Introduction to LabVIEW

TOPICS

A. What Is LabVIEW?
B. Background Information
C. LabVIEW Example Demo
D. Basic File Types
E. Parts of a VI
F. Front Panel Design
G. Block Diagram and Searching
H. Tool, Wiring and Debugging
I. LabVIEW Data Types
J. Program Flow Control

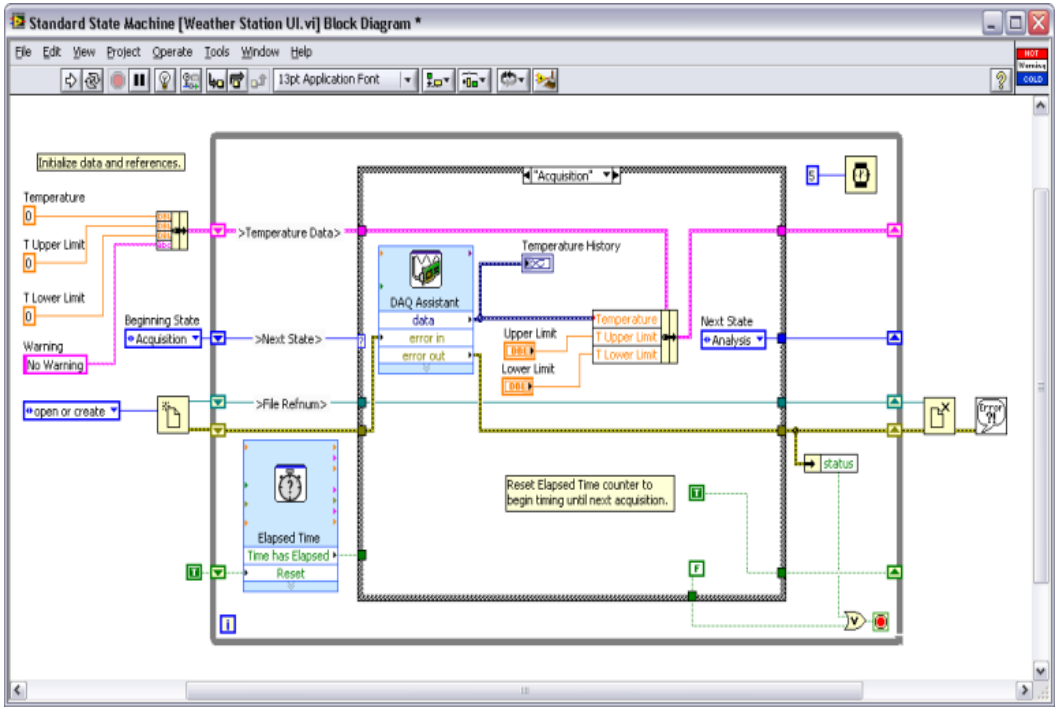
A. What Is LabVIEW?

Lab Virtual Instrument Engineering Workbench

LabVIEW Language Feature

What Is LabVIEW?

— A graphical programming environment used to develop sophisticated measurement, test, and control systems.



LabVIEW:

- Interfaces with wide variety of hardware
- Scales across different targets and OSs
- Provides built-in analysis libraries

LabVIEW Language Characteristics

Basic	Advanced
• Graphical • Dataflow-oriented • Multi-threaded • Multi-platform • Synchronous	• Event-driven • Object-oriented • Multi-language • Multi-target • Memory-Managed

B. Background Information

Hardware Integration

LabVIEW Projects

NI Hardware

PXI

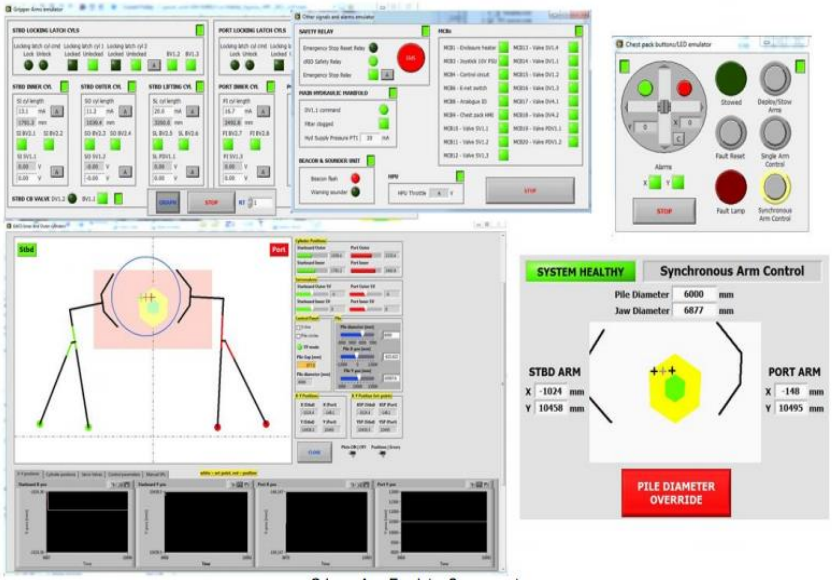
myRIO

cRIO

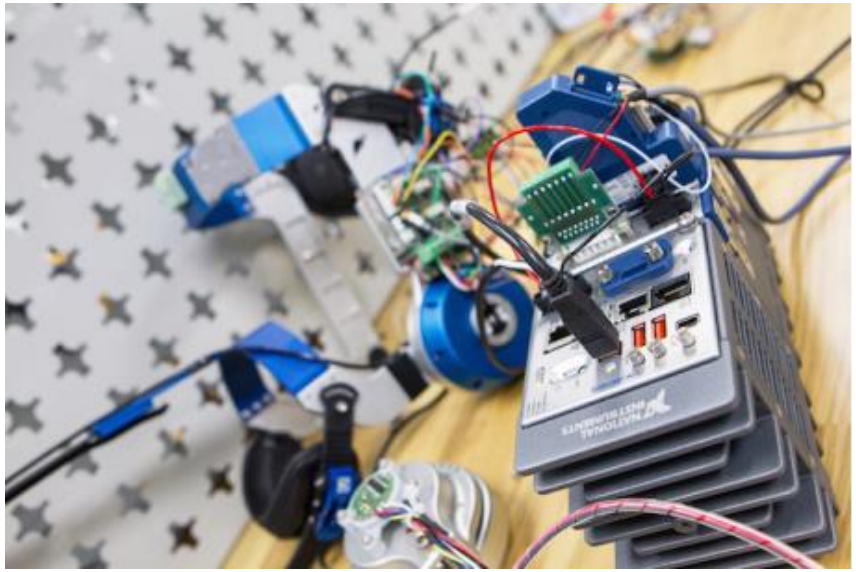
DAQ

SpaceX Control Center

Wind Turbine Installation



Wearable Robotics with cRIO



MyRIO System Overview

Features	Resources
• Portable • Programed with LabVIEW • Powerful • Multifunctional • Expandable	• Real Time System • On board FPGA • Wifi capability • ADC/DAC • UART/SPI/I2C

C. LabVIEW Example Demo

3D Bouncing Ball

2D Robot Manipulator

Quadcopter Dynamics

D. Basic File Types

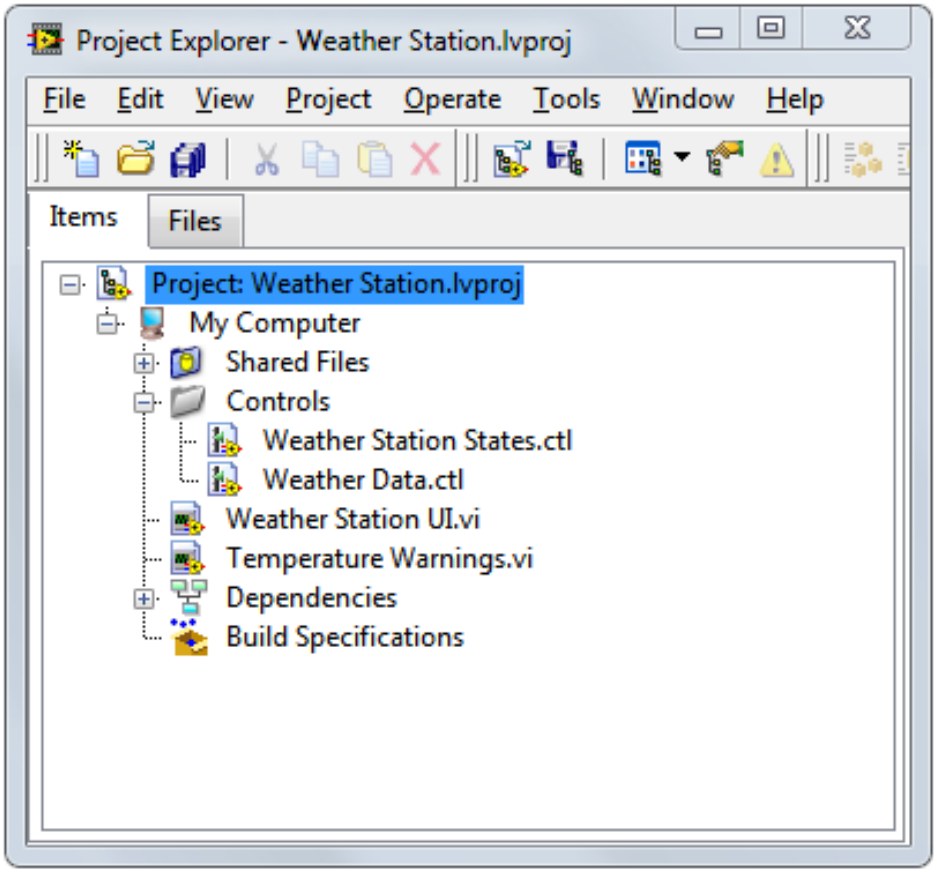
Project Explorer Window

Virtual Instrument

Custom Controls

Project Explorer

- Find, access, and organize project files
- Prevent, detect, and resolve incorrect links
- Deploy or download files to targets
- Manage code for build options
  - Executables, installers, and zip files
- Integrate with source code control providers



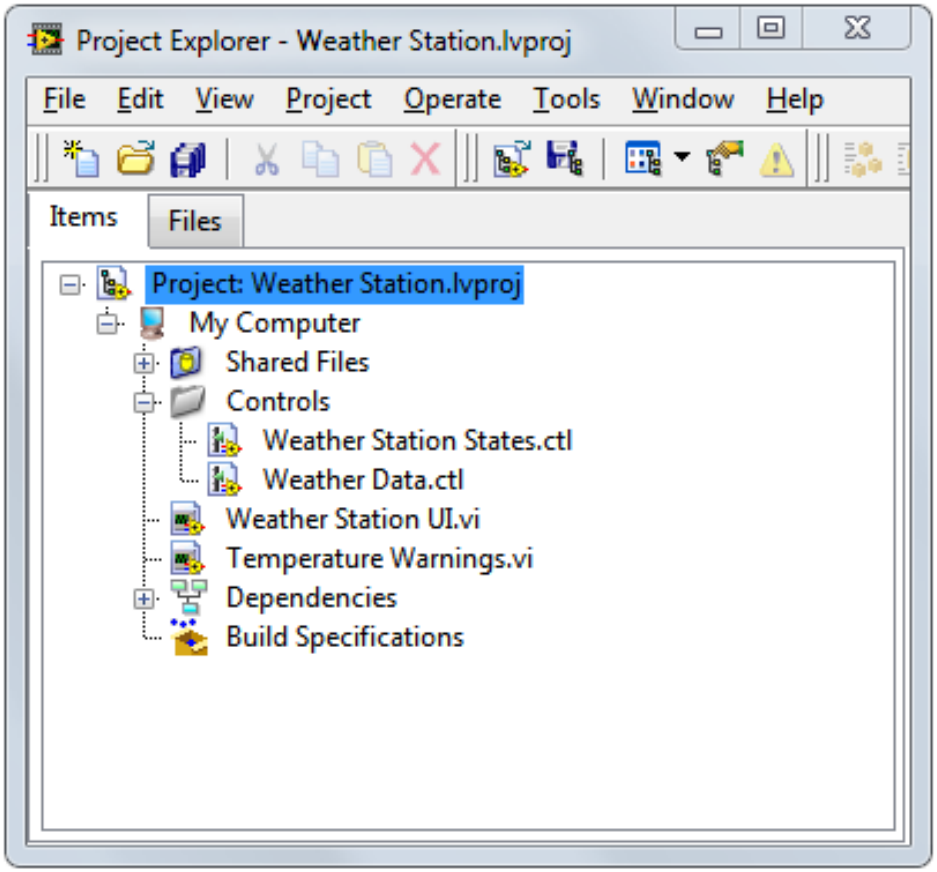
LabVIEW Files

Common LabVIEW file extensions:

LabVIEW project —.lvproj

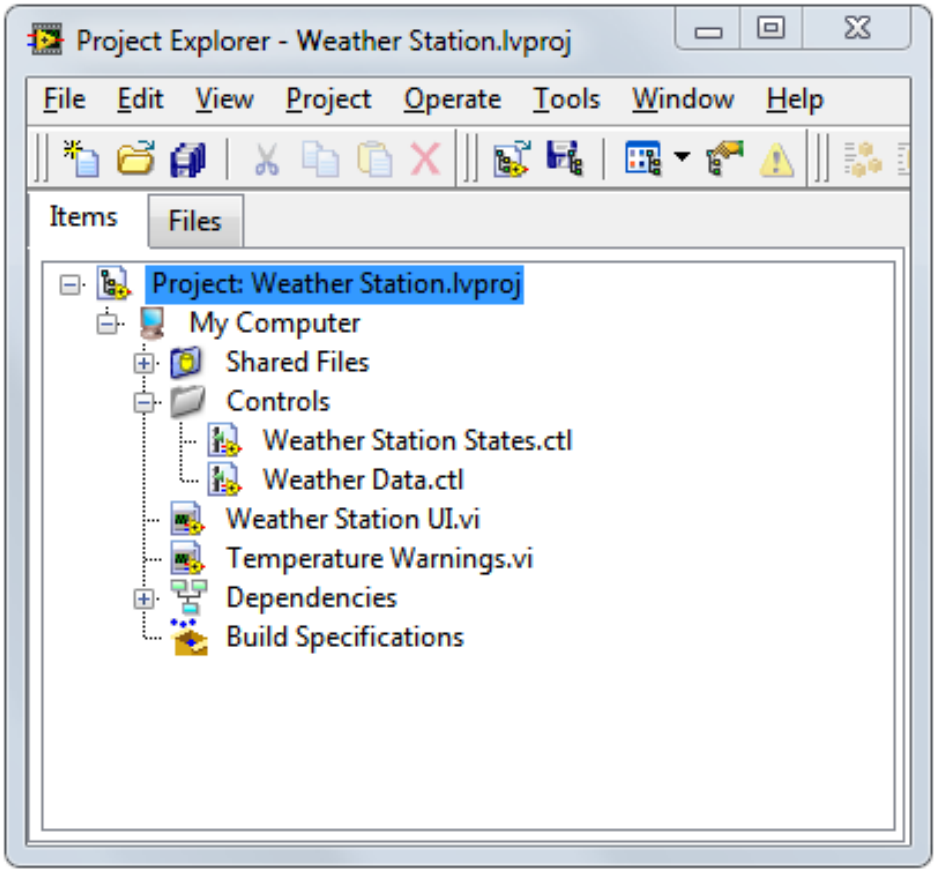
Virtual instrument (VI) — .vi

Custom control — .ctl



Adding Folders to a Project

- Virtual folder
  - Organizes project items and does not represent files on disk
- Auto-populating folder
  - Adds a directory on disk to the project
  - LabVIEW continuously monitors and updates the folder according to changes made in the project and on disk



