

Introduction to LabVIEW – Exercise-1

Objective

In this Laboratory, you will write simple VIs to incorporate basic programming structures in LabVIEW. This section will teach you fundamentals of LabVIEW front panel, block diagram and tool palette, and also explains data flow in the block diagram. It will teach you the featured structures include While Loop, For Loop, Case Structure and Sequence Structure.

You will learn the following:

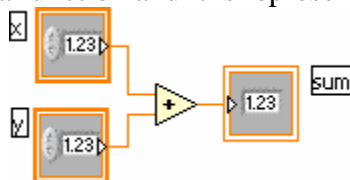
- Simple arithmetic functions and their use to perform different operations
- Differences between indicators and controls and how they can be used
- Differences between a While Loop and a For Loop
- How shift registers access values from previous iterations.
- How the Case Structures work.

Introduction

LabVIEW is written on graphical structure. If you think of “summation”, means adding two values. It can be written in BASIC or in C language as,

```
double x, y, sum;  
x = 2;  
y = 2;  
sum = x + y;  
printf sum;  
end.
```

While in LabVIEW summation is a function and it is represent by following symbol.



In LabVIEW, such mathematical and logical functions are represented graphically. Likewise, one can design his own function according to his application.

LabVIEW Example # 1

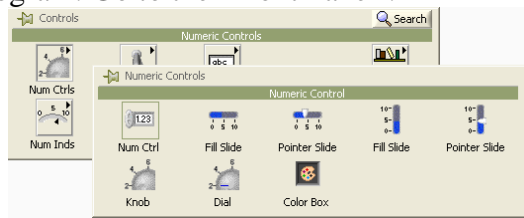
Objective

Build a VI to add and multiply two given numbers and display the results

1. Start 'LabVIEW' program.
2. Open a 'blank VI'

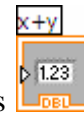
- You will have two screens available. One is the 'front panel' and the second is called 'block diagram'. Front Panel will always have the 'controls' and 'indicators' required by the program to show the input and output parameters. 'Block diagram' will have the detailed graphical representation of the actual program. Go to the 'Front Panel'.

- The proposed program needs two numbers as inputs for 'addition' and 'multiplication'. Remember all inputs are 'controls' and all outputs are 'indicators'. Right click your mouse keeping the cursor anywhere on the front panel. That opens the 'controls' palette. Then select 'Numerical controls' and a 'Numeric control', which brings a numeric control display



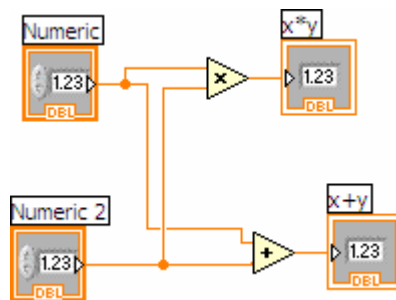
control display onto the front panel. This can be used to input the required number as input 1. These steps may be repeated to obtain a second control input, where you can input the second number. This will be your input.2. Now shift over to the 'Block diagram' by pressing CTRL+E characters on your keyboard.

- You will see the icons representing the two numeric control displays provided on the front panel. Right click the mouse (keeping the cursor in the free space on the block diagram). That opens up the 'functions' palette. Select Arith/compare palette. This opens up the Arithmetic/comparison palette. Select 'Numeric' palette. This opens up the 'express numeric' palette, which shows all the arithmetic functions. Select and drag the 'add' and 'multiply' functions on to the block diagram.
- Right click on output side of the 'add' and 'multiply' functions and create indicator by



selecting **Create>Indicator**. This provides the display indicators for showing the results of the respective functions.

- Get wiring tool and wire both the numeric controls to both the input terminals of the 'add' and 'multiply' functions. The block diagram will then show the following graphical program on the screen.



- Go to the front panel. Your program is now ready for execution. Input the numbers you want to add and multiply in the respective numeric input boxes. Hit the run button on the tool bar and the output display boxes will show the results of addition and multiplication functions.
- Save the VI using the 'File' pull down menu as 'example1-add+multiply'

Now we will learn how loops are designed and how it works in LabVIEW. Structures are graphical representations of the loops. Loops use structures on the block diagram **to repeat** code and to execute code conditionally or in a specific order.

Like other nodes, structures have terminals that connect them to other block diagram nodes, execute automatically when input data are available, and supply data to output wires when execution completes.

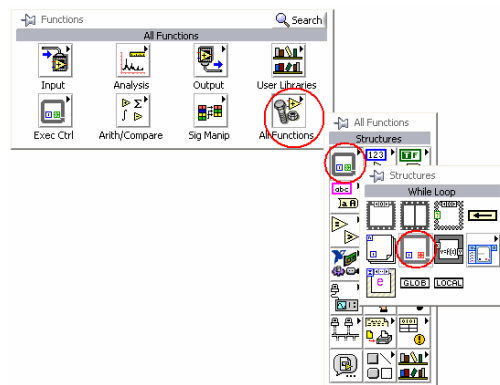
- | | |
|---------------------------|--|
| For Loop | Executes a code a set number of times. |
| While Loop | Executes a code until a condition is met. |
| Case structure | executes depending on the input value passed to the structure. |
| Sequence structure | Contains one or more sequential code, which execute in sequential order. |

LabVIEW Example # 2

Objective



To write a VI for the Multiplication of a random number with 10 and displaying the result continuously, until it is stopped.

