1

Introduction to Excel

Excel made its debut on the Macintosh in 1985 and has never lost its position as the most popular spreadsheet application in the Mac environment. In 1987, Excel was ported to the PC, running under Windows. It took many years for Excel to overtake Lotus 1-2-3, which was one of the most successful software systems in the history of computing at that time.

There were a number of spreadsheet applications that enjoyed success prior to the release of the IBM PC in 1981. Among these were VisiCalc and Multiplan. VisiCalc started it all, but fell by the wayside early on. Multiplan was Microsoft's predecessor to Excel, using the R1C1 cell addressing which is still available as an option in Excel. But it was 1-2-3 that shot to stardom very soon after its release in 1982 and came to dominate the PC spreadsheet market.

Early Spreadsheet Macros

1-2-3 was the first spreadsheet application to offer spreadsheet, charting, and database capabilities in one package. However, the main reason for its run-away success was its macro capability. Legend has it that the 1-2-3 developers set up macros as a debugging and testing mechanism for the product. It is said that they only realized the potential of macros at the last minute, and included them into the final release pretty much as an afterthought.

Whatever their origins, macros gave non-programmers a simple way to become programmers and automate their spreadsheets. They grabbed the opportunity and ran. At last they had a measure of independence from the computer department.

The original 1-2-3 macros performed a task by executing the same keystrokes that a user would use to carry out the same task. It was, therefore, very simple to create a macro as there was virtually nothing new to learn to progress from normal spreadsheet manipulation to programmed manipulation. All you had to do was remember what keys to press and write them down. The only concessions to traditional programming were eight extra commands, the `/x` commands. The `/x` commands provided some primitive decision making and branching capabilities, a way to get input from a user, and a way to construct menus.

One major problem with 1-2-3 macros was their vulnerability. The multi-sheet workbook had not yet been invented and macros had to be written directly into the cells of the spreadsheet they supported, along with input data and calculations. Macros were at the mercy of the user. For example, they could be inadvertently disrupted when a user inserted or deleted rows or columns. Macros were also at the mercy of the programmer. A badly designed macro could destroy itself quite easily while trying to edit spreadsheet data.

Despite the problems, users reveled in their newfound programming ability and millions of lines of code were written in this cryptic language, using arcane techniques to get around its many limitations. The world came to rely on code that was often badly designed, nearly always poorly documented, and at all times highly vulnerable, often supporting enterprise-critical control systems.

The XLM Macro Language

The original Excel macro language required you to write your macros in a macro sheet that was saved in a file with an `.xlm` extension. In this way, macros were kept separate from the worksheet, which was saved in a file with an `.xls` extension. These macros are now often referred to as XLM macros, or Excel 4 macros, to distinguish them from the VBA macro language introduced in Excel Version 5.

The XLM macro language consisted of function calls, arranged in columns in the macro sheet. There were many hundreds of functions necessary to provide all the features of Excel and allow programmatic control. The XLM language was far more sophisticated and powerful than the 1-2-3 macro language, even allowing for the enhancements made in 1-2-3 Releases 2 and 3. However, the code produced was not much more intelligible.

The sophistication of Excel's macro language was a two edged sword. It appealed to those with high programming aptitude, who could tap the language's power, but was a barrier to most users. There was no simple relationship between the way you would manually operate Excel and the way you programmed it. There was a very steep learning curve involved in mastering the XLM language.

Another barrier to Excel's acceptance on the PC was that it required Windows. The early versions of Windows were restricted by limited access to memory, and Windows required much more horsepower to operate than DOS. The Graphical User Interface was appealing, but the tradeoffs in hardware cost and operating speed were perceived as problems.

Lotus made the mistake of assuming that Windows was a flash in the pan, soon to be replaced by OS/2, and did not bother to plan a Windows version of 1-2-3. Lotus put its energy into 1-2-3/G, a very nice GUI version of 1-2-3 that only operated under OS/2. This one horse bet was to prove the undoing of 1-2-3.

