What's Your Status?

Click & Retrieve Source CODE!

Use VB4's new StatusBar control to give the user constant feedback about your application.

by Chris Barlow

Several new controls that have been added to Visual Basic 4.0 allow you to give your application the look and feel of the most recent professional Windows applications. In the last few columns I've looked at the RichTextBox, CommonDialog, and ToolBar controls. This time I'll cover the StatusBar control, which is important because it gives the user constant feedback about your application and the environment.

I'll use the text editor I created in the last few columns, but you can use the code examples in this column to add a status bar to any of your existing applications. The text-editor source code as well as the code discussed in this column is available online in a title called GS0496.ZIP. Download the file from *VBPJ's* Development Exchange on the World Wide Web at http://www.windx.com, or from the *VBPJ* CompuServe Forum, or MSN site. For details, see "How to Reach Us" in the Letters to the Editor.

The application's status bar usually sits aligned at the bottom of the application's main window. In the past it consisted of a single "panel" where you could display text, usually indicating the action the application was performing such as "Saving the document" or "Processing record #123."

In more recent applications the style of the status bar has been changed to include several panels that display other information for the user, such as the position of the Caps Lock key. In Excel 7 one of the panels can even constantly display the sum of the currently selected range of cells.

The StatusBar control included with VB4 can support both the simple single-panel style and the more complex multiple-panel style. When you set the Style property to sbrSimple, you can use the SimpleText property to display text in the single panel. This makes for an easy transition from your older applications. However, when the Style property is set to sbrNormal, you must control the information displayed on the status bar by manipulating its Panel objects.

Yes, the StatusBar control is similar to the ToolBar control in its architecture. Just as the toolbar buttons on the ToolBar control are actually a collection of Button objects with their own properties and methods, the StatusBar control is made up of a collection of Panel objects with their own properties and methods (see Figure 1).

As you continue to work with Visual Basic 4.0, you should be getting comfortable with the syntax for dealing with these collections and objects. As you may know, you can add a button to the Buttons collection of a toolbar with the Add method, and then refer to that button within the collection using its Key property or its index:

```
Set btnX = Toolbar1.Buttons.Add _
   (, "open", , tbrDefault, "open")
Set btnX = Toolbar1.Buttons("open")
Set btnX = Toolbar1.Buttons(3)
```

In the same fashion, you can add a new Panel object to the Panels collection of the StatusBar control at run time or change the text of the first panel object with code like this:

```
Set pnlX = StatusBar1.Panels.Add _
   (,,,sbrTime,LoadPicture("icons\misc\clock03.ico")
StatusBar1.Panels(1).Text = "Loading file..."
```

ADDING THE STATUS BAR

Add the StatusBar control to your form by dragging it from the toolbox to the bottom of your form. Then press F4 to view the control's properties and set the Align property to "2 - Align Bottom" so that the StatusBar control will "stick" to the bottom of the form as it is resized. Then click in the Custom property row or right-click on the control to display the StatusBar control's property dialog. On the General tab you can set the Style property and, if you plan to use the sbrSimple style, you can set the SimpleText property.

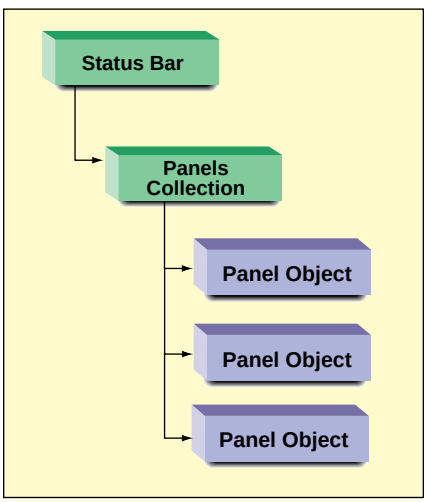

FIGURE 1 ***Don't Object to Collections.*** *You'll need to know how these objects relate to each other to develop a working status bar. The status-bar buttons are actually a collection of Panel objects with their own properties and methods.*

Remember that each status bar has a collection of Panel objects. At design time you can add a panel to the Panels