E. Parts of a VI

Front Panel

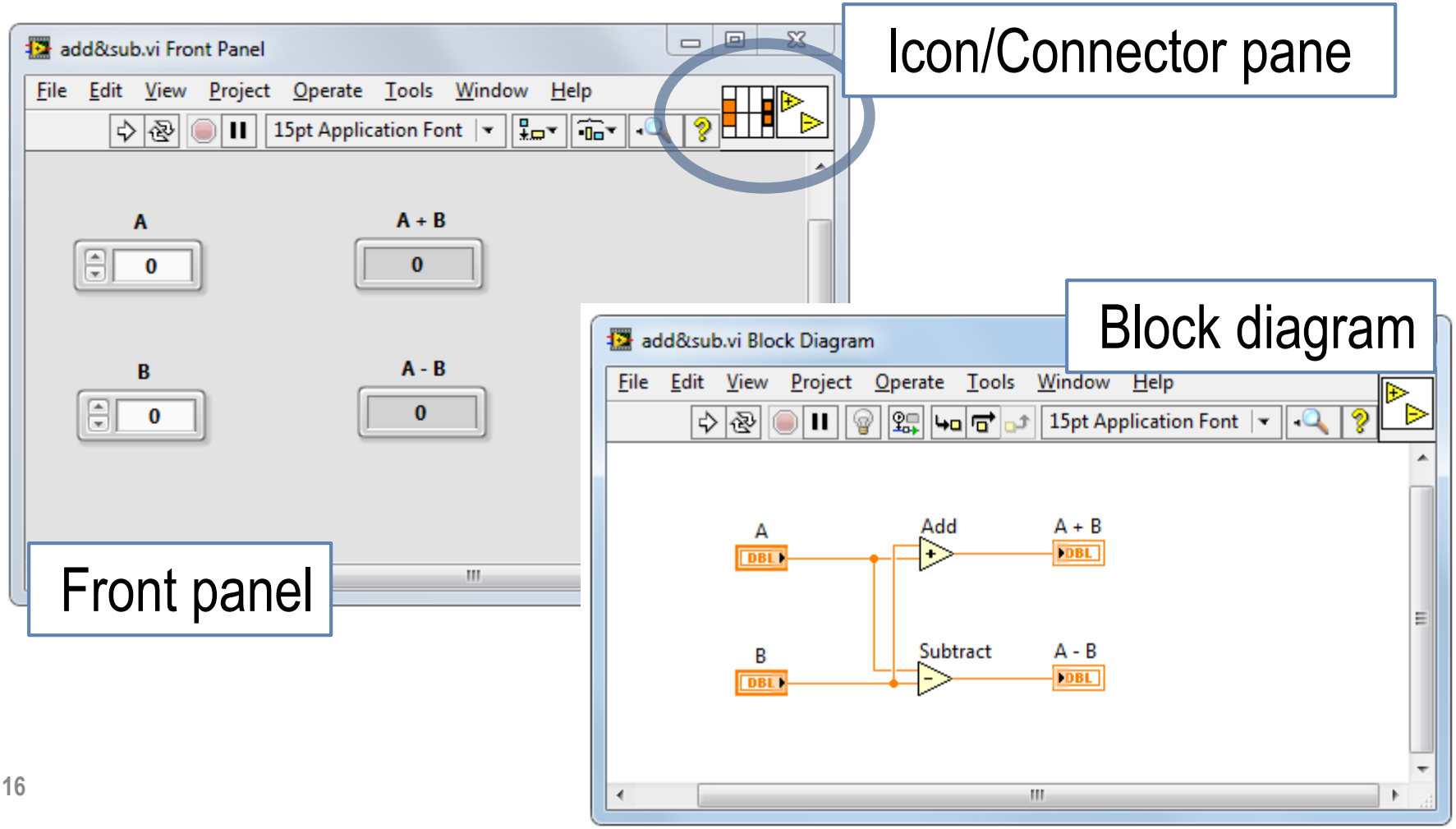
Block Diagram

Icon

Connector Pane

Parts of a VI

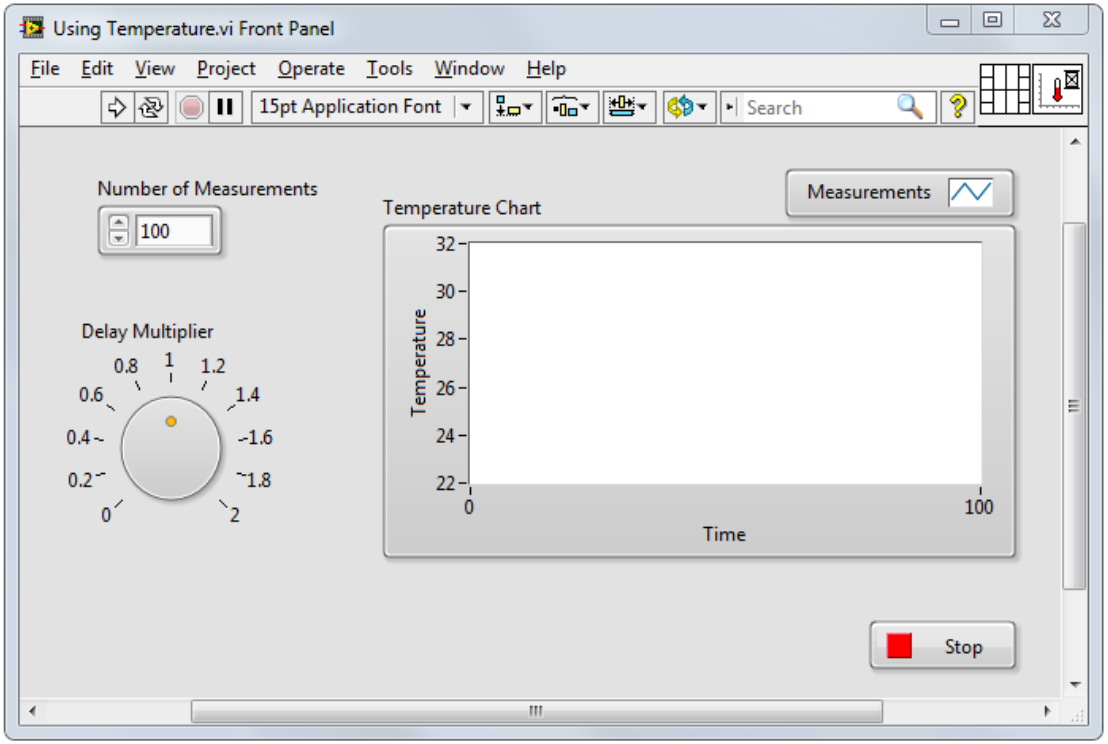
VIs have 3 main components:



Parts of a VI – Front Panel

Front Panel – User interface for the VI

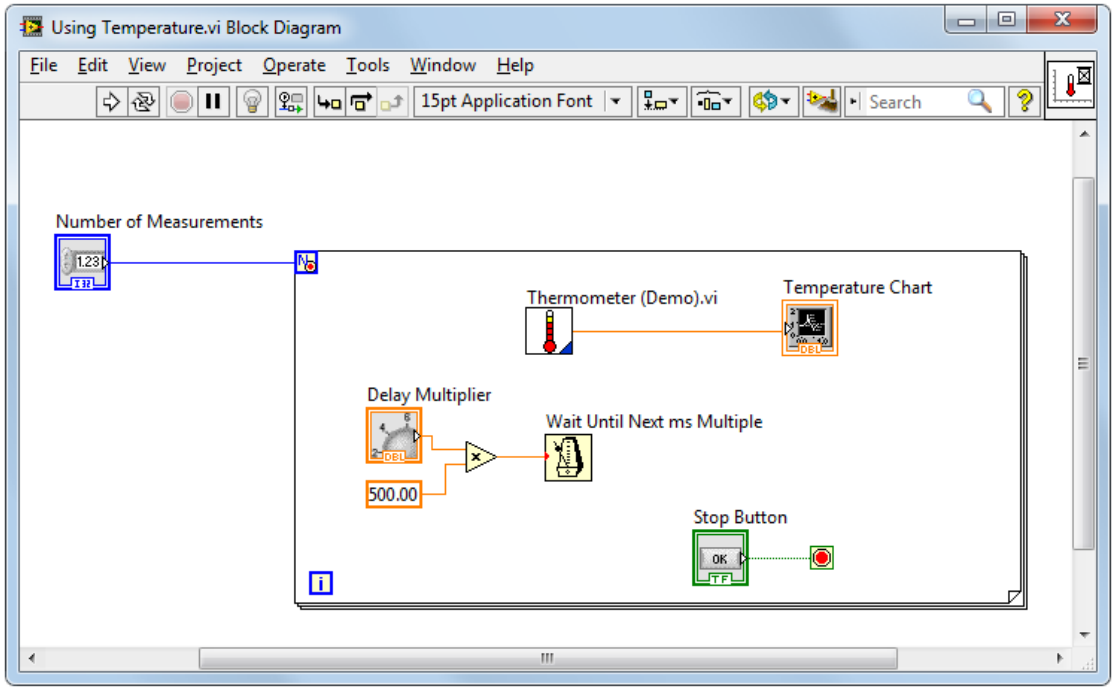
You build the front panel with controls (inputs) and indicators (outputs).



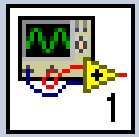
Parts of a VI – Block Diagram

Block Diagram – Contains the graphical source code

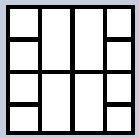
Front panel objects appear as terminals on the block diagram.



Parts of a VI – Icon/Connector Pane



Icon – Graphical representation of a VI



Connector Pane – Map of the inputs and outputs of a VI

Icons and connector panes are necessary to use a VI as a subVI.

- A subVI is a VI that appears on the block diagram of another VI.
- A subVI is similar to a subroutine or function in a text-based programming language.

F. Front Panel Design

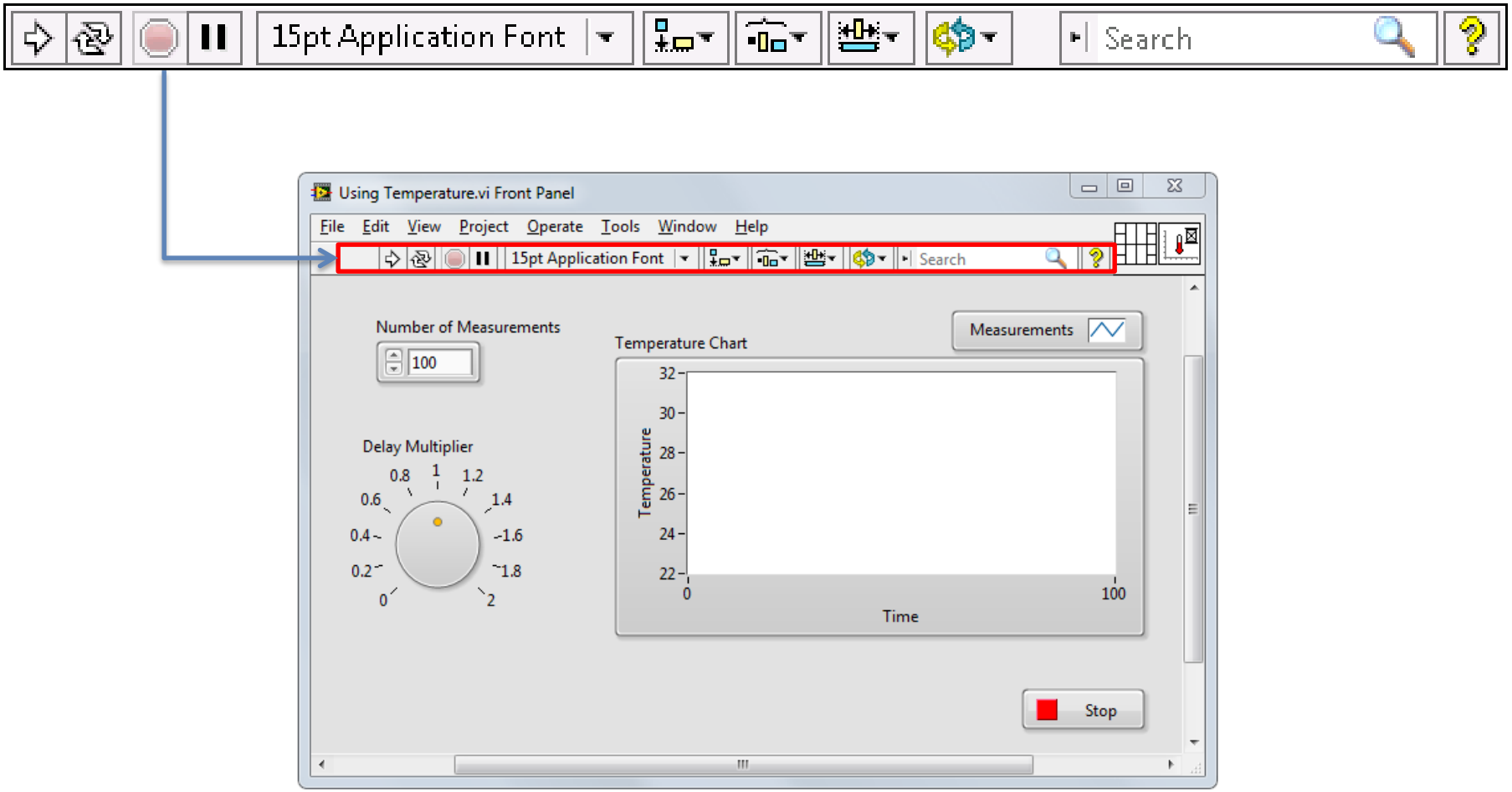
Controls and Indicators

Object Styles

Object Types

- Boolean
- Numeric
- String

Front Panel



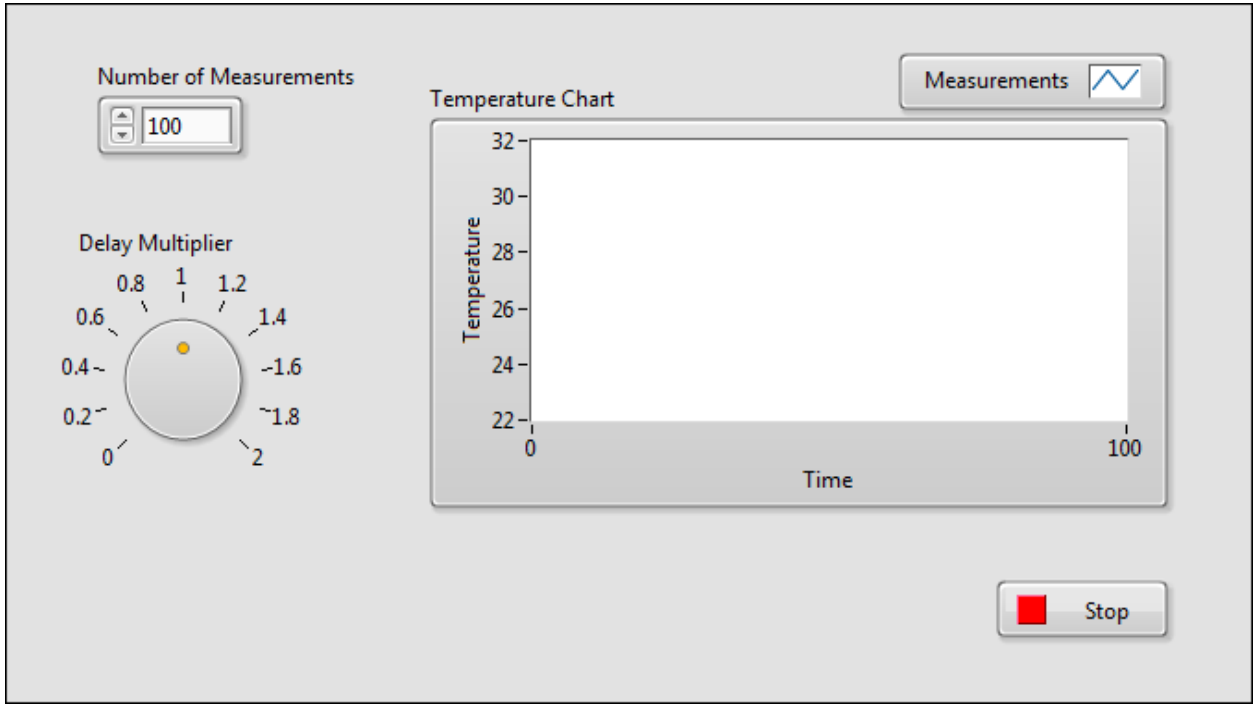
Controls and Indicators

Controls

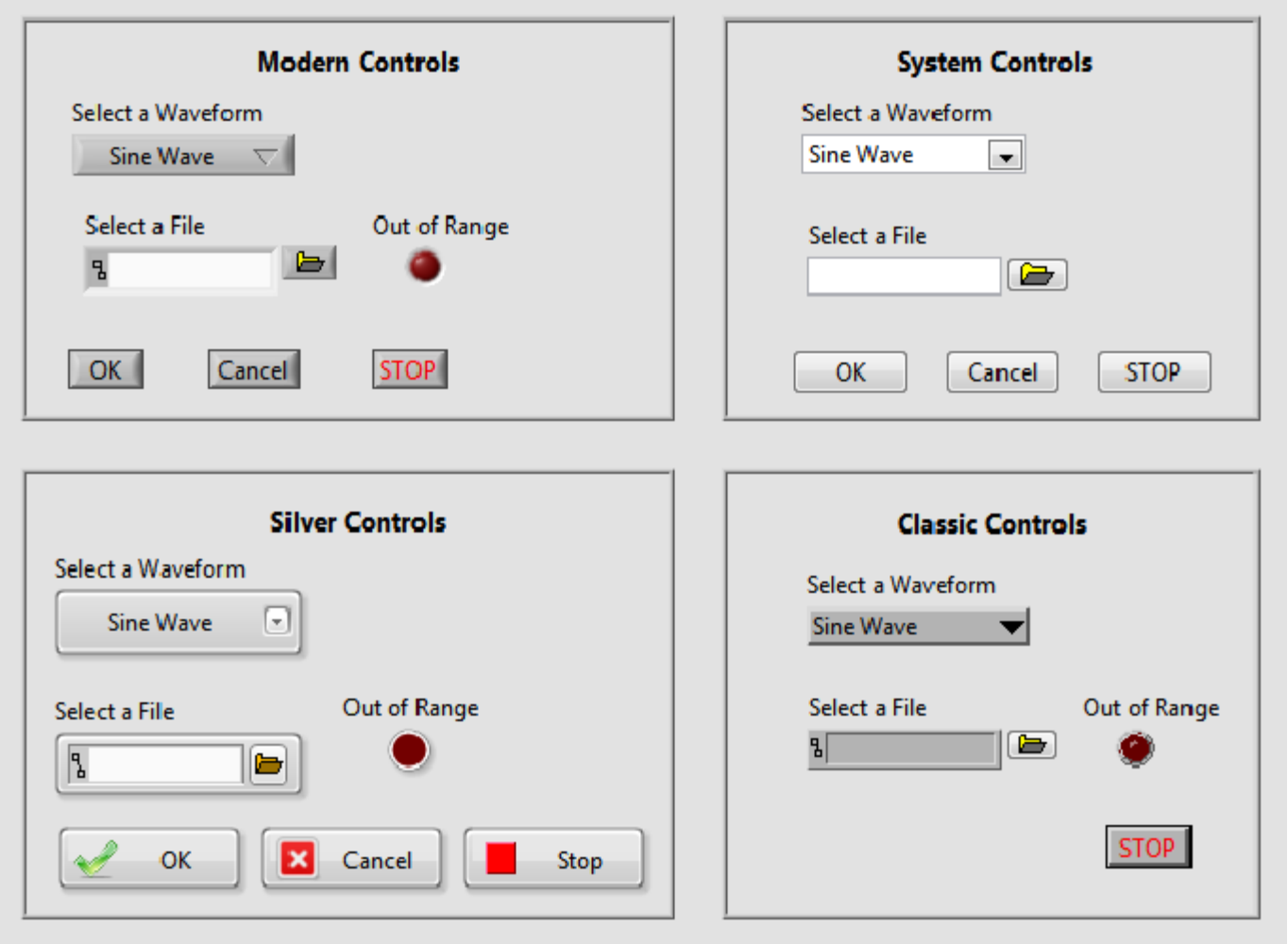
- Input devices
- Knobs, buttons, slides
- Supply data to the block diagram

Indicators

- Output devices
- Graphs, LEDs
- Display data the block diagram acquires or generates

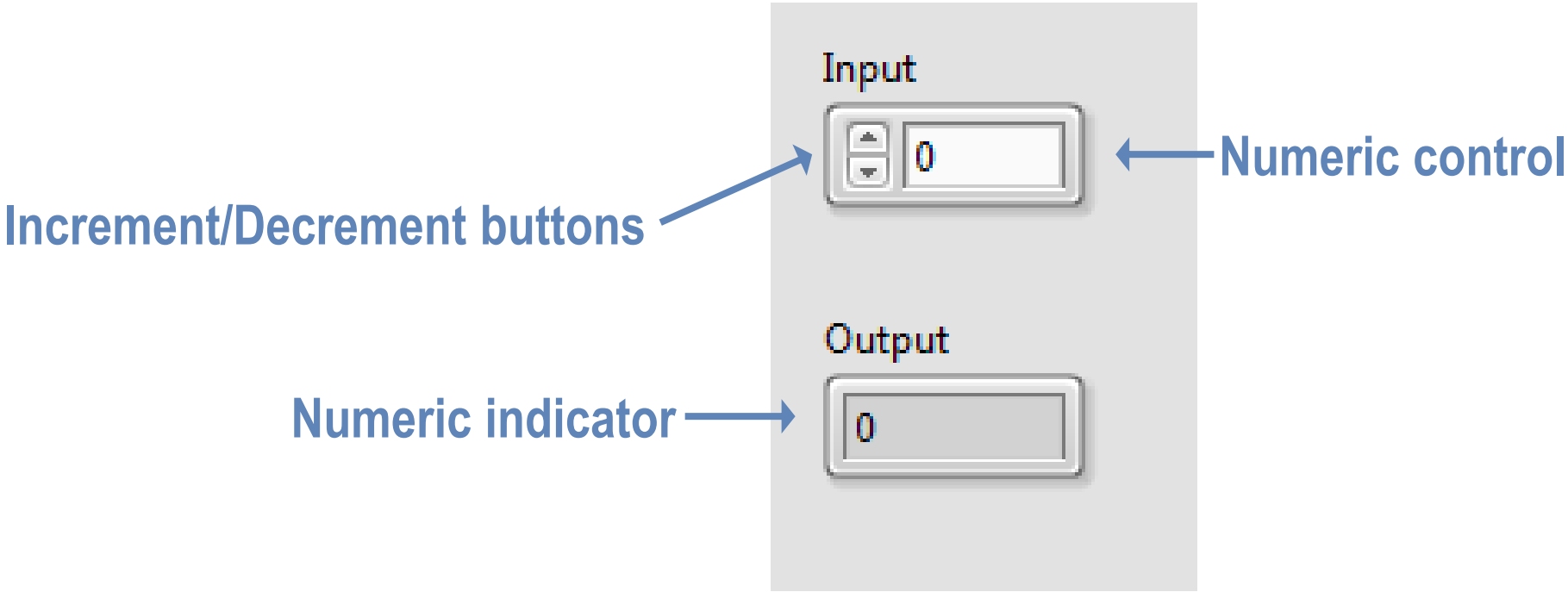


Front Panel Object Styles



Numeric Controls and Indicators

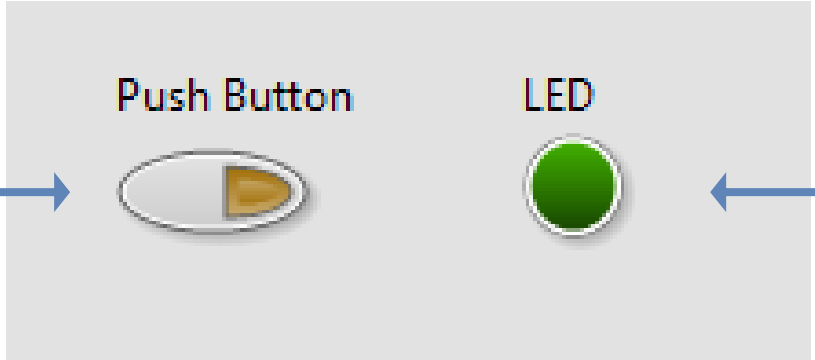
The numeric data in a control or indicator can represent numbers of various types, such as integer or floating-point.



