
# TIPS & TRICKS

## OFFICE XP MACROS


# TIPS & TRICKS

Compiled by

MUHAMMAD AJMAL BEIG NAZ

*******

# Table of Contents

# OFFICE XP MACROS

## ABOUT MACROS

If you perform a task repeatedly in Microsoft Office, you can automate the task by using a macro. A macro is a series of Word commands and instructions that you group together as a single command to accomplish a task automatically. Here are some typical uses for macros:

- To speed up routine editing and formatting
- To combine multiple commands; for example, inserting a table with a specific size and borders, and with a specific number of rows and columns
- To make an option in a dialog box more accessible
- To automate a complex series of tasks

Macros can be created by the macro recorder or using the Visual Basic Editor. For quick access to your macro, you can assign it to a **Toolbar**, a **Menu**, or **Shortcut Keys**. Running the macro is as simple as clicking the toolbar button or menu command or pressing the shortcut keys.

If you give a new macro the same name as an existing built-in command, the new macro actions will replace the existing actions. For example, if you record a new macro and name it **FileClose**, it becomes attached to the **Close** command. When you choose the **Close** command, it performs the new actions you recorded.

**IMPORTANT:**
**All macro codes are given in a `different font` for easy recognition. You must type the code exactly in the same manner and punctuation to create a successful macro.**

## MICROSOFT OUTLOOK XP

### Create a Macro in Outlook 2002 to delete junk e-mails

To create the macro, you must first have added at least one junk e-mail message sender name to your junk e-mail message senders list.

1. On the **Tools** menu, point to **Macro** and click **Macros**.
2. In the **Macro name** box, type **MultipleJunkEMailSenders**.
3. Click **Create**.
4. Find the code that looks like this:

```
Public Sub MultipleJunkEMailSenders()
End Sub
```

## Modify the code so that it looks like this (Exactly the same):

```
Sub MultipleJunkEMailSenders()
```

> **Note:** This code works only with Microsoft Outlook 2002. The list of your junk e-mail message senders is stored in a file on your hard drive named "Junk Senders.txt".

Replace this string of text with the path to your Junk Senders.txt file.

```
   Dim objExplorer As Outlook.Explorer
   Dim objMailItem As Outlook.MailItem
   Dim intItem As Integer
   Dim objFSO As Scripting.FileSystemObject
   Dim objTextStream As Scripting.TextStream

   Set objFSO = New Scripting.FileSystemObject

   If MsgBox(Prompt:="Are you sure you want to " & _
      "add all of the selected senders to your " & _
      "junk e-mail sender list and then " & _
      "delete all of the selected e-mail " & _
      "messages? Caution: This action is not " & _
      "easily reversible!", Buttons:=vbYesNo) = vbYes Then

   If objFSO.FileExists(FileSpec:=JUNK_SENDERS_FILE) = False
   Then
   Else
   Set objTextStream = objFSO.OpenTextFile _
   FileName:=JUNK_SENDERS_FILE, IOMode:=ForAppending)
   End If

   Set objExplorer = Application.ActiveExplorer
   For Each objMailItem In objExplorer.Selection

objMailItem.Delete

Next objMailItem
MsgBox Prompt:="All selected senders have been added " & _
   "to your junk e-mail senders list and " & _
   "all selected e-mails have been deleted."

End If
End Sub
```

5. On the **Tools** menu, click **References**.
6. In the **Available References** list, check the **Microsoft Scripting Runtime** box.
7. Click **OK**.
8. On the **File** menu, click **Close and Return to Microsoft Outlook**.
9. Select one or more junk e-mail messages in your Inbox.
10. On the **Tools** menu, point to **Macro**, and then click **Macros**.
11. Click **MultipleJunkEMailSenders**, and then click **Run**.
12. Click **Yes**.

13. If the message **A program is trying to access e-mail addresses you have stored in Outlook. Do you want to allow this?** appears, click **Yes**.
14. Click **OK**.

