Excerpted from VBA and Macros for Microsoft Excel, by Bill Jelen and Tracy Syrstad

Case Study for Using VBA with Excel

Let's say you work in an accounting department. Each day you receive a text file from the company system showing all the invoices produced the prior day. This text file has commas separating each field. The columns in the file are InvoiceDate, InvoiceNumber, SalesRepNumber, CustomerNumber, ProductRevenue, ServiceRevenue, and ProductCost.

As you arrive at work each morning, you manually import this file into Excel. You add a total row to the data, bold the headings, and then print the report for distribution to a few managers.

Preparing to Record the Macro

This is a task that is perfect for a macro. Before you record any macro, you should think about the steps that you will use before you begin. In our case, these steps are as follows:

1. From the menu, select File, Open.
2. Navigate to the proper folder.
3. Choose All Files(*.*) from the Files of Type: dropdown list.
4. Select Invoice.txt.
5. Click Open.
6. In the Text Import Wizard—Step 1 of 3, select Delimited from the Original data type section.
7. Click Next.
8. In the Text Import Wizard—Step 2 of 3, uncheck the Tab key and check Comma in the Delimiters section.
9. Click Next.
10. In the Text Import Wizard—Step 3 of 3, select General in the Column data format section and change it to Date: MDY.
11. Click Finish to import the file.
12. Press the End key followed by the down arrow to move to the last row of data.
13. Press the down arrow one more time to move to the total row.
14. Type the word Total.
15. Press the right arrow key four times to move to column E of the total row.
16. Click the Autosum button and press Ctrl+Enter to add a total to the Product Revenue column

while remaining in that cell.

17. Grab the Autofill handle and drag from Column E over to Column G to copy the total formula over to Columns F and G.

18. Highlight Row 1 and click the Bold icon to set the headings in bold.

19. Highlight the Total row and click the Bold icon to set the totals in bold.

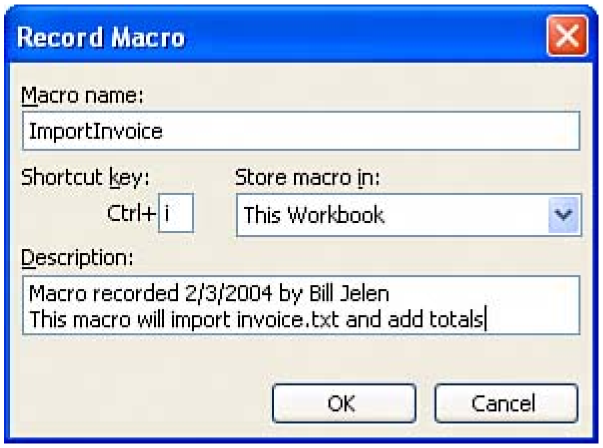
20. Press Ctrl+A to select all cells.

21. From the menu, select Format, Column, Autofit Selection.

After you've rehearsed these steps in your head, you are ready to record your first macro. Open a blank workbook and save it with a name like MacroToImportInvoices.xls. Click the Record button or select Tools, Macro, Record New Macro.

In the Record Macro dialog, the default macro name is Macro1. Change this to something descriptive like ImportInvoice. Make sure that the macros will be stored in This Workbook. You might want an easy way to run this macro later, so enter the letter i in the Shortcut Key field. In the Description field, you will, by default, have your name and date. Add a little descriptive text to tell what the macro is doing (see Figure 1.10). When you are ready, click OK.

Figure 1.10. Before recording your macro, complete the Record Macro dialog box.



Macro name: ImportInvoice

Shortcut key: Ctrl+ i

Store macro in: This Workbook

Description:
Macro recorded 2/3/2004 by Bill Jelen
This macro will import invoice.txt and add totals

Recording the Macro

Don't be nervous, but the Macro Recorder is now recording your every move. You want to try to perform your steps in exact order without extraneous actions. If you accidentally move to column F and then back to E to enter the first total, the recorded macro blindly makes that same mistake day after day after day. Recorded macros move fast, but this is nothing like having to watch your same mistakes played out by the macro recorder again and again.

CAUTION

Resist the temptation to hide the Stop Recording toolbar. If the floating toolbar gets in the way, grab the blue title bar to move it to an out-of-the-way place. (The action of moving the Stop Recording Toolbar is not recorded.) If you do happen to hide the toolbar with the "X," then you will have to use the hard-to-find Tools, Macro, Stop Recording option on the menu.

