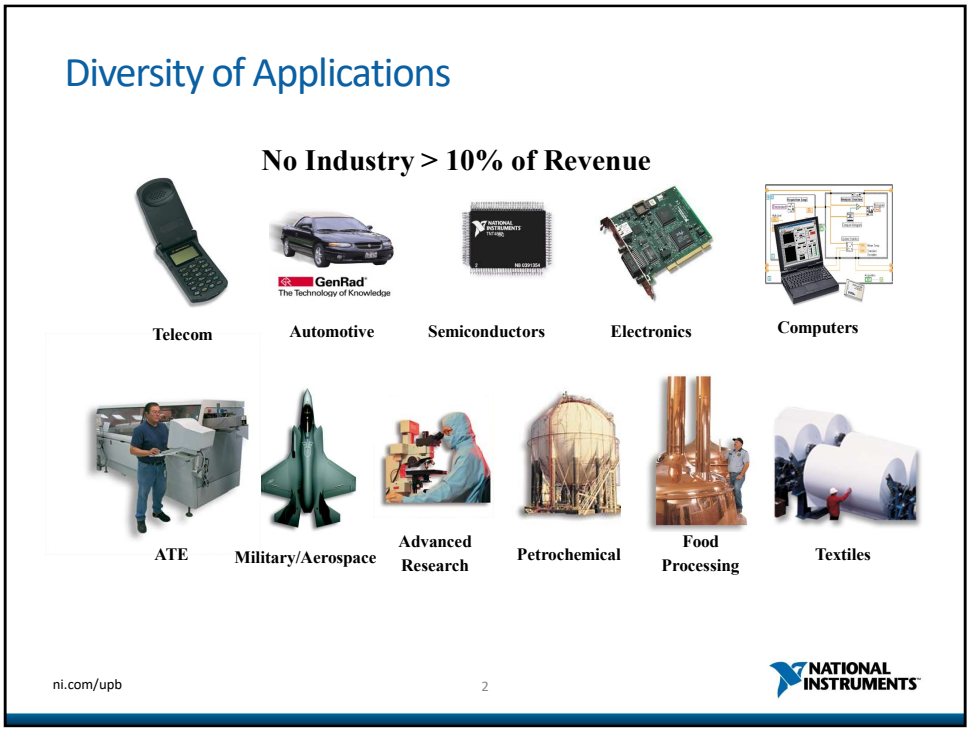

POLITEHNICA
UNIVERSITATEA
BUCUREȘTI
UPB
1818
UNIVERSITY POLITEHNICA
OF BUCHAREST
NATIONAL INSTRUMENTS
LabVIEW
STUDENT AMBASSADOR
LabVIEW Student Ambassador (2012) Tranca Dumitru-Cristian
NATIONAL
INSTRUMENTS
Conținutul materialului este parte din trainingurile ținute în UPB prin programul
LabVIEW Student Ambassador
ni.com/upb

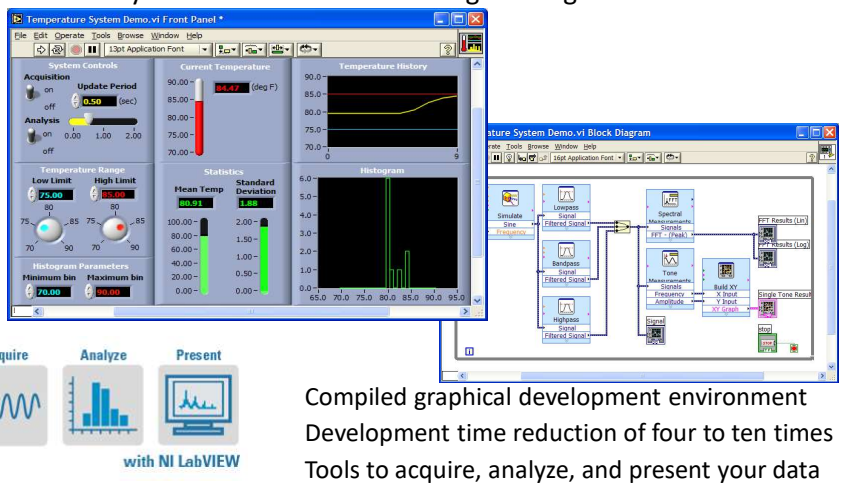


One of the reasons we've been able to weather adverse conditions relatively well is that we're not tied to any particular industry. We make hardware and software for engineers and scientists useful to just about any company that makes things. Any group building something or inventing something requires testing to make sure that what they're making is of good quality, or that they understand properties of things they're trying to invent. They also might need some sort of automated control so they aren't building things by hand. Our hardware and software measures and controls things like temperature, pressure, electrical properties, etc. to help ensure quality of products. Sometimes we do make products more aligned with particular industries, but for the most part, our products can be generically used across them, like Microsoft Office or personal computers.

ATE=Automatic Test Equipment

What is LabVIEW?

Laboratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench



Compiled graphical development environment
Development time reduction of four to ten times
Tools to acquire, analyze, and present your data

ni.com/upb 3 NATIONAL INSTRUMENTS

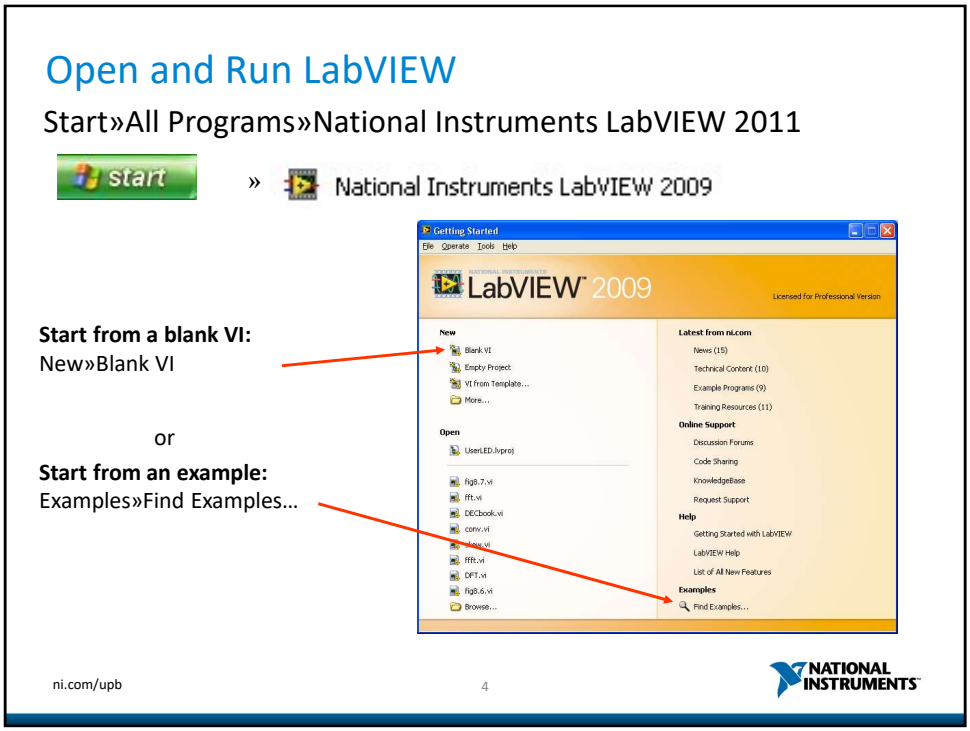
What is NI LabVIEW? LabVIEW is a highly productive graphical development environment with the performance and flexibility of a programming language, as well as high-level functionality and configuration utilities designed specifically for measurement and automation applications.

In general-purpose programming languages, the code is as much of a concern as the application. You must pay close attention to the syntax (commas, periods, semicolons, square brackets, curly brackets, round brackets, etc.). In contrast, with LabVIEW you use icons to represent functions, and you wire them together to determine the flow of data through your program, similar to creating flowcharts. It has all the breadth and depth of a general-purpose programming language, but it is easy to use, increasing your productivity by decreasing the time required to develop your applications.

You can easily divide measurement and automation application into three main parts: acquisition, analysis, and presentation of data. LabVIEW provides a seamless way to acquire your data, perform necessary analysis on that data, and present the information in a chosen format. Throughout the seminar, we touch upon each of these three components of a measurement and automation application.

Each program in LabVIEW is called a virtual instrument, or VI. The VI serves as the primary building block of a LabVIEW application, and you can use it to modularize your code for efficient design, clear and concise documentation, and simplified maintenance. Each LabVIEW VI is made up of three main components: the Front Panel, Block Diagram, and the Palettes.

In the coming slides, we continue our discussion of the actual development environment of LabVIEW.



This is the screen you will see when LabVIEW is opened. From the Getting Started window you can do a number of things to begin development, including creating a new VI, a new project, or opening an example from the Example Finder. Other resources are available through quick links as well, such as online articles and forums. Recently opened files are also listed for easy access.

To create a new VI, you have a number of methods: (1) click the Blank VI link under New (2) go to File>>New VI (3) go to File >> New… and then select Blank VI in the outline/list. You can also create an empty project and create a VI from the project as described in the next slide.

LabVIEW Example Finder

LabVIEW features hundreds of example VIs you can use and incorporate into VIs that you create. In addition to the example VIs that are shipped with LabVIEW, you can access hundreds of example VIs on the NI Developer Zone (zone.ni.com). You can modify an example VI to fit an application, or you can copy and paste from one or more examples into a VI that you create.

What is a Virtual Instrument (VI)?

Answer: a LabVIEW program

1. Front Panel

User interface (UI)

- Controls = inputs
- Indicators = outputs

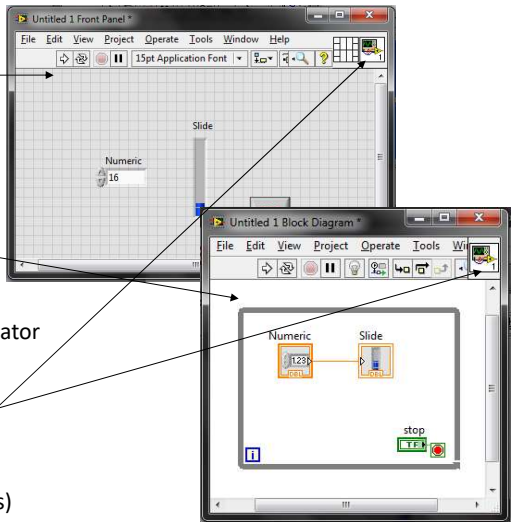
2. Block Diagram

Graphical source code

- Data travels on wires from control terminals through functions to indicator terminals
- Blocks execute by data flow

3. Icon & Connector Pane

- Graphical representation of a VI
- Means of connecting VIs (subVIs)



ni.com/upb 5 NATIONAL INSTRUMENTS

LabVIEW programs are called virtual instruments (VIs).

Each VI contains three main parts:

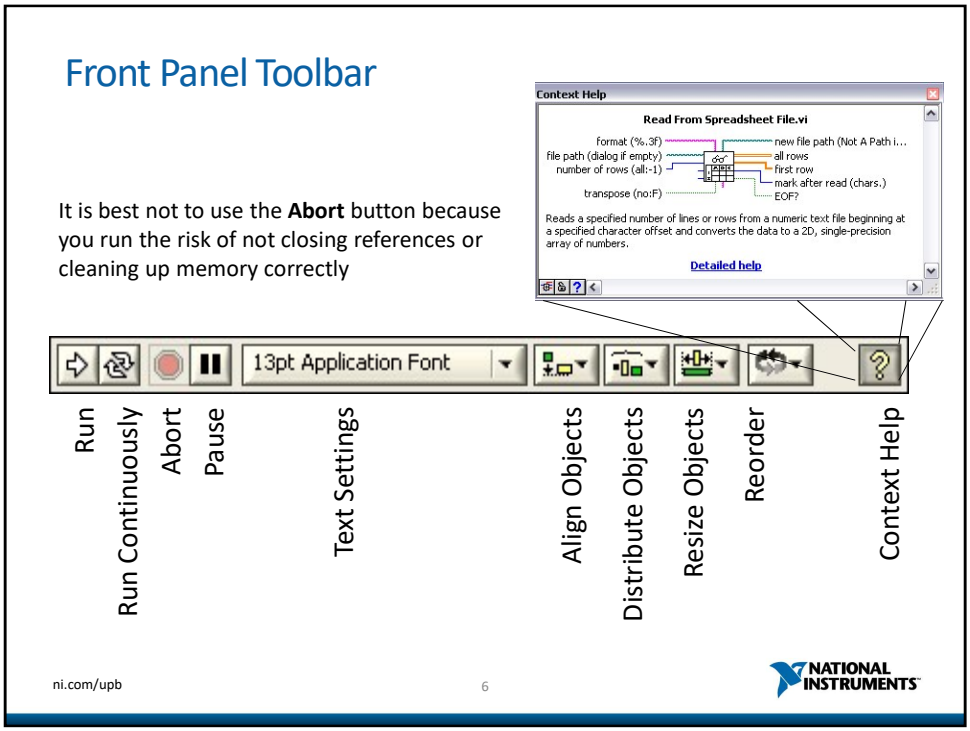
- Front panel – How the user interacts with the VI
- Block diagram – The code that controls the program
- Icon & connector – The means of connecting a VI to other Vis and the graphical representation of the VI

In LabVIEW, you build a user interface by using a set of tools and objects. The user interface is known as the front panel. You then add code using graphical representations of functions to control the front panel objects. The block diagram contains this code. In some ways, the block diagram resembles a flowchart.

You interact with the front panel when the program is running. You can control the program, change inputs, and see data updated in real time. Controls are used for inputs such as 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 display output 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 you run a VI, 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 through wires.

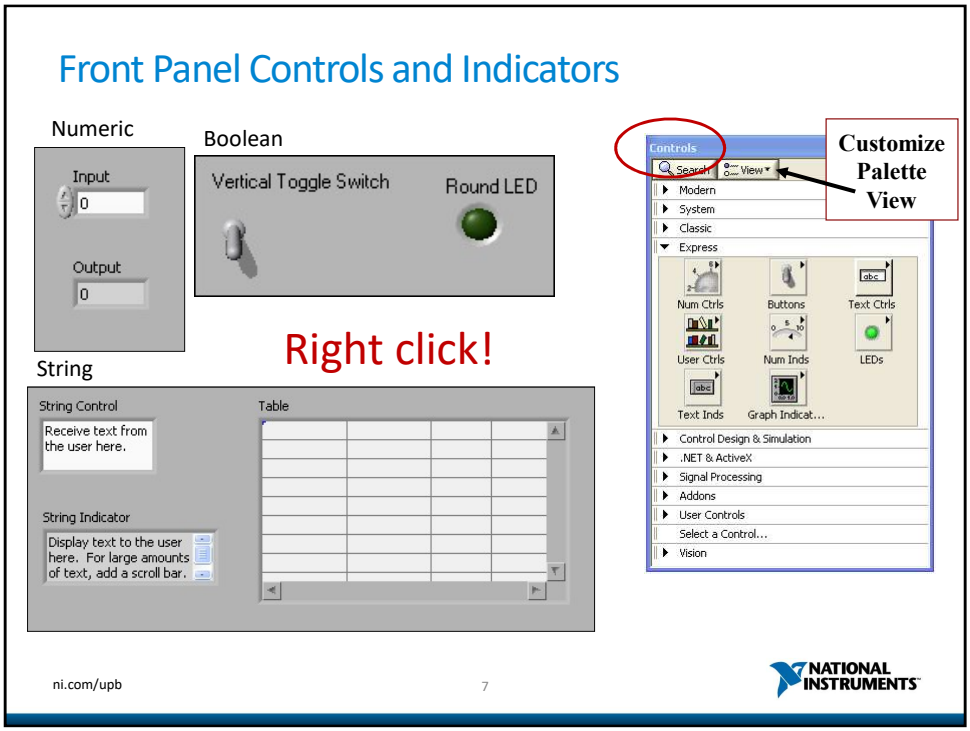
- Left-click to edit and customize Icon



-Using the Abort button is not recommended because it does not allow your VI to properly finish.