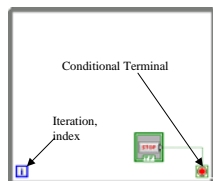
1. Open LabVIEW and start a new VI. A blank VI opens the front panel and block diagram simultaneously. If you save one VI, second one will be saved automatically. Put a while loop from the function palette on the block diagram. This comes from **Functions>>Structures>>While Loop**






2. Right click on the conditional terminal and Select **Create>>Control**. This provides a True or False Control for the conditional terminal.




- In text base programming Do Loops repeat-until a condition is met. While loops iterate as long as a condition is true. The condition can be tested at the beginning or the end of the loop. In LabVIEW, the while Loop executes a graphical code while a condition is true or until a **condition** is met. A LabVIEW while loop always executes at least once.
- Look at the Loop and identify the iteration index, conditional terminal and Boolean operator from the figure.
- **Stop If True**. When a conditional terminal is in the **Stop If True**  state, the While Loop executes its Block diagram until the conditional terminal receives a TRUE value from the front panel. The VI checks the conditional terminal at the end of each iteration.
- **Continue If True** . When a conditional terminal is in the **Continue If**

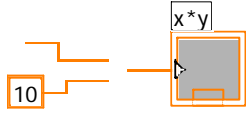







True  state, the While Loop executes its sub-diagram while the conditional terminal receives a True value.

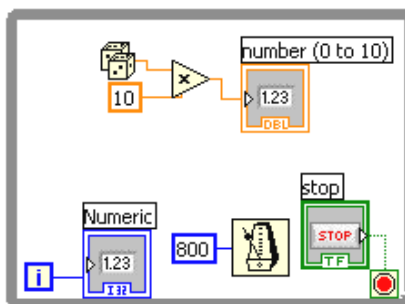
- Get a random number function  by right clicking on blank space from **All function » Functions » Numeric » Random number**. Same way get multiply function  from the numeric palette.
- Get wiring tool and wire the random number function to one input of ‘multiply’ function. [By default, you are in automatic tool selection mode, which provides you as soon as you bring the cursor near input/output terminal of any functional operator. In case, you can’t get the wiring tool automatically, go to windows pull down menu and click on the ‘show tools palette’. Then click on ‘connect wire’ tool on the palette.] Right click on 2nd input terminal of the multiply function and select **Create»Constant**, and type 10 in the blank

box. You should now have this: 

- Right click on the output side of the multiply function and create an indicator by selecting

Create»Indicator. Your block diagram will now show: . This step will show the output value from the multiply function.

- Right click on the iteration  and create indicator to see number of iterations so far done. [Follow the same procedure you did for creating the indicator in the last step].
- Double click on the stop button, this will show you location of the stop button in front panel. Identify the indicator of the number of iterations and Output number indicator and change the title of these indicators, if necessary.
- Hit the run  button on the tool bar and observe the running program. It will be too fast to see the numbers.
- Stop the program by pressing the  button and go to block diagram. Use CTRL-E whenever you want to move from front panel to block diagram and vice versa. Get a metronome  by right clicking **Functions »All Function »Time & Dialog » Wait**
- until next ms multiple**. Right click on left hand side (at its input) of  and create a constant and type 800 in blue box.
- Your block diagram should look like the diagram below.



- Go to the front panel and run the program and observe the changes, note the changes.
- Save your work as ‘example-2-random-while’ file name.



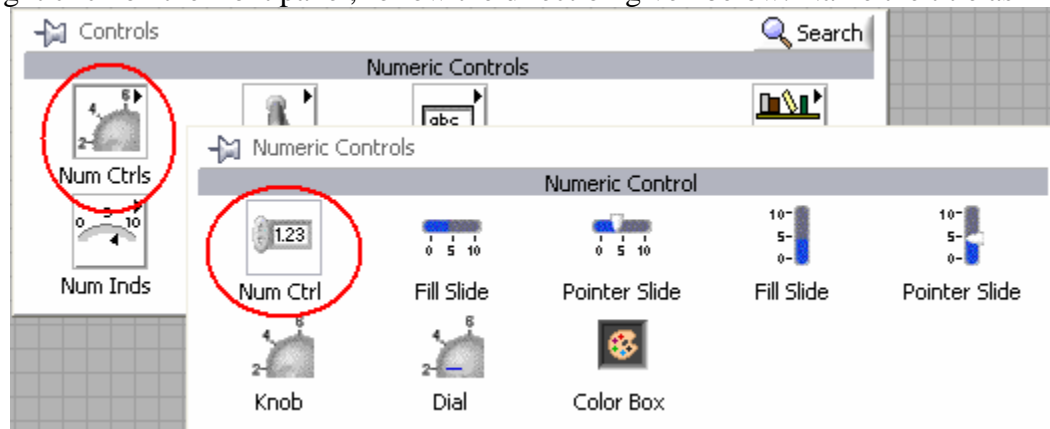
- Each iteration in the While loop will be delayed by 800 milliseconds, now you will easily see the iterations and the output values.
- The iteration terminal contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns 0.

