Appendix C

Drawing an Integrated Circuit Chip

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**
- **Automatically Selecting and Arraying Drawing Entities**
- **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 cmdPickPoint 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 continue to learn how to use the Visual Basic Application (VBA) program to create a form and then to generate a drawing automatically. We reiterate many elements of the previous lesson, but now we have 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.

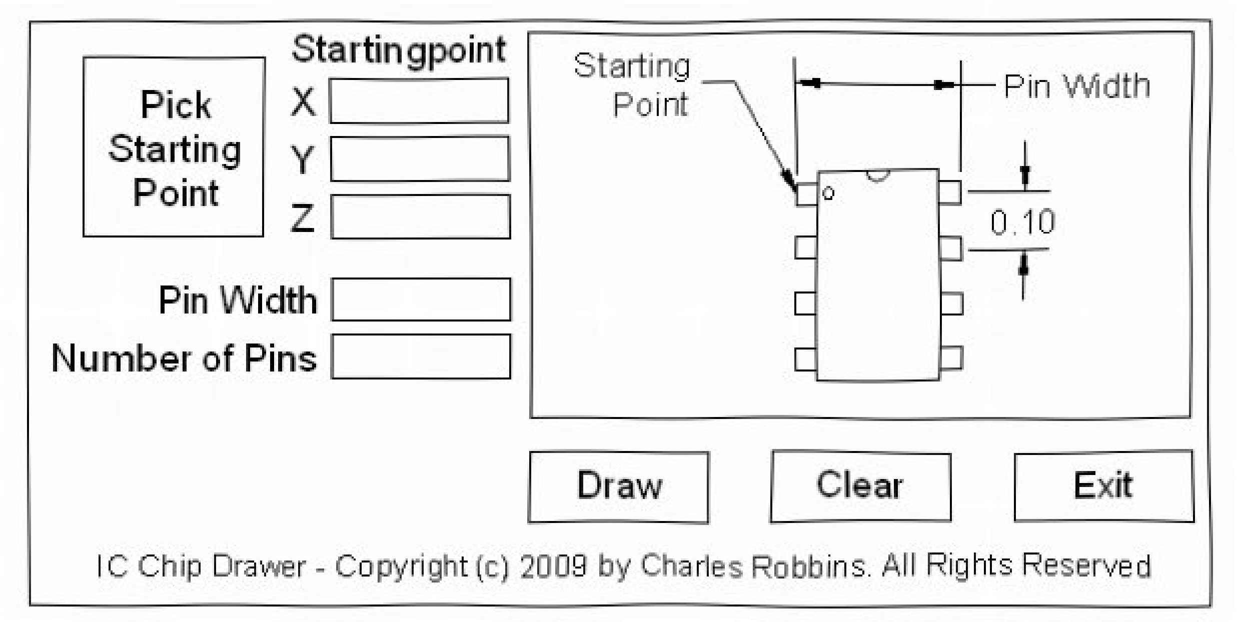


Figure C.1 – Rough Sketch of the Integrated Circuit Chip Form

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, Integrated Circuit Chip, 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, **Integrated Circuit Chip**. We will place five textboxes on the left side of the form to input the starting point, the pin width and the number of pins. On the right side of the form, we will place an image of the chip. 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 C.1.

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 2009, 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.

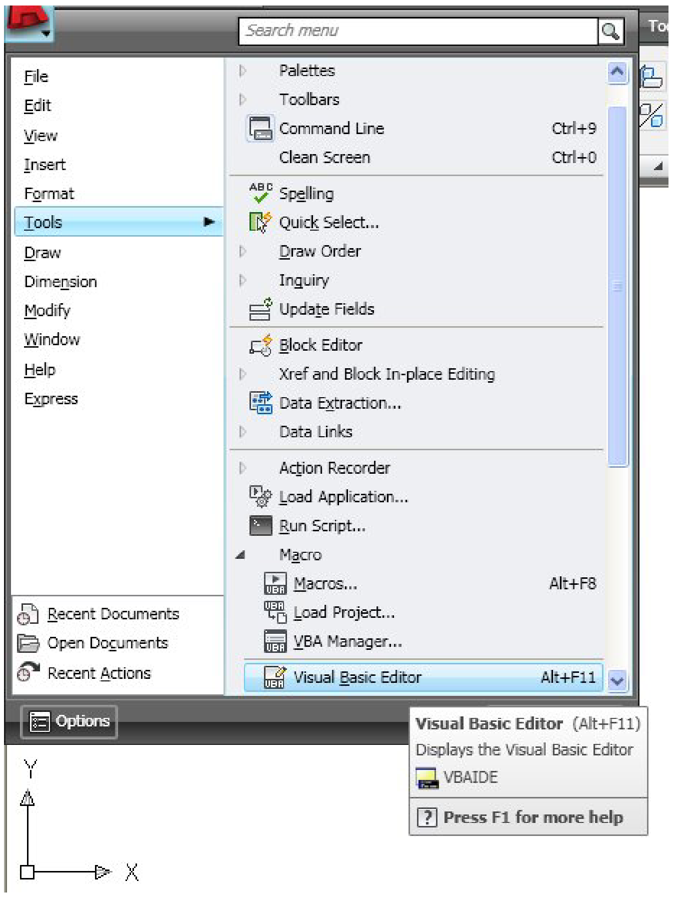


Figure C.2 – Launching the Visual Basic Editor

Select Tools on the AutoCAD main menu; 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 C.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.

