```
End

Function Slider1Proc(sa) : SliderControl  // Action procedure for slider1
   STRUCT WMSliderAction &sa

   switch(sa.eventCode)
      case -3:       // Control received keyboard focus
      case -2:       // Control lost keyboard focus
      case -1:       // Control being killed
         break
      default:
         if (sa.eventCode & 1)    // Value set
            Printf "Value = %g, event code = %d\r", sa.curval, sa.eventCode
         endif
         if (sa.eventCode & 8)    // Mouse moved or arrow key moved the slider
            Printf "Value = %g, event code = %d\r", sa.curval, sa.eventCode
         endif
         break
   endswitch

   return 0
End
```

Creating TabControl Controls

The **TabControl** creates or modifies a TabControl control. Tabs are used to group controls into visible and hidden groups.

The tabs are numbered. The first tab is tab 0, the second is tab 1, etc.

A default tab control has one tab:

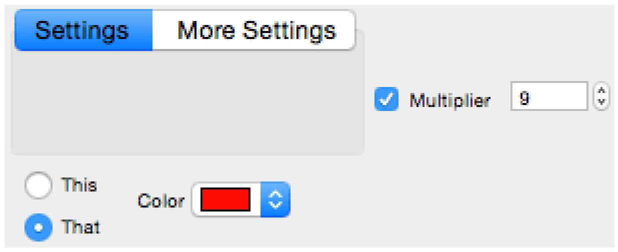
```
NewPanel/W=(150,50,650,400)
TabControl tb, tabLabel(0)="Settings", size={400,250}
```

You add tabs to the control by providing additional tab labels:

```
TabControl tb, tabLabel(1)="More Settings"
```

When you click on a tab, the control's action procedure receives the number of the clicked-on tab.

The showing and hiding of the controls are accomplished by your action procedure. In this example, the This, That, and Color controls are shown when the Settings tab is clicked, and the Multiplier checkbox is hidden. When the More Settings tab is clicked, the action procedure makes the opposite occur.



The simplest way to create a tabbed user interface is to create an over-sized panel with all the controls visible and outside of the tab control. Place controls in their approximate positions relative to one another. By positioning the controls this way you can more easily modify each control until you are satisfied with them.

Before you put the controls into the tab control, get a list of the non-tab control names:

```
Print ControlNameList("" ,"\r", "!tb")       // all but "tb"
thisCheck
thatCheck
colorPop
multCheck
multVar
```

Determine which controls are to be visible in which tabs:

Tab 0: Settings	Tab 1: More Settings
thisCheck	multCheck
thatCheck	multVar
colorPop	

Write the action procedure for the tab control to show and hide the controls:

```
Function TabProc(tca) : TabControl

   STRUCT WMTabControlAction &tca

   switch (tca.eventCode)
      case 2:                          // Mouse up
         Variable tabNum = tca.tab     // Active tab number
         Variable isTab0 = tabNum==0
         Variable isTab1 = tabNum==1

         ModifyControl thisCheck disable=!isTab0      // Hide if not Tab 0

         ModifyControl thatCheck disable=!isTab0      // Hide if not Tab 0
         ModifyControl colorPop disable=!isTab0       // Hide if not Tab 0

         ModifyControl multCheck disable=!isTab1      // Hide if not Tab 1
         ModifyControl multVar disable=!isTab1        // Hide if not Tab 1
         break
   endswitch

   return 0
End
```

A more elegant method, useful when you have many controls, is to systematically name the controls inside each tab using a prefix or suffix that is unique to that tab, such as tab0_thisCheck, tab0_thatCheck, tab1_-multVar. Then use the ModifyControlList operation to show and hide the controls. See the **ModifyControl-List** operation for an example.

Assign the action procedure to the tab control:

```
TabControl tb, proc=TabProc
```

Click on the tabs to see whether the showing and hiding is working correctly.

Verify that the action procedure correctly shows and hides controls as you click the tabs. When this works correctly, move the controls into their final positions, inside the tab control.

During this process, the "temporary selection" shortcut comes in handy. While you are in operate mode, pressing Command-Option (*Macintosh*) or Ctrl+Alt (*Windows*) temporarily switchs to select mode, allowing you to select and drag controls.