



# Astute Excel: Intro to Macros Part 2

Written and Presented by  
David H. Ringstrom, CPA  
Accounting Advisors, Inc.  
[www.accountingadvisors.com](http://www.accountingadvisors.com)



**About the speaker:**

David H. Ringstrom, CPA, is an author and nationally recognized instructor who teaches scores of webinars each year. His Excel courses are based on over 25 years of consulting and teaching experience. His mantra is “Either you work Excel, or it works you.” David offers spreadsheet and database consulting services nationwide.

# Excel Versions

I'll be teaching from the Office 365 version of Excel, and noting any differences or limitations in the legacy versions of Excel.

## The Future of Excel

### Office 365

Subscription-based version of Microsoft Office, which includes Excel. Use the software on multiple devices, new calculation engine, services and more features are rolling out that will dramatically set this version apart from past Excel versions.

[www.office365.com](http://www.office365.com)

## Legacy Versions

### Perpetual Licenses

Pay once for the software to use on a single computer. No new features added until you buy a new license.

Includes

Excel 2019

Excel 2016

Excel 2013

Excel 2010

Excel 2007 and earlier

1

File Home Insert Page Layout

Cut Copy

I demonstrate each technique at least twice, first by way of numbered steps in PowerPoint, and then in Excel.

Handouts include PDF of today's presentation along with an example workbook that contains most examples I'll teach from as well as related articles.

Email [ask@davidringstrom.com](mailto:ask@davidringstrom.com) if you haven't received the handouts.

7 Excel Tip: Data Entry Shortcut

Related Articles

Car Payme

Sheet1





# Enabling the Developer menu

**1** File

**2** Options

**3** Customize Ribbon

**4** Developer

**5** OK

**6** The Developer menu now appears on the Ribbon.

**Excel 2007 alternate steps 3 and 4: Select 'Popular' tab and tick mark the 'Show Developer tab in the Ribbon' checkbox.**

Popular

Top options for working with Excel

Show Developer tab in the Ribbon

Review View **Developer** Help



# Macros Streamline Repetitive Tasks

**1** Data

**2** What-If Analysis

**3** Goal Seek...

**4** Set cell: \$B\$4

**5** To value: 500

**6** By changing cell: \$B\$3

**7** OK

**8** OK

**A** Goal Seek

**Macro button**

**We can streamline Goal Seek:**

- launch the feature with a Macro button
- gather input with an InputBox

**Macro buttons and InputBoxes document the spreadsheet, simplify inputs, and democratize Excel tasks and features.**

**Goal Seek Macro**

Enter desired payment amount

**C** OK

**B** 500

**InputBox**

	A	B	C	D
1	Interest	5.25%		
2	Term	36		
3	Principal	16620.533		
4	Payment	\$500.00		

	A	B
1	Interest	5.25%
2	Term	36
3	Principal	15,000.00
4	Payment	\$451.25



# Recording a Goal Seek Macro

**1** View Developer

**2** Record Macro

**3** RunGoalSeek

**4** This Workbook

**5** Automates the Goal Seek Feature

**6** OK

You can also choose View, Macros, and then Record Macro in lieu of steps 1-2.

You can also click here in lieu of steps 1-2. If this icon doesn't appear, right-click on the status bar and choose Macro Recording.

Feature names, such as GoalSeek, are reserved words in Excel's programming environment. Always add a prefix, such as Run, to avoid conflicts that are troublesome for beginners to resolve.

Once you click OK most actions you take in Excel are recorded.

	A	B	C	D	E	F	G	H	I
1	Interest	5.25%							
2	Term	36							
3	Principal	15,000.00							
4	Payment	\$451.25							
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									
20									
21									
22									



# Recording a Goal Seek Macro

7 Data

8 What-If Analysis

9 Goal Seek...

10 Set cell: \$B\$4

11 To value: 500

12 By changing cell: \$B\$3

13 OK

14 OK

15 Ready

Goal Seek Status

Goal Seeking with Cell B4 found a solution.

Target value: 500

Current value: \$500.00

Step

Pause

Cancel

Goal Seek

Ready

If you haven't enabled this icon, choose Developer and then click Stop Recording, or choose View, Macros, and then Stop Recording.