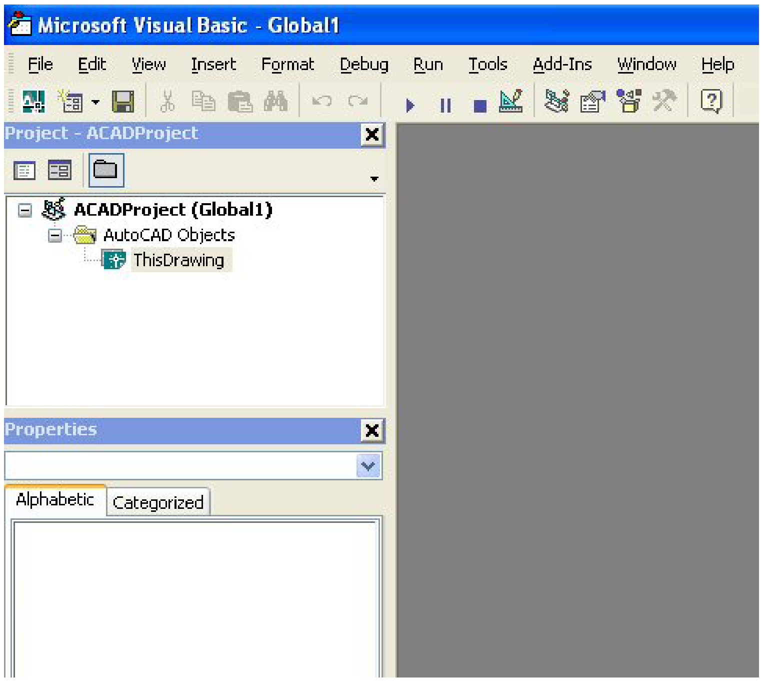


Figure C.3 – The Visual Basic Editor

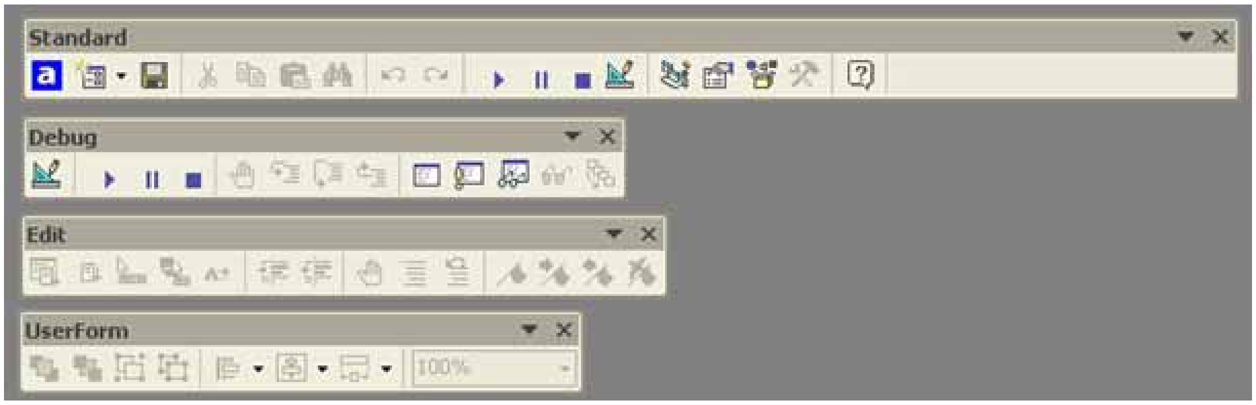


Figure C.4 – Toolbars in the Visual Basic Editor

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 "IC_chip". The file has an extension called *dvb* which means DCL and Visual Basic programs as shown in Figure C.5.

