Programming Microsoft Excel 2016 Using Visual Basic for Applications Course Notes

Chris Page

11 River Gardens, Shawbury, Shrewsbury, SY4 4LA
Tel 01939 251094
www.pagecommunication.co.uk

CONTENTS

COURSE OBJECTIVES

By the end of this course you will be able to

- Accurately record a macro using the Record macro feature, without referring to notes.
- Correctly edit a macro to make it more efficient, referring to notes if necessary.
- Briefly describe the difference between macro storage locations.
- Briefly describe what an object's Properties, Methods and Events mean.
- Accurately navigate an object hierarchy, without reference to notes.
- Briefly describe how *Application*, *Workbooks*, *Worksheets*, and *Range* objects are related.
- Correctly edit a workbook's VBA code as specified, referring to notes if necessary.
- Correctly create variables for use within VBA code without reference to notes.
- Briefly describe the difference between procedure-level variable, module level-variables and public variables.
- Briefly describe the different Boolean operators without reference to notes.
- Correctly use If..Else, and Select Case constructs, referring to notes if necessary.
- Correctly repeat blocks of code using looping structures, referring to notes if necessary.
- Briefly describe the difference between a sub procedure and a function procedure.
- Correctly create VBA code that uses a sub procedure, referring to notes if necessary.
- Correctly create VBA code that uses a message box and input box to interact with the user, referring to notes if necessary.
- Correctly create a specified customised dialog box to interact with a user, referring to notes if necessary.
- Correctly add an ActiveX control to a worksheet, without reference to notes.
- Briefly describe the difference between the *Resume*, *Resume Next* and *Resume Label* statements.
- Create VBA code that correctly traps specified errors, referring to notes if necessary.

MODULE 1 - MACROS

OBJECTIVES

At the end of this section you will be able to:

- Accurately record a macro using the Record macro feature, without referring to notes.
- Correctly edit a macro to make it more efficient, referring to notes if necessary.
- Briefly describe the difference between macro storage locations.

DIFFERENCES BETWEEN VBA AND VB

There is a difference between using Excel with associated Visual Basic for Applications (VBA) code and using normal Visual Basic (VB) code in a VB Development Environment. The syntax, or language grammar, is essentially the same. VB can create stand-alone executable program files, whereas VBA needs to reside within an Excel workbook. Excel is regarded as a host application, as it *hosts* the running of VBA.

INTRODUCTION TO MACROS

Macros are sequences of commands that automatically run Excel operations. Macros are based on the VBA programming language that enables it to control the different types of objects in Excel. It consists of a series of instructions that the computer follows in a sequence we specify. The macro is given a name that is used to run the instructions it contains.

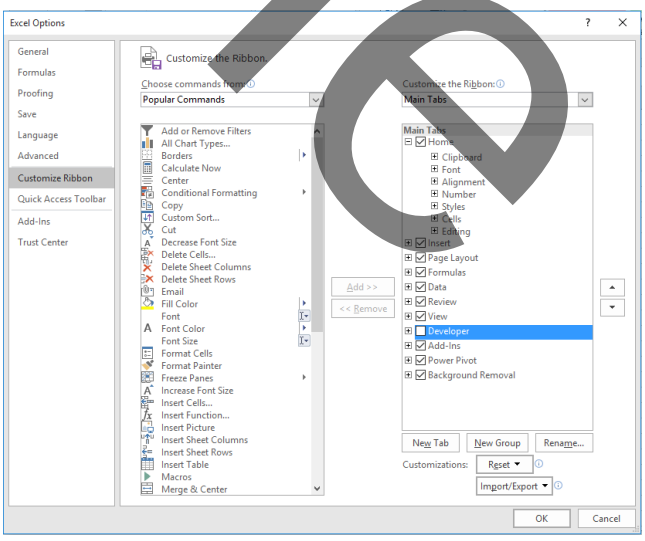
Anything we do in Excel that is of a repetitive nature is a good candidate for a macro. For instance, we might have the job of creating financial analysis reports for our company, and we want to create a macro that will enter the company name in the current cell and format it using the proper font. This is easily done with a macro, and is a simple way to show how macros work.

