King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318: Computer Control Systems

# Laboratory Exercises

Prepared by

Dr. Lahouari Cheded
Mr. Mohammad Shahab
Mr Ameenuddin Hussain

Term 082
02 February, 2009

# Table of Contents

King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 1: Introduction to LabVIEW

**LabVIEW** (short for **Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) is a platform and development environment for a visual programming language from National Instruments. The graphical language is named "G". LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, various flavors of UNIX, Linux, and Mac OS.

One benefit of LabVIEW over other development environments is the extensive support for accessing instrumentation hardware. Drivers and abstraction layers for many different types of instruments and buses are included or are available for inclusion. These present themselves as graphical nodes. The abstraction layers offer standard software interfaces to communicate with hardware devices. The provided driver interfaces save program development time. Many libraries with a large number of functions for data acquisition, signal generation, mathematics, statistics, signal conditioning, analysis, etc., along with numerous graphical interface elements are provided in several LabVIEW package options. Another benefit of the LabVIEW environment is the platform independent nature of the *G*-code (LabVIEW programming language), which is (with the exception of a few platform-specific functions) portable between the different LabVIEW systems for different operating systems (Windows, MacOSX and Linux).

## *Dataflow*

The programming language used in LabVIEW, called "G", is a dataflow language. Execution is determined by the structure of a graphical block diagram (the LV-source code) on which the programmer connects different function-nodes by drawing wires. These wires propagate variables and any node can execute as soon as all its input data become available. Since this might be the case for multiple nodes simultaneously, G is inherently capable of parallel execution.

The afore-mentioned data-flow (which can be "forced", typically by linking inputs and outputs of nodes) completely defines the execution sequence, and that can be fully controlled by the programmer. Thus, the execution sequence of the LabVIEW graphical syntax is as well-defined as with any textually coded language such as C, Visual BASIC, etc. Furthermore, LabVIEW does not require type definition of the variables; the wire type is defined by the data-supplying node. LabVIEW supports polymorphism in that wires automatically adjust to various types of data.

Screenshot of a simple LabVIEW program that generates, synthesizes, analyzes and displays waveforms, showing the block diagram and front panel. Each symbol on the block diagram represents a LabVIEW subroutine (subVI) which can be another LabVIEW program or a LV library function.

## *Graphical programming*

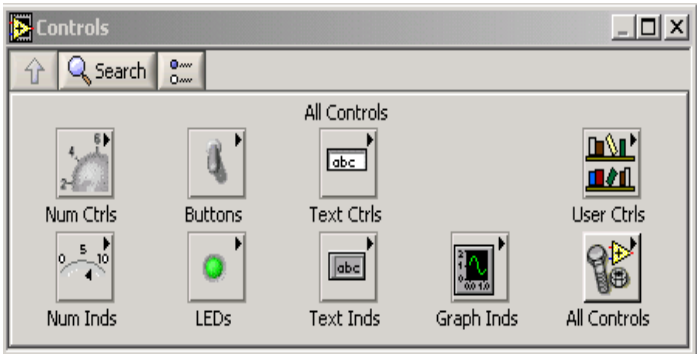
LabVIEW ties the creation of user interfaces (called front panels) into the development cycle. LabVIEW programs/subroutines are called virtual instruments (VIs). Each VI has three components: Each VI contains three main parts:

1. Front Panel - How the user interacts with the VI.
2. Block Diagram - The code that controls the program.
3. Icon/Connector - Means of connecting a VI to other VIs.

The **Front Panel** is the user interface of the VI, to interact with the user when the program is running. Users can control the program, change inputs, and see data updated in real time. You build the front panel with controls and indicators (available on the **Control Palette**), which are the interactive input and output terminals of the VI, respectively. Controls are knobs, pushbuttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays.

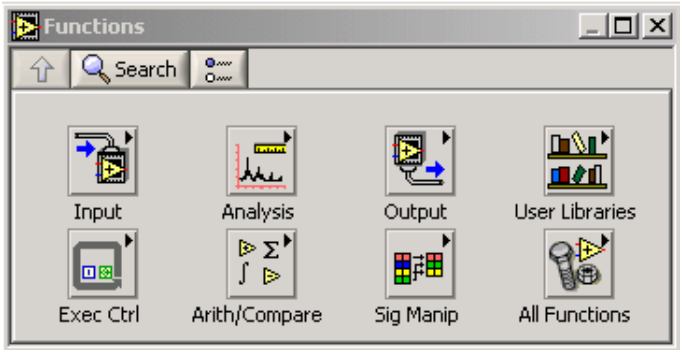
Controls simulate instrument input devices and supply values to the block diagram of the VI. Indicators simulate instrument output devices and display values the block diagram acquires or generates. These may include data, program states, and other information. Every front panel control or indicator has a corresponding terminal on the block diagram. When a VI is run, values from controls flow through the block diagram, where they are used in the functions on the diagram, and the results are passed into other functions or indicators.

**The Controls** palette is available only on the front panel. Go to **Window»Show Controls Palette** or right-click the front panel workspace to display the **Controls** palette.




The **Block Diagram** contains this graphical source code. Front panel objects appear as terminals on the block diagram. Additionally, the block diagram contains functions and structures from built-in LabVIEW VI libraries. These can be accessed from the **Functions Palette**. Wires connect each of the nodes on the block diagram, including control and indicator terminals, functions, and structures.

The **Functions** palette is available only on the block diagram. Select **Window»Show Functions Palette** or right-click the block diagram workspace to display the **Functions** palette




The **Icon/Connector** may represent the VI as a subVI in block diagrams of calling VIs. Controls and indicators on the front panel allow an operator to input data into or extract data from a running virtual instrument. However, the front panel can also serve as a programmatic interface. Thus a virtual instrument can either be run as a program, with the front panel serving as a user interface, or, when

dropped as a node onto the block diagram, the front panel defines the inputs and outputs for the given node through the connector pane. This implies each VI can be easily tested before being embedded as a subroutine into a larger program.

The graphical approach also allows non-programmers to build programs by simply dragging and dropping virtual representations of the lab equipment with which they are already familiar. The LabVIEW programming environment, with the included examples and the documentation, makes it easy to create small applications. For complex algorithms or large-scale code it is important that the programmer possess an extensive knowledge of the special LabVIEW syntax and the topology of its memory management. The most advanced LabVIEW development systems offer the possibility of building stand-alone applications.

Other features that you will need include the **Tools Palette** which is a Floating Palette that is used to operate and modify front panel and block diagram objects and the **Status Toolbar**. They contain the following tools and buttons, respectively:

**Tools Palette:**


Automatic Selection Tool

Operating Tool

Positioning/Resizing Tool

Labeling Tool

Wiring Tool

Shortcut Menu Tool

Scrolling Tool

Breakpoint Tool

Probe Tool

Color Copy Tool

Coloring Tool

**Status Bar:**


Run Button

Continuous Run Button

Abort Execution

Pause/Continue Button

Text Settings

Align Objects

Distribute Objects

Reorder

Resize front panel objects

Execution Highlighting Button

Step Into Button

Step Over Button

Step Out Button

## Exercise 0 - Open and Run a Virtual Instrument

Examine the **Signal Generation and Processing** VI and run it. Change the frequencies and types of the input signals and notice how the display on the graph changes. Change the Signal Processing Window and Filter options. After you have examined the VI and the different options you can change, stop the VI by pressing the Stop button.

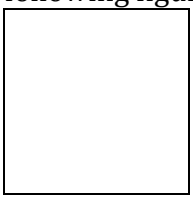
1. Select **Start»Programs»National Instruments»LabVIEW 7.0»LabVIEW** to launch LabVIEW. The **LabVIEW** dialog box appears.

2. Select **Help»Find Examples.** The dialog box that appears lists and links to all available LabVIEW example VIs.

3. On the Browse Tab, select browse according to task. Choose **Analyzing and Processing Signals**, then **Signal Processing**, then **Signal Generation and Processing.vi.**
This will open the Signal Generation and Processing VI Front Panel.

**Front Panel**

4. Click the **Run** button on the toolbar, shown at left, to run this VI.
This VI determines the result of filtering and windowing a generated signal. This example also displays the power spectrum for the generated signal. The resulting signals are displayed in the graphs on the front panel, as shown in the following figure.



5. Use the Operating tool, shown at left, to change the Input Signal and the Signal Processing, use the increment or decrement arrows on the control, and drag the pointer to the desired Frequency.

6. Press the **More Info…** button or [F5] to read more about the analysis functions.

7. Press the **Stop** button or [F4] to stop the VI.

**Block Diagram**

8. Select **Window»Show Diagram** or press the <Ctrl-E> keys to display the block diagram for the Signal Generation and Processing VI.
This block diagram contains several of the basic block diagram elements, including subVIs, functions, and structures, which you will learn about later in this course.

9. Select **Window»Show Panel** or press the <Ctrl-E> keys to return to the Front Panel.

10. Close the VI and do not save changes.

## End of Exercise

King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 2: Building a VI

Before building our own VI we will first examine the **Signal Generation and Processing** VI and run it. Change the frequencies and types of the input signals and notice how the display on the graph changes. Change the Signal Processing Window and Filter options. After you have examined the VI and the different options you can change, stop the VI by pressing the Stop button.

1. Select **Start»Programs»National Instruments»LabVIEW 8.0 » LabVIEW** to launch LabVIEW. The **LabVIEW** dialog box appears.

2. Select **Help»Find Examples.** The dialog box that appears lists and links to all available LabVIEW example VIs.

3. On the Browse Tab, select browse according to task. Choose **Analyzing and Processing Signals**, then **Signal Processing**, then **Signal Generation and Processing.vi.**
This will open the Signal Generation and Processing VI Front Panel.

**Note** You also can open the VI by clicking the **Open VI** button and navigating to labview\examples\apps\demos.llb\Signal Generation and Processing.vi.

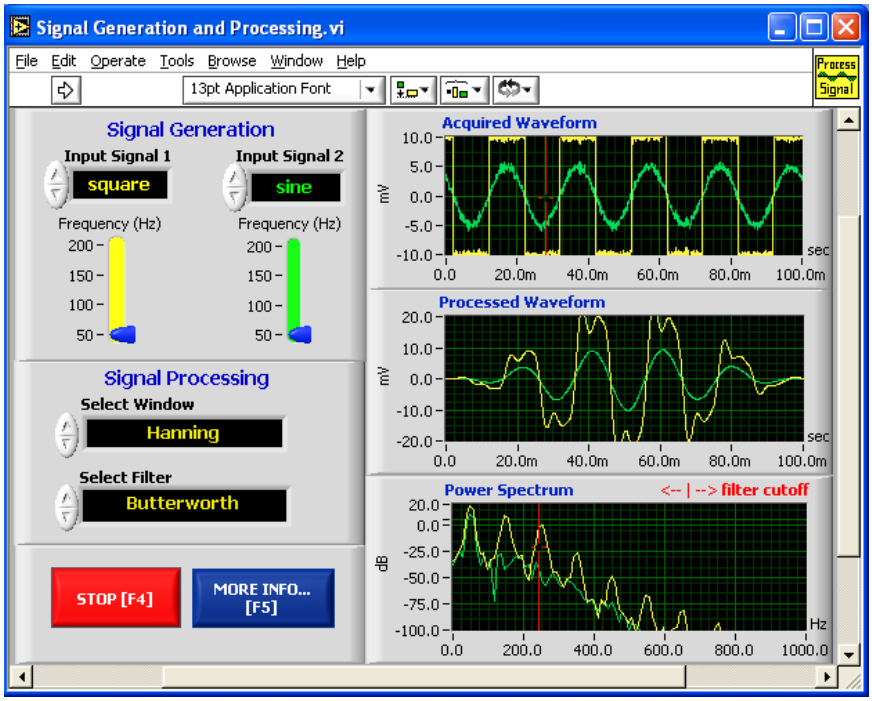
**Front Panel**

4. Click the **Run** button on the toolbar, shown at left, to run this VI.
This VI determines the result of filtering and windowing a generated signal. This example also displays the power spectrum for the generated signal. The resulting signals are displayed in the graphs on the front panel, as shown in the following figure.

5. Use the Operating tool, shown at left, to change the Input Signal and the Signal Processing, use the increment or decrement arrows on the control, and drag the pointer to the desired Frequency.

6. Press the **More Info…** button or [F5] to read more about the analysis functions.

7. Press the **Stop** button or [F4] to stop the VI.




**Block Diagram**

8. Select **Window»Show Diagram** or press the <Ctrl-E> keys to display the block diagram for the Signal Generation and Processing VI.

This block diagram contains several of the basic block diagram elements, including subVIs, functions, and structures, which you will learn about later in this course.

9. Select **Window»Show Panel** or press the <Ctrl-E> keys to return to the Front Panel.

10. Close the VI and do not save changes.

**Creating your own VI**

When you create an object on the Front Panel, a terminal will be created on the Block Diagram. These terminals give you access to the Front Panel objects from the Block Diagram code. Each terminal contains useful information about the Front Panel object it corresponds to. For example, the color and symbols provide the data type. Double-precision, floating point numbers are represented with orange terminals and the letters DBL, boolean terminals are green with TF lettering. In general, orange terminals should wire to orange terminals, green to green, and so on. This is not a hard-and-fast rule; LabVIEW will allow a user to connect a blue terminal (integer value) to an orange terminal (fractional value), for example.

For more help with the terminals, right-click on the function and select **Visible Items»Terminal**s. The functions picture will be pulled back to reveal the connection terminals. For additional help, select **Help»Show Context Hel**p, or press <Ctrl-H>. As you move your mouse over the function, this window will show you the function, terminals, and a brief help description in the context help window. For organization you can, right-click on the particular wire in question and choose **Clean Up Wire** to automatically re-route that wire.

## Exercise 1- Convert °C to °F

Complete the following steps to create a VI that takes a number representing degrees Celsius and converts it to a number representing degrees Fahrenheit.

In wiring illustrations, the arrow at the end of this mouse icon shows where to click and the number on the arrow indicates how many times to click.

**Front Panel**

1. Select **File»New** to open a new front panel.

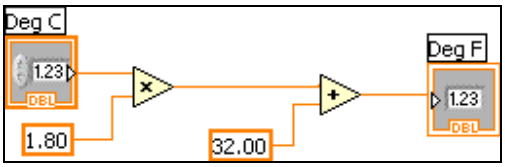

2. (Optional) Select **Window»Tile Left and Right** to display the front panel and block diagram side by side.
3. Create a numeric digital control. You will use this control to enter the value for degrees Centigrade.
    a. Select the digital control on the **Controls»Numeric Controls** palette. If the **Controls** palette is not visible, right-click an open area on the front panel to display it.
    b. Move the control to the front panel and click to place the control.
    c. Type deg C inside the label and click outside the label or click the **Enter** button on the toolbar. If you do not type the name immediately, LabVIEW uses a default label. You can edit a label at any time by using the Labeling tool.
4. Create a numeric digital indicator. You will use this indicator to display the value for degrees Fahrenheit.
    a. Select the digital indicator on the **Controls»Numeric Indicators** palette.
    b. Move the indicator to the front panel and click to place the indicator.
    c. Type deg F inside the label and click outside the label or click the **Enter** button.

LabVIEW creates corresponding control and indicator terminals on the block diagram. The terminals represent the data type of the control or indicator. For example, a DBL terminal represents a double-precision, floating-point numeric control or indicator.

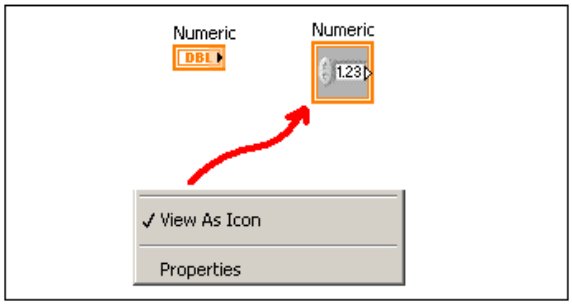
**Note** Control terminals have a thicker border than indicator terminals.

**Block Diagram**

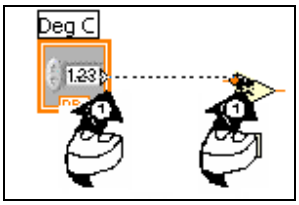
5. Display the block diagram by clicking it or by selecting **Window»Show Diagram**.




**Note:** Block Diagram terminals can be viewed as icons or as terminals. To change the way LabVIEW displays these objects right click on a terminal and select **View As Icon**.




6. Select the Multiply and Add functions on the **Functions»Numeric** palette and place them on the block diagram. If the **Functions** palette is not visible, right-click an open area on the block diagram to display it.
7. Select the numeric constant on the **Functions»Numeric** palette and place two of them on the block diagram. When you first place the numeric constant, it is highlighted so you can type a value.
8. Type 1.8 in one constant and 32.0 in the other. If you moved the constants before you typed a value, use the Labeling tool to enter the values.
9. Use the Wiring tool to wire the icons as shown in the previous block diagram.
    - To wire from one terminal to another, use the Wiring tool to click the first terminal, move the tool to the second terminal, and click the second terminal, as shown in the following illustration. You can start wiring at either terminal.




    - You can bend a wire by clicking to tack the wire down and moving the cursor in a perpendicular direction. Press the spacebar to toggle the wire direction.
    - To identify terminals on the nodes, right-click the Multiply and Add functions and select **Visible Items»Terminals** from the shortcut menu to display the connector pane. Return to the icons after wiring by right-clicking the functions and selecting **Visible Items»**
    - **Terminals** from the shortcut menu to remove the checkmark.
    - When you move the Wiring tool over a terminal, the terminal area blinks, indicating that clicking will connect the wire to that terminal and a tip strip appears, listing the name of the terminal.
    - To cancel a wire you started, press the <Esc> key, right-click, or click the source terminal.
10. Display the front panel by clicking it or by selecting **Window»Show Panel**.
11. Save the VI for later use (have a folder by your own name).
12. Enter a number in the digital control and run the VI.
    a. Use the Operating tool or the Labeling tool to double-click the digital control and type a new number.
    b. Click the **Run** button to run the VI.
    c. Try several different numbers and run the VI again.

