Chapter

4

Stamping with Four Holes

In this chapter, you will learn how to use the following VBA functions to World Class standards:

- **Beginning a New Visual Basic Application**
- **Opening the Visual Basic Editor in AutoCAD**
- **Laying Out a User Input Form in Visual Basic**
- **Creating and Inserting an Image into a Form in Visual Basic**
- **Insert a Label into a Form**
- **Insert a Textbox into a Form**
- **Insert Command Buttons into a Form**
- **Adding a Copyright Statement to a Form**
- **Adding Comments in Visual Basic to Communicate the Copyright**
- **Declaring Variables in a Program with the Dimension Statement**
- **Setting Variables in a Program**
- **Assigning Values to the Variables**
- **Inputting the Code to Draw in Visual Basic**
- **Resetting the Data with the cmdClear Command Button**
- **Exiting the Program with the cmdExit Command Button**
- **Exiting the Program with the cmdExit Command Button**
- **Executing a Subroutine with the cmdDraw Command Button**
- **Inserting a Module into a Visual Basic Application**
- **Running the Program**

Beginning a New Visual Basic Application

In this chapter, we will learn how to use the Visual Basic Application (VBA) program to first create a form and then to generate drawings automatically. We reiterate many elements of the previous lesson, but now we add the capability to add lines, circles and arcs in AutoCAD Model Space. Eventually in following chapters, we add text and dimensions, placing entities on specific layers, having multiple views and soon we will be completing entire drawings in seconds.

At the beginning of every chapter, we will start a new Visual Basic Application project, use a sketch to determine the extent of what the program will do, create the form and then write the code. Once the code is finished, we will run the program and an orthographic drawing will appear on the graphical display.

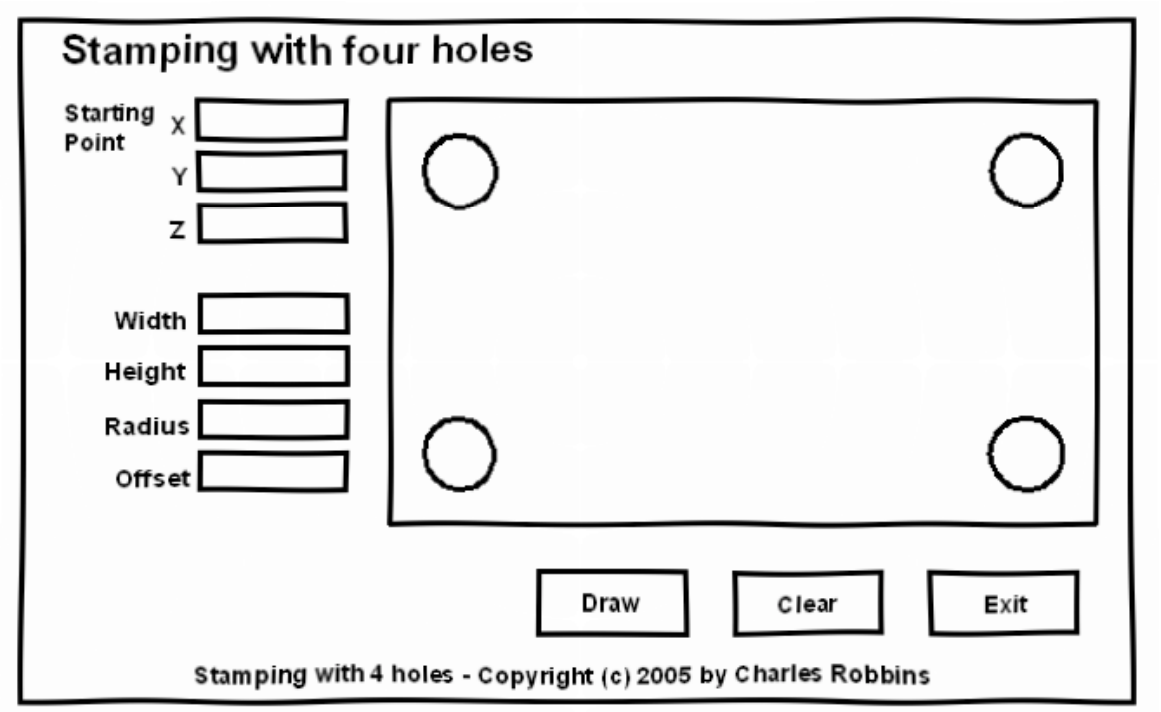

Figure 4.1 – Rough Sketch of the Stamping with 4 Holes and Arc Form

Remember, that all programming projects begin with one or more sketches, with one portraying the part, detail, or assembly and the other being the user input form. In this Visual Basic Project, Stamping with 4 Holes, we will be running a user input form inside the AutoCAD application, so we need to sketch the structure of this special dialogue box. We will name the Input form, **Stamping with 4 holes**. We will place seven textboxes on the left side of the form to input the starting point, width, height, radius of the holes and the offset distance from the edge of the part to the hole. On the right side of the form, we will place an image of the stamping. We will have three command buttons, **Draw**, **Clear** and **Exit**. On the bottom of the form, we will write the copyright statement using another label. On this presentation, we can help ourselves by being as accurate as possible, by displaying sizes, fonts, colors and any other

specific details which will enable us to quickly create the form. From the beginning of inserting the form into the project, we need to refer to our sketch. The sketch of the form is shown in Figure 4.5.

Remember, we should train new programmers initially in the art of form building. When using the editor, we insert and size the form, and selecting the Controls Toolbox, we will place all the various input tools and properly label them. Whenever we place an input tool, the properties window will display a list of every attribute associated with the tool, and we will take every effort to arrange the tool by performing such actions as naming, labeling and sizing the visual input device.

Opening the Visual Basic Editor in AutoCAD

Opening the Visual Basic Editor in AutoCAD is essential to creating the program to automate the drawing process. In this version of the World Class CAD – Visual Basic Applications for AutoCAD, we are using AutoCAD 2007, but we just finished using all the programs in this text with a group programming in AutoCAD 2002. Their drawings were automatically made just as efficiently as if they were using the most recent version of the Autodesk software.

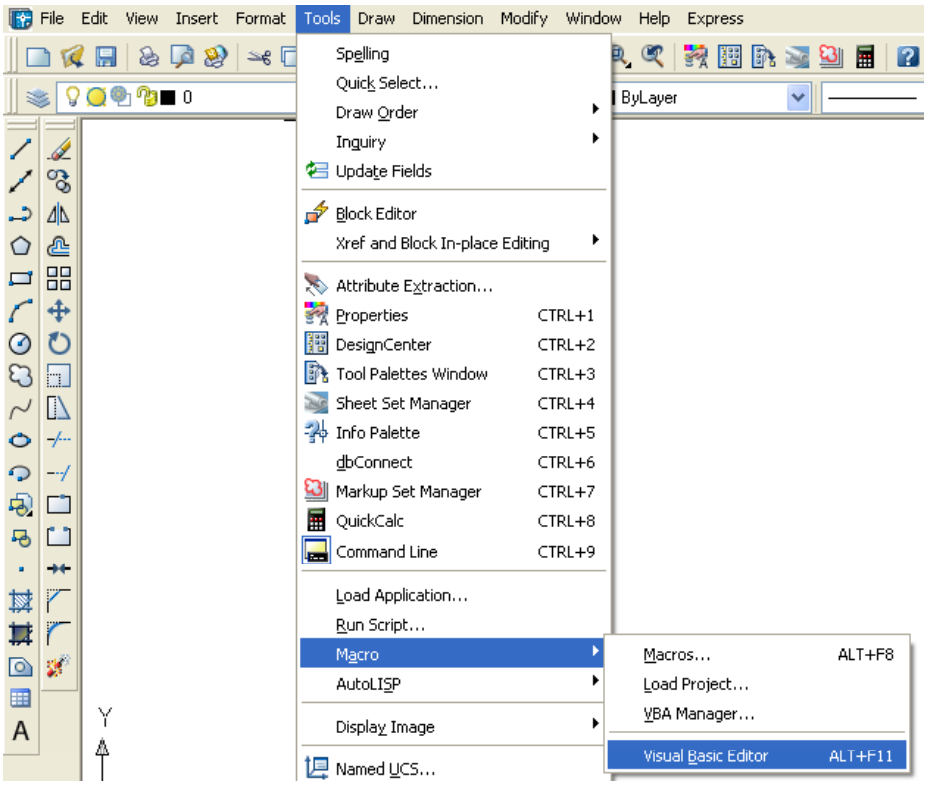

Figure 4.2 – Launching the Visual Basic Editor

Select Tools on the Menu bar; pick Macro and then choose the Visual Basic Editor. Look to the right of the phrase, Visual Basic Editor and the shortcut keys Alt – F11 is noted. For quick launching of the editor, press Alt – F11

The Visual Basic Editor will appear on the computer desktop as a new program application. Looking down on the computer's Taskbar, we can see the AutoCAD and Microsoft Visual Basic Editor program tabs. Just single click either program tab to switch between any applications. However, if we close the AutoCAD drawing, unlike a stand alone version of Visual Basic, the Visual Basic Editor will also close.

For those individuals with previous Visual Basic experience, the Visual Basic Editor in AutoCAD has the same layout as in other VB programs. The Menu Bar contains tools for our use as well as the four toolbars shown in Figure 4.4, which are Standard, Debug, Edit and Userform. Presently, only the Standard toolbar is showing. On the left side of the workspace is the Project menu, which shows the files pertaining to this project. Below the Project menu is the Properties pane. If we remember the Properties tool in AutoCAD, using this device will be simple.

