



# NI LabView

**READ THIS DOCUMENT CAREFULLY AND FOLLOW THE INSTRUCTIONS IN THE EXERCISES**

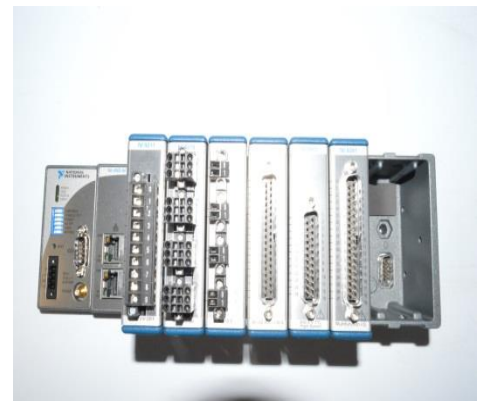
## Introduction

According to National Instruments description: *“LabVIEW is a graphical programming platform that helps engineers scale from design to test and from small to large systems. It offers unprecedented integration with existing legacy software, IP, and hardware while capitalizing on the latest computing technologies. LabVIEW provides tools to solve today’s problems—and the capacity for future innovation—faster and more effectively”* (National Instruments, 2013)

NI LabVIEW software is the foundation of the National Instruments products. The software integrates a huge number of tools necessary for building any measurement, data acquisition or control application. LabVIEW software makes easier the integration software- hardware bringing together a flexible and friendlier platform to measure and control systems on time and cost friendly.

## CompactRIO

CompactRIO is a reconfigurable embedded control and acquisition system. The CompactRIO system’s rugged hardware architecture includes I/O modules, a reconfigurable FPGA chassis, and an embedded controller. Additionally, CompactRIO is programmed with NI LabVIEW graphical



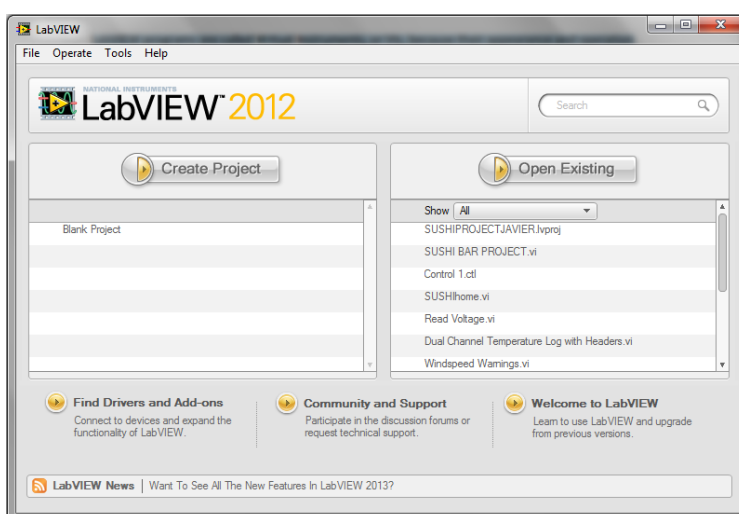
## LABVIEW Environment

LabVIEW (short for Laboratory Virtual Instrumentation Engineering Workbench) is a platform and development environment for a visual programming language from National Instruments. The graphical language is named "G". Originally released for the Apple Macintosh in 1986, LabVIEW is commonly used for data acquisition, instrument control, and industrial automation on a variety of platforms including Microsoft Windows, UNIX, Linux, and Mac OS X. The latest version of LabVIEW is version LabVIEW 2013. Visit National Instruments at [www.ni.com](http://www.ni.com).

LabVIEW contains a comprehensive set of tools for acquiring, analysing, displaying, and storing data, as well as tools to help you troubleshoot your code. The code files have the extension “.vi”, which is an abbreviation for “Virtual Instrument”.

LabVIEW programs are called Virtual Instruments, or VIs, because their appearance and operation imitate physical instruments. LabVIEW contains a comprehensive set of tools for acquiring analysing, displaying, and storing data, as well as tools to help you troubleshoot your code.

