

- » Why use macros
- » Recording macros
- » Understanding macro security
- » Examples of macros in action

Chapter **1**

Macro Fundamentals

A *macro* is essentially a set of instructions or code that you create to tell Excel to execute any number of actions. In Excel, macros can be written or recorded. The key word here is recorded.

Recording a macro is like programming a phone number into your cell phone. You first manually dial and save a number. Then when you want, you can redial those numbers with the touch of a button. Just as on a cell phone, you can record your actions in Excel while you perform them. While you record, Excel gets busy in the background, translating your keystrokes and mouse clicks to written code (also known as Visual Basic for Applications (VBA)). After a macro is recorded, you can play back those actions anytime you want.

In this chapter, you'll explore macros and learn how you can use macros to automate your recurring processes to simplify your life.

Why Use a Macro?

The first step in using macros is admitting you have a problem. Actually, you may have several problems:

- » **Problem 1 - Repetitive tasks:** As each new month rolls around, you have to make the donuts (that is, crank out those reports). You have to import that data.

You have to update those PivotTables. You have to delete those columns, and so on. Wouldn't it be nice if you could fire up a macro and have those more redundant parts of your dashboard processes done automatically?

- » **Problem 2 - You're making mistakes:** When you go hand-to-hand combat with Excel, you're bound to make mistakes. When you're repeatedly applying formulas, sorting, and moving things around manually, there's always that risk of catastrophe. Add to that the looming deadlines and constant change requests, and your error rate goes up. Why not calmly record a macro, ensure that everything is running correctly, and then forget it? The macro is sure to perform every action the same way every time you run it, reducing the chance of errors.
- » **Problem 3 - Awkward navigation:** You often create reports for an audience that probably has a limited knowledge of Excel. It's always helpful to make your reports more user-friendly. Macros can be used to dynamically format and print worksheets, navigate to specific sheets in your workbook, or even save the open document in a specified location. Your audience will appreciate these little touches that help make perusal of your workbooks a bit more pleasant.

Macro Recording Basics

To start recording your first macro, you need to first find the Macro Recorder, which is on the Developer tab. Unfortunately, Excel comes out of the box with the Developer tab hidden — you may not see it on your version of Excel at first. If you plan to work with VBA macros, you'll want to make sure that the Developer tab is visible. To display this tab

1. Choose File ⇨ Excel Options.
2. In the Excel Options dialog box, select Customize Ribbon.
3. In the list box on the right, place a check mark next to Developer.
4. Click OK to return to Excel.

Now that you have the Developer tab showing in the Excel Ribbon, you can start up the Macro Recorder by selecting Record Macro from the Developer tab. This activates the Record Macro dialog box, as shown in Figure 1-1.

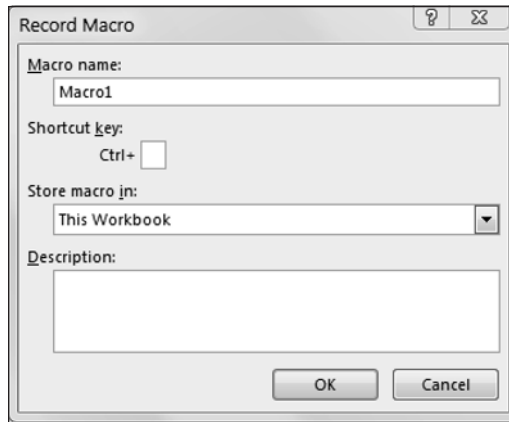


FIGURE 1-1:
The Record
Macro dialog box.

Here are the four parts of the Record Macro dialog box:

- » **Macro Name:** This should be self-explanatory. Excel gives a default name to your macro, such as Macro1, but you should give your macro a name more descriptive of what it actually does. For example, you might name a macro that formats a generic table as FormatTable.
- » **Shortcut Key:** Every macro needs an event, or something to happen, for it to run. This event can be a button press, a workbook opening, or in this case, a keystroke combination. When you assign a shortcut key to your macro, entering that combination of keys triggers your macro to run. This is an optional field.
- » **Store Macro In:** This Workbook is the default option. Storing your macro in This Workbook simply means that the macro is stored along with the active Excel file. The next time you open that particular workbook, the macro is available to run. Similarly, if you send the workbook to another user, that user can run the macro as well (provided the macro security is properly set by your user — more on that later in this chapter).
- » **Description:** This is an optional field, but it can come in handy if you have numerous macros in a spreadsheet or if you need to give a user a more detailed description about what the macro does.

With the Record Macro dialog box open, follow these steps to create a simple macro that enters your name into a worksheet cell:

- 1. Enter a new single-word name for the macro to replace the default Macro1 name.**

A good name for this example is MyName.

