Excel® 2007 VBA Programmer's Reference

John Green
Stephen Bullen
Rob Bovey
Michael Alexander

Wiley Publishing, Inc.

Excel® 2007 VBA
Programmer's Reference

Excel® 2007 VBA Programmer's Reference

John Green
Stephen Bullen
Rob Bovey
Michael Alexander

Wiley Publishing, Inc.

Excel® 2007 VBA Programmer's Reference

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2007 by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-04643-2

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

Library of Congress Cataloging-in-Publication Data

Excel 2007 VBA programmer's reference / John Green ... [et al.].
p. cm.
Includes index.
ISBN 978-0-470-04643-2 (paper/website)
1. Microsoft Excel (Computer file) 2. Business—Computer programs. I. Green, John, 1945-
HF5548.4.M523E92988 2007
005.54—dc22
2007004976

About the Authors

John Green lives and works in Sydney, Australia, as an independent computer consultant, specializing in Excel and Access. He has 35 years of computing experience, a Chemical Engineering degree, and an MBA.

He wrote his first programs in FORTRAN, took a part in the evolution of specialized planning languages on mainframes and, in the early '80s, became interested in spreadsheet systems, including 1-2-3 and Excel.

John established his company, Execuplan Consulting, in 1980, specializing in developing computer-based planning applications and in training. He has led training seminars for software applications and operating systems both in Australia and overseas.

John has had regular columns in a number of Australian magazines and has contributed chapters to a number of books including *Excel Expert Solutions* and *Using Visual Basic for Applications 5*. He also co-authored *Professional Excel Development* with Stephen Bullen and Rob Bovey.

From 1995 to 2005 he was accorded the status of MVP (Most Valuable Professional) by Microsoft for his contributions to the CompuServe Excel forum and MS Internet newsgroups.

John Green contributed the Introduction, Chapters 1–11, 13, 15–17, and 19 to this book.

Stephen Bullen lives in Woodford Green, London, England, with his partner Clare, daughter Becky, and their dogs, Fluffy and Charlie. He has two other daughters, Jane and Katie, from his first marriage.

A graduate of Oxford University, Stephen has an MA in Engineering, Economics, and Management, providing a unique blend of both business and technical skills. He has been providing Excel consulting and application development services since 1994, originally as an employee of Price Waterhouse Management Consultants and later as an independent consultant trading under the names of Business Modelling Solutions Limited and Office Automation Limited. Stephen now works for Barclays Capital in London, developing trading systems for complex exotic derivative products.

The Office Automation web site, `www.oaltd.co.uk`, provides a number of helpful and interesting utilities, examples, tips and techniques to help in your use of Excel and development of Excel applications.

As well as co-authoring previous editions of the *Excel VBA Programmer's Reference*, Stephen co-authored *Professional Excel Development*.

In addition to his consulting and writing assignments, Stephen actively supports the Excel user community in Microsoft's peer-to-peer support newsgroups and the Daily Dose of Excel blog. In recognition of his knowledge, skills and contributions, Microsoft has awarded him the title of Most Valuable Professional each year since 1996.

Stephen Bullen contributed Chapters 14, 18, 24–27, and Appendix B to this book.

Rob Bovey is president of Application Professionals, a software development company specializing in Microsoft Office, Visual Basic, and SQL Server applications. He brings many years' experience creating financial, accounting, and executive information systems for corporate users to Application Professionals. You can visit the Application Professionals web site at `www.appspro.com`.

Rob developed several add-ins shipped by Microsoft for Microsoft Excel and co-authored the *Microsoft Excel 97 Developers Kit* and *Professional Excel Development*. He earned his Bachelor of Science degree from The Rochester Institute of Technology and his MBA from the University of North Carolina at Chapel Hill. He is a Microsoft Certified Systems Engineer (MCSE) and a Microsoft Certified Solution Developer (MCSD). Microsoft has awarded him the title of Most Valuable Professional each year since 1995.