Boolean Controls and Indicators

- The Boolean data type represents data that has only two options, such as True/False or On/Off.
- Use Boolean controls and indicators to enter and display Boolean (TRUE/FALSE) values.
- Boolean objects simulate switches, push buttons, and LEDs.

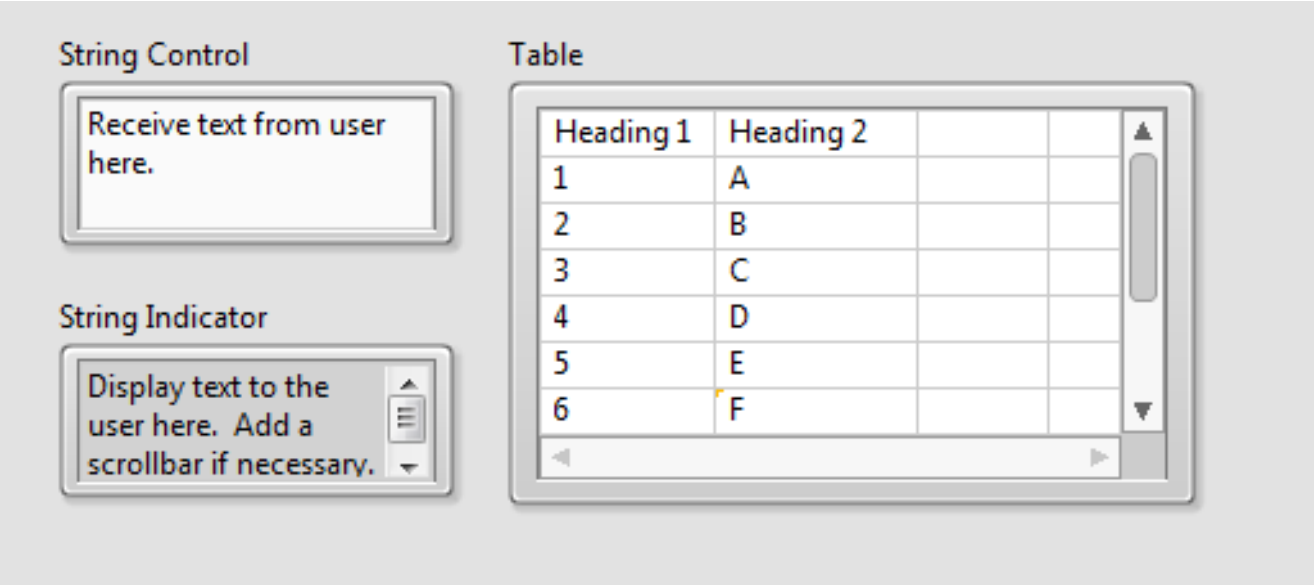
Boolean control



Boolean indicator

Strings

- The string data type is a sequence of ASCII characters .
- Use string controls to receive text from the user, such as a password or user name.
- Use string indicators to display text to the user.



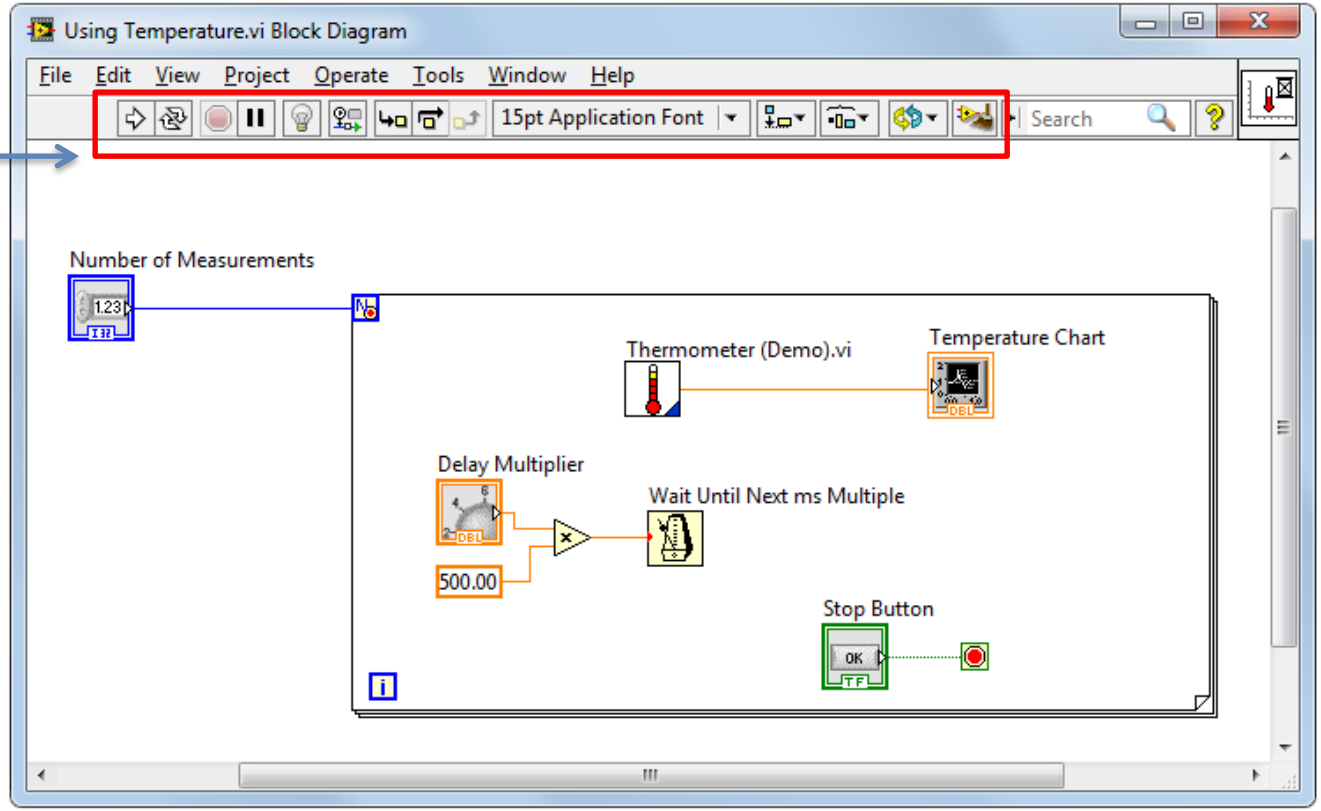
G. Block Diagram and Searching

Terminals, Nodes, SubVIs, Wires

Context Help, LabVIEW Help, Example

Searching on Palettes, Quick Drop

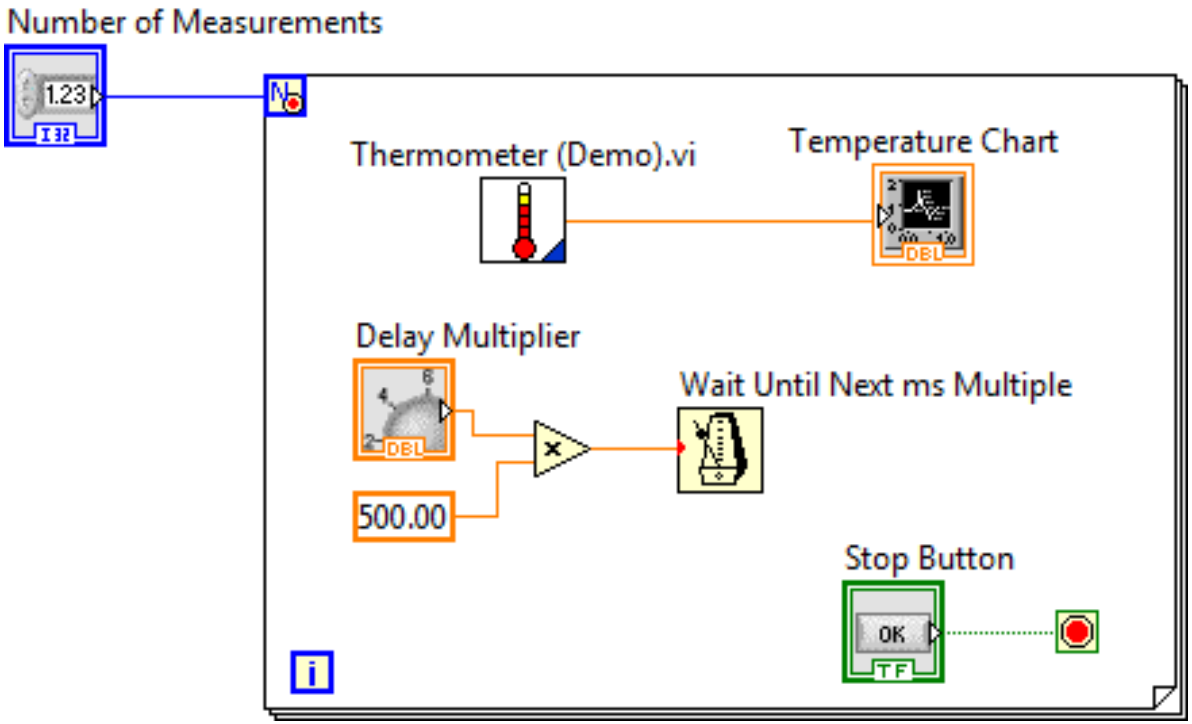
Block Diagram



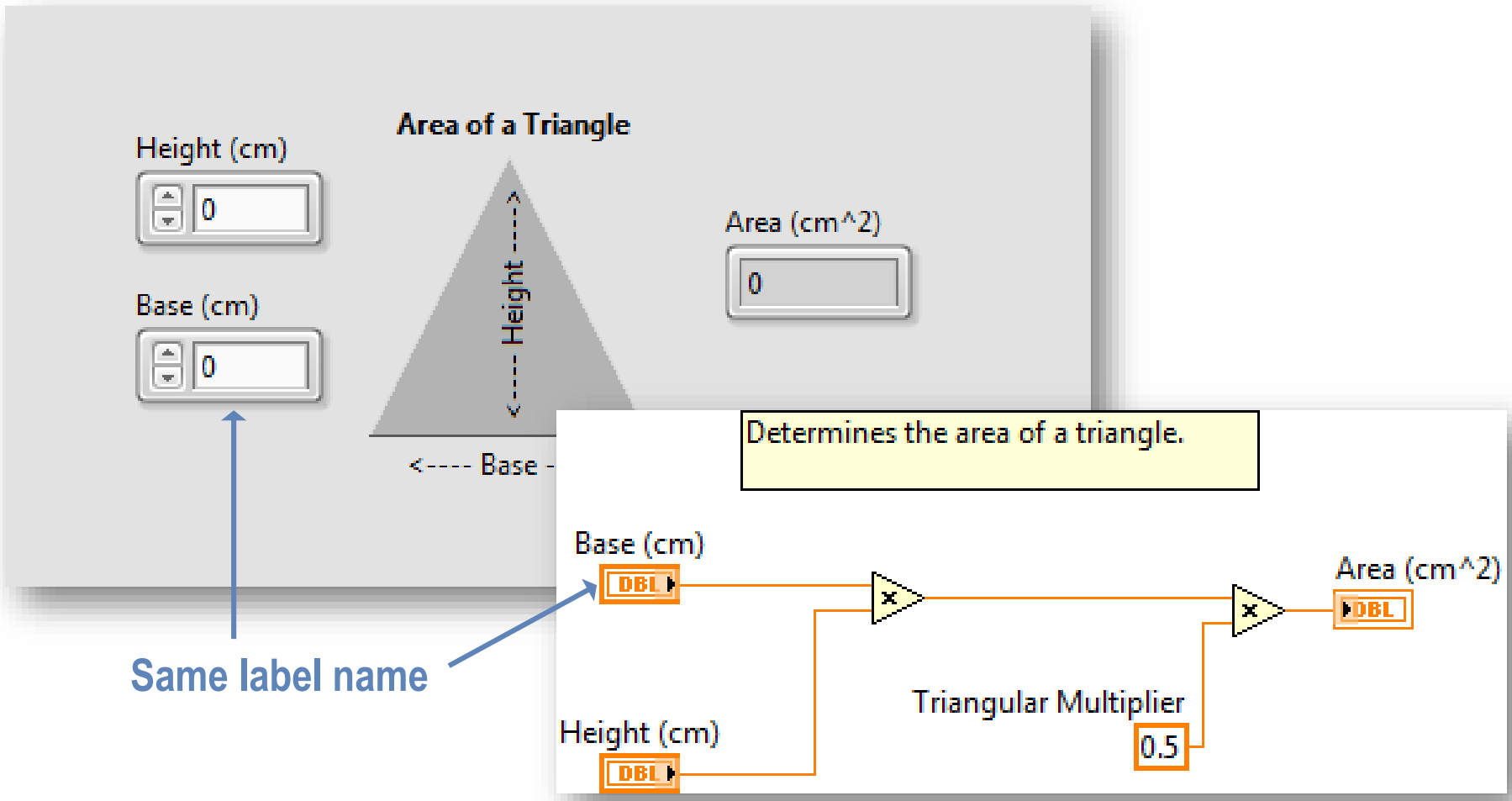
Block Diagram

Block diagram items:

- Terminals
- Constants
- Nodes
  - Functions
  - SubVIs
  - Structures
- Wires
- Free labels

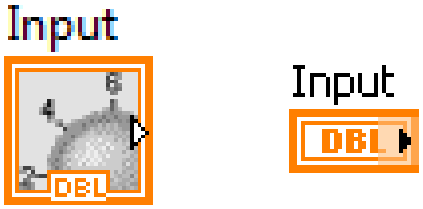


Terminals



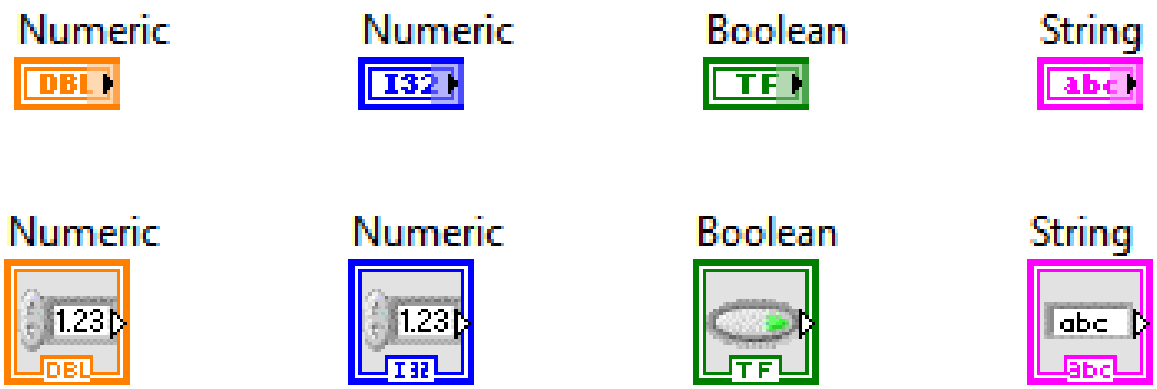
Terminals for Front Panel Objects

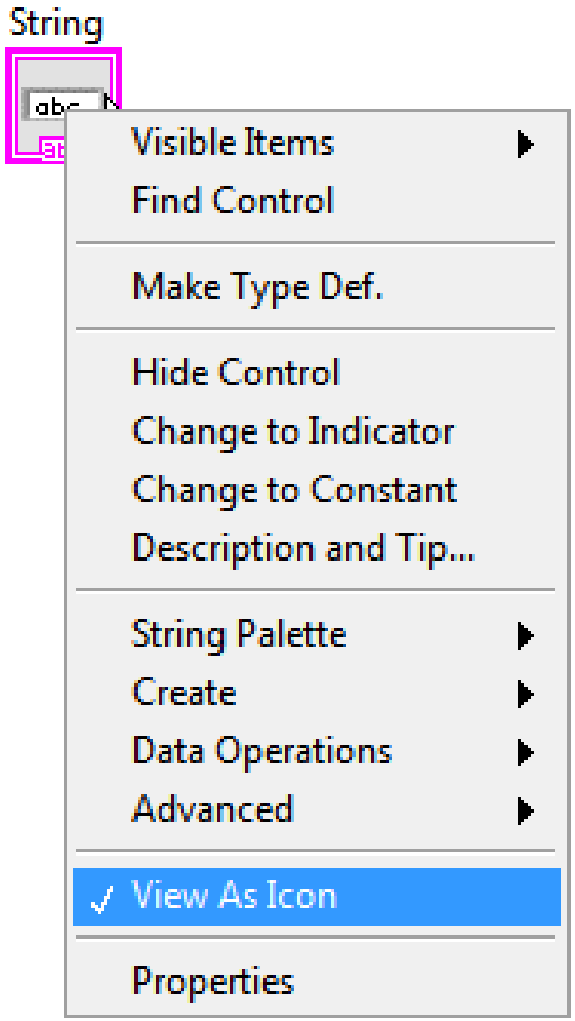
- Terminals are:
  - Entry and exit ports that exchange information between the front panel and block diagram.
  - Analogous to parameters in text-based programming languages.
- Double-click a terminal to locate the corresponding front panel object.



