

0 OVERVIEW

In this chapter, you will investigate the LabVIEW environment and learn how its three parts—the front panel, block diagram, and icon/connector—work together. When all three main components are properly developed, you have a VI that can stand alone or be used as a subVI in another program. You will also learn about the LabVIEW environment: pull-down and pop-up menus, floating palettes and subpalettes, the Toolbar, and how to get help. To finish up, we will discuss the power of subVIs and why you should use them.

GOALS

- Understand and practice using the front panel, block diagram, and icon/connector
- Learn the difference between controls and indicators
- Be able to recognize the difference between the block diagram terminals of controls and indicators
- Understand the principle of dataflow programming
- Become familiar with LabVIEW menus, both pop-up and pull-down
- Learn about the capabilities and uses of the Toolbar, **Tools** palette, **Controls** palette, **Functions** palette, and subpalettes
- Learn why the **Help** windows can be your most valuable ally
- Understand what a subVI is and why it's useful
- Work through the activities to get a feel for how LabVIEW works

KEY TERMS

- | | | | |
|-------------|------------|----------------|---------------|
| ■ Control | ■ SubVI | ■ Dataflow | ■ Palette |
| ■ Indicator | ■ Terminal | ■ Pop-up menus | ■ Subpalette |
| ■ Wire | ■ Node | ■ Toolbar | ■ Help window |

The LabVIEW Environment: Building Your Own Workbench



3.1 Front Panels

Simply put, the *front panel* is the window through which the user interacts with the program. When you run a VI, you must have the front panel open so that you can input data to the executing program. You will also find the front panel indispensable because that's where you see your program's output. Figure 3.1 shows an example of a LabVIEW front panel.

3.1.1 Controls and Indicators

The front panel is primarily a combination of *controls* and *indicators*. **Controls** simulate typical input objects you might find on a conventional instrument, such as knobs and switches. Controls allow the user to input values; they supply data to the block diagram of the VI. **Indicators** show output values produced by the program. Consider this simple way to think about controls and indicators:

Controls = Inputs from the User = Source Terminals
Indicators = Outputs to the User = Destinations or "Sinks"

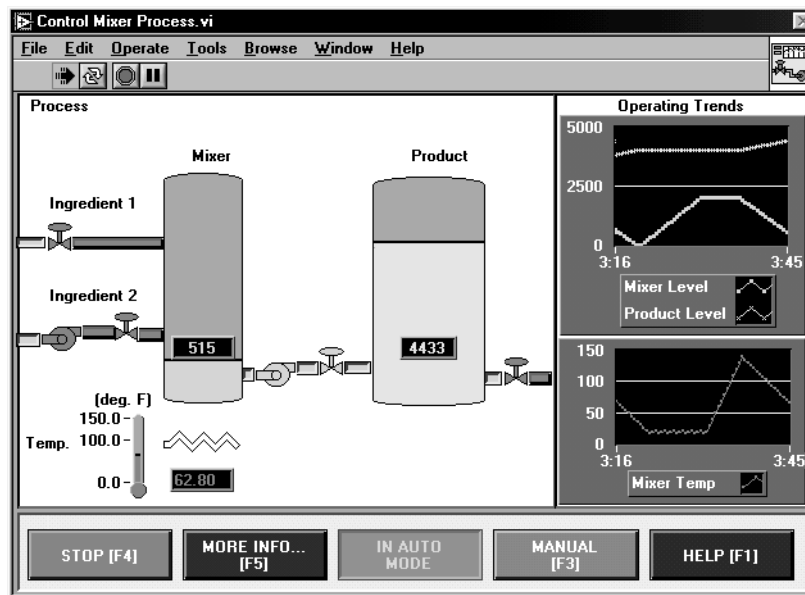


Figure 3.1
LabVIEW front panel.

They are generally not interchangeable, so make sure you understand the difference.

You “drop” controls and indicators onto the front panel by selecting them from a *subpalette* of the floating **Controls** palette window and placing them in a desired spot. Once an object is on the front panel, you can easily adjust its size, shape, position, color, and other attributes.

3.2 Block Diagrams

The *block diagram* window holds the graphical source code of a LabVIEW VI. LabVIEW’s block diagram corresponds to the lines of text found in a more conventional language like C or BASIC—it is the actual executable code. You construct the block diagram by wiring together objects that perform specific functions. In this section, we will discuss the various components of a block diagram: *terminals*, *nodes*, and *wires*.

The simple VI shown in Figure 3.2 computes the sum of two numbers. Its diagram in Figure 3.3 shows examples of terminals, nodes, and wires.

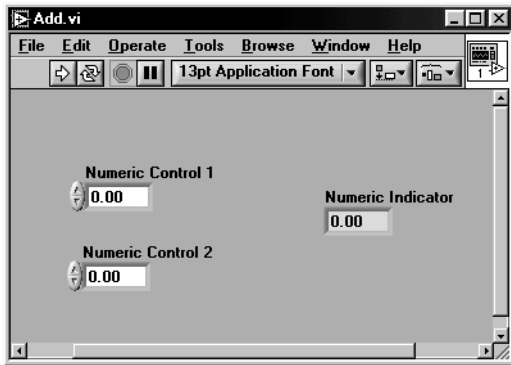


Figure 3.2

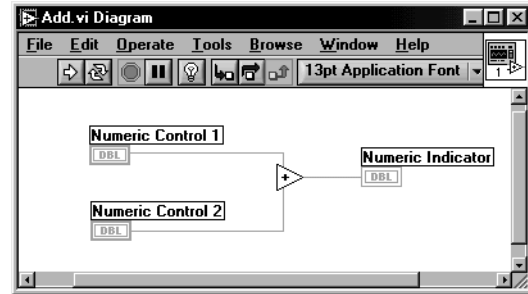


Figure 3.3

3.2.1 Terminals

When you place a control or indicator on the front panel, LabVIEW automatically creates a corresponding *terminal* on the block diagram. By default, you cannot delete a block diagram terminal that belongs to a control or indicator, although you may try to your heart's content. The terminal disappears only when you delete its corresponding control or indicator on the front panel.



Control terminals have thick borders, while indicator terminal borders are thin. It is very important to distinguish between the two since they are not functionally equivalent (Control = Input, Indicator = Output, and so they are not interchangeable).



Figure 3.4

You can think of terminals as entry and exit ports in the block diagram, or as sources and destinations. Data that you enter into Numeric Control 1 (shown in the Figure 3.3) exits the front panel and enters the block diagram through the Numeric Control 1 terminal on the diagram. The data from Numeric Control 1 follows the wire and enters the **Add** function input terminal. When the **Add** function completes its internal calculations, it produces new data values at its exit terminal. The data flows to the Numeric Indicator terminal and reenters the front panel, where it is displayed for the user.

