

A Plot Accounting Macro – Q4 2000 C-A-D

By Mark Stefanchuk

The advantage of tracking plot information for billing purposes is obvious, you know how much money to charge your client. Even if you decide not to charge your clients, you may still want to track your plot output. Do you know who creates the largest number of plots? Which plotters are being used? Do you really need to pay maintenance on ten plotters, or can eight carry the plot load? By tracking plot information you will always know and you will begin to save money.

Maybe plot accounting does not have to be difficult. Maybe plot accounting does not have to be sophisticated nor expensive. Unfortunately, expensive software is the only way to get plotting accountability. Or is it?

Capturing data such as project number, user name, sheet size, plotter name, plot date, and filename, can be accomplished using a basic macro. We just need to figure out where to get the data. Here are some strategies for finding and saving plot data.

Project Number

The project number determines which client is charged for the plotting service. There are a couple of ways to grab the project number. The first method uses the directory path and requires that your project directory names contain the project number.

For example, a project started in 2000 might use a numbering scheme as simple as 00472000 to indicate that this is the 47th project started in 2000. The directory structure may then be defined as p:\projects\00472000\. Our macro can save the directory as a string, and strip the directory name. Using the macro string property MbeDgnInfo.dgnFileName yields the path and file name. Next, use VB string manipulation tools to extract the project number from the path. The macro will look something like this.

```
Sub main ()
    Dim fileInfo As String, rt As String, lt As String
    Dim spot As String

    fileInfo = MbeDgnInfo.dgnFileName
    spot = getLastSlash (fileInfo)
    lt = left$ (fileInfo, spot - 1 )
    spot = getLastSlash (lt)
    rt = right$(lt, len(lt) - spot)
    print rt
End Sub
```

GetLastSlash is a function created using the Visual Basic statement inStr. The function steps through the string until the last back slash is found and then returns the position of

the slash. MicroStation's basic macro language does not have an inStrRev statement so finding the last occurrence of a character requires a little more code. Feel free to download the source code from cadgurus.com to study the function.

When the project number is not contained in the directory path use a project text file. Simply create a new text file and add the project number to the file, 00472000. Save the file with the name project.txt. To read the text file just use the Visual Basic Open statement to access the project number as shown in the following code fragment.

```
Sub main ()
    Dim fileInfo As String, projnum As String, thedirectory As String
    Dim spot As String

    ' get the project directory location
    fileInfo = MbeDgnInfo.dgnFileName
    spot = getLastSlash (fileInfo)
    thedirectory = left$ (fileInfo, spot )

    ' now append project.txt to thedirectory

    open thedirectory + "project.txt" for input as #1
        input #1, projnum
    close #1
    print projnum
End sub ()
```

User Name

The purpose of the user name is to tell us who submitted the plot. The variable for the current user name is USERNAME. The value is extracted easily with the basic statement Environ\$.

```
Sub main ()
    Dim whoplots as String

    whoplots = Environ$ ("USERNAME")
    Print whoplots
End sub
```

Sheet Size

Obtaining the sheet size may appear tricky at first, but using MicroStation's C expression strings, retrieving plot information is quite easy. Upon examining the Print/Plot dialog it

becomes apparent that MicroStation contains all the plot information needed by our accounting macro.

<< insert image01.jpg here >>

“But wait, I heard something about C, and I don’t know anything about C...” you cry. That’s okay, because we don’t really need to know how or why it works, or that we will be extracting information using a C structure. All we need to know is the syntax, and a little about where the information comes from, a road map of sorts. The process does get quite involved, especially when you write it down on paper, but with some digging and study a pattern will emerge. So, an example:

```
sizeX = MbeCExpressionString("plotStat->plotSpec.size.x")
sizeY = MbeCExpressionString("plotStat->plotSpec.size.y")
```

sizeX and sizeY are just string variables which means that the function on the right hand side of the equality must return a string.

Regarding the string passed to the function, “where did plotStat->plotSpec.size.x come from?” An easy answer is to say that it came from a file located in MicroStation’s mdl\include directory called pltstrc.h. The file is a text file defining all of the variables that MicroStation uses when plotting. This file is our road map.