View Terminals as Icons

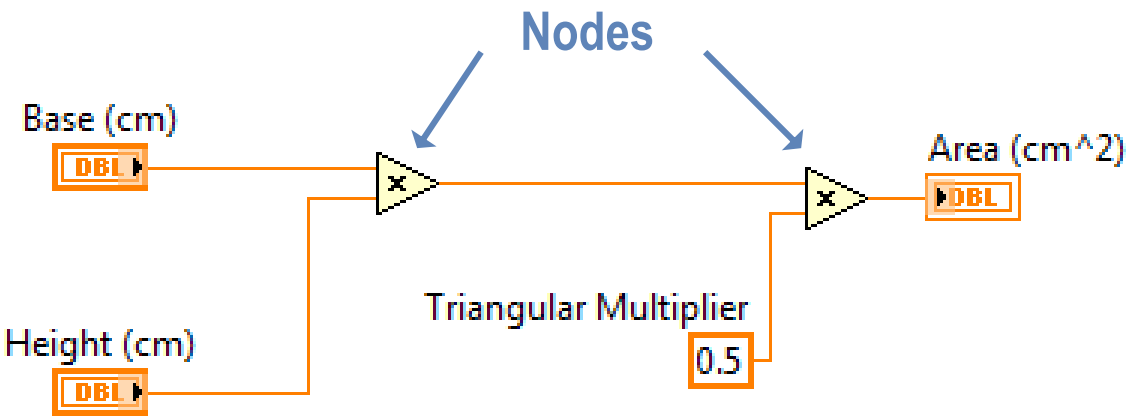
- By default, **View as Icon** option enabled.
- Deselect **View as Icon** for a more compact view.



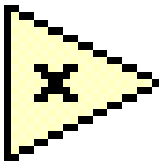


Nodes

Nodes are objects on the block diagram that have inputs and/or outputs and perform operations when a VI runs.

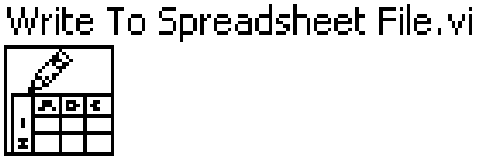


Function Nodes



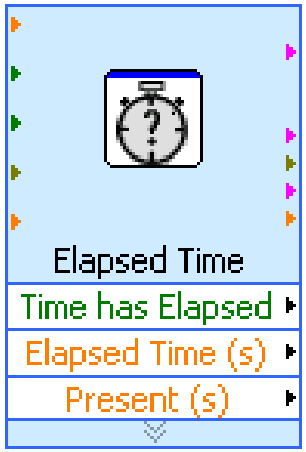
- Functions are:
  - Fundamental operating elements of LabVIEW.
  - Do not have front panels or block diagrams, but do have connector panes.
  - Has a pale yellow background on its icon.
- Double-clicking a function only selects the function.
- Functions do not open like VIs and subVIs.

SubVI Nodes



- SubVIs :
  - Are VIs that you use on the block diagram of another VI.
  - Have front panels and block diagrams.
  - Use the icon from the upper-right corner of the front panel as the icon that appears when you place the subVI on a block diagram.
- When you double-click a subVI, the front panel and block diagram open.
- Any VI has the potential to be used as a subVI.

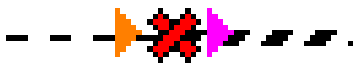
Express VIs



- Express VIs:
  - Are a special type of subVI.
  - Require minimal wiring because you configure them with dialog boxes.
  - Save each configuration as a subVI.
- Icons for Express VIs appear on the block diagram as icons surrounded by a blue field.

Wires

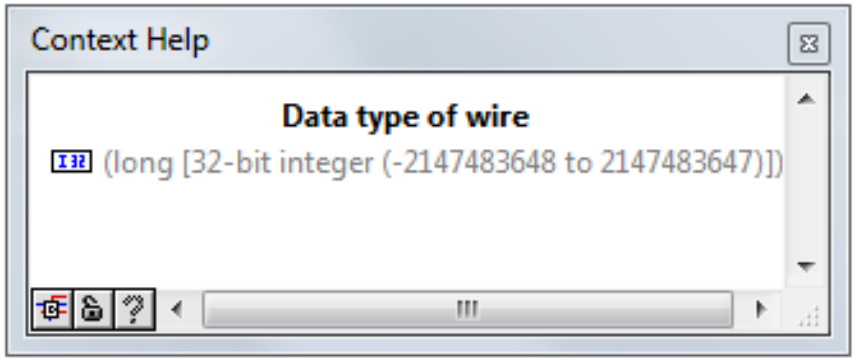
- Wires transfer data between block diagram objects.
- Wires are different colors, styles, and thicknesses, depending on their data types.
- A broken wire appears as a dashed black line with a red X in the middle.

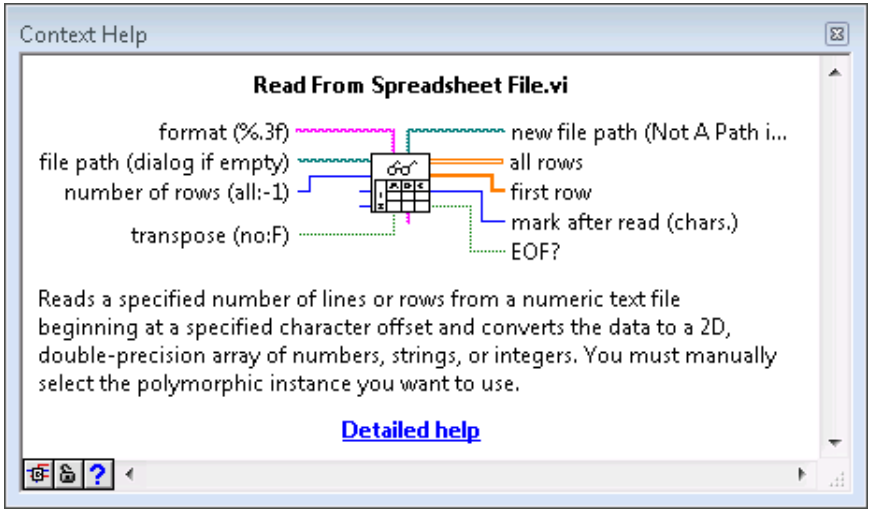


	Floating-point	Integer	String	Boolean
Scalar				
1-D Array				
2-D Array				

Context Help

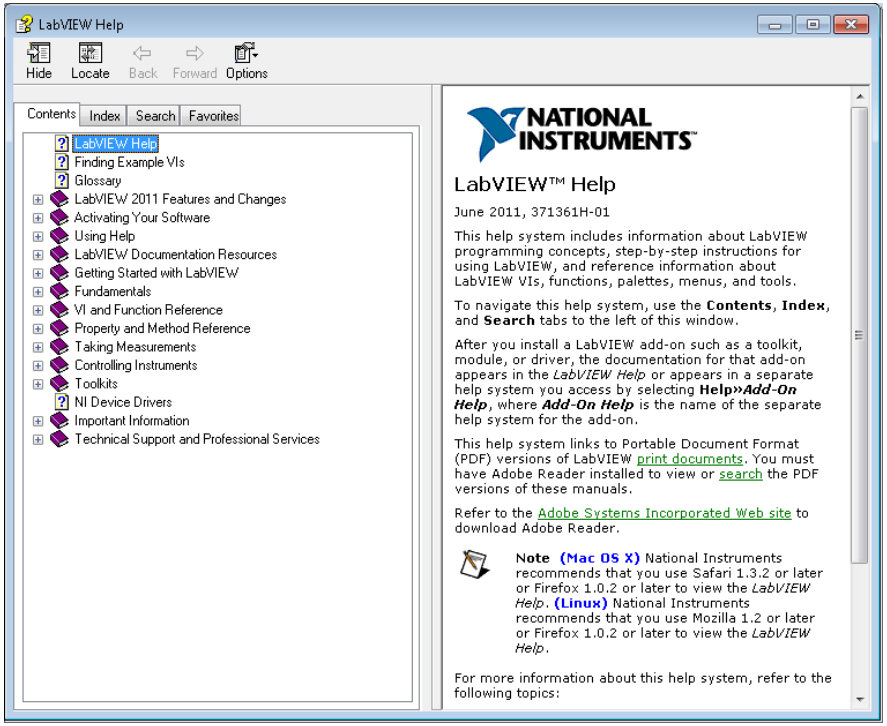
- Displays basic information about wires and nodes when you move the cursor over an object.
- Can be shown or hidden in the following ways.
  - Select **Help»Show Context Help** from the LabVIEW menu.
  - Press <Ctrl-H>.
  - Click the following button on the toolbar: 





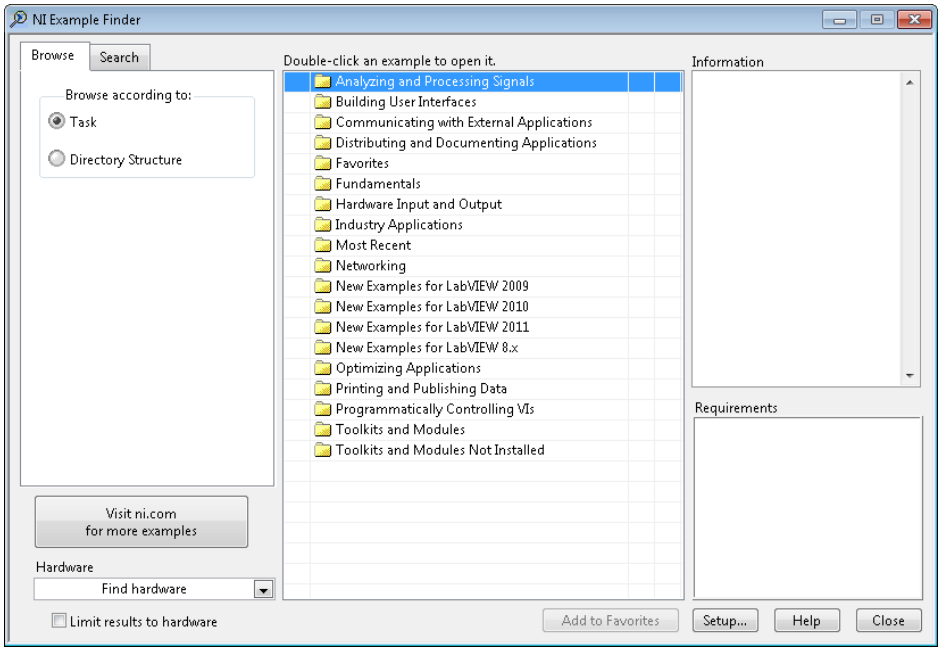
LabVIEW Help

- Contains detailed descriptions and instructions for most palettes, menus, tools, VIs, and functions.
- Can be accessed by:
  - Selecting **Help» LabVIEW Help** from the menu.
  - Clicking the **Detailed help** link in the **Context Help** window.
  - Right-clicking an object and selecting **Help** from the shortcut menu.



Examples

- LabVIEW includes hundreds of example VIs.
- Use NI Example Finder to browse and search installed examples.
  - Select **Help»Find Examples** in the menu.



- Click the example buttons in *LabVIEW Help* topics.

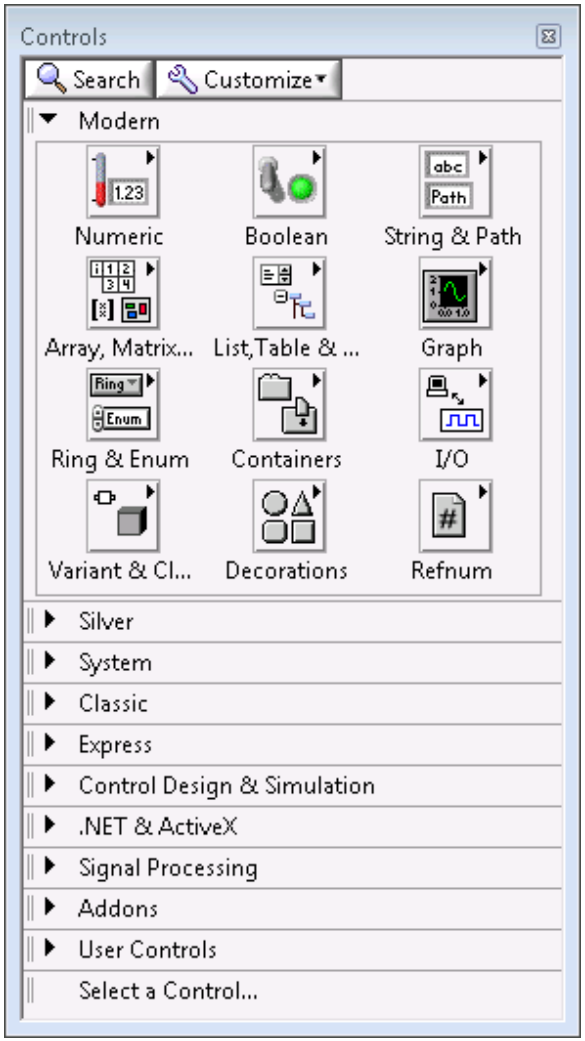
Searching for Controls, VIs, and Functions

Ways to find controls, VIs, and functions:

- Search or navigate the palettes.
  - Controls palette
  - Functions palette
- Search by name of object.
  - Quick Drop dialog box
- Search palettes, *LabVIEW Help*, and `ni.com`.
  - Search text box in toolbar

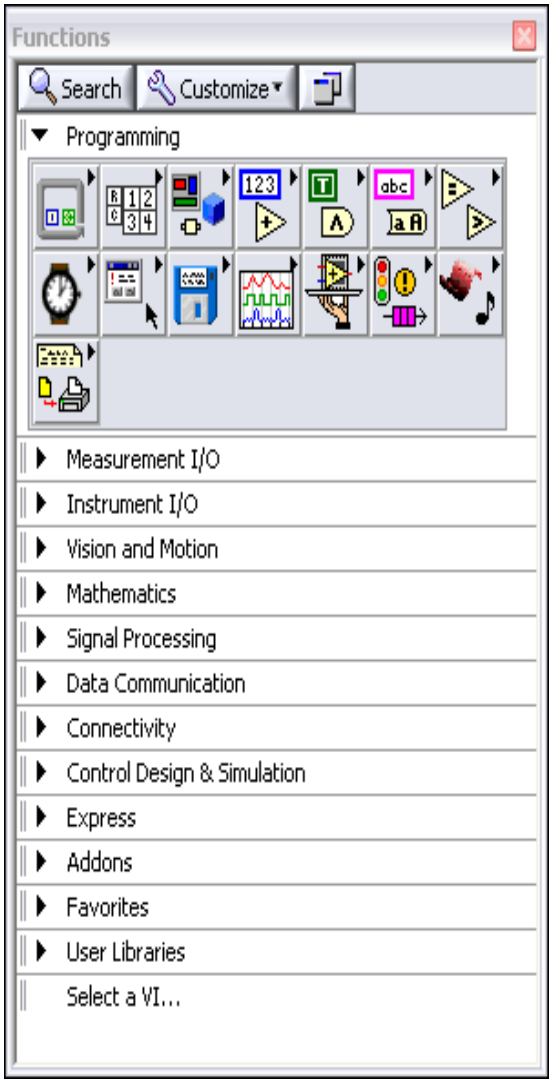
Controls Palette

- Contains the controls and indicators you use to create the front panel.
- Navigate the subpalettes or use the **Search** button to search the Controls palette.



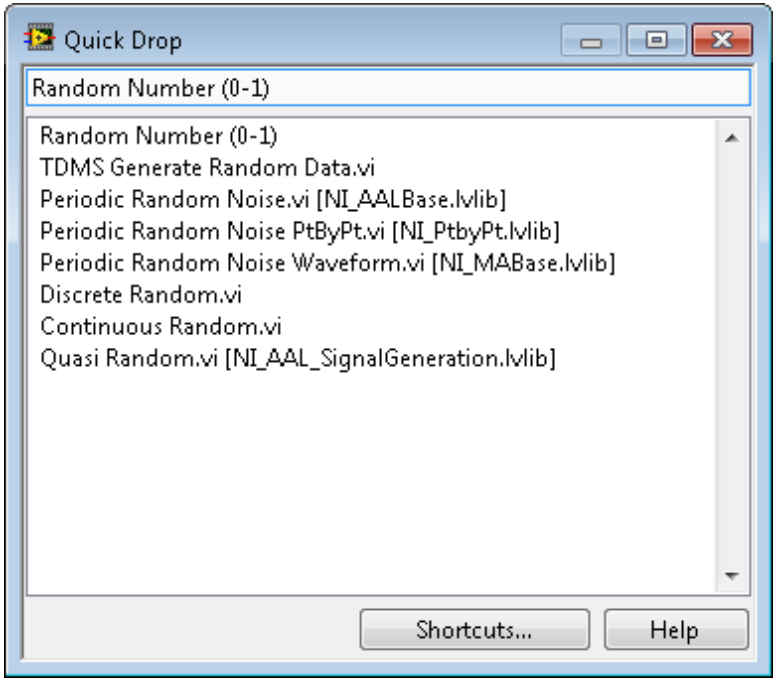
Functions Palette

- Contains the VIs, functions, and constants you use to create the block diagram.
- Navigate the subpalettes or use the **Search** button to search the Functions palette.



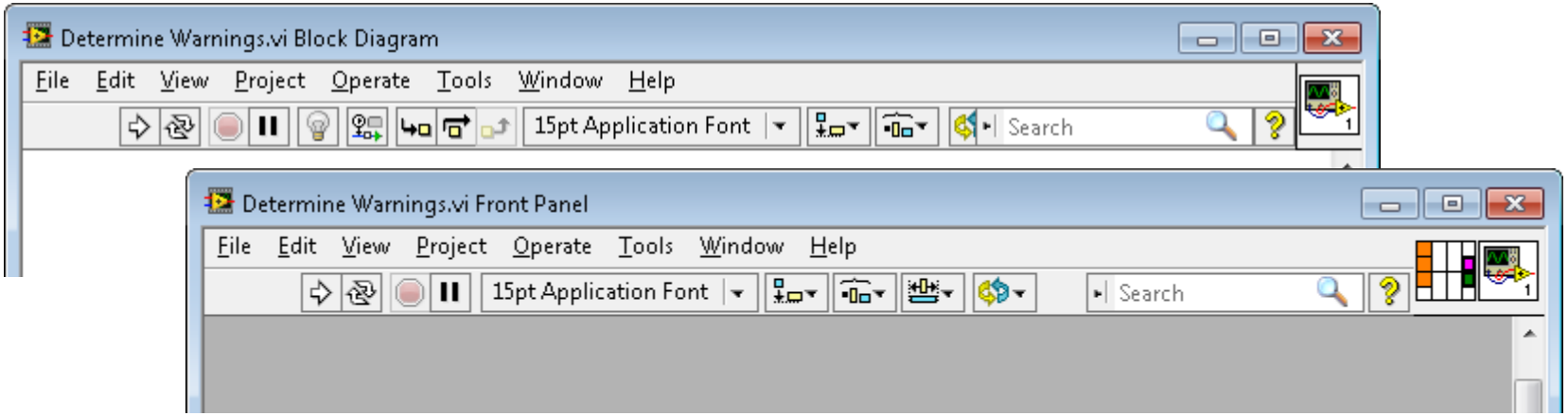
Searching with Quick Drop

- Lets you quickly find controls, functions, VIs, and other items by name.
- Press the <Ctrl-Space> keys to display the Quick Drop dialog box.



Global Search

Use the Search bar in the top right of the front panel and block diagram windows to search palettes, *LabVIEW Help*, and `ni.com`.



