In this example, the text for the menu item is computed by the MacrosMenuItem function. It computes text for item 1 and for item 2 of the menu. Item 1 can be enabled or disabled. Item 2 can be checked or unchecked.

The dynamic keyword specifies that the menu definition contains a string expression that needs to be reevaluated each time the menu item is drawn. This rebuilds the user-defined menu each time the user clicks in the menu bar. Under the current implementation, it rebuilds *all* user menus each time the user clicks in the menu bar if *any* user-defined menu is declared dynamic. If you use a large number of user-defined items, the time to rebuild the menu items may be noticeable.

There is another technique for making menu items change. You define a menu item using a string expression rather than a literal string but you do not declare the menu dynamic. Instead, you call the BuildMenu operation whenever you need the menu item to be rebuilt. Here is an example:

```
Function ToggleItem1()
   String item1Str = StrVarOrDefault("root:MacrosItem1Str","On")
   if (CmpStr(item1Str,"On") == 0)     // Item is now "On"?
      String/G root:MacrosItem1Str = "Off"
   else
      String/G root:MacrosItem1Str = "On"
   endif
   BuildMenu "Macros"
End

Menu "Macros"
   StrVarOrDefault("root:MacrosItem1Str","On"), /Q, ToggleItem1()
End
```

Here, the menu item is controlled by the global string variable MacrosItem1Str. When the user chooses the menu item, the ToggleItem1 function runs. This function changes the MacrosItem1Str string and then calls BuildMenu, which rebuilds the user-defined menu the next time the user clicks in the menu bar. Under the current implementation, it rebuilds *all* user-defined menus if BuildMenu is called for *any* user-defined menu.

Optional Menu Items

A dynamic user-defined menu item *disappears* from the menu if the menu item string expression evaluates to ""; the remainder of the menu definition line is then ignored. This makes possible a variable number of items in a user-defined menu list. This example adds a menu listing the names of up to 8 waves in the current data folder. If the current data folder contains less than 8 waves, then only those that exist are shown in the menu:

```
Menu "Waves", dynamic
   WaveName("",0,4), DoSomething($WaveName("",0,4))
   WaveName("",1,4), DoSomething($WaveName("",1,4))
   WaveName("",2,4), DoSomething($WaveName("",2,4))
   WaveName("",3,4), DoSomething($WaveName("",3,4))
   WaveName("",4,4), DoSomething($WaveName("",4,4))
   WaveName("",5,4), DoSomething($WaveName("",5,4))
   WaveName("",6,4), DoSomething($WaveName("",6,4))
   WaveName("",7,4), DoSomething($WaveName("",7,4))
End

Function DoSomething(w)
   Wave/Z w

   if( WaveExists(w) )
      Print "DoSomething: wave's name is "+NameOfWave(w)
   endif
End
```

This works because WaveName returns "" if the indexed wave doesn't exist.

Note that each potential item must have a menu definition line that either appears or disappears.