Here is a working example:

```
#pragma IndependentModule = IndependentModuleA

Function ButtonProc(ba) : ButtonControl   // Must not be static
   STRUCT WMButtonAction &ba

   switch (ba.eventCode)
      case 2: // mouse up
         Print "Running IndependentModuleA#ButtonProc"
         break
   endswitch

   return 0
End

Function CreatePanel()
   NewPanel /W=(375,148,677,228)
   Button button0,pos={106,23},size={98,20},title="Click Me"
   Button button0,proc=ButtonProc
End
```

Independent Modules and User-Defined Menus

Independent modules can contain user-defined menus. When you choose a user-defined menu item, Igor determines if the menu item was defined in an independent module. If so, and if the menu item's execution text starts with a call to a function defined in the independent module, then Igor prepends the independent module name before executing the text. This means that the second and third menu items in the following example both call IndependentModuleA#DoAnalysis:

```
#pragma IndependentModule = IndependentModuleA

Menu "Macros"
   "Load Data File/1", Beep; LoadWave/G
   "Do Analysis/2", DoAnalysis() // Igor automatically prepends IndependentModuleA#
   "Do Analysis/3", IndependentModuleA#DoAnalysis()
End

Function DoAnalysis()
   Print "DoAnalysis in IndependentModuleA"
End
```

This behavior on Igor's part makes it possible to #include a procedure file that creates menu items into an independent module and have the menu items work. However, in many cases you will not want a #included file's menu items to appear. You can suppress them using `menus=0` option in the #include statement. See **Turning the Included File's Menus Off** on page IV-167.

Note: If a procedure file with menu definitions is included into multiple independent modules, the menus are repeatedly defined (see **Independent Modules and #include** on page IV-240). Judicious use of the menus=0 option in the #include statements helps prevent this. See **Turning the Included File's Menus Off** on page IV-167.

When the execution text doesn't start with a user-defined function name, as for the first menu item in this example, Igor executes the text without altering it.

Independent Modules and Popup Menus

In an independent module, implementing a popup menu whose items are determined by a function call at click time requires special care. For example, outside of an independent module, this works:

```
Function/S MyPopMenuList()
   return "Item 1;Item2;"
```