Multiple Menu Items

A menu item string that contains a semicolon-separated “string list” (see **StringFromList** on page V-998) generates a menu item for each item in the list. For example:

```
Menu "Multi-Menu"
   "first item;second item;", DoItem()
End
```

Multi-Menu is a two-item menu. When either item is selected the DoItem procedure is called.

This begs the question: How does the DoItem procedure know which item was selected? The answer is that DoItem must call the **GetLastUserMenuInfo** operation (see page V-306) and examine the appropriate returned variables, usually V_value (the selected item’s one-based item number) or S_Value (the selected item’s text).

The string list can be dynamic, too. The above "Waves" example can be rewritten to handle an arbitrary number of waves using this definition:

```
Menu "Waves", dynamic
   WaveList("*",";",""), DoItem()
End

Function DoItem()
   GetLastUserMenuInfo           // Sets S_value, V_value, etc.
   WAVE/Z w= $S_value
   if( WaveExists(w) )
      Print "The wave's name is "+NameOfWave(w)
   endif
End
```

Consolidating Menu Items Into a Submenu

It is common to have many utility procedure files open at the same time. Each procedure file could add menu items which would clutter Igor’s menus. When you create a utility procedure file that adds multiple menu items, it is usually a good idea to consolidate all of the menu items into one submenu. Here is an example.

Let’s say we have a procedure file full of utilities for doing frequency-domain data analysis and that it contains the following:

```
Function ParzenDialog()
   …
End

Function WelchDialog()
   …
End

Function KaiserDialog()
   …
End
```

We can consolidate all of the menu items into a single submenu in the Analysis menu:

```
Menu "Analysis"
   Submenu "Windows"
      "Parzen…", ParzenDialog()
      "Welch…", WelchDialog()
      "Kaiser…", KaiserDialog()
   End
End
```