The junk e-mail messages are moved to your Deleted Items folder, the sender names are added to your junk e-mail message senders list, and future e-mail messages from these senders are moved directly to your Deleted Items folder. Your junk e-mail message problems are now under control!

## Add a Last Contacted Date to Your Contacts

How can you record the date you last spoke with or e-mailed someone in your Outlook 2002 Contacts list? We will learn how to create an Outlook macro that adds a last contacted date to one of your contacts.

Many people occasionally use the **Journal** feature in Microsoft Outlook® 2002 to associate **e-mail messages**, **meeting items**, **task items**, and **Office files** with their **contacts**. Some people create individual Journal items and associate them with individual contacts as well. However, sometimes you want to quickly make a note of when you last contacted someone without taking the time to create a complete Journal item. The following steps enable you to quickly make a tick mark that notes when you last spoke with or e-mailed one of your Outlook contacts. To do so, follow these steps:

## Create a macro to add a last contacted date to a contact

1. From inside Outlook, open the Microsoft Office Visual Basic® Editor. (On the **Tools** menu, point to **Macro** and then click **Visual Basic Editor**.)
2. If the **Project Explorer** window is not visible, on the **View** menu, click **Project Explorer**.
3. In the **Project Explorer** window, expand the folder tree to open the **Microsoft Outlook Objects** folder and then double-click **ThisOutlookSession**.
4. Type the following code into the **Code** window (the large, blank window on the right side of the screen):

```
Public Sub LastContacted()

' Purpose: Adds a "Last Contacted" property to
' the active contact item.
' Note: You must select a contact item before
' running this macro.

  Dim olApp As Outlook.Application
  Dim oSel As Outlook.Selection
  Dim oContact As Outlook.ContactItem
  Dim oPrp As Outlook.ItemProperty
  Dim bFieldExists As Boolean

    Const FIELD_NAME As String = "Last Contacted"
    On Error GoTo LastContacted_Err
```

```
    Set olApp = Outlook.Application
    Set oSel = olApp.ActiveExplorer.Selection

'Only one item can be selected.
If oSel.Count > 1 Then
MsgBox "You selected more than one item. " & _
"Select only one item and try again."
Exit Sub

End If
' Selected item must be a contact item.
Set oContact = oSel.Item(1)

' Check for a custom "Last Contacted" field.
For Each oPrp In oContact.ItemProperties

If oPrp.Name = FIELD_NAME Then

bFieldExists = True
oPrp.Value = Now
MsgBox FIELD_NAME & " property updated."

End If

Next oPrp
' Create "Last Contacted" field if it doesn't already exist.
If bFieldExists = False Then

Set oPrp = _
oContact.ItemProperties.Add(FIELD_NAME, olDateTime)

oPrp.Value = Now
MsgBox FIELD_NAME & " property created and updated."

End If

oContact.Save

LastContacted_End:

Set oPrp = Nothing
Set oContact = Nothing
Set oSel = Nothing
Set olApp = Nothing

Exit Sub

LastContacted_Err:

Select Case Err.Number

Case 13 ' Not a contact item.

MsgBox "You can only run this macro on contact " & _
"items. Select one contact item and try again."
```

```
Case Else    ' Unanticipated error.

MsgBox "Error " & Err.Number & " in LastContacted " & _
"subroutine: " & Err.Description
End Select
Resume LastContacted_Err

End Sub
```

5. On the **File** menu, click **Close and Return to Microsoft Outlook**.

## To run the macro

1. Open your Outlook **Contacts** folder.
2. Select a single contact item (do not open the contact item).
3. On the **Tools** menu, point to **Macro** and then click **Macros**.
4. Click **LastContacted**, and then click **Run**.
5. Click **OK** when the dialog box appears.
6. Open the selected Outlook contact item.
7. On the **All Fields** tab, in the **Select from** list, click **User-defined fields in this item**.

The **Last Contacted** field appears along with the date and time you ran the **LastContacted** macro. Be sure to run this macro at or near the time, you make your call or send e-mail, so that the date and time entered by the macro are correct.