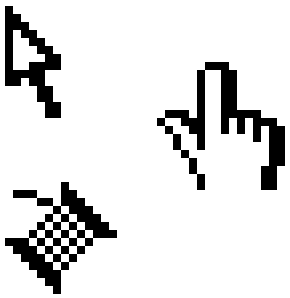
H. Tool, Wiring and Debugging

Selecting a Tool

Wiring Clean-Up

Debugging

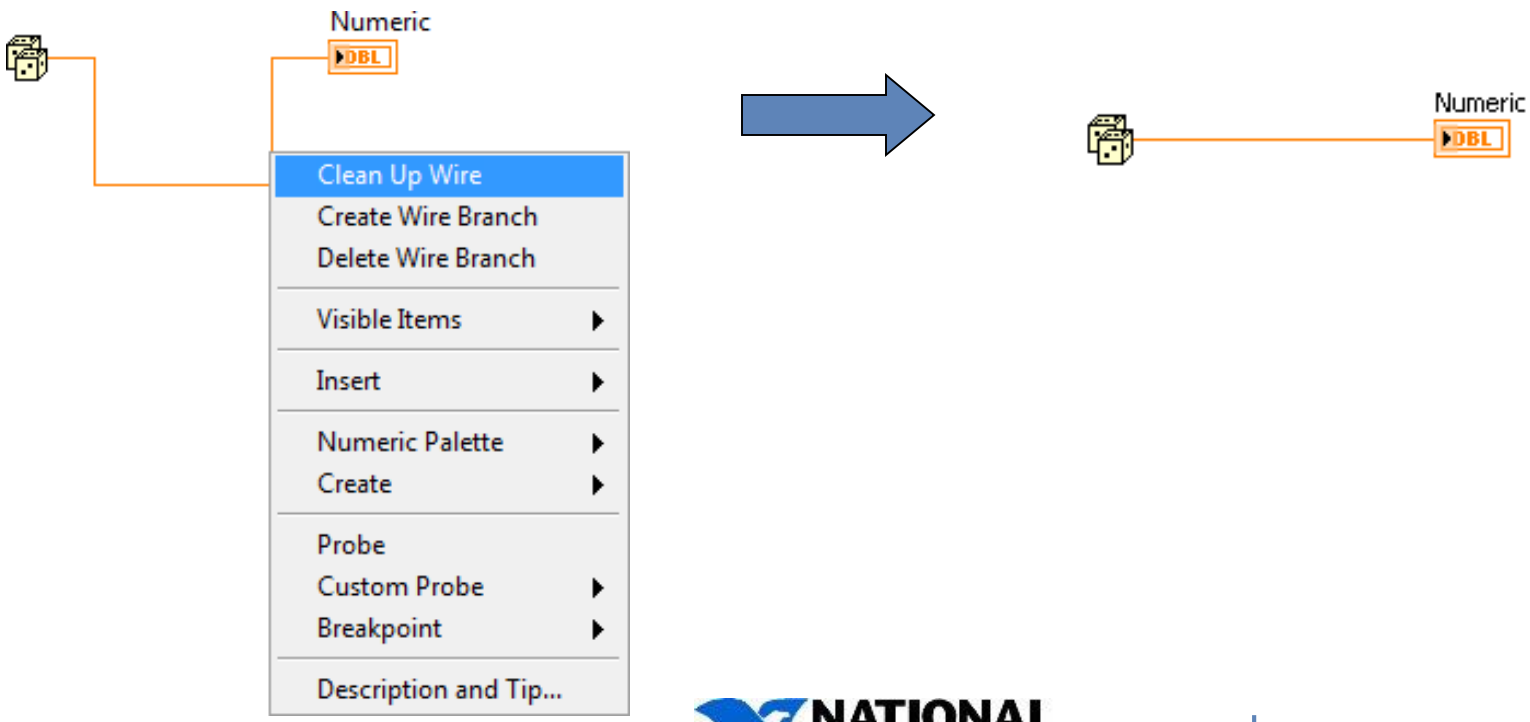
Selecting a Tool



- A tool is a special operating mode of the mouse cursor.
- Create, modify, and debug VIs using the tools provided by LabVIEW.
- By default, LabVIEW automatically selects tools based on the context of the cursor.
- If you need more control, use the **Tools** palette to select a specific tool.
  - Select **View»Tools Palette** to open the **Tools** palette.

Wiring Tips

- Press <Ctrl-B> to delete all broken wires.
- Right-click and select **Clean Up Wire** to reroute the wire.



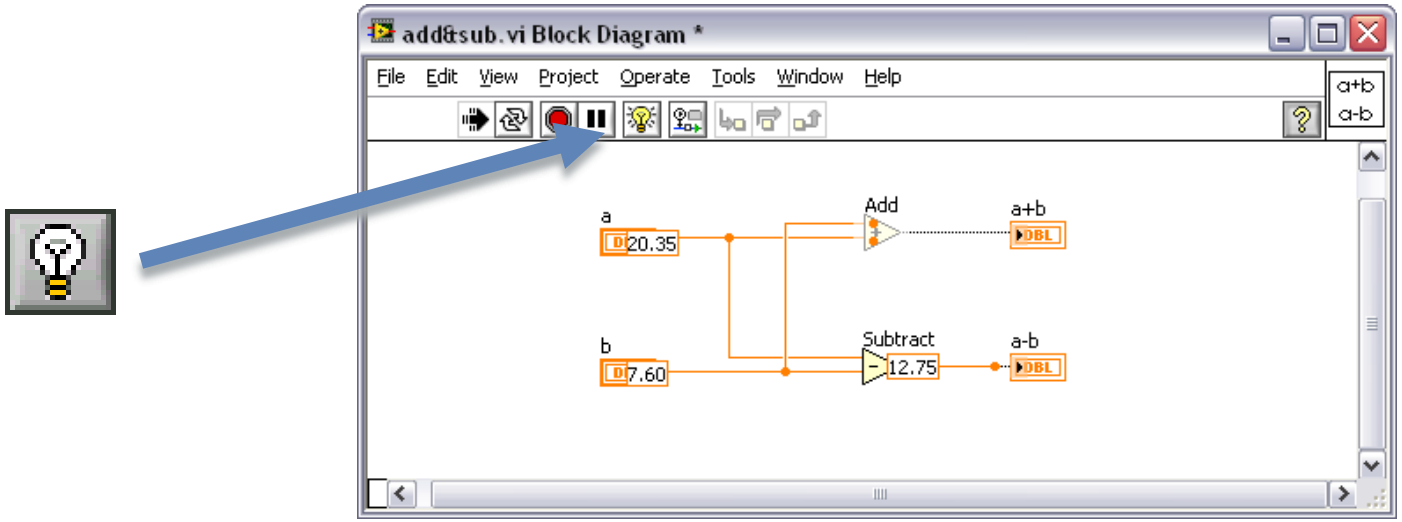
Debugging Techniques

What to look for if a VI produces unexpected data or behavior:

- Are there any unwired or hidden subVIs?
- Is the default data correct?
- Does the VI pass undefined data?
- Are numeric representations correct?
- Are node executed in the correct order?

Execution Highlighting

- Use execution highlighting to watch the data flow through the block diagram.
- If the VI runs more slowly than expected, confirm that you turned off execution highlighting in subVIs.



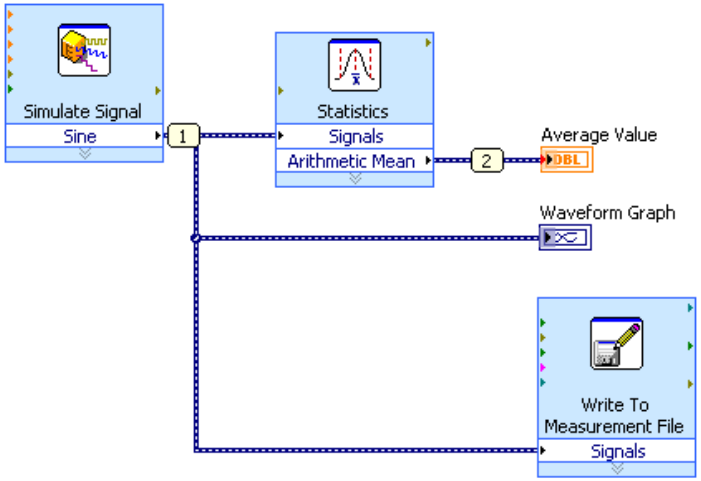
Single-Stepping

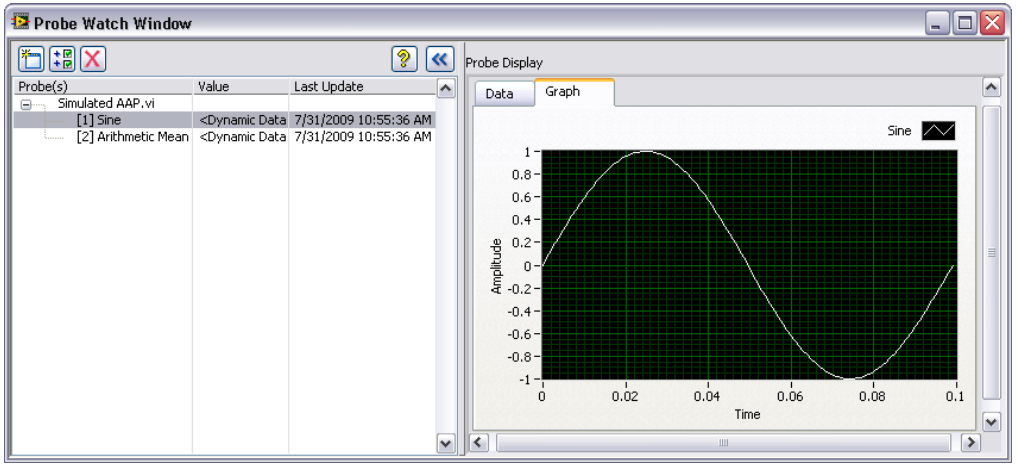
- Single-step through the VI to view each action of the VI on the block diagram.
- Suspend the execution of a subVI to edit values of controls and indicators, to control the number of times it runs, or to go back to the beginning of the execution of the subVI.
  - Open subVI and select **Operate»Suspend When Called** from the shortcut menu.

Probes

- Use the Probe tool to observe intermediate data values and check the error output of VIs and functions, especially those performing I/O.

- Specify to retain the values in the wires so that you can probe wires for data after execution.





Breakpoints

- When you reach a breakpoint during execution, the VI pauses and the **Pause** button appears red.

- You can take the following actions at a breakpoint:
  - Single-step through execution using the single-stepping buttons.
  - Probe wires to check intermediate values.
  - Change values of front panel controls.
  - Click the **Pause** button to continue running to the next breakpoint or until the VI finishes running.

I. LabVIEW Data Types

Shortcut Menu and Properties Dialog Box

Numeric Types

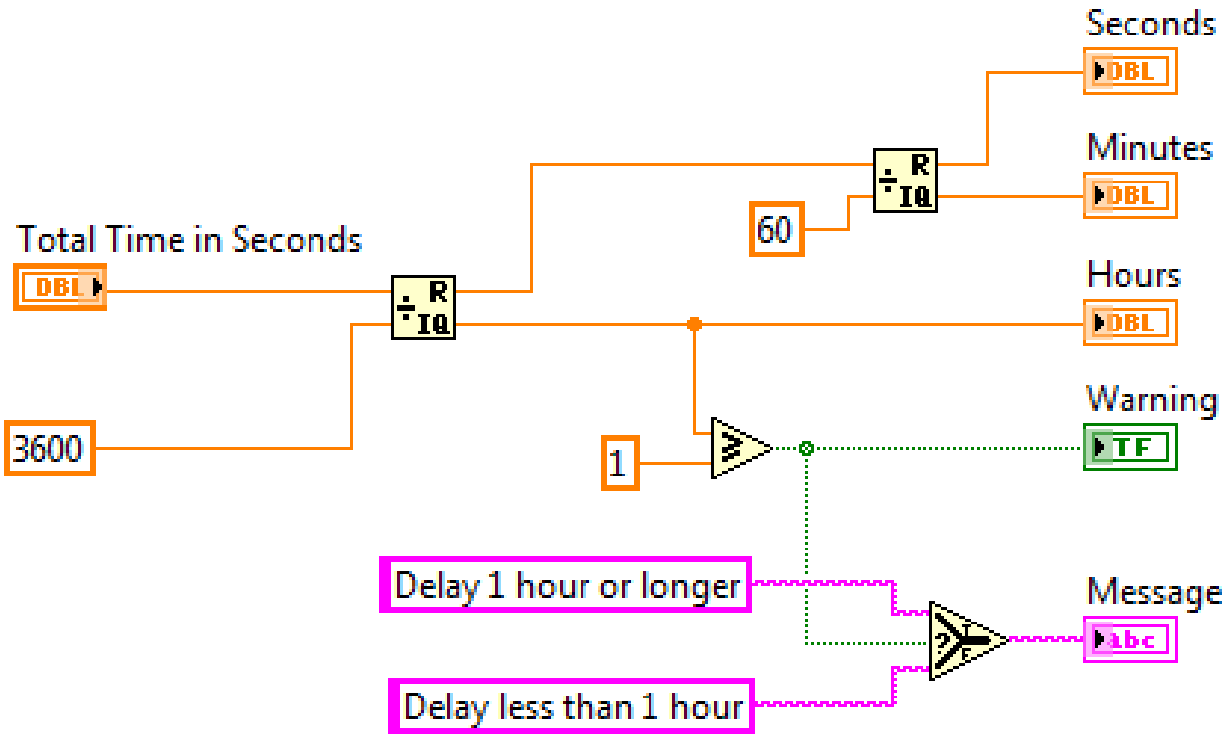
Boolean Types

String Types

Enums and Other Types

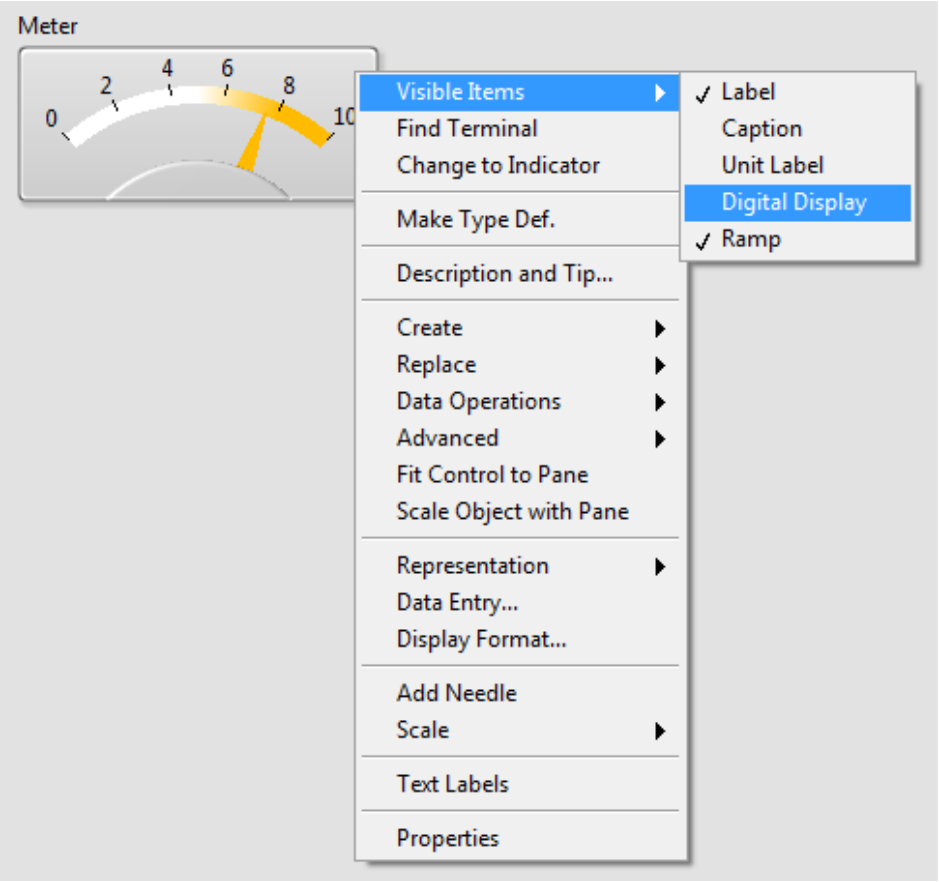
LabVIEW Data Types

Terminals visually communicate information about the data type represented



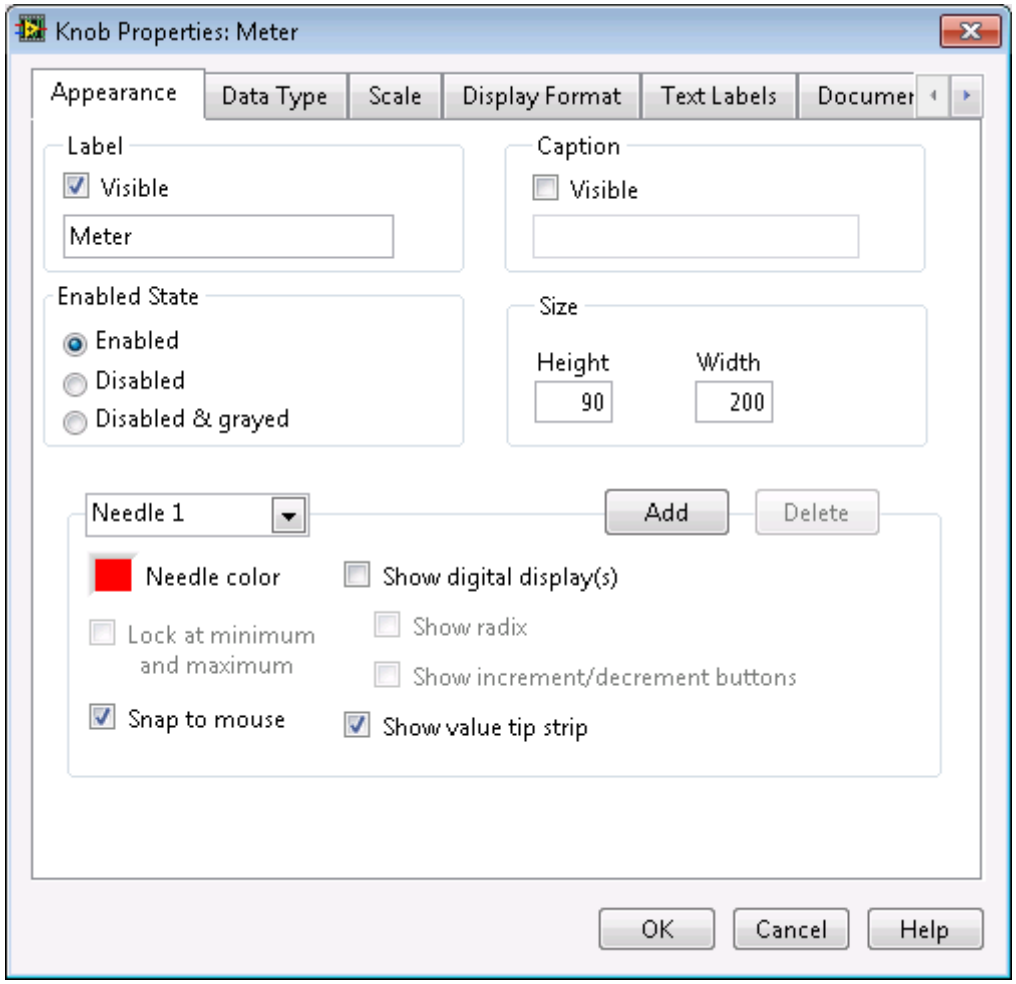
Shortcut Menus

- All LabVIEW objects have associated shortcut menus.
- Use shortcut menu items to change the look or behavior of objects.
- To access the shortcut menu, right-click the object.