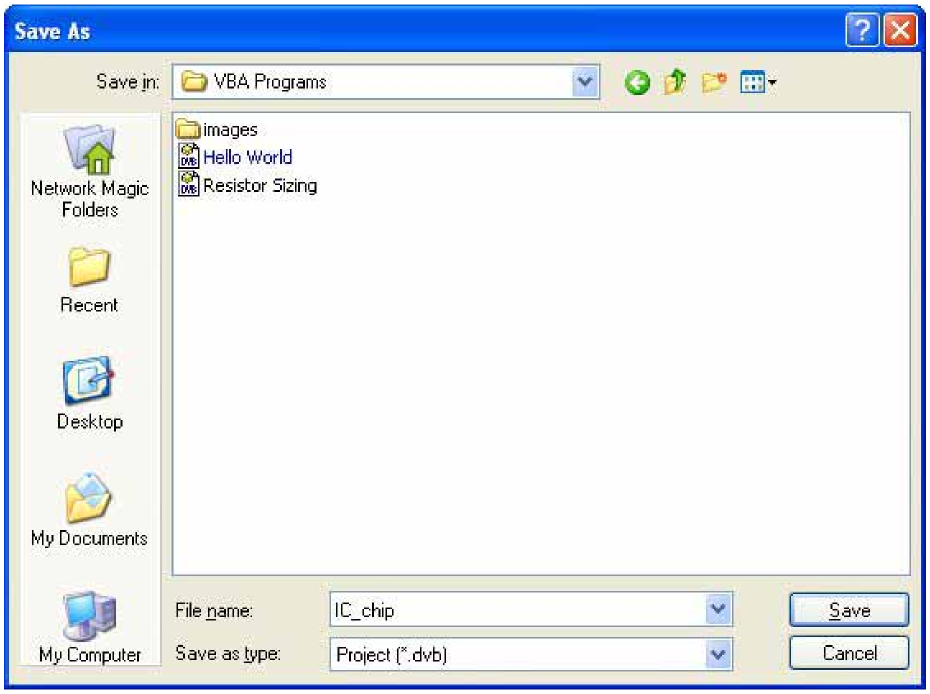


Figure C.5 – Saving the Integrated Circuit Chip Program

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 C.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 C.7)

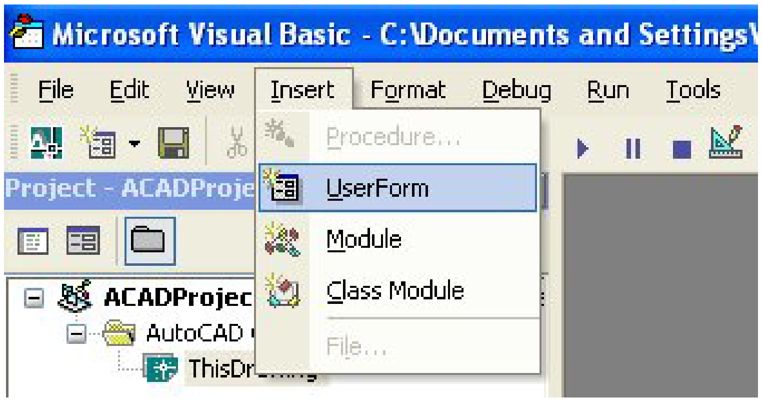
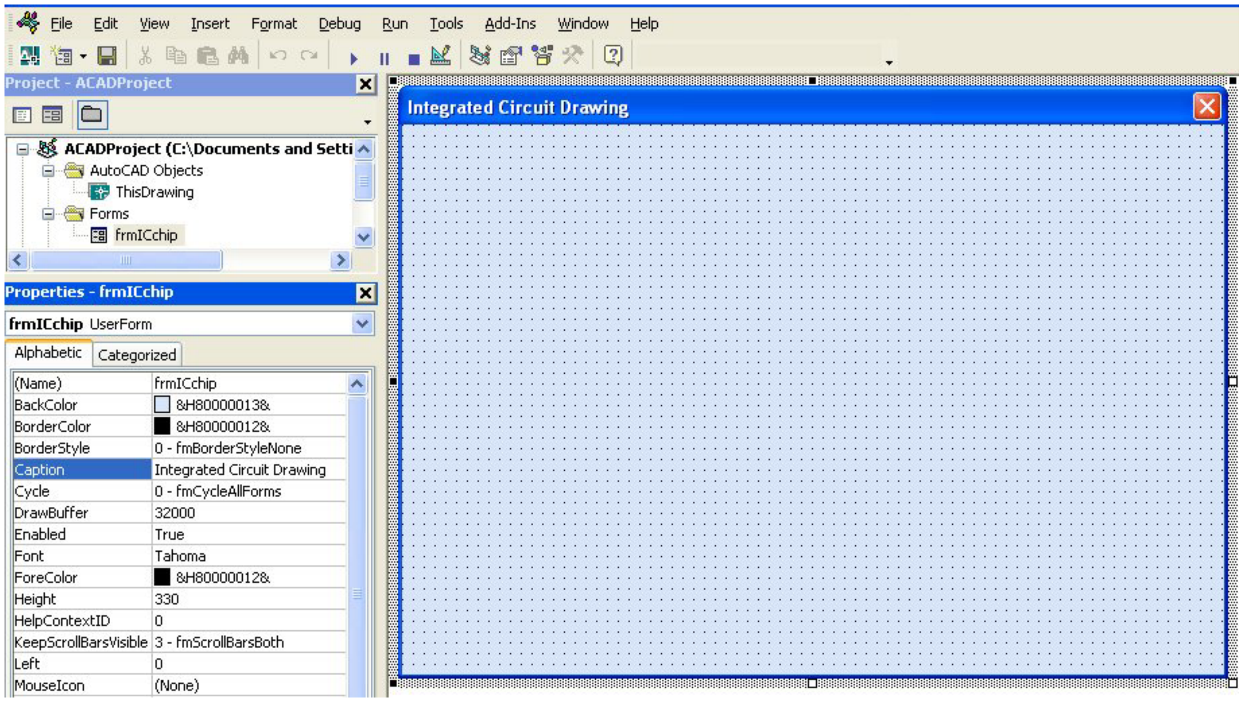


Figure C.6 – Inserting a User Form

Change the name of the user form to frmICchip. We use the frm prefix in front of all of the form names in Visual Basic. Change the background of the form to light blue by setting the BackColor in the Properties Pane on the left side of the Visual Basic Application window to "&H80000013&".

