Tutorial: Passing Data and Debugging

Publish Date: Jul 02, 2008

Overview

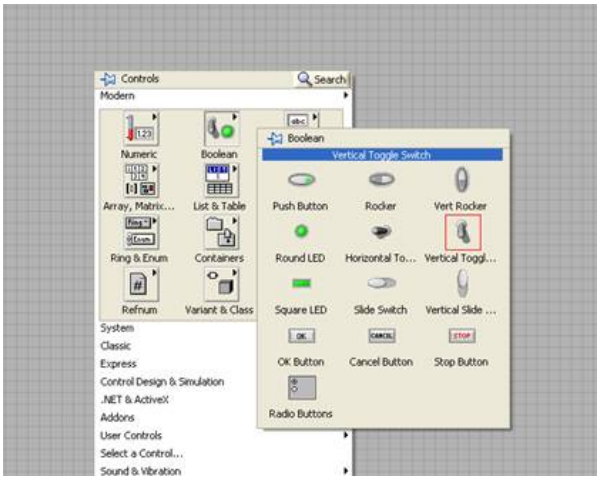
Just as in any other programming languages, it is important to have a basic understanding of the different data types in NI LabVIEW software and how to access them when composing a VI. When you are unsure of a wire's data type or of the inputs and outputs a certain VI accepts, the Context Help window is a very powerful tool for finding this information. Once you wire a VI, it is important to pay attention to the data flow of the VI to determine the program's sequence of events. When you are unsure of the exact sequence of events in a VI, you can use the Highlight Debugging feature to slow down the code and visually observe the data flow of the VI.

Table of Contents

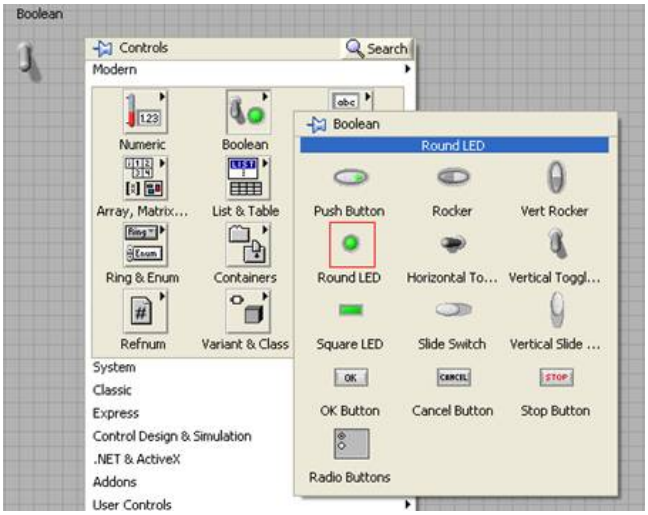
LabVIEW Data Types

1. Open a blank VI from the toolbar by selecting **File»New VI**.

2. Right-click on the front panel to open the **Controls** palette and select **Modern»Boolean»Vertical Toggle Switch**, and place the switch on the front panel. This control is of the Boolean data type, which means it can take on one of two values, TRUE or FALSE.



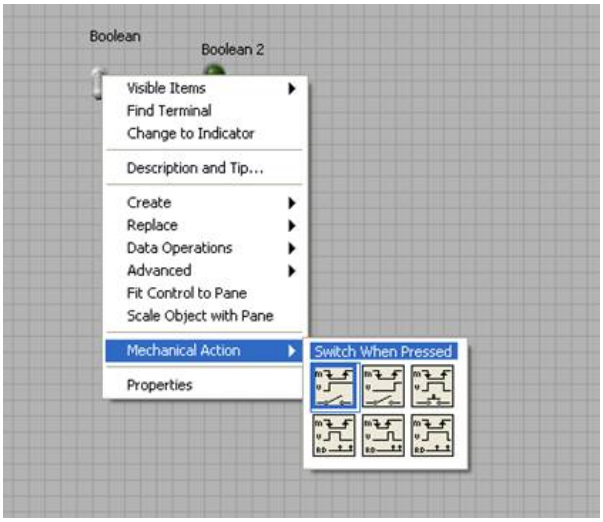
3. Open the **Controls** palette and select **Modern»Boolean»Round LED**, and place the LED on the front panel. This indicator is also a Boolean data type.



