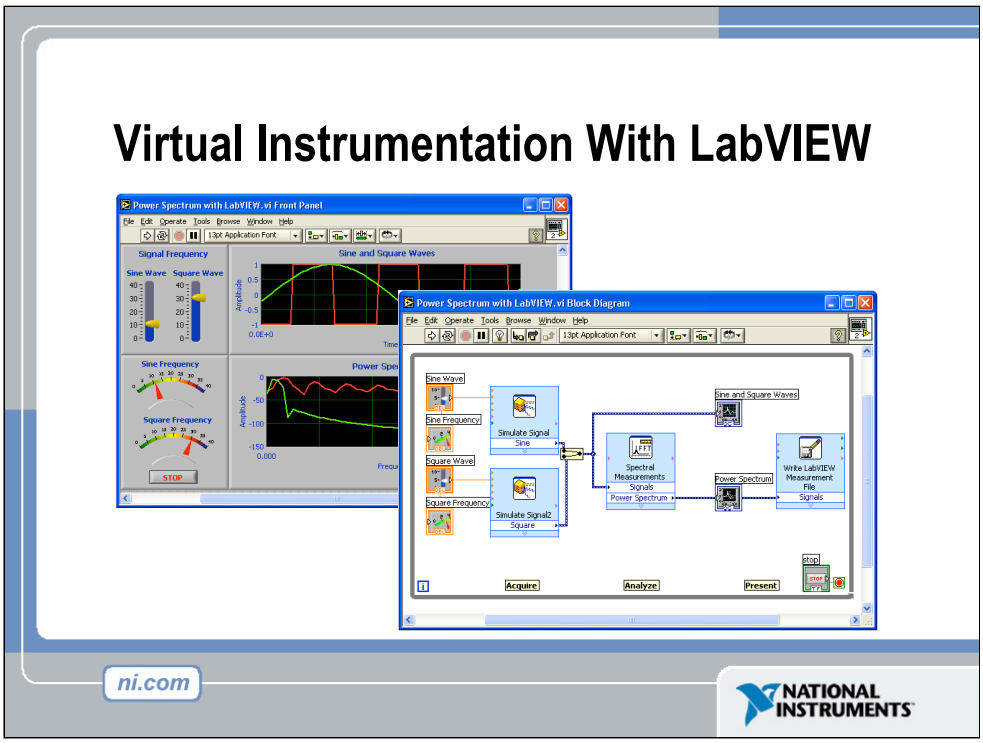
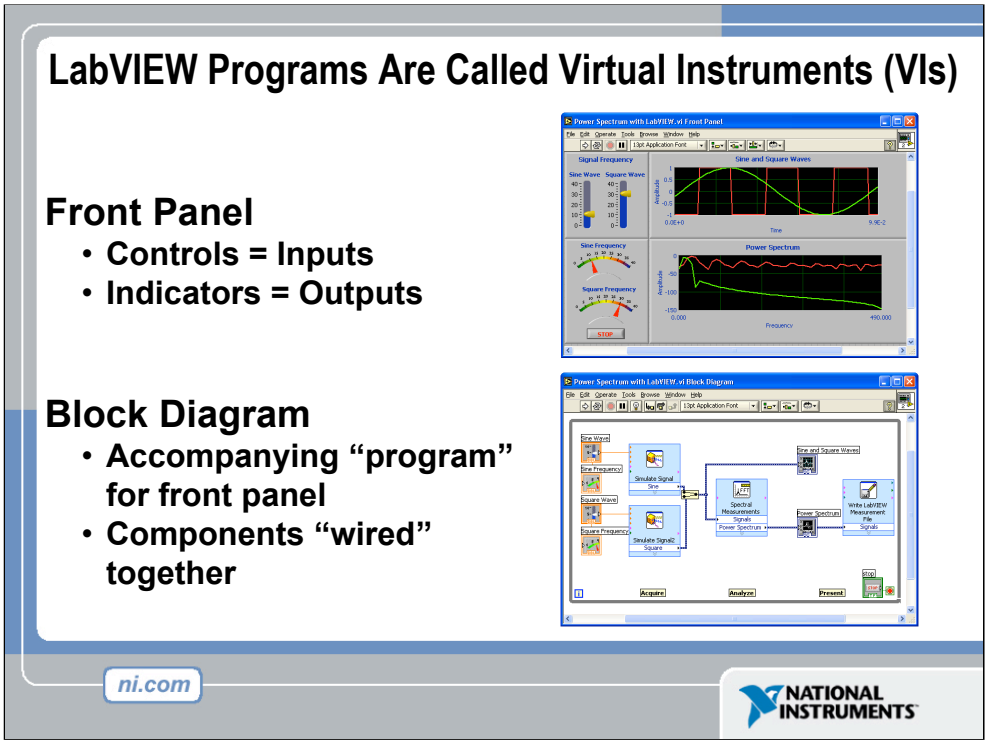


LabVIEW is a graphical development environment with built-in functionality for simulation, data acquisition, instrument control, measurement analysis, and data presentation. LabVIEW gives you the flexibility of a powerful programming language without the complexity of traditional development environments. LabVIEW delivers extensive acquisition, analysis, and presentation capabilities in a single environment, so you can seamlessly develop a complete solution on the platform of your choice.

In the workshop we will be using LabVIEW to acquire, analyze, and present data from the Vernier LabPro. The following pages will serve as a brief introduction to LabVIEW. Do not be alarmed if this seems a bit foreign initially. Review the materials with the goal of familiarity not expertise. The workshop will allow for step-by-step instruction and hands-on learning to prepare to give your students a powerful learning and experimentation tool.



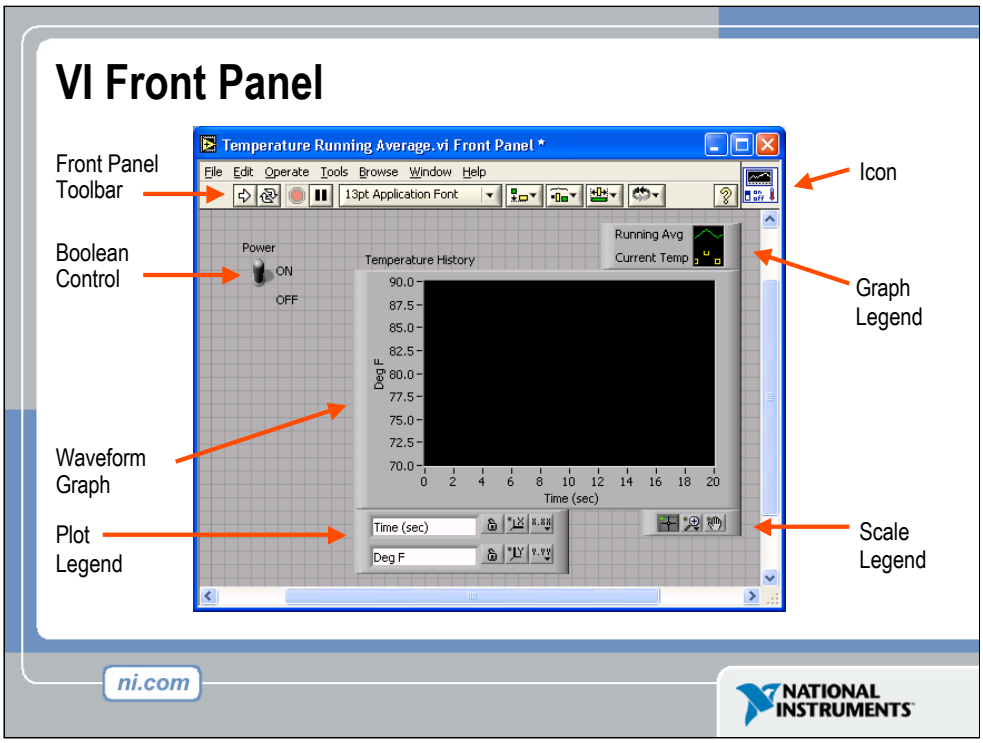
LabVIEW programs are called virtual instruments (VIs).