	A	B	C	D	E	F
1	Interest	5.25%				
2	Term	36				
3	Principal	16,620.53				
4	Payment	\$500.00				
5						
6						
7						
8						
9						
10						
11						
12						
13						
14						
15						
16						



# Viewing Our Recorded Macro

**1** Developer

**2** Visual Basic

**3** Macros

**4** Run Dialog

	A	B
1	Interest	5.25%
2	Term	36
3	Principal	16,620.53
4	Payment	\$500.
5		
6		
7		

**Macro**

Macro name: RunGoalSeek

PERSONAL.XLSB!CenterTitle  
PERSONAL.XLSB!ContactInfo  
PERSONAL.XLSB!FormatAsNumber  
PERSONAL.XLSB!Macro1  
**RunGoalSeek**

Macros in: All Open Workbooks

Description: Automates the Goal Seek Feature

**In lieu of steps 1-4, you can also type the macro name in the Name Box and press Enter. Note: this trick only works when the macro is in the active workbook.**

**2** Name Box

A1

**Microsoft Visual Basic for Applications - Goal Seek.xlsm - [Module1 (Code)]**

File Edit View Insert Format Debug Run Tools Add-Ins Window

```

Sub RunGoalSeek()
    '
    ' RunGoalSeek Macro
    ' Automates the Goal Seek Feature
    '
    Application.CutCopyMode = False
    Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
End Sub
          
```

**If you don't see your macro on the list, choose All Open Workbooks. Remember, macros are only available when the workbook containing the macro is open.**



# View the Code as Recorded

The Visual Basic Editor (VBE) looks like a separate program, but is part of Excel. Clicking Save in the VBE saves your entire workbook.

Every macro begins with Sub, and ends with End Sub.  
Sub is short for Subroutine

Green text signifies a comment. You can manually enter a comment by typing an apostrophe (') at the beginning of a line of text.

```
Sub RunGoalSeek()
    ' RunGoalSeek Macro
    ' Automates the Goal Seek Feature

    Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
End Sub
```

This is the line of programming code derived from the steps that were taken when recording the macro.





# Deconstructing the Goal Seek Macro

Microsoft Visual Basic for Applications - Goal Seek.xlsm - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 33, Col 1

(General) RunGoalSeek

```

Sub RunGoalSeek()
    ' RunGoalSeek Macro
    ' Automates the Goal Seek Feature
    '
    Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
End Sub

```

1 Object      2 Method      3 Required Parameter      4 Required Parameter

Goal Seek - Excel

File Home Insert Page Layout Formulas Data Review View

Goal Seek

	A	B
1	Interest	5.25%
2	Term	36
3	Principal	16,620.53
4	Payment	\$500.00

Goal Seek

Set cell: B4

To value: 500

By changing cell: B3

OK Cancel

Goal Seek



# Adding a Macro button

**1** Developer

**2** Insert > Form Controls > Button

**3** Button placed on worksheet

**4** Left-click on the worksheet where you want the button to be placed.

**5** Assign Macro dialog box, RunGoalSeek selected

**6** OK

**7** Right-click on the button.

**8** Edit Text

**9** Enter Goal Seek after you choose Edit Text.

**10** Our finished button.

	A	B	C	D
1	Interest	5.25%		
2	Term	36	Goal Seek	
3	Principal	16,620.53		
4	Payment	\$500.00		
5				
6				
7				
8				
9				
10				



# Make the Goal Seek Macro Interactive

**1** Add a couple of blank lines (like you would in Word or an email), and then add this line of code.

```
Sub RunGoalSeek
    RunGoalSeek Macro
    Automates the Goal Seek Feature

    result = InputBox("Payment amount?", "Goal Seek", Range("B4"))

    Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
End Sub
```

**2** Click here to run your macro.

**3** Enter any amount.

**4** OK

**5** The amount will remain unchanged because we need to modify the second line of our macro.

	A	B	C
1	Interest	5.25%	
2	Term	36	
3	Principal	16,620.53	
4	Payment	\$500.00	
5			
6			
7			
8			
9			
10			

	A	B	C	D	E	F	G	H	I
1	Interest	5.25%							
2	Term	36							
3	Principal	16,620.53							
4	Payment	\$500.00							
5									
6									
7									
8									
9									
10									



# Understanding InputBox

**1** InputBox is a method of the Application object, but we don't have to list the object in this case.

**2** "Payment Amount?" is the optional Prompt parameter.

**3** "Goal Seek" is the optional Title parameter.

**4** Range("B4") is the optional default value to appear in the InputBox. Omit to leave the field blank.

```

RunGoals Seek Macro
Automates the Goal Seek Feature

result = InputBox("Payment amount?", "Goal Seek", Range("B4"))
          InputBox(Prompt, [Title], [Default], [XPos], [YPos], [HelpFile], [Context]) As String
          Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
End Sub

```

Result is a variable. This lets us store the input from the user for use later in our macro.