2. Assign this macro to the shortcut key Ctrl+Shift+N.

You do this by entering uppercase N in the edit box labeled Shortcut Key.

3. Click OK.

This closes the Record Macro dialog box and begins recording your actions.

4. Select any cell on your Excel spreadsheet, type your name into the selected cell, and then press Enter.

5. Choose Developer ⇨ Code ⇨ Stop Recording (or click the Stop Recording button in the status bar).

Examining the macro

The macro was recorded in a new module named Module1. To view the code in this module, you must activate the Visual Basic Editor. You can activate the VB Editor in either of two ways:

- » Press Alt+F11.
- » Choose Developer ⇨ Code ⇨ Visual Basic.

In the VB Editor, the Project window displays a list of all open workbooks and add-ins. This list is displayed as a tree diagram, which you can expand or collapse. The code that you recorded previously is stored in Module1 in the current workbook. When you double-click Module1, the code in the module appears in the Code window.

The macro should look something like this:

```
Sub MyName()  
,  
,  
    ' MyName Macro  
,  
    ' Keyboard Shortcut: Ctrl+Shift+N  
,  
    ActiveCell.FormulaR1C1 = "Michael Alexander"  
  
End Sub
```

The macro recorded is a Sub procedure named MyName. The statements tell Excel what to do when the macro is executed.

Notice that Excel inserted some comments at the top of the procedure. These comments are some of the information that appeared in the Record Macro dialog

box. These comment lines (which begin with an apostrophe) aren't really necessary, and deleting them has no effect on how the macro runs. If you ignore the comments, you'll see that this procedure has only one VBA statement:

```
ActiveCell.FormulaR1C1 = "Michael Alexander"
```

This single statement causes the name you typed while recording to be inserted into the active cell.

Testing the macro

Before you recorded this macro, you set an option that assigned the macro to the Ctrl+Shift+N shortcut key combination. To test the macro, return to Excel by using either of the following methods:

- » Press Alt+F11.
- » Click the View Microsoft Excel button on the VB Editor toolbar.

When Excel is active, activate a worksheet. (It can be in the workbook that contains the VBA module or in any other workbook.) Select a cell and press Ctrl+Shift+N. The macro immediately enters your name into the cell.



REMEMBER

In the preceding example, notice that you selected the cell to be altered before you started recording your macro. This step is important. If you select a cell while the macro recorder is turned on, the actual cell that you selected is recorded into the macro. In such a case, the macro would always format that particular cell, and it would not be a general-purpose macro.

Editing the macro

After you record a macro, you can make changes to it (although you must know what you're doing). For example, assume that you want your name to be bold. You could re-record the macro, but this modification is simple, so editing the code is more efficient. Press Alt+F11 to activate the VB Editor window. Then activate Module1 and insert the following statement before the End Sub statement:

```
ActiveCell.Font.Bold = True
```

The edited macro appears as follows:

```
Sub MyName()  
,
```

```
' MyName Macro
'
' Keyboard Shortcut: Ctrl+Shift+N
'
    ActiveCell.Font.Bold = True

    ActiveCell.FormulaR1C1 = "Michael Alexander"

End Sub
```

Test this new macro, and you see that it performs as it should.

Comparing Absolute and Relative Macro Recording

Now that you've read about the basics of the Macro Recorder interface, it's time to go deeper and begin recording macros. The first thing you need to understand before you begin is that Excel has two modes for recording — absolute reference and relative reference.

Recording macros with absolute references

Excel's default recording mode is in absolute reference. As you may know, the term absolute reference is often used in the context of cell references found in formulas. When a cell reference in a formula is an absolute reference, it does not automatically adjust when the formula is pasted to a new location.

The best way to understand how this concept applies to macros is to try it out. Open the Chapter 1 Sample File.xlsx file and record a macro that counts the rows in the Branchlist worksheet. (See Figure 1-2.)



TIP

The sample dataset used in this chapter can be found on this book's companion website at www.dummies.com/go/excelmacros.

Follow these steps to record the macro:

1. **Before recording, make sure cell A1 is selected.**
2. **Select Record Macro from the Developer tab.**
3. **Name the macro AddTotal.**
4. **Choose This Workbook for the save location.**

FIGURE 1-2:
Your pre-totaled
worksheet
containing two
tables.

