Learn more about Microsoft Word

Start to delve into VBA and create your own macros

Communicator

The Institute of Scientific and Technical Communicators
Summer 2015

What's a technical metaphor? Find out more.

Create and implement personalised learning

Re-think your accessibility requirements with SVG

Discover how a security technology team engaged with customers

Enhancing Word with VBA macros

An introduction to using VBA to fix documents by Mike Mee.

Working within a team of four technical authors at CDL, we do a lot of work with Microsoft Word.

The types of Word-based work can range from being sent a document to proofread and send back with some simple changes (we like these!) through to being sent what could only be described as a *dog's dinner* of a document that is begging to be cleaned up.

I am sure that you have all seen those types of document: the ones that you can almost guarantee will have some combination of the wrong font, an abhorrent justification setting, portrait page footers incorrectly formatted as landscape, and maybe the wrong header styles.

It is in our team's job description to fix these documents: we call it 'CDLifying'. We dive in and start applying the necessary fixes. However, the fixes are achieved quickly by the use of Word macros.

For as long as I have been using Word, I have always been interested in how it 'ticks' under the hood and how to speed up some of the more mundane formatting and fixes that we need to do.

This article will be the first of several in which I will give you some examples to follow so that you can take your first tentative steps towards coding your own macros in VBA (Visual Basic for Applications).

Macros can be recorded by yourself either as you perform a task (via the Macro Recorder), or you can create them directly using VBA.

VBA is an enhanced version of BASIC (Beginner's All-purpose Symbolic Instruction Code) that is similar to Microsoft's Visual BASIC 6. I have been coding in BASIC since the day my Dad brought home a ZX Spectrum from the Kingston-upon-Thames branch of Laskys back in 1983.

This article is based around Word 2013. However, the majority of the functions will be the same in Word 2010.

For those of you using any older version (for example, Word 2007), the macro code will generally be the same, although some of the commands may have altered – or worse, have been removed altogether!

A potted history of the BASIC behind Office

In Office 95, Microsoft added a programming option to enable people to tinker with it and to replace some of the duller and more repetitious tasks with code that just 'does it all for you' in an instant.

Word BASIC was the first variant and it was based on Microsoft's Quick BASIC.

Using simple commands, you could create Word macros to automate some of the more tedious tasks.

A simple example of the usual 'Hello World' test in Word BASIC:

```
Sub MAIN
      FormatFont .Name = “Arial”, .Points = 10
      Insert “Hello World”
End Sub
```

In Office 97, the Office suite was upgraded to a new and enhanced version of BASIC. This was similar to their main BASIC programming suite, Visual BASIC 6. It was called “Visual Basic for Applications”(VBA).

The code below is the same 'Hello World' example but this time using VBA:

```
Sub Main()
      With Selection.Font
              .Name = “Arial”
              .Size = 10
      End With
      Selection.TypeText Text:=”Hello World”
End Sub
```

While there are some slight differences between the two examples due to the commands used, they both inform the relevant versions of Word that you want the 'Hello World' text displayed in Arial, size 10.

Initial steps: switching on the Developer tab

By default, Word 2010 and 2013 both hide the **Developer** tab from the initial Ribbon bar. In order to use VBA, it needs to be on display.

1. On the **File** tab, choose **Options** to open the **Word Options** dialog box.
2. Click the **Customize Ribbon** option on the left-hand side.
3. On the right-hand side of the dialog box will be a list of available tabs. Select the **Developer** check box and click **OK**.

Word will now display the **Developer** tab. Within the Ribbon, you will be able to see the Visual Basic, Macros, Macro Security and Record Macro buttons.

Creating some target material

Before you create your macro, you need some test text to use with it. If you go into the main Word document display and type[1]:

```
=rand(10)
```

Press ENTER on the keyboard and 10 paragraphs of random text are created. You will use these paragraphs as the target material for your macro.

Save the result as a new document, for example, `MyMacroTest.docx`, creating a backup copy just in case.

Your first macro

Word (along with all other Microsoft Office applications) has a built-in Macro Recording option.

This feature will 'watch' what you do and record it step-by-step and then recreate these steps in VBA code.

The only downsides are that sometimes the Macro Recorder can:

1. 'Over-produce' some of the functionality, which, with some deft editing, you can cut down to size.
2. Not record everything you want to. Sometimes you have to code the functionality yourself.

This initial article will cover the creation of a simple macro that can be altered afterwards.

Start recording

As a simple example for your first macro, you are going to open up your test document, select all the text and then change the alignment to be right-aligned.

Open up the `MyMacroTest.docx` file you created earlier. On the **Developer** tab, click the **Record Macro** button.

The first dialog box that pops up will be one to ask what name you would like to call this macro.

By default, Word stores all of your macros inside the **Normal** template. However, for our example, you need to store your first macro inside the same document as your random text.

