



AUGI Training Program

VBA, Access & AutoCAD

Mike DeGraw
"A" Segment

Sponsored by:



**CAD
Technology
Corp.**

P.O. Box 1117
Franklin, NC 28744

Voice 828-369-3979
Fax 828-369-3972
johnbo@cadtechcorp.com

**Your drawings are only as good as
the symbols that complete them...**

Trust CAD Technology Corp. for symbols. For 17 years, CAD Technology Corp. has delivered high-quality symbols to the CAD community. We offer more than 40 symbols and parts libraries, each featuring thousands of individual symbols, for every conceivable application in Mechanical, Piping, Electrical, Hydraulics, Electronic, Ergonomics, Architectural, Civil, Structural, Landscaping, and Drafting.

For details and examples visit our website at www.cadtechcorp.com.

★ **SPECIAL OFFER FOR AUGI MEMBERS** ★
10% off all orders placed in October 2002

Autodesk User Group International

AUGI Training Program (ATP)

This course (and every course you are registered for) will only continue if you re-register for it after completing every segment. To do this (and you can re-register for all the courses you're taking at one time for each segment) use the ATP Registration webpage. It will be updated with each segment (A, B, C, D and E) and you will need to re-register using it to keep the courses you signed up for going this semester. No one is forcing you to do this, but if you don't, your courses may get cancelled. It's that simple.

If you have a question or comment about the content of any segment, we encourage you to post a message to your course's corresponding ATP Mail List. Teachers moderate their class mail list and may take up to 2 days to respond to questions you put to them. Proper use of any ATP Mail List is outlined on the ATP Mail List web page. If you have a question or comment about the ATP program in general, you can write to atpstaff@augi.com. Please do not write to atpstaff@augi.com with questions about course content unless there is a typo or a technical problem with a file.

You are welcome to publish the content of this or any ATP course segment for the benefit of your Local User Group or co-workers so long as AUGI and the course instructor receive full credit for generating it. The content of this course may not be sold in any form and is copyrighted. This lesson and other AUGI Training Program materials have been generated by the faculty and staff of Autodesk User Group International, a non-profit user-focused organization which depends on your participation.

Autodesk, the Autodesk logo, AutoCAD, 3D Studio MAX, Autodesk VIZ, AutoCAD LT, Visual LISP and AutoLISP are registered trademarks of Autodesk, Inc., in the USA and/or other countries. AUGI is a servicemark of Autodesk, Inc., licensed exclusively to the Autodesk User Group International. Windows95, Windows98, Windows XP, Windows NT and Windows 2000, Visual Basic, Microsoft Word, Microsoft Excel, and Microsoft Access are trademarks of Microsoft Corporation. All other brand names, product names, or trademarks belong to their respective holders. Copyright 2002 AUGI. All rights reserved.

-The ATP Staff

ATP608 - VBA, Access & AutoCAD

Faculty: Mike DeGraw

Welcome to AUGI training. This is one of the many resources that I use to learn new things. Little background on myself. I've been using AutoCAD since release 2.6 and programming since release 10. I work for one of the largest Trade Show General Contractors in the US and serve here as the director of programming in AutoCAD. I have a degree in Mechanical Engineering and a background in several languages from Fortran to VBA. I enjoy VBA as it runs very well in the AutoCAD environment and has a very refined user interface. Like you I rely on documentation to program in any language. Sometimes experience shared with others clears up the fog that sometimes exists in the documentation or provides information that is just not documented. I hope my experience in this area will expand the possibilities of programming for you to a new level.

So here are my assumptions about you and your system, or in other words here is the environment I program in and would hope yours is similar.

ACAD2000 as a minimum.

Windows NT or Higher (I've never used Windows 95/98 so if your using those platforms this information might or might not work). (I'm using Windows 2000 so the screen captures at the system level may not look like yours if you are using some other OS).

Your programming experience is beyond the beginner stage and you understand the difference between a regular module and a class module, and know how to reference other objects.

You should know how to set up and program a userform, and be familiar with methods, events, & properties.

I'll assume you know very little about Access or Databases in general.

With that said let's go.

First some terminology and information. **You Don't Need Access loaded on your machine to use the Access Database!**

If you don't have access or if the machine/s you want to run your program on do not have Access loaded you can still run the program with an Access driver. These drivers are loaded in the "ODBC Data Source Administrator" section of your OS. It can be found in the control panel either in "ODBC settings" or Administrative tools or something similar. You should find a Microsoft Access driver (*.mdb) loaded in that section. If you don't find that driver it can be downloaded from the Microsoft website. This driver allows AutoCAD to create, read, and write to an .MDB (Microsoft Data Base) file. If you have Access loaded then you have the convenience of also viewing the database if needed.