3.2.2 Nodes

A *node* is just a fancy word for a program execution element. Nodes are analogous to statements, operators, functions, and subroutines in standard programming languages. The **Add** and **Subtract** functions represent one type of node. A structure is another type of node. Structures can execute code repeatedly or conditionally, similar to loops and Case statements in traditional programming languages. LabVIEW also has special nodes, called Formula Nodes, which are useful for evaluating mathematical formulas or expressions.

3.2.3 Wires

A LabVIEW VI is held together by *wires* connecting nodes and terminals. Wires are the data paths between source and destination terminals; they deliver data from one source terminal to one or more destination terminals. If you connect more than one source or no source at all to a wire, LabVIEW disagrees with what you're doing, and the wire will appear broken.



This principle of wires connecting source and destination terminals explains why controls and indicators are not interchangeable. Controls are source terminals, whereas indicators are destinations, or "sinks."

Each wire has a different style or color, depending on the data type that flows through the wire. The block diagram shown in Figure 3.3 depicts the wire style for a numeric scalar value—a thin, solid line. The chart in Figure 3.5 shows a few wires and corresponding types.

	Scalar	1D Array	2D Array	Color
Floating-point number				Orange
Integer number				Blue
Boolean				Green
String				Pink

Figure 3.5
Basic wire styles used in block diagrams.

To avoid confusing your data types, simply match up the colors and styles!

3.2.4 Dataflow Programming—Going with the Flow

Since LabVIEW is not a text-based language, its code cannot execute “line by line.” The principle that governs LabVIEW program execution is called *dataflow*. Stated simply, a node executes only when data arrives at all its input terminals; the node supplies data to all of its output terminals when it finishes executing; and the data pass immediately from source to destination terminals. Dataflow contrasts strikingly with the control flow method of executing a text-based program, in which instructions are executed in the sequence in which they are written. This difference may take some getting used to. Whereas traditional execution flow is instruction driven, dataflow execution is data driven or *data dependent*.

3.3 The Icon and the Connector

When your VI operates as a *subVI*, its controls and indicators receive data from and return data to the VI that calls it. A VI’s *icon* represents it as a subVI in the block diagram of another VI. An icon can include a pictorial representation or a small textual description of the VI, or a combination of both.

The VI’s *connector* functions much like the parameter list of a C or Pascal function call; the connector terminals act like little graphical parameters to pass data to and from the subVI. Each terminal corresponds to its very own control or indicator on the front panel. During the subVI call, the input parameter terminals are copied to the connected controls, and the subVI executes. At completion, the indicator values are copied to the output parameter terminals.



Figure 3.6

An icon and its underlying connector.

Every VI has a default icon, which is displayed in the icon pane in the upper-right corner of the panel and diagram windows. The default icon is depicted in Figure 3.7.

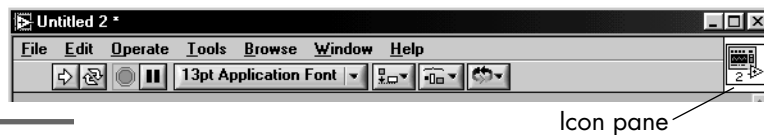


Figure 3.7

A VI's connector is hidden under the icon; access it by choosing **Show Connector** from the front panel icon pane pop-up menu (we'll talk more about pop-up menus later). When you show the connector for the first time, LabVIEW helpfully suggests a connector pattern that has one terminal for each control and indicator currently on the front panel. You can select a different pattern if you desire, and you can assign up to 28 terminals before you run out of real estate on the connector.

3.3.1 Activity 3-1: Getting Started

Okay, you've read enough for now. It's time to get some hands-on experience. Go ahead and launch LabVIEW. You will step through the creation of a simple LabVIEW VI that generates a random number and plots its value on a waveform chart. You'll learn more in the next chapter about the steps you'll be taking; for now, just get a feel for the environment.

If you are using the full version of LabVIEW, just launch it and you'll be ready to start building your first VI.

If you are using the evaluation version of LabVIEW, you can still do these activities, since the evaluation version of LabVIEW has almost no restrictions on creating and editing VIs. Just be aware that your VI cannot run more than 5 minutes, and after 30 days your evaluation version will quit working.



If you are not comfortable working through the activities in this chapter without more background information or if you have trouble getting them to work, read Chapter 4 and then come back and try again.

1. At the LabVIEW dialog box during launch, click **New VI**. You should have an "Untitled 1" front panel on your screen.

Go to the floating **Controls** palette and click on the **Graph** button to access the Graph subpalette. If the **Controls** palette isn't visible, select **Show Controls Palette** from the **Windows** menu. Also make sure the front panel window is active, or you will see the **Functions** palette instead of the **Controls** palette. On the **Graph** subpalette, select **Waveform Chart** by releasing the mouse button. You will notice that, as you run the cursor over the icons in the Controls palette and subpalettes, the selected button or icon's name appears at the top of the palette, as shown in the Figures 3.8 and 3.9.



Figure 3.8



Figure 3.9

Positioning
Tool

You will see the outline of a chart with the cursor "holding" it. Position the cursor in a desirable spot on your front panel and click. The chart magically appears exactly where you placed it. If you want to move it, select the **Positioning** tool from the **Tools** palette, and then drag the chart to its new home. If the **Tools** palette isn't visible, select **Show Tools Palette** from the **Windows** menu.

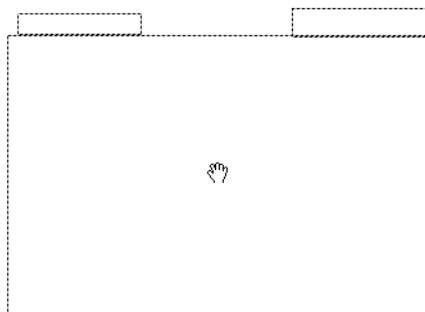


Figure 3.10

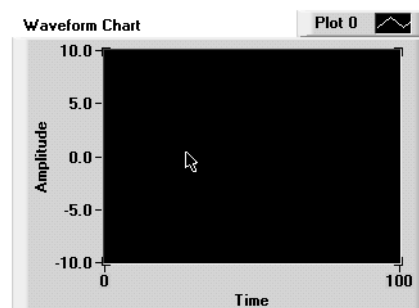


Figure 3.11

- Go back to the floating **Controls** palette, by clicking on the “Up to Owning Palette” arrow on the **Graph** subpalette (this arrow is at the top-left corner of all control palettes). From the **Controls** palette, select the **Boolean** subpalette, and choose **Vertical Toggle Switch**.



Figure 3.12

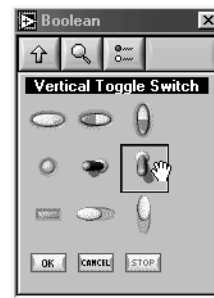


Figure 3.13

Place it next to the chart as shown in Figure 3.14.