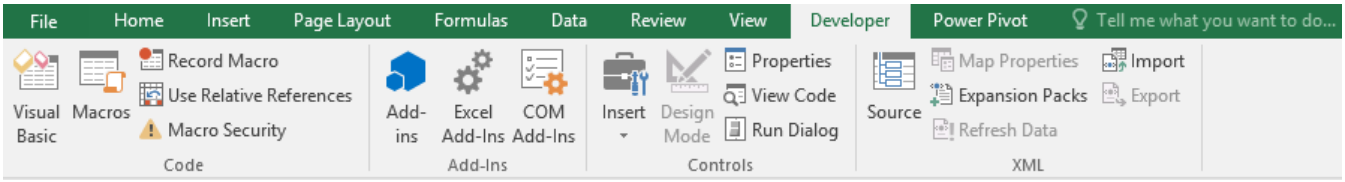
Showing the Developer Tab in the Ribbon

You might want to see the developer tab in the Ribbon. This gives you more options for working with Macros, VBA and ActiveX controls on worksheets (covered later in the course).

To display the developer tab

1. Right-click on the Ribbon and choose the **Customize the Ribbon** option that is displayed in the shortcut.
2. The Ribbon tabs are shown in the right-hand pane.
3. Tick the *Developer* box.
4. Click on **OK**.
5. The devloper tab now is shown on the Ribbon whenever Excel is run.





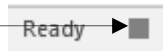
Recording and Running a Visual Basic Macro

To record a macro

1. Click on the **Record Macro** button displayed on the status bar, or if the *Developer* tab is displayed, in the *Code* group click on the **Record Macro** button.

2. Give the macro a name and type in a description.
3. Choose a **Shortcut Key** if you would like to run the macro using just Ctrl followed by a single letter. Only e, j, l, m, q and t are spare without overwriting default assignments, so consider using Upper-Case letters.
4. Select **Store Macro In** to determine where the macro is saved. This can be in:
   a. This Workbook (The current workbook).
   b. New Workbook (A different workbook).
   c. Personal Workbook (Your personal workbook)

   If you select *Personal Workbook*, Excel will store it in a file called PERSONAL.XLSB which is saved in the XLSTART folder. This will always be available whenever Excel is run.
5. Click on **OK**.
6. Perform the keystrokes that are to be saved as the Macro. Whilst the macro is recording, the **Stop Macro** button will appear in the status bar.
7. When all the keystrokes have been carried out click on the **Stop Macro** button.



*Note: If you click on the **Use Relative References** button in the **Code** group before you begin to record a macro, then cell references within the macro are all relative. This means that if, for example, you move from active cell A1 to cell C1, this will be recorded, within your recorded macro as "Select the cell that is two cells to the right of the current active cell". However, if the Use Relative References option has not been selected, this action will recorded as "Select cell C1", regardless of which cell is initially the active cell.*

To run a macro

1. Activate the worksheet from which you want to run the procedure
2. Select **View** tab in the Ribbon, and in the *Macros* group click on the **Macros** button, or if the *Developer* tab is displayed, in the *Code* group click on the **Macros** button.
3. Select the name of the macro and click on **Run.**