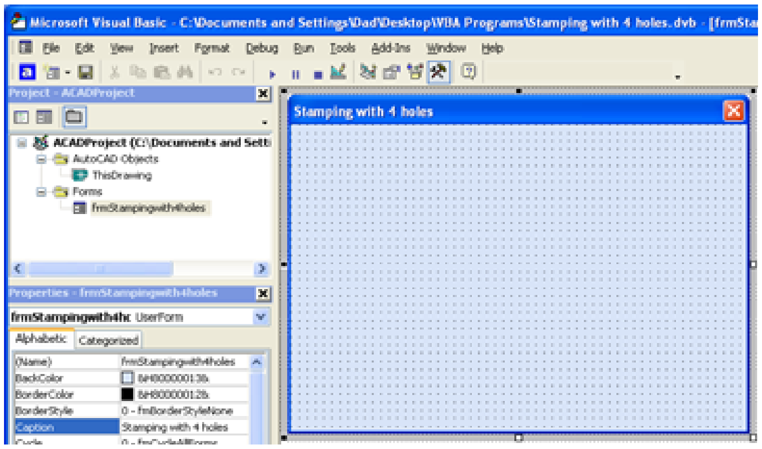

File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - ACADProject
ACADProject (C:\Documents and Setti
AutoCAD Objects
ThisDrawing
Forms
frmICchip
Properties - frmICchip
frmICchip UserForm
Alphabetic
Categorized
Integrated Circuit Drawing

(Name)	frmICchip
BackColor	&H80000013&
BorderColor	&H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	Integrated Circuit Drawing
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	&H80000012&
Height	330
HelpContextID	0
KeepScrollBarsVisible	3 - fmScrollBarsBoth
Left	0
MouseIcon	(None)

Figure C.7 – Designing the Integrated Circuit Chip Form in Visual Basic

Next, we will change the **Caption** in the Properties pane to **Integrated Circuit Drawing** to agree with the sketch in Figure C.8. Go ahead and change the form in two other aspects, Height and Width.

Alphabetic	
(Name)	frmICcip
BackColor	&H80000013&
Caption	Integrated Circuit Drawing
Height	300
Width	470


Microsoft Visual Basic - C:\Documents and Settings\Dad\Desktop\VBA Programs\Stamping with 4 holes.dvb - [frmSta
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - ACADProject
ACADProject (C:\Documents and Setti
AutoCAD Objects
ThisDrawing
Forms
frmStampingwith4holes
Properties - frmStampingwith4holes
frmStampingwith4hc UserForm
Alphabetic
Categorized
Stamping with 4 holes

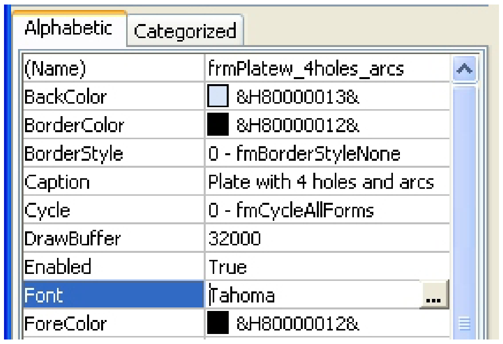
(Name)	frmStampingwith4holes
BackColor	&H80000013&
BorderColor	&H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	Stamping with 4 holes
Cycle	0 - fmCycleAllForms

Figure C.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.

In previous chapters, we set the Font and Font size for the labels, textboxes and command buttons after creating those specific interfaces. If we set the Font to Tahoma and the Font size to 14 on the form, then all of the labels, textboxes and command buttons that we insert from the Control Toolbox will already be set to those attributes.

