Graphical Programming: Detailed Explanation

Data Flow

LabVIEW follows a dataflow model for running VIs. A block diagram node executes when it receives all required inputs. When a node executes, it produces output data and passes the data to the next node in the dataflow path. The movement of data through the nodes determines the execution order of the VIs and functions on the block diagram.

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.

For a dataflow programming example, consider a block diagram that adds two numbers and then subtracts 50.00 from the result of the addition, as shown in Figure 1. In this case, the block diagram executes from left to right, not because the objects are placed in that order, but because the Subtract function cannot execute until the Add function finishes executing and passes the data to the Subtract function. Remember that a node executes only when data is available at all of its input terminals and supplies data to the output terminals only when the node finishes execution.

![Figure 1. Dataflow Programming Example](image)

In Figure 2, 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 other programming methods, such as sequence structures or error clusters, to force the order of execution.

![Figure 2. Dataflow Example for Multiple Code Segments](image)
Wires

You transfer data among block diagram objects through wires. In figures 1 and 2, wires connect the control and indicator terminals to the Add and Subtract function. Each wire has a single data source, but you can wire it to many VIs and functions that read the data. Wires are different colors, styles, and thicknesses, depending on their data types.

![Figure 3. Broken Wire Example](image)

A broken wire appears as a dashed black line with a red X in the middle, as shown in Figure 3. Broken wires occur for a variety of reasons, such as when you try to wire two objects with incompatible data types. Table 1 shows the most common wire types.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Scalar</th>
<th>1D Array</th>
<th>2D Array</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric - Floating Point</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>Orange</td>
</tr>
<tr>
<td>Numeric - Integer</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>Blue</td>
</tr>
<tr>
<td>Boolean</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>Green</td>
</tr>
<tr>
<td>String</td>
<td>[Image]</td>
<td>[Image]</td>
<td>[Image]</td>
<td>Pink</td>
</tr>
</tbody>
</table>

Table 1. Common Wire Types

In LabVIEW, you use wires to connect multiple terminals together to pass data in a VI. You must connect the wires to inputs and outputs that are compatible with the data that is transferred with the wire. For example, you cannot wire an array output to a numeric input. In addition, the direction of the wires must be correct. You must connect the wires to only one input and at least one output. For example, you cannot wire two indicators together. The components that determine wiring compatibility include the data type of the control and/or the indicator and the data type of the terminal.

For example, if a switch has a green border, you can wire a switch to any input with a green label on an Express VI. If a knob has an orange border, you can wire a knob to any input with an orange label. However, you cannot wire an orange knob to an input with a green label. Notice the wires are the same color as the terminal.

Automatically Wiring Objects

As you move a selected object close to other objects on the block diagram, LabVIEW draws temporary wires to show you valid connections. When you release the mouse button to place the object on the block diagram, LabVIEW automatically connects the wires. You also can automatically wire objects already on the block diagram. LabVIEW connects the terminals that best match and does not connect the terminals that do not match.

By default, automatic wiring is enabled when you select an object from the Functions palette or when you copy an object already on the block diagram by pressing the <Ctrl> key and dragging the object. Automatic wiring is disabled by default when you use the Positioning tool to move an object already on the block diagram.

You can adjust the automatic wiring settings by selecting Tools >> Options and selecting Block Diagram from the Category list.
Manually Wiring Objects

When you pass the Wiring tool over a terminal, a tip strip appears with the name of the terminal. In addition, the terminal blinks in the Context Help window and on the icon to help you verify that you are wiring to the correct terminal. To wire objects together, pass the Wiring tool over the first terminal, click, pass the cursor over the second terminal, and click again.

After wiring, you can right-click the wire and select Clean Up Wire from the shortcut menu to have LabVIEW automatically choose a path for the wire. If you have broken wires to remove, press <Ctrl-B> to delete all the broken wires on the block diagram.