# MICROSOFT WORD XP

## Create Macro that replaces the CTRL+V Shortcut

First, let us create the macro, which will allow us to paste unformatted text from the Clipboard into your documents. If you have never created a macro before, this one provides both a good learning experience and a useful function.

## To create the macro to cut and paste without formatting

1. Start Word.
2. On the **Tools** menu, point to **Macro**, and then click **Macros** to display the **Macros** dialog box.
3. In the **Macro name** box, type **PasteUnformattedText**.
4. Make sure that **All active templates and documents** is displayed in the **Macros in** list, and then click **Create**. The Microsoft Visual Basic® Editor is displayed.
5. Directly above the `End Sub` statement in the `PasteUnformattedText` subroutine, type the following line of code:
   `Selection.PasteSpecial DataType:=wdPasteText`
6. On the **File** menu, click **Close and Return to Microsoft Word**.

Now you need to instruct Word to run the `PasteUnformattedText` macro each time you press the **CTRL+V** keyboard shortcut.

## To run the macro each time you press CTRL+V

1. On the **Tools** menu, click **Customize**.
2. Click the **Keyboard** button.
3. Make sure the **Save changes in** box displays **Normal.dot**.
4. In the **Categories** list, click **Macros**.
5. In the **Macros** list, click **PasteUnformattedText**.
6. Click in the **Press new shortcut key** box, press and hold the **CTRL** key, and then press and hold the V key at the same time. The **Press new shortcut key** box displays **Ctrl+V**.
7. Click **Assign**. Click **Close**, and then click **Close** again.

That's it! Now, every time you press **CTRL+V**, Word will paste text from the Clipboard without the formatting. If you ever want to return the **CTRL+V** keyboard shortcut to its default behavior, it's very easy to do.

## To return the CTRL+V shortcut to its default behavior

1. On the **Tools** menu, click **Customize**.
2. Click the **Keyboard** button.
3. Make sure the **Save changes in** box displays **Normal.dot**.
4. In the **Categories** list, click **Macros**.
5. In the **Macros** list, click **PasteUnformattedText**.
6. In the **Current keys** box, click **Ctrl+V** and then click **Remove**.
7. Click **Close**, and then click **Close** again.

The **CTRL+V** keyboard shortcut now pastes text and any formatting from the Clipboard by default. However, the `PasteUnformattedText` macro is still available if you ever want to use it again.

## Macro that turn Line Numbering On or Off Automatically

You can create a macro to turn line numbering on or off automatically, so that you don't have to adjust your page setup each time. If line numbers are not being displayed, the macro can add them; or, if line numbers are already being displayed, the same macro can remove them.

1. Start Word if it is not already running.
2. On the **Tools** menu, point to **Macro** and then click **Macros**.
3. In the **Macro name** box, type **ToggleLineNumbering**.
4. In the **Macros in** list, select **Normal.dot (global template)**.
5. Click **Create**. The Microsoft Visual Basic® Editor appears.
6. Between the lines of code Sub ToggleLineNumbering() and End Sub, type the following code:

```
With ActiveDocument.PageSetup.LineNumbering
If .Active = False Then
.Active = True
.StartingNumber = 1
.CountBy = 1
.RestartMode = wdRestartContinuous
Else
.Active = False
End If
End With
```

7. On the **File** menu of the Visual Basic Editor, click **Close and Return to Microsoft Word**.
8. On the **Tools** menu, point to **Macro** and then click **Macros**.
9. In the list of macros, click **ToggleLineNumbering**, and then click **Run**. If line numbers were not visible, they are now added; if line numbers were visible, they are now removed.

## Bonus Tips:

- To start line numbering at a number other than 1, you can change the number `1` in the line of code `.StartingNumber = 1` .
- To change line numbers from multiples of one to different multiples, you can change the number `1` in the line of code `.CountBy = 1` .
- To change the line numbering behavior, you can change the word `wdRestartContinuous` in the line of code `.RestartMode = wdRestartContinuous` :
  - To restart line numbering after each page, change the word to `wdRestartPage`.
  - To restart line numbering after each section, change the word to `wdRestartSection`.

