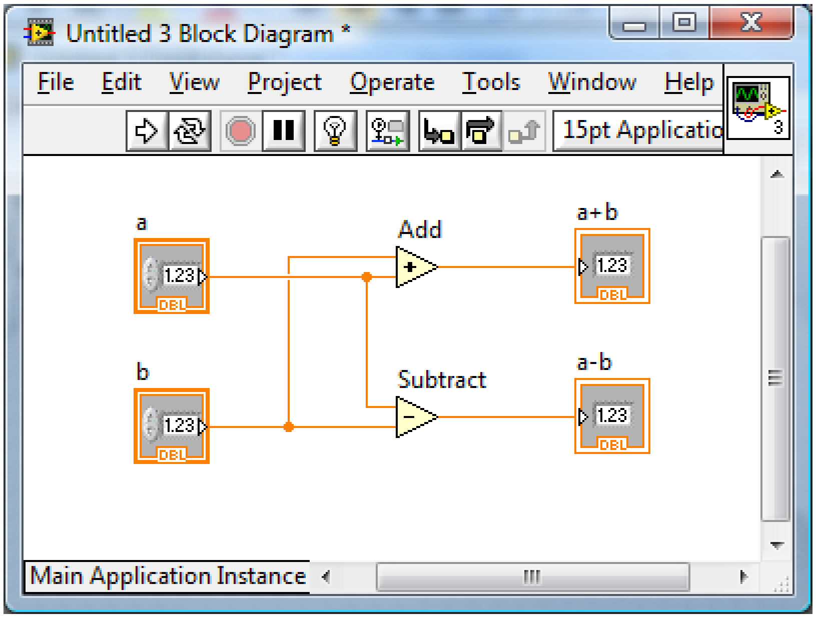
Chapter
3

LabVIEW Programming

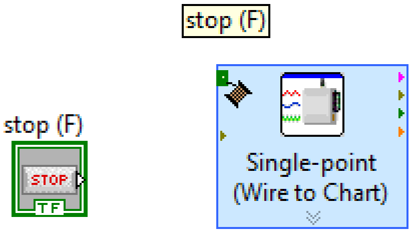
WIRES

Think of wires as a path for data to flow. Data comes into block diagram objects through a wire and can leave only through a wire. In the figure below, wires connect the control and indicator terminals to the Add and Subtract functions. As you can see, each wire has a single data source or starting point. However, you can branch off one wire, represented by a dot on the wire, and wire it to many VIs and functions. By branching off the main wire, you can send data to multiple destinations. Note that the color, style, and thickness of wires changes depending on the type of data the wire is transmitting.

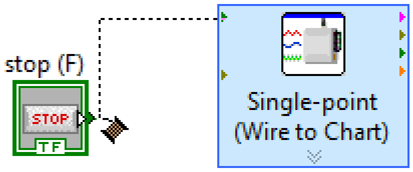


Logic rules apply to wiring in LabVIEW: each wire must have one (but only one) source (or control), and each wire may have multiple destinations (or indicators). For example, you cannot wire two indicators together, but you may wire one control to two indicators.

When you pass the Wiring tool over a terminal, the terminal blinks and a tip strip appears with the name of the terminal. Use this feature to ensure that you are selecting the correct terminal before wiring.



To wire objects together, pass the Wiring tool over the first terminal, click on the terminal, and then drag the cursor to the second terminal. If the Tools palette is on auto-select, the wiring tool automatically appears when you hold the mouse over a terminal. Click again on the destination terminal to terminate the wire. You may also click the mouse before you get to the second terminal. This will pin the wire to the block diagram at the location at which you click. This allows you to make a turn in the wire path.