LabVIEW Example # 3

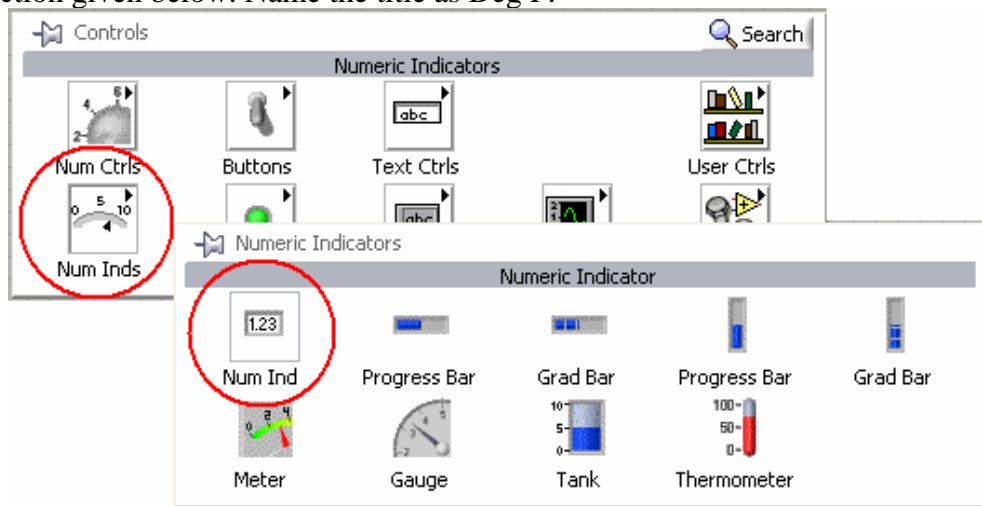
Objective

Write a VI to convert a given temperature value from Degrees C to Degrees F. This VI will teach you:






- (a) Methodology for writing small VIs targeting a conversion in units.
 - (b) The procedure for using a small program as a sub VI in another program by creating Icons. You will use this VI as a sub VI in another program later to build a thermometer demo VI.
1. Open a Blank VI and be in the front panel. Get one **numerical control** for *degree C* by right click on the front panel, follow the direction given below. Name the title as Deg C.

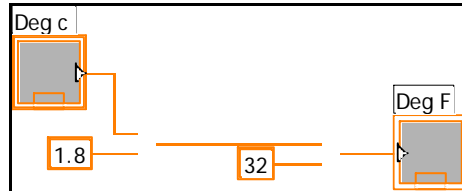


2. Get one **numerical Indicator** for *degree F* by right click on the front panel, follow the direction given below. Name the title as Deg F.

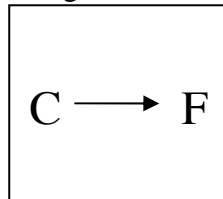


3. Ctrl-E to Block Diagram

4. Use Arithmetic/comparison palette to obtain all mathematical (, , , , ) functions and constants required for this conversion, following the procedure used in the earlier example. The constants when you drag on to the block diagram will show a zero inside and are blue in color. As you change the zero inside to the required number, it will change the color from blue to orange, which means the representation type of the constant has changed from integer to a real number. These will have to be organized in the block diagram. Connect the constants, functions and indicators in the required order.
5. Your block diagram should look like,



6. Run the VI from the front panel and check the conversion for different deg C values.
7. Edit the Icon on the top right hand corner of the front panel as shown below. Use tools palette for typing C and F and writing the arrow as shown in the diagram below.



8. Right click on the icon and select “Show Connector”
- You will see two terminals, left hand side is input terminal (Deg C) and Right hand side is output terminal (Deg F).
9. Select left terminal of the icon and click on Deg C numerical control in the front panel. After that, click on right side terminal and click on Deg F. This will create two connector for sub VI, one is Deg C and second is Deg F
10. If you did not get this correct, get help from your Lab supervisor. You will be using this in the next exercise.
- Save this file as ‘c to f converter.vi’

As you can see in the block diagram, this VI implemented the formula for converting temperature from ‘degrees Celcius’ to ‘degrees Fahrenheit’ using mathematical functions.