A database consists of several components that we need to understand quite well to successfully program in VBA. First you don't reference Access as an application like you would Excel or Word. Access is referenced using an engine that drives Access. The engine provides addition properties,

events, and methods in VBA that are specific to database (DB) programming. There are several engines available for DB programming each a little different in its application. We will be using the Microsoft DAO 3.6 Object Library. DAO stands for Data Access Objects and is one of the older slower engines, but provides direct access to the Access Database file. Some of the newer engines ActiveX Data Objects (ADO) and Remote Data Objects (RDO) outperform the DAO but I've found them a little more complex to use and DAO has very much suited my needs.

So from the VBA interface you select the **Tools** drop down button and select the **References** Option. From figure 1 you can see the correct selection of the DAO engine.

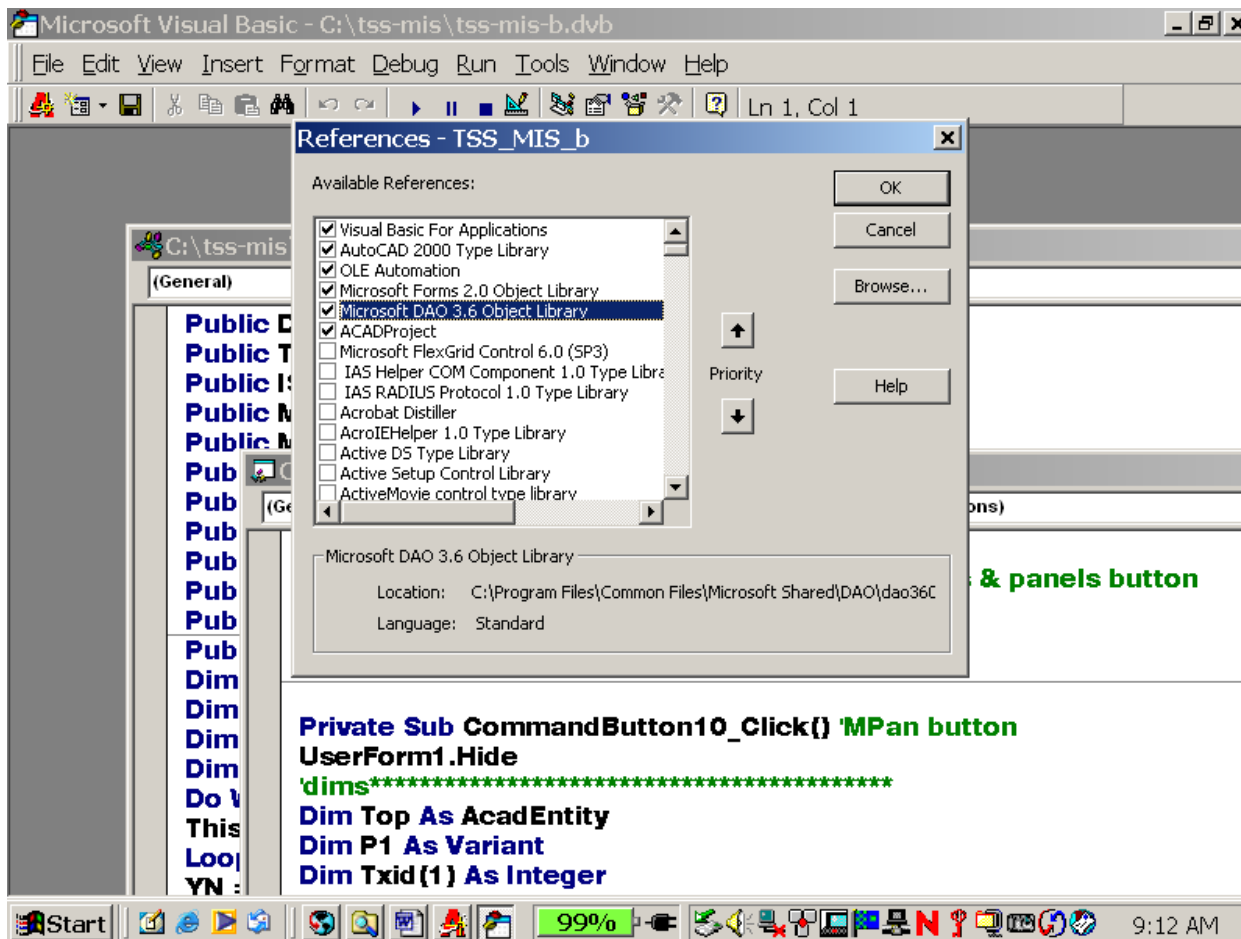


Figure 1:

Once this is referenced we can start programming in VBA using the DAO engine to link AutoCAD to an Access Database file. I'll hereafter refer to these as MDB files.