The underscore signifies that the code continues on to the next line.

A more formal way of writing this code would be  
`result=Application.InputBox(Prompt:="Payment Amount", Title:="Goal Seek", Default:=Range("B4"))`

	A	B	C
1	Interest	5.25%	
2	Term	36	
3	Principal	16,620.53	
4	Payment	\$500.00	
5			
6			
7			
14			
15			
16			
17			
18			

**1** InputBox

**2** "Payment Amount?"

**3** "Goal Seek"

**4** Range("B4")



# Revise the Code to Use a Variable

**1** Add a single quote before the first line, and then add the second line of code.

```

RunGoalSeek Macro
Automates the Goal Seek Feature

result = InputBox("Payment amount?", "Goal Seek", Range("B4"))
'Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
Range("B4").GoalSeek result, Range("B3")
End Sub

```

**2** Click here to run your macro (or press the F5 key).

	A	B	C
1	Interest	5.25%	
2	Term	36	
3	Principal	11,634.37	
4	Payment	\$350.00	
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

**5** The amount reflects what you entered in the InputBox.

	A	B	C	D	E	F	G	H	I
1	Interest	5.25%							
2	Term	36							
3	Principal	16,620.53							
4	Payment	\$500.00							
5									
6									
7									
8									
9									
10									

**3** Enter any amount.

**4** OK



# Dealing with Errors/Debug Prompt

	A	B	C	D
1	Interest	5.25%		
2	Term	36		
3	Principal	11,634.37		
4	Payment	\$350.00		
5				
6				
7				
8				
9				
10				

**1** Goal Seek

**2** OK Cancel

**3** Continue End Debug Help

**4** Microsoft Visual Basic for Applications - Goal Seek.xlsm [break] - [Module1 (Code)]  
Excel highlights the line of code that's causing an issue in yellow.

**5** Move your mouse over the word result to see the contents of the variable. The variable doesn't contain a value because the user clicked Cancel.

```

Sub RunGoalSeek()
    RunGoalSeek Macro
    ' Automates the Goal Seek Feature

    result = InputBox("Payment amount?", "Goal Seek", Range("B4"))

    Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
    Range("B4").GoalSeek result, Range("B3")
End Sub

```

**6** Click Reset to stop your macro.

**6** Microsoft Visual Basic for Applications - Goal Seek.xlsm - [Module1 (Code)]

```

Sub RunGoalSeek()
    ' RunGoalSeek Macro
    ' Automates the Goal Seek Feature

    result = InputBox("Payment amount?", "Goal Seek", Range("B4"))

    'Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
    Range("B4").GoalSeek result, Range("B3")
End Sub

```



# Handling The Cancel Button

Microsoft Visual Basic for Applications - Goal Seek.xlsm - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 35, Col 1

RunGoalSeek

1 Add this line of code. If the user clicks Cancel, result will be blank, so the macro will terminate.

2 Click here to run your macro (or press the F5 key).

```

Automates the Goal Seek Feature
result = InputBox("Payment amount?", "Goal Seek", Range("B4"))
If result = "" Then Exit Sub
'Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
Range("B4").GoalSeek result, Range("B3")
End Sub

```

	A	B	C	D
1	Interest	5.25%		
2	Term	36		
3	Principal	11,634.37	Goal Seek	
4	Payment	\$350.00		
5				
6				
7				
8				
9				

Goal Seek

Payment amount?

OK

350

3 Cancel

17 Now the macro ends gracefully when you click Cancel.

18

19

20

Goal Seek



# Handling Invalid Inputs

**1** Add all code circled in red.

```

Range("B4").GoalSeek Goal:=500, C
On Error Resume Next
Range("B4").GoalSeek result:=Range("B3")
If Err <> 0 Then
    MsgBox result & " is not a valid input.", vbCritical, "Invalid Input"
End If
On Error GoTo 0
  
```

**2** Click Run (or press F5).

	A	B	C	D
1	Interest	5.25%		
2	Term	36	Goal Seek	
3	Principal	11,634.37		
4	Payment	\$350.00		
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				

