# Static Macros

*by George Lungu*

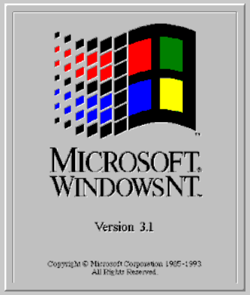
**<excelunusual.com>**

RIT – my old school in Rochester NY

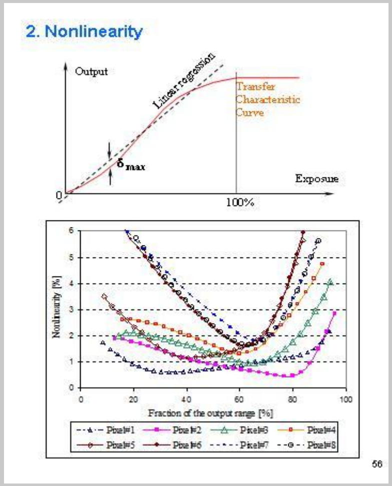
I started using Excel around 1993 while in graduate school.

<excelunusual.com>

All Windows stuff seemed awkward not to say annoying at the beginning. That means navigating through directories, opening files, saving and renaming and overall I wasn't too impressed with anything Excel or the Office suite had to offer except for the fact that I didn't have to waste paper like with a type writer



An old Windows logo



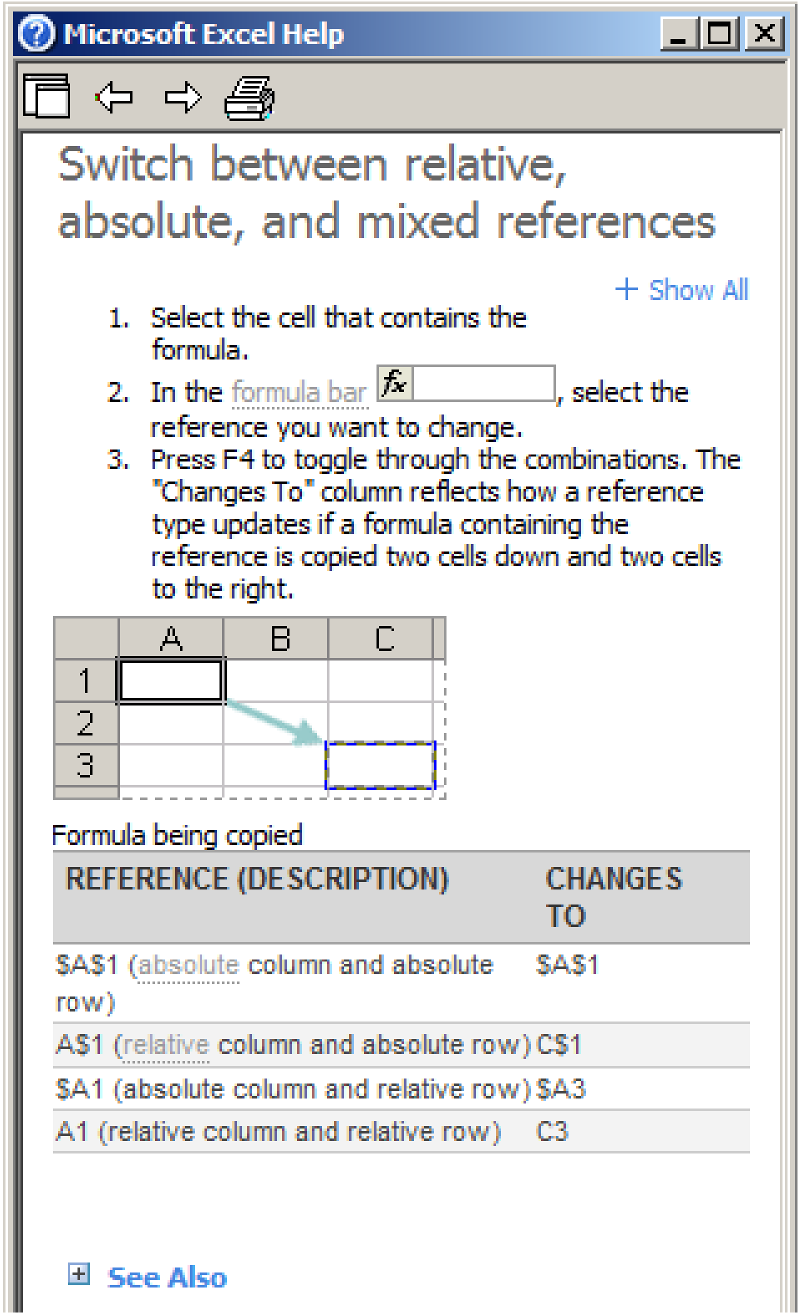
Later I employed Excel for processing data in my thesis work but now I realize I was using it in a very inefficient fashion. You can see all these books and sites around promising increased productivity…, well I was not too productive but I got the work done anyway.

