

Macros In Excel

What is Macro in Excel?

Macro in Excel is a programming language used to automate tasks in Microsoft Excel. It helps users to automate repetitive tasks and create custom applications within Excel. It is a powerful tool that helps users to create complex calculations and automate tasks quickly and easily.

Suppose that every month, you create a report for your accounting manager. You want to format the names of the customers with overdue accounts in red and apply bold formatting. You can create and then run a macro that quickly applies these formatting changes to the cells you select.

Creating Macros in Excel

Creating macros in Excel is a simple process. Users can create macros by recording their actions in Excel or by writing code in the Visual Basic for Applications (VBA) language. Macro code can be written in the VBA editor and then saved as a macro file.

Automate tasks with the Macro Recorder

To automate a repetitive task, you can record a macro with the Macro Recorder in Microsoft Excel. Imagine you have dates in random formats, and you want to apply a single format to all of them. A macro can do that for you. You can record a macro applying the format you want, and then replay the macro whenever needed.

When you record a macro, the macro recorder records all the steps in Visual Basic for Applications (VBA) code. These steps can include typing text or numbers, clicking cells or commands on the ribbon or on menus, formatting cells, rows, or columns, or even importing data from an external source, say, Microsoft Access. Visual Basic Application (VBA) is a subset of the powerful Visual Basic programming language and is included with most Office applications. Although VBA gives you the ability to automate processes within

Macros In Excel

and between Office applications, it is not necessary to know VBA code or computer programming if the Macro Recorder does what you want.

It is important to know that when you record a macro, the Macro Recorder captures almost every move you make. So, if you make a mistake in your sequence, for example, clicking a button that you did not intend to click, the Macro Recorder will record it. The resolution is to re-record the entire sequence or modify the VBA code itself. This is why whenever you record something, it's best to record a process with which you're highly familiar. The more smoothly you record a sequence, the more efficiently the macro will run when you play it back.

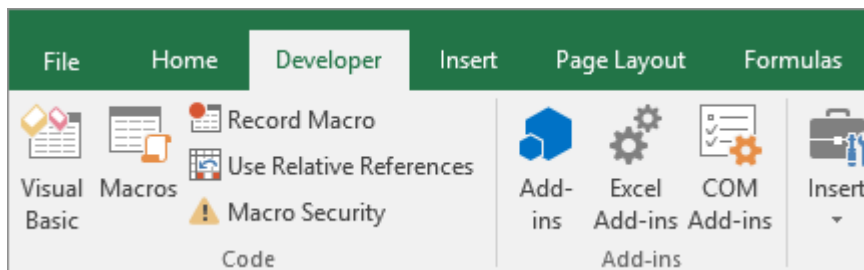
Macros and VBA tools can be found on the **Developer** tab, which is hidden by default, so the first step is to enable it.

Show the Developer tab.

The **Developer** tab isn't displayed by default, but you can add it to the ribbon.

1. On the **File** tab, go to **Options > Customize Ribbon**.
2. Under **Customize the Ribbon** and under **Main Tabs**, select the **Developer** check box.

After you show the tab, the **Developer** tab stays visible, unless you clear the check box or have to reinstall a Microsoft 365 program.



Macros In Excel

Record a macro.

There are a few helpful things you should know about macros:

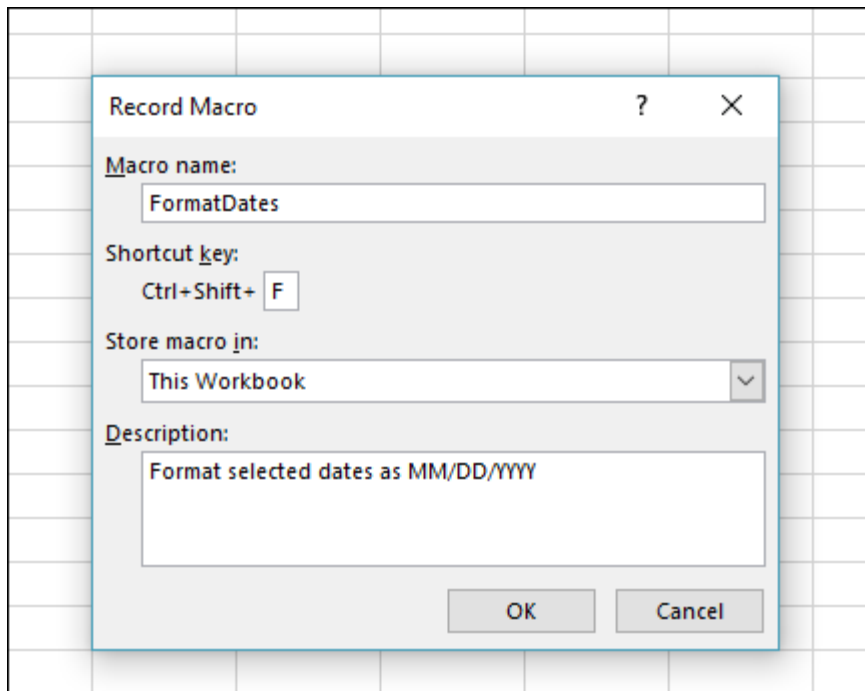
- When you record a macro for performing a set of tasks in a range in Excel, the macro will only run on the cells within the range. So if you added an extra row to the range, the macro will not run the process on the new row, but only the cells within the range.
- If you have planned a long process of tasks to record, plan to have smaller relevant macros instead of having one long macro.
- It is not necessary that only tasks in Excel can be recorded in a macro. Your macro process can extend to other Office applications and any other applications that support Visual Basic Application (VBA). For example, you can record a macro where you first update a table in Excel and then open Outlook to email the table to an email address.

