



# EXCEL

**Mike Rippin**

**TESSELLA SUPPORT SERVICES PLC**

Issue V1.R2.M1

January 2001



## EXCEL

### Introduction

Microsoft Excel defines the industry standard for desktop Spreadsheet applications. An integral part of the Microsoft Office family, Excel has a surprising depth of application well beyond its immediate role as the “number cruncher” of the Office suite. This article aims to highlight how this potential can be realised by describing the ways in which Excel can be extended using the Visual Basic for Applications (VBA) language to create customisable applications tailored to suit business needs.

### Inherently Powerful

Excel has a whole variety of built-in worksheet functionality to manipulate and process data, a facility for user defined functions, advanced statistical and financial analysis capabilities, numerical analysis facilities, sophisticated sheet formatting and report generation, database access and query functionality, two and three-dimensional graphing of data, effortless integration with web technology, and integrated email facilities for data distribution.

Excel is easy to learn, is user friendly, has the familiar “look and feel” of all the Office family of products, and has application at all levels of sophistication from a simple personal accounts spreadsheet, to project monitoring systems, to non-linear curve fitting analysis of experimental data.

### Visual Basic for Applications

Despite the inherent breadth of application which Excel already supplies “out of the box”, it provides one more additional feature which gives it almost limitless scope: Visual Basic for Applications, or VBA. VBA is a programming language, derived from Visual Basic, but modified to encompass all the functionality of Excel. The use of VBA allows Excel to be “customised” and to provide bespoke features just as if Excel had provided them itself – menus can be created, toolbars added and dialogs displayed. Extending this further, a whole series of dialogs may be created together forming a “mini-application” running within Excel in much the same way as the many “Wizards” Excel provides already.

### Bespoke Development

Why would this be useful? The answer is simple – every business is different, and has different data processing requirements. The purchase of, or indeed the search for, a package which provides *precisely* a given set of business needs is

unlikely to succeed. Even if it does, it will be yet another package unfamiliar to end users of the system, and its unqualified acceptance cannot be guaranteed. Even the functionality provided by Excel will often not optimally accommodate a given specific requirement. Being able to *extend* Excel to provide precisely the required functionality greatly reduces the need for pre-supplied packages.

Excel VBA allows the programmer to “take control” of the many features offered in Excel, and to achieve in a single step, a whole series of operations which may take several, repetitive, menu selections. It is this power to build in bespoke functionality which matches precisely the needs of the business that is so useful. For example, pre-defined “print” functions which automatically format and set up the worksheet pages to fit a given specification; functions to process all the data in a simple accounts system to produce end of quarter summary documents; functions to simultaneously curve fit the data from several experimental measurements, post process the fitted physical parameters and plot the summary results on worksheet charts.

### **The Macro**

There are two mechanisms to achieve this “single step” functionality. The first is by using a tool provided as is by Excel – the “macro recorder”. If the user finds himself repeatedly running the same series of menu operations, starting the macro recorder first allows a macro name to be given to the set of operations which are literally recorded as the user steps through them. When they need to be performed again, the one single macro is called and *voila* – the whole series of operations is repeated exactly as before. There is an obvious drawback to this – the series of operations really is *exactly* the same, so if the user wishes to do something *even slightly* different, a new macro will need to be recorded first. There will also be no validation of data. Nevertheless, the macro recorder is rather clever, and actually creates Visual Basic code, which it writes to “code modules” in the same workbook.

The second method is through direct use of the VBA language itself, using the VBA editor and writing to the code modules directly.

### **VBA Development**

Visual Basic is now a very rich language, which has evolved over the years into one of the most popular computer languages in the world. VBA is found in many applications, not just Excel – Word, Access and Powerpoint all have a VBA element to them. It is also very easy to learn – the language syntax is simple and

intuitive, it provides the standard framework of all procedural languages (subroutines and functions, loops, 'IF-THEN' constructs and so on), and from Office '97 onwards, provides 'object-oriented' features in the form of 'class modules'. But this simplicity does not in any way undermine its power – VB is an extremely broad, 'component based' language which has allowed the basic elements to grow as fast as people can develop them. Within Excel, this is all available through VBA.

Excel has associated with it an 'object model' – the user is able to access all the Excel objects such as cells, worksheets, charts and so on from within VBA in an intuitive and straightforward manner. For example, to change the value of the cell A1 in the visible worksheet to "10" the code is:

```
ActiveSheet.Range("A1").Value = "10". Simple as that.
```

Modification of other Excel objects is just as straightforward, though there is some learning to be done – the object model is quite extensive, and allows everything to be accessed and modified: fonts, colours, selections, charts – everything.