A database has several components that need to be understood in order to successfully reference them from VBA. These are the building blocks:

Application. In the real world the application is called "Access".

From VBA the application (Access) is opened by setting a variable to **DBEngine.Workspace(i)**. This is the programmatic method of “running” the Access program. Once the application is “running” we can open MDB files.

Tables. Within the MDB there are tables. A table is a collection of records. From VBA these are referred to as a **TableDef**. There can be many tableDef’s in each MDB and stored as a collection, each TableDef is a collection has a Name.

Records. Within each TableDef there are records. They are referenced in VBA as a RecordSet and stored as a collection. Each RecordSet can have many lines of records.

Fields. Each Record in a RecordSet contains Fields that can store data. Each field can be accessed once a record in the record set is made active. (It’s been my experience that Field Names cannot contain spaces. Therefore I use the underscore a lot to define field names.)

Data. The data contained in a field can be declared as a type ie String, Double, Date, \$, etc.

Lets look at a sample MDB file and see how all this fits together.

In this example we are going to get an MDB file that already exists. Later we’ll dive into how to create an MDB file on the fly.

Our file is **C:\Sample\MyFile.mdb**

This database file has one table defined called “Booth”, see figure 2

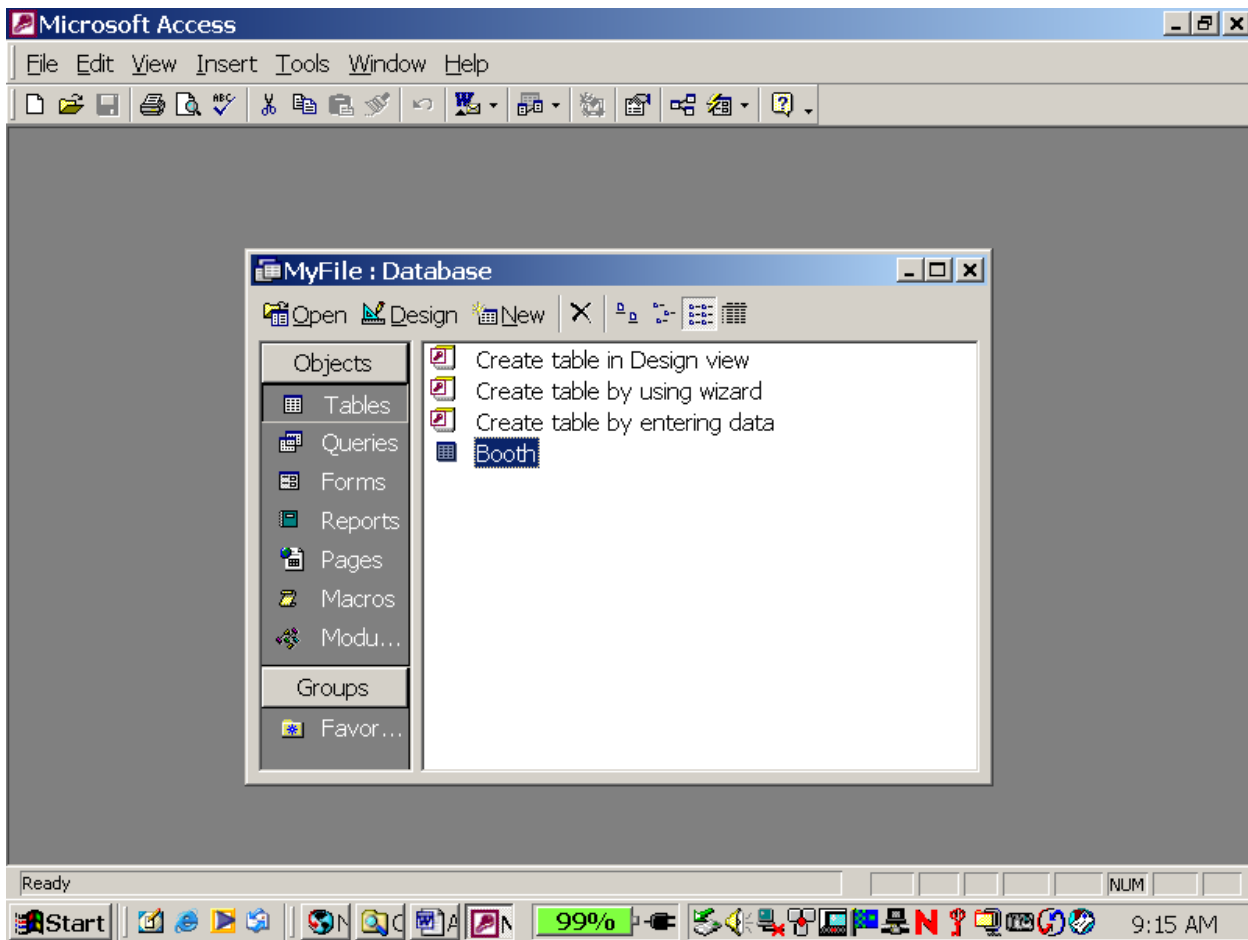


Figure 2:

“Booth” has two fields:

“Booth Number” (text field)

“Handle” (text field)

Booth has 4 Records as seen in figure 3.

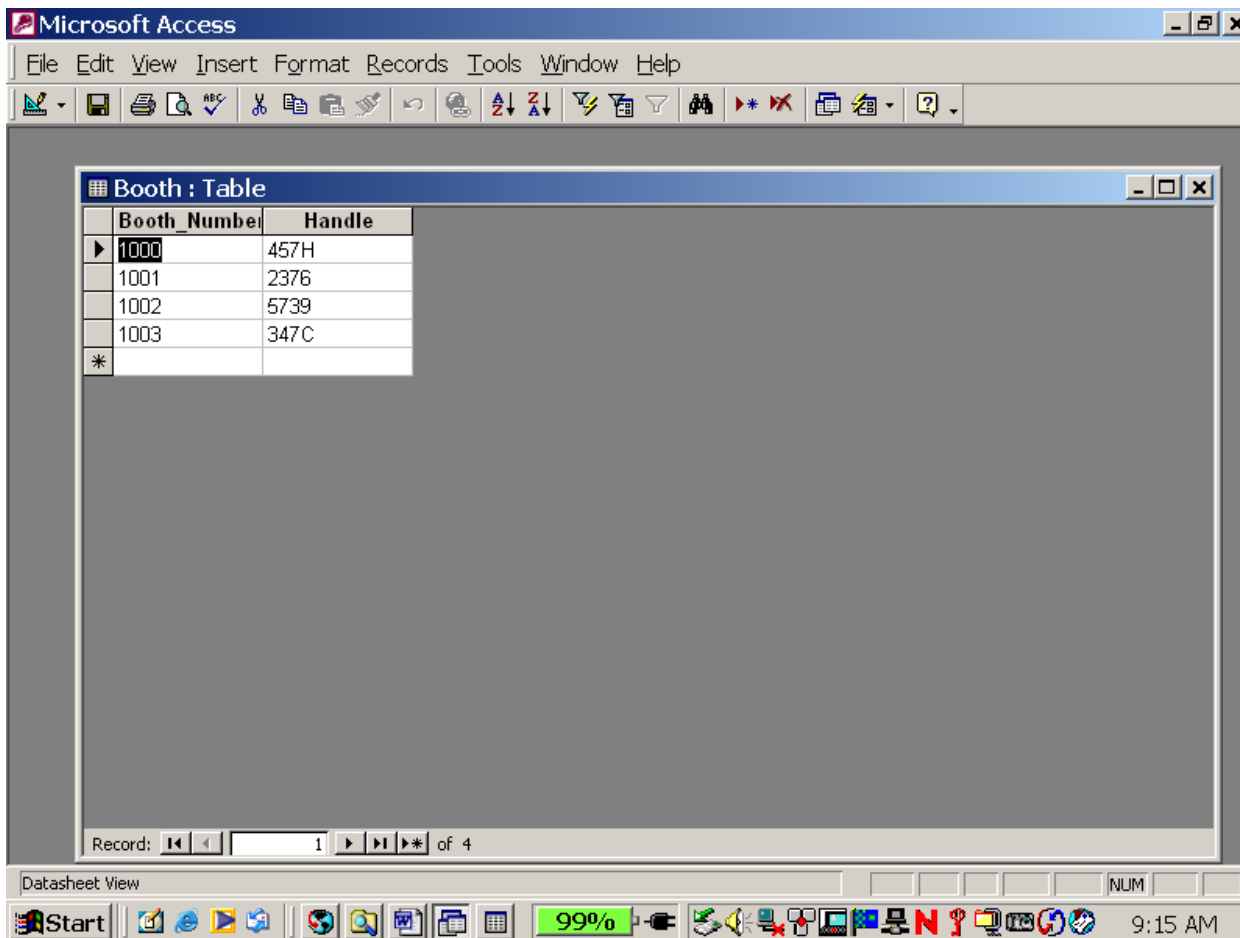


Figure 3:

The Programmatic steps of connecting and accessing data in an MDB file is as follows:

First. We must attach to the Application. In VBA this is done as follows:

```
Dim WS as Workspace 'Dimension a variable to represent Access  
Set WS = DBEngine.Workspace(0) 'set WS up as the application
```

With the DAO engine properly referenced we now have a workspace object that can be dimensioned. The WS variable name is arbitrary. This is the Access program. We are now attached to that application and can begin using DB functions to get at tables and records. This would be the equivalent of "running" Access in Windows with no current file.

Second. As stated above, once the Access program is "running" we want to open an MDB file. The file we want is **MyFile.mdb** In VBA we get to this file using the application or WS (remember WS represents the Access program). WS has several methods now, one of which is:

```
Dim DB as Database 'dimension a variable to represent the MDB  
Set DB = WS.OpenDatabase("C:\Sample\MyFile.mdb")
```

This will open the MDB file that has already been created and set it to the DB variable. DB is now the reference to the MDB file "MyFile.mdb". The Method .OpenDatabase is part of the DAO Engine.

Third. Now that the MDB file is open we would want to get at the data in the MDB file.

We must first open the table "Booth" and then start reading the records in "Booth". To do this we use the method:

Dim RS as Recordset 'set a variable to represent the records
Set RS = DB.OpenRecordset("Booth") where "Booth" is the TableDef already defined in the database. If we were creating a new DB we would have to define and create the TableDef, but since we are reading an MDB with a predefined table, the TableDef is already there. The Method .OpenRecordset is part of the DAO Engine.

Forth. We now have the table "Booth" opened in Access and can get at the data using the methods attached to the RS object. The methodology of getting at each individual record is as follows:

Once you open the table there is no active record. (Remember each line in figure 3 represents a record.) You make a record active with one of several methods:

```
.MoveFirst  
.MoveLast  
.MoveNext
```

These methods allow you to "step" around in the table. To make the first record active you would:

```
RS.MoveFirst
```

Now the first line in figure 3 is the active record and each field can be read as follows:

To read a field in the active record the format looks like:

```
Recordset!FieldName
```

So to read the "Booth Number" value and set it to a variable in VBA the code would look like:

```
Dim BN as String  
Dim BNH as String
```