To save a worksheet containing a macro

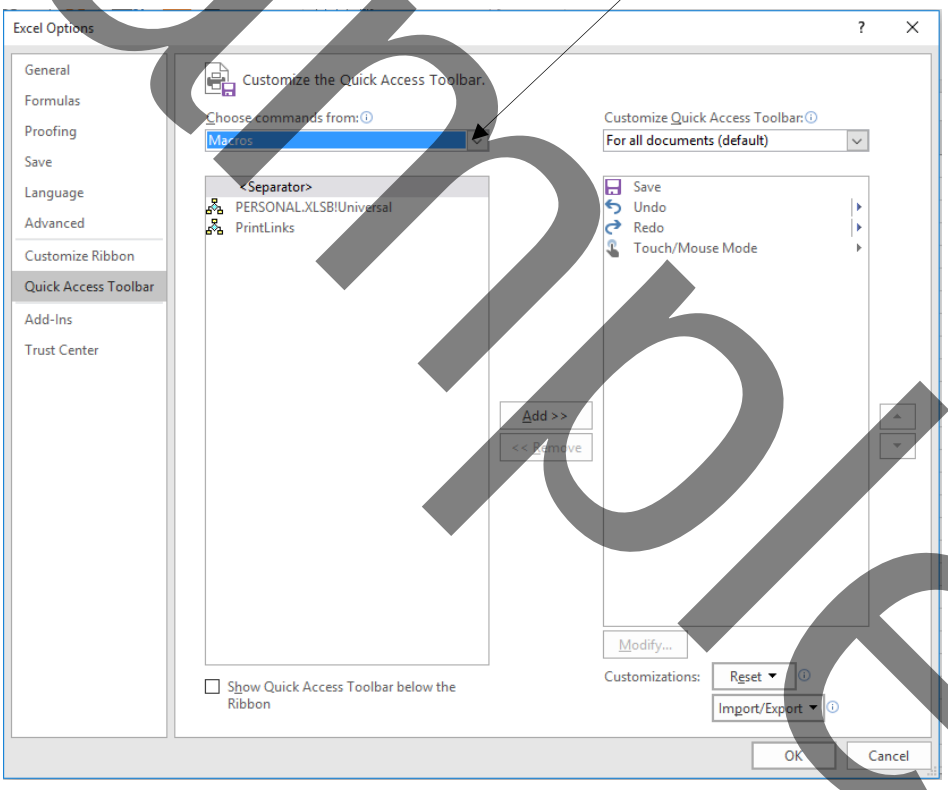
1. Click on the **File** tab, then select **Save As**.
2. Using the drop-down selector of the **Save as type** box, choose the *Excel Macro-Enabled Workbook* option. If you are saving a very large worksheet, you could also save the file as an *Excel Binary Workbook* which can also hold macros.
3. Type the required name in the File name text box. Do not worry about adding any extension of *.xlsm* or *.xlsb* as Excel will add it for you.
4. Click on **Save**, you will need to confirm the save if the filename exists already.

Macros on the Quick Access toolbar, Graphics or Form Controls on a Worksheet

Adding macros to Quick Access toolbar, and graphics or buttons on a worksheet helps to make VBA procedures easy to operate.

To attach a macro to the Quick Access Toolbar

1. Click on the **Customize Quick Access Toolbar** button.
2. Select the **More Commands…**option. The *Customize* dialog is displayed.
3. Using the *Choose commands from* drop-down selector at the top select the *Macros* option.



4. The right-hand pane shows the buttons that currently exist in the Quick Access toolbar. The left-hand pane now shows the macros that can be added.
5. Select the macro to add to the Quick Access toolbar.
6. Click on the **Add** button that is between the two panes.

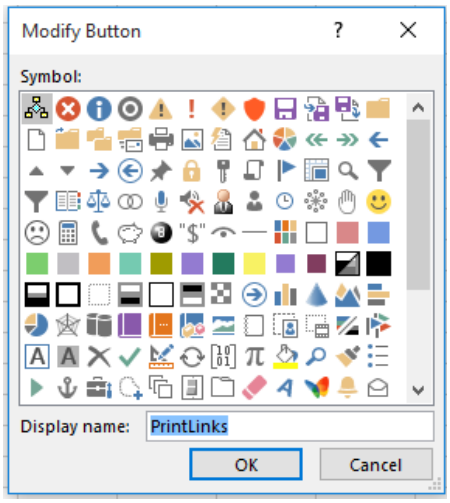
7. Once you have added the macro button, you can use the two move buttons – located to the right of the right-hand pane – to change the order of buttons.

8. To rename the tooltip that the button shows and also change the icon click on the **Modify** button.

9. The *Modify Button* dialog is displayed as shown here. Type a new name in the *Display name* box at the bottom to change the tooltip.

10. If you want the button to have an associated icon, choose one by clicking on one from the displayed *Symbol* collection.

11. Finally, click on **OK**.



Notes:

You cannot increase the size of the buttons. The only way to increase the size of the buttons is to lower the screen resolution you use.