Stress that controls equal inputs, indicators equal outputs.

Each VI contains three main parts:
- Front Panel – How the user interacts with the VI.
- Block Diagram – The code that controls the program.
- Icon/Connector – Means of connecting a VI to other VIs.

The Front Panel is used to interact with the user when the program is running. Users can control the program, change inputs, and see data updated in real time. Stress that controls are used for inputs- adjusting a slide control to set an alarm value, turning a switch on or off, or stopping a program. Indicators are used as outputs. Thermometers, lights, and other indicators indicate values from the program. These may include data, program states, and other information.

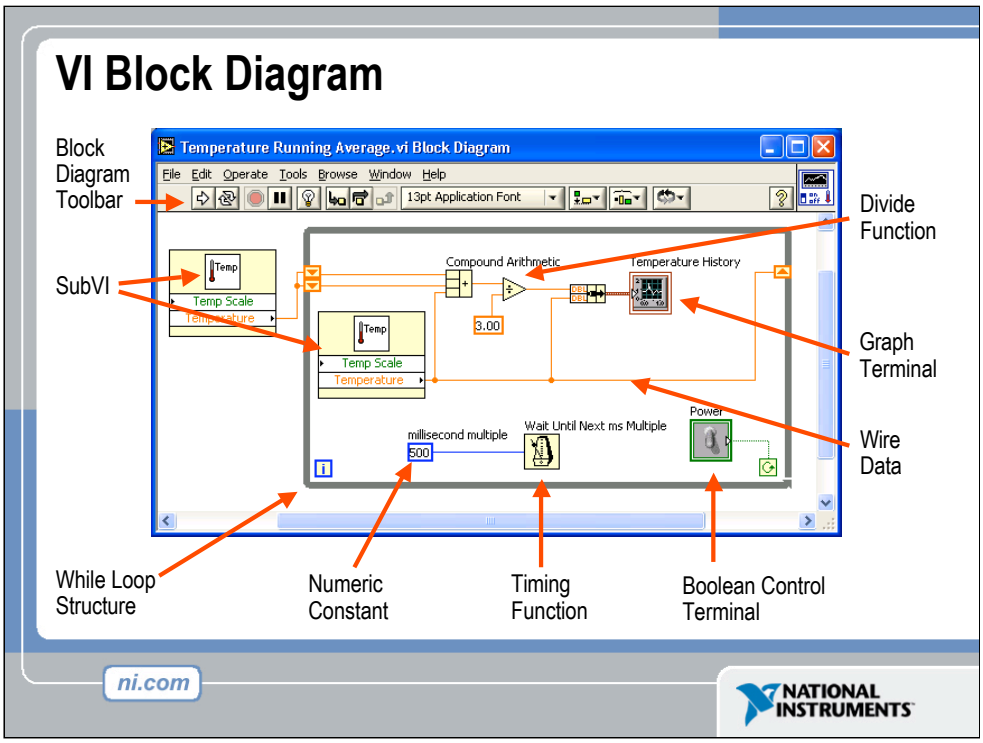
Every front panel control or indicator has a corresponding terminal on the block diagram. When a VI is run, values from controls flow through the block diagram, where they are used in the functions on the diagram, and the results are passed into other functions or indicators.



The front panel is the user interface of the VI. You build the front panel with controls and indicators, 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 data to the block diagram of the VI. Indicators simulate instrument output devices and display data the block diagram acquires or generates.

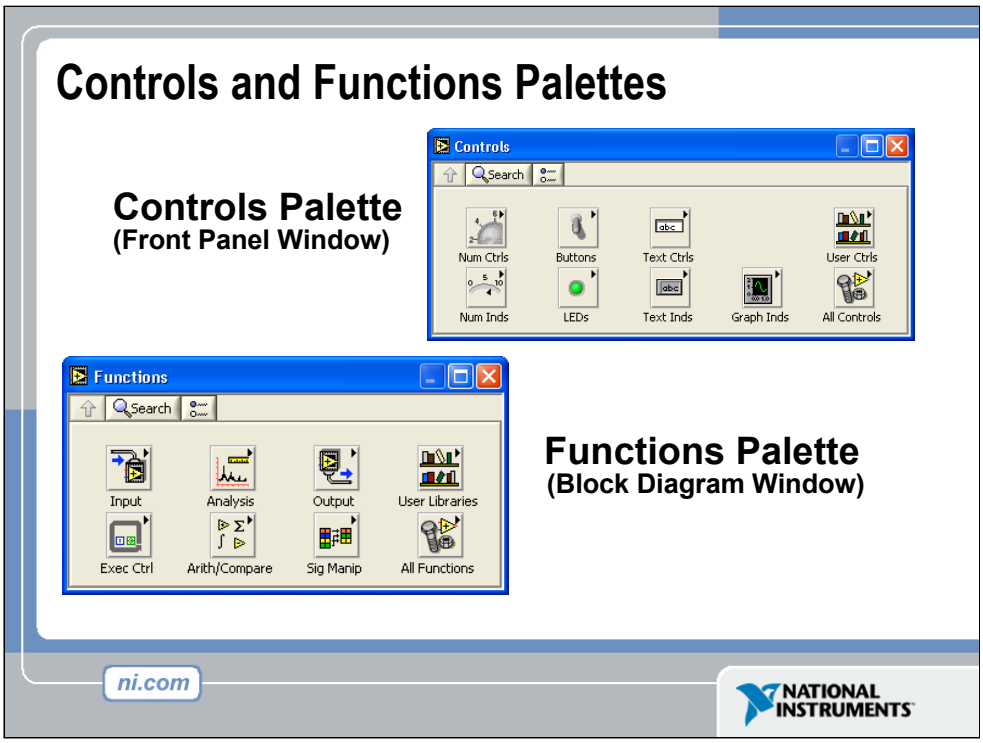
In this picture, the **Power switch** is a boolean control. A boolean contains either a true or false value. The value is false until the switch is pressed. When the switch is pressed, the value becomes true. The **temperature history** indicator is a waveform graph. It displays multiple numbers. In this case, the graph will plot Deg F versus Time (sec).

The front panel also contains a toolbar, whose functions we will discuss later.



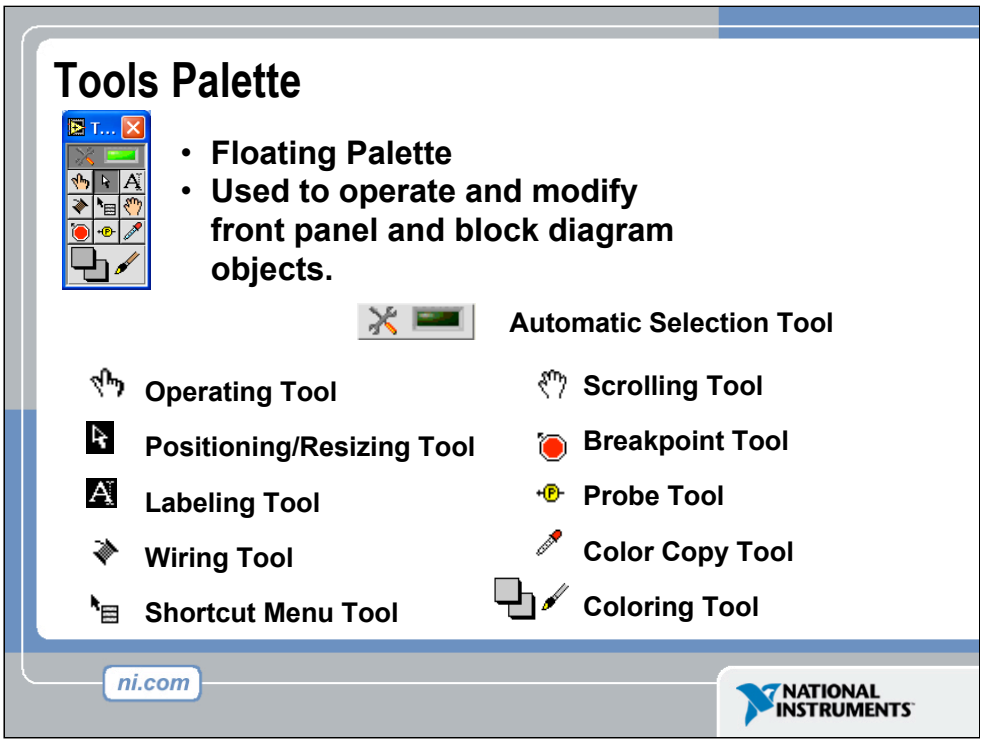
The block diagram contains this graphical source code. Front panel objects appear as terminals on the block diagram. Additionally, the block diagram contains functions and structures from built-in LabVIEW VI libraries. Wires connect each of the nodes on the block diagram, including control and indicator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

In this block diagram, the subVI **Temp** calls the subroutine which retrieves a temperature from a Data Acquisition (DAQ) board. This temperature is plotted along with the running average temperature on the waveform graph **Temperature History**. The **Power** switch is a boolean control on the Front Panel which will stop execution of the While Loop. The While Loop also contains a Timing Function to control how frequently the loop iterates.



