CHAPTER 3

Sub VI

3.1 Definition of Sub VI

In the regular text based programming languages such as FORTRAN, PASCAL, C, and others, the concept of subroutine is so essential that one may not be able to complete his or her code without using subroutines. Similarly, LabVIEW has a VI which corresponds to the subroutines and is called sub VI. As subroutines do, a LabVIEW sub VI can take parameters, perform a task, and return resulting values. Of course, sub VI can have no input or no output parameters. Since LabVIEW is a G-language (Graphical language) one should expect different ways of invoking the sub VIs. Since the VIs represent virtual instruments, you have wires for connection and objects for individual components. Therefore, you pass the parameters using wires, and call subroutines by including objects (sub VIs) in the diagram window. In other words, controlling the sub VIs is done *graphically*. In this chapter, you will see how to create sub VIs and control them using their settings.

3.2 Creating a Sub VI

Invoke LabVIEW if it is not running already. Create a new VI by pressing the button **New VI** in the startup window of LabVIEW or by choosing **New** under the **File** pull-down menu. Go to the front panel and place objects as shown in Figure 3.1 and save it as **First Sub VI.vi**. There are two digital controls and one digital indicator that enable this sub VI to calculate the square of the sum of two variables *x* and *y*. When you finish this sub VI, you will be able to call on it whenever $(x + y)^2$ needs to be calculated. You can find the digital control and the digital indicator in the **Numeric** sub-palette within the **Controls** palette. (See Figure 3.2.) Right clicking the mouse button in the front panel window gives access to these palettes.

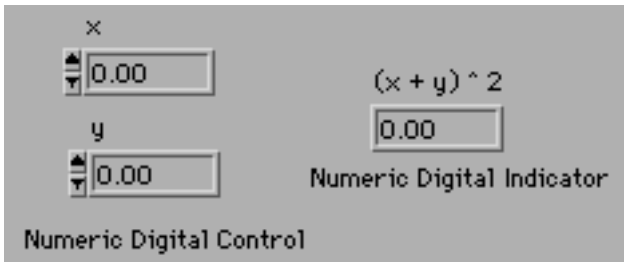


Figure 3.1 Front panel of **First Sub VI.vi**

To distinguish each object, make sure to label them as **x**, **y**, and **(x + y)^2**. You can display the labels if they are not shown by right clicking on the object and selecting **Show** >> **Label** from the pop-up menu.

> When in doubt, right click on the object of interest. This will display a pop-up menu from which you may choose an action. Online help is also available and can be accessed with the **Ctrl-H** shortcut while the cursor rests on the object of interest.

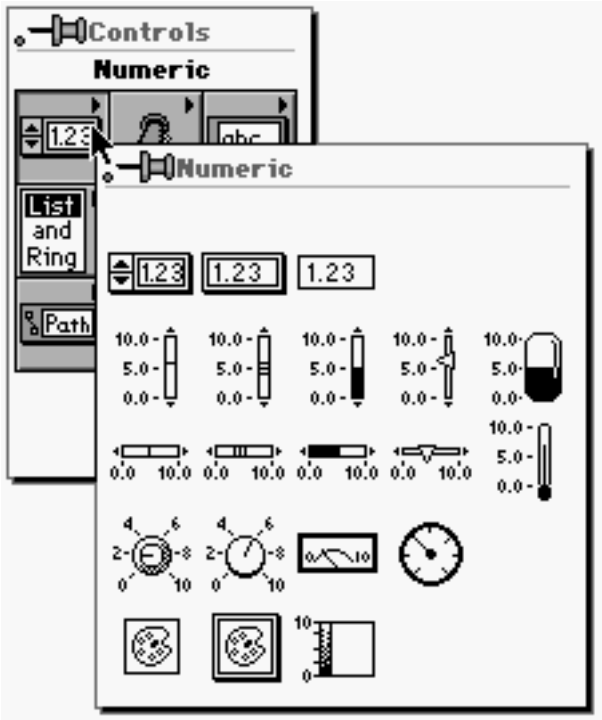


Figure 3.2 You can find the digital control and the digital indicator from **Controls** >> **Numeric**.

Now, go to the diagram window and complete it as shown in Figure 3.3. To do so, right click in the diagram window and access the **Functions** palette. Select **Numeric** to locate the **Add** feature; it appears as a triangularly shaped icon with a plus sign in the center. To find **Power Of X** that calculates b^c for given b and c, pop up the **Functions** palette and go to **Numeric** >> **Logarithmic**. Once you have placed these objects, press either the tab key or space bar to change the cursor to the Wire Tool. Now wire them as shown in Figure 3.3.