Follow these steps to record a macro.

1. On the **Developer** tab, in the **Code** group, click **Record Macro**.

-OR-

Press **Alt+T+M+R** .



Macros In Excel

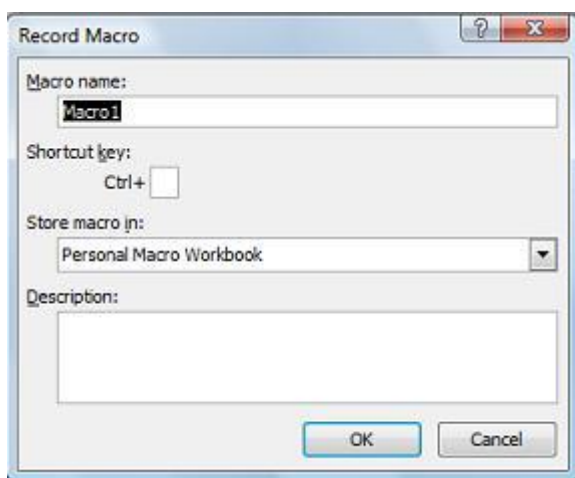
2. In the **Macro name** box, enter a name for the macro. Make the name as descriptive as possible so you can quickly find it if you create more than one macro.

Note: The first character of the macro name must be a letter. Subsequent characters can be letters, numbers, or underscore characters. Spaces cannot be used in a macro name; an underscore character works well as a word separator. If you use a macro name that is also a cell reference, you may get an error message that the macro name is not valid.

3. To assign a keyboard shortcut to run the macro, in the **Shortcut key** box, type any letter (both uppercase and lowercase will work) that you want to use. It is best to use **Ctrl + Shift** (uppercase) key combinations because the macro shortcut key will override any equivalent default Excel shortcut key while the workbook that contains the macro is open. For instance, if you use **Ctrl+Z** (Undo), you will lose the ability to Undo in that Excel instance.
4. In the **Store macro in** the list, select where you want to store the macro.

In general, you'll save your macro in the **This Workbook** location, but if you want a macro to be available whenever you use Excel, select [Personal Macro Workbook](#). When you select **Personal Macro Workbook**, Excel creates a hidden personal macro workbook (Personal.xlsm) if it does not already exist, and saves the macro in this workbook.

In the **Store macro in** the box, select **Personal Macro Workbook**.



Macros In Excel

5. In the **Description** box, optionally type a brief description of what the macro does.

Although the description field is optional, it is recommended you enter one. Also, try to enter a meaningful description with any information that may be useful to you or other users who will be running the macro. If you create a lot of macros, the description can help you quickly identify which macro does what, otherwise you might have to guess.

