DoEvents
Do You DoEvents
Some version only use
 Application.DoEvents()

Lots of programmers don't even know about Visual Basic's DoEvents function. This is not surprising because'few Visual Basic programs need it. DoEvents returns control to the operating system temporarily, allowing it to process other events that may have occurred. In my experience, the only time DoEvents is needed is when a program has code that takes a long time to execute, such as certain complex mathematical calculations. By calling DoEvents at strategic locations in your code you can improve program responsiveness.

To see what I mean, create a Standard EXE project in Visual Basic and place one CommandButton and one TextBox on the form. Then, put the following code in the Command Button's Click event procedure:

```
Private Sub Command1_Click()

        Dim i As Long, j As Long

        For i = 1 To 100
            Text1.Text = i
            For j = 1 To 100000
            Next
        Next

        Text1.Text = "Done"

    End Sub
```

You can see that the code has one loop within another, loops that will take a few seconds to complete (You may want to adjust the value that the inner loop counts to depending on the speed of your system). Each time the outer loop iterates, the current value of i is displayed in the text box. When the loops are finished, "done" is displayed.

What actually happens when you run the program, however, is that the text box does not change until "done" is displayed. The problem is that the system was so busy executing the loops that the requests to display i in the text box got stalled in Windows queue. When the loops were finished, all these requests were processed, too quickly for you to see on the screen.

Now, place a call to DoEvents in the code, just after the Text1.Text = i statement. When you run the program you will see that the text box "counts up" the values of i, just as you would expect. Calling DoEvents frees up the system to process the request, then returns control to the Visual Basic program.

DoEvents is not without potential problems. For example, if you call DoEvents from a procedure you must be sure that the same procedure cannot be called again before execution returns from the first call - otherwise unpredictable results may occur. Likewise, DoEvents should be avoided if other applications could interact with the procedure in unforeseen ways. Use of a Timer control or isolation of long-running code in an ActiveX component are two other approaches to improving program responsiveness.