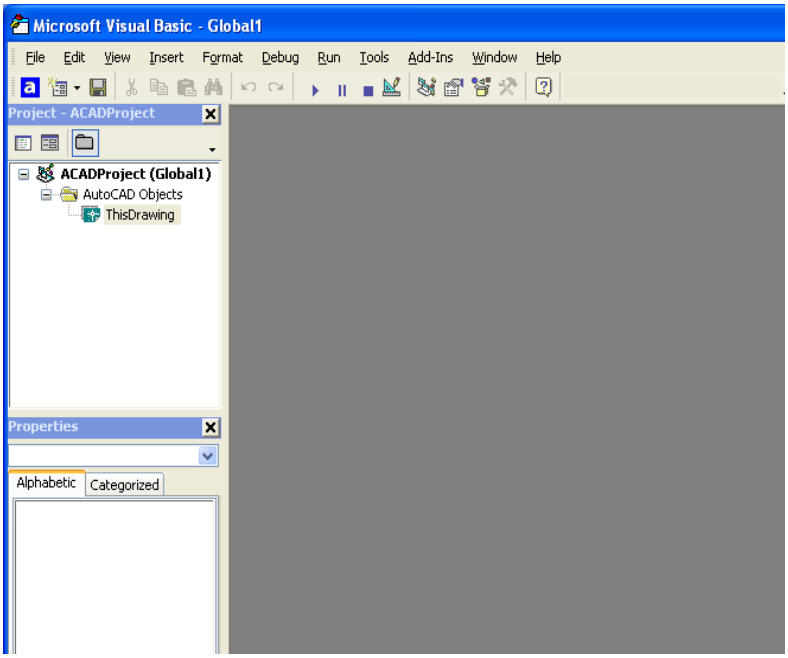

Figure 4.3 – The Visual Basic Editor

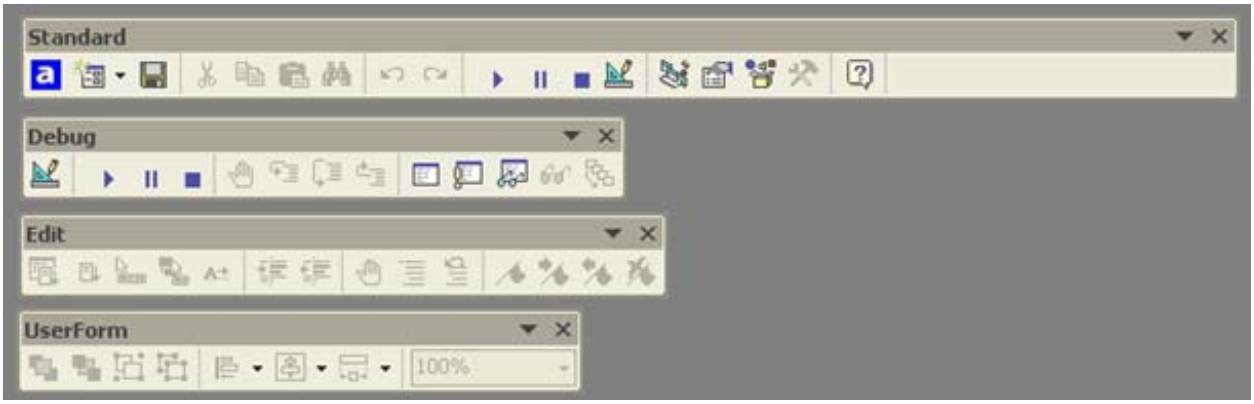

Figure 4.4 – Toolbars in the Visual Basic Editor

With the Visual Basic Editor open, select **File** on the Menu Bar and select **Save Project**. Remember, we have a folder on either the desktop or in the My Documents folder called "VBA Programs". Save the project with the filename "Stamping with 4 holes". The file has an extension called *dvb* which means DCL and Visual Basic programs as shown in Figure 4.5.

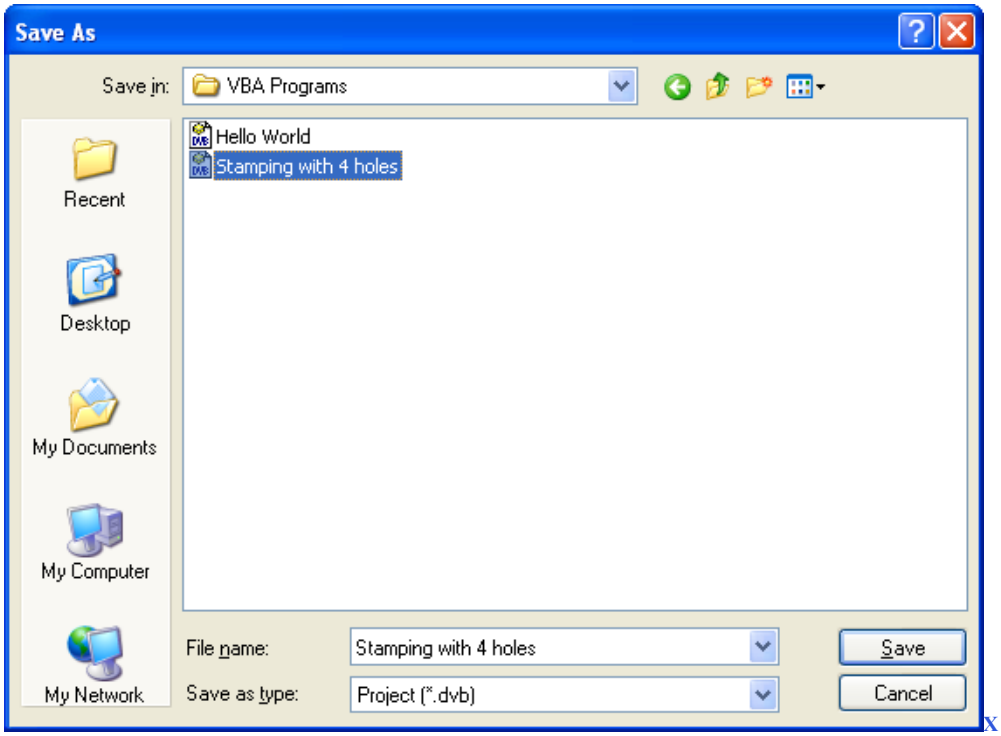

Figure 4.5 – Saving the Stamping with 4 Holes Program

Laying Out a User Input Form in Visual Basic

Now that we have an idea of what the dialogue box in our program will look like, select the **Insert UserForm** button on the Standard toolbar to insert a new form as shown in Figure 4.6. Instantaneously, the once grey work area is changed to contain our UserForm1. A Form folder with Userform1 is now in the Project menu and the Properties pane contains the attributes associated with UserForm1. (See Figure 4.7)

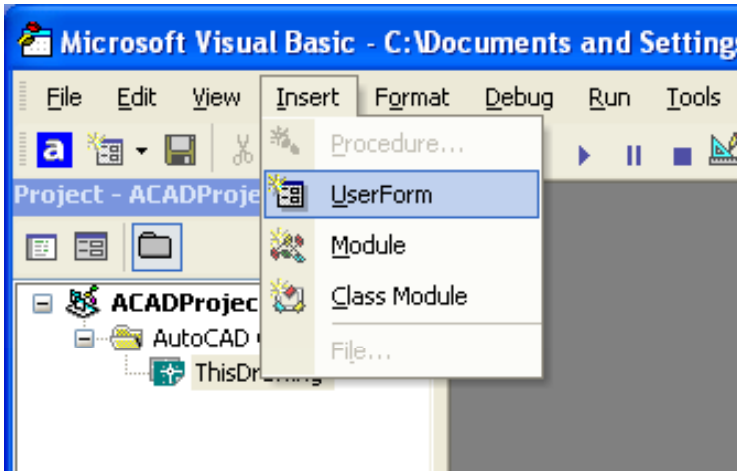

Figure 4.6 – Inserting a User Form

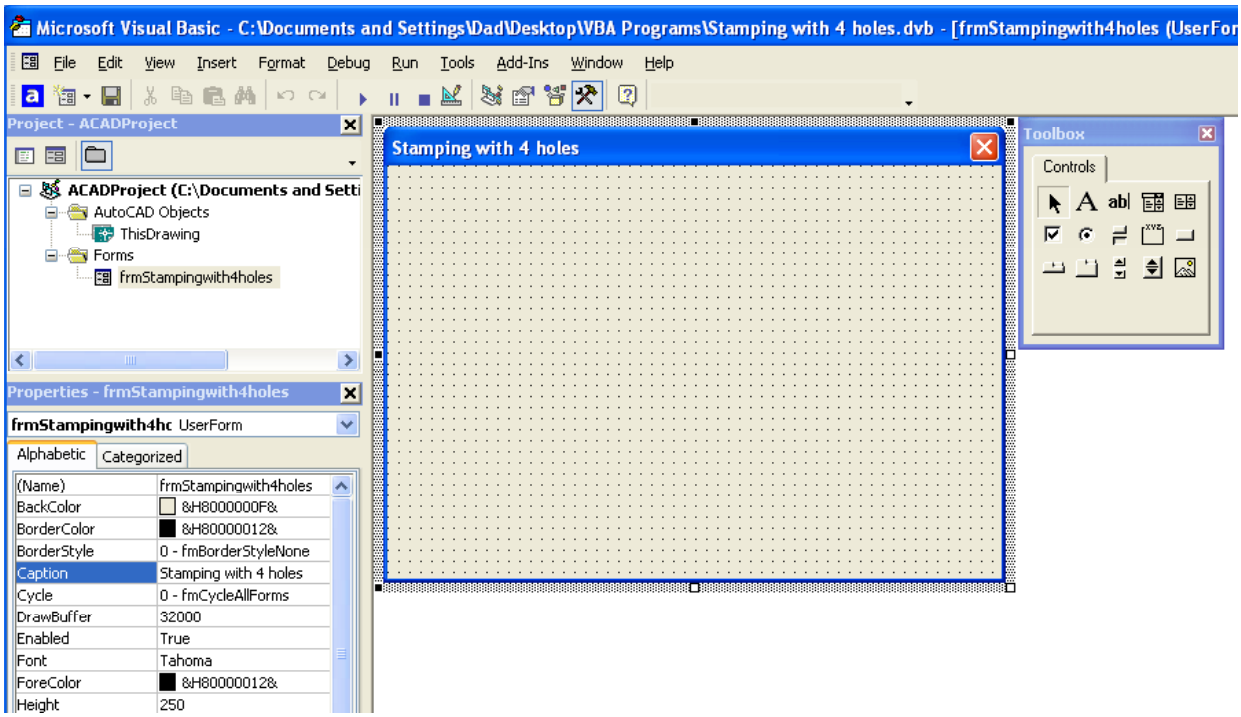

Figure 4.7 – Designing the Stamping with 4 Holes Form in Visual Basic

Next, we will change the **Caption** in the Properties pane to **Stamping with 4 Holes and Arc** to agree with the sketch in Figure 4.5. Go ahead and change the form in two other aspects, Height and Width.