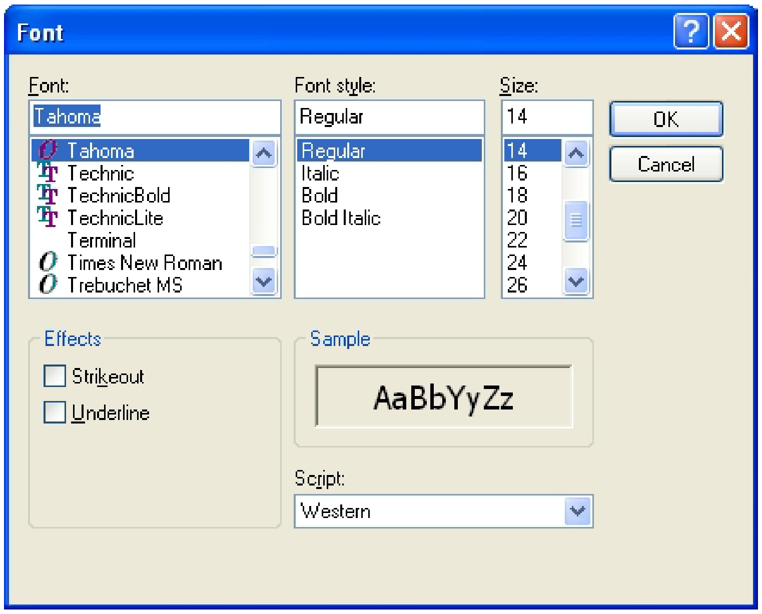
On the left side of the Visual Basic Editor, locate the property that controls the font and font size in the Properties window. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Tahoma. Click on the three dotted button to open the Visual Basic Font window.



(Name)	frmPlatew_4holes_arcs
BackColor	&H80000013&
BorderColor	&H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	Plate with 4 holes and arcs
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	&H80000012&

Figure C.9 – Changing the Font to Tahoma

We will select the Tahoma font, Regular font style and 14 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.



Font: Tahoma
Tahoma
Technic
TechnicBold
TechnicLite
Terminal
Times New Roman
Trebuchet MS

Font style: Regular
Regular
Italic
Bold
Bold Italic

Size: 14
14
16
18
20
22
24
26

Script: Western

Figure C.10 – The Font Window in Visual Basic

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 C.11, we show dimensions that associate the Pin Width, Number of Pins and Startingpoint textboxes with the image. 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, ic_chip. We saved the file as a Bitmap with a width of 340 pixels and a height of 200 pixels.

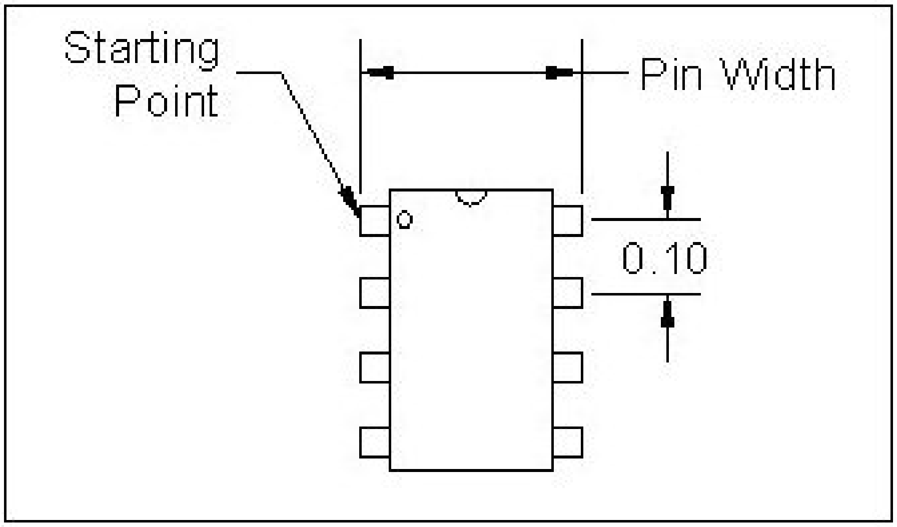


Figure C.11 – Creating the Integrated Circuit Chip Form Image in AutoCAD

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 C.13. 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, ic_chip and it will appear in the picture frame.