Properties Dialog Box

- All LabVIEW objects have properties.
- To access properties, right-click the object and select **Properties**.
- Property options are similar to shortcut menu options.
- Select multiple objects to simultaneously configure shared properties.

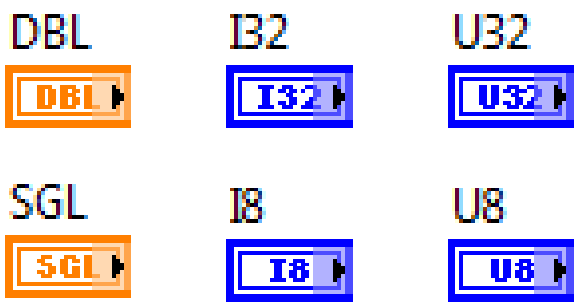


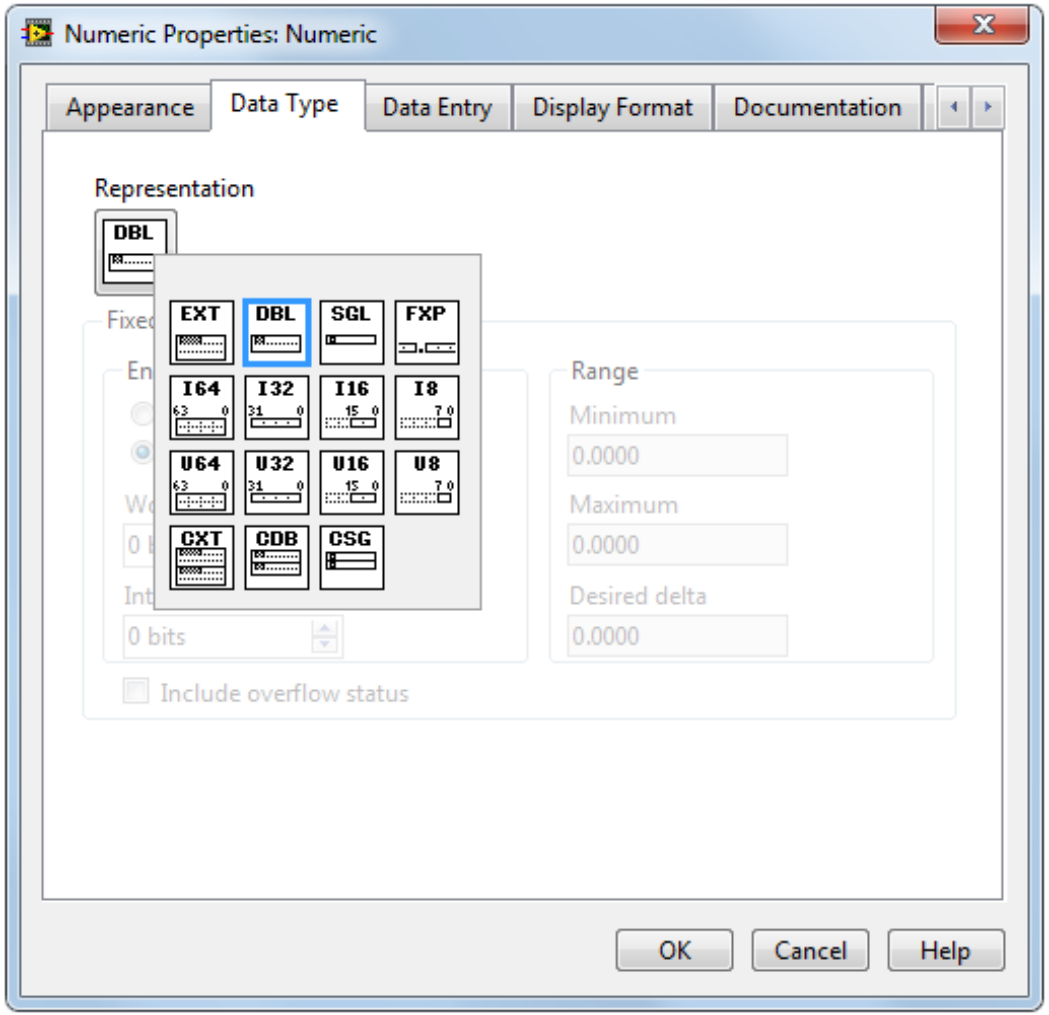
57

Numerics

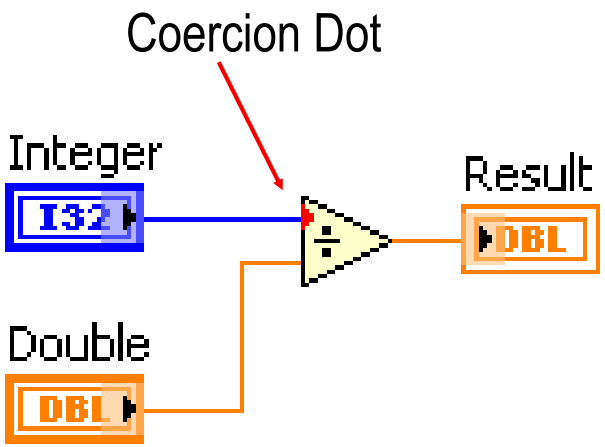
Various data type representations:

- Floating-point
- Unsigned integers
- Signed integers

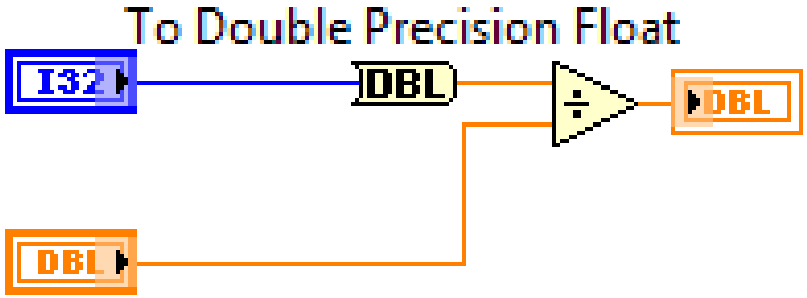




Numeric Conversion

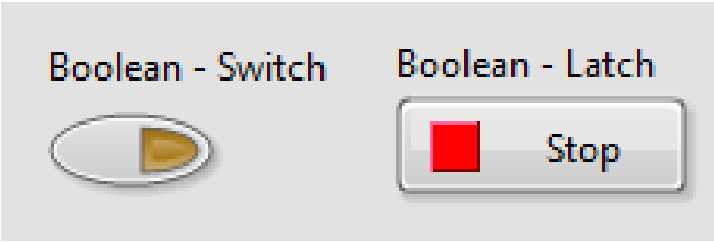


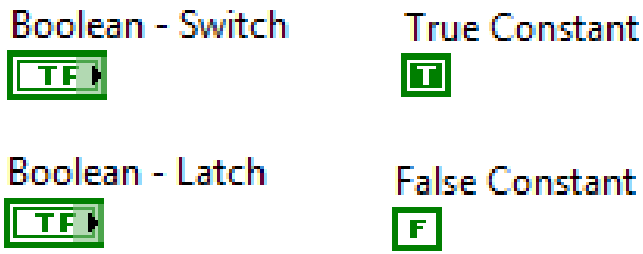
- Coercion dots indicate that LabVIEW converted the value passed into a node to a different representation.
  - Occurs when a node expects an input with a different representation.
- LabVIEW chooses the representation that uses more bits.
- Avoid coercion by programmatically converting to a matching data type.

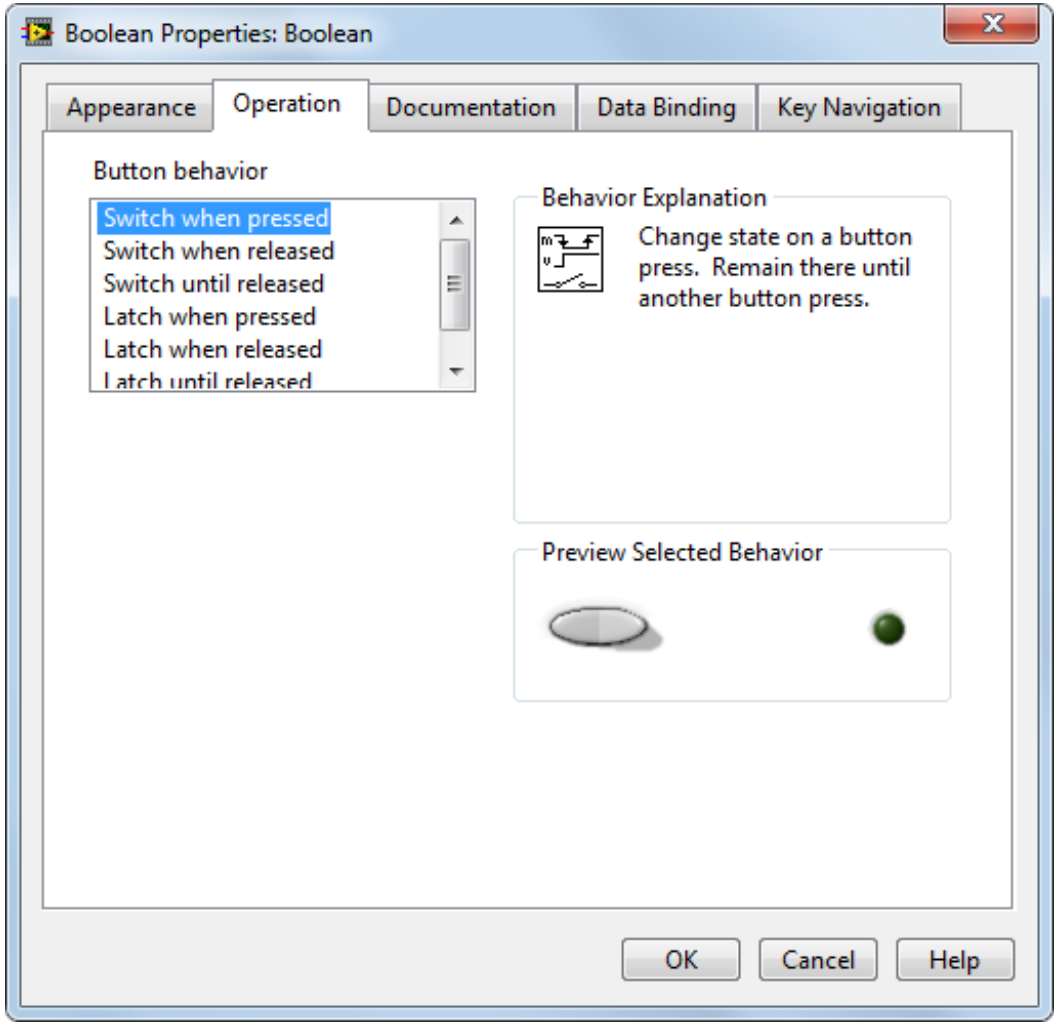


Booleans

- Behavior of Boolean controls is specified by the mechanical action.
- Boolean have only TRUE/FALSE values.

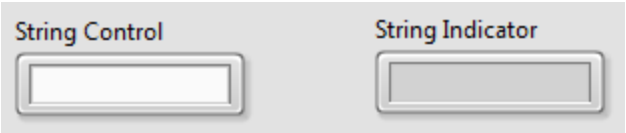


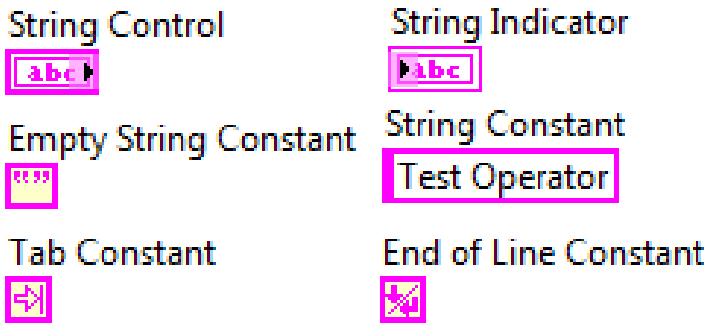


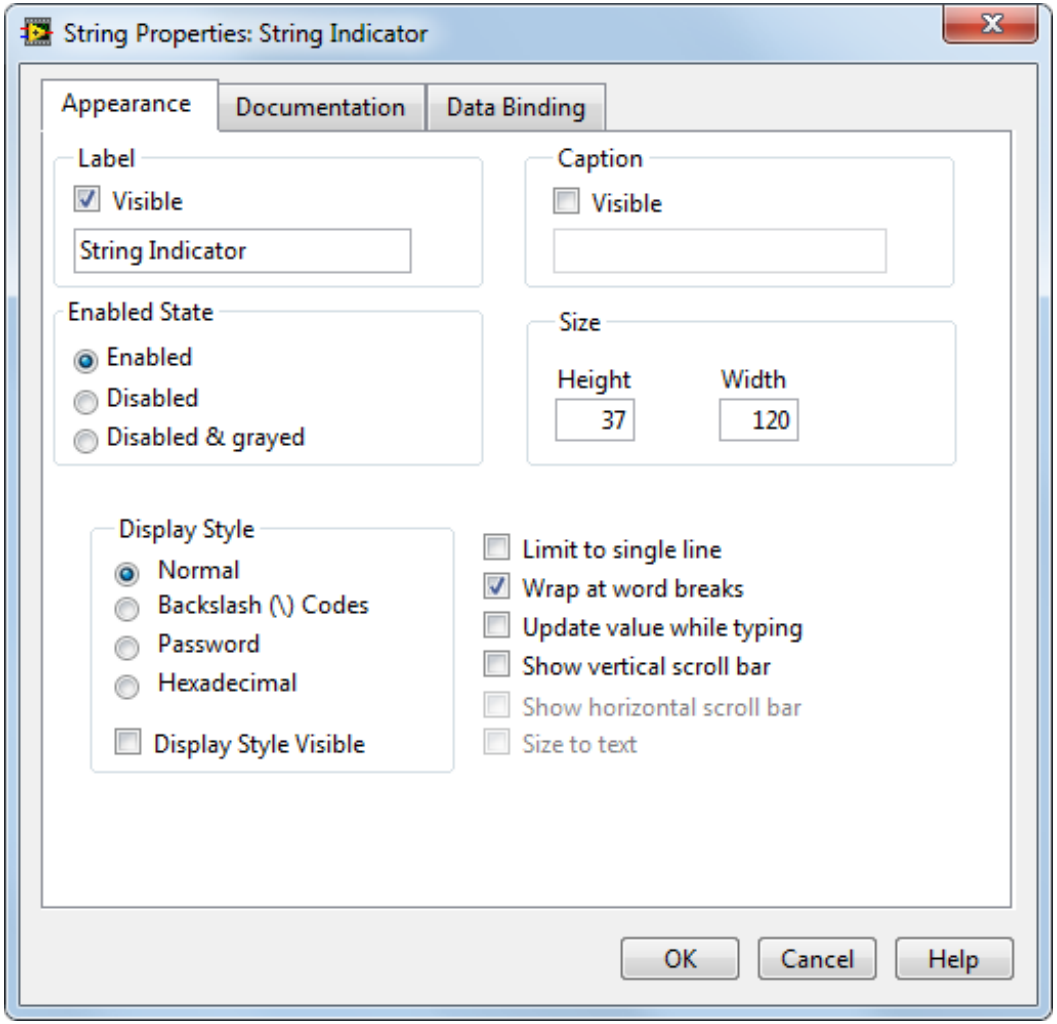


Strings

- A string is a sequence of ASCII characters.
- Strings have various display styles.
  - Backslash codes
  - Password
  - Hex

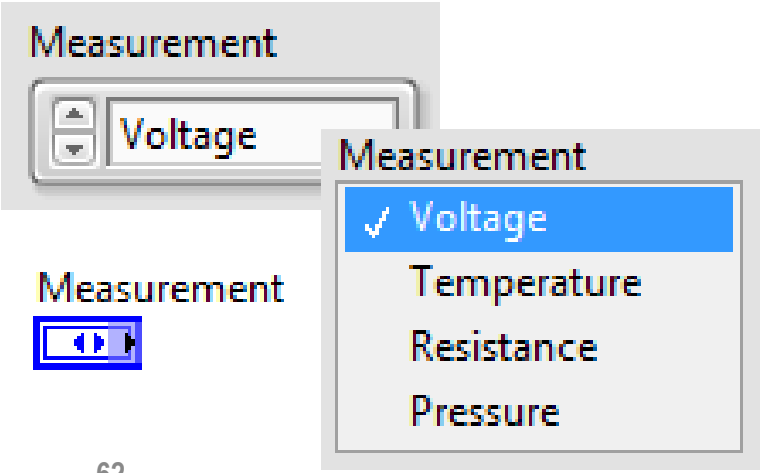


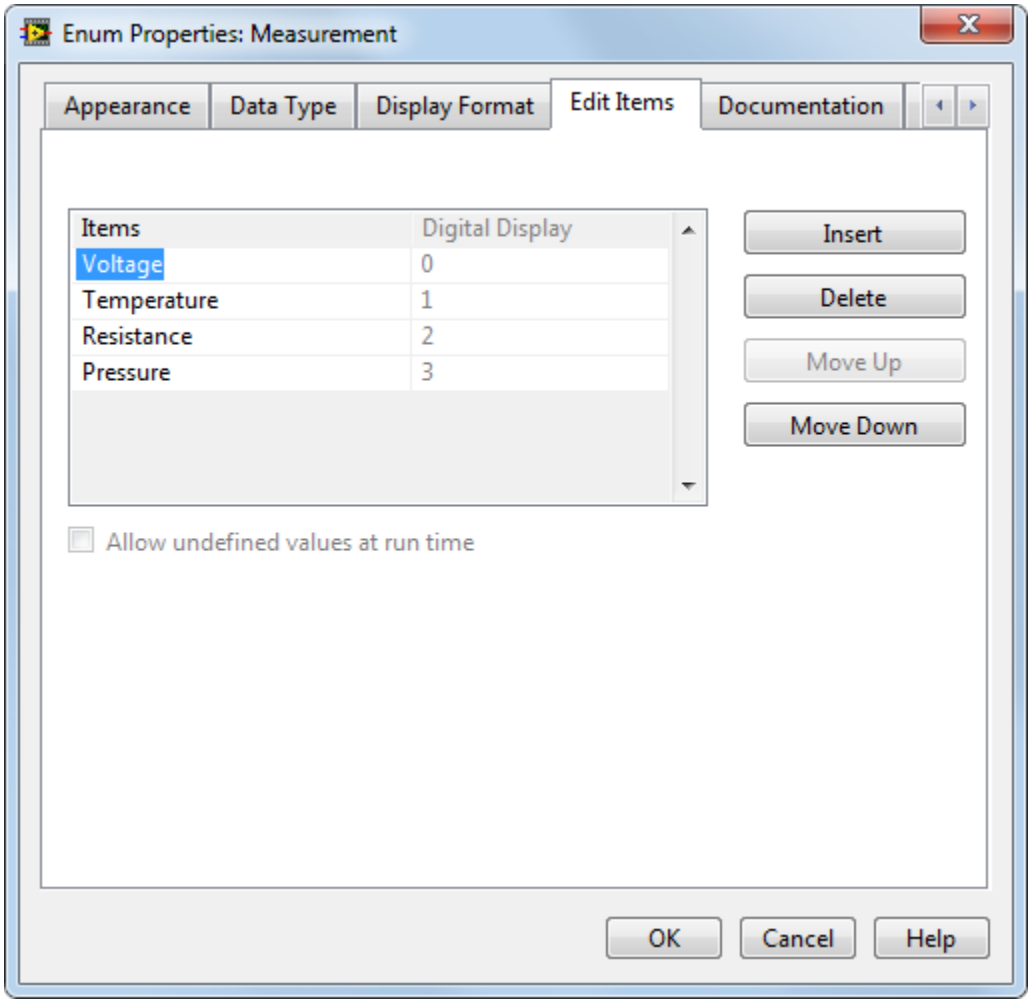




Enums

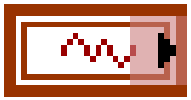
- Enums give users a list of items from which to select.
- Each item represents a pair of values.
  - String
  - 16-bit Integer





Other Data Types

Refer to *LabVIEW Help* for complete list of terminal symbols for different types of controls and indicators.