 **Connect Wire** (Wire Tool)

Since wiring technique is important, here are the steps again.

Step 1 Place the Wire Tool on top of **x** (digital control: how do you tell if it is a control or an indicator?). It will start blinking.

Step 2 While the icon blinks, click once; that is, press the left mouse button once and release it.

Step 3 Drag the Wire Tool onto the top input of **Add**. Note that you just move the mouse without holding the button.

Step 4 When the Wire Tool is on top of the first input of **Add**, it (the top input) will also start blinking. Note that the label of the first input **x** is also shown. While the icon is blinking, click on the left mouse button, and it will complete the wire between **x** and the top input of **Add**.

Step 5 To wire the rest, follow the same procedure.

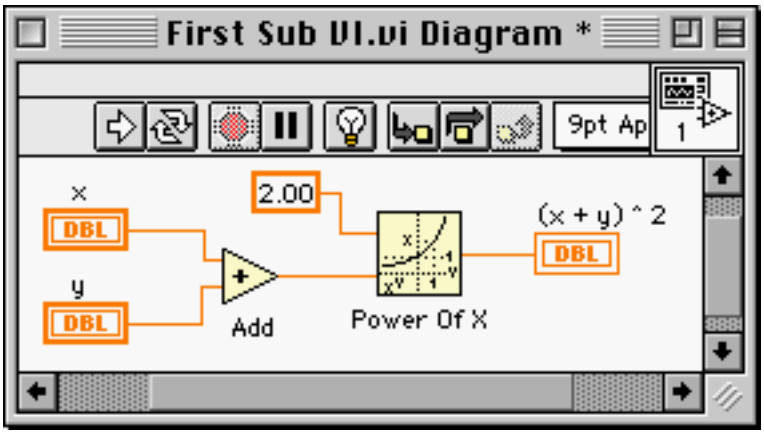


Figure 3.3 Diagram window of **First Sub VI.vi**

Now, you need to create terminals for the inputs and the output of this sub VI. Go to the front panel by using the shortcut **Ctrl-E** that jumps between the front panel and the diagram window. Locate the square icon in the upper right corner of the window, as shown in Figure 3.4. Right click on the icon and select **Show Connector**. This will turn the icon into a picture with three terminals as shown in Figure 3.4. Note that LabVIEW automatically knows how many terminals you need based on the number of controls and indicators in your VI. In the **First Sub VI.vi**, you have two controls **x** and **y**, and one indicator, **(x + y)^2**; therefore, two input terminals and one output terminal are chosen by LabVIEW. However, a different pattern of terminals can be manually chosen.

Figure 3.4 Select **Show Connector** from the pop-up menu to display the terminals.

To select a different terminal pattern, right click on the icon in the upper right corner of the front panel while the terminals are displayed. Go to **Patterns** and you will see all of the possible patterns of terminals. Ask yourself the following question about selecting terminal patterns: What happens if you need 50 input terminals and 50 output terminals? You immediately see a problem because no such pattern is available for selection. Furthermore, it will be difficult to identify each terminal so wiring would be challenging if not impossible. In order to avoid such a situation, you could use different data structure such as Array or Cluster (see Chapter 10), and bundle multiple terminals into a single data type.

> Try to minimize the number of terminals as much as possible. One way to do so is to use a data structure such as Array or Cluster that can have more than one element for a single object.

Keeping this in mind, finish your first sub VI example. When you select **Show Connector** from the pop-up menu after right clicking on the icon in the upper right corner of the front panel, LabVIEW automatically switches the mouse pointer to the Wire Tool. If not, you can select it from the **Tools** palette. After selecting the Wire Tool, click one terminal and click an object that you want to assign the terminal to. Continue to assign the terminals to the remaining controls and indicators. Note that you do not need an equal number of terminals and objects; therefore, it would be a good idea to have a few extra terminals available when you initially create them in case you need to modify subroutines (sub VIs). This will prevent you from having to reassign terminals and rewire other VIs to which the sub VIs were connected. Also, there is no requirement for input terminals to be placed on the left and output terminals on the right; however, this seems to be conventional for most LabVIEW programmers. When you finish assigning all of the terminals to each object, save the VI. By the way, you may have recognized the asterisk (*) next to the title of the VI in Figure 3.3. This indicates whether changes have been made since the last save. Therefore, it will disappear after saving the changes in the VI.