LabVIEW Example # 4

Objective

Build a thermometer, which measures temperature and displays temperature values using the C to F converter you built in the earlier exercise.

You will be doing this exercise using two methods. In the first method, you use a random number generator function and use those inputs for displaying as temperatures.

In the second method, you will be using a 'DAQ assistant from LabVIEW' for displaying the real temperature values acquired from a thermocouple. It will be discussed in Lab #2.

In this exercise you will learn following.

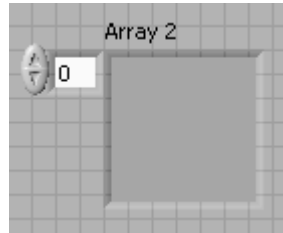
1. Use of the 'for loop'
2. How to use 'sub VI' in another program.
3. Plotting the data using a waveform graph, instead of displaying as values.

Thermometer takes input from another sub-vi, which generates values. First we'll build a 'generate Random values .vi'

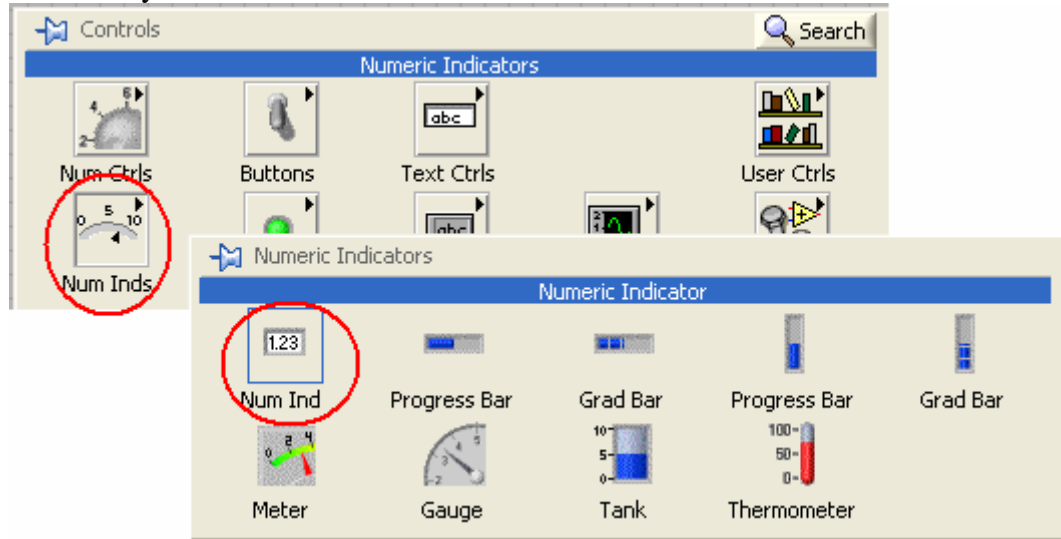
Procedure to write 'generate Random values .vi'

Front Panel

1. Open a new blank VI, Go to the 'Front Panel'
 - a) Right click in front panel, that brings up controls palette and following further through **Controls» All controls» Array & cluster» Array**, select the array. It should look like



- b) Now from the Controls palette, drag numerical indicator and put in square space of the array container.



- c) Your result should look like,

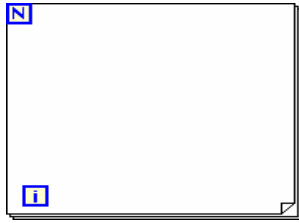


This is called array.



Read First

Now you need to use a 'for loop' in this example.



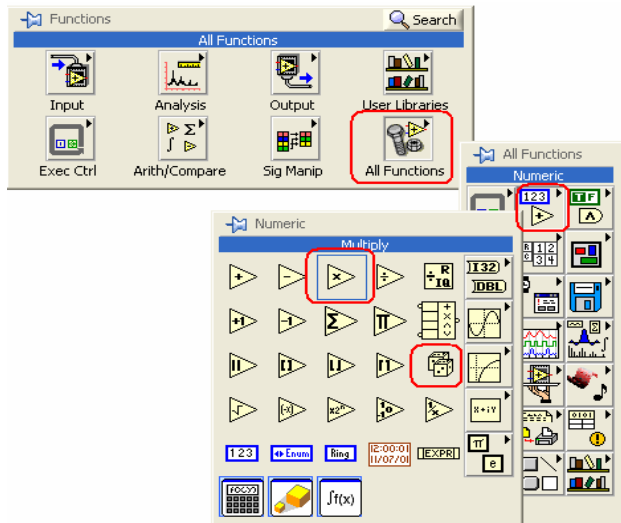
A 'For Loop' executes a code for a set number of times.

The value (input) to the count terminal [N] indicates how many times to repeat the loop. Set the count explicitly by wiring a value from outside the loop on the left.

