Word Macro Examples & VBA Tutorial

https://www.automateexcel.com/vba/word/macro-examples

IN THIS ARTICLE

SELECT / GO TO

Description	VBA Code
Backspace	Selection.TypeBackspace
Select Entire Document	Selection.HomeKey Unit:=wdStory Selection.Extend
Copy	Selection.Copy
Delete	Selection.Delete Unit:=wdCharacter, Count:=1
Insert After	Selection.InsertAfter "text"
Beginning of Line	Selection.HomeKey Unit:=wdLine
End of Line	Selection.EndKey Unit:=wdLine
Paste	Selection.Paste
Select All	Selection.WholeStory
Select Entire Line	Selection.EndKey Unit:=wdLine, Extend:=wdExtend
Move Up Paragraph	Selection.MoveUp Unit:=wdParagraph, Count:=1
Move Right One Character	Selection.MoveRight Unit:=wdCharacter, Count:=1
Move Right One Cell in Table	Selection.MoveRight Unit:=wdCell
Go To Start of Doc	Selection.HomeKey Unit:=wdStory
Go To End of Doc	Selection.EndKey Unit:=wdStory
Go To Page 1	Selection.GoTo What:=wdGoToPage, Which:=wdGoToNext, Name:="1"
Go To Top of Page	Selection.GoTo What:=wdGoToBookmark, Name:="\Page" Selection.MoveLeft Unit:=wdCharacter, Count:=1

COLUMNS

Description	VBA Code
Save As	Documents("Example.doc").SaveAs ("C:\Example\Example.doc")
Save	Documents("Example.doc").Save
Protect	Documents("Example.doc").Protect Password:="password"
Unprotect	Documents("Example.doc").UnProtect Password:="password"
Number of Pages	Dim varNumberPages as Variant varNumberPages = _ ActiveDocument.Content.Information(w-dActiveEndAdjustedPageNumber)
Print	Documents("Example.doc").Print

LOOPS

Description	VBA Code
Do Until End of Doc	Do Until ActiveDocument.Bookmarks("\Sel") = ActiveDocument.Bookmarks("\EndOfDoc") 'Do Something Sub
For Each Doc in Docs	Dim doc As Document ForEach doc In Documents 'Do Something Next doc
Loop Through Paragraphs	Sub through Paragraphs Dim i As Long, iParCount As Long iParCount = ActiveDocument.Paragraphs. CountFori = 1 To iParCount ActiveDocument.Paragraphs(i).Alignment = wdAlignParagraphLeft Next i

BOOKMARKS

Description	VBA Code
Add	With ActiveDocument.Bookmarks .Add Range:=Selection.Range, Name:="Name" .DefaultSorting = wdSortByName .ShowHidden = False End With
Count	Dim n as Integer n = ActiveDocument.Bookmarks.Count
Delete	ActiveDocument.Bookmarks("BookmarkName").Delete

Exists?	If ActiveDocument.Bookmarks.Exists("BookmarkName") = True then
	'Do something
	End If

Go To	Selection.GoTo What:=wdGoToBookmark, Name:="BookmarkName"
Select	ActiveDocument.Bookmarks("BookmarkName").Select
Replace Text	Selection.GoTo What:=wdGoToBookmark, Name:="BookmarkName" Selection.Delete Unit:=wdCharacter, Count:=1 Selection.InsertAfter "New Text" ActiveDocument.Bookmarks.Add Range:=Selection.Range, _ Name:="BookmarkName"