**3** Enter a non-numeric value.

**4** OK

**5** The macro ends gracefully after notifying the user.

**6** This amount remains unchanged due to the invalid input.

	A	B	C	D
1	Interest	5.25%		
2	Term	36	Goal Seek	
3	Principal	11,634.37		
4	Payment	\$350.00		
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				





# Error Handling

Microsoft Visual Basic for Applications - Goal Seek

File Edit View Insert Format Debug Run

(General)

```
Sub RunGoalSeek()
    '
    'Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
    '
    On Error Resume Next
    Range("B4").GoalSeek result,
    If Err <> 0 Then
        MsgBox result & " is not a valid input.", vbCritical, "Invalid Input"
    End If
    On Error GoTo 0
End Sub
```

On Error Resume Next instructs the macro to skip over lines of code that trigger an error and continue on.

On Error Resume Next

On Error GoTo 0

Err is equal to zero if the previous line of code executed properly, otherwise it contains an error number.

Microsoft Visual Basic for Applications - Goal Seek.xlsm - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 42, Col 1

(General) RunGoalSeek

```
If result = "" Then Exit Sub
'Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")
On Error Resume Next
Range("B4").
If Err <> 0 Then
    MsgBox result & " is not a valid input.", vbCritical, "Invalid Input"
End If
On Error GoTo 0
End Sub
```

MsgBox allows us to provide feedback to the user.

Each If must have an End If unless the If is on a single line.

On Error GoTo 0 turns off the error handling, so other lines of code that cause an error won't get skipped. Always be sure to use On Error GoTo 0 in conjunction with On Error Resume Next.



# Understanding MsgBox

Microsoft Visual Basic for Applications - Goal Seek.xlsm - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 35, Col 1

General) RunGoalSeek

1 MsgBox is a method of the Application object, but we don't have to list the object in this case.

2 result & " is not a valid input" is the optional Prompt parameter.

3 vbCritical signifies that we want a red X icon and an OK button for the optional Buttons parameter.

4 "Invalid Input" is the optional Title parameter. The title defaults to Microsoft Excel if you omit the title.

```
MsgBox result & " is not a valid input.", vbCritical, "Invalid Input"
MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult
End If

On Error GoTo 0

End Sub
```

result & " is not an invalid input" allows us to report back to the user exactly what they did wrong. The ampersand joins the contents of the results variable together with some additional text.

	A	B	C	D
1	Interest	5.25%		
2	Term	36		
3	Principal	11,634.37		
4	Payment	\$350.00		
5				
6				

1 MsgBox

2 Prompt Parameter

3 Buttons Parameter

4 Title Parameter

Invalid Input

David is not a valid input.

OK

Other Icon Options

- vbInformation:
- vbExclamation:
- vbQuestion:



# Stepping Through a Macro

**1** Click inside the macro.

**2** Click the **Debug** menu.

**3** Click **Step Into** (F8).

Excel highlights the macro name and then stops. Press F8 to move to the next line of code.

**7** Position your cursor over the result variable to see its contents. Continue pressing F8 to step through the macro if needed.

**4** Press F8 to execute the next line of code, which will display the InputBox.

**5** Enter an amount.

**6** Click **OK**.

**8** Click **Reset** to stop stepping through your macro.

You'll encounter this prompt if you skip Step 8 and try to run your macro again. Click the **Reset** button to resolve.

```

Sub RunGoalSeek()
    ' RunGoalSeek Macro
    ' Automates the Goal Seek Feature
    result =
    If result = "" Then Exit Sub
    Range("B4").GoalSeek Goal:=500, ChangingCell:=Range("B3")

```



# Compile Errors

	A	B	C	D
1	Interest	5.25%		
2	Term	36		
3	Principal	11,634.37		
4	Payment	\$350.00		
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				

**1** Goal Seek

**2** Excel highlights the problem in blue. In this case we inadvertently used the same name for our variable as the name of our macro.

**3** OK

**4** Excel highlights the name of the macro that has a problem.

Compile errors generally signify typographical errors, invalid use of reserved words, or missing parameters.

