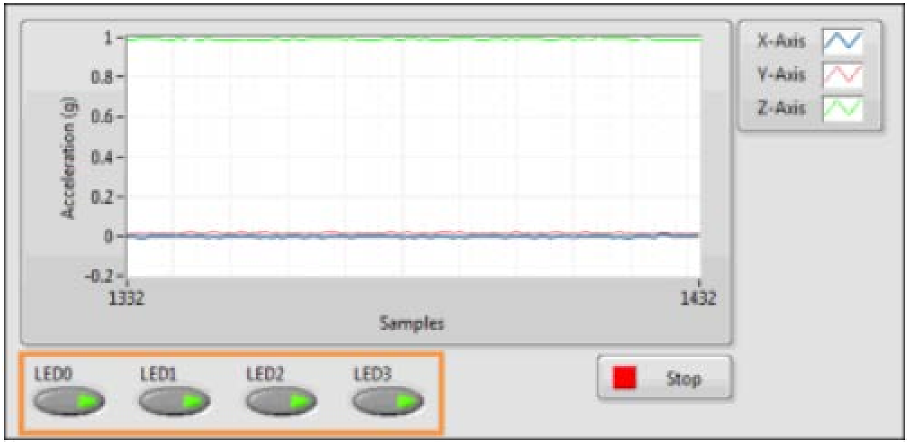
Part 3: Cont'd

- Right-click each block diagram input of the LED Express VI and select **Create>>Control** to create Boolean controls for the four LEDs. You can click a Boolean control to toggle between the TRUE and FALSE states, which determined the ON and OFF states of an onboard LED.



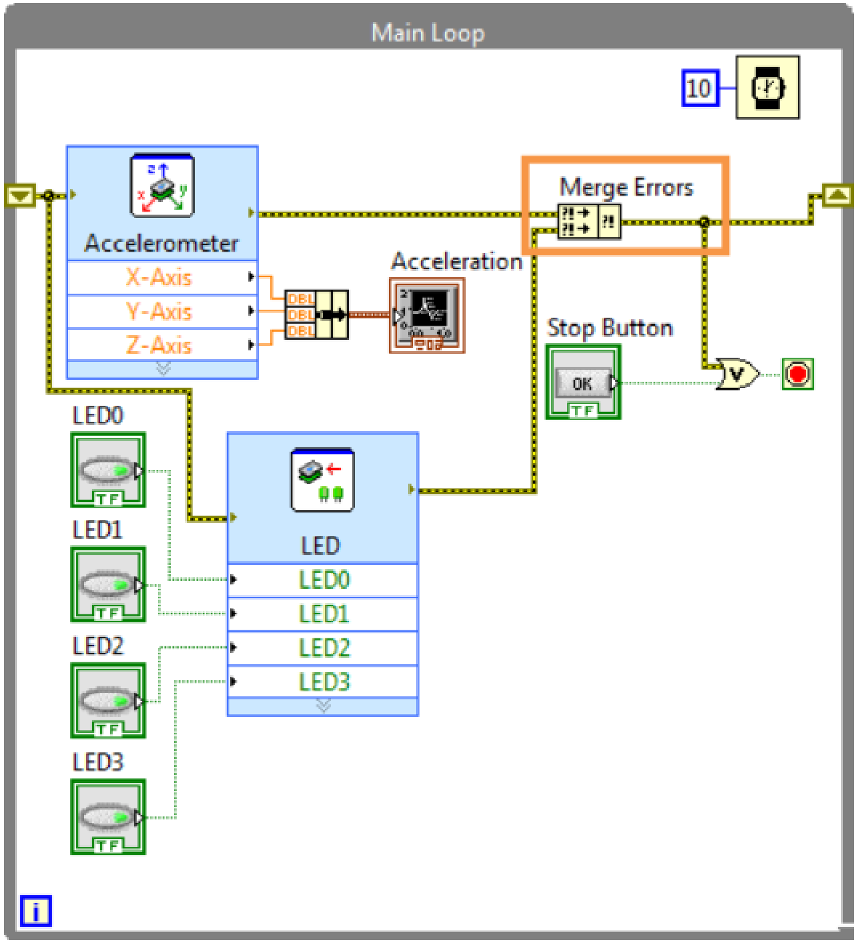
Part 3: Cont’d

- Press <Ctrl-E> to switch to the front panel. You can arrange the four Boolean controls in a layout that you like.
- Click **Run**.
- Click the four Boolean controls and see the status changes of the onboard LEDs.
- Click **Stop**.
- Click **File>>Save**.



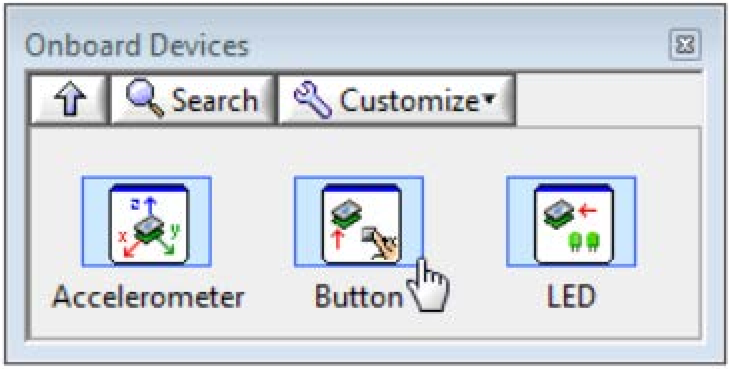
Error checking in LabVIEW

- Right now Main.vi stops if the Accelerometer Express VI returns an error. You need to add code to handle errors that the LED Express VI might return. You can use the Merge Errors function, available by selecting **Functions>>Programming>>Dialog & User Interface>> Merge Errors**, to merge errors from the Accelerometer and LED Express VIs.
- Error checking is important because it can tell you why and where errors occur.



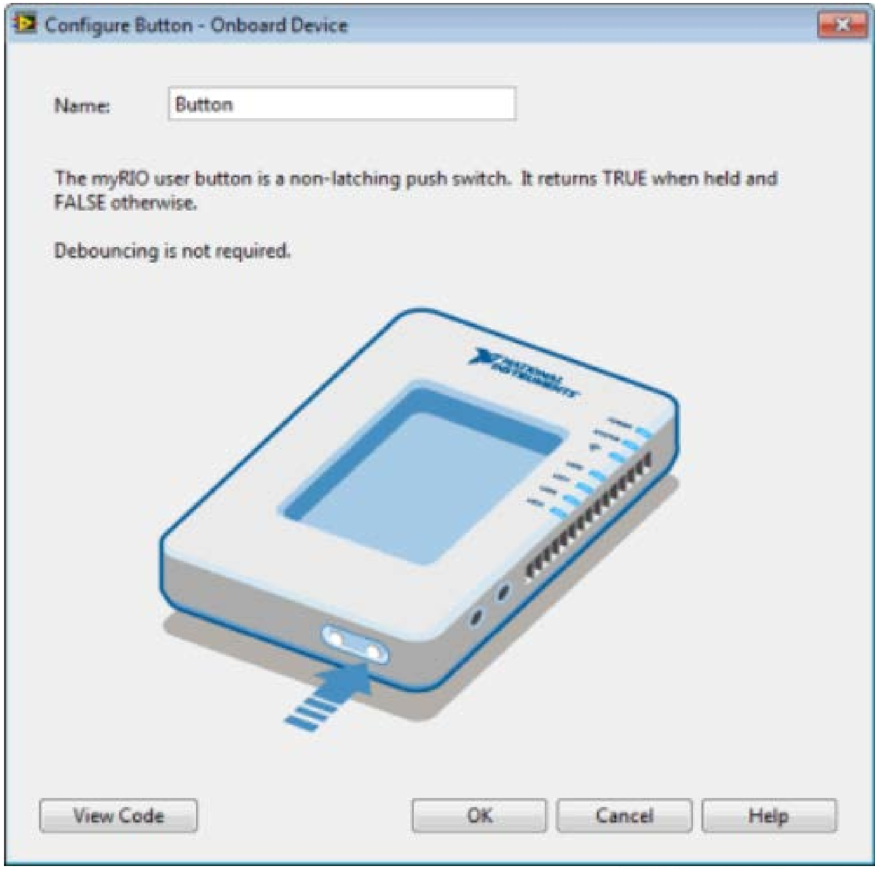
Part 4: Testing the Button

- In the block diagram window of Main.vi, click **View>>Functions Palette and select Functions>>myRIO>>Onboard Devices** to locate the button Express VI.
- Click the button Express VI and add it to the Whil Loop in Main.vi.



