AUGI CAD Camp - Denmark
Copenhagen, Denmark - 13 March, 2007
Network. Learn. Share.

AutoCAD

A Hands-On Introduction to VBA Programming

Paul Oakley

S4-1

Course Summary:

Outstanding - VBA has never been explained better. If you are new to VBA, this is the class for you. 75 minutes of the essentials of VBA programming is perfect for beginning your journey into the wonderful world of VBA for AutoCAD

Instructor:

Paul Oakley is Director of Oakley-CAD Services Ltd and has 20 years' extensive experience within the AEC industry, working as both architect and CAD manager for major Architectural practices such as Broadway Malyan and PRP Architects before starting his own CAD management consultancy. As a consultant he has provided services to clients ranging from Autodesk to individual Architectural practices. Paul runs CAD management, bespoke AutoCAD and ADT training courses and has also presented at major conferences, plus featuring in industry press. Paul has been involved within the development of ADT with Autodesk for several years and chairs the technical forum for the ADT Community group. He has also been involved in various industry led Building Information Modelling initiatives, such as Teamwork and more recently Avanti.

Autodesk User Group International

www.AUGI.com

Introduction

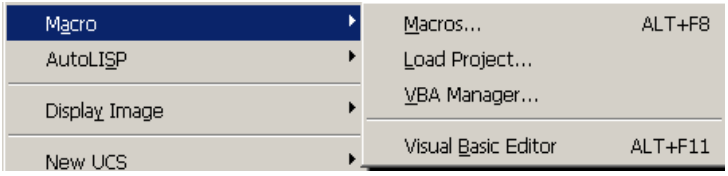
Welcome to the Hands-On introduction to AutoCAD VBA. The class will be a quick overview of how VBA works and a demonstration of some of the capabilities of VBA and what is achievable with it. There are many good books on the subject and plenty of online help including the Autodesk discussion groups. Some references are listed at the end of this paper.

Visual Basic for Application (VBA) was developed by Microsoft as an extension of the basic programming language that developed into Visual Basic (VB). It has been around since the early 1990s but was first introduced into AutoCAD in R14 and developed through AutoCAD 2000. Slowly more features of AutoCAD have become exposed for use by VBA, which has now become a widely used customization tool. The fact that many software venders have incorporated VBA into their products by licensing from Microsoft the VBA integral Development Environment means that data can be manipulated across software via VBA applications. This not only goes for Microsoft products, AutoCAD and its vertical products, but also many other software products and file formats.

NOTE: AutoCAD VBA is not available in AutoCAD LT although Visual Basic (VB) can be used to communicate with AutoCAD LT through the Windows environment

Using VBA

The Tools Menu in AutoCAD holds the 4 main commands used for VBA within AutoCAD.



1. The Macros button (VBARUN) will allow loaded Macros to be run.
2. Load project command (VBALOAD) opens a VBA project (Dvb File type)
3. VBA Manager (VBAMAN) opens the Manager for accessing all aspects of VBA
4. Visual Basic Editor (VBAIDE) to open or create new Visual Basic projects

