MicroStation V8 and VBA

by Mark Stefanchuk, Cadgurus.com

The recent release of MicroStation V8 introduces another automation feature, Visual Basic for Applications (VBA). This new implementation of VBA provides rapid development features for designing applications and tools for MicroStation.

Organizations can leverage built in knowledge of Windows programming to improve the CAD workflow and environment. Casual developers, such as CAD managers, can take advantage of visual interface construction tools to build Windows like programs quickly.

Despite these advantages however, CAD managers may still opt to defer development of more sophisticated applications to professional programmers. Although VBA makes automation easier, there are still problems that require programming skill and experience. Compared to the alternatives, VBA is still easier.

MDL, a C programming language has always had an advantage over MicroStation BASIC Macros for a couple of reasons.

- First, MDL creates elements dynamically displaying as they are dragged around the screen.
- And second, macros must be started each time you need them, where as MDL applications are loaded, but run “silent” until a command is requested. However, MDL requires considerable programming expertise, which usually is outside of the average users scope of knowledge.

MicroStation VBA will run until requested, can handle complex dynamics, so building applications in this easy to use programming environment sounds like the way to proceed. But, is ease of programming really the most important aspect of VBA?

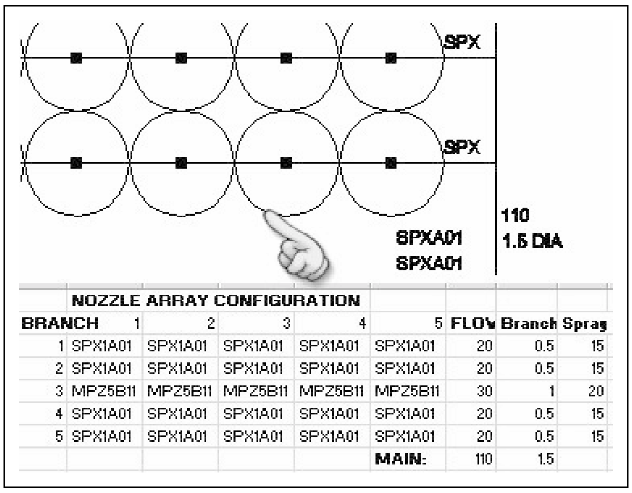
Well, maybe. But, considering that MDL, Macros, and don’t forget JMDL can already handle automation needs do we need VBA?

In a word, yes. VBA has at least one more advantage. It has the ability to read and write to MicroStation from other applications.

For example, a VBA application in Excel can scan the contents of a MicroStation file directly and manipulate the spreadsheet or the design file based on conditions discovered in either file.

A manufacturer of spray nozzles may use a spreadsheet to calculate the flow rate required in a main supplying a nozzle array. The spreadsheet could be arranged such that the total flow rate and the pipe main size are calculated based on nozzle model numbers entered into the spreadsheet.

An Excel VBA could automatically draw the array and label the drawing with size and flow parameters of the array. The spreadsheet user doesn’t need to open the MicroStation drawing or even know how to use MicroStation.



	NOZZLE ARRAY CONFIGURATION							
BRANCH	1	2	3	4	5	FLOW	Branch	Spray
1	SPX1A01	SPX1A01	SPX1A01	SPX1A01	SPX1A01	20	0.5	15
2	SPX1A01	SPX1A01	SPX1A01	SPX1A01	SPX1A01	20	0.5	15
3	MPZ5B11	MPZ5B11	MPZ5B11	MPZ5B11	MPZ5B11	30	1	20
4	SPX1A01	SPX1A01	SPX1A01	SPX1A01	SPX1A01	20	0.5	15
5	SPX1A01	SPX1A01	SPX1A01	SPX1A01	SPX1A01	20	0.5	15
					MAIN:	110	1.5	

Figure 1 *Create Drawing From Excel VBA*

In the next few pages we will examine the mechanics of the MicroStation VBA process. Using a less complex example we will build an Excel VBA to extract cell names from a design file and lists the names in the spreadsheet.

Of course this is a simple example, but it can be extended to create a quantity take off application, or an equipment schedule tool. The possibilities are endless. Imagine linking specification documents to schedules, or comparing vendor drawing data to design data in your MicroStation file.