Use the **Controls** palette to place controls and indicators on the front panel. The **Controls** palette is available only on the front panel. Select **Window»Show Controls Palette** or right-click the front panel workspace to display the **Controls** palette. You also can display the **Controls** palette by right-clicking an open area on the front panel. Tack down the **Controls** palette by clicking the pushpin on the top left corner of the palette.

Use the **Functions** palette, to build the block diagram. 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. You also can display the **Functions** palette by right-clicking an open area on the block diagram. Tack down the **Functions** palette by clicking the pushpin on the top left corner of the palette.



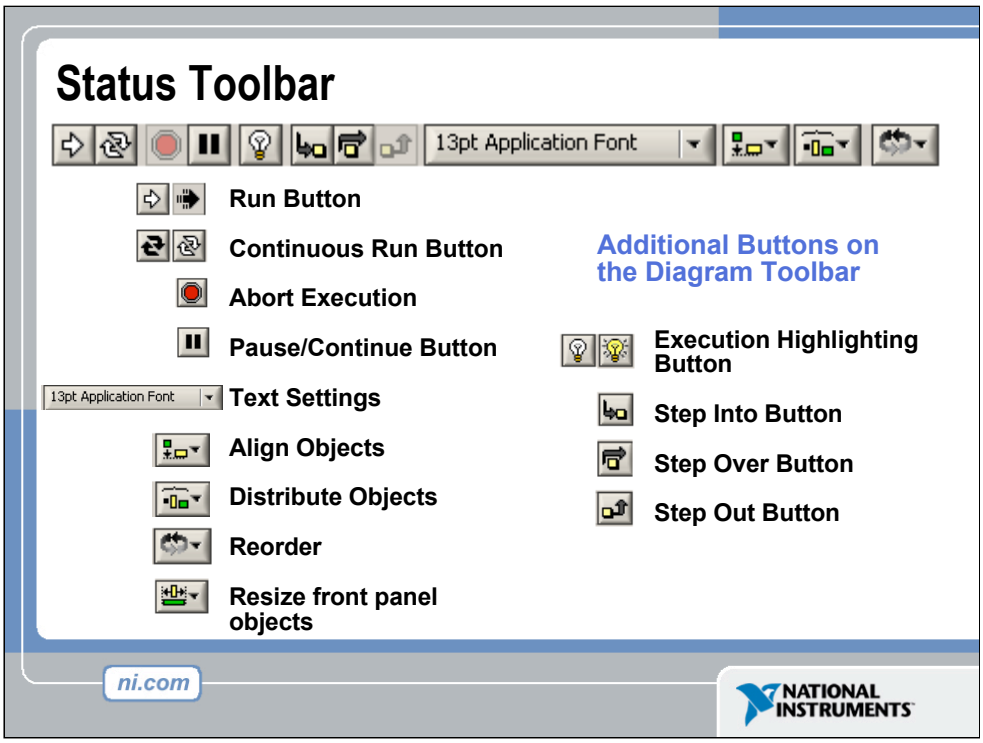
If automatic tool selection is enabled and you move the cursor over objects on the front panel or block diagram, LabVIEW automatically selects the corresponding tool from the **Tools** palette. Toggle automatic tool selection by clicking the **Automatic Tool Selection** button in the **Tools** palette.

Use the Operating tool to change the values of a control or select the text within a control.

Use the Positioning tool to select, move, or resize objects. The Positioning tool changes shape when it moves over a corner of a resizable object.

Use the Labeling tool to edit text and create free labels. The Labeling tool changes to a cursor when you create free labels.

Use the Wiring tool to wire objects together on the block diagram.



•Click the **Run** button to run the VI. While the VI runs, the **Run** button appears with a black arrow if the VI is a top-level VI, meaning it has no callers and therefore is not a subVI.

•Click the **Continuous Run** button to run the VI until you abort or pause it. You also can click the button again to disable continuous running.

•While the VI runs, the **Abort Execution** button appears. Click this button to stop the VI immediately.

> **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 to pause a running VI. When you click the **Pause** button, LabVIEW highlights on the block diagram the location where you paused execution. Click the **Pause** button again to continue running the VI.

•Select the **Text Settings** pull-down menu to change the font settings for the VI, including size, style, and color.

•Select The **Align Objects** pull-down menu to align objects along axes, including vertical, top edge, left, and so on.

•Select the **Distribute Objects** pull-down menu to space objects evenly, including gaps, compression, and so on.

•Select the **Resize Objects** pull-down menu to change the width and height of front panel objects.

•Select the **Reorder** pull-down menu when you have objects that overlap each other and you want to define which one is in front or back of another. Select one of the objects with the Positioning tool and then select from **Move Forward**, **Move Backward**, **Move To Front**, and **Move To Back**.

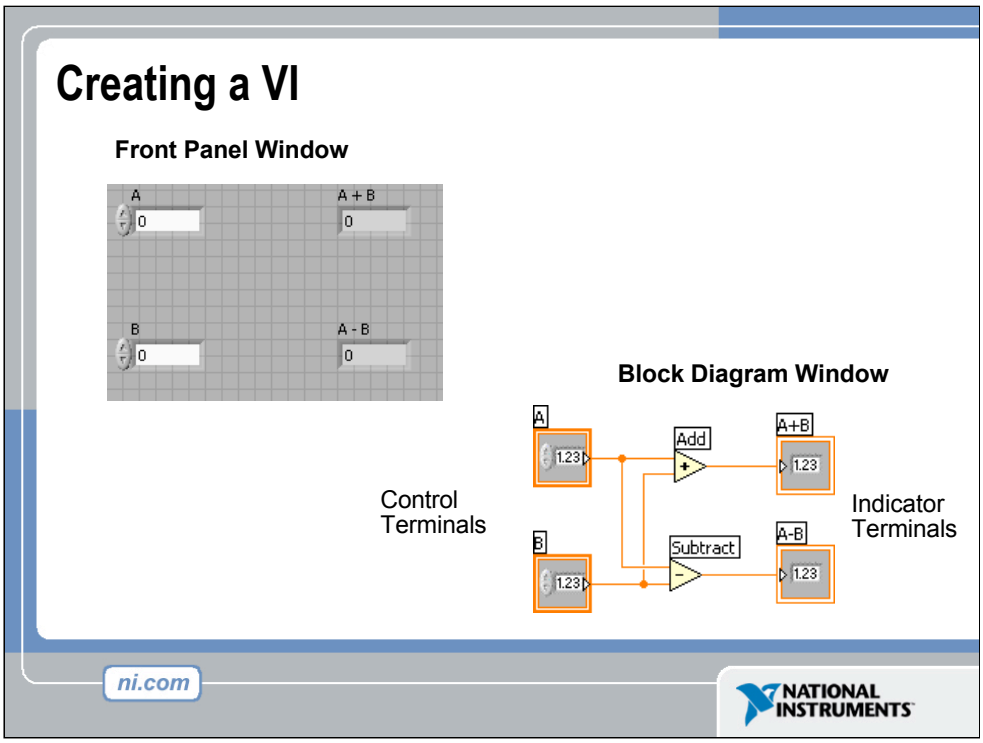
<the following items only appear on the bock diagram toolbar>

•Click the **Highlight Execution** button to see the flow of data through the block diagram. Click the button again to disable execution highlighting.