Now build another VI where you will call your first sub VI. Close **First Sub VI.vi** completely, create a new VI, and save it as **Main.vi**. (Saving changes on a regular basis, such as every 10 minutes, is a good programming habit.) Place objects as shown in Figure 3.5. Label each object as shown to distinguish between them easily. Both the **Knob** and the **Gauge** can be found under the **Controls** >> **Numeric** sub-palette. By default, the maximum value of the knob

is 10.0. Change the value to 1.0 by entering 1.0 over the 10.0. You will notice that the knob scale adjusts automatically. Explore the feature of the gauge for a moment. Notice that it can be rotated, resized, and the values can be changed. You can also show or hide the digital display from the pop-up menu of each item. When you have finished the front panel, switch to the diagram window by using **Ctrl-E**. You should have three objects in the diagram window: two controls and one indicator. (How do you tell if an object is a control or an indicator in the diagram window?) In text-based programming languages, you invoke subroutines by calling their names.

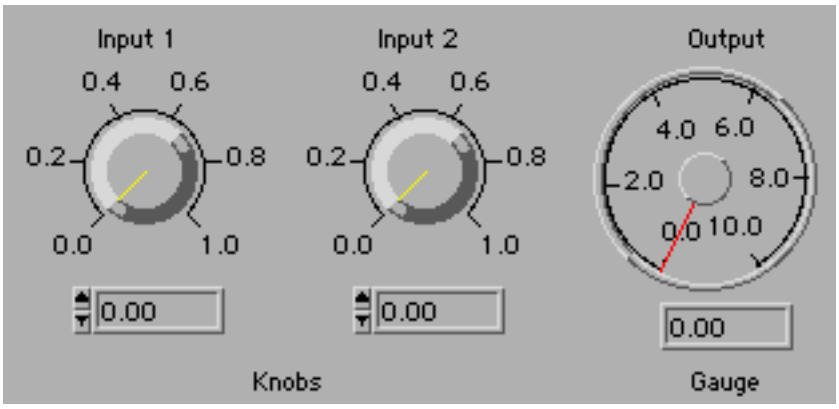


Figure 3.5 Front panel of **Main.vi**

Similarly, you call sub VIs in LabVIEW by placing it in the diagram window. To do so, bring up the **Functions** panel by right clicking in the diagram window and locate the **Select a VI …** sub-palette. Selecting it will bring up a dialog box asking for a VI to open. In the dialog window similar to Figure 3.6, locate and select **First Sub VI.vi** and place it in the diagram window. It has the default LabVIEW icon, but it can be modified with the **Edit Icon**. This option can be found in the pop-up menu of the icon in the upper right corner of **First Sub VI.vi**.

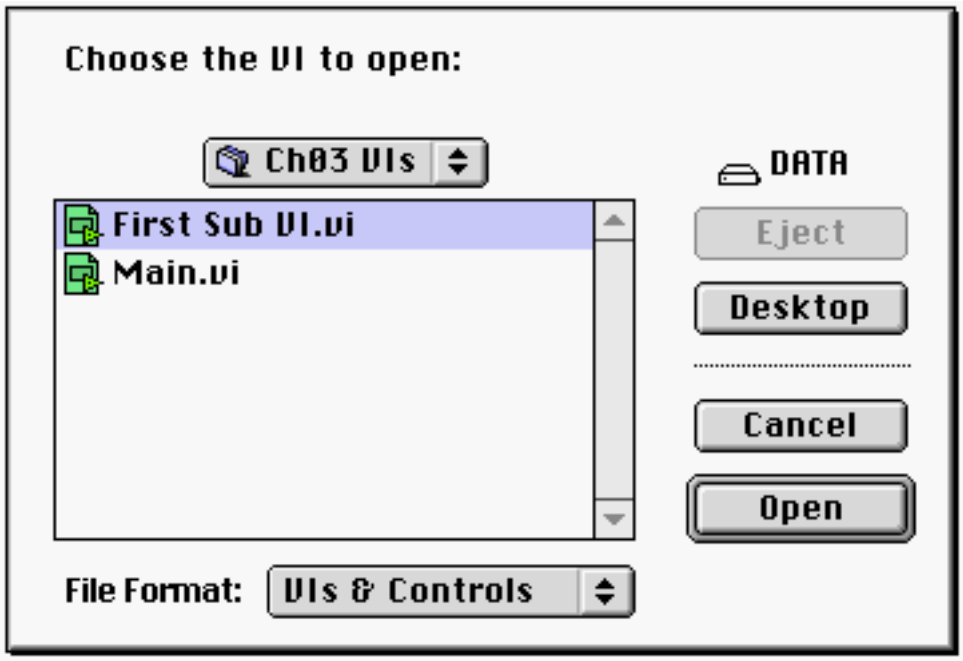


Figure 3.6 Dialog box to choose a VI (Macintosh platform)