By the time it became clear that Windows was here to stay, Lotus was in real trouble as it watched users flocking to Excel. The first attempt at a Windows version of 1-2-3, released in 1991, was really 1-2-3 Release 3 for DOS in a thin GUI shell. Succeeding releases have closed the gap between 1-2-3 and Excel, but have been too late to stop the almost universal adoption of Microsoft Office by the market.

Excel 5

Microsoft took a brave decision to unify the programming code behind its Office applications by introducing **VBA (Visual Basic for Applications)** as the common macro language in Office. Excel 5, released in 1993, was the first application to include VBA. It has been gradually introduced into the other Office applications in subsequent versions of Office. Excel, Word, Access, PowerPoint, and Outlook all use VBA as their macro language in Office XP.

Since the release of Excel 5, Excel has supported both the XLM and the VBA macro languages, and the support for XLM should continue into the foreseeable future, but will decrease in significance as users switch to VBA.

VBA is an object-oriented programming language that is identical to the Visual Basic programming language in the way it is structured and in the way it handles objects. If you learn to use VBA in Excel, you know how to use it in the other Office applications.

The Office applications differ in the objects they expose to VBA. To program an application, you need to be familiar with its **object model**. The object model is a hierarchy of all the objects that you find in the application. For example, part of the Excel Object Model tells us that there is an `Application` object that contains a `Workbook` object that contains a `Worksheet` object that contains a `Range` object.

> *VBA is somewhat easier to learn than the XLM macro language, is more powerful, is generally more efficient, and allows us to write well-structured code. We can also write badly structured code, but by following a few principles, we should be able to produce code that is readily understood by others and is reasonably easy to maintain.*

In Excel 5, VBA code was written in modules, which were sheets in a workbook. Worksheets, chart sheets, and dialog sheets were other types of sheets that could be contained in an Excel 5 workbook.

> **A module is really just a word-processing document with some special characteristics that help you write and test code.**

Excel 97

In Excel 97, Microsoft introduced some dramatic changes in the VBA interface and some changes in the Excel Object Model. From Excel 97 onwards, modules are not visible in the Excel application window and modules are no longer objects contained by the `Workbook` object. Modules are contained in the VBA project associated with the workbook and can only be viewed and edited in the Visual Basic Editor (VBE) window.

In addition to the standard modules, class modules were introduced, which allow you to create your own objects and access application events. Commandbars were introduced to replace menus and toolbars, and UserForms replaced dialog sheets. Like modules, UserForms can only be edited in the VBE window. As usual, the replaced objects are still supported in Excel, but are considered to be hidden objects and are not documented in the Help screens.

In previous versions of Excel, objects such as buttons embedded in worksheets could only respond to a single event, usually the `Click` event. Excel 97 greatly increased the number of events that VBA code can respond to and formalised the way in which this is done by providing event procedures for the workbook, worksheet and chart sheet objects. For example, workbooks now have 20 events they can respond to, such as `BeforeSave`, `BeforePrint`, and `BeforeClose`. Excel 97 also introduced ActiveX controls that can be embedded in worksheets and UserForms. ActiveX controls can respond to a wide range of events such as `GotFocus`, `MouseMove`, and `DblClick`.

The VBE provides users with much more help than was previously available. For example, as we write code, popups appear with lists of appropriate methods and properties for objects, and arguments and parameter values for functions and methods. The **Object Browser** is much better than previous versions, allowing us to search for entries, for example, and providing comprehensive information on intrinsic constants.

Microsoft has provided an Extensibility library that makes it possible to write VBA code that manipulates the VBE environment and VBA projects. This makes it possible to write code that can directly access code modules and UserForms. It is possible to set up applications that indent module code or export code from modules to text files, for example.

Excel 97 has been ported to the Macintosh in the form of Excel 98. Unfortunately, many of the VBE help features that make life easy for programmers have not been included. The VBE Extensibility features have not made it to the Mac either.

