IN THIS CHAPTER

- Choosing macros
- Recording macros
- Understanding macro security
- Finding out where to store and how to run macros
- Exploring macro examples

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 smartphone. You first manually dial and save a number. Then when you want, you can redial those numbers with the touch of a button. With macro recording, 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 code (also known as Visual Basic for Applications or VBA). After a macro is recorded, you can play back those actions anytime you want.

In this chapter, you explore macros and find out how you can use macros to automate your recurring processes to simplify your life.

Choosing to 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. With a macro you could have those more redundant parts of your monthly process 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. Or you could calmly record a macro, ensure that everything is running correctly, and then forget it. The macro performs 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 ➪ Options.**
2. **In the Excel Options dialog box, click 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.

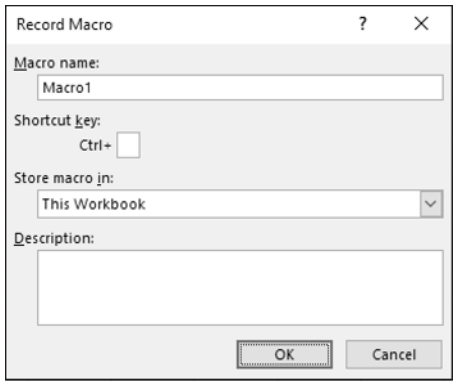


FIGURE 1-1: The Record Macro dialog box.

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

- **Macro Name:** 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.

  REMEMBER

  You have to follow a few rules when naming a macro. The first character must be a letter. Generally, special characters other than underscore aren't allowed. And the total number of characters can't be more than 255, although hopefully you don't get close to that limit.

- **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 if you use this field, 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). You can also choose New Workbook to tell Excel to create a new workbook to store the macro or Personal Macro Workbook, a special workbook used to store macros you want access to all the time. See "Storing and Running Macros" later in this chapter for more on the Personal Macro Workbook.

- **Description:** This field is optional, but it can come in handy if you have numerous macros in a workbook 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 cell B3 on your worksheet, 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. (See Chapter 2 to find out more about the Visual Basic Editor.) You can activate the VB Editor in one of three ways:

- Press Alt+F11.
- Choose Developer ➪ Code ➪ Visual Basic.
- Choose Developer ➪ Code ➪ Macros, select a macro, and click Edit.

In the VB Editor, the Project window displays a list of all open workbooks and add-ins. If the Project Explorer isn't visible, choose View ➪ Project Explorer. 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
'
```

```
    Range("B3").Select
    ActiveCell.FormulaR1C1 = "Dick Kusleika"
    Range("B4").Select
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 three VBA statements, the second of which is:

```
ActiveCell.FormulaR1C1 = "Dick Kusleika"
```

The first statement is selecting cell B3. The last statement is pressing Enter after you enter your name, which moves the active cell down one row. The middle statement, the one that does all the work, causes the name you typed while recording to be inserted into the active cell.

Editing the macro

After you record a macro, you can make changes to it. The macro you recorded in the previous section always inserts your name into cell B3. Edit the macro so that it enters your name in whatever cell you happen to be in when you run it. To do that, delete the first and third lines of the macro. The edited macro appears as follows:

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

This macro inserts text into the active cell because the first Select statement was removed. That same cell remains active because the second Select statement was removed.

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.

Comparing Absolute and Relative Macro Recording

Excel has two modes for recording — absolute reference and relative reference. These modes affect what statements Excel generates when you select cell while recording. In this section, I discuss the two modes and when to use them.

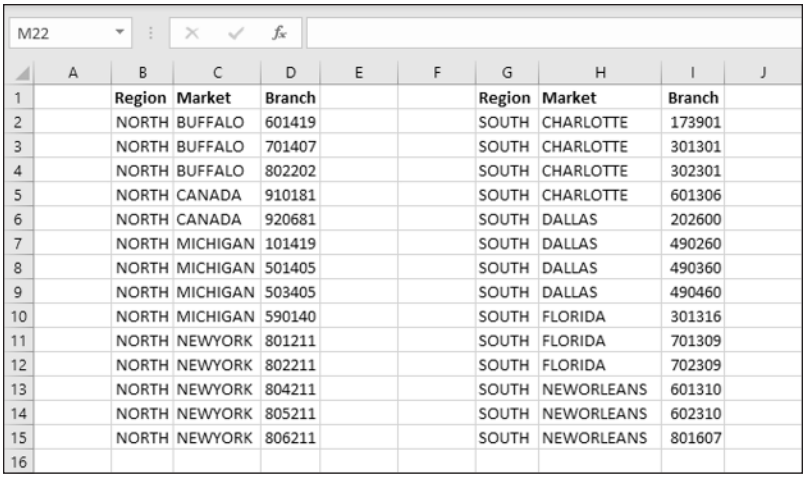
Recording macros with absolute references

In the example in the preceding section, you selected cell B3 while recording a macro and Excel dutifully recorded a statement that selects cell B3. This is an example of an absolute reference and it's the default mode when recording macros. 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. 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 the book's companion website.