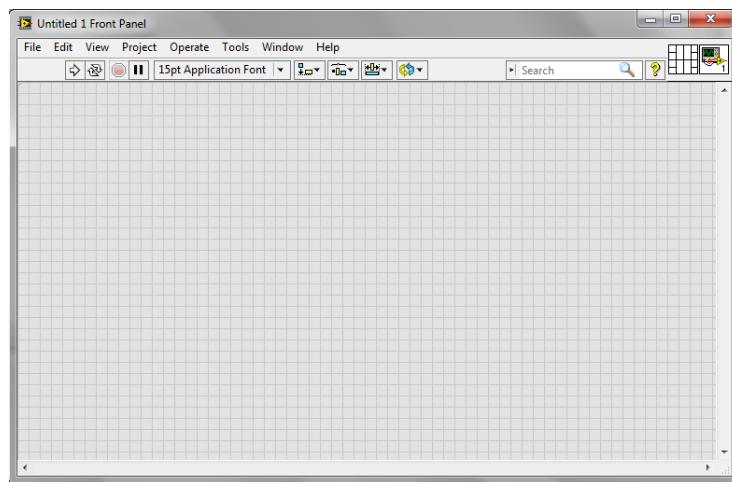
When opening LabVIEW, you first come to the “Getting Started” window.



In order to create a new LabVIEW project, select “Create project” and then select “Blank VI”, an untitled front panel window appears. This window displays the front panel; this is one of the two LabVIEW windows you use to build a VI. The other window contains the block diagram.

## Front Panel

When you have created a new VI or selected an existing VI, the Front Panel and the Block Diagram for that specific VI will appear



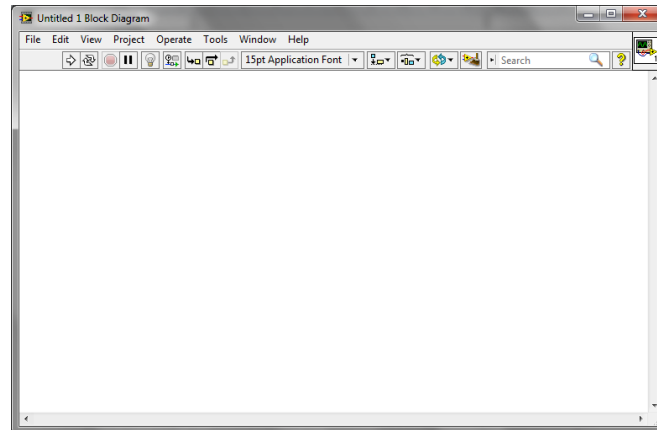
In LabVIEW, you build a user interface, or front panel, with controls and indicators. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays.

You build the front panel with controls and indicators, which are the interactive input and output terminals of the VI, respectively. Controls are knobs, push buttons, dials, and other input devices. Indicators are graphs, LEDs, and other displays. Controls simulate instrument input devices and supply data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.

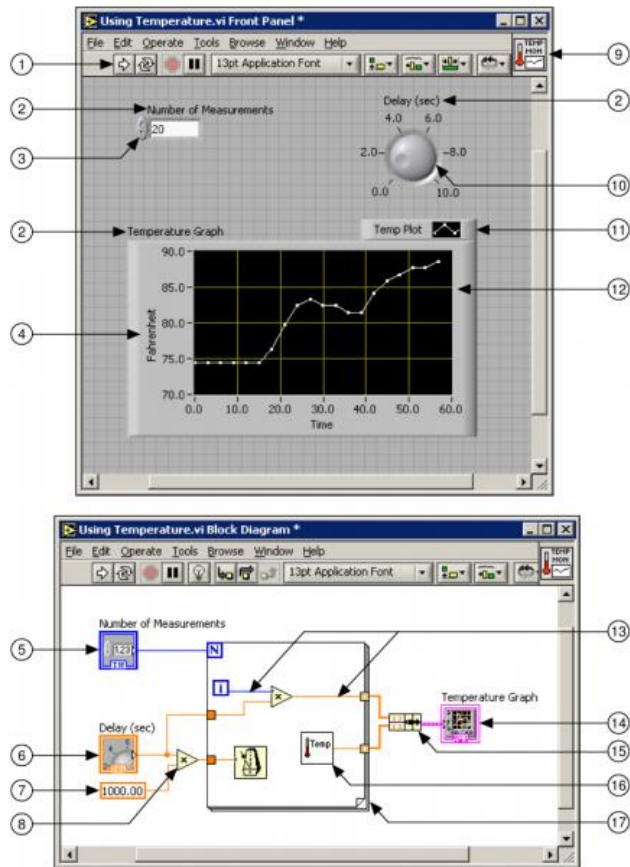
## The Block Diagram

After you build the user interface, you add code using VIs and structures to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