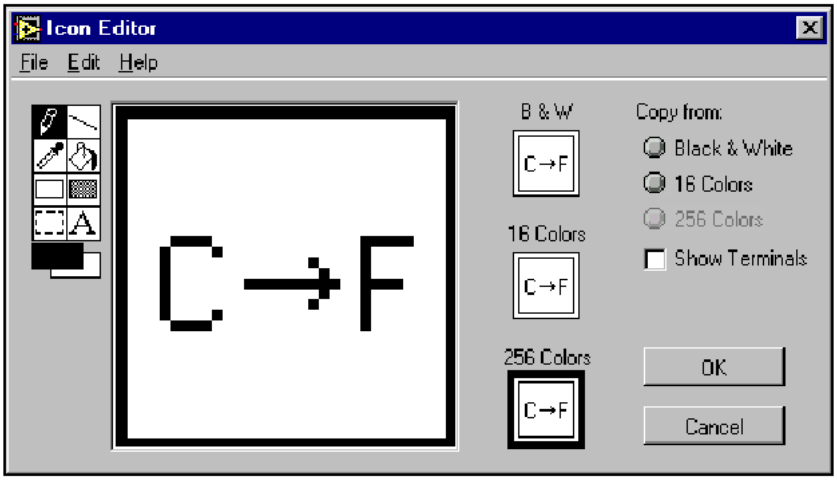
## Exercise 2 – Create a SubVI

### Front Panel

1. Select **File»Open** and navigate to your folder to open the C to F VI.

The following front panel appears.


2. Right-click the icon in the upper right corner of the front panel and select **Edit Icon** from the shortcut menu. The **Icon Editor** dialog box appears.
3. Double-click the Select tool on the left side of the **Icon Editor** dialog box to select the default icon.
4. Press the <Delete> key to remove the default icon.
5. Double-click the Rectangle tool to redraw the border.
6. Create the following icon.




   a. Use the Text tool to click the editing area.
   b. Type C and F.
   c. Double-click the Text tool and change the font to **Small Fonts**.
   d. Use the Pencil tool to create the arrow. (To draw horizontal or vertical straight lines, press the <Shift> key while you use the Pencil tool to drag the cursor).
   e. Use the Select tool and the arrow keys to move the text and arrow you created.
   f. Select the **B&W** icon and select **256 Colors** in the **Copy from** field to create a black and white icon, which LabVIEW uses for printing unless you have a color printer.
   g. When the icon is complete, click the **OK** button to close the **Icon Editor** dialog box. The icon appears in the upper right corner of the front panel and block diagram.

7. Right-click the icon on the front panel and select **Show Connector** from the shortcut menu to define the connector pane terminal pattern.
   LabVIEW selects a connector pane pattern based on the number of controls and indicators on the front panel. For example, this front panel has two terminals, **deg C** and **deg F**, so LabVIEW selects a connector pane pattern with two terminals.
8. Assign the terminals to the digital control and digital indicator.
   a. Select **Help»Show Context Help** to display the **Context Help** window. View each connection in the **Context Help** window as you make it.

b. Click the left terminal in the connector pane. The tool automatically changes to the Wiring tool, and the terminal turns black.
c. Click the **deg C** control. The left terminal turns orange, and a marquee highlights the control.
d. Click an open area of the front panel. The marquee disappears, and the terminal changes to the data type color of the control to indicate that you connected the terminal.
e. Click the right terminal in the connector pane and click the **deg F** indicator. The right terminal turns orange.
f. Click an open area on the front panel. Both terminals are orange.
g. Move the cursor over the connector pane. The **Context Help** window shows that both terminals are connected to floating-point values.

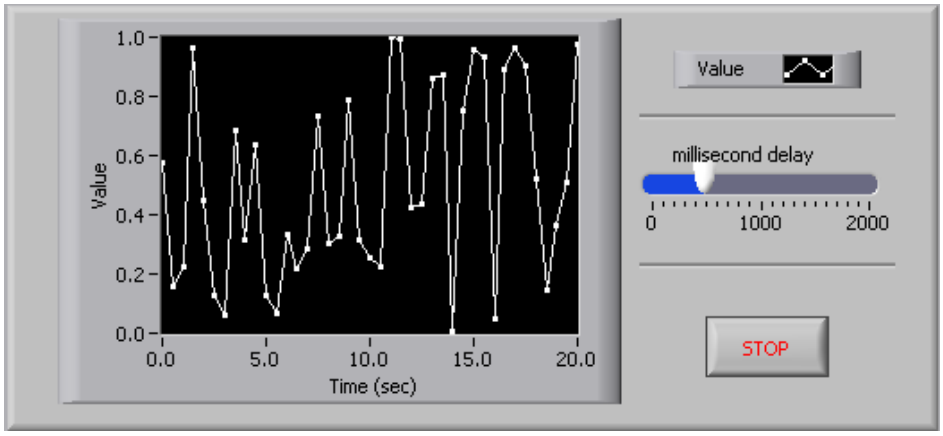
9. Select **File»Save** to save the VI.

## Exercise 3 – Using Loops

Use a while loop and a waveform chart to build a VI that demonstrates software timing.

## Front Panel

1. Open a new VI.
2. Build the following front panel.



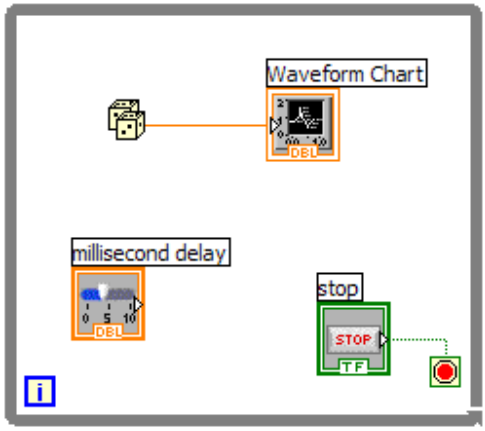

a. Select the horizontal pointer slide on the **Controls»Numeric Controls** palette and place it on the front panel. You will use the slide to change the software timing.
b. Type millisecond delay inside the label and click outside the label or click the **Enter** button on the toolbar, shown at left.
c. Place a Stop Button from the **Controls»Buttons** palette.
d. Select a waveform chart on the **Controls»Graph Indicators** palette and place it on the front panel. The waveform chart will display the data in real time.
e. Type Value History inside the label and click outside the label or click the **Enter** button.
f. The waveform chart legend labels the plot Plot 0. Use the Labeling tool to triple-click Plot 0 in the chart legend, type Value, and click outside the label or click the **Enter** button to relabel the legend.
g. The random number generator generates numbers between 0 and 1, in a classroom setting you could replace this with a data acquisition VI. Use the Labeling tool to double-click 10.0 in the y-axis, type 1, and click outside the label or click the **Enter** button to rescale the chart.

h. Change –10.0 in the y-axis to 0.
i. Label the y-axis Value and the x-axis Time (sec).

## Block Diagram

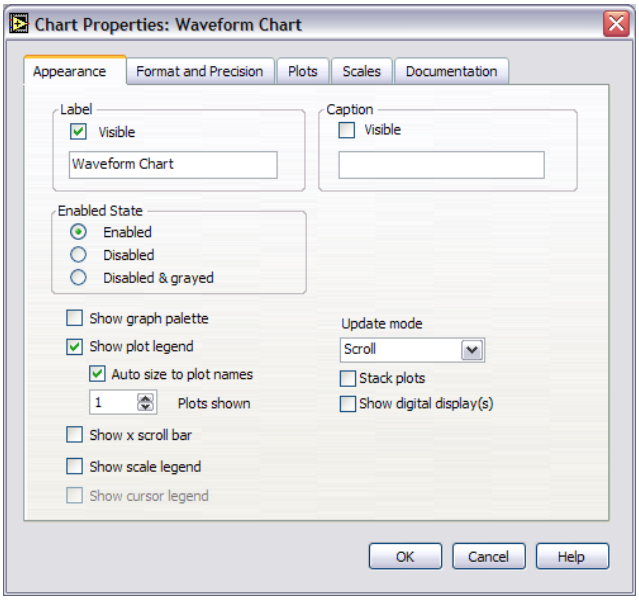
3. Select **Window»Show Diagram** to display the block diagram.

4. Enclose the two terminals in a While Loop, as shown in the following block diagram.




   a. Select the While Loop on the **Functions»Execution Control** palette.
   b. Click and drag a selection rectangle around the two terminals.
   c. Use the Positioning tool to resize the loop, if necessary.

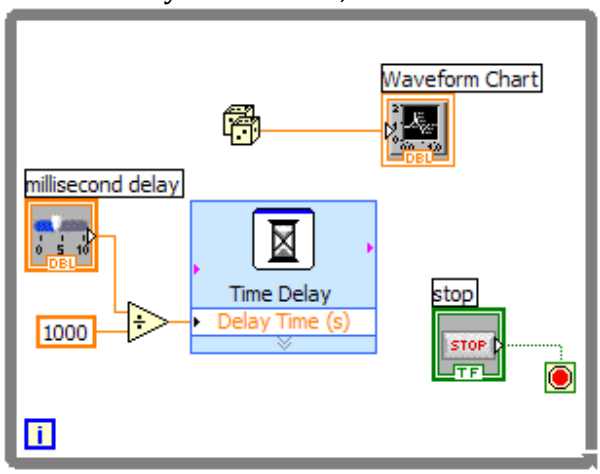
5. Select the Random Number (0-1) on the **Functions»Arithmetic and Comparison»Numeric** palette. Alternatively you could use a VI that is gathering data from an external sensor.
6. Wire the block diagram objects as shown in the previous block diagram.
7. Save the VI as Use a Loop.vi in your folder
8. Display the front panel by clicking it or by selecting **Window»Show Panel**.
9. Run the VI. The section of the block diagram within the While Loop border executes until the specified condition is TRUE. For example, while the STOP button is not pressed, the VI returns a new number and displays it on the waveform chart.
10. Click the STOP button to stop the acquisition. The condition is FALSE, and the loop stops executing.
11. Format and customize the X and Y scales of the waveform chart.
    a. Right-click the chart and select **Properties** from the shortcut menu. The following dialog box appears.
    b. Click the **Scale** tab and select different styles for the y-axis. You also can select different mapping modes, grid options, scaling factors, and formats and precisions. Notice that these will update interactively on the waveform chart
    c. Select the options you desire and click the **OK** button.




12. Right-click the waveform chart and select **Data Operations»Clear Chart** from the shortcut menu to clear the display buffer and reset the waveform chart. If the VI is running, you can select **Clear Chart** from the shortcut menu.

## Adding Timing

When this VI runs, the While Loop executes as quickly as possible. Complete the following steps to take data at certain intervals, such as once every half-second, as shown in the following block diagram.




a. Place the Time Delay Express VI located on the **Functions»Execution Control** palette. In the dialog box that appears, insert 0.5. This function would make sure that each iteration occurs every half-second (500 ms).
b. Divide the millisecond delay by 1000 to get time in seconds. Connect the output of the divide function to the Delay Time (s) input of the Time Delay Express VI. This will allow you to adjust the speed of the execution from the pointer slide on the front panel.

13. Save the VI, because you will use this VI later in the course.
14. Run the VI.
15. Try different values for the millisecond delay and run the VI again. Notice how this effects the speed of the number generation and display.

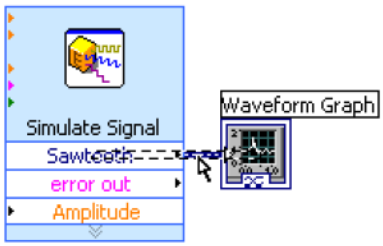
King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 2 (Extra)

## Modifying a Signal

Complete the following steps to scale the signal by 10 and display the results in the graph on the front panel.

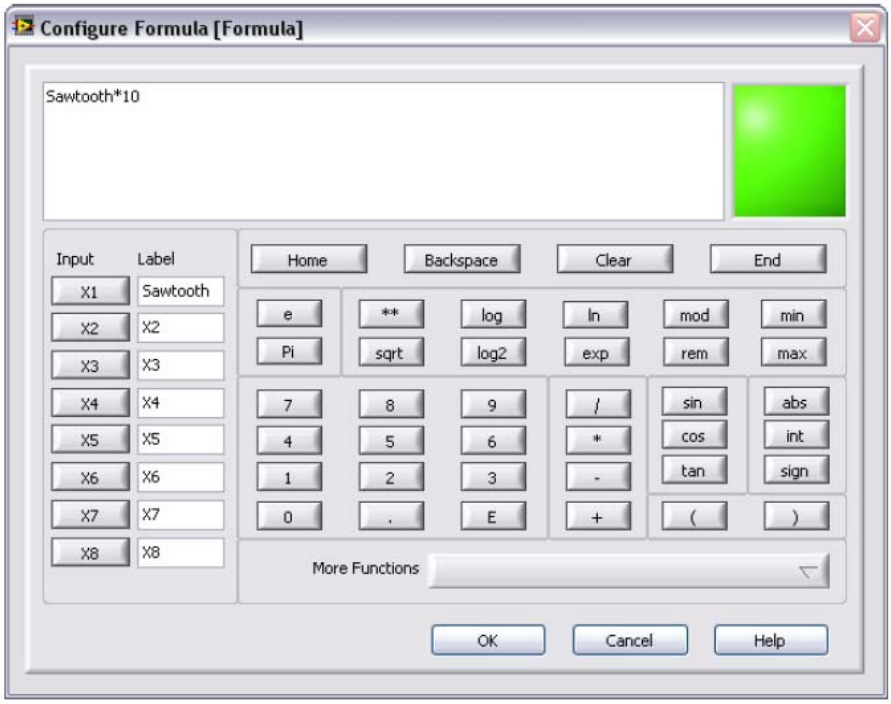



1. The **Functions** palette opens with the **Express** subpalette visible by default. If you have selected another subpalette, you can return to the **Express** subpalette by clicking **Express** on the **Functions** palette.
2. On the **Arithmetic & Comparison** palette, select the Formula Express VI, shown, and place it on the block diagram inside the loop between the Simulate Signal Express VI and the **Waveform Graph** terminal. You can move the **Waveform Graph** terminal to the right to make more room between the Express VI and the terminal. The **Configure Formula** dialog box appears when you place the Express VI on the block diagram. When you place an Express VI on the block diagram, the configuration dialog box for that VI always appears automatically.

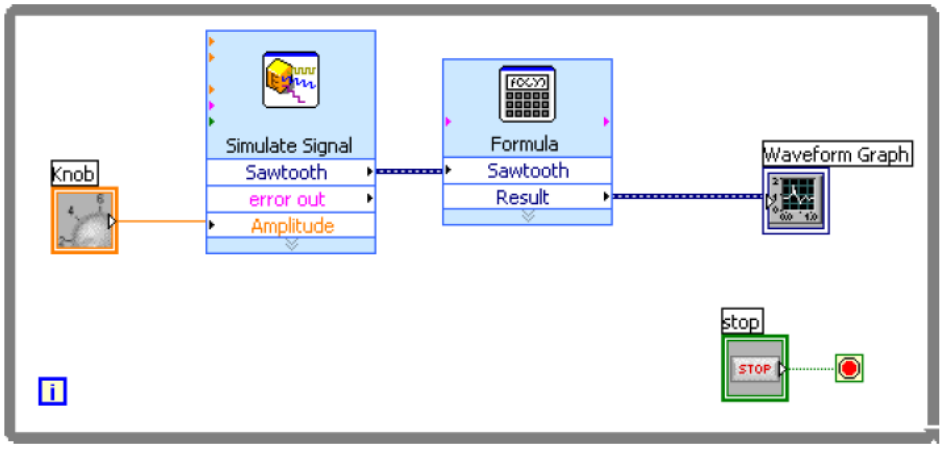

3. Click the **Help** button, in the bottom right corner of the **Configure Formula** dialog box to display the *LabVIEW Help* topic for this Express VI. The *Formula* help topic describes the Express VI, the configuration dialog box options, and the inputs and outputs of the Express VI. Each Express VI has a corresponding help topic you can access by clicking the **Help** button in the configuration dialog box or by right-clicking the Express VI and selecting **Help** from the shortcut menu.
4. In the *Formula* topic, find the dialog box option whose description indicates that it enters a variable into the formula.
5. Minimize the *LabVIEW Help* to return to the **Configure Formula** dialog box.
6. Change the text in the **Label** text box of the dialog box option you read about from **X1** to Sawtooth to indicate the input value to the Formula Express VI. When you click in the **String** text box at the top of the **Configure Formula** dialog box, the text changes to match the label you entered.
7. Define the value of the scaling factor by entering *10 after **Sawtooth** in the **String** text box. You can use the **Input** buttons in the configuration dialog box or you can use the *, 1, and 0 keyboard

buttons to enter the scaling factor. If you use the **Input** buttons in the configuration dialog box, LabVIEW places the formula input after the **Sawtooth** input in the **String** text box. If you use the keyboard, click in the **String** text box after **Sawtooth** and enter the formula you want to appear in the text box.

The **Configure Formula** dialog box should appear similar to




8. Click the **OK** button to save the current configuration and close the **Configure Formula** dialog box.
9. Move the cursor over the arrow on the **Sawtooth** output of the Simulate Signal Express VI.
10. When the Wiring tool appears, click the arrow on the **Sawtooth** output and then click the arrow on the **Sawtooth** input of the Formula Express VI, shown at left, to wire the two objects together.
11. Use the Wiring tool to wire the **Result** output of the Formula Express VI to the **Waveform Graph** terminal.

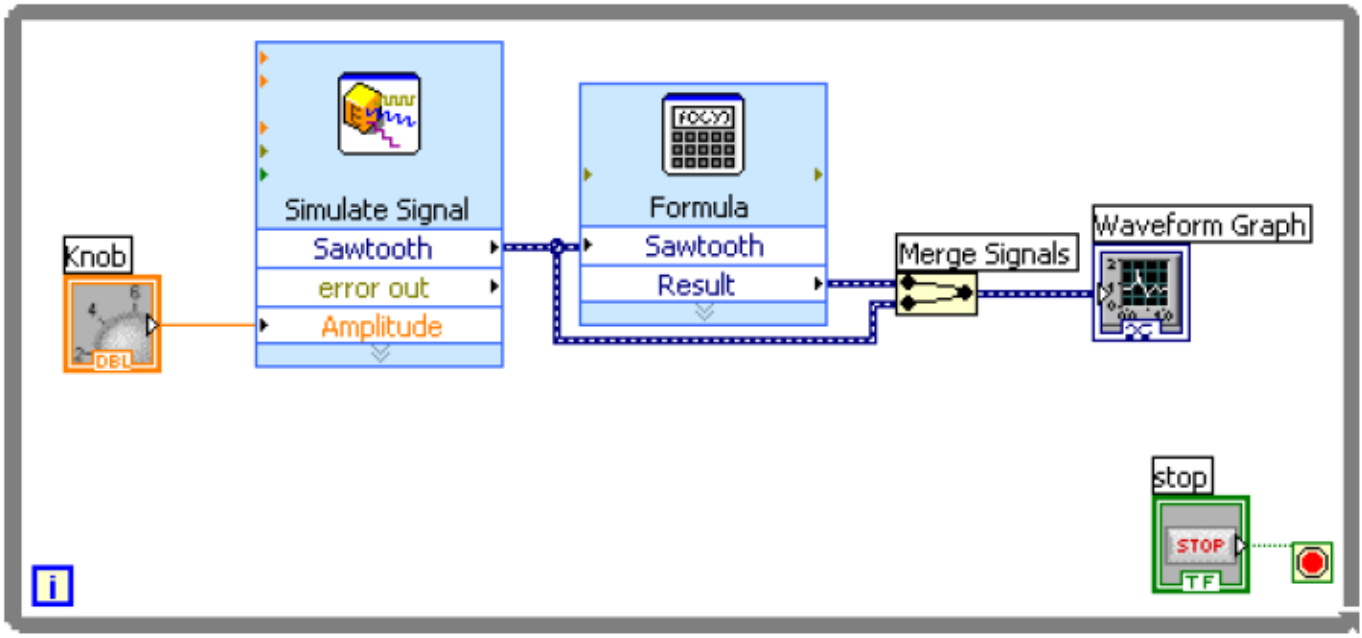


## ➢ Displaying Two Signals on a Graph

To compare the signal generated by the Simulate Signal Express VI and the signal modified by the Formula Express VI on the same graph, use the Merge Signals function. Complete the following steps to display two signals on the same graph.