## Display Word Counts in the Title Bar

This macro allows you to display the number of words, characters (not including spaces), and paragraphs in the title bar of an open Microsoft Word 2002 document. This is a simple replacement of the current Word Count command on the Tools menu.

**To display word, characters, and paragraph counts in the title bar, follow these steps:**

1. Start Word.
2. Press ALT+F8. Or, on the **Tools** menu, point to **Macro** and then click **Macros**.
3. In the **Macros in** box, select **Normal.dot (global template)**.
4. In the **Macro name** box, type **ShowDocStats** and then click **Create**.
5. Find the code that looks like this:

```
Sub ShowDocStats()
'
' ShowDocStats Macro
' Macro created [date] by [author]
'
End Sub
```

**Make this code look like the following by typing in the six new lines in the middle:**

```
Sub ShowDocStats()
'
' ShowDocStats Macro
' Macro created [date] by [author]
'
    With ActiveDocument.ActiveWindow
        .Caption = .Document.Name & " / " & _

.Document.BuiltInDocumentProperties(wdPropertyWords) & "
Words, " & _

.Document.BuiltInDocumentProperties(wdPropertyCharacters)
& " Characters, " & _

.Document.BuiltInDocumentProperties(wdPropertyParas) & "
Paragraphs"
    End With
End Sub
```

6. Press ALT+F11. Or, click **Return to Microsoft Word** on the **Standard** toolbar.
7. Open the **Macros** dialog box again (as in steps 2 through 4) and click **Run**. You should see the number of words, characters (without spaces), and paragraphs in the title bar of the Word document that is currently open and visible.

## To Protect a Word Document using a Macro

1. In Word, on the **Tools** menu, point to **Macro** and then click **Macros**.
2. In the **Macro name** box, type **SetWordOpenPassword**.
3. In the **Macros in** list, select **Normal.dot (global template)**.
4. Click **Create**. The Microsoft Visual Basic® Editor opens.
5. Between the code `Sub SetWordOpenPassword()` and `End Sub`, type the following code:

```
Dim strPassword As String

strPassword = InputBox(Prompt:="Type " & _
   "a case-sensitive password that " & _
   "must be used to open this document " & _
   "from now on, and then click OK.")

ActiveDocument.Password = strPassword

MsgBox Prompt:="Open password set to '" & _
strPassword & "'."
```

6. Click anywhere inside the macro code, and then on the **Run** menu, click **Run Sub/UserForm**.
7. Enter your password, click **OK**, and then save and close the document. When you open it again, you will be prompted for the password.

If you want to protect the document from being modified rather than opened, repeat the previous steps but use the **SetWordWritePassword** macro. Here's the code:

```
Dim strPassword As String

strPassword = InputBox(Prompt:="Type " & _
   "a case-sensitive password that " & _
   "must be used to make changes to " & _
   "this document from now on, " & _
   "and then click OK.")

ActiveDocument.WritePassword = strPassword

MsgBox Prompt:="Write password set to '" & _
strPassword & "'."
```

# MICROSOFT EXCEL XP

## Protect an Excel 2002 workbook by using a Macro

You can also password-protect a workbook for opening or modifying by using a macro. Just as in Word, this is quicker than clicking the **Tools** menu, pointing to **Protection**, clicking **Protect Workbook**, and so on.

## To protect a workbook using a macro

1. In Excel, on the **Tools** menu, point to **Macro** and then click **Macros**.
2. Type **SetExcelOpenPassword**.
3. In the **Macros in** list, select **This Workbook**.
4. Click **Create**. The Visual Basic Editor opens.
5. Between the code `Sub SetExcelOpenPassword()` and `End Sub`, type the code **exactly** as below:

```
Dim strPassword As String

strPassword = InputBox(Prompt:="Type " & _
   "a case-sensitive password that " & _
   "must be used to open this workbook " & _
   "from now on, and then click OK.")

ActiveWorkbook.Password = strPassword

MsgBox Prompt:="Open password set to '" & _
strPassword & "'."
```

6. Click anywhere inside the macro code, and then on the **Run** menu, click **Run Sub/UserForm**.
7. Enter your password, click **OK**, and then save and close the document.

