

Making Smart MicroStation Drawings

by Mark Stefanchuk, Cadgurus.com

Imagine for a moment a future scene between a designer and a project manager. They are reviewing a plan. The project manager points to a box on the drawing and asks, “What’s that?” As in some science fiction movie, our designer makes a couple of clicks and the screen fills with the specifications. The designer replies, “It’s an 8 gph water fountain. Type WF-1, and the preferred vender is Grainger, model 3H013. They are manufactured by Elkay, and cost \$501.50.” And, “How many do we need?” A couple of more clicks, “52”.

I’ve introduced this “future” situation for two reasons. First and most importantly, you can do this right now, with MicroStation without writing any software and you don’t need a sophisticated 3D application to have access to these intelligent drawings. And secondly, you can make it better using tags, and a little VBA development. After all where would this column be without VBA? We’ll get to this later.

First let’s follow our scenario, of the designer and project manager along the MicroStation path. We will implement a simple strategy using only MicroStation. Then later I’ll present some strategies to improve upon it.

To make this example work you will need a design file with at least one element in it. Draw a box or place a cell to represent the water fountain. Open engineering links. To start Engineering Links, go to the *Tools>Engineering Links*. This will open the E-Links toolbox.

Today we will use only three commands in Engineering links, “Attach Engineering Link”, “Follow Engineering Link”, and “Connect to Browser”.

Tips

Listing of AccuDraw Shortcuts

a PDF of the Quick Reference Guide is included on the commercial CDs. The last page has a listing of AccuDraw shortcuts.

CAD

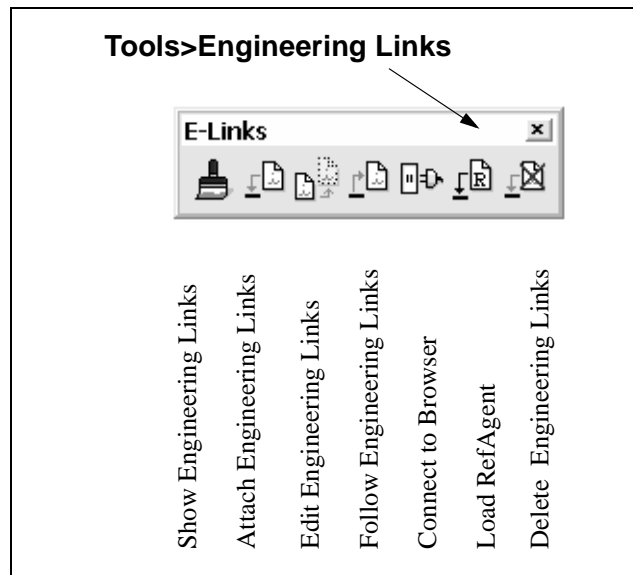


Figure 1 E-Links Toolbox

- Click on the Connect to Browser tool to open your default internet browser. Click on the MicroStation window to bring it back to the front, but do not close the browser window.
- Click on the Attach Engineering Link tool to open the Attach Engineering Link dialog. There is a field for the URL. To get the URL we need to surf the internet for one.
- Bring the web browser back to the top of the desktop. Use the window that MicroStation opened. Do not open a new browser window.
- In the address field type **www.grainger.com**. On the top right hand corner of the Grainger web page is a search field, Type “Water Cooler” in this field.
- Click the Go button to display a list of water fountains. Pick one of the water fountains by clicking on the Manufacturer’s name.

This will take you to a catalog page which will display the specifications for this water fountain. I chose this one.

<http://www.grainger.com/Grainger/productdetail.jsp?xi=xi&ItemId=1611591699>

- Copy (highlight the text then ctrl+c) and paste the new address from the browser address field into the URL text field located on the Attach Engineering Link dialog.

Remember that to paste into a MicroStation text field is shift+insert and not ctrl-v like it is in Windows.

Next, attach the link. As soon as you clicked on the “Attach Engineering Link” command icon, the prompt field has been telling you to “Identify Element”.

- So go ahead, click on and accept your water fountain. This will associate the web page from the Grainger catalog with your water fountain.