-Context Help is your friend! Use it! Context Help will provide information about whatever your mouse is hovering over (if the button in the bottom left hand corner of the Context Help window is in the Unlock position). You can get inputs/outputs of VIs including data types (indicated by the color of the wire) and a brief description of the function of the VI. The Detailed Help link or the blue ? Button (bottom left corner of Context Help window) will take you to LabVIEW's Help file about that item. The button in the far bottom left corner of the Context Help window will show/hide optional input items.

-The Run Continuously button is also used rarely- a loop should be used instead.



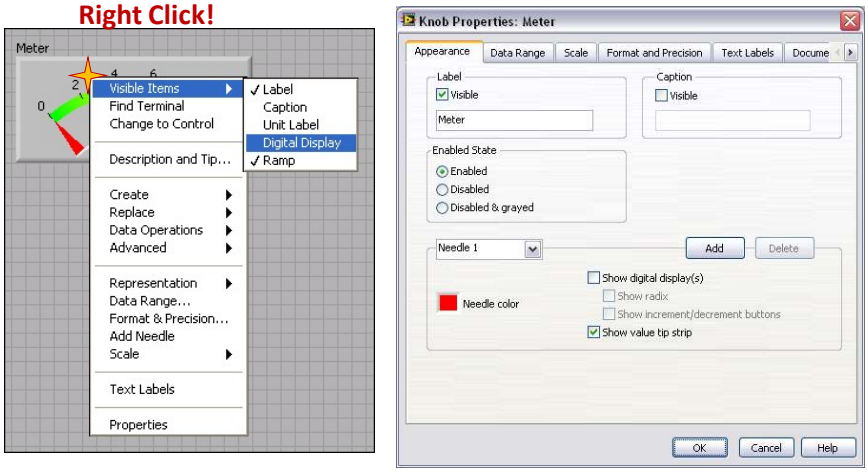
Controls and Indicators are the main items you will find on the front panel (decorations and free labels are examples of other possible items)

- Controls (Input): typically have white background (uninitialized arrays are an exception), numeric controls have the increment/decrement buttons showing by default. Controls can come in all shapes, sizes and types (i.e. numeric, boolean, string, array, cluster – switches, dials, tables)

- Indicators (Output): usually have a gray background and also come in all shapes, sizes, and types (gauges, LEDs, tanks)

Use the **Controls** palette to place controls and indicators on the front panel. The **Controls** palette is available only on the front panel and contains item that can be placed on the front panel. To view the Controls palette, right-click on the gray area of the front panel or select **View»Controls Palette**. Tack down the **Controls** palette by clicking the pushpin on the top left corner of the palette; customize the categories displayed by clicking the View Button and checking the categories you wish to view. *Recommendation* go to Tools>>Options>> Controls/Functions Palette and change the default setting to be Icons and Text for the display.

