

CONTENTS

| | | |
|-------------------|--|----|
| Chapter 1 | Introduction to Excel VBA | 1 |
| Chapter 2 | Working with Variables in Excel VBA | 7 |
| Chapter 3 | Using Message box and Input box in Excel VBA | 15 |
| Chapter 4 | Using If....Then....Else in Excel VBA | 22 |
| Chapter 5 | For.....Next Loop | 28 |
| Chapter 6 | Do.....Loop | 35 |
| Chapter 7 | Select Case.....End Select | 38 |
| Chapter 8 | Excel VBA Objects Part 1–An Introduction | 40 |
| Chapter 9 | Excel VBA Objects Part 2 –The Workbook Object | 48 |
| Chapter 10 | Excel VBA Objects Part 3 –The Worksheet Object | 53 |
| Chapter 11 | Excel VBA Objects Part 4–The Range Object | 58 |
| Chapter 12 | Working with Excel VBA Controls | 66 |
| Chapter 13 | VBA Procedures Part 1-Functions | 75 |
| Chapter 14 | VBA Procedures Part 2-Sub Procedures | 84 |
| Chapter 15 | String Handling Functions | 87 |
| Chapter 16 | Date and Time Functions | 90 |
| Chapter 17 | Sample Excel VBA Programs | 96 |

Chapter 1

Introduction to Excel VBA

1.1 The Concept of Excel VBA

VBA is the acronym for Visual Basic for Applications. It is an integration of the Microsoft's event-driven programming language Visual Basic with Microsoft Office applications such as Microsoft Excel, Microsoft Word, Microsoft PowerPoint and more. By running Visual Basic IDE within the Microsoft Office applications, we can build customized solutions and programs to enhance the capabilities of those applications.

Among the Visual Basic for applications, Microsoft Excel VBA is the most popular. There are many reasons why we should learn VBA for Microsoft Excel, among them is you can learn the fundamentals of Visual Basic programming within the MS Excel environment, without having to purchase a copy of Microsoft Visual Basic software. Another reason is by learning Excel VBA; you can build custom made functions to complement the built-in formulae and functions of Microsoft Excel. Although MS Excel has a lot of built-in formulae and functions, it is still not enough for certain complex calculations and applications. For example, it is very hard to calculate monthly payment for a loan taken using Excel's built-in formulas, but it is relatively easy to program a VBA for such calculation. This book is written in such a way that you can learn VBA for MS Excel in an easy manner, and everyone shall master it in a short time!

You can program Excel VBA in every version of Microsoft Office, including MS Office 97, MS Office2000, MS Office2002, MS Office2003, MS Office XP and MS Office 2007. The reason VBA is needed is due to the limitations in using the built-in functions of MS Excel and macro recording. By using VBA, you can build some very powerful tools in MS Excel, including financial and

scientific applications such as getting financial data from the Internet as well as linear programming.

1.2 The Visual Basic Editor in MS Excel

There are two ways which you can start programming VBA in MS Excel. The first way is to place a command button on the spreadsheet and start programming by clicking the command button to launch the Visual Basic Editor. The second way is to launch the Visual Basic Editor by clicking on the Tools menu then select Macro from the drop-down menu and choose Visual Basic Editor. Lets start with the command button first. In order to place a command button on the MS Excel spreadsheet, you need to click View on the MS Excel menu bar and then click on toolbars and finally select the Control Toolbox after which the control toolbox bar will appear, as shown in Figure 1.1. Then you click on the command button and draw it on the spreadsheet, as shown in Figure 1.2.