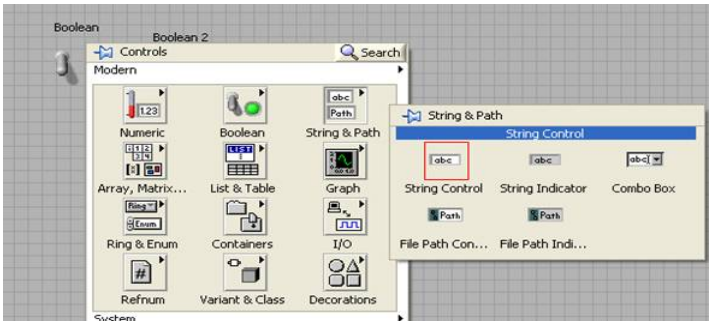
4. Open the block diagram of the VI by selecting **Window»Show Block Diagram**, and notice that the Boolean control and indicator are both represented by green icons. This is the color for all Boolean data types in LabVIEW.

5. Wire the output of the toggle switch control to the input of the LED indicator. Note that the wire connecting the two icons is also green.

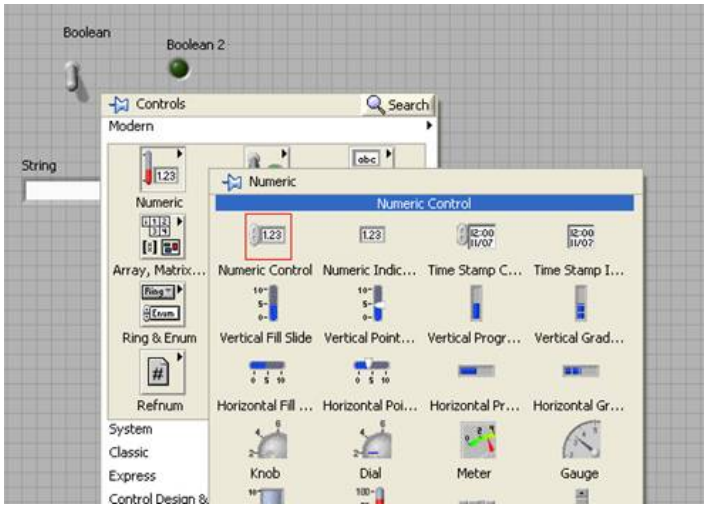
6. Return to the front panel by selecting **Window»Show Front Panel**. Right-click on the toggle switch control and hover your mouse over **Mechanical Action** to view the available switching actions for switch when pressed. Leave the action as **Switch When Pressed**.

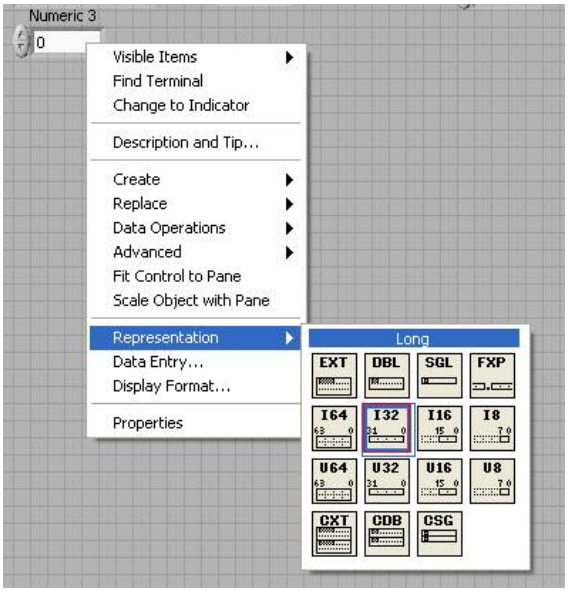


7. Open the **Controls** palette, select **Modern»String & Path»String Control**, and place a string control on the front panel. The string data type is composed of a sequence of displayable or non-displayable ASCII characters.

