

WORD HACKS™

*Tips & Tools for
Taming Your Text*



O'REILLY®

Andrew Savikas

HACK
#65

Show Progress from VBA

When macros take a long time to run, people get nervous. Did it crash? How much longer will it take? Do I have time to run to the bathroom? Relax. This hack shows you two ways to create a macro progress bar using VBA.

Before adding a full-fledged progress bar to your macro, consider whether something more subtle might be effective enough to keep the macro user informed. Within a macro, you can use the `StatusBar` property to display text in Word's *status bar*—the little area at the bottom of the window that displays the current page, line count, and so forth.

The following macro displays a personalized message in the status bar. Put the macro in the [template of your choice](#) [Hack #50] and run it from Tools → Macro → Macros:

```
Sub SayHello()  
    StatusBar = "Hello, " & Application.UserName & _  
        ". My that's a nice shirt you're wearing."  
End Sub
```

You can take a tip from Word, which often displays messages in the status bar (e.g., when you save a document), and use the status bar as a means of communication from within a macro.

For example, the following macro uses a `For Each` loop [Hack #66] to highlight any paragraph set to outline Level 2 that contains more than 10 words. As it completes this task, it prints the text of the paragraph to the status bar:

```
Sub HighlightLongHeadings()  
    Dim para As Paragraph  
    For Each para In ActiveDocument.Paragraphs  
        StatusBar = "Checking: " & para.Range.Text  
        If para.OutlineLevel = wdOutlineLevel2 Then  
            If para.Range.Words.Count > 10 Then  
                para.Range.HighlightColorIndex = wdBrightGreen  
            End If  
        End If  
    Next para  
    StatusBar = ""  
End Sub
```

This solution usually provides enough visual feedback to keep users assured that the macro's still hard at work and that Word hasn't crashed.

If you want a more specific, or just less subtle, feedback method, you can create a custom progress bar that appears in its own dialog box while your macro runs. The following sections describe two ways to create your own progress bar using VBA. Both adapt the `HighlightLongHeadings` macro shown above.

Continuous Progress

The first technique combines the code for the progress bar with the code for the macro.

To keep the example simple, you should put this code in your Normal template. Select Tools → Macro → Visual Basic Editor, choose Normal in the Project Explorer (near the top left of the window), and then select Insert → UserForm. Next, choose View → Toolbox to display the Toolbox (it may already be showing). Select the Label control (the one with the “A” on it). Now move your cursor to the UserForm and drag the cursor to create a new label, like the one shown in Figure 7-9. Try to position the top-left corner of the label near the top left of the UserForm.