Rob Bovey contributed Chapters 20–22 to this book.

Michael Alexander is a Microsoft Certified Application Developer (MCAD) with more than 14 years' experience consulting and developing office solutions. He parlayed his experience with VBA and VB into a successful consulting practice in the private sector, developing middleware and reporting solutions for a wide variety of industries. He currently lives in Frisco, Texas, where he serves as a Senior Program Manager for a top technology firm. Michael is the author of several books on Microsoft Access and Excel, and is the principle behind DataPig Technologies, where he shares Access and Excel knowledge with the Office community.

Michael Alexander contributed Chapters 12 and 23 and Appendices A and C to this book.

Credits

Acquisitions Editor
Katie Mohr

Development Editor
Brian Herrmann

Technical Editor
Dick Kusleika

Production Editor
William A. Barton

Copy Editor
Kim Cofer

Editorial Manager
Mary Beth Wakefield

Production Manager
Tim Tate

Vice President and Executive Group Publisher
Richard Swadley

Vice President and Executive Publisher
Joseph B. Wikert

Project Coordinator
Jennifer Theriot

Graphics and Production Specialists
Carrie A. Foster
Denny Hager
Joyce Haughey
Jennifer Mayberry
Barbara Moore
Barry Offringa
Heather Ryan

Quality Control Technicians
Jessica Kramer
Christine Pingleton

Proofreading and Indexing
Kevin Broccoli
Sean Medlock

Contents

Acknowledgments

John Green

Thanks to Katie Mohr and Michael Alexander for getting us back together, and thanks to Brian Herrmann for melding us into a coherent whole.

Dick Kusleika deserves special mention as our technical editor. He has saved us from some embarrassment and suggested numerous improvements in the examples and text. Thank you, Dick.

I would like to thank Michael Beale for seeding some of the examples of interaction with other Office applications.

Finally, a heartfelt thank you to my fellow authors. I have handled the basics and Michael, Rob, and Stephen have supplied the benefits of their specialized knowledge in the higher-level topics to take us further than I would have ever dared on my own.

Stephen Bullen

First and foremost, I'd like to thank my long-suffering girlfriend, Clare, for putting up with all the late nights and lonely evenings she endured while I wrote this update. Thanks also goes to Mike Alexander and Katie Mohr for their efforts in resurrecting the original author team to write this update to the book, and to John and Rob for agreeing to do it—your professionalism leaves me humbled.

Dick Kusleika is the unsung hero of this book. While the four authors could concentrate on our own chapters, Dick had to carefully read every word and check its accuracy. The credit for the amazingly high quality of this work goes to him, while any remaining errors are ours.

Of course, without the Excel team at Microsoft, we wouldn't have had anything to write about, so thanks goes to David Gainer and his team for crafting an amazing update to a quite mature product, and for being so open with the Excel MVPs and wider public over the past few years. The Ribbon is the biggest change that has happened to Office for many years and Jensen Harris and Savraj Dhanjal and their teams have done a brilliant job in designing the Ribbon's UI and programmability model, respectively. I'd particularly like to thank them for listening to the (sometimes harsh) criticism from the beta testers, and for updating their designs in response.

Last, I'd like to thank you, the reader, for buying this book, writing the five-star reviews on Amazon and recommending it to all your friends and colleagues!

Mike Alexander

I would like to first thank the original authors—John Green, Stephen Bullen, and Rob Bovey—for agreeing to reclaim their work. Believe me when I say that these men are very well respected among professional Excel developers, and it is an absolute honor to be associated with their work.

Acknowledgments

A big thank you goes to Katie Mohr for joining me in lobbying to get the original author team back on board. It is safe to say that without her efforts, this title would not be the superb product it is today. I would also like to thank Brian Herrmann and the professionals at Wiley for all of their time and resources in helping this ambitious title come to fruition.

