

Feature

Easy Scan and Easy Open

by Mark Stefanchuk

Has anyone else noticed that VBA doesn't necessarily mean easy? Granted it is a much faster programming tool than MDL, and absolutely a more complete programming tool than MicroStation Macros, but easy? MicroStation VBA is a complete development environment, and like its counter parts in Word, or Excel it provides automation tools for almost every feature within its parent program.

Completeness in this case translates into a complex integrated development system, in other words hard. But, fear not. Here, for your use and modification are two easy to implement but very useful VBA applications.

Easy Scanner

Probably the most powerful tool in design automation is the ability to perform the same operation on multiple elements. Here's an example.

```
Option Explicit

Public Sub EasyScan()
    ' think of enumerators as a bag full of elements
    Dim oScanEnumerator As ElementEnumerator

    ' declare a variable to hold the element
    Dim oElement As Element

    ' handle errors
    On Error GoTo EasyScanError

    ' get all elements in the active model - fill bag
    Set oScanEnumerator = ActiveModelReference.Scan

    ' examine each element one at a time - from bag
    Do While oScanEnumerator.MoveNext

        ' make oElement the current element
        Set oElement = oScanEnumerator.Current

        ' since we scanned all elements make sure
        ' we only act on graphic elements.
        If oElement.IsGraphical Then

            ' test for snap = false
            If Not oElement.IsSnappable Then

                ' set snap to true
                oElement.IsSnappable = True

                ' when you change an element
                ' you must rewrite it.
                oElement.Rewrite
            End If
        End If
    End While
End Sub
```

```

Loop

' stop the VBA macro and invoke the default command
' the default command is usually (it is a preference)
' the selection tool
CommandState.StartDefaultCommand

' tell the user we are done
ShowStatus "Finished!"

' clean up and exit
Set oElement = Nothing
Set oScanEnumerator = Nothing
Exit Sub

```

```

EasyScanError:
    Debug.Print "Something happened in EasyScan"
    MsgBox "Unable to process elements"
End Sub

```

EasyScan is called a slow scanner because every element in the file is examined. In fact this slow scanner doesn't use any of MicroStation's scanning tools. The scanning tools filter elements you do not want to consider before looping through the elements. This implies a smaller list, thus a faster scan. But EasyScan is still faster than changing one element at a time.

The EasyScan sub filters elements using "If-Then" conditional tests. For example,

```
If oElement.IsGraphical Then
```

By nesting tests you can modify EasyScan to only modify lines. For example,

```

' we only act on graphic elements.
If oElement.IsGraphical Then
    ' test for line element
    If oElement.IsLineElement Then
        ...
    End If
End If

```

A Note Regarding Error Handling

EasyScan implements code to manage errors. Normally, you won't see this code in examples. It tends to complicate the discussion. But it is very important. Here are the lines I am referring to.

```

' handle errors
    On Error GoTo EasyScanError

...

EasyScanError:
    Debug.Print "Something happened in EasyScan"

```

The on Error GoTo EasyScanError tells the sub what to do if an error is encountered. An error might be an unreadable element, or a file input/output operation that fails. Really it could be anything.

If removed and the program generates an error VBA stops execution and exits the program. And usually VBA presents your user with a very cryptic message. Try this. Using EasyScan comment out the error handling lines we just looked at and the following if statement and its associated end if.

```

'If oElement.IsGraphical Then
...
'End If

```

Now run the program. In this case EasyScan is making a graphics modification but it is attempting to make that change on non-graphic elements. An error results.

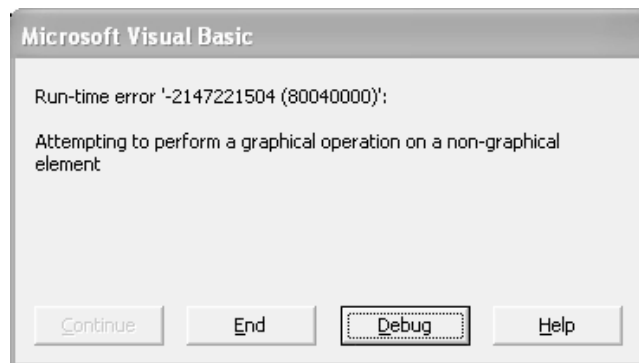


Figure 1 VBA Debug Error

And if you lock the application for viewing, as you should when deploying to others then the user will get a non-debug version of the error.

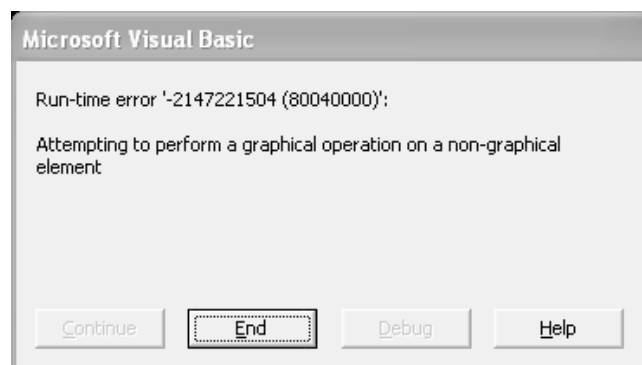


Figure 2 VBA Message Without Debug

Clicking help will tell your user about automation error 440. Still not very useful. So, what happens then if the error handler is uncommented? Well, in this example not much, but you don't get this poorly formed error message, the program doesn't exit, and you have control over the error messaging. Meaning you can tell the user whatever you want when something doesn't work.