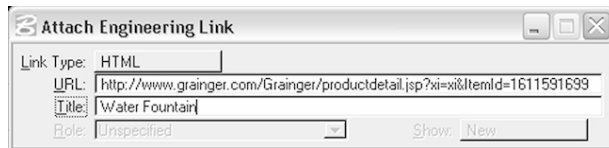


Figure 2 Attach Engineering Links Dialog

To trace the link back use the command “Follow Engineering Link”. To demonstrate, return to your default home page by clicking on your browser’s home icon.

- Now, in MicroStation click on the Follow Engineering Links tool. Click on your water fountain. The browser will be redirected back to the Grainger catalog page displaying the water fountain.

Now the next time your project lead leans over your shoulder and asks, “What’s that?” you can tell him, with just a couple of clicks. And, what about the number of water fountains?

Here’s one method. Draw the elements of the fountain on a unique level. Using power selector you can choose level name to filter out elements. In the bottom right hand corner of the MicroStation window is the number of items selected. See Figure 3.

Tips

Using the Modify tool on Dimensions

Using the Modify tool, you can adjust the extension line location, text position and dimension line location. It is also possible to change the extension line if you have Association Lock *On*. Then snap to the new position and accept; it will update to the new location and re-associate.

CAD

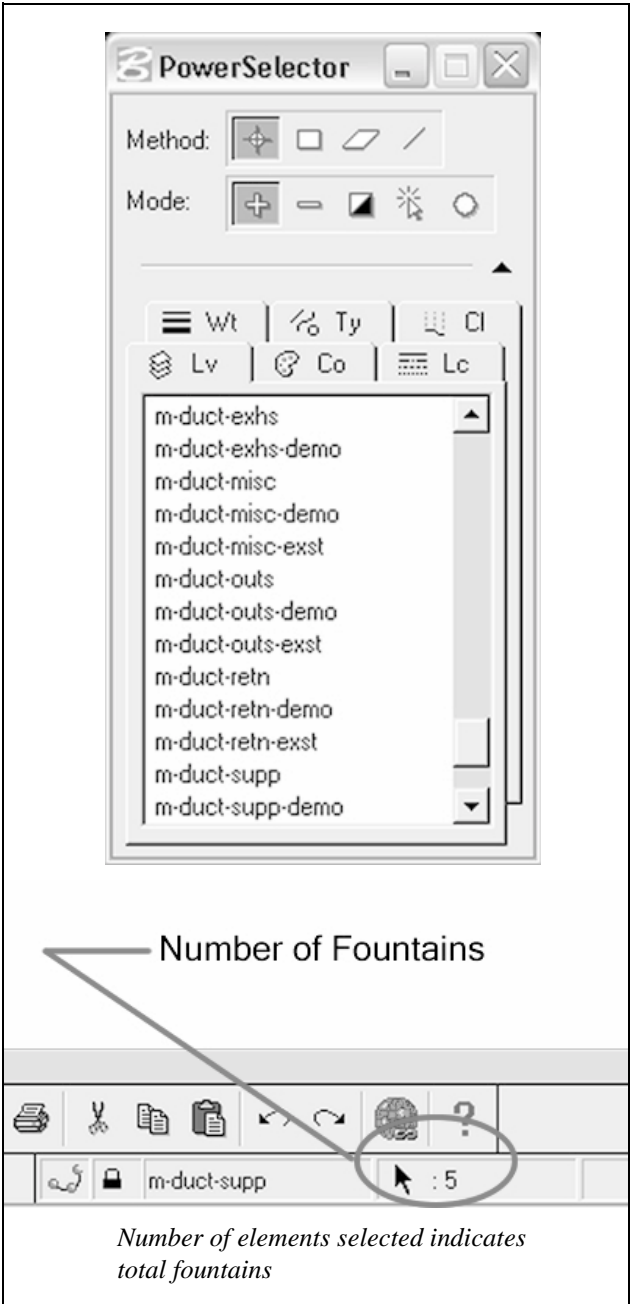


Figure 3 Power Selector Element Level

So far we’ve used MicroStation tools to associate non-graphic, or intelligent data to elements. And to the boss it looks very futuristic. But, although answering important questions this data only provides a small degree of design support.