Excel 2000

Excel 2000 did not introduce dramatic changes from a VBA programming perspective. There were a large number of improvements in the Office 2000 and Excel 2000 user interfaces and improvements in some Excel features such as PivotTables. A new PivotChart feature was added. Web users benefited the most from Excel 2000, especially through the ability to save workbooks as web pages. There were also improvements for users with a need to share information, through new online collaboration features.

One long awaited improvement for VBA users was the introduction of modeless UserForms. Previously, Excel only supported modal dialog boxes, which take the focus when they are on screen so that no other activity can take place until they are closed. Modeless dialog boxes allow the user to continue with other work while the dialog box floats above the worksheet. Modeless dialog boxes can be used to show a "splash" screen when an application written in Excel is loaded and to display a progress indicator while a lengthy macro runs.

Excel 2002

Excel 2002 has also introduced only incremental changes. Once more, the major improvements have been in the user interface rather than in programming features. Microsoft continues to concentrate on improving web-related features to make it easier to access and distribute data using the Internet. New features that could be useful for VBA programmers include a new `Protection` object, SmartTags, RTD (Real Time Data), and improved support for XML.

The new `Protection` object lets us selectively control the features that are accessible to users when we protect a worksheet. We can decide whether users can sort, alter cell formatting, or insert and delete rows and columns, for example. There is also a new `AllowEditRange` object that we can use to specify which users can edit specific ranges and whether they must use a password to do so. We can apply different combinations of permissions to different ranges.

SmartTags allow Excel to recognize data typed into cells as having special significance. For example, Excel 2002 can recognize stock market abbreviations, such as MSFT for Microsoft Corporation. When Excel sees an item like this, it displays a SmartTag symbol that has a popup menu. We can use the menu to obtain related information, such as the latest stock price or a summary report on the company. Microsoft provides a kit that allows developers to create new SmartTag software, so we could see a whole new class of tools appearing that use SmartTags to make data available throughout an organization or across the Internet.

RTD allows developers to create sources of information that users can draw from. Once you establish a link to a worksheet, changes in the source data are automatically passed on. An obvious use for this is to obtain stock prices that change in real time during the course of trading. Other possible applications include the ability to log data from scientific instruments or industrial process controllers. As with SmartTags, we will probably see a host of applications developed to make it easy for Excel users to gain access to dynamic information.

Improved XML support means it is getting easier to create applications that exchange data through the Internet and intranets. As we all become more dependent on these burgeoning technologies, this will become of increasing importance.

Excel 2002 VBA Programmer's Reference

This book is aimed squarely at Excel users who want to harness the power of the VBA language in their Excel applications. At all times, the VBA language is presented in the context of Excel, not just as a general application programming language.

The pages that follow have been divided into three sections:

- Primer
- Working with Specific Objects
- Object Model References

The Primer has been written for those who are new to VBA programming and the Excel Object Model. It introduces the VBA language and the features of the language that are common to all VBA applications. It explains the relationship between collections, objects, properties, methods, and events and shows how to relate these concepts to Excel through its object model. It also shows how to use the Visual Basic Editor and its multitude of tools, including how to obtain help.

The middle section of the book takes the key objects in Excel and shows, through many practical examples, how to go about working with those objects. The techniques presented have been developed through the exchange of ideas of many talented Excel VBA programmers over many years and show the best way to gain access to workbooks, worksheets, charts, ranges, etc. The emphasis is on efficiency, that is how to write code that is readable and easy to maintain and that runs at maximum speed. In addition Rob Bovey has written a chapter on Excel and ADO that details techniques for accessing data, independent of its format.

The final four chapters of this section, written by Stephen Bullen, address the following advanced issues: linking Excel to the Internet, writing code for international compatibility, programming the Visual Basic Editor, and how to use the functions in the Win32 API (Windows 32 bit Application Programming Interface).

The final section of the book is a comprehensive reference to the Excel 2002 Object Model, as well as the Visual Basic Editor and Office Object Models. All the objects in the models are presented together with all their properties, methods, and events. I trust that this book will become a well-thumbed resource that you can dig into, as needed, to reveal that elusive bit of code that you must have right now.