	A	B	C	D	E	F	G	H	I	J
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										

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

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.**
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.

6. **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.

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

1. **Click Macros on 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 the macro using absolute references.

To understand what this means, examine the underlying code. To examine the code, click Macros on the Developer tab to open the Macro dialog box, as shown in Figure 1-4.

	A	B	C	D	E	F	G	H	I	J
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						
17										

FIGURE 1-3: Your post-totaled worksheet.

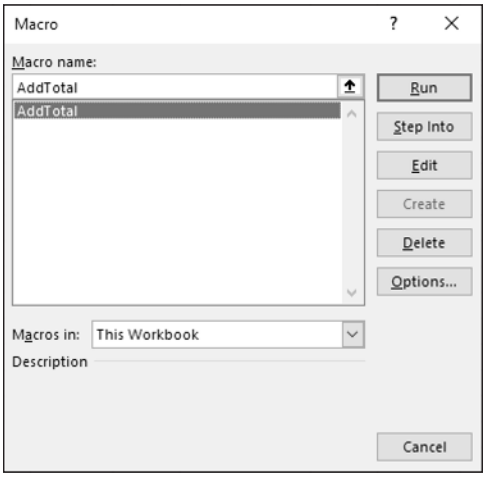


FIGURE 1-4: The Excel Macro dialog box.

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

  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:

1. **Click Use Relative References from the Developer tab, as shown in Figure 1-5.**
2. **Before recording, make sure cell A1 is selected.**
3. **Click Record Macro from the Developer tab.**
4. **Name the macro AddTotalRelative.**
5. **Choose This Workbook in the Macros In drop-down list.**
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.**

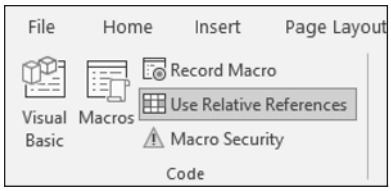


FIGURE 1-5: Recording a macro with relative references.

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

Click 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. Take a look at what the relevant parts of this VBA code really mean.

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). Excel doesn't select a cell with a specific address as it did when recording an absolute reference macro.

Between Offset and Select on the second line is Range("A1"). This is Excel recording that you only selected one cell — the first cell of the range that is offset from the active cell. It's a quirk of the Macro Recorder and isn't necessary when you

select only one cell. (The Macro Recorder records a lot of unnecessary code.) If you had selected, say A16:B17 instead of just A16, it would have recorded:

```
ActiveCell.Offset(15, 0).Range("A1:B2").Select
```

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

1. **Select cell A1.**
2. **Click Macros on the Developer tab.**
3. **Select the AddTotalRelative macro.**
4. **Click the Run button.**
5. **Select cell F1.**
6. **Click Macros on the Developer tab.**
7. **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.

Understanding Macro Security

At this point, you should feel comfortable recording your own Excel macros. In the wrong hands, macros can be destructive. Because macros are so powerful, some people have used them to create macro viruses that can be harmful to your computer. For that reason, Microsoft has built some security measures around macros. Here are some important security concepts you need to keep in mind when working with macros.

Macro-enabled file extensions

Excel's default file format, called Excel Workbook, has a .xlsx file extension. 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.

Trusted documents

Excel allows you to flag a document as trusted. 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, you see a yellow bar message under the Ribbon stating that macros (active content) have been disabled. If you have the Visual Basic Editor open, you'll get a dialog box instead of the yellow bar.

If you click Enable, the workbook 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 users of your macros. 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 workbook 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. **Click the Macro Security button on the Developer tab.**

   The Trust Center dialog box opens.