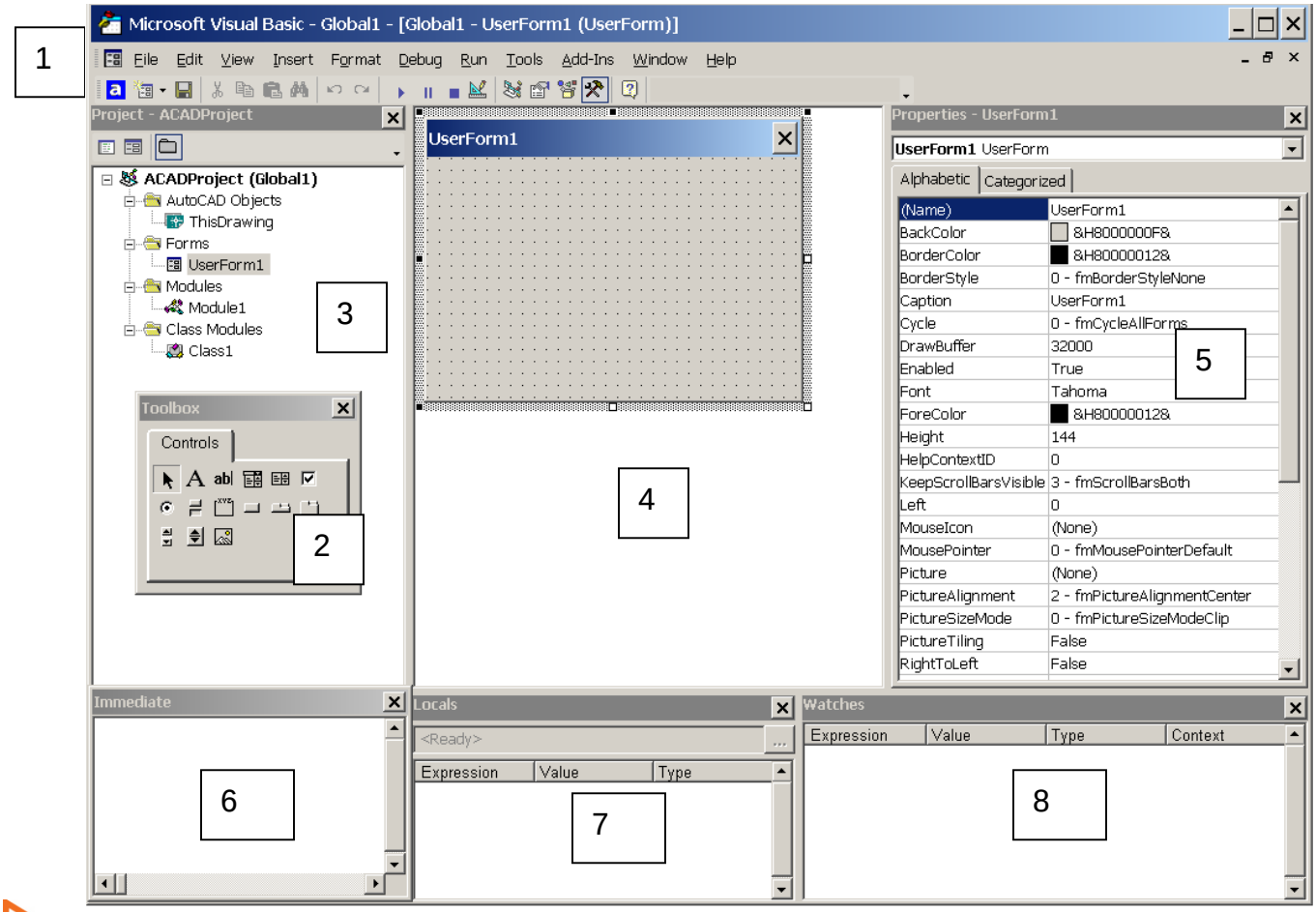
Macros

A macro are a piece of Code that will perform a specific task such as starting up a VBA application. Macros can be either stored in the current drawing (Thisdrawing object) or stored in VBA projects, which are DVB files.

Integrated Development Environment (IDE)

The IDE consists of the following components many of which exist in standard industry software. The Components are:

1. Menus / Tool Bars
2. Toolbox
3. Project Explorer
4. Code Window
    - Code Window
    - The Object and procedures boxes
    - Forms View
5. Properties Window
6. Immediate Window (debug window)
7. Watches Window (View defined values)
8. Locals Window (View Variables)

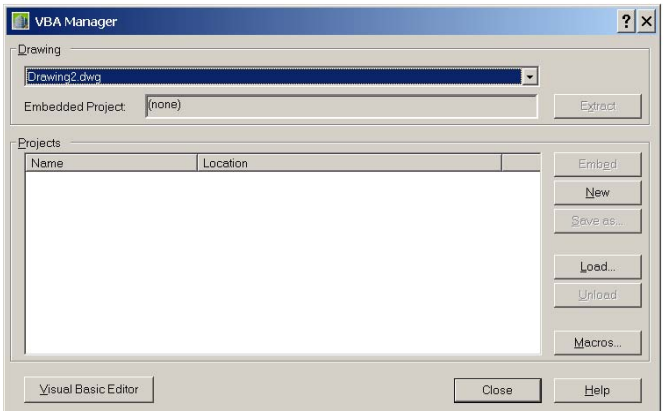


Creating, Opening and Saving VBA Projects

Use the VBAMAN command or access the VBA Manager from the Tools menu.