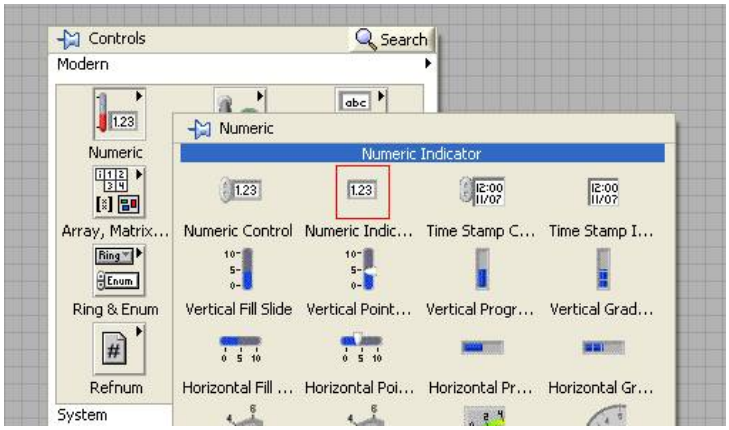


8. Open the **Controls** palette, select **Modern»Numeric»Numeric Control**, and place a numeric control on the front panel. The numeric data type is composed of numbers, and can take on several different representations. Right-click on the numeric control and select **Representation»Long (I32)** to change the control to a 32-bit integer.



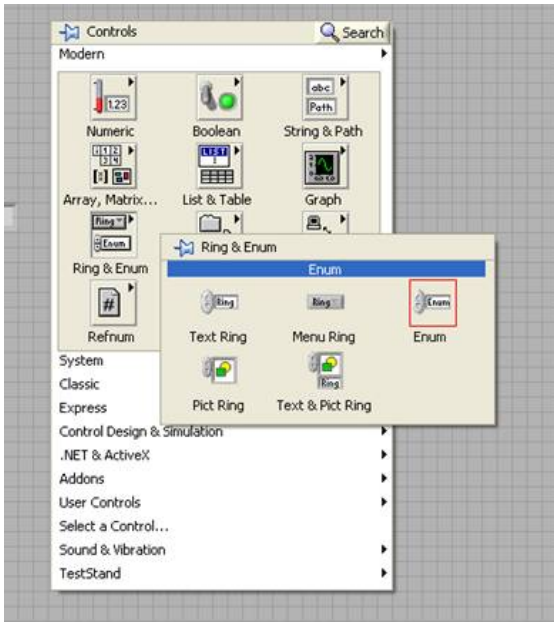


9. From the **Controls** palette, select **Modern»Numeric»Numeric Indicator** and place the numeric indicator on the front panel. Leave this indicator's representation as the default double precision (DBL).

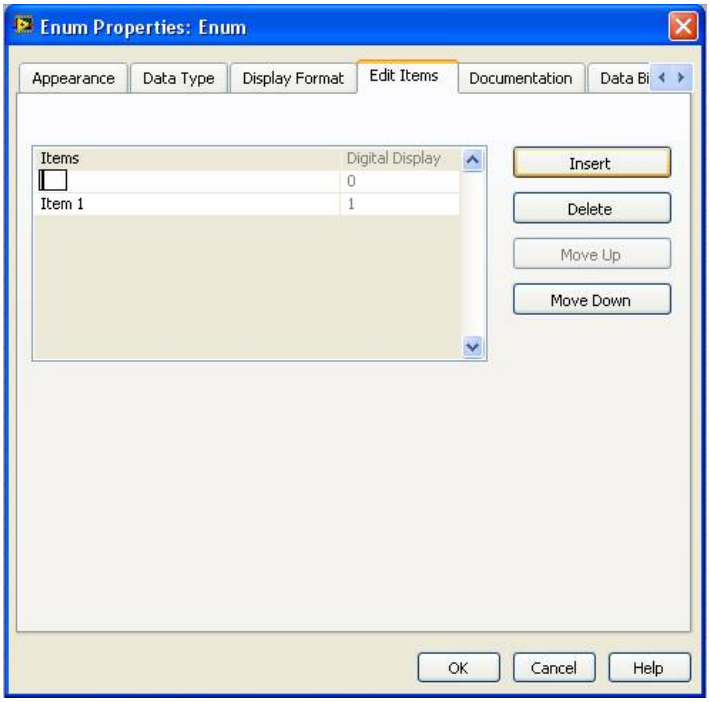


10. Open the block diagram and observe the colors associated with each new control and indicator. The string data type is pink, the long numeric data type is blue, and the double precision numeric data type is orange.

11. Open the front panel and the **Controls** palette, select **Modern»Ring & Enum»Enum**, and place the Enum control on the front panel. The Enum control is an enumerated list of string values, where each string value has a numeric value associated with it.