6. Click **OK** to start recording.
7. Perform the actions that you want to record.

8. On the **Developer** tab, in the **Code** group, click **Stop Recording**.

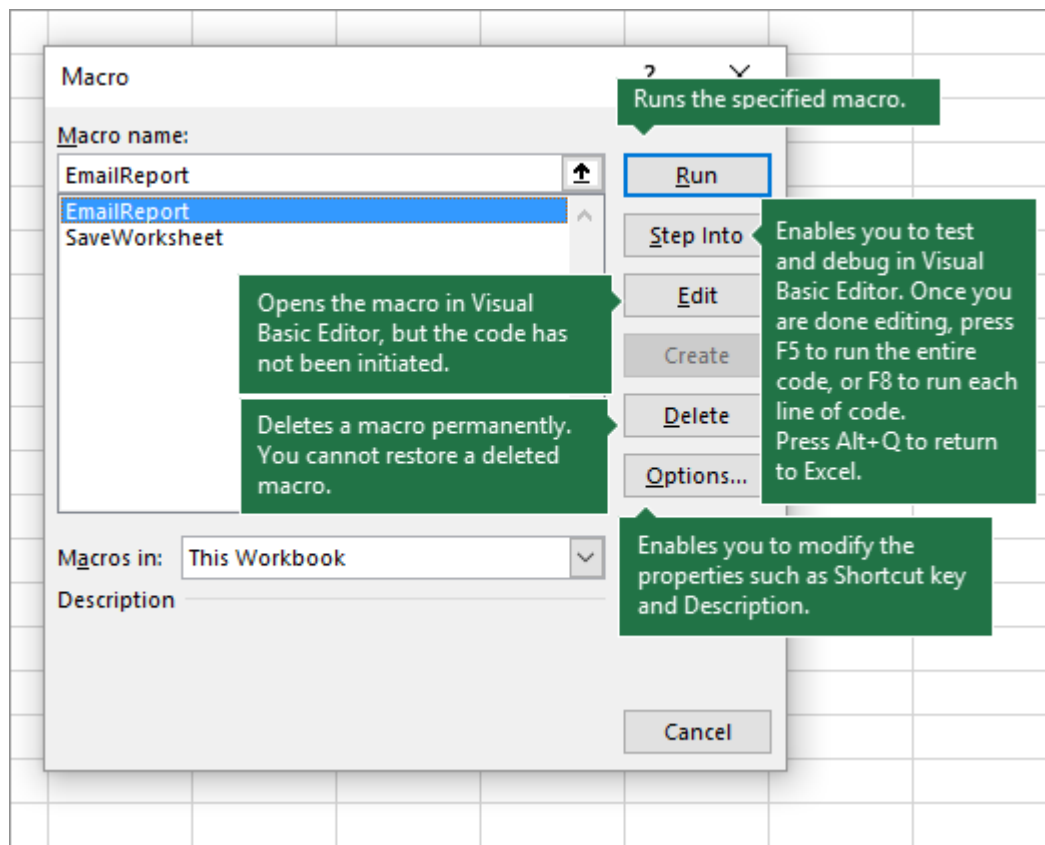
-OR-

Press **Alt+T+M+R**.

Working with recorded macros in Excel

In the **Developer** tab, click **Macros** to view macros associated to a workbook. Or press **Alt+ F8**. This opens the **Macro** dialog box.

Macros In Excel



Caution: Macros cannot be undone. Before you run a recorded macro for the first time, make sure that you've either saved the workbook where you want to run the macro or better yet worked on a copy of the workbook to prevent unwanted changes. If you run a macro and it doesn't do what you want, you can close the workbook without saving it.

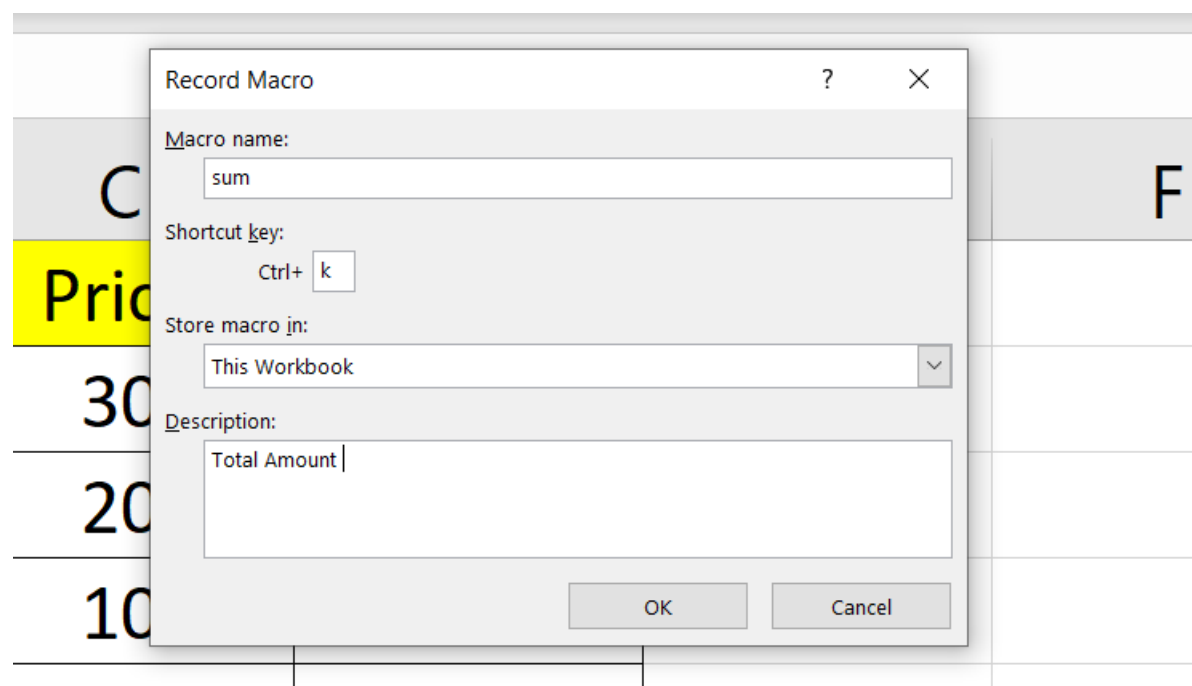
Macros In Excel

Let's Create One Simple Macro to Perform the Total Amount of Product

1.

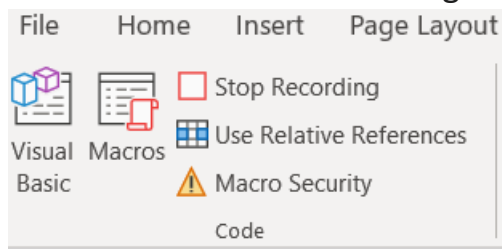
Product	Quantity	Price	Amount
Milk	10	30	
Ice Cream	2	20	
Chips	5	10	
Pasta	2	45	
Noodle	6	80	

2. On the **Developer** tab, in the **Code** group, click **Record Macro**.



Macros In Excel

3. Click on ok to start Recording.



4. Apply Custom Formula in Amount Column.

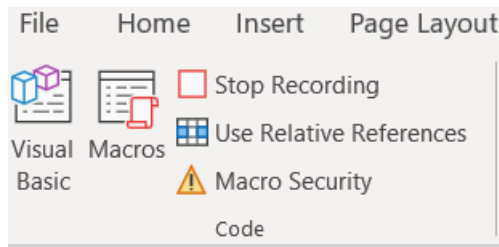
Product	Quantity	Price	Amount
Milk	10	30	=B2*C2
Ice Cream	2	20	
Chips	5	10	
Pasta	2	45	
Noodle	6	80	

- 5.

