Lab VIEW

LabVIEW is short for **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench. LabVIEW is used by engineers and scientists in research, development, testing, and production in PC controlled equipments and machinery. We intend to briefly introduce LabVIEW to enable students of easily navigate through the software environment. Discussed here will be LabVIEW running on Windows operating system.

The Startup Screen:

When one launches LabVIEW either by double-clicking on its icon on the desktop or from Start/All Programs/National Instruments/LabVIEW/LabVIEW8.X, a startup window launches as shown below. Two important functions are Open/Blank VI (VI stands for Virtual Instrument) and Find Examples. The first Open/Blank VI is used which is used to open a new VI. The Find Examples is an important function especially for beginners; it is a comprehensive library of examples. Students are encourages to navigate through the examples to learn more.

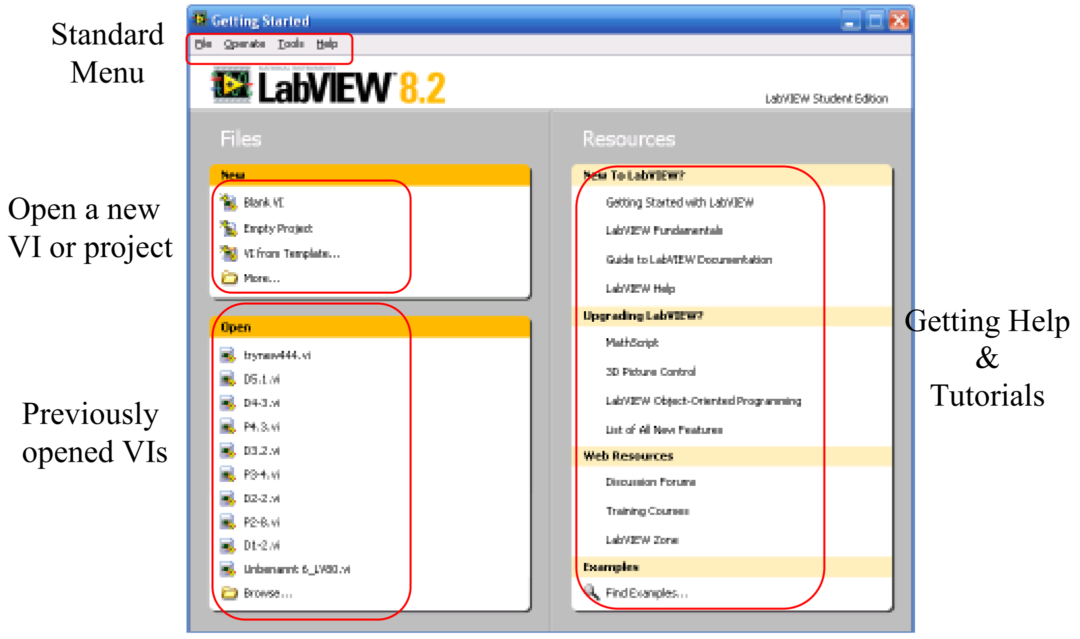


Figure 1: Startup Window

Front Panel and Block Diagram Windows:

When you create a new VI, LabVIEW opens two windows Front Panel and Block Diagram windows. As implied by their names, the Front panel is the user interface to the code. It is where the users pass in inputs and observe outputs. One can think of the Front Panel window as the monitor of your PC. On the other hand, the Block Diagram window is where the code resides. Since LabVIEW is mainly considered as graphical programming languages, functions, loops and analysis tools are represented by blocks wired together with soft wires (lines). These soft wires running between block depends on the data types and will be discussed in more details later.