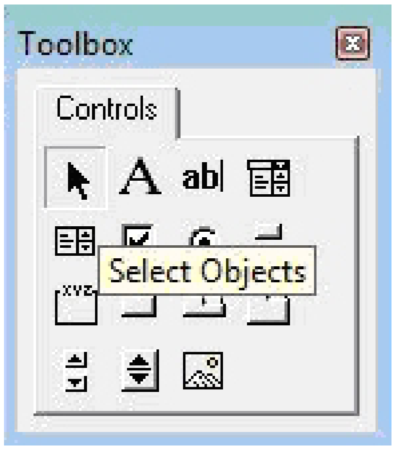
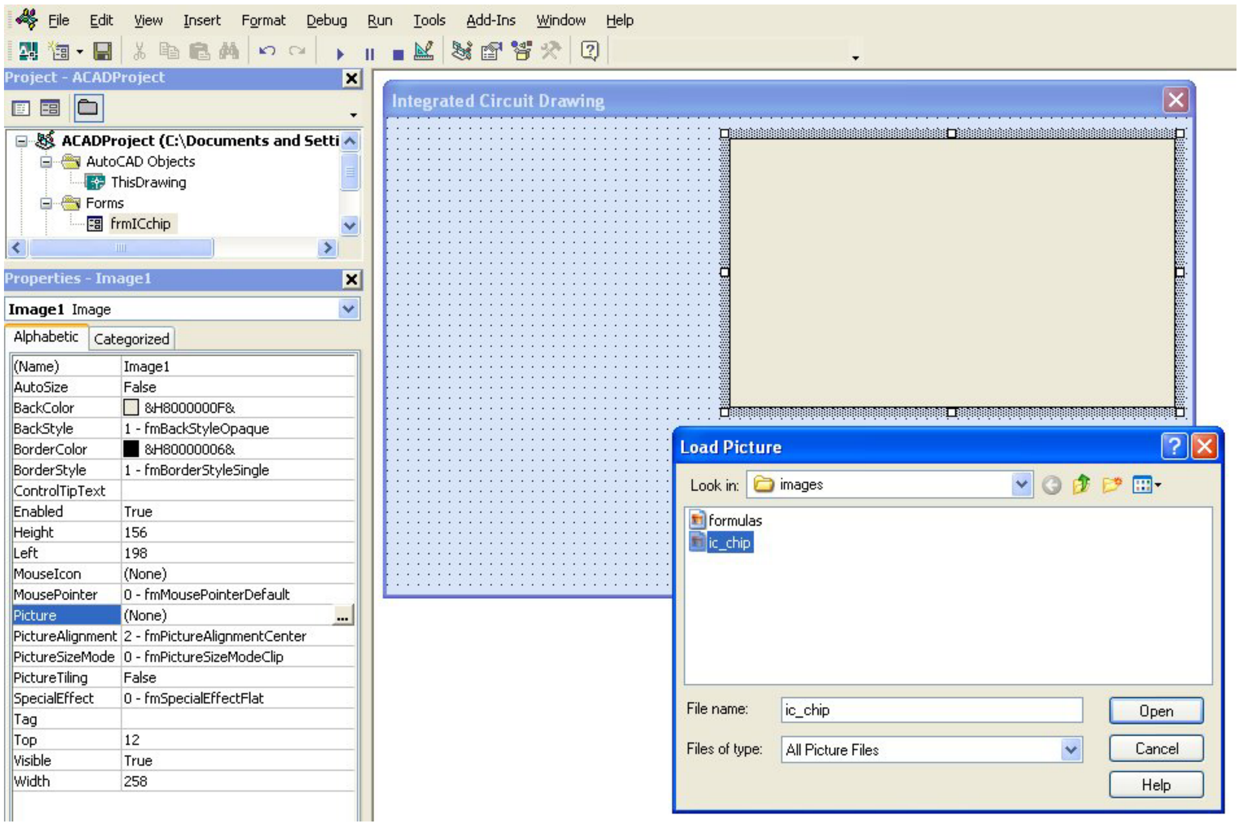


Figure C.12 – The Control Toolbox

In the Properties pane set the image name to imgChip, the width to 340 and the height to 200. The image will finally appear as shown in Figure C.14.


File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project - ACADProject
ACADProject (C:\Documents and Setti
AutoCAD Objects
ThisDrawing
Forms
frmICchip
Integrated Circuit Drawing
Load Picture
Look in: images
formulas
ic_chip
Open
Cancel
Help

Properties - Image1

Image1 Image

Alphabetic | Categorized

(Name)	Image1
AutoSize	False
BackColor	&H8000000F&
BackStyle	1 - fmBackStyleOpaque
BorderColor	&H80000006&
BorderStyle	1 - fmBorderStyleSingle
ControlTipText	
Enabled	True
Height	156
Left	198
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)
PictureAlignment	2 - fmPictureAlignmentCenter
PictureSizeMode	0 - fmPictureSizeModeClip
PictureTiling	False
SpecialEffect	0 - fmSpecialEffectFlat
Tag	
Top	12
Visible	True
Width	258