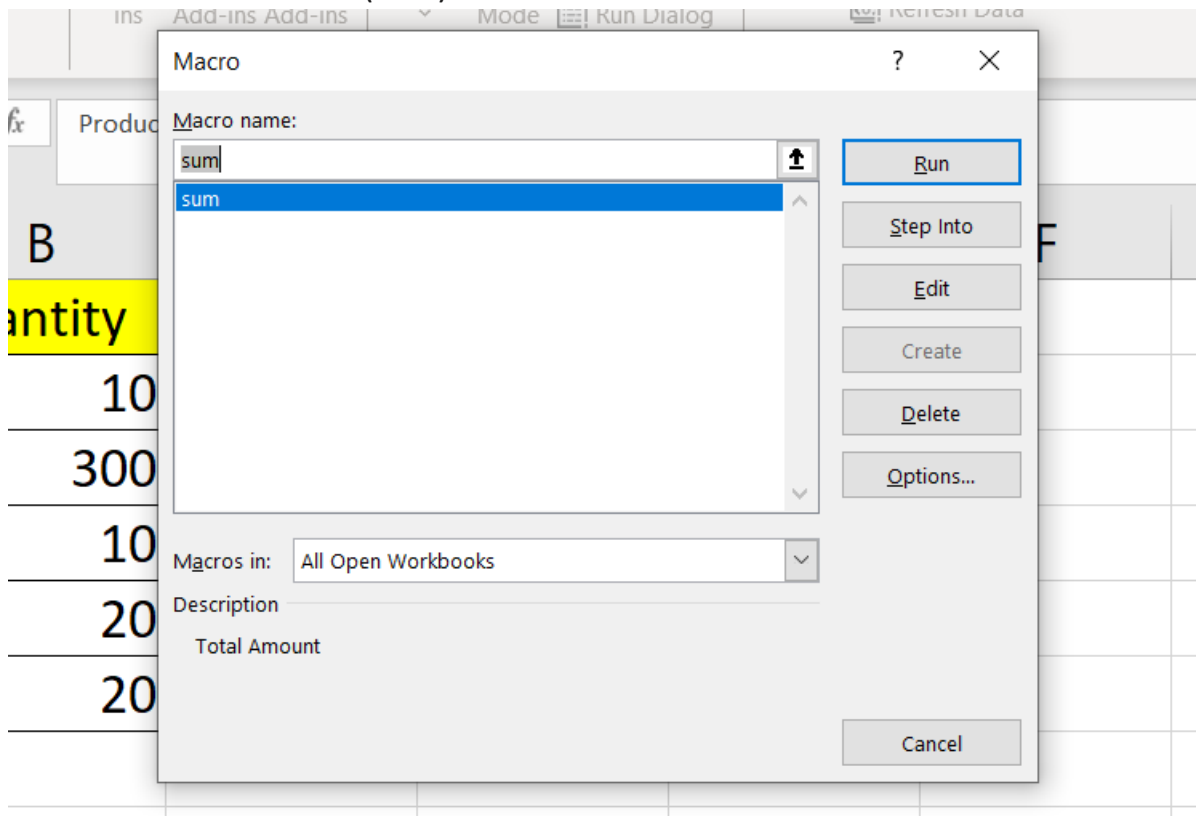
Product	Quantity	Price	Amount
Milk	10	30	300
Ice Cream	2	20	40
Chips	5	10	50
Pasta	2	45	90
Noodle	6	80	480

5. Stop Recording by clicking on Stop Recording

Macros In Excel



6. Now, we had to get a new Record in the same format in the new sheet and we want to calculate the total Amount.
7. It is very easy to do that work.
8. Just click on that sheet, go to the top corner of that sheet, and press Ctrl+k a shortcut key that we have created or go to Developer click on Macros, chose the macro name (sum) then run it.



9.

Macros In Excel

Product	Quantity	Price	Amount
Bolt	10	2	
Screw	300	2	
Tape	10	20	
paint	20	320	
rubber	20	200	

10.

Product	Quantity	Price	Amount
Bolt	10	2	20
Screw	300	2	600
Tape	10	20	200
paint	20	320	6400
rubber	20	200	4000

Remember Macros only work for the number of columns and the number of rows which has been recorded.

Run a macro.

There are several ways to run a macro in Microsoft Excel. A macro is an action or a set of actions that you can use to automate tasks. Macros are recorded in the Visual Basic for Applications programming language. You can always run a macro by clicking the **Macros** command on the **Developer** tab on the ribbon. Depending on how a macro is assigned to run, you might also be able to run it by pressing a combination shortcut key, by clicking a button on the Quick Access Toolbar or in a custom group on the ribbon, or by clicking on an object, graphic, or control. In addition, you can run a macro automatically whenever you open a workbook.

Run a macro from the Developer tab.

1. Open the workbook that contains the macro.
2. On the **Developer** tab, in the **Code** group, click **Macros**.
3. In the **Macro name** box, click the macro that you want to run, and press the **Run** button.
4. You also have other choices:
 - **Options** - Add a shortcut key or a macro description.
 - **Step** - This will open the Visual Basic Editor to the first line of the macro. Pressing **F8** will let you step through the macro code one line at a time.
 - **Edit** - This will open the Visual Basic Editor and let you edit the macro code as needed. Once you've made changes, you can press **F5** to run the macro from the editor.