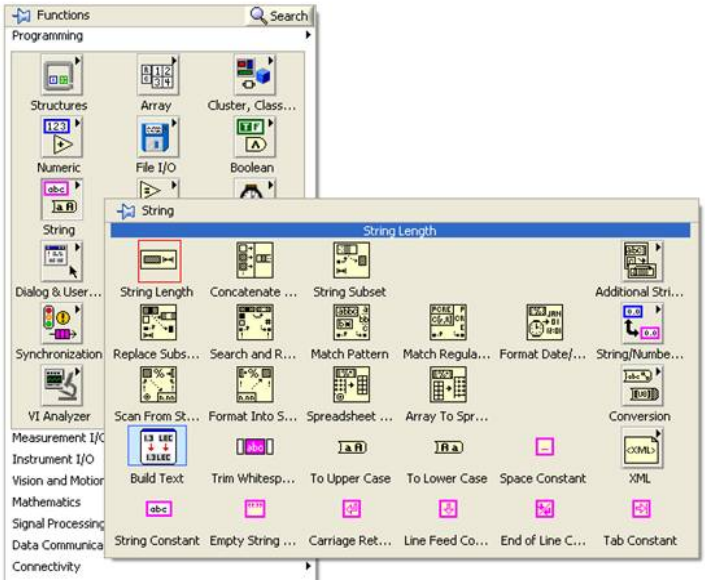
12. Right-click on the Enum control and select **Edit Items…** to open up the Edit Items tab under Enum Properties. On the Edit Items tab, click on **Insert** and type in a string with the keyboard. Pres the **<Enter>** key to create a new item below the current item, or click on **Insert** to insert a new item above the current item. Repeat the process until you have at least a few items in the Enum.



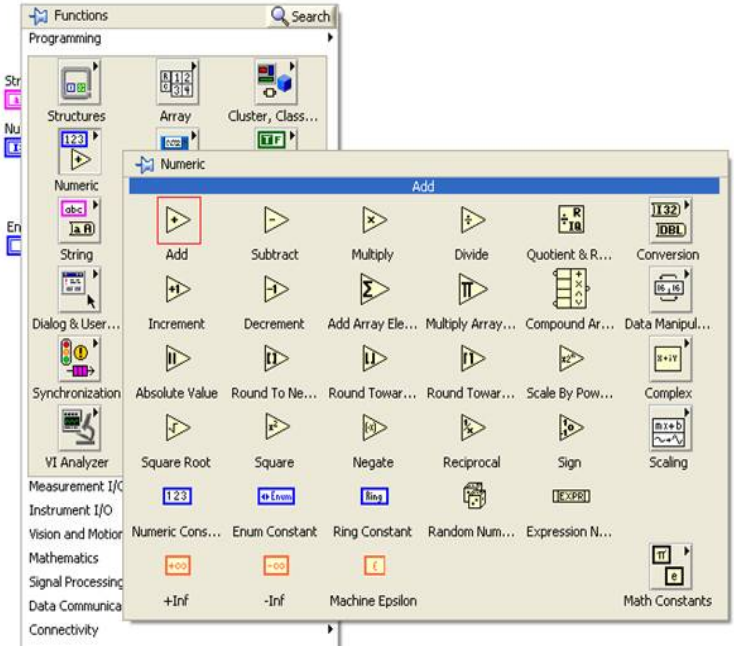
13. Click on the **OK** button when finished entering items into the Enum.

14. Open up the block diagram, right-click on the output pin of the Enum control, and select **Create»Indicator** to create an indicator that is wired to the Enum control. Note that the indicator is also an Enum.

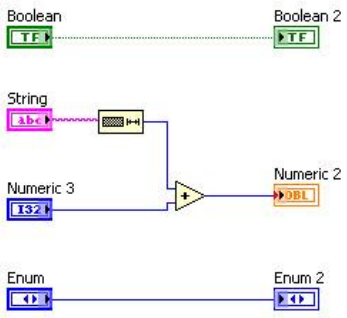
15. In the block diagram of the VI from above, right-click on a blank space to open the **Functions** palette, select **Programming»String»String Length**, and place the function on the block diagran



16. Open the **Functions** palette again, select **Programming»Numeric»Add,** and place an add function on the block diagram.



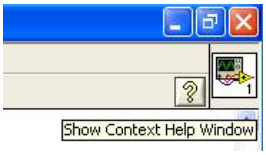
17. Wire the newly placed functions with the unconnected controls and indicators as seen in the figure below.