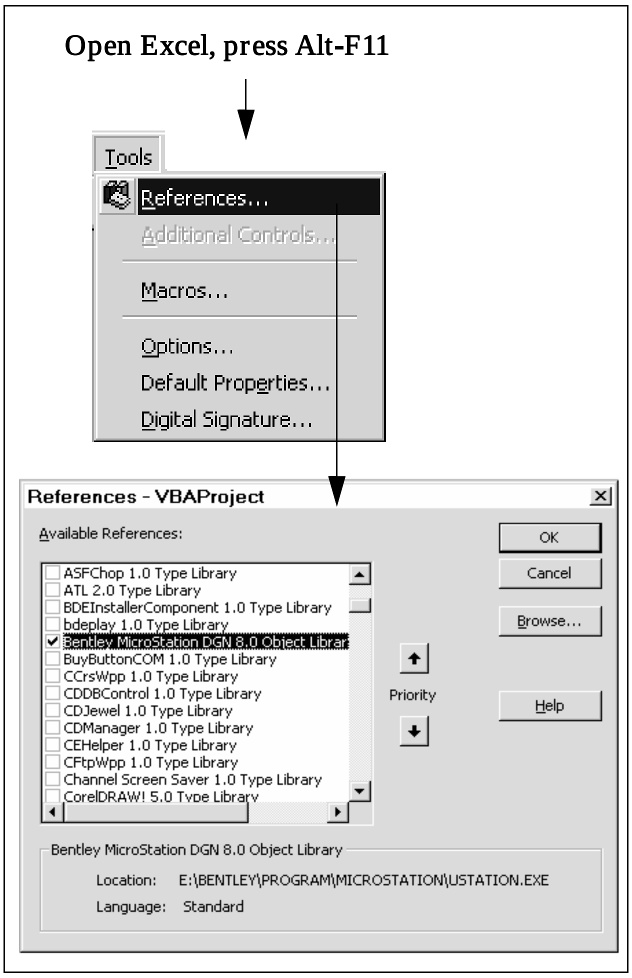
"What? Your vendor drawings are in AutoCAD format not MicroStation?" No problem, AutoCAD implements VBA too, so use MicroStation VBA applications to read data from the AutoCAD files.

Teaching Excel VBA about MicroStation

Before you can reference MicroStation objects in Excel, VBA needs to know that MicroStation exists. We do this by creating a reference to the DGN 8.0 library. Using the following steps:

1. Open the VBA editor in the application you want to reference MicroStation from (Excel – the keyboard shortcut **Alt-F11** will open the editor.)

2 Under the Tools pull down menu, choose References. A dialog appears with a list of references for your computer.



- ASFChop 1.0 Type Library
- ATL 2.0 Type Library
- BDEInstallerComponent 1.0 Type Library
- bdeplay 1.0 Type Library
- ✓ Bentley MicroStation DGN 8.0 Object Librar
- BuyButtonCOM 1.0 Type Library
- CCrsWpp 1.0 Type Library
- CDDBControl 1.0 Type Library
- CDJewel 1.0 Type Library
- CDManager 1.0 Type Library
- CEHelper 1.0 Type Library
- CFtpWpp 1.0 Type Library
- Channel Screen Saver 1.0 Type Library
- CorelDRAW! 5.0 Type Library

Bentley MicroStation DGN 8.0 Object Library
Location: E:\BENTLEY\PROGRAM\MICROSTATION\USTATION.EXE
Language: Standard

Figure 2 Object Library with Small Box Checked

3 Scroll through this list to find "Bentley MicroStation DGN 8.0 Object Library"

4 Check the small box next to Bentley MicroStation DGN 8.0 Object Library and then click **Ok** on the References dialog.

Now that there is an object Library we no longer need to create an OLE application object.

Some of you may recall that using createObject in VB 6 to edit MicroStation. msApp = createObject("MicroStation.Application")

Creating an Interface

- Add a form to the VBA by choosing the Pull Down Menu "Insert – UserForm".