Part 4: Cont’d

- In the **Configure Button** dialog box, click **OK** to enable testing the user button.



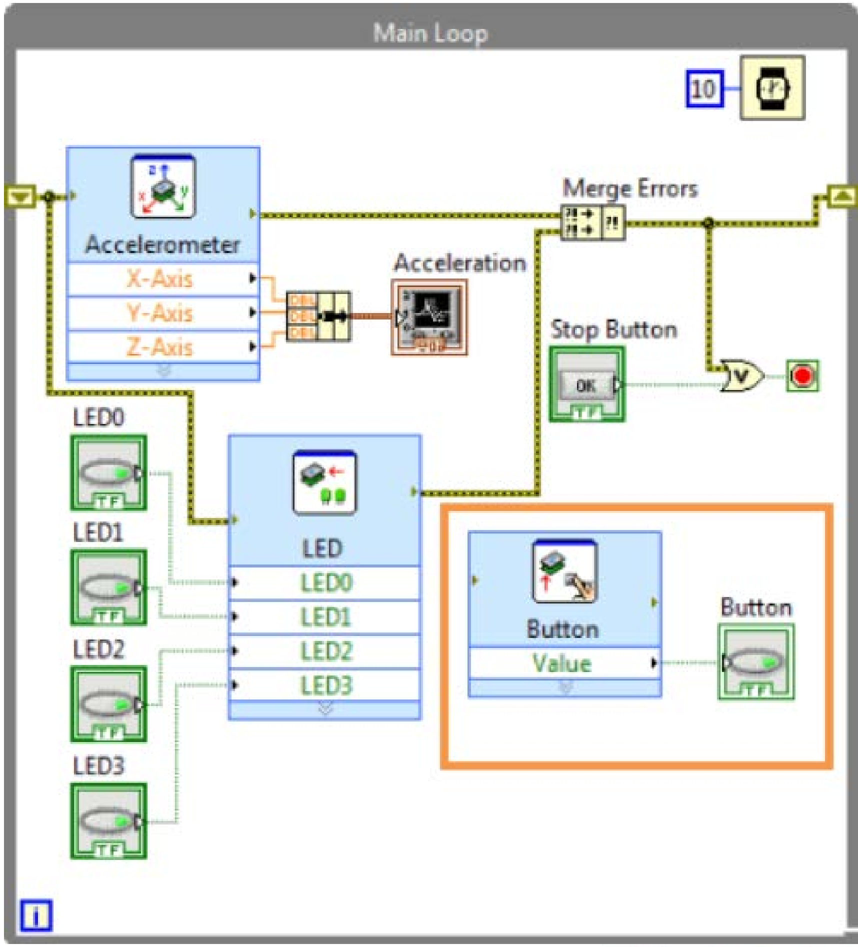
Name: Button

The myRIO user button is a non-latching push switch. It returns TRUE when held and FALSE otherwise.

Debouncing is not required.

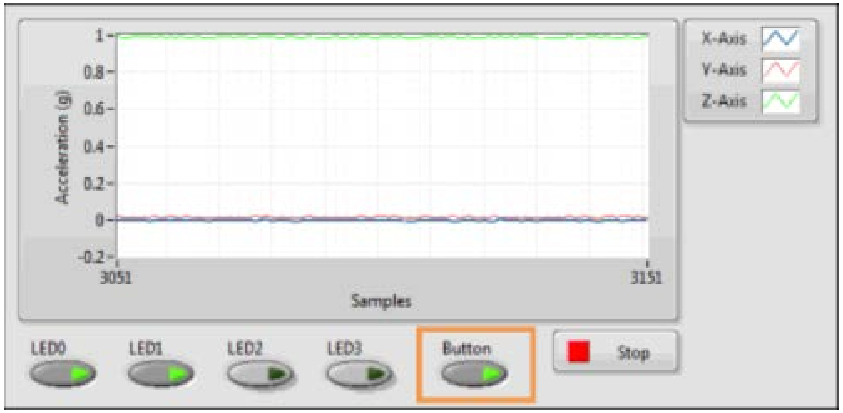
Part 4: Cont'd

- Right-click the **Value** output of the Button Express VI and select **Create>>Indicator** to create a Boolean indicator for the user button. The TRUE and FALSE states of the Boolean indicator represent the ON and OFF states of the user button.
- Rename the indicator as **Button**.
- (Optional) Add error checking for the Button Express VI.



Part 4: Cont'd

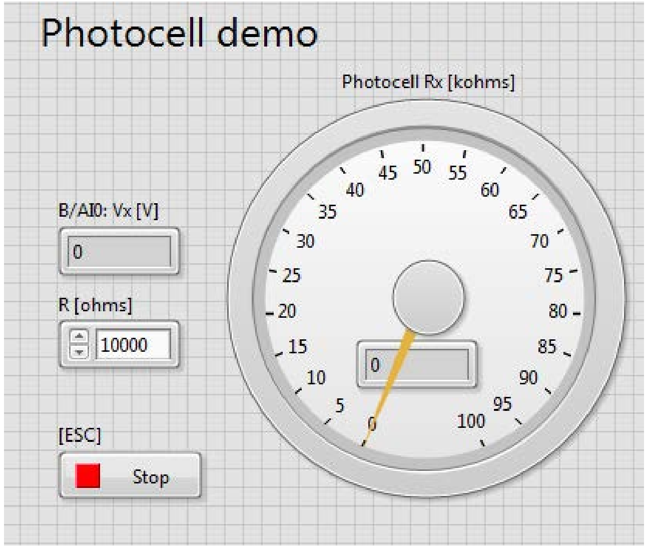
- Press <Ctrl-E> to switch to the front panel and click **Run**.
- Press the user button on the NI myRIO and observe the state change of the **Button** indicator.
- Click **Stop**.
- Click **File>>Save**.

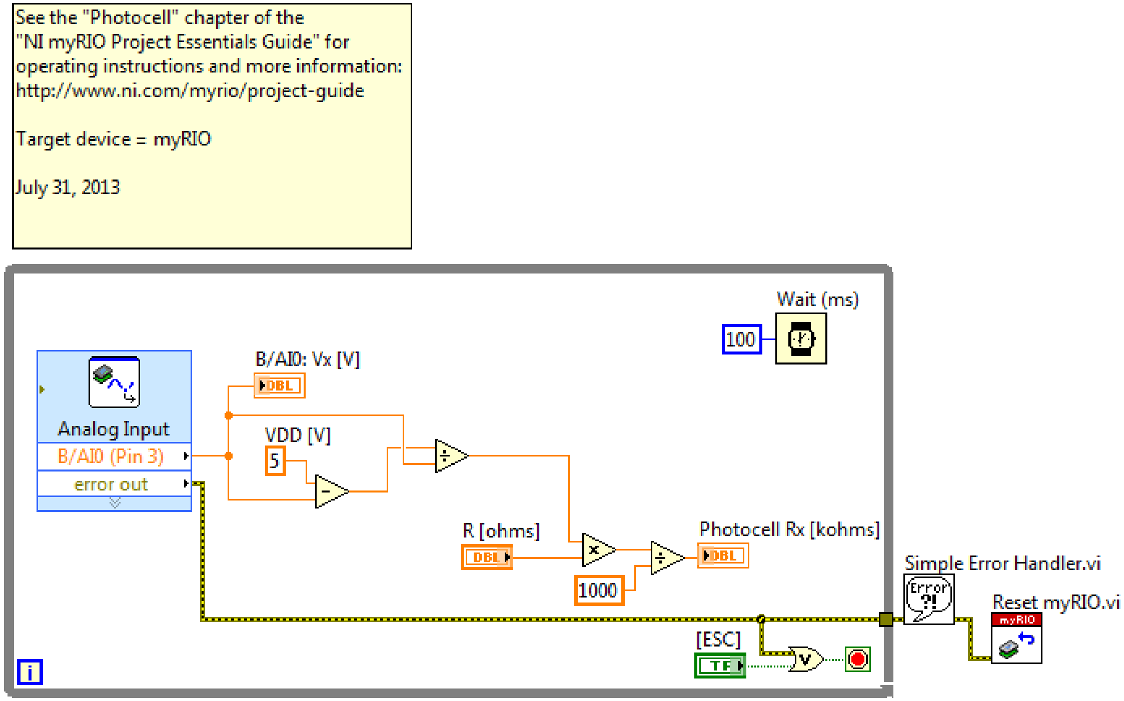


Good job! You have successfully created a myRIO project and tested all the NI myRIO onboard devices. You're ready to get started creating applications of your own!