For example, you might send a message to MicroStation's prompt field.

```
ShowStatus "Unable to process elements."
```

Or you may choose to display your own message box.

```
MsgBox "Unable to process elements"
```

Easy Open

There are excellent tools in MicroStation v8 to help you batch process your design files. Batch process simply refers to running the same process on more than one design file, like batch plotting. Sometimes however, it is necessary for your applications to manage batch processing on its own.

Here is an example of a VBA macro for v8 that opens design files in a directory, fits view 1, saves settings, and then opens the next file. It is simple in that it only considers files in one directory. But like our EasyScan macro, it gets the job done.

For this example I have created a small user form. It's primary purpose is to give the user a way to specify the folder of design files.

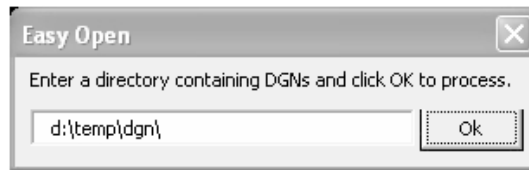


Figure 3: Easy Open User Form

In the code view of the form is the following Private sub routine. This sub is called when the **Ok** button is pressed.

```
Private Sub EasyOpen()  
    ' declare files system variables  
    Dim fso As Object, f As Object, fc As Object, f1 As Object  
  
    ' trap all errors  
    On Error GoTo EasyOpenError  
  
    ' get file system object  
    Set fso = CreateObject("Scripting.FileSystemObject")  
  
    ' get the folder defined by user  
    Set f = fso.getFolder(Me.TextBox1.Text)  
  
    ' get a list of files in this folder  
    Set fc = f.files  
  
    ' open each file in the list  
    For Each f1 In fc  
        ' make sure that the file is a dgn - simple check  
        If InStr(1, f1.Name, ".dgn", vbTextCompare) > 0 Then  
            ' open the file  
            OpenDesignFile Me.TextBox1.Text + f1.Name  
  
            ' fit view 1  
            CadInputQueue.SendKeyin "fit all;selview 1"  
  
            ' save settings  
            SaveSettings  
        End If  
    Next  
  
    ' tell the user we are done  
    ShowStatus "Finished!"  
  
    ' clean up and exit sub  
    Set fso = Nothing  
    Set f = Nothing  
    Set fc = Nothing  
    Set f1 = Nothing  
    Exit Sub  
  
EasyOpenError:  
    Debug.Print "Error running EasyOpen"  
    MsgBox "Error running EasyOpen"  
End Sub
```

This routine uses Windows file system object to find the folder specified by the user (in the form's text box) and all of the files within that folder.

For each file three MicroStation commands are executed. The first opens the file.

```
OpenDesignFile Me.TextBox1.Text + f1.Name
```

This command opens the file name described by the strings Me . TextBox1 . Text (the folder) plus f1 . Name. I must point out that no validation is performed on the user input or on the string. As a minimum the input in the field should make sure that a final backslash is included in the location of the folder.

For example d:\temp\dgn will cause an error because the combined filename would look something like d:\temp\dgndesignfile.dgn. In this case OpenDesignFile can't find the file and will cause an error. The easiest check to make would be to test that the file exists using the following.

```
If fso.fileExists(Me.TextBox1.Text + f1.Name) Then
```

Put this test just before OpenDesignFile and end it after SaveSettings. Use an Else to provide a prompt that tells the user the filename was incorrect. Be careful, don't use a message box. If you have several hundred files to process you will be clicking **OK** several hundred times. In this case ShowStatus is better practice.

Enhancement - A more complex VBA

EasyScan modifies many elements in a single file while the EasyOpen performs a single operation on many files. Now, what if you added the following line after OpenDesignFile in EasyOpen?

```
' run easy scan  
EasyScan
```

Get it? Now the application applies an operation to multiple elements in multiple files. A very "easy" change makes this a much more sophisticated VBA macro. Changing EasyScan to add filters, or to modify colors, or levels are also easy changes that make the macro applicable to new problems or tasks.

About The Author

Mark Stefanчук is a consultant and contract programmer specializing in business and design workflow automation. He was co-founder and principle architect of cadgurus.com a web community for MicroStation professionals.

More recently Mr. Stefanчук continues to work on projects involving MDL, VBA, and Microsoft Windows technology. Mark may be contacted by sending email to markastefanchuk@yahoo.com.

CAD

Tips

Toggle between a white background and a black background

We regularly need to toggle between a white background and a black background to view external consultant files (we use a black background for most of our work). Is there a key-in I can assign to a function key which will toggle the background from white to black and vice versa?

You can do this on a view-by-view basis. For view 1, you can use

```
mdl keyin calculat calculator  
userPrefsP->extFlags.invertBackground=!userPrefsP->extFlags.invertBackground  
;view off 1;view on 1
```

to toggle the background colour from black to white to black, etc.

CAD

TIPS IN HERE