Using VBA with MicroStation V8

Unleash the Power

By Mark Stefanchuk

The Phocaz Group, LLC

Creating Amazing Apps for Engineers

709 Chambers Drive, Huntsville, AL 35801

mark.stefanchuk@phocaz.com

Contents

The following workshop was originally published in spring of 2002. It has been edited and updated where needed.

Introduction:

I'm going to start out by bragging a little.

My hockey team placed second this year. We were a great team, but the first-place team was better, and their average age was about 20. If you have watched or played hockey you understand that it is a game of speed, and transition. I mention this because as older novice players we have trouble with both. Transition means you move from defense to offence. By passing the puck from deep in your zone to a forward you can take advantage of the other team's offensive positioning and get an opportunity to score.

Now just bear with me a moment and I will connect this to CAD.

To keep it simple I tell my team to transition from defense to offense by making outlet passes up the boards so that even if the opposing player intercepts the puck they can't get a shot on net. It requires effort but it works. Despite my excellent advice it seems like every shift someone tries the outlet pass up the middle. Nine times out of ten the puck is intercepted. And since we don't have Dominic Hasek in net it usually means a goal. (Hasek: Detroit Red Wings Goalie – you may recall that in 1998 he single handedly won Olympic gold for Czech Republic)

Why do we repeat the same things over and over again when we know it's not the best approach? Because it's easy and sometimes it works. That tenth try we get a cool break away and maybe we score. It looks good. It appears to get the job done. So, MicroStation VBA is like that outlet pass up the boards. It solves problems, it works, it requires some effort but you will always have the advantage. All of us have had a similar experience using MicroStation. It might go something like this: You're the CAD lead for your discipline and there is a 50 % submittal due. At some time before the Fedex deadline you notice that your entire package says 25% submittal instead of 50%. And, you notice that the string exists in every file instead of in a common reference file. Oops!

What's the fix? Well when there's a deadline you just pop the string into the border file since every file references the same border, you open every file and delete the submittal stamp, re-plot and make 12 copies. So you start, you make the change to the border, that was easy, now you begin going file to file – open, delete, fit, save settings, compress - repeat. 500 files later... Right?

Not much time left for plotting.

So, welcome to our session - Unleash the Power of MicroStation VBA! Or as Bentley prefers to call it – Using VBA with MicroStation V8 …

In the next hour and 45 minutes I will introduce you to VBA and allow you to roll up your sleeves and dig into the VBA interface and the MicroStation object library. I'll be assisted today by Mark Anderson of Bentley Systems. Mark is a super sized storehouse of programming knowledge. He's been with Bentley for many years, and I consider him to be the most important link between Bentley and those of us tasked with automating MicroStation.

Now, before we get to me, I need some information so let's do a …

Quick Survey – How many have done some form of MicroStation programming – UCM, MDL, Macro's? How about any programming at all? Has anyone here tried to use MicroStation VBA?

Great, so some of you know who I am. For everyone else, my name is Mark Stefanchuk. I work for a The Phocaz Group, LLC and, I am one of two co-founders of cadgurus.com (an old web destination for CAD users, shut down in 2002).

Phocaz (pronounced focus) is a development studio. We help engineers, and architects solve their design and business workflow problems through software development. Our practice is centered around

CAD – Autodesk, MicroStation and related vertical software. We do branch out from time to time especially now with the increased demand for mobile applications.

About this Session:

This hands-on workshop has its origins in two seminars I presented in early April. Today's session however, differs in two distinct ways – First, it is interactive where you will have an opportunity to try out some VBA features, and second it focuses on a single detailed example demonstrating a few features of the MVBA object model.

My intent is to make sure you leave today with an introduction to the VBA integrated development environment, and the MicroStation Object Library.

If you are not a VBA programmer, this session will not turn you into a programmer or developer. But, you will leave this session with an excellent understanding of the process. That is the process of defining and managing (whether you "do it yourself" or outsource) a VBA project.

Ok, I've broken the session into the following lessons.

In Lesson 1 I'll ask you some important questions designed to make you to think about the project, the concept, what we will call "The Idea". We'll spend about 15 more minutes here.

Lesson 2 is where you get to test drive MicroStation VBA. We will spend about an hour creating a draw routine. Here you will learn the key components needed to build an interface and draw graphics in MicroStation.

Lesson 1: The Idea

Identifying the problem and solution is step one of our process. And, there are some important considerations before you begin coding.

1. What do you want to build?

< Are there any volunteers? No? Great! I get to pick someone. >

We had several suggestions from NEMSUG – this one was interesting because the solution isn't obvious and requires some MDL…

i.) Open and close the reference dialog from a function key. In this case, the problem could be described this way, "We want a tool that will help us to open and close the reference dialog quickly." To further complicate matters the VBA record and playback of this operation only records the open event. So only the first half of the problem is solved automatically.

One of the most interesting things about developing for Windows is the close interaction between development environments. Visual Basic can access function libraries delivered by other applications. In our case we are interested in MDL functions delivered by MicroStation. This leads us to the second half of our solution.

Using MDL in our VBA application we can access programming tools that are not available in Visual Basic. In this case we are talking about unloading the reference dialog box. Without attempting to explain the details, here's how it's done…

```
Option Explicit
Declare Function mdlDialog_closeCommandQueue Lib "stdmdlbltin.dll" (ByVal dbP As Long) As Long
Declare Function mdlDialog_find Lib "stdmdlbltin.dll" (ByVal dialogId As Long, ByRef ownerId As Integer)
As Long

Sub openRefDlg()
    CadInputQueue.SendCommand "MDL KEYIN REF DIALOG REFERENCE"
```

```
End Sub


Sub closeRefDlg()
    'DIALOGID_ReferenceFile (-7) ::: this is from mdl\include\dlogids.h
    mdlDialog_closeCommandQueue mdlDialog_find(-7, 0)
    CommandState.StartDefaultCommand
End Sub
```

Lines 3 and 4 tell visual basic how to interpret the new functions. The rest of the code is just two function calls. The first is the open reference dialog sub routine created using the record features. The second routine calls the close request for the reference dialog using the MDL functions.

Since the solution wasn't very difficult go ahead and create the VBA. A few minutes of programming and you're done. Email the solution to everyone that has the same problem along with instructions so that they can set up the function keys. And be sure to tell the boss so that you can get a bonus for saving the company several hours of production time per year, per week, per day!

ii.) Draw footings based on design parameters.