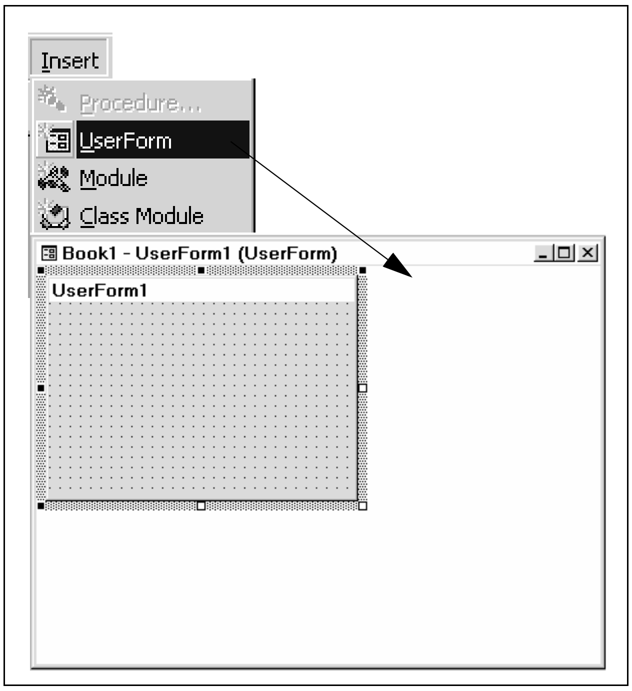


Figure 3 Tools – UserForm Pull Down Menu

- Place a Button on the Form and change the Caption to "Get Cell Names".

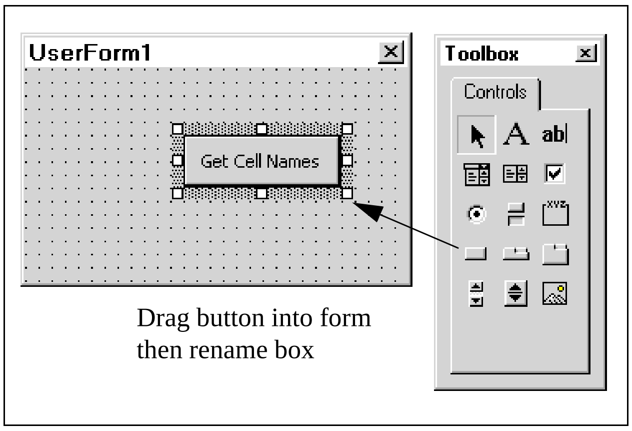


Figure 4 Add Button "Get Cell Names" to UserForm1

Tips

Converting Raster to Vector

Can I convert a raster file to a vector file in microstation J?

You need the module MicroStation Descartes or IRAS/B to the job.

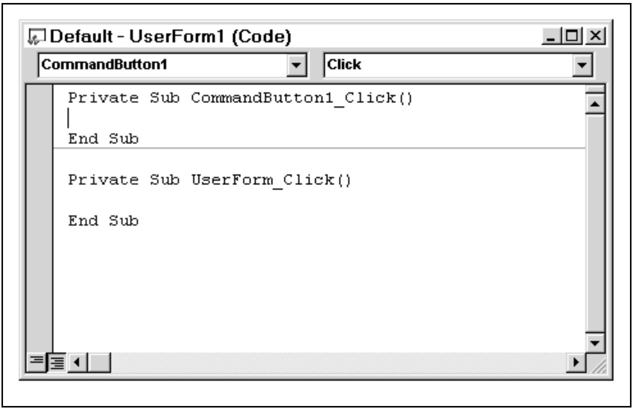
CAD

Building the Command

- Double click on the “Get Cell Names” button to open the source code window (see Figure 5).

A template sub routine is automatically created. It will look like this…

```
Private Sub CommandButton1_Click()
End Sub
```



```
Private Sub CommandButton1_Click()
|
End Sub

Private Sub UserForm_Click()

End Sub
```