Run a macro by pressing a combination shortcut key.

You can add a combination shortcut key to a macro when you record it, and you can also add one to an existing macro:

1. On the **Developer** tab, in the **Code** group, click **Macros**.
2. In the **Macro name** box, click the macro that you want to assign to a combination shortcut key.
3. Click **Options**.

The **Macro Options** dialog box appears.

4. In the **Shortcut key** box, type any lowercase or uppercase letter that you want to use with the shortcut key.
5. In the **Description** box, type a description of the macro.
6. Click **OK** to save your changes, and then click **Cancel** to close the **Macro** dialog box.

Macros In Excel

Run a macro by clicking a button on the Quick Access Toolbar

To run a macro from a button on the Quick Access toolbar, you first have to add the button to the toolbar.

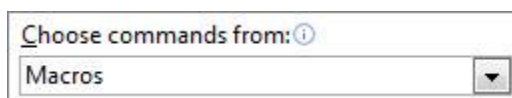
You can use a button (a form control) to run a macro that performs an action when a user clicks it. For example, you might use a button to automate the printing of a worksheet, the filtering of data, or the calculation of numbers.

After you [create a macro](#), you can assign it to a button you click to [run the macro](#). You can assign a macro to a button on the Quick Access Toolbar or to a button in your own personal group on the ribbon.

If you want a macro button to be available in other workbooks, assign it to a macro that was [created in a personal workbook](#).

Add a macro button to the Quick Access Toolbar

1. Click **File > Options > Quick Access Toolbar**.
2. In the **Choose commands from** list, click **Macros**.



3. Select the macro you want to assign a button to.
4. Click **Add** to move the macro to the list of buttons on the Quick Access Toolbar.
5. To replace the default macro icon with a different button for your macro, click **Modify**.
6. Under **Symbol**, select a button icon for your macro.

Macros In Excel



7. To use a friendlier name for the button, in the **Display name** box, enter the name you want.

You can enter a space in the button name.

8. Click **OK** twice.

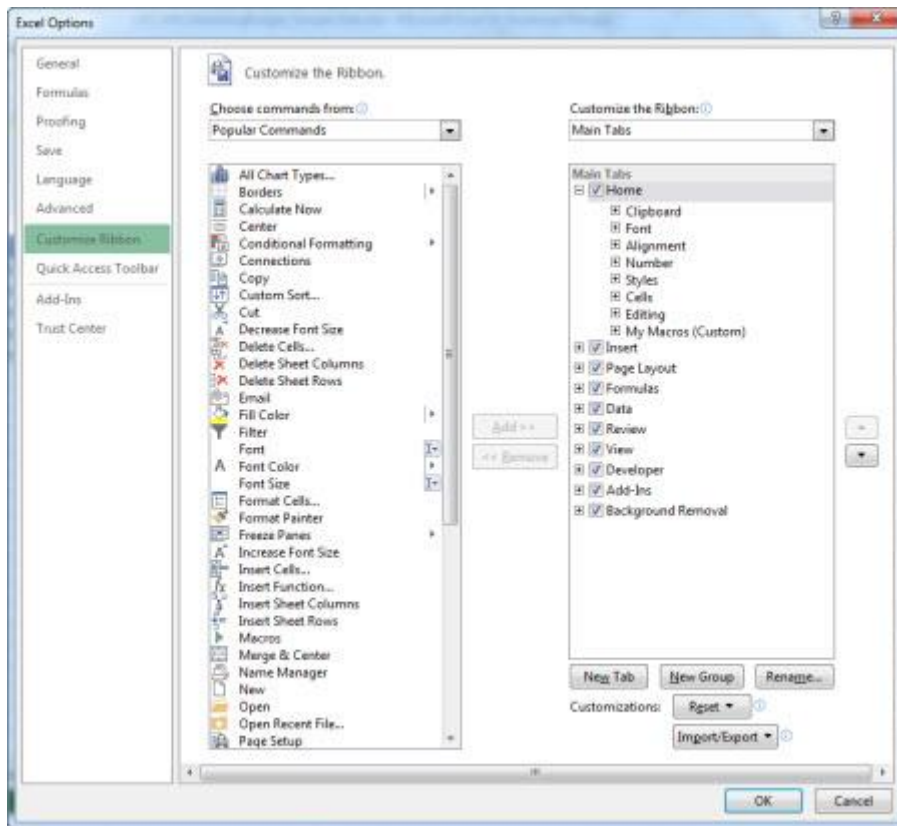
The new button appears on the Quick Access Toolbar, where you can click it to run the macro.

Tip: When you [save the workbook](#), buttons you assign to macros in the personal workbook will be available in every workbook you open.

Add a macro button to your own group on the ribbon

1. Click **File > Options > Customize Ribbon**.
2. Under **Customize the Ribbon**, in the **Main Tabs list**, check the **Developer** box if it is not already checked.

Macros In Excel



3. Pick the tab where you want to add your own group.

For example, pick **Home**, to add your group to the **Home** tab.

4. Select **New Group**.

That adds **New Group (Custom)** to the tab you picked.

5. To use a better name for your new group, click **Rename**, type the name you want in the **Display name** box, and then click **OK**.

You can enter a space in the name. For example, type **My Macros**.

