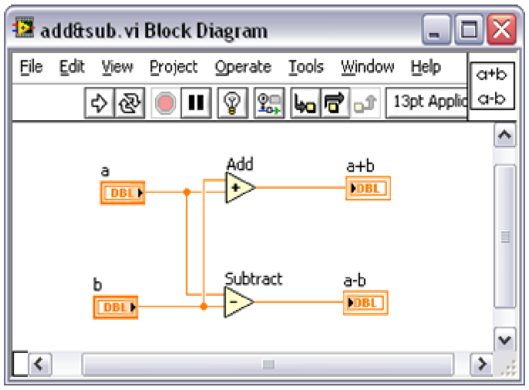
Tutorial: Wires

Publish Date: Jul 29, 2010 | 8 Ratings | **4.00** out of 5

Overview

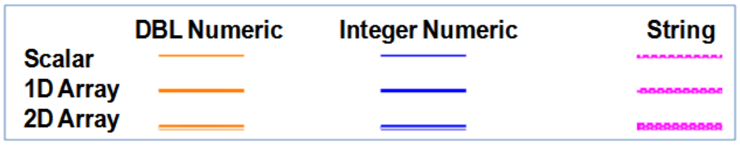
In text-based programming languages, you store and access data with functions through the use of variables. In the NI LabVIEW graphical programming language, wires implicitly handle all of the data storage and access that are associated with variables in text-based languages. Think of wires as a path 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 and wire it to many VIs and functions. By branching off the main wire, you can send data to multiple destinations. Note that wires are different colors, styles, and thicknesses. Color, style, and thickness change depending on the type of data the wire is transmitting.

Table of Contents



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.

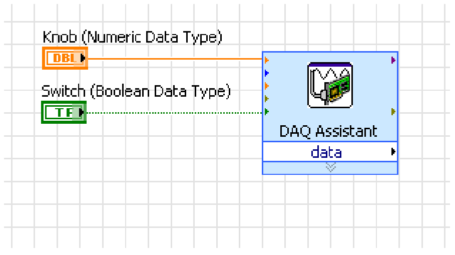
Broken wires occur for a variety of reasons such as wiring two objects with different or incompatible data types. For example, you cannot wire an array output to a numeric input. The data in the wire is an array and the input is expecting a single numeric, so the data types are not compatible. When connecting block diagram objects with wires, you must connect a wire to one input and at least one output. For example, you cannot wire two indicators together. The figure below shows the most common wire types. Notice the different colors and thicknesses of the wires indicating the different data types.



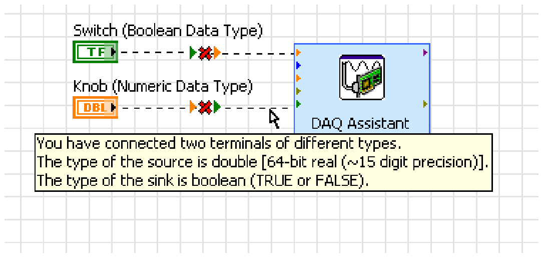
	DBL Numeric	Integer Numeric	String
Scalar	─	─	┈
1D Array	━	━	┅
2D Array	═	═	┉

Data Types

Every wire 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 switch has a green border, you can wire a switch to any input with a green label on an Express VI or function. 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 with an orange label and the wire will be orange. You cannot wire an orange knob to an input with a green label because the data types are not compatible. Note that the wires are the same color as the terminal. The figure below shows the correct wiring of two controls to a subVI. Note that the color of the wire and control matches the color of the input terminal.



The figure below shows the incorrect wiring of two controls to a subVI. The data type of the control does not match the data type of the input terminal on the subVI. Hold your mouse over the broken wire to see the data types required. Note that the source is a double and the sink, or input, is a Boolean. The wire is broken because the data types are different.