After you build the front panel, you add code using graphical representations of functions to control the front panel objects. The block diagram contains this graphical source code. Front panel objects appear as terminals, on the block diagram. Block diagram objects include terminals, sub VIs, functions, constants, structures, and wires, which transfer data among other block diagram objects.



The Figure below shows a front panel and its corresponding block diagram with front panel and block diagram components.



The different components are as follows:

- |                    |                      |                        |
|--------------------|----------------------|------------------------|
| 1. Toolbar         | 7. Numeric Constant  | 13. Wire Data Path     |
| 2. Owned Label     | 8. Multiply Function | 14. XY Graph Terminal  |
| 3. Numeric Control | 9. Icon              | 15. Bundle Function    |
| 4. Free Label      | 10. Knob Control     | 16. Sub VI             |
| 5. Numeric Control | 11. Plot Legend      | 17. For Loop Structure |
| Terminal           | 12. XY Graph         |                        |
| 6. Knob Terminal   |                      |                        |

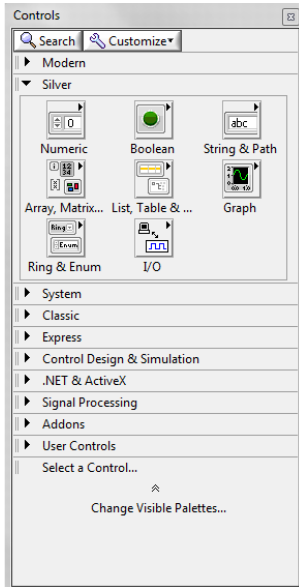
## 2.4 LABVIEW PALETTES:

### Controls Palette

The Controls and Functions palettes contain sub palettes of objects that you can use to create a VI. When you click a sub palette icon, the entire palette changes to the sub palette you selected. To use an object on the palettes, click the object and place it on the front panel or

block diagram. The Controls palette is available only on the front panel. The Controls palette contains the controls and indicators you use to build the front panel.

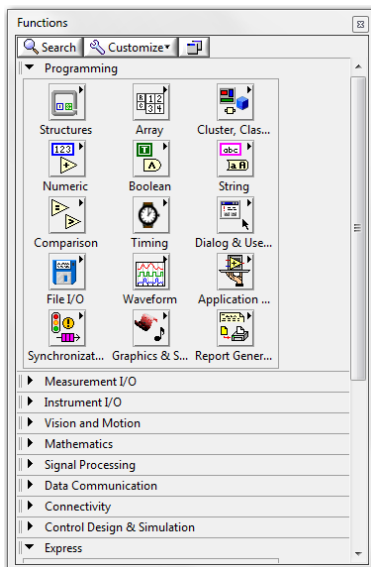
To get the control pallet just click on the right button of the mouse on the front panel.



## Function Palette

The Functions palette is available only on the block diagram. The Functions palette contains the VIs and functions you use to build the block diagram.

To get the Function pallet just click on the right button of the mouse on the Block Diagram.

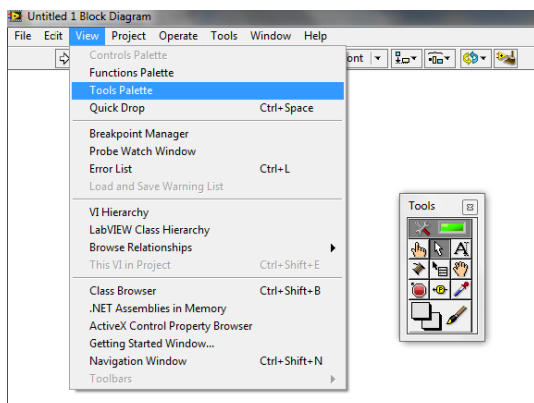


## Tools Palette

You can create, modify, and debug VIs using the tools located on the floating Tools palette. The Tools palette is available on both the front panel and the block diagram. A tool is a special operating mode of the mouse cursor. The cursor corresponds to the icon of the tool selected in the Tools palette. Use the tools to operate and modify front panel and block diagram objects.