•Click the **Step Into** button to single-step into a loop, subVI, and so on. Single-stepping through a VI steps through the VI node to node. Each node blinks to denote when it is ready to execute. By stepping into the node, you are ready to single-step inside the node.

•Click the **Step Over** button to step over a loop, subVI, and so on. By stepping over the node, you execute the node without single-stepping through the node.

•Click the **Step Out** button to step out of a loop, subVI, and so on. By stepping out of a node, you complete single-stepping through the node and go to the next node.



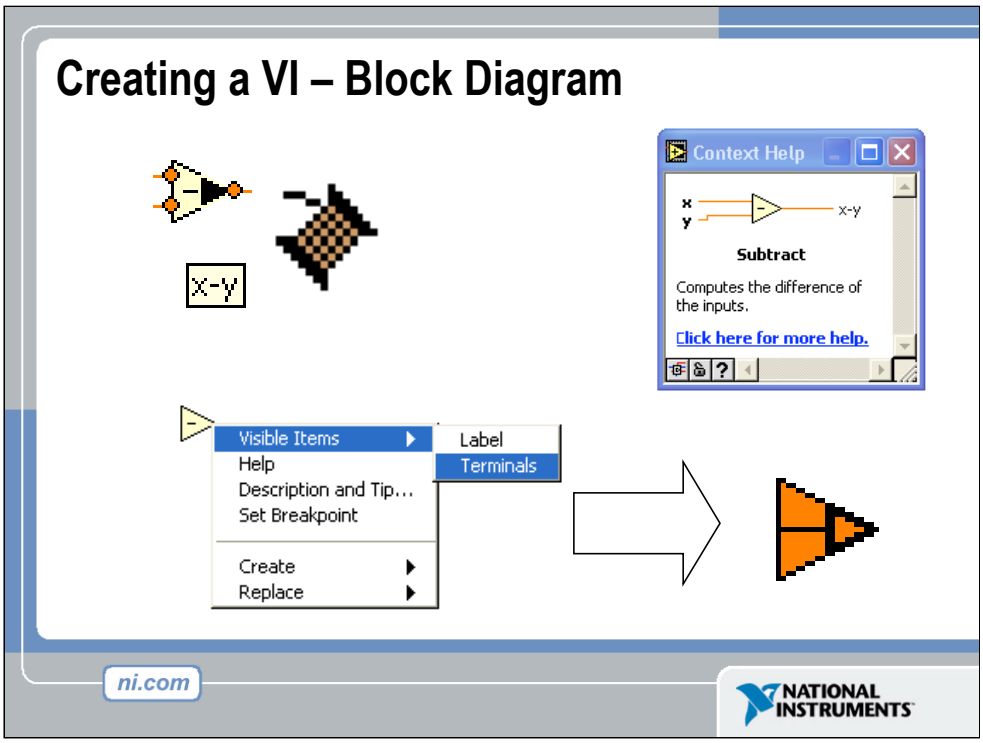
When you create an object on the Front Panel, a terminal will be created on the Block Diagram. These terminals give you access to the Front Panel objects from the Block Diagram code.

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. But in most cases, look for a match in colors.

Controls have an arrow on the right side and have a thick border. Indicators have an arrow on the left and a thin border. Logic rules apply to wiring in LabVIEW: Each wire must have one (but only one) source (or control), and each wire may have multiple destinations (or indicators).

The program in this slide takes data from A and B and passes the values to both an Add function and a subtract function. The results are displayed on the appropriate indicators.



In addition to Front Panel terminals, the Block diagram contains functions. Each function may have multiple input and output terminals. Wiring to these terminals is an important part of LabVIEW programming.

Once you have some experience programming in LabVIEW, wiring will become easy. At first, you may need some assistance. Here are some tips to get you started:

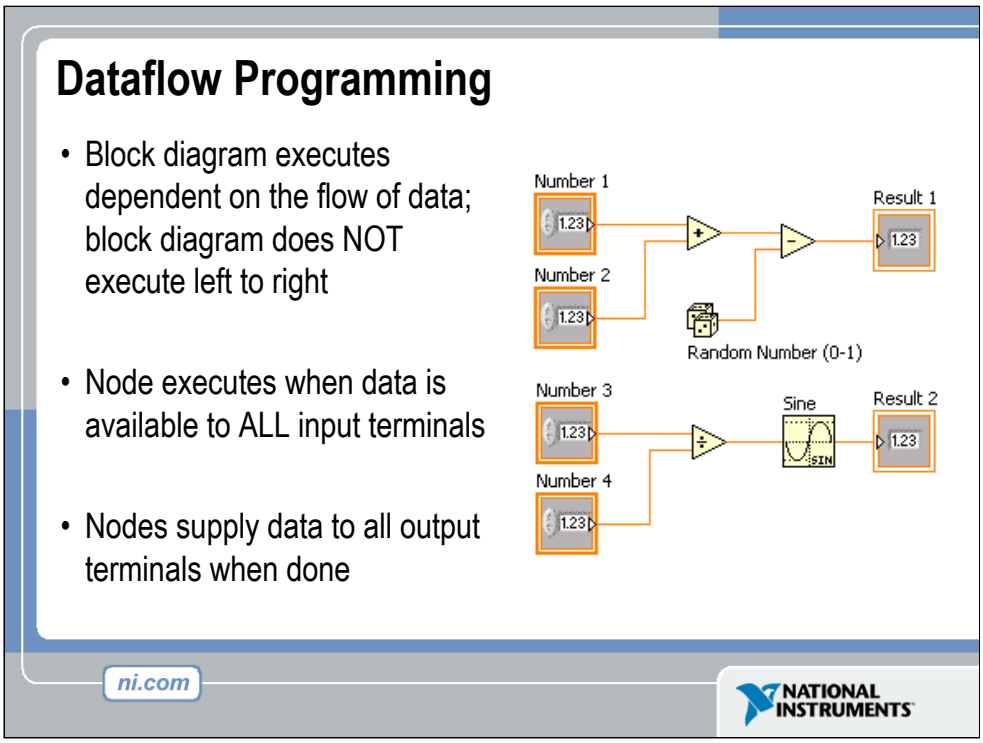
•The wiring tool is used to wire to the nodes of the functions. When you "aim" with the wiring tool, aim with the end of the wire hanging from the spool. This is where the wire will be placed.

•As you move the wiring tool over functions, watch for the yellow tip strip. This will tell you the name of the terminal you are wiring to.

•As you move the wiring tool over a terminal, it will flash. This will help you identify where the wire will attach.

•For more help with the terminals, right-click on the function and select **Visible Items>>Terminals**. The function's picture will be pulled back to reveal the connection terminals. Notice the colors- these match the data types used by the front panel terminals.