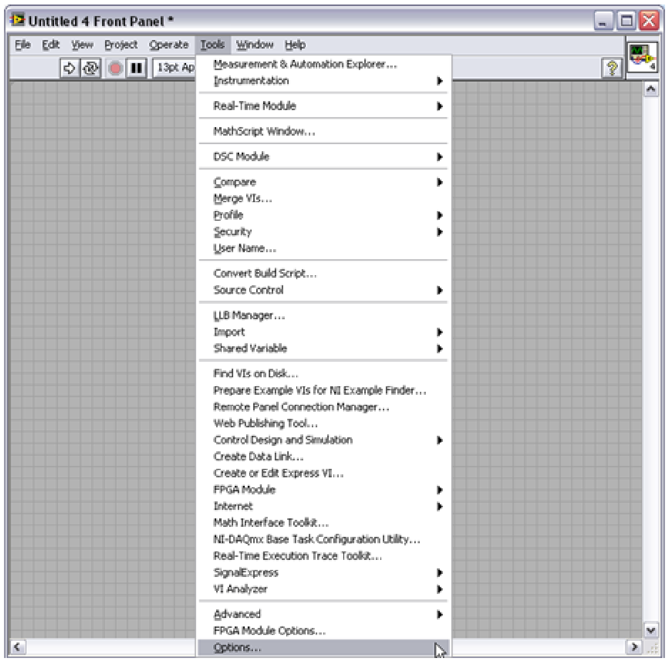
You have connected two terminals of different types.
The type of the source is double [64-bit real (~15 digit precision)].
The type of the sink is boolean (TRUE or FALSE).

Tip: You can quickly delete all broken wires from your block diagram by pressing **<ctr-B>**.

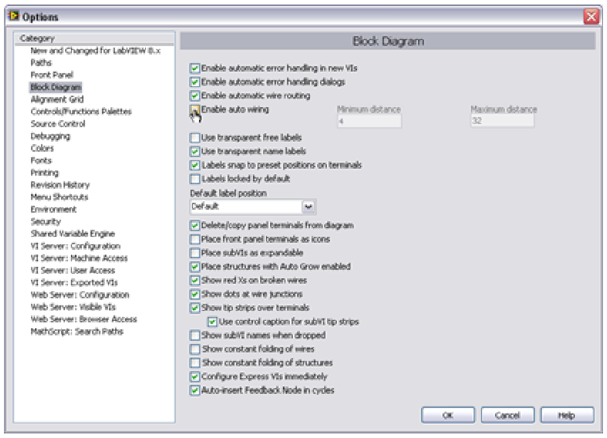
Automatically Wiring Objects

You can use the LabVIEW auto wiring feature to connect objects on the block diagram faster. 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 can also automatically wire objects already on the block diagram. LabVIEW connects only the terminals that match. Toggle automatic wiring by pressing the spacebar while you move an object using the Positioning tool.

You enable automatic wiring by default when you select an object from the **Functions** palette or when you copy an object already on the block diagram. You disable automatic wiring by default when you use the Positioning tool. You can adjust these settings by selecting **Tools»Options** and selecting **Block Diagram** from the **Category** list.



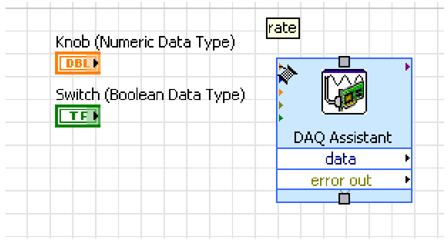
You can enable or disable automatic wiring by checking or unchecking the checkbox as shown.