Enter '`MyFirstMacro`' as the name of your new macro in the **Macro name** field. Change the filename in the **Store macro in** drop-down box so that it shows `MyMacroTest.docx (document)` instead of `Normal` and then click **OK**.

Macro recording is enabled, and Word is now recording every action you perform.

Note: you **cannot** use the right-mouse button when recording a macro, so you will need to use the paragraph alignment button on the **Home** tab.

1. Select the whole document (for example, CTRL + A).
2. Change the alignment of the paragraphs to be right-aligned.

3. Go back to the **Developer** tab and click **Stop Recording**.

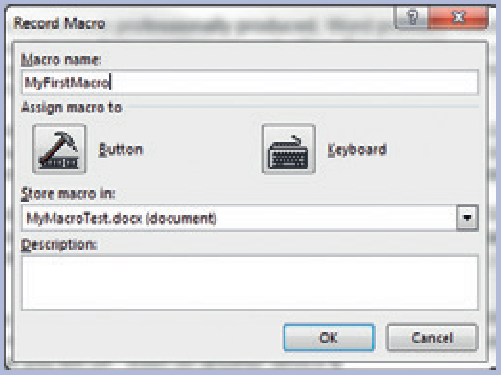
That's your first macro recorded. Now you can examine the code that the macro recorder has generated.

VBA Editor: the hidden window

A separate area within Word displays the generated VBA code. To view this area (the VBA Editor window), press the keys **ALT + F11**, or click the **Visual Basic** button on the **Developer** tab.

If you have two screens connected to your PC, it might open on a separate screen to where Word is currently running.

Within the VBA Editor (Figure 2) you will see several windows. The first one you need to look for is called **Project** and underneath will be a



Record Macro

Macro name: MyFirstMacro

Assign macro to

Store macro in: MyMacroTest.docx (document)

Description:

Figure 1. Record Macro dialog

tree structure showing the current files you have open, including your test document and the Normal template.

The code you have recorded will be stored under the sub-section **Modules**. Under this entry will be **New Macros**. If you double-click on **New Macros** you will be shown the code that was generated by the macro recorder.

These are the lines of VBA commands that represent the actions that have been recorded.

Viewing the code

It should look like this:

```
Sub MyFirstMacro()
'
' MyFirstMacro Macro
'
  Selection.WholeStory
  Selection.ParagraphFormat.Alignment = wdAlignParagraphRight

End Sub
```

Understanding the code

Your simple macro can be broken down into three separate parts:

1. The outside, which consists of the `Sub` and `End Sub` commands.
2. The comments, which must be preceded by an apostrophe.
3. The actual VBA code.

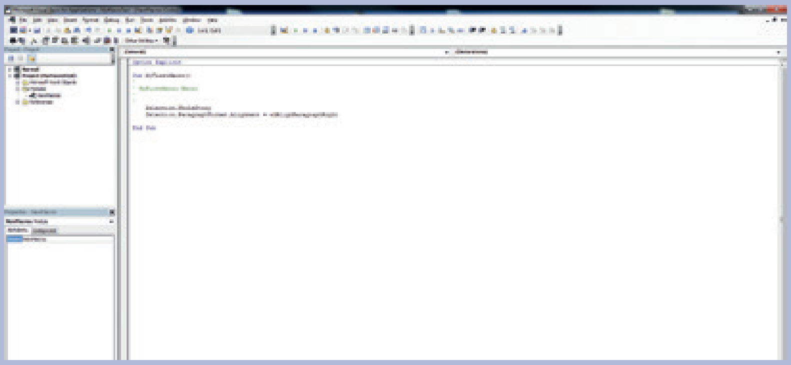


Figure 2. VBA Editor

Note: All your macros will begin with `Sub` and end with `End Sub`.

Word automatically adds the comments (displayed in green) when it converts your recorded actions into VBA code. It is entirely up to you if you want to leave them there, remove them or alter them to something more relevant.

Underneath the comments will be the actual VBA code. It consists of just two lines of code:

1. The first line ensures that the next bit of code operates on the whole document.

```
Selection.WholeStory
```

2. The second line uses the `ParagraphFormat.Alignment` command to change everything to be right-aligned.

```
Selection.ParagraphFormat.Alignment =
wdAlignParagraphRight
```

That's it! You have now created your first VBA macro.

Running your macro

There are several ways to run your macro, but I will cover the two easiest ones in this first article. There are other methods, but they are way out of scope for someone who is just starting out creating macros in VBA.

- Directly from the VBA Editor: click anywhere between the `Sub` and `End Sub` that are each end of your macro. Press F5 and your macro will be run against the text.
- Alternatively, click on the **Macros** button on the **Developer** tab. Find the name of your macro in the dialog box and select it, then click the **Run** button to start the macro.