If you want to protect the document workbook from being modified rather than opened, repeat the previous steps for the **SetExcelWritePassword** macro. Here is the code:

```
Dim strPassword As String

strPassword = InputBox(Prompt:="Type " & _
   "a case-sensitive password that " & _
   "must be used to make changes to " & _
   "this workbook from now on, " & _
   "and then click OK.")
ActiveWorkbook.WritePassword = strPassword

MsgBox Prompt:="Write password set to '" & _
strPassword & "'."
```

## Build a Macro for a Timesheet with a Simple Function

Excel offers a simple yet powerful way to collect employees' timesheet entries: the **NOW()** function. Using this function, you can create a macro that enables an employee to clock in or clock out with the click of a button. To record a macro that enters and updates the **NOW()** function, do the following:

1. On the **Tools menu**, point to **Macro**, and then click **Record New Macro**.
2. In the **Macro name** box, enter a name for the macro, such as "Timesheet".
3. In the **Store macro in** box, click the location where you want to store the macro. If you want a macro to be available whenever you use Excel, select **Personal Macro Workbook**.
4. If you want to include a description of the macro, type it in the **Description** box.
5. Click **OK**.
6. In the worksheet, select the cell in which the employee's clock-in time should appear, type the formula **=Now()**, and press **ENTER**.
7. **Copy** the cell.
8. **Right-click** the same cell, and click **Paste Special** on the shortcut menu. Under **Paste**, select **Values**, and then click **OK**. Doing this freezes the clock-in/out time so it cannot be altered by the employee.
9. Press **ENTER**.
10. On the **Stop Recording** toolbar, click **Stop Recording**.

Now you have a macro that updates a selected cell with the current time. The next step is to assign that macro to a button, so that the entry can be accomplished with a single click. To create a custom toolbar button and assign the new macro:

1. On the **Tools** menu, click **Customize**, and then click **Commands** tab.
2. In the **Categories** box, click **Macros**.
3. Drag the **Custom Button** icon from the **Commands** box to a toolbar. Leaving the **Customize** dialog box open, do the following:

    a. Right-click the new button and then type a name, such as "ClockInOut", in the **Name** box on the shortcut menu.

b. Right-click the new button, click **Change Button Image**, and then click an image. Or, to display the button name instead of an image, click **Text Only (Always)**.

c. Right-click the new button and click **Assign Macro**. Under **Macro Name**, click the name of the macro you just created, and click **OK**.

4. Close the **Customize** dialog box.

Now all the employee has to do is selecting the appropriate cell and click the **"Clock In/Out**" button.

## A macro to give your Workbooks a Professional Look

To maintain a consistent and professional look in the documents you send to your company's clients, you can create a macro that automatically formats your workbooks with certain elements. Among other things, your macro sets the page layout to landscape, specifies the page margins, and adds standard elements such as copyright information and page numbers to page headers and footers.

Identifying repetitive tasks and recording them as macros saves you a lot of time, helps to maintain consistency, and reduces mistakes. The following procedure demonstrates how to create a macro you can use to insert a custom footer into your documents. To create the macro, follow these steps:

1. Open a new Excel workbook.
2. On the **Tools menu**, point to **Macro**, and then click **Record New Macro**.
3. In the **Macro name** text box, type the name for the macro, such as *FormatPage*.
4. In the **Store macro in** list, select **Personal Macro Workbook**. (Note: You must save the macro in your Personal Macro Workbook, or it will be lost.)
5. Click **OK**.
6. On the **View** menu, click **Header and Footer**.
7. Click the **Custom Footer** button.
8. Click in the **Left section**, **Center section**, or **Right section** box, and then click the buttons to insert the header or footer information you want in that section; or, type in your own information.
9. Click the **Font** button (the button with a large A) to change the font attributes.
10. Click **OK**.
11. On the **Tools** menu, point to **Macro**, and then click **Stop Recording**.