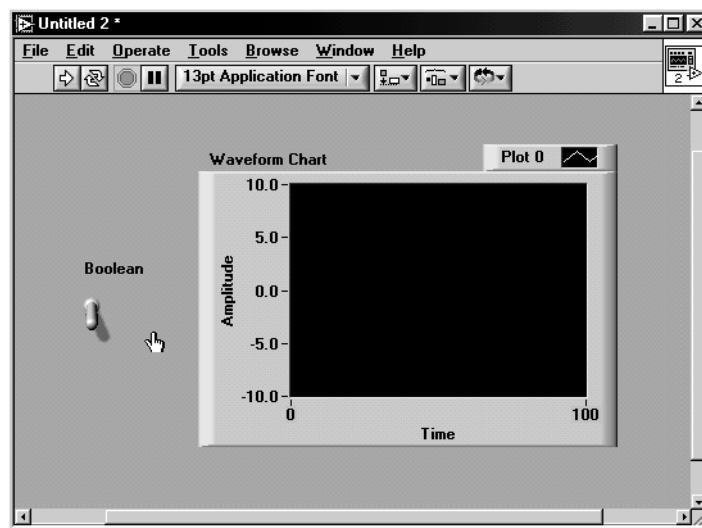
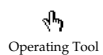


Figure 3.14



Operating Tool

- Select the Operating tool from the floating **Tools** palette.



Figure 3.15



Now change the scale on the chart. Highlight the number “10” by click-dragging or by double-clicking on it with the Operating tool. Now type in 1.0 and click on the enter button that appears in the Toolbar at the top of the window.

- Switch to the block diagram by selecting **Show Diagram** from the **Windows** menu. You should see two terminals already there (Figure 3.16).

Figure 3.16



- Now you will put the terminals inside a While Loop to repeat execution of a segment of your program. Go to the **Structures** subpalette of the floating **Functions** palette and select the **While Loop**. Make sure the block diagram window is active, or you will see the **Controls** palette instead of the **Functions** palette.



Figure 3.17

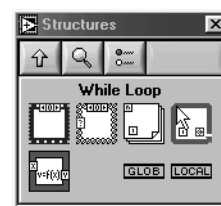


Figure 3.18

Your cursor will change to a little loop icon. Now enclose the DBL and TF terminals: Click and hold down the mouse button while you drag the cursor from the upper-left to the lower-right corners of the objects you wish to enclose.

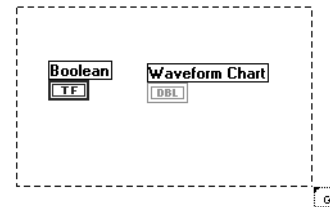


Figure 3.19

When you release the mouse button, the dashed line that is drawn as you drag will change into the While Loop border. Make sure to leave some extra room inside the loop.

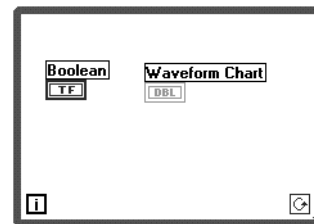


Figure 3.20

- Go to the **Functions** palette and select **Random Number (0–1)** from the **Numeric** subpalette. Place it inside the While Loop.

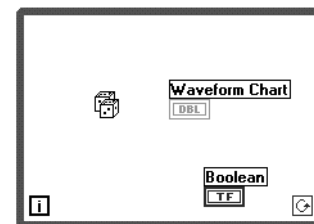
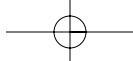




Figure 3.21

The While Loop is a special LabVIEW structure that repeats the code inside its borders until it reads a FALSE value. It is the equivalent of a Do-While Loop in a more conventional language. You'll learn more about loops in Chapter 6.



- 
 7. Select the Positioning tool from the floating Tools palette and arrange your diagram objects so that they look like the previously shown block diagram.
- 
 8. Now select the Wiring tool from the **Tools** palette. Click once on the **Random Number (0–1)** icon, drag the mouse over to the DBL terminal, and click again.

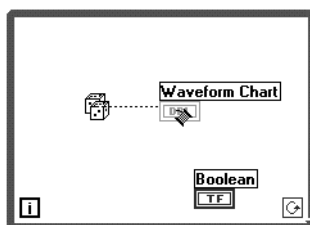




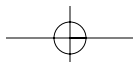
Figure 3.22

You should now have a solid orange wire connecting the two icons. If you mess up, you can select the wire or wire fragment with the Positioning tool and then hit the <delete> key to get rid of it. Now wire the Boolean TF terminal to the conditional terminal of the While Loop. The loop will execute while the switch on the front panel is TRUE (in the “up” position) and stop when the switch becomes FALSE.



Figure 3.23

- 
 9. You should be about ready to run your VI. First, switch back to the front panel by selecting **Show Panel** from the **Windows** menu. Using the Operating tool, flip the switch to the “up” position. Now click on the run button to run your VI. You will see a series of random
- 



numbers plotted continuously across the chart. When you want to stop, click on the switch to flip it to the down position.

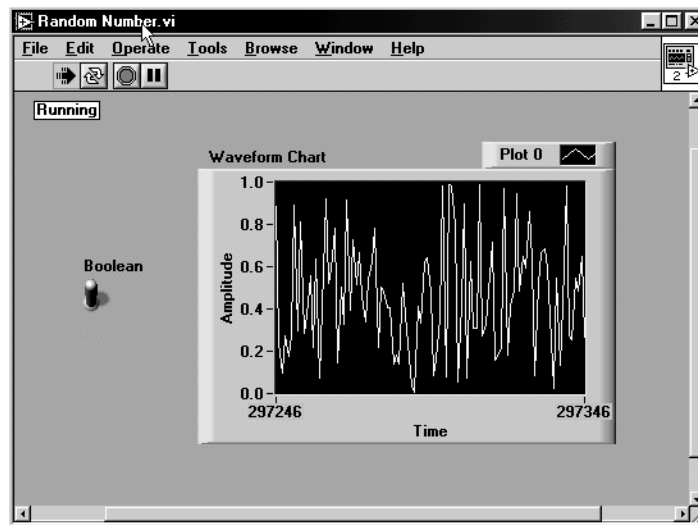


Figure 3.24

10. Create a directory or folder called `MYWORK` in your `LabVIEW` directory. Save your VI in your `MYWORK` directory or folder by selecting **Save** from the **File** menu and pointing out the proper location to save to. Name it **Random Number.vi**.



Save all of your subsequent activities in `MYWORK` so you can find them easily!



Remember, if you get stuck or just want to compare your work, the solutions to every activity in this book can be found in the `EVERYONE` directory or folder on the accompanying CD. You can view them in the sample version or the full version of LabVIEW.

Congratulate yourself—you've just written your first LabVIEW program! Don't worry that it doesn't actually do much—your programs will be more powerful and have more of a purpose soon enough!