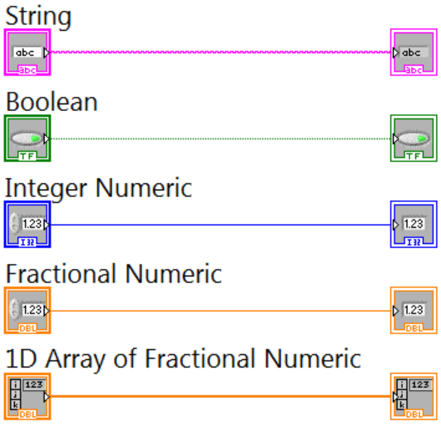
After wiring, you may want to clean up the path of the wire. Move your cursor to the wire. When the cursor turns into an arrow, click on the wire and drag the wire to relocate it. In addition, you can right-click the wire and select Clean Up Wire from the shortcut menu. LabVIEW automatically chooses a path for the wire. And finally, the wires and objects on a block diagram can be organized using the Clean Up Diagram button on the toolbar.

DATA TYPES

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 icon contains useful information about the front panel object it corresponds to. For example, the color and symbols provide information about the data type. The dynamic data type, used by many Express VIs, is a data type represented by dark blue terminals; Boolean terminals are green with TF lettering; and Strings are pink.

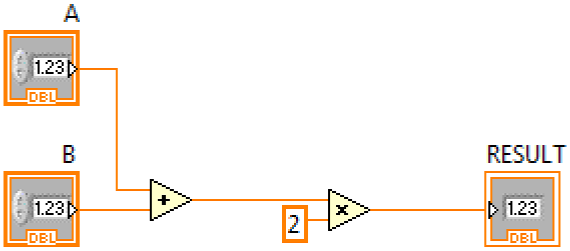
Every wire also has a type based on the data that it is transmitting. The data type of a wire determines which object, indicators, or functions you can connect a wire to. For example, if a Boolean switch has a green border, you can wire that switch to any input terminal with a green label. Note that the wire will also be green, reflecting the Boolean data type. If a knob has an orange border, you can wire a knob to any input terminal with an orange label and the wire will be orange. You cannot wire an orange knob to an input terminal with a green label because the data types are not compatible. In most cases, look for a match in color, but this is not a hard-and-fast rule; LabVIEW will allow a user to connect an Express VI's dark blue terminal to an orange terminal that represents a real number (fractional numeric value), for example.

The figure below shows the correct wiring of common data types. Note that when working in LabVIEW, the color of the wire and control matches the color of the input terminal.

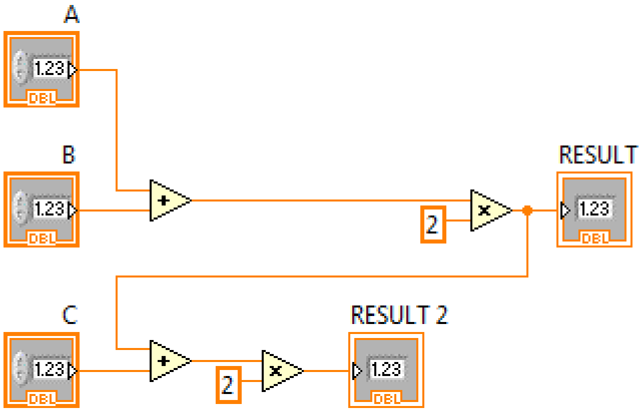


DATAFLOW PROGRAMMING

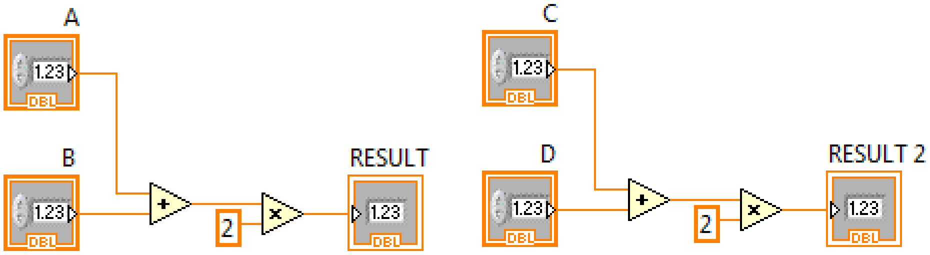
LabVIEW follows a dataflow model for running VIs. A block diagram object executes when all its inputs are available. When an object completes execution, it supplies data to its output terminals and passes the output data to the next object in the dataflow path.