Having put **First Sub VI.vi** in the diagram window of **Main.vi**, wire the two controls **Input 1** and **Input 2** to the input terminals (**x** and **y**) and the indicator **Output** to the output (**(x + y)^2**). (What is the shortcut to toggle between the Arrow Tool and the Wire Tool?) Right click on **First Sub VI.vi** to bring up its pop-up menu and select **Show** >> **Label** to display its name **First Sub VI.vi**. Note that when you place the Wire Tool on the first input terminal of the sub VI, LabVIEW displays **x** and similarly **y** on the second input terminal. In other words, LabVIEW displays the original labels that you assigned to each input and output terminals in your sub VI. This helps users identify terminals.

If you have followed the instructions correctly so far, the diagram window of **Main.vi** should look like Figure 3.7. Go to the front panel using the shortcut and select the Hand Tool either from the pop-up menu (shortcut: **Ctrl**-**Shift**-right click) or by tapping the tab key. Using the Hand Tool, set some values to the knobs (**Input 1** and **Input 2**) and run the VI by clicking on the **Run** button. Examine the VI further by playing with each object and different buttons. If you have a broken run arrow indicating that an error or errors have occurred, debug the VI using the tools such as **Highlight Execution**  , **Step Into**  , **Step Out**  , or **Probe Data**  . (See

Chapter 2 for their roles.)

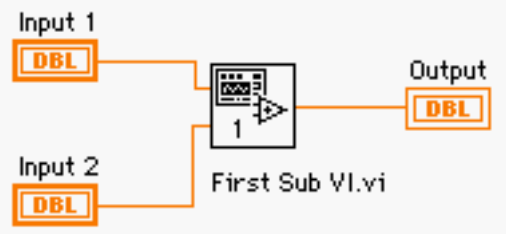


Figure 3.7 Diagram window of **Main.vi**

By completing the preceding steps, you have just learned how to create a sub VI. Sub VIs are important stepping stones in LabVIEW programming. You can now call (invoke) **First Sub VI.vi** when you need to compute the square of the sum of two variables in other VIs just by placing the sub VI in their diagram windows. If the values you have set for all of the input parameters in the sub VI are values you will always start with, select **Make Current Values Default** from the **Operate** menu and save the VI, or you can save an individual control's default values by popping up the menu on the control, selecting **Data Operations**, then selecting **Make Current Value Default**, and saving the VI.

3.3 Creating Online Help Text for a Sub VI

LabVIEW provides online help for almost all of the objects and the shortcut **Ctrl-H** will bring up the help text while pointing at objects of interest. You can also create an online help text for your own sub VIs that can be instantly accessed through the **Ctrl-H** shortcut. This can be illustrated by using the sub VI example you just created (**First Sub VI.vi**).

Open the example **First Sub VI.vi** by either double-clicking the icon in **Main.vi** or opening it manually. Select **Show VI Info…** from the **Windows** pull-down menu. This will bring up the window **VI Information** with a text box under **Description**. Then, the comments you enter in the **Description** will show up as the online help when you use **Ctrl-H** while the LabVIEW cursor is on **First Sub VI.vi**.

If you want to keep a history of modification of a VI, select **Show History** from the **Windows** pull-down menu. This will bring up a window revealing the VI history where you can

add a new list or reset the previous one. This is helpful when multiple users are working on the same VI and one user wants to see what other users have done.

3.4 Security Settings of a Sub VI

One of the new features of LabVIEW 5.0 is the password protection capability of VIs. If you select **Show VI Info…** from the **Windows** pull-down menu, it will bring up the window **VI Information** with the password setting window as shown in Figure 3.8. Using the four setting buttons, you can lock or unlock the VI, or set, clear, or change its passwords. Locking a VI will allow users to open the diagram window, but will not allow them to make any changes. In this mode, users only have limited debugging capability. (Debugging with light bulb  will be disabled.) This is useful if you want to prevent any accidental modification during the debugging process.