2. **Click the Trusted Locations option on the left.**

   This opens the Trusted Locations window (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.**

5. **Add an optional description, click OK to close the Microsoft Office Trusted Location dialog, and then click OK again to close the Trust Center dialog.**

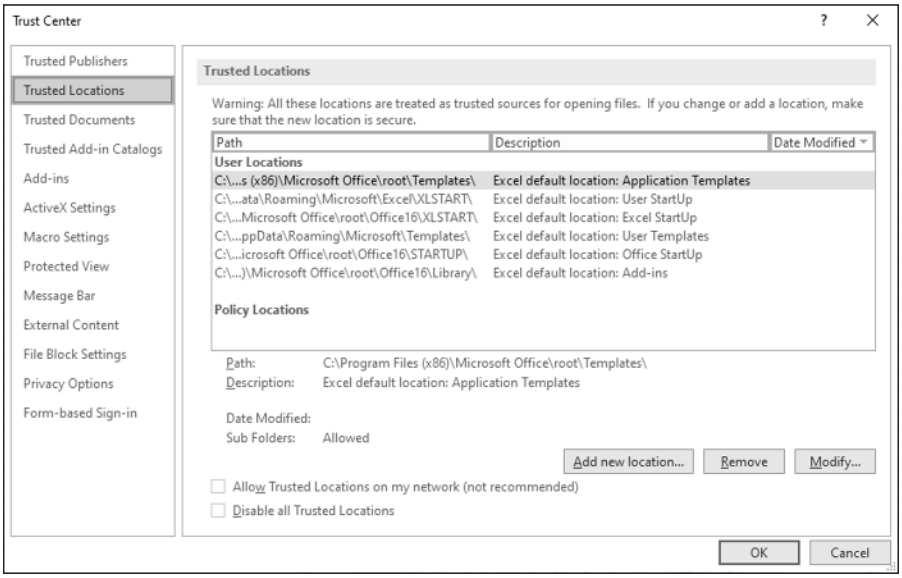


FIGURE 1-6: The Trusted Locations window allows you to add directories that are considered trusted.

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

Storing and Running Macros

Macros are stored in a workbook. Even though you work with macros in the Visual Basic Editor, the code is stored in the same file with all the worksheets, cells, shapes, or whatever else you have inside your workbook. This makes macros very

easy to distribute. You simply send someone the Excel file and the macros are included.

All macros need an event to start them running. Earlier in this chapter, I discussed assigning a shortcut key to a macro you record. Pressing the shortcut key is one type of event that can trigger a macro. In this section, I discuss the Personal Macro Workbook, a special workbook for storing macros, and a few common ways to trigger macros.

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 This file, named personal.xlsb, doesn't exist until you record a macro using Personal Macro Workbook as the destination. The Personal Macro Workbook is loaded whenever you start Excel.

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 worksheets. 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 worksheet 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 on 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 worksheet, the Assign Macro dialog box, shown in Figure 1-8, opens 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.**

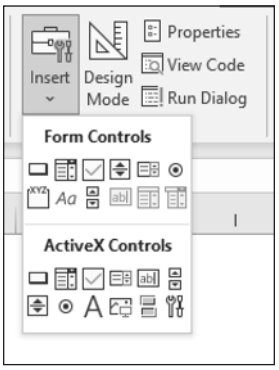


FIGURE 1-7: You can find the form controls on the Developer tab.

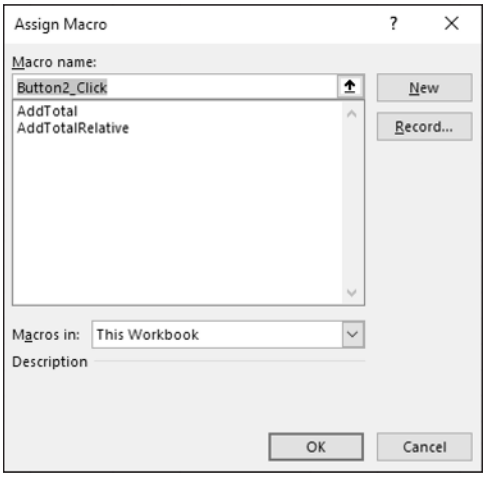


FIGURE 1-8: Assign a macro to the newly added button.

FORM CONTROLS VERSUS ACTIVEX CONTROLS

The Insert command on the Developer tab shows both form controls and ActiveX controls (refer to Figure 1-7). Although they look similar, they're quite different. Form controls are designed specifically for use on a worksheet, and ActiveX controls are typically used on Excel user forms (custom dialog boxes that are beyond the scope of this book). As a general rule, you should always use form controls when working on a worksheet unless there is a specific function that only ActiveX controls have. This is because form controls need less overhead, so they perform better, and configuring form controls is far easier than configuring their ActiveX counterparts.