PARAGRAPH

Description	VBA Code
KeepLines Together	Selection.ParagraphFormat.KeepTogether = True
KeepWithNext	Selection.ParagraphFormat.KeepWithNext = True
Space After	Selection.ParagraphFormat.SpaceAfter = 12
Space Before	Selection.ParagraphFormat.SpaceBefore = 0
Align Center	Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
Align Right	Selection.ParagraphFormat.Alignment = wdAlignParagraphRight
Align Left	Selection.ParagraphFormat.Alignment = wdAlignParagraphLeft
Left Indent	Selection.ParagraphFormat.LeftIndent = InchesToPoints(3.75)
Right Indent	Selection.ParagraphFormat.RightIndent = InchesToPoints(1)
Line Spacing	With Selection.ParagraphFormat .LineSpacingRule = wdLineSpaceExactly .LineSpacing = 12 End With
Loop Through All Paragraphs	Sub through Paragraphs Dim i As Long, iParCount As Long iParCount = ActiveDocument.Paragraphs. CountFori = 1 To iParCount ActiveDocument.Paragraphs(i).Alignment = wdAlignParagraphLeft Next i

DOCUMENT

Description	VBA Code
Activate	Documents("Example.doc").Activate
Add to Variable	Dim doc As Document Set doc = Documents.Add
Add	Documents.Add
Add (From Another Doc)	Documents.Add Template:="C:\Forms\FormDoc.doc", _ NewTemplate:=False
Close	Documents("Example.doc").Close
Close – Save Changes	Documents("Example.doc").Close SaveChanges:=wdSaveChanges
Close – Do Not Save	Documents("Example.doc").Close SaveChanges:=wdDoNotSaveChanges
Close – Prompt to Save	Documents("Example.doc").Close SaveChanges:=wdPromptToSaveChanges

FONT

Description	VBA Code
Size	Selection.Font.Size = 12
Bold	Selection.Font.Bold = True
Italics	Selection.Font.Italic = True
Underline	Selection.Font.Underline = wdUnderlineSingle
All Caps	Selection.Font.AllCaps = True
Color	Selection.Font.TextColor = vbRed
Subscript	Selection.Font.Subscript = True
SuperScript	Selection.Font.Superscript = True
Highlight Color	Selection.Range.HighlightColorIndex = wdYellow
Style	Selection.Style = ActiveDocument.Styles("Normal")

INSERT

Description	VBA Code
Insert AutoText	Selection.TypeText Text:="a3" Selection.Range.InsertAutoText
Insert Date Code	
Insert File	Selection.InsertFile ("C:\Docs\Something.doc")
Insert Page Break	Selection.InsertBreak Type:=wdPageBreak
Insert Paragraph Symbol	Selection.TypeText Text:=Chr$(182)
Insert Tab	Selection.TypeText Text:=vbTab
Insert Text	Selection.TypeText Text:="Any Text"
Insert Type Paragraph	Selection.TypeParagraph
Insert Paragraph	Selection.InsertParagraph

Word VBA Macro Tutorial

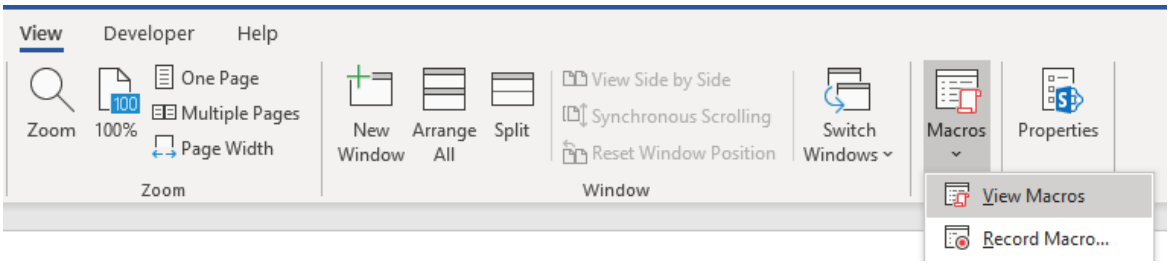
This is a tutorial for using VBA with Microsoft Word. This tutorial will teach you how to write a simple Macro and interact with Documents, Ranges, Selections, and Paragraphs.

Note: If you're brand new to Macros / VBA you might also find this article useful: How to write VBA Macros from Scratch.