Either of these two options will run your macro against the text in your document.

Saving your macro

The `.docx` format is one that does not allow macros to be stored, so you will need to save the document in the `.docm` format instead. Luckily, Word recognises if your current document has macros inside it and will warn you that it cannot save it in the original format. Change the format to 'Word Macro-Enabled Document' and save your file as `MyMacroTest.docm.`

Whenever you open this new file up, you will be able to access and tweak your recorded macro.

Editing your macro

Now that you have created your first macro, you can add some extra features to it. Instead of recording the additions via the Macro Recorder, you are going to type them in.

Open your file, the `.docm` variant, not the `.docx` one, and go into the VBA Editor.

After the last line of the code, before the `End Sub`, press Enter a few times so that you have some space to insert extra VBA commands.

1. Firstly you can increase the size of the text in your document to size 20-point text. Insert this command into the blank space:

```
Selection.Font.Size = 20
```

2. The next thing you can do is to change the background highlight to yellow, so you enter the following command:

```
Selection.Range.HighlightColorIndex
= wdYellow
```

3. Finally, in a mad rush of VBA power to your head, you need to make sure each word is displayed using bold. Therefore, your final command to add into the listing is:

```
Selection.Font.Bold = True
```

4. Close the VBA Editor. Now select all of the text and change the alignment back so that it is left-aligned. Save your updated document.

The changes to your text will not appear until you run the macro with the new changes.

Additional suggestions:

1. Note that "Selection" is the VBA object that refers to the currently selected text. It gives the context for the following method (action) or property.
2. Try typing "Selection." and see what you are offered: a huge list of all the methods and properties that might apply to selected text.
3. Start typing Font – see how the prompt helps you pick up a valid property – this now gives you a new object to work with (we call it "drilling down").
4. Type the . and see the list of valid properties for the font object and complete that line.
5. Read back – you've changed the size OF THE font OF THE selection.

Running your edited macro

If you use the same methods as described earlier to run your macro, you should end up with your test text displayed in bold, size 20-point font and with yellow highlighting behind. This will probably look a bit of a mess, but there is nothing stopping you changing the parameters of the code and re-running the macro.

If this does not happen and instead you are presented with a message box telling you that you have an error in your code, click the Debug option and you will be taken to the line with the error. Check that you have not made any typing errors, and fix them if you have.

Microsoft Trust Center

Years ago, in the late 90s, I was earning my crust as a helpdesk bod. I used to be sent to a national broadcaster based in London, who accepted scripts and whatnot in Word format from outside sources without really checking them. In those days, Word 2000's defences were quite weak and a particularly nasty Word virus spread throughout the entire organisation within a few hours!

The Trust Center was introduced in Office 2007 and it is Microsoft's answer to preventing self-copying VBA viruses from infecting your version of Word, and that of your other colleagues. The default settings are for any document that is opened is blocked from running any of the macros that might be present. The Trust Center's messages appear above the document in the same style of message that the Office Compatibility 'warning' does with the bright yellow background.

You have to click the 'Allow' button to unblock the macro code. You can change the level of security implemented via the **File>Options>Trust Center** configuration. Naturally, if you work in an organisation with a strict policy on macros, you might not be allowed to change this. Personally, mine is switched off completely, but you might not want that – or be allowed it!

Other examples you could try doing

Now that you have the hang of recording a simple macro, it is time to either dive in and add more steps to the macro that you have been editing and running.

Remember to reset your text to the standard format that you began with (use Word's Undo feature) as this helps you work out what is happening each time you change the macro code. There will be a new area on the ISTC website in the Resources section which will contain some hints, tips and new macros to either try and create, or go through step-by-step to help you understand how they were created.

What next?

In the follow-up to this article, I will be explaining how to enhance your macros by the use of message boxes and input boxes. The use of input boxes requires variables, so I will also be covering the multiple types of variable that you can use in VBA.

From there I will cover the use of loops to go through the elements (paragraphs, sections, headers/footers etc.) of your document.

After these are covered, who knows what functions of Word I might be requested to breakdown into VBA.