What if though you could count all of the water coolers of that type WF-1 in all drawings, create a schedule of all cooler types, and provide a running estimate of fountain costs. To do that we need to investigate some more advanced features of MicroStation, including VBA.

Tags, and VBA – Even Smarter Drawings

Tags are the simplest method available in MicroStation for associating non-graphic design information with a graphic element. In fact, Engineering Links uses tag technology to associate the web page to an element.

All elements including lines, boxes, and cells may be tagged with design information. Vendor, model, size, color, material, price, design type, and specification reference are all non-graphic information that you may want to have available with a mouse click.

Smart Tags Program

To attach a tag we don't need to write any software, but I find this limiting. When you are in a hurry to complete a design layout you don't want to stop to make sure that the tag definition exists in the design file, and you don't want to have to remember to attach the tag.

The idea then is to create a command, say for placing a water fountain, so that when the fountain is placed the tag is placed too. But this command should be generic enough to accept any cell and any tag set. That way you can use it to customize your interface like tool boxes and pull down menus without writing more code. Here's one way to do it.

Create a primitive command. To do this add a class-module and call this module `classCellPT`. The class module contains the handlers to place cell. To review the code download it from the cadgurus.com download page. Look for the download called **Smart Tags**.

To run the command:

```
Vba run placeCellAndTag wf1,waterfountain
```

Where `wf1` is the water fountain cell name (a cell with this name must exist in one of your cell libraries). And where `waterfountain` is the name of the tag set.

When the cell is placed the add tag function is executed. We called this function `TagCellPT`.

```
TagCellPT oElt, GetTagSet(gTagSetName)
```

The first thing that happens is that `GetTagSet` is called. This function checks to make sure that the tag set exists. If it does the function returns the tag set, but if not a tag set is added to the design file. **See Code 1** below.

`AddTagStringDefinition` takes the new tag set and adds it to the design file. `TagCellPT` attaches the tag to the element. **See Code 2** below.

Now that our water fountains all have tags we can use the data in the design process. We now have the ability to look at all of the files within our project and count all of the WF-1 fountains and ignore WF-2, and WF-3 fountain types.

Count the Fountains

We can use VBA to create a macro that counts all of the elements with the "Type" set to "WF-1". Here's a script you may use to step through each file and count the fountains. **See Code 3** on page 29.

Quickie Tip - Keyboard Shortcuts

```
{Ctrl}+[Z] -----> Undo
{Ctrl}+[R] -----> Redo
{Ctrl}+[X] -----> Cut
{Ctrl}+[C] -----> Copy
{Ctrl}+[V] -----> Paste
```

CAD

Code 1

```
Function GetTagSet(strName As String) As TagSet
    Dim oTagSets As TagSets
    Set oTagSets = ActiveDesignFile.TagSets
    On Error Resume Next
    Set GetTagSet = oTagSets(strName)
    If GetTagSet Is Nothing Then
        Set GetTagSet = oTagSets.Add(strName)
        addTagStringDefinition GetTagSet, "Type", "WF-1", False, True, ""
        addTagStringDefinition GetTagSet, "Mfr", "Eklay", False, True, ""
        addTagStringDefinition GetTagSet, "Size", "8 gph", False, True, ""
        addTagStringDefinition GetTagSet, "Price", "550.50", False, True, ""
        addTagStringDefinition GetTagSet, "Model", "3H013", False, True, ""
        addTagStringDefinition GetTagSet, "Vendor", "Grainger", False, True, ""
    End If
End Function
```

Code 2

```
Sub TagCellPT(ele As CellElement, oTagSet As TagSet)
    Dim oTags() As TagElement
    setTagSetString oTagSet, "Type", "WF-1", False, True, ""
    setTagSetString oTagSet, "Mfr", "Eklay", False, True, ""
    setTagSetString oTagSet, "Size", "8 gph", False, True, ""
    setTagSetString oTagSet, "Price", "550.50", False, True, ""
    setTagSetString oTagSet, "Model", "3H013", False, True, ""
    setTagSetString oTagSet, "Vendor", "Grainger", False, True, ""
    oTags = ele.AddTags(oTagSet)
End Sub
```