Consider the block diagram in the figure above. In this program, the block diagram executes from left to right, not because the objects are placed in that order, but because the upper input of the Multiply function is not valid until the Add function has finished executing and passed the data to the Multiply function. Remember that an object 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 figure above, the flow of the code dictates that the addition of C with RESULT does not occur until A and B have been added together and this sum is multiplied by 2. This means that RESULT will happen before RESULT 2. The length of the wires does not slow or speed the code.

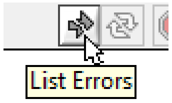


And in this final example, the code that generates RESULT 2 is to the right of the code that generates RESULT, but the position of code does not matter. What matters is the flow of the code and whether an object has received all of the required data input. Both will begin executing at the same time and run independent of one another. The two results will happen at approximately the same time.

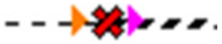
DEBUGGING TECHNIQUES

LabVIEW has several useful tools for debugging a program, a few of which are discussed below.

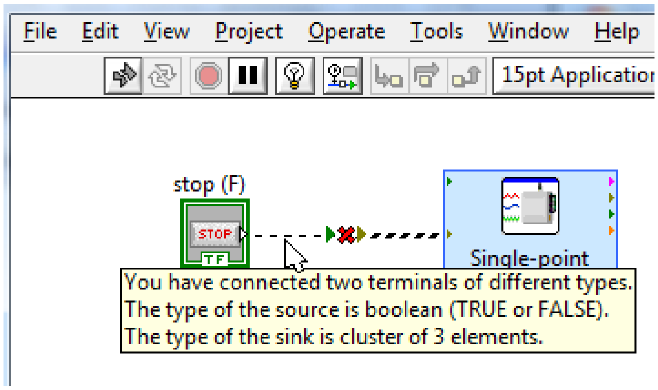
When your VI is not executable, a broken arrow is displayed in the Run button in the palette. To list errors, click the broken arrow. An Error List dialog box will appear. To locate the bad object, double-click the error message that is displayed in the dialog box.



Broken wires (wires that are not connected properly) prevent your VI from running. A broken wire appears as a dashed black line with a red X in the middle, as shown below.



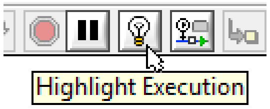
Broken wires occur for a variety of reasons, such as wiring two objects with different or incompatible data types; for example, you cannot wire a Boolean control to a numeric input terminal. The data in the wire is a Boolean and the input is expecting a numeric value, so the data types are not compatible.



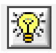
You have connected two terminals of different types.
The type of the source is boolean (TRUE or FALSE).
The type of the sink is cluster of 3 elements.

You must manually clean up a broken wire by selecting it and deleting it. A quick way to get rid of all broken wires is to use the shortcut <Ctrl-B> or choose Remove Broken Wires from the Edit menu.

When trying to visualize the data flow of a particular VI, a very useful tool is Highlight Execution. Implement execution highlighting by clicking the Highlight Execution button on the toolbar.

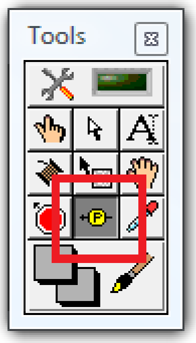


When Highlight Execution is selected, the icon will change to represent a bulb turned on.