- Helpful resources
  - **myRIO Module Help**
    - You can access this help file by selecting **Help»LabVIEW Help** from LabVIEW.
  - **Templates and Sample Projects**
    - You can customize templates and sample projects according to the needs of your application. In LabVIEW, select **File»Create Project** to display the **Create Project** dialog box. Look for the templates and sample projects under the **myRIO** categories.

Tutorial: Photo Cell Demo





See the "Photocell" chapter of the "NI myRIO Project Essentials Guide" for operating instructions and more information:
http://www.ni.com/myrio/project-guide

Target device = myRIO

July 31, 2013

THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

Part 3

NI myRIO Examples using Control Design and Simulation Toolkit

NATIONAL INSTRUMENTS™

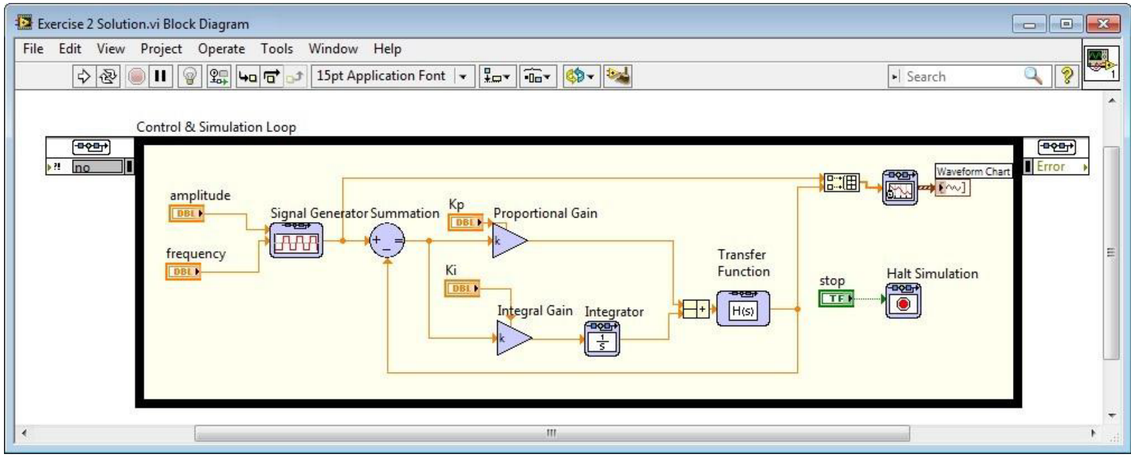
VEHICLE & ROBOTICS ENGINEERING LABORATORY

Introduction

- Control design is a process that involves developing mathematical models that describe a physical system, analyzing the models to learn about their dynamics characteristics, and creating a controller to achieve certain dynamic characteristics.
- Control systems are traditionally modeled as block diagrams.

Introduction

- CD&SIM contains a control and simulation loop, which is essentially a while loop with additional functionality for controls running in the background, including various ODE solvers.
  - Once a controller has been analysed in simulation, the simulated plant can be replaced with actual I/O pins.



THE UNIVERSITY OF ALABAMA AT BIRMINGHAM
Knowledge that will change your world

Example 1: RC Circuit Modeling and Simulation

NI myRIO Examples using Control Design and Simulation Toolkit

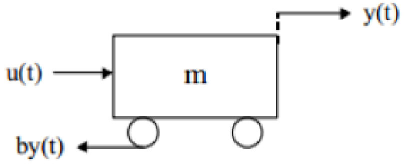
NATIONAL INSTRUMENTS

VEHICLE & ROBOTICS ENGINEERING LABORATORY

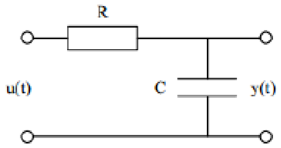
Develop a simulated control system that executes on the development machine

- Create a simple PI controller to control a voltage across the capacitor.
- We will model the RC circuit using a simple 1st order transfer function.
- The input is u(t) and the output is y(t).
- RC circuit time constant = 2.5.

Example: 1st Order Mechanical system



Example: 1st Order Electrical system

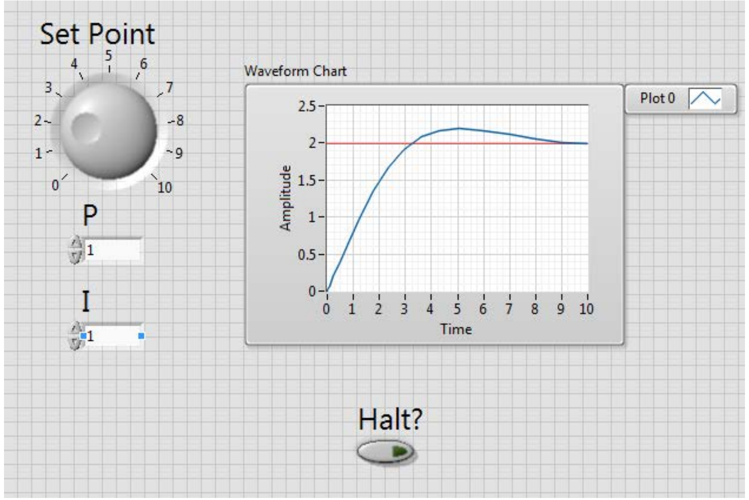


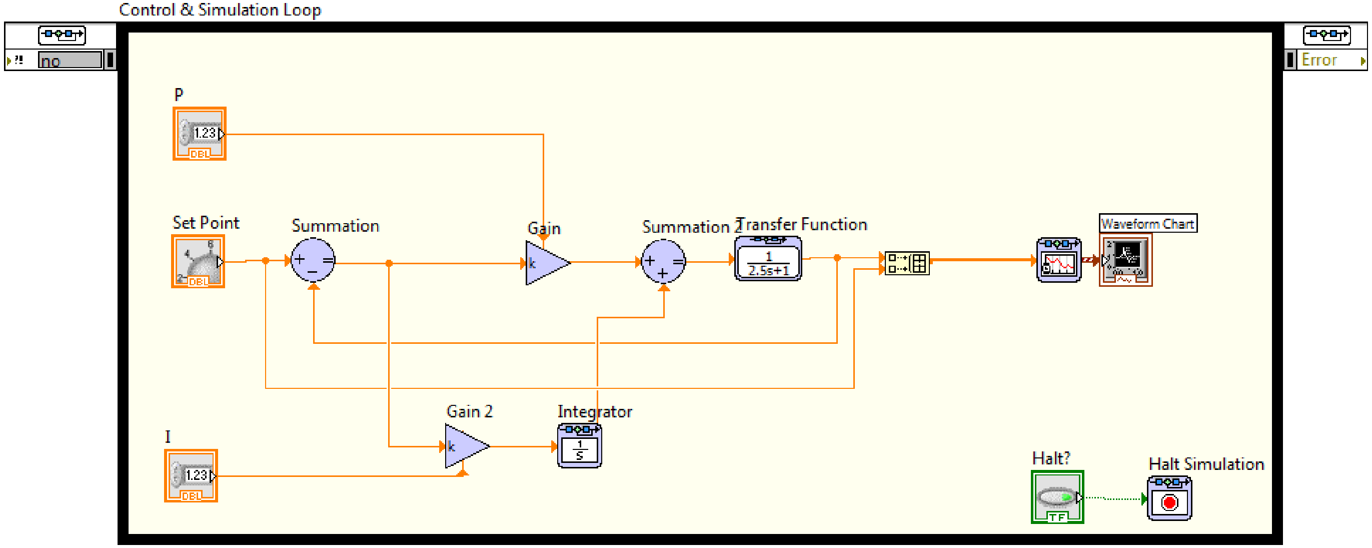
$$H_C(s) = \frac{V_C(s)}{V_{in}(s)} = \frac{1}{1 + RCs}$$