Chris Barlow is president and CEO of SunOpTech, a developer of manufacturing decision-support applications including the ObjectBank and the ObjectJob Systems, where he and Ken Henderson hold a software patent related to decentralized distributed asynchronous object-oriented systems. Chris holds degrees from Harvard Business School and Dartmouth College where he worked with Drs. Kemeny and Kurtz on the BASIC language. Reach Chris on the Internet at ChrisB@SunOpTech.com or through SunOpTech's World Wide Web server at www.SunOpTech.com.

collection using the panels tab of the StatusBar control's property dialog. Just click on the Insert Panel button to add a Panel object (see Figure 2).

You can use this dialog to set the properties of the Panel object at design time. In addition to the Index property, each panel has a Key property that you can use to refer to this particular panel within the panels collection at run time.

The power of this control is apparent when you look at the Style property. You can choose one of eight different styles for each panel on the status bar (see Table 1). When I first saw these styles I was excited! I remember the code I used to have to write so that I could display the time on a status bar for the user: add a timer control to the form, in the timer event get the current system time, format the time, set the text property to this formatted time, and so on. Now you only have to set the style property of one of your panels to sbrTime and forget it. You can also easily display the status of the Caps Lock, Num Lock, and Scroll Lock keys just as Word does. Neat!

If you are going to use a panel to display text, you can set the Text and Alignment properties to center text within the panel, for example. You can also set the panel's AutoSize property to sbrContents so that the panel will size itself to the content of the panel. Plus, you can set the AutoSize property to sbrSpring so that all panels will divide any extra space on the panel and resize themselves accordingly. To prevent a panel from getting too small when the form is resized, you can set its MinWidth property. Panel objects can also contain an image that you set with the Picture property.

For your text editor application it would make sense to have at least three panels. You can use the first panel to display text indicating to the user what the application is doing, such as "Loading document," "Saving document," and "Printing document." To do this, you'll add lines of code in the various applications' events to display certain text strings (see Listing 1).

Whenever you are going to display these kind of text messages, it is a good practice to put them in the Declarations section of a form or module rather than code the strings directly into the lines of code. This makes it a lot easier for you to find the text later if you want to update or change it. For example, for this application you could add these constants in the Declarations section of the form:

```
Const msgLoad = "Loading document..."
Const msgSave = "Saving document..."
Const msgPrint = "Printing document..."
Const msgFind = "Finding text..."
```

Then you can add code to the appropriate events to display these constants in the first panel of the of the status bar. For example, you could add two lines to the mnuSave event—one at the beginning to display the text contained in the constant msgSave and one at the end of the event to clear the text:

```
Private Sub mnuSave_Click()
StatusBar1.Panels(1).Text = msgSave
CommonDialog1.ShowSave
RichTextBox1.SaveFile (CommonDialog1.filename)
```

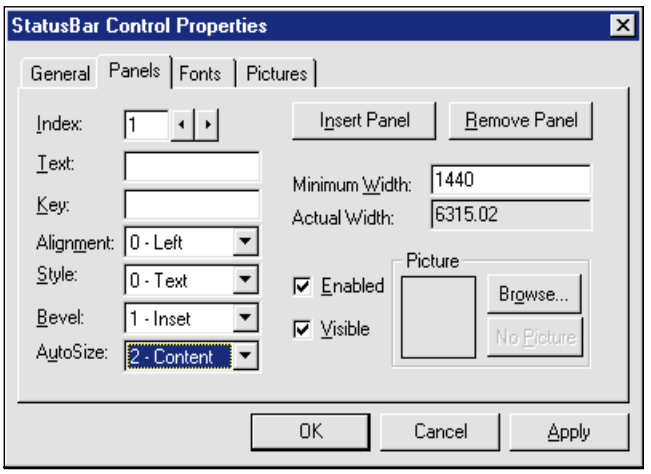

FIGURE 2 ***Control your Panels.*** *The Panels tab of the StatusBar Control Properties dialog lets you add or remove Panel objects to the Panels collection and set their properites at design time.*