## To use the macro in a new document:

1. Open a document.
2. On the **Tools** menu, point to **Macro**, and then click **Macros**.

3. In the **Macro name** box, click the name of the macro you want to run.
4. Click **Run**.

To view your results, click **Print Preview** on the **Standard** toolbar.

> **Note:** To use **Print Preview**, you must have filled in at least one cell in the workbook.

## Display Word Counts in the Title Bar

This macro allows you to display the number of words, characters (not including spaces), and paragraphs in the title bar of an open Microsoft Word 2002 document. This is a simple replacement of the current Word Count command on the **Tools** menu.

**To display word, characters, and paragraph counts in the title bar, follow these steps:**

8. Start Word.
9. Press **ALT+F8**. or, on the **Tools** menu, point to **Macro** and then click **Macros**.
10. In the **Macros in** box, select **Normal.dot (global template)**.
11. In the **Macro name** box, type **ShowDocStats** and then click **Create**. The Visual Basic Program will be startup.

**Find the code that looks like this:**

```
Sub ShowDocStats()
'
' ShowDocStats Macro
' Macro created [date] by [author]
'
End Sub
```

**Make this code look like the following by typing in the new lines in the middle:**

```
Sub ShowDocStats()
'
' ShowDocStats Macro
' Macro created [date] by [author]
'
    With ActiveDocument.ActiveWindow
        .Caption = .Document.Name & " / " & _

.Document.BuiltInDocumentProperties(wdPropertyWords) & "
Words, " & _

.Document.BuiltInDocumentProperties(wdPropertyCharacters)
& " Characters, " & _

.Document.BuiltInDocumentProperties(wdPropertyParas) & "
```

```
Paragraphs"
    End With
End Sub
```

12. Press **ALT+F11**. or, click **Return to Microsoft Word** on the **Standard** toolbar.
13. Open the **Macros** dialog box again (as in steps 2 through 4) and click **Run**. You should see the number of words, characters (without spaces), and paragraphs in the title bar of the Word document that is currently open and visible.

## Display Full Path Name in Title Bar

When you open a document in Word, the file name for the document is displayed in the title bar. It would be nice to display more than a simple file name in the title bar. Many readers could profit by a way to display a full path name along with the file name in the title bar. Unfortunately, Word does not provide a way to do this easily.

There are a couple of ways that this can be approached. If you only need to know the full path name occasionally, then you can create a very simple macro and assign it to a toolbar button. When you click on the button, the information in the title bar for the active window is changed to reflect the full path name. This macro is called ChangeCaption.

## To display full path name in the title:

1. Start Word.
2. Press ALT+F8. or, on the **Tools** menu, point to **Macro** and then click **Macros**.
3. In the **Macros in** box, select **Normal.dot (global template)**.
4. In the **Macro name** box, type **ChangeCaption** and then click **Create**. The Visual Basic Program will be startup.

   **Find the code that looks like this:**

```
Sub ShowDocStats()
'
' ShowDocStats Macro
' Macro created [date] by [author]
'
End Sub
```

   **Make this code look like the following:**

```
Sub ChangeCaption()

  ActiveWindow.Caption = ActiveDocument.FullName
End Sub
```

5. Press ALT+F11. or, click **Return to Microsoft Word** on the **Standard** toolbar.

6. Open the **Macros** dialog box again and click **Run**. You should see the full path name in the title bar of the Word document that is currently open and visible.

## To assign the macro to a toolbar button:

1. Right-click on any toolbar and choose **Customize**, or use the **Tools**>**Customize** menu options from the main menu.
2. Choose the **Commands** tab.
3. In the **Categories** list, click on **Macros**. In the **Commands** list to the right, find **ChangeCaption**. Drag this to the desired location on your toolbar.
4. Select the **Close** button.
5. Click the toolbar button to run the macro manually.