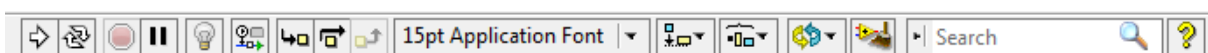


The tools palette can be accessed holding the “shift” key and the right button of the mouse or from the VI menu



## 2.5 Toolbar

Below we see the LabVIEW Toolbar:



The behaviours of the different buttons are as follows:



Click the Run button to run a VI. LabVIEW compiles the VI, if necessary. You can run a VI if the Run button appears as a solid white arrow. The solid white arrow, shown above, also indicates you can use the VI as a sub VI if you create a connector pane for the VI.



While the VI runs, the Run button appears as shown at left if the VI is a top-level VI, meaning it has no callers and therefore is not a sub VI.



If the VI that is running is a sub VI, the Run button appears as shown at left.

The Run button appears broken, shown at left, when the VI you are creating or editing contains errors. If the Run button still appears broken after you finish wiring the block diagram, the VI is broken and cannot run. Click this button to display the Error list window,



which contains all errors and warnings.



Click the Run Continuously button, shown at left, to run the VI until you abort or pause execution. You also can click the button again to disable continuous running.

While the VI runs, the Abort Execution button, shown at left, appears. Click this button to stop the VI immediately if there is no other way to stop the VI. If more than one running top-level VI uses the VI, the button is dimmed.

Note: Avoid using the Abort Execution button to stop a VI. Either let the VI complete its data flow or design a method to stop the VI programmatically. By doing so, the VI is at a known state. For example, place a button on the front panel that stops the VI when you click it.



Click the Pause button, shown at left, to pause a running VI. When you click the Pause button, LabVIEW highlights the location of the block diagram where you halted the execution, and the Pause button appears red. Click the button again to continue running the VI.



Highlight execution, use this function to slow down the data flow and check the data running through the code. You can use it to debug the code and edit any error.



## Controls and Functions Palettes

The Controls and Functions palettes contain sub palettes of objects you can use to create a VI. When you click a sub palette icon, the entire palette changes to the sub palette you selected. To use an object on the palettes, click the object and place it on the front panel or block diagram.

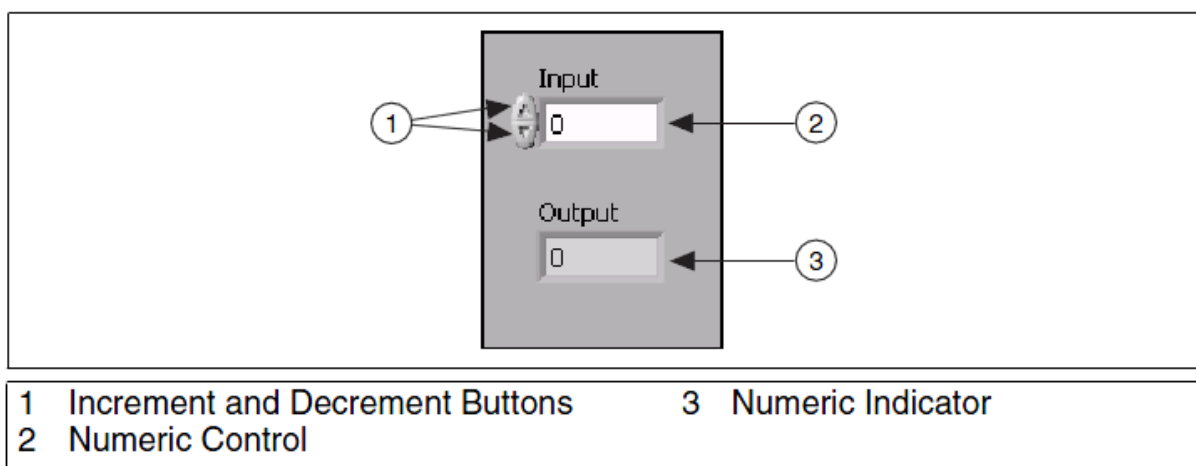
## Controls Palette

The Controls palette is available only on the front panel. The Controls palette contains the controls and indicators you use to create the front panel.