The VBA Manager provides the means to Extract, Embed, Create, Saveas, Load or Unload a DVB project.

The IDE menu only gives options to Save.



SAVE – BEFORE RUNNING
Remember to always SAVE before running a project.
If you have forgotten to exit a loop you may need to crash the software to exit the routine and will loose any work.

VBA Projects (DVB File)

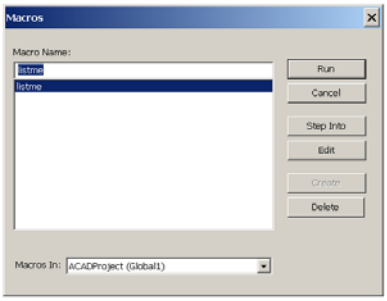
VBA projects consist of modules that are containers for your program code. These modules can be viewed within the project explorer and comes as three types:

- UserForm Module – This holds the code for your GUI interface which is your form with Controls. The Module holds code such as declaration of constants, variables, procedures and event procedures.
- Standard Modules are best used to hold Macros and routines that do not fit into a Form module or a Class Module. They don't have a visible interface and can run without a User realizing they are taking place.
- Class Modules are used to create your own objects along with the associated methods, properties and events that go with that object

The use of modules and where you put your code is similar to working with AutoCAD. Most AutoCAD beginners put all their data in one drawing file. Only when they need to share data with others for multiple working will they split the work into multiple files and the benefits of Xrefs and blocks are realized. Most VBA beginners do a similar thing and will place all their code in the UserForm Module. As the program becomes more complicated the benefits of having separate Modules, Class modules, Procedures and Macros will start to become more obvious and the benefits to break your code down appropriately are realized.

Our First VBA Project

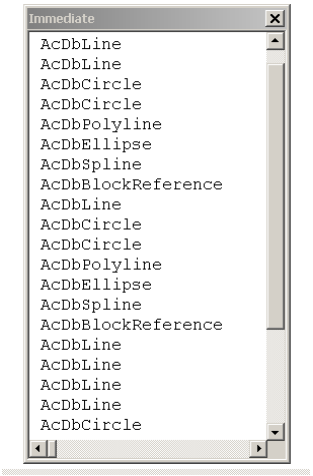
List everything in a drawing



Create a new VBA project using the VBA Manager.
Within the VBA IDE select the Tools Menu and the Macro option.
This should bring up the following Macro Form.
Call the Macro "Listme" and use the create button.
Add the following code into the Standard Module within the Sub Listme

```
Sub listme()
Dim MyEntity As AcadEntity
For Each MyEntity In ThisDrawing.ModelSpace
    Debug.Print MyEntity.ObjectName
Next MyEntity
End Sub
```

Add Toggle Breakpoint (F9)



Run the Macro and the object names of the entities within the current drawing are listed within the Intermediate Window. If you add a toggle breakpoint to the code and run again. The properties of each of the entities can be viewed within the Locals Windows.

Additional Properties can be printed to the debug window as long as all these properties exist for each entity. These are properties such as Color or Linetypes. Alternatively these properties can be changed. For example by adding the line *"MyEntity.color = acByLayer"* within the For / Next loop all entities can have their color property changed to ByLayer.

Language

Unfortunately Visual Basic being a programming Language has it's own terminology that goes with it. AutoCAD also has its own terminology some of which will be known to Users but other sections are unique to the VBA environment. I have attempted to cover some of the terms covered but please refer to the help files or the book listed at the back for further guidance.

Variables and Data Types

Variables let you retain values and are a name that is used to refer to a location in memory that contains an item of Data. Variables have specific data types that can range from a simple "Byte" to a complex "Variant".

It is important to use the correct data type for your variables or your code may fail. There are often situations where a value used such as 0 (zero) is assumed to be a integer (number) but is in fact a

string “0” and therefore cannot be used by your code. The immediate window locals windows let you view your variables as your code runs and states the Data types.