6. To add a macro to the group, in the **Choose commands from** list, click **Macros**.
7. Select the macro you want to add to your new group, and then click **Add**. The macro is added to the **My Macros** group.
8. To use a friendlier name, click **Rename**, and then type the name you want in the **Display name** box.

You can enter a space in the name.

9. Under **Symbol**, select a button icon for your macro.

Macros In Excel

10. Click **OK** twice.


Run a macro by clicking an area on a graphic object.


You can create a hotspot on a graphic that users can click to run a macro.

1. In the worksheet, insert a graphic object, such as a picture, or draw a shape. A common scenario is to draw a Rounded Rectangle shape, and format it so it looks like a button.

To learn about inserting a graphic object, see [Add, change, or delete shapes](#).

2. Right-click the hotspot that you created, and then click **Assign Macro**.
3. Do one of the following:

- To assign an existing macro to the graphic object, double-click the macro or enter its name in the **Macro name** box.
- To record a new macro to assign to the selected graphic object, click **Record**, type a name for the macro in the **Record Macro** dialog box, and then click **OK** to begin recording your macro. When you finish recording the macro, click **Stop Recording**  on the **Developer** tab in the **Code** group.

Tip: You can also click **Stop Recording**  on the left side of the status bar.

- To edit an existing macro, click the name of the macro in the **Macro name** box, and then click **Edit**.
4. Click **OK**.

Run a macro from the Visual Basic Editor (VBE)

On the **Developer** tab, click Visual Basic to launch the **Visual Basic Editor (VBE)**. Browse the **Project Explorer** to the module that contains the macro you want to run, and open it. All of the macros in that module will be listed in the pane on the right. Select the macro you want to run, by placing your cursor anywhere within the macro, and press **F5**, or on the menu, go to **Run > Run Macro**.

Configure a macro to run automatically upon opening a workbook.

Create a Workbook Open event.

The following example uses the **Open** event to run a macro when you open the workbook.

1. Open the workbook where you want to add the macro, or create a new workbook.
2. On the **Developer** tab, in the **Code** group, click **Visual Basic**.
3. In the **Project Explorer** window, right-click the **This Workbook** object, and then click **View Code**.

Tip: If the Project Explorer window is not visible, on the **View** menu, click **Project Explorer**.

4. In the **Object** list above the Code window, select **Workbook**.

This automatically creates an empty procedure for the **Open** event, such as this:

```
Private Sub Workbook_Open()
```

```
End Sub
```

5. Add the following lines of code to the procedure:

```
Private Sub Workbook_Open()  
MsgBox Date  
Worksheets("Sheet1").Range("A1").Value = Date  
End Sub
```


Macros In Excel

6. Switch to Excel and save the workbook as a macro-enabled workbook (.xlsm).
7. Close and reopen the workbook. When you open the workbook again, Excel runs the **Workbook_Open** procedure, which displays today's date in a message box.
8. Click **OK** in the message box.

Edit a macro.

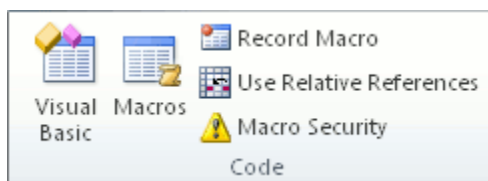
To edit a macro that is attached to a Microsoft Excel workbook, you use the Visual Basic Editor.

Important: Before you can work with macros, you have to enable the Developer tab. For more information, see [Show the Developer tab](#).

Change macro security settings

To edit and run macros, you must set the security level to temporarily enable all macros:

1. On the **Developer** tab, in the **Code** group, click **Macro Security**.



2. Under **Macro Settings**, click **Enable all macros (not recommended, potentially dangerous code can run)**, and then click **OK**.

Warning: To help prevent potentially dangerous code from running, we recommend that you return to any of the settings that disable all macros after you finish working with macros.

Macros In Excel

Edit the macro.

1. On the **Developer** tab, in the **Code** group, click **Macros**.
2. In the **Macro name** box, click the macro that you want to edit.
3. Click **Edit**. The Visual Basic Editor appears.

Copy a macro module to another workbook.

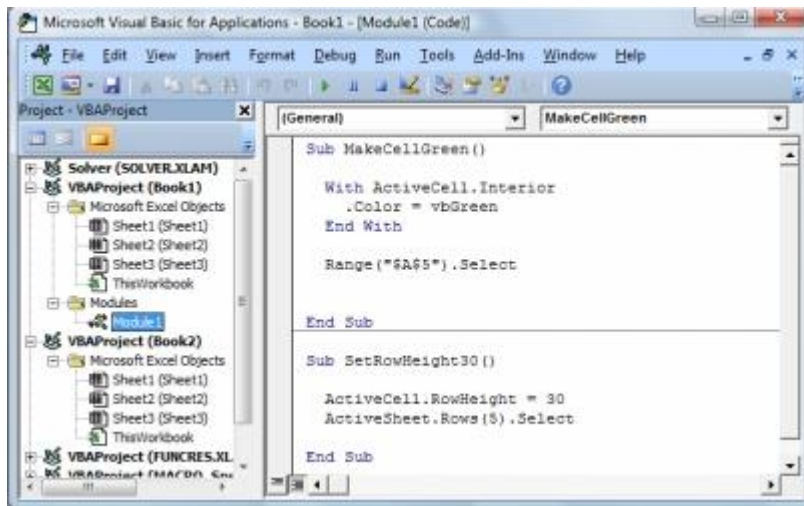
If a workbook contains a **Visual Basic for Applications (VBA)** macro that you would like to use elsewhere, you can copy the module that contains that macro to another open workbook by using the **Visual Basic Editor (VBE)**.