One of the first ***important things*** I learned is that Excel will normally copy formulas in relative reference with neighboring cells. You can make a ***reference absolute*** with respect to a column, a row or both a column and a row by using the dollar sign ($).

You can find out more by opening the ***"Help"*** menu in Excel and typing ***"absolute referencing"***

If you still need more clarification you can search Google or YouTube.

<excelunusual.com>

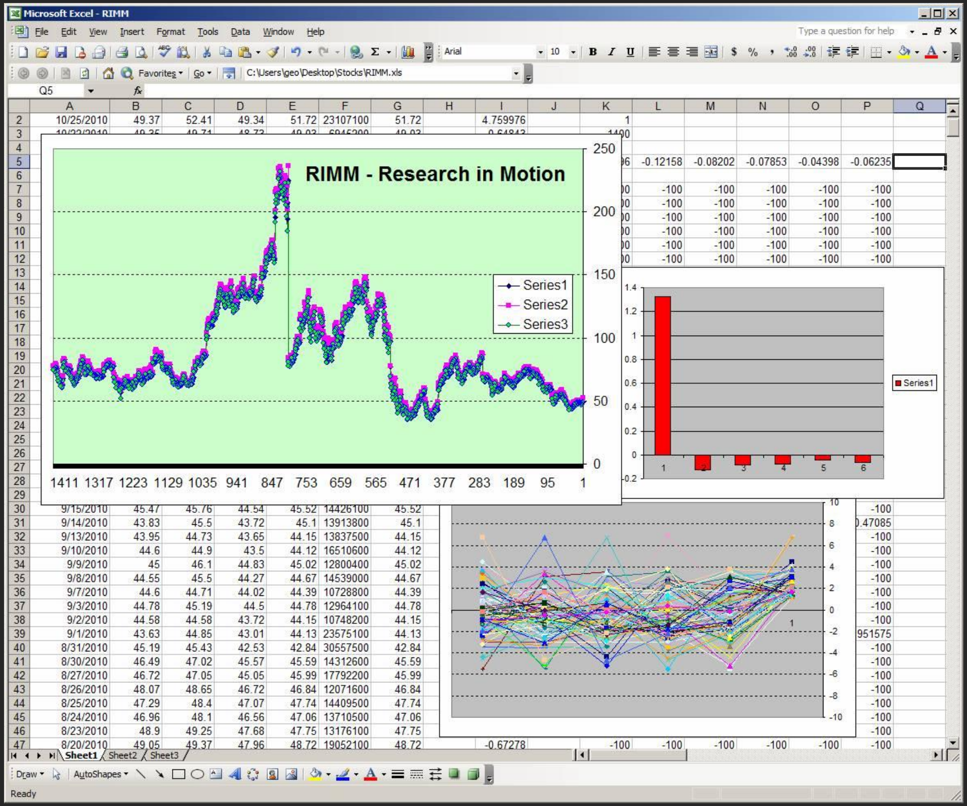


Switch between relative, absolute, and mixed references

1. Select the cell that contains the formula.
2. In the formula bar *fx*, select the reference you want to change.
3. Press F4 to toggle through the combinations. The "Changes To" column reflects how a reference type updates if a formula containing the reference is copied two cells down and two cells to the right.

Formula being copied

| REFERENCE (DESCRIPTION) | CHANGES TO |
|---|---|
| $A$1 (absolute column and absolute row) | $A$1 |
| A$1 (relative column and absolute row) | C$1 |
| $A1 (absolute column and relative row) | $A3 |
| A1 (relative column and relative row) | C3 |