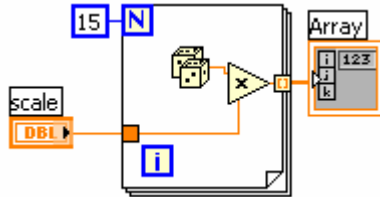
The iteration terminal [i] (an output terminal) contains the number of completed iterations. The iteration count always starts at zero.

Both the count and iteration terminals are signed long integers. **If you wire 0 or a negative number to the count terminal, the loop does not execute.**

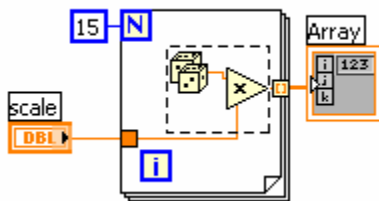
- Switch to block diagram and build a 'for loop' as given below. Get 'all functions' from the 'Functions' palette and connect all wire as given in the figure.



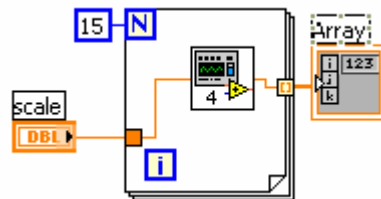
- Connect all the wires properly and create the block diagram as shown.



- Save this VI as 'Generate Random Values.VI'. Open the front panel. Run the program and see the results.
 - Right click on the upper right corner icon and select show connector, assign array as output connector by clicking icon connector followed by clicking array indicator in front panel.
 - Save your changes, you will use this VI as sub VI in the next example.
- Draw a selection box around the random and the multiply

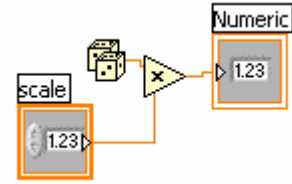


then from the main menu select **EDIT>>Create SubVI**.



You should then see :

Open the newly created subVI by double clicking it. Here you will find a complete subVI all terminals connected and

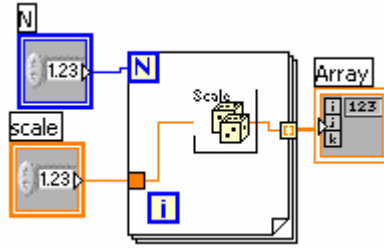


generic names added where there was none supplied.

Change

the Numeric to 'Scaled random' and save as *Scaled Random*. You can also decorate the icon.

- Back on *Generate Random Values.VI*, right click the constant 15 and change to control. Name it N. Notice that the value from the constant becomes the default value for the



control.

- Add and wire a terminal on the icon for N.



Don't forget to

save *Generate Random Values.VI*.

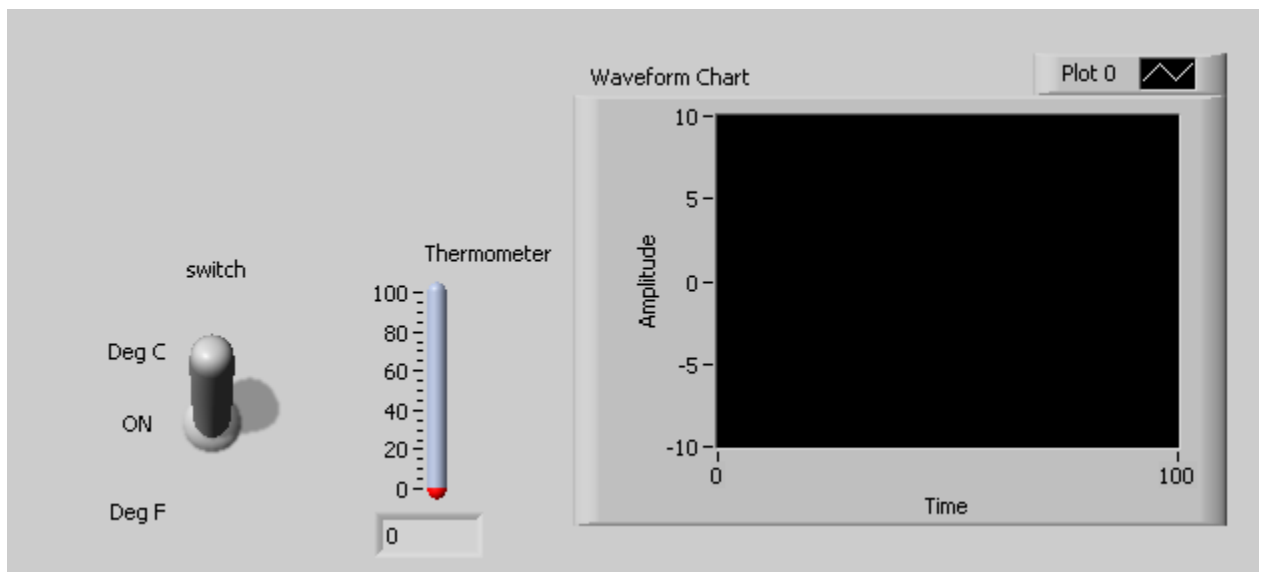
LabVIEW Example # 5

Objective

Create a thermometer using the above two VIs and plot the output temperature values on a chart to display the variation of temperature values. At same time show the mean of the temperature values on a thermometer.