You cannot display the Quick Access Toolbar on multiple lines.

Only commands & macros can be added to the Quick Access Toolbar.

To attach a macro to a graphic or form control on a worksheet

Macros that we frequently run can easily be attached to graphics and form controls on the worksheet.

1. Right-click on the item and select **Assign macro**.
2. Select the macro name in the displayed dialog.
3. Click on **OK**.

Time for Exercise 1

LIMITATIONS OF MACROS

While recorded Excel macros allow a certain amount of automation they still have limitations.

- In the event of an unplanned error the message displayed could confuse the user.
- Complex condition checking during recording is not possible.
- Excel macros run from start to finish. It is not possible to repeat steps within a macro (looping).
- Macros cannot define and use their own functions or solve complex mathematical problems.
- Macros may be inefficient in the way they execute as they literally record everything the user does.

The only way that these limitations can be overcome, and to achieve total control on how the automation is carried out, is to use Visual basic for Applications (VBA) code instead of macros.

VISUAL BASIC EDITOR

The VB Editor is a separate window for working with the VBA code in Excel. It is the same editor that exists within other Office applications.

To open the VB Editor

1. Either press Alt + F11, or select or if the *Developer* tab is displayed, in the *Code* group click on the **Visual Basic** button.

To return to Excel from the VB Editor

1. Either press Alt + F11, or select **Tools, Macro,** and **Visual Basic Editor** from the Menu bar, or click on the Excel icon on the toolbar.

Examining and Running the Macro

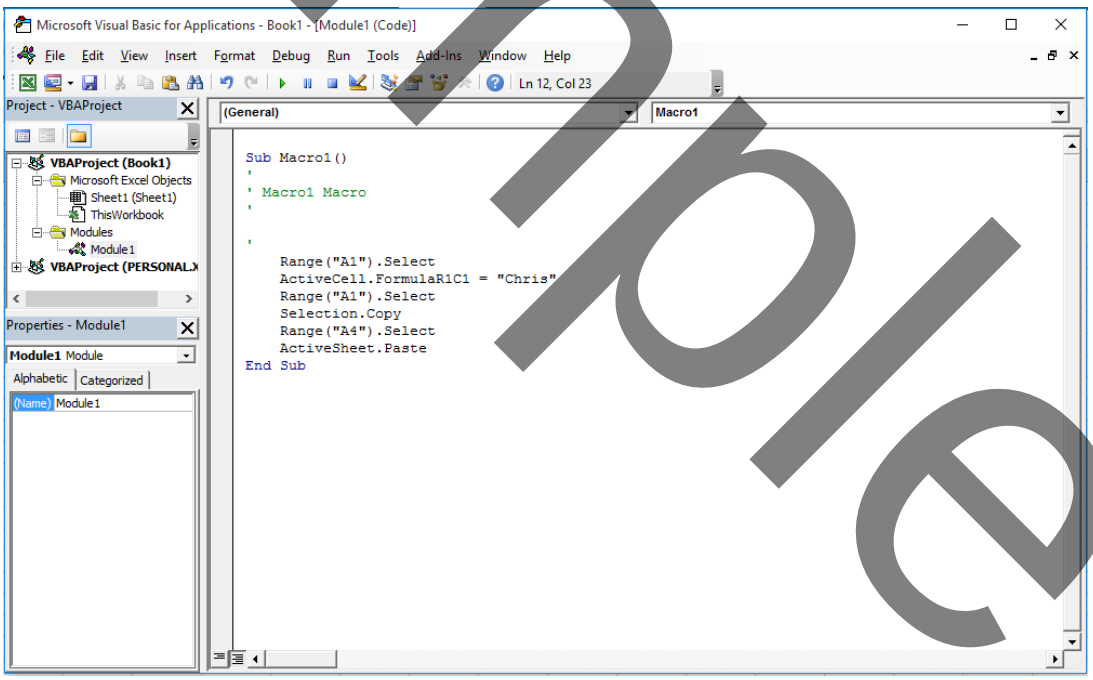
When we record a Visual Basic macro, the procedure will be placed in a Visual Basic Module.