Steps

1. To save time we have provided a starting point for the application. Select **File» Open Project** and navigate to L\ME360\Section2C\Lecture\LabVIEW Lecture\Lecture 3\RC circuit\RC circuit.vi.
2. This will execute the VI on the development machine rather than the myRIO.
3. To execute on the myRIO, select File>> RC circuit.lvproj when the myRIO is connected to the computer machine using USB cable.

Front panel and Block diagram



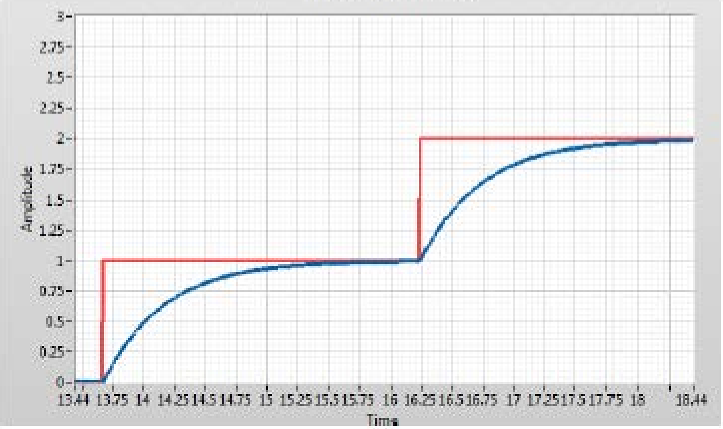


Steps

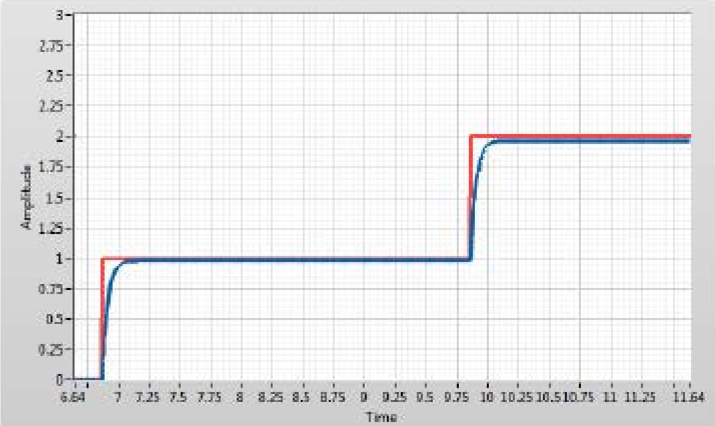
- The application is now complete. **Save** the application, switch to the front panel, and click **Run**. Remember that this application is now running on the host processor.
- While the application is executing, try interacting with the set point control and experimenting with the control gain values.

Results

Using the modest default gains
(P=5, I=2)



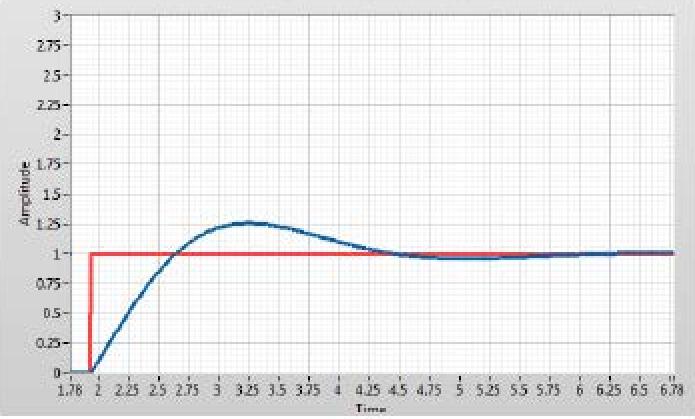
Increasing P causes a faster response, but can cause instabilities and oscillations
(P=50, I=2)



Results

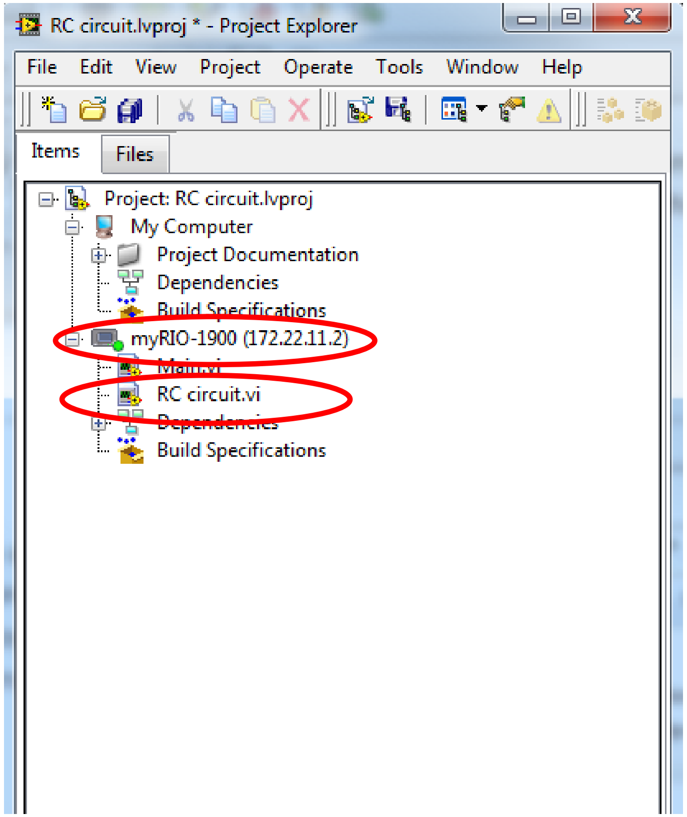
Increasing I decreases steady state error, but can increase overshoot and settling time

(P=4, I=10)



Develop a simulated control system that executes on the myRIO processor

- Select **File» Open Project** and navigate to L\ME360\Section2C\Lecture\LabVIEW Lecture\Lecture 3\RC circuit\RC circuit.lvproj.
- Right click on myRIO to connect.
- Double click on RC circuit.vi to deploy the code on myRIO processor.



Example 2: DC Motor Simulation Control using PID

NI myRIO Examples using Control Design and Simulation Toolkit

PID Control

- Proportional Integral Derivative (PID) is one of the most commonly used control algorithms due its ease of use and minimal required knowledge of the system or plant to be controlled.
- National Instruments provides ready-to-run, advanced (PID) control algorithms with the NI LabVIEW PID Control Toolkit. Combined with the LabVIEW Control Design & Simulation Module, the LabVIEW PID Control Toolkit can help you simulate and tune your PID controllers without implementing them in real-world systems, thus avoiding possible problems such as instability during application development.

Objective

- In this tutorial, learn how to use the LabVIEW PID Control Toolkit with the LabVIEW Control Design & Simulation Module and design the PID gains for the position controller of a DC motor in a Real-Time system.

