After my previous article on creating forms in Excel (Weiss, 2004) came out, I received some e-mail asking for more information on how to aggregate the data from the collected forms. Surely there’s an easy way to do that, isn’t there? There sure is, but it means delving into the big and often misunderstood topic of programming Excel. The topic itself is vast enough in scope that I have at least three thick books on my bookshelf that cover various aspects of it with varying specificity to Excel, and none purport to offer any kind of encyclopedic reference. Given the few short pages that I have here, I can’t do much more than introduce some of the ideas and illustrate their power. That said, you can do some amazing, time-saving things with only a little knowledge, and Excel makes it easy to get started.

It is important to carefully define the scope of this article. With some practice, you can use Microsoft’s Visual Basic for Applications (VBA) programming language to create amazingly complex applications that can even draw on other applications such as Word and have unique interfaces that in no way resemble Excel spreadsheets. That’s at the high end, where the power users play. We’ll keep things simple for the sake of this article and just focus on creating Excel macros simply and implementing macro code created by others. Let’s start by getting a better understanding of what macros do and why they’re useful.

What Are Macros, and Can They Bite?

In PC desktop software, macros are best known as a means for capturing a set of user actions for later use. A simple example of a macro is provided by my long lost Gateway “AnyKey” keyboard. As I recall, this keyboard enabled the user to record keystrokes and save the recording to a specific key combination. This was a wonderful help in speeding exceedingly dull tasks like cleaning up data for entry into statistical software. Rather than repeat the key combination <DOWN ARROW>, <LEFT ARROW>, <F2>, <HOME>, <CTRL>+<SHIFT>+<RIGHT ARROW>, <SHIFT>+<RIGHT ARROW>, <DELETE>, <ENTER> to delete a set of characters that began each cell in a column for each and every one of a thousand rows in a spreadsheet, I could record the keystrokes into a keyboard macro and just hit the
activating keystroke a thousand times. This latter approach was much less painful, error-prone, and mind-numbing than trying to repeatedly duplicate a long sequence of keystrokes.

The keyboard macros described above are effective, but unquestionably limited. They only applied to keystrokes on a keyboard and were stored in the keyboard’s memory, unaware of the software in which they were used. This latter fact usually came rocketing home when I would accidentally invoke the wrong keyboard macro and watch helplessly as the unwitting macro mutilated my spreadsheet. In contrast to keyboard macros, Excel macros represent recorded actions within Excel, whether by keyboard or mouse. Say for example that you record an Excel macro that invokes a dialog box (say, the *Format Cells* dialog) and changes its settings. The resulting macro depicts your actions as formatting cells, not just as a selection of key-presses and mouse clicks on buttons.

Macros can be stored directly with a spreadsheet, or in your personal macro workbook if you’d like to use them across a selection of spreadsheets. Unfortunately, this former ability to store macros in a spreadsheet that you can e-mail to others is attractive to virus writers, whose macros tend toward the destructive. To prevent users from accidentally unleashing viruses, most installations of Excel are set by default to block all macros stored with spreadsheets. You can back away from this appropriately paranoid approach by setting Excel to offer you the choice for files containing macros. This is done by selecting the **Tools** menu and choosing **Macro | Security… | Medium**. Each time you open a spreadsheet file with macros in it, Excel will then ask you if you want them to be enabled. If you store macros in your spreadsheets, be sure to advise others to whom you distribute the files as to their presence and the above means of enabling their use.