Draw routines are an excellent example of how you can really utilize the power of a tool like VBA. In this case the problem has both production and design aspects. With a good program you can improve both the quality of design and decrease the time it takes to draw footings.

With a combination of dynamics and some engineering calculation you can introduce what-if scenarios. For example, you may want to know how the dimensions of a footing need to change if the load on the footing should increase. When you change the value from 2000 psi to 2400 psi you can see the graphics change.

iii.) Set up drawing sheets automatically.

A very common desire for many CAD coordinators and Managers is the ability to complete the first 5% of a drawing package by filling in Title Block information. A sheet set up program can automate the process based on a drawing list (which you will create anyway) and do it in a few seconds. Image how long it might take to attach, scale, and fill in title block date for even 100 sheets. And to think you already have the data in a spreadsheet.

A less complicated problem is the automation of a border attachment. There are a couple of prominent methodologies used in MicroStation. Roadway and civil site projects tend to place border cells into a drawing, while Building designers tend to attach the border as a reference file.

In building design the border is usually applied a scale, such as 96:1 for for 1’-0” = 1/8" sheets or 48:1 for 1’-0” = 1/4" sheets. Automating border attachment is a task well suited to VBA, macros or MDL, and at least in the case of building design it can save a bunch of time.

So, we can identify a problem and we can always find a way to solve it using VBA. But, before we dive in a start coding, there are a few more considerations.

MMAC (Atlanta's TMC): We had several excellent suggestions from this group as well. Our discussion keyed on handling repetitive tasks: for example, setting the same reference clip mask for multiple drawings.

2. Does the solution already exist?

In other words, is there a software application commercially available that does exactly what you want to do? If the answer is yes, then buy it. Don't waste money on software development if there is an off the shelf solution that will do the things you need to do.

Whether done in house or out sourced, developing a new software application will always cost more than buying it from someone else. Any questions?

Remove the negative…

Can't buy it because it's too expensive? Then you don't have a problem. Think about it. My boss won't let me buy anything. See question 5 then review 1 through 4.

My company will never do that. My company doesn't listen to me. I can never get time for development. My company is too cheap. I'll never get time to do that. Or some other statement along the same vein… Change your attitude, change your mind… Then review question 5.

3. How do I find out if the solution already exists?

Currently the best search tool available is www.google.com. The search tool indexes almost 2 billion web pages and a search on the word MicroStation returned about 178,000 pages. (dogpile.com – combination of search engines)

Check out comp.cad.microstation, also available at www.google.com. If you still haven't found it, then post a message on one of the active news groups, like CCM - comp.cad.microstation, or bentley.microstation.*. **Update 2017: These communities are no longer active. You may find them, but it's better today to visit Bentley.com Communities. https://communities.bentley.com/**

4. Who is the Program for?

Will you be the only user? Or, will everyone else in your office be able to benefit from this project?

If you are the only one who will use it then will the time and quality improvements pay for the cost of development? Or, will you be able to do something, a calculation, a design, that you haven't been able to do before, something that will land you new customers. If it doesn't, maybe this isn't the correct solution for your problem.

On the other hand, the VBA application you propose may decrease the time it takes to draw HVAC duct, and there are 10 HVAC designers in your office. Drawing elbows you estimate takes five minutes and you can reduce that to almost zero time. You have just paid for a couple of salaries. Umm, of course you may no longer need 10 HVAC designers. I'm sure they can draw something else…

There's a pattern here and it is directly related to the return on investment. You wouldn't pay for MicroStation if you didn't make money using it. And the same goes for software you want to develop.

To summarize – your application must improve quality, save money, or make money – or all of the above.

5. How do I sell the project?

What? I'm not a sales person, I am a designer, architect, engineer. Well, that's where you're wrong. In fact you always need to be selling, whether it's the design you're working on or your outstanding abilities that show your boss you deserve that next raise.

But seriously, if you have done your homework and answered the first four questions then this step is easy. You know what the problem is, you know what it will take to solve it, and you also know how much money it will save and/or make. Go get ‘em.

Here are a couple of strategies that will help in the internal sales process. First, make a features list. These are the solutions to the problem(s) you are solving with your application. A 10 point bullet item list works well. Keep it simple. After this session is over you will know enough to create the interface and even connect a couple of controls for a simple demo. Cut and paste the dialog into you feature sheet. It looks really good. (Use Alt-Print Screen to capture any “Active” windows dialog. Then use ctrl-v to paste the dialog into a word document.)

Now, go tell everyone about it. Ask their opinion. Modify your feature sheet to match their feedback. Ask them if they will use it. Finally, you have everything you need to approach the boss. (For an example see the iCell Feature sheet at the end of this document.)

Lesson 2: The Workshop

I'm about to step you through the process of creating a draw routine in MicroStation. We will learn how to draw a door. The application will have a user form that allows you to dynamically change the size of the door. But, before we do that lets figure out how to create and edit a MicroStation VBA project.

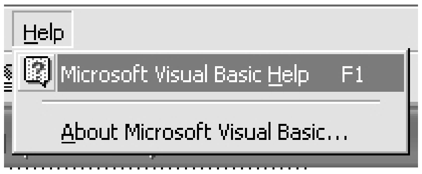
(Portions of this article were originally published by Control-Alt-Delete Magazine, Pen & Brush Publishers - Hawthorn, Australia, 1st Quarter 2002.)

Developing a Visual Basic for Applications (VBA) tool is quite easy. That's mostly a true statement. On the surface the drag and drop interface development is outstanding.

Creating the interface for simple and even complex user input and visual queues is intuitive for the developer. The code that drives the application is the complicated part. Each application that implements VBA, Word, Excel, AutoCAD, and now MicroStation all require special functions to manipulate the applications interface and data. It is definitely a powerful and extremely flexible development environment, but to create sophisticated applications requires the software management skills of a professional programmer.

Fortunately, for the CAD manager who has invested time to learn MicroStation Basic Macros, or Visual Basic Script (VBS) learning to manipulate graphics and data using VBA is a simple extension of what they already know. And fortunately for the rest of us most of the code we need to write is as simple as setting line weight, color, style and level.

Before getting started you need to know where to look for examples and help. For VBA help there are two places. The first is accessed from the VBA editor, under the help pull down menu.