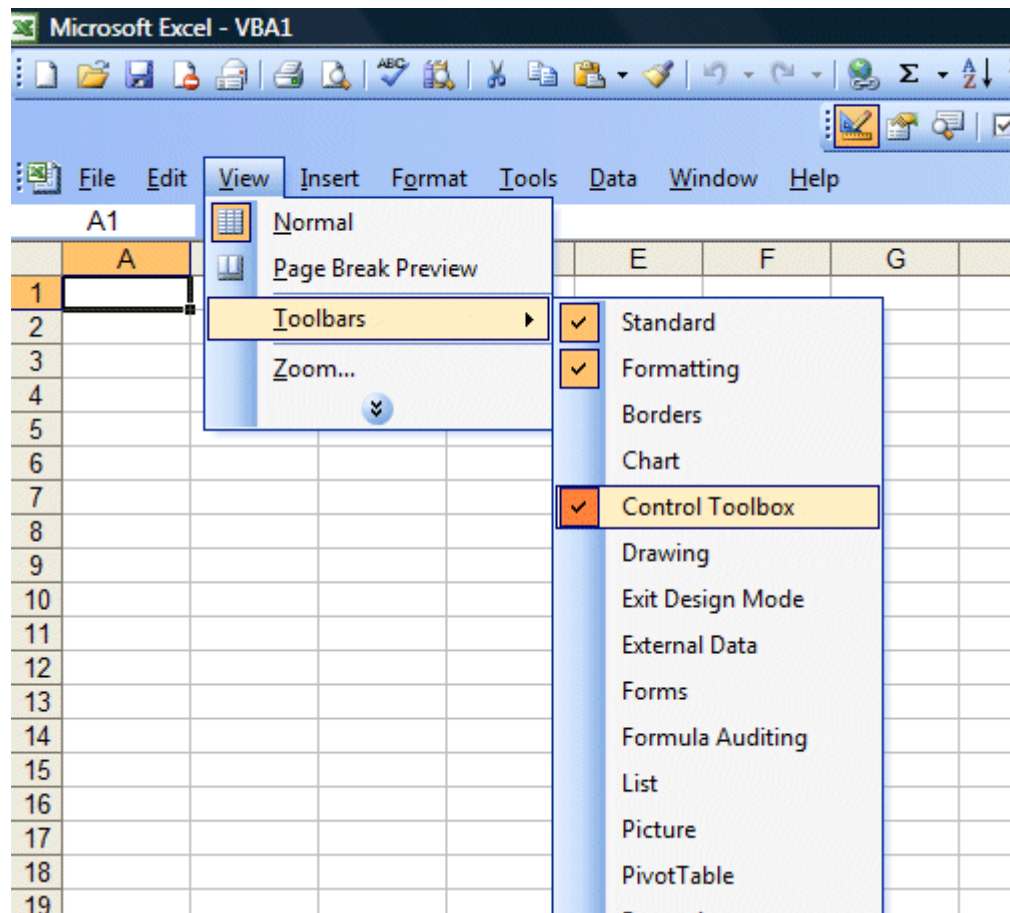


Figure 1.1: Displaying Control Toolbox in MS Excel.