Overview of macros and VBA

If you are unfamiliar with macros and VBA in general, you may find the following information helpful.


- A macro is an action or set of actions that you can use to automate tasks.
- You can record macros by using the **Record Macro** command on the **Developer** tab.
- Macros are recorded in the VBA programming language.
- You can inspect and edit your macros in the Visual Basic Editor, a window that is opened by Excel. Here's an example of the VBE window

Macros In Excel

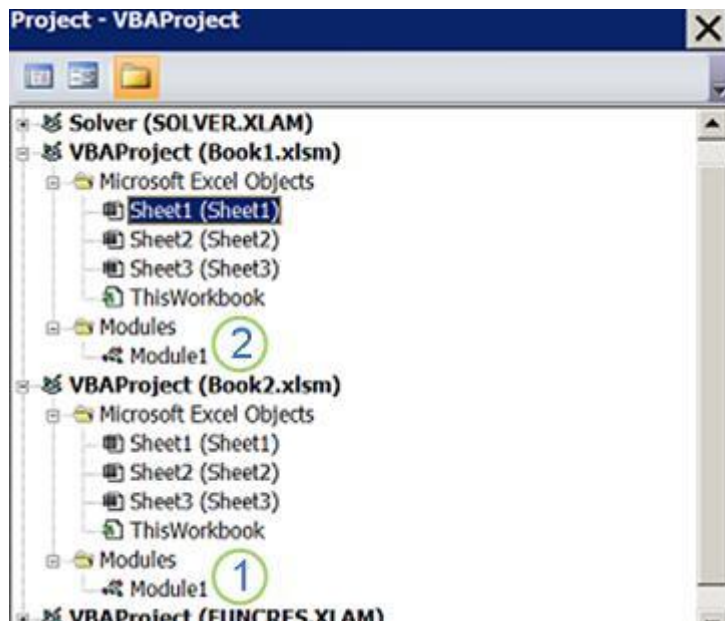


Macros named ***MakeCellGreen*** and ***SetRowHeight*** are in a module named **Module1**, which is stored in Book1.

Copying a module from one workbook to another

1. Open both the workbook that contains the macro you want to copy, and the workbook where you want to copy it.
2. On the **Developer** tab, click **Visual Basic** to open the **Visual Basic Editor**.
3. In the Visual Basic Editor, on the **View** menu, click **Project Explorer** , or press **CTRL+R**.
4. In the **Project Explorer** pane, drag the module containing the macro you want to copy to the destination workbook. In this case, we're copying Module1 from Book2.xlsm to Book1.xlsm.

Macros In Excel



1. Module1 copied from Book2.xlsm
2. Copy of Module1 copied to Book1.xlsm

Assign a macro to an object, shape or graphic.

1. On a worksheet, right-click the object, graphic, shape, or the item to which you want to assign an existing macro, and then click **Assign Macro**.
2. In the **Assign Macro** box, click the macro that you want to assign.

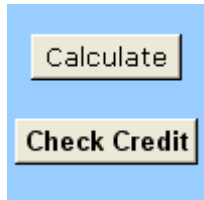
Assign a macro to a Form or a Control button.

You can use a Form control button or a command button (an ActiveX control) to run a macro that performs an action when a user clicks it. Both these buttons are also known as a push button, which can be set up to automate the printing

Macros In Excel

of a worksheet, filtering data, or calculating numbers. In general, a Form control button and an ActiveX control command button are similar in appearance and function. However, they do have a few differences, which are explained in the following sections.

Button (Form control)

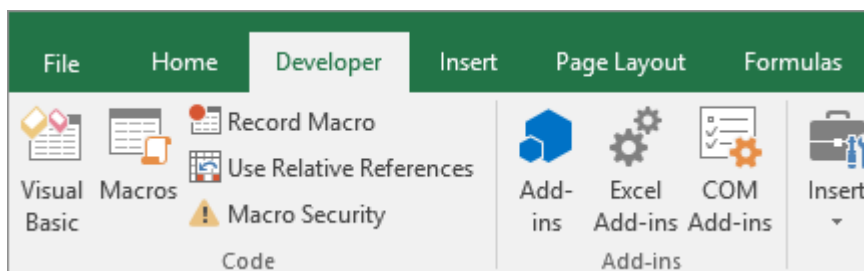


Command button (ActiveX control)




Macros and VBA tools can be found on the **Developer** tab, which is hidden by default.

The first step is to enable it. For more information, see the article: [Show the Developer tab](#).