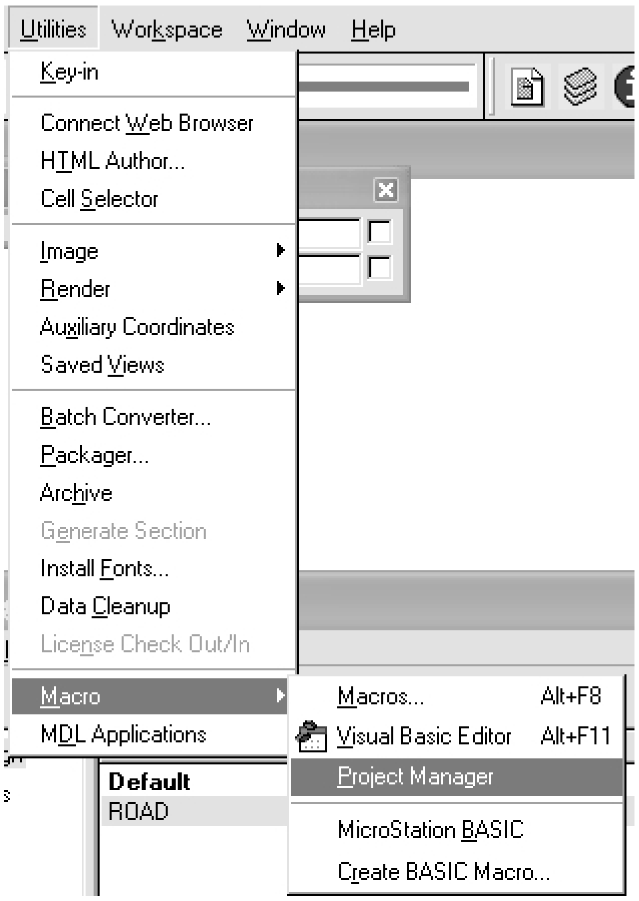

Help
Microsoft Visual Basic Help F1
About Microsoft Visual Basic...

Opening VBA Help

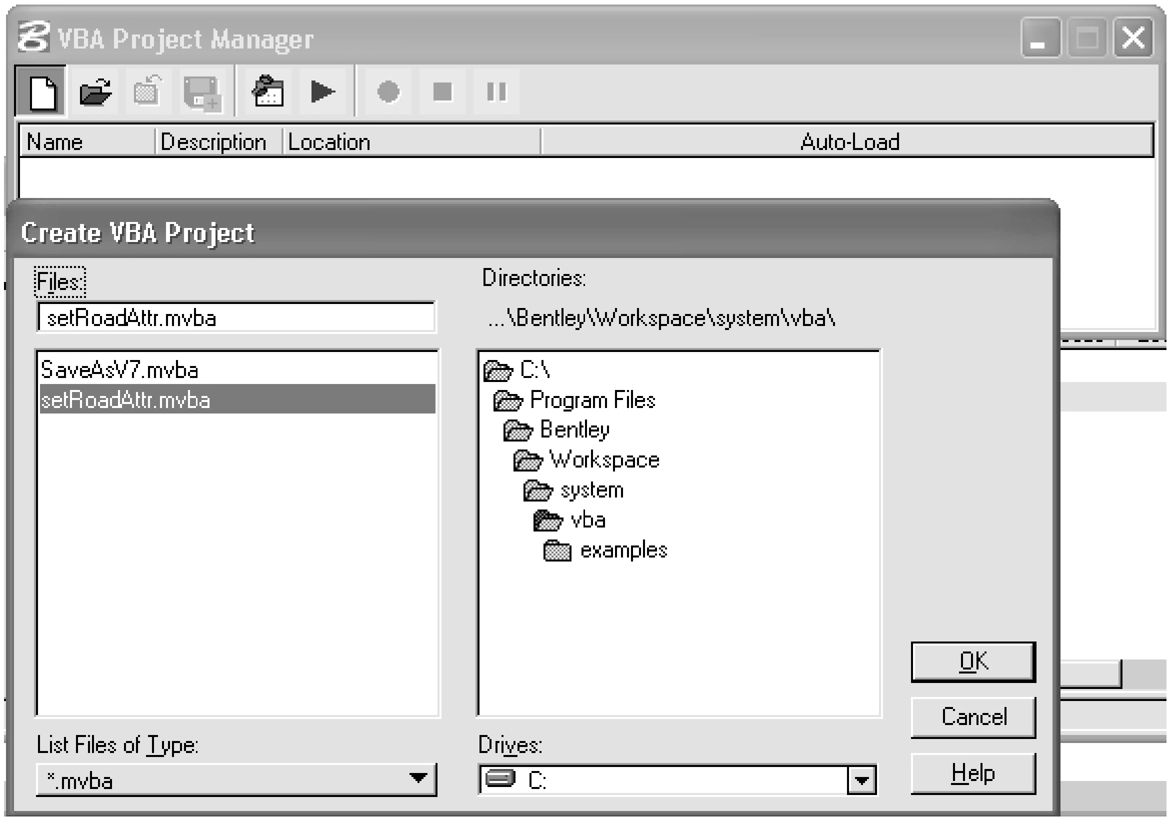
Use the VBA help to find information about VBA specific methods, objects, functions, and controls.

The second is the MicroStation VBA help which is currently located in the MicroStation root directory. The file is called MicroStation VBA.chm. Use this help file to research MicroStation VBA methods, objects, functions, and controls. The help files often are not the best place to start trying to solve a coding problem. If you need an example, or the correct syntax for a command, then by far the best method to use is a web search. I prefer www.google.com and have recently started to use www.google.com/ie for faster, trimmed down searches. When you use a search engine include key words like VB, VB6, or VBS. An example of a search I made today to find the options for CreateTextFile: "VB CreateTextFile". The sixth result was a Beginner article describing how to use the file system object. We will learn more about that when we create our second program.

And yesterday I downloaded examples posted by Bentley. The first thing to do is to create a new MVBA project. In MicroStation select Utilities > Macro > Project Manager. Right click in the project manager form and select New… from the pop up menu. A create file form displays. Type in the name setRoadAttr.mvba. The default directory is C:\Program Files\Bentley\Program\Workspace\System\VBA\.



Starting the VBA Project Manager



Files: setRoadAttr.mvba

SaveAsV7.mvba
setRoadAttr.mvba

Directories: ...\Bentley\Workspace\system\vba\

C:\
Program Files
Bentley
Workspace
system
vba
examples

List Files of Type: *.mvba

Drives: C:

Create a New Project

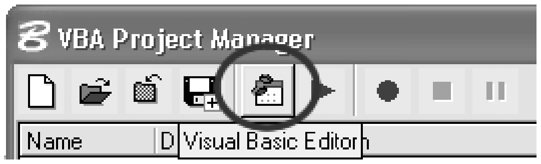
Exercise 1 – create a new project called pDoor.mvba.

