You can select multiple controls, mix selections with drawing objects, and perform operations such as move, cut, copy, paste, delete, and align. These operations are undoable. You can't group buttons or use Send to Back as you can with drawing objects.

In panels, when you do a Select All, the selectin includes all controls and drawing objects, but in the case of graphs, only drawing objects are selected. This is because drawing objects in graphs are used for presentation graphics whereas in panels they are used to construct the user interface.

If you want to copy controls from one window to another, simply use the Edit menu to copy and paste. You can also duplicate controls using copy and paste.

When you copy controls to the clipboard, the command and control names are also copied as text. This is handy for programming.

Press Option (*Macintosh*) or Alt (*Windows*) while choosing Edit-Copy to copy the complete commands for creating the copied controls.

If you copy a control from the extreme right side or bottom of a window, it may not be visible when you paste it into a smaller window. Use the smaller window's Retrieve submenu in the Mover tool palette icon to make it visible.

General Command Syntax

All of the control commands use the following general syntax:

```
ControlOperation Name [,keyword[=value] [,keyword[=value]]…]
```

Name is the control's name. It must be unique among controls in the window containing the control. If *Name* is not already in use then a new control is created. If a control with the same name already exists then that control is modified, so that multiple commands using the same name result in only one control. This is useful for creating controls that require many keywords.

All keywords are optional. Not all controls accept all keywords, and some controls accept a keyword but do not actually use the value. The value for a keyword with one control can have a different form from the value for the same keyword used with a different type of control. See the specific control operation documentation in Chapter V-1, **Igor Reference** for details.

Some controls utilize a format keyword to set a format string. The format string can be any printf-style format that expects a single numeric value. Think of the output as being the result of the following command:

```
Printf formatString, value_being_displayed
```

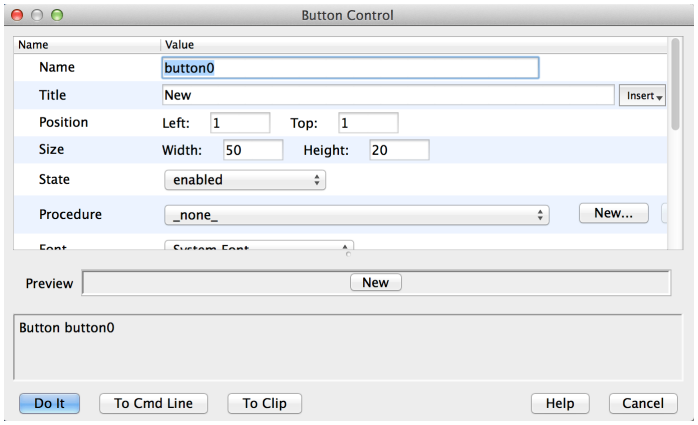
See the **printf** operation on page V-770 for a discussion of printf format strings. The maximum length of the format string is 63 bytes. The format is used only for controls that display numeric values.

All of the clickable controls can optionally call a user-defined function when the user releases the mouse button. We use the term *action procedure* for such a function. Each control passes one or more parameters to the action procedure. The dialogs for each control can create a blank user function with the correct parameters.

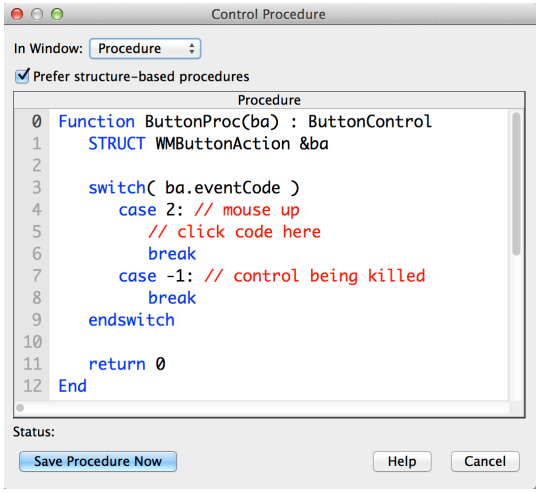
Creating Button Controls

The **Button** operation (page V-55) creates or modifies a rounded-edge or custom button with the title text centered in the button. The default font depends on the operating system, but you can change the font, font size, text color and use annotation-like escape codes (see **Annotation Escape Codes** on page III-53). The amount of text does not change the button size, which you can set to what you want.

Here we create a simple button that will just emit a beep when pressed. Start by choosing the Add Button menu item in the Graph or Panel menu to invoke the Button Control dialog:



Clicking the procedure's New button brings up a dialog containing a procedure template that you can edit, rename, and save. Here we started with the standard ButtonControl template, replaced the default name with `MyBeepProc`, and added the `Beep` command:



The controls can work with procedures using two formats: the old procedure format used in Igor 3 and 4, and the "structure-based" format introduced in Igor 5.

Selecting the "Prefer structure-based procedures" checkbox creates new procedure templates using the structure-based format.

The "Prefer structure-based procedures" checkbox is checked by default because structure-based procedures are recommended for all new code. If you uncheck this checkbox before editing the template, Igor switches the template to the old procedure format. Use of the old format is discouraged.

Click Help to get help about the Button operation. In the Details section of the help, you can find information about the WMButtonAction structure.

The fact that you can create the action procedure for a control in a dialog may lead you to believe that the procedure is stored with the button. This is not true. The procedure is actually stored in a procedure window. This way you can use the same action procedure for several controls. The parameters which are passed to a given procedure can be used to differentiate the individual controls.

For more information on using action procedures, see **Control Structures** on page III-437, the **Button** operation (page V-55), and **Using Structures with Windows and Controls** on page IV-103.

Button Example

Here is how to make a button whose title alternates between Start and Stop.

Enter the following in the procedure window:

```
Function MyStartProc()
   Print "Start"
End

Function MyStopProc()
   Print "Stop"
End

Function StartStopButton(ba) : ButtonControl
   STRUCT WMButtonAction &ba

   switch(ba.eventCode)
      case 2:                          // Mouse up
         if (CmpStr(ba.ctrlName,"bStart") == 0)
            Button $ba.ctrlName,title="Stop",rename=bStop
            MyStartProc()
         else
            Button $ba.ctrlName,title="Start",rename=bStart
            MyStopProc()
         endif
         break
   endswitch

   return 0
End
```

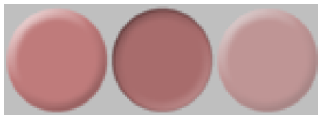
Now execute:

```
NewPanel
Button bStart,size={50,20},proc=StartStopButton,title="Start"
```

Custom Button Control Example

You can create custom buttons by following these steps:

1. Using a graphics-editing program, create a picture that shows the button in its normal (“relaxed”) state, then in the pressed-in state, and then in the disabled state. Each portion of the picture should be the same size:



If the button blends into the background it will look better if the buttons are created on the background you will use in the panel. Igor looks at the pixels in the upper left corner, and if they are a light neutral color, Igor omits those pixels when the button is drawn.

2. Copy the picture to the clipboard.
3. Switch to Igor,choose Misc→Pictures, click Load and then From Clipboard.