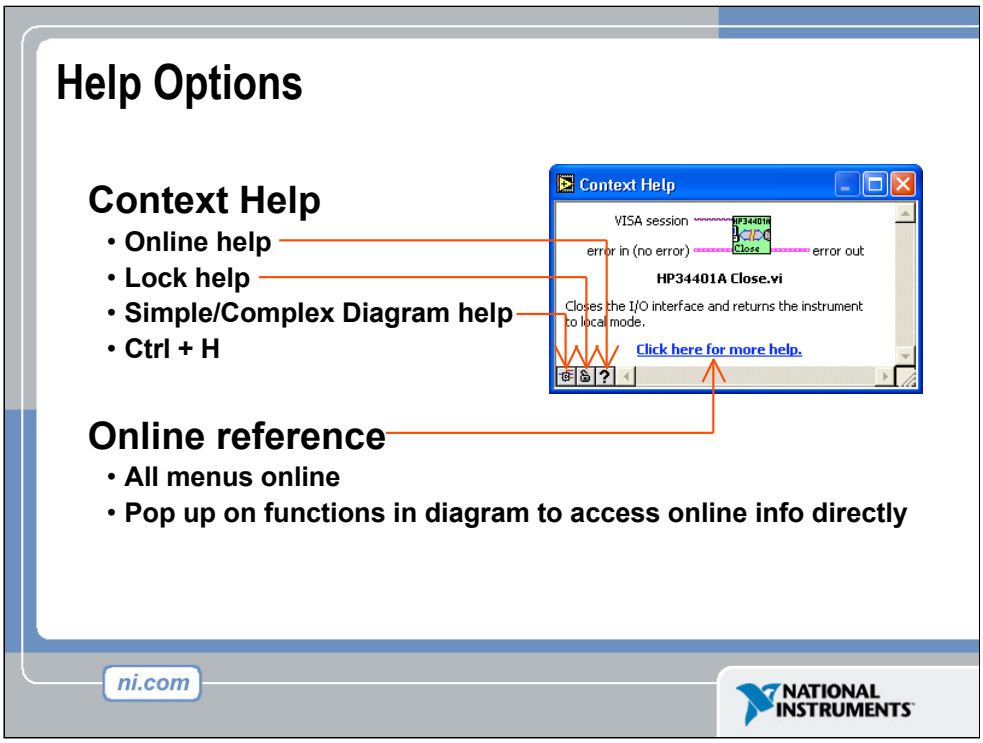
•For additional help, select **Help>>Show Context Help**, or press **CTRL+H**. This will bring up the context help window. As you move your mouse over the function, this window will show you the function, terminals, and a brief help description. Use this with the other tools to help you as you wire.



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.

Consider the block diagram above. It adds two numbers and then subtracts 50.0 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 one of the inputs of the Subtract function is not valid until the Add function has finished executing and passed 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 code to the right, 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 a Sequence structure to force the order of execution.



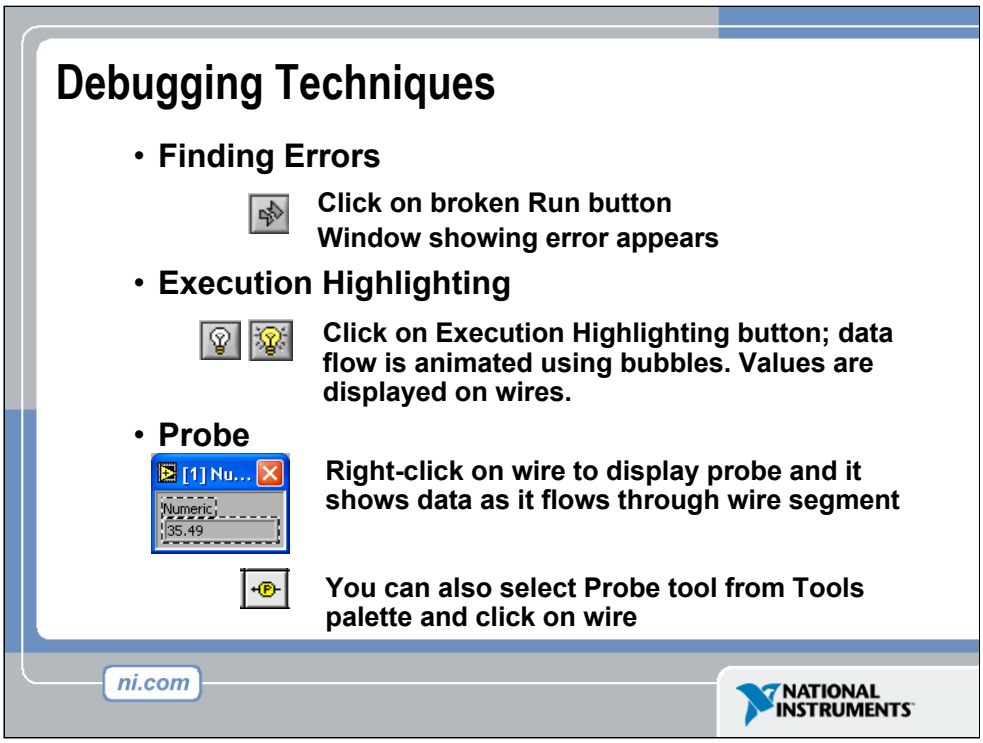
Use the **Context Help** window and the **LabVIEW Help** to help you build and edit VIs. Refer to the **LabVIEW Help** and manuals for more information.

Context Help Window

To display the **Context Help** window, select **Help»Show Context Help** or press the <Ctrl-H> keys. 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. Above is an example **Context Help** window.

Click the **Simple/Detailed Context Help** button located on the lower left corner of the **Context Help** window to change between simple and detailed context help. The simple mode emphasizes the important connections. Optional terminals are shown by wire stubs, informing you that other connections exist.

Click the **Lock Context Help** button to lock the current contents of the **Context Help** window. When the contents are locked, moving the cursor over another object does not change the contents of the window. To unlock the window, click the button again. You also can access this option from the **Help** menu.



When your VI is not executable, a broken arrow is displayed in the Run button in the palette.

Finding Errors: To list errors, click on the broken arrow. To locate the bad object, click on the error message.

Execution Highlighting: Animates the diagram and traces the flow of the data, allowing you to view intermediate values.

Click on the **light bulb** on the toolbar.

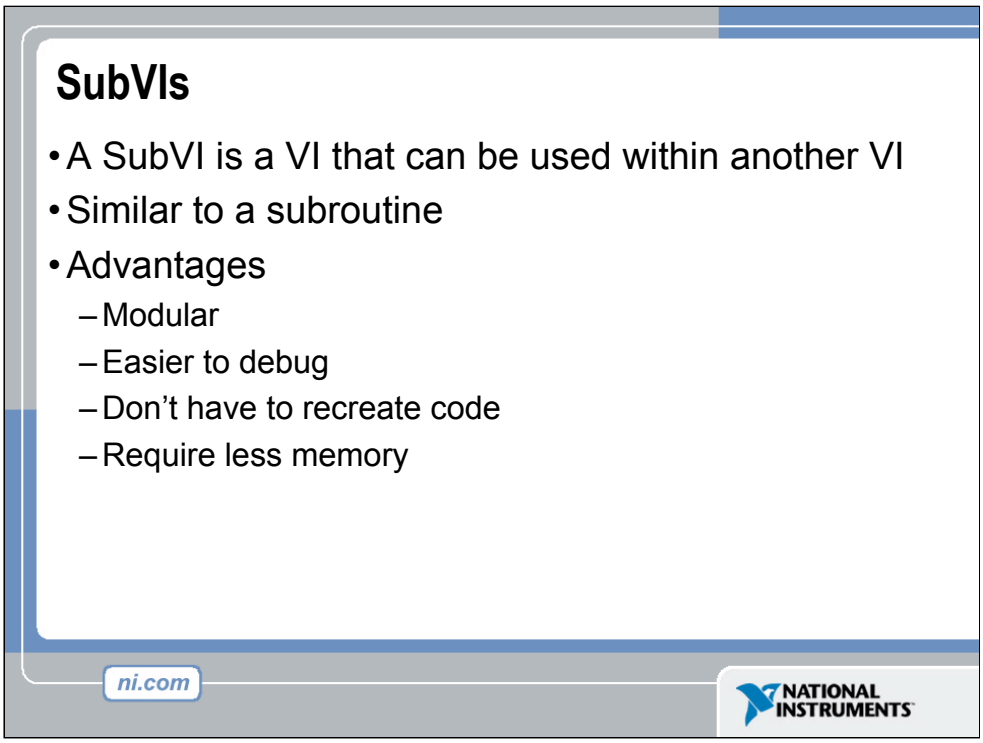
Probe: Used to view values in arrays and clusters.

Click on wires with the **Probe** tool or right-click on the wire to set probes.