1. On the block diagram, move the cursor over the arrow on the **Sawtooth** output of the Simulate Signal Express VI.
2. Use the Wiring tool to wire the **Sawtooth** output to the **Waveform Graph** terminal. The Merge Signals function, shown, appears where the two wires connect. A function is a built-in execution element, comparable to an operator, function, or statement in a text-based programming language. The Merge Signals function takes the two separate signals and combines them so that both can display on the same graph.

3. Press the <Ctrl-S> keys or select **File»Save** to save the VI.
4. Return to the front panel, run the VI, and turn the knob control. The graph plots the sawtooth wave and the scaled signal. The maximum value on the y-axis automatically changes to be 10 times the knob value. This scaling occurs because you configured the Formula Express VI to generate a slope of 10.
5. Click the **STOP** button to stop the VI.

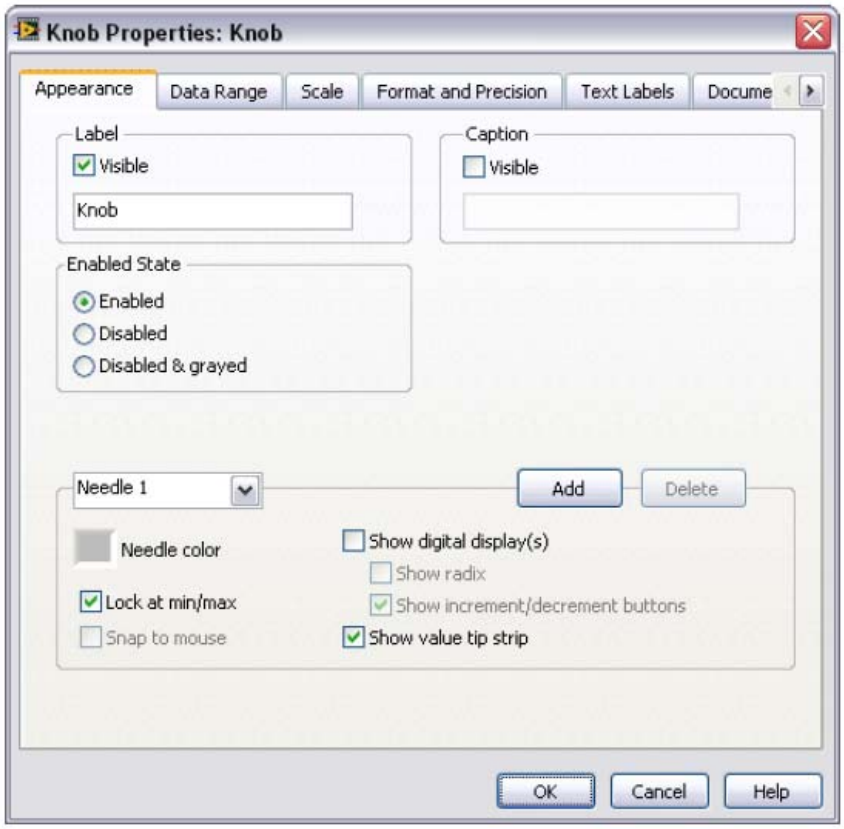



## ➢ Customizing a Knob Control

The knob control changes the amplitude of the sawtooth wave, so labeling it **Amplitude** accurately describes the behavior of the knob. Complete the following steps to customize the appearance of the knob.

1. On the front panel, right-click the knob and select **Properties** from the shortcut menu to display the **Knob Properties** dialog box.

2. In the **Label** section on the **Appearance** page, delete the label **Knob**, and enter Amplitude in the text box.



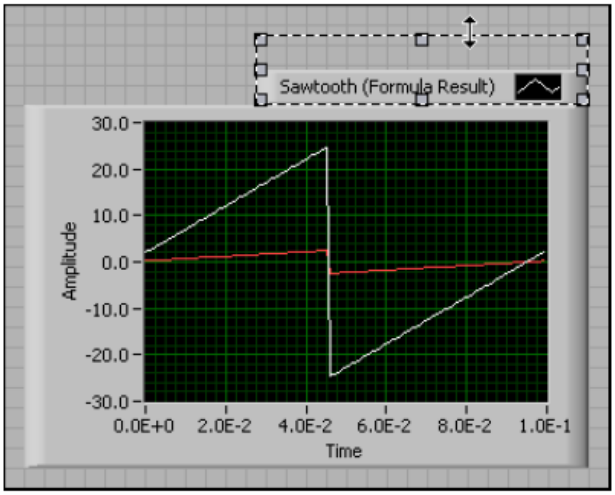

3. Click the **Scale** tab and in the **Scale Style** section, place a checkmark in the **Show color ramp** checkbox. The knob on the front panel updates to reflect these changes.
4. Click the **OK** button to save the current configuration and close the **Knob Properties** dialog box.
5. Save the VI.
6. Reopen the **Knob Properties** dialog box and experiment with other properties of the knob. For example, on the **Scale** page, try changing the colors for the **Marker text color** by clicking the color box.
7. Click the **Cancel** button to avoid applying any changes you made while experimenting. If you want to keep the changes you made, click the **OK** button.

## ➢ Customizing a Waveform Graph

The waveform graph indicator displays the two signals. To indicate which plot is the scaled signal and which is the simulated signal, you can customize the plots. Complete the following steps to customize the appearance of the waveform graph indicator.

1. On the front panel, move the cursor over the top of the plot legend on the waveform graph. Though the graph has two plots, the plot legend displays only one plot.
2. When a double-headed arrow appears, shown in Figure 1-11, click and drag the border of the plot legend to add one item to the legend. When you release the mouse button, the second plot name appears.
3. Right-click the waveform graph and select **Properties** from the shortcut menu to display the **Waveform Graph Properties** dialog box.

4. On the **Plots** page, select **Sawtooth** from the pull-down menu. In the **Colors** section, click the **Line** color box to display the color picker. Select a new line color.
5. Select **Sawtooth (Formula Result)** from the pull-down menu.
6. Place a checkmark in the **Do not use waveform names for plot names** checkbox.
7. In the **Name** text box, delete the current label and change the name of this plot to Scaled Sawtooth.
8. Click the **OK** button to save the current configuration and close the **Waveform Graph Properties** dialog box. The plot color on the front panel changes.
9. Reopen the **Waveform Graph Properties** dialog box and experiment with other properties of the graph. For example, on the **Scales** page, try disabling automatic scaling and changing the minimum and maximum value of the y-axis.
10. Click the **Cancel** button to avoid applying any changes you made while experimenting. If you want to keep the changes you made, click the **OK** button.
11. Save and close the VI.



King Fahd University of Petroleum and Minerals
Systems Engineering Department

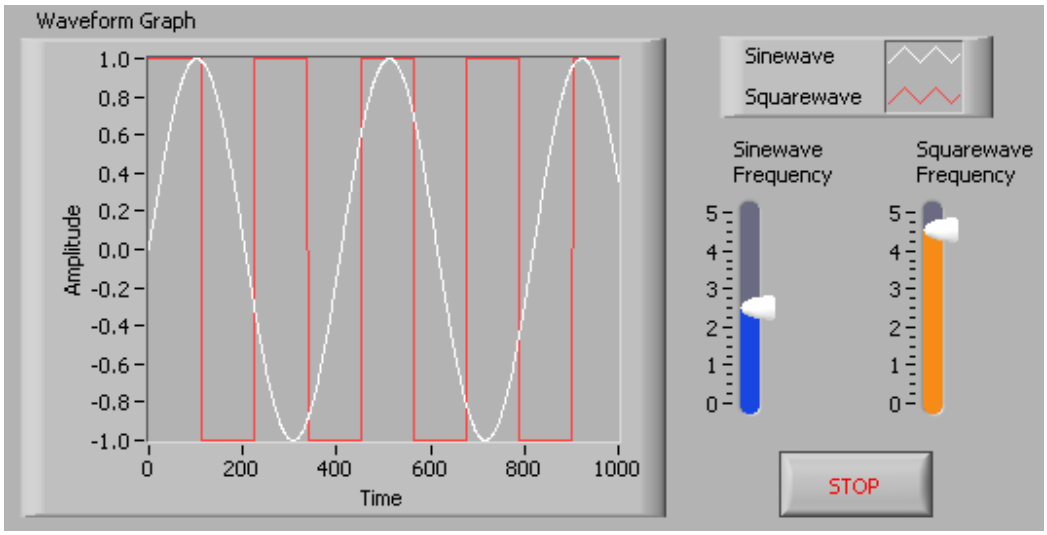
CISE 318
Computer Control Systems

# Lab 3: Exercises in LabVIEW

## Exercise 1 - Using Waveform Graphs

## Front Panel

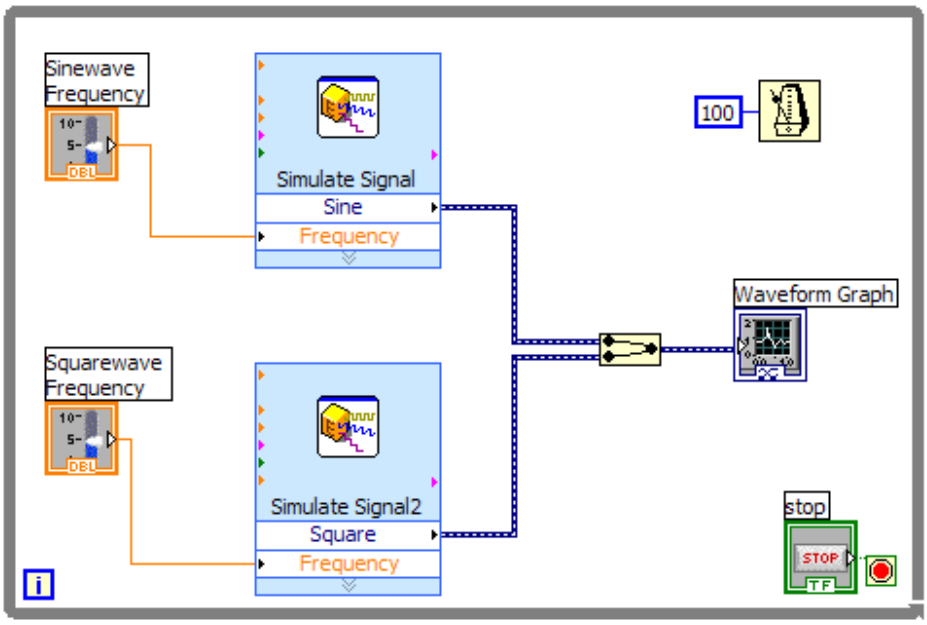
1. Open a new VI and build the following front panel using the following tips.




a. Create a waveform graph indicator from the **Controls»Graph Indicators** palette. Use the position/size/select tool to move the plot legend to the side, and expand it to display two plots. Use the labeling tool to change the plot names and the properties page to choose different colors for your plots.
b. Place a Stop button on the front panel.
c. Place two vertical pointer slides from the **Controls»Numeric Controls** palette. Use the properties page again to change the slide fill color.

## Block Diagram

2. Build the following block diagram.




a. Place a While Loop from **Functions»Execution Control** palette.
b. Place a **Wait Until Next ms Multiple** from **Functions»All Functions »Time & Dialog** and create a constant with a value of 100.
c. Place two Simulate Signal Express VIs from the **Functions»Input** and leave the Signal type as Sine for the first Simulate Signal VI and change the Signal Type to Square for the second VI. Wire both of the outputs into the waveform graph. A Merge Signals function will automatically be inserted.
d. Expand the Simulate Signal Express VIs to show another Input/Output. By default, error out should appear. Change this to Frequency by clicking on error out and choosing **Frequency**.

3. Save the VI as Multiplot Graph.vi.

4. Display the front panel and run the VI.

5. Save and close the VI.

## Exercise 2 - Matching Dice using a While Loop

i. This VI will simulate the rolling of two dice and determining how many rolls it takes to roll matching dice.

ii. Starting with the **Block Diagram.**

1. Create a **While Loop**
   - **Functions Palette** → **Exec Ctrl** → **While Loop** → **Left-Click** and **Hold** on block diagram → **Drag** to create a fairly large box, which will represent your while loop *(be sure to make it large enough to fit several terminals inside)*.
2. Delete the **Stop Button Terminal** that is connected to the **Stop If True Terminal** *(automatically created when creating a while loop)*.
3. Insert the following inside the while loop:

   a. **2 Random Number Generators**
      - **Functions Palette** → **All Functions** → **Arith/Compare** → **Numeric** → **Random Num**

   b. **2 Multiplication Functions**
      - **Functions Palette** → **Arith/Compare** → **Numeric** → **Multiply**

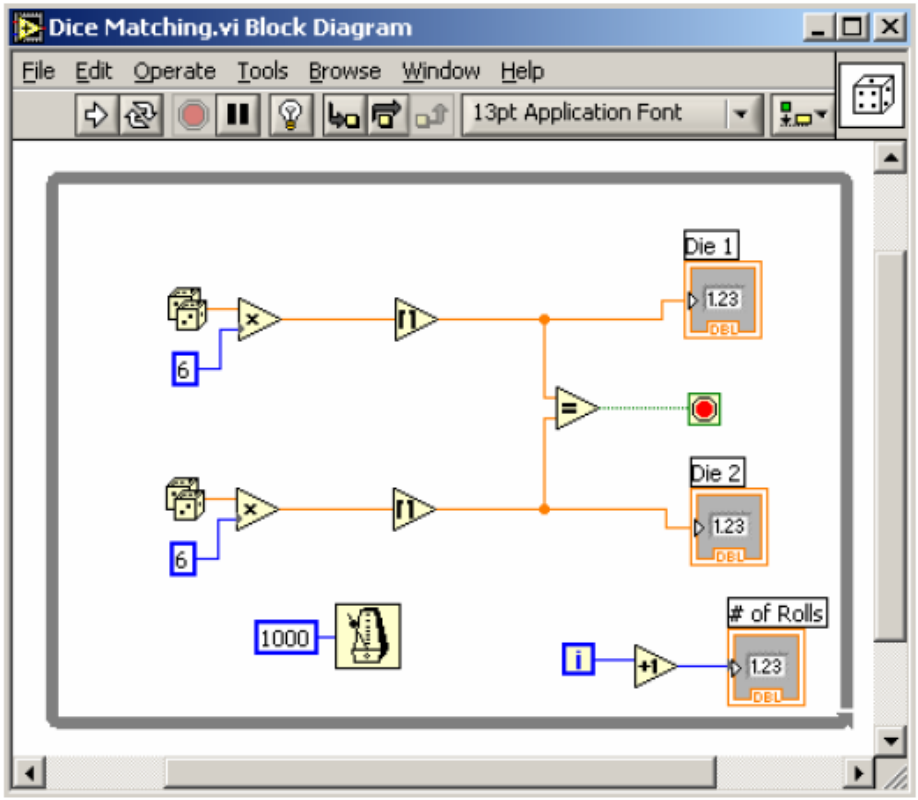
   c. **2 Round To +Inf Functions**
      - **Functions Palette** → **Arith/Compare** → **Numeric** → **Round To +Inf**

   d. An **Equals Function**
      - **Functions Palette** → **Arith/Compare** → **Compare** → **Equal To**

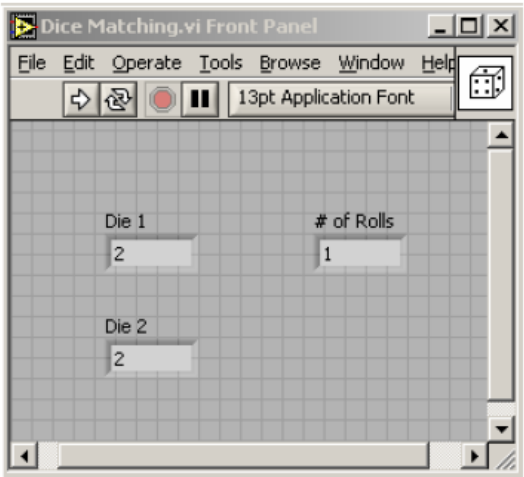
   e. A **Wait Until Next MS Multiple Function**
      - **Functions Palette** → **All-Functions** → **Time & Dialog** → **Wait Until Next MS Multiple**

   f. An **Increment Function**
      - **Functions Palette** → **Arith/Compare** → **Numeric** → **Increment**.

4. Wire and arrange the terminals as shown i ıure 3.2, excluding the numerical indicators (Die 1, Die 2, # of Rolls). You will have to create a few constants as well, which can be done by **Right-Clicking** on the appropriate terminal and selecting **Create → Constant**.




iii. On the **Front Panel**, we need 3 **Numerical Indicators** *(two will show the number on each die and the other will indicate the number of rolls taken)*. Label each output as follows: **"Die 1"**, **"Die 2"**, and **"# of Rolls"**



## Exercise 3 – Using Arrays

What is an array?

- An array can either resemble a vector or a matrix. As does a vector and a matrix, an array groups similar pieces of data.
- An array consists of two different components, the elements (pieces of data) and the dimension (the size of the array).
- Arrays may contain numeric, Boolean, path, string, waveform, and cluster data types. They may be used as an indicator (output) or a control (input).
- The elements in the array are ordered or assigned a certain index. Arrays begin indexing the elements at zero, so the first element will have an index of zero instead of one.