Select Window» Show Controls Palette or right-click the front panel workspace, it displays the Controls palette. Tack down the Controls palette by clicking the thumbtack on the top left corner of the palette. By default, the Controls palette starts in the Express view.

## Numeric Controls and Indicators

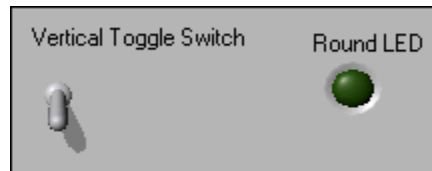
The two most commonly used numeric objects are the numeric control and the numeric indicator, as shown in the following illustration.



To enter or change values in a numeric control, click the increment and decrement buttons with the Operating tool or double-click the number with either the Labelling tool or the Operating tool, type a new number, and press the <Enter> key.

## Boolean Controls and Indicators

Use Boolean controls and indicators to enter and display Boolean (True or False) values. Boolean objects simulate switches, push buttons, and LEDs. The most common Boolean objects are the vertical toggle switch and the round LED, as shown in the following example.



### **Express VIs, VIs, and Functions**

LabVIEW uses coloured icons to distinguish between Express VIs, VIs, and functions on the block diagram. By default, icons for Express VIs appear on the block diagram as expandable nodes with icons surrounded by a blue field. Icons for VIs have white backgrounds, and icons for functions have pale yellow backgrounds.

By default, most functions and VIs on the block diagram appear as icons that are not expandable, unlike Express VIs.

### **Express VIs**

Use Express VIs for common measurement tasks. Express VIs are nodes that require minimal wiring because you configure them with dialog boxes. You can save the configuration of an Express VI as a subVI.

### **VIs**

When you place a VI on the block diagram, LabVIEW considers the VI to be a subVI. When you double-click a subVI, its front panel and block diagram appear, rather than a dialog box in which you can configure options. The front panel includes controls and indicators. The block diagram includes wires, front panel icons, functions, possibly subVIs, and other LabVIEW objects.

The upper right corner of the front panel and block diagram displays the icon for the VI. This is the icon that appears when you place the VI on the block diagram.

You can create a VI to use as a subVI.

## Functions

Functions are the fundamental operating elements of LabVIEW. Functions do not have front panels or block diagrams but have connector panes. Double-clicking on a function only selects the that particular function.

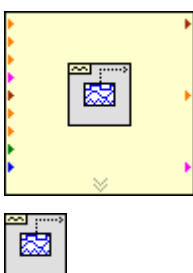
## Nodes

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

Nodes can be functions, subVIs, or structures. Structures are process control elements, such as Case structures, For Loops, or While Loops.

## Expandable Nodes versus Icons

You can display VIs and Express VIs as icons or as expandable nodes. Expandable nodes appear as icons surrounded by a coloured field. SubVIs appear with a yellow field, and Express VIs appear with a blue field. Use icons, such as the Basic Function Generator VI icon shown at left, if you want to conserve space on the block diagram. Use expandable nodes, such as the Basic Function Generator VI expandable node shown at left, to make wiring easier and to aid in documenting block diagrams. By default, subVIs appear as icons on the block diagram, and Express VIs appear as expandable nodes.

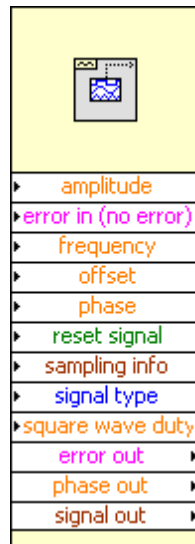


To display a subVI or Express VI as an expandable node, right-click the subVI or Express VI and select **View As Icon** from the shortcut menu to remove the checkmark. You can resize the expandable node to make wiring even easier, but it also takes a large amount of space on the block diagram. Complete the following steps to resize a node on the block diagram.

1. Move the Positioning tool over the node. Resizing handles appear at the top and bottom of the node.

2. Move the cursor over a resizing handle to change the cursor to the resizing cursor.
3. Use the resizing cursor to drag the border of the node down to display additional terminals.
4. Release the mouse button.