Alphabetic	
BackColor	&H80000013&
Height	250
Width	400

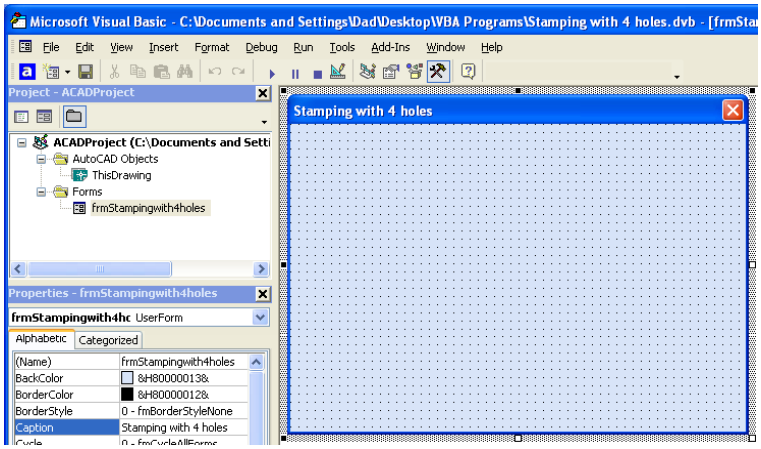

Figure 4.8 – Setting the Caption and other Properties

The form will change in size to the height and width measurement. The background color will change to a light blue. There are many more attributes in the Properties pane that we will use on future projects.

Creating and Inserting an Image into a Form in Visual Basic

Different from the last chapter, this form will have a picture of the part that we will create automatically, so we need to make a drawing of part in AutoCAD. Dimension the drawing as we do in any other drawing, but we will use the Edit Text tool to remove the actual dimension and write in the word that matched the textbox label. In Figure 4.8, we show dimensions that associate the Width, Height, Offset and Radius textbox with the drawing. When the drawing is finished, we need to save the drawing as an image file. Use the **Saveimg** command to save file on the VBA Programs folder. Create a folder named Images in the VBA Programs folder and save the file as the same name as the program for matching purposes, Stamping with four holes. We saved the file as a Bitmap with a width of 300 pixels and a height of 200 pixels.

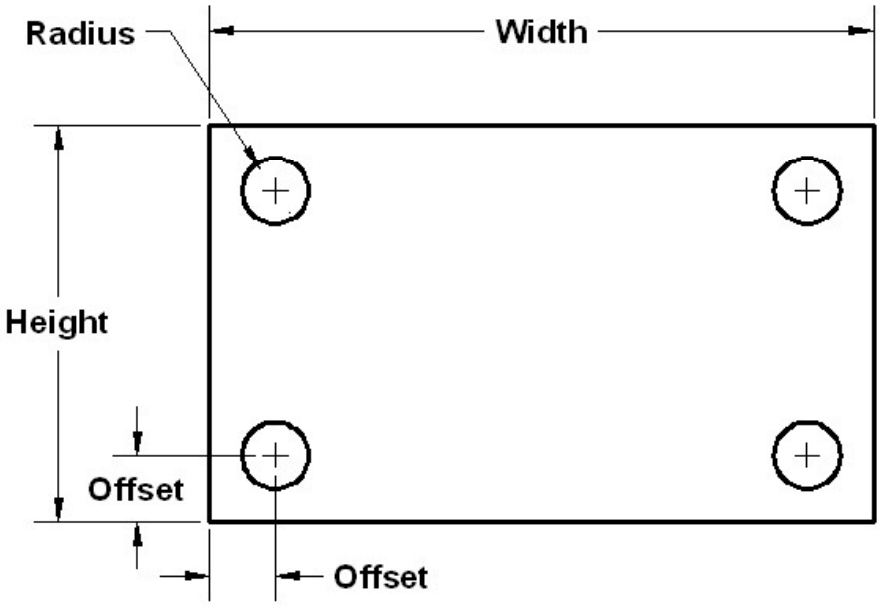

Figure 4.9 – Creating the Stamping with 4 Holes Form Image in AutoCAD

On the control toolbox, select the Image tool and then draw a rectangular box on the form in the upper right corner as shown in Figure 4.11. After outlining the size of the image, we will direct the program to the folder and filename of the digital image. In the Properties – Image pane, select the attribute named Picture. With the mouse, select the three dot box in the empty cell to the right of Picture. The Load Picture window appears on the screen. Go to the VBA Programs folder and then the Images folder. Select the file, Stamping with four holes and it will appear in the picture frame.

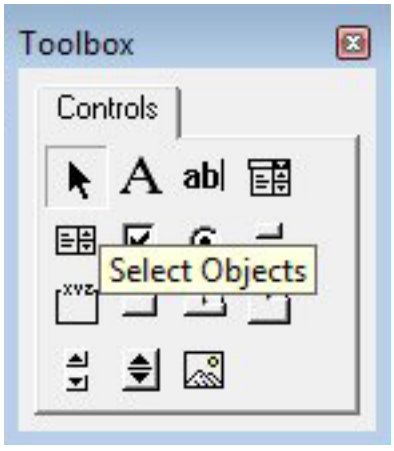

Figure 4.10 – The Control Toolbox

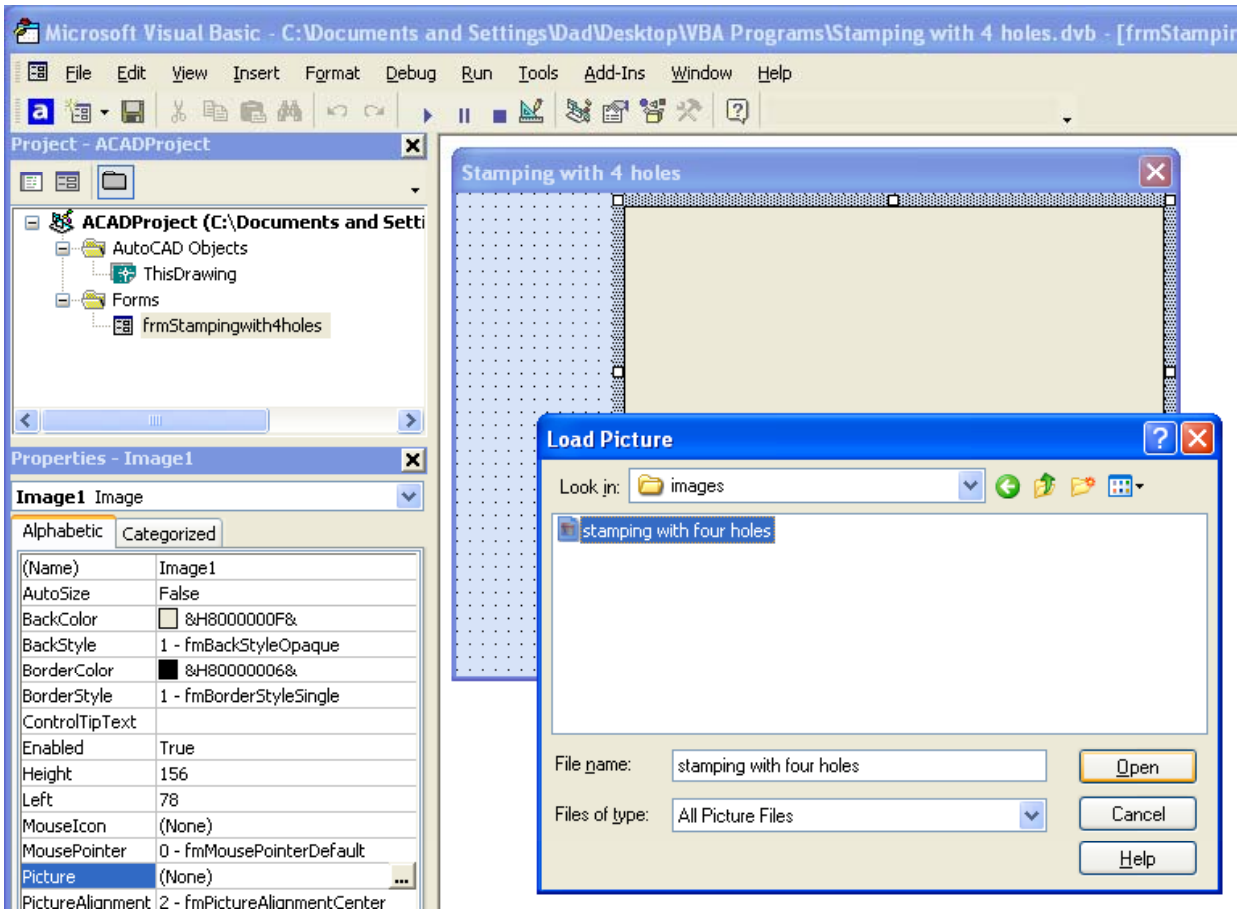

Figure 4.11 – Placing an Image on the Form

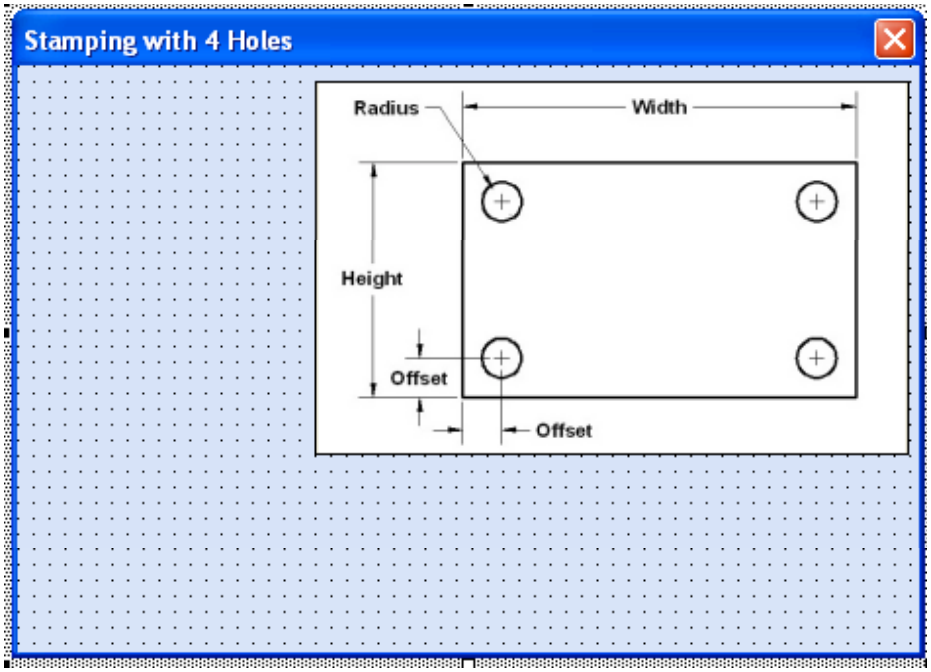

Figure 4.12 – Placing an Image on the Form

Inserting a Label into a Form

A good form is easy to figure out by the user, so when we are attempting to provide information on the window that will run in AutoCAD; we add labels to textboxes to explain our intent. Press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box as shown in the sketch.