Code 3

```
Sub CountAllFountains(projectLocation As String)
    Dim fso As Object, folder As Object, f As Object, flist As Object,
        totalWF As Integer

    Dim dFile As DesignFile, i As Integer
    Set fso = CreateObject("Scripting.FileSystemObject")
    Set folder = fso.GetFolder(projectLocation)
    Set flist = folder.Files
    i = 0
    For Each f In flist
        If VBA.InStr(1, f.Name, ".dgn", vbTextCompare) > 0 Then
            Set dFile = OpenDesignFile(f, True)
            totalWF = totalWF + CountFountains
            i = i + 1
            If i < flist.Count Then
                dFile.Close
            End If
        End If
    Next
    Debug.Print Str(totalWF)
End Sub

Function CountFountains() As Integer
    Dim ee As Enumerator
    Dim el As Element
    Dim i As Integer, gwfcnt As Integer, oTagElems() As TagElement
    Set ee = ActiveModelReference.Scan
    Do While ee.MoveNext
        Set el = ee.Current
        If el.IsGraphical Then
            If el.HasAnyTags Then
                oTagElems() = el.GetTags
                For i = 0 To UBound(oTagElems)
                    If StrComp(Trim(oTagElems(i).TagDefinitionName), "Type") = 0 Then
                        If Instr(1, Trim(oTagElems(i).Value), "WF", vbTextCompare) = 0 Then
                            gwfcnt = gwfcnt + 1
                        End If
                    End If
                Next
            End If
        End If
    Loop
    CountFountains = gwfcnt
End Function
```

Here's some homework. Can you modify the script to accept any tag definition name, and any value? In other words can you make it more generic? Can you build a form to display, start, and present the results instead of using the debug.print?

Creating Schedules

Schedules report and provide general specifications for a "type" of a equipment. Finding and reporting all of the tags found on elements is relatively simple, but for the schedule we only want to report the item once. This poses some interesting problems which we will look at next.

First, it is a good idea to build an array of tags for the component we are about to build the schedule. To do this we want to scan the project files once counting the number of components. We already know how to do that. The next step processes the files again to load the tag data into the array.

I created a VBA form to display execute our reports and collect the results.

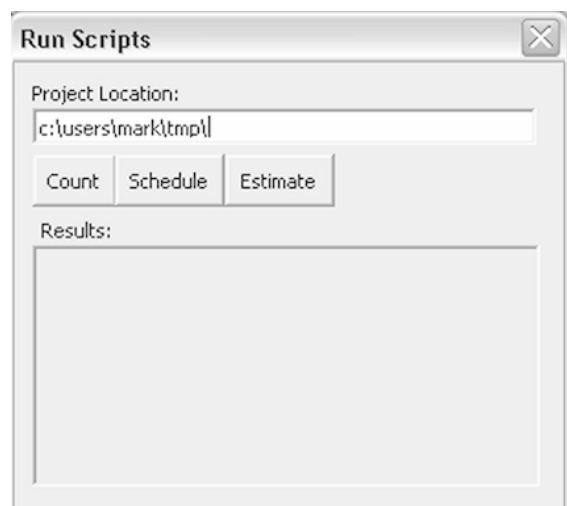


Figure 6 Sample VBA Form for Running Reports

The first two lines are cleaning up the report list box on our VBA form and reporting the total number of fountains it found.

```
formRunScripts.lstStatus.Clear
formRunScripts.lstStatus.AddItem Str(totalWF)
```

The array variable is dimensioned to hold the total number of fountains.

```
ReDim schedListHolder(0 To totalWF - 1)
Set fso = CreateObject("Scripting.FileSystemObject")
Set folder = fso.GetFolder(projectLocation)
Set flist = folder.Files
```

The program calls the subroutine `FountainTags` to process each file. `NextTag` is initialized to `0` and is used as the array index. We want the index to be set to the number of fountains found in the file - `1` when `FountainTags` returns control back to this procedure. See **Code 4** below.