	A	B	C	D	E	F	G	H	I
1		Region	Market	Branch			Region	Market	Branch
2		NORTH	BUFFALO	601419			SOUTH	CHARLOTTE	173901
3		NORTH	BUFFALO	701407			SOUTH	CHARLOTTE	301301
4		NORTH	BUFFALO	802202			SOUTH	CHARLOTTE	302301
5		NORTH	CANADA	910181			SOUTH	CHARLOTTE	601306
6		NORTH	CANADA	920681			SOUTH	DALLAS	202600
7		NORTH	MICHIGAN	101419			SOUTH	DALLAS	490260
8		NORTH	MICHIGAN	501405			SOUTH	DALLAS	490360
9		NORTH	MICHIGAN	503405			SOUTH	DALLAS	490460
10		NORTH	MICHIGAN	590140			SOUTH	FLORIDA	301316
11		NORTH	NEWYORK	801211			SOUTH	FLORIDA	701309
12		NORTH	NEWYORK	802211			SOUTH	FLORIDA	702309
13		NORTH	NEWYORK	804211			SOUTH	NEWORLEANS	601310
14		NORTH	NEWYORK	805211			SOUTH	NEWORLEANS	602310
15		NORTH	NEWYORK	806211			SOUTH	NEWORLEANS	801607

5. Click OK to start recording.

At this point, Excel is recording your actions. While Excel is recording, perform the following steps:

1. Select cell A16 and type **Total** in the cell.
2. Select the first empty cell in Column D (D16) and enter = **COUNTA(D2:D15)**.

This gives a count of branch numbers at the bottom of column D. You need to use the COUNTA function because the branch numbers are stored as text.

3. Click **Stop Recording** on the **Developer** tab to stop recording the macro.

The formatted worksheet should look something like the one in Figure 1-3.

	A	B	C	D	E	F	G	H	I
1		Region	Market	Branch			Region	Market	Branch
2		NORTH	BUFFALO	601419			SOUTH	CHARLOTTE	173901
3		NORTH	BUFFALO	701407			SOUTH	CHARLOTTE	301301
4		NORTH	BUFFALO	802202			SOUTH	CHARLOTTE	302301
5		NORTH	CANADA	910181			SOUTH	CHARLOTTE	601306
6		NORTH	CANADA	920681			SOUTH	DALLAS	202600
7		NORTH	MICHIGAN	101419			SOUTH	DALLAS	490260
8		NORTH	MICHIGAN	501405			SOUTH	DALLAS	490360
9		NORTH	MICHIGAN	503405			SOUTH	DALLAS	490460
10		NORTH	MICHIGAN	590140			SOUTH	FLORIDA	301316
11		NORTH	NEWYORK	801211			SOUTH	FLORIDA	701309
12		NORTH	NEWYORK	802211			SOUTH	FLORIDA	702309
13		NORTH	NEWYORK	804211			SOUTH	NEWORLEANS	601310
14		NORTH	NEWYORK	805211			SOUTH	NEWORLEANS	602310
15		NORTH	NEWYORK	806211			SOUTH	NEWORLEANS	801607
16	Total			14					

FIGURE 1-3:
Your post-totaled
worksheet.

To see your macro in action, delete the total row you just added and play back your macro by following these steps:

1. **Select Macros from the Developer tab.**
2. **Find and select the AddTotal macro you just recorded.**
3. **Click the Run button.**

If all goes well, the macro plays back your actions to a T and gives your table a total. Now here's the thing: No matter how hard you try, you can't make the AddTotal macro work on the second table (G1:I15 in Figure 1-3). Why? Because you recorded it as an absolute macro.

To understand what this means, examine the underlying code. To examine the code, select Macros from the Developer tab to get the Macro dialog box you see in Figure 1-4.

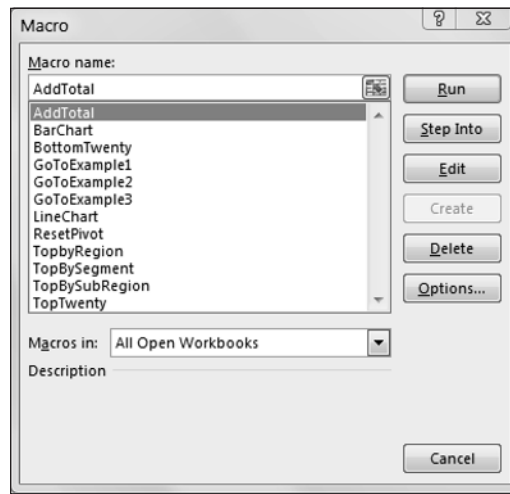


FIGURE 1-4:
The Excel Macro dialog box.

Select the AddTotal macro and click the Edit button. This opens the Visual Basic Editor to show you the code that was written when you recorded your macro:

```
Sub AddTotal()  
  
    Range("A16").Select  
  
    ActiveCell.FormulaR1C1 = "Total"  
  
    Range("D16").Select  
  
End Sub
```



```
ActiveCell.FormulaR1C1 = "=COUNTA(R[-14]C:R[-1]C)"
```

```
End Sub
```

Pay particular attention to lines 2 and 4 of the macro. When you asked Excel to select cell range A16 and then D16, those cells are exactly what it selected. Because the macro was recorded in absolute reference mode, Excel interpreted your range selection as absolute. In other words, if you select cell A16, that cell is what Excel gives you. In the next section, you take a look at what the same macro looks like when recorded in relative reference mode.

