

VBA Foundations, Part 4

A Tutorial in VBA for Beginners—The Fourth in a Twelve Part Series

Richard L. Binning / rbinning@atbi.com

In the last issue we explored the Visual Basic Editor in depth. In this issue we will cover some basic (pun intended) VB concepts including the anatomy of a macro and how to write a simple macro. We will utilize the VBE learning about some of the enhancements contained in the code editor windows. As part of our exploration we will see how Microsoft's "IntelliSense" can save time, eliminate spelling errors, and reduce the amount of typing required.

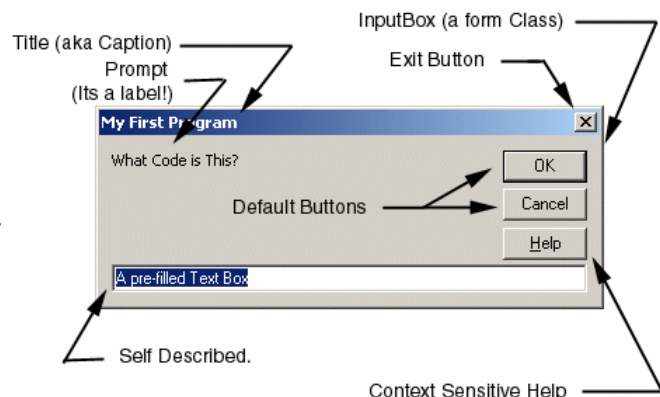
Alright! I hear you out there saying, "I thought this was a programming article, so lets start programming!" Calm down, this one is for those who work in parenthesis. If you've got AutoCAD running type the following at the command line. **Vbastmt**. Now press the "Enter" button (Don't worry my salty friends, I haven't forgotten about you, you can hit that space bar with your thumb!). You should now see your prompt change from the ever-present **Command:** to a new prompt **Expression:**. Now type the following string shown below at the **Expression:** prompt and press the enter key. (Yes that's right! The space bar stopped functioning as an enter key while you were typing in the following code.)

```
InputBox "What Code is This?", "My First Program", "A pre-filled Text Box",,"Help.htm",1000
```

The following graphic is a snapshot of your first VBA code in action. Was AutoLISP ever that easy? I think not! Take a minute and compare the code above and notice where it appeared in the graphic as shown. You, my friend are a programmer.

You have just:

- Created Your First Object
- Called A Class
- Set Properties Of An Object (Mandatory & Optional)
- Passed Arguments Into A Function.
- Returned a Value from a Function



Where is the return value you say? It was whatever you typed into the text box before you clicked the "OK" button. We didn't declare any variable to receive the value so you'll just have to take my word for it.

Are you excited yet? Good! Now before we start slinging code like a one year old with applesauce in his hands, lets lay down some ground rules so that we begin by writing good code. It has been said, "Writing code is easy. Writing Good Code is Hard." So as tempting as it may be to just slap down a form, drag a command button on to it and double click it, lets dive into the code pool with some understanding. "What is Good Code?" you say. I say, "Thanks for playing! Good Code is all the following and more."

- **Good Code Works.** (*Preferably bug-free, but we'll just call them features when they are discovered*)
- **Good Code Is Documented.** (*This will enable you or others to fix that code when your features are reported back to you. It makes your job much easier when that feature is reported 6 months after you put this routine out in the wild and you can't even remember what it is supposed to do, let alone what it is doing now!*)

(Continued on page 2)

VBA Foundations, Part 4

A Tutorial in VBA for Beginners—The Fourth in a Twelve Part Series

Richard L. Binning / rbinning@atbi.com

- **Good Code Executes Fast.** (*Faster than Lisp or Visual Lisp in AutoCAD's environment.*)
- **Good Code Is Re-Usable.** (*Often you'll find some other use for the code you just wrote. If you keep it simple and separate you can use it in other projects easily and efficiently.*)

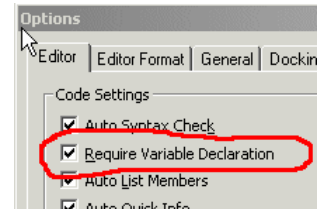
So if you'll bear with me, strive for good code, and adhere to the following rules you'll find this learning adventure will be fun, profitable, and sensible. In fact you'll be swimming across the deep end before you consider how deep that water really is.