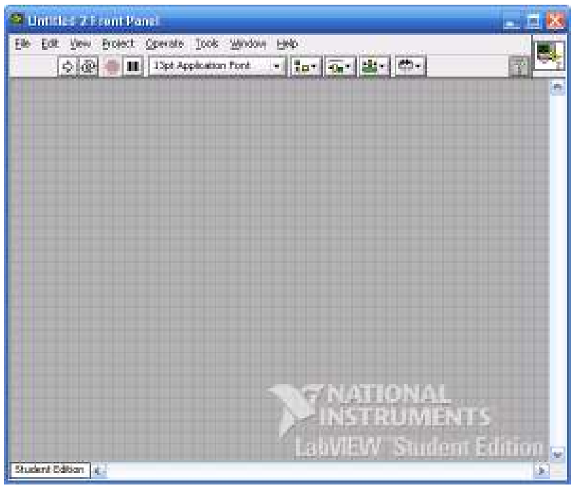


Figure 2: Front Panel Window

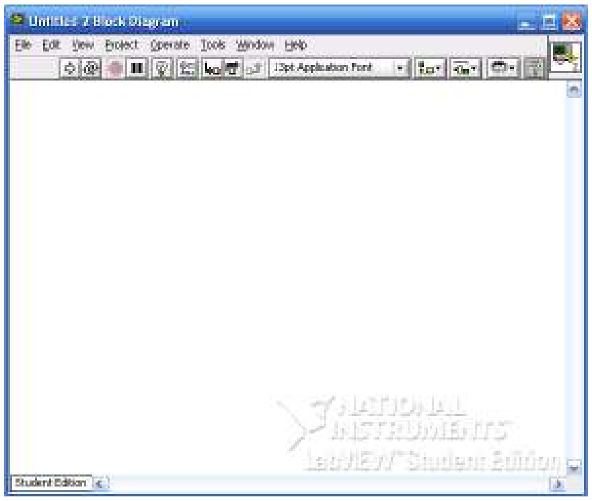


Figure 3: Block Diagram Window

LabVIEW has a very important help tool that it is recommend it to turn it on before using the software. Go to Help/Show Context Help (Ctrl+H), it will show Context Help window that displays each block brief help information as soon you pass mouse over the block.

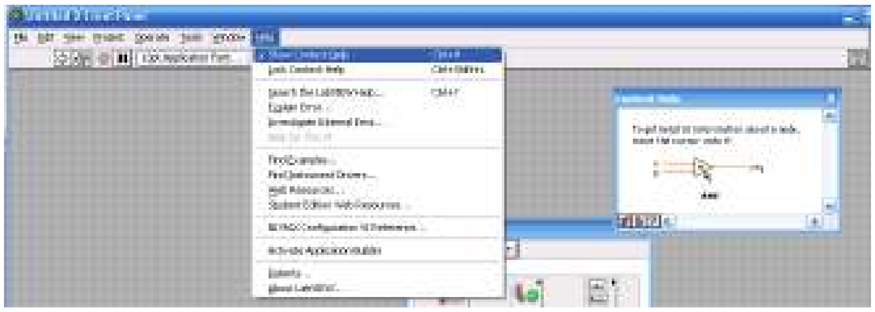


Figure 4: How to open Context Help

Palettes:

In LabVIEW, there are three main palettes you need to get familiar with from which you will access controls, functions and tools.

Tools Palette:

A tool is a special operating mode of the mouse cursor. By default the Tools Palette is hidden (not visible). To show the Tool Palette, select Tool Palette from View menu on the main tool bar.

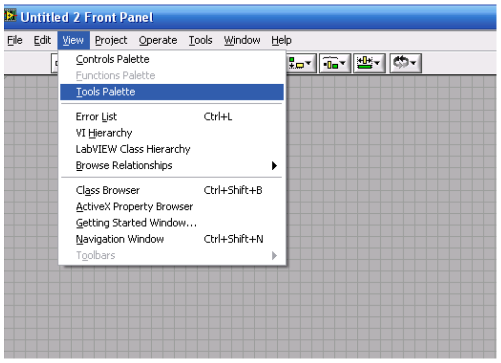


Figure 5: Displaying Tool Palette

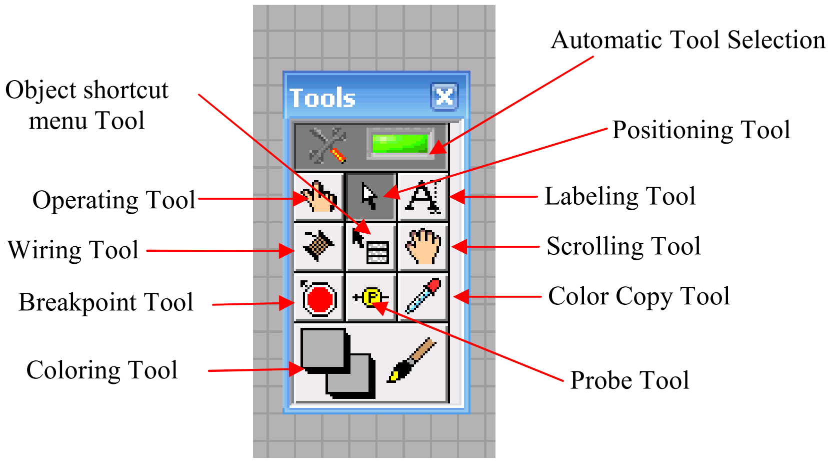


Figure 6: Tools Palette

When the Automatic Tool Selection button is enabled, LabVIEW automatically selects the corresponding tool from the Tools Palette as you move the cursor over the objects on either the front panel or the block diagram.

Controls Palette:

If the Controls Palette is not shown, you can right-click on an open area in the front panel window. The Controls are located on sub-palettes based on the type of the control. It includes but not limited to Modern, System, Classic, Express and Users Defined.