Front Panel

1. Open a new front panel, go to **File»New**.
2. Create the Front panel thermometer indicator, as shown on the following figure.




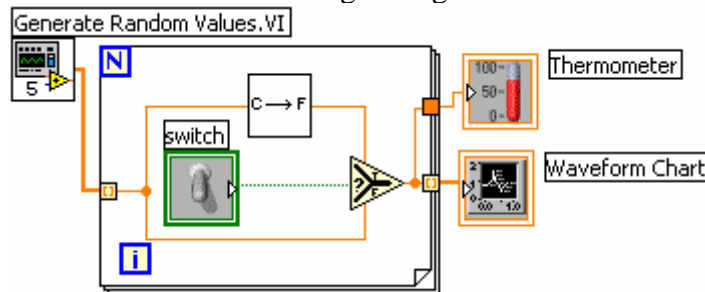
- a) Select the thermometer on the **Controls»Numeric Indicators** palette and place it on the front panel.
- b) Right-click the thermometer and select **Visible Items»Digital Display** from the shortcut menu to display the digital display for the thermometer.
- c) Get waveform chart following **Controls»graph indicators»>>chart**.
3. Create the vertical switch control.
 - a) Select the vertical toggle switch on the **Controls»Buttons** palette.
 - b) Place a label, deg C, next to the TRUE position of the switch, as shown in the previous front panel.
 - c) Place a free label, deg F, next to the FALSE position of the switch.

Block Diagram

4. Using CTRL-E, to switch over to the block diagram.
5. Build the following block diagram.
 - a. To include a sub-VI, Go to **Functions»>>All Functions»>>Select a VI**. It gives you a dialog box asking you to type in the subVI file name. Ensure to enter the correct directory, where you have saved the sub VI file from the last two

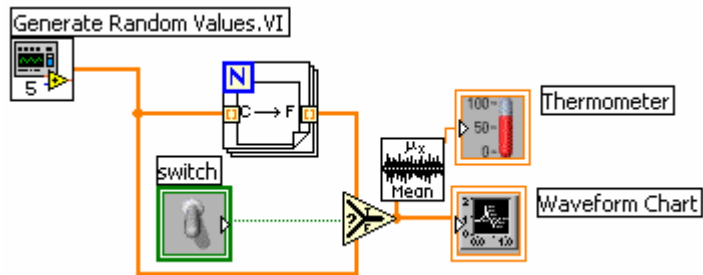
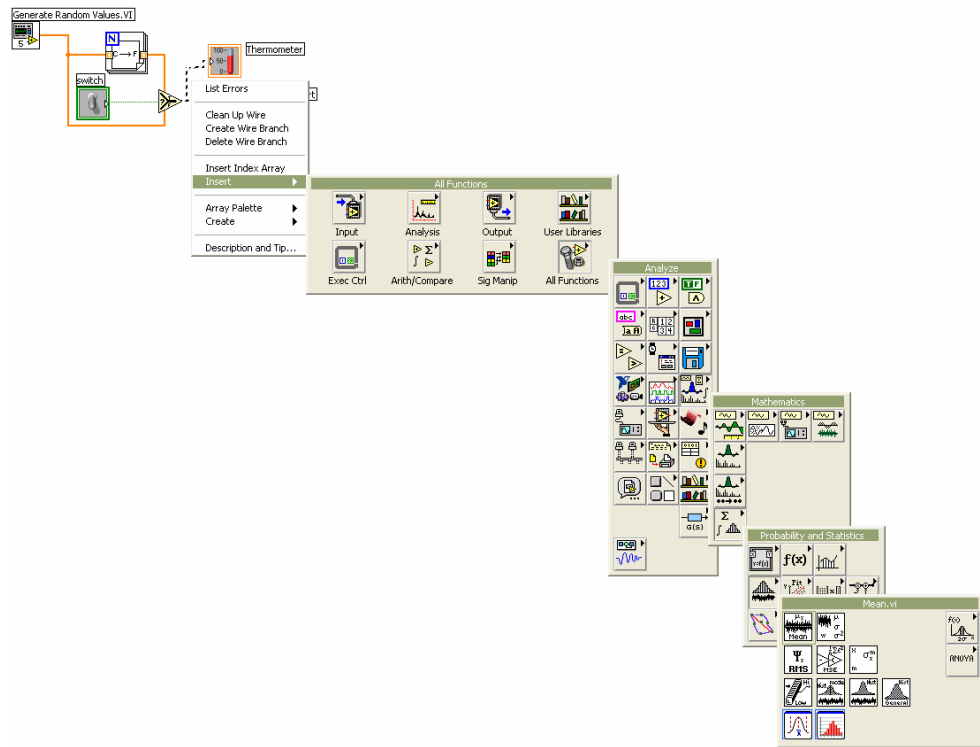
examples. Following this procedure, place the ‘generate Random values.vi’ and ‘C to F converter.vi’ in this block diagram. (Note that we are assuming the random values generated by the sub-VI ‘generate random values. VI’ as temperature values. In the next laboratory, we will measure the real temperature through a Data Acquisition Board and use real temperature values.)