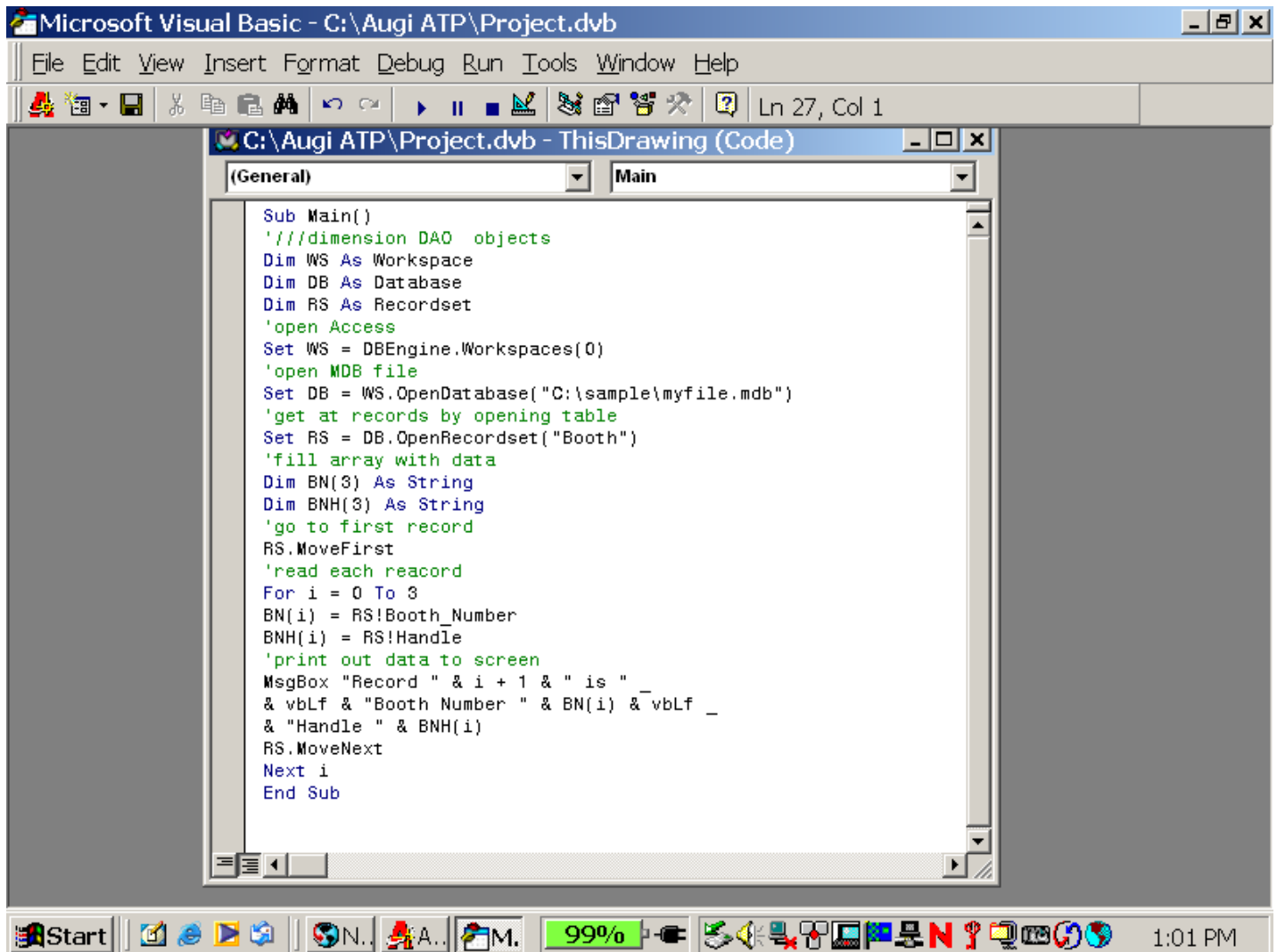
```
BN = RS!booth number  
BNH = RS!handle
```


This would set BN = 1000 and BNH = 457H (line 1 in figure 3)

I can now step to the next record in figure 3 by:

RS.MoveNext

You should be getting the picture by now. Lets put this all together.



```
Microsoft Visual Basic - C:\Augi ATP\Project.dvb
File Edit View Insert Format Debug Run Tools Window Help
Ln 27, Col 1
C:\Augi ATP\Project.dvb - ThisDrawing (Code)
(Main) Main
Sub Main()
'///dimension DAO objects
Dim WS As Workspace
Dim DB As Database
Dim RS As Recordset
'open Access
Set WS = DBEngine.Workspaces(0)
'open MDB file
Set DB = WS.OpenDatabase("C:\sample\myfile.mdb")
'get at records by opening table
Set RS = DB.OpenRecordset("Booth")
'fill array with data
Dim BN(3) As String
Dim BNH(3) As String
'go to first record
RS.MoveFirst
'read each record
For i = 0 To 3
BN(i) = RS!Booth_Number
BNH(i) = RS!Handle
'print out data to screen
MsgBox "Record " & i + 1 & " is " _
& vbCrLf & "Booth Number " & BN(i) & vbCrLf _
& "Handle " & BNH(i)
RS.MoveNext
Next i
End Sub
```

If you did not know how many records were in the table you could loop through the table as follows:

Do While Not RS.EOF

Code

RS.MoveNext

Loop

This would read each record in order until it found the End Of Record.

This is the basics of attaching to an ACCESS DATABASE and reading the table and records in an MDB file using VBA.

Some of the other methods available are:

- .AddNew
- .Edit
- .Update
- .Delete
- .MovePrevious

Many of the other methods deal with SQL strings which is beyond the scope of this course. But we will be looking into the .Seek method which makes finding records much easier than stepping through them in order.

With the above code create you own MDB file and start getting yourself familiar with the many properties and methods available using the DAO engine.

Next segment we'll dive in a little deeper and look at the methods of editing and updating, adding, & deleting records in a table and creating a New MDB file in VBA.