To examine the macro

1. Select **View** tab in the Ribbon, and in the *Macros* group click on the **Macros** button, or if the *Developer* tab is displayed, in the *Code* group click on the **Macros** button.
2. Select the Macro name you wish to examine.
3. Click on the *Edit* button (this does not work for macros saved to the Personal Workbook – open the VB Editor instead).

When we first look at a Visual Basic procedure we will see that each step we took may have resulted in one or more lines of code. We may, in addition, see many lines which we did not actually record. The procedure itself appears in colour, with comments in green, keywords in blue and everything else in black.

The VB Editor Layout



The VB Editor consists of several windows all designed to help us, the code developers, work with and create VBA code for automating Excel.

The Visual Basic Editor consists of the following elements - the Menu bar, the Toolbar, Project Explorer, and the Properties Window. Additional windows will be covered later in the course.

Programming Microsoft Excel 2013 Using Visual Basic for Applications Exercises

Contents

EXERCISE 1

1. Open a new workbook.
2. Type three figures in cells A1, A2 and A3 respectively.
3. Position in cell B1 – this is needed as the start of the recorded macro will position in cell A4. *Note: You don't need to specifically position in B1; it can be any cell as long as it is not A1.*
4. Record a macro entitled **MyFirstMacro**. Give it a shortcut key of Ctrl + e. Store it in *This Workbook*. Once recording has started position in cell A4 and use Autosum to add up the three figures. Once you have pressed Enter↵, stop recording the macro.
5. Delete the contents of cell A4 (the sum).
6. Close and save the file as a macro-enabled workbook with the name **Macros**
7. Open a new workbook.
8. Position in cell B1 – this is needed as the start of the recorded macro will position in cell A1.

   Note: You don't need to specifically position in B1; it can be any cell as long as it is not A1.
9. Record a macro entitled **MyPersonalMacro**. Give it a shortcut key of Ctrl + j. Store it in *Personal Macro Workbook*. Once recording has started position in cell A1 and type *January*. Use the fill handle to copy this down to create 12 months. Stop recording the macro after you have created the 12 months.
10. Close Excel and do not save the changes to the current workbook, but make sure that you DO save the changes to the Personal Macro Workbook.
11. Start up Excel again.
12. Hold down Ctrl + e. It should not work.
13. Hold down Ctrl + j. It should work.
14. Open up **Macros.**
15. Hold down Ctrl + e. It should now work correctly. Macros stored in This Workbook only work when the workbook they are stored in are open. Macros stored in the Personal Macro Workbook work anytime.
16. Close Excel down without saving any changes.
17. Stop here.

EXERCISE 2

1. Open up the workbook named **Ex2**.
2. In the Security Warning bar, click on the **Enable content** button.
3. Press Alt + F11 to open the VBA Editor.
4. Position in the *Ex2Macro* procedure.
5. Press F5 to run the macro.
6. Record how long it takes. _____
7. Now modify the macro – within the loop - so it looks like this:

```
Sub Ex2Macro()
  'Assign the time since midnight to the variable called Start
  Start = Timer
  'Now repeat some code 1000 times
  For Count = 1 To 1000
    Range("A4").FormulaR1C1 = "=SUM(R[-3]C:R[-1]C)"
  Next Count
  'Show how long it took by taking the new
  'time from midnight away from the previous one
  MsgBox "This took " & Timer - Start & " seconds"
End Sub
```

8. Press F5 to run the macro again.
9. Record how long it takes. _____
10. The macro should be much quicker this time.
11. Close Excel and save the changes.
12. To avoid making the Project Explorer cluttered we will now delete the Personal Macro Workbook. On the Windows desktop click on Start, Programs, Accessories, Windows Explorer.
13. Navigate to the following folder:

    `C:\Users\Student\AppData\Roaming\Microsoft\Excel\XLSTART`

If you cannot see the AppData *folder you need to change the setting to show hidden items. Do this in Windows Explorer by clicking on **View** tab of the Ribbon and in the **Show/Hide** group put a check mark in the **Hidden items** check box.*

14. Delete the **Personal** Excel file.
15. Stop here.