We will name the Label using a common Visual Basic naming convention where the programming object is a three letter prefix followed by the name or phrase of the tool. For our first label on this form, the name is **lblStartingPoint.**

Alphabetic	
(Name)	lblStartingPoint
BackColor	&H80000013&
Caption	Type your name:
Font	Arial

On the sketch, the label's caption is "**Starting Point:**" The font on the sketch is 10 point, Arial. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Tahoma. Click on the three dotted button to open the Visual Basic Font window.

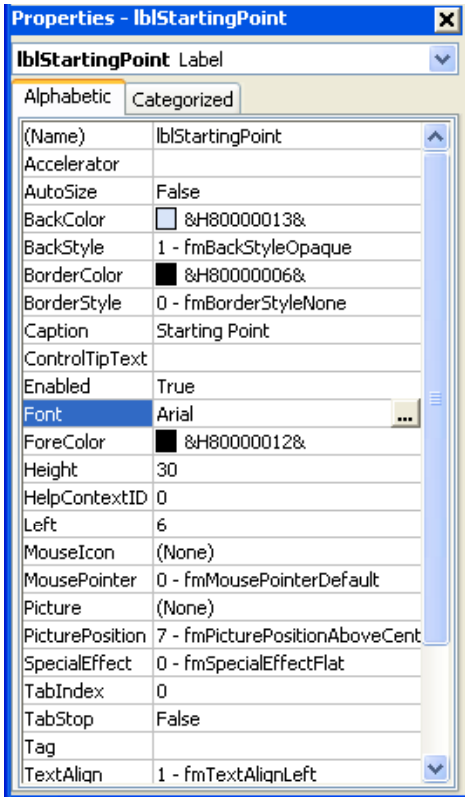

Figure 4.13 – Changing the Font to Arial

We will select the Arial font, Regular font style and 10 size for this project to agree with the initial sketch if the user input form. When we adjust the attributes for the label, these changes do not alter globally for the other objects on the form. If we wish to underline the text or phrase in the label, add a check to the Underline checkbox in the Effects section of the Font window. When we finish making changes to the font property, select the OK command button to return to the work area.

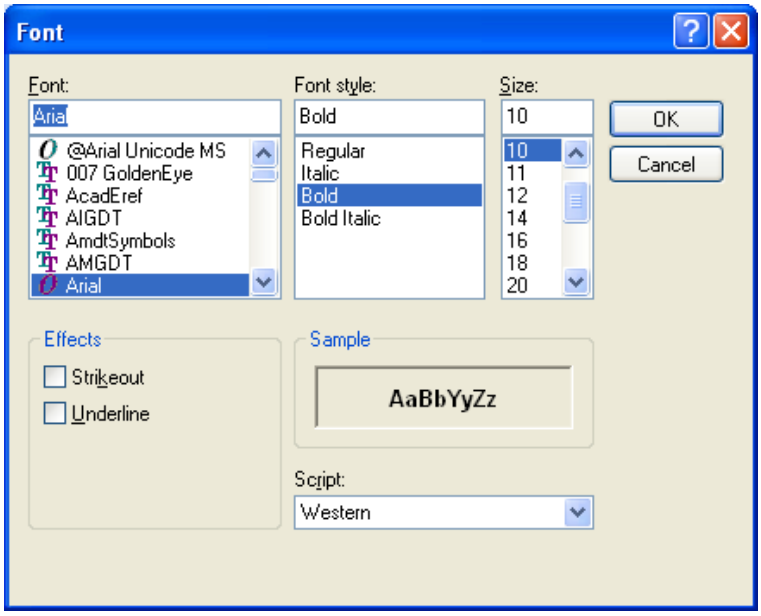

Figure 4.14 – The Font Window in Visual Basic

When the first label is done, the background color of the label matches the background color of the form. In many cases that effect is visually pleasing to the eye, versus introducing another color. Both color and shape will direct the user in completing the form along with the explanation we place on the window to guide the designer in using the automated programs. Use colors and shape strategically to communicate well.

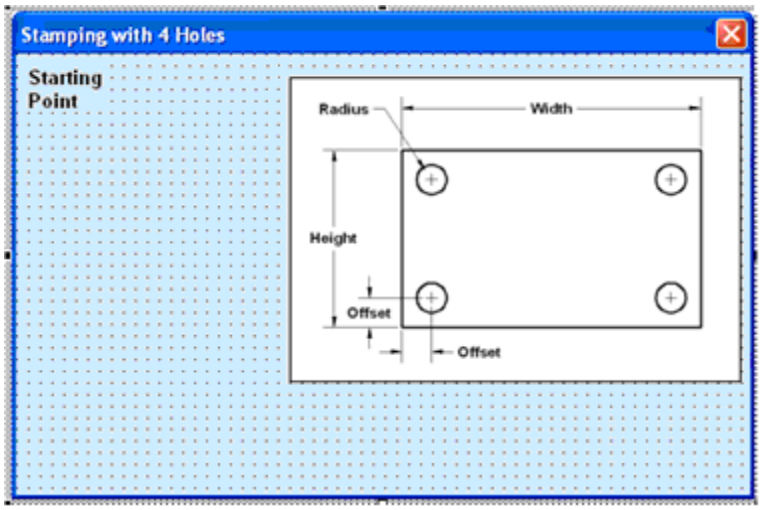

Figure 4.15 – The Finished Label on the Form

Inserting a Textbox into a Form

A textbox is used so that a user of the computer program can input data in the form of words, numbers or a mixture of both. Press the TextBox (ab) button on the Control Toolbar to add a textbox. To size the textbox area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted textbox as shown in Figure 4.15.

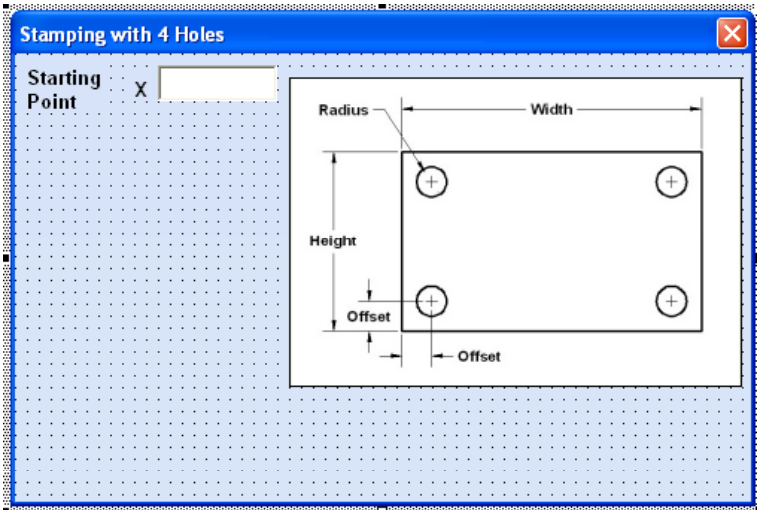

Figure 4.16 – Placing a TextBox on the Form

We will name the TextBox using the three letter prefix followed by the name or phrase of the tool. For our first textbox, the name is **txtXcoord.**

Alphabetic	
(Name)	txtXcoord
Height	18
Width	60

The font on the sketch is 12 point, Arial. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Tahoma. Click on the three dotted button to open the Visual Basic Font window. Make the changes like we did on the Label and press OK to save the property.

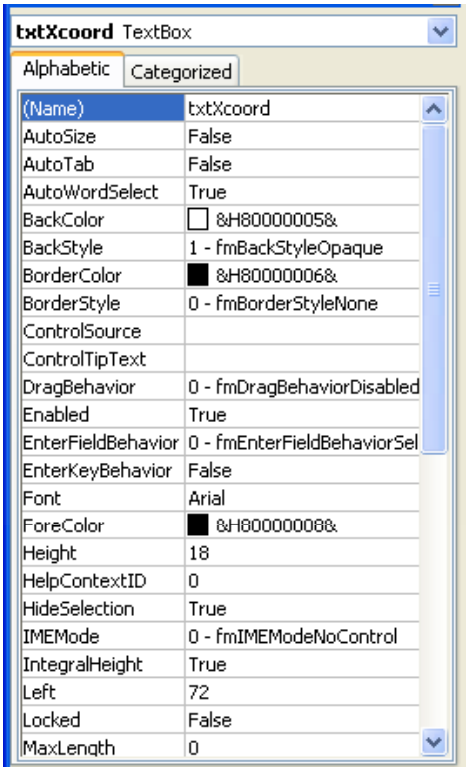

Figure 4.17 – Changing the (Name) to txtName

We place a Label using a common Visual Basic naming convention **lblXcoord** just to the left of the Textbox. The Caption for the Label will be **X.** On all of the labels that are just to the left of the Textboxes, we will align the text to the right by setting the **TextAlign** property to right align.