With any map we need to know what we are looking for, a road or city. In the above example our destination is paper size. Searching pltstrc.h for the word “size” will yield twenty-seven variations of the word size. Twenty of these occurrences are contained in comments indicated by the “/* */” combination. Of the remaining seven, six are variables that contain the word size as part of a longer variable name and the last is simply the word size. Upon closer examination the comment explains that this variable is, “size of plot in plot units”. That sounds like the variable we are looking for, so let’s try it. First, we need to know a little more about the pltstrc.h file.

The pltstrc.h file is like a big variable declaration file. The declarations have a similar function to the Dim statement in basic. If you scroll through the file you will see many examples of declarations and several that you might recognize. “Int replots” declares replots as an integer, and “double scale” declares scale as a double precision floating point. But the variable size is declared as Dpoint2d. Well, Dpoint2d happens to be a MicroStation defined variable and in this case is another structure. So, the pltstrc.h file is made up of structures and the structures are made up of other structures and simpler variables like integers (int).

The Dpoint2d structure is defined in a different file, but is simple to figure out. In 2d there is an x and y coordinate, so Dpoint2d likely contain values which define the size of the plot in the x and the y directions. That means size has more than one value, an x and y value. To tell MicroStation which one you want, use a dot between size and the variable, as in size.x and size.y. Further, MicroStation conveniently defines a variable plotStat

where is stores the plotting information. Size is actually part of a structure called Plot_spec and is referenced by yet another structure called PlotStatic as plotSpec.

To simplify, here's a rule. To obtain plot information from a variable contained in the structure Plot_spec then use the string "plotStat->plotSpec.<variable you want goes here>" or if you want a variable from the structure Pltr_cfg then the string becomes "plotStat->plotter.<variable you want goes here>".

Using the plot structure file pltstrc.h is an involved process. Try not to get overwhelmed by the details. News groups are a great resource for ideas and sample code fragments. Don't forget to check comp.cad.microstation.programmer the next time you get stumped.

Plotter Name

Here too the best technique for obtaining the plotter name is to use the plot structure. To get the name of the plotter used by Print/Plot first get the name of the "plt" file and use the file name to indicate the plotter. A more sophisticated solution would have the macro read the plt file and extract the value of "default_outFile/auto_overwrite ". Extracting default_outFile/auto_overwrite the macro means that the plt file can have any name. But, to keep our macro simple we will just use the plt filename.

```
Sub main ()
```

```
    Dim theplotter as String, plotdevice as String
```

```
    theplotter = MbeCExpressionString("plotStat->plotSpec.config_name")
```

```
    plotdevice = FileParse$( theplotter,3)
```

```
    print plotdevice
```

```
End sub
```

Plot Date

Extracting the plot date is just a matter of using the basic Date\$() function. The date function returns the current system date as a ten-character string. The format is MM-DD-YYYY.

```
Sub main ()
```

```
    CurrentDate as String
```

```
    CurrentDate = Date$()
```

```
    Print CurrentDate
```

```
End sub
```

Filename

The filename is derived from the current MicroStation file. Use the design file information object again, MbeDgnInfo.dgnFileName. To illustrate,

```
Sub main ()
    Dim designFile as String

    designFile = MbeDgnInfo.dgnFileName

    Print designFile
End sub
```

Bringing It All Together

Our discussion looked at saving six pieces of plot information. In all of the example code fragments the information was sent to a text window. We really need to save the data. A quick solution is to append the data to a text file using the basic Open and Close statements. Here is a simple procedure for writing the plot data to a text file.

```
Sub writePlotInfo (outputLine as String)

    Open projectTxtFile for Output As #1
        Print #1, outputLine
    Close #1

End sub
```

The text file name, projectTxtFile is a string containing the path name of the output file. Refer to any Visual Basic text regarding ini files. Or simply use the quick procedure outlined in the “Project Number” section. Adding up all of the plot information strings creates outputLine, the string sent to writePlotInfo.

A pen table may be used to run the macro, but to make sure that it is run each time a plot is used, create a plot macro that calls Print/Plot directly. Edit MicroStation’s File pull down menu and call the new macro instead of starting Print/Plot. Incorporate the plot accounting features into all batch-plotting macros too.

An alternative is to incorporate your macros into your pen table definitions, but make sure that pen tables are always used when plotting and that each pen table incorporates the plot accounting procedures.

So, plot accounting doesn’t have to be sophisticated. A few strategies are required to find and extract the right data, and with a few basic statements the data is available without expensive plotting software.