In this first exercise we create a project and open the VBA IDE, find MicroStation help, and VBA help.

MicroStation VBA help provides documentation of the Bentley MicroStation V8 object model, and Windows VBA help provides documentation from Microsoft regarding VBA objects, methods, procedures, and settings.

Step 1 – use the instructions included in the previous discussion to create a project called drawDoor.mvba.

Step 2 – Open the project editor (the VBA IDE). There are a row of icons along the top of the project manager form. Select the one that looks like a visual basic form (the fifth button from the left). This is the command button which opens up the VBA editor.

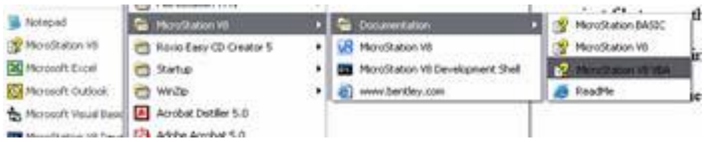


Opening the VBA Editor

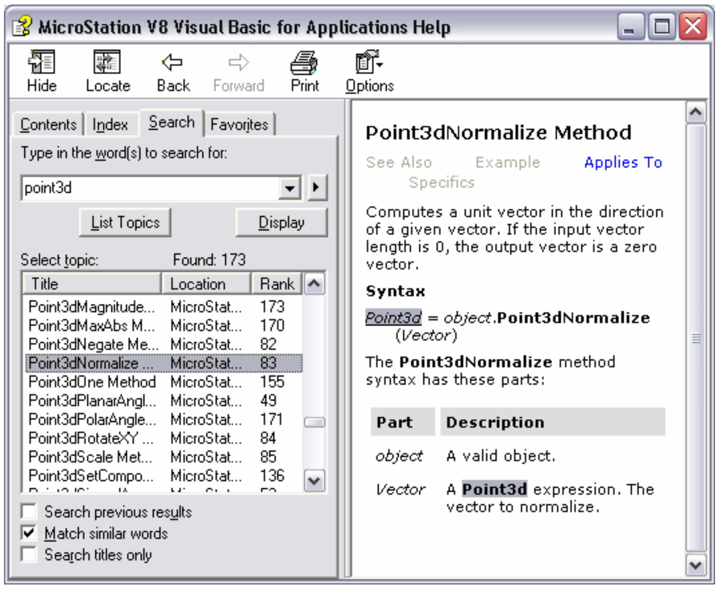
The editor is configured to show you a project file tree on the left side of the main window. There you will see a file tree that resembles A Windows Explore Form. It will show one file unexpanded. The file name is the name of the project you just created.

Step 3 – Getting help…

MicroStation object library help file can be accessed from the Windows “Start” button. Go to Start > Programs > MicroStation V8 > Documentation > MicroStationV8 VBA.



This opens the MicroStation V8 VBA Help menu. Use the search tab to find all objects, methods, and properties, related to a particular feature. For example, say you want to find all references to point3d. I use this search to list point3d methods. Click on the Title column label to sort the titles alphabetically.


MicroStation V8 Visual Basic for Applications Help

Hide | Locate | Back | Forward | Print | Options

Contents | Index | Search | Favorites

Type in the word(s) to search for:

point3d

List Topics | Display

Select topic: Found: 173

Title	Location	Rank
Point3dMagnitude...	MicroStat...	173
Point3dMaxAbs M...	MicroStat...	170
Point3dNegate Me...	MicroStat...	82
Point3dNormalize ...	MicroStat...	83
Point3dOne Method	MicroStat...	155
Point3dPlanarAngl...	MicroStat...	49
Point3dPolarAngle...	MicroStat...	171
Point3dRotateXY ...	MicroStat...	84
Point3dScale Met...	MicroStat...	85
Point3dSetCompo...	MicroStat...	136

- [ ] Search previous results
- [x] Match similar words
- [ ] Search titles only

Point3dNormalize Method

See Also Example Applies To Specifics

Computes a unit vector in the direction of a given vector. If the input vector length is 0, the output vector is a zero vector.

Syntax

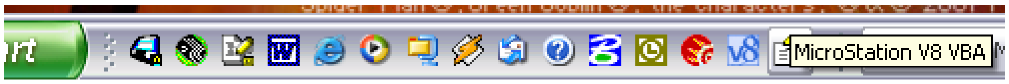
Point3d = *object*.**Point3dNormalize** (*Vector*)

The **Point3dNormalize** method syntax has these parts:

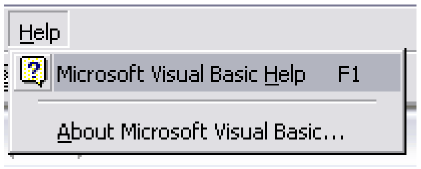
Part	Description
object	A valid object.
Vector	A **Point3d** expression. The vector to normalize.

In addition to this search tool the Index tab lists all objects, methods, and properties alphabetically. Typing in the first few characters of an MVBA component will move you to that position in the index.

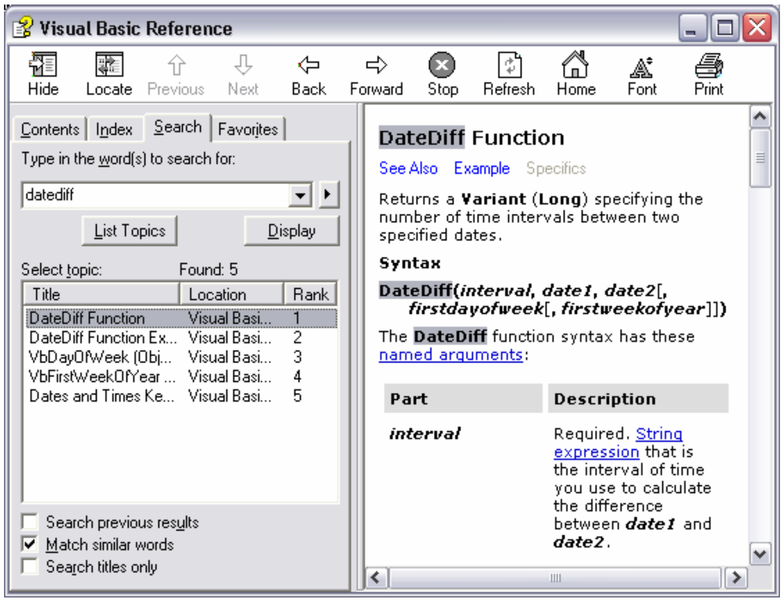
One more tip – put the MicroStationV8 VBA help icon into the quick launch section of the task bar. This will allow you to access the help file with a single click. It will also help you to minimize the number of open applications.