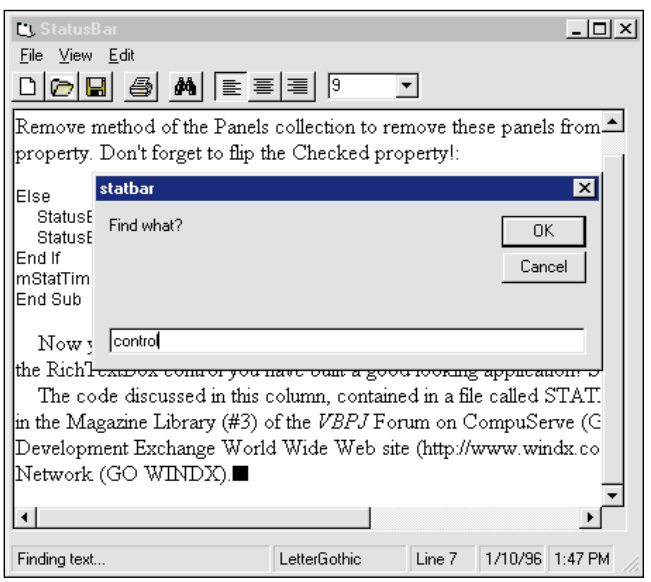

FIGURE 3 ***The Completed Application.*** *Notice how the panels resize when the font size is changed but do not shrink to less than the specified minimum width.*

Constant	Value	Description
sbrText		(Default). Text and/or a bitmap. Set text with the Text property.
SbrCaps	1	Caps Lock key. Displays the letters CAPS in bold when Caps Lock is enabled, and dimmed when disabled.
sbrNum	2	Number Lock. Displays the letters NUM in bold when the Number Lock key is enabled, and dimmed when disabled.
sbrIns	3	Insert key. Displays the letters INS in bold when the Insert key is enabled, and dimmed when disabled.
sbrScrl	4	Scroll Lock key. Displays the letters SCRL in bold when Scroll Lock is enabled, and dimmed when disabled.
sbrTime	5	Time. Displays the current time in the system format.
sbrDate	6	Date. Displays the current date in the system format.
sbrKANA	7	KANA. Displays KANA when Scroll Lock is enabled.

TABLE 1 ***Panel Object Styles.*** *I've expanded this table from the Visual Basic help file to include the new KANA style that was added after the help file was created.*

```
StatusBar1.Panels(1).Text = ""
End Sub
```

In the second panel you can display the name of the font at the current cursor position. The RichTextBox control has a property called SelFontName for the font name of the selected font. Simply add this line of code to the SelChange event of your RichTextBox control:

```
StatusBar1.Panels(2).Text = RichTextBox1.SelFontName
```

THE VB4 STATUSBAR CONTROL CAN SUPPORT BOTH THE SIMPLE SINGLE-PANEL STYLE AND THE MORE COMPLEX MULTIPLE-PANEL STYLE.

In the third panel you can display the current line number of the cursor. The RichTextBox control has a method called GetLineFromChar that returns the line number given a character position. The SelStart property of the RichTextBox control always contains the cursor's character position. So this code in the SelChange event of your RichTextBox control updates the third panel with the proper line number:

```
StatusBar1.Panels(3).Text = "Line " & _
   RichTextBox1.GetLineFromChar _
   (RichTextBox1.SelStart) + 1
```

MAKE PANELS AN OPTION

Your users may want to have the date and time displayed in panels on the status bar. But they may just want to use the default time display on the Windows 95 tray. You can use code at run time to add or remove date and time panels from the status bar.

Add to your form a top-level menu item called View, then add a menu item on the View menu with a caption of StatusBar Time and a name of mStatTime. Then you can add code in the mStatTime_click event to check and uncheck this menu item as you add and remove the date and time panels. First, if this menu item is not checked, use the Add method of the Panels collection to add a Panel object with a key of "date" and a style of "sbrDate" by passing these parameters to the Add method:

```
Private Sub mStatTime_Click()
Dim pnl As Panel
If Not mStatTime.CHECKED Then
   Set pnl = StatusBar1.Panels.Add (, "date", , sbrDate)
   pnl.AutoSize = sbrContents
   pnl.MinWidth = 720
```

After the Add method returns the new Panel object, you can use

TEXT CONTINUED ON PAGE 86.