To check your macro for errors before you run it, choose Debug, and then Compile. If no prompt appears then no compile errors were found.

```

RunGoalSeek = InputBox("Payment amount?", "Goal Seek", Range("B4"))

If result = "" Then Exit Sub
  
```



# Syntax Errors

	A	B	C	D
1	Interest	5.25%		
2	Term	36		
3	Principal	11,634.37		
4	Payment	\$350.00		
5				
6				
7				
8				
9				
10	<b>Microsoft Visual Basic</b>			
11	Run-time error '1004':			
12	Method 'Range' of object '_Global' failed			
13				
14	Continue End <b>Debug</b> Help			
15				
16				
17				
18				
19				
20				

**Microsoft Visual Basic for Applications - Goal Seek.xlsm [break] - [Module1 (Code)]**

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 6, Col 1

(General) GoalSeek

```

Sub RunGoalSeek()
'
' RunGoalSeek Macro
' Automates the Goal Seek Feature
'
result = InputBox("Payment amount?", "Goal Seek", Range("B"))
If result = "" Then Exit Sub
'Range("B4").GoalSeek Goal
On Error Resume Next
Range("B4").GoalSeek result, Range("B3")
If Err <> 0 Then
MsgBox result & " is not a valid input.", vbCritical, "Invalid"
End If
On Error GoTo 0
End Sub

```

**1** Goal Seek

**2** Debug

**3** In this case Range ("B") should be Range ("B4"). Fix the typo and then press F5 to continue the macro.

Click Reset if you need to stop the macro from running.



# Syntax Errors

**1** Sometimes, but not always, the Visual Basic Editor will alert you to problems with your code. Any programming code in red is invalid. The code will turn to blue/black when the problem is corrected.

```

Sub RunGoalSeek()
    RunGoalSeek Macro
    Automates the Goal Seek process
    '
    result = InputBox("Payment amount?", "Goal Seek", Range("B4"))
    If result = "" Then Exit Sub
    'Range("B4").GoalSeek Goal
    On Error Resume Next
    Range("B4").GoalSeek
    If Err <> 0 Then
        MsgBox result
    End If
    On Error GoTo 0
End Sub

```

**2** This prompt appears when you leave a given line of code if it isn't syntactically correct.

Microsoft Visual Basic for Applications

Compile error:  
Syntax error

OK Help

**3** In this case a parenthesis is missing.

**3** Red lettering vanishes when programming code is syntactically correct.

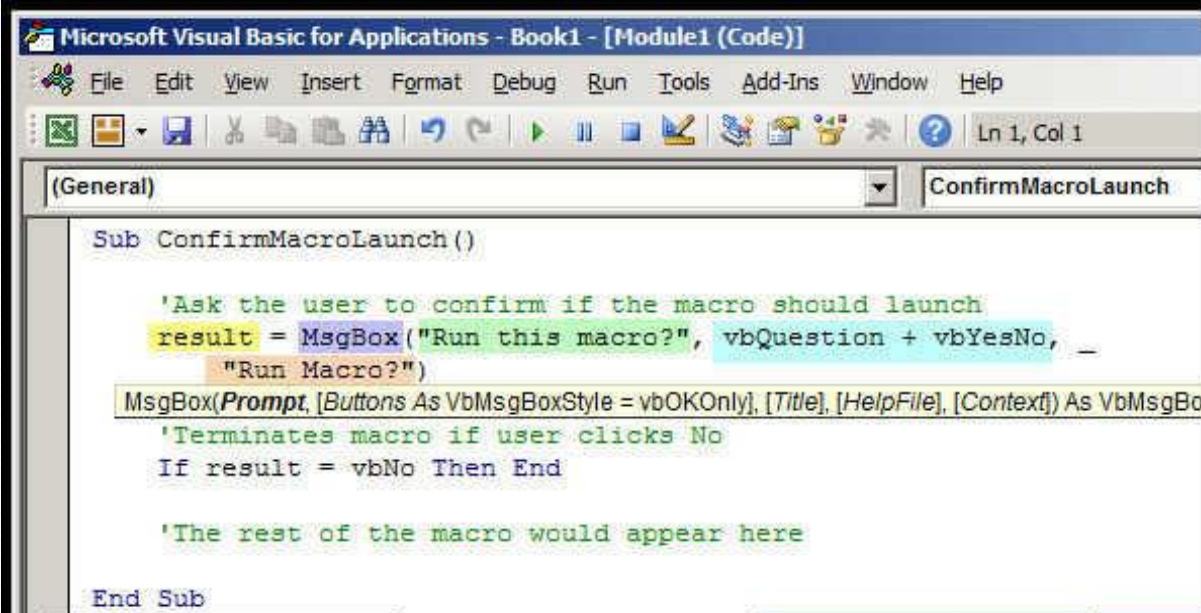
```

Sub RunGoalSeek()
    RunGoalSeek Macro
    Automates the Goal Seek process
    '
    result = InputBox("Payment amount?", "Goal Seek", Range("B4"))
    If result = "" Then Exit Sub
    'Range("B4").GoalSeek Goal
    On Error Resume Next
    Range("B4").GoalSeek
    If Err <> 0 Then
        MsgBox result & "Invalid"
    End If
    On Error GoTo 0
End Sub