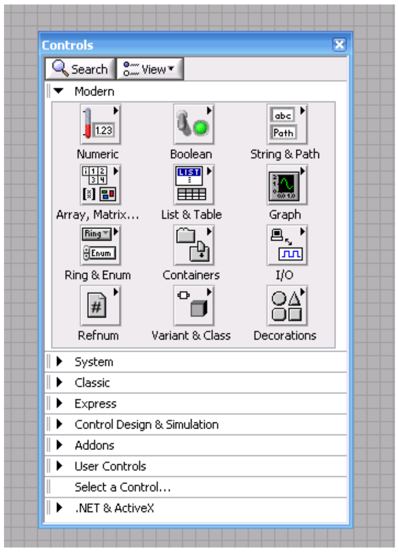


Figure 7: Controls Palette

Then each sub-palette is divided into categories. For example, Modern sub-palette is divided into Numeric, Boolean, String&Path etc.

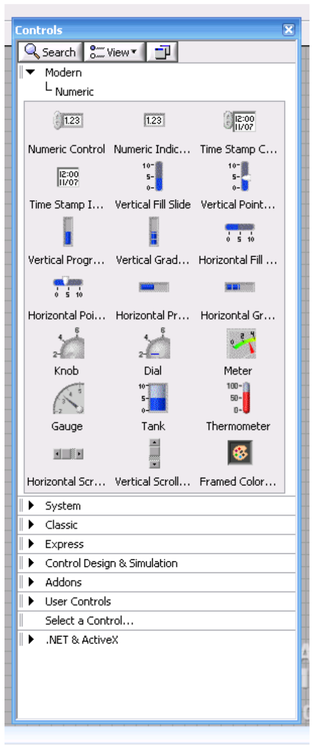


Figure 8: Modern sub-palette items

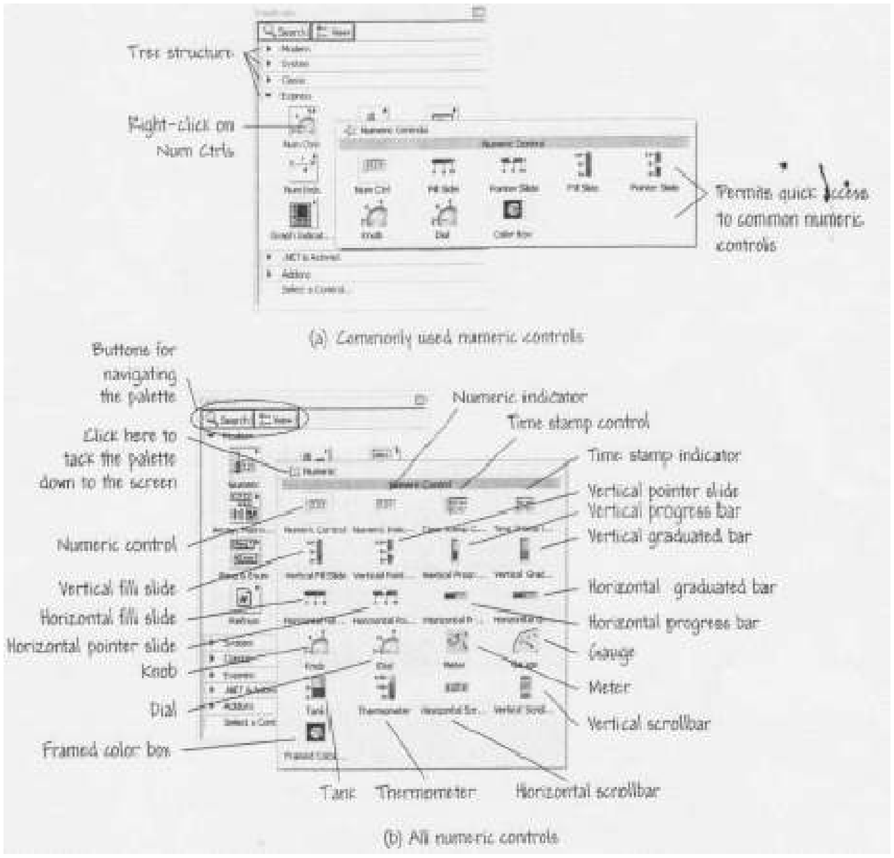


Figure 9: Numeric controls and indicator subpalette

Functions Palette:

The Functions Palette is similar to the Controls Palette but it is accessible from the block diagram. It contains all function libraries from simple mathematics to advanced analysis tools. Some of the sub-palettes included Programming, Measurements I/O, Instruments I/O, Mathematics and Signal Processing.