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) can work in the same way as the command button, in that you assign a macro to run when the control is selected.

Placing a macro on the Quick Access Toolbar

You can also assign a macro to a button on the 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. **Select Macros from the Choose Commands From drop-down list on the left.**
3. **Select the macro you want to add and click the Add button.**
4. **(Optional) Click the Modify button to change the icon or display name, and then click OK to close the Modify Button dialog box.**
5. **Click OK to close the Excel Options dialog box.**

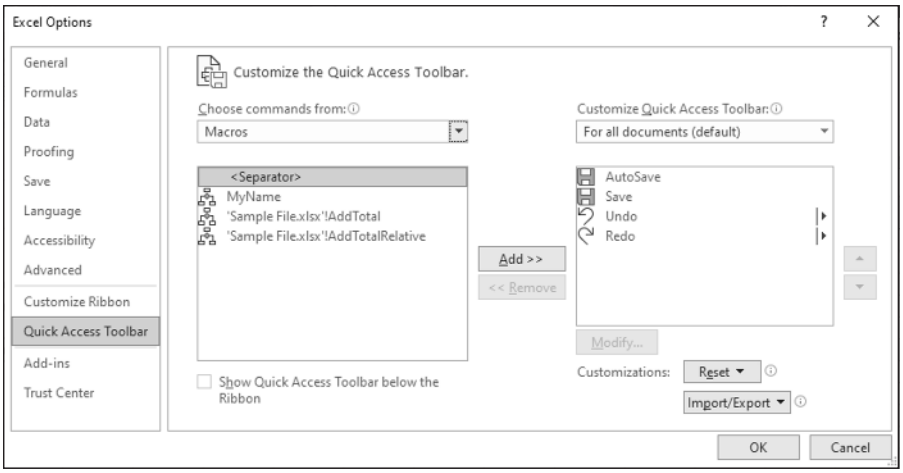


FIGURE 1-9: Adding a macro to the Quick Access Toolbar.

Exploring Macro Examples

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 Chapter 1 Samples.xlsm to follow along in the next section.

Building navigation buttons

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

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.**

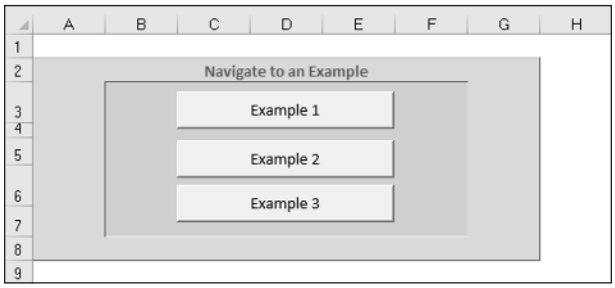


FIGURE 1-10: Use macros to build buttons that help users navigate your reports.

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 (such as 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 can be used with any Excel object normally used in reporting. For instance, you can use a macro to give your users a way to dynamically change aPivotTable. In the example shown in Figure 1-11, macros allow a user to change the perspective of the chart simply by clicking any one of the buttons.

Figure 1-12 reveals that the chart is actually a PivotChart 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.

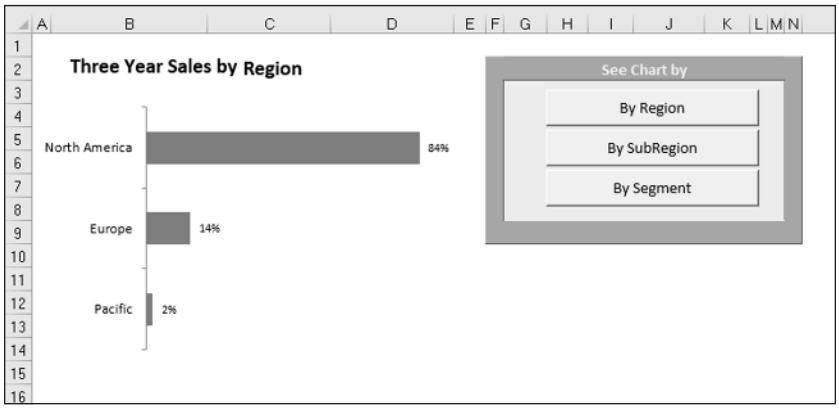


FIGURE 1-11: This report allows users to choose their perspective.