### **User Forms and Dialogs**

Perhaps the most powerful feature provided by VBA is that of 'Forms'. Forms are the building blocks of all the Wizard dialogs that Excel provides, all the message boxes, and the various Open and Save dialogs. VBA allows *customised* forms to be created, providing the means for professional looking and robust 'applications' to be written. A Form is the 'canvas' on which buttons, labels, textboxes, lists and so on can be placed. VBA functionality can then be applied to them. For example, a button can be added, then a macro 'assigned' to the button 'press' which changes the print area of the active worksheet to print one page wide and one page tall, then to display the built-in print preview dialog.

Constructing an 'application' using forms to manipulate workbooks, rather than allowing the user to edit and modify worksheets themselves, provides a level of precise consistency and repeatability for all users, as they can only modify the workbook as defined by the 'application'. All user selections are controlled and can be validated before they are used, according to predefined requirements. This eliminates user errors (e.g. writing string characters in cells which require numbers for numerical purposes), ensures that only tried and tested techniques are used for data analysis (the user does not use the 'wrong' algorithm), and vastly speeds up the data processing sequence.

## Applications and Addins

An ‘application’ in Excel is not really an application in the usual ‘standalone’ sense. Applications written in Visual Basic are standalone – they run by themselves. Applications written in Excel are actually written *for* Excel **and can only run within Excel**. Excel ‘applications’ are called ‘addins’ – these are ‘normal’ Excel workbooks whose worksheets are invisible to the user, but whose ‘code modules’ can be accessed through public macros and events. Addins can be as simple or as complicated as the developer wishes – from a single code module or form defining a simple utility dialog (e.g to pre-format, or re-arrange, the selected range of cells), to tens of code modules and/or forms. In addition to forms and code modules, menus and toolbars can also be created and manipulated from VBA. This provides the user with seamless access to bespoke functionality amongst the already familiar menu and toolbar system provided by Excel. More importantly, bespoke menus can be controlled – items can be enabled or disabled depending on certain worksheet conditions, or the menu can be altered as particular options are selected by the user. This is another area where user options can be restricted to ensure that only those processes defined as part of the ‘application’ are executed, and only when the ‘application’ allows it.

## Interacting Systems

One very powerful feature is the ability to develop interacting systems of workbooks and addins. This is achieved by defining ‘references’ between workbooks. These references allow the code modules in one workbook, A, (which may be an addin) to be accessed from another workbook, B, as if the code was actually inside workbook B. In a system of interacting workbooks this is a very useful feature – code common to lots of addins can be placed in a single addin and referenced by all the others. This cuts down code duplication, encourages re-use, and importantly keeps down the size of the addins.

## COM

COM underlies Microsoft’s Active-X technology. The “Component Object Model” provides a standard for developing components in any language which can then be used seamlessly within any other application that understands the COM specification. As a Microsoft product, Excel benefits from being “COM-based” – this not only allows Excel worksheets and charts, or even workbooks, to be embedded in other COM aware applications (such as Word or Powerpoint), it allows these other Microsoft “objects” to be embedded within Excel. More powerful than this is the ability to embed *any* COM component within an Excel addin, and to use it as if it was part of the Excel system. This has wide-ranging

implications, and extends the boundaries of an Excel addin well beyond that provided by the Excel object model. Using COM components written by third party suppliers, or developed in-house using Visual Basic, can provide access to a huge array of external systems – access to databases, software to control machinery, reading and writing data files to and from sophisticated scientific equipment, or accessing the world wide web, to mention a few.

### **Bringing It Together**

Many of the above ideas have been used by Tessella in the development of a system of addins for a large pharmaceutical company. The situation is this: a series of experiments are carried out each with differing data processing requirements. Each experiment is well defined and conforms to clear scientific processes. A series of Excel “template” workbooks were defined within which all the experimental data is stored for a given experiment. At various stages in the experiment, new worksheets are either dynamically created to store new data, or fixed format, predefined worksheets are populated with experimental results automatically imported into the workbook. Each experimental template has associated with it an addin, which it references, containing VBA code and Forms to guide the user through each stage of the experiment, validate the data, and post process the results. Also associated with each template is a bespoke menu item which is added to the main menu bar when the workbook is visible, and removed when it is invisible – this allows many different experimental workbooks to be opened while avoiding having several unnecessary menu items. The whole suite of experimental templates is managed by an additional addin which is set to open when Excel starts up, and this adds its own menu item from which the different experimental workbooks can be selected from a dialog.