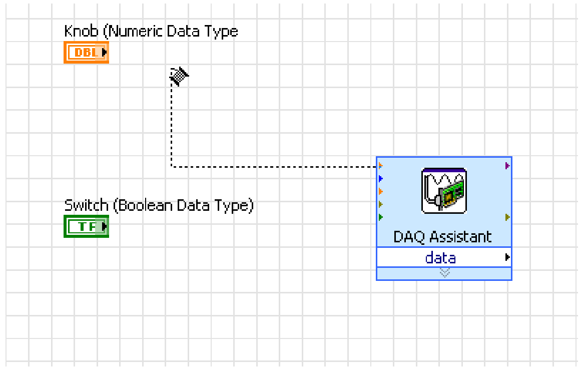
- ☑ Enable automatic error handling in new VIs
- ☑ Enable automatic error handling dialogs
- ☑ Enable automatic wire routing
- Enable auto wiring — Minimum distance 4 — Maximum distance 32
- ☐ Use transparent free labels
- ☑ Use transparent name labels
- ☑ Labels snap to preset positions on terminals
- ☐ Labels locked by default
- Default label position: Default
- ☑ Delete/copy panel terminals from diagram
- ☐ Place front panel terminals as icons
- ☐ Place subVIs as expandable
- ☑ Place structures with Auto Grow enabled
- ☑ Show red Xs on broken wires
- ☑ Show dots at wire junctions
- ☑ Show tip strips over terminals
  - ☑ Use control caption for subVI tip strips
- ☐ Show subVI names when dropped
- ☐ Show constant folding of wires
- ☐ Show constant folding of structures
- ☑ Configure Express VIs immediately
- ☑ Auto-insert Feedback Node in cycles

Manually Wiring Objects

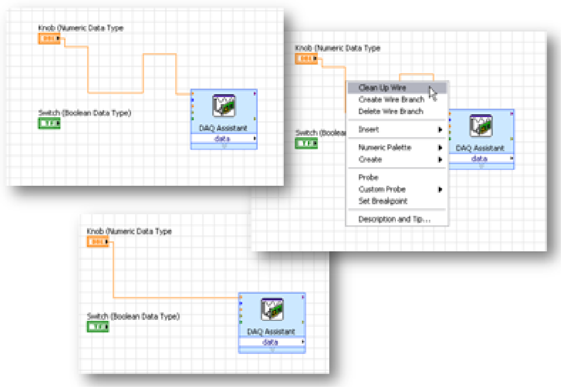
Often you may want to manually wire the objects on the block diagram together. Note that when you pass the Wiring tool over a terminal, a tip strip appears with the name of the terminal. Use this feature to ensure that you are selecting the correct terminal before wiring.



To wire objects together, pass the Wiring tool over the first terminal and click and drag the cursor over 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.

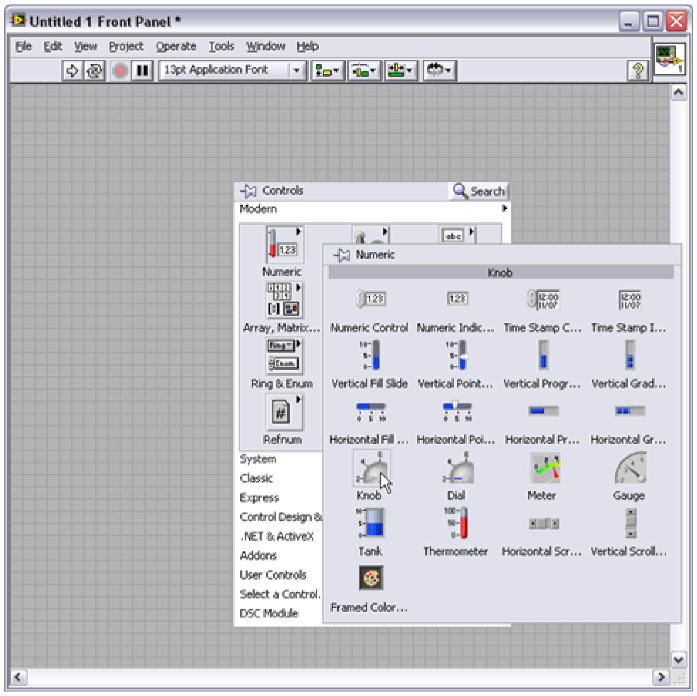


After wiring, you may want to clean up the path of the wire. Simply right-click the wire and select **Clean Up Wire** from the shortcut menu. LabVIEW automatically chooses a path for the wire.