Shortcut Menus and Properties Dialog

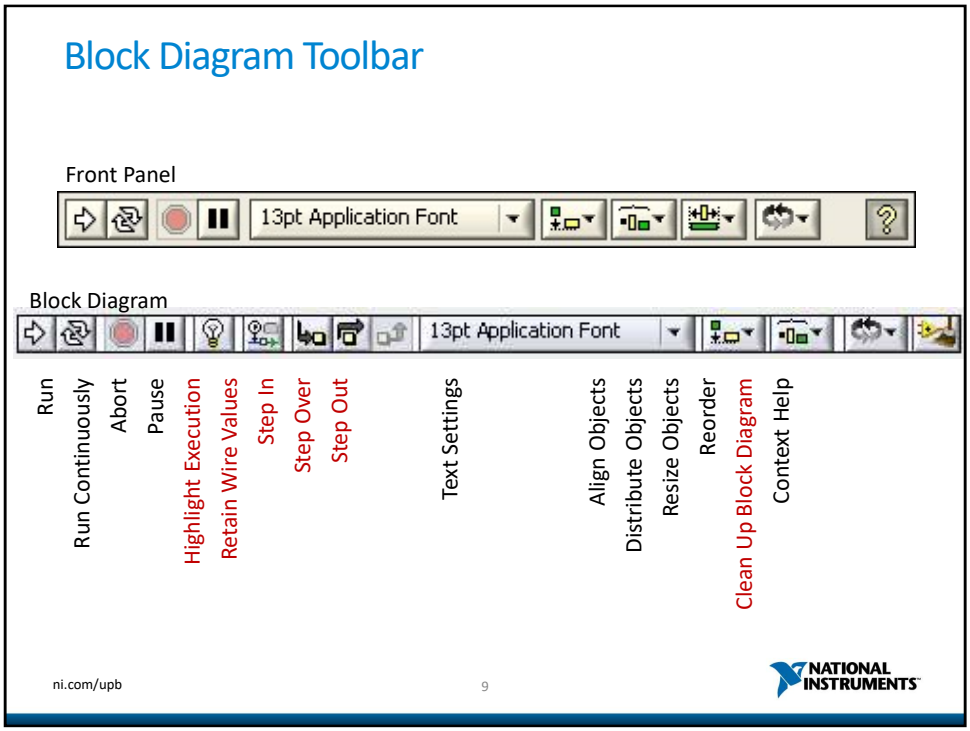


ni.com/upb 8 NATIONAL INSTRUMENTS

All LabVIEW objects have associated shortcut menus and you access the shortcut menu by right-clicking the object. Examples of things that can be done from the short cut menu are: selecting which aspects of the object is visible, change the object to a control or indicator, navigate to the related palette, change the representation (for numerics), change the display type (for strings, ie to hex display), access the properties window.

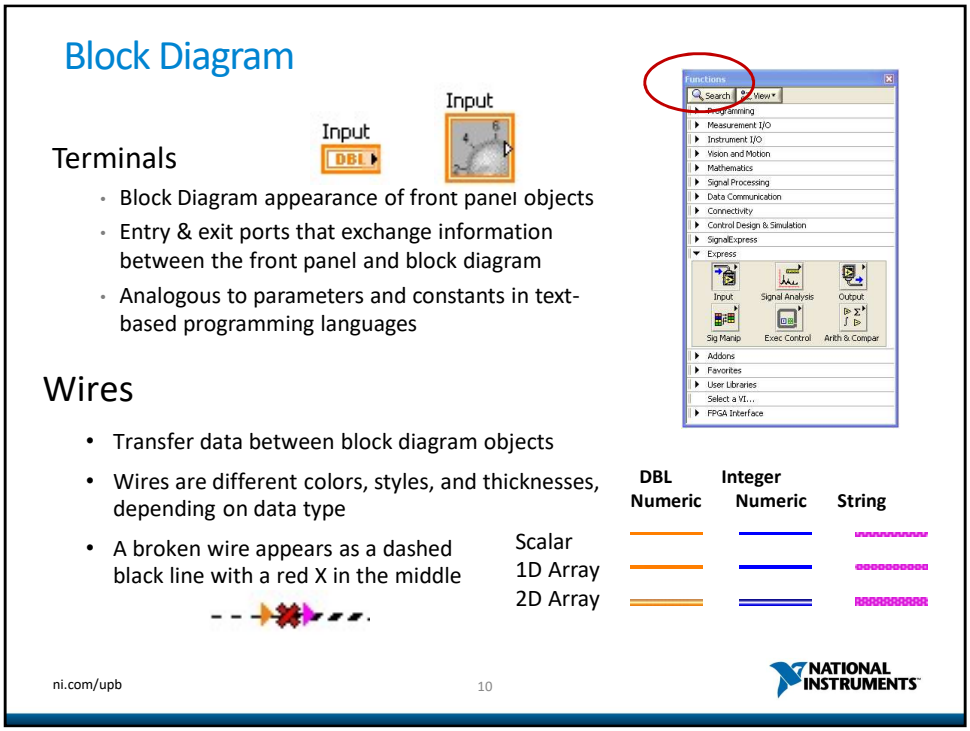
The properties window allows you to configure the object. Many of the commonly-changed properties are available from the short-cut menu. Properties include: color of the object, text on the object (buttons), representation (numerics), data range, descriptions.

Note: if multiple items are selected, the properties window will only display the properties shared amongst all of the objects. For example, if two numerics are selected and the properties window is opened, then it will be possible to set the data range. However, if a string and a numeric are selected, data range will not be available for changing because data range is not a property of a string.



The block diagram toolbar is very similar to the front panel with a few extras. Most important are:
- Highlight Execution: slows down code execution and displays beads on the wires as data flows through them. Output values are also displayed in small boxes. Great tool for debugging (except when debugging timing issues- it slows the execution for you to visualize, so timing will not be correct)
- Clean Up Block Diagram: arrange block diagram objects in an orderly fashion (according to data flow) and eliminates bends in wires. It's not perfect, but it does a pretty good job. Note that it will take longer to clean up block diagrams with many objects

(note that the Context Help button is not displayed on the slide, but is present on the BD toolbar)



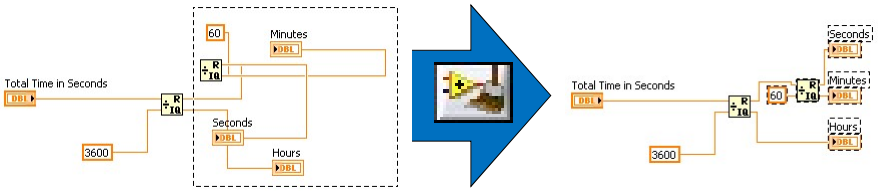
Terminals:
Terminals are the connection to front panel objects. Two views: small icon and large icon (Tools>>Options to change the default view). The color represents the data type of the terminal. The terminals on this slide have a small arrow on the right side pointing out, which tells you that it is connected to a control (data comes from the terminal, out to the right) A terminal connected to an indicator will have an arrow on the left side pointing in (receives data from the wire to be sent to the front panel)

