Because of click-time reevaluation, the pop-up menu does not automatically update if the value of the string expression changes. Normally this is not a problem, but you can use the **ControlUpdate** operation (page V-94) to force the pop-up menu to update. Here is an example:

```
NewPanel/N=PanelX
String/G gPopupList="First;Second;Third"
PopupMenu oneOfThree value=gPopupList  // pop-up shows "First"
gPopupList="1;2;3"                     // pop-up is unchanged
ControlUpdate/W=PanelX oneOfThree      // pop-up shows "1"
```

If the string expression can not be evaluated at the time the command is compiled, you can defer the evaluation of the expression by enclosing the value this way:

In some cases, the string expression can not be compiled at the time the PopupMenu command is compiled because it references a global object that does not yet exist. In this case, you can prevent a compile-time error by using this special syntax:

```
PopupMenu name value= #"pathToNonExistentGlobalString"
```

If a deferred expression has quotes in it, they need to be escaped with backslashes:

```
PopupMenu name value= #"\"_none_;\"+UserFunc(\"foo\")"
```

The optional user defined action procedure is called after the user makes a selection from the popup menu. Popup menu procedures have the following form:

```
Function PopMenuProc(pa) : PopupMenuControl
   STRUCT WMPopupAction pa

   switch(pa.eventCode)
      case 2:        // Mouse up
         Variable popNum = pa.popNum      // 1-based item number
         String popStr = pa.popStr        // Text of selected item
         break
      case -1:       // Control being killed
         break
   endswitch
End
```

`pa.popNum` is the item number, *starting from one*, and pa.`popStr` is the text of the selected item.

For the color pop-up menus the easiest way to determine the selected color is to use the **ControlInfo**.

Creating SetVariable Controls

The **SetVariable** operation (page V-854) creates or modifies a SetVariable control. SetVariable controls are useful for both viewing and setting values.

SetVariable controls are tied to numeric or string global variables, to a single element of a wave, or to an internal value stored in the control itself. To minimize clutter, you should use internal values in most cases.

When used with numeric variables, Igor draws up and down arrows that the user can use to increment or decrement the value.

You can set the width of the control but the height is determined from the font and font size. The width of the readout area is the width of the control less the width of the title and up/down arrows. However, you can use the bodyWidth keyword to specify a fixed width for the body (nontitle) portion of the control.

For example, executing the commands:

```
Variable/G globalVar=99
SetVariable setvar0 size={120,20},frame=1,font="Helvetica", value=globalVar
```

creates the following SetVariable control: