

Feature

VBA - Batch Processor

by Mark Stefanchuk, Cadgurus.com

I just wrote a quick MicroStation VBA application to handle turning off levels in a specific reference file. The application itself was not complicated but it demonstrates a couple of very powerful features of VBA. First, I noticed that with a little background in CAD Management and in VBA you can automate some very tedious processes. And the second, you can do it in a very short period of time.

I'd like to share that process with you in the next few pages. My hope is that this demonstration can give any MicroStation user, or at least you power users some tools to take your CAD Management skills to the next level. I believe too that any MicroStation user can follow this example, even towards the end where where I show you how to read user input.

Now, turning off a level in one reference file is easy. All trained MicroStation users know how to do that. But let's say you have 300 files. Now what? Are you going to open each one individually and change the level for the same reference file in each design file? I don't think so. Although I must admit to having done similar things to a project folder full of design files many years ago. Eventually I learned of MSBatch and running batch processes from a command line.

With V8, Bentley has created an interface for batch processing files. If you know the key-in that's great. But if your processing is a little more complicated, you can insert a VBA macro into the batch process.

Here's how you might use the batch processor (*Utilities>Batch Process*) to change the color of the text in all selected design files.

Free Program

Getrefnames.bas

The `Getrefnames.bas` program gets reference filenames and save as configuration variable. Download from:

<http://www.h-stegeman.demon.nl/index.html>

CAD

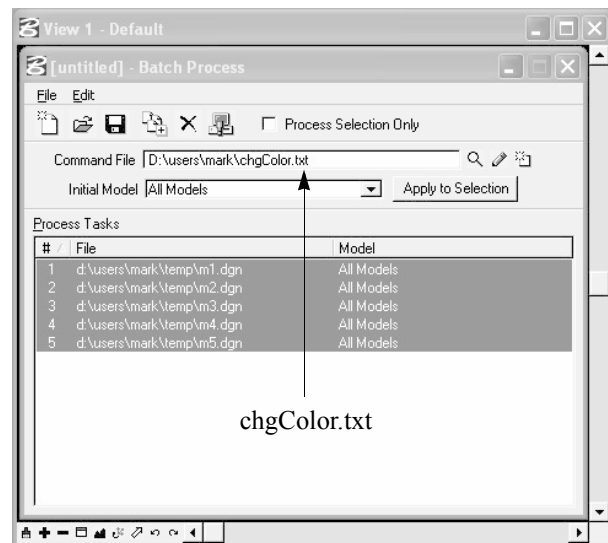


Figure 1 Utilities>Batch Process

The text file `chgColor.txt` contains a single line "VBA run `chgColor`". The assumption here is that a MicroStation VBA application is loaded and contains a macro (a sub routine) called `chgColor`. In this case I borrowed the example from the Bentley news group posted by Gino Cortesi and modified it to change text elements instead of arcs.

```
Public Sub chgColor()  
    Dim ee As ElementEnumerator, el As Element  
    Set ee = ActiveModelReference.Scan  
    Do While ee.MoveNext  
        Set el = ee.Current  
        If el.IsGraphical Then  
            If el.IsTextElement Then  
                el.Color = 1  
                el.Rewrite  
                el.Redraw  
            End If  
        End If  
    Loop  
End Sub
```

All this sub routine does is find every element in the file,

```
Set ee = ActiveModelReference.Scan
```

It then steps through each element that it found,

```
Do While ee.MoveNext
```

Set `el=ee.Current` just sets an element object to the current element. We test to make sure that the elements we found were the ones we are looking for. Since we didn't filter the scan (perhaps the subject of a future article) we find all MicroStation elements including non-graphic elements so that's why this example uses the `IsGraphic` property.

Finally the element is updated,

```
el.Color = 1
el.Rewrite
el.Redraw
```

If you don't rewrite the element then the changes will be discarded when the file is closed. Redraw updates the element display without updating the entire view.

But, this text file can contain any key-in. So if you are better at MicroStation Macro programming then use it. Or, if your change is as simple as fitting a view and saving the settings then your text file could contain a key-in like,

```
Fit all;selview 1;file
```

Of course we want to do some things that are more complicated. Like turn *Off* the level in a single reference file. And we want to apply this change to multiple files. Here's the VBA macro I wrote earlier today. See **Code 1 below**.

As you can see even this macro is quite simple. But a tool that took only a few minutes to write will save many hours. Use MicroStation batch processor and a command file with the following key-in

```
VBA run levelOff
```

`ActiveModelReference.Attachments` represents all of the attached reference files in the design file. The `For Each` statement is used to step through each reference file. The file name is compared to the file we want to process, and then the level property `IsDisplayed` is turned off. To turn on levels create another macro called `levelOn`. Use the same code contained in `levelOff`, but change the `IsDisplayed` property to `True`.

The batch processor is quite powerful and allows the user to select multiple files from more than one directory. Further, batch processor lets you save combinations of command files and drawing lists. The procedures are well documented and for power users, quite intuitive and worth studying on your own.

As a rule software developers hate building scripts where data that may change, like a file name or level name is included in the program source code. In the `levelOff` example, I included the line, `If att.AttachName = "02046TTL.dgn"`. The file name really should be variable. One way to accomplish this is to define the variable inside the macro.

Something like that shown in **code 2 below**.

Here I have assigned the variable filename with the value "02046TTL.dgn". In a more complicated macro, where filename is used more than once, I only have to go to one place to change it. This approach doesn't give any more control to the end user. If the end user is the developer then assigning filename in this way is probably good enough. More likely however, your end user will want to be able to indicate the filename at least from command line.

A quick way to accomplish this is to use the built in property `KeyinArguments`. When used with the VBA `RUN <macro name>`, `KeyinArguments` holds the string that follows `<macro name>`.

Code 1

```
Public Sub levelOff()
    Dim att As Attachment
    For Each att In ActiveModelReference.Attachments
        If strcomp(att.AttachName, "02046TTL.dgn", vbTextCompare)=0 Then
            att.Levels("g-mech-seal").IsDisplayed=False
            att.Rewrite
        End If
    Next att
End Sub
```

Code 2

```
Public Sub levelOff()
    Dim att As Attachment, filename as String
    filename = "02046TTL.dgn"
    For Each att In ActiveModelReference.Attachments
        If strcomp (att.AttachName, filename, vbTextCompare) = 0 Then
            att.Levels("g-mech-seal").IsDisplayed = False
            att.Rewrite
        End If
    Next att
End Sub
```

For example, VBA RUN levelOff 02046TTL.dgn will set KeyinArguments to 02046TTL.dgn. Using KeyinArguments, the *levelOff* macro is as shown in **Code 3 below**.

Notice too that I've added a test for the length of the filename to make sure that a name was entered. And if the name was not entered I used MicroStation's ShowMessage method to tell me what happened. ShowMessage is added to MicroStation's Message Center so I can look for errors after running batch processor.

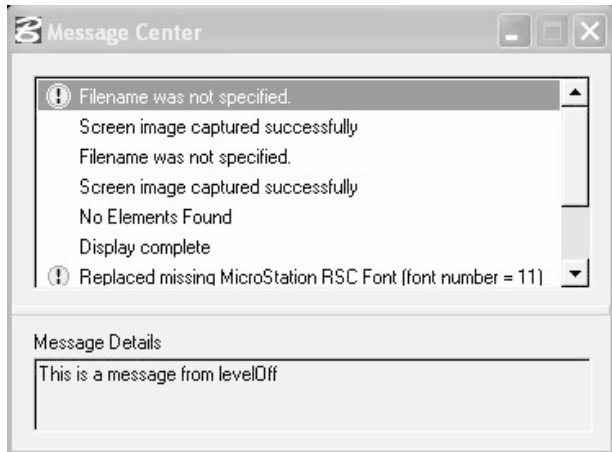


Figure 2 Message Center

Message center can be displayed by clicking on the exclamation mark (box may be empty if the message is only informational and not a warning) positioned bottom center of the MicroStation window.

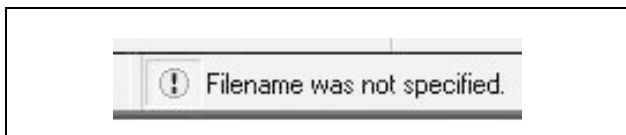


Figure 3

There is at least one more enhancement you may want to consider. Just like the file name the level name is included in the macro source. Instead of using att.Levels("g-mech-seal").IsDisplayed=False we want to use att.Levels(levelname).IsDisplayed=False.

To accomplish this quickly, we will impose a command line syntax that looks like,

VBA RUN <macro name> <filename>,<levelname>

For example, using our *levelOff* command we would have,

VBA RUN levelOff 02046TTL.dgn,g-mech-seal

To get our macro to recognize that there are two inputs we can use the VBA "split" method. See **Code 4 on page 29**.

Tips

AccuSnap and Fence

Is there a reason that the fence tool doesn't work in conjunction with AccuSnap? Is there a switch or checkbox somewhere?

The default behavior for AccuSnap when placing a fence is *Off*. You can, however, press Ctrl+Shift to *enable* AccuSnap while placing a fence. Note that you can do the same thing for any tool that doesn't use AccuSnap by default).

CAD

The Home of MicroStation Books and Magazine

<http://www.penbrush.com>

Code 3

```
Public Sub levelOff()
    Dim att As Attachment, filename as String
    filename = KeyinArguments
    If len(filename) > 0 then
        For Each att In ActiveModelReference.Attachments
            If strcomp (att.AttachName, filename, vbTextCompare) = 0 Then
                att.Levels("g-mech-seal").IsDisplayed = False
                att.Rewrite
            End If
        Next att
    Else
        ShowMessage "Filename was not specified.", _
            "This is a message from levelOff", _
            msdMessageCenterPriorityWarning
    End If
End Sub
```

Code 4

```
Public Sub levelOff()  
    Dim att As Attachment  
    Dim filename As String  
    Dim levelname As String  
    Dim tmpStr() As String  
  
    On Error Resume Next ' continue when user doesn't enter arguments  
    tmpStr = Split(KeyinArguments, ",", 2, vbTextCompare)  
    filename = tmpStr(0)  
    levelname = tmpStr(1)  
    If Len(filename) > 0 And Len(levelname) > 0 Then  
        For Each att In ActiveModelReference.Attachments  
            If strcmp (att.AttachName, filename, vbTextCompare) = 0 Then  
                att.Levels(levelname).IsDisplayed = False  
                att.Rewrite  
            End If  
        Next att  
    Else  
        ShowMessage "Arguments not specified.", _  
            "This is a message from levelOff", _  
            msdMessageCenterPriorityWarning  
    End If  
End Sub
```

The only significant changes were to add the `tmpStr()` variable and the `levelname` variable. `tmpStr` is defined by the `split` method which expects `KeyinArguments` to have two values separated by a comma. In this case we have assume that `tmpStr(0)` contains the filename, and `tmpStr(1)` contains the level name. We've only tested for the existence of these strings and haven't included a mechanism for testing whether the file name or the level name are valid entries.

Without overcomplicating the macro, the best test would be to insert a *True/False* flag into the `strcmp` test for the filename. Initialize the flag to *False* before entering for each loop. If you find a match set the flag to *True*. When you exit the for each loop test the flag.

If *False* send a message indicating filename doesn't exist. This doesn't mean that you entered the arguments incorrectly. It could also simply indicate that the file you have processed doesn't contain the reference file specified. But, it is a simple way to alert you that there may have been a problem.

Of course I'll leave you with some homework. Read up on batch processor and how to create a command file. And, then see if you can update the macro to show the filename not found message.

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

If you have MicroStation tips,
please email details to
penbrush@ozemail.com.au



MicroStation Training --- Software Development :: MDL MACROS VB --- Your CAD Gurus