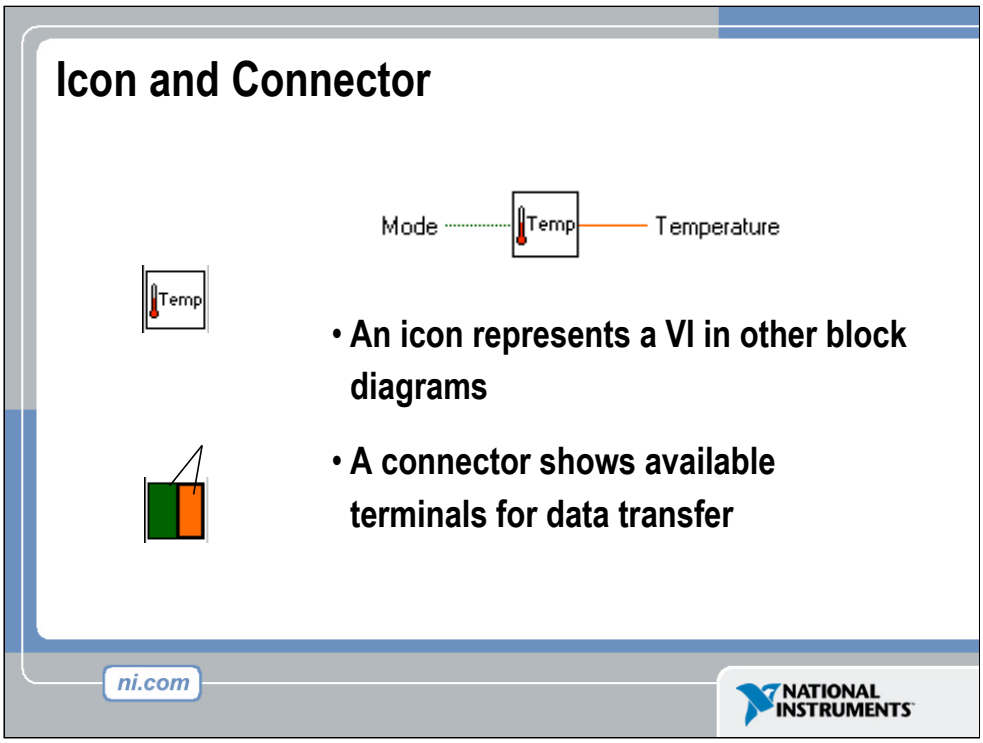
Breakpoint: Set pauses at different locations on the diagram.

Click on wires or objects with the **Breakpoint** tool to set breakpoints.

Use **Debug Demonstrate** VI from BASICS.LLB to demonstrate the options and tools.

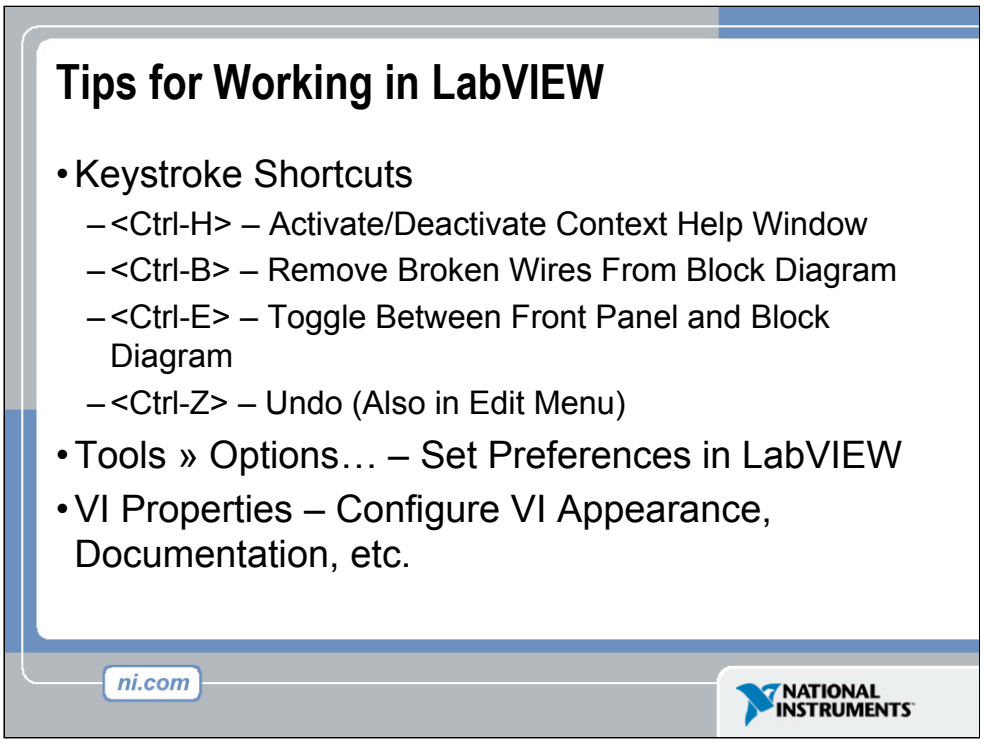


After you build a VI and create its icon and connector pane, you can use it in another VI. A VI within another VI is called a subVI. A subVI corresponds to a subroutine in text-based programming languages. Using subVIs helps you manage changes and debug the block diagram quickly.



Every VI displays an icon, shown above, in the upper right corner of the front panel and block diagram windows. An icon is a graphical representation of a VI. It can contain text, images, or a combination of both. If you use a VI as a subVI, the icon identifies the subVI on the block diagram of the VI.

The connector shows terminals available for transfer or data to and from the subVI. There are several connector patterns to choose from. Right click on the connector and select the pattern from the **Patterns** menu. From there you can assign controls and indicators on the front panel to the connector terminal, as we will see later.

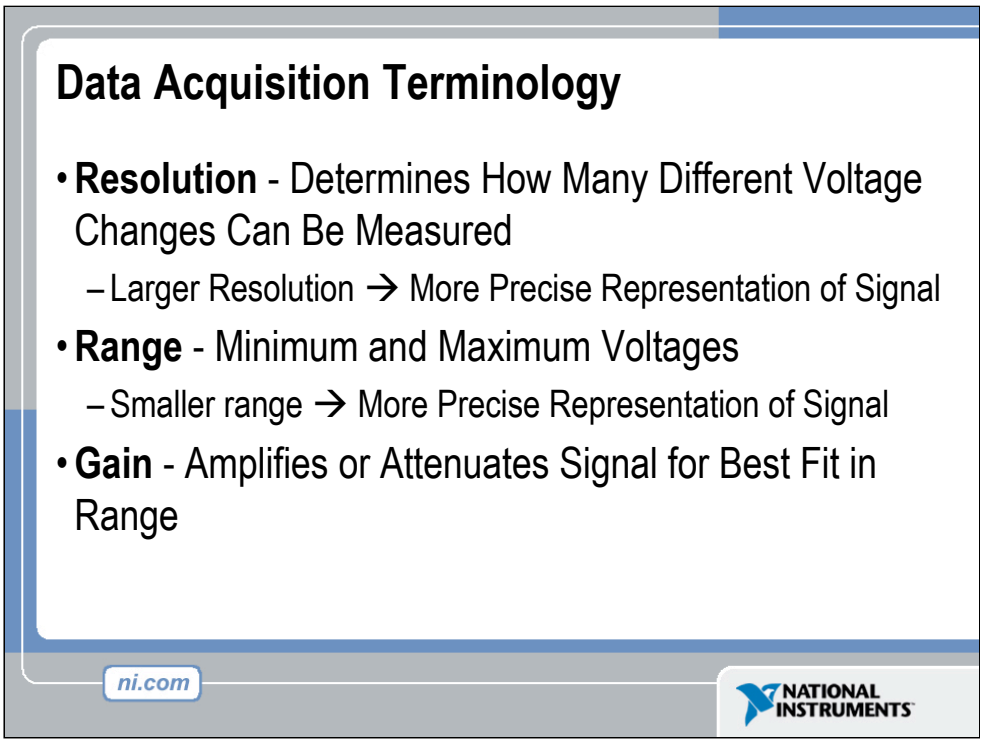


LabVIEW has many keystroke shortcuts that make working easier. The most common shortcuts are listed above.

