Advancing Your Office 2007 Power: Using Visual Basic

By Greg Perry

Author of *Sams Teach Yourself Office 2007 All-in-One* **(ISBN 0-672-32901-8)**

You can automate many routine tasks that you perform with Office 2007's products by recording your keystrokes and storing them in a macro. You can easily assign the macro to a keystroke such as Ctrl+Shift+F12 and Word or Excel will zip through your automated commands the moment you press the keys.

Recorded macro keystrokes have been around for years and many Office users are familiar with them. The only drawback is they are simple and don't go the extra mile when you want something really automated.

For example, what if you consolidate a group of worksheets at the end of the month, performing some calculations, and printing them individually as well as a summary worksheet you create? You cannot automate such a major task no matter how routine it may be by recording your keystrokes.

You need more macro power. You need to use Visual Basic for Applications, or VBA.

In addition to the keyboard-recorded macros, many macros provide additional commands that go beyond the simple selection of menu options. Microsoft standardized the Visual Basic for Application *programming language* several years ago to form the foundation behind macros.

When you record a macro, your Office application isn't actually recording your keystrokes and mouse clicks, although that is the practical upshot of what it is doing. In reality, every time you press a key, select from a menu, type text, or click a button, the application adds a new VBA command to that macro's *program*.

Learn These New Terms

Programming language– A set of commands that you can put together to control a computer or another application such as Microsoft Excel.

Program– A set of instructions written in a programming language such as VBA. The program makes your computer do something.

Extra Info

Microsoft often uses the shortened name *Visual Basic* instead of *Visual Basic for Applications*. Microsoft offers a standalone programming language named Visual Basic that is identical to VBA. When used in the context of an Office application such as Excel, however, VBA and Visual Basic for Applications are both accepted.

Until you learn a programming language, its programs can be cryptic indeed. For example, a macro you record that colors a selected paragraph's background yellow and converts the paragraph's first letter to a drop cap letter is actually a VBA program that looks like this:

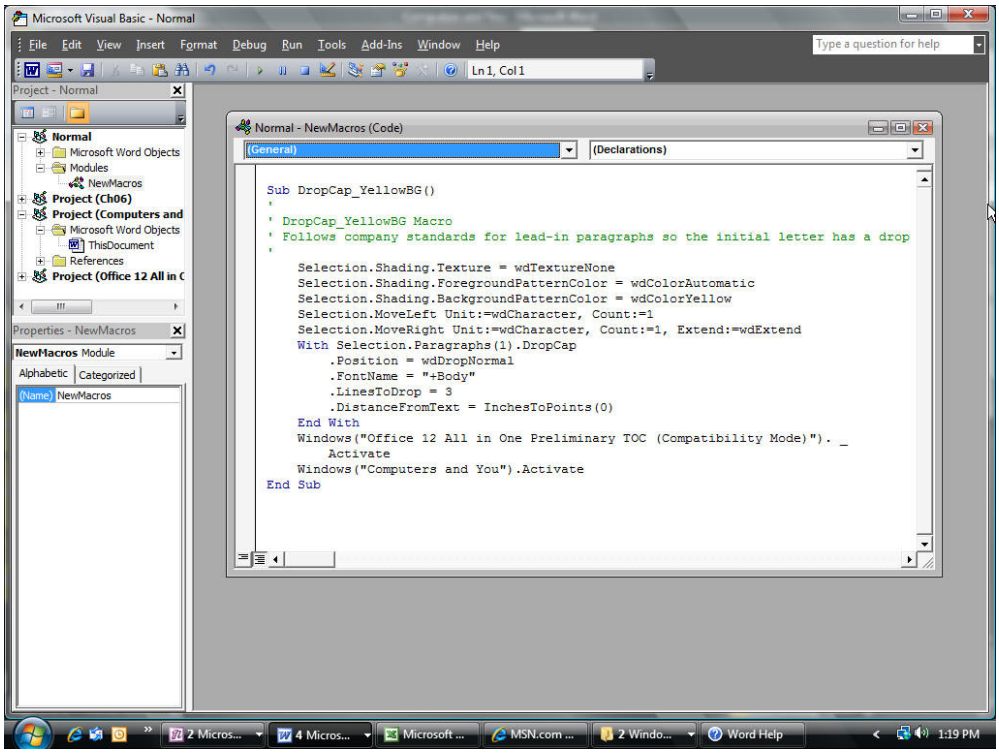
```
Sub DropCap_YellowBN()
'
' DropCap_YellowBG Macro
' Follows company standards for lead-in paragraphs so the
' initial letter uses a drop cap and the paragraph is
' colored with a yellow background.
'
   Selection.Shading.Texture = wdTextureNone
   Selection.Shading.ForegroundPatternColor = wdColorAutomatic
   Selection.Shading.BackgroundPatternColor = wdColorYellow
   Selection.MoveLeft Unit:=wdCharacter, Count:=1
   Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
   With Selection.Paragraphs(1).DropCap
        .Position = wdDropNormal
        .FontName = "+Body"
        .LinesToDrop = 3
        .DistanceFromtext = InchesToPoints(0)
   End With
   Windows("Computers and You").Activate
End Sub
```

You've got to admit, recording your keystrokes and mouse clicks is a lot easier than writing all those VBA programming language statements! Nevertheless, if you knew VBA, you could have written this code instead of recording the macro and achieved the same results.

Extra Info

Not everything in a macro's program is cryptic. Programming languages use common words and phrases for commands and program elements. Although you may not be able to write VBA programs from scratch, you can look through this program and tell what much of it is doing, such as shading the background yellow, and adding the drop cap.

In Office 2007, you get a complete Visual Basic environment from which you can write VBA programs that control virtually every aspect of your document work. From the Developer ribbon you can click the Visual Basic button to see the following programming environment open up on your screen.



When you write a Visual Basic program to control an Office application, you'll use Visual Basic's own environment.

You might wonder how Visual Basic can supplement and add functionality to an Office application other than automating simple, routine tasks. After all, Excel supports VBA, but with all the worksheet power and commands that Excel already provides, how could a VBA program controlling Excel's environment and worksheet add any benefit to the Excel user?

If all VBA did was automate simple, routine tasks in recorded macros, VBA would be a welcome addition to Office. Simple recorded macros, though, must run in exactly the same environment and with the same set of worksheets and columns each time. The keyboard macro has no room for ambiguity. For example, what if one of your company divisions was to shut down for remodeling one month? The keyboard macro that you may have recorded to consolidate all your company's four divisions would consolidate either a blank worksheet or an old one for the missing division's new data, producing an error.

Because Visual Basic is a complete programming language, the employee who needs such a consolidated report, for instance, can write a series of commands that handle unexpected conditions more gracefully than keyboard-recorded macros can. Perhaps the VBA program could read each worksheet and, if data other than zeros appears for the month totals, add that worksheet to the summary but ignore any other worksheet in which the division had no activity for the period. Such a macro – a Visual Basic program written in the VBA language – would work when a recorded macro would not.