File name: ic_chip

Files of type: All Picture Files

Figure C.13 – Placing an Image on the Form

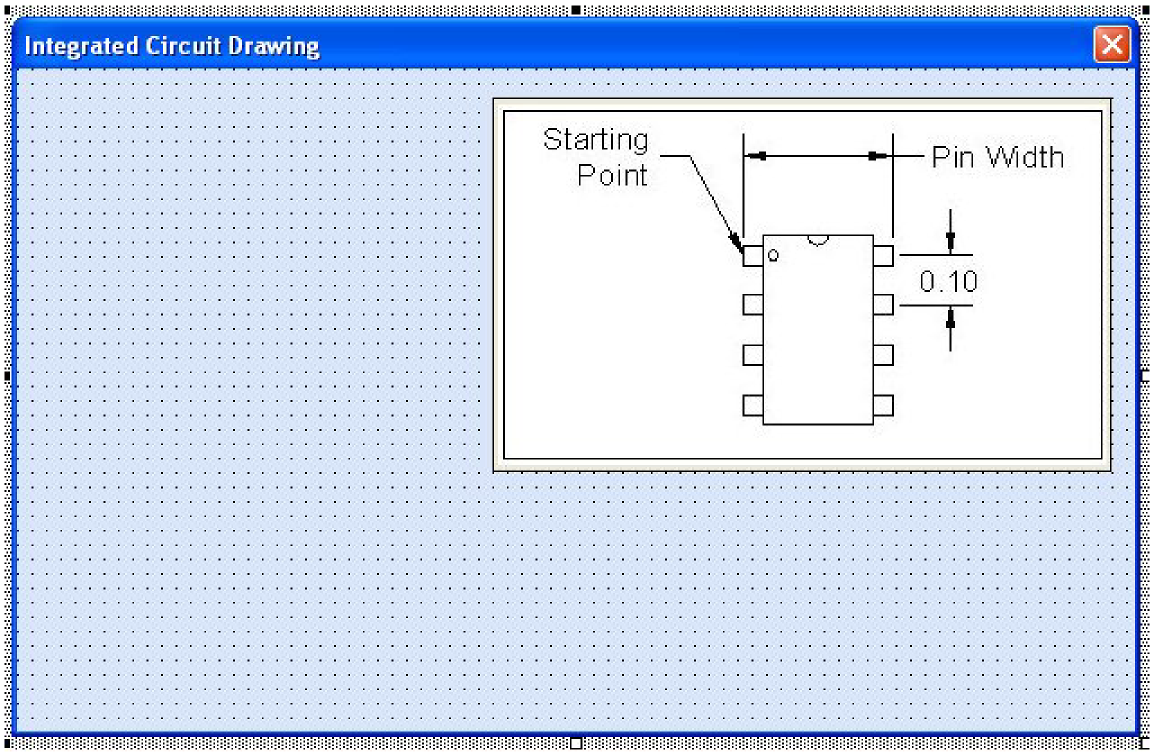


Figure C.14 – Placing an Image on the Form

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.

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.

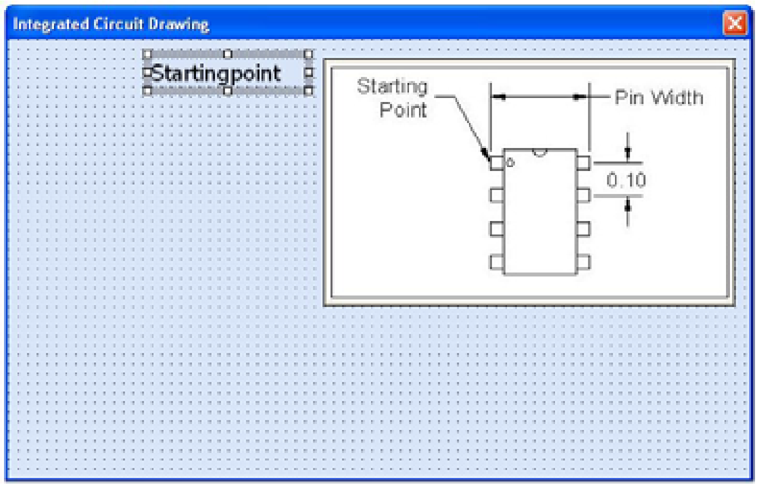


Figure C.15 – The Finished Label on the Form

For the first label, set the name as **lblStartingpoint** and the caption as Startingpoint. The width of the textbox is 100 and the height is 24. For labels on the top side of the textbox, set the TextAlign attribute to center justification.

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 label 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 C.16.

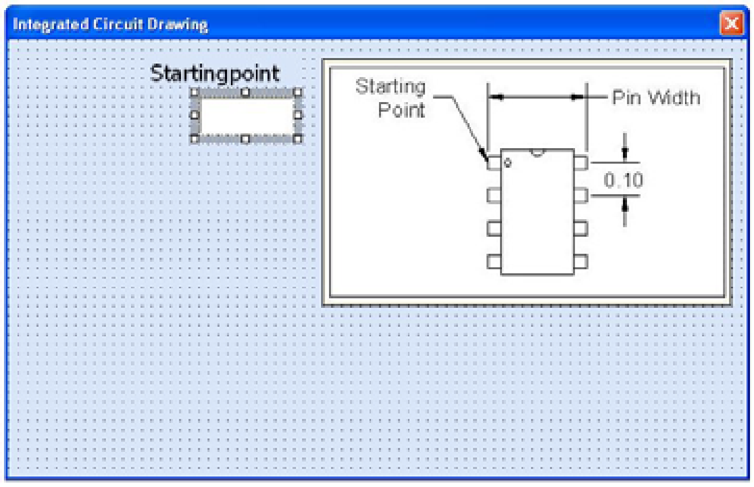
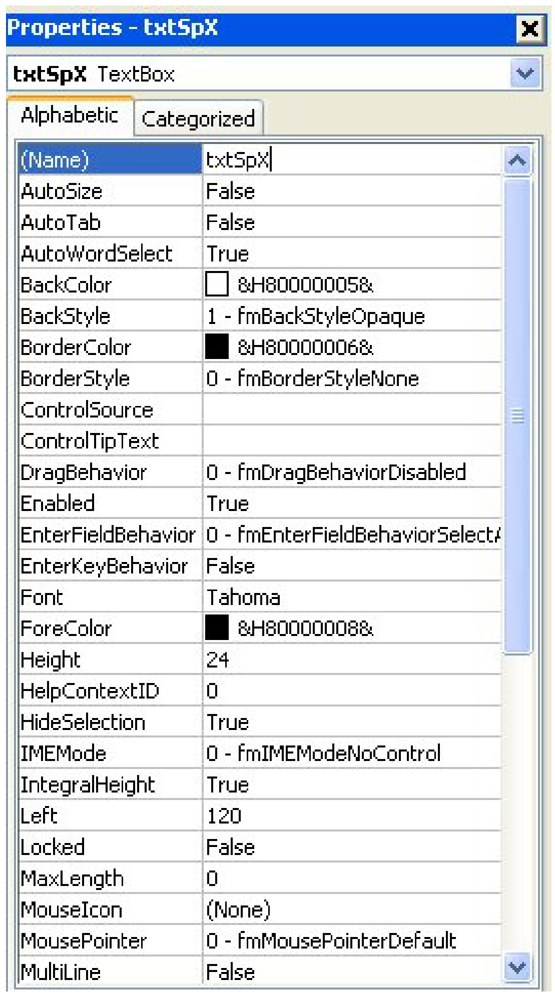