3.4 Pull-Down Menus

Keep in mind that LabVIEW's capabilities are many and varied. This book by no means provides an exhaustive list of all of LabVIEW's ins and outs (it would be several thousand pages long if that were the case); instead, we try to get you up to speed comfortably and give you an overview of what you can do. If you want to know everything there is to know about a subject, we'd recommend looking it up in one of LabVIEW's many manuals, attending a seminar, or going to ni.com/labview. See Appendix A for an exhaustive list of other resources. Feel free to skim through this section and some of the subsequent ones, but remember that they're here if you need a reference.

LabVIEW has two main types of menus: pull-down and pop-up. You used some of them in the last activity, and you will use both extensively in all of your program development henceforth. Now you will learn more about what they can do. We'll cover pull-down menu items very briefly in this section. You might find it helpful to look through the menus on your computer as we explain them, and maybe experiment a little.

The menu bar at the top of a VI window contains several pull-down menus. When you click on a menu bar item, a menu appears below the bar. The pull-down menus contain items common to many applications, such as **Open**, **Save**, **Copy**, and **Paste**, and many other functions particular to LabVIEW. We'll discuss some basic pull-down menu functions here. You'll learn more about the advanced capabilities later.

Many menus also list shortcut keyboard combinations for you to use if you choose. To use keyboard shortcuts, press the appropriate key in conjunction with the <control> key on PCs, the <command> key on Macs, the <meta> key on Suns, and the <alt> key on HP machines.



Many of the menu items show keyboard shortcuts to the right of their corresponding commands. You may want to use the shortcuts instead of the menus.



Figure 3.25



Figure 3.26



Figure 3.27

File Menu

Pull down the **File** menu, which contains commands common to many applications such as **Save** and **Print**. You can also create new VIs or open existing ones from the **File** menu.

Edit Menu

Take a look at the **Edit** menu. It has some universal commands, like **Undo**, **Cut**, **Copy**, and **Paste**, that let you edit your window. You can also search for objects with the **Find...** command and remove bad wires from the block diagram.

Operate Menu

You can run or stop your program from the **Operate** menu (although you'll usually use Toolbar buttons). You can also change a VI's default values, control 'print and log at completion' features, and switch between run mode and edit mode.

Tools Menu

The **Tools** menu lets you access built-in and add-on tools and utilities that work with LabVIEW, such as the **Measurement & Automation Explorer**, where you configure your DAQ devices, or the **Web Publishing Tool** for



Figure 3.28

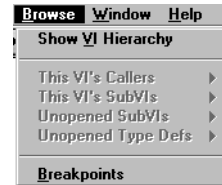


Figure 3.29



Figure 3.30

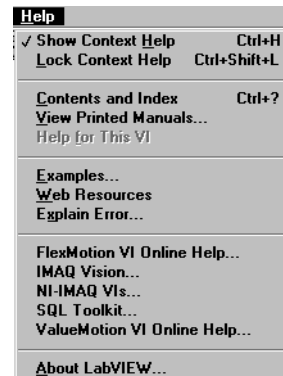


Figure 3.31

creating HTML pages from LabVIEW. You can view and change the myriad LabVIEW **Options...**

Browse Menu

The **Browse** menu contains features to simplify navigation among large sets of VIs. You can see VI hierarchy, determine all of a VI's subVIs, and view debugging breakpoints.

Windows Menu

Pull down the **Windows** menu. Here you can toggle between the panel and diagram windows, show the error list and the clipboard, "tile" both windows

so you can see them at the same time, and switch between open VIs. You can also bring up floating palettes if you've closed them. In addition, you can show VI information and development history from this menu.

Help Menu

You can show, hide, or lock the contents of the Help window using the **Help** menu. You can also access LabVIEW's online reference information and view the About LabVIEW information window.

3.5 Floating Palettes

LabVIEW has three often-used floating palettes that you can place in a convenient spot on your screen: the **Tools** palette, the **Controls** palette, and the **Functions** palette. You can move them around by clicking on their title bar and dragging. Close them just like you would close any window in your operating system. If you decide you want them back, use the **Show... Palette** function in the **Windows** menu.

3.5.1 Controls and Functions Palettes

You will be using the **Controls** palette a lot, since that's where you select the controls and indicators that you want on your front panel. You will probably use the **Functions** palette even more often, since it contains the functions and structures used to build a VI.

The Controls and Functions palettes are unique in several ways. *Most importantly, the Controls palette is only visible when the front panel window is active, and the Functions palette is only visible when the block diagram window is active.* Both palettes have *subpalettes* containing the objects you need to access. As you pass the cursor over each subpalette button in the Controls and Functions palettes, you will notice that the subpalette's name appears at the top of the window.



Figure 3.32

If you click on a button the associated subpalette appears and replaces the previous active palette. To select an object in the subpalette, click the mouse button over the object, and then click on the front panel or block diagram to place it where you want it. Like palette button names, subpalette object names appear when you run the cursor over them. To return to the previous (“owning”) palette, just click on the top-left arrow on each palette. You can search for a specific item in a palette by clicking on the spyglass icon, and you can edit your own palettes by clicking the options button.

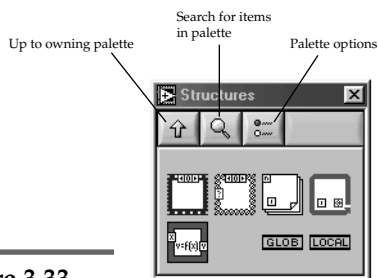


Figure 3.33

There is another way to navigate palettes that some people find a little easier. Instead of having each subpalette replace the current palette, you can pass through subpalettes in a hierarchical manner without them replacing their parent palettes. You can do this by right-clicking (Windows) or command-clicking (MacOS) the buttons on palettes.

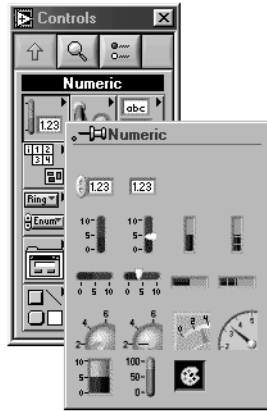


Figure 3.34

Note that some subpalettes have subpalettes containing more objects; these are denoted by a little triangle in the upper-right corner of the icon and a raised appearance. We'll discuss specific subpalettes and their objects in the next chapter.

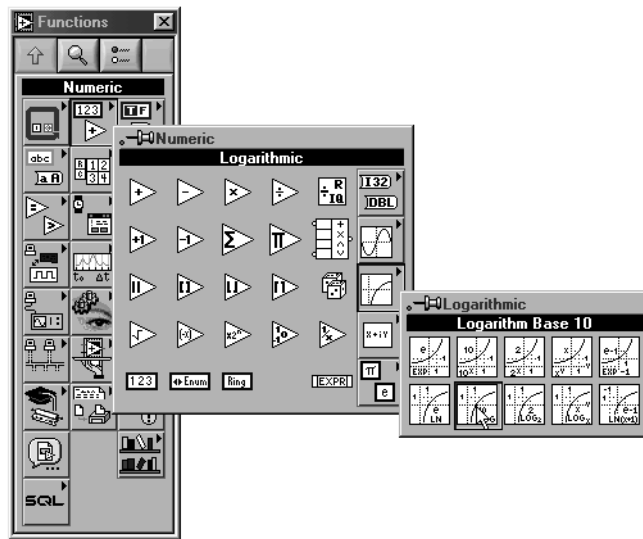


Figure 3.35



The Controls and Functions palettes can also be accessed by popping up in an empty area of the front panel or block diagram. "Popping up" is defined as right-mouse-clicking on the PC, Sun, and HP machines, and <command>-clicking on the Mac. You can also pop up with the soon-to-be-discussed Pop-up Tool.