DC Motor Modeling

- In this tutorial, we will design the velocity controller for a DC motor. For the sake of simplicity consider a basic transfer function for a DC motor where effects such as friction and disturbances are being considered:

$$\frac{\Phi(s)}{V(s)} = \frac{\frac{K}{JL}}{s^2 + \left(\frac{JR + BL}{JL}\right)s + \frac{BR + K^2}{JL}}$$

DC Motor Modeling

- Where
- $\Phi(s)$ is the angular velocity (rad/sec)
- $V(s)$ is applied voltage (V)
- J is the rotor inertia (9.64E-6)
- R is the rotor resistance (3.3 Ώ)
- K is the torque constant (0.028 N-m\A)
- L is the Inductance (4.64E-3 H)
- B is the Friction Torque Constant (1.8E-6 N-m-s)

DC Motor Modeling

- If you replace the numeric values, you get the following transfer function:

$$G(s) = \frac{3.5276\ 10^6}{s^2 + 1591.46s + 109711}$$

- Your goal is to implement a PID algorithm that is going to run on a Real-Time controller with a loop rate of 1000 Hz (0.001 second period).

Step 1

- Start by opening the LabVIEW Development Environment and navigating to the Block Diagram. On the Functions Palette, select *Control Design & Simulation->Simulation->Control & Simulation Loop* then click and drag to size and create a **Control & Simulation Loop.**

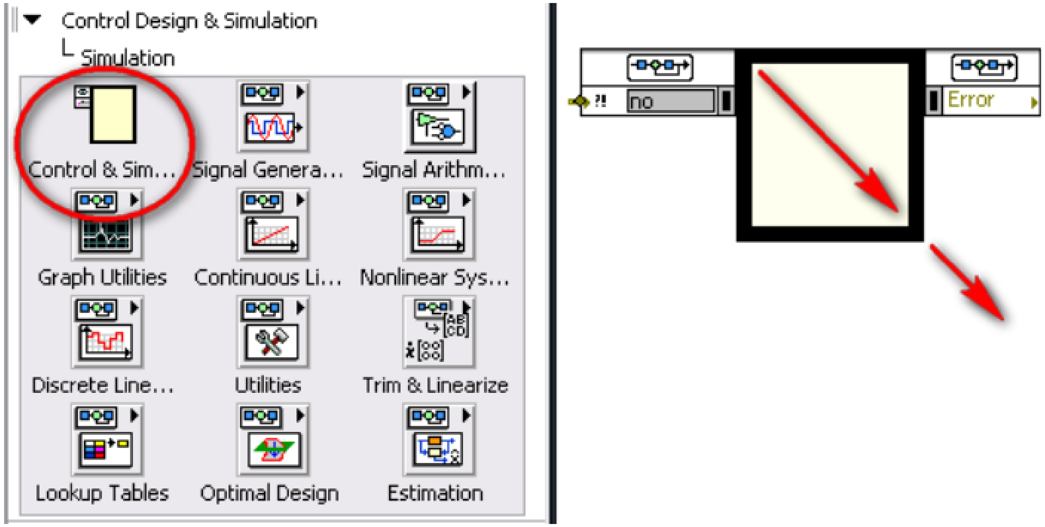


Figure 1. Create a Control & Simulation Loop.

Step 2

- Again on the *Simulation* subpalette, select *Continuous Linear Systems* and click once on *Transfer Function* and once inside the Control & Simulation Loop you created previously. This places a Transfer Function block inside the Control & Simulation Loop. Now double-click on the Transfer Function block to input the transfer function parameters.

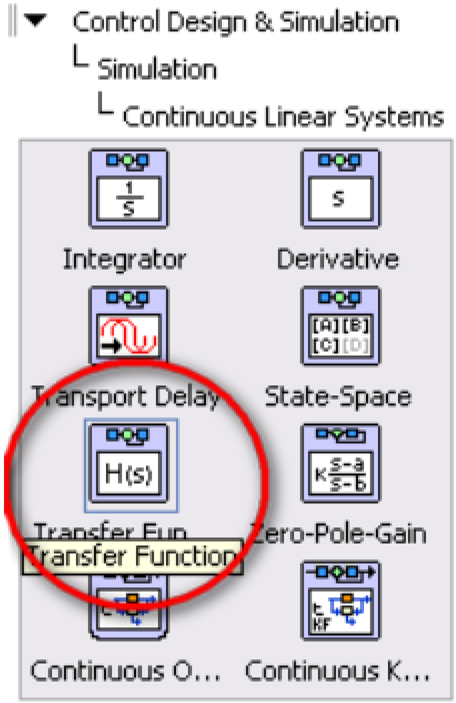
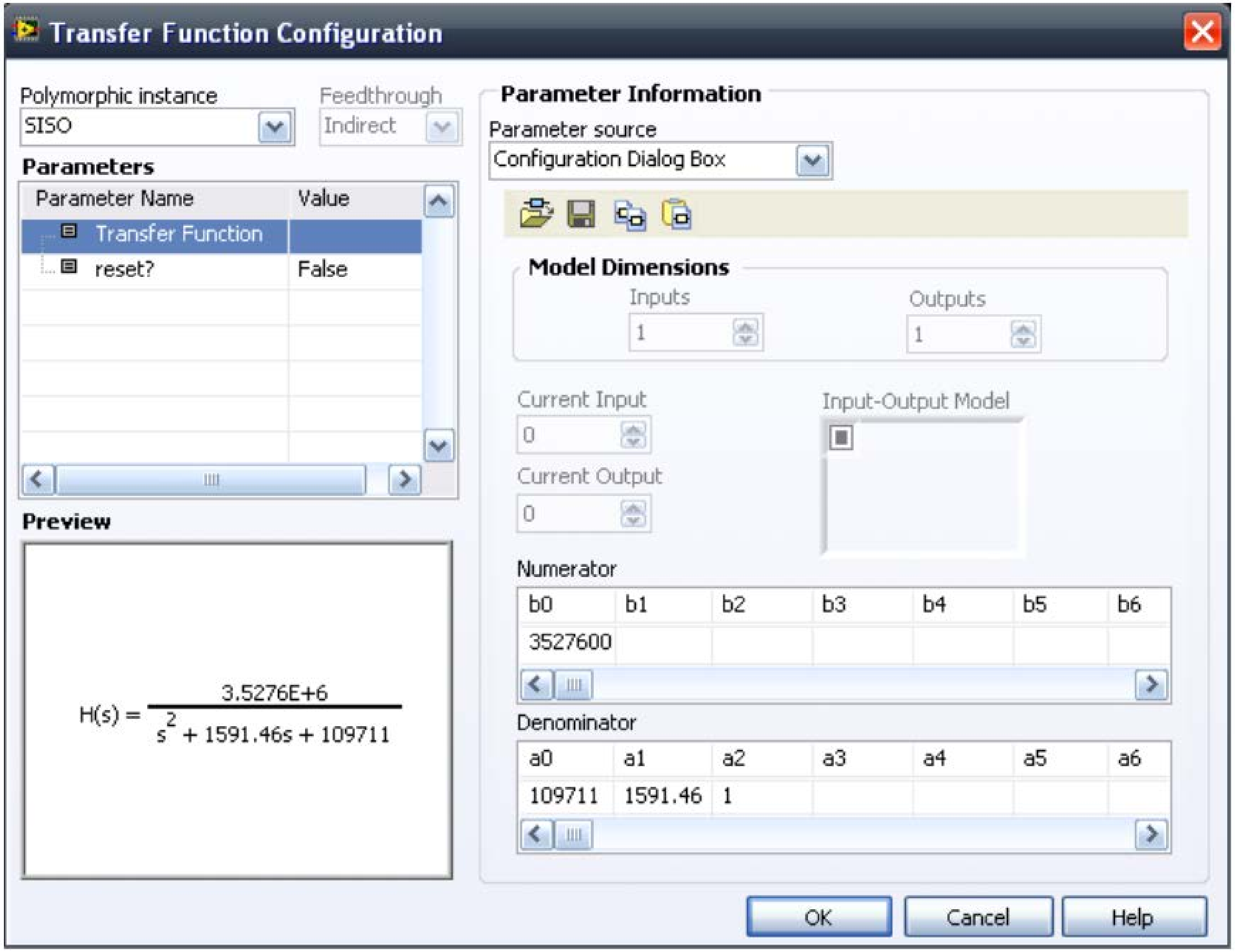


Figure 2. Select a Transfer Function.