**To create any Array:**

1. **Controls Palette → All-Controls → Array & Cluster → Array**
2. The array you create will have a control on the top-left side that will allow you to navigate through the array. The number in the control box will indicate the index of the element shown in the leftmost cell.
3. Notice when you first put the array on the front panel that it is empty. You can determine your array type by inserting either a control or indicator inside the array. *For example*, for a numerical indicator array:
   - **Controls Palette → Num Inds→ Num Ind → Place inside Array**

**Do the following steps:**

i. This VI will incorporate many of the available array functions all in one program. Many others exist that can be explored using the help tool, which will explain what each function does and what parameters are required.

ii. Beginning with the **Front Panel** of a **New Blank VI**.

1. Create **5 Arrays**

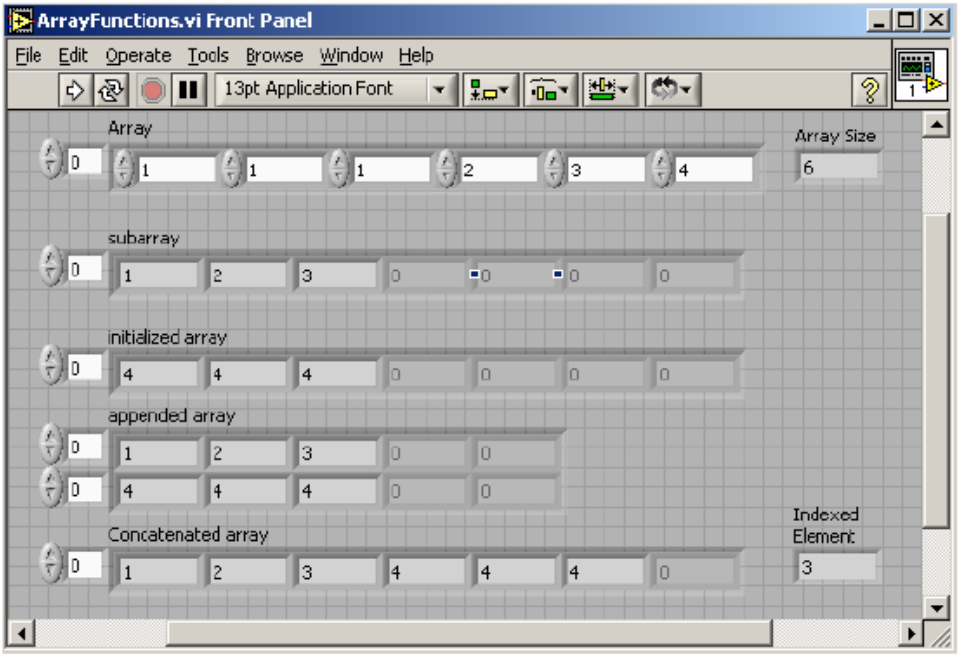
   a. The **First Array** will be a **Numerical Control Array**, the **Second**, **Third** and **Fifth** will be **Numerical Indicator Arrays**, and the **Forth** will be a **2-D Numerical Indicator Array**.

   b. Name the **Arrays** as follows: "**Array**", "**Subarray**", "**Initialized Array**", "**Appended Array**", and "**Concatenated Array**"

2. Create **2 Numerical Indicators**

   a. Name the **Numerical Indicators** as follows: "**Array Size**" and "**Indexed Element**"

   b. Arrange your **Front Panel** similar to the one shown in Figure




iii. Switch to the **Block Diagram**

1. Using the **Array Size Function** 
   - **Functions Palette → All-Functions → Arrays and Clusters → Array Size**
   - Wire **"Array" Terminal → Array Size Function → "Array Size" Numerical Indicator**

2. Using the **Array Subset Function** 
   - **Functions Palette → All-Functions → Arrays and Clusters → Array Subset**
   - Wire **"Array" Terminal → Array Subset Function → "Subarray" Array Indicator**

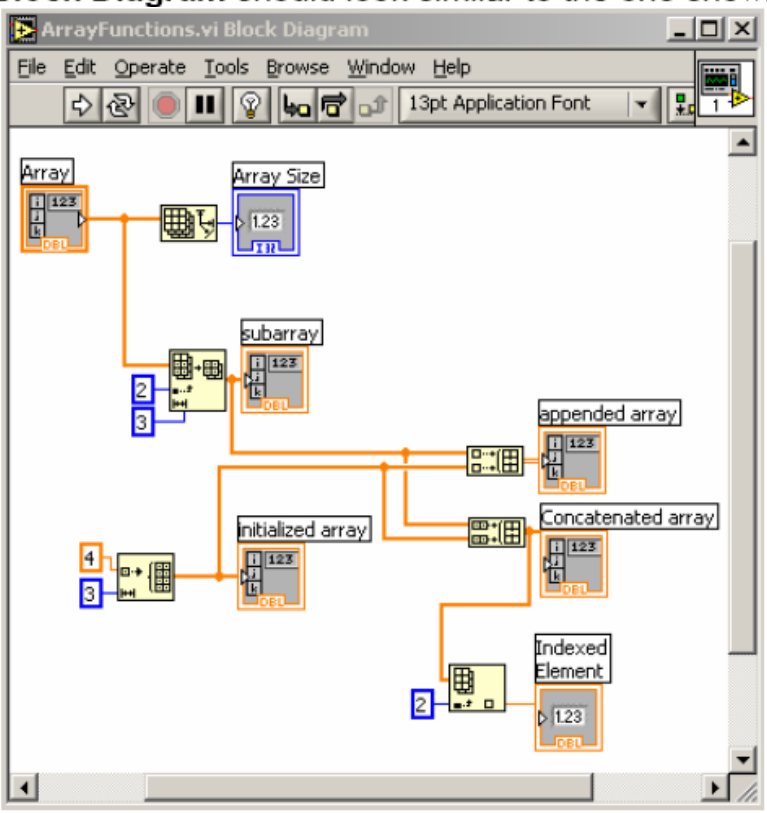
3. Using the **Initialize Array Function** 
   - **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Initialize Array**
   - Wire **Numerical Constant** of **4** → **Element Terminal** of **Initialize Array Function**
   - Wire **Numerical Constant** of **3** → **Length Terminal** of **Initialize Array Function**
   - Wire **Initialize Array Function** → "**Initialized Array**" **Array Indicator**

4. Using the **Build Array Function** 

   a. The default for this function is to **Append** to the original array.
      - **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Build Array**
      - Wire **"Array" Terminal** → **Top Terminal** of **Append/Build Array Function**
      - Branch a wire from the **Output** of the **Initialize Array Function** → **Bottom Terminal** of **Append/Build Array Function**
      - Wire **Output** of **Append/Build Array Function** → "**Appended Array**" **Array Indicator**

   b. The **Build Array Function** can also be used to **Concatenate** to the original array.
      - **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Build Array** → **Right-Click** on **Build Array Function** → **Concatenate**
      - Wire **"Array" Terminal** → **Top Terminal** of **Concatenate/Build Array Function**
      - Branch a wire from the **Output** of the **Initialize Array Function** → **Bottom Terminal** of **Concatenate/Build Array Function**
      - Wire **Output** of **Concatenate/Build Array Function** → "**Concatenated Array**" **Array Indicator**

5. Using the **Index Array Function** 
   - **Functions Palette** → **All-Functions** → **Arrays and Clusters** → **Index Array**
   - **Branch** a wire from the **Output** of the **Concatenated/Build Array Function** → **Top Terminal** of **Index Array Function**
   - Wire **Numerical Constant** of **2** → **Bottom Terminal** of **Index Array Function**
   - Wire **Output** of **Index Function** to **"Indexed Element" Numerical Indicator Terminal**
6. Your **Block Diagram** should look similar to the one shown in Figure




iv. Switch to the **Front Panel**

1. Save as "**ArrayFunctions.VI**"
2. Initialize the values in the "**Array**" **Control Array** to **1**, **1**, **1**, **2**, **3**, and **4** and then, **Run**.
3. Observe the outputs and investigate how the functions work.

King Fahd University of Petroleum and Minerals
Systems Engineering Department

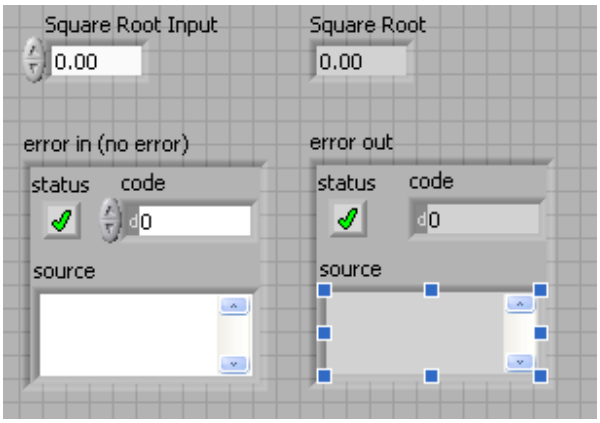
CISE 318
Computer Control Systems

# Lab 4: Exercises in LabVIEW

## Exercise 1 - Error Clusters & Handling

## Front Panel

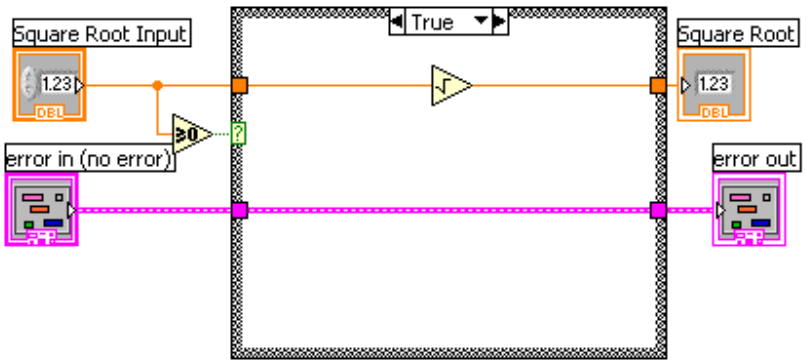
1. Open a new VI and build the following front panel using the following tips.




a. Create a numeric control and change the Label to Square Root Input. Create a numeric indicator for Square Root.
b. Place Error In 3D.ctl from **Controls»All Controls»Arrays & Clusters.**
c. Place Error Out 3D.ctl from **Controls» All Controls»Arrays & Clusters.**

## Block Diagram

2. Build the following block diagram.



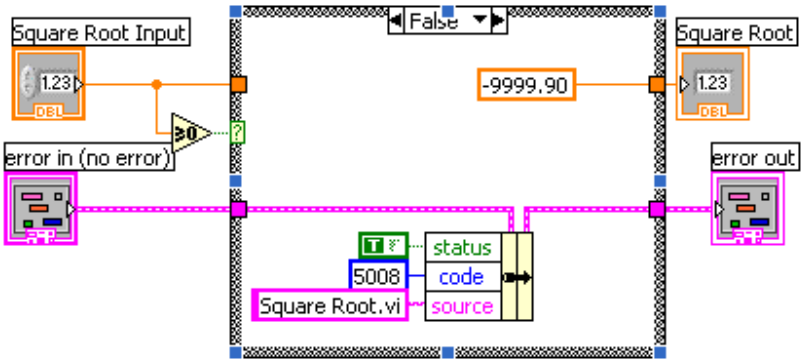

a. Place a Case Structure from **Functions»Execution Control** palette.

b. Place a **Greater or Equal to 0?** from the **Functions»Arithmetic and Comparison»Comparison** palette and wire it to the condition terminal of the case structure.

In the True Case:

c. Place the Square Root function from **Functions»Arithmetic and Comparison»Numeric** palette.

In the False Case:




d. Create a numeric constant from **Functions»Arithmetic and Comparison»Numeric** palette and type -9999.90.

e. Place the Bundle By Name from **Functions»All Functions»Arrays & Clusters** palette. Wire from Error in to the center terminal of Bundle by Name to make status show up. Create constants. Wire from the Error Out indicator to the output of Bundle By name.

3. Save the VI as Square Root.vi.
4. Display the front panel and run the VI.
5. Save and close the VI.

## End of Exercise

## Exercise 2- Simple State Machine

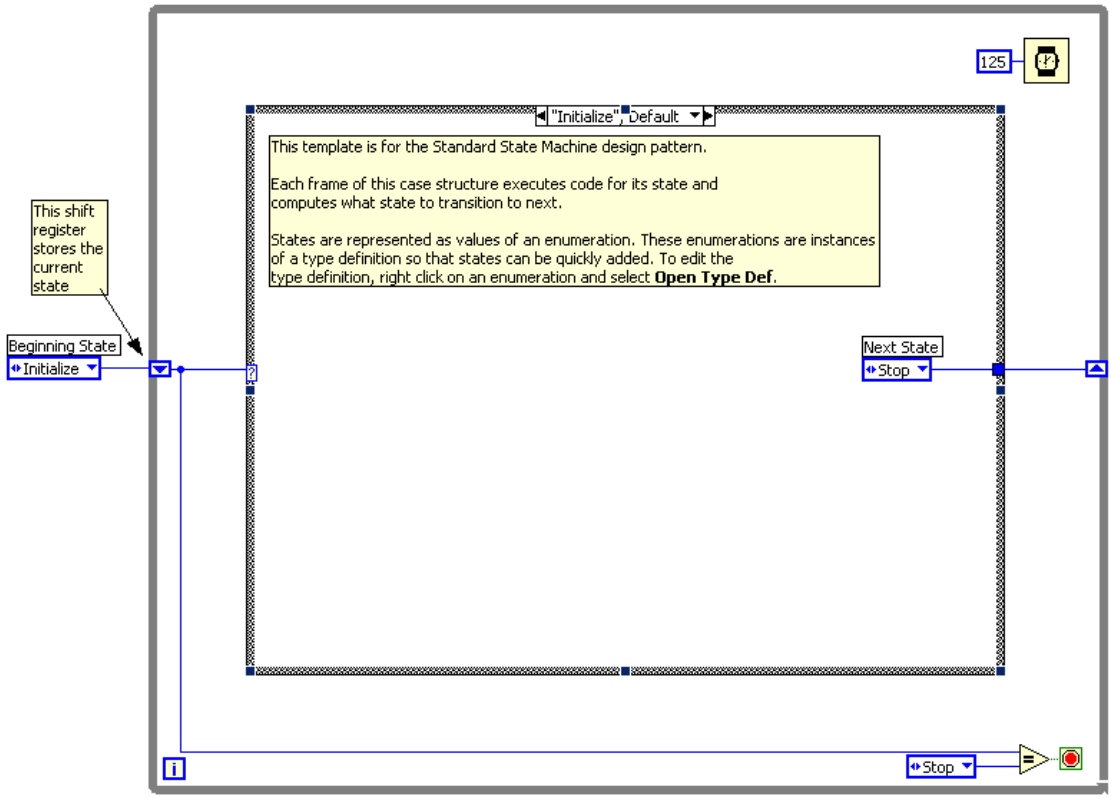
Create a VI using state machine architecture that simulates a simple test sequence. The VI will have an initial state, where it will display a pop-up message indicating that it is starting the test. Then it will proceed to the next case and then to the final state where it will ask the user whether to start over or end the test.

## Front Panel

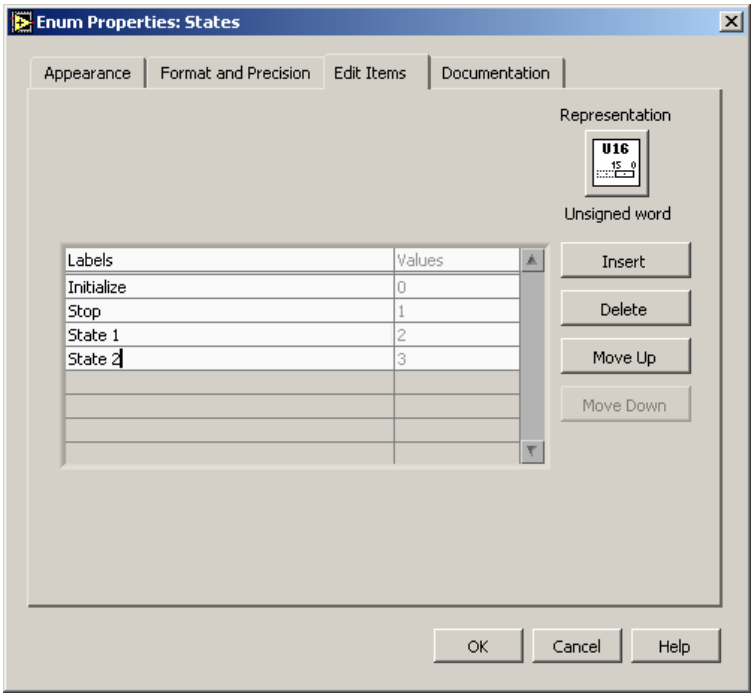
Rather than start from scratch, we will use a VI template to create our state machine.

1. From the initial LabVIEW screen click on **New…**, and choose Standard State Machine, which is located under the **VI from Template » Frameworks » Design Patterns** heading.
2. Examine the template, and then save it in another directory before you begin working on it.

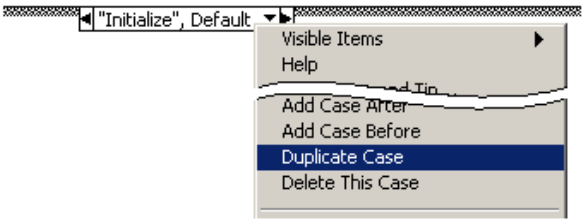
## Block Diagram



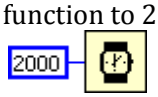

3. Right click on the enum constant labeled Next State and select Open Type Def.
4. On the front panel of the StateMachinesStates.ctl Type Def VI, right click on the States enum control and choose **Edit Items**.
5. Add two more states. Call them “State 1” and “State 2”



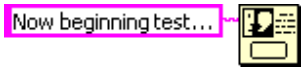

6. Close the State Machines.ctl Type Def Front panel and save the control with the default name when prompted.
7. Right click on the Case Selector Label of the case structure and choose Duplicate case. Do this one additional time so that there are four cases: Initialize, State 1, State 2, and Stop.