VB4

```
Option Explicit
Public sFind As String
Const msgLoad = "Loading document..."
Const msgSave = "Saving document..."
Const msgPrint = "Printing document..."
Const msgFind = "Finding text..."

Private Sub Combo1_Change()
RichTextBox1.SelFontSize = Combo1
RichTextBox1.SetFocus
End Sub

Private Sub Combo1_Click()
RichTextBox1.SelFontSize = Combo1
RichTextBox1.SetFocus
End Sub

Private Sub Form_Load()
'Initialize the combo box
Show
With Combo1
.Width = Toolbar1.Buttons("combo1").Width
.Left = Toolbar1.Buttons("combo1").Left
.Top = Toolbar1.Buttons("combo1").Top
.AddItem "10"
.AddItem "12"
.AddItem "14"
.AddItem "16"
.ListIndex = 0
.ZOrder
End With
End Sub

Private Sub mStatTime_Click()
Dim pnl As Panel
If Not mStatTime.CHECKED Then
  Set pnl = StatusBar1.Panels.Add (, "date", , sbrDate)
  pnl.AutoSize = sbrContents
  pnl.MinWidth = 720

  Set pnl = StatusBar1.Panels.Add
  With pnl
     .Key = "time"
     .Style = sbrTime     ' Time style
     .AutoSize = sbrContents
     .MinWidth = 720
  End With
Else
   StatusBar1.Panels.Remove ("date")
   StatusBar1.Panels.Remove ("time")
End If
mStatTime.CHECKED = Not mStatTime.CHECKED
End Sub

Private Sub Form_Resize()
With Combo1
.Width = Toolbar1.Buttons("combo1").Width
.Left = Toolbar1.Buttons("combo1").Left
.Top = Toolbar1.Buttons("combo1").Top
End With
End Sub

Private Sub mnuExit_Click()
Unload Me
End
End Sub

Private Sub mnuFind_Click()
StatusBar1.Panels(1).Text = msgFind
sFind = InputBox("Find what?", , sFind)
RichTextBox1.Find sFind
StatusBar1.Panels(1).Text = ""
End Sub

Private Sub mnuFont_Click()
CommonDialog1.Flags = cdlCFBoth + cdlCFEffects
CommonDialog1.ShowFont
With RichTextBox1
  .SelFontName = CommonDialog1.FontName
  .SelFontSize = CommonDialog1.FontSize
  .SelBold = CommonDialog1.FontBold
  .SelItalic = CommonDialog1.FontItalic
  .SelStrikethru = CommonDialog1.FontStrikethru
  .SelUnderline = CommonDialog1.FontUnderline
End With
End Sub

Private Sub mnuNew_Click()
RichTextBox1.Text = ""
End Sub

Private Sub mnuNext_Click()
RichTextBox1.SelStart = RichTextBox1.SelStart + _
  RichTextBox1.SelLength + 1
RichTextBox1.Find sFind, , Len(RichTextBox1)
End Sub

Private Sub mnuOpen_Click()
StatusBar1.Panels(1).Text = msgLoad
CommonDialog1.ShowOpen
RichTextBox1.LoadFile (CommonDialog1.filename)
StatusBar1.Panels(1).Text = ""
End Sub

Private Sub mnuPrint_Click()
StatusBar1.Panels(1).Text = msgPrint
CommonDialog1.Flags = cdlPDReturnDC + cdlPDNoPageNums
If RichTextBox1.SelLength = 0 Then
  CommonDialog1.Flags = CommonDialog1.Flags + cdlPDAllPages
Else
  CommonDialog1.Flags = CommonDialog1.Flags + cdlPDSelection
End If
CommonDialog1.ShowPrinter
RichTextBox1.SelPrint CommonDialog1.hDC
StatusBar1.Panels(1).Text = ""
End Sub

Private Sub mnuSave_Click()
StatusBar1.Panels(1).Text = msgSave
CommonDialog1.ShowSave
RichTextBox1.SaveFile (CommonDialog1.filename)
StatusBar1.Panels(1).Text = ""
End Sub

Private Sub RichTextBox1_SelChange()
Select Case RichTextBox1.SelAlignment
Case rtfLeft
  Toolbar1.Buttons("Left").Value = tbrPressed
Case rtfCenter
  Toolbar1.Buttons("Center").Value = tbrPressed
Case rtfRight
  Toolbar1.Buttons("Right").Value = tbrPressed
Case Else
  Toolbar1.Buttons("Left").Value = tbrUnpressed
  Toolbar1.Buttons("Center").Value = tbrUnpressed
  Toolbar1.Buttons("Right").Value = tbrUnpressed
End Select
Combo1.Text = RichTextBox1.SelFontSize
StatusBar1.Panels(2).Text = RichTextBox1.SelFontName
StatusBar1.Panels(3).Text = "Line " & _
  RichTextBox1.GetLineFromChar (RichTextBox1.SelStart) + 1
End Sub

Private Sub Toolbar1_ButtonClick(ByVal Button As Button)
Select Case Button.Key
Case "New": mnuNew_Click
Case "Open": mnuOpen_Click
Case "Save": mnuSave_Click
Case "Print": mnuPrint_Click
Case "Find": mnuFind_Click
Case "Left": RichTextBox1.SelAlignment = rtfLeft
Case "Center": RichTextBox1.SelAlignment = rtfCenter
Case "Right": RichTextBox1.SelAlignment = rtfRight
End Select
End Sub
```