Variable are declared using the Dim Statement

Dim MyEnt as AcadEntity
Dim MyResult as String

To avoid issues with variables you can force visual basic to force you to declare all your variables by adding OPTION EXPLICIT at the top of your code module.

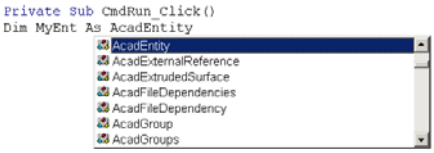
Declaring Variables
You do not have to declare variable in Visual Basic as a specific data type but the largest Variant data type is assigned which can lead to memory issues. If you misspell a variable another variable will be created which can lead to debugging problems.

See the VBA help file for more information on Variables and Data Types.

VBA Help

When programming in VBA there is text sensitive help, which will select software specific help. For AutoCAD VBA help select an AutoCAD specific data types such as “AcadEntity” and then press F1. For General VBA help just press F1

VBA will provide Autolists and the Auto quick info feature will provide context help on what values VBA is looking for.



Ensure that the project Variables are set correctly and that the appropriate References have been include in your VBA Project.

Procedures

There are three types of Procedures, which are blocks of code that have a specific purpose. The three types are:-

- Sub Procedures – These are blocks of code that are called from other procedure. They may contain parameters when called but will not return a value
- Functions Procedure – These procedures are similar to Sub Procedure but they return a value to the calling procedure.
- Events Procedures - These procedures are triggered when an event happens such as a Button is clicked, something is changed or a command is invoked.

Decision Structures

Decision Structures are used to test conditions, values or state of a variable before carrying out an operation. Typical Decision structures are:

- If, then
- If, then, else, else if
- If Not
- Select Case

If we look back at our first VBA project we can start to introduce Decision structures to test specific conditions. By using an "If then" statement the value of our Entity can be checked and then certain properties changed. An example would be to change the Layer of all block references to Layer 0 by adding the code

```
Sub listme()
Dim MyEntity As AcadEntity
For Each MyEntity In ThisDrawing.ModelSpace
   Debug.Print MyEntity.ObjectName
      If MyEntity.ObjectName = "AcDbBlockReference" Then
            MyEntity.Layer = 0
      End if
Next MyEntity
End Sub
```

Added If Then statement

To deal with multiple object types a Select Case Statement would be used. Here we will check for multiple object types and change the color for each type accordingly.

```
Sub Runme()
Dim MyEntity As AcadEntity
For Each MyEntity In ThisDrawing.ModelSpace
   Select Case MyEntity.ObjectName
   Case "AcDbLine"
      MyEntity.color = acBlue
   Case "AcDbCircle"
      MyEntity.color = acGreen
   Case "AcDbPolyline"
      MyEntity.color = acMagenta
   Case "AcDbBlockReference"
      MyEntity.color = acRed
      End Select
Next MyEntity
End Sub
```

Loop Structures

In order to carry out code repeatable or to execute loops based upon conditions the various Loop structures are used. Specifically useful is the for each loop which can be used to iterate through collections. Most of AutoCAD's data is stored in collections.

Do While Loop

- Do While Ini1 >1 ' test value
- Code to make something happen
- Loop
-

Do Loop While

- Do
- Code to make something happen
- Loop While Ini1 >1 ' test value
-

For Next

- For I = 1 to 10
- Code to make something happen 10 times
  I = I + 1
- Next

For Each Next

- For Each Element in Collection
- Code to make something happen
- Next

How to get out of Loops

Exit

- Exit Do
- Exit For

Nested Structures

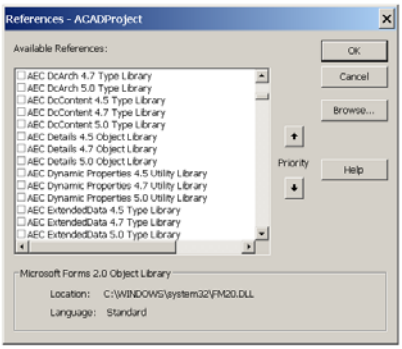
It is possibly to nest various loops within one another along with decision structures. It is generally good practice when nesting control structure to indent the code so that the start and end are recognizable.

Useful Tip:
Add a toggle break point inside a loop to test that it works.
If you have created no exit to a loop then you will need to crash VBA and AutoCAD and all work that is not saved will be lost.

Comments
It is good practice and often necessary to add comments to your code using the (') symbol.
Comments should turn green on the code screen.

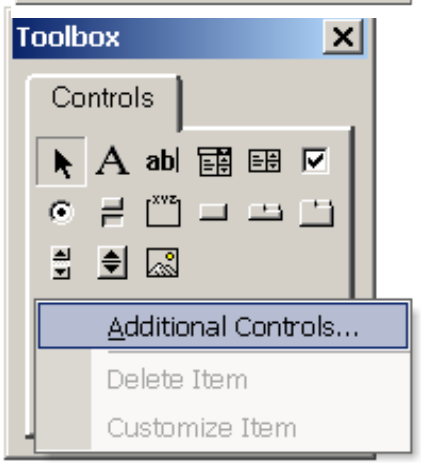
References

In order to use different VBA objects the relevant references for that application need to be loaded in VBA. Different versions of AutoCAD and its vertical products such as ADT and ABS have there own libraries of objects. These can be loaded via selecting the references option from the Tools menu.



Forms and Controls

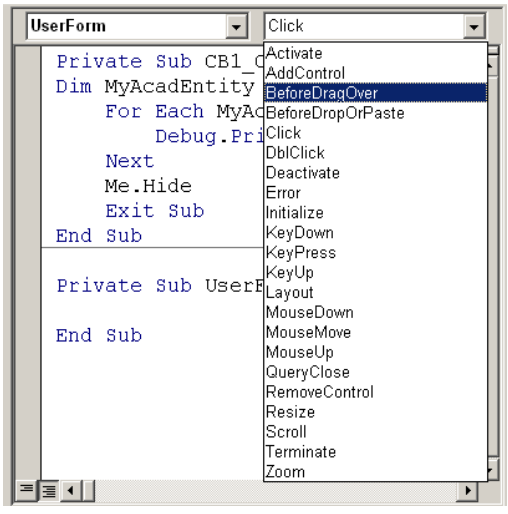
Top Row
Select Objects / Label / Text Box / Combo Box / List Box / Check Box
Middle Row
Option Button / Radio Button / Frame Command Button / Tab strip / Multipage
Bottom Row
Scroll Bar / Spin Button / Image



Try adding the various controls onto a form. Then select the controls and view the properties available for the specific controls. Additional Controls can be added depending upon Software by right clicking the Toolbox and selecting the Additional controls option.

Values for the Controls can either be edited via the property dialog or the values can be set by code within your modules

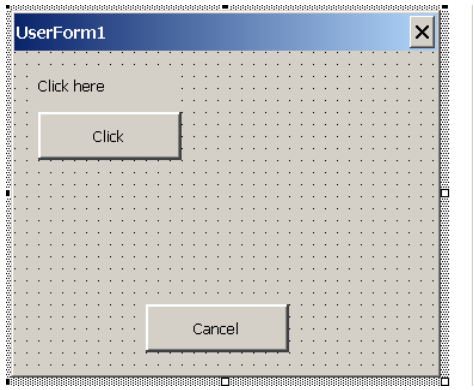
Forms and Controls have identifiable events, which are viewable by the object and procedure boxes. If the object is selected such as “UserForm” then the various events such as “click” for that object can then be chosen.
The appropriate Event procedure is then automatically created and is ready for further code to be written. UserForms and Controls are objects. It is possible to add new Controls to the form at runtime from code and to also change their properties.