Standard VBA help is launched from within the VBA IDE. It will provide you with information on VB script, VBA objects, methods, and properties.


Help
Microsoft Visual Basic Help F1
About Microsoft Visual Basic...

The VBA help is similar in appearance to the MicroStation VBA help and has the same search and index features.



Type in the word(s) to search for: datediff

Select topic: Found: 5

Title	Location	Rank
DateDiff Function	Visual Basi...	1
DateDiff Function Ex...	Visual Basi...	2
VbDayOfWeek (Obj...	Visual Basi...	3
VbFirstWeekOfYear ...	Visual Basi...	4
Dates and Times Ke...	Visual Basi...	5

☐ Search previous results
☑ Match similar words
☐ Search titles only

DateDiff Function

Returns a **Variant** (**Long**) specifying the number of time intervals between two specified dates.

Syntax

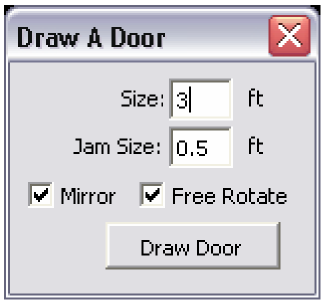
DateDiff(***interval***, ***date1***, ***date2***[, ***firstdayofweek***[, ***firstweekofyear***]])

The **DateDiff** function syntax has these named arguments:

Part	Description
interval	Required. String expression that is the interval of time you use to calculate the difference between ***date1*** and ***date2***.

Exercise 2 – create a new user form.

Ok, next we need to create a user form. This form is our dialog box – it is the place where we will change the door dimensions, set the placement options, and tell the command to start the draw routine. I almost always start here when specifying a program and when writing an application.

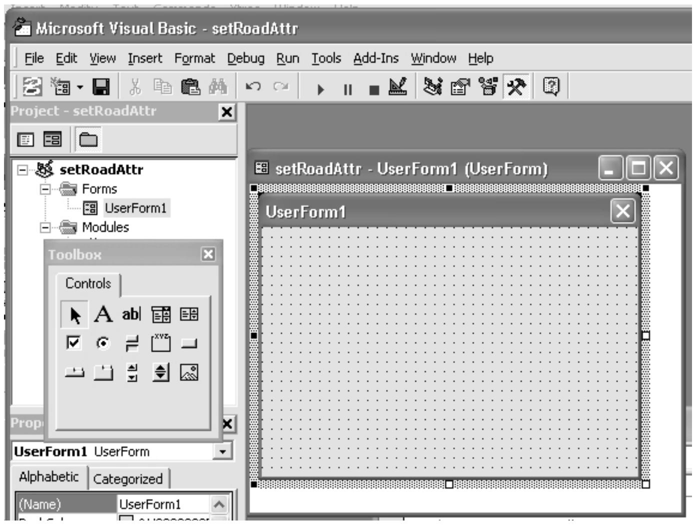


Size: 3 ft
Jam Size: 0.5 ft
☑ Mirror ☑ Free Rotate

Step 1 – Create the User Form

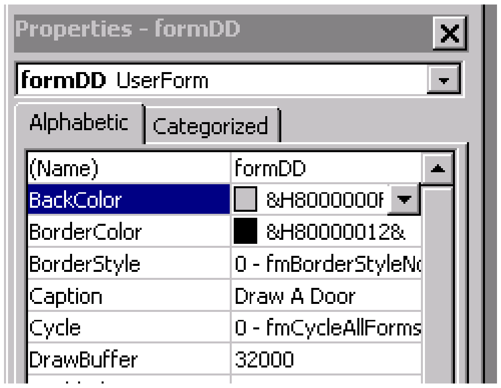
At the top of the VBA editor window is a pull down menu called Insert. Select it and then the sub menu called User Form. When you do this a new form is added to the project.

Congratulations. You have just created your first dialog box.


Microsoft Visual Basic - setRoadAttr
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - setRoadAttr
setRoadAttr
Forms
UserForm1
Modules
Toolbox
Controls
setRoadAttr - UserForm1 (UserForm)
UserForm1
UserForm1 UserForm
Alphabetic
Categorized
(Name)
UserForm1

VBA Editor with New User Form

Step 2 – Changing the user form caption and name.



Properties - formDD

formDD UserForm

(Name)	formDD
BackColor	&H8000000F
BorderColor	&H80000012&
BorderStyle	0 - fmBorderStyleNo
Caption	Draw A Door
Cycle	0 - fmCycleAllForms
DrawBuffer	32000

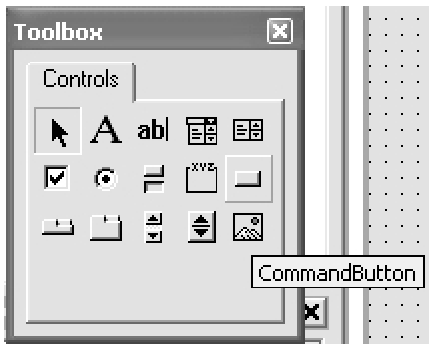
Change the name of the form to “formDD”. Change

the caption to “Draw Door”. You will find these settings in the Properties window. If your properties window is closed you can open it by typing F4, or by selecting the options from the View pull down and choosing Properties Window.

The property showModal is set to true when you create a user form. This is a read-only property and can not be set at run-time. So, you must decide at design time if the form will be modal – no other actions allowed until the form is dismissed – or modeless (the way most MicroStation dialogs work) – do other things while the dialog is open. Modal dialogs will allow you to interact with MicroStation so we need these for most commands like the place door draw routine we will build today.