When you set a password to a VI, you won't be able to open the diagram window without the correct password. The only way to recover the diagram window without the correct password is to *rewrite* the VI; therefore, passwords must be kept safe and used with caution. To activate this security feature, save the VI with your new password and then close the VI. If you would like an immediate effect, select **Clear Password Cache** from the **Edit** pull-down menu. Save the VI, and close only the diagram window without closing the VI completely. Next time you try to open the diagram window using the shortcut **Ctrl-E**, you will be prompted to enter the password.

Figure 3.8 Password protection window of a VI

3.5 Option Settings of a Sub VI

Knowing how to create sub VIs in LabVIEW would lead one to expect graphical control over them since LabVIEW is a G-language. What would you expect to happen if one calls the *same* sub VI simultaneously with different input values? Can you somehow control the front panel of sub VIs such as its location on screen? This section on controlling sub VIs will answer these questions and more. However, before you proceed, you should recognize that you can bring up the front panel of sub VIs by double clicking on their icons. Also, if you want to change the settings of a particular sub VI, you *must* change them *in* that particular sub VI. This implies that you need to bring up its front panel and its diagram window to modify it and save changes therein. Local options will be discussed in the latter part of this chapter.

Go back to the **Main.vi** created in the previous section. Refer to the diagram window in Figure 3.7. In order to change the feature of **First Sub VI.vi**, you need to bring up its front panel by double clicking the icon. Proceed to do so to have its front panel on screen. If you need to change its diagram, you may use the shortcut **Ctrl-E** to go to the diagram window. For the moment, stay with its front panel. Right click on the icon in the upper right corner of the front panel, as shown in Figure 3.4, to bring up the pop-up menu. Selecting **VI Setup …** from the menu will bring up a window in Figure 3.9. If you click on the **Execution Options**, you will see the other two available options, but only the first two, **Execution Options** and **Window Options**, are discussed here. For those who are interested in the **Documentation** option, you may refer to the LabVIEW User Manual. Note that the menu type that contains **Execution Options**, **Window Options**, and **Documentation** is called *menu ring*.