The UserForms Exercise highlights some of the possibilities for Form and Controls and show what can be done with them.
As this code makes no reference to AutoCAD Objects it would work the same in either Excel or Word VBA

UserForms Exercise.

```
Private Sub cmdClick_MouseDown(ByVal Button As Integer, ByVal Shift As Integer, ByVal X As Single, ByVal Y As Single)
ChangeColour 'Run ChangeColour procedure
End Sub
---------------------------------------
Private Function ChangeColour()
If frmForm1.Left = 10 Then
    frmForm1.Move 500
    frmForm1.Caption = "Right"
    'Change Background Colour
    frmForm1.BackColor = RGB(100, 100, 150)
    Label1.BackColor = RGB(100, 100, 150)
    'Move Label and button positions
    Label1.Left = 12
    CmdClick.Left = 12
    'Change button text
    CmdClick.Caption = "Right"
Else
    frmForm1.Move 10
    frmForm1.Caption = "Left"
    'Add Change Colour
    frmForm1.BackColor = RGB(150, 200, 150)
    Label1.BackColor = RGB(150, 200, 150)
    'Move Label and button positions
    Label1.Left = 130
    CmdClick.Left = 130
    'Change button text
    CmdClick.Caption = "Left"
End If
End Function
---------------------------------------
Private Sub UserForm_Click()
Dim Mytxb As TextBox
Set Mytxb = Controls.Add("Forms.Textbox.1")
    Mytxb.Left = 18
    Mytxb.top = 100
    Mytxb.Width = 175
    Mytxb.Height = 20
    Mytxb.Text = "Did you mean to Click me?"
End Sub
Private Sub cmdCancel_Click()
Unload Me
End Sub
```



Create a new User Form called UserForm1.

Add a Label called Label1

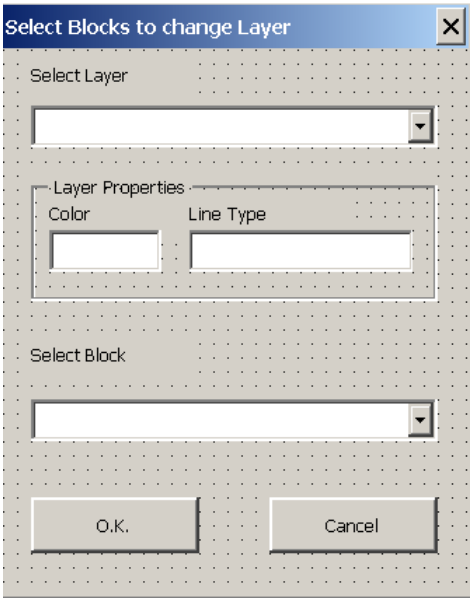
Add a Button called CmdClick.
Add a Button Called CmdCancel

Add the code.

Run the Code
Click on the buttons to see what happens.

Also click directly on the form!

Routine to Change Layer of Selected Blocks



This routine demonstrates how to access objects in Collections and uses the following collections:

- Model Space
- Layers
- Blocks

Collections are accessed via Loop structures using a “For Each / Next Loop”. Decision Structure such as “If then Else” or “If Not Then Else” are used to Test each object in the collection.
When the form is loaded this triggers the Activate event and two procedures are used to interrogate the Layer and Block collections. The Contents are then added to the relevant combo boxes and in the case of the Layers the properties (Color and Line Type) are added for the selected Layer.

When the O.K. button is selected the routine searches the drawing collection for any entities and then tests the “objectname” property by using “If statements”. This checks that it is a block reference and checks the name of the block using the “EffectiveName” property (“Name” for pre 2006 versions) to ensure that it matches the value in the combo box.

If the names match then the Entities Layer property is changed to match the name in the Layer combo box. Other properties of the Entity such as Color could also be changed at this point if required.

Note
As Labels and Frames will not be changed via Code default names are acceptable.

Add Form Objects
Create a new User Form called “UserForm1”. Change caption to “Select Blocks to change Layer”

Add a Label & change caption to “Select Layer”

Add a ComboBox called “CmbBlock”

Add a Frame and change caption to “Layer Properties”

Add a Label and change Caption to “Color”
Add a Label and change Caption to “Line Type”