Recording macros with relative references

In the context of Excel macros, relative means relative to the currently active cell. So you should use caution with your active cell choice — both when you record the relative reference macro and when you run it.

First, make sure the Chapter 1 Sample File.xlsx file is open. Then, use the following steps to record a relative-reference macro:



REMEMBER

To download the Chapter 1 Sample file, visit www.dummies.com/go/excelmacros.

1. **Select the Use Relative References option from the Developer tab, as shown in Figure 1-5.**
2. **Before recording, make sure cell A1 is selected.**
3. **Select Record Macro from the Developer tab.**
4. **Name the macro AddTotalRelative.**
5. **Choose This Workbook for the save location.**
6. **Click OK to start recording.**
7. **Select cell A16 and type Total in the cell.**
8. **Select the first empty cell in Column D (D16) and type = COUNTA(D2:D15).**
9. **Click Stop Recording on the Developer tab to stop recording the macro.**

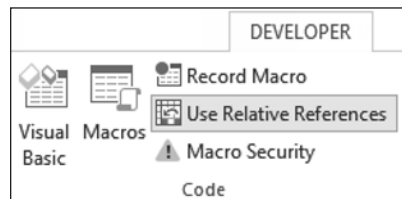


FIGURE 1-5: Recording a macro with relative references.

At this point, you have recorded two macros. Take a moment to examine the code for your newly created macro.

Select Macros from the Developer tab to open the Macro dialog box. Here, choose the AddTotalRelative macro and click Edit.

Again, this opens the Visual Basic Editor to show you the code that was written when you recorded your macro. This time, your code looks something like the following:

```
Sub AddTotalRelative()  
  
    ActiveCell.Offset(15, 0).Range("A1").Select  
  
    ActiveCell.FormulaR1C1 = "Total"  
  
    ActiveCell.Offset(0, 3).Range("A1").Select  
  
    ActiveCell.FormulaR1C1 = "=COUNTA(R[-14]C:R[-1]C)"  
  
End Sub
```

Notice that there are no references to any specific cell ranges at all (other than the starting point "A1"). Let's take a quick look at what the relevant parts of this VBA code really mean.

Notice that in line 2, Excel uses the Offset property of the active cell. This property tells the cursor to move a certain number of cells up or down and a certain number of cells left or right.

The Offset property code tells Excel to move 15 rows down and 0 columns across from the active cell (in this case, A1). There's no need for Excel to explicitly select a cell as it did when recording an absolute reference macro.

To see this macro in action, delete the total row for both tables and do the following:

- 1. Select cell A1.**
- 2. Select Macros from the Developer tab.**
- 3. Find and select the AddTotalRelative macro.**
- 4. Click the Run button.**
- 5. Now select cell F1.**
- 6. Select Macros from the Developer tab.**

7. Find and select the `AddTotalRelative` macro.
8. Click the Run button.

Notice that this macro, unlike your previous macro, works on both sets of data. Because the macro applies the totals relative to the currently active cell, the totals are applied correctly.

For this macro to work, you simply need to ensure that

- » You've selected the correct starting cell before running the macro.
- » The block of data has the same number of rows and columns as the data on which you recorded the macro.

Hopefully, this simple example has given you a firm grasp of macro recording with both absolute and relative references.

Other Macro Recording Concepts

At this point, you should feel comfortable recording your own Excel macros. Here are some of other important concepts you'll need to keep in mind when working with macros.

Macro-enabled file extensions

Beginning with Excel 2007, Excel workbooks were given the standard file extension `.xlsx`. Files with the `.xlsx` extension cannot contain macros. If your workbook contains macros and you then save that workbook as an `.xlsx` file, your macros are removed automatically. Excel warns you that macro content will be removed when saving a workbook with macros as an `.xlsx` file.

If you want to retain the macros, you must save your file as an Excel Macro-Enabled Workbook. This gives your file an `.xlsm` extension. The idea is that all workbooks with an `.xlsx` file extension are automatically known to be safe, whereas you can recognize `.xlsm` files as a potential threat.

Macro security in Excel 2010

With the release of Office 2010, Microsoft introduced significant changes to its Office security model. One of the most significant changes is the concept of trusted

documents. Without getting into the technical minutia, a trusted document is essentially a workbook you have deemed safe by enabling macros.

If you open a workbook that contains macros in Excel 2010, you see a yellow bar message under the Ribbon stating that macros (active content) have, in effect, been disabled.

If you click Enable, it automatically becomes a trusted document. This means you no longer are prompted to enable the content as long as you open that file on your computer. The basic idea is that if you told Excel that you “trust” a particular workbook by enabling macros, it is highly likely that you will enable macros each time you open it. Thus, Excel remembers that you’ve enabled macros before and inhibits any further messages about macros for that workbook.