3.5.2 The Thumbtack

If you use a subpalette frequently, you may want to “tear it off” by releasing the mouse button over the *thumbtack* located at the upper left of the palette. This thumbtack is available when you navigate hierarchically through palettes by right-clicking (<command>-clicking on the Mac), as we just described. You now have a stand-alone window that you can position anywhere and then close when you’re done with it. You can leave open as many subpalettes as you like.

3.5.3 Customizable Palettes

If LabVIEW’s default organization of the **Controls** and **Functions** palettes doesn’t fit your needs, you can customize them according to your whim. Access the menu editor by clicking on the “options” icon for a palette. From here, you can create your own palettes and customize existing views by adding new subpalettes, hiding items, or moving them from one palette to another. For example, if you create a VI using trigonometric functions, you can place it in the existing **Trigonometric** subpalette for easy access. Editing the palettes is handy for placing your most frequently used functions at the top level for easy access and burying those pesky functions you never want to see again at the bottom of a subpalette. You can also choose whether to show the icons, the text, or both on palettes.

You’ll learn more about how to customize palettes in Chapter 4. You can also use the built-in “Data Acquisition” or “Test and Measurement” sets if those configurations are more convenient for you.

3.5.4 Tools Palette

A *tool* is a special operating mode of the mouse cursor. You use tools to perform specific editing and operation functions, similar to how you would use them in a standard paint program.



Figure 3.36

Like the **Controls** and **Functions** palettes, the **Tools** palette window can be relocated or closed. To select a tool, click the appropriate button on the **Tools** palette and your mouse cursor will change accordingly. If you're not sure which tool is which, hold your cursor over the button until a tip *strip* appears describing the tool.



The Operating tool lets you change values of front panel controls and indicators. You can operate knobs, switches, and other objects with the Operating tool—hence the name. It is the only front panel tool available when your VI is running or in run mode (described shortly).



The Positioning tool selects, moves, and resizes objects.



The Labeling tool creates and edits text labels.



The Wiring tool wires objects together on the block diagram. It is also used to assign controls and indicators on the front panel to terminals on the VI's connector.



The Color tool brightens objects and backgrounds by allowing you to choose from a multitude of hues. You can set both foreground and background colors by clicking on the appropriate color area in the Tools palette. If you pop up on an object with the Color tool, you can choose a hue from the color palette that appears.



The Pop-up tool opens an object's pop-up menu when you click on the object with it. You can use it to access pop-up menus instead of the standard method for popping up (right-clicking under Windows and Unix and <command>-clicking on MacOS).



The Scroll tool lets you scroll in the active window.



The Breakpoint tool sets breakpoints on VI diagrams to help you debug your code. It causes execution to suspend so that you can see what is going on and change input values if you need to.



The Probe tool creates probes on wires so that you can view the data traveling through them while your VI is running.



Use the Color Copy tool to pick up a color from an existing object, and then use the Color tool to paste that color onto other objects. This technique is very useful if you need to duplicate an exact shade but can't remember which one it was. You can also access the Color Copy tool

when the Color tool is active by holding down the <control> key on Windows, <option> on MacOS, <meta> on Sun, and <alt> on Linux and HP-UX.



You can use the <tab> key to tab through the Tools palette instead of clicking on the appropriate tool button to access a particular tool. Or press the space bar to toggle between the Operating tool and the Positioning tool when the panel window is active and between the Wiring tool and the Positioning tool when the diagram window is active. The <tab> and space bar shortcuts cycle through the most frequently used tool for your convenience—try using them, and see how they save you time!

You can also access a temporary copy of the Tools palette by pop-up clicking (<shift>-right click for Windows and Unix and <command-shift>-click on MacOS).

3.6 The Toolbar

The *Toolbar*, located at the top of LabVIEW windows, contains buttons you will use to control the execution of your VI, as well as text configuration options and commands to control the alignment and distribution of objects. You'll notice that the Toolbar has a few more options in the block diagram than in the front panel and that a few editing-related options disappear when you run your VI. If you're not sure what a button does, hold the cursor over it until a tip strip appears describing its function.



Figure 3.37



Run Button



Run Button
(active)



Run Button
(broken)

The Run button, which looks like an arrow, starts VI execution when you click on it. It changes appearance when a VI is actually running. When a VI won't compile, the run button is broken.



The Continuous Run button causes the VI to execute over and over until you hit the stop button. It's kind of like a GOTO statement (sort of a programming "no-no"), so use it sparingly.



The Abort button, easily recognizable because it looks like a tiny stop sign, becomes active when a VI begins to execute; otherwise the Abort button is grayed out. You can click on this button to halt the VI.



Using the Abort button is like pulling the power cord on your computer. Your program will stop immediately rather than coming to a graceful end, and data integrity can be lost this way. You should always code a more appropriate stopping mechanism into your program, as we will demonstrate later.



The Pause button pauses the VI so that you can use single-step debugging options such as step into, step over, and step out. Hit the pause button again to continue execution.



The single-step buttons, Step Into, Step Over, and Step Out, force your VI to execute one step at a time so you can troubleshoot. We'll talk more about how to use them in Chapter 5.



The Execution Highlight button causes the VI to highlight the flow of data as it passes through the diagram. When execution highlighting is on, you can see intermediate data values in your block diagram that would not otherwise appear.



The Warning button appears if you have configured your VI to show warnings and you have any warnings outstanding. You can list the warnings by clicking on the button. A warning is not an error; it just alerts you that you are doing something you may not have intended (for example, you have a front panel control with nothing wired to it).

You can change the font, size, style, justification, and color of LabVIEW text from the Font ring on the Toolbar.

Figure 3.38

LabVIEW has an automatic alignment mechanism to help you line up and evenly space your icons. Select the objects you want to align by dragging around them with the Positioning tool, and then go to the Alignment ring on the Toolbar and choose how you want to align them (top edges flush, left edges flush, vertical centers, etc.). If you want to set uniform spacing between objects, use the Distribution ring in a similar fashion.



Alignment ring Distribution ring

Figure 3.39

In a similar fashion, LabVIEW lets you group objects together to treat them as one control for graphical editing purposes, as well as set the depth order of objects, so that you can specify which objects should go in front of or behind others. You can do this with the Reorder ring.