CODE

```
Option Explicit
'----------------------------------------
Private Sub UserForm_Activate()
'Call Procedures AddLayer and AddBlock
AddLayers
AddBlocks
End Sub
'----------------------------------------
Private Sub AddLayers()
Dim MyLayer As AcadLayer
CmbLayers.Clear
'Iterate through the Layers collection and add to the combo box
For Each MyLayer In ThisDrawing.Layers
    CmbLayers.AddItem MyLayer.Name
Next MyLayer
CmbLayers.ListIndex = 0
```

```
End Sub
'---------------------------------------
Private Sub AddBlocks()
Dim MyBlock As AcadBlock
CmbBlocks.Clear
'Iterate through the blocks collection and add all blocks that do not start with "*" to the combo box
For Each MyBlock In ThisDrawing.Blocks
    If Not Left(MyBlock.Name, 1) = "*" Then
        CmbBlocks.AddItem MyBlock.Name
    End If
Next MyBlock
If CmbBlocks.ListCount > 0 Then
    CmbBlocks.ListIndex = 0
Else
    CmbBlocks.ListIndex = -1
End If
End Sub
'---------------------------------------
Private Sub CmbLayers_Change()
Dim MyLayer As AcadLayer

'When Selected Layer in Combo box changes
'Iterate through Layers collection to find Layer name
'in Combo box and add properties to the Textboxes
For Each MyLayer In ThisDrawing.Layers
    If MyLayer.Name = CmbLayers.Value Then
        TxbLineType.Text = MyLayer.Linetype
        TxbColor.Text = MyLayer.color
        Exit For
    End If
Next
End Sub
'---------------------------------------
Private Sub CmdRun_Click()
Dim MyEnt As AcadEntity
Me.Hide
'Iterate through Modelspace and find all objects that
'are Block References
For Each MyEnt In ThisDrawing.ModelSpace
If MyEnt.ObjectName = "AcDbBlockReference" Then
    'Check to see that Block Name equals BlockReference Name
    If MyEnt.EffectiveName = CmbBlocks.Value Then
        'Change Layer of Block reference
        MyEnt.Layer = CmbLayers.Value
```

Form Objects (Cont....)

Add a Textbox called “TxbColor”
Add a Textbox called “TxbLineType”

Add a Label and change Label caption to “Select Block”

Add a ComboBox called CmbBlock

Add a Command button called “CmdCancel”.
Add a Command Button called “CmdRun”

Add the code.

```
    End If
End If
Next MyEnt
Unload Me
End Sub
'----------------------------------------
Private Sub CmdCancel_Click()
'Quit VBA macro
Unload Me
End Sub
```

This routine shows some of the power of VBA in AutoCAD where you can now start to develop your own applications. The ability to iterate through the various collections available within the DWG database provides the power to either view or change the various properties.

I doubt if this limited 75 minute class will be able to do much other than wet your appetite of the capabilities of VBA. VBA in itself has a limited life span as Microsoft is replacing VBA within their 64 bit applications. Until then AutoCAD and Microsoft applications can all take advantage of the abilities of VBA and I am sure that the skill you learn with VBA will help in learning whatever future programming languages become available.

Hopefully this class has at least given you a useful introduction to VBA and the power that it can bring to your AutoCAD use. Some other useful sources of information are: -

Books

The most up to date book I have found is AutoCAD 2006 VBA A Programmers Reference by Joe Sutphin. Other Useful books include:

Using Visual Basic with AutoCAD
Mastering AutoCAD VBA – Marion Cottingham
VBA for AutoCAD 2005 - Guide for the Non-Programmer - Jerry Winters

AutoDesk Discussion Groups

There are also plenty of examples and help from the Autodesk discussion groups which can be accessed via : http://discussion.autodesk.com/

Select “AutoCAD” and then “Visual Basic Customization”

Web Sites

There are many examples on the Web and within the AUGI web site of VBA code and what can be done with it. Try searching Google or Yahoo for "AutoCAD VBA". Some of the results include:
http://vbcad.com/
http://www.contractcaddgroup.com/download/
http://www.activedwg.com/html/VBA.htm

AUGI

and Don't forget the AUGI Web Site Http://www.AUGI.com.
There are often VBA articles within AUGI World and there are also various sources of VBA Code specifically in the old AU Classes that are available.

Finally

I hope you enjoyed the Class. If you need any assistance feel free to contact me at:

Paul Oakley BA(Hons), Dip Arch, RIBA.
Oakley CAD Services Ltd

Email: Paul@Oakley-CAD.co.uk

Web Site: Http://www.oakley-cad.co.ukT