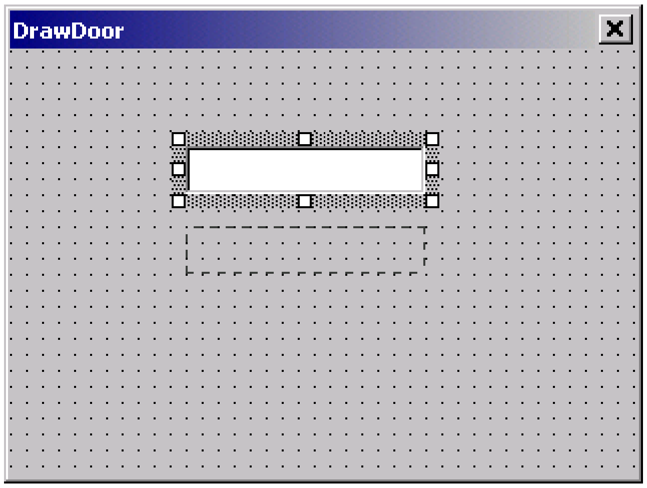
Step 3 – Add the text boxes.

There are two text boxes in this example. The first takes the overall door size dimension and the second field takes the door jam size. For simplicity we will design our interface to accept these in master units. We will learn how to check the input later. When you inserted the form, several things happened to the interface. The form was added, a second form called Toolbox opened, and a new docked window with a bunch of text in it was displayed. This last item describes the attributes of a VBA object and is referred to as the Properties Window. Controls, like buttons, text boxes, and user forms are VBA objects. Other things can be objects too, but for our purposes this is a good enough definition.



VBA Controls Toolbox

Click the TextBox control button on the toolbox and draw a text box on the user form. This is similar to drawing a box in MicroStation. Identify the first point and then the opposite corner.



VBA Drawing a Textbox

Using the properties Window change the name of the first text box to "txtDoorSize" and the name of the second textbox to "txtJamSize".

Text boxes in VBA add a selection tab before the text begins. I think this is to make selecting text easier. I don't like this. I prefer my text to be absolutely left justified. So I turn this property off.

Step 4 – Add the labels.

Next we will add four labels. The first two will identify the text boxes and the second set of labels will indicate the units for each text box.

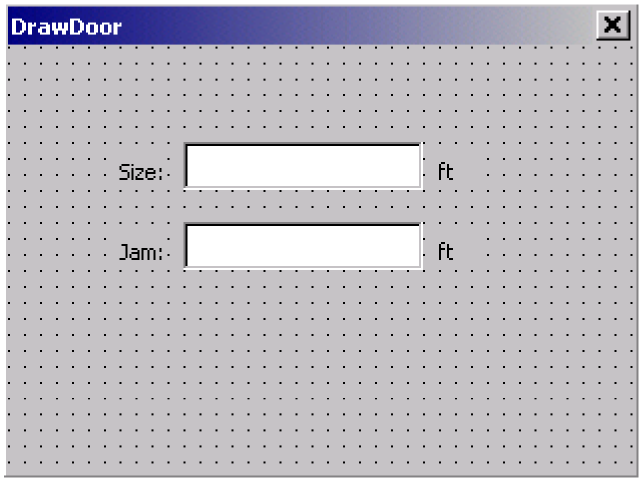
Can you add the labels? Can you figure out how to name them and change their captions? Here are the names. I've used a typical naming convention using an "lbl" prefix. This allows us to quickly identify and manage interface variables when scattered throughout our code.

lblDoorSize

lblJamSize

lblDoorUnit

lblJamUnit



Notice how I made the label names similar to the text box names? I did this so that when I am reading my code that I can mentally group the objects together. I know that txtDoorSize has labels lblDoorSize and lblDoorUnits.

When you are done you will have a form similar to the one above.

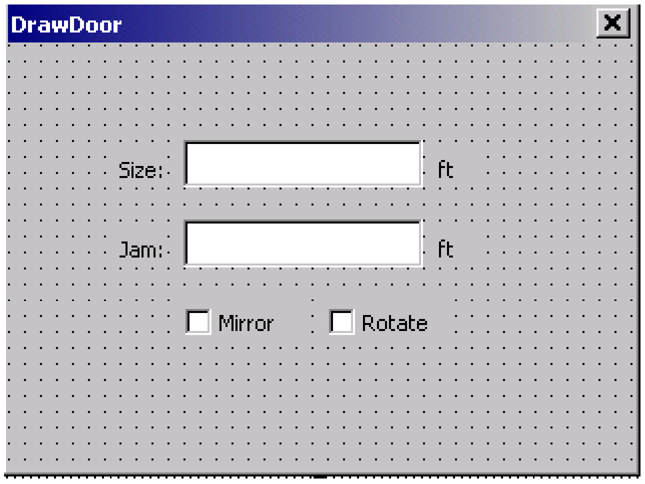
Step 5 – Add the check boxes.

Our draw door command will have two placement options. Option one will allow the user to mirror the door about the placement point by dragging the cursor around the placement point. The second option tells the place door command to do a free rotation about the placement point.

We will add two check boxes that will be used to tell the place door command to do these things.

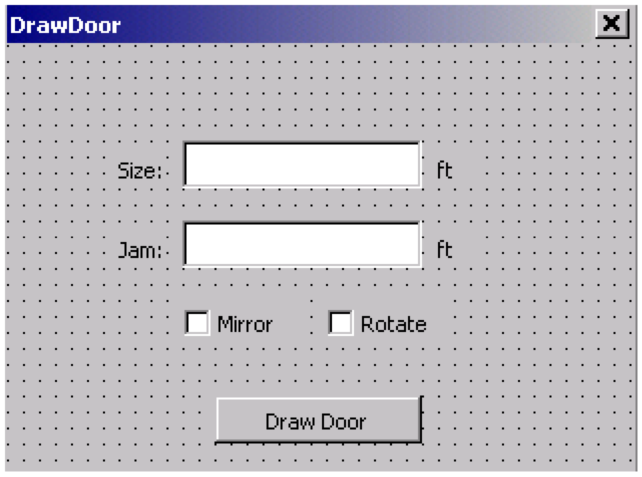
Add a check box named “chkMirror” with the caption “Mirror Door”.

Add a second check box named “chkRotate” with the caption “Rotate”. When you are done the form will look similar to the one shown here.



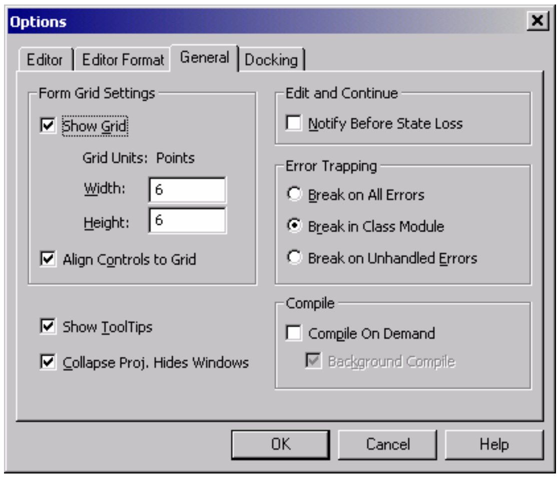
Step 6 – Add the command button.