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 2 | 10/25/2010 | 49.37 | 52.41 | 49.34 | 51.72 | 23107100 | 51.72 |
| 30 | 9/15/2010 | 45.47 | 45.76 | 44.54 | 45.52 | 14426100 | 45.52 |
| 31 | 9/14/2010 | 43.83 | 45.5 | 43.72 | 45.1 | 13913800 | 45.1 |
| 32 | 9/13/2010 | 43.95 | 44.73 | 43.65 | 44.15 | 13837500 | 44.15 |
| 33 | 9/10/2010 | 44.6 | 44.9 | 43.5 | 44.12 | 16510600 | 44.12 |
| 34 | 9/9/2010 | 45 | 46.1 | 44.83 | 45.02 | 12800400 | 45.02 |
| 35 | 9/8/2010 | 44.55 | 45.5 | 44.27 | 44.67 | 14539000 | 44.67 |
| 36 | 9/7/2010 | 44.6 | 44.71 | 44.02 | 44.39 | 10728800 | 44.39 |
| 37 | 9/3/2010 | 44.78 | 45.19 | 44.5 | 44.78 | 12964100 | 44.78 |
| 38 | 9/2/2010 | 44.58 | 44.58 | 43.72 | 44.15 | 10748200 | 44.15 |
| 39 | 9/1/2010 | 43.63 | 44.85 | 43.01 | 44.13 | 23575100 | 44.13 |
| 40 | 8/31/2010 | 45.19 | 45.43 | 42.53 | 42.84 | 30557500 | 42.84 |
| 41 | 8/30/2010 | 46.49 | 47.02 | 45.57 | 45.59 | 14312600 | 45.59 |
| 42 | 8/27/2010 | 46.72 | 47.05 | 45.05 | 45.99 | 17792200 | 45.99 |
| 43 | 8/26/2010 | 48.07 | 48.65 | 46.72 | 46.84 | 12071600 | 46.84 |
| 44 | 8/25/2010 | 47.29 | 48.4 | 47.07 | 47.74 | 14409500 | 47.74 |
| 45 | 8/24/2010 | 46.96 | 48.1 | 46.56 | 47.06 | 13710500 | 47.06 |
| 46 | 8/23/2010 | 48.9 | 49.25 | 47.68 | 47.75 | 13176100 | 47.75 |
| 47 | 8/20/2010 | 49.05 | 49.37 | 47.96 | 48.72 | 19052100 | 48.72 |

In 2003-2004 I was looking for trends in the stock market using Excel. I needed to do parametric analysis which meant calculating a certain trading strategy work sheet for a family of constants and storing/plotting the results as an array.

It was then when I was introduced to visual basic. I was shown how to create a button in Excel and associate it with a simple Visual Basic code called a macro.

**And there was light!**

# ***Enabling macros in Excel***

***In Excel 2003 or older:***

Tools => Macro => Security => Low (or Medium) => OK => Save => Close => Reopen Workbook

<excelunusual.com>

***In Excel 2007:***

Right Click the Office Button => Excel Options => Trust Center => Trust Center Settings => Macro Settings => Disable all macros with notification => OK => Save and Close => Reopen Workbook

When you reopen a file with macros if you get a "Security Warning" do the following: Options => Enable this content => OK

***In Excel 2010** – being done in two stages**:***

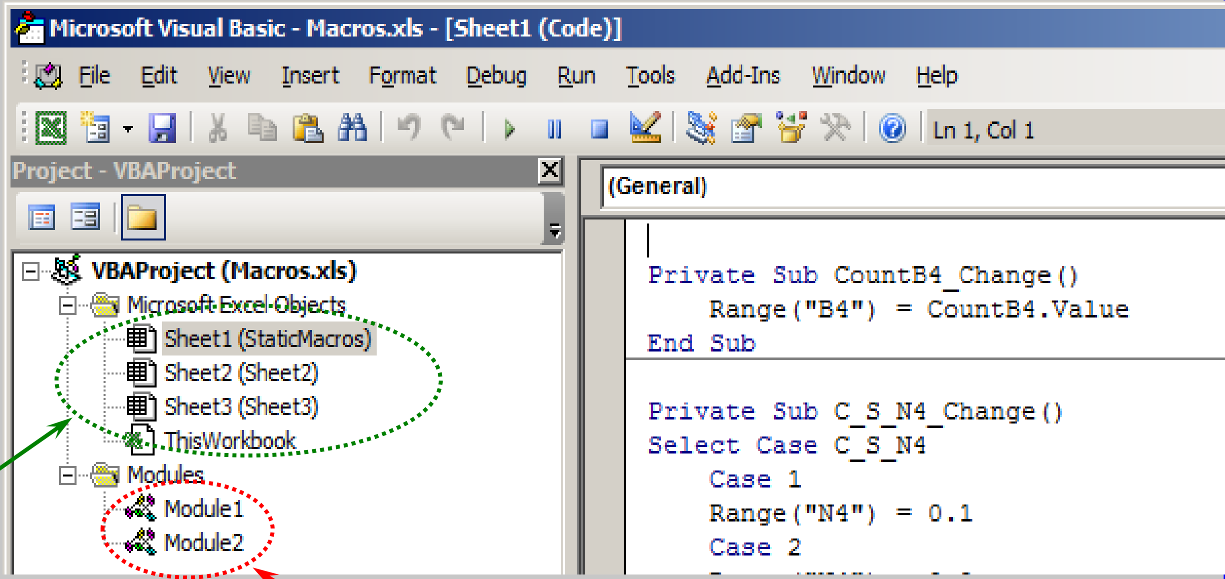
A) Open File => File => Options => Trust Center => Trust Center Settings => Macro Settings => Enable all macros … => OK => OK => Save

B) File => Options => Trust Center => Trust Center Settings => Protected View => Disable them … => OK => OK => Save => Close => Reopen Workbook

## ***Where to find and create macros:***

<excelunusual.com>

In Excel 2003 and earlier go to Tools => Macro => Visual Basic Editor or simply type **"Alt F11"**



```
Private Sub CountB4_Change()
    Range("B4") = CountB4.Value
End Sub

Private Sub C_S_N4_Change()
Select Case C_S_N4
    Case 1
    Range("N4") = 0.1
    Case 2
```

**Workbook And Sheet Modules** are special modules tied directly to the Workbook object and to each Sheet object.

**Standard Code Modules**, also called Modules, are where you put most of your VBA code. Your basic macros and your custom function (User Defined Functions) should be in these modules.