Figure 3.40



Run Mode and Edit Mode

When you open a VI, it opens in *edit mode* so that you can make changes to it. When you run a VI, it automatically goes into *run mode* and you can no longer edit. Only the **Operating** tool is available on the front panel when the VI is in run mode. When your VI completes execution, your VI reverts to edit mode (unless you manually switched it to run mode before you ran it—then it stays in run mode). You can switch to run mode by selecting **Change to Run Mode** from the **Operate** menu; switch back to edit mode by choosing **Change to Edit Mode**. To draw a parallel with text-based languages, if a VI is in run mode, it has been successfully compiled and awaits your command to execute. Most of the time, you will not need to concern yourself with run and edit modes. But if you accidentally find that you suddenly have only the Operating tool, and you can't make any changes, at least now you'll know why.



If you prefer to open VIs in run mode (perhaps so uninvited users can't make changes), select Options... from the Tool menu. Go to the Miscellaneous options and choose "Open VIs in Run Mode."

3.7 Pop-Up Menus

As if pull-down menus didn't give you enough to learn about, we will now discuss the other type of LabVIEW menu, the pop-up menu. You will probably use pop-up menus more often than any other LabVIEW menu. To pop up, position the cursor over the object whose menu you desire; then click the right mouse button on Windows and UNIX machines, or hold down the <command> key and click on the Mac. You can also click on the object with the Pop-up tool. A pop-up menu will appear.

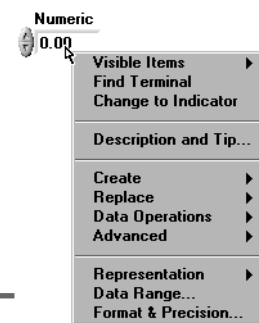


Figure 3.41

Virtually every LabVIEW object has a pop-up menu of options and commands. Options available in this pop-up menu depend on the kind of object, and they are different when the VI is in edit mode or run mode. For example, a numeric control will have a very different pop-up menu than a graph indicator. If you pop up on empty space in a front panel or block diagram, you will get the **Controls** or **Functions** palette, respectively.

You will find that instructions throughout this book guide you to select a command or option from an object pop-up menu—so try popping up now!



How to Pop Up

Windows and UNIX: right mouse click on the object

Mac: <command>-click on the object

All Platforms: Click on the object with the Pop-up tool

Pop-up menus are ever-present in LabVIEW. They contain most configuration options for an object. So remember, when in doubt about how to do something, try popping up!



If the Color tool is active, you will see a color palette when you pop up instead of the pop-up menu that appears when other tools are active.

3.7.1 Pop-Up Menu Features to Keep in Mind

Many pop-up menu items expand into submenus called hierarchical menus, denoted by a right arrowhead (see Figure 3.42). Hierarchical menus sometimes have a selection of mutually exclusive options. The currently selected option is denoted by a check mark for text-displayed options or is surrounded by a box for graphical options.

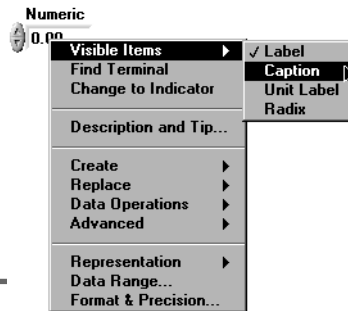


Figure 3.42

Some menu items pop up dialog boxes containing options for you to configure. Menu items leading to dialog boxes are denoted by ellipses (...).

Menu items without right arrowheads or ellipses are usually commands that execute immediately upon selection. A command usually appears in verb form, such as **Change to Indicator**. When selected, some commands are replaced in the menu by their inverse commands. For example, after you choose **Change to Indicator**, the menu selection becomes **Change to Control**.



Sometimes different parts of an object have different pop-up menus. For example, if you pop up on an object's label, the menu contains only a *Size to Text* option. Popping up elsewhere on the object gives you a full menu of options. So if you pop up and don't see the menu you want, try popping up elsewhere on the object.

3.7.2 Pop-Up Features Described

Pop-up menus allow you to specify many traits of an object. The following options appear in numerous pop-up menus, and we thought they were important enough to describe them individually. We'll let you figure out the other options, since we would put you to sleep detailing them all. Feel free to skim over this section and refer back to it when necessary.

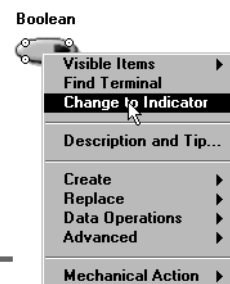


Figure 3.43

Visible Items

Many items have **Visible Items** menus with which you can show or hide certain cosmetic features like labels, captions, scrollbars, or wiring terminals. If you select **Visible Items**, you will get another menu off to the side, listing options of what can be shown (this list varies depending on the object). If an option has a check next to it, that option is currently visible; if it has no check, it is hidden. Release the mouse on an option to toggle its status.

Find Terminal and Find Control/Indicator

If you select **Find Terminal** from a front panel pop-up menu, LabVIEW will locate and highlight its corresponding block diagram terminal. If you select **Find Control/Indicator** from a block diagram pop-up menu, LabVIEW will show you its corresponding front panel object.

Change to Control and Change to Indicator

By selecting **Change to Indicator**, you can turn an existing control (an input object) into an indicator (an output object), or vice versa if you select **Change**

to Control. When an object is a control, its pop-up menu contains the option to **Change to Indicator**. When it is an indicator, the pop-up menu reads **Change to Control**.



Since **Change to Control/Indicator** is an option in the pop-up menu, it is easy to accidentally select it without realizing what you've done. Because controls and indicators are not functionally interchangeable in a block diagram, the resulting errors may befuddle you.

A control terminal in the block diagram has a thicker border than an indicator terminal. Always pay attention to whether your objects are controls or indicators to avoid confusion!

Description and Tip

Selecting this option will allow you to enter a description and a “tip.” The description will appear in the Help window for that control, and the tip will show up when you place the mouse cursor over this control (this type of help is sometimes called tool-tip or hesitation help).

Create...

The **Create...** option is an easy way for you to create a property node, local variable, or reference for a given object (these advanced topics will be covered in detail in Chapter 12.)

Replace

The **Replace** option is extremely useful. It gives you access to the **Controls** or **Functions** palette (depending on whether you're in the front panel or block diagram) and allows you to replace the object you popped up on with one of your choice. Where possible, wires will remain intact.

Data Operations

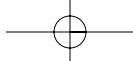
The Data Operations pop-up menu has several handy options to let you manipulate the data in a control or indicator:

- **Reinitialize to Default** returns an object to its default value, while **Make Current Value Default** sets the default value to whatever data are currently there.
- Use **Cut Data**, **Copy Data**, and **Paste Data** to take data out of or put data into a control or indicator.
- **DataSocket Connection...** brings up a dialog box where you can configure this control to be connected to a DataSocket URL. We'll talk more about DataSocket in Chapter 14.