8. Change the value connected to the Wait function to 2000.

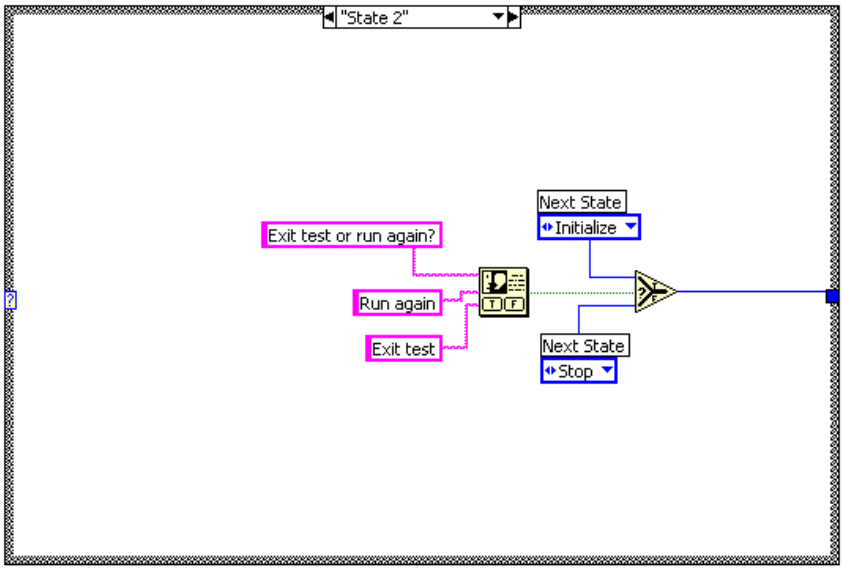



9. Right click on the shift register on the left side of the while loop and create an indicator. Change it's name to "Current State".
10. In the "Initialize", Default case place a One Button Dialog function and wire a string constant into the **Message** input. Type "Now beginning test…" into the string constant.




11. Change the enum constant labled Next State to "State 1".
12. Change to the next state in the case structure ("State 1") and change the enum constant labled Next State to "State 2".
13. Change to the next state ("State2") and add the following code.
    a. Place a Select function and connect two enum constants
    *(Tip: Copy the enum constants from one of the previous cases)*

b. Place a Two Button Dialog and wire create the constants as illustrated below.




14. Run the VI.
15. Save and close the VI.

## End of Exercise

King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 5: Data Acquisition

- ## Data Acquisition

The NI USB-6009 multifunction I/O is a device that can be used to connect LabVIEW to external devices for Data Acquisition. The NI USB-6009 is a USB based data acquisition (DAQ) and control device with analog input and output and digital input and output.

The main features of NI USB-6009 are as follows:

- **Analog input (AI):** 8 inputs with referenced single ended signal coupling or 4 inputs with differential signal coupling. Software-configurable voltage ranges: ±20V, ±10V, ±5V, ±4V, ±2.5V, ±2V, ±1.25V, ±1V. Max sampling rate is 48kS/s (48000 samples per second). 14 bits AD converter (USB-6008: 12 bits).
- **Analog output (AO):** 2 outputs. Voltage range is 0 - 5V (fixed). Output rate is 150Hz (samples/second). 12 bits DA converter.
- **Digital input (DI) and digital output (DO):** 12 channels which can be used as either DI or DO (configured individually). These 12 channels are organized in ports, with Port 0 having lines 0, .., 7, and Port 1 having lines 0, .., 3. Input low is between -0.3V and +0.8V. Input high is between 2.0V and +5.8V. Output low is below 0.8V. Output high is above 2V (with open-drain and push-pull as options).
- **Counter: 32 bits:** Counting on falling edge.
- **On-board voltage sources** (available at individual terminals): 2.5V and 5.0V
- **Power:** USB-6009 is powered via the USB cable.
- **Configuring and testing:** USB-6009 can be configured and tested using MAX (Measurement and Automation Explorer) 4.0, which is installed with NI-DAQ 8.0.
- **Application software:** LabVIEW, C, or Visual Studio.
- **Platforms**: Windows, Mac, Linux.

## *Installing the DAQ*

After inserting the CD, the NI-DAQmx 7.*x* installer should open automatically. If not, select **Start»Run**. Enter *x*:\setup.exe, where *x* is the letter of the CD drive. Complete the instructions in the installer, including rebooting the computer if necessary. For troubleshooting instructions, refer to the Hardware Installation/Configuration Troubleshooter at ni.com/support/install.

**Note:** Install your driver software *before* installing new hardware devices, so Windows can detect your device.

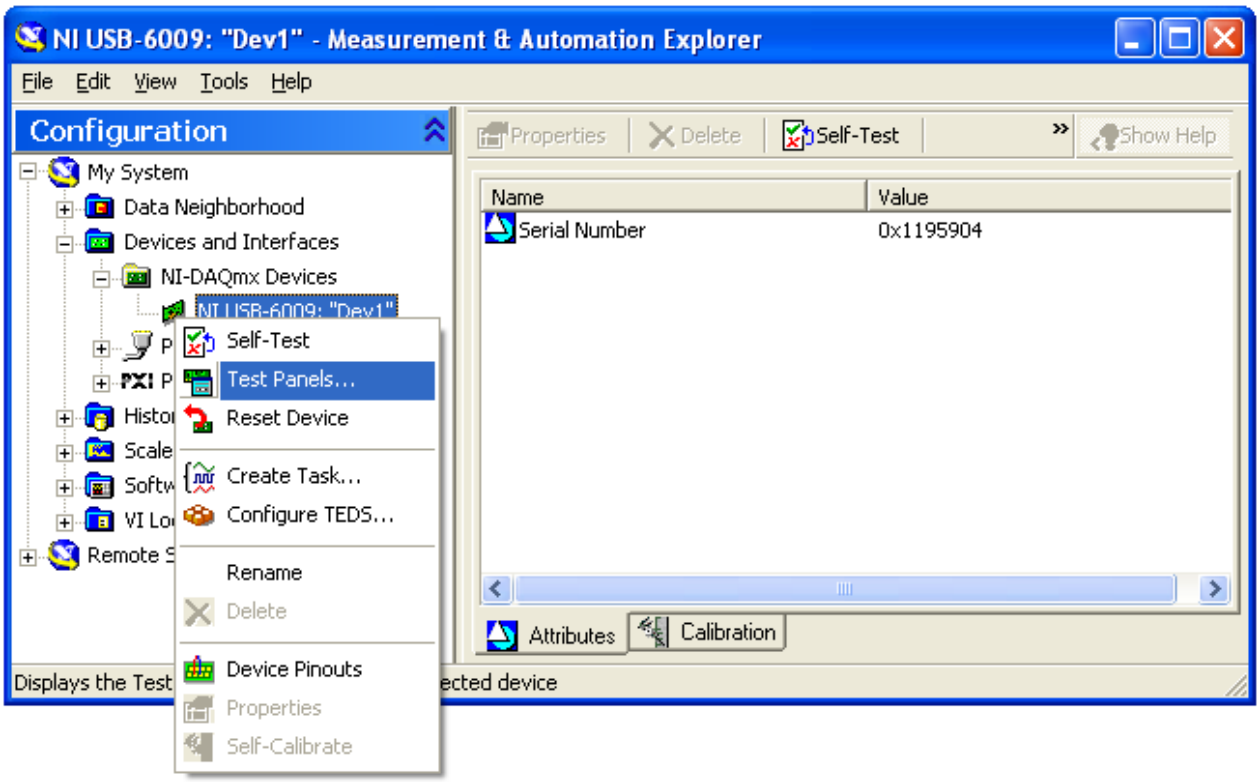
After installing the software, the actual device should be installed. This is done quite easily by simply plugging in the USB cable into PC and device, and then click **Next** on any dialog that appears. Next click **Finish**.

**Note USB-6008/6009 Only**—If this is the first time that a USB-6008/6009 device is installed on your computer, you might be prompted to install a USB-6*xxx* Firmware Loader. Please complete the Windows Hardware Wizard prompts to completely install this device.

**Device Confirmation**

Complete the following steps:

1. Double-click the **Measurement & Automation** icon on the desktop to open MAX.
2. Expand **Devices and Interfaces**, then expand **NI-DAQmx Devices**.

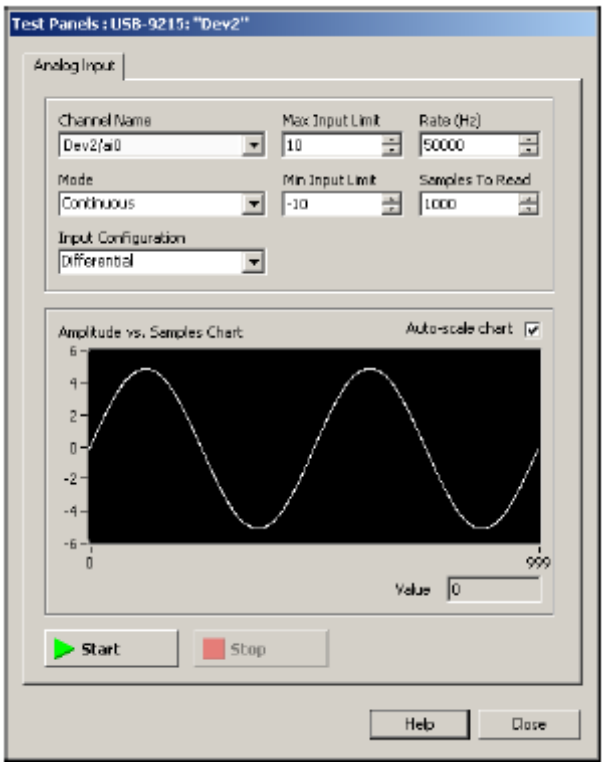



3. Check that your device appears under Devices and Interfaces. If your device does not appear, press **<F5>** to refresh the view in MAX. If the device is still not recognized, refer to ni.com/support/install for troubleshooting information.
4. Right-click your device and select **Self-Test**.

If you need help during the self-test, select **Help»HelpTopics»NI-DAQmx** and click *Max Help for NI-DAQmx*. When the self-test finishes, a message indicates successful verification or if an error occurred.

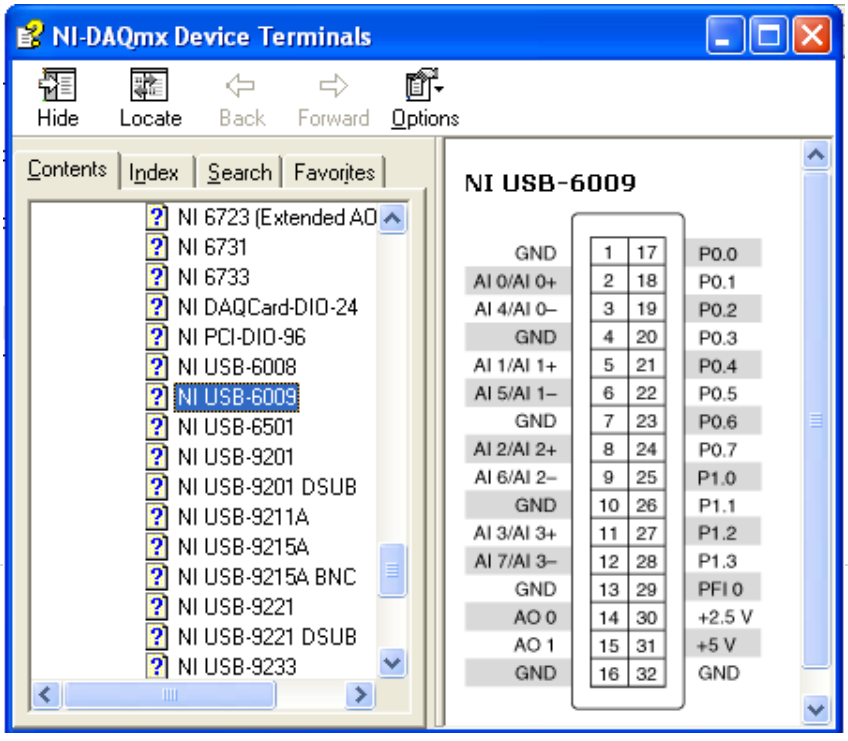
You can also use a test panel for testing specific device functionality, such as the ability to acquire and generate signals. It can be done as follows:

1. In MAX, expand **Devices and Interfaces»NI-DAQmx Devices**.
2. Right-click the device to test.
3. Select **Test Panels** to open a test panel for the selected device.




4. Click **Start** to test different functions of the device. Click **Help** for instructions on operating the test panels.
5. The test panel displays a message indicating whether an error occurred.

To see the terminals of the USB-6009, select **Device Pinouts** in the menu shown below. The terminals or pins are shown in the NI-DAQmx Device Terminals window.



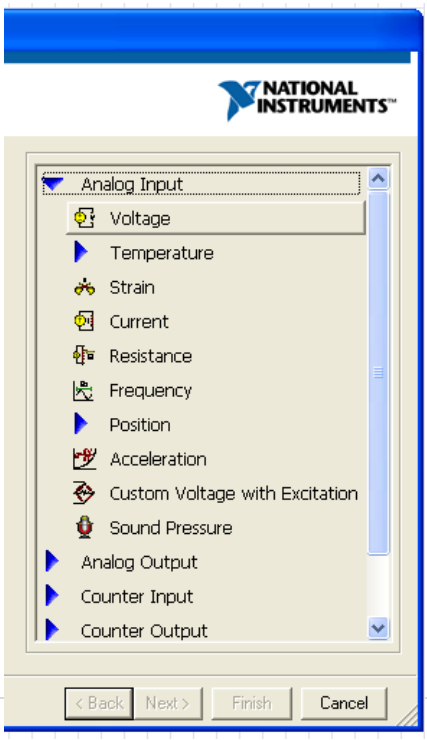

The DAQ is now ready to connect external devices to LabVIEW.

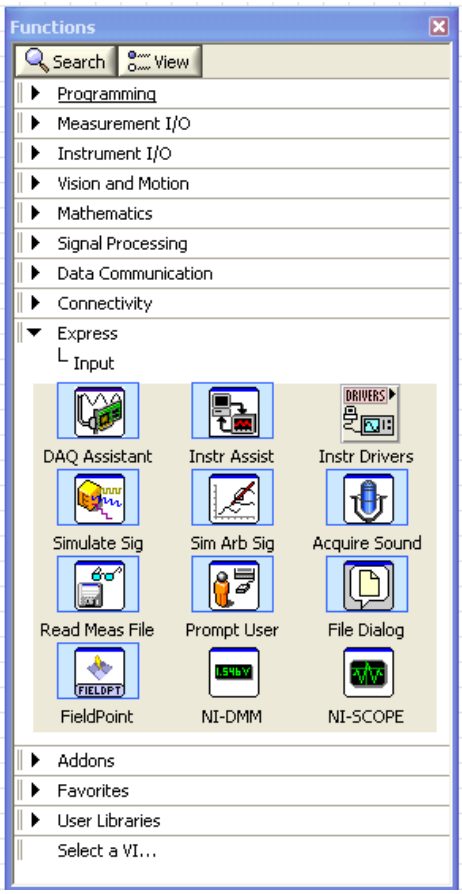
## Exersise :

**Reading a analog signal from a signal generator and displaying in Labview.**

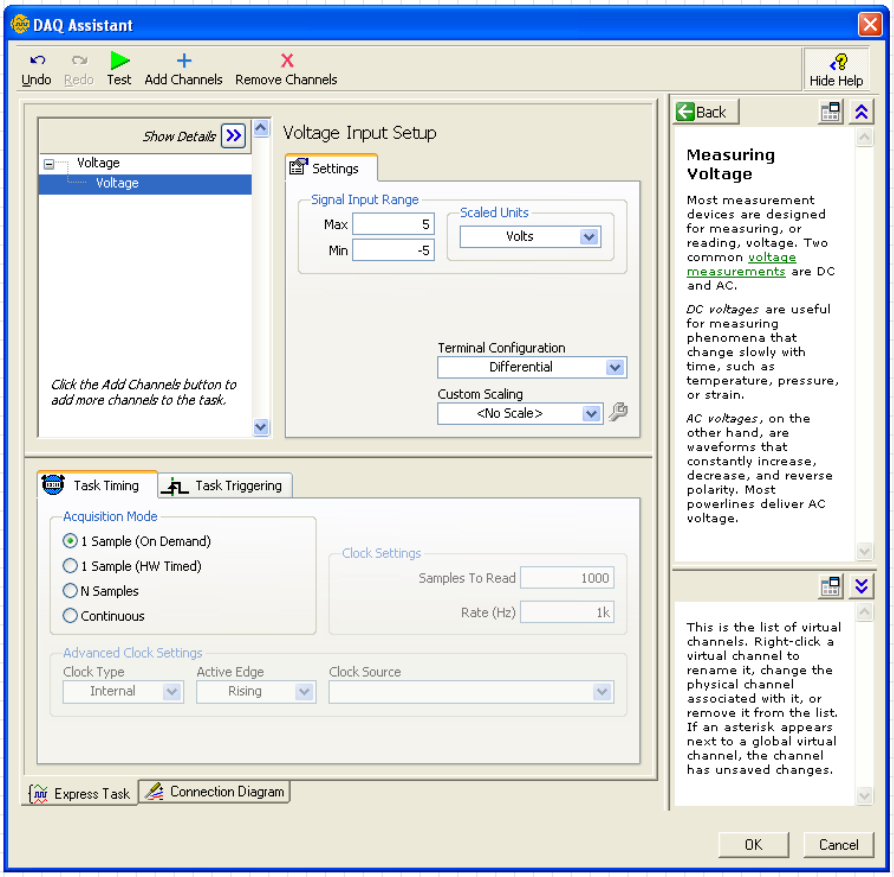
The purpose of this exersice is to teach you how to build a LabVIEW application to retrieve voltage input values using the NI USB-6009 data acquisition device.

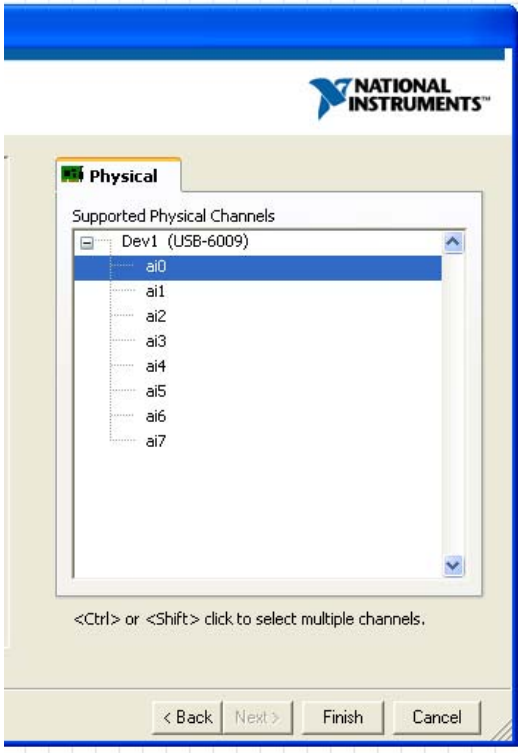
1. Click “New VI” button to create a new blank LabVIEW program
2. Select Front Panel and enable Controls Palette to choose a “Waveform Chart” indicator in “Graph” group, add to front panel and name it “Voltage”.
3. Switch to Block Diagram to verify that a new data terminal was created (named “Voltage”).
4. In the Block diagram, open the Functions Palette to get the DAQ Assistant. The path goes as Functions>>Express>>Input>>DAQ Assistant.
   Note: If you have any trouble in finding a block use the Search function available just below the Title Bar of the Functions Palette.
5. When the DAQ Assistant is placed in the Block diagram window, a window with “Cretae New…” in the title bar appears. Select Analog Input and then Voltage.