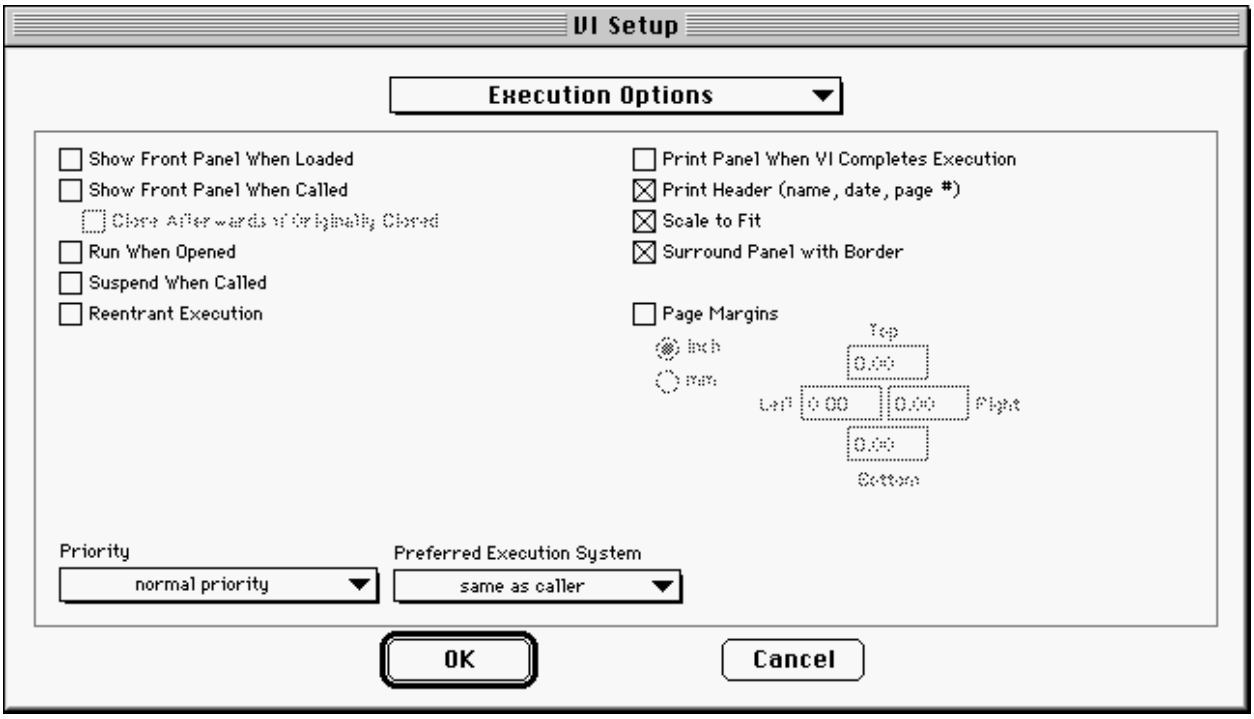


Figure 3.9 **Execution Options** in **VI Setup …** window

The most common feature that LabVIEW programmers use is the second and the third option in the left column in Figure 3.9: **Show Front Panel When Called** and **Close Afterwards if Originally Closed**. The option **Close Afterwards if Originally Closed** becomes available once you select **Show Front Panel When Called**. To understand their usage, consider an example. Suppose you are writing a database program for your customer using LabVIEW. In the front panel, your customer wants to display multiple buttons for tasks such as Enter Password, Update Address, Display Check-In and Check-Out Time, and so forth. Your task is to write a VI that brings up the address page when the user presses the Update Address button. It could easily be done by using a sub VI containing the address information if you select **Show Front Panel When Called** and **Close Afterwards if Originally Closed**. (In order to have a button waiting for an action in this case, you will need to use a loop, which will be discussed in Chapter 4.) You can adjust the location of the window when it shows up on screen using the **Window Options** that will be discussed in a moment. Another option you must be aware of whenever you create your own sub VI is the **Reentrant Execution** option.

In the beginning of this section, you asked what would happen if you simultaneously

called the same sub VI in multiple places in one VI? When you open a VI (main VI) which calls (contains) other VIs (sub VIs), LabVIEW will load the main VI as well as all of the sub VIs in the memory. If you call the same VI (sub VI) in different places intending to run these simultaneously, LabVIEW may return wrong answers if you are not using the **Reentrant Execution** option. In fact, each copy of the sub VI will execute sequentially in random order without the option **Reentrant Execution** selected. Therefore, if the functionality of a sub VI is related to a timed simultaneous operation, it may bring an unexpected result. Also, if the sub VI has memory, the data from the previous call will be reused in the next call! (See Problem 4.2 and Problem 4.3 for an example about the effect of **Reentrant Execution**.) To avoid such an incident, you must select **Reentrant Execution** so that LabVIEW will allocate separate memory blocks and treat each copy of the same VI as different ones. Having the option **Reentrant Execution** checked, identical copies of a sub VI will run independently of one another. However, the remaining choices in the **Execution Options** and the debugging buttons of a sub VI will become unavailable with the **Reentrant Execution** option selected.

Another option is **Window Options**, and its functionality is self-explanatory. With it, you may hide the toolbar of a VI, move or resize the windows to fit the entire screen, or perform many other tasks related to the windows of VIs. One thing to remember is that these options will take effect when you *execute* the VI. If a VI is a sub VI, then these options will apply when it gets *called* by the caller VI. **Window Options** is shown in Figure 3.10. Lastly, the option **Dialog Box** forces the front panel of the sub VI to be active on the top of the caller VI. This feature is useful if you want users not to lose the front panel of the sub VI. It can happen if the users accidentally activate the caller window by clicking in it during the VI execution, then the front panel of the sub VI will be hidden behind that of the caller VI. In order to avoid such an inconvenience – you have to move the front panel of the Caller VI to retrieve the window of the sub VI – you can use the option **Dialog Box** selected in the sub VI.