We will add another TextBox named **txtYcoord** under the first one and the Label to the left of the textbox is called **lblYcoord.** The Caption for the Label will be **Y.**

We will add yet another TextBox named **txtZcoord** under the first one and the Label to the left of the textbox is called **lblZcoord.** The Caption for the Label will be **Z.**

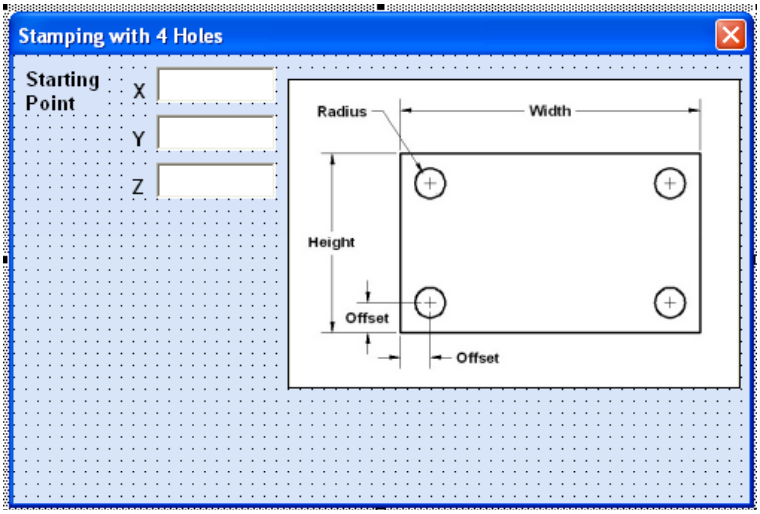

Figure 4.18 – Adding the Y and Z Textboxes

We will add four more textboxes named **Width, Height, Radius** and **Offset** under the X, Y and Z textboxes. The labels to the left of the textbox are called **lblWidth, lblHeight, lblRadius** and **lblOffset.** The Captions for the Labels are shown in Figure 4.19.

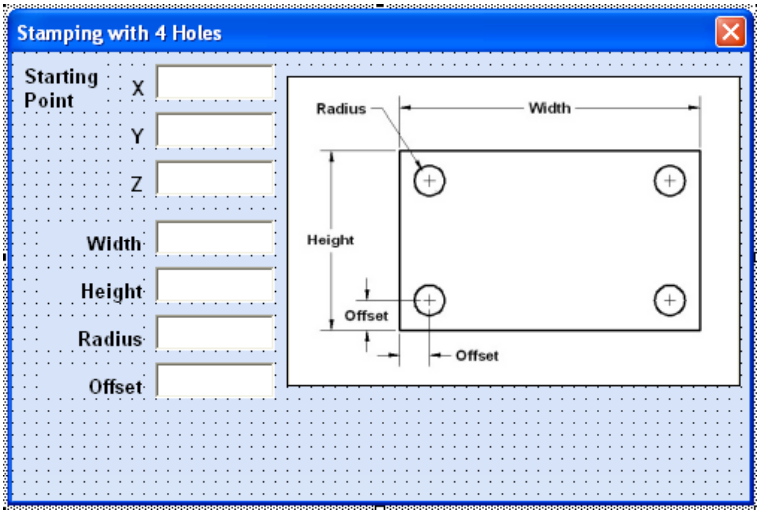

Figure 4.19 – Adding Four More Textboxes

Inserting a Command Buttons into a Form

A command button is used so that a user will execute the application. Press the Command button on the Control Toolbar to add a command button. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the command button as shown in Figure 4.20.

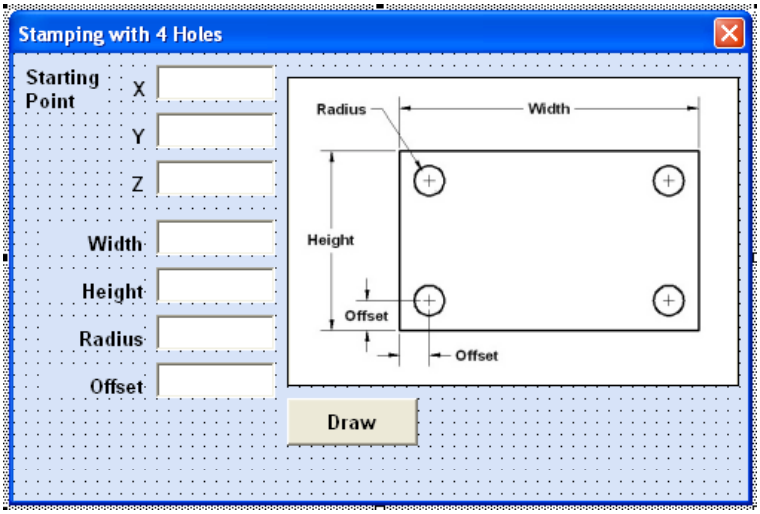

Figure 4.20 – Insert a Command Button onto a Form

We will name the command button using the name is **cmdDraw.**

Alphabetic	
(Name)	cmdDraw
Caption	Draw
Font	Arial
Height	24
Width	66

The font on the sketch is 18 point, Arial. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Tahoma. Click on the three dotted button to open the Visual Basic Font window. Make the changes as we did before and press OK to save the property.

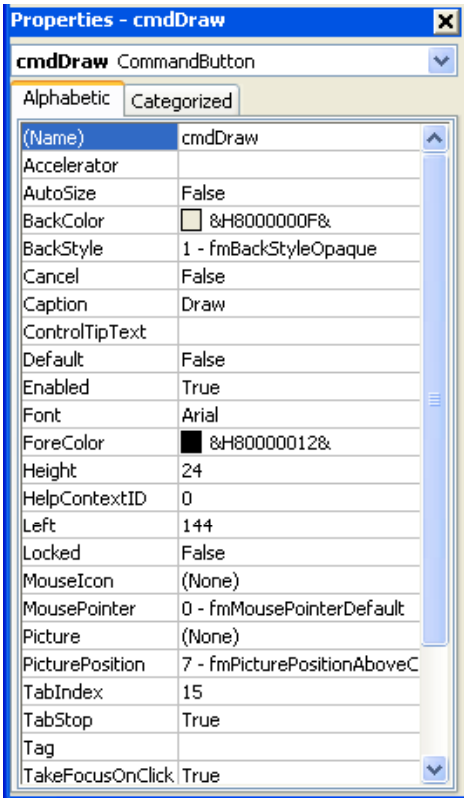

Figure 4.21 – Changing the (Name) to cmdDraw

Add a second Command button; named cmdClear is for clearing the Staring Point, Width, Height, Radius and Offset objects. The third command button is to exit the program. When the user presses the Exit command button, the application closes and full control of the manual AutoCAD program returns to the user. Notice the equal spacing between the command buttons gives a visually friendly appearance.

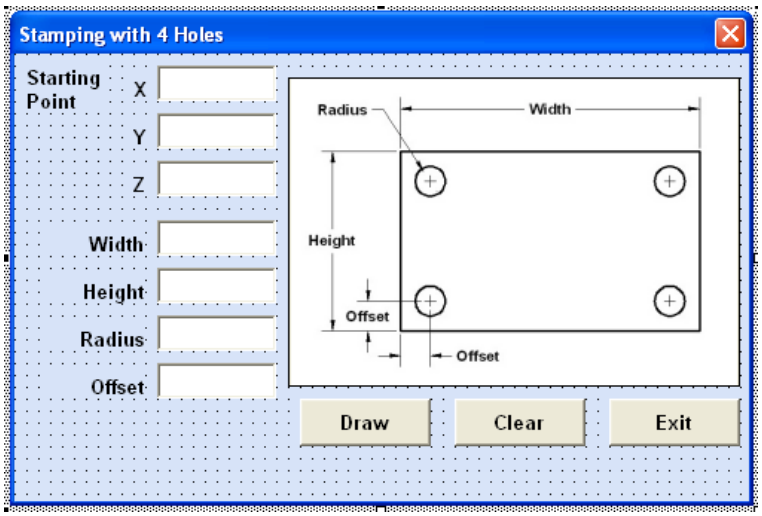

Figure 4.22 – Insert Two More Command Buttons

Adding a Copyright Statement to a Form

At the beginning of a new program, we will expect to see an explanation or any special instructions in the form of comments such as copyright, permissions or other legal notices to inform programmers what are the rules dealing with running the code. Comments at the opening of the code could help an individual determine whether the program is right for their application or is legal to use. The message box is a great tool when properly utilized to inform someone if they are breaking a copyright law when running the code.

Finish the form with the following copyright information.

'Stamping with 4 holes - copyright (c) 2006 by charles robbins

If there are special rules or instructions that the user needs to know, place that information on the bottom of the form.

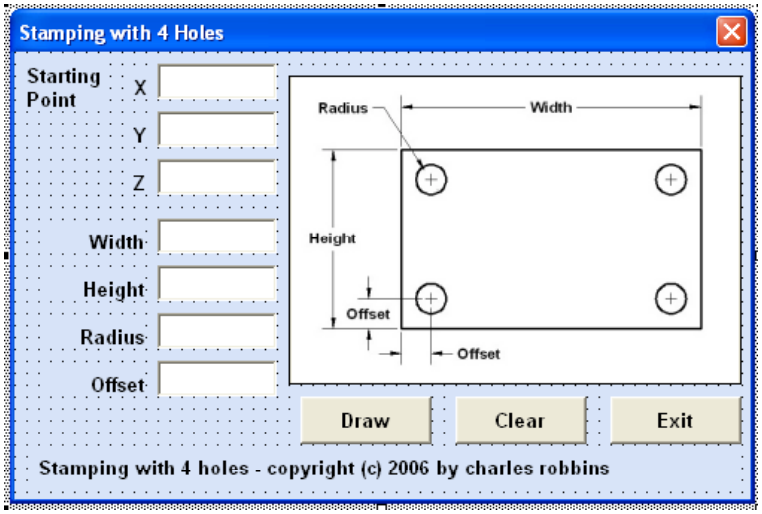

Figure 4.23 – Adding a Copyright Statement

Now that the form is complete, we will begin to write the code that actually interfaces the content of the form using logic and computations to draw the stamping in the AutoCAD graphical display. We will begin the program with comments and place addition phrases throughout the program to assist ourselves or others in the future when modifying the code.

Adding Comments in Visual Basic to Communicate the Copyright

The comments we placed in the first three lines of the program will inform the individual opening and reading the code, but those user that may run the application without checking, the label on the bottom of the form with the copyright information is a great tool to alert the client to the rules of the program and what will the application do.