For the novice programmer, *all* your code should be in standard modules. For example a function will work only while written in a module. The modules have a much larger scope and they could even be shared between workbooks.

## ***A Spin Button Macro in Visual Basic for Applications (VBA)***

This macro will be controlled by a “spin button” and it will adjust the integer value of a certain cell between the lower limit of 1 and the higher limit of 10.

<excelunusual.com>

*Instructions:*

- Open a new workbook
- View -> Toolbox -> Control Toolbox -> Design Mode (click)
- In the control toolbox click “Spin Button” and trace out (drag) a button somewhere around cell “C3”
- Right click the button and select “Properties”
- In the property box replace the name “SpinButton1” with “CountB4” (or some other name you like)
- In the property box replace “min” with 1 and “max” with 10

<excelunusual.com>

# ***A Spin Button Macro***

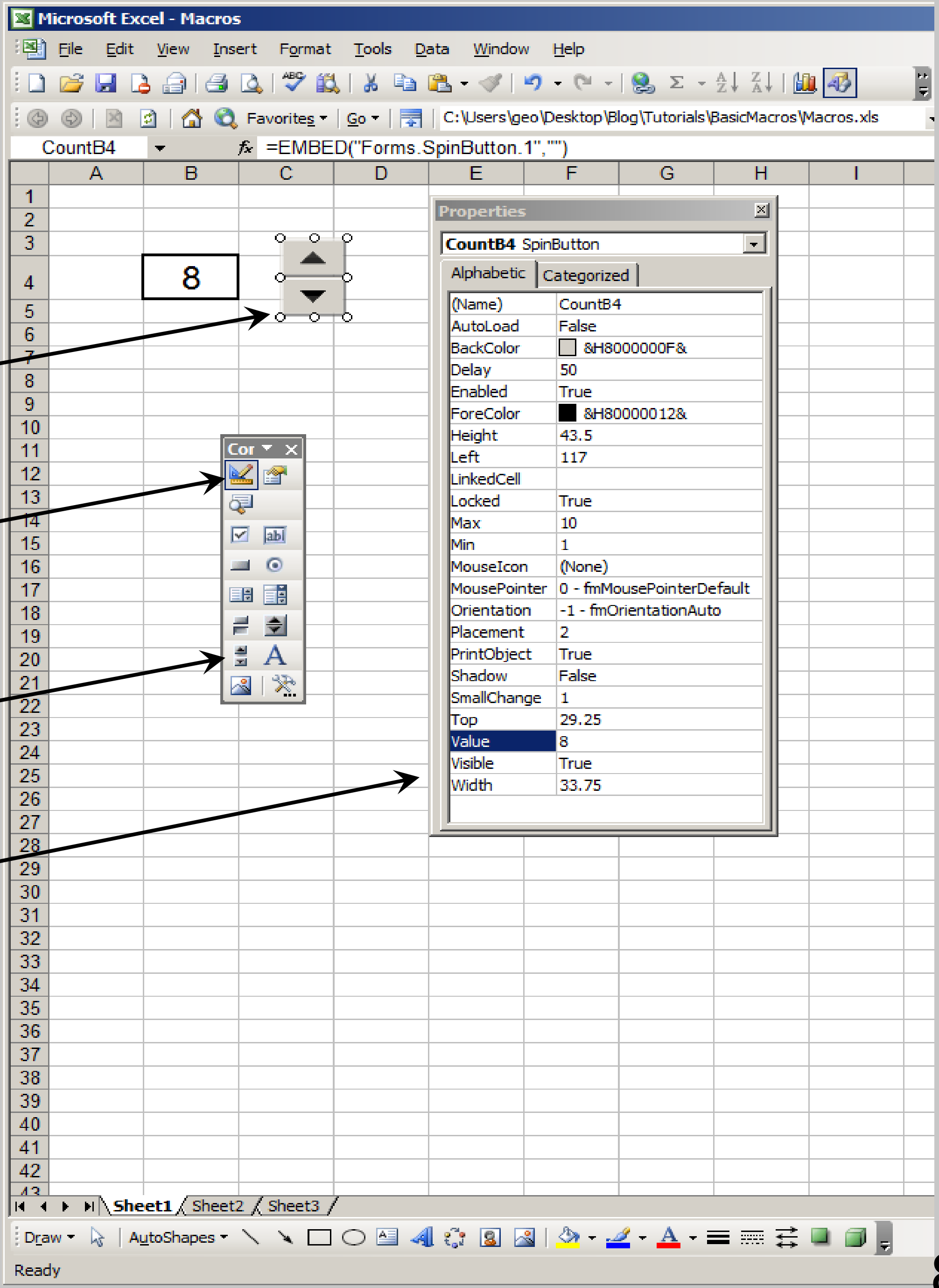
*(continuation)*

Spin button

Design mode icon

Control toolbox

Properties box



Microsoft Excel - Macros

C:\Users\geo\Desktop\Blog\Tutorials\BasicMacros\Macros.xls

CountB4 =EMBED("Forms.SpinButton.1","")