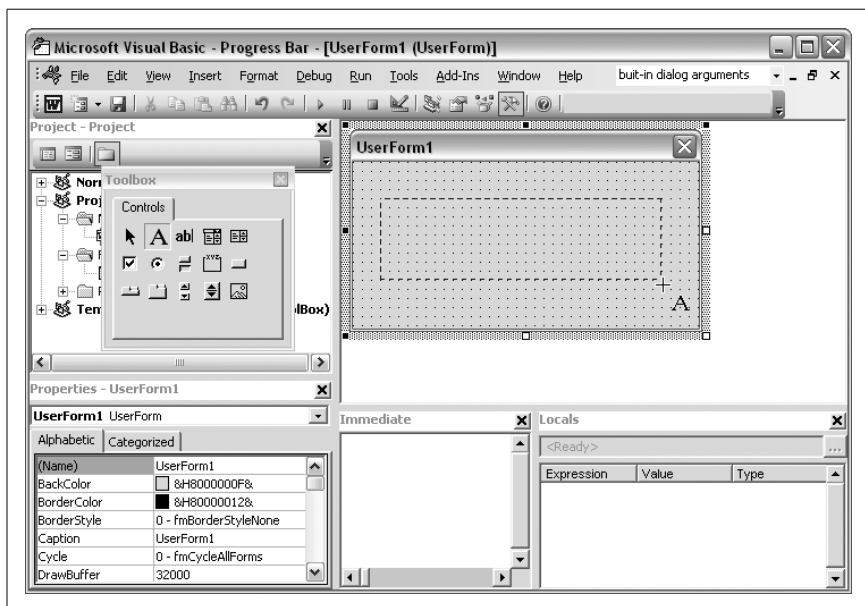


Figure 7-9. Creating a simple progress bar with a UserForm

Next, select View → Code and insert the following code:

```
Private Sub UserForm_Activate()  
    Dim lParaCount As Long  
    Dim i As Integer  
    Dim para As Paragraph  
    Dim lMaxProgressBarWidth As Long  
    Dim sIncrement As Single  
  
    ' Resize the UserForm
```

```

Me.Width = 240
Me.Height = 120

' Resize the label
Me.Label1.Height = 50
Me.Label1.Caption = ""
Me.Label1.Width = 0
Me.Label1.BackColor = wdColorBlue

lMaxProgressBarWidth = 200
lParaCount = ActiveDocument.Paragraphs.Count
sIncrement = lMaxProgressBarWidth / lParaCount
i = 1

For Each para In ActiveDocument.Paragraphs
    Me.Label1.Width = Format(Me.Label1.Width + sIncrement, "#.##")
    Me.Caption = "Checking " & CStr(i) & " of " & CStr(lParaCount)
    Me.Repaint
    If para.OutlineLevel = wdOutlineLevel2 Then
        If para.Range.Words.Count > 10 Then
            para.Range.HighlightColorIndex = wdBrightGreen
        End If
    End If
    i = i + 1
Next para

Unload Me

End Sub

```

From the Project Explorer, select one of the code modules in Normal, as shown in Figure 7-10. If you don't have any code modules in Normal, select Insert → Module to create one.

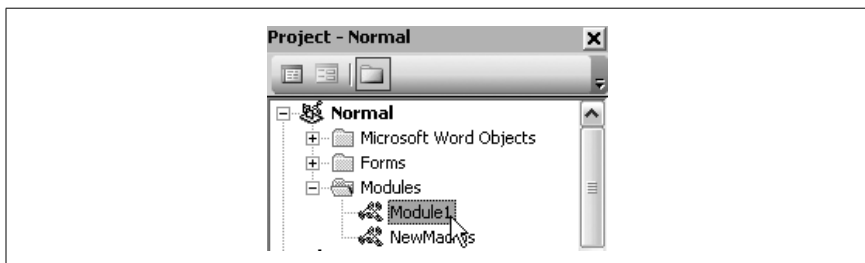


Figure 7-10. Select one of the code modules in your Normal template

In the code module you've selected, insert the following code:

```

Sub HighlightLongHeadings()
    UserForm1.Show
End Sub

```

Show Progress from VBA

Now select File → Close and Return to Microsoft Word. To run the macro, select Tools → Macro → Macros and choose HighlightLongHeadings. When you run the macro, you'll see a progress bar like the one shown in Figure 7-11.

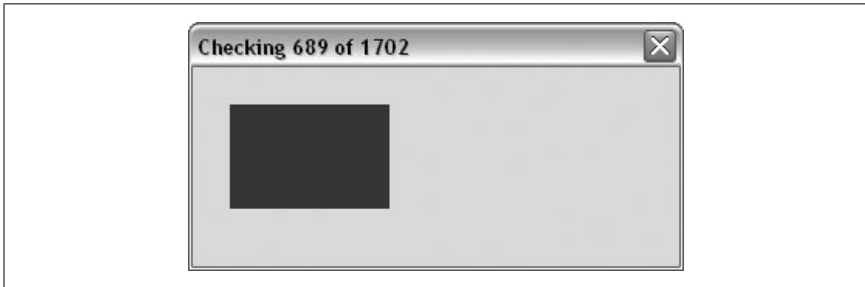


Figure 7-11. A simple progress bar in action



If the document is very short, you probably won't see the progress bar—it'll finish filling in too fast. Test this out on a long document to really see it in action.

One of the lines in the UserForm code deserves a closer look:

```
Format(Me.Label1.Width + sIncrement, "#.##")
```

The variable `sIncrement` is the final width of the progress bar divided by the total number of paragraphs in the document. As the macro visits each paragraph in the document, the width of the bar increases by the value of `sIncrement`. Since the maximum width of the bar in this example is 200 pixels (as defined in the variable `lMaxProgressBarWidth`), if there are 10 paragraphs in the document, the width of the bar will increase by 20 pixels as each paragraph is examined.

If there are hundreds or thousands of paragraphs in a document, the value of `sIncrement` can become quite small—smaller than the measurements UserForms are designed to handle. When that happens, VBA will round the number according to its own internal rounding rules, which can cause the width of the progress bar to eventually exceed the width of the UserForm. However, if you use the `Format` function, the increment amount will be rounded more precisely, keeping it confined to the boundaries of the UserForm.

Incremental Progress

One drawback to the technique described in the previous section is that the code for the progress bar is mixed with the code used to modify the document. To create another macro that displays a similar progress bar, you'd need to create another, similar UserForm. But by separating the code for the progress bar from the code that works on the document, you can reuse your progress bar in a variety of situations.