This is great news for you and your clients. After enabling your macros just one time, they won’t be annoyed at the constant messages about macros, and you won’t have to worry that your macro-enabled dashboard will fall flat because macros have been disabled.

Trusted locations

If the thought of any macro message coming up (even one time) unnerves you, you can set up a trusted location for your files. A trusted location is a directory that is deemed a safe zone where only trusted workbooks are placed. A trusted location allows you and your clients to run a macro-enabled workbook with no security restrictions as long as the workbook is in that location.

To set up a trusted location, follow these steps:

- 1. Select the Macro Security button on the Developer tab.**

This activates the Trust Center dialog box.

- 2. Click the Trusted Locations button.**

This opens the Trusted Locations menu (see Figure 1-6), which shows you all the directories that are considered trusted.

- 3. Click the Add New Location button.**

- 4. Click Browse to find and specify the directory that will be considered a trusted location.**

After you specify a trusted location, any Excel file opened from this location will have macros automatically enabled.

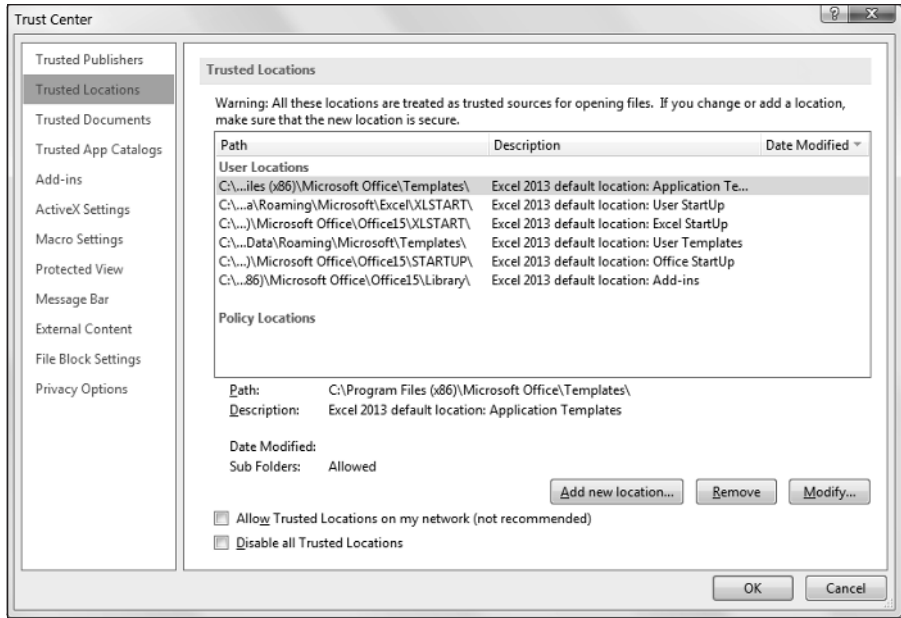


FIGURE 1-6:
The Trusted Locations menu allows you to add directories that are considered trusted.

Storing macros in your Personal Macro Workbook

Most user-created macros are designed for use in a specific workbook, but you may want to use some macros in all your work. You can store these general-purpose macros in the Personal Macro Workbook so that they're always available to you. The Personal Macro Workbook is loaded whenever you start Excel. This file, named `personal.xlsb`, doesn't exist until you record a macro using Personal Macro Workbook as the destination.

To record the macro in your Personal Macro Workbook, select the Personal Macro Workbook option in the Record Macro dialog box before you start recording. This option is in the Store Macro In drop-down list (refer to Figure 1-1).

If you store macros in the Personal Macro Workbook, you don't have to remember to open the Personal Macro Workbook when you load a workbook that uses macros. When you want to exit, Excel asks whether you want to save changes to the Personal Macro Workbook.



REMEMBER

The Personal Macro Workbook normally is in a hidden window to keep it out of the way.

Assigning a macro to a button and other form controls

When you create macros, you may want to have a clear and easy way to run each macro. A basic button can provide a simple but effective user interface.

As luck would have it, Excel offers a set of form controls designed specifically for creating user interfaces directly on spreadsheets. There are several different types of form controls, from buttons (the most commonly used control) to scrollbars.

The idea behind using a form control is simple: You place a form control on a spreadsheet and then assign a macro to it — that is, a macro you’ve already recorded. When a macro is assigned to the control, that macro is executed, or played, when the control is clicked.