Step 3



Transfer Function Configuration

Polymorphic instance: SISO

Feedthrough: Indirect

Parameters

Parameter Name	Value
Transfer Function	
reset?	False

Preview

$$H(s) = \frac{3.5276E+6}{s^2 + 1591.46s + 109711}$$

Parameter Information

Parameter source: Configuration Dialog Box

Model Dimensions

Inputs: 1

Outputs: 1

Current Input: 0

Current Output: 0

Numerator

b0	b1	b2	b3	b4	b5	b6
3527600						

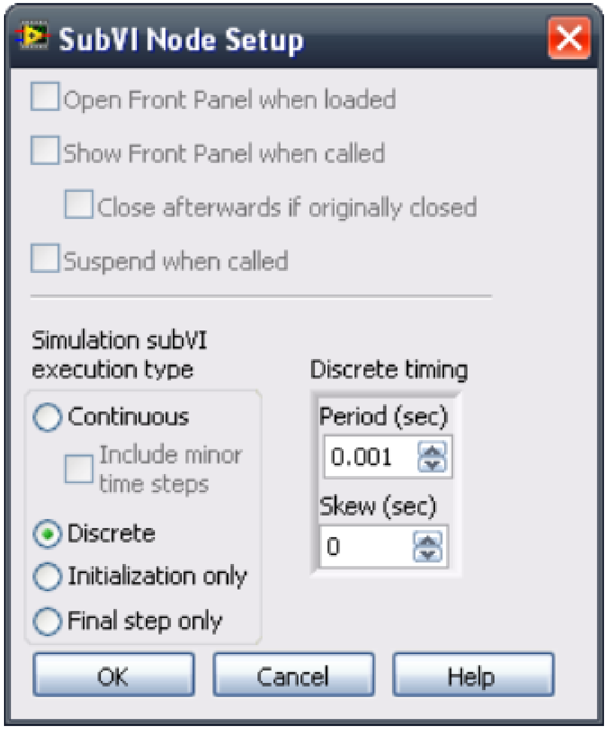
Denominator

a0	a1	a2	a3	a4	a5	a6
109711	1591.46	1				

Figure 3. Input Transfer Function parameters.

Step 4: Implement the PID algorithm

- On the function palette, select the *Control Design & Simulation->PID* subpalette and drag and drop the **PID.vi** into the Control & Simulation Loop.
- Because the PID algorithm is going to run on a Real-Time based operating system with a fixed loop rate, right-click on the **PID.vi** and select *SubVI Node Setup….* to bring up a configuration dialog window.
- You can use this window to configure the simulation loop to handle timing with this particular VI.
- Assume the controller is going to run at a 1000 Hz loop rate, so select *discrete* with a period value of 0.001 seconds.



SubVI Node Setup

☐ Open Front Panel when loaded
☐ Show Front Panel when called
  ☐ Close afterwards if originally closed
☐ Suspend when called

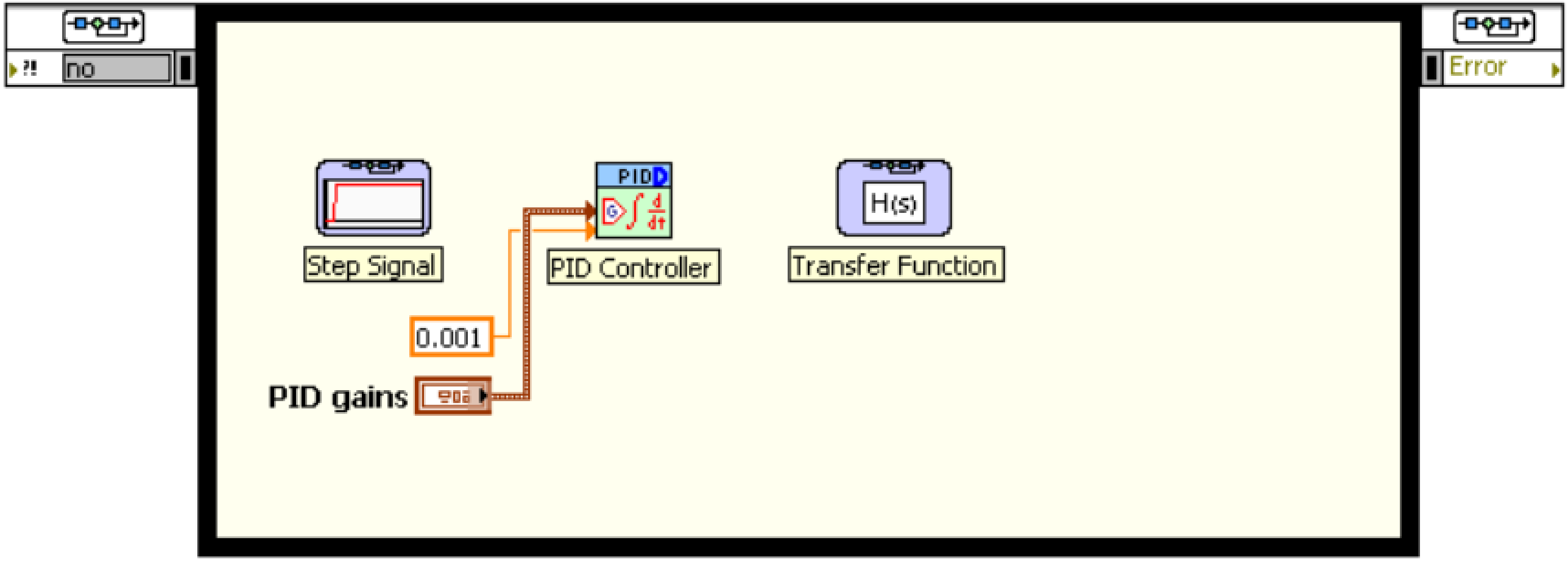
Simulation subVI execution type
○ Continuous
  ☐ Include minor time steps
● Discrete
○ Initialization only
○ Final step only

Discrete timing
Period (sec): 0.001
Skew (sec): 0

Step 4: Implement the PID algorithm

- The “D” that appears on the PID VI indicates that it is being handled as a discrete system.
- Run the cursor over the PID VI until you are on top of the *PID Gains Terminal* (you might type CTRL+H to Show Context Help if you cannot find it). Right-click and select *Create->Control*. This creates a control on the Front Panel that you can use to change the PID gains interactively. Finally, right-click on the *dt(s)* terminal and create a constant. This should be the same as the digital period you created previously, 0.001 seconds.
- To create an input signal, use a step signal. From *Control Design & Simulation->Simulation->Signal Generation*, select *Step Signal* and drop it into the simulation loop. Leave parameters as they are configured by default.