6. The Physical Tab will appear showing the devices attached and the corresponding pins. Select the required pin and click Finish.
7. Another DAQ Assistant window will appear. Under the Voltage Input setup, Settings Tab select the appropriate maximum & minimum input signal range and units.(preferably, -5 to 5 volts). Also, in the Task Timing Tab under Acquisition mode select 1 Sample (On Demand).Click OK.

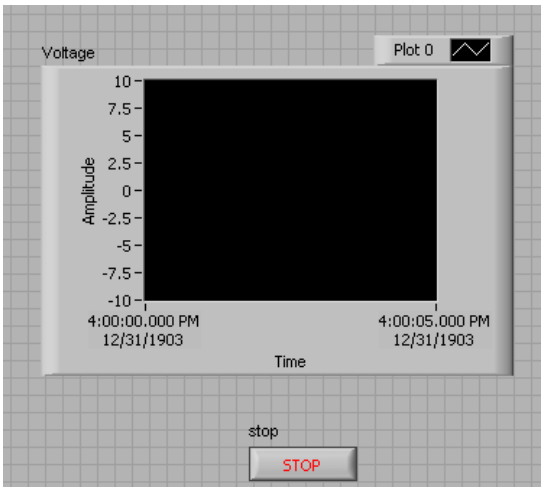


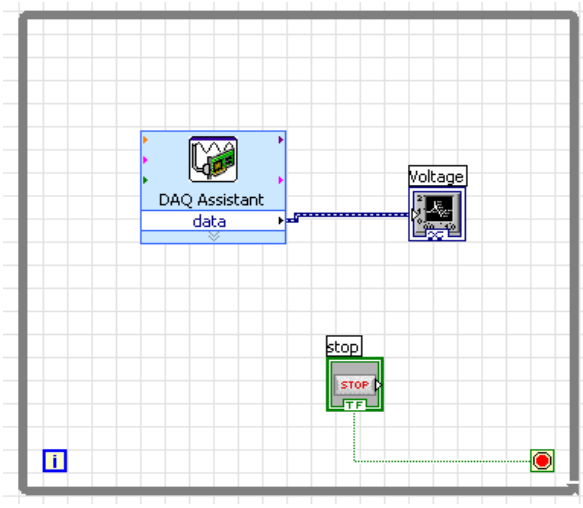




8. Connect the data output from the DAQ Assistant to the Waveform chart terminal, using the wiring tool. If the wiring tool does not appear automatically on moving the cursor over the Data output of DAQ, go to View>>Tools Palette and select the wiring tool. To make wiring easier, you can invoke the Help window to support it. The Help window shows input and output parameters of each selected VI. (The way to display the Help window is to click on the Help menu selection and select the "Show Context Help" option)
9. You can change the window to the Front Panel window and click the "Run" (or "Run Continuously") button to execute the VI.
10. If your waveform chart does not show the whole graph, you can change the scale of the Yaxis. To change the Y axis scale on the waveform chart, move the cursor to the graphic origin, click the left mouse button, and type in the desired value
11. The "Run Continuously" button executes the VI continuously. You can also add a "While Loop" to enhance your program. First, move mouse to the Block Diagram. Open the Functions Palette and select the Structures group and then the While Loop. Add the While Loop to the Block Diagram

and enlarge it to include the DAQ Assistant and the Voltage Waveform data terminal. On the front panel add a Stop button (from Controls>>Boolean>>Stop Button) to control the execution of the While Loop. In the block diagram, wire the stop button of the front panel to the stop button of the while loop.

12. After wiring, switch to Front Panel and press the RUN button to execute the VI.




**Make sure that all your external wiring is connected properly and all devices have a common ground.**

# Exersise :

**Sending an analog signal from the NI USB 6009 and displaying in a Oscilloscope.**

The purpose of this exersice is to teach you how to build a LabVIEW application to send voltage outputs values from NI USB 6009 data acquisition cards.

1. Click “New VI” button to create a new blank LabVIEW program
2. Select Front Panel and enable Controls Palette to choose a “Waveform Chart” indicator in “Graph” group, add to front panel and name it “Voltage”.
3. Switch to Block Diagram to verify that a new data terminal was created (named “Voltage”).
4. In the Block diagram, open the Functions Palette to get the DAQ Assistant. The path goes as Functions>>Express>>Input>> DAQ Assistant.
   Note: If you have any trouble in finding a block use the Search function available just below the Title Bar of the Functions Palette.
5. When the DAQ Assistant is placed in the Block diagram window, a window with “Cretae New…” in the title bar appears. Select Analog Output and then Voltage.
6. The Physical Tab will appear showing the devices attached and the corresponding pins. Select the required pin and click Finish.
7. Another DAQ Assistant window will appear. Under the Voltage Output setup, Settings Tab select the appropriate maximum & minimum input signal range and units.(preferably, 0 to 10 volts). Also, in the Task Timing Tab under Acquisition mode select 1 Sample (On Demand).Click OK.
8. Generate a sinusoidal signal using the express VI using the “Simulate sig” express VI. The path to access these **“Functions” >> Express>> “Inputs” >> “Simulate sig”.** Select a type of signal, its

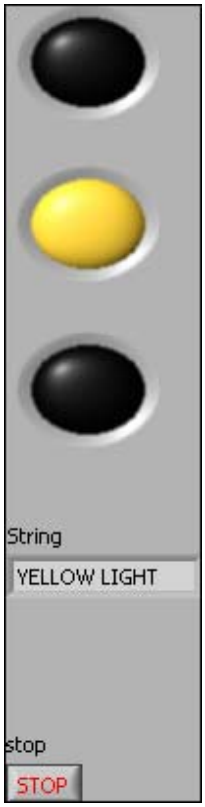
amplitude and frequency. Select a number of "samples per second" as 1000 and uncheck the automatic for number of samples and select **"1"** as number of samples. Now connect the output of "**Simulate sig"** to the voltage input of the DAQ Assistant.

9. Also connect the Voltage waveform to the output of "**Simulate sig"**.
10. Complete your wiring and run to see the result in your graph as well as the oscilloscope.
11. The "Run Continuously" button executes the VI continuously. You can also add a "While Loop" to enhance your program. First, move mouse to the Block Diagram. Open the Functions Palette and select the Structures group and then the While Loop. Add the While Loop to the Block Diagram and enlarge it to include the DAQ Assistant and the Voltage Waveform data terminal. On the front panel add a Stop button (from Controls>>Boolean>>Stop Button) to control the execution of the While Loop. In the block diagram, wire the stop button of the front panel to the stop button of the while loop.
12. After wiring, switch to Front Panel and press the RUN button to execute the VI.

**Make sure that all your external wiring is connected properly and all devices have a common ground.**

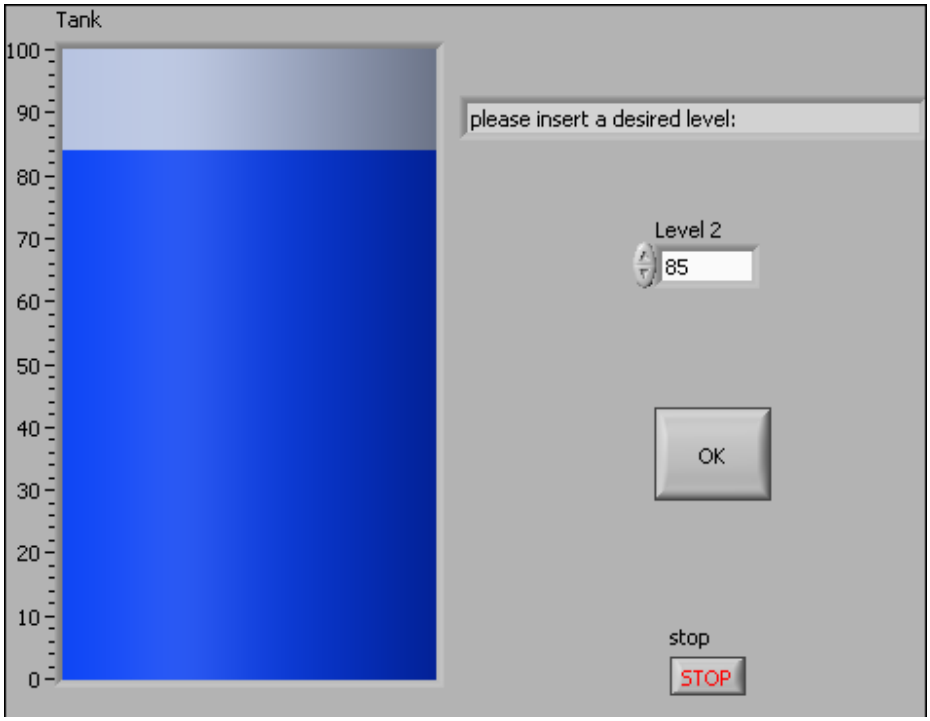
## Exercise: Traffic Light

Construct a VI that will display a traffic light in the Front Panel. See figure below. Light transitions are timed. Let red take 2s, yellow 1s and green 3s. Try to utilize your knowledge of LabVIEW in constructing this VI.



## Exercise: Filling and emptying a Tank

Build a VI that display a Front Panel similar to which shown below. The front panel should have a Numeric Control to set a desired tank level. The filling and emptying tasks should be displayed gradually. Your VI should decide by itself if it is to fill or to empty the tank. An OK button is to start the task.



King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 6: Digital Input / Output

Digital signals are digital representations of discrete-time signals, which are often derived from analog signals. An analog signal is a datum that changes over time—say, the temperature at a given location; the depth of a certain point in a pond; or the amplitude of the voltage at some node in a circuit—that can be represented as a mathematical function, with time as the free variable (abscissa) and the signal itself as the dependent variable (ordinate). A discrete-time signal is a sampled version of an analog signal: the value of the datum is noted at fixed intervals (for example, every microsecond) rather than continuously.

If individual time values of the discrete-time signal, instead of being measured precisely (which would require an infinite number of digits), are approximated to a certain precision—which, therefore, only requires a specific number of digits—then the resultant data stream is termed a digital signal. The process of approximating the precise value within a fixed number of digits, or bits, is called quantization. In conceptual summary, a digital signal is a quantized discrete-time signal; a discrete-time signal is a sampled analog signal.

In most applications, digital signals are represented as binary numbers, so their precision of quantization is measured in bits. Suppose, for example, that we wish to measure a signal to two significant decimal digits. Since seven bits, or binary digits, can record 128 discrete values (*viz.*, from 0 to 127), those seven bits are more than sufficient to express a range of one hundred values.

**Exercise: Sending and receiving digital signals**

The purpose of this exercise is to teach you how to build a LabVIEW application to send and receive digital values from NI USB 6009 data acquisition cards.

- **Part 1: Sending**
  1. Click "New VI" button to create a new blank LabVIEW program
  2. Select Front Panel and enable Controls Palette to choose a "Push Button" indicator in "Boolean" group (you can also use a toggle or slide switch, etc. as well), add to front panel and name it "Status".
  3. Switch to Block Diagram to verify that a new data terminal was created (named "Status").
  4. In the Block diagram, open the Functions Palette to get the DAQ Assistant. The path goes as *Functions>>Express>>Input>> DAQ Assistant.*
  5. When the DAQ Assistant is placed in the Block diagram window, a window with "Create New…" in the title bar appears. Select Digital I/O and then Line Output.
  6. The Physical Tab will appear showing the devices attached and the corresponding pins. Select the required pin(s) and click Finish.
  7. Another DAQ Assistant window will appear. Under the Task Timing Tab under Acquisition mode select 1 Sample (On Demand).Click OK.

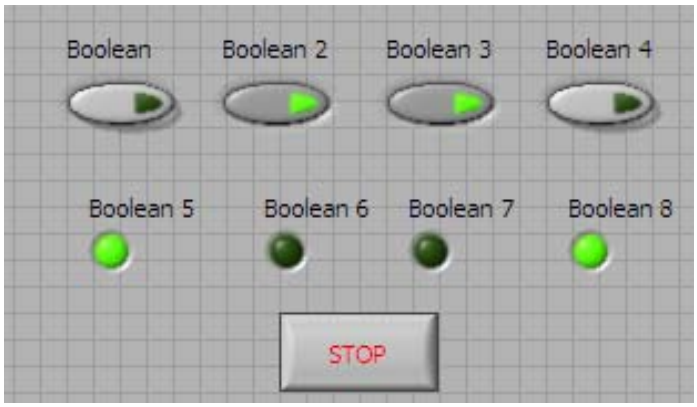
8. If you want to send more than 1 signal at a time, then first put more Boolean switches/buttons on the front panel. In the Block diagram, connect all the Boolean terminals into one using *"Functions" >> Programming>> "Arrays" >> "Build arrays".* Extend the size of the array according to the number of signals to send.
9. Next connect the array output to the data pin of the DAQ assistant. Note: Make sure that the correct pin numbers are assigned to the Boolean switches in the DAQ Assistant.
10. You can also add LEDs (from *Controls>>Boolean*) and connect them to the switches in the block diagram if you want to view the result on the front panel as well.
11. The "Run Continuously" button executes the VI continuously. You can also add a "While Loop" to enhance your program.
12. After wiring, switch to Front Panel and press the RUN button to execute the VI.

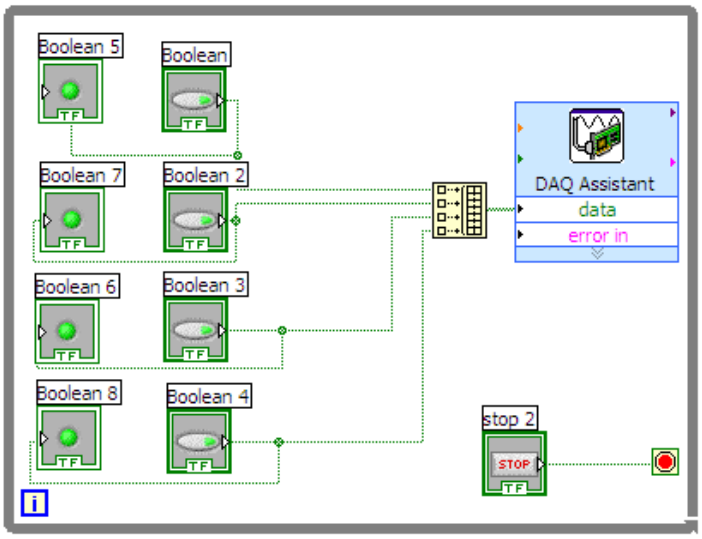
- **Part 2: Receiving**

For testing Digital input,

- Select *Digital I/O>>Line Input* in the DAQ assistant
- And instead of Boolean switches use LEDs.
- Also Index array can be used.

**Note**: In hard wire, real LEDs and switches can be used for demonstration. Make sure that all your external wiring is connected properly and all devices have a common ground.



King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 7: Two Tank Experimental System

## • Interfacing the Two tank


In this exercise, two-tank system is introduced. The two tank system is as shown in the figure. It has a water pump which takes in 0..10 voltages and pumps in water with speeds depending at the voltage supplied. Two outputs which are the speed of flow and the level of water in the tank are shown visually. There are speed and level sensors that provide voltages between 0 and 10 voltages to indicate speed and voltage. The yellow filled area shows the flow of water from pump to tank to reservoir through valves. The flow from tank to reservoir can be controlled using the value. At "0" indicator the valve is "fully closed" and at "5" it is "fully opened".

This exercise will interface the tanks output to analog inputs to measure the tank level and speed of flow and use the analog voltage output of the USB 6009 to voltage input of the tank to run the pump motor. Use your knowledge of previous experiments to send a constant voltage out of the USB card and receive the two analog signals.

To conduct this experiment, we will have to first connect the 2-Tank system to LabVIEW through the NI DAQ card. The steps are as follows:

1. Connect the sensor of the tank system (top-most pin) to any Analog Input (AI) pin of the DAQ card.
2. Next connect the motor (2nd last/above ground) to an Analog Output (AO) pin.
3. Connect the ground of the tank (bottom most) pin to a ground of the DAQ.

Now the hardware wire connections are complete and we can start building the VI:

1. Use two sets of **"DAQ Assistant"** to capture the analog signal of level at the channel. And use one **"DAQ Assistant"** to send a signal from USB 6009 to the tank.
2. Use a **Knob** to select the voltage being sent from the USB 6009 to Tank. "**Knob**" can be found at **"Controls" >> "Num Ctrls" >> "Knob"**
3. Use the tank level indicator from **"Controls" >> "Numerical Ind" >> "Tank"** to display the output of tank level.
4. Use the "**Flat sequence**" in the block diagram from "**Functions**"**>> "Structures" >> "Flat Sequence"** to send the analog out signal first from computer to two tank and then read the two analog inputs signals from two tank to computer.
5. Add frames on the flat sequence by right clicking on the border of the flat sequence and selecting "Add frame after" from the menu.
6. Connect all the wiring and use a **while** loop and **stop** button to run the VI.

- **On-Off Control**

In this experiment we will implement a simple control strategy, namely On-Off Control. This strategy is based on a simple condition by which the maximum control signal is given if the output has not reached the desired set point. When the output reaches the desired value a zero control signal is sent to the plant.