```



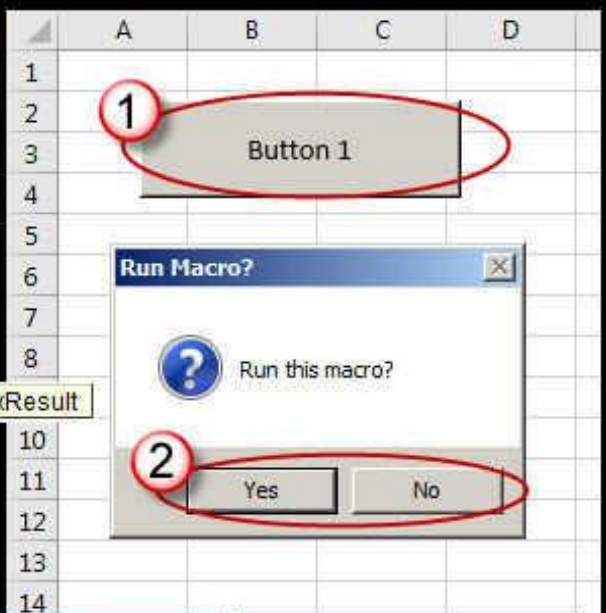
# Confirm User Wants to Run Macro



```

Sub ConfirmMacroLaunch()
    'Ask the user to confirm if the macro should launch
    result = MsgBox("Run this macro?", vbQuestion + vbYesNo, _
        "Run Macro?")
    MsgBox(Prompt, [Buttons As VbMsgBoxStyle = vbOKOnly], [Title], [HelpFile], [Context]) As VbMsgBoxResult
    'Terminates macro if user clicks No
    If result = vbNo Then End
    'The rest of the macro would appear here
End Sub

```



**result**

A user-defined variable where we can store vbYes or vbNo.

**MsgBox**

A method of the Application object, enclosed in parentheses because we're capturing a response.

**"Run this macro?"**

Prompt required by MsgBox.

**vbQuestion + vbNo**

Optional Buttons argument, vbQuestion displays icon, vbYesNo adds buttons.

**"Run Macro?"**

Optional Title argument, if omitted MsgBox title is Microsoft Excel.



# Determine if User Clicked No

Microsoft Visual Basic for Applications - Book1 - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 13, Col 1

(General) ConfirmMacroLaunch

```

Sub ConfirmMacroLaunch ()

    'Ask the user to confirm if the macro should launch
    result = MsgBox("Run this macro?", vbQuestion + vbYesNo, _
        "Run Macro?")

    'Terminates macro if user clicks No
    If result = vbNo Then End

    'The rest of the macro would appear here

End Sub

```

<p><b>If</b></p> <p>Decision-making command in VBA, requires logical test that evaluates to TRUE or FALSE.</p>	<p><b>result</b></p> <p>A user-defined variable where MsgBox stored vbYes or vbNo</p>	<p><b>vbNo</b></p> <p>MsgBox stores vbNo if the user clicks the No button.</p>	<p><b>Then</b></p> <p>Decision-making command in VBA.</p>	<p><b>End</b></p> <p>Terminates the macro. Can sometimes be exchanged for Exit Sub.</p>
--	---	--	---	---





# Prompt User to Choose a File

Microsoft Visual Basic for Applications - Book3 - [Module1 (Code) ...]

File Edit View Insert Format Debug Run Tools Ad...

(General)

```
Sub ChooseFileToOpen ()
    strFile = Application.GetOpenFilename ("*.xls?;*.xlsx?", , "Select Workbook")
    If strFile = False Then Exit Sub
    Workbooks.Open strFile
End Sub
```

As shown in the name, Application.GetOpenFileName doesn't actually open any documents, but instead allows you to display a dialog box from which a user can select one more more file names that can be processed by a macro.

