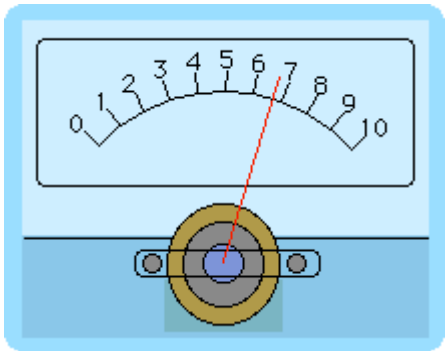
You can create even more sophisticated controls, such as this voltage meter control.



Choose File→Example Experiments→Feature Demos 2→Custom Control Demo to try this control and see the code that implements it.

Creating GroupBox Controls

The GroupBox operation creates or modifies a listbox control. See the **GroupBox** operation on page V-334 for a complete description and examples.

Creating ListBox Controls

The **ListBox** operation creates or modifies a listbox control.

We will illustrate listbox creation by example.

1. Create a panel with a listbox control:

   ```
   NewPanel
   ListBox list0 size={200,60}, mode=1
   ```

   The simplest functional listbox needs at least one text wave to contain the list items. Without the text wave, a listbox control has no list items. In this state, the listbox is drawn with a red X over the control.

2. We need a text wave to contain the list items:

   ```
   Make/O/T textWave0 = {"first list item", "second list item", "etc..."}
   ```

3. Choose Panel→Show Tools.

   This puts the panel in edit mode so you can modify controls.

4. Double-click the listbox control to invoke the ListBox Control dialog.
5. For the List Text Wave property, select the wave you created to assign it as the list's text wave.
6. Click Do It.

   You now have a functional listbox control.

7. Click the operate (top) icon, or choose Panel→Hide Tools, so you can use, rather than edit, the list.

In this example, we created a single-selection list. You can query the selection by calling **ControlInfo** and checking the V_Value output variable.

See the **ListBox** for a complete description and further examples.

Right-clicking (*Windows*) or Control-clicking (*Macintosh*) a listbox shows a contextual menu with commands for editing the list waves and action procedure, and for creating a numeric selection wave, if the control is a multi-selection listbox.

Creating PopupMenu Controls

The **PopupMenu** creates or modifies a pop-up menu control. Pop-up menus are usually used to provide a choice of text items but can also present colors, line styles, patterns, and markers.

The control automatically sizes itself as a function of the title or the currently selected menu item. You can specify the bodyWidth keyword to force the body (non-title portion) of the pop-up menu to be a fixed size. You might do this to get a set of pop-up menus of nicely aligned with equal width. The bodywidth keyword also affects the non-text pop-up menus.

The font and fsize keywords affect only the title of a pop-up menu. The pop-up menu itself uses standard system fonts.

Unlike color, line style, pattern, or marker pop-up menus, text pop-up menu controls can operate in two distinct modes as set by the mode keyword's value.

If the argument to the mode keyword is nonzero then it is considered to be the number of the menu item to be the initial current item *and* displays the current item in the pop-up menu box. This is the *selector mode*. There is often no need for an action procedure since the value of the current item can be read at any time using the **ControlInfo** operation (page V-89).

If mode is zero then the title appears inside the pop-up menu box, hence the name *title-in-box mode*. This mode is generally used to select a command for the action procedure to execute. The current item has no meaning except when the pop-up menu is activated and the selected item is passed to the action procedure.

The menu that pops up when the control is clicked is determined by a string expression that you pass as the argument to the value keyword. For example:

```
PopupMenu name value="Item 1;Item 2;Item 3;"
```

To create the color, line style, pattern or marker pop-up menus, set the string expression to one of these fixed values:

```
"*COLORPOP*"
"*LINESTYLEPOP*"
"*MARKERPOP*"
"*PATTERNPOP*"
```

For text pop-up menus, the string expression must evaluate to a list of items separated by semicolons. This can be a fixed literal string or a dynamically-calculated string. For example:

```
PopupMenu name value="Item 1;Item 2;Item 3;"
PopupMenu name value="_none_;" + WaveList("*",";","")
```

It is possible to apply certain special effects to the menu items, such as disabling an item or marking an item with a check. See **Special Characters in Menu Item Strings** on page IV-133 for details.

The literal text of the string expression is stored with the control rather than the results of the evaluation of the expression. Igor evaluates the expression when the PopupMenu value=<value> command runs and reevaluates it every time the user clicks on the pop-up menu box. This reevaluation ensures that dynamic menus, such as created by the WaveList example above, reflect the conditions at click time rather than the conditions that were in effect when the PopupMenu control was created.

When the user clicks and Igor reevaluates the string expression, the procedure that created the pop-up menu is no longer running. Consequently, its local variables no longer exist, so the string expression can not reference them. To incorporate the value of local variables in the value expression use the **Execute** operation:

```
String str = <code that generates item list>    // str is a local variable
Execute "PopupMenu name value=" + str
```

Igor evaluates the string expression as if it were typed on the command line. You can not know what the current data folder will be when the user clicks the pop-up menu. Consequently, if you want to refer to objects in specific data folders, you must use full paths. For example:

```
PopupMenu name value=#"func(root:DF234:wave0, root:gVar)"
```