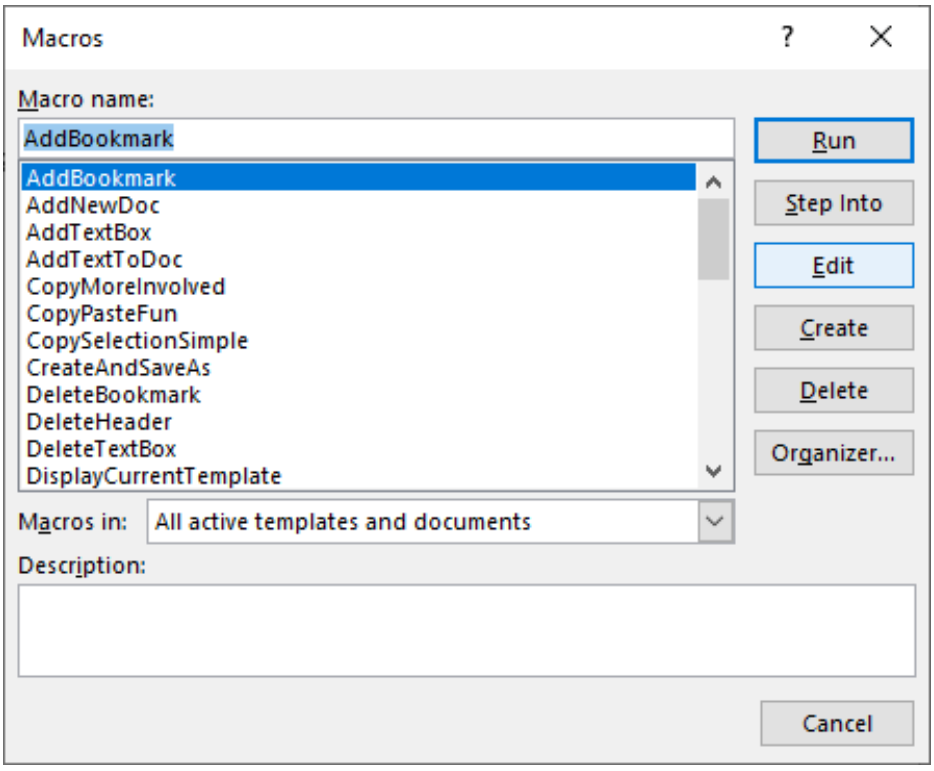
VBA is the programming language used to automate Microsoft Office programs including Word, Excel, Outlook, *PowerPoint*, and Access.

Macros are blocks of VBA code that perform specific tasks.

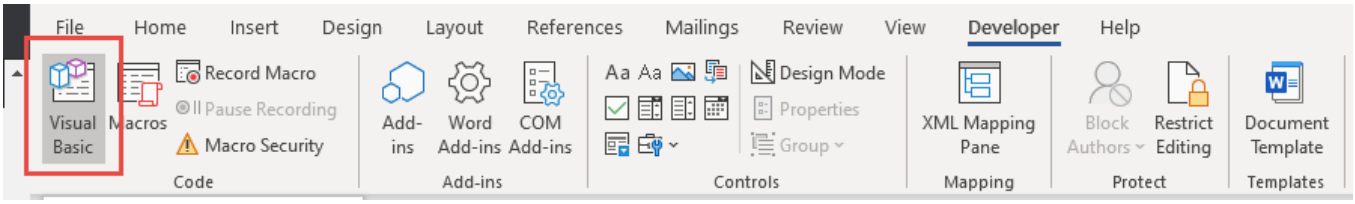
When you *Record a Macro,* Word will write VBA code into a Macro, allowing you to repeat your actions. You can see a list of all available Macros from View > Macros.