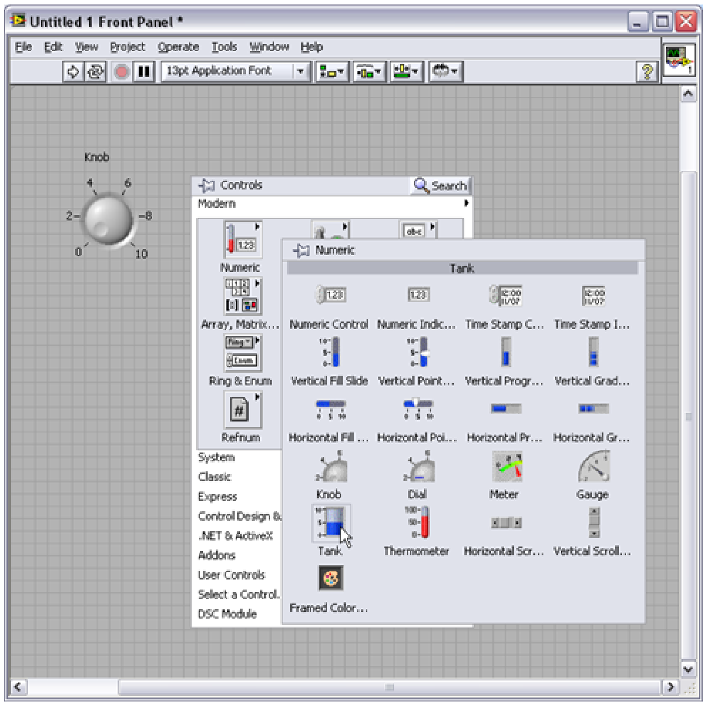


To learn more about wiring by manually wiring several controls and indicators, follow the steps below:

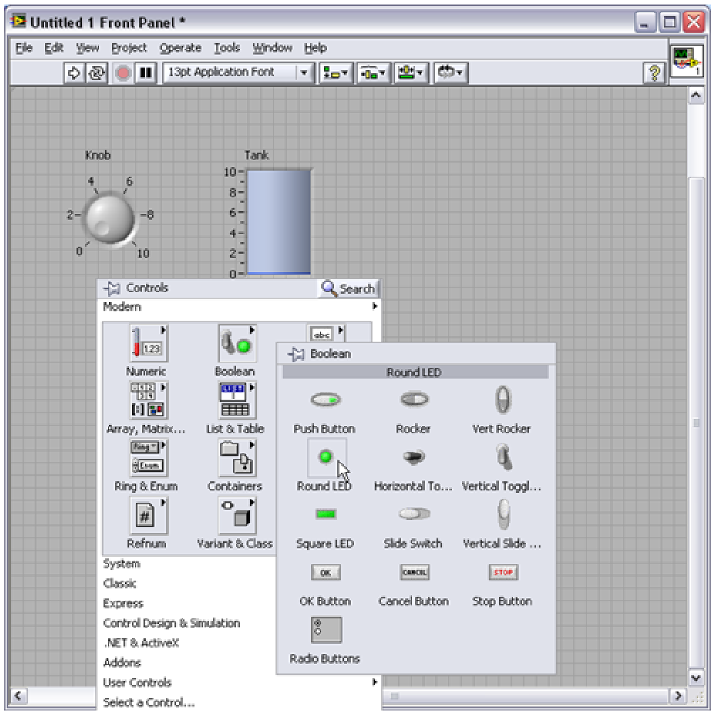
1. Create a new VI and place a knob on the front panel by selecting it from the numeric subpalette in the **Controls** palette.