8

Properties

CountB4 SpinButton

Alphabetic | Categorized

| | |
|---|---|
| (Name) | CountB4 |
| AutoLoad | False |
| BackColor | □ &H8000000F& |
| Delay | 50 |
| Enabled | True |
| ForeColor | ■ &H80000012& |
| Height | 43.5 |
| Left | 117 |
| LinkedCell | |
| Locked | True |
| Max | 10 |
| Min | 1 |
| MouseIcon | (None) |
| MousePointer | 0 - fmMousePointerDefault |
| Orientation | -1 - fmOrientationAuto |
| Placement | 2 |
| PrintObject | True |
| Shadow | False |
| SmallChange | 1 |
| Top | 29.25 |
| Value | 8 |
| Visible | True |
| Width | 33.75 |

<excelunusual.com>

## *A Spin Button Macro*
*(continuation)*

<excelunusual.com>

- You can change colors and other properties to your taste
    - Double click the button. The Visual Basic editor window comes up with the following code:

```
Private Sub CountB4_Change()

End Sub
```

    - Replace the code with the following:

```
Private Sub CountB4_Change()
            Range("B4") = CountB4.Value
End Sub
```

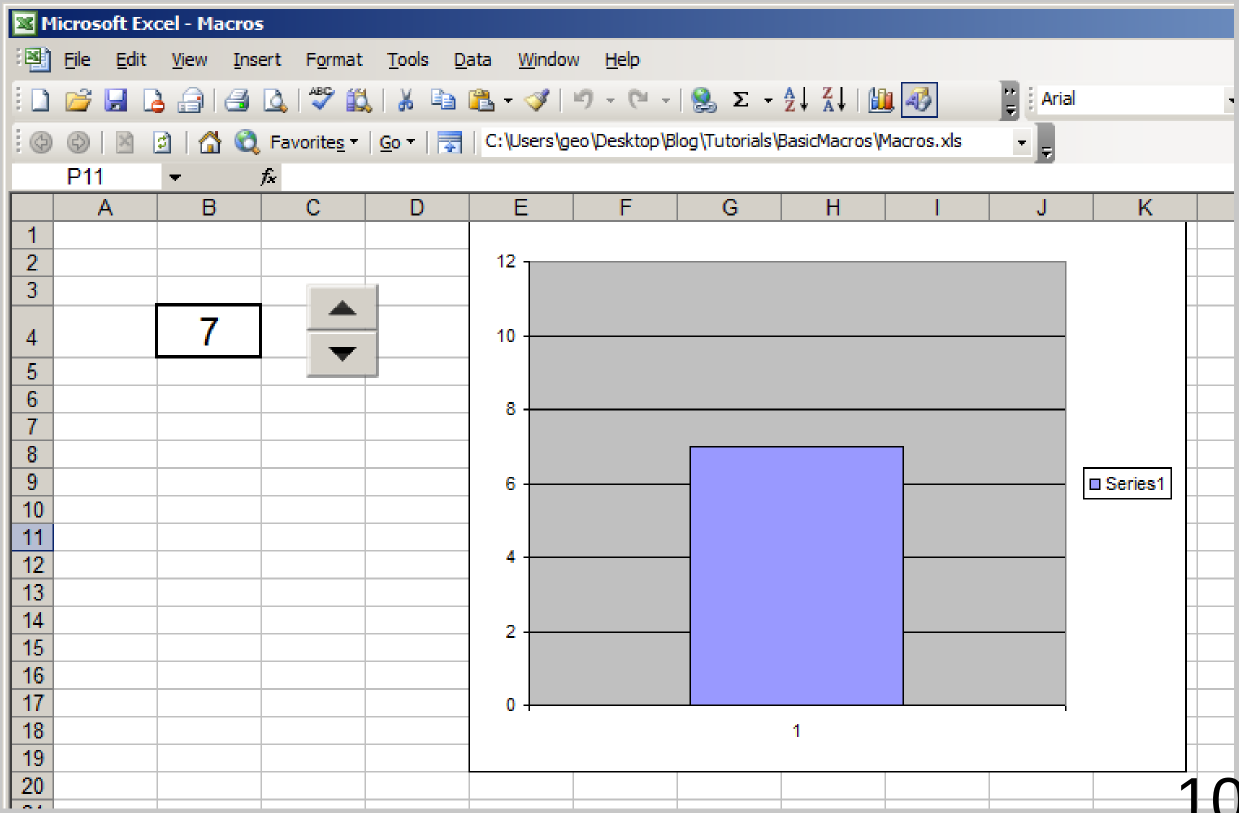
    - Now get back to the “Control Toolbox” and exit the design mode by clicking on the icon with the same name.
    - Close the control toolbox window and try to see if the button you just created works as expected