We are going to start with some simple rules that you may not fully appreciate until much later, but they are enacted to help us in the long run. I will explain these rules more fully in later articles as the reasons for them start to become evident, so please be patient. If you want a more thorough understanding of the reasons for these rules and want to do some extracurricular reading then please direct your browser to the following websites for an in-depth discussion of Naming conventions Reddick (RVBA) Naming Conventions and Coding conventions Reddick (RVBA) Coding Conventions.

A Beginner's Rules for Writing Good Code (for Experts too!)

1. Option Explicit – All projects should be created and maintained with this VBE option turned on. You should type this phrase at the top of each form code window, module, and class. To turn this on by default and let the VBE type it automatically for you do the following inside the VBE.

- Select the "Tools" menu item and click "Options". Ensure that every project has "Require Variable Declaration" checked on as shown in the graphic.



2. Always Include Error Handling – Every procedure should have some level of Error handling no matter how trivial the procedure may seem when it is written. You will be surprised the things users of your routines will try which can lead to errors.

3. Write Well Structured Code – Limit the Go To statements to the Error Handlers and only allow one exit point for each procedure.

4. Use Meaningful Variable Names – With VBA's built in "Intellisense" and "Code Completion" features creating abbreviated variables no longer make sense. These acronyms also make your code harder to read, understand and debug later by you or other members of your team. We'll explore this Rule later. In the meantime consider this, which is easier to understand as a string variable? strUserName or UNS

5. Comment, Comment, Comment, then Comment Some More – Many times the purpose of a function or procedure is not readily apparent on the first read through the code. If you or others have to re-visit code in the future, wouldn't it be practical and efficient to provide a roadmap for their use? You'll thank me for this later.

6. Use Standard Indention – Make your code easier to interpret by indenting as you progress through your code. Which example is harder to read, *a* or *b* below?

- If Cint(strReturn) = intValue Then MsgBox strTest Else MsgBox "Thanks for Playing!" End If
- If Cint(strReturn) = intValue Then
 MsgBox strTest
Else
 MsgBox "Thanks for Playing!"
End If

(Continued on page 3)

VBA Foundations, Part 4

A Tutorial in VBA for Beginners—The Fourth in a Twelve Part Series

Richard L. Binning / rbinning@atbi.com

Okay, enough rules for now. We'll discuss other rules as we progress through out these articles. Lets begin our code adventure by diving into the deep end by writing a routine you'll be able to use everyday. How many of you open more than one drawing at a time? Okay how many keep those drawings open while you switch back and forth between them. Ever want to shut down all the drawings and purge them and save them at the same time? Can we make it automatic? You bet!!!!

Lets open the VBE by selecting "Tools" from the AutoCAD menu bar, select "Macros" and click on "Visual Basic Editor". We should now have an empty project ready and waiting for some code. (If not switch back to your AutoCAD window, remember "Focus"?, and type "VBAMAN" at the command line and select the "New" button, then switch "Focus" back to the VBE). Okay, everyone on the same page now? Lets do it. We should think through this routine before we get started. Things we know:

- We have multiple drawings open.
- We wish to purge all drawings.
- We wish to save all drawings.
- We wish to close all drawings.
- We wish to close AutoCAD.

If we had to write a simple series of sentences to describe this routine we might write the following: "For each drawing that is currently open in AutoCAD, I would like to automatically switch to each drawing, purge the drawing, save the drawing, close the drawing, continue doing these steps until no drawings are open then close AutoCAD." Okay since I did describe this routine this way lets use it to start our documentation (we'll also convert it into the code in our routine...this is called writing Pseudo Code and can help us to get a head start on our comments!)

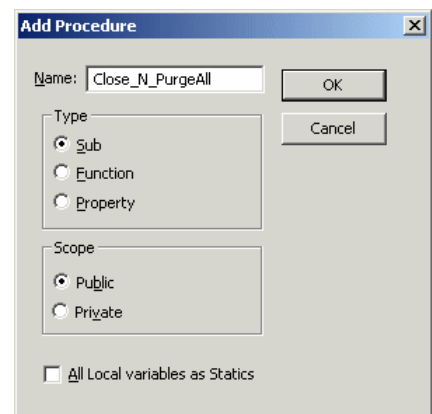
Step 1.) Block in the description from the paragraph above and place it on the clipboard (Ctrl + C or Edit > copy).