## Alternately, you can use this option without a macro:

1. Right-click on any toolbar and choose **Customize**, or use the **Tools**>**Customize** menu options from the main menu.
2. Choose the **Commands** tab.
3. In the **Categories** list, click on **Web**. In the **Commands** list to the right, find **Address**. (It should be first in the list, and contains a drop-down list as part of the command.) Drag Address to the desired location on your toolbar.
4. Select the **Close** button.
5. Now when you open a document, **Address** window show the full path name of the document that is currently open and visible.

Normally, the Address button and drop-down area are used to enter a URL you want to visit (enter a URL, press Enter, and your browser opens to that address). The drop-down box lets you re-visit sites. However, if you don't use the button to visit the Web, Word displays the full path of the open document, but once you use that button, Word goes into "Web" mode and lists Web sites instead.

## Convert All Hyperlinks to Plain Text

Use the following macro that converts existing hyperlinks to plain text. The macro seeks out hyperlinks no matter where they are. It can found any hyperlinks in the header, footer, text boxes, and different sections of the main body of the document.

```
Sub RemoveHyperlinks()

  Dim oStory As Range
  For Each oStory In ActiveDocument.StoryRanges
     Do While oStory.Hyperlinks.Count > 0
     oStory.Hyperlinks(1).Delete
     Loop
  Next oStory

End Sub
```

## Toggle Button for Picture Placeholder

Create the macro below, and then put a button on the toolbar, (using Tools>Customize>Commands>Macro) that executes the macro, which acts as a toggle - changing the value from true to false (or back to true) each times it is run.

```
Sub TogglePlaceholders()
  With ActiveWindow
    With .View
         .ShowPicturePlaceHolders = Not
.ShowPicturePlaceHolders
    End With
  End With
End Sub
```

## Automatically start a Word document where you left off

While we know of no option you can set to do this automatically, it is possible to achieve the same thing by creating an **AutoOpen** macro.

If you want to open only a particular document to the last edited position, you would save the AutoOpen macro to just that document. To make sure that it runs all the time, add it to the **normal.dot** global template.

> **Note:** You can override the AutoOpen behavior: just hold down the **Shift** key when you open a document.

## Defining a Macro:

Before creating a fresh macro, check whether you have an AutoOpen macro already defined. Do the following steps to check it out:

1. From the main menu, select **Tools/Macro/Macros**.
2. In the **Macros in** drop down, the default will be **All active templates and documents**.
3. If **AutoOpen** appears in the list, then click it in the list and then select the **Edit** button.
4. Add the following lines of code between the **Sub** and **End Sub** lines:

   ```
   MsgBox "Moving to the last position edited",
   MsgBoxSetForeground
   Application.GoBack
   ```

   **Note:** the first line -- with the MsgBox command -- will pop up each time you open a document to remind you that you are being moved to the last position edited. If you do not want to be notified, include only the second line in the macro. (`Application.GoBack`)

5. Save the macro and return to your document.

## Creating a Macro:

1. Use the **Tools/Macro/Record New Macro** menu sequence.

2. In the Record Macro dialog box, type **AutoOpen** in the **Macro name** field, and click on **OK**. By default, the macro is saved in the Normal.dot template so it will be available to all documents you work on in Word.
3. A small two-button toolbar appears. Press **Shift + F5** (this records the "return to last position edited" command). Click the square button to Stop Recording.
6. Steps 4 through 6 are optional. They cause word to display a pop-up window reminding you that you are being moved to the last position edited. To omit this reminder, skip to step 7.
4. From the main menu, **select Tools/Macro/Macros**. Choose **AutoOpen** in the list of macros and click **Edit**.
5. Add the following code to the AutoOpen macro:

```
MsgBox "Moving to the last position edited",
vbMsgBoxSetForeground
```

6. On the File menu in the Microsoft Visual Basic Editor, click Save Normal, and then close the Visual Basic Editor.
7. There is no need to save the current (new) document, so you can delete it if you like or use it as the start of a new document. Continue working in Word as you normally would.

*****