<p><b>strFile</b></p> <p>A user-defined variable name used to store the file name selected (or False if the user clicks Cancel)</p>	<p><b>Application.GetOpenFileName</b></p> <p>A method used to open documents programmatically in Excel</p>	<p><b>"*.xls?;*.xlsx?"</b></p> <p>FileFilter argument used to specify the type of document to open. Include ? to allow the user to choose XLS, XLSX, XLSM, or XLSB files</p>	<p><b>FilterIndex argument isn't needed in this context, so just use a comma as a placeholder</b></p>	<p><b>"Select Workbook"</b></p> <p>Title argument allows you to provide a title for the Open dialog box that appears.</p>
---	--	--	---	---



# Determine if User Clicked Cancel

Microsoft Visual Basic for Applications - Book3 - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 10, Col 1

(General) ChooseFileToOpen

```
Sub ChooseFileToOpen()
    strFile = Application.GetOpenFilename("**.xls?;*.xls?", , "Select Workbook")
    If strFile = False Then Exit Sub
    Workbooks.Open strFile
End Sub
```

<p><b>If</b></p> <p>Decision-making command in VBA, requires logical test that evaluates to TRUE or FALSE.</p>	<p><b>strFile</b></p> <p>A user-defined variable that in this context will contain a file name or the word False.</p>	<p><b>False</b></p> <p>Within VBA the word False in blue text signifies a Boolean value, which is either TRUE or FALSE.</p>	<p><b>Then</b></p> <p>Decision-making command in VBA.</p>	<p><b>Exit Sub</b></p> <p>Terminates the current macro. Unlike End which ends all macros, Exit Sub only exits the current macro.</p>
--	---	---	---	--



# Opening the File the User Chose

Microsoft Visual Basic for Applications - Book3 - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window Help

Ln 10, Col 1

(General) ChooseFileToOpen

```
Sub ChooseFileToOpen()  
    strFile = Application.GetOpenFilename("**.xls?;*.xls?", , "Select Workbook")  
    If strFile = False Then Exit Sub  
    Workbooks.Open strFile  
End Sub
```

**Workbooks.Open**  
In this case the Open method of the Workbooks object will open the file of our choice.

**strFile**  
A user-defined variable that in this context will contain a file name or the word False.



# Notifying User of Macro Completion

Microsoft Visual Basic for Applications - Book3 - [Module1 (Code)]

File Edit View Insert Format Debug Run Tools Add-Ins Window

(General)

```
Sub ChooseFileToOpen()
    strFile = Application.GetOpenFilename("**.xls?;*.xls?", , "Select Workbook")
    If strFile = False Then Exit Sub
    Workbooks.Open strFile
    MsgBox "The file " & strFile & " has been opened.", vbInformation, "Macro complete"
End Sub
```

It's a good practice to include a MsgBox command at the end of a macro so that the user isn't left wondering if the macro has completed or not, especially for both particularly short or especially long macros.

MsgBox	"The file " & strFile & " has been opened"	vbInformation	Title
A method used to display onscreen prompts. Formal wording would be Application.MsgBox	The Prompt argument for MsgBox. Use & to combine pieces of text together	The Buttons prompt for MsgBox, can be used to display icons and/or buttons. vbInformation shows an	The Title prompt is an optional title to appear at the top of the dialog box.



# Saving a Macro-Enabled Workbook

**1** File

**2** Save As

**3** File name: Goal Seek Macro

**4** Save as type: Excel Macro-Enabled Workbook (\*.xism)

**5** Save

**6** Click No and change the Save As Type as shown in Step 4 if you encounter this prompt unless you purposely want to remove macros from a file.

Choose .XLSM, .XLSB, or .XLS here.

The following features cannot be saved in macro-free workbooks:

- VB project

To save a file with these features, click No, and then choose a macro-enabled file type in the File Type list.

To continue saving as a macro-free workbook, click Yes.

Yes No Help

# Thank you for attending!

I'm happy to hear from you. In particular let me know if you did not receive the handouts for this presentation.

**David Ringstrom, CPA**



**[ask@davidringstrom.com](mailto:ask@davidringstrom.com)**



**[www.twitter.com/excelwriter](http://www.twitter.com/excelwriter)**



**[www.linkedin.com/in/davidringstrom](http://www.linkedin.com/in/davidringstrom)**

Is there something you were hoping to learn today but didn't?  
Please let me know. You can ask me anything about Excel.