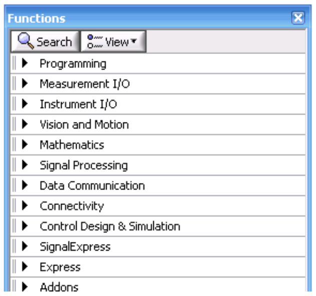


Figure 10: Function Palette

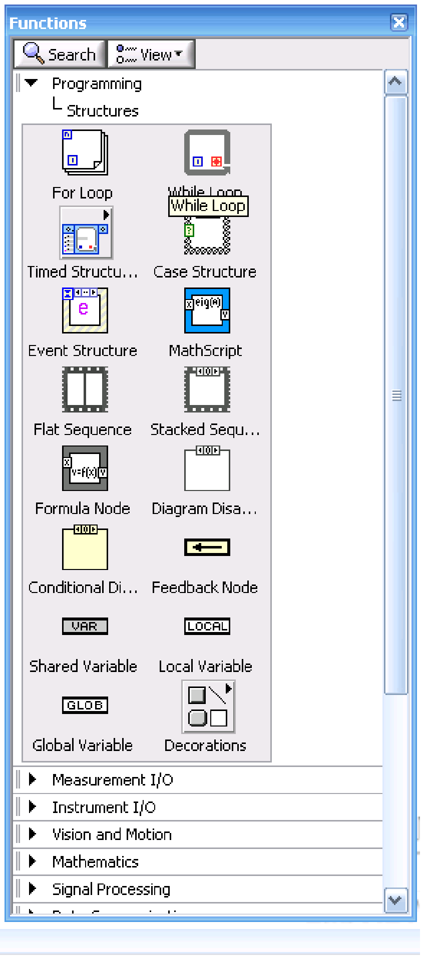


Figure 11: Functions Palette sub-palette items

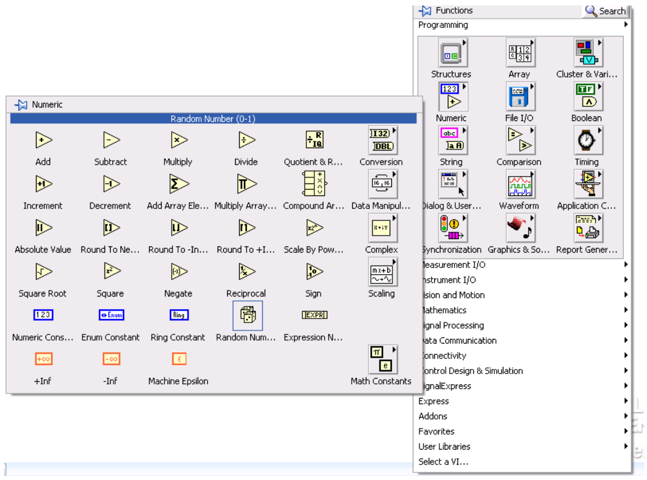


Figure 12: Selecting random number from Function palette

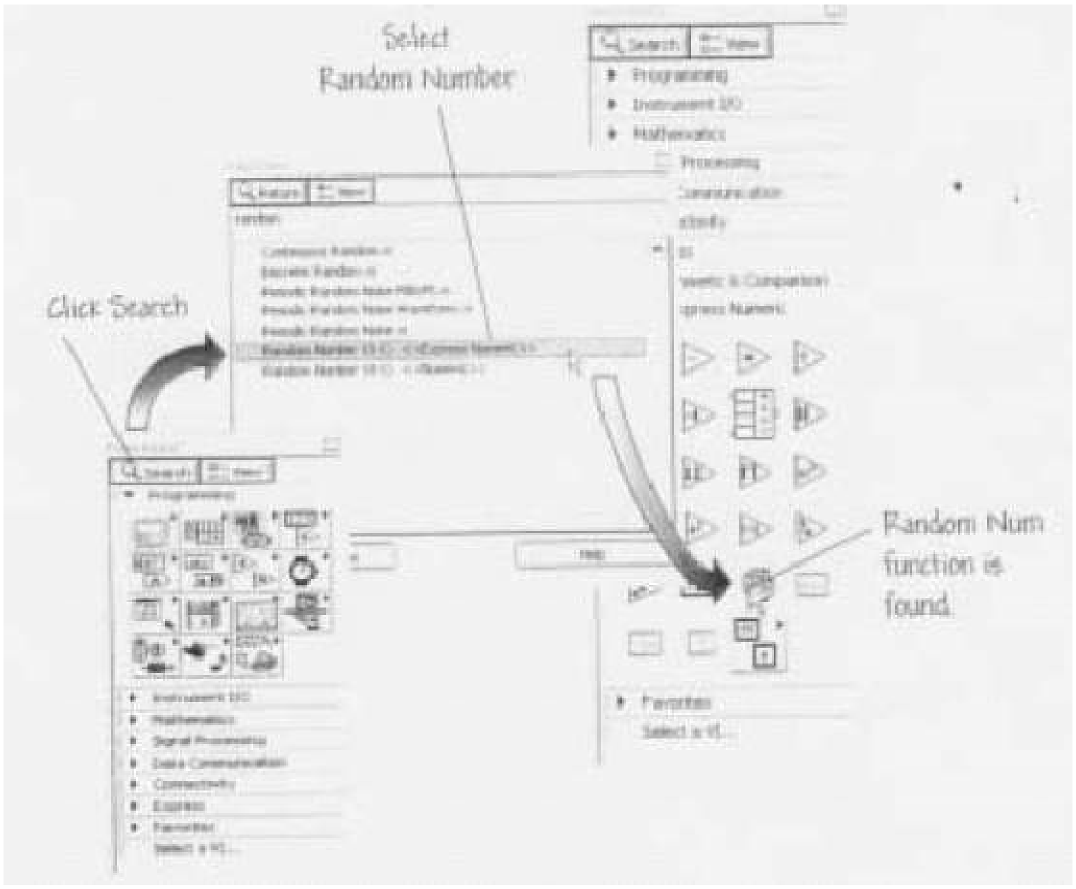


Figure 13: Searhing Function palette for random number

A common feature between the Functions Palette and Controls Palette is the search option indicated by the magnifying glasses. Use these features whenever you can't find a control or function.

Configuring Function Palette

Click on the **View Button**, near the upper right corner of the Function Palette. In the menu that appears, select **Change Visible Categories/Select All** then press the **OK** button.