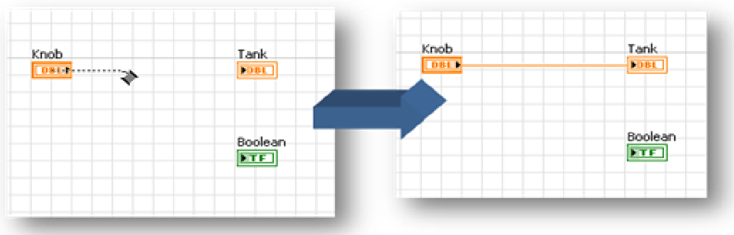
2. Place a tank indicator on the front panel to the right of the knob.



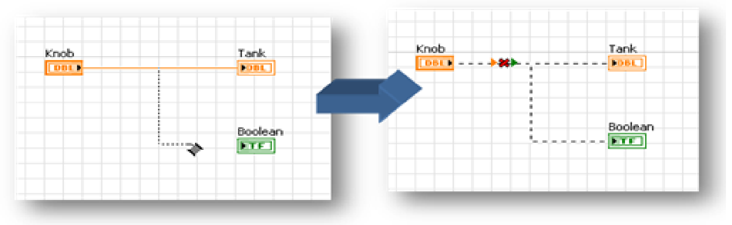
3. Place a round LED on the front panel beneath the tank.



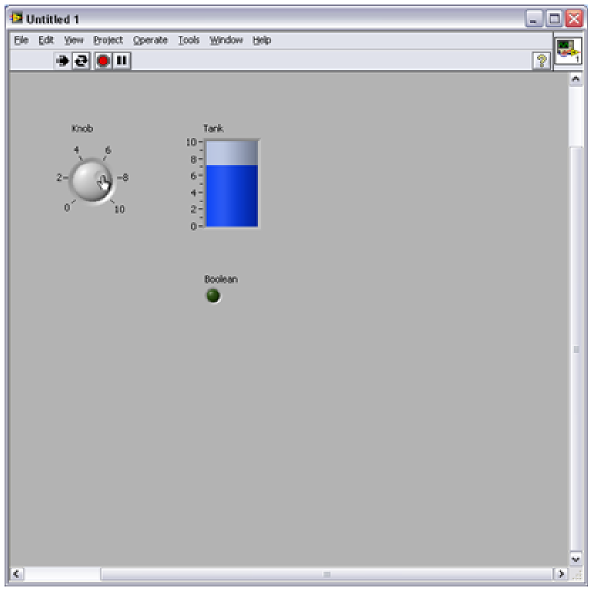
4. Now switch to the block diagram. Note that all three front panel objects are on the block diagram. Both **Knob** and **Tank** are orange and **Boolean** is green. Remember that the color represents the data type. **Knob** and **Tank** are both a data type of DBL while **Boolean** is a data type of Boolean. Hold the mouse over the output terminal of **Knob** so that the wiring tool appears. Click and drag the wire to the input terminal of **Tank**.