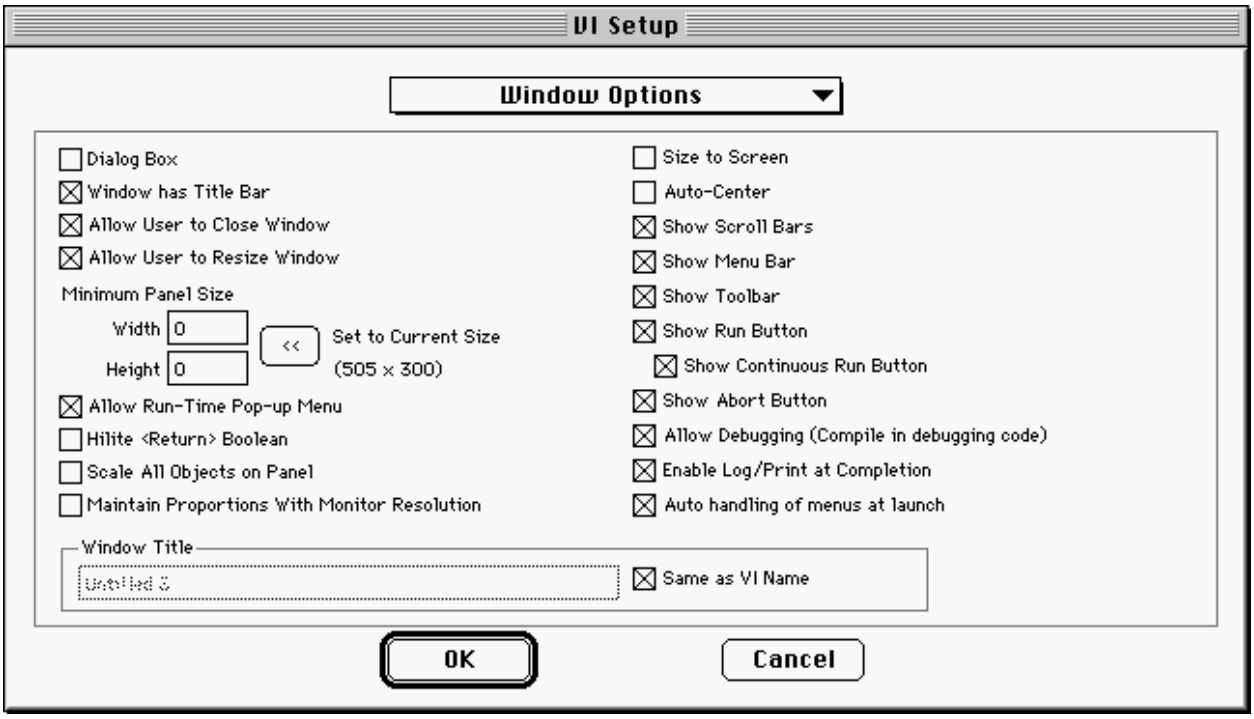


Figure 3.10 **Window Options** in **VI Setup …** window

3.6 SubVI Node Setup

The settings you make on a sub VI in its **VI Setup …** are global. In other words, if you pop up the front panel of a sub VI, bring up **VI Setup …,** and change any setting, it will apply whenever you call that sub VI. However, you may want to apply some settings to only a few of the copies of the same sub VI. You can do this by using **SubVI Node Setup**. Locate the sub VI to which you want to apply local settings. Right click on that sub VI to bring up the pop-up menu and select **SubVI Node Setup**; this will bring up a window as shown in Figure 3.11. You may realize that the settings in this window are four of the options available in the **Execution Option** in **VI Setup** that was shown in Figure 3.9. Therefore, you can only have a small set of options from **Execution Options** for sub VI local settings. These settings will not affect the other copies of the sub VI, but will apply only to the one that has the settings invoked.

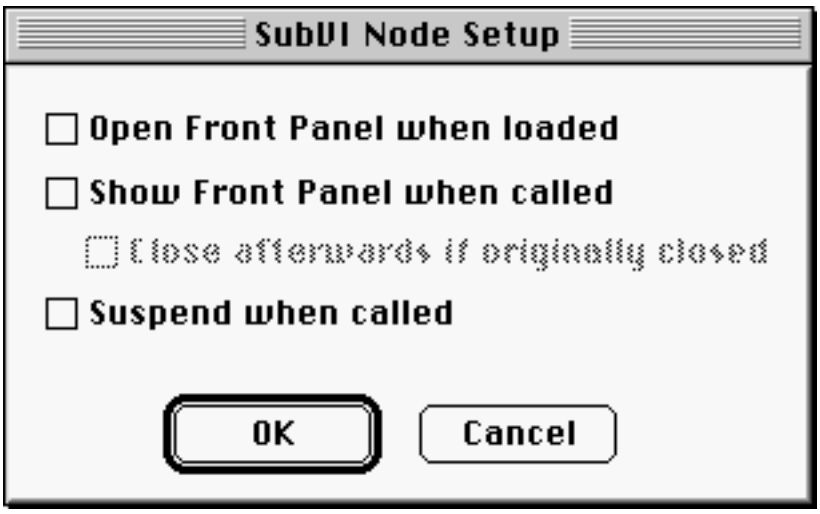


Figure 3.11 **SubVI Node Setup** window

3.7 Alternative Way of Creating a Sub VI

There is an alternative way to create sub VIs, and it can be illustrated with the **First Sub VI.vi** which computes **(x+y)^2** for given **x** and **y**. Suppose you have a VI whose diagram is Figure 3.12.