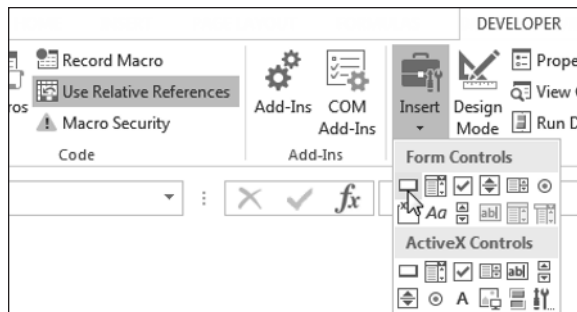
Take a moment to create a button for the AddTotalRelative macro you created earlier. Here’s how:

1. **Click the Insert button under the Developer tab. (See Figure 1-7.)**
2. **Select the Button Form Control from the drop-down list that appears.**
3. **Click the location where you want to place your button.**

When you drop the button control onto your spreadsheet, the Assign Macro dialog box, shown in Figure 1-8, activates and asks you to assign a macro to this button.

4. **Select the macro you want to assign to the button and then click OK.**

FIGURE 1-7:
You can find the form controls in the Developer tab.



At this point, you have a button that runs your macro when you click it! Keep in mind that all the controls in the Form Controls group (shown in Figure 1-7) work in the same way as the command button, in that you assign a macro to run when the control is selected.

FORM CONTROLS VERSUS ActiveX CONTROLS

Notice the form controls and ActiveX controls in Figure 1-7. Although they look similar, they're quite different. Form controls are designed specifically for use on a spreadsheet, and ActiveX controls are typically used on Excel user forms. As a general rule, you should always use form controls when working on a spreadsheet. Why? Form controls need less overhead, so they perform better, and configuring form controls is far easier than configuring their ActiveX counterparts.

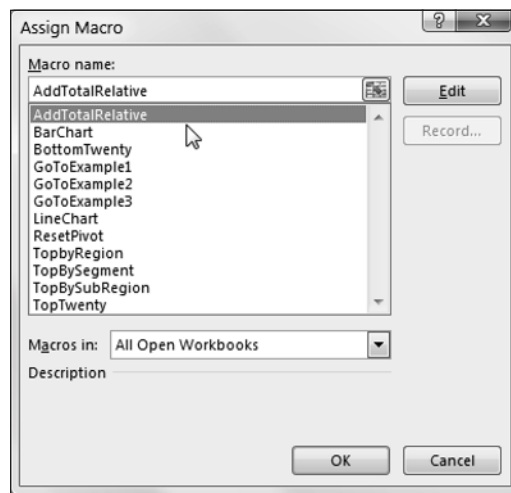


FIGURE 1-8:
Assign a macro
to the newly
added button.

Placing a macro on the Quick Access Toolbar

You can also assign a macro to a button in Excel's Quick Access Toolbar. The Quick Access Toolbar sits either above or below the Ribbon. You can add a custom button that runs your macro by following these steps:

- 1. Right-click your Quick Access Toolbar and select Customize Quick Access Toolbar.**
This opens the dialog box shown in Figure 1-9.
- 2. Click the Quick Access Toolbar button on the left of the Excel Options dialog box.**
- 3. Select Macros from the Choose Commands From drop-down list on the left.**

4. Select the macro you want to add and click the Add button.
5. Change the icon by clicking the Modify button.

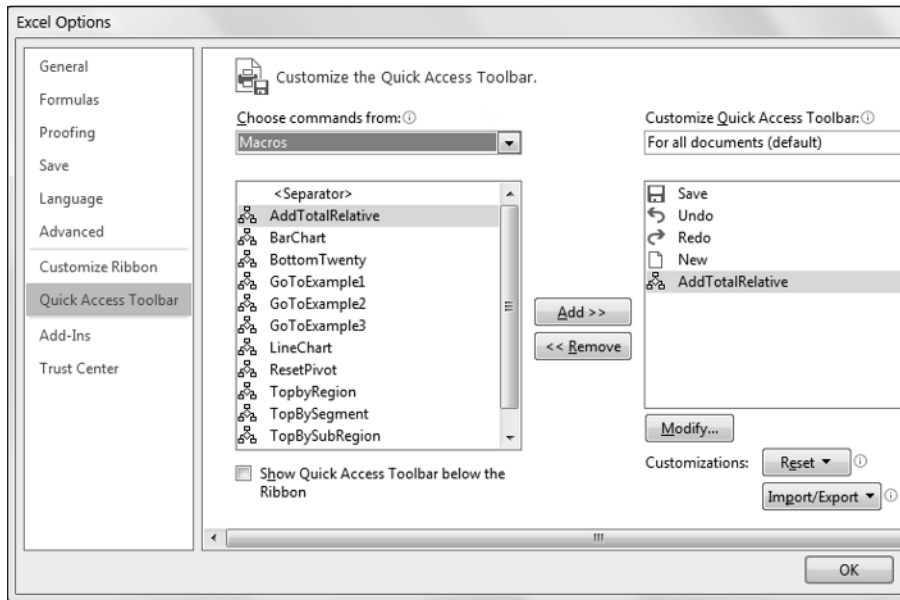


FIGURE 1-9:
Adding a macro
to the Quick
Access Toolbar.