Getting in touch

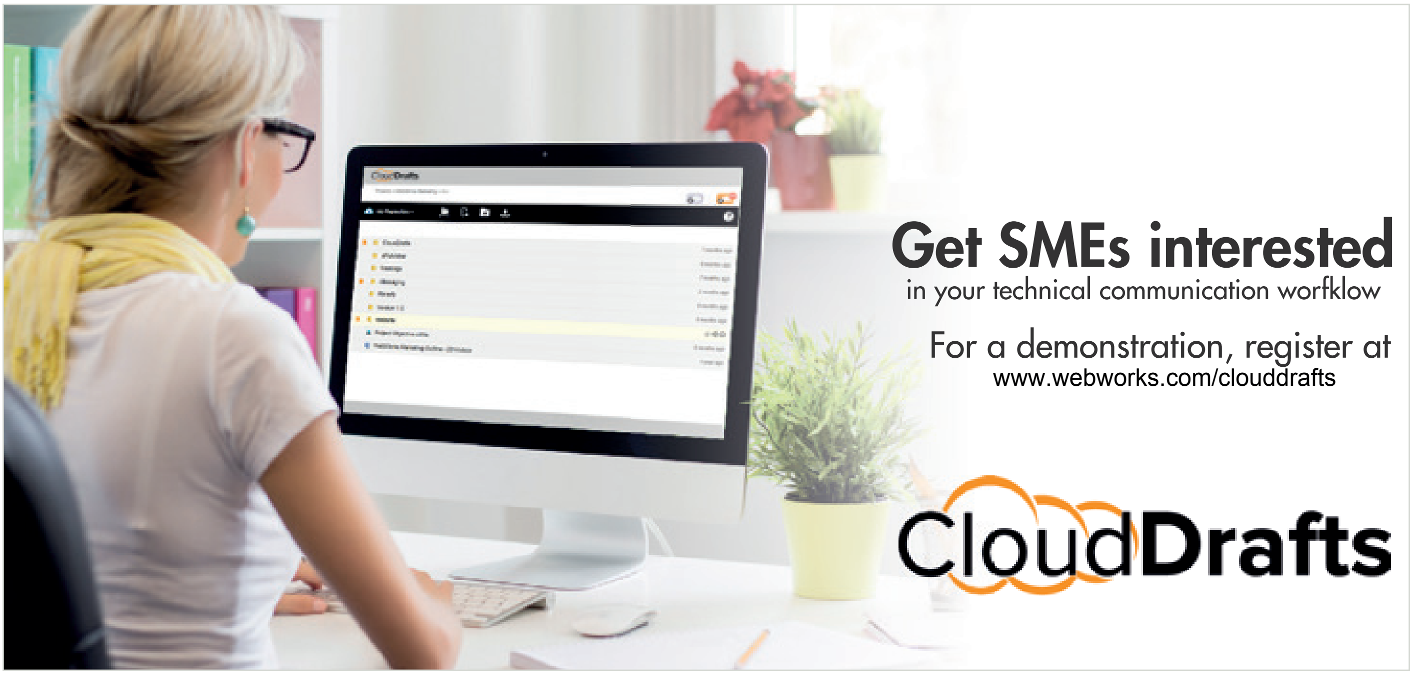
I am more than happy to lend a hand to budding VBA code creators. I recently joined the ISTC Yahoo group, so it is probably best to post your query in there first.

My Twitter ID is a shortened version of both 'Mugger Boot' and 'Mug Of Tea'. Two nicknames I picked up at school as I was the 'only northerner' in a south London comprehensive. I used to play football in Doc Martens boots and I apparently drank copious amounts of tea! C

Further reading

[1] Microsoft (2011) How to insert sample text into a document in Word https://support.microsoft.com/en-us/kb/212251 (accessed April 2015)

Mike Mee MISTC is is a technical author working at CDL in Stockport.
E: mike.mee@cdl.co.uk
T: @Mug_UK
W: http://mikestoolbox.weebly.com – my toolkit for Word 2010/2013 (the full source code is also available on the website).


Get SMEs interested
in your technical communication worfklow
For a demonstration, register at
www.webworks.com/clouddrafts
CloudDrafts

See what else we offer...

If you enjoyed this article, visit our website to see what else we do.

The Institute of Scientific and Technical Communicators is the largest UK body representing information development professionals, serving both our members and the wider technical communication community.

What the ISTC offers

Professional development and recognition

Resources and opportunities to develop and diversify skills, stay up to date with trends and technologies, and get recognition for achievements.

Our CPD (Continuous Professional Development) framework enables you to provide evidence of your learning in all its forms, and our Awards programme gives you the opportunity to showcase excellent work.

Communicator professional journal

Communicator is the ISTC's award-winning quarterly professional journal, covering the breadth of technical communications, offering in-depth articles, case studies, book and product reviews.

Now you've read a sample article, would you like to see more? The journal is free to our members and is also available on subscription.

ISTC Community

The ISTC offers opportunities to network, exchange expertise, and stay in touch with the UK technical communication industry – through a range of online groups, local events, and InfoPlus+ (our monthly newsletter).

You can find us on LinkedIn, Eventbrite, YouTube and Twitter (@ISTC_org).

Technical Communication UK

The ISTC hosts Technical Communication UK, the annual conference that aims to meet the needs of technical communicators, their managers and clients, from every corner of the industry.

Open to all, visit www.technicalcommunicationuk.com for the latest news.

ISTC Resources

The ISTC offers access to a range of resources, including our own books, various templates, articles summarising key technical communication issues and discounted British Standards publications.