Wires:
Do not be afraid of broken wires! Hover over them with Context Help on and Context Help will tell you why the wire is broken (eg datatype mismatch)

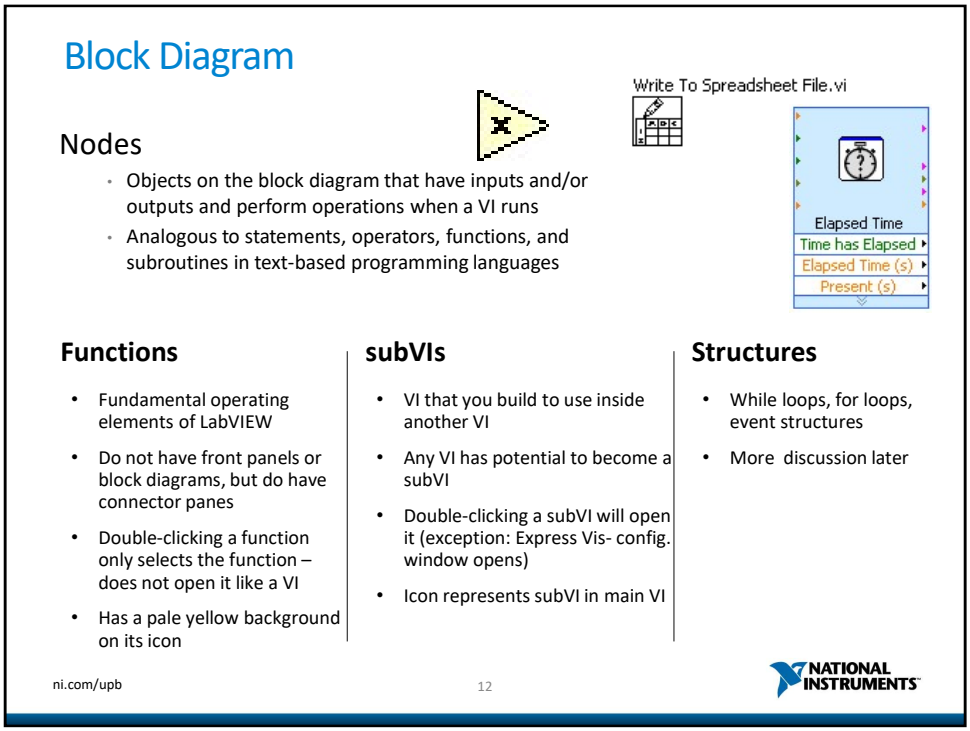
Use the **Functions** palette to build the block diagram. The **Functions** palette is available *only* on the block diagram. To view the palette, select **View»Functions Palette**. You also can display the **Functions** palette by right-clicking an open area on the block diagram. Tack down the **Functions** palette by clicking the pushpin on the top left corner of the palette.

Block Diagram: Wiring Tips

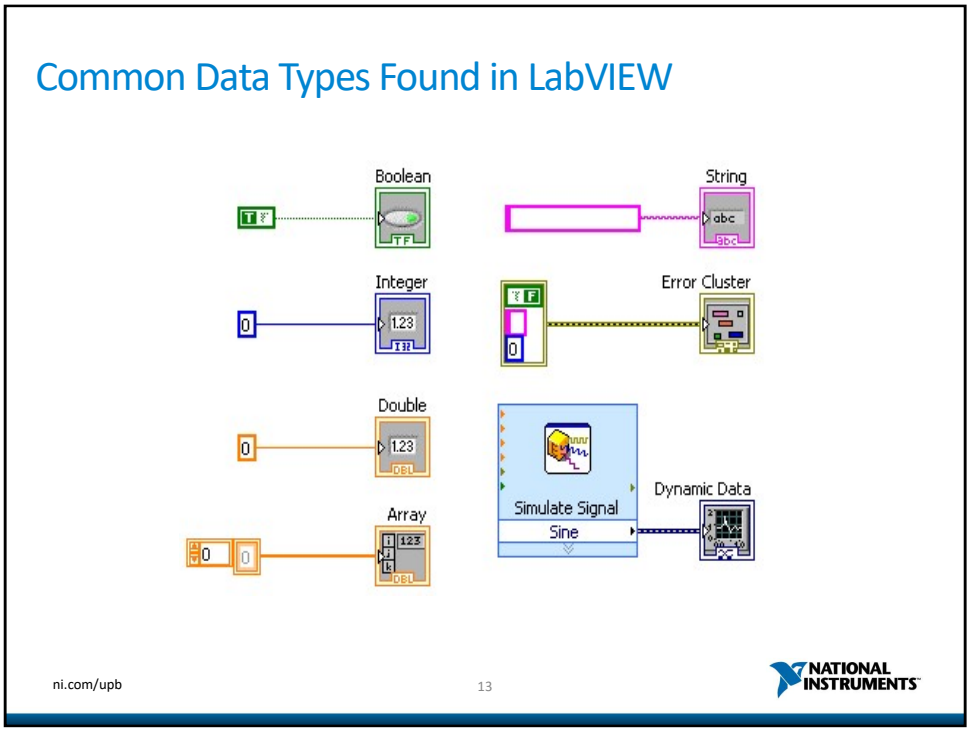
- Press <Ctrl>-B to delete all broken wires
- Right-click and select **Clean Up Wire** to reroute the wire
- Use the Clean Up Diagram tool to reroute multiple wires and objects to improve readability
  - Select a section of your block diagram
  - Click the Clean Up Diagram button on the block diagram toolbar (or <Ctrl>-U)



Note: Block diagram cleanup is only available in LV version 8.6 and later (2009, 2010 have partial block diagram clean up, which allows you to select a region to clean up as show in this slide)



An express VI has a configuration window that pops up when you double click it. Values wired into an Express VI will override the configured setting (for example, if you configure the express vi to output a sine wave, then wire in a setting for a triangle wave, the express vi will output a triangle wave.