Step 2.) Switch to the VBE. Locate the "Project Explorer" and double click on the "ThisDrawing" class under your project. This will open the code window where we can drop our description. See Graphic.

Step 3.) As always there are multiple methods for creating a new procedure in our project. Click in the white space under the words "Option Explicit", now your cursor should start flashing on and off with a vertical bar. Select the "Insert" menu and choose "Procedure" from the pull down. This will pop up the Add Procedure dialog box. Notice the options available for future use and make sure that your selections are as shown (right). We are creating a Sub (We'll discuss the differences between Subs, Functions, Properties at a later time). For now we will leave the scope Public (Another topic for later). Please type in the name as shown (note: hyphens are unacceptable) and select the OK button. Now look at the sub that was just created under the words "Option Explicit" it should look as shown below. Like magic isn't it? (All you keyboard mongers can just type it all in can't you!)

Option Explicit

```
Public Sub Close_N_PurgeAll()  
End Sub
```



(Continued on page 4)

VBA Foundations, Part 4

A Tutorial in VBA for Beginners—The Fourth in a Twelve Part Series

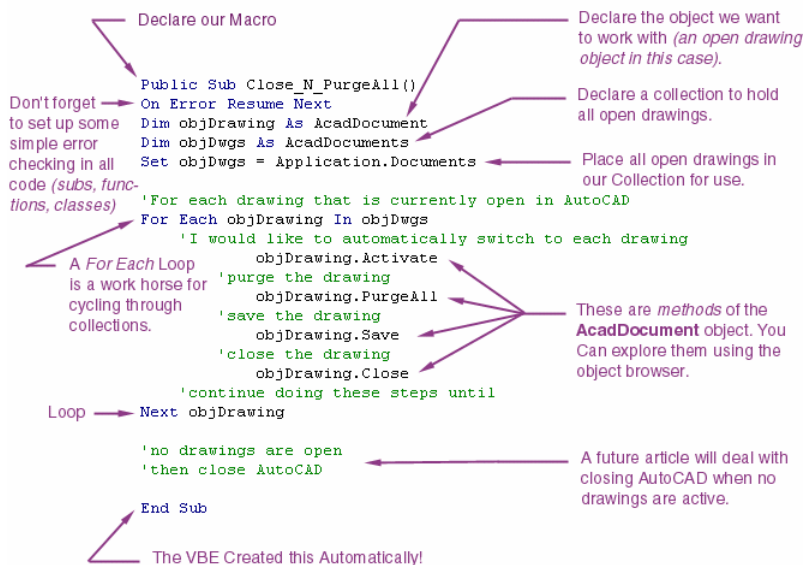
Richard L. Binning / rbinning@atthi.com

Step 4.) Click directly under the word Public then paste (“Ctrl” + “V”) in the description from Step 1 above so that your sub is identical to that shown below: (Notice you’ll get a compile error upon hitting return and the text will turn red. This is normal, just place a single quote before the line and then start adding returns so that our description is spaced out on multiple lines as shown in the graphic, remember each “comment” line must be prefaced with a single quote).

Option Explicit

```
Public Sub Close_N_PurgeAll()  
  
'For each drawing that is currently open in AutoCAD  
' I would like to automatically switch to each drawing  
'purge the drawing  
'save the drawing  
'close the drawing  
'continue doing these steps until  
  
'no drawings are open  
'then close AutoCAD  
  
End Sub
```

Step 5.) We have now created the basic commented skeleton for the new procedure in our project. In order to work with anything in the AutoCAD object model we have to have an object to work with. Looking at our description what objects do you think we are going to need? We will need a drawing object. Lets look further at our description and identify any action words (verbs). Lets see we have switch, purge, save, close, continue and our final close for AutoCAD itself. What else can we see in our description? What does the word continue conjure up? Repeat, loop, next? Yes! There is some kind of repetition. What else? What about the word until? That is a condition right? So we need to meet a condition and when the condition (no more open drawings) is True we can do our final close. Now look at the rewritten code example as shown. Can you see where I have created the code directly from our initial description? Good. Now type this code in exactly as shown. When you are finished, please take the opportunity to save this project (Filename: Purgeall.dvb) somewhere on your hard drive, preferably within your AutoCAD support file search path.