Carefully, execute all the actions necessary to produce the report. After you have performed the final step, click the Stop button on the Stop Recording toolbar. The toolbar disappears.

CAUTION

Again, resist the temptation to close the toolbar. This does not stop the recording. If you do accidentally close the toolbar, you can stop recording by choosing Tools, Macro, Stop Recording from the main menu.

It is time to take a look at your code. Switch to the Visual Basic Editor by choosing Tools, Macro, Visual Basic Editor or pressing Alt+F11.

Examining Code in the Programming Window

Let's take a look at the code you just recorded from the case study; don't worry if it doesn't make sense yet.

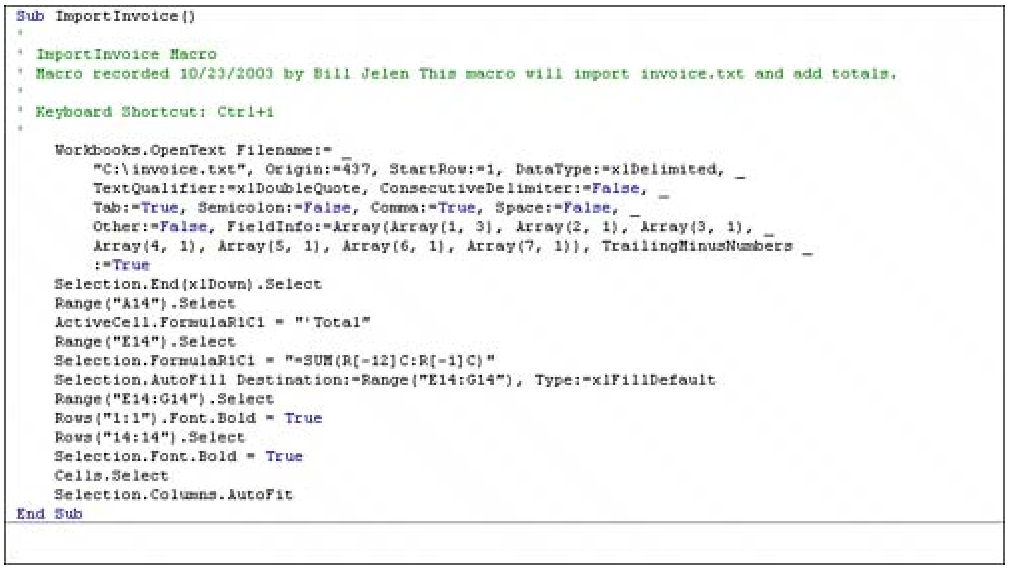
To open the Visual Basic Editor press Alt+F11. In our VBA Project (MacroToImportInvoices.xls), find the component Module1, right-click on it, and select View Code. Notice that some lines start with an apostrophe—these are comments and are ignored by the program. The macro recorder starts your macros with a few comments, using the description that you entered in the Record Macro dialog. The comment for the Keyboard Shortcut is there to remind you of the shortcut.

CAUTION

The comment does not assign the shortcut. If you change the comment to be Ctrl+j, it does not change the shortcut. This has to be done in the Macro dialog box from Excel.

Recorded macro code is usually pretty neat (see Figure 1.11). Each non-comment line of code is indented four characters. If a line is longer than 100 characters, the recorder breaks it into multiple lines and indents the lines an additional four characters. To continue a line of code, you type a space and an underscore at the end of the line. Note that the physical limitations of this book do not allow 100 characters on a single line. I will break the lines at 80 characters so that they fit on this page. Your recorded macro might look slightly different than the ones that appear here.