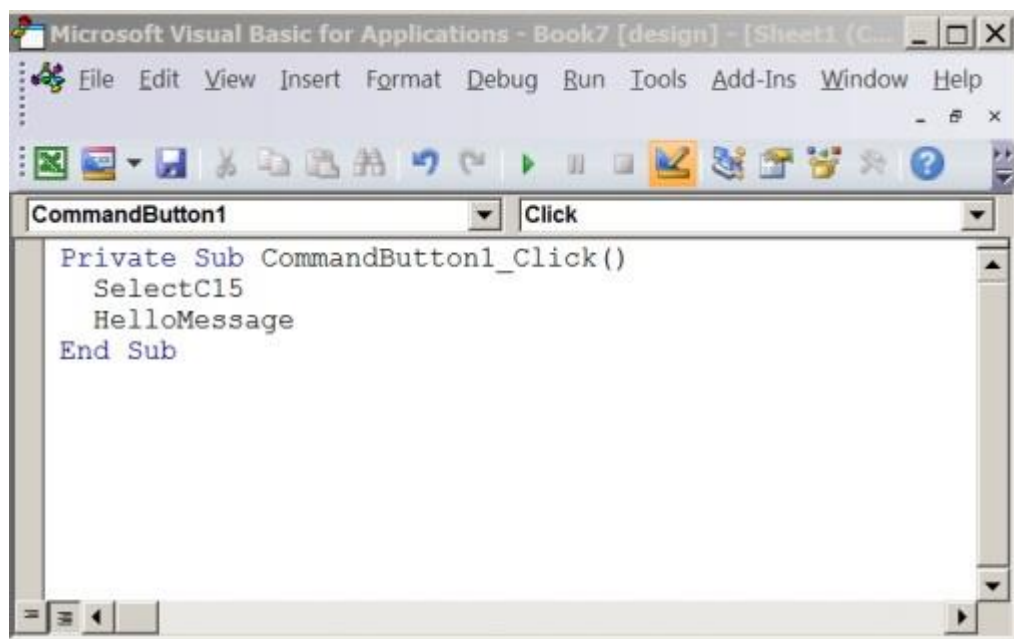
Add a button (Form control)


1. On the **Developer** tab, in the **Controls** group, click **Insert**, and then under **Form Controls**, click **Button** .
2. Click the worksheet location where you want the upper-left corner of the button to appear. The **Assign Macro** popup window appears.
3. Assign a macro to the button, and then click **OK**.
4. To specify the control properties of the button, right-click the button, and then click **Format Control**.

Macros In Excel


Add a command button (ActiveX control)

1. On the **Developer** tab, in the **Controls** group, click **Insert**, and then under **ActiveX Controls**, click **Command Button** .
2. Click the worksheet location at which you want the upper-left corner of the command button to appear.
3. In the **Controls** group, click **View Code**. This launches the Visual Basic Editor. Ensure that **Click** is chosen in the drop-down list on the right. The sub procedure **CommandButton1_Click** (see the figure below) runs these two macros when the button is clicked: **SelectC15** and **HelloMessage**.



4. In the subprocedure for the command button, do either of the following:
 - Enter the name of an existing macro in the workbook. You can find macros by clicking **Macros** in the **Code** group. You can run multiple macros from a button by entering the macro names on separate lines inside the subprocedure.
 - As necessary, add your own VBA code.
5. Close the Visual Basic Editor, and click **Design Mode**  to ensure design mode is off.
6. To run the VBA code that is now part of the button, click the ActiveX command button that you just created.
7. To edit the ActiveX control, make sure that you are in design mode. On the **Developer** tab, in the **Controls** group, turn on **Design Mode**.

Macros In Excel

8. To specify the control properties of the command button, on the **Developer** tab, in the **Controls** group, click **Properties** . You can also right-click the command button, and then click **Properties**.

Note: Before you click **Properties**, make sure that the object for which you want to examine or change properties is already selected.

The **Properties** box appears. For detailed information about each property, select the property, and then press F1 to display a Visual Basic Help topic. You can also type the property name in the Visual Basic Help **Search** box. The following table summarizes the properties that are available.

If you want to specify	Use this property
General:	
Whether the control is loaded when the workbook is opened. (Ignored for ActiveX controls.)	AutoLoad (Excel)
Whether the control can receive focus and respond to user-generated events.	Enabled (Form)
Whether the control can be edited.	Locked (Form)
The name of the control.	Name (Form)
The way the control is attached to the cells below it (free floating, move but do not size, or move and size).	Placement (Excel)
Whether the control can be printed.	PrintObject (Excel)
Whether the control is visible or hidden.	Visible (Form)
Text:	
Font attributes (bold, italic, size, strikethrough, underline, and weight).	Bold, Italic, Size, StrikeThrough, Underline, Weight (Form)
Descriptive text on the control that identifies or describes it.	Caption (Form)
Whether the contents of the control automatically wrap at the end of a line.	WordWrap (Form)

Macros In Excel

If you want to specify	Use this property
Size and Position:	
Whether the size of the control automatically adjusts to display all the contents.	AutoSize (Form)
The height or width in points.	Height, Width (Form)
The distance between the control and the left or top edge of the worksheet.	Left, Top (Form)
Formatting:	
The background color.	BackColor (Form)
The background style (transparent or opaque).	BackStyle (Form)
The foreground color.	ForeColor (Form)
Whether the control has a shadow.	Shadow (Excel)
Image:	
The bitmap to display in the control.	Picture (Form)
The location of the picture relative to its caption (left, top, right, and so on).	PicturePosition (Form)
Keyboard and Mouse:	
The shortcut key for the control.	Accelerator (Form)
A custom mouse icon.	Mouselcon (Form)
The type of pointer that is displayed when the user positions the mouse over a particular object (standard, arrow, I-beam, and so on).	MousePointer (Form)
Whether the control takes the focus when clicked.	TakeFocusOnClick (Form)

Macros In Excel