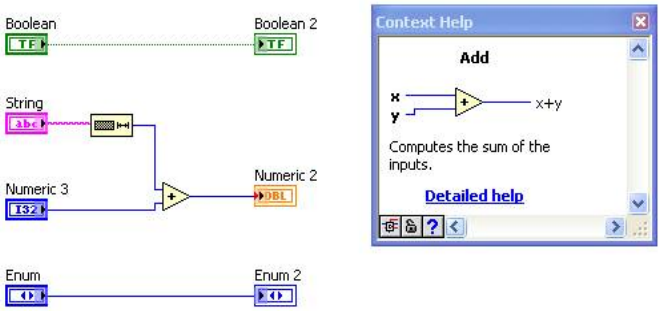
Using Context Help

The **Context Help** window displays basic information about LabVIEW objects when you hover your mouse over each object.

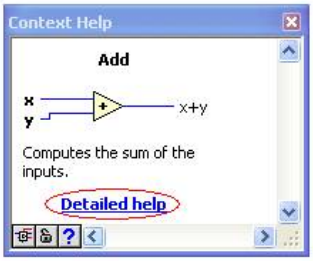
1. Open the **Context Help** window by selecting **Help»Show Context Help**, pressing the **<ctr-H>** keys on the keyboard, or by clicking on the **Show Context Help Window** button on the toolbar, as seen in the figure below.



2. Hover your mouse over the different wires, controls, and indicators to see their data types in the **Context Help** window. Hover your mouse over the functions to see a brief explanation of their functions and the inputs and outputs they accept.

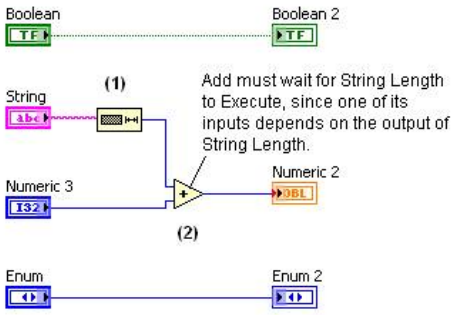


3. Click on the **Detailed Help** link at the bottom of the **Context Help** window to get a more detailed description of the function and its inputs and outputs.



Data Flow

As mentioned earlier, the concept of data flow is important to keep in mind when programming. Just as a line of text-based code must wait for data to reach it before executing, a node (function or V must wait until data reaches all of its input terminals before executing. Once that node finishes execution, it passes its output to the next node in the execution sequence. In the VI that you built, although one of the inputs to the add function is immediately available, the node must wait for the string length function to execute first because its output is the other input to the add function.



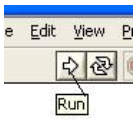
So data flow can sometimes determine the order of execution of LabVIEW code. In this case, it forces the string length function to execute before the add function, as seen in the figure above.

Debugging with Highlight Execution

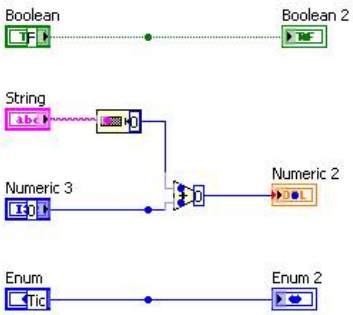
When trying to visualize the data flow of a particular VI, a very useful tool is highlight execution or execution highlighting. Implement execution highlighting by clicking on the **Highlight Execution** button on the toolbar, as seen in the figure below.

1. Click on the **Highlight Execution** button on the toolbar. Note that the light bulb on the button is now filled in.

2. Run the VI by selecting **Operate»Run**, pressing the **<ctr-R>** keys on the keyboard, or by clicking on the **Run** button on the toolbar.



3. Observe the data flow of the VI on the block diagram. Observe the sequence of execution of the nodes.



Note: Not only does execution highlighting show the sequence of execution, but it also slows down the speed of execution so that it is visible to the user. This execution speed impacts the overall performance of the VI, and should be turned off to return execution to normal speeds.

4. Click on the **Highlight Execution** button to turn off execution highlighting. The light bulb on the button is no longer filled in.

Note: When an application appears to be running suspiciously slowly, it is always a good idea to check the **Highlight Execution** button to see if execution highlighting is enabled.