Examples of Macros in Action

Covering the fundamentals of building and using macros is one thing. Coming up with good ways to incorporate them into your reporting processes is another. Take a moment to review a few examples of how macros automate simple reporting tasks.



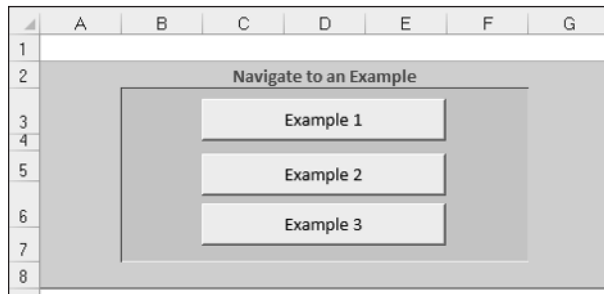
REMEMBER

Open the Chapter 1 Samples.xlsm file to follow along in the next section. To download the Chapter 1 Sample file, visit www.dummies.com/go/excelmacros.

Building navigation buttons

The most common use of macros is navigation. Workbooks that have many worksheets or tabs can be frustrating to navigate. To help your audience, you can create some sort of a switchboard, like the one shown in Figure 1-10. When a user clicks the Example 1 button, he's taken to the Example 1 sheet.

FIGURE 1-10: Use macros to build buttons that help users navigate your reports.



Creating a macro to navigate to a sheet is quite simple.

- 1. Start at the sheet that will become your switchboard or starting point.**
- 2. Start recording a macro.**
- 3. While recording, click the destination sheet (the sheet this macro will navigate to).**
- 4. After you click in the destination sheet, stop recording the macro.**
- 5. Assign the macro to a button.**



TIP

It's useful to know that Excel has a built-in hyperlink feature, allowing you to convert the contents of a cell into a hyperlink that links to another location. That location can be a separate Excel workbook, a website, or even another tab in the current workbook. Although using a hyperlink may be easier than setting up a macro, you can't apply a hyperlink to form controls (like buttons). Instead of a button, you'd use text to let users know where they'll go when they click the link.

Dynamically rearranging PivotTable data

Macros be used with any Excel object normally used in reporting. For instance, you can use a macro to give your audience a way to dynamically change a pivot table. In the example illustrated in Figure 1-11, macros allow a user to change the perspective of the chart simply by selecting any one of the buttons shown.

Figure 1-12 reveals that the chart is actually a pivot chart tied to a PivotTable. The recorded macros assigned to each button are doing nothing more than rearranging the PivotTable to slice the data using various pivot fields.

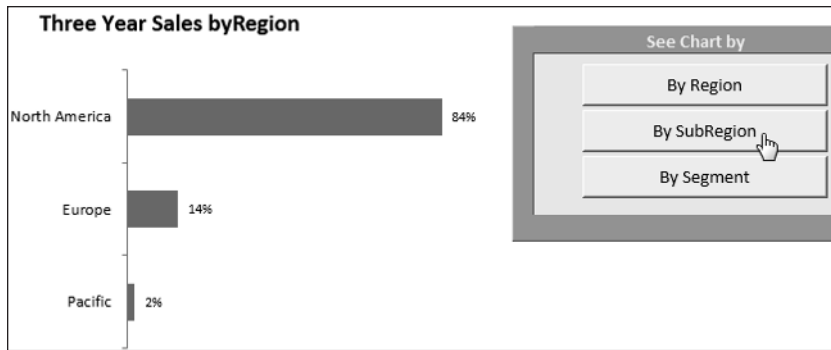


FIGURE 1-11: This report allows users to choose their perspective.

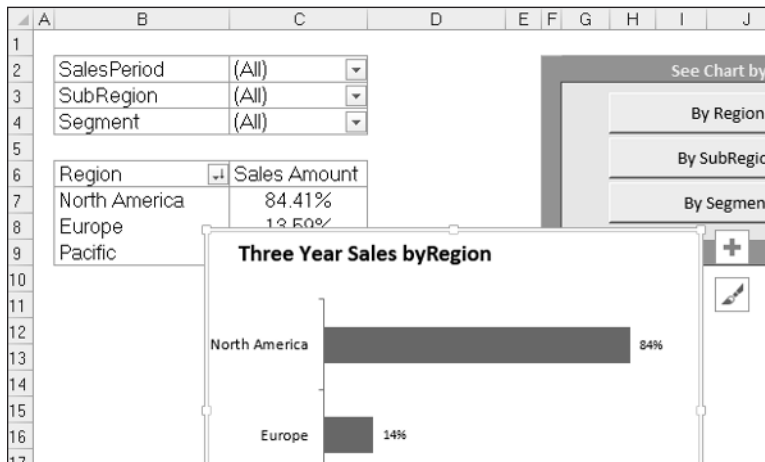


FIGURE 1-12: The macros behind these buttons rearrange the data fields in a PivotTable.