To cancel a resizing operation, drag the node border past the block diagram window before you release the mouse button. The following figure shows the Basic Function Generator VI as a resized expandable node.



## Terminals

Front panel objects appear as terminals on the block diagram. The terminals represent the data type of the control or indicator. You can configure front panel controls or indicators to appear as icon or data type terminals on the block diagram. By default, front panel objects appear as icon terminals. For example, a knob icon terminal, shown at left, represents a knob on the front panel.



The DBL at the bottom of the terminal represents a data type of double-precision, floating-point numeric. To display a terminal as a data type on the block diagram, right-click the terminal and select **View as Icon** from the shortcut menu to remove the checkmark. A DBL data type terminal, shown at left, represents a double-precision, floating-point numeric control or indicator.








## Wires

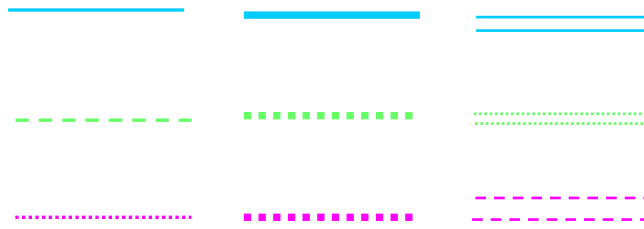
Data is transferred among block diagram objects through wires. Wires are similar to variables in text-based programming languages. In Figure 1-9, wires connect the control and indicator terminals to the Add and Subtract functions. Each wire has a single data source, but you can wire it to many VIs and functions that read the data. Wires are different colours, styles, and thicknesses, depending on their data types. A broken wire appears as a dashed black line with a red X in the middle. The following examples are the most common wire types.

In LabVIEW, you use wires to connect multiple terminals together to pass data in a VI. The wires must be connected to inputs and outputs that are compatible with the data that is transferred with the wire. For example, you cannot wire an array output to a numeric input. In addition the direction of the wires must be correct. The wires must be connected to only one input and at least one output. For example, you cannot wire two indicators together. The components that determine wiring compatibility include the data type of the control and/or indicator and the data type of the terminal.

## DATA TYPES

-  **Numeric:** This data type contains either an integer or floating point numbers
-  **Boolean:** This data type contains logical combinations of TRUE/FALSE, AND, OR, NOT, Etc.
-  **String:** Contains alpha-numeric characters only
-  **Cluster.** Encloses several data types. Cluster data types appear brown if all elements in the cluster are numeric or pink if all elements of the cluster are of different types.
-  These data types can be in either Scalar, 1D Array or 2D Array form.

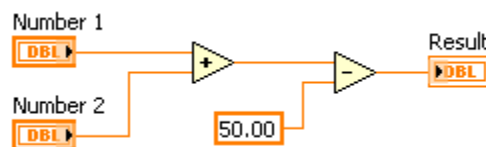
	Scalar	1D array	2D array	Color
Floating-point number				orange
Integer number				blue
Boolean				green
String				pink



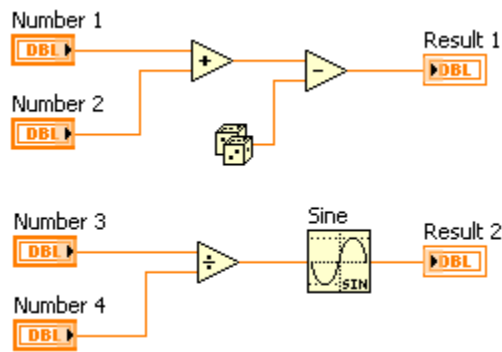
## Dataflow Programming

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when all its inputs are available. When a node completes execution, it supplies data to its output terminals and passes the output data to the next node in the dataflow path.

Visual Basic, C++, JAVA, and most other text-based programming languages follow a control flow model of program execution. In control flow, the sequential order of program elements determines the execution order of a program. For a dataflow programming example, consider a block diagram that adds two numbers and then subtracts 50.00 from the result of the addition. In this case, the block diagram executes from left to right, not because the objects are placed in that order, but because the Subtract function cannot execute until the Add function finishes executing and passes the data to the Subtract function. Remember that a node executes only when data are available at all of its input terminals, and it supplies data to its output terminals only when it finishes execution.