Click on the **View Button** again, this time, in the menu, click on **Options/Controls/Functions Palette**, then in the **Format** box, choose **Category(Standard)**, then press **OK** button.

Downloading leaning Directory:

A set of examples beside the one supplied by LabVIEW upon installation are provided when you purchase LabVIEW8.X Student Edition book by Robert Bishop (ISBN: 0-13-199918-4). These examples are complied in a Learning Directory, which you can down load from http://www.prenhall.com/bishop. Please download this folder because we will refer to some the examples listed in this folder.

Working with Controls:

1- Numeric Control:

Used to pass numerical inputs to the code. You can access the control form Controls/Modern/Numeric/Numeric Control.

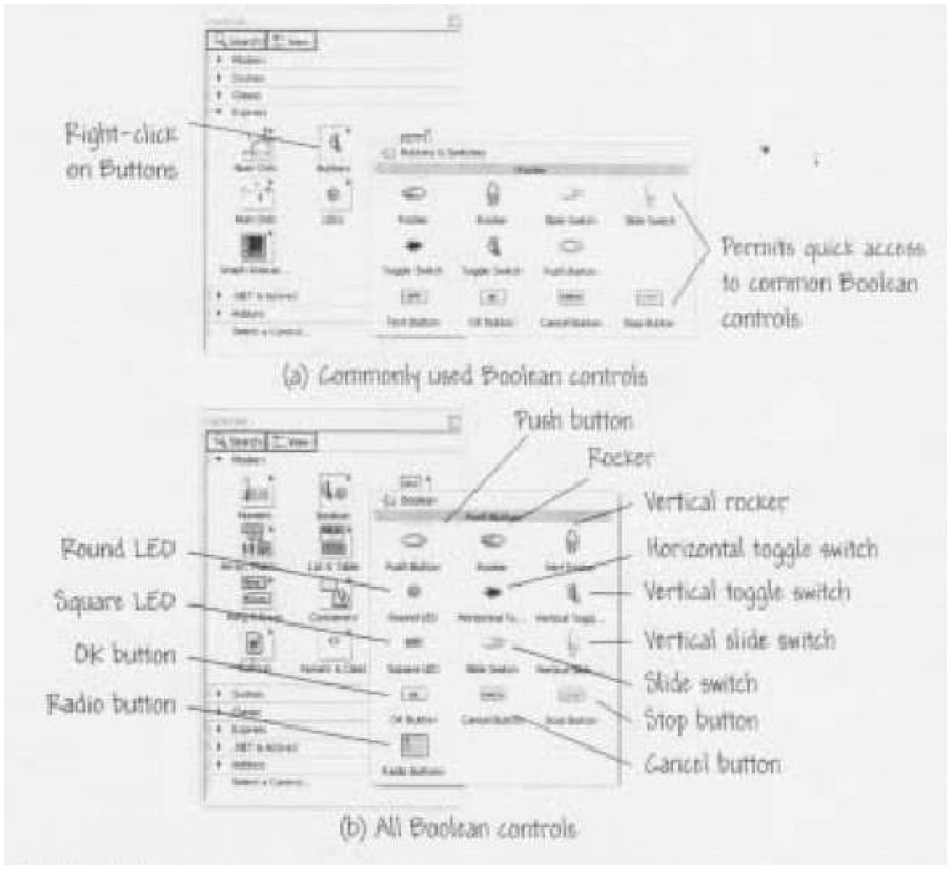
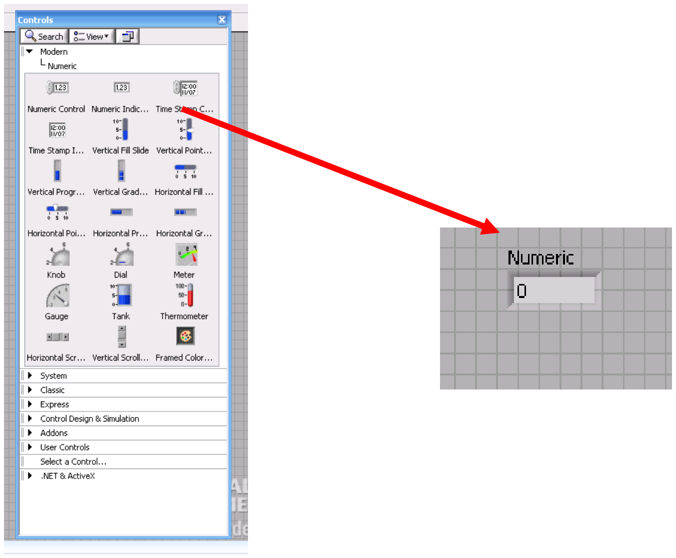
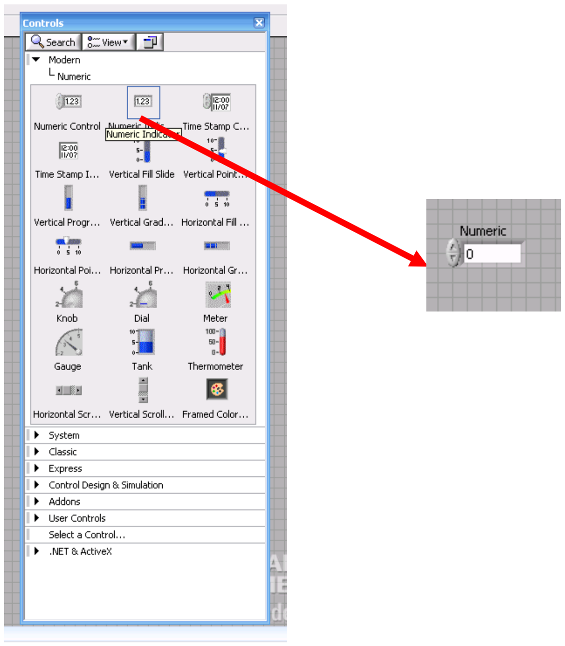