After recording a Macro, you will be able to *edit the Macro* from the Macro List:



When you click **Edit**, you open the **VBA Editor**. Using the VBA Editor you can edit recorded Macros or write a Word Macro from scratch. To access the VBA Editor use the shortcut **ALT + F11** or click **Visual Basic** from the **Developer Ribbon.**



Simple Word Macro Example

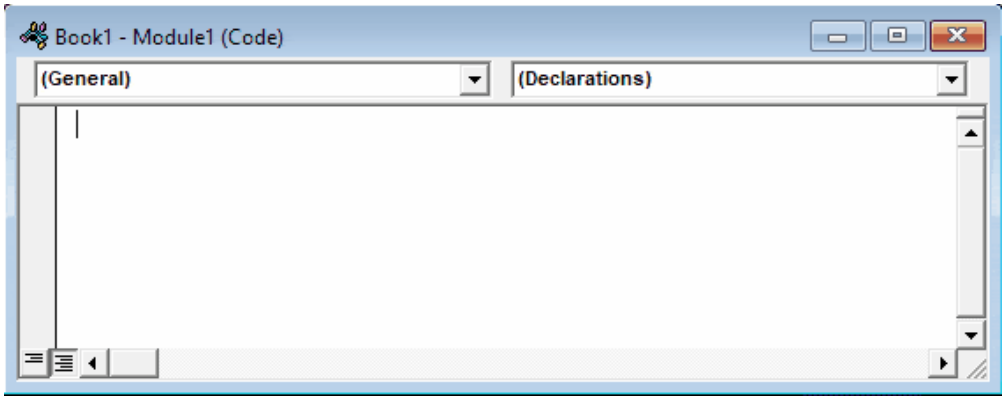
This is a simple example of a Word VBA Macro. It performs the following tasks:

- Opens a Word Document
- Writes to Document
- Closes and Saves the Word Document.

```
Sub WordMacroExample()

    'Open Doc & Assign to Variable
    Dim oDoc As Document
    Set oDoc = Documents.Open("c:\Users\someone\NewDocument.docx")

    'Write To Doc
    Selection.TypeText "www.automateexcel.com"
    Selection.TypeParagraph

    'Save and Close Doc
    oDoc.Save
    oDoc.Close

End Sub
```

Word Macro Basics

All VBA code must be stored within procedures like this. To create a procedure in VBA type "Sub WordMacroExample" (Where "WordMacroExample" is your desired Macro name) and press ENTER. VBA will automatically add the parenthesis and End Sub.



Word Document Object

When interacting with Microsoft Word in VBA, you will frequently reference Word "Objects". The most common objects are:

Application Object – Microsoft Word itself

Document Object – A Word document

Range Object – A part of a Word document

Selection Object – A selected range or cursor location.

Application

Application is the "top-level" object. All other objects in Word can be reached through it.

In addition to accessing other Word objects, there are "application-level" settings that can be applied:

```
Application.Options.AllowDragAndDrop = True
```

This is an example of accessing the "Selection" of "Windows(1)" with in the Application:

```
Application.Windows(1).Selection.Characters.Count
```

However, the most common Word objects can be accessed directly, without typing the full hierarchy. So instead, you can (and should) just type:

Documents

ActiveDocument

Often, you will have two or more documents opened in Word and you will need specify which specific Word Document to interact with. One way to specify which document is to use ActiveDocument. For example:

```
ActiveDocument.PrintOut
```

...would print the ActiveDocument. The ActiveDocument is the document in Word which "has focus"

To switch the ActiveDocument, use the Activate command:

```
Documents("Example.docx").Activate
```

ThisDocument

Instead of using ActiveDocument to reference the active document, you can use ThisDocument to reference the document where the macro is stored. ThisDocument will never change.

```
ThisDocument.PrintOut
```

Document Variables

However, for more complicated macros, it can be hard to keep track of the Active Document. It can also be frustrating to switch back and forth between documents.

Instead, you can use Document variables.