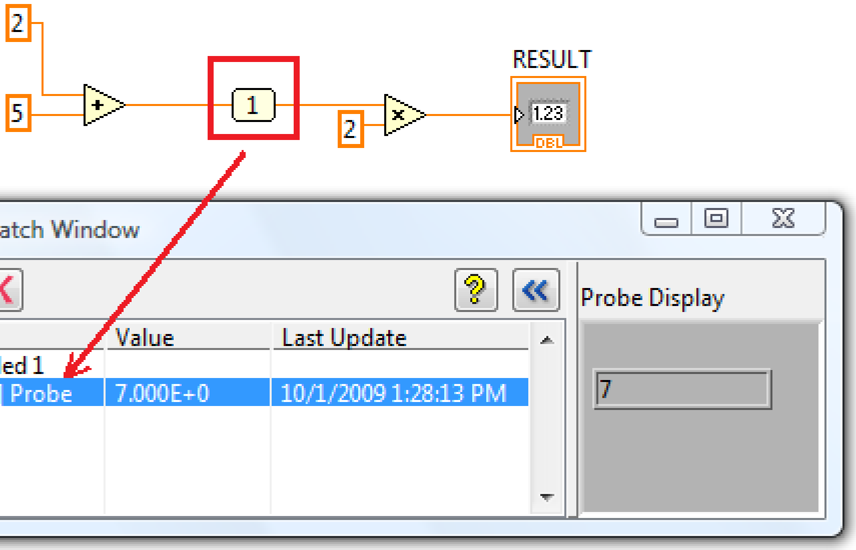


Now, when you run the VI, observe the data flow of the program's block diagram. Not only does execution highlighting show the sequence of execution, it also slows down the speed of execution so that it is visible to the user. Beware that this execution speed impacts the overall performance of the VI, and should be turned off to return execution to normal speeds.

Execution highlighting allows you to view the data as it flows through the wires, but sometimes it is not easy to view just a single wire's data with execution highlighting. In addition, it may be necessary to view a wire's data as a program runs at full speed. To view a wire's data as a program runs at full speed, click the wire with the Probe Data tool that is found in the Tools Palette (highlighted in the figure below), or by selecting Probe from the shortcut menu that appears by right-clicking the wire.



After placing a probe on a wire, the Probe Watch Window dialog box appears. This window will display the value of the wire as the program runs. Note in the figure below that a probe was placed on the wire coming out of the Addition function, and it created a yellow box with a number 1. This corresponds with the probe number in the dialog box.

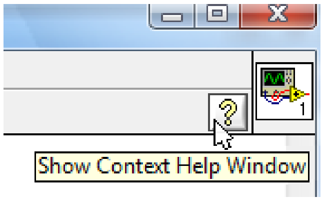


Probe(s)	Value	Last Update
Untitled 1		
[1] Probe	7.000E+0	10/1/2009 1:28:13 PM

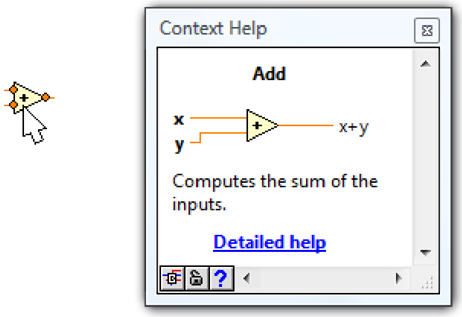
7

CONTEXT HELP

The Context Help window displays basic information about LabVIEW objects when you hover the mouse over each object. Open the Context Help window by choosing Show Context Help from the Help menu, pressing the <Ctrl-H> keys on the keyboard, or by clicking on the Show Context Help Window button on the toolbar, as seen in the figure below.



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. 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.



Computes the sum of the inputs.

TIPS FOR WORKING IN LABVIEW

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

<Ctrl-E>	Toggle between the front panel and block diagram
<Ctrl-T>	Tiles the front panel and block diagram to easily see both windows side by side
<Ctrl-B>	Remove broken wires from the block diagram
<Ctrl-H>	Display and close the context help window
<Ctrl-Z>	Undo

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. Once the Automatic Selection Tool is turned off, manually select the appropriate tool. Once you are finished with the tool you choose, you can click the Automatic Tool Selection button on the Tools palette, or press <Shift+Tab> to turn the Automatic Selection Tool back on.

In the Options dialog from the Tools menu, there are many configurable options for customizing your front panel, block diagram, colors, printing, and much more.