- b. Place the ‘for loop’ following the directions given in the last example.
- c. Place a ‘Select Switch’  on the block diagram from **Functions>>Arithmetic & Comparison>>Comparison>>Select**. This function returns either the Fahrenheit (FALSE) or Celsius (TRUE) temperature value, depending on the value selected on front panel.
- d. Keeping the cursor at the center terminal of the select switch, right click the mouse and create ‘control’, which provides the true or false control switch for the select.
- e. Connect all the wires using wiring tool as shown in the block diagram.



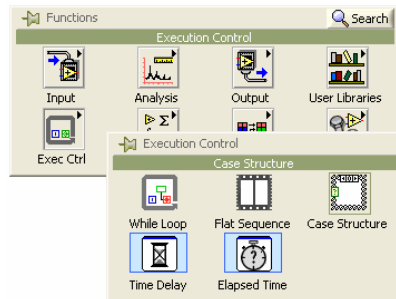
This VI illustrates LabVIEW’s automatic determination of N. When an array is passed into a *For loop*, LabVIEW uses the array length to determine N and indexes over each of the array elements. This is why the wire from the array becomes thin inside the loop. Inside the loop, we are working only with the 1th value. This loop evaluates the switch for each of the values and keeps the C or F value accordingly. As our data leaves the loop, the tunnel assembles the results into a new array and sends it on to the Waveform Chart. This is the *For loop tunnel’s* default behavior. The thermometer however can only show a single value so here we have disabled the indexing at the tunnel leading to the thermometer. Indexing is a property of the tunnel. Right click the tunnel itself to modify the property.

6. Go to front panel and run the program and observe the results.
 - a. Although this VI works, there is no point in evaluating the switch each time through the loop. It is not desirable to change the state of the switch during the loop nor is it humanly possible without a time delay. We can pull it out of the loop.
 - b. There is another undesirable effect. In order to satisfy the thermometer’s need for a single value, we have discarded all but the last value calculated in the loop. Right click on the broken wire leading to the thermometer and select **insert>>All_functions>>Analyze>>Mathematics>>probability_and_Statistics>>Mean.vi**



You then see:

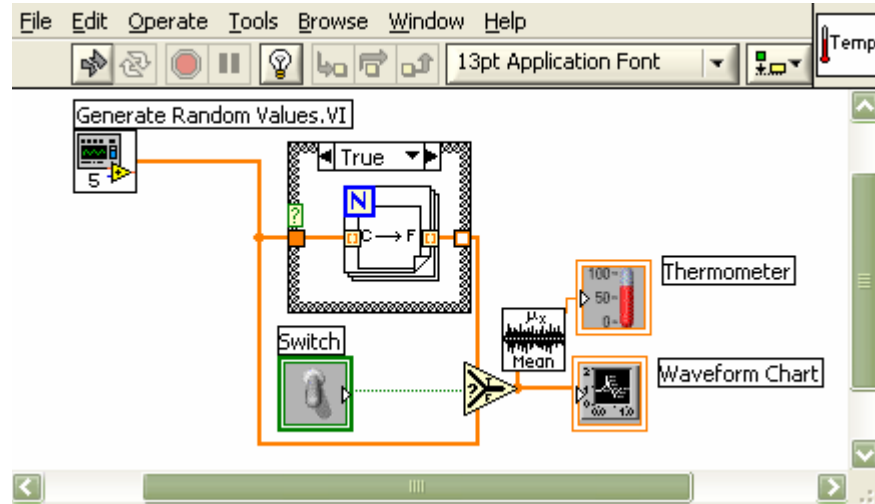
Notice that we are still evaluating the C to F conversion for each value regardless of the switch setting. Lets make a little room around our loop and stretch a case





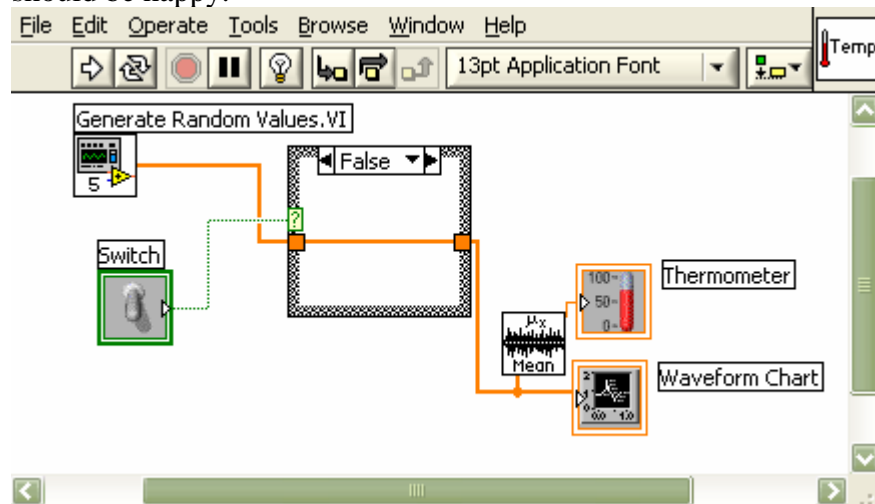
structure around the loop.