Here are the high-level steps needed to create this type of setup:

1. **Create your PivotTable and a pivot chart.**
2. **Start recording a macro.**
3. **While recording, move a pivot field from one area of the PivotTable to the other. When you're done, stop recording the macro.**
4. **Record another macro to move the data field back to its original position.**
5. **After both macros are set up, assign each one to a separate button.**

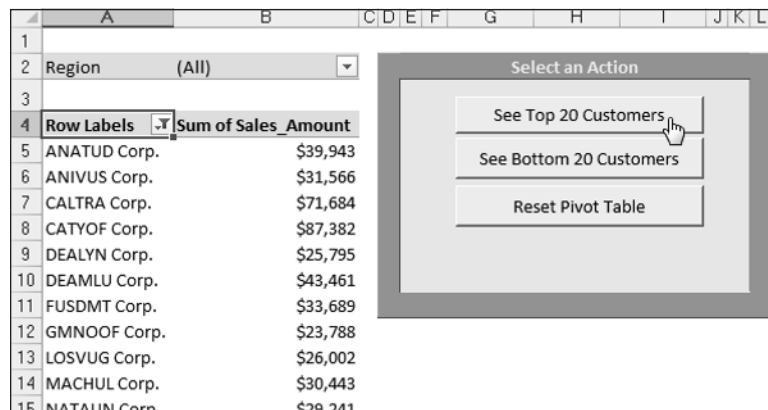
You can fire your new macros in turn to see your pivot field dynamically move back and forth.

Offering one-touch reporting options

The last two examples demonstrate that you can record any action that you find of value. That is, if you think users would appreciate a certain feature being automated for them, why not record a macro to do so?

In Figure 1-13, notice that you can filter the PivotTable for the top or bottom 20 customers. Because the steps to filter a PivotTable for the top and bottom 20 have been recorded, users can get the benefit of this functionality without knowing how to do it themselves. Also, recording a specific action allows you to manage risk a bit. That is to say, you'll know that your users will interact with your reports in a method that has been developed and tested by you.

FIGURE 1-13: Offering prerecorded views not only saves time and effort, but it also allows users that don't know how to use advanced features to benefit from them.



Row Labels	Sum of Sales_Amount
ANATUD Corp.	\$39,943
ANIVUS Corp.	\$31,566
CALTRA Corp.	\$71,684
CATYOF Corp.	\$87,382
DEALYN Corp.	\$25,795
DEAMLU Corp.	\$43,461
FUSDMT Corp.	\$33,689
GMNOOF Corp.	\$23,788
LOSVUG Corp.	\$26,002
MACHUL Corp.	\$30,443
NATAUN Corp.	\$28,241

This not only saves them time and effort, but it also allows users that don't know how to take these actions to benefit from them.

Figure 1-14 demonstrates how you can give your audience a quick and easy way to see the same data on different charts. Don't laugh too quickly at the uselessness of this example. It's not uncommon to be asked to see the same data different ways. Instead of taking up real estate, just record a macro that changes the Chart Type of the chart. Your clients can switch views to their heart's content.

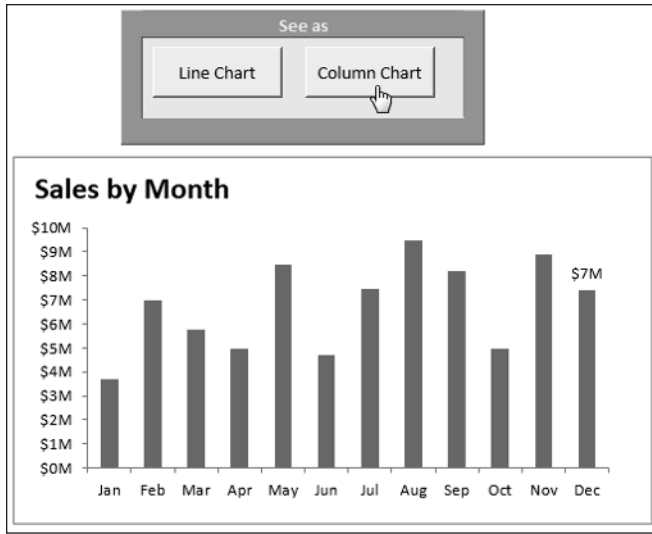


FIGURE 1-14: You can give your audience a choice in how they view data.