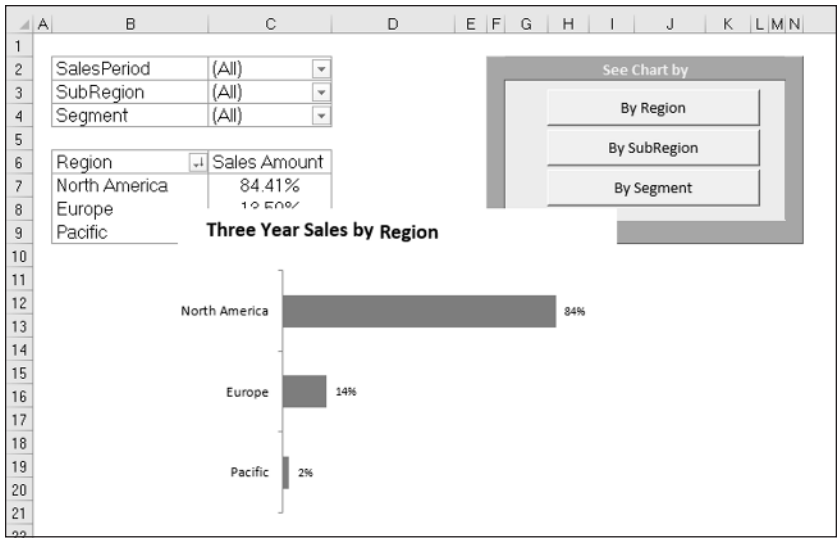


FIGURE 1-12: The macros behind these buttons rearrange the data fields in a PivotTable.

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

1. **Create your PivotTable and PivotChart.**
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 run 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, buttons are tied to macros that 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, your users can interact with your reports in a method that has been developed and tested by you.

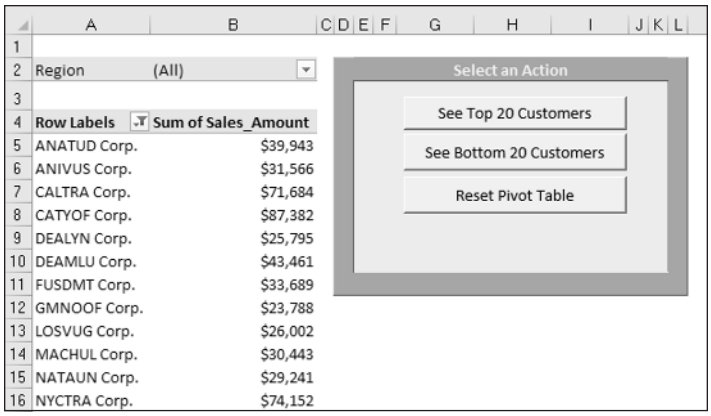


FIGURE 1-13: Offering prerecorded views not only saves time and effort, but it also allows users that don't know how to use advanced features to benefit from them.

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. It's not uncommon to be asked to see the same data different ways. Instead of taking up real estate by showing both charts, just record a macro that changes the Chart Type of the chart. Your clients can switch views to their heart's content.

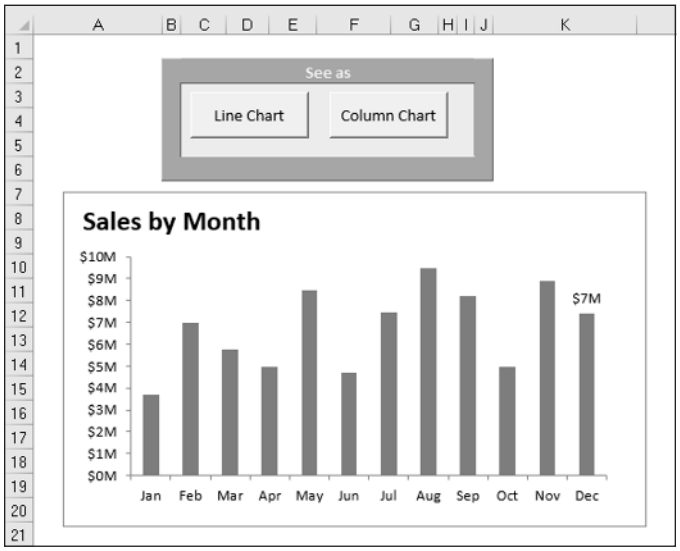


FIGURE 1-14: You can give your audience a choice in how they view data.