Version Issues

This book was first written for Excel 2000 and has now been extended to Excel 2002 as a component of Office XP. As the changes in the Excel Object Model, compared to Excel 97, have been relatively minor most of this book is applicable to all three versions. Where we discuss a feature that is not supported in previous versions, we make that clear.

What You Need to Use this Book

Nearly everything discussed in this book has examples with it. All the code is written out and there are plenty of screenshots where they are appropriate. The version of Windows you use is not important. It is important to have a full installation of Excel and, if you want to try the more advanced chapters involving communication between Excel and other Office applications, you will need a full installation of Office. Make sure your installation includes access to the Visual Basic Editor and the VBA Help files. It is possible to exclude these items during the installation process.

Note that Chapters 17 and 18 also require you to have VB6 installed as they cover the topics of COM Addins and SmartTags.

Conventions Used

We've used a number of different styles of text and layout in the book, to help differentiate between different kinds of information. Here are some of the styles and an explanation of what they mean:

These boxes hold important, not-to-be forgotten, mission-critical details that are directly relevant to the surrounding text.

Background information, asides, and references appear in text like this.

- **Important Words** are in a bold font
- Words that appear on the screen, such as menu options, are in a similar font to the one used on screen, for example, the Tools menu
- All object names, function names, and other code snippets are in this style: `SELECT`

Code that is new or important is presented like this:

```
SELECT CustomerID, ContactName, Phone
FROM Customers
```

whereas code that we've seen before or has little to do with the matter being discussed, looks like this:

```
SELECT ProductName FROM Products
```

In Case of a Crisis...

There are number of places you can turn to if you encounter a problem. The best source of information on all aspects of Excel is from your peers. You can find them in a number of newsgroups across the Internet. Try pointing your newsreader to the following site where you will find us all actively participating:

- msnews.microsoft.com

Subscribe to `microsoft.public.excel.programming` or any of the groups that appeal. You can submit questions and generally receive answers within a hour or so.

Stephen Bullen and Rob Bovey maintain very useful web sites, where you will find a great deal of information and free downloadable files, at the following addresses:

- http://www.bmsltd.co.uk
- http://www.appspro.com

Another useful site is maintained by John Walkenbach at:

- http://www.j-walk.com

Wrox can be contacted directly at:

- http://www.wrox.com – for downloadable source code and support
- http://p2p.wrox.com/list.asp?list=vba_excel – for open Excel VBA discussion

Other useful Microsoft information sources can be found at:

- http://www.microsoft.com/office/ – for up-to-the-minute news and support
- http://msdn.microsoft.com/office/ – for developer news and good articles about how to work with Microsoft products
- http://www.microsoft.com/technet – for Microsoft Knowledge Base articles, security information, and a bevy of other more admin-related items

Feedback

We've tried, as far as possible, to write this book as though we were sitting down next to each other. We've made a concerted effort to keep it from getting "too heavy" while still maintaining a fairly quick pace. We'd like to think that we've been successful at it, but encourage you to e-mail us and let us know what you think one way or the other. Constructive criticism is always appreciated, and can only help future versions of this book. You can contact us either by e-mail (`support@wrox.com`) or via the Wrox web site.

Questions?

Seems like there are always some, eh? From the last edition of this book, we received hundreds of questions. We have tried to respond to every one of them as best as possible. What we ask is that you give it your best shot to understand the problem based on the explanations in the book.

If the book fails you, then you can either e-mail Wrox (`support@wrox.com`) or us personally (`jgreen@enternet.com.au`, `RobBovey@AppsPro.com`, `Stephen@BMSLtd.ie`). You can also ask questions on the vba_excel list at http://p2p.wrox.com. Wrox has a dedicated team of support staff and we personally **try** (no guarantees!) to answer all the mail that comes to us. For the last book, we responded to about 98% of the questions asked – but life sometimes becomes demanding enough that we can't get to them all. Just realize that the response may take a few days (as we get an awful lot of mail).