In the following example, consider which code segment would execute first—the Add, Random Number, or Divide function. You cannot know because inputs to the Add and Divide functions are available at the same time, and the Random Number function has no inputs. In a situation where one code segment must execute before another and no data dependency exists between the functions, use other programming methods, such as error clusters, to force the order of execution.



## Help

The Context Help window (Ctrl +H) displays basic information about LabVIEW objects when you move the cursor over each object. The Context Help window is visible by default. To toggle display of the Context Help window, select Help-Show Context Help, press the Ctrl-H keys, or click the Show Context Help Window button on the toolbar. When you move the cursor over front panel and block diagram objects, the Context Help window displays the icon for subVIs, functions, constants, controls, and indicators, with wires attached to each terminal. When you move the cursor over dialog box options, the Context Help window displays descriptions of those options. In the window, required connections are bold, recommended connections are plain text, and optional connections are dimmed or do not appear. The Figure below shows an example of the Context Help window.

# Exercises

## Exercise 1 (Convert C to F)

Create VI that take a numeric value representing degrees Celsius and Convert it to degrees Fahrenheit.



### List of functions



#### **Add**

Function palette > Numeric > Add

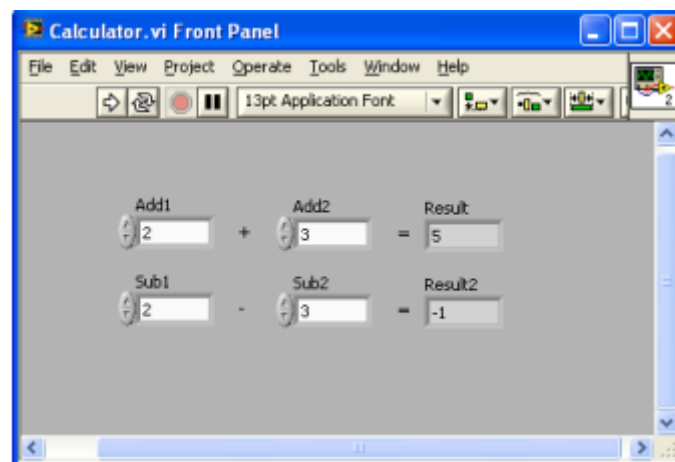


#### **Multiply**

Function palette > Numeric > Add

## Exercise 2: Create a simple Calculator

Create a simple calculator that Add and Subtract 2 numbers like this:



Start the program with the Run button.