This macro will assign the ActiveDocument to a variable and then print the document using the variable:

```
Sub VarExample()
    Dim oDoc As Document
    Set oDoc = ActiveDocument
    oDoc.PrintOut
End Sub
```

Document Methods

Open Document

To Open a Word Document:

Documents.Open “c:\Users\SomeOne\Desktop\Test PM.docx”
We recommend always assigning a Document to a variable upon opening it:

```
Dim oDoc as Document
Set oDoc = Documents.Open(“c:\Users\SomeOne\Desktop\Test PM.docx”)
```

Create New Document

To create a new Word Document:

```
Documents.Add
```

We can instruct Word to create a new doc based on some template:

```
Documents.Add Template:=”C:\Program Files\Microsoft Office\Templates\MyTemplate.dotx”
```

As always, it is useful and huge problem saver to assign document to variable upon creating or opening:

```
Dim oDoc as Document
Set oDoc = Documents.Add (Template:=”C:\Program Files\Microsoft Office\Templates
MyTemplate.dotx”)
```

Save Document

To save a document:

```
ActiveDocument.Save
```

or SaveAs:

```
ActiveDocument.SaveAs FileName:= c:\Users\SomeOne\Desktop\test2.docx”,
FileFormat:=wdFormatDocument
```

Close Document

To close a Document and save changes:

```
ActiveDocument.Close wdSaveChanges
```

or without saving changes:

```
ActiveDocument.Close wdDoNotSaveChanges
```

Print Document

This will print the active Document:

```
ActiveDocument.PrintOut
```

Range, Selection, Paragraphs

Range and **Selection** are probably the most important objects in Word VBA, certainly the most used.

Range refers to some portion of document, usually, but not necessarily, text.

Selection refers to selected text (or other object like pictures) or, if nothing is selected, an insertion point.

Paragraphs represent paragraphs in document. Its less important than it sounds, because you can't directly access paragraph text (you need to access particular paragraph range to make modifications).

Range

Range can be any part of document, including entire document:

```
Dim oRange As Range
Set oRange = ActiveDocument.Content
```

or it can be small as one character.

Another example, this range would refer to first word in document:

```
Dim oRange As Range
Set oRange = ActiveDocument.Range.Words(1)
```

Usually, you would want to get range which refers to specific part of document and then modify it.

In the following example we will make the first word of second paragraph bold:

```
Dim oRange As Range
Set oRange = ActiveDocument.Paragraphs(2).Range.Words(1)
oRange.Bold = True
```

Set Range Text

To set the text value of a Range:

```
Dim oRange As Range
Set oRange = ActiveDocument.Paragraphs(2).Range.Words(1)
oRange.Text = "Hello "
```

(Tip: Note the space after "Hello". Because word object includes space after word, with just "hello" we would get "Hellonext word")

There are hundreds of things which you can do with ranges. Just a few examples (these assume you are already made object variable oRange referring to range of interest):

Change font

```
oRange.Font.Name = "Arial"
```

Display in message box number of characters in particular range

```
MsgBox oRange.Characters.Count
```

Insert some text before it

```
oRange.InsertBefore "this is inserted text "
```

Add a footnote to range

```
ActiveDocument.Footnotes.Add Range:=oRange, _
Text:="Read more at automateexcel.com."
```

Copy it to clipboard

```
oRange.Copy
```

Often you need *to* change *to* what *is* particular range referring. So you can start it's start *and end*

```
oRange.Start = 5
oRange.End = 50
```

After above code, oRange would refer to text starting with fifth and ending with 50th character in document.

Selection

Selection is even more widely used than Range, because it is easier to work with Selections than Ranges, IF your macro ONLY interacts with the ActiveDocument.

First select the desired part of your document. For example select the second paragraph in active document:

```
ActiveDocument.Paragraphs(2).Range.Select
```

Then you can use the Selection Object to type some text:

```
Selection.TypeText "Some text"
```

We can type some paragraphs bellow "Some text":

```
Selection.TypeText "Some text"
Selection.TypeParagraph
```

Often, it's necessary to know if some text is selected or we have just a insertion point:

```
If Selection.Type <> wdSelectionIP Then
    Selection.Font.Bold = True
Else
    MsgBox "You need to select some text."
End If
```

When working with Selection object we want to place insertion point to particular place, and issue commands starting from this point.