LabVIEW uses many common data types:

Boolean, numeric, strings, clusters, and so on.

The color and symbol of each terminal indicate the data type of the control or indicator. Control terminals have a thicker border than indicator terminals. Also, arrows appear on front panel terminals to indicate whether the terminal is a control or an indicator. An arrow appears on the right if the terminal is a control and on the left if the terminal is an indicator.

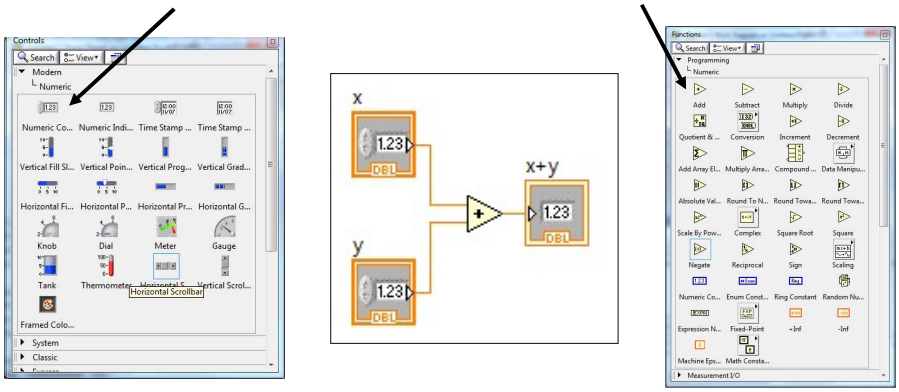
Dynamic data is used in conjunction with Express VIs.

The *LabVIEW User Manual* on `ni.com` provides additional references for data types found in LabVIEW.

Numeric Controls and Functions

(Front Panel) From the **Controls»Modern»Numeric** subpalette, select the **Numeric Control** icon.

(Block Diagram) From the **Functions»Programming»Numeric** subpalette, select the **Add** icon.



ni.com/upb

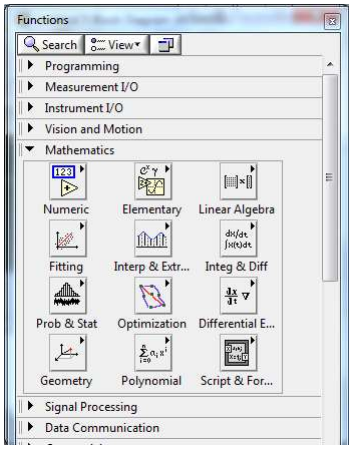
14

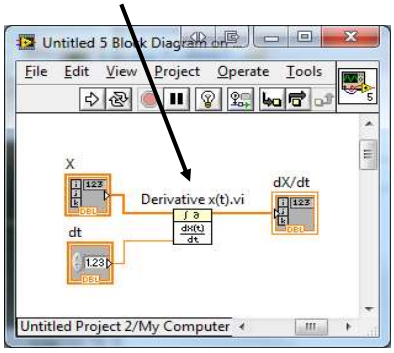
NATIONAL INSTRUMENTS

Arithmetic controls including numeric controls, vertical pointers, and knobs can be accessed through the **Controls»Modern»Numeric** subpalette and placed on the front panel. These controls are operated on by the numeric functions in the block diagram, which is accessed through the **Functions»Programming»Numeric** subpalette.

Mathematical Operations

(Block Diagram) From the **Functions»Mathematics»Integration and Differentiation** subpalette, select the **Derivative x(t).vi**





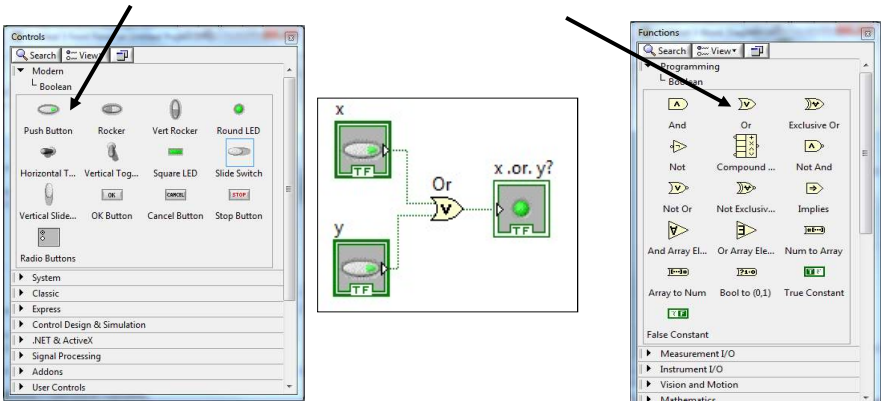
ni.com/upb 15

Use the Mathematics VIs to perform many different kinds of mathematical analysis. You also can interface real-world measurements to the mathematical algorithms in order to obtain practical solutions. Mathematical analysis in LabVIEW include, but in not limited to, Linear Algebra, Interpolation, Integration and Differentiation, Optimization, etc.