## ***A Spin Button Macro***
*(continuation)*

<excelunusual.com>

*Verification:*

- Let’s associate a chart with cell B4
- Select cell “B4” => Insert => Chart => Column => Finish
- Select the chart => Format Y axis => Maximum = 0, Minimum = 12
- Click the spin button and see the effects on the chart



Microsoft Excel - Macros
File Edit View Insert Format Tools Data Window Help
Arial
Favorites Go
C:\Users\geo\Desktop\Blog\Tutorials\BasicMacros\Macros.xls
P11
7
12
10
8
6
4
2
0
1
Series1

## ***A "Select-Case" Macro in Visual Basic for Applications (VBA)***

Sometimes the correspondence between the value of the cell and the value of the macro argument is not a linear one but can be described as a look-up table. In this case we will use the "Select Case" construct in VBA. The syntax for the **Case** statement is:

<excelunusual.com>

```
Select Case test_expression
          Case condition_1
                    result_1
          ……………………
          ……………………
          Case condition_n
                    result_n
          Case Else
                    result_else
End Select
```

*test_expression* is a string or numeric value. It is the value that you are comparing to the list of conditions (labeled from 1 to n). Once a *condition* is found to be true, the **Case** statement will execute the corresponding code and not evaluate the conditions any further.

<excelunusual.com>

## ***The "Select-Case" Macro*** *- continuation*

<excelunusual.com>

*Instructions:*

1. In the existing work sheet (Sheet_1) enter the design mode by doing the following:
 View -> Toolbox -> Control Toolbox -> Design Mode (click)
2. In the control toolbox click "Spin Button" and trace out (drag) a button somewhere around cell "O3"
3. Right click the button and select "Properties"
4. In the property box replace the name "SpinButton1" with "Select_CaseN4" (or some other name you like)
5. In the property box replace "min" with 1 and "max" with 7
6. You can change colors and other properties to your taste
7. Double click the button. The Visual Basic editor window comes up with the following code:

```
Private Sub CountB4_Change()

End Sub
```

<excelunusual.com>

## ***The "Select-Case" Macro*** *- continuation*

*Instructions:*

- Replace the macro code with the following:

```
Private Sub C_S_N4_Change()
        Select Case Case_SelectN4
                Case 1
                Range("N4") = 0.1
                Case 2
                Range("N4") = 0.2
                Case 3
                Range("N4") = 0.5
                Case 4
                Range("N4") = 1
                Case 5
                Range("N4") = 2
                Case 6
                Range("N4") = 5
                Case Else
                Range("N4") = 10
        End Select
End Sub
```

<excelunusual.com>

## ***The "Select-Case" Macro*** *- continuation*

*Verification:*

- Now get back to the "Control Toolbox" and exit the design mode by clicking on the icon with the same name.
- Close the control toolbox window and try to see if the button you just created works as expected

**You are supposed to be able to go through the following combination of values in Cell "N4" in any diraction:**

***0.1 => 0.2 => 0.5 => 1 => 2 => 5 =>10***

***0.1 <= 0.2 <= 0.5 <= 1 <= 2 <= 5 <=10***

<excelunusual.com>

## ***The “Application” Macro in Visual Basic for Applications (VBA)***

Often times, the output of a macro must be expressed as a function of the macro’s argument. This function might not exist in VBA and in this case we will use the following construct: ***Application.Function()***, where ***“Function”*** is the name of the equivalent spreadsheet function. This way we can use a large variety of functions nonexistent in VBA.

<excelunusual.com>

*Instructions:*

- After creating a spin button, right click it and select “Properties”
- In the property box replace the name “SpinButton1” with “RotateB4” (or some other name you like)
- Double click the button and replace the existing Visual Basic code with:

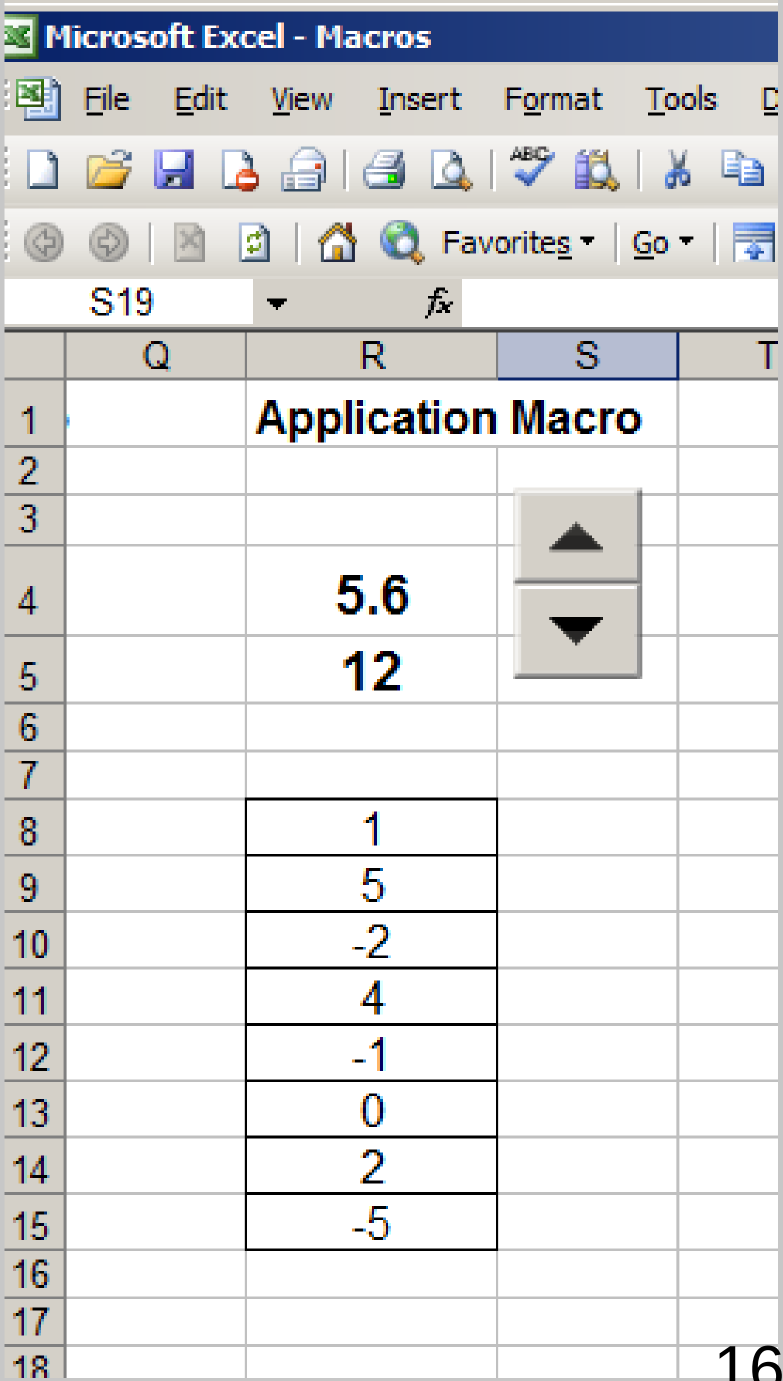
```
Private Sub AppR4_Change()
            Range("R4") = Application.Round(AppR4.Value / 17, 1)
            Range("R5") = Application.SumIf(Range("R8:R15"), ">0")
End Sub
```

<excelunusual.com>

## ***<u>The "Application" Macro</u>*** *(continuation)*

This macro uses the spreadsheet function "Round" to find the truncated value (with one digit past the decimal point) of the macro argument divided by 17 and display it in cell "R4". (search Help for the function "Round")

The macro also takes the positive values in the range "R8:R15" and sums them up, displaying the result in cell "R5" (search Help for the function "Sumif")



| | Q | R | S |
|---|---|---|---|
| 1 | | **Application Macro** | |
| 2 | | | |
| 3 | | | |
| 4 | | **5.6** | |
| 5 | | **12** | |
| 6 | | | |
| 7 | | | |
| 8 | | 1 | |
| 9 | | 5 | |
| 10 | | -2 | |
| 11 | | 4 | |
| 12 | | -1 | |
| 13 | | 0 | |
| 14 | | 2 | |
| 15 | | -5 | |
| 16 | | | |
| 17 | | | |
| 18 | | | |

<excelunusual.com>

## ***The “Rotation” Macro in Visual Basic for Applications (VBA)***

This macro will be controlled by a “spin button” and it will adjust the integer value of a certain cell between the lower limit of 0 and the higher limit of 355 in increments of 5 degrees as an infinite loop. This macro uses two “If” statements in order to get the angle to loop around between the values 0 and 355.