Figure 14: Boolean controls and indicators



2- Numeric Indicator:

Used to display numerical outputs. You can access the indicator from Controls/Modern/Numeric/Numeric Indicator. Changing the properties of the numeric indicator can be done by right-clicking the indicator and selecting Properties from the shortcut menu.



3- Graphs

You can access graphs options from Controls/Modern/Graphs. LabVIEW includes the following types of graphs and charts:

- **Waveform Graphs and Charts**—Display data typically acquired at a constant rate.
- **XY Graphs**—Display data acquired at a non-constant rate and data for multivalued functions.
- **Intensity Graphs and Charts**—Display 3D data on a 2D plot by using color to display the values of the third dimension.
- **Digital Waveform Graphs**—Display data as pulses or groups of digital lines.
- **Mixed Signal Graphs**—Display data types accepted by waveform graphs, XY graphs, and digital waveform graphs. Also accept clusters that contain any combination of those data types.
- **3D Graphs**—Display 3D data on a 3D plot in an ActiveX object on the front panel.

Shortcut: right-click any button on the control palette permits quick access to common controls.

Wiring:

Wires are connection between functions. They are the data path between terminals and are similar to internal program variables in scripted languages. They take different shapes, some of which is shown in the table below.

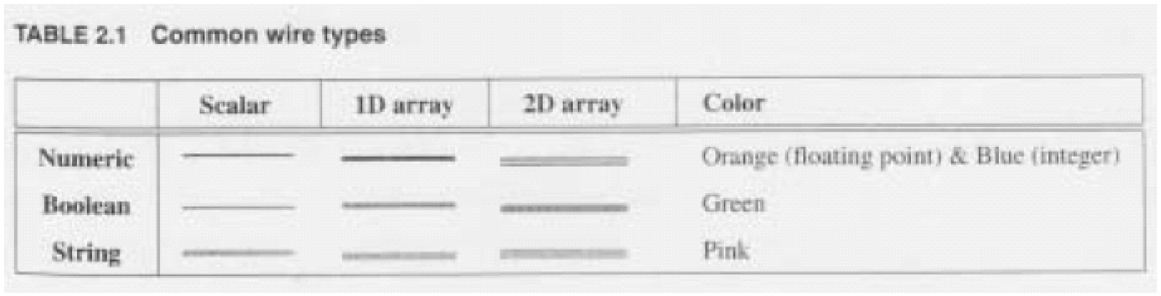


TABLE 2.1 Common wire types

	Scalar	1D array	2D array	Color
Numeric	—	━	═	Orange (floating point) & Blue (integer)
Boolean	—	━	═	Green
String	—	━	═	Pink

Tips for wiring:

1- To wire the object together with Wiring tool, click and release on the source terminal and drag the Wiring tool to the destination terminal. When the destination terminal is blinking, click and release the left mouse.
2- To identify terminals on the function, right click on the icon and select Visible Items/Terminals to see connectors. When the wiring is finished, follow the same procedure to show the icon again.