Dick Kusleika is definitely the "the fifth Beatle" of this book. Dick clearly put a lot of time and effort into keeping us honest and ensuring that our work is as clean as possible. A solid technical editor is paramount for an all-encompassing reference like this one, and Dick Kusleika really came through for all of us.

A very special thank you to Mary for putting up with all of my crazy projects. The royalty checks are in the mail, my love.

Introduction

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.

A number of spreadsheet applications 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 runaway 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 in 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 because 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 manually operated 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 made 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 was 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.

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 has decreased 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 you to write well-structured code. You can also write badly structured code, but by following a few principles, you 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 onward, 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 formalized the way in which this is done by providing event procedures for the workbook, worksheet, and chart sheet objects. For example, in Excel 2007 workbooks have 29 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 you write code, pop-ups 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 you 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 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 onscreen 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 also introduced only incremental changes. Once more, the major improvements were in the user interface rather than in programming features. Microsoft continued to concentrate on improving web-related features to make it easier to access and distribute data using the Internet. New features that can be useful for VBA programmers included a new `Protection` object, SmartTags, RTD (Real Time Data), and improved support for XML.

The `Protection` object allows selective control over the features that are accessible to users when you protect a worksheet. You 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 you can use to specify which users can edit specific ranges and whether they must use a password to do so. You 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 pop-up menu. You 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 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.

Improved XML support meant that it was getting easier to create applications that exchange data through the Internet and intranets. As everyone becomes more dependent on these burgeoning technologies, XML support becomes of increasing importance.

Excel 2003

Excel 2003 continued to introduce new web-orientated features, including improved support for XML and improved online help and the ability to share and update data using Windows SharePoint Services.

Excel 2003 introduced corrected versions of a number of Excel's statistical functions.

The List feature was introduced to allow easier management of a database table. Lists make it easier to sort, filter, and edit data. Lists can also be integrated into SharePoint to share data via the Internet.

New features were introduced to enhance document sharing and management of access rights. Side-by-side comparison of workbooks was also introduced.

Excel 2007

Excel 2007 represents the greatest change in Excel since Excel 97. The most impact will be made by the new user interface, which uses the Ribbon as the primary navigation tool, replacing menus and toolbars. Although the Ribbon is probably much easier to digest for new users, it means that experienced users need to be re-educated. From a developer's point of view, the Ribbon is a major challenge requiring a whole new approach in application interfaces and a completely new set of programming rules.

Excel 2007 lifts many of the old limits, supporting 1,048,576 rows and 16,384 columns, for example. There are many changes to the way features are accessed so that PivotTables and charts are more accessible and easier to manipulate, as are many other features.

The List feature of Excel 2003, which handles database tables, has become the Table feature in Excel 2007 and is easier to use and has more capabilities. Sorting and filtering have been redesigned. You can sort on up to 64 keys simultaneously, for example. Enhancements have also been made in the range of external data sources that are now accessible, and the ways in which the data is accessed have been improved.

New file formats are used in Excel 2007, which are not compatible with previous versions although data can be saved back to older formats with the loss of any new features. If you want to have VBA code saved with a workbook, the format of the file is different compared with a standard workbook file.

Security concepts have been redesigned, introducing the Trust Center. You can now designate folders as "trusted," and macros in these folders will be allowed to run without needing digital certificates.

For a VBA programmer there are a number of new objects to be discovered and new concepts to be learned.

Excel 2007 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 loosely divided into three sections:

- Primer (Chapter 1)
- Working with Specific Objects (Chapters 2–27)
- Object Model References (Appendices A–C)

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, and so on. 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, the chapters devoted to accessing external databases detail techniques for accessing data in a range of formats.

The final four chapters of the book 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).

Finally, the appendices are a comprehensive reference to the Excel 2007 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

Previous editions of this book were able to cover all versions of Excel from Excel 97 onward, because the changes in the Excel object model and user interface were relatively minor. The changes in Excel 2007 have meant that it is no longer possible to do this without filling the book with complicated alternatives. This book applies to Excel 2007.

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