Figure 5 *Source Code Window*

Inside this subroutine we will place the script that opens a file, scans the cells in the file, reads the cell names and then places the names into the spreadsheet file. Here’s what it will look like.

```
 Private Sub CommandButton1_Click()
 Dim oCell As CellElement
 Dim oScanCriteria As ElementScanCriteria
 Dim dFile As DesignFile
 Dim curCell As Object
 'OPEN THE DESIGN FILE
 Set dFile = OpenDesignFile
                         ("c:\temp\test3d.dgn", True)
 Set oScanCriteria = New ElementScanCriteria
' SET THE SCAN CRITERIA
oScanCriteria.ExcludeAllTypes
oScanCriteria.IncludeType msdElementTypeCellHeader
Dim oScanEnumerator As ElementEnumerator
Set oScanEnumerator =
          ActiveModelReference.Scan(oScanCriteria)
i = 1
Do While oScanEnumerator.MoveNext
Set oCell = oScanEnumerator.Current
Set curCell = Worksheets(1).Cells(i, 1)
curCell.Value = oCell.Name
i = i + 1
Loop
dFile.Close
End Sub
```

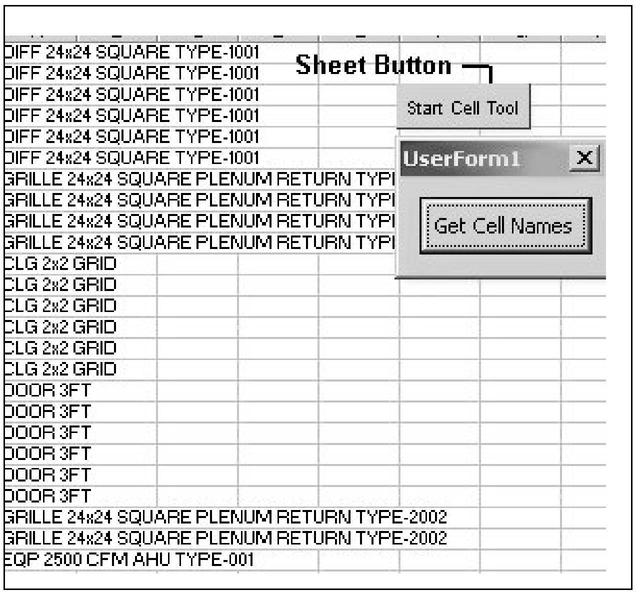
After opening the design file, a scan criteria is set. Think of this as a filter. In this case we only want to look for cells. The scan enumerator can be thought of as a list of elements that the search finds, which is executed by the `ActiveModelReference.Scan` procedure.

Once the elements are loaded into the scan enumerator (list). Each element can be processed. In this example we use a “do while” condition to step through each cell found.

The code that populates the spreadsheet is contained inside the “do while” and is just two lines which sets the value of the spreadsheet cell (careful, it’s just a row-column location in the spreadsheet, and not the same as a MicroStation cell.) to the name of the MicroStation cell found by the scanner.

Running the Application

A Workbook sheets can also contain buttons. In Figure 6, the spreadsheet shows a button labeled “Get Cell Tool”.



```
DIFF 24x24 SQUARE TYPE-1001
DIFF 24x24 SQUARE TYPE-1001
DIFF 24x24 SQUARE TYPE-1001
DIFF 24x24 SQUARE TYPE-1001
DIFF 24x24 SQUARE TYPE-1001
DIFF 24x24 SQUARE TYPE-1001
GRILLE 24x24 SQUARE PLENUM RETURN TYPI
GRILLE 24x24 SQUARE PLENUM RETURN TYPI
GRILLE 24x24 SQUARE PLENUM RETURN TYPI
GRILLE 24x24 SQUARE PLENUM RETURN TYPI
CLG 2x2 GRID
CLG 2x2 GRID
CLG 2x2 GRID
CLG 2x2 GRID
CLG 2x2 GRID
CLG 2x2 GRID
DOOR 3FT
DOOR 3FT
DOOR 3FT
DOOR 3FT
DOOR 3FT
DOOR 3FT
GRILLE 24x24 SQUARE PLENUM RETURN TYPE-2002
GRILLE 24x24 SQUARE PLENUM RETURN TYPE-2002
EQP 2500 CFM AHU TYPE-001
```

Figure 6 *Sheet containing button*

Tips

Cpwattr.bas

Is there a program that will excute the "copy parallel by distance" command, then change that newly copied element to the active level & symbology, all at once?

For a Copy Parallel With Active Attributes.program, cpwattr.bas is the BASIC macro.