A command button will be used to start the place door routine. Find the command button icon on the toolbox and draw a button on the form. Name it drawDD, and set the caption to "Draw Door".



Step 7 – Align, Move, Arrange Controls

By now you have already had seen how controls are moved. The grid lock is probably turned on (this is the default option) so controls when moved snap to the grid locations.



Form Grid Settings
- [✓] Show Grid
- Grid Units: Points
- Width: 6
- Height: 6
- [✓] Align Controls to Grid

- [✓] Show ToolTips
- [✓] Collapse Proj. Hides Windows

Edit and Continue
- [ ] Notify Before State Loss

Error Trapping
- ○ Break on All Errors
- ● Break in Class Module
- ○ Break on Unhandled Errors

Compile
- [ ] Compile On Demand
- [✓] Background Compile

You can turn this option off by going to the options form located under Tools>Options and selecting the General tab.

Uncheck the option that says “Align Controls to Grid”. Controls can be groupedtogether by selecting more than one control at a time – works in much the same way as the selection tool in MicroStation.

I prefer to leave align to grid turned on. If I am having particular trouble getting labels and textboxes to line up I’ll use the alignment tools located under Format > Align. The first item selected is the element aligned. For example – if I want the middles of the label lblSize and the textbox txtSize to match but I like the location of the textbox, first select the label and then the textbox. Align > Middles, in this case will move the label to match the textbox.

Exercise 3 – Adding the Code

I’ve simplified this exercise due to time constraints. Instead of writing, studying, and understanding all of the code required to build our place door command we will import the module and class module for this application. And, once we’ve done that we’ll modify the code to make it a little more user friendly.

Step 1 – Importing modules As you begin to develop multiple VBA projects you will find that some code is the same.

It would be great if you group and save common code into a file so you can use it in a future project. Exporting is the mechanism in VBA used to save modules to text files and Importing allows you to retrieve and include these files into other projects.

By the way, get in the habit of saving and compiling often. And, don't leave your VBA projects open over night. If there is a power failure you could lose work from the previous day. Just because you have saved the VBA doesn't mean that the changes have been saved to the MVBA file.

So, lets import two files – modDD.bas and classDD.cls. The .bas file is a module and the .cls file is a class module.

To simplify:

A module contains visual basic code fragments, subroutines and functions used to make your program run – in general this is where you would put subroutines used to launch your MicroStation VBA.

A class module is used to create your own objects. Remember objects are things like lines, and design files. So in our example we are creating a door.

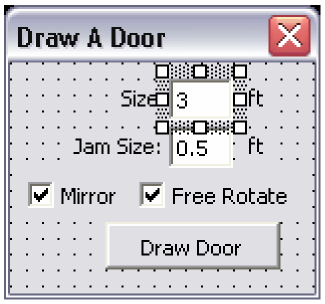
But, don't worry about the different module types at this time. Just remember that for most projects there will be modules and two types of class modules – one used to make primitive commands (like placing lines, circles, boxes, and text), and one used to create locate commands (like delete, move, and copy).

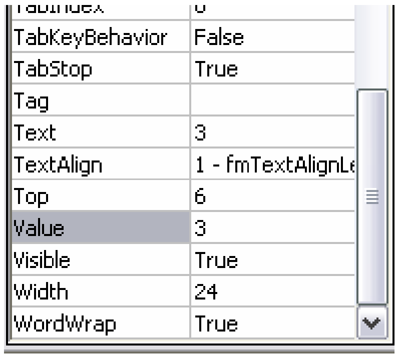
Step 2 – Setting the Default Text Box Value Properties

We don't want our users to put just any value in our text boxes. In fact we want them to place dimensional data, numbers. So, some of our users will need help, because they might accidentally type "8 feet" instead of 8.0.

The first thing we can do to help them out is to provide a default size for the door and jam.

Select the Size text box and then in the properties window scroll down until you find the Value property. Type 3 in this field.





TabKeyBehavior	False
TabStop	True
Tag	
Text	3
TextAlign	1 - fmTextAlignLe
Top	6
Value	3
Visible	True
Width	24
WordWrap	True

Now do the same for the Jam size text box, but type 0.5 for the value.

Step 3 – Controlling Input with Code

In this case we want the text fields to always be numeric. So, as the user is typing we want the application to test the input. Double click on the Size text box.

When you double click on a control for the first time it opens the code window and automatically writes some code for you. In this example VBA assumes that code for text boxes should act when the value of the text box changes. But, you can watch for different events too, like mouse clicks or keystrokes.

In this case we will use the default change event. Shown here…

```
Private Sub txtDoorSize_Change()
End Sub
```

To check for numeric values we will use a VBA procedure called isNumeric. Make your code look like this…

```
Private Sub txtDoorSize_Change()
  If Not IsNumeric(formDD.txtDoorSize.Text) Then
    formDD.txtDoorSize.Text = 3
  End If
End Sub
```

Do the same thing for the Jam Size – can you figure out what the test should be? Hint: The name of the Jam textbox is txtJamSize.

Step 4 – Starting the Draw Routine

We need to hook up the Draw Door command button to our draw door object. When the user clicks the command button the door will place.

Double click on the Draw Door command button. Like the text boxes some code will be written for you. For command buttons the default is a click event. Modify the event so it looks like this…

```
Private Sub drawDD_Click()
  CommandState.StartPrimitive New classDD
End Sub
```

This tells MicroStation to start the draw door routine when the you click the command button.

Step 5 – Testing the Application

In the module called modDD there is a subroutine called drawDoor. It's located at the bottom of the file. Open the module and scroll to the bottom of the code window. The subroutine looks like this…

```
Sub drawDoor()
  Load formDD
  formDD.Show
End Sub
```

All this guy does is load and display the user form we created. It can be activated a couple of ways in MicroStation.

1) Using the project manager double click on the project drawDoor. In the dialog that opens there will be a macro name listed called drawDoor. Double click on it. When you do your user form is displayed.

2) Alternatively, you can key in vba run drawDoor and the same thing will happen. If the project drawDoor is not loaded then this keyin will not work and an error prompt will be displayed. You must first key in vba load drawDoor (the name of the project).