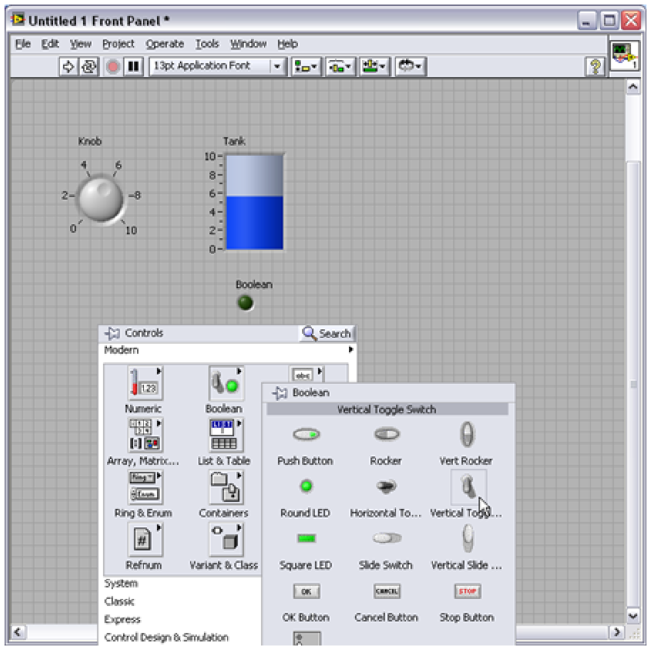
5. Hold your mouse over the wire connecting **Knob** and **Tank** so that the wiring tool appears. Click and drag to create a new wire coming off the existing wire. Try to connect the new wire to **Boolean**. Note that doing this causes the wire to break. Because **Knob** is a data type of DBL and **Boolean** is a data type of Boolean, you cannot connect them. Also notice that the **Run** arrow on the toolbar is broken. A broken wire prevents your VI from running.



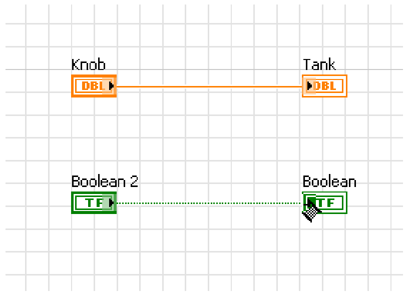
6. Delete the broken wires by pressing **<ctr-B>** and reconnect **Knob** to **Tank**. Go to the front panel and select **Run Continuously**. Use your mouse to move the **Knob** control to different values. Notice how it updates the level in the tank. This happens because the value of the **Knob** control is being passed by the wire on the block diagram to the **Tank** indicator. Because there is no data wired to **Boolean**, it does not change value.



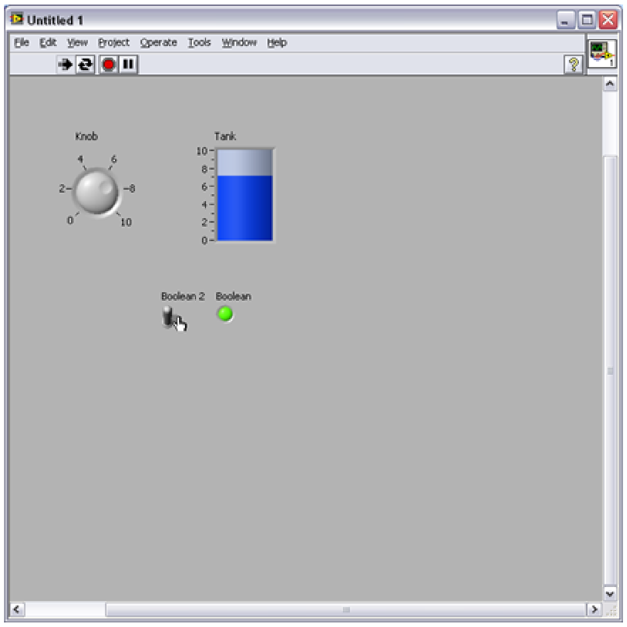
7. Select **Abort Execution** to stop the VI. From the **Controls** palette, select a vertical toggle switch and place it next to the LED.



8. Note that they are both labeled as Booleans. Switch to the block diagram and wire the output of the vertical toggle switch (**Boolean2**) to the input of the LED (**Boolean**).



9. Now select **Run Continuously** from the front panel. Use your mouse to toggle the vertical toggle switch from on to off. Observe how the LED changes from off to on.



10. Close the block diagram and front panel without saving changes.

Wires are one of the most fundamental building blocks of a virtual instrument. They determine how the elements on the block diagram interact with each other and determine the flow of the execution in general.

Video Exercise Passing Data, Debug, and SubVIs Modules Home FIRST Community

Related Links

For more information on using wires in LabVIEW, see the LabVIEW Help.