Advanced

The Advanced pop-up option gives you access to less-frequently used features that let you fine-tune the appearance and behavior of the control or indicator:

- Use **Key Navigation...** to associate a keyboard key combination with a front panel object. When a user enters that key combination while a VI is running, LabVIEW acts as if the user had clicked on that object, and the object becomes the key focus (key focus means the cursor is active in that field).
- **Synchronous Display** is a selectable option that forces LabVIEW to refresh the display of this control or indicator on every update. This adds significant overhead, so you shouldn't use this option unless you have a good reason.
- **Customize...** will bring up the *Control Editor* to allow you to customize the graphical appearance of the control. We'll talk about creating your own custom controls in Chapter 15.
- **Hide Control/Indicator.** You can choose to hide a front panel object using this option, which comes in handy when you don't want the user to see the front panel object but still need it in the diagram. If you need to show the front panel object again, you must select Show Control/Indicator on the pop-up menu of the block diagram terminal.
- **Enabled State** allows you to set a control's state as **enabled**, **disabled**, or **disabled & grayed**. This comes in handy if you still want to show a control or indicator on the front panel, but you don't want the user to use it.



There are some other pop-up options that are specific to different types of controls (numeric, Boolean, etc.), but we'll leave them for later.

Don't worry about memorizing all of these features right now—you'll come across them as you work with LabVIEW and they'll make a lot more sense!



The same object will have a different pop-up menu in run mode than it will in edit mode. If you cannot find a certain pop-up option, it may not present for that object, you may need to switch modes, or you should pop up elsewhere on the object.

3.8 Help!

3.8.1 The Context Help Window

The LabVIEW *Context Help window* offers indispensable help information for functions, constants, subVIs, and controls and indicators. To display the window, choose **Show Context Help** from the **Help** menu or use the keyboard shortcut: <control-H> on Windows, <command-H> on the Mac, <meta-H> on the Sun, and <alt-H> on HP-UX and Linux. If your keyboard has a <help> key, you can press that instead. You can resize the Help window and move it anywhere on your screen to keep it out of the way.

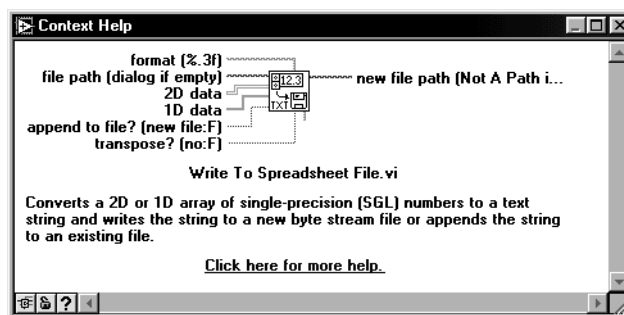


Figure 3.44
Help window.

When you hold the cursor over a function, a subVI node, or a VI icon (including the icon of the VI you have open, at the top-right corner of the VI window), the Help window shows the icon for the function or subVI with wires of the appropriate data type attached to each terminal. *Input wires point to the left, and output wires point to the right.* Terminal names appear beside each wire. If the VI has a description associated with it, this description is also displayed.



Lock Button

For some subVIs or functions, the Help window will show the names of required inputs in bold, with default values shown in parentheses. In some cases, the default value can be used and you do not need to wire an input at all. You can lock the Help window so that its contents do not change when you move the mouse by selecting **Lock Context Help** from the **Help** menu or by pressing the Lock button in the Help window.

If you position the Wiring tool over a specific node on a function or subVI, the Help window will flash the labeled corresponding node so that you can make sure you are wiring to the right place. Sometimes you may need to use the scrollbar to see all of the text in the Help window.

For VIs and functions with large numbers of inputs and outputs, the Help window can be overwhelming, so LabVIEW gives you the choice between simple or detailed views. You can use the simple view to emphasize the important connections and deemphasize less commonly used connections.

Simple/
Detailed Help
Button

Switch between views by pressing the Simple/Detailed Diagram Help button on the lower-left corner of the Help window. In simple help view, required connections appear in bold text; recommended connections appear in plain text; and optional connections are not shown. Wire stubs appear in the place of inputs and outputs that are not displayed, to inform you that additional connections exist (and you can see them in the detailed help view).

In detailed help view, required connections appear in bold text; recommended connections appear in plain text; and optional connections appear as disabled text.

If a function input does not need to be wired, the default value often appears in parentheses next to the input name. If the function can accept multiple data types, the Help window shows the most common type.

Online Help



LabVIEW's Help window provides a quick reference to functions, VIs, controls, and indicators. However, there are times when you'd prefer to look at a more detailed, indexed description for information on using a VI or function. LabVIEW has extensive online help that you can access by selecting **Contents and Index...** from the **Help** menu or by pressing the Online Help button in the Help window.

You can type in a key word to search for, view an extensive keyword index, or choose from a variety of topics to browse through. You can also set your own links to online help documents, which we'll talk about in Chapter 15.



Currently, not all LabVIEW VIs link to online help; if this is the case the online help menu item and online help button will be grayed out

3.9 A Word about SubVIs

If you want to take full advantage of LabVIEW's abilities, you must understand and use the hierarchical nature of the VI. A *subVI* is simply a stand-alone program that is used by another program. After you create a VI, you can use it as a subVI in the block diagram of a higher-level VI as long as you give it an icon and define its connector. A LabVIEW subVI is analogous to a subroutine in C or another text-based language. Just as there is no limit to the number of subroutines you can use in a C program, there is no limit to the number of subVIs you can use in a LabVIEW program (memory permitting, of course).

If a block diagram has a large number of icons, you can group them into a subVI to maintain the simplicity of the block diagram. You can also use one subVI to accomplish a function common to several different top-level VIs. This modular approach makes applications easy to debug, understand, and modify. We'll talk more about how to build subVIs later, but it's such an important part of the LabVIEW programming environment that we want you to keep it in mind as you're learning the basics.

3.10 Activity 3-2: Front Panel and Block Diagram Basics

In this activity, you will practice some simple exercises to get a feel for the LabVIEW environment. Try to do the following basic things on your own. If you have any trouble, glance back through the chapter for clues.

1. Open a new VI and toggle between the front panel and block diagram.



Use the keyboard shortcuts listed in the pull-down menus!

2. Resize the windows so that both front panel and block diagram are visible simultaneously. You may need to move them around.



Do this using the standard resizing technique for your platform. Or try the Tile function!

3. Drop a digital control, a string control, and a Boolean indicator on the front panel by selecting them from the Controls palette.

To get the digital control, click on the **Numeric** palette button in the **Controls** palette and select **Digital Control** from the subpalette that appears.



Figure 3.45

Now click your mouse on the front panel in the location where you want your digital control to appear. Voilà—there it is! Now create the string control and Boolean indicator in the same fashion.