<excelunusual.com>

*Instructions:*

- View -> Toolbox -> Control Toolbox -> Design Mode (click)
- In the control toolbox click “Spin Button” and trace out (drag) a button somewhere around cell “W3”
- Replace the name “SpinButton1” with “RotV4”
- In the property box replace “min” with -1 and “max” with 80
- Double click the button and replace the existing Visual Basic code with:

```
Private Sub RotV4_Change()
        If RotV4 > 71 Then RotV4 = 0
        If RotV4 < 0 Then RotV4 = 71
        Range("V4") = 5 * RotV4.Value
End Sub
```

<excelunusual.com>

## ***The "Rotation" macro*** *(continuation)*

In order to easily verify the functionality we will have to add a 2x2 table:

- Cell "V9"=0
- Cell "W9"=0
- Cell "V10"= cos(radians(V4))
- Cell "W10"= sin(radians(V4))

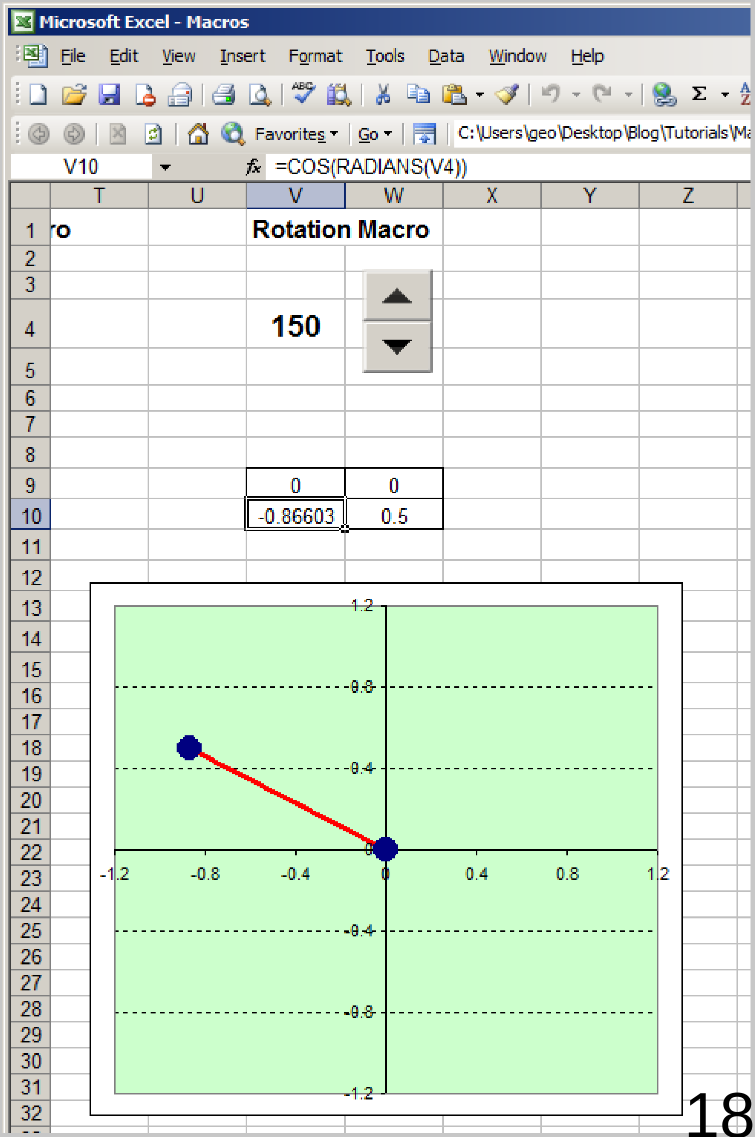
We'll add an XY scatter plot too:

The "X values" will be from the range "V9:V10"

The "Y values" will be from the range "W9:W10"

The axes will have a range between -1.2 to 1.2

Clicking the spin button will rotate the charted segment in both directions.



| | V | W |
|---|---|---|
| 9 | 0 | 0 |
| 10 | -0.86603 | 0.5 |

<excelunusual.com>

**Conclusions:** Five different shape alterations were shown in this presentation:

<excelunusual.com>

1. ***Resizing*** – which can be done independently on each axis by multiplying the coordinates by a certain factor while the shape is centered in the origin.
2. ***Translation*** – which can be done independently on each axis by adding or subtracting a term to each coordinate.
3. ***Simple rotation*** – formulas were derived for rotation around the origin by an angle $\alpha$ of a shape centered in origin.
4. ***Rotation of a shape around its own axis*** – the exact derivation was not given but the recipe is to translate the shape back to origin, then rotate it and then translate the shape to it's final position (translation + rotation + translation)
5. ***Scene rotation*** – this is just a simple rotation but the shapes in the scene are held fixed at their own coordinates.

***A MS Excel 2003 work book with all the examples can be downloaded. The reader is advised to examine the functionality of each section of the work sheet and try to rebuild it from scratch reading as little as possible. This presentation should be seen as a lure or a lifeline. The farther you can reach with the least reading, the more knowledge you'll gain. Your only real teacher lives in yourself.*** **Spend few days, don't rush!**