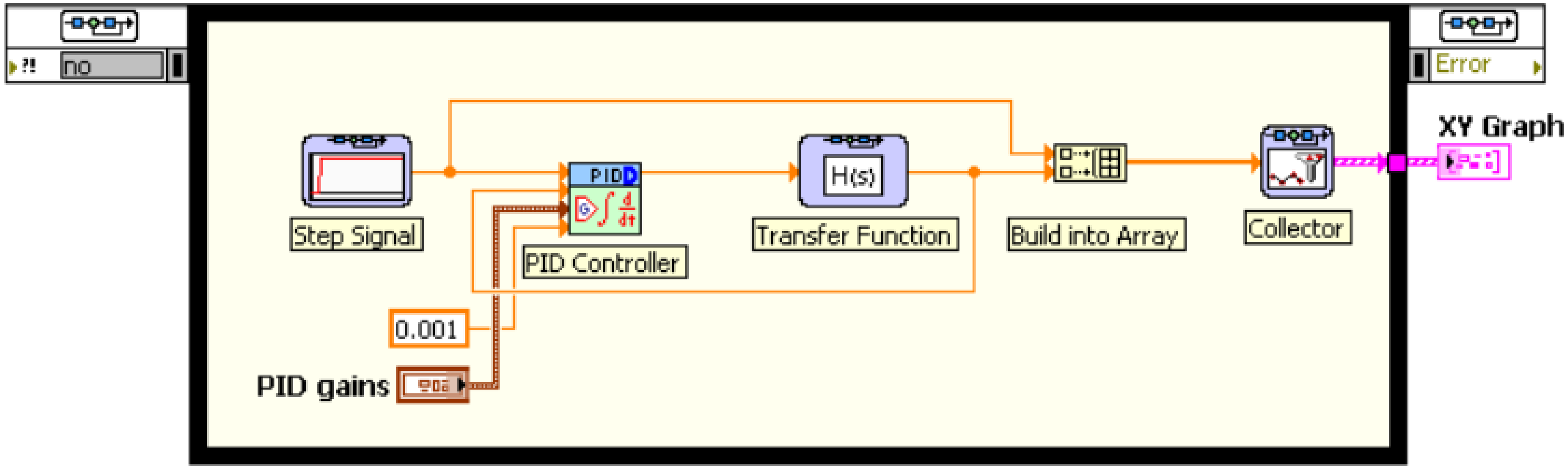
Step 4: Implement the PID algorithm



Step 5: View Simulation Results

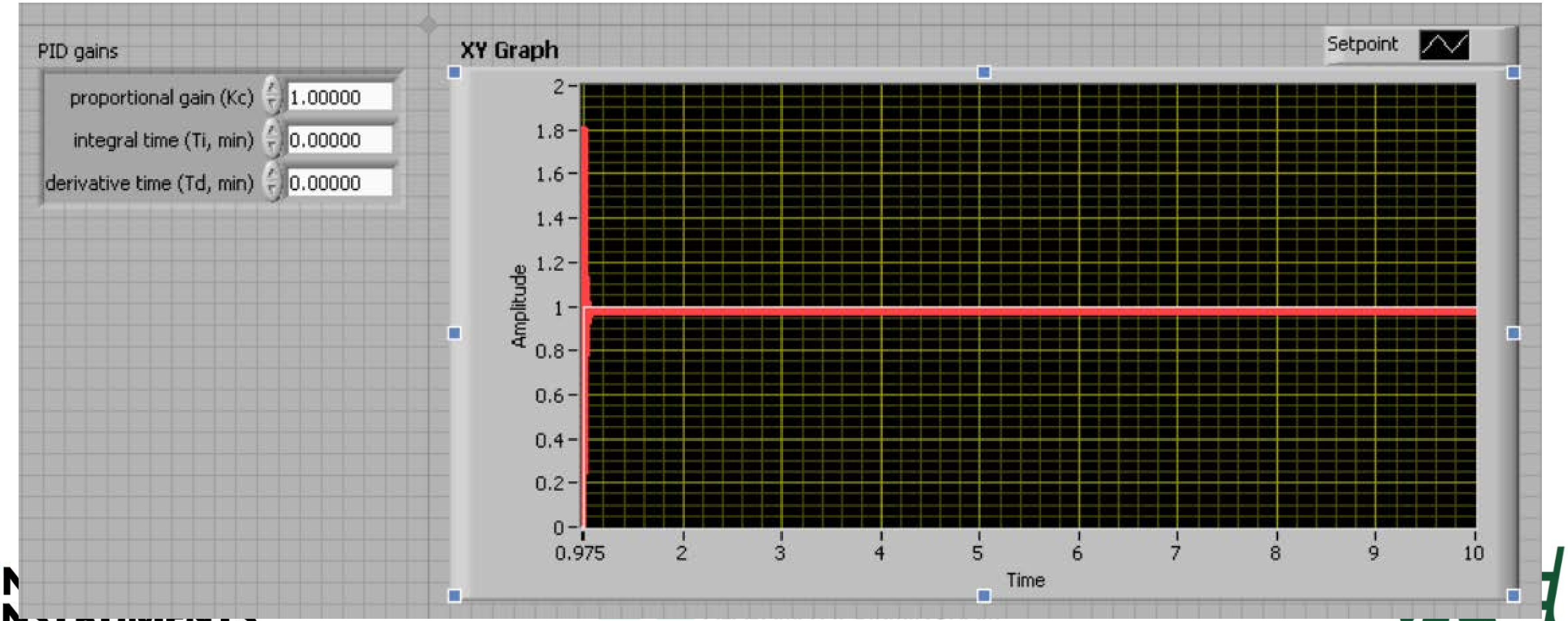
- First bundle the input (Step Signal) with the output from the motor transfer function into a *Build Array* node, which you can find on the *Programming->Arrays* subpalette.
- Collect these signals and plot them on a graph on the Front Panel.
- To do so, select *Control Design & Simulation->Simulation->Utilities* then select and drop *Collector*.
- On the Front Panel, create an XY Graph to display the simulation results.
- Connect all the signals as shown in Figure 5.

Step 6: Simulation loop



Step 6: Front Panel

- If you rearrange the Front Panel elements and use the default values, you will end up with a graph similar to Figure 6:



PID gains

proportional gain (Kc) 1.00000

integral time (Ti, min) 0.00000

derivative time (Td, min) 0.00000

Step 7: Fine-Tune the Simulation

- Use LabVIEW native graphical capabilities to improve the simulation and fine-tune the PID gains.
- First, change axis properties to have a better view of the simulation results. Right-click on the border of the XY Graph and uncheck the *AutoScale X* property under the *X Scale* option