To begin the actual coding of the program, double click on the Draw command button to enter the programming list. At the top of the program and before the line of code with **Sub CreateStampingwith4Holes ()**, place the following comments with the single quote (‘) character. Remember, the single quote character (‘) will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
Sub CreateStampingwith4Holes ()

'Stamping with 4 holes.dvb copyright (c) 2006 by Charles W. Robbins
'This program will open a dialogue box in AutoCAD, allow the user to enter a starting point (x, y z)
'Width, Height, Radius and Offset and then draw a four holed stamping
```

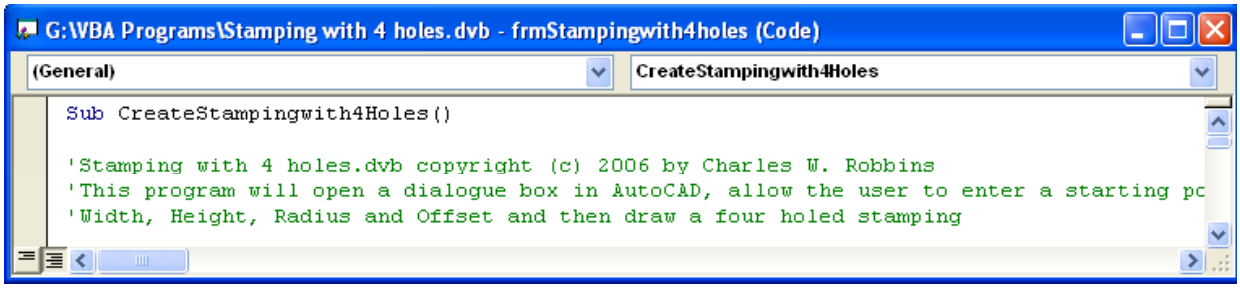

Figure 4.24 – Adding Comments into the Code

Declaring Variables in a Program with the Dimension Statement

When we are going to use a number, text string or object that may change throughout the life of the code, we create a variable to hold the value of that changing entity. In Visual Basic, the dimension or dim statement is one of the ways to declare a variable at the script of procedure level. The other two ways are the Private and Public statements, which we will use in later chapters.

Type the following lines of code after the comment.

```
'define the starting and centerpoint arrays, width, height and radius

    Dim CircleObject As AcadCircle
    Dim LineObject As AcadLine
    Dim Startingpoint(0 To 2) As Double
```

```
Dim P2(0 To 2) As Double
Dim P3(0 To 2) As Double
Dim P4(0 To 2) As Double
Dim Center1(0 To 2) As Double
Dim Center2(0 To 2) As Double
Dim Center3(0 To 2) As Double
Dim Center4(0 To 2) As Double
Dim Width As Double
Dim Height As Double
Dim Radius As Double
Dim Offset As Double
```

In our program, we will declare a variable to enable us to draw circles and lines, a variable for each vertex and a variable for the height, width, radius and offset. As we can see below, the made up name CircleObject is an AutoCAD Circle by definition and the contrived name LineObject is a line.

The vertices are declared as double integers (As Double) with an array of zero to two (0 to 2). The vertex StartingPoint(0) represents the X coordinate, the StartingPoint(1) represents the Y coordinate and StartingPoint(2) represents the Z coordinate. Some may think that it is a waste of time to involve the Z-axis in a two dimension drawing, but we will incorporate the Z coordinate for designers that work in all three dimensions. For everyone else, we will just enter zero (0) in the Z coordinate textbox.

Lastly, we declare Width, Height, Radius and Offset as double integers (As Double).

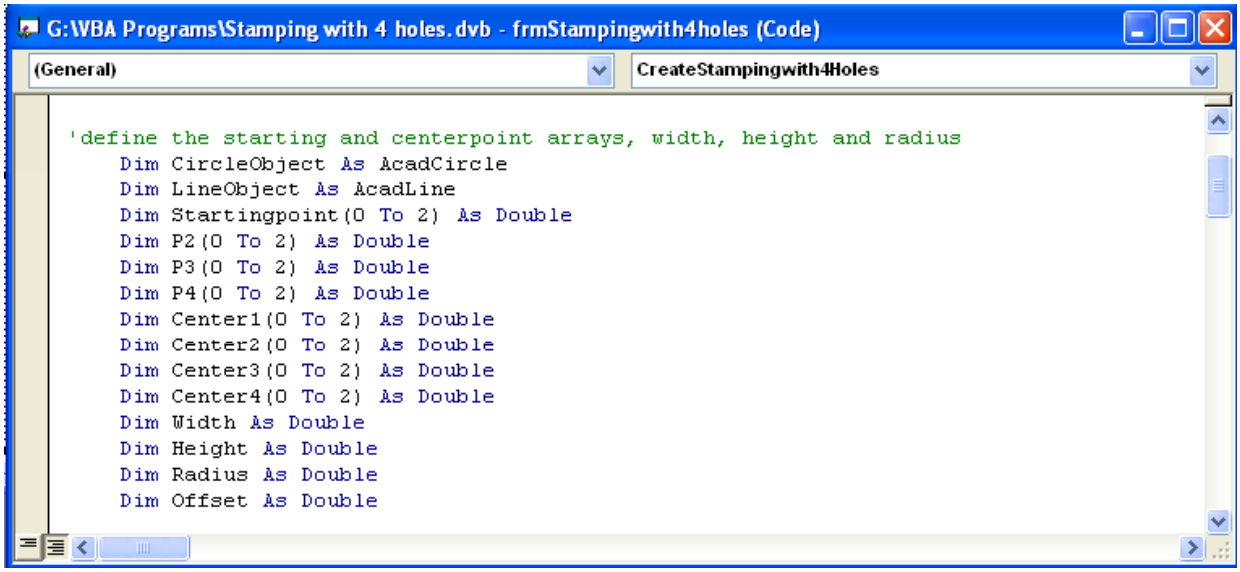

Figure 4.25 – Declaring Variables with Dim Statements

When selecting variable names, they should be a word or a phrase without spaces that represents the value that the variable contains. If we want to hold a value of one's date of birth, we can call the variable, DateofBirth. The keywords Date and Birth are in sentence case with the first letter capitalized. There are no spaces in the name. Some programmers use the

underscore character (_) to separate words in phrases. This is acceptable, but a double underscore (__) can cause errors if we do not detect the repeated character.

Assigning Values to the Variables

After we declare the variables and before we start drawing, we will assign the variables from the input the user types in the textboxes on the launched user form and then assign values to each of the vertices in the set of construction points.

Type the following code right below the declared variables.

```
'assigning values to the variables
   Width = txtWidth
   Height = txtHeight
   Radius = txtRadius
   Offset = txtOffset
   Startingpoint(0) = txtXcoord
   Startingpoint(1) = txtYcoord
   Startingpoint(2) = txtZcoord

'point assignments and math
   Center1(0) = txtStartingpointX + Offset
   Center1(1) = txtStartingpointY + Offset
   Center1(2) = txtStartingpointZ
   Center2(0) = txtStartingpointX + Width - Offset
   Center2(1) = txtStartingpointY + Offset
   Center2(2) = txtStartingpointZ
   Center3(0) = txtStartingpointX + Width - Offset
   Center3(1) = txtStartingpointY + Height - Offset
   Center3(2) = txtStartingpointZ
   Center4(0) = txtStartingpointX + Offset
   Center4(1) = txtStartingpointY + Height - Offset
   Center4(2) = txtStartingpointZ
   P2(0) = Startingpoint(0) + Width
   P2(1) = Startingpoint(1)
   P2(2) = Startingpoint(2)
   P3(0) = Startingpoint(0) + Width
   P3(1) = Startingpoint(1) + Height
   P3(2) = Startingpoint(2)
   P4(0) = Startingpoint(0)
   P4(1) = Startingpoint(1) + Height
   P4(2) = Startingpoint(2)
```

As we can see below, the first seven equal the values form the textbox. After that, we assign each point's X, Y and Z coordinate a number either from the variable or from a mathematical calculation that we arrive from the sketch in Figure 4.26. We use the variables Width, Height and Offset to measure the distance from one point to another.

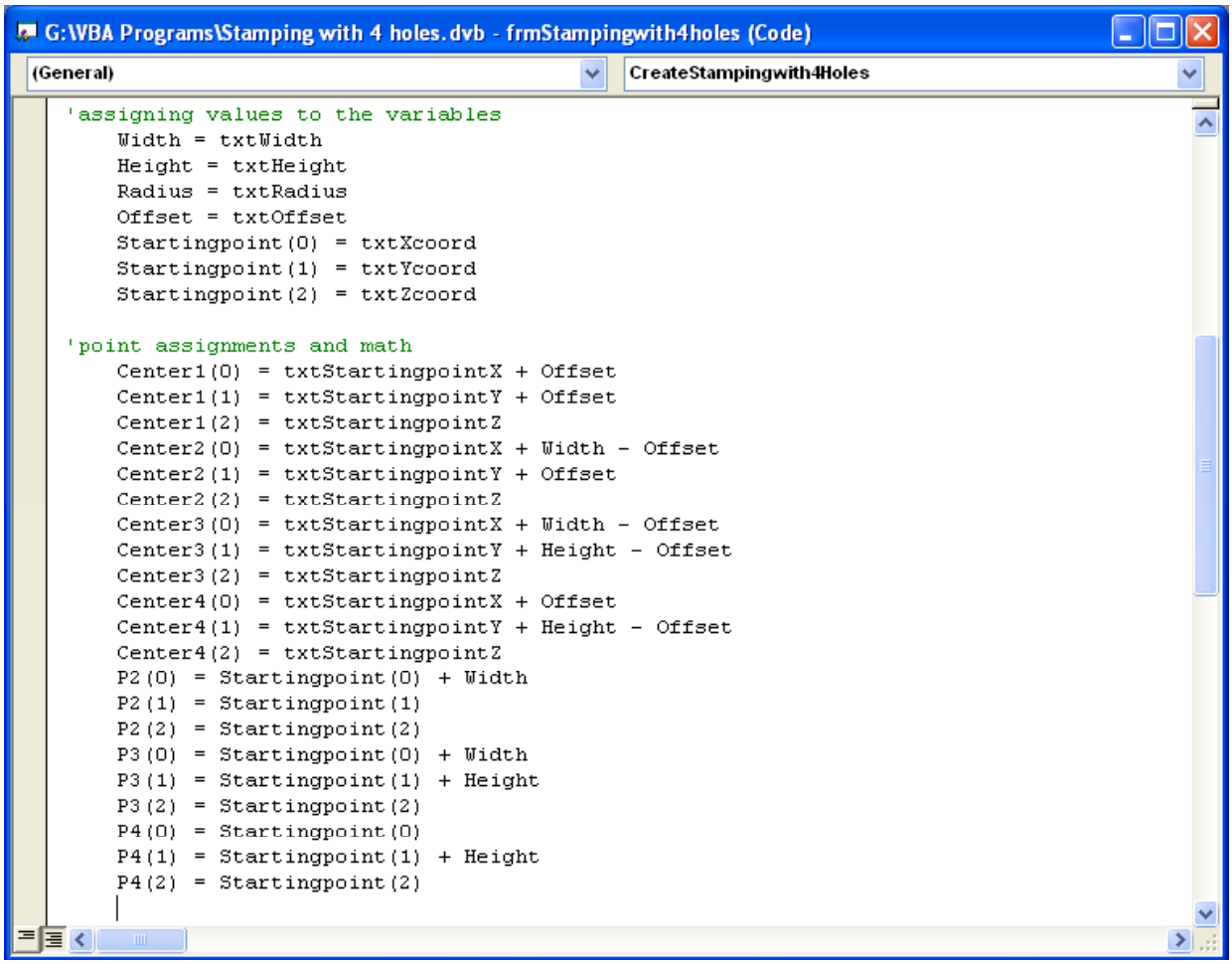

Figure 4.26 – Setting the Variables in the Code

Inputting the Code to Draw in Visual Basic

Now we want to enter the code that will actually draw lines and circles in the AutoCAD Model Space. We use the Set function to draw a line by typing **Set LineObject** and then we tell the computer that it will draw in Modelspace by adding a line from the starting point to point P2.

Go ahead and type the following comments and drawing code:

```
'Execute the stamping with 4 hole
'Draw lines
  Set LineObject = ThisDrawing.ModelSpace.AddLine(Startingpoint, P2)
  Set LineObject = ThisDrawing.ModelSpace.AddLine(P2, P3)
  Set LineObject = ThisDrawing.ModelSpace.AddLine(P3, P4)
  Set LineObject = ThisDrawing.ModelSpace.AddLine(P4, Startingpoint)

'Draw circle
   Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center1, Radius)
```

```
Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center2, Radius)
Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center3, Radius)
Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center4, Radius)
```

We draw three more lines from P2 to P3, P3 to P4 and finally from P4 to the startingpoint.

We use the Set function to draw a circle by typing **Set CirlceObject** and then we tell the computer that it will draw in Modelspace by adding a circle from the center point number 1 with a radius that contains the value from the radius textbox. Then we draw three more circles with the radius at center point 2, center point 3, and center point 4.

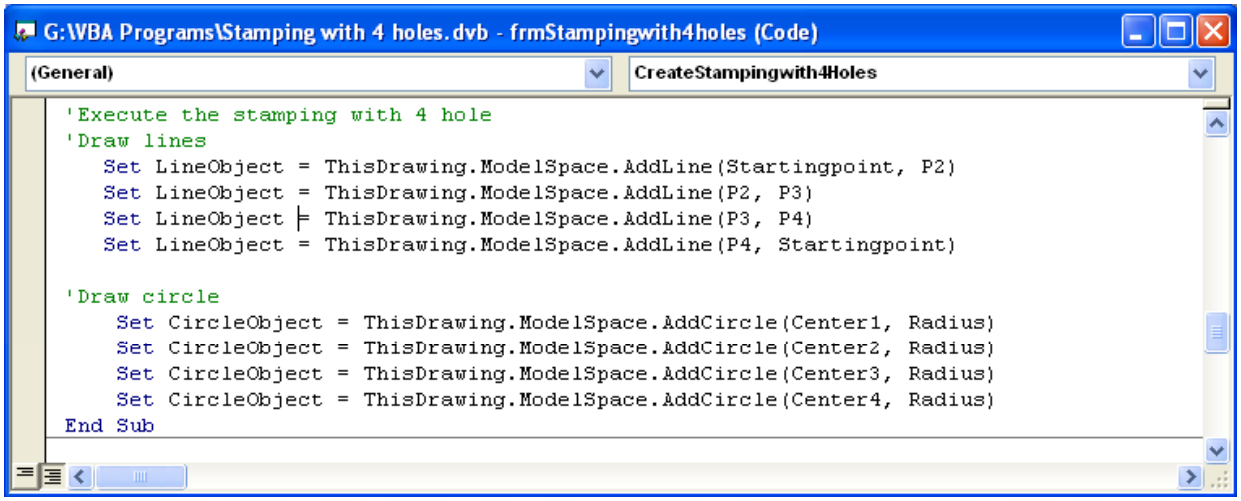

Figure 4.27 – Drawing the Lines and Circles with the Code

To end this Visual Basic subroutine, we will type a comment saying so. In the future, this will be more elaborate, but for now we will just get used to announcing the natural divisions of the script.

Type the following code:

```
'End of program
End Sub
```

Resetting the Data with the cmdClear Command Button

To clear the textboxes containing the user input, we will first set the textbox for txtXcoord, txtXcoord.text property to a "0.00" entry by using the equal sign "=".This makes the property equal zero as a default. We do this also for the Y and Z coordinates. We will set the textboxes for txtWidth, txtWidth.text property to a black entry by using the equal sign "=" and the null string "", and this will make that property blank. Notice that after the control object name the dot (.) separates the suffix which is the name of the property for that object.

Key the following code as a new subroutine **Private Sub cmdClear_Click().**

```
Private Sub cmdClear_Click()
'clear the form
   txtXcoord = "0.00"
   txtYcoord = "0.00"
   txtZcoord = "0.00"
   txtWidth = ""
   txtHeight = ""
   txtRadius = ""
   txtOffset = ""
End Sub
```

Figure 4.28 – Computing the Reset Button by Clearing Textboxes

Exiting the Program with the cmdExit Command Button

To exit this program, we will unload the application and end the program.
Type the following code:

```
Private Sub cmdExit_Click()
'unload and end program
   Unload Me
   End
End Sub
```

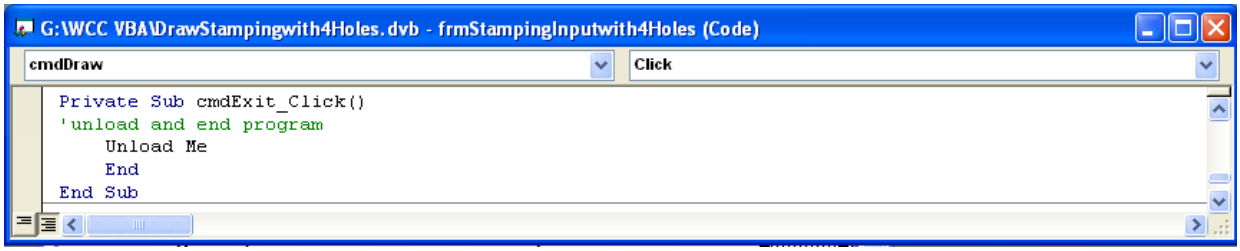

Figure 4.29 – Coding the Exit Button

Executing a Subroutine with the cmdDraw Command Button

In this program, we use a subroutine which is executed by the Draw command button, so type the following code to execute the subroutine, **CreateStampingwith4Holes**

```
Private Sub cmdDraw_Click()
'draw the stamping
   CreateStampingwith4Holes
End Sub
```

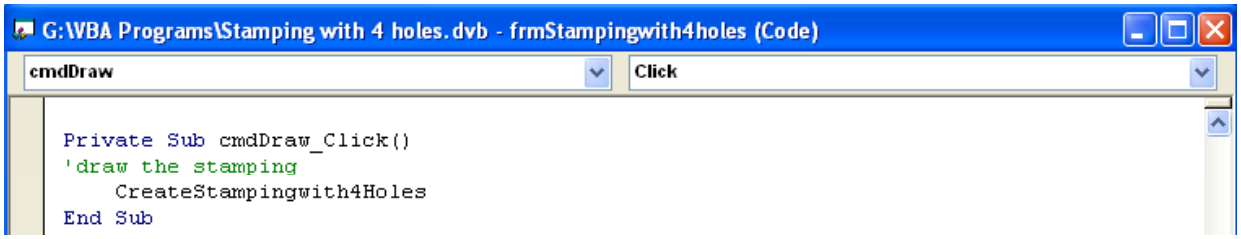

Figure 4.30 – Coding the Draw Button

Written below is the entire program for creating the Stamping with 4 holes. Next, we will insert a module to launch the form.

```
Sub CreateStampingwith4Holes()
'define the starting and centerpoint arrays, width, height and radius
   Dim CircleObject As AcadCircle
   Dim LineObject As AcadLine
   Dim Startingpoint(0 To 2) As Double
   Dim P2(0 To 2) As Double
   Dim P3(0 To 2) As Double
   Dim P4(0 To 2) As Double
   Dim Center1(0 To 2) As Double
   Dim Center2(0 To 2) As Double
   Dim Center3(0 To 2) As Double
   Dim Center4(0 To 2) As Double
   Dim Width As Double
   Dim Height As Double
   Dim Radius As Double
   Dim Offset As Double

'assigning values to the variables
   Width = txtWidth
   Height = txtHeight
   Radius = txtRadius
   Offset = txtOffset
   Startingpoint(0) = txtXcoord
   Startingpoint(1) = txtYcoord
   Startingpoint(2) = txtZcoord
```

```
'point assignments and math
   Center1(0) = txtStartingpointX + Offset
   Center1(1) = txtStartingpointY + Offset
   Center1(2) = txtStartingpointZ
   Center2(0) = txtStartingpointX + Width - Offset
   Center2(1) = txtStartingpointY + Offset
   Center2(2) = txtStartingpointZ
   Center3(0) = txtStartingpointX + Width - Offset
   Center3(1) = txtStartingpointY + Height - Offset
   Center3(2) = txtStartingpointZ
   Center4(0) = txtStartingpointX + Offset
   Center4(1) = txtStartingpointY + Height - Offset
   Center4(2) = txtStartingpointZ
   P2(0) = Startingpoint(0) + Width
   P2(1) = Startingpoint(1)
   P2(2) = Startingpoint(2)
   P3(0) = Startingpoint(0) + Width
   P3(1) = Startingpoint(1) + Height
   P3(2) = Startingpoint(2)
   P4(0) = Startingpoint(0)
   P4(1) = Startingpoint(1) + Height
   P4(2) = Startingpoint(2)

'Execute the stamping with 4 hole
'Draw lines
  Set LineObject = ThisDrawing.ModelSpace.AddLine(Startingpoint, P2)
  Set LineObject = ThisDrawing.ModelSpace.AddLine(P2, P3)
  Set LineObject = ThisDrawing.ModelSpace.AddLine(P3, P4)
  Set LineObject = ThisDrawing.ModelSpace.AddLine(P4, Startingpoint)

'Draw circle
   Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center1, Radius)
   Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center2, Radius)
   Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center3, Radius)
   Set CircleObject = ThisDrawing.ModelSpace.AddCircle(Center4, Radius)
End Sub

Private Sub cmdClear_Click()
'clear the form
   txtXcoord = "0.00"
   txtYcoord = "0.00"
   txtZcoord = "0.00"
   txtWidth = ""
   txtHeight = ""
   txtRadius = ""
   txtOffset = ""
End Sub
```

```
Private Sub cmdDraw_Click()
'draw the stamping
   CreateStampingwith4Holes
End Sub

Private Sub cmdExit_Click()
'unload and end program
   Unload Me
   End
End Sub
```

Inserting a Module into a Visual Basic Application

Insert a Module by selecting Insert on the Menu Bar and select Module as shown in Figure 4.31. In the Project Menu, double click on the Module and type the following code.