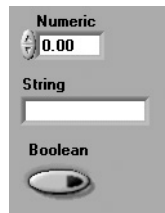


Figure 3.46

Notice how LabVIEW creates corresponding terminals on the block diagram when you create a front panel object. Also notice that floating-point numeric terminals are orange (integer numerics will be blue), strings are pink, and Booleans are green. This color-coding makes it easier for you to distinguish among data types.

4. Now pop up on the digital numeric control (by right-mouse-clicking on Windows and UNIX platforms or <command>-clicking on Mac) and select **Change to Indicator** from the pop-up menu. Notice how the appearance of the numeric's front panel changes (the little arrows go away). Also notice how the terminal on the block diagram changes (the border is much thinner for indicators). Switch the object back and forth between control and indicator until you can easily recognize the differences on both front panel and block diagram. Note that for some objects (like a few Booleans), front panel indicators and controls can look the same, but their block diagram terminals will always be different.
5. Choose the Positioning tool from the floating **Tools** palette, and then select an object on the front panel. Hit the <delete> key to remove it. Delete all front panel objects so that you have an empty front panel and block diagram.
6. Drop another digital control from the **Numeric** subpalette of the **Controls** palette onto the front panel. If you don't click on anything first, you should see a little box above the control. Type `Number 1`, and you will see this text appear in the box. Click the Enter button on the Toolbar to enter the text. You have just created a label. Now create another digital control labeled `Number 2`, a digital indicator labeled `N1+N2`, and a digital indicator labeled `N1-N2`.





Use the Operating tool to click on the increment arrow of Number 1 until it contains the value "4.00." Give Number 2 a value of "3.00."

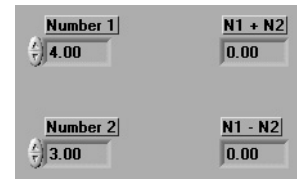


Figure 3.47



7. Switch back to the diagram. Drop an **Add** function from the **Nu-meric** subpalette of the **Functions** palette in the block diagram (this works just like creating front panel objects). Now repeat the process and drop a **Subtract** function.
8. Pop up on the **Add** function and select the **Visible Items>>Terminals** option (you'll notice that before you select it, the option is not checked, indicating that terminals are not currently shown). Once you show them, observe how the input and output terminals are arranged; then redisplay the standard icon by again selecting **Visible Items>>Terminals** (this time the option appears with a check mark next to it, indicating that terminals are currently shown).
9. Bring up the Help window by using either the keyboard shortcut or the **Show Context Help** command from the **Help** menu. Position the cursor over the **Add** function. The Help window provides valuable information about the function's use and wiring pattern. Now move the cursor over the **Subtract** function and watch the Help window change.
10. You may have to use the Positioning tool to reposition some of the terminals as shown in Figure 3.47. Then use the Wiring tool to wire the terminals together. First select it from the **Tools** palette; then click once on the DBL terminal and once on the appropriate terminal on the **Add** function to draw a wire. A solid orange line should appear. If you mess up and get a dashed black line instead of a solid orange one, select the wire fragment with the Positioning tool and hit the <delete> key; then try again. Click once and release to start the wire, click any time you want to add a new segment (which turns a corner), and click on a destination to finish the wire.

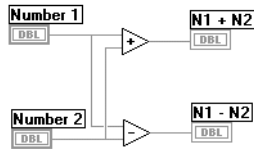


Figure 3.48

Notice that when you pass the Wiring tool over the **Add** and **Subtract** functions, little wire stubs appear showing where the terminals are located. In addition, as you pass the cursor over a terminal, its name appears in a tip strip. Like learning to type, wiring can be kind of tricky until you get the hang of it, so don't worry if it feels a little awkward right now.

11. Switch back to the front panel and pop up on the icon pane (the little window in the upper-right corner). Select **Show Connector** from the menu. Observe the connector that appears. If you can't get the pop-up menu to appear, you are probably trying to pop up in the icon pane of the block diagram.



Figure 3.49

Now pop up again on the connector and look at its menu to see the configuration options you have. The connector defines the input and output parameters of a VI so that you can use it as a subVI and pass data to it. You can choose different patterns for your connectors depending on how many parameters you need to pass. Show the icon again by selecting **Show Icon**. Remember, the icon is just the pictorial representation of a VI; when you use a VI as a subVI, you will wire to this icon in the block diagram of the top-level VI just like you wired to the **Add** function.



Run Button

12. Run the VI by clicking on the Run button. The N1+N2 indicator should display a value of "7.00" and N1-N2 should be "1.00." Feel free to change the input values and run it over and over.

13. Save the VI by selecting **Save** from the **File** menu. Call it **Add.vi** and place it in your `MYWORK` directory or VI library.

Congratulations! You have now mastered several important basic LabVIEW skills!

3.11 Wrap It Up!

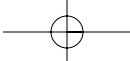
The LabVIEW environment has three main parts: the *front panel*, the *block diagram*, and the *icon/connector*. The front panel is the user interface of the program—you can input data through *controls* and observe output data through *indicators*. When you place an object on the front panel using the **Controls** palette, a corresponding terminal appears in the block diagram, making the front panel data available for use by the program. Wires carry data between *nodes*, which are LabVIEW program execution elements. A node will execute only when all input data are available to it, a principle called *dataflow*.

A VI should also have an *icon* and a *connector*. When you use a VI as a subVI, its icon represents it in the block diagram of the VI you use it in. Its connector, usually hidden under the icon, defines the input and output parameters of the subVI.

LabVIEW has two types of menus: pull-down and pop-up. *Pull-down* menus are located in the usual menu spot at the top of your window or screen, while *pop-up* menus can be accessed by “popping up” on an object. To pop up, right-mouse-click on Windows and UNIX machines and <command>-click on the Mac, or click with the Pop-up tool. Pull-down menus tend to have more universal commands, whereas pop-up menu commands affect only the object you pop up on. Remember, when in doubt about how to do something, pop up to see its menu options!

The **Tools** palette gives you access to the special operating modes of the mouse cursor. You use these tools to perform specific editing and operation functions, similar to how you would use them in a standard paint program. You will find front panel control and indicator graphics located in the **Controls** palette and block diagram constants, functions, and structures in the **Functions** palette. These palettes often have objects nestled several layers down in *subpalettes*, so make sure you don't give up your search for an object too soon.

The Help window provides priceless information about functions and how to wire them up; you can access it from the **Help** menu. LabVIEW also



contains extensive online help that you can call up from the **Help** menu or by pressing the online help button in the Help window. Between these two features, your questions should never go unanswered!

You can easily turn any VI into a subVI by creating its icon and connector and placing it in the block diagram of another VI. Completely stand-alone and modular, subVIs offer many advantages: They facilitate debugging, allow many VIs to call the same function without duplicating code, and offer an alternative to huge messy diagrams.

Don't worry if this seems like a lot to remember. It will all become natural to you as you work your way through the book.