This will break the VI,

but we'll fix it.



The select switch  is no longer needed, but the tunnel out of the case structure must be satisfied. Hook the switch to the conditional terminal , stretch a wire across the false case between the tunnels and reconnect the output tunnel from the case to the chart. Now hit CTRL-B to clear the broken wires and your VI should be happy.



Now when the switch is false, it makes no conversion and when the switch is true it converts C to F. What's wrong with that?

Create an icon, as shown below, so you can use this as a subVI later. Also provide the connectors at the input or output as required.



7. Select **File>Save** to save the VI as Thermometer.VI.
8. Select **File>Close** to close the VI.

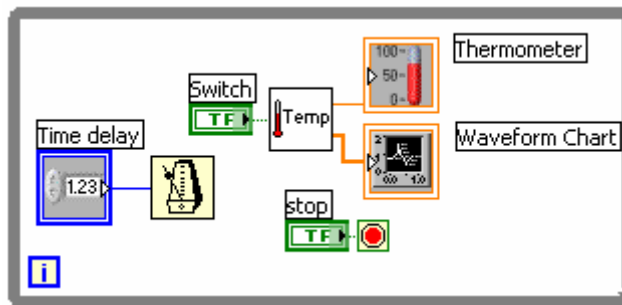
LabVIEW Example #6:

Objective


To build a VI to monitor the temperature continuously

Block Diagram

1. Move over to Block diagram
2. Place a 'while' loop on the block diagram from 'all functions-structures palette'
 - Place the 'Thermometer.VI' on the diagram.
 - Create an indicator for chart, and thermometer, and a control for switch.
 - Provide a 500 ms delay using 'wait until next ms multiple' function




3. Run it from the front panel. It plots the temperature values on the chart, as long as the stop button has not been hit.
4. Save this as 'temperature monitor.vi'

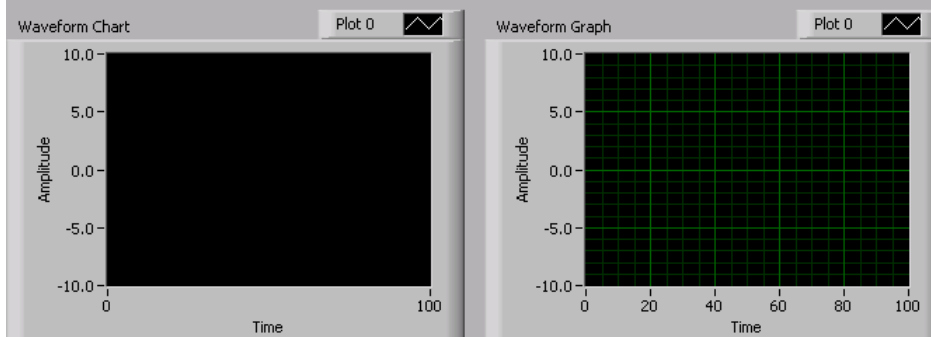


Graphs and Charts

Use graphs and charts to display plots of data in a graphical form.



Graphs and charts differ in the way they display and update data. VIs with graphs require an array of data. In contrast, a chart appends new data points to a history buffer. On a chart, you can see the current reading or measurement in context with data previously acquired. Chart takes updates the information displayed on the chart each iteration.



The graphs and charts located on the **Controls»Graph** palette include the following types:

- **Waveform Chart and Graph**—Displays data acquired at a constant rate.
- **XY Graph**—Displays data acquired at a non-constant rate, such as data acquired when a trigger occurs.
- **Intensity Chart and Graph**—Displays 3D data on a 2D plot by using color to display the values of the third dimension.

You can customize the appearance of graphs and charts by showing or hiding options. Right-click the graph or chart and select **Visible Items** from the shortcut menu to display or hide the following options:

- **Plot Legend**—Defines the color and style of the plot(s). Resize the legend to display multiple plots.
 - **Scale Legend**—Defines labels for scales and configures scale properties.
 - **Graph Palette**—Changes scaling and formatting while a VI is running.
 - **X Scale and Y Scale**—Formats the x- and y-scales.
 - **Cursor Legend (graph only)**—Displays a marker at a defined point coordinate. You can display multiple cursors on a graph.
-