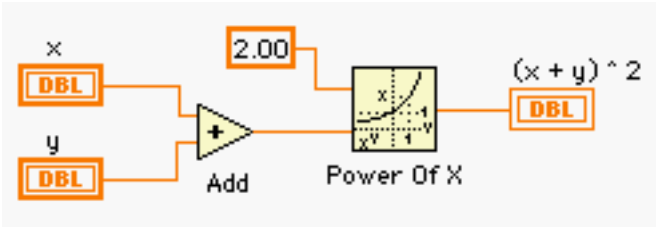


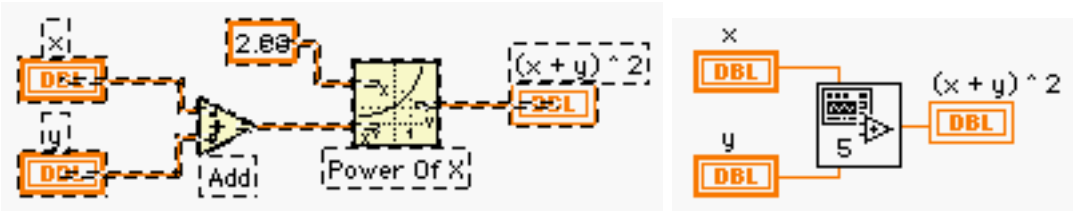
Figure 3.12 Diagram window of a VI which performs the same computation as **First Sub VI.vi**.

Note that Figure 3.12 performs the same computation as **First Sub VI.vi**. In the previous section, you manually assigned each element to each terminal to complete **First Sub VI.vi**. However, LabVIEW provides an alternative way to create sub VIs instantly. The steps are:

Step 1 Select the portion from which you would like to create a sub VI. In Figure 3.12, you will select the entire diagram.

Step 2 Select **Edit** >> **Create SubVI**. This will create a sub VI

with three terminals **x**, **y**, and **(x + y) ^ 2**.



(a) (b)

Figure 3.13 (a) A portion from which a sub VI is needed is selected in the diagram window. (b) Selecting **Edit** >> **Create SubVI** will create a VI from the portion selected.

Each step is shown in Figure 3.13. This method is very convenient since you do not have to assign each terminal to the objects in your VI; however, this can cause some confusion if the structure of input and output becomes complicated. Therefore, this method should not be abused in creating sub VIs unless the sub VI is as simple as the example in Figure 3.13.

> In order to create sub VIs, always try to use the method described in section 3.2, and minimize the use the method using **Edit** >> **Create Sub VI** unless the structure of input and output is extremely simple.

PROBLEMS

3.1 In the following equation, identify control(s) and indicator(s). Then, create a sub VI **P03_01.vi** that performs the equation, assuming that x is guaranteed to be nonzero.

$$y = 20 \log_{10} (|x|)$$

3.2 You want to expand the sub VI you created in Problem 3.1 so it can now return the result of the following equation as well as y:

$$z = 3 \cos(x)$$

Save the modified sub VI as **P03_02.vi**. If you created a few extra terminals in Problem 3.1, expanding **P03_01.vi** could be trivial. Therefore, it is a good idea to have a few extra terminals available when you create sub VIs for the first time.

3.3 Create a sub VI **P03_03.vi** that shows its front panel when called to display a message HELLO in the front panel for 3 seconds. Then, the front panel disappears afterwards. Test the sub VI to confirm that it behaves as described above by calling **P03_03.vi** from another VI. Hint: You may use the following LabVIEW functions and options:

Controls >> **String & Table** >> **Simple String**
Functions >> **Time & Dialog** >> **Wait (ms)**
pull-down menu **Operate** >> **Make Current Values Default**
VI Setup … >> **Execution Options** >> **Show Front Panel When Called**, and **Close Afterwards if Originally Closed**

3.4 Modify the setting(s) of **P03_03.vi** so that the front panel shows up in the center of the screen when it is called and disappears afterwards. Save the VI as **P03_04.vi**.

3.5 Create a sub VI **P03_05.vi** that takes an angle in degree as input, converts it in radian, and returns the sine and cosine of the angle in radian as output. You may use the following LabVIEW functions:

Functions >> **Numeric** >> **Additional Numeric Constants** >> **Pi**
Functions >> **Numeric** >> **Trigonometric** >> **Sine**, and **Cosine**