This section shows you how to create a dialog that reports the progress of a macro as a percentage, in increments of 10%, as shown in Figure 7-12. You can use this same progress bar from within any macro whose progress can be translated into a percentage.

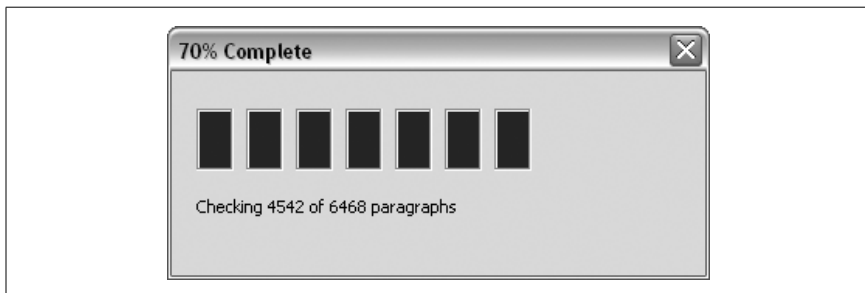


Figure 7-12. A progress bar that displays percentage increments

To keep the example simple, you should put this code in your Normal template. Select Tools → Macro → Visual Basic Editor, choose Normal in the Project Explorer (near the top left of the window), and then select Insert → UserForm. Next, choose View → Toolbox to display the Toolbox (it may already be showing).

On the Toolbox, select the Frame control (the box with “xyz” at the top), and then draw a single frame on your blank UserForm. With the frame selected, go to the Properties window. Change the frame’s height to 30 and its width to 18 and set its Visible property to False. Then delete the frame’s caption and change its background color to blue, as shown in Figure 7-13.

In the listbox at the top of the Properties window, select UserForm1 instead of Frame1, and then change the ShowModal property to False. While in the Properties window, change the name of the UserForm to IncrementalProgress.

Now go back to the UserForm itself and select the frame. Choose Edit → Copy, and then paste the frame nine times. Align the 10 frames as best you can in a single row. While holding down the Ctrl key, select all of the

Show Progress from VBA

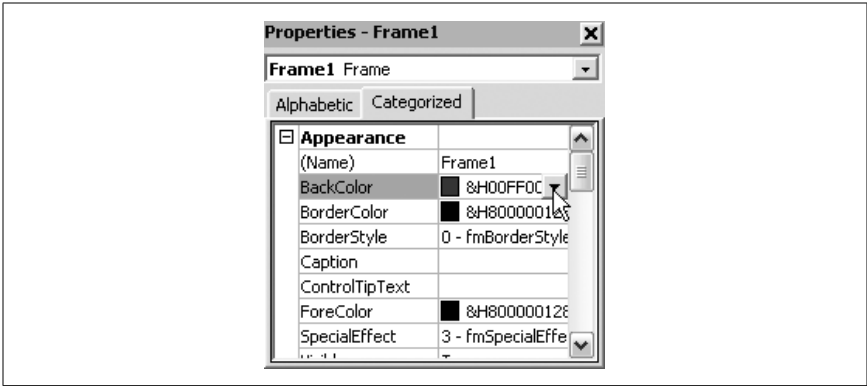


Figure 7-13. Change the frame’s caption and background color from the Properties window

frames. Then select Format → Align and align the centers and tops of all the frames.

Next, select the Label control from the Toolbox (the one with the “A” on it) and draw a label underneath the frames, as shown in Figure 7-14. From the Properties window, delete the label’s caption.