Boolean Controls and Functions

(Front Panel) From the **Controls»Modern»Boolean** subpalette, select the **Push Button** icon.

(Block diagram) From the **Function»Programming»Boolean** subpalette, select the **OR** icon.



ni.com/upb

16

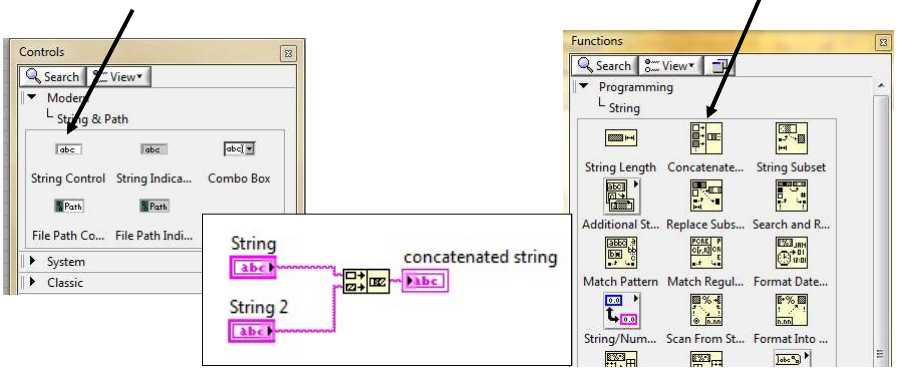
NATIONAL INSTRUMENTS

Boolean logic is also implementable in LabVIEW. From the front panel, you can access boolean controls, such as push buttons, toggle switches, and sliders and boolean indicators, such as LEDs. Boolean controls and indicators are found via the **Controls»Modern»Boolean** subpalette on the front panel. Boolean logic functions, such as AND, OR, and NOT functions, operate on these boolean controls. You can access the boolean functions on from the Function**»Programming»Boolean** subpalette on the block diagram.

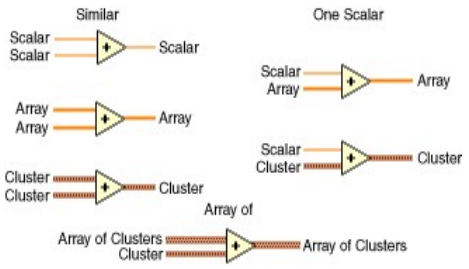
String Controls and Functions

(Front Panel) From the **Controls»Modern»String** subpalette, select the **String Control** icon.

(Block diagram) From the **Function»Programming»String** subpalette, select the **Concatenate** icon.



Polymorphism

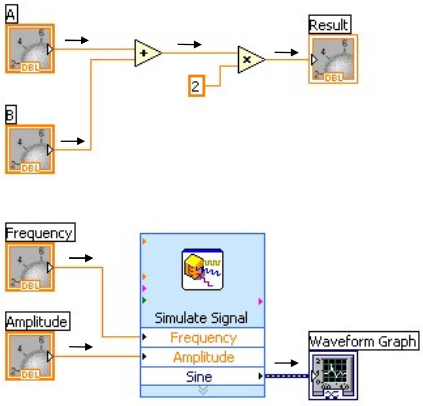


- **Definition**: a programming language feature that allows values of different data types to be handled using a uniform interface.
- **In LabVIEW**: the ability of VIs and functions to automatically adapt to accept input data of different data types
  - i.e. Numeric Functions
  - Useful when performing the same operation on different data types

In short, polymorphism is the ability of a node to accept/handle various data types.

Data Flow

- Block diagram execution is dependent on the flow of data
- Block diagram does NOT execute left to right
- Node executes when data is available to ALL input terminals
- Nodes supply data to all output terminals when done
- If the computer running this code had multiple processors, these two pieces of code could run independently without additional coding



ni.com/upb 19 NATIONAL INSTRUMENTS

- It is good LV programming practice to have your program go from left to right, but data flow is not dependent on location of terminals or nodes on the block diagram.
- Node executes when data is available to ALL input terminals
- Nodes supply data to ALL output terminals when done (not before) - this includes loops! - a loop must completely finish all iterations before data will be passed out of its output tunnels
- Data flow being dependent on this and two major rules allows Multiple Independent sections of code to operate in parallel

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 multiplies by 2 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 Multiply function is not valid until the Add function has finished executing and passed the data to the Multiply function. Remember that a node executes only when all data are available at all of its input terminals, and it supplies data to its output terminals only when it finishes execution. In the second piece of code, the Simulate Signal Express VI receives input from the controls and passes its result to the graph.

You may consider the add-multiply and the simulate signal code to coexist on the same block diagram in parallel. This means that they begin executing at the same time and run independently of one another. If the computer running this code had multiple processors, these two pieces of code could run independently of one another (each on its own processor) without any additional coding.