LISTING 1 ***Give Yourself Some Status.*** *Here's the code you need to add a status bar to your application. It's also available on the VBCD, in the Magazine Library (#3) of the* VBPJ *Forum on CompuServe, the* VBPJ *Development Exchange World Wide Web site, or the* VBPJ *site on The Microsoft Network.*

TEXT CONTINUED FROM PAGE 82.
the method to set the object's AutoSize and MinWidth properties. You could also use the Add method without any parameters and set each property individually for the Time panel:

```
Set pnl = StatusBar1.Panels.Add
   With pnl
      .Key = "time"
      .Style = sbrTime ' Time style
      .AutoSize = sbrContents
      'Content
      .MinWidth = 720
   End With
```

I think this second bit of code, in which you use the Add method without any parameters and then set the properties, is a bit easier to follow. Next, if this menu item is already checked, you can use the Remove method of the Panels collection to remove these panels from the status bar by referencing their Key property. Don't forget to flip the Checked property:

```
Else
   StatusBar1.Panels.Remove ("date")
   StatusBar1.Panels.Remove ("time")
End If
mStatTime.CHECKED = Not
mStatTime.CHECKED
End Sub
```

Now you have a fully functional status bar for your application (see Figure 3). Combine it with the ToolBar control and the RichTextBox control, and you've built a good looking application!

YOU CAN CHOOSE ONE OF EIGHT DIFFERENT STYLES FOR EACH PANEL ON THE STATUS BAR.

When my February column, "Your First VB4 App," appeared with instructions for developing a simple text editor in about five minutes, I received a lot of e-mail from readers with their own times and tips. Lee Nelson reported a bug in the print routine using the RichTextBox control. The hDC handed to the RichTextBox control must be the printer.hDC rather than the Common-Dialog control hDC. Here is the updated code:

```
Private Sub mnuPrint_Click()
On Error Resume Next
CommonDialog1.Flags = cdlPDReturnDC + _
   cdlPDNoPageNums
CommonDialog1.CancelError = True
If RichTextBox1.SelLength = 0 Then
   CommonDialog1.Flags = _
      CommonDialog1.Flags + _
      cdlPDAllPages
Else
   CommonDialog1.Flags = _
      CommonDialog1.Flags + _
      cdlPDSelection
End If
CommonDialog1.ShowPrinter
If Err = 0 Then
   Printer.Print ""
   RichTextBox1.SelPrint Printer.hDC
   Printer.EndDoc
End If
End Sub
```