The data processing within the addins varies from simple worksheet formatting, to the reading and writing of data files to and from network drives, to dynamic chart creation and formatting, and bespoke print/preview functionality. A COM component is used to access a chemical database on the network, and an in-house COM component is used to read raw data from a variety of Mass Spectrometer machines, as well as to post-process the results for the suite of experimental workbooks.

This type of system not only allows data from several different sources on a network, in several different formats, to be integrated together in one place, it also provides the means for controlling and automating this data access from within the same system. The template workbook acts as a data ‘hub’. The users

do not need to access the network – the template will do this. They do not need to access the database, to reformat mass spec data into a user friendly form, or to plot and format experimental graphs. The template takes care of all this, and does so consistently for all users, and all experiments. More importantly, the users can stop worrying about where their data is and how much time, effort and trouble will be required to process and format it – they can concentrate on the science.

**Tessella Support Services plc**  
**Creating Software for Science and Engineering**

Tessella's services range from feasibility studies, through system design, development, implementation and ongoing support. Our expertise includes:

Data Analysis Software  
Data Capture  
Simulation Software  
Advanced Graphics  
Systems Support  
Database Applications

**Other Technical Supplements available include:**

- |   |  |
|---|--|
| <input type="checkbox"/> Archiving of Electronic Info   | <input type="checkbox"/> Object Oriented Programming   |
| <input type="checkbox"/> Active Server Pages            | <input type="checkbox"/> Pocket PC                     |
| <input type="checkbox"/> Automated GUI Testing          | <input type="checkbox"/> Portable GUI Development      |
| <input type="checkbox"/> Bayesian Statistics            | <input type="checkbox"/> Printer Technology Guide      |
| <input type="checkbox"/> Beowulf Clusters               | <input type="checkbox"/> Real Time Systems             |
| <input type="checkbox"/> C++                            | <input type="checkbox"/> Regression Testing            |
| <input type="checkbox"/> Client-Server Technology       | <input type="checkbox"/> Security and the Internet     |
| <input type="checkbox"/> COM                            | <input type="checkbox"/> Simulation                    |
| <input type="checkbox"/> Computational Fluid Dynamics   | <input type="checkbox"/> Soft Computing                |
| <input type="checkbox"/> Computer Image Processing      | <input type="checkbox"/> Software Design Methodologies |
| <input type="checkbox"/> Decision Support Systems       | <input type="checkbox"/> Software Development Cycle    |
| <input type="checkbox"/> Electronic Data Capture        | <input type="checkbox"/> Software Documentation        |
| <input type="checkbox"/> Electronic Lab Notebooks       | <input type="checkbox"/> Software Portability          |
| <input type="checkbox"/> Excel                          | <input type="checkbox"/> Software Re-engineering       |
| <input type="checkbox"/> Extending the Life of Software | <input type="checkbox"/> Software Specification        |
| <input type="checkbox"/> Federal Drug Administration    | <input type="checkbox"/> SQL                           |
| <input type="checkbox"/> FORTRAN 90                     | <input type="checkbox"/> UNIX Inter-Process Comms      |
| <input type="checkbox"/> Grid Computing                 | <input type="checkbox"/> UNIX Systems Performance      |
| <input type="checkbox"/> High Throughput Screening      | <input type="checkbox"/> UNIX Workstations             |
| <input type="checkbox"/> Instrumentation                | <input type="checkbox"/> Visual Basic 6                |
| <input type="checkbox"/> Integrated Lab Systems         | <input type="checkbox"/> WAP                           |
| <input type="checkbox"/> J2EE                           | <input type="checkbox"/> Web Services                  |
| <input type="checkbox"/> Java                           | <input type="checkbox"/> Windows 2000 Services         |
| <input type="checkbox"/> Lims                           | <input type="checkbox"/> XML                           |
| <input type="checkbox"/> Linux                          | <input type="checkbox"/> X Windows                     |
| <input type="checkbox"/> Microsoft Net                  |  |



INVESTOR IN PEOPLE

**Tessella Support Services plc**

3 Vineyard Chambers, Abingdon, Oxon, OX14 3PX, England

Tel: (+44) (0) 1235 555511 Fax: (+44) (0) 1235 553301

E-mail: [info@tessella.com](mailto:info@tessella.com) Web Address: <http://www.tessella.com>