Incrementing "i" and comparing it to the total number of files process allows us to leave MicroStation open when the report ends. If you don't do this, the user will be left with a blank MicroStation Window.

```
        i = i + 1
        If i < flist.Count Then
            dFile.Close
        End If
    End If
Next
```

`FountainTags` is called by the previous routine. It builds the array of fountains. See **Code 5** below.

`FountainTags` scans for all elements in a design file. It checks that the element found has tags and then verifies that tag definition "Type" is part of the tag. Finally, if type is found on the tag, then the definition value is evaluated to see if it contains the fountain code "WF". See **Code 6** below.

Due to the way that the tag set and definitions were created, the "Type" definition is the last indexed. Although our strategy could be to create "Type" first when defining a tag set, we don't really want our code to depend on the user to follow this convention. So, instead, if one of the tag definitions meets our criteria and it's name is "Type", then step through all of the definitions again to build our tag string. See **Code 7** on page 31,

Tips

Equal Spacing Text Program

Is there some kind of utility for providing equal spacing between multiple items of single line text? We have files imported from other CAD programs where all text comes in as single line entities. The line spacing is frequently not uniform, etc.

Yes there is, try `txt2node` from Cadgurus

<http://www.cadgurus.com/learn/freedwnl/txt2Node.asp>

CAD

Code 4

```
nextTag = 0
i = 0
For Each f In flist
    If VBA.InStr(1, f.Name, ".dgn", vbTextCompare) > 0 Then
        Set dFile = OpenDesignFile(f, False)
        FountainTags schedListHolder, nextTag
```

Code 5

```
Function FountainTags(wfType() As String, nextTag As Integer) As Integer
    Dim ee As ElementEnumerator
    Dim el As Element, k As Integer, tmpStr As String, okContinue As Boolean
    Dim i As Integer, gwfCnt As Integer, oTagElems() As TagElement
```

Code 6

```
Set ee = ActiveModelReference.Scan
Do While ee.MoveNext
    Set el = ee.Current
    If el.IsGraphical Then
        If el.HasAnyTags Then
            tmpStr = ""
            oTagElems() = el.GetTags
            okContinue = False
            For i = 0 To UBound(oTagElems)
                If StrComp(Trim(oTagElems(i).TagDefinitionName), "Type") = 0 Then
                    If InStr(1, Trim(oTagElems(i).Value), "WF", vbTextCompare) > 0 Then
                        okContinue = True
                    End If
                End If
            Next
        End If
    End If
Next
```

Code 7

```
    If okContinue Then ' do over to get all definitions
        For i = 0 To UBound(oTagElems)
            tmpStr = tmpStr + Trim(oTagElems(i).Value)
            If i < UBound(oTagElems) Then
                tmpStr = tmpStr + ","
            End If
        Next
        wfType(nextTag) = tmpStr
        nextTag = nextTag + 1
    End If
End If
End If
Loop
End Function
```

The last part of the FountainTags function removes all duplicate entries reporting only one of each type of fountain. If we say that any change in specification – a change in size, price, or model – represents a new fountain “Type” then we only need to compare exact differences between each record.

Let’s say we have 10 fountains in our project. We run our program to find all of the fountains and save our schedule data from each into a string array as described above. That means the array has 10 strings. Let’s also say that there are four different fountain types used on the project, WF-1 through WF-4. When the project files are scanned we may get an array that looks something like this,

```
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H012,4,26.50,211.00, 3gph,Eklay,WF-3
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H445,4,26.50,355.00, 5gph,Eklay,WF-4
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H012,4,26.50,211.00, 3gph,Eklay,WF-3
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H565,4,26.50,483.00, 6gph,Eklay,WF-2
Grainger,3H565,4,26.50,483.00, 6gph,Eklay,WF-2
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
```

If we step through this array reporting new types, when a change occurs, we would get nine of the fountains back.

```
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H012,4,26.50,211.00, 3gph,Eklay,WF-3
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H445,4,26.50,355.00, 5gph,Eklay,WF-4
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H012,4,26.50,211.00, 3gph,Eklay,WF-3
```

```
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H565,4,26.50,483.00, 6gph,Eklay,WF-2
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
```

Better results are possible however, if the data were sorted. In this case we could count the number of rows that follow which are the same as the one reported and then move our index to the row that follows when a different type is encountered. The next code fragment demonstrates how we do this, start by sorting the array.