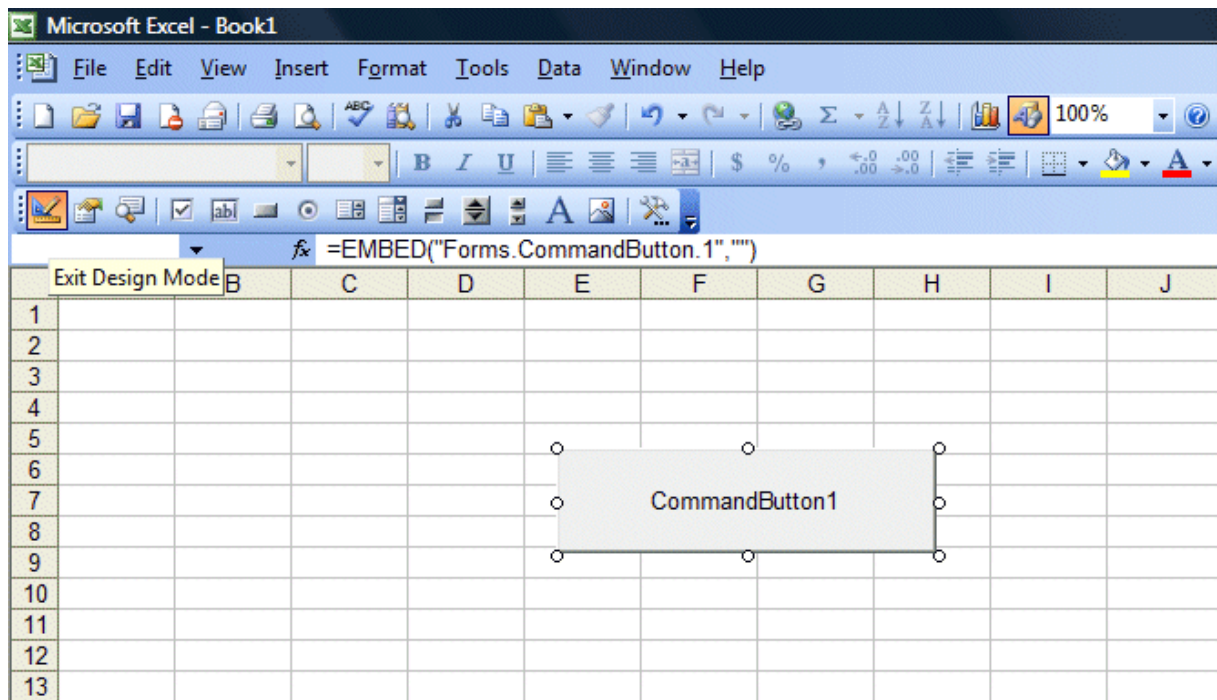


Figure 1.2: The Command Button in Design Mode

Now you select the command button and make sure the design button on the far left of the control toolbox is depressed. Next, click on the command button to launch the Visual Basic Editor. Enter the statements as shown in figure 1.3. Let's write out the code here:

Example 1.1

```
Private Sub CommandButton1_Click ()
```

```
    Range ("A1:A10").Value="Visual Basic "
```

```
    Range ("C11").Value=Range (" A11").Value +Range ("B11").Value
```

```
End Sub
```

The first statement will fill up cell A1 to cell A10 with the phrase "Visual Basic" while the second statement add the values in cell A11 and cell B11 and then show the sum in cell C11. To run the program, you need to exit the Visual Basic Editor by click the Excel button on the far left corner of the tool bar.

When you are in the MS Excel environment, you can exit the design mode by clicking the design button, then click on the command button.

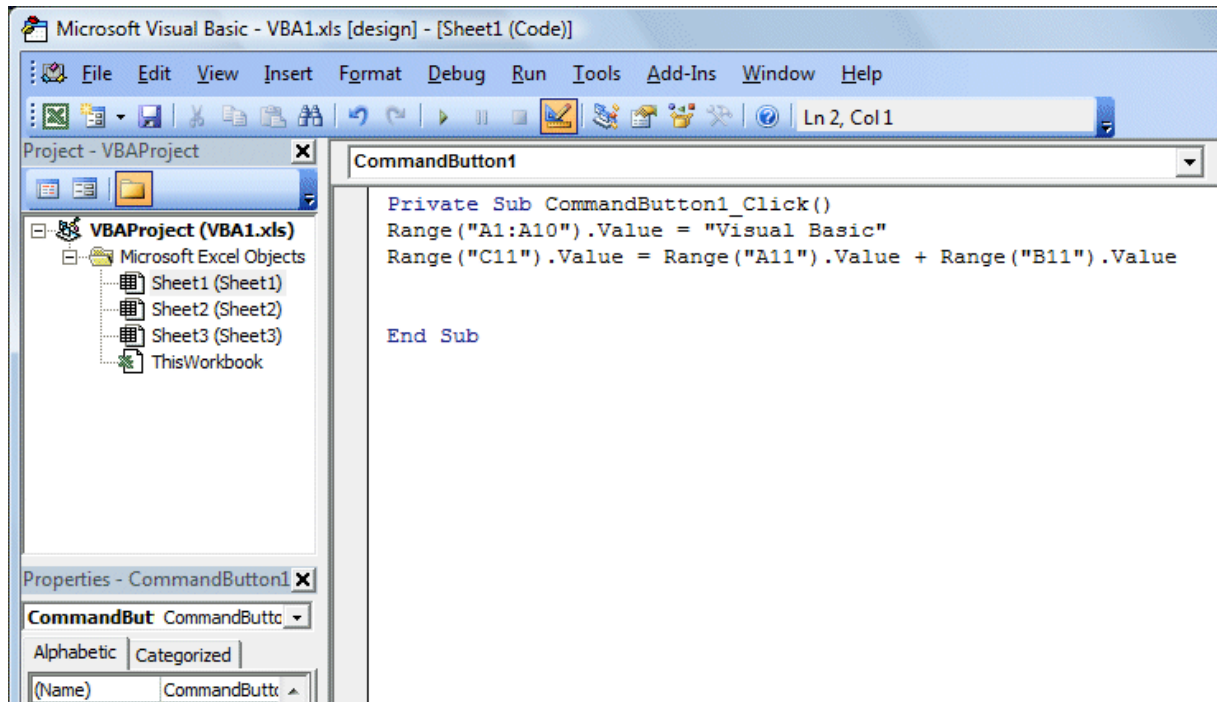


Figure 1.3: The Visual Basic Editor IDE in MS Excel

Running the above VBA will give you the following output.