3- To bend the wires as you connect two objects, click with the left mouse button on the bend location with the Wiring tool.
4- To remove broken wires, press Ctrl+B or select Edit/Remove Broken Wires.
5- To organize and clean wires, click with the mouse button and select Clean Up Wire.

Wiring Structure problems: **(be aware)**

1- Failing to wire a tunnel in all cases of the Case Structure.
2- Overlapping tunnels.
3- Wiring underneath rather than through a structure.

<u>Data Types:</u>

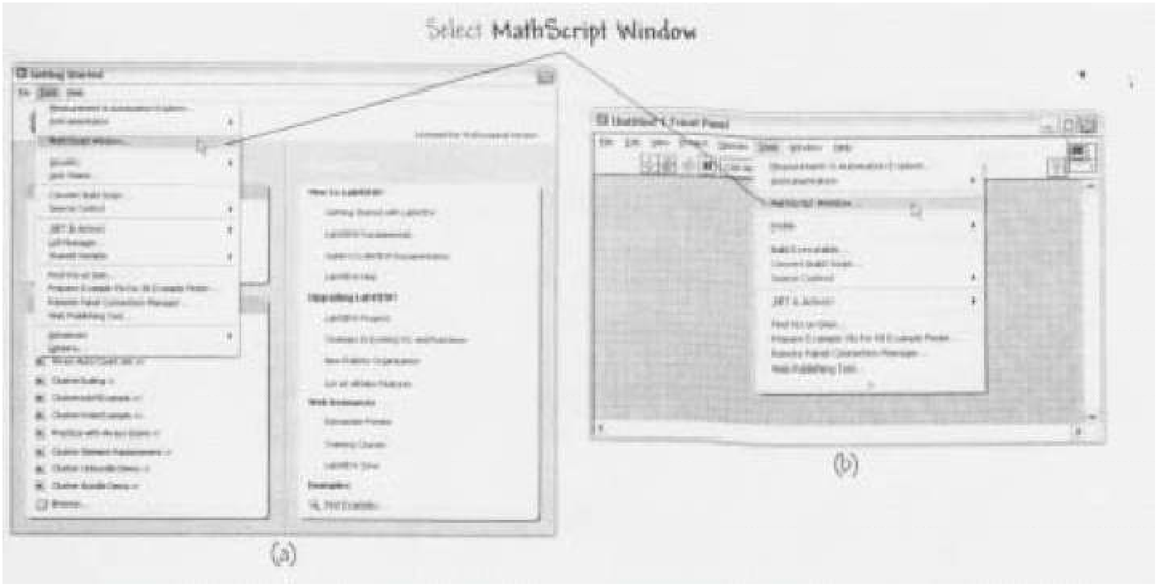
SGL	Single-precision, floating-point	32	6	Minimum positive number: 1.40e–45 Maximum positive number: 3.40e+38 Minimum negative number: –1.40e–45 Maximum negative number: –3.40e+38
DBL	Double-precision, floating-point	64	15	Minimum positive number: 4.94e–324 Maximum positive number: 1.79e+308 Minimum negative number: –4.94e–324 Maximum negative number: –1.79e+308
EXT	Extended-precision, floating-point	128	varies from 15 to 20 by platform	Minimum positive number: 6.48e–4966 Maximum positive number: 1.19e+4932 Minimum negative number: –6.48e–4966 Maximum negative number: –1.19e+4932
CSG	Complex single-precision, floating-point	64	6	Same as single-precision, floating-point for each (real and imaginary) part
CDB	Complex double-precision, floating-point	128	15	Same as double-precision, floating-point for each (real and imaginary) part
CXT	Complex extended-precision, floating-point	256	varies from 15 to 20 by platform	Same as extended-precision, floating-point for eac (real and imaginary) part
I8	Byte signed integer	8	2	–128 to 127
I16	Word signed integer	16	4	–32,768 to 32,767
I32	Long signed integer	32	9	–2,147,483,648 to 2,147,483,647
I64	Quad signed integer	64	18	–1e19 to 1e19
U8	Byte unsigned integer	8	2	0 to 255
U16	Word unsigned integer	16	4	0 to 65,535
U32	Long unsigned integer	32	9	0 to 4,294,967,295
U64	Quad unsigned integer	64	19	0 to 2e19
⌛	128-bit time stamp	<64.64>	15	Minimum time (in seconds): 5.4210108624275221700372640043497e–20

Mathscript:

Mathscript is text base math oriented with command prompt from within the LabVIEW development environment. It is similar to Mathlab environment.