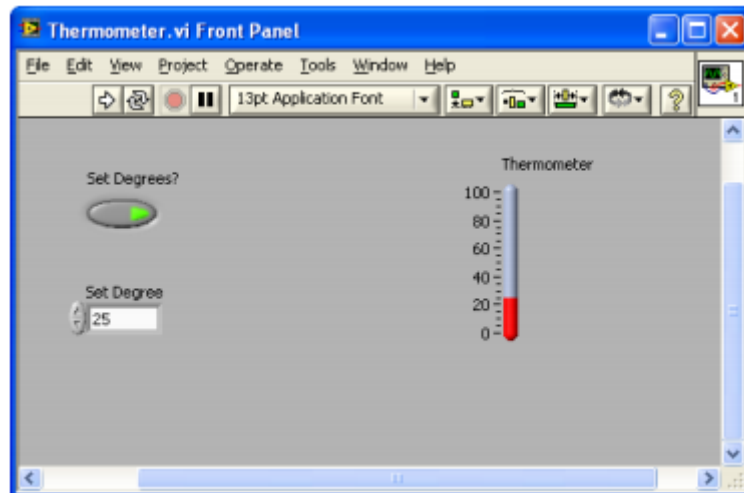


### Exercise 3: Create a Thermometer with using a Selector (VI)

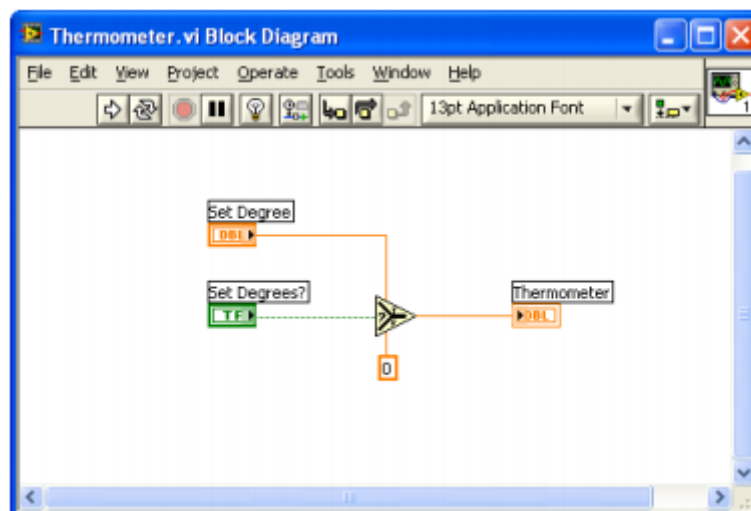
Create a simple LabVIEW application (VI) with a Front Panel with some Controls and Indicators.


Create the logic by connecting the Terminals on the Block Diagram.

The Front Panel could look something like this:

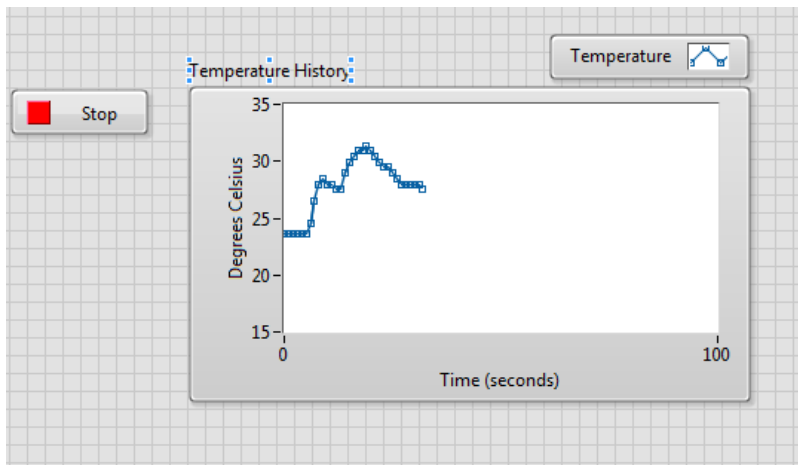
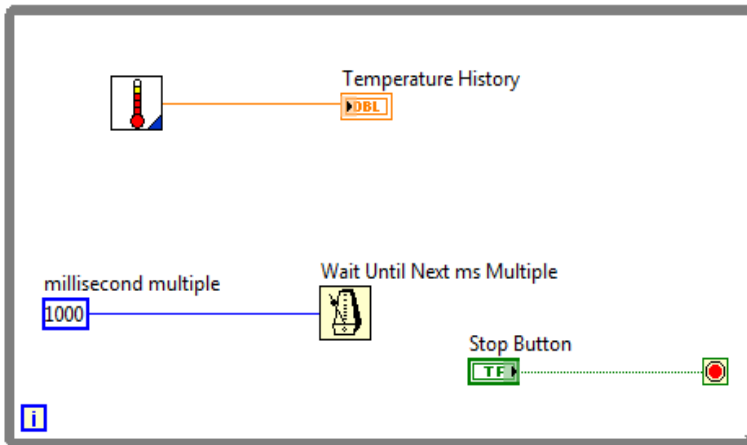


The Block Diagram could look something like this:



 Start the program with the Run button.

**Exercise 4: Create a Temperature History monitoring using with While loop.**



**Exercise 5: Create below VI**

Create a VI which its output is random number between 0-100 when a selector key is on and 0 when the selector key is off.

Hints: Use Case Structure.

## **Source:**

Labview Basic introduction Course manual:

<http://www.unife.it/ing/informazione/Sistemi-acquisizione-dati/manuale-di-programmazione-labview/manuale-labview-8-0-inglese>

List of tutorials for LabVIEW

[http://labviewwiki.org/LabVIEW\\_tutorial](http://labviewwiki.org/LabVIEW_tutorial)

Tutorial labVIEW (upscale)

<http://www.upscale.utoronto.ca/GeneralInterest/LabView.html>