Many different sort algorithms are available for free on the internet or from most programming text books. I like this bubble sort routine which I found at Chris Rae's VBA Code Archive - <http://chrisrae.com/vb>. It does the job and was already written in visual basic.

```
BubbleSort schedListHolder, True
formRunScripts.lstStatus.Clear
Dim matches As Integer, unqids As Integer
unqids = totalWF
```

After the sort is complete, enter the array at the top and report the first type of fountain found. See **Code 8** below.

Count the number of fountains that match the current fountain.

```
matches = matches + 1
End If
Next
```

Move the index to the record in the list that is different from the previous record reported.

```
i = i + matches - 1
unqids = unqids - matches
Next
```

Code 8

```
For i = 0 To UBound(schedListHolder)

    ' always report the first
    formRunScripts.lstStatus.AddItem schedListHolder(i)
    matches = 0
    For k = i To UBound(schedListHolder)
        If StrComp(schedListHolder(i), schedListHolder(k), vbTextCompare) = 0 Then
            ' increment counter to the next unique
```

At the beginning of the previous code fragment the sorted array groups all "like" types together.

```
Grainger,3H012,4,26.50,211.00, 3gph,Eklay,WF-3
Grainger,3H012,4,26.50,211.00, 3gph,Eklay,WF-3
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
Grainger,3H445,4,26.50,355.00, 5gph,Eklay,WF-4
Grainger,3H565,4,26.50,483.00, 6gph,Eklay,WF-2
Grainger,3H565,4,26.50,483.00, 6gph,Eklay,WF-2
```

On the first pass through the array the index is at the top so the first line is reported.

```
Grainger,3H012,4,26.50,211.00, 3gph,Eklay,WF-3
```

The next line is skipped because it is the same as the first so the matches variable moves the index to the third line. This line is reported.

```
Grainger,3H013,4,26.50,550.50, 8gph,Eklay,WF-1
```

The matches variable is re-initialized and counts off 4 more rows and increments the index to point at the next different row,

```
Grainger,3H445,4,26.50,355.00, 5gph,Eklay,WF-4
```

Finally, there are no matches for the WF-4 record so the index increments by 1 and reports the last fountain type.

```
Grainger,3H565,4,26.50,483.00, 6gph,Eklay,WF-2
```

The schedule is complete.

Estimate the Cost

To make the tags even more valuable to the project, they should be able to generate an estimate. For the water fountain example here is a simple calculation based on the material and labor cost of the project. All of the data may be extracted from the tags.

labor * hours + unit price = estimate.

Add these to get a total estimate.

The script is a combination of the counter and schedule procedures. The function `EstFountains` returns a double representing the total cost instead of an integer representing the number of fountains. The element scan is the same, as that found in the schedule scan, but since we don't need all of the definitions, we specify which ones we need for the calculation. See **Code 9** below.

Now you have several futuristic tools to impress your project manager. You can apply these same tools to any component in the design – doors, windows, pumps, pipe. All of the data stays with the MicroStation file and with a little ingenuity you can use the same strategy to design, build, and manage your building, roadway, or map.

About The Author

Mark Stefanчук 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

Code 9

```
If okContinue Then ' do over to get all definitions
  For i = 0 To UBound(oTagElems)
    If StrComp(Trim(oTagElems(i).TagDefinitionName), "Price") = 0 Then
      gwfPrice = Val(Trim(oTagElems(i).Value))
    End If
    If StrComp(Trim(oTagElems(i).TagDefinitionName), "Labor") = 0 Then
      gwfLabor = Val(Trim(oTagElems(i).Value))
    End If
    If StrComp(Trim(oTagElems(i).TagDefinitionName), "Hours") = 0 Then
      gwfHours = Val(Trim(oTagElems(i).Value))
    End If
  Next
  gwfTotalCost = gwfTotalCost + gwfPrice + (gwfHours * gwfLabor)
End If
```



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