

## Bonus Chapter 2

# Ten VBA Do's and Don'ts

.....

**I**f you are reading this bonus chapter, you've probably read most of the content of this book and are familiar with Excel VBA. This chapter gives you some advice you should take into account when you start developing your own VBA solutions. Following these guidelines is no panacea to keep you out of (programming) trouble, but can help you avoid pitfalls that others have stumbled over.

### *Do Declare All Variables*

How convenient it is: simply start typing your VBA code without having to go through the tedious chore of declaring each and every variable you want to use. Although Excel allows you to use undeclared variables, doing so is simply asking for trouble.

If you lack self-discipline, add "Option Explicit" at the top of your modules. That way, your code won't even run if it includes one or more undeclared variables. Not declaring all variables has only one advantage: You save a few seconds of time. But using undeclared variables will eventually come back to haunt you. And I guarantee that it will take you more than a few seconds to figure out the problem.

### *Don't Confuse Passwords with Security*

You spent months creating a killer Excel app, with some amazing macros. You're ready to release it to the world, but you don't want others to see your incredible macro programming. Just password-protect the VBA Project and you're safe, right? Wrong.

Using a VBA password can keep most casual users from viewing your code. But if someone *really* wants to check it, he'll figure out how to crack the password. Bottom line? If you absolutely, positively need to keep your code a secret, Excel isn't for you.

## Do Clean Up Your Code

After your app is working to your satisfaction, you can clean it up. Code housekeeping tasks include the following:

- ✓ Make sure every variable is declared.
- ✓ Make sure all the lines are indented properly so the code structure is apparent.
- ✓ Rename any poorly named variables. For example, if you use the variable *MyVariable*, there's a pretty good chance that you can make the variable name more descriptive. You'll thank yourself later.
- ✓ If you're like me, your modules probably have a few "test" procedures that you wrote while trying to figure something out. They've served their purpose, so delete them.
- ✓ Add comments so you'll understand how the code works when you revisit it six months from now.
- ✓ Make sure everything is spelled correctly — especially text in UserForms and messages boxes.
- ✓ Check for redundant code. If you have two or more procedures that have identical blocks of code, consider creating a new procedure that others can call.

## Don't Put Everything in One Procedure

Want to make an unintelligible program? An efficient way to accomplish that is by putting all your code inside one nice big procedure. If you ever revisit this program again to make changes to it, you're bound to make mistakes and introduce some fine-looking bugs in there.

Do you see the problem? The solution is modular code. Split your program into smaller chunks, where each chunk is designed to perform a specific task. After you pick up this habit, you find that writing bug-free code is easier than ever.

## Do Consider Other Software

Excel is an amazingly versatile program, but it's not suitable for everything. When you're ready to undertake a new project, take some time and consider all your options. To paraphrase an old saying, "When all you know is Excel VBA, everything looks like a VBA macro."

## ***Don't Assume That Everyone Enables Macros***

As you know, Excel allows you to open a workbook with its macros disabled. In fact, it's almost as if the designers of Excel 2007 want to *encourage* users to disable macros.

Enabling macros when you open a workbook from an unknown source is not a good idea, of course. So you need to know your users. In some corporate environments, all Microsoft Office macros are disabled and the user has no choice in the matter.

One thing to consider is adding a digital signature to the workbooks that you distribute to others. That way, the user can be assured that it actually comes from you, and that it hasn't been altered. Consult the Help system for more information about digital signatures.

## ***Do Get in the Habit of Experimenting***

When I work on a large-scale Excel project, I usually spend a significant amount of time writing small VBA "experiments." For example, if I'm trying to find out about a new object, method, or property, I'll just write a simple Sub procedure and play around with it until I'm satisfied that I have a thorough understanding of how it works — and the potential problems. Setting up simple experiments is almost always much more efficient than incorporating a new idea into your existing code without the understanding that those experiments bring.

## ***Don't Assume That Your Code Will Work with Other Excel Versions***

Currently, at least five different versions of Excel for Windows are in use around the world. When you create an Excel app, you have absolutely no guarantee that it will work flawlessly in older versions or in newer versions. In some cases, the incompatibilities will be obvious (for example, if your code refers to cell XDY877322, you know that it won't work in versions prior to Excel 2007 because those versions used a smaller worksheet grid. But, you'll also find that things that *should* work with an earlier version, *don't* work.

And if Excel for Macintosh users will use your application, you should definitely plan on incompatibilities.

The only way to be sure that your application works with versions other than the one you created it with is to test it on those versions.

## *Do Keep Your Users in Mind*

Excel apps fall into two main categories: those that you develop for yourself, and those that you develop for others to use. If you develop apps for others, your job is much more difficult because you can't make the same types of assumptions. For example, you can be more lax with error handling if you're the only user. If an error crops up, you'll have a pretty good idea of where to look so you can fix it. If someone else is using your app and the same error appears, they'll be out of luck. And, when working with your own application, you can usually get by without instructions.

You need to understand the skill level of those who will be using your workbooks, and try to anticipate problems that they might have. Try to picture yourself as a new user of your application, and identify all areas that may cause confusion or problems.

## *Don't Forget About Backups*

Nothing is more discouraging than a hard drive crash without a backup. If you're working on an important project, ask yourself a simple question: If my computer dies tonight, what will I have lost? If your answer is more than a few hours' work, then you need to take a close look at your data backup procedure. You *do* have a data backup procedure, right?