Be careful – if you key in vba load drawDoor more than once the project will be loaded more than once.

3) Instead of using the macro drawDoor, you can use the built in macro onProjectLoad. This macro will be called when a project is loaded. Go ahead and try this. In the module modDD change drawDoor to onProjectLoad. Save and compile. Exit the project.

Now load the project – the user form is displayed automatically.

Exercise 4 – Settings for the IDE, by Jeff Brown of Bentley Systems

This document was originally posted on the Bentley news group. Bentley.microsation.v8.vba.

I'm sure that it is a FAQ by now, but I thought it was important enough to include here…

This is for all of our VBA and Visual Basic 6 developers. (Hereafter, I'll refer to both platforms as VB.) I strongly recommend that you make the following changes to your VB editor options. These recommendations apply to VB development in general, regardless of whether you're programming MicroStation.

The Options dialog can be found under the menu selection, Tools / Options, within your VB editor.

RECOMMENDATION # 1 - REQUIRE VARIABLE DECLARATION *ON*

Require Variable Declaration can be found on the Editor tab of the Options dialog. I recommend that you turn it ON.

When this option is ON, VB will automatically add the statement "Option Explicit" at the top of every newly-created code unit (form, class or module). The "Option Explicit" statement, in turn, tells VB that every variable must be explicitly declared (via "Dim") prior to its use. Otherwise, whenever VB encounters a variable name it doesn't recognize, it spontaneously creates a new variable of that name!

You can imagine how this behavior (without the "Option Explicit" statement) might lead to some difficultto-find bugs. For example:

```
Public Sub ShowAllVertices(oElement As ShapeElement)
  Dim atVertices() As Point3d
  Dim nVertexNumber As Long
  atVertices = oElement.GetVertices
  For nVertexNumber = LBound(atVertices) To UBound(atVertices)

    With atVertices(nVertexNubmer)
      Debug.Print .X, .Y, .Z
    End With
    Next nVertexNumber
End Sub
```

Without "Option Explicit", the preceding code will run just fine. However, it will simply display the values of one item, atVertices(0), over and over again. VB will perceive the misspelled name, "nVertexNubmer", as a new variable, whose value is always 0! WITH "Option Explicit", VB will detect the misspelled name, and report it as an error, before it attempts to execute that statement.

So I recommend that you turn this option, Require Variable Declaration, ON. Please realize that turning this option ON affects only NEWLY-CREATED code units. To apply the benefits of "Option Explicit" to your EXISTING code units (forms, classes or modules), you must manually add the statement "Option Explicit" at the top of each unit. (Each "Option Explicit" statement applies to only the code unit that it appears in.)

RECOMMENDATION # 2 - COMPILE ON DEMAND *OFF*

Compile On Demand can be found on the General tab of the Options dialog. I recommend that you turn it OFF. When this option is OFF, VB performs a "safety check" of your entire project before it executes a single statement. Don't worry; this check is performed so quickly that you won't notice.

If VB detects any compilation problems ANYWHERE in your project, VB alerts you before it attempts to run ANY of your project. Thus, you have the confidence that your ENTIRE project passes the compilation phase (at least). Otherwise (when this option is ON), VB does not check each statement for correctness until it encounters that statement for the first time. This can be a nuisance: In the middle of an extensive test run, your program comes to a screeching halt because of an obvious error that could have been detected by the compiler, and which you can easily fix, but which forces you to start your program over again.

So, I recommend that you turn this option, Compile On Demand, OFF.

RECOMMENDATION # 3 - AUTO SYNTAX CHECK *OFF*

Auto Syntax Check can be found on the Editor tab of the Options dialog. I recommend that you turn it OFF.

This recommendation is not as severe as the preceding ones. Some developers may still prefer it ON, after trying it OFF. When this option is ON (the default), VB checks each statement for syntactical correctness as you're entering it. This sounds good on paper, but I find it to be very annoying.

Whenever I type new lines of code, of course I make typographical errors along the way. When I'm done typing, I proofread what I've entered, and correct the typographical errors. What I don't need is for VB to stand over my shoulder, notice every typographical error almost at the moment I make it, and yell at me about it (i.e, display a dialog that I must dismiss).

So, I recomment that you turn this option, Auto Syntax Check, OFF.

Note that, even when this option is OFF, VB will still highlight in red those lines that have syntax errors. So you still have the benefits of the VB's syntax checking, just not the annoying dialog box.

Lesson 2A: References

A) We already mentioned comp.cad.microstation, but there is a sub news group called comp.cad.microstation.programmer. If you are trying to solve a programming problem the answer is likely here, especially if it relates to MDL, or basic macros. **Update 2017: These communities are no longer active. You may find them, but it's better today to visit Bentley.com Communities.** https://communities.bentley.com/

B) Bentley.microstation.v8.vba – password and login are viecon, and viecon. Works well with Outlook Express. If the answer isn't there a Bentley programmer will get the answer for you. These guys are very responsive and very good at what they do. If they don't have an answer then there probably isn't one.

C) Visual Basic Language Developers Handbook, by Ken Getz and Mike Gibert, SYBEX – available from amazon.com. Excellent examples for doing those programming things we should have learned in college, but didn't because we were going to engineering school not computer science. These guys discuss

Visual Basic related technologies for sorting, encryption, calling Windows API's. This is not a book about Visual Basic's IDE, it's intended for the intermediate to advanced programmer. Buy it even if you are new to this stuff. You'll understand it next year.

D) Bentley's delivered help. I know, Bentley's help files are not very good. But, although you won't find very many examples the help files do follow a pattern, and there is more information in them than you might realize at first glance. The VBA help file for MicroStation is located in MicroStation's root directory (at least it was last month. I think it's supposed to be located in …Bentley\help\) The file name is MicroStationVBA.chm and covers all available objects in MVBA.

E) Take any generic course on VBA, especially if you are new to programming.

F) And, by the way – VBA is not VBA.NET. Be prepared for a new development environment in a couple of years called C# (C-Sharp) and VB(A).net.

G) One more – this is a fun reference (not related to VBA) and although you probably know much of the information in this book there will still be something in here you don't know. The book is called "Poor Leo's 2002 Computer Almanac" by Leo Laporte, Published by QUE, Indiana. If you're not familiar with the Screen Savers on Techtv, call your cable company and find out why not. Mr. Laporte is a co-host of the screen savers – the show focuses on operating systems, computer hardware, and web tips and tricks. Very cool for the geek in all of us.