While the Automatic Selection Tool is great for choosing the tool you would like to use in LabVIEW, there are sometimes cases when you want manual control. Use the Tab key to toggle between the four most common tools (Operate Value, Position/Size/Select, Edit Text, Set Color on Front Panel and Operate Value, Position/Size/Select, Edit Text, Connect Wire on Block Diagram). Once you are finished with the tool you choose, you can press <Shift-Tab> to turn the Automatic Selection Tool on.

In the Tools » Options… dialog, there are many configurable options for customizing your Front Panel, Block Diagram, Colors, Printing, and much more.

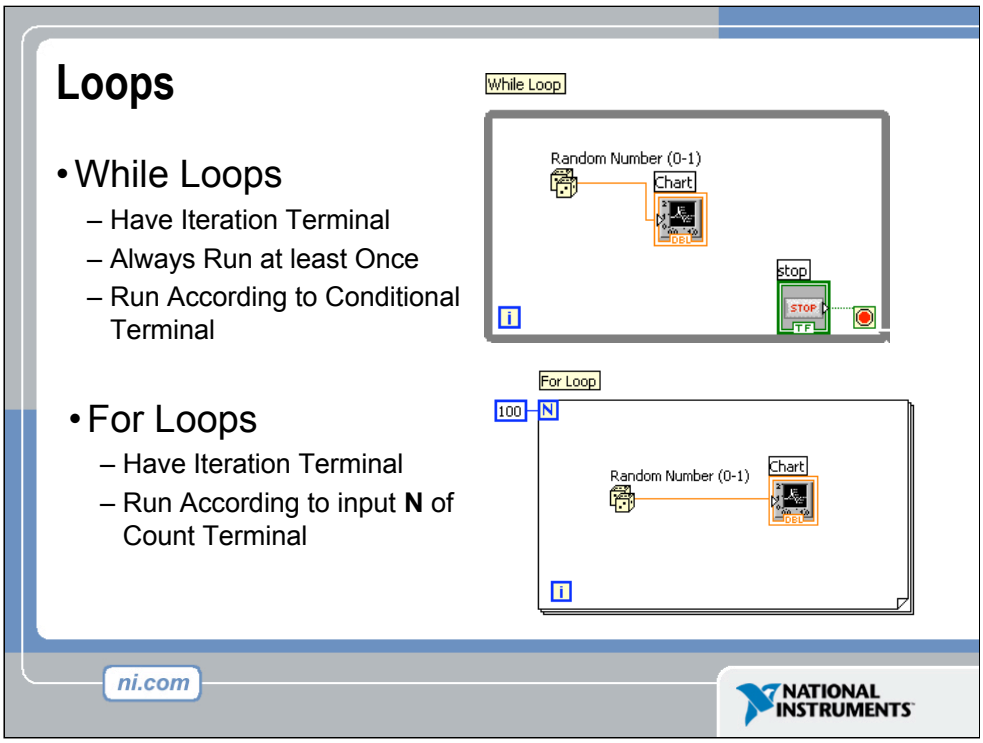
Similar to the LabVIEW Options, you can configure VI specific properties by going to File » VI Properties… There you can document the VI, change the appearance of the window, and customize it in several other ways.



Resolution: When acquiring data to a computer, an Analog-to-Digital Converter (ADC) takes an analog signal and turns it into a binary number. Therefore, each binary number from the ADC represents a certain voltage level. The ADC returns the highest possible level without going over the actual voltage level of the analog signal. Resolution refers to the number of binary levels the ADC can use to represent a signal. To figure out the number of binary levels available based on the resolution you simply take $2^{Resolution}$. Therefore, the higher the resolution, the more levels you will have to represent your signal. For instance, an ADC with 3-bit resolution can measure 2^3 or 8 voltage levels, while an ADC with 12-bit resolution can measure 2^{12} or 4096 voltage levels.

Range: Unlike the resolution of the ADC, the range of the ADC is selectable. Most DAQ devices offer a range from 0 - +10 or -10 to +10. The range is chosen when you configure your device in NI-DAQ. Keep in mind that the resolution of the ADC will be spread over whatever range you choose. The larger the range, the more spread out your resolution will be, and you will get a worse representation of your signal. Thus it is important to pick your range to properly fit your input signal.

Gain: Properly choosing the range of your ADC is one way to make sure you are maximizing the resolution of your ADC. Another way to help your signal maximize the resolution of the ADC is by applying a gain. Gain refers to any amplification or attenuation of a signal. The gain setting is a scaling factor. Each voltage level on your incoming signal is multiplied by the gain setting to achieve the amplified or attenuated signal. Unlike resolution that is a fixed setting of the ADC, and range that is chosen when the DAQ device is configured, the gain is specified indirectly through a setting called input limits. Input limits refers to the minimum and maximum values of your actual analog input signal. Based on the input limits you set, the largest possible gain is applied to your signal that will keep the signal within the chosen range of the ADC. So instead of needing to calculate the best gain based on your signal and the chosen range, all you need to know is the minimum and maximum values of your signal.



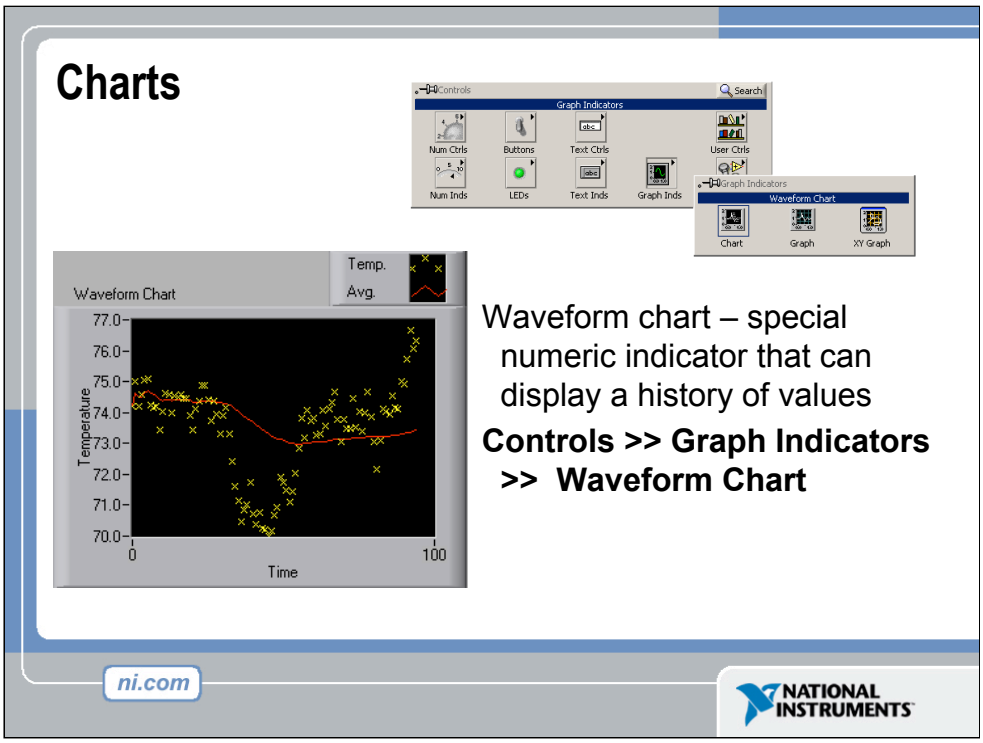
Both the While and For Loops are located on the **Functions»Structures** palette. The For Loop differs from the While Loop in that the For Loop executes a set number of times. A While Loop stops executing the subdiagram only if the value at the conditional terminal exists.

While Loops

Similar to a Do Loop or a Repeat-Until Loop in text-based programming languages, a While Loop, shown at the top right, executes a subdiagram until a condition is met. The While Loop executes the sub diagram until the conditional terminal, an input terminal, receives a specific Boolean value. The default behavior and appearance of the conditional terminal is **Continue If True**, shown at left. When a conditional terminal is **Continue If True**, the While Loop executes its subdiagram until the conditional terminal receives a FALSE value. The iteration terminal (an output terminal), shown at left, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns 0.