- Dynamic 
  - Stores the information generated or acquired by an Express VI.
- Path 
  - Stores the location of a file or directory using the standard syntax for the platform you are using.
- Waveform 
  - Carries the data, start time, and dt of a waveform.

J. Program Flow

Data Flow

Case Structure

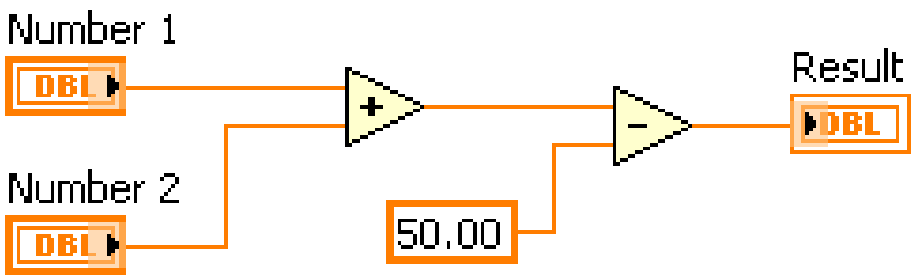
Loops

Timing

Dataflow

LabVIEW follows a dataflow model for running VIs.

- A node executes only when data are available at all of its required input terminals.
- A node supplies data to the output terminals only when the node finishes execution.



Dataflow – Quiz

Which node executes first?

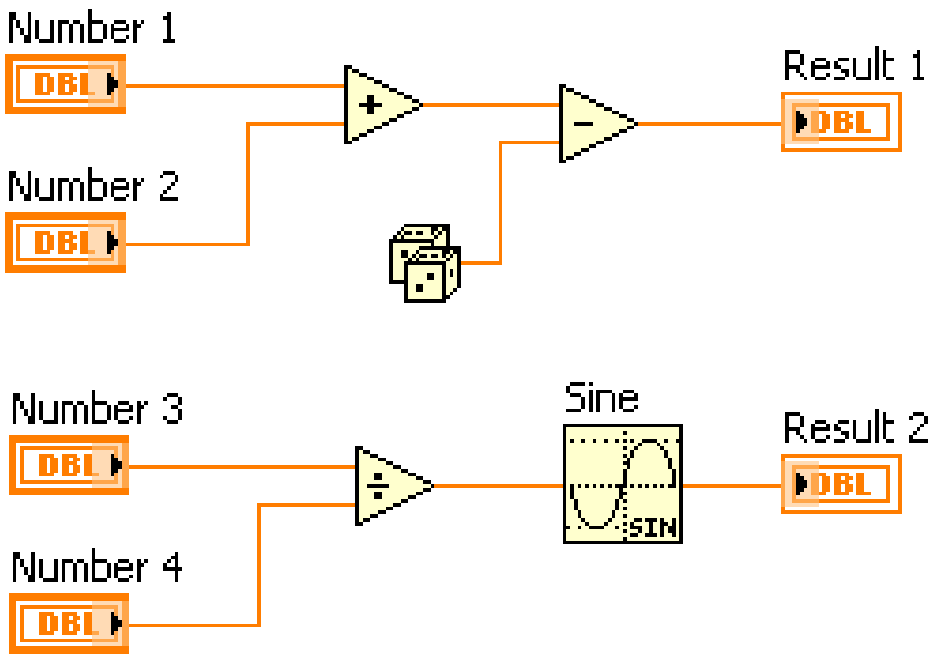
a) Add

b) Subtract

c) Random Number

d) Divide

e) Sine



Dataflow – Quiz Answers

No single correct answer.

Which node executes first?

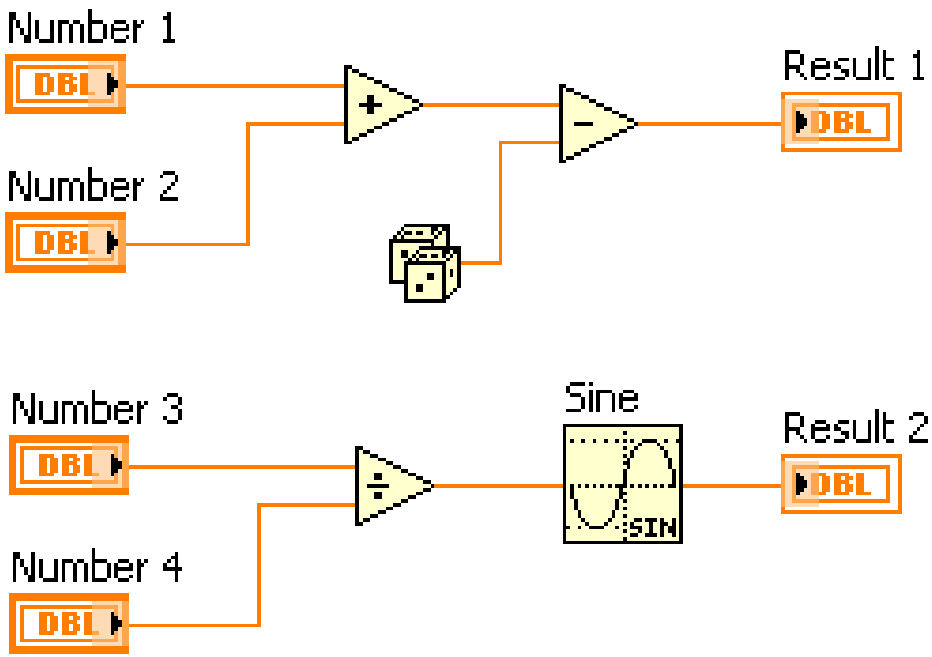
a) Add – **Possibly**

b) Subtract – **Definitely not**

c) Random Number – **Possibly**

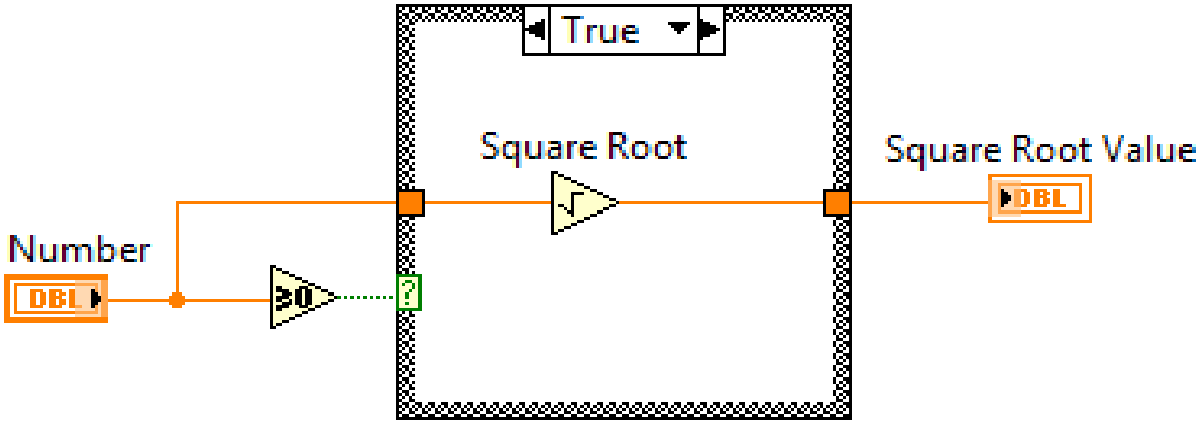
d) Divide – **Possibly**

e) Sine – **Definitely not**



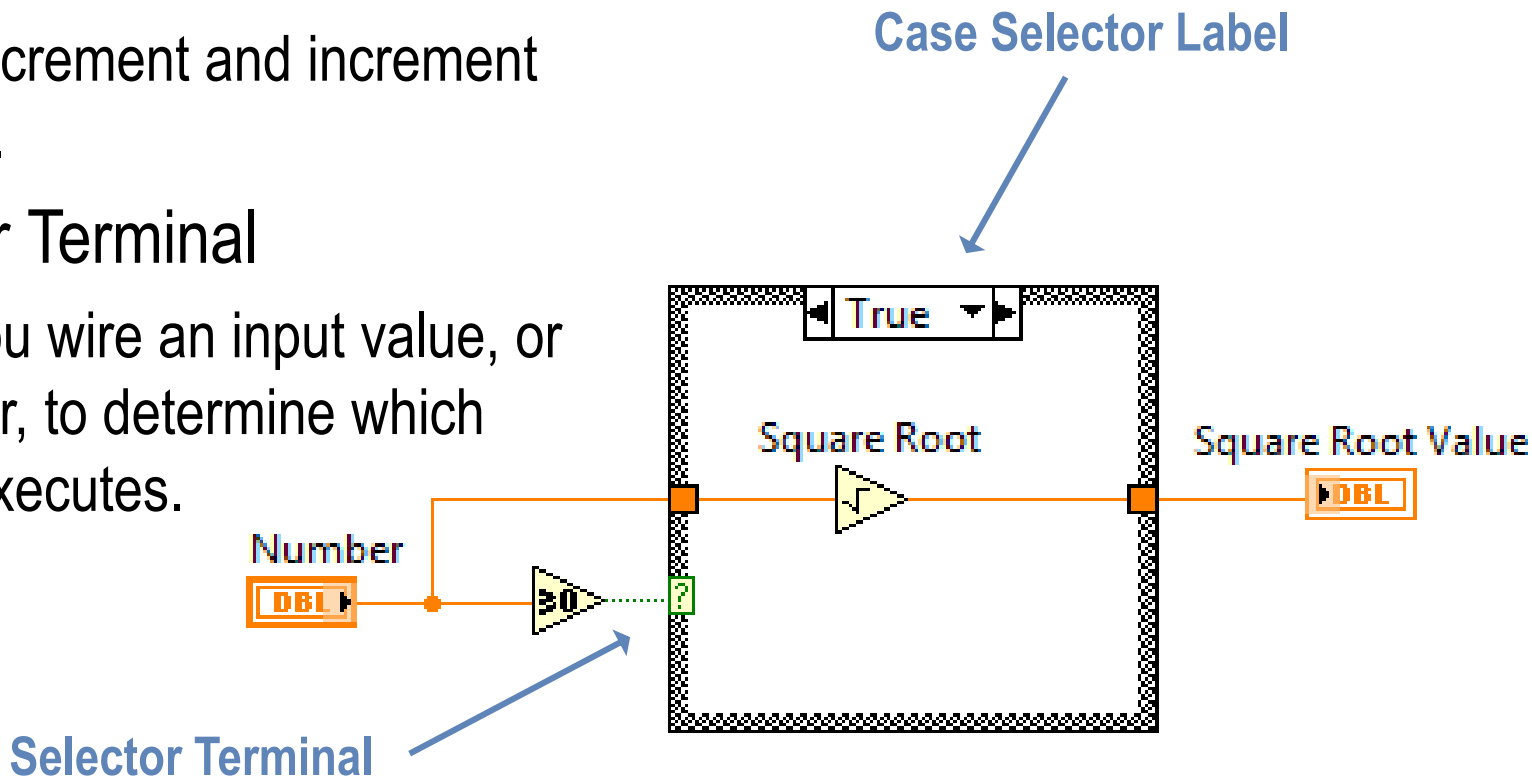
Case Structures

- Have two or more subdiagrams or cases.
- Use an input value to determine which case to execute.
- Execute and display only one case at a time.
- Are similar to **case** statements or **if...then...else** statements in text-based programming languages.



Case Structures

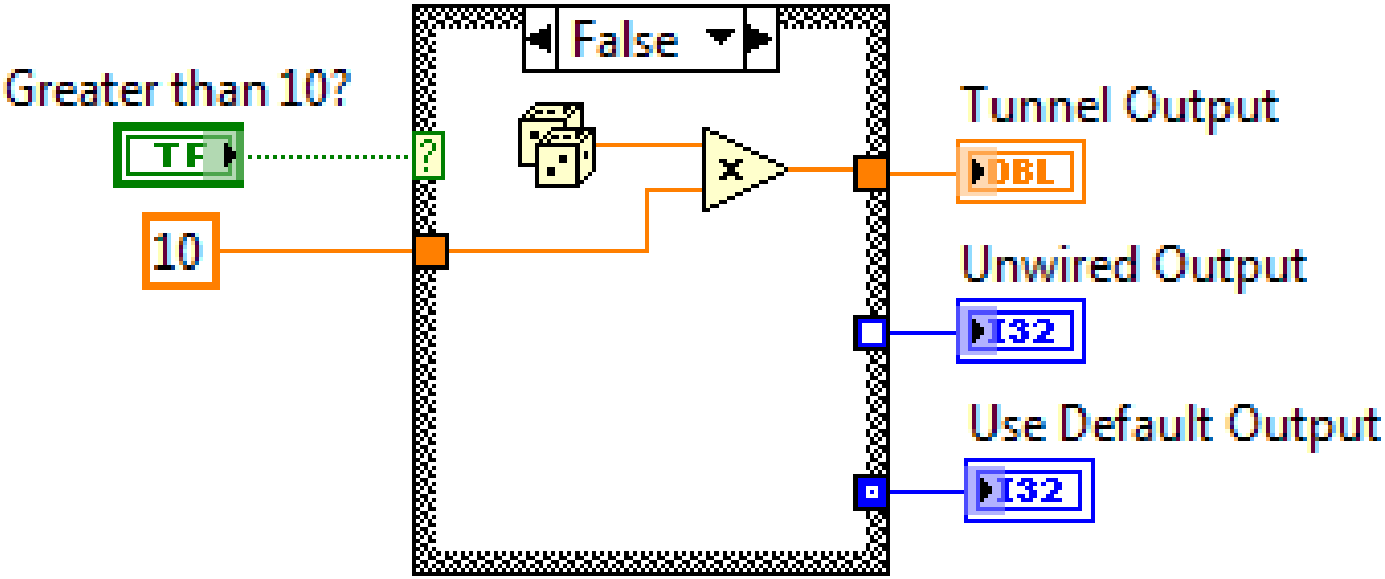
- Case Selector Label
  - Contains the name of the current case.
  - Has decrement and increment arrows.
- Selector Terminal
  - Lets you wire an input value, or selector, to determine which case executes.



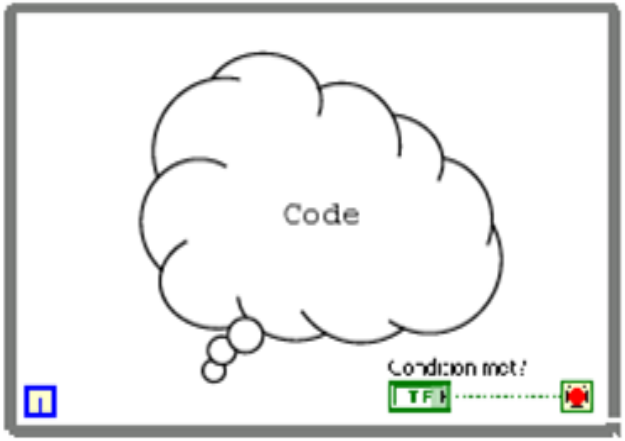
Input and Output Tunnels

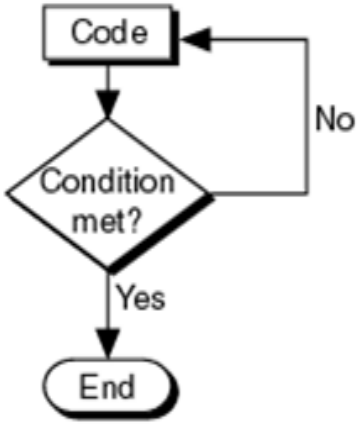
You can create multiple input and output tunnels.

- Inputs tunnels are available to all cases if needed.
- You must define each output tunnel for each case.



While Loops





Repeat (code);

Until Condition met;

End;

LabVIEW While Loop

Flowchart

Pseudo Code

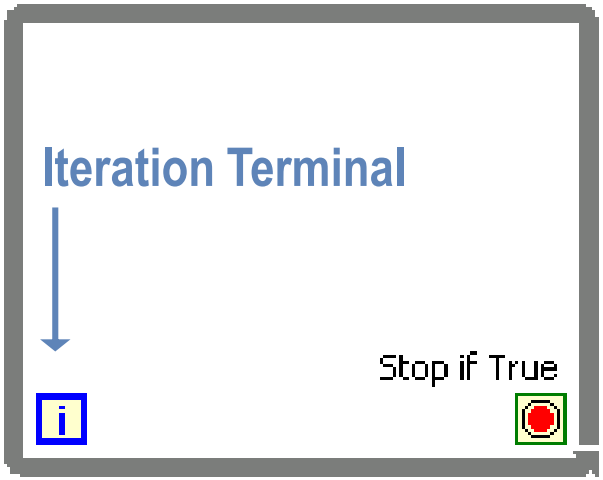
While Loops

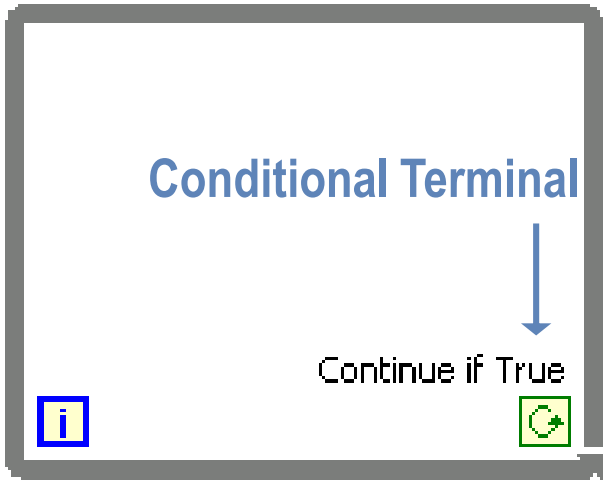
Iteration terminal

- Returns number of times loop has executed.
- Is zero-indexed.