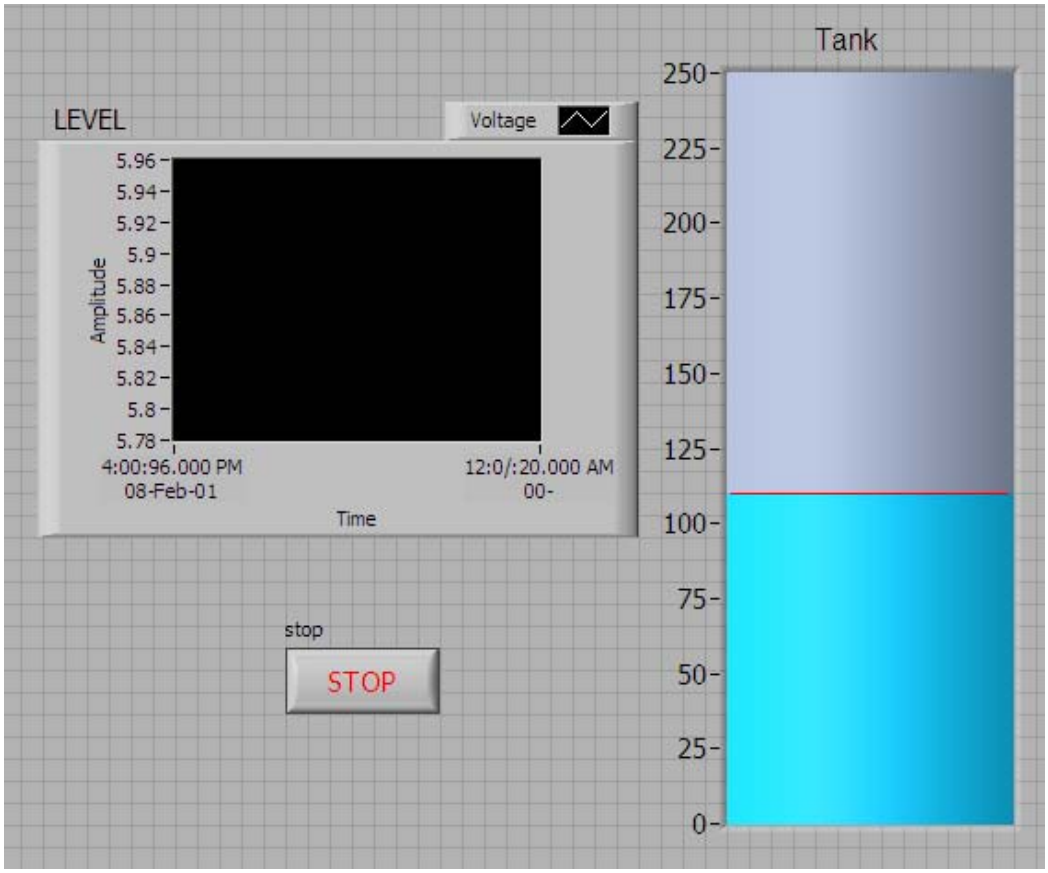
To conduct this experiment, we will have to first connect the 2-Tank system to LabVIEW through the NI DAQ card. The steps are as follows:

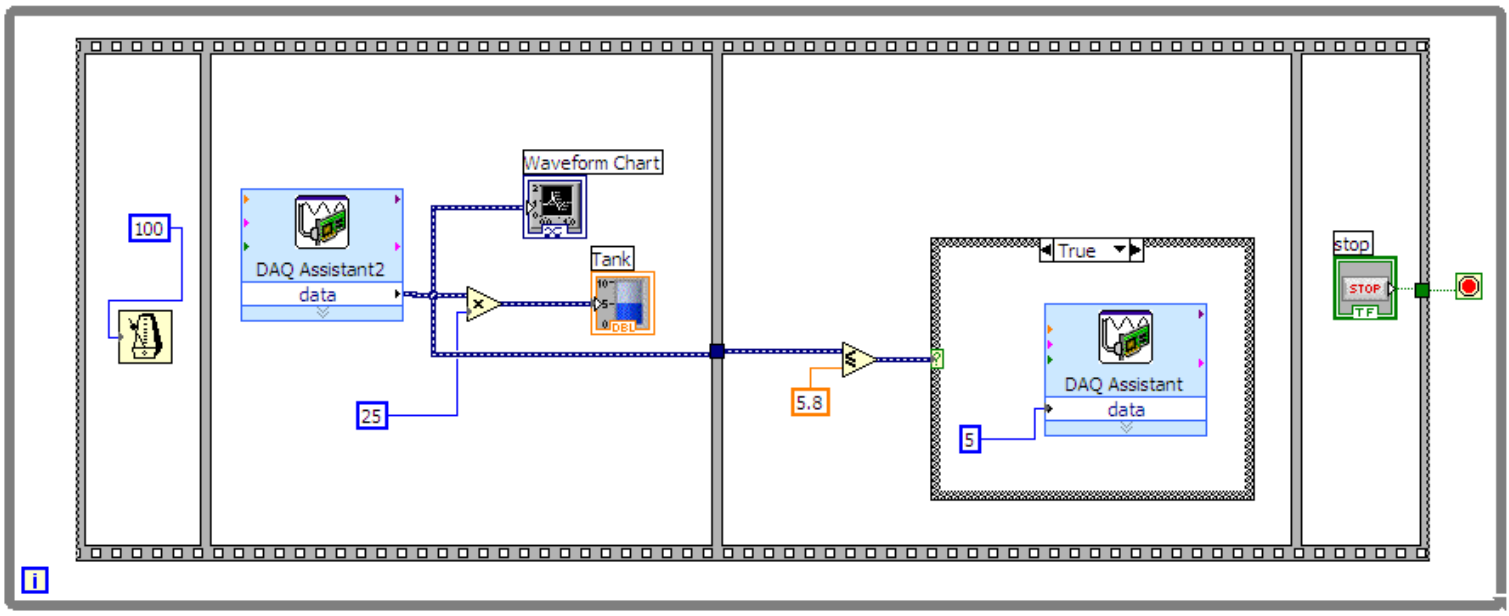
1. Connect the sensor of the tank system (top-most pin) to any Analog Input (AI) pin of the DAQ card.
2. Next connect the motor (2nd last/above ground) to an Analog Output (AO) pin.
3. Connect the ground of the tank (bottom most) pin to a ground of the DAQ.

To build the VI follow the following steps:

1. Click "New VI" button to create a new blank LabVIEW program
2. Select Front Panel and enable Controls Palette to choose a "Waveform Chart" icon in the "Graph" group, add to front panel and name it "Level".
3. Switch to Block Diagram to verify that a new data terminal was created (named "Level").
4. In the Block diagram, go to **"Functions" > Programming> "Structures" > "Flat Sequence"** and place it in the window. Add 4 frames to it by right clicking on the border of the frame and selecting Add Frame After
5. In the first frame go to **"Functions" > Programming> "Timing" > "Wait Until Next ms Multiple"** and place the icon in the frame. Then go to **"Functions" >Programming> "Numeric" >"Numeric Constant"** in the frame and enter the value 100 in it. Wire it to the timing icon. This will set the sampling time to 100ms.
6. In the next frame, open the Functions Palette to get the DAQ Assistant. The path goes as Functions>>Express>>Input>> DAQ Assistant.
7. Select Analog Input and then Voltage. The Physical Tab will appear showing the devices attached and the corresponding pins. Select the required pin (plant sensor) and click Finish.
8. Another DAQ Assistant window will appear. Under the Voltage Input setup, Settings Tab select the appropriate maximum & minimum input signal range and units.(preferably, 0 to 10 volts). Also, in the Task Timing Tab under Acquisition mode select 1 Sample (On Demand).Click OK.
9. Wire the data pin of the DAQ Assistant i.e. sensor output to the waveform chart (Level).
10. You can also add a Tank indicator on the front panel by going to Controls>> Modern>>Numeric>>Tank. But for this you will have to properly scale the tank in order to give you accurate readings. For this multiply the sensor reading by 25 and add a constant to it before sending it to the tank and on the front panel manually change the maximum tank reading to 250. (Method of scaling can be different also.)
11. In the third frame, go to **"Functions" >> Programming>> "Comparison" >> "Greater Or Equal?"** and place the icon. Connect the top pin (x) of the icon to the sensor output from DAQ Assistant and connect the bottom pin (y) to a numeric constant-from the Numeric Palette. Enter your set point as this constant.
12. Go to **"Functions" >> Programming>> "Structures" >> "Case Structure"** and place it in the frame. Connect the Case selector i.e. the question mark on the left border of the Case Structure to the output of the Comparison Icon.
13. In the True window of the Case structure place another DAQ Assistant similar to the first one, except that this should be an Analog Output. Send the value of 0 to the Data pin by using the numeric constant function. Similarly, in the false window, send a value of 5 to the DAQ Assistant. (You can copy-paste it from the True window.)
14. Add a "While Loop" to enhance your program. On the front panel add a Stop button to control the execution of the While Loop.

15. In the last frame place the Stop button terminal and connect it to the stop button of the while loop.
16. To make sure that the motor is turned off when the program is terminated, add another case structure in the last frame and connect the Case Selector (question mark on the left border of the Case Structure) to the Stop button. In the True window of the Case structure place a DAQ Assistant similar to the one in the previous frame (Analog Output). Next, send the value of 0 to its Data pin by using the numeric constant function.
17. In the 2-Tank system open the exit valve a little to see the proper effect of On-Off Control. The valve can be opened to the number 2 position or another one depending on the speed of the motor.
18. After wiring, switch to Front Panel and press the RUN button to execute the VI.



King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

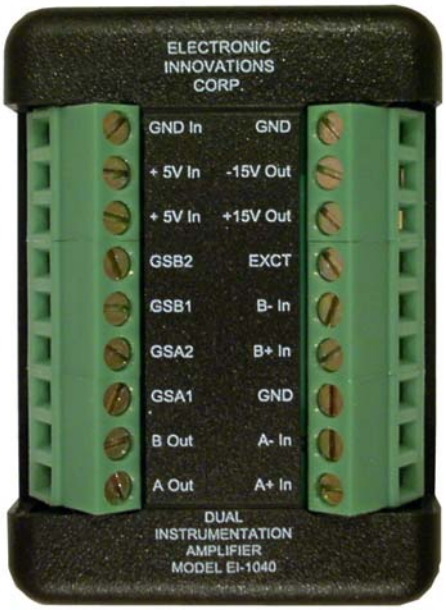
# Lab 8: On-Off Control with Dead Zone

This experiment is similar to the previous one with the exception that instead of having one fixed set-point value, it will have a fixed **"Range"** determined by 2 values i.e. the Upper and Lower limits. This is done since constant switching On and Off of the motor can be damaging and so the frequency of switching must be reduced.

Another difference in this experiment is that we will be using an "Instrumentation Amplifier"- the **EI-1040 Dual Programmable Gain Instrumentation Amplifier**. This is because the NI DAQ card gives a maximum voltage of 5V whereas the Tank system can take up to 10V. So, we will amplify the signal from the DAQ (by a factor of 10) before sending it to the Tank. The pin configuration of the Amplifier is as follows:

| LABEL | DESCRIPTION | LABEL | DESCRIPTION |
|---|---|---|---|
| GND In | POWER SOURCE GROUND | GND | SIGNAL GROUND |
| +5 In | 5 VOLT FROM POWER SOURCE | -15V OUT | -15 VOLT USER AT 8 ma * |
| +5 In | SPARE 5 VOLT TERMINAL | +15V OUT | +15 VOLT USER AT 8 ma * |
| GSB2 | GAIN STATE B2 | EXCT | 4.096 VOLT FOR EXCITATION ** |
| GSB1 | GAIN STATE B1 | B- In | B AMP MINUS INPUT |
| GSA2 | GAIN STATE A2 | B+ In | B AMP PLUS INPUT |
| GSA1 | GAIN STATE A1 | GND | SIGNAL GROUND |
| B Out | B AMPLIFIER OUTPUT | A- In | A AMP MINUS INPUT |
| A Out | A AMPLIFIER OUTPUT | A+ In | A AMP PLUS INPUT |

* Worst case current availability - actual current availability may be greater
** Current availability is 3 mA max




The Amplifier needs a 5V power supply and its gain can be set using the Gain State pins. Combinations of gain can be selected so that for example one amplifier can have a gain of 10 and the other have a gain of 100. The different gains are 1, 10, 100, and 1000 and can be set through the following combinations:

| GAIN | TERMINAL | TERMINAL |
|---|---|---|
| | GSA1 or B1 | GSA2 or B2 |
| 1 | 0 | 0 |
| 10 | 1 | 0 |
| 100 | 0 | 1 |
| 1000 | 1 | 1 |

To conduct the experiment, we will have to first connect the 2-Tank system to LabVIEW through the NI DAQ card and Amplifier set up. The steps are as follows:

1. Connect the level sensor of the tank system to any Analog Input (AI) pin of the DAQ.
2. Connect the **"+5 In"** and **"GND In"** pins of the Amplifier to the 5V and ground terminals of the **Power supply**.
3. Connect the negative input of the amplifier you are using, **"A- In"** to ground and connect the positive inputs, **"A+ In"** to an Analog Output (AO) pin.
4. Next connect the motor to the Amplifier Output i.e. **"A Out"**.
5. Connect **"GSA1"** and **"GSA2"** to 2 Digital (DO) pins.
6. Connect the ground of the Amplifier to the ground of the DAQ.
7. Connect the ground of the tank (bottom most) pin to a ground of the DAQ.

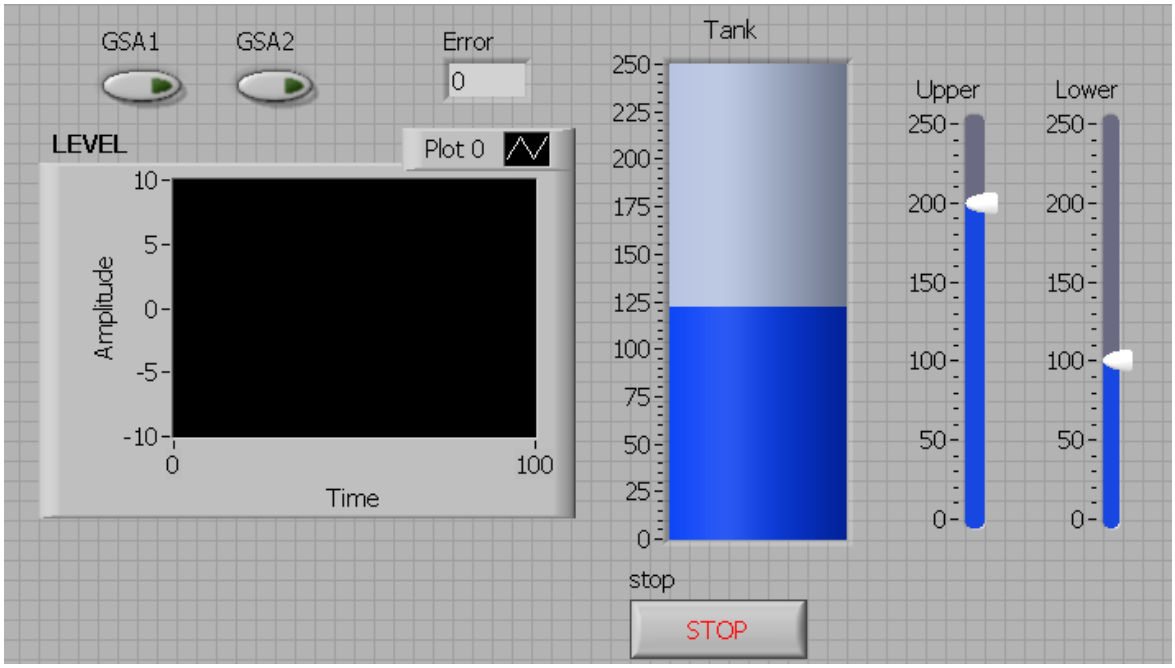
**Note: Make sure that ALL devices are connected to a common ground.**

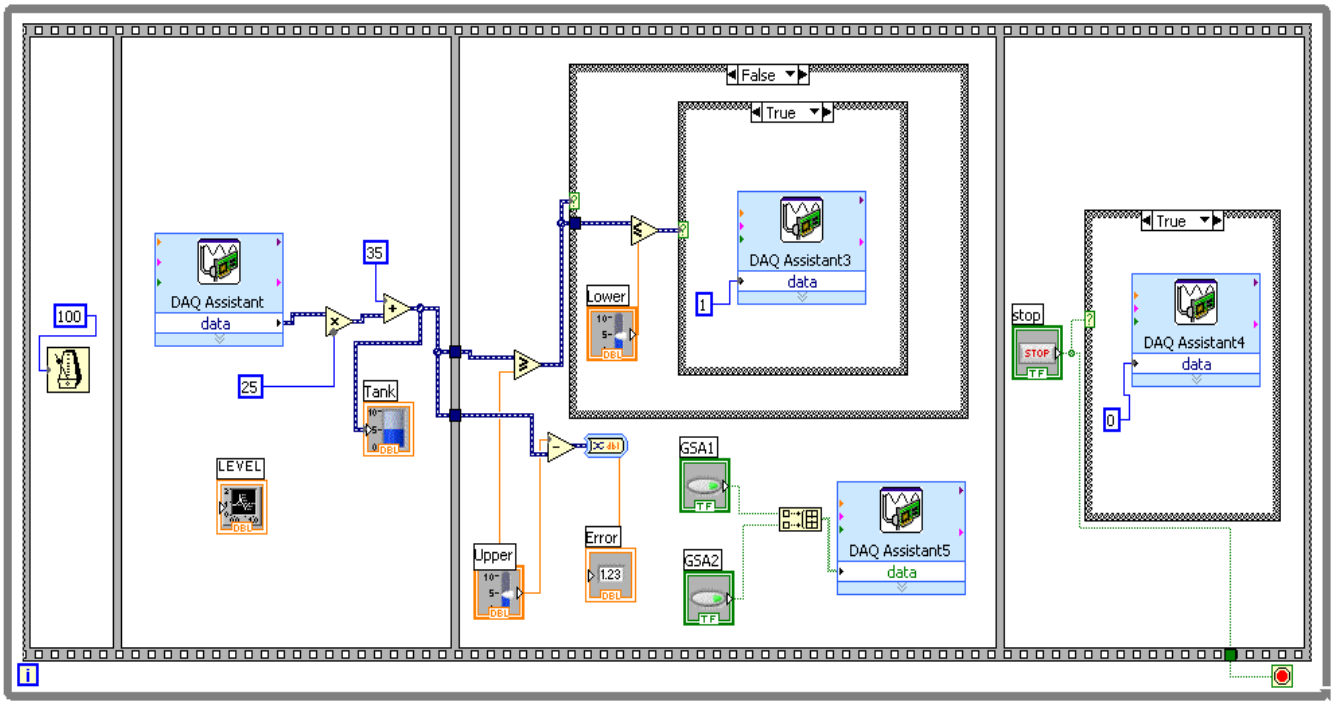
To build the VI follow the following steps:

1. Click "New VI" button to create a new blank LabVIEW program
2. Select Front Panel and choose a **"Waveform Chart"** icon, name it "Level".
3. In the Block diagram, add **"Flat Sequence"** and put it in the window. Next, add 4 frames by right clicking on the border of the frame and selecting Add Frame After (or Before).
4. In the first frame add **"Wait Until Next ms Multiple"** and place the icon in the frame. Then add **"Numeric Constant"** of value 100. Wire it to the timing icon. This will set the sampling time to 100ms.
5. In the next frame, add **DAQ Assistant**. Select Analog Input and then Voltage.
    a. The Physical Tab will appear. Select the required pin (Level sensor) and click Finish.
    b. Another DAQ Assistant window will appear. Under the Voltage Input setup, Settings Tab select the appropriate maximum & minimum input signal range and units (preferably, 0 to 10 volts). Also, in the Task Timing Tab under Acquisition mode select 1 Sample (On Demand).Click OK.
6. Wire the data pin of the DAQ Assistant i.e. sensor output to the waveform chart (Level).
   **Note**: You can also add a Tank indicator on the front panel. But for this you will have to properly scale the tank in order to give you accurate readings. For this multiply the sensor reading by 25 and add a constant to it before sending it to the tank and on the front panel manually change the maximum tank reading to 250. (Method of scaling can be different also.)
7. In the third frame, add **"Greater Or Equal?"**. Connect the top pin (x) of the icon to the sensor output from DAQ Assistant and connect the bottom pin (y) to Numeric control from the Numeric Palette. Enter your Upper limit as this.
   Note: You can also place **"Vertical Pointer Slides"** on the Front Panel from **Controls>>Modern>>Numeric** and use them to set the Upper/Lower limits.
8. Add **"Case Structure"** and place it in the frame. Connect the Case selector i.e. the question mark on the border of the Case Structure to the output of the Comparison Icon.
9. In the *True* window of the Case structure place another DAQ Assistant similar to the first one, except that this should be an Analog Output (AO). Send value of 0 to the Data pin.
10. In the *False* window, place another Case structure. Place the **"Less Or Equal?"** icon outside the structure and connect its output to the Case Selector (question mark). Connect the top pin (x) of the icon to the sensor output from DAQ Assistant and connect the bottom pin (y) to numeric control from the Numeric Palette. Enter your Lower limit.
    Note: You can use **"Vertical Pointer Slides"** as well.
11. In the *True* window of the inner structure place another similar DAQ Assistant (Analog Output). Send the value of 1 (it will be later amplified to 10V) to the Data pin. Leave the other window blank.
12. In the Front Panel, add **"Push Button"** indicator in "Boolean" group. Add 2 of them and name them "GSA1" and "GSA2". (***Refer to Gains table in previous page***)

13. In the Block Diagram (3rd frame), connect all the Boolean terminals into one **"Build arrays".** Place a DAQ Assistant and select Digital I/O and then Line Output. Select the required pins, select 1 Sample (On Demand) and click OK. Finally connect the array output to the data pin of the DAQ assistant. **Note**: Make sure that the correct pin numbers are assigned to the Boolean switches in the DAQ Assistant.
14. Add "While Loop" to enhance your program.
15. In the last frame place the Stop button terminal and connect it to the stop button of the while loop.
16. To make sure that the motor is turned off when the program is terminated, add another case structure in the last frame and connect the Case Selector (question mark on the left border of the Case Structure) to the Stop button. In the True window of the Case structure place a DAQ Assistant similar to the one in the previous frame (Analog Output). Next, send the value of 0 to its Data pin by using the numeric constant function.
17. After wiring, switch to Front Panel and press the RUN button to execute the VI.