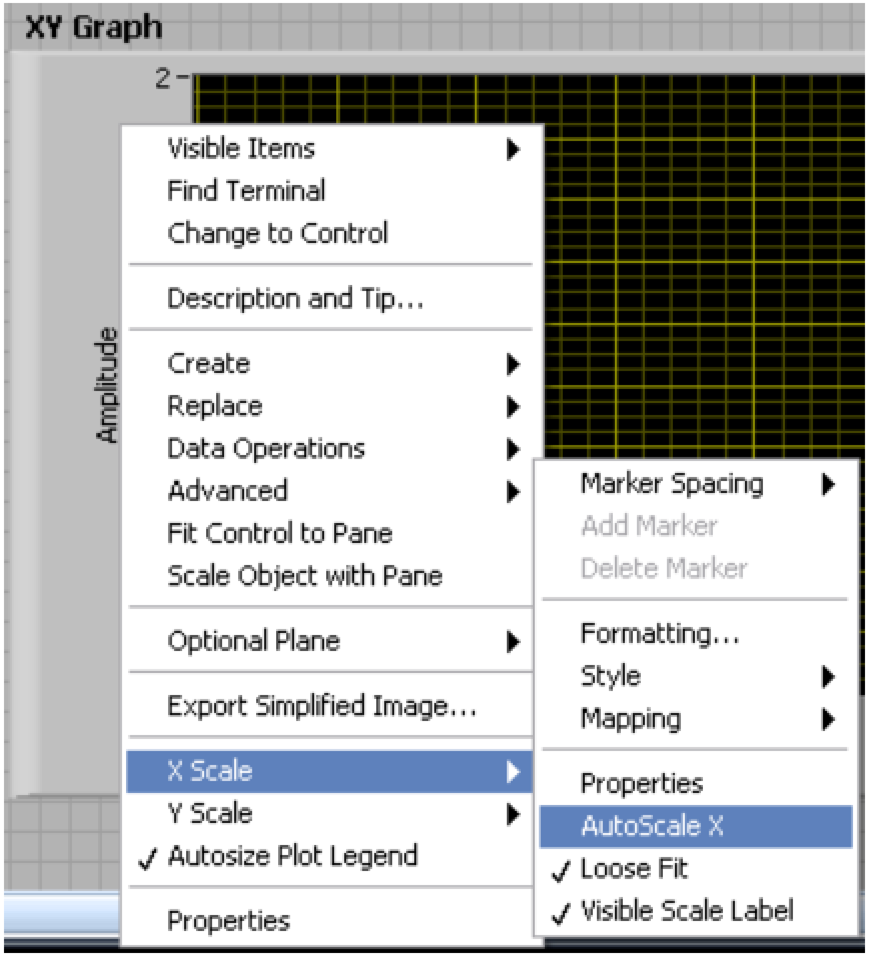
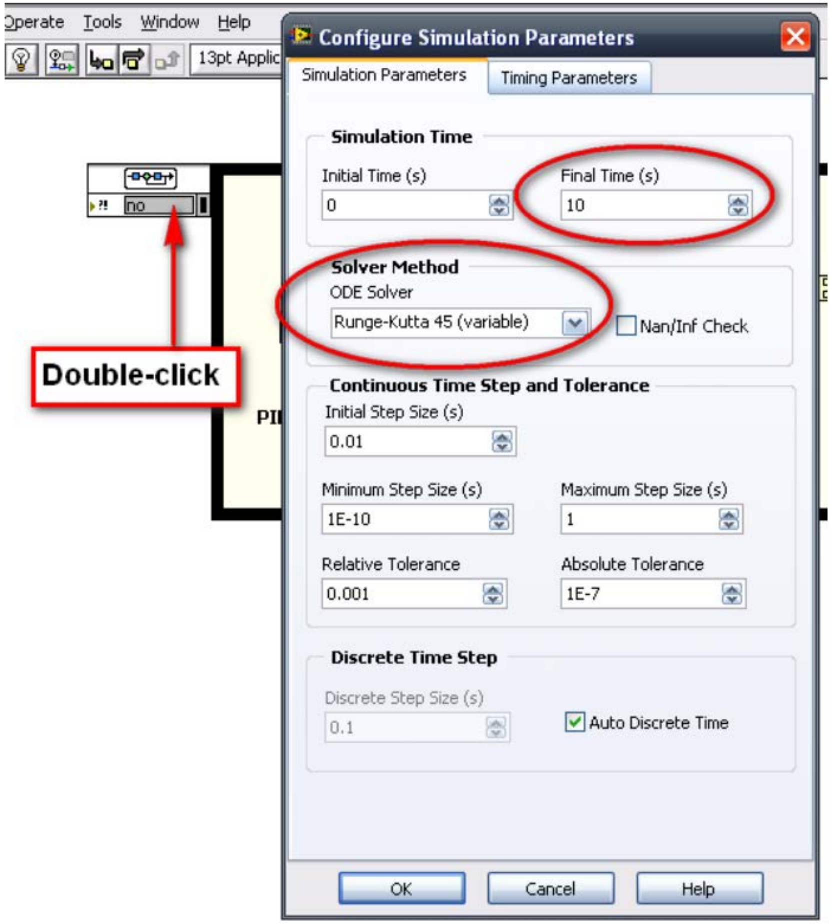


Figure 7. Graph properties.

Step 8: Configure Simulation Parameters

- Before making changes to the PID controller, we will make the simulation more efficient. As seen on Figure 6 there is no need to simulate the default 10 seconds; the plant is fast enough so that a final simulation time of 2 seconds is enough. Now modify the simulation parameters by double-clicking on the Control & Simulation Loop configuration pane which calls up the Configure Simulation Parameters dialog window. Implement the parameters as shown in Figure 8.



Double-click

Configure Simulation Parameters

Simulation Parameters | Timing Parameters

Simulation Time

Initial Time (s): 0

Final Time (s): 10

Solver Method

ODE Solver: Runge-Kutta 45 (variable)

☐ Nan/Inf Check

Continuous Time Step and Tolerance

Initial Step Size (s): 0.01

Minimum Step Size (s): 1E-10

Maximum Step Size (s): 1

Relative Tolerance: 0.001

Absolute Tolerance: 1E-7

Discrete Time Step

Discrete Step Size (s): 0.1

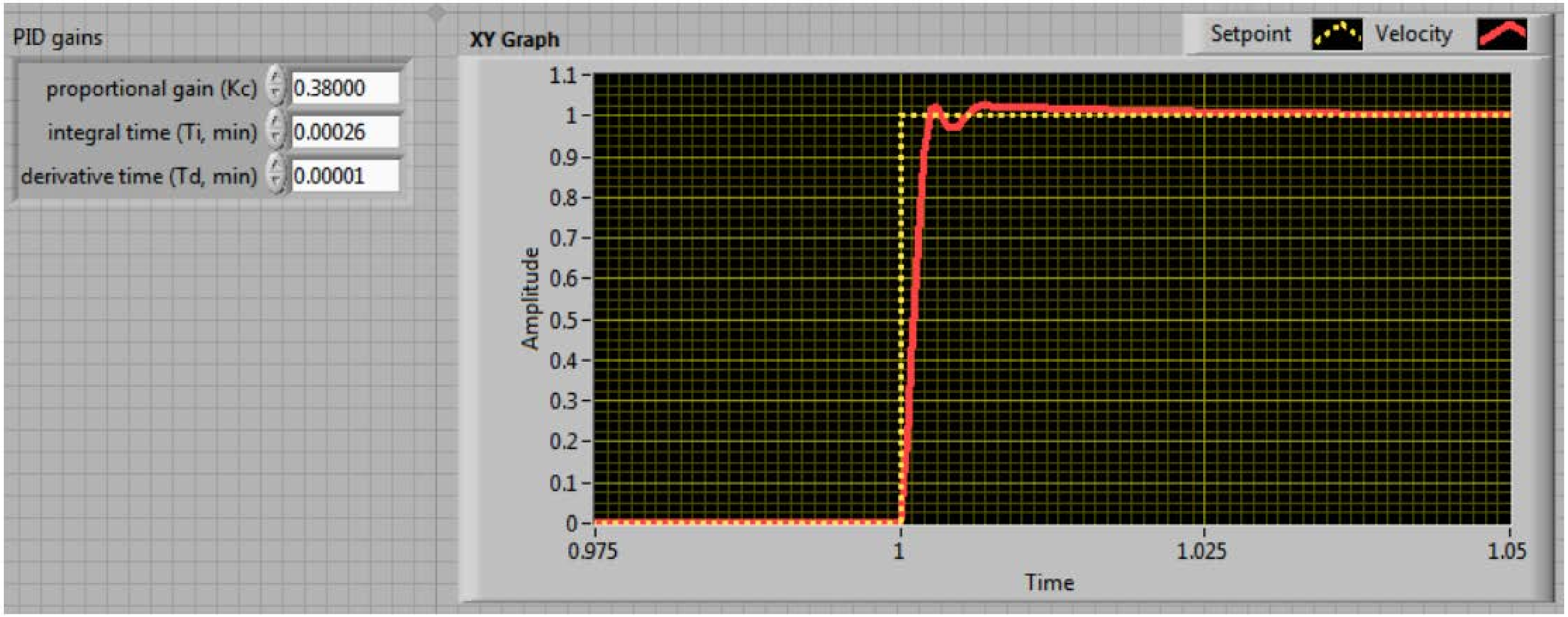
☑ Auto Discrete Time

Figure 8. Change the simulation parameters.

Step 9: PID Tuning

- You now can run the VI continuously and change the PID gains until you are satisfied with the results.
- A typical procedure to tune a PID controller would be

1. Kc to 1 and Ti, Td to zero. Keep increasing/decreasing Kc until the response has some overshoot
2. Modify Td to make the system faster and compensate the overshoot
3. Modify Ti to remove any steady-state error on the step response



PID gains

proportional gain (Kc) 0.38000
integral time (Ti, min) 0.00026
derivative time (Td, min) 0.00001

Figure 8. Simulation results with Kc=0.38, Ti=0.00026, and Td=0.0001.

Conclusion

- You now know how to simulate a discrete-based PID controller with the continuous dynamic system behavior of a DC motor. You can apply this technique to any kind of hybrid system where continuous and discrete behavior is mixed. One of the benefits of the procedure shown is that the control algorithm you used is exactly the same as the one you would use in a Real-Time implementation, and you can take advantage of many of its features, such as integral anti windup.

THANK YOU

Questions..?!