Check out
www.cadgurus.com/learn
/freedwnl/cpwattr.asp **CAD**

Quickie Tip

What is a SF.DAT file in MicroStation?

It is an Bentley internal system file related to licensing. Users need not to know about it.

CAD

This button also has a code subroutine. The subroutine opens the UserForm1. The code to open the form looks like this…

```
Private Sub CommandButton1_Click()
Load UserForm1
UserForm1.Show
    End Sub
```

To access place buttons on sheets and access the code window for these buttons, click on "Design Mode". This is the triangle icon located on the Controls Toolbox (*View> Toolbars*).

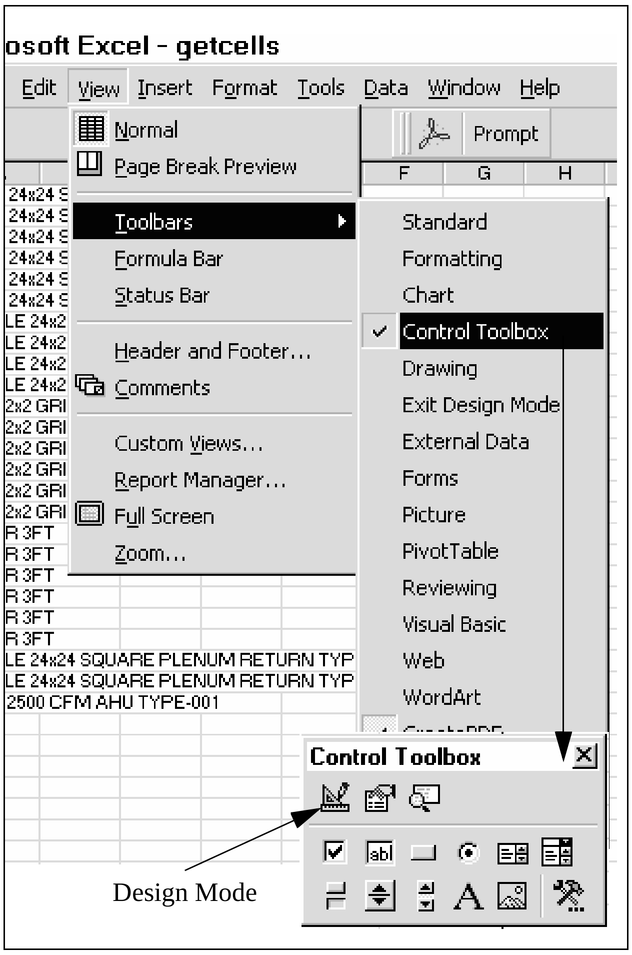


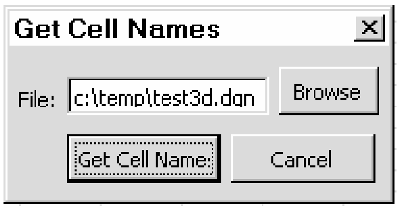
Figure 7 *Design Mode Toggle on Control Tools Toolbox*

Once the button has been placed and you add the code to the button, "Exit Design Mode" by clicking on the design mode button in the Control Tools toolbox again.

- Now run the application by clicking on the sheet button. UserForm1 will display.
- Click the Get Cell Names button. Column 1 is populated with cell names.

Final Comments

In designing `UserForm1` it would have been appropriate to include a text box to prompt the user for the location of the design file. Further, a browse button helps the user search for a file. Do you think you could add these to your form? Try it. The completed code is available from **cadgurus.com**.



File: c:\temp\test3d.dgn

Figure 8 *Form with File Text Box, Browse ,Cancel Button*

VBA opens up new tool kit for office automation. Accessing MicroStation objects in this way can lead to many new applications.

And, although you may need to consult a programmer for more sophisticated applications, you will have no problem creating user- friendly applications that take care of every day problems. You know, like extracting cell names from a design file.

About The Author

Mark Stefanchuk is a partner with Ramsey Systems, Inc., the developers of cadgurus.com. Mark can be contacted by email on mark@cadgurus.com.

Please email Mark with any feedback or suggestions for future articles.

CAD

cadgurus.com
by Ramsey Systems
INCORPORATED
MicroStation Training --- Software Development :: MDL MACROS VB --- Your CAD Gurus