Beginning of document:

```
Selection.HomeKey Unit:=wdStory, Extend:=wdMove
```

Beginning of current line:

```
Selection.HomeKey Unit:=wdLine, Extend:=wdMove
```

The Extend parameter wdMove moves the insertion point. Instead, you could use wdExtend which will select all text between the current insertion point.

```
Selection.HomeKey Unit:=wdLine, Extend:=wdExtend
```

Move Selection

The most useful method for changing position of insertion point is Move. To move Selection two characters forward:

```
Selection.Move Unit:=wdCharacter, Count:=2
```

to move it backwards, use negative number for Count parameter:

```
Selection.Move Unit:=wdCharacter, Count:=-2
```

Unit parameter can be wdCharacter, wdWord, wdLine, or more (use Word VBA help to see others).

To move words instead:

```
Selection.Move unit:=wdWord, Count:=2
```

Selection is easier to work with (compared to ranges) because it is like a robot using Word, mimicking human user. Where Insertion point is – some action would take place. But, this means that you must take care where insertion point is! This is not easy after many steps in code. Otherwise, Word would change text in not desired place.

In the case you need some property or method not available in Selection object you can always easily obtain range associated with selection:

```
Set oRange = Selection.Range
```

TIP: Using Selection is often easier than using ranges, but also it's way slower (important when you deal with big documents)

Paragraphs

You can't directly use Paragraphs object to change text:

```
ActiveDocument.Paragraphs(1).Text = "No, it wouldn't work"
```

Above wouldn't work (actually it will throw an error). You need to first obtain range associated with particular paragraph:

```
ActiveDocument.Paragraphs(1).Range.Text = "It works now :)"
```

But you can directly change its style:

```
ActiveDocument.Paragraphs(1).Style = "Normal"
```

or change its paragraph level formatting:

```
ActiveDocument.Paragraphs(1).LeftIndent = 10
```

or maybe you want to keep this paragraph on the same line with next paragraph:

```
ActiveDocument.Paragraphs(1).KeepWithNext = True
```

Make paragraph centered:

```
ActiveDocument.Paragraphs(1).Alignment = wdAlignParagraphCenter
```

It is VERY useful to assign a particular paragraph to object variable. If we assign particular paragraph to variable we don't have to worry if the first paragraph becomes the second because we inserted one paragraph before it:

```
dim oPara as Paragraph
Set oPara = Selection.Paragraphs(1) 'here we assign first paragraph of current selection to
variable
```

Here is an example where we insert a paragraph above the first paragraph, but we can still reference the old first paragraph because it was assigned to a variable:

```
Sub ParagraphExample()
    Dim oPara As Paragraph
    Set oPara = ActiveDocument.Paragraphs(1)
    MsgBox oPara.Range.Text
    oPara.Range.InsertParagraphBefore 'Insert Paragraph
    MsgBox oPara.Range.Text
End Sub
```

Paragraph object is very frequently used in loops:

```
Sub LoopThroughParagraphs()

    Dim oPara As Paragraph
    For Each oPara In ActiveDocument.Paragraphs
        'do something with it. We will just display
        'paragraph text if its style is "Heading 4"
        If oPara.Style = "Heading 4" Then
            MsgBox oPara.Range.Text
        End If
    Next oPara

End Sub
```

Word VBA Tutorial Conclusion

This tutorial covered the basics of Word VBA. If you're new to VBA, you should also review our general VBA Tutorial to learn more about Variables, Loops, MessageBoxes, Settings, Conditional Logic and much more.

Word Macro Examples

When interacting with Microsoft Word in VBA, you will frequently reference Word "Objects". The most common objects are:

Application Object – Microsoft Word itself

Document Object – A Word document

Range Object – A part of a Word document

Selection Object – A selected range or cursor location.

WORD MACRO EXAMPLES
Templates
Add New Documents
Count Words in Selection
TextBoxes
SaveAs PDF
Bookmarks
Tables
Find and Find and Replace
Open Documents