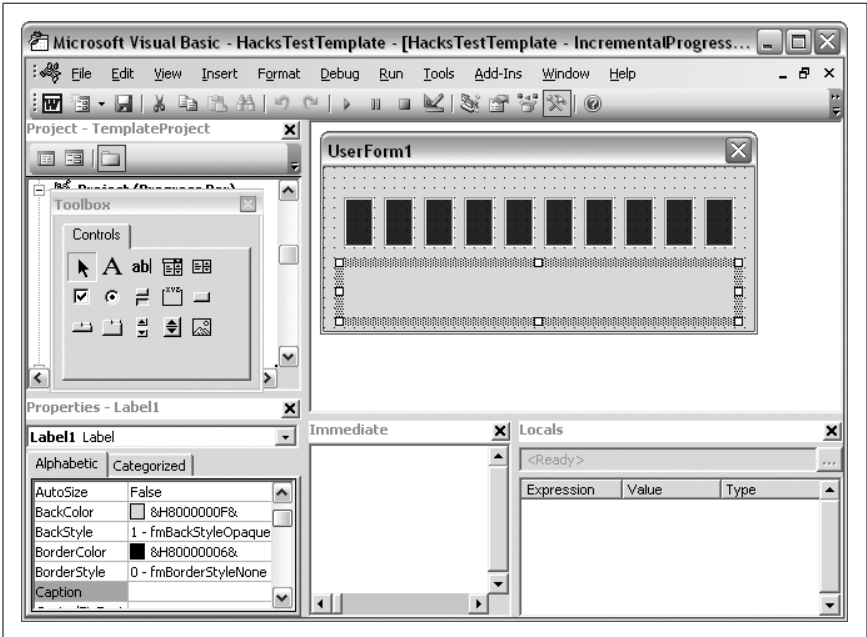


Figure 7-14. Creating an incremental progress bar

With this method, you display the dialog when your macro starts, then periodically increment its progress as a percentage. It involves more code, but it's more versatile than the first method.

Now select View → Code and insert the following:

```
Private Sub UserForm_Initialize()  
Me.Caption = "0% Complete"  
End Sub  
  
Public Function Increment(sPercentComplete As Single, _  
    sDescription As String)  
On Error Resume Next  
Me.Label1.Caption = sDescription  
Me.Repaint  
Dim iPercentIncrement As Integer  
iPercentIncrement = Format(sPercentComplete, "#")  
  
Select Case iPercentIncrement  
    Case 10  
        Me.Frame1.visible = True  
        Me.Caption = "10% Complete"  
        Me.Repaint  
    Case 20  
        Me.Frame2.visible = True  
        Me.Caption = "20% Complete"  
        Me.Repaint  
    Case 30  
        Me.Frame3.visible = True  
        Me.Caption = "30% Complete"  
        Me.Repaint  
    Case 40  
        Me.Frame4.visible = True  
        Me.Caption = "40% Complete"  
        Me.Repaint  
    Case 50  
        Me.Frame5.visible = True  
        Me.Caption = "50% Complete"  
        Me.Repaint  
    Case 60  
        Me.Frame6.visible = True  
        Me.Caption = "60% Complete"  
        Me.Repaint  
    Case 70  
        Me.Frame7.visible = True  
        Me.Caption = "70% Complete"  
        Me.Repaint  
    Case 80  
        Me.Frame8.visible = True  
        Me.Caption = "80% Complete"  
        Me.Repaint  
    Case 90
```

Show Progress from VBA

```

        Me.Frame9.visible = True
        Me.Caption = "90% Complete"
        Me.Repaint
    Case 100
        Me.Frame10.visible = True
        Me.Caption = "100% Complete"
        Me.Repaint
    End Select
End Function

```

You can now use the progress bar from within your macros. All you need to do is provide the percentage and any text you'd like displayed underneath the progress bars.

The following is the `HighlightLongHeadings` macro, revised to use this progress bar. The lines shown in bold are the ones that interact with the progress bar.

```

Sub HighlightLongHeadings()
    Dim lParaCount As Long
    Dim sPercentage As Single
    Dim i As Integer
    Dim para As Paragraph
    Dim sStatus As String

```

IncrementalProgress.Show

```

lParaCount = ActiveDocument.Paragraphs.Count
i = 1

For Each para In ActiveDocument.Paragraphs

    sPercentage = (i / lParaCount) * 100
    sStatus = "Checking " & i & " of " & lParaCount & " paragraphs"
    IncrementalProgress.Increment sPercentage, sStatus

    If para.OutlineLevel = wdOutlineLevel2 Then
        If para.Range.Words.Count > 10 Then
            para.Range.HighlightColorIndex = wdBrightGreen
        End If
    End If

    i = i + 1
Next para

```

Unload IncrementalProgress

```

End Sub

```

Running this macro will display the progress bar shown in [Figure 7-12](#).

Your macros will take longer to run, because the progress bar adds overhead. You should test versions of your macros with and without the progress bar to determine whether you find the performance hit acceptable.



The above code assumes you will hit each percentage stop along the way. If you expect to skip increments, modify the code to make sure you “turn on” all the increment frames lower than the current one. For example:

```
...
Case 40
  With Me
    .Frame1.Visible = True
    .Frame2.Visible = True
    .Frame3.Visible = True
    .Frame4.Visible = True
    .Caption = "40% Complete"
    .Repaint
  End With
...

```