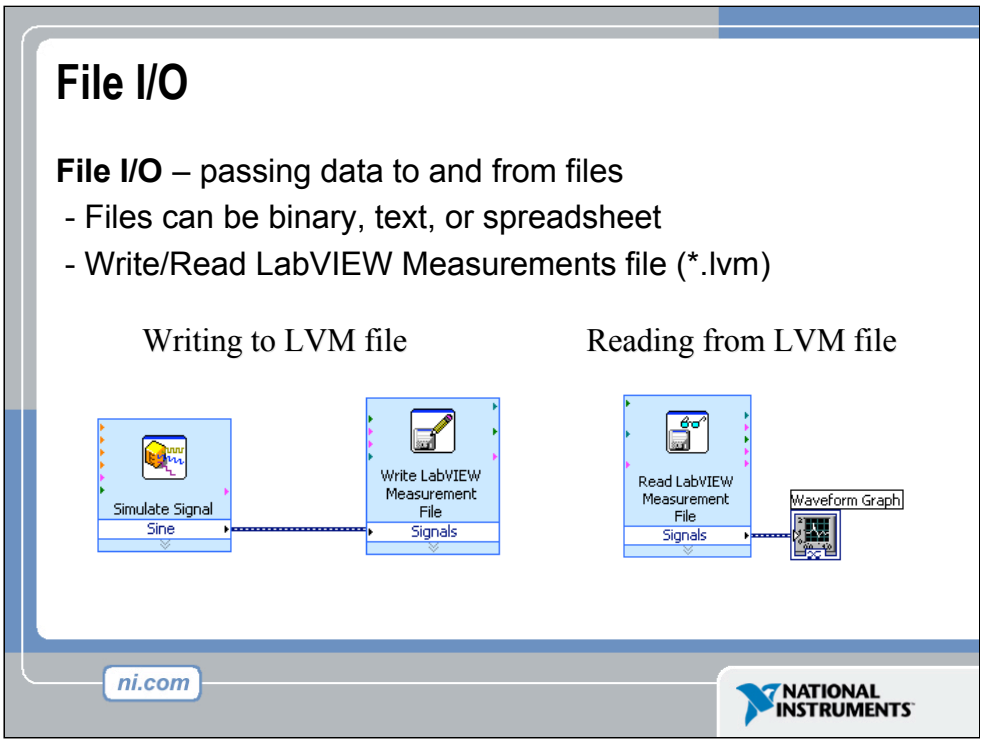
For Loops

A For Loop, shown at left, executes a subdiagram a set number of times. The value in the count terminal (an input terminal) represented by the N, indicates how many times to repeat the subdiagram. The iteration terminal (an output terminal), shown at left, contains the number of completed iterations. The iteration count always starts at zero. During the first iteration, the iteration terminal returns 0.



The waveform chart is a special numeric indicator that displays one or more plots. The waveform chart is located on the **Controls»Graph Indicators** palette. Waveform charts can display single or multiple plots. The following front panel shows an example of a multi-plot waveform chart.

You can change the min and max values of either the x or y axis by double clicking on the value with the labeling tool and typing the new value. Similarly, you can change the label of the axis. You can also right click the plot legend and change the style, shape, and color of the trace that is displayed on the chart.



File I/O operations pass data to and from files. In LabVIEW, you can use File I/O functions to:

Open and close data files

Read data from and write data to files

Read from and writ to spreadsheet-formatted files

Move and rename files and directories

Change file characteristics

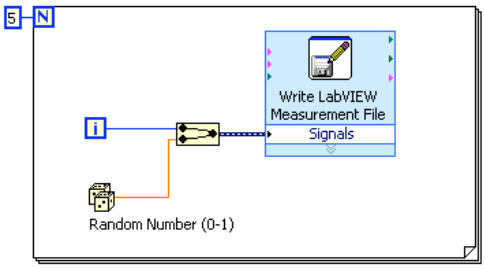
Create, modify, and read a configuration file

Write to or read from LabVIEW Measurements files.

In this course we will examine how to write to or read from LabVIEW Measurements files (*.lvm files).

Write LabVIEW Measurement File

- Includes the open, write, close and error handling functions
- Handles formatting the string with either a tab or comma delimiter
- Merge Signals function is used to combine data into the dynamic data type

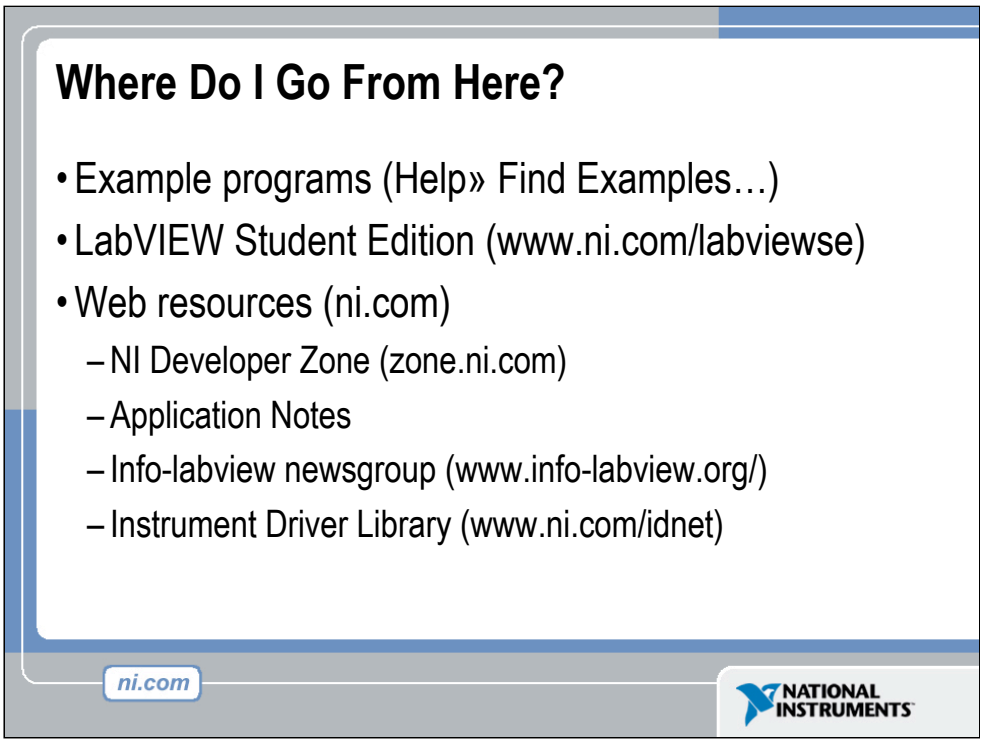


	A	B	C	D
1		0	0.385055	
2		1	0.23516	
3		2	0.985184	
4		3	0.177893	
5		4	0.935915	
6				
7				

ni.com

NATIONAL INSTRUMENTS

The Write LVM file can write to spreadsheet files. However, its main purpose is for logging data, that will be used in LabVIEW. This VI creates a .lvm file which can be opened in a spreadsheet application. For simple spreadsheet files, use the Express VIs: Write LVM and Read LVM.



Where do you go from here?

•National Instruments offers a wide range of instructional courseware to expand your knowledge. Please visit ni.com/academic for programs and resources available.
•The LabVIEW Student Edition is a available from our website. In includes *Learning With LabVIEW*, a textbook written by Dr. Bob Bishop from the University of Texas at Austin.
•The web is the best place to turn in order to find information on LabVIEW. Ni.com is designed to be the one stop resource to find information.

- •The NI Developer Zone ("NIDZ") is a place for developers to meet, discuss design issues, and post content.
- •Application Notes can be downloaded from ni.com/support on a variety of topics.
- •Info-labview is a newsgroup maintained by a third party.
- •There is an exhaustive library of LabVIEW instrument drivers available for download from NIDZ.