TABLE 3.1 Overview of the Features of MathScript

MathScript Feature	Description
Powerful textual math	MathScript includes more than 500 functions for math, signal processing, and analysis; functions cover areas such as linear algebra, curve fitting, digital filters, differential equations, probability/statistics, and much more
Math-oriented data types	MathScript uses matrices as fundamental data types, with built-in operators for generating data, accessing elements, and other operations
Compatible	MathScript is generally compatible with the m-file script syntax used by The MathWorks, Inc. MATLAB® software, Comsol, Inc. COMSOL Script™ software, and others
Extensible	You can extend MathScript by defining your own custom functions
Part of LabVIEW	MathScript does not require additional third-party software to compile and execute



<u>Exercise #1: Building your first VI – Sine wave</u>

1- Insert a **While loop** from **Function Palette/Structures/Programming**, then select **While Loop**. The **While Loop** structure is used to control repetitive operation. By default, it will repeatedly execute the subprogram (the code inside the loop) until a Boolean values in no longer FALSE. You can change the position and size of the loop by clicking on the loop border, then resize or reposition as if you are editing an image in any word processing application.
2- When you place the While loop on the Block Diagram, two objects (items) are included in the loop (1) Stop if True conditional terminal and (2) iteration terminal (iteration number counter).
3- To generate a Sine wave plot, place a Sine icon within the loop. You can find the Sine icon in **Function Palette/Mathematics/Elementary & Special Functions/Trigonometric Functions**.

<u>Shortcuts:</u>

1- To move an object horizontally or vertically only, hold down <shift> key then drag the object.
2- To move the object in small increment, highlight the object and move the object by pressing <Arrow> keys.
3- To show the context help about any object on the block diagram or frontl panel, either press <Ctrl>+<H> or Help/Show Context Help.

4- To allow the While loop to generate the Sine wave within its border without external stimulus, go to Function Paletter/Programming/Boolean and select False Constant. Connect False Constant to Stop if True terminal of the While Loop.

<u>Shortcuts:</u>

1- To auto connect two objects together, when you select the object from the function palette bring it closer to the object to which it suppose to connect, Labview will auto connect the two objects together. For example, to connect False Constant to Stop if True (step 4 above), bring the False Constant close to Stop if True and watch LabView connecting them automatically. This is a *very handy shortcut, because LabView will also match the data-type between the two objects.*

5- We will use the iteration number as a source of ever-increasing argument x for the Sine function. So wire While Loop iteration number to the input of the Sine object. This step completes machine task, what left is how the user would see the generated Sine wave.

6- To display the Sine wave, toggle the screens to Front Panel *(a keyboard shortcut to toggle between front panel and block diagram <Ctrl+E>)*, then place a Waveform Chart, which can be found in **Controls/Modern/Graph**, then select a **Waveform Chart**.

7- You can edit the Chart's Label, by simply clicking on the label and edit it to more descriptive name.

8- Since our Sine wave is Unity amplitude Sine wave, we need to change the Y-scale to range from 1 to -1, which can be done simply by clicking on 10 (or -10) then type 1 (or -1) and press <Enter>.

9- To allow access to several useful functions that determine the scaling and labeling of the x- and y-axis, Right Click on the chart region and select Visible/Scale Legend.

10- To finish up the program, toggle back to the Block Diagram and connect the sine wave to the Waveform chart.

Suggested improvement:

a. Place **Stop Button** on the Front Panel and connect it to Stop if True terminal instead of False Constant. **Stop Button** can be found in **Controls/Modern/Boolean/Stop Butto**n.

b. Controlling iteration rate, by adding a **Wait (ms)** function and numeric constant to indicate the rate. Wait (ms) can be found in **Functions/Programming/Timing** and select **Wait**. Then to fins the constant, **Functions/Programming/Numeric**, then select **Numeric Constant**.

c. Experiment with Chart properties to learn more about available chart's features.

<u>Exercise #2:</u> (Design Problem #D1.2 from [1])

Open new VI. Go to the block diagram and place a Compound Arithmetic function, found in the Programming/Numeric subpalette. By default this function returns the sum of two inputs.

(a) Using LabVIEW Context Help, find out how to configure the function to compute other operations besides addition.
(b) Using what you learned in Step (a), condifure the function to multiply two inputs.
(c) Using LabVIEW Context help, find out how to change the Compound Arithmetic function to compute operations on more than two inputs.
(d) Using what you learned in Step (c), configure the function to multiply three inputs together.