**Baby’s First Macro**

I described conditional formatting in my first article on Excel (Weiss, 2003). The idea is that you can define up to three sets of criteria by which Excel will automatically format a cell. For example, let’s say you’ve set up your budget in Excel, and you want to keep a careful eye on spending in certain categories. To help you keep track at a glance, you institute a simple conditional format: If you are over budget for a given category, you want the cell background to be colored red; if you are within your budget, the background should be green. Let’s assume further that you have a number of such categories (*hobby expenses; house repairs due to hobby projects gone terribly wrong: apologetic gifts to spouse, etc.*) and that you change which categories you choose to focus on each month. Rather than go through the effort of repeatedly invoking the *Conditional Formatting* dialog box and filling in the exact same information, it makes sense to create a macro to do the work.
Let’s assume that you have a cell for each category of expenditure that calculates the difference between your budget and actual outlays. If the value of that cell is zero or negative, then you are under budget. If it is positive, then you are over budget. We will go through the steps to set up conditional formatting for this cell and have Excel record our actions. Here are the steps to take.

1. **Pick a cell.** Start Excel and select a target cell.

2. **Launch the macro recorder.** Select the Tools menu and choose **Macro | Record New Macro…** Under **Macro name** in the Record Macro dialog box, type BudgetWatch. Note that macro names cannot have spaces. Under **Store macro in**, select **Personal Macro Workbook**. Click **OK** to exit the Record Macro dialog box. The dialog box will be replaced by a tiny, illegibly-titled Stop Recording toolbar with two buttons on it, the first of which has a blue square like a Stop button on a CD or DVD player. The macro recorder is now recording your every movement within Excel.

3. **Perform your conditional formatting magic.** Select the Format menu and choose **Conditional Formatting…** to bring up the Conditional Formatting dialog box. The box is designed to let you define conditions in a sentence-like manner. Click on the second drop-down box from the left, set by default to **between**. Set it to **greater than**. Enter 0 in the final box on the right. The three boxes, left to right, should read “Cell Value Is” “greater than” “0”. Click the **Format…** button to begin defining the cell formatting. Choose the Patterns tab and click on a red square. Click **OK**. Next, click the Add>> button to add a second condition where we will indicate the “within-budget” status. Follow the same process as just outlined and set three boxes to “Cell Value Is” “less than or equal to” ”0”, and the cell background pattern to green. Click **OK** to close the Conditional Formatting dialog box and click the Stop button on the Stop Recording toolbar. Your macro is now complete and ready to use.

4. **Try the macro.** First, let’s make sure the macro works as planned. Enter positive and negative values into your conditionally formatted cell and observe the changes in color. Note that you must hit Enter or exit the cell for the conditional formatting to operate following each change. Now go to a different cell and invoke the macro as follows: Select the Format menu and choose **Macro | Macros…** to launch the Macro dialog box. Select the macro labeled **Personal.xls!BudgetWatch** and click on **Run** to run it. Try entering positive and negative values into the new cell. Wasn’t that easy and fun?

**A note on the other button.** The Stop Recording toolbar has two buttons. The first, as we have seen, stops the macro recording process. The second is an interesting critter. By default, Excel records the selection of cells in absolute terms—if you click the down arrow on your keyboard and the cursor lands on cell B6, Excel records this as something like “Select cell B6.” If you want your macro to record relative movements in your macros (e.g., from whatever cell you are on, go down one cell), select that second button in the
Stop Recording toolbar, called the Relative Reference button prior to any related cursor movement. Once activated, the Relative Reference button remains activated until you click it again.

Running Macros

Now that you know how to create a simple macro—albeit one that can save you lots of time—you should know that there are many ways to invoke it. The first is through the Macro dialog box, as outlined above. Following are some faster ways to invoke macros.

Assign a shortcut key. Launch the Macro dialog box as outlined above. Select the macro for which you wish to define a shortcut key and click on the Options… button. In the Shortcut key box, enter a letter. You can then invoke the macro by holding down the <CTRL> key and typing that letter. A couple of notes are in order here. First, your choice of letter here will override any standard key combinations in place. For example, if you regularly rely on <CTRL>+<C> for copying a selection and you then define that key combination for a macro, you may be surprised if you accidentally try to use it for copying. However, if you change the shortcut key to something else, the default key combination will return. Second, you can use the <SHIFT> key for an additional shortcut associated with capital letters. Therefore, <CTRL>+<W> and <CTRL>+<SHIFT>+<W> can refer to two different macros.

Use a form control. Each of the form controls on the Forms toolbar (see Weiss, 2004) offer the option to run a macro when they are clicked. It is probably most useful to assign macros to buttons, which have no other use and were consequently not considered in Weiss (2004). However, the other controls can also accept macros. Right click on a form control and choose Assign Macro… from the context-sensitive menu. The Assign Macro dialog box will then appear. Select the desired macro and click OK. You can also create or record a new macro from this dialog box.

Assign a macro to a toolbar button or a menu item. This is a little complex. Select the Tools menu and choose Customize… Click on the Commands tab and select Macros near the bottom of the Categories listbox. For a toolbar button, click and drag the happy face Custom Button item from the Commands listbox to a toolbar. Click the Modify Selection button and choose Assign Macro… at the bottom. The Assign Macro dialog box will then appear. Select the desired macro and click OK. A number of the other options available from the Modify Selection button allow you to change the appearance of the button to text only or a combination of image and text and to choose a different button image. Creating a menu item is much the same, with the exception that you will drag the Custom Menu Item entry from the Commands listbox to a menu, and then drop it into position on the menu. As before, use the Modify Selection button to assign a macro and further define the menu item.
Learn the Code

Generating macros through the macro recorder is a very useful bit of functionality. Anything that can cut repetitive tasks—especially complex, repetitive tasks—down to size is welcome in my book. The fact of the matter is that even skilled Excel programmers use the macro recorder to get a start on their macros. We are all taught to use the right tool in the right situation, and the macro recorder is a very handy tool for situations in which you want to replicate specific actions within the Excel interface.

The macro recorder’s strength is that it is simple—it allows the user to create macros without having to learn VBA, Excel’s scripting language. VBA is a very powerful language; as discussed above, it can be used to create complex solutions that tie together elements of disparate software applications, as well as viruses that turn your computer into nothing more than a very expensive paperweight. With power naturally comes complexity—VBA is a complete programming language and must be learned as such to be used flexibly. Though the macro recorder hides VBA’s complexity from you, you will likely begin to find situations in which you need the power of a programmed solution.

The scope of this article does not permit any real treatment of VBA. Entire books are devoted to the topic, and it is one area in which I believe that you need more than one book to have all of the resources you need. Though I will not attempt to teach VBA here, there is still some merit to a brief discussion of the coding process. Many magazines, Web sites, and fellow users will share macros in the form of VBA code. If nothing else, it is helpful to know how to enter those macros so you can use them. Let’s take a look at how to do this.

To work with VBA code, you use the Visual Basic Editor, which may be launched via either of two methods. The first is to tell Excel that you want to create or edit a macro in VBA. Launch the Macro dialog box by choosing the Tools menu and selecting Macro | Macros… To create a new macro in VBA (not through the macro recorder), type its name into the box under Macro name and click the Create button. Remember, macros cannot have spaces or other nonalphanumeric characters in their names and must begin with an alphabetical character. When the Visual Basic Editor launches, you will see a new window that looks like Figure 1. To edit an existing macro, select it by name and click on the Edit button. The other way to launch the Visual Basic Editor is even more straightforward. Select the Tools menu and choose Macro | Visual Basic Editor. The difference between the two means of launching the Editor is that the former method is used for macros stored with a spreadsheet. The latter must be used for macros used across spreadsheets, which are stored in a file called Personal.xls.
Let’s try a simple example. Above, we created a macro to conditionally format cells for a budget spreadsheet. We noted that it was important to be able to change which cells were highlighted each month. The macro we created adds conditional formatting to cells we want to focus on that month. We therefore need to create another macro to eliminate the conditional formatting we applied the previous month.

As noted above, macros may be stored within a spreadsheet, for use with that spreadsheet only, or they may be stored in a separate location and used across spreadsheets. We wish to use the latter approach here. Launch VBA by selecting the Tools menu and choosing Macro | Visual Basic Editor. In the Project panel at left, you should see two projects called VBAProject. Select VBAProject (PERSONAL.XLS) and click on Modules, then double-click on Module1. You should see the code for your original Budget-Watch macro.

Select the Insert menu and choose Procedure. Type DeleteBudgetWatch into the Name box and click the OK button. The Visual Basic Editor will create the framework, which consists of the lines Sub DeleteBudgetWatch() and End Sub. In VBA terms, we are creating a “subroutine,” which is abbreviated as Sub in code. The VBA code for DeleteBudgetWatch consists of a single line, which you will type between the two framework lines. The line of code to type is:

Selection.FormatConditions.Delete

If you do not use capital letters within the code, Visual Basic will automatically reformat it. Note as well that it is the typical practice to indent code
within the Sub... End Sub construct. My copy of the macro therefore looks like the following:

```vba
Sub DeleteBudgetWatch()
    Selection.FormatConditions.Delete
End Sub
```

Select the File menu and choose Close and Return to Microsoft Excel. Try out the new macro and see how well it works!

**Compiling Excel Form Data**

As noted in the beginning of this article, I received some e-mail asking for information on aggregating form data. This is a fairly easy application of VBA. Let’s assume that the target data are stored in a single row of the spreadsheet with the form and that you wish to copy the data over to the first free row of a spreadsheet compiling all responses across respondents. For each new form spreadsheet received, we will open it, run our macro, and then wait while the macro does all the work and reports back its success.

Here’s the macro I created. I have numbered the lines for easier reference in my explanation below. **Do not number the lines in your code!** Since you will want to use the macro across spreadsheets, enter it into Personal.xls as described in the example immediately above.

```vba
1 Sub CopyFormData()
2   Range(”Responses!2:2”).Copy
3   Workbooks.Open Filename:=”C:\FormData.xls”
4   Range(”A1”).Select
5   Selection.End(xlDown).Select
6   ActiveCell.Offset(1, 0).Range(”A1”).Select
7   ActiveSheet.Paste
8   Application.CutCopyMode = False
9   ActiveWorkbook.Save
10  ActiveWindow.Close
11  MsgBox (”Record successfully copied”)
12  End Sub
```

Here’s what the code means:

**Line 1:** Define a macro called CopyFormData.

**Line 2:** On the Responses sheet, where I keep the responses from the form, I have my variable names in Row 1 and the actual responses in Row 2. Line 2 tells Excel to select and copy Row 2.

**Line 3:** Open the file in which I store responses across participants.

**Line 4–6:** Navigate to the first open row by starting at cell A1 (Line 4), moving down to the last row of data (Line 5), and then moving down one additional cell (Line 6).
I hope this article provided you with a good way to get up to speed on Excel macros. I am always interested in feedback and suggestions for future articles, and can be reached at jason.weiss@ddiworld.com. I regret, of course, that I cannot offer specific help on your Excel applications. You will find, however, that there are many resources online that are ready to help you. See my Web site for information on finding these resources: http://www.jasonweiss.net. I also use the Web site to provide additional examples and materials on Excel and to comment on the rare bugs that may have appeared in my articles.

**Suggested Resources**

Though I’m not a big fan of buying books about software, VBA is a topic that needs good guidance and reference material. If you are starting from scratch, I’d suggest a book on programming in VBA, a reference book on VBA code, and a third book on programming Excel. The last will have special detail on coding elements that are exclusive to Excel. The following are some books that you might find useful.


There are a number of Web sites that offer macros and guidance on programming in Excel. Here are several I find particularly useful:

http://www.exceltip.com/

A very useful site with lots of macros and a vibrant user community.

http://ca.geocities.com/b_davidso/Web_Page_Files/excel_vba.html

Barrie Davidson’s Excel VBA page. Lots of interesting and diverse applications.


Microsoft’s knowledge base—always handy for any Excel question, whether VBA-related or not.

**References**