writing extensive VBA routines. The code will run as shown, although it won't be able to close AutoCAD for you yet. We will come back to this example in a later issue and complete it so that it will match our description in full. Until then you can run it as is, it will purge all drawings, save them (Only if they are named...Drawing#.dwg that is created by default will not be able to be saved because AutoCAD doesn't think it is a named drawing. More on that later.), and close them until no drawings are left open. For now you will continue to exit AutoCAD as you normally would. We will discover the secret to closing AutoCAD under all conditions in a future article.

Step 6.) So how do we run this macro? Since we are already in the VB editor we can simply go to the toolbar and select the “play” button. (It is a right facing solid triangle) What if we want to run it later when the

(Continued on page 5)

VBA Foundations, Part 4

A Tutorial in VBA for Beginners—The Fourth in a Twelve Part Series

Richard L. Binning / rbinning@atthi.com

VBE is not already open. We have many ways to load it and run it, using the VBA Manager or typing VBA-LOAD at the command line and following with VBARUN after we have it loaded. If you are like me you prefer to create a shortcut in AutoLISP or Visual Lisp to load it. For those who already have some customization skills in AutoLISP you can load the following AutoLISP macro into your custom lisp file. Those that don't have any custom tools running or don't know how to create such an animal can email me at the address above and request a copy of the macro and a custom lisp file for loading it by putting the words "VBA-PurgeAll" in the subject line of your email. The next step describes how this AutoLISP file is setup and used.

Step 7.) A simple macro loader can be created by utilizing the "findfile" function and using the following format:

```
(defun c:qc (/ Purge-it) ;create the short keys and declare a local variable
  (prompt "\nMacro: Purge, Save, Close all Open Drawings Loaded!...") ;tell the user what is going on
  (vl-load-com) ;make sure the vlisp editor is awake
  (setq Purge-it (strcat (findfile "Purgeall.dvb") "!ThisDrawing.Close_N_PurgeAll")) ;store the loading string
  (vl-vbarun Purge-it) ;using visual lisp call the macro and run it
  (princ) ;prevent the word nil from appearing at the command line
)
```

If you copy the preceding code and place it in a file named acad.lsp and save this file and "Purgeall.dvb" to a folder in your search path. You could add a folder to your search path by right clicking and selecting "options" – "support file search path." Click the "add" button then click the "browse" button and search for the acad.lsp file you just created. We will discuss how this loader works in a later article. Now restart AutoCAD and this time and every time after you can simply type QC at the command line to run this macro. (Note: a dialog box may open asking if you want to load "acad.lsp" with every drawing? If so, then you should choose the second option to load with every open drawing. If you choose the first option then this utility will only be available from the first drawing you open.

Step 8.) Not really a step but lets review what we have done so far. Since I knew you wanted to jump right in, we wrote this macro without discussing some of the key concepts of the VBA language such as conditional looping, variables, objects, etc. We will cover these topics in more depth in future articles.

Now for the quiz. As you were typing in the VBA code shown above you may have noticed that when you typed the word objDrawing and typed the period (a.k.a. dot operator) immediately the VBE popped up a window with available code for you as a guideline. This is the "Intellisense" feature I mentioned earlier. When it is onscreen you can simply use your mouse or "arrow" keys to select the method, property, or event associated with the object and let the VBE complete that portion of code for you. Now for the code completion that is extremely useful when you are declaring meaningful variable names as recommended. Go back to your code window and type in the following string "objD" (without the quotes). Now hold the "Ctrl" button down and click the space bar. A box similar to the "Intellisense" window will appear with a list of objects, properties, constants, etc. This is "Code Completion" in action and eliminates the justification for abbreviating variables and functions as well as ensuring correct spelling.

In summary, we have "basically" covered some beginning VB concepts including; Defining Good Code, Reviewing Common Coding Guidelines, Exploring the anatomy of a macro, Discovering some of the code editor enhancements like Microsoft's "IntelliSense" and "Command Completion" utilities, and creating our first useful Macro. We will return next month with even more code and begin explaining some of the things we mentioned today such as, variables, objects, loops, etc. See you next month.