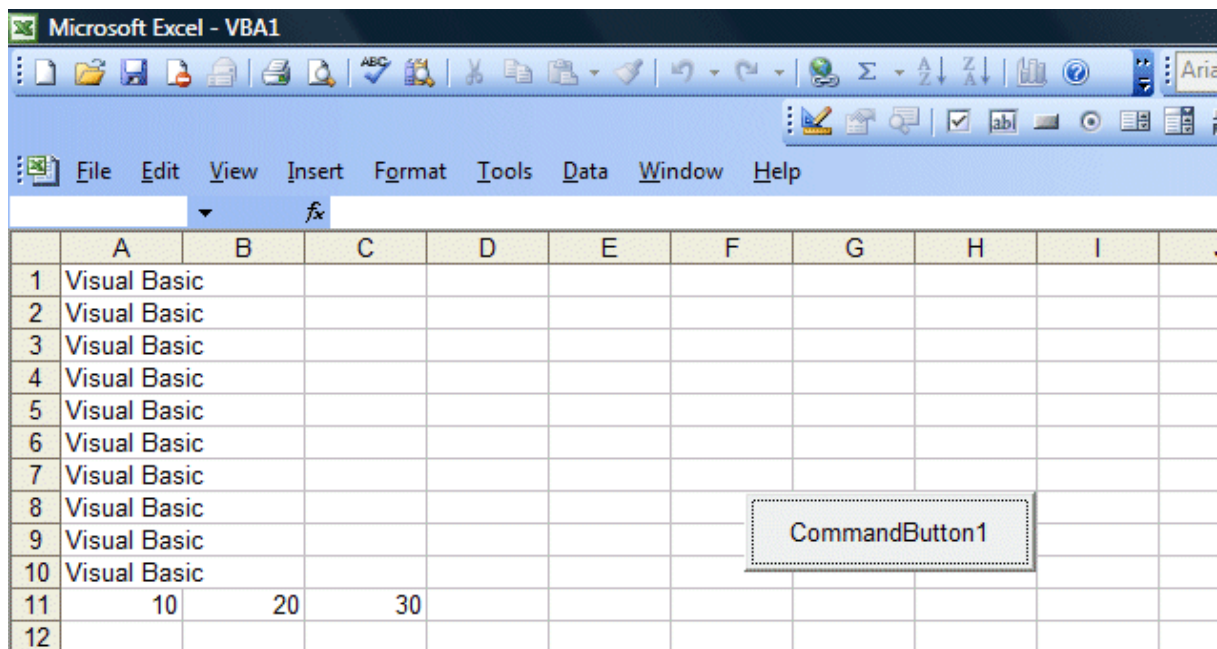


Figure 1.4:

1.3 The Excel VBA Code

Writing Excel VBA code is almost exactly the same as writing code in Visual Basic, which means you have to use syntax similar to Visual Basic. However, there are codes specially designed for use in MS Excel, such as the use of the object or function called **Range**. It is the function that specifies the value of a cell or a range of cells in MS Excel spreadsheet. The format of using Range is as follows:

```
Range("cell Name").Value=K or Range("Range of Cells").Value=K
```

Where Value is the property of Range and k can be a numeric value or a string

Example 1.2

```
Private Sub CommandButton1_Click ()
```

```
    Range ("A1").Value= "VBA"
```

```
End Sub
```

The above example will enter the text "VBA" into cell A1 of the MS Excel spreadsheet when the user presses the command button. You can also use Range without the Value property, as shown in Example 1.3:

Example 1.3

```
Private Sub CommandButton1_Click ()
```

```
    Range ("A1") = 100
```

```
End Sub
```

In the above example, clicking the command button with enter the value of 100 into cell A1 of the MS Excel spreadsheet.

The follow example demonstrates how to input values into a range of cells:

Example 1.4

'Input the value of 100 from cell A1 to cell A10

```
Private Sub CommandButton1_Click ()
```

```
    Range ("A1:A10") = 100
```

```
End Sub
```