**Note: push "GSA1" only to enable gain of 10. (Refer to 'Gains' table in previous page)**



King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 9: Proportional Control

**Proportional Controller (part of the PID controller)** is a common feedback loop component in industrial control systems. The controller takes a measured value from a process or other apparatus and compares it with a reference setpoint value. The difference (or "error" signal) is then used to adjust some input to the process in order to bring the process' measured value back to its desired setpoint. Basically, when the controller reads a sensor, it subtracts this measurement from the "setpoint" to determine the "error". It then uses the error to calculate a correction to the process's input variable (the "action") so that this correction will remove the error from the process's output measurement.

It is used mainly to handle the immediate error, which is multiplied by a constant *P* (for "proportional"), and added to the controlled quantity. *P* is only valid in the band over which a controller's output is proportional to the error of the system. This is known as the Propotional Band, often abbreviated as $P_b$. A controller setting of 100% proportional band means that a 100% change of the error signal (setpoint – process variable) will result in 100% change of the output, which is a gain of 1.0. A 20% proportional band indicates that 20% change in error gives a 100% output change, which is a gain of 5.

$P_b = 100/\text{gain}$ OR $K_p = \frac{1}{P_b}$

With proportional band, the controller output is proportional to the error or a change in measurement (depending on the controller). So,

(controller output) = (error)*100/(proportional band)

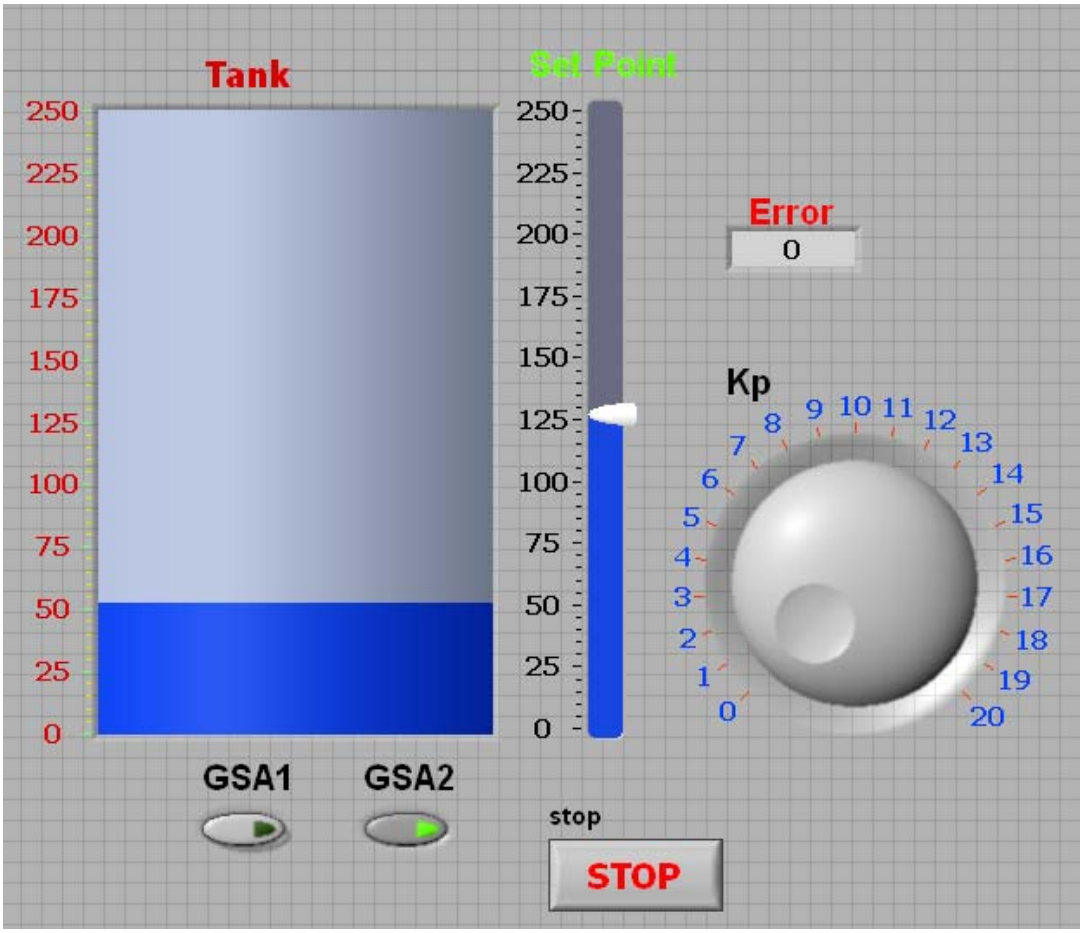
This theory will be implemented on the 2-Tank system in this experiment. The controller will be designed in a VI while the hardware connections remain the same- as shown below:

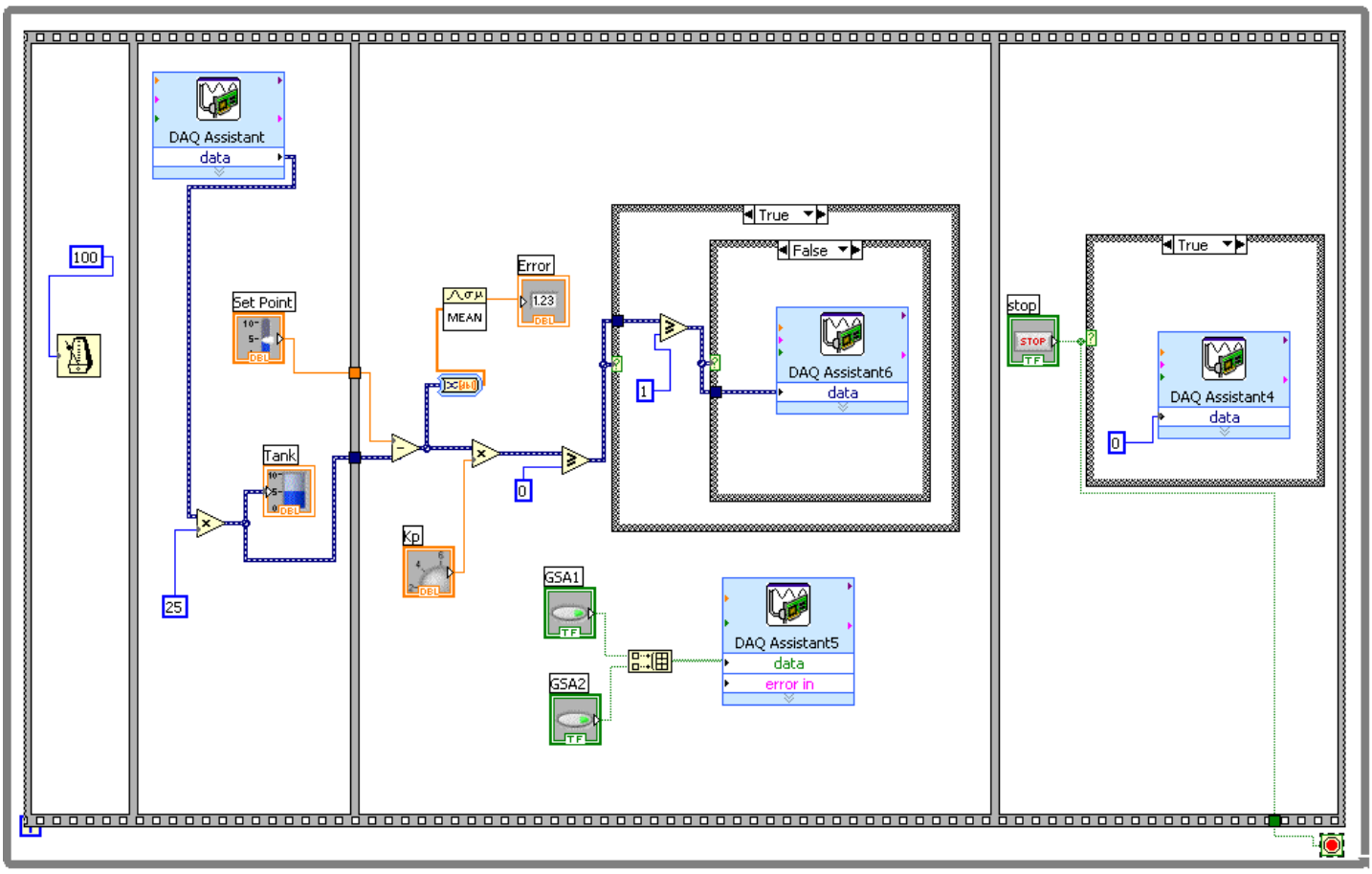
1. Connect the sensor of the tank system (top-most pin) to any Analog Input (AI) pin of the DAQ card.
2. Connect the +5 In and GND In pins of the Amplifier to the 5V and ground terminals of the Power supply.
3. Connect the negative input of the amplifier you are using, A- In or B- In, to ground and connect the positive inputs, A+ In or B+ In, to an Analog Output (AO) pin of the DAQ.
4. Next connect the motor (2nd last/above ground) to the Amplifier Output i.e. A Out or B Out.
5. Connect the ground of the Amplifier to the ground of the DAQ.
6. Connect the ground of the tank (bottom most) pin to a ground of the DAQ.

Note: Make sure that ALL devices are connected to a common ground.

The VI will be build as follows:

1. Click “New VI” button to create a new blank LabVIEW program.
2. In the Block diagram, go to **“Functions” >> Programming>> “Structures” >> “Flat Sequence”** and place it in the window. Next add 4 frames to it by right clicking on the border of the frame and selecting Add Frame After (or Before).
3. Keep the 1st two frames and the last frame the same as they were for On-Off Control viz.
   - The 1st frame to set the sampling time to 100ms- using **Wait Until Next ms Multiple** and **Numeric Constant**.
   - The 2nd frame to receive the sensor signal from the Tank, scale it properly and display it on the front panel in a graph as well as tank format- using **DAQ Assistant, Waveform Chart, Tank** and other numeric icons.
   - The last frame to manually terminate the execution of the program through a stop button on the front panel and make sure the motor is turned off at the end- using **DAQ Assistant**, **Case Structure** and **Numeric Constant.** (The entire Flat Sequence must be included in the while loop and the Stop button terminal must be connected to the stop button of the while loop)
4. Calculate the Error by subtracting (**“Functions” >> Programming>> “Numeric” >> “Subtract”**) the sensor value or level from the desired set point. The set point can be given in the form of a Numeric Constant in the Block diagram or through Vertical pointer slides, Numeric controls, etc. on the Front panel. This can be done in the 2nd or 3rd frame.
5. On the front panel, add a Control Knob from the Numeric palette. This will be used to control the Proportional gain $K_p$. In the 3rd frame of the Block diagram sequence, multiply the error with the gain- connect the error and gain terminal to a multiplication block.
6. In the same frame check the above product (input to controller) and if it is greater than 1 send one to the Tank system- using DAQ. If it is lesser than 0 send the tank 0. If it is between 0 and 1, send the control input as it is. The comparison can be done using “Greater Or Equal?” and “Lesser Or Equal?” functions along with a Case Structure having another Case Structure inside (as in the On-Off Control). Here the control input is connected to the Case Selector.
7. After all the wiring is complete switch to Front Panel and press the RUN button to execute the VI.


Tank
Set Point
250
225
200
175
150
125
100
75
50
25
0
Error
0
Kp
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
GSA1
GSA2
stop
STOP


DAQ Assistant
data
100
Set Point
Tank
25
MEAN
Error
1.23
Kp
True
False
DAQ Assistant6
data
1
0
GSA1
GSA2
DAQ Assistant5
data
error in
stop
STOP
TF
True
DAQ Assistant4
data
0

King Fahd University of Petroleum and Minerals
Systems Engineering Department

CISE 318
Computer Control Systems

# Lab 10: PI Controller

The next step in PID control is the inclusion of the **Integral** component – It is needed to learn from the past. The error is integrated (added up) over a period of time, and then multiplied by a constant $K_i$ (making an average), and added to the controlled quantity. A simple proportional system either oscillates, moving back and forth around the setpoint because there's nothing to remove the error when it overshoots, or oscillates and/or stabilizes at a too low or too high value. By adding a proportion of the average error to the process input, the average difference between the process output and the setpoint is continually reduced. Therefore, eventually, a well-tuned PID loop's process output will settle down at the setpoint. As an example, a system that has a tendency for a lower value (heater in a cold environment), a simple proportional system would oscillate and/or stabilize at a too low value because when zero error is reached $P$ is also zero thereby halting the system until it again is too low. Larger $K_i$ implies steady state errors are eliminated quicker. The tradeoff is larger overshoot: any negative error integrated during transient response must be integrated away by positive error before we reach steady state. The integral component is always used with the proportional one and is so referred to as **PI controller**.

This theory will be implemented on the 2-Tank system in this experiment. The controller will be designed in a VI while the hardware connections remain the same- as shown below:

1. Connect the sensor of the tank system (top-most pin) to any Analog Input (AI) pin of the DAQ card.
2. Connect the +5 In and GND In pins of the Amplifier to the 5V and ground terminals of the Power supply.
3. Connect the negative input of the amplifier you are using, A- In or B- In, to ground and connect the positive inputs, A+ In or B+ In, to an Analog Output (AO) pin of the DAQ.
4. Next connect the motor (2nd last/above ground) to the Amplifier Output i.e. A Out or B Out.
5. Connect the ground of the Amplifier to the ground of the DAQ.
6. Connect the ground of the tank (bottom most) pin to a ground of the DAQ.

Note: Make sure that ALL devices are connected to a common ground.

The VI will be build as follows:

1. Click “New VI” button to create a new blank LabVIEW program.
2. In the Block diagram, go to **“Functions” >> Programming>> “Structures” >> “Flat Sequence”** and place it in the window. Next add 4 frames to it by right clicking on the border of the frame and selecting Add Frame After (or Before).
3. Keep the 1st two frames and the last frame the same as they were for On-Off Control viz.
1. The 1st frame to set the sampling time to 100ms- using **Wait Until Next ms Multiple** and **Numeric Constant**.

2. The 2nd frame to receive the sensor signal from the Tank, scale it properly and display it on the front panel in a graph as well as tank format- using **DAQ Assistant, Waveform Chart, Tank** and other numeric icons.
3. The last frame to manually terminate the execution of the program through a stop button on the front panel and make sure the motor is turned off at the end- using **DAQ Assistant**, **Case Structure** and **Numeric Constant.**
4. (The entire Flat Sequence must be included in the while loop and the Stop button terminal must be connected to the stop button of the while loop)
5. Calculate the Error by subtracting (**"Functions" >> Programming>> "Numeric" >> "Subtract"**) the sensor value or level from the desired set point. The set point can be given in the form of a Numeric Constant in the Block diagram or through Vertical pointer slides, Numeric controls, etc. on the Front panel. This can be done in the 2nd or 3rd frame.
6. On the front panel, add 2 Control Knobs from the Numeric palette. This will be used to control the Proportional gain $K_p$ and the Integral Gain $K_I$.
7. In the 3rd frame of the Block diagram sequence, check if the error is less than zero. If it is, then send the tank 0.If not go to the next step. The comparison can be made using the Case Structure and the "Lesser Or Equal?" function.
8. Multiply the error with the gain by connecting the error and gain terminal to a multiplication block. Also, integrate the error by sending it to the Integral block (**"Functions" >> Mathematics>> Integ & diff >>Time Domain Math-** select **Integral** in this block) and then multiply the integrated error with the Integral gain as was done with the Proportional gain. Next, add the 2 products together (use compound arithmetic or 2 add functions). Send the sum to the Tank through the DAQ Assistant.
9. In the same frame check the above product (input to controller) and if it is greater than 1 send one to the Tank system- using DAQ. If it is lesser than 0 send the tank 0. If it is between 0 and 1, send the control input as it is. The comparison can be done using "Greater Or Equal?" and "Lesser Or Equal?" functions along with a Case Structure having another Case Structure inside (as in the On-Off Control). Here the control input is connected to the Case Selector.
10. In the 2-Tank system open the exit valve a little to see the proper effect of the PI Control. The valve can be opened to the number 2 position or another one depending on the speed of the motor.
11. After all the wiring is complete switch to Front Panel and press the RUN button to execute the VI.