Figure C.16 – Placing a TextBox on the Form

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 **txtSpX.**

Alphabetic	
(Name)	txtSpX
Height	24
Width	60

We place a Label using a common Visual Basic naming convention **lblSpX** 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.



Properties - txtSpX

txtSpX TextBox

Alphabetic | Categorized

(Name)	txtSpX
AutoSize	False
AutoTab	False
AutoWordSelect	True
BackColor	□ &H80000005&
BackStyle	1 - fmBackStyleOpaque
BorderColor	■ &H80000006&
BorderStyle	0 - fmBorderStyleNone
ControlSource	
ControlTipText	
DragBehavior	0 - fmDragBehaviorDisabled
Enabled	True
EnterFieldBehavior	0 - fmEnterFieldBehaviorSelect
EnterKeyBehavior	False
Font	Tahoma
ForeColor	■ &H80000008&
Height	24
HelpContextID	0
HideSelection	True
IMEMode	0 - fmIMEModeNoControl
IntegralHeight	True
Left	120
Locked	False
MaxLength	0
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
MultiLine	False

Figure C.17 – Changing the (Name) to txtName

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

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

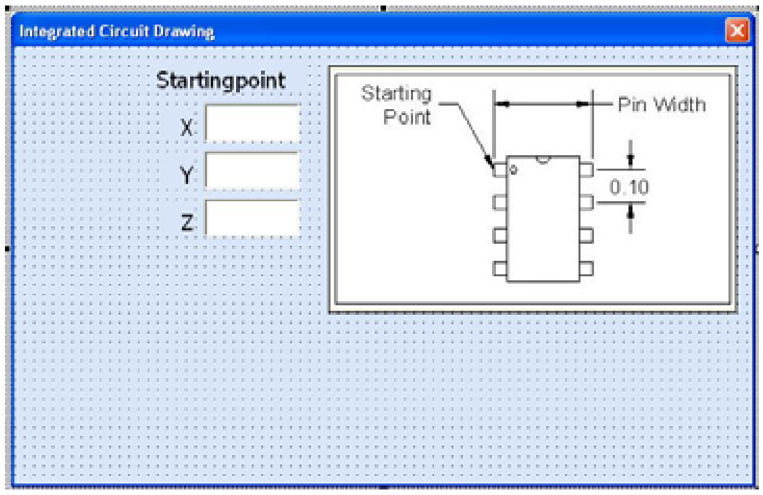


Figure C.18 – Adding the Y and Z Textboxes

We will add two more textboxes named **txtPinWidth** and **txtNumber** under the X, Y and Z textboxes. The labels to the left of the textbox are called **lblPin Width** and **lblNumber.** The Captions for the Labels are shown in Figure C.19.

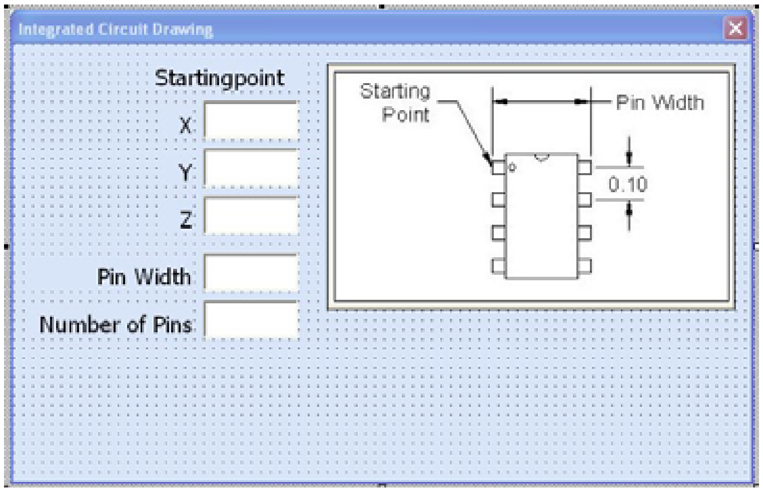


Figure C.19 – Adding Two More Textboxes

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 C.20.