Conditional terminal

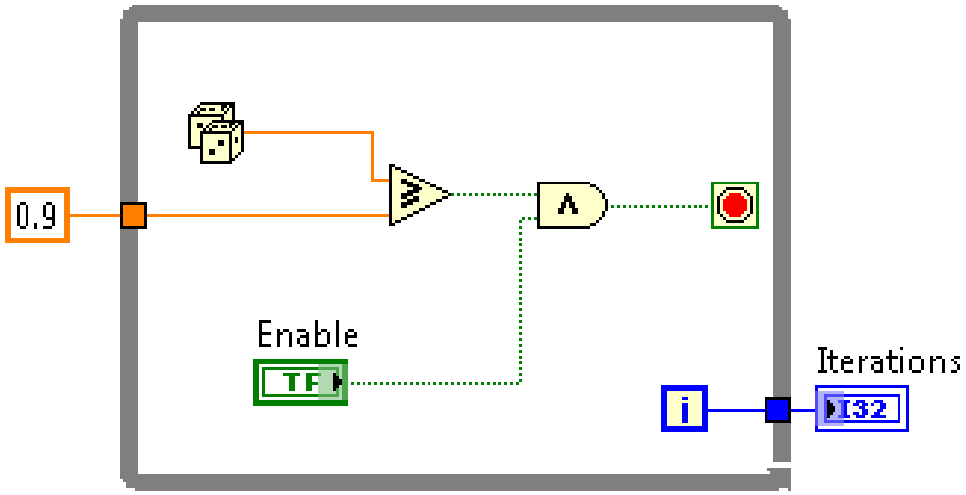
- Defines when the loop stops.
- Has two options.
  - Stop if True
  - Continue if True



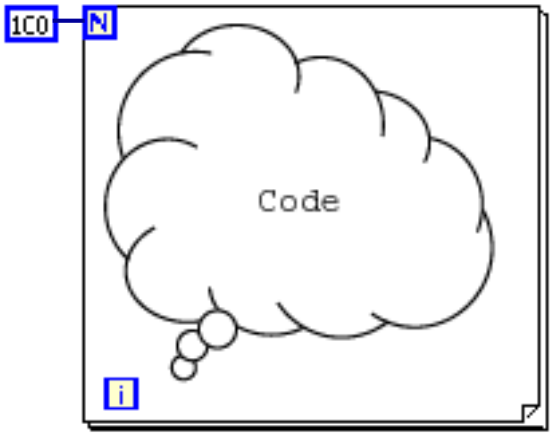


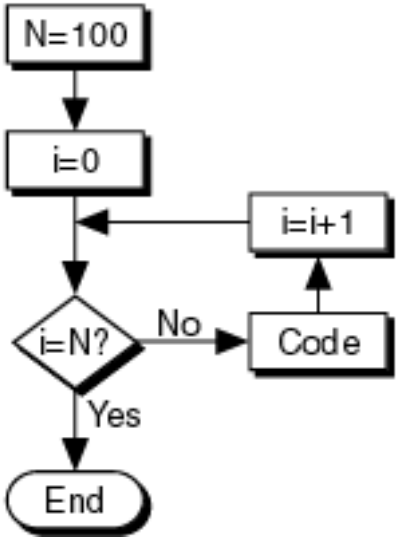
While Loops – Tunnels

- Tunnels transfer data into and out of structures.
- Data pass out of a loop after the loop terminates.
- When a tunnel passes data into a loop, the loop executes only after data arrive at the tunnel.



For Loops






```
N=100;
i=0;
Until i=N:
    Repeat (code;i=i+1);
End;
```

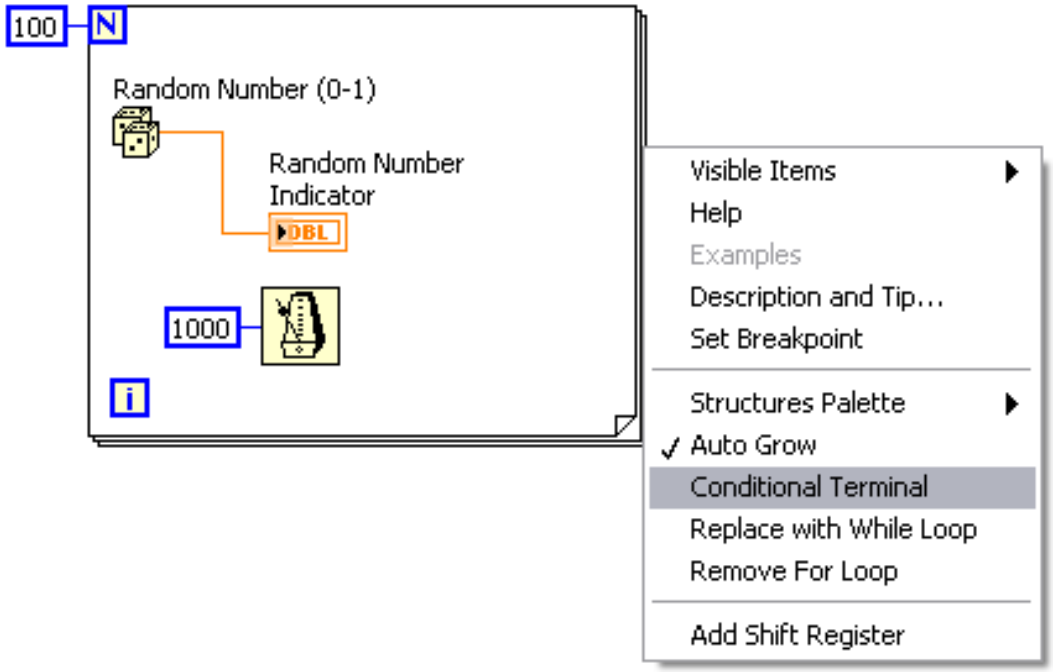
LabVIEW For Loop

Flowchart

Pseudo Code

For Loops – Conditional Terminal

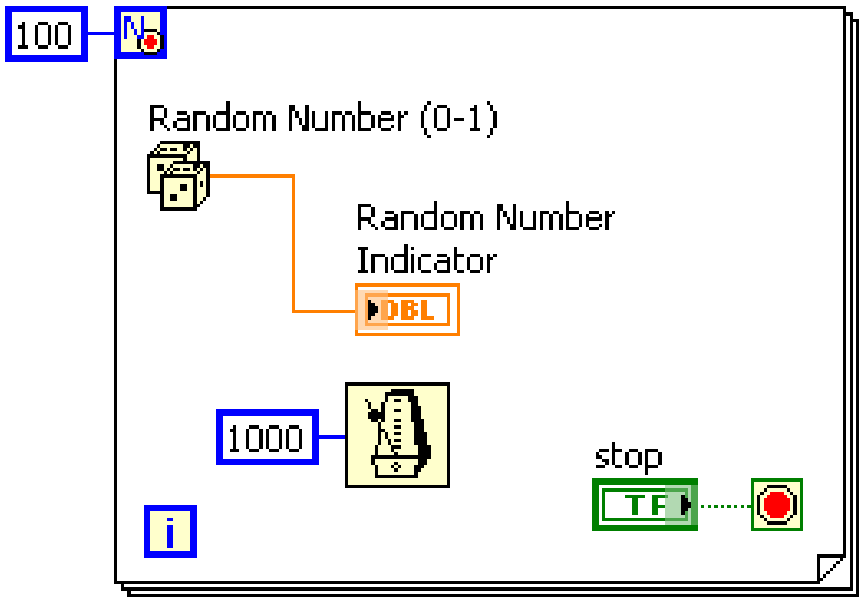
You can add a conditional terminal to configure a For Loop to stop when a Boolean condition is true or an error occurs.



For Loops – Conditional Terminal

For Loops configured with a conditional terminal have:

- A red glyph next to the count terminal.
- A conditional terminal in the lower right corner



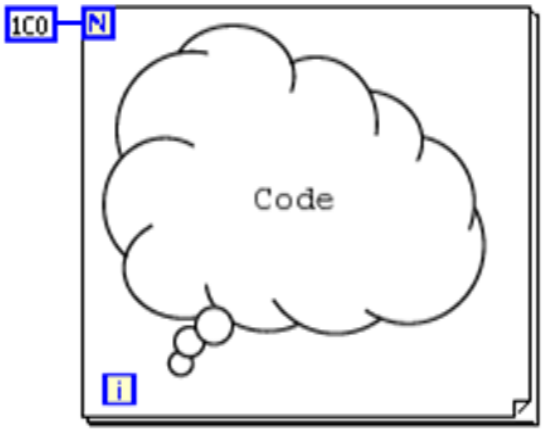
For Loop/While Loop Comparison

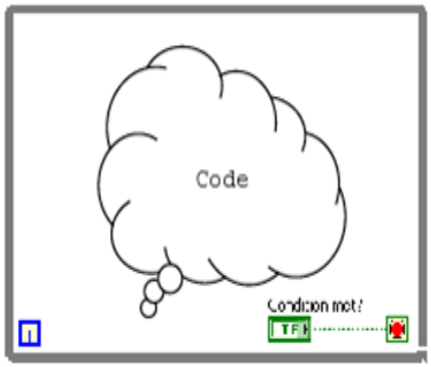
For Loop

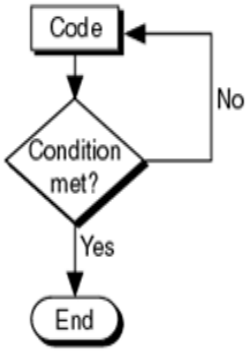
- Executes a set number of times unless a conditional terminal is added.
- Can execute zero times.
- Tunnels automatically output an array of data.

While Loop

- Stops executing only if the value at the conditional terminal meets the condition.
- Must execute at least once.
- Tunnels automatically output the last value.

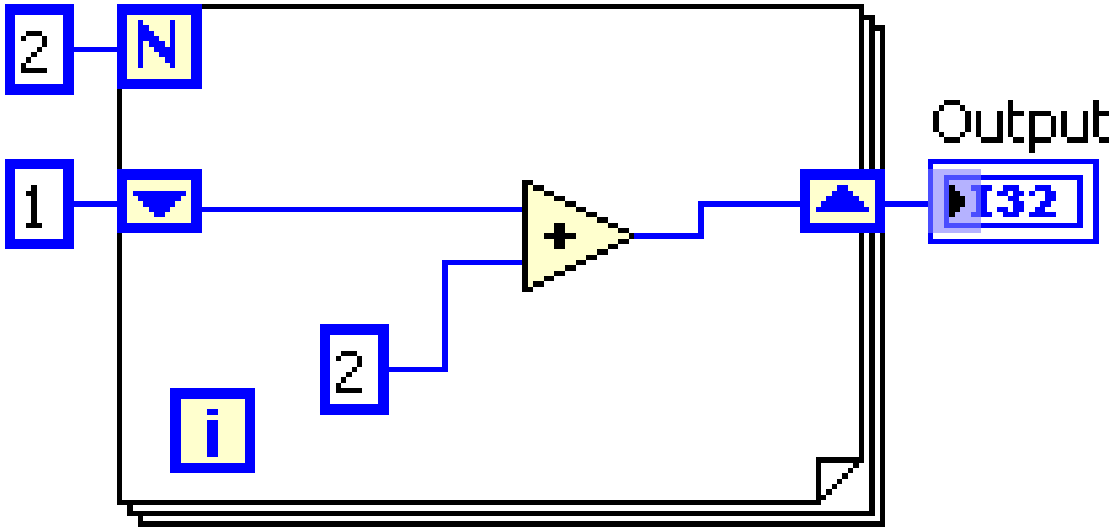






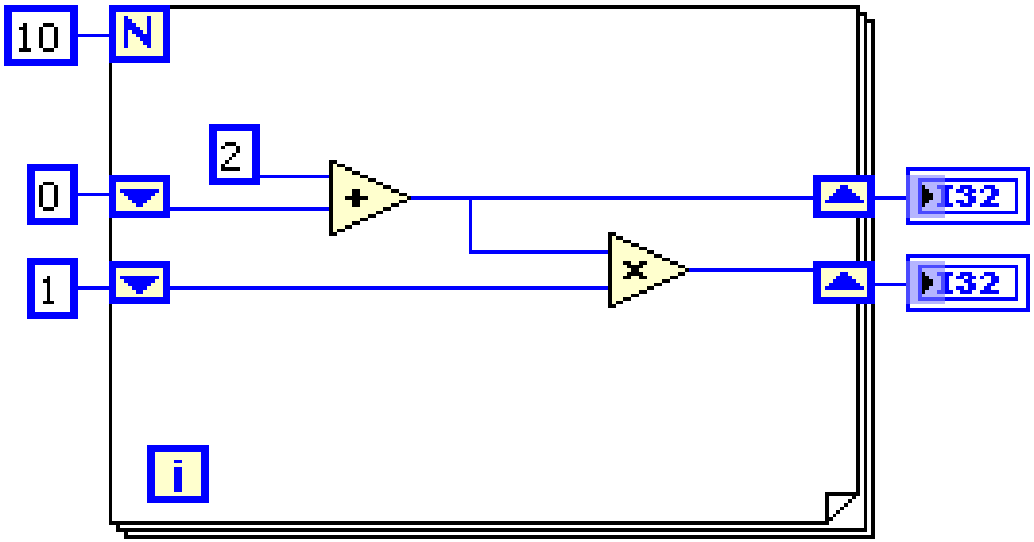
Data Feedback in Loops

- When programming with loops, you often need to know the values of data from previous iterations of the loop.
- Shift registers transfer values from one loop iteration to the next.



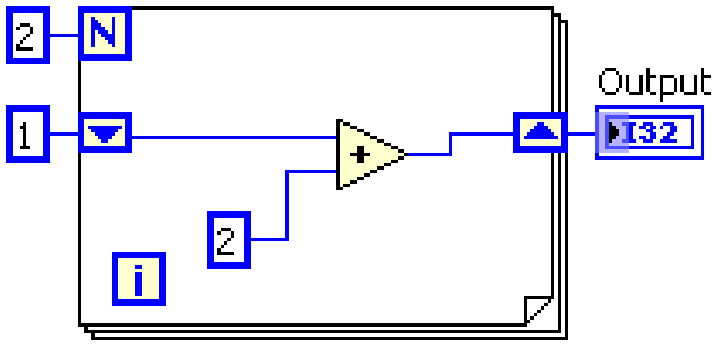
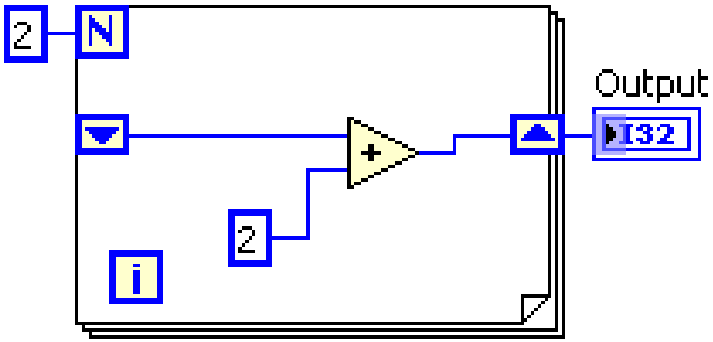
Shift Registers

- Right-click the border and select **Add Shift Register** from the shortcut menu.
- Right shift register stores data on completion of an iteration.
- Left shift register provides stored data at beginning of the next iteration.



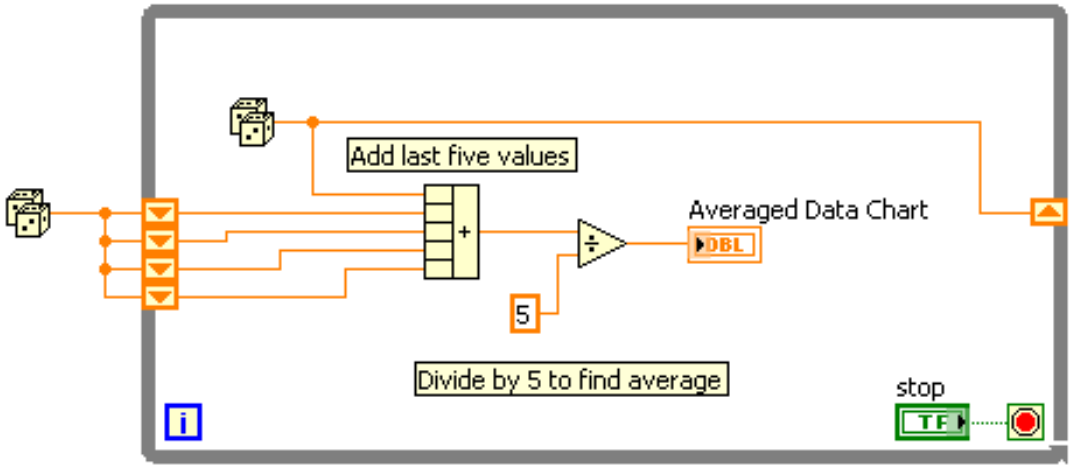
Initializing Shift Registers

Run once → VI finishes → Run again

Block Diagram	1st run	2nd run
Initialized Shift Register 	Output = 5	Output = 5
Not Initialized Shift Register 	Output = 4	Output = 8

Multiple Previous Iterations

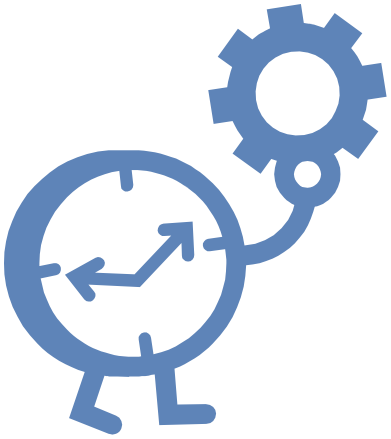
- Stacked shift registers remember values from multiple previous iterations and carry those values to the next iterations.
- Right-click the left shift register and select **Add Element** from the shortcut menu to stack a shift register.



Timing a VI

Why do you need timing in a VI?

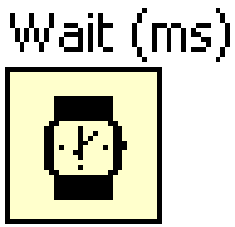
- To control the frequency at which a loop executes.
- To provide the processor with time to complete other tasks, such as processing the user interface.

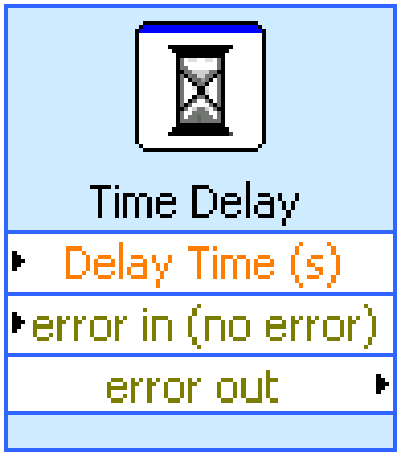


Wait Functions

A wait function inside a loop:

- Allows the VI to sleep for a set amount of time.
- Allows the processor to address other tasks during the wait time.
- Uses the operating system millisecond clock.





Elapsed Time Express VI

- Determines how much time elapses after some point in your VI.
- Keeps track of time while the VI continues to execute.
- Does not provide the processor with time to complete other tasks.

Homework

- Complete the LabVIEW Beginner Homework

- Open-ended Problem:
  - Install the Arduino Package for LabVIEW
  - Establish communication between Arduino and LabVIEW
  - Use any sensor that you learned earlier and plot the measurement on the screen in real time with LabVIEW

Questions?

- Online Resources:
  - Self-Paced Online Training
  - http://www.learnni.com/
  - LabVIEW 101 Slides