```
Sub Drawstampingwithhole ()
'draw the stamping
   frmStampingwith4Holes.Show
End Sub
```

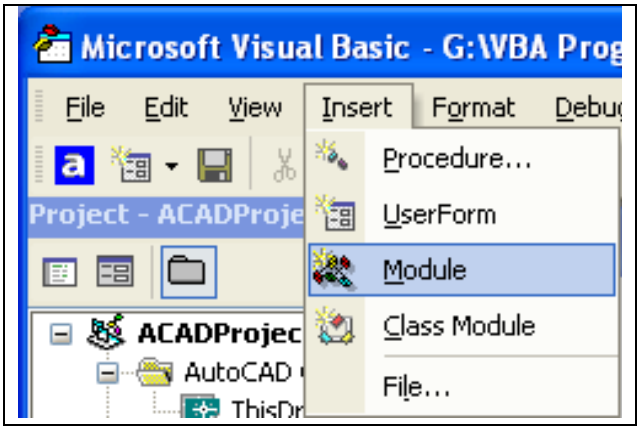

Figure 4.31 – Inserting a Module

The line of code, **frmStampingwith4Holes.Show** will display the form at the beginning of the program.

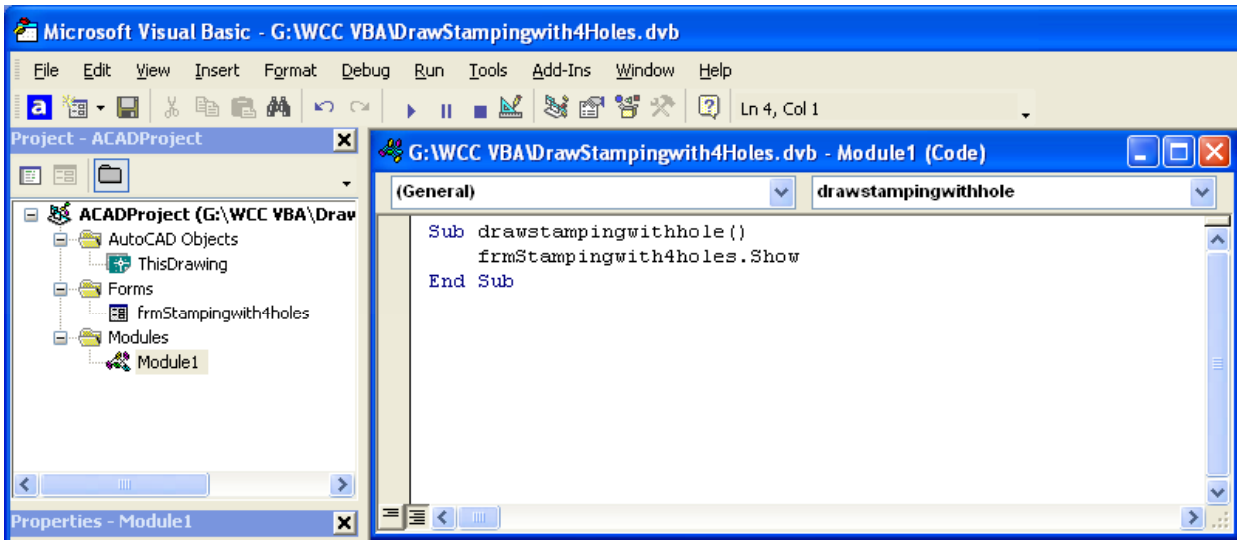

Figure 4.32 – Coding the Module

Running the Program

After noting that the program is saved, press the F5 to run the Stamping with 4 Holes application. Stamping with 4 Holes window will appear on the graphical display in AutoCAD as shown in Figure 4.33.

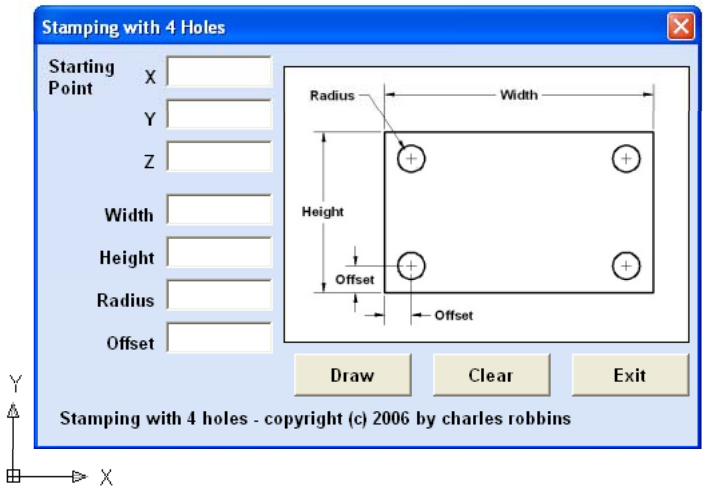

Figure 4.33 – Launching the Program

Type the following data or something similar into the textboxes and select the Draw Command Button to execute the program.

X	0
Y	0
Z	0
Width	6
Height	4
Radius	0.25
Offset	0.5

Figure 4.34 – Input Data

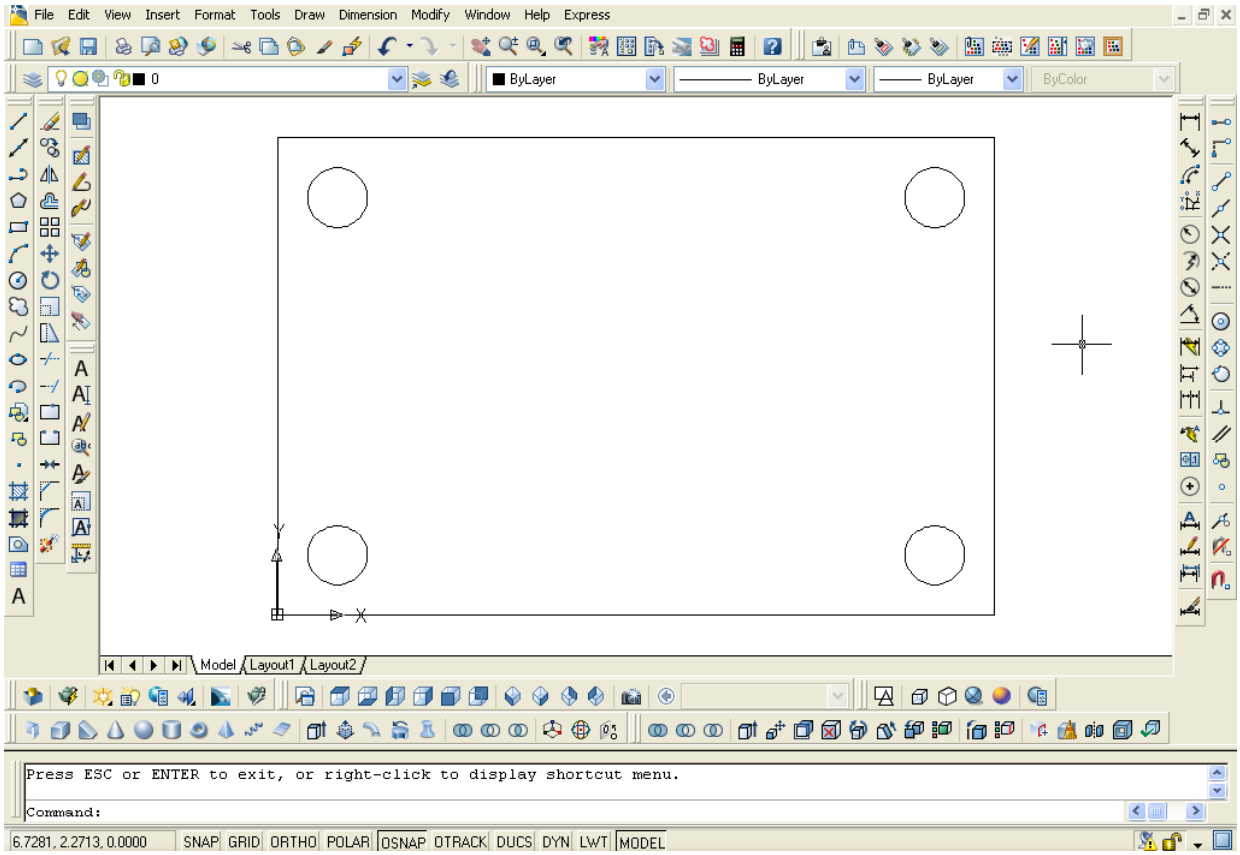

Figure 4.35 – The Finished Draw

There are many variations of this Visual Basic Application we can practice and draw many single view orthographic drawings. While we are practicing with forms, we can learn how to use variables, make point assignments and draw just about anything we desire. These are skills that we want to commit to memory.

*** World Class CAD Challenge 5-2 * - Write a Visual Basic Application that draws a rectangle with four holes and is executed by a inputting data in a form. Complete the program in less than 120 minutes to maintain your World Class ranking.**

Continue this drill four times making other shapes and simple orthographic views with lines and circles, each time completing the Visual Basic Application in less than 120 minutes to maintain your World Class ranking.