Figure 1.11. The recorded macro is neat looking and nicely indented.



```
Sub ImportInvoice()
'
' ImportInvoice Macro
' Macro recorded 10/23/2003 by Bill Jelen This macro will import invoice.txt and add totals.
'
' Keyboard Shortcut: Ctrl+i
'
    Workbooks.OpenText Filename:= _
        "C:\invoice.txt", Origin:=437, StartRow:=1, DataType:=xlDelimited, _
        TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False, _
        Tab:=True, Semicolon:=False, Comma:=True, Space:=False, _
        Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), Array(3, 1), _
        Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), TrailingMinusNumbers _
        :=True
    Selection.End(xlDown).Select
    Range("A14").Select
    ActiveCell.FormulaR1C1 = "'Total"
    Range("E14").Select
    Selection.FormulaR1C1 = "=SUM(R[-12]C:R[-1]C)"
    Selection.AutoFill Destination:=Range("E14:G14"), Type:=xlFillDefault
    Range("E14:G14").Select
    Rows("1:1").Font.Bold = True
    Rows("14:14").Select
    Selection.Font.Bold = True
    Cells.Select
    Selection.Columns.AutoFit
End Sub
```

Consider that the following seven lines of recorded code is actually only one line of code that has been broken down into seven lines for readability:

```
Workbooks.OpenText Filename:= _
   "C:\invoice.txt", Origin:=437, StartRow:=1, DataType:=xlDelimited, _
   TextQualifier:=xlDoubleQuote, ConsecutiveDelimiter:=False,  _
   Tab:=True, Semicolon:=False, Comma:=True, Space:=False,  _
   Other:=False, FieldInfo:=Array(Array(1, 3), Array(2, 1), Array(3, 1), _
   Array(4, 1), Array(5, 1), Array(6, 1), Array(7, 1)), _
  TrailingMinusNumbers:=True
```

Counting the above as one line, the macro recorder was able to record our 21-step process in 14 lines of code, which is pretty impressive.

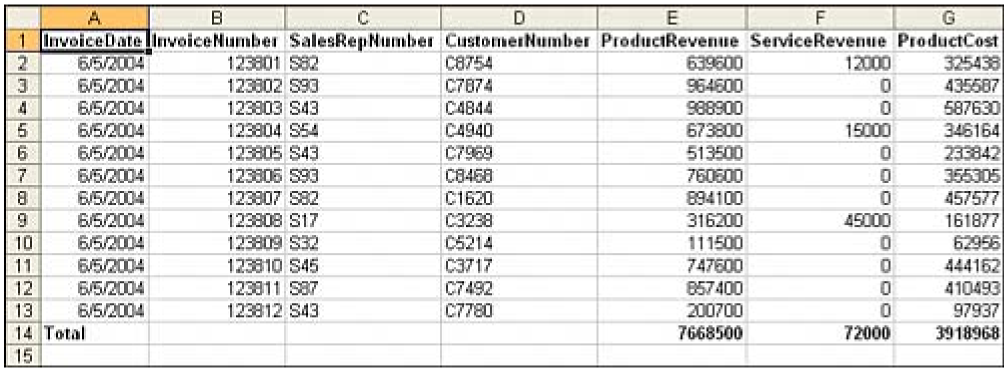
TIP

Each action that you perform in the Excel user interface may equate to one or more lines of recorded code. Some actions might generate a dozen lines of code.

It is always a good idea to test out your macro. Return to the regular Excel interface, using Alt+F11. Close Invoice.txt without saving any changes. You still have MacroToImportInvoices.xls open.

Press Ctrl+i to run the recorded macro. It works beautifully! The data is imported, totals are added, bold formatting is applied, and the columns are made wider. This seems like a perfect solution (see Figure 1.12).

Figure 1.12. The macro formats the data in the sheet beautifully.



	A	B	C	D	E	F	G
1	**InvoiceDate**	**InvoiceNumber**	**SalesRepNumber**	**CustomerNumber**	**ProductRevenue**	**ServiceRevenue**	**ProductCost**
2	6/5/2004	123801	S82	C8754	639600	12000	325438
3	6/5/2004	123802	S93	C7874	964600	0	435587
4	6/5/2004	123803	S43	C4844	988900	0	587630
5	6/5/2004	123804	S54	C4940	673800	15000	346164
6	6/5/2004	123805	S43	C7969	513500	0	233842
7	6/5/2004	123806	S93	C8468	760600	0	355305
8	6/5/2004	123807	S82	C1620	894100	0	457577
9	6/5/2004	123808	S17	C3238	316200	45000	161877
10	6/5/2004	123809	S32	C5214	111500	0	62956
11	6/5/2004	123810	S45	C3717	747600	0	444162
12	6/5/2004	123811	S87	C7492	857400	0	410493
13	6/5/2004	123812	S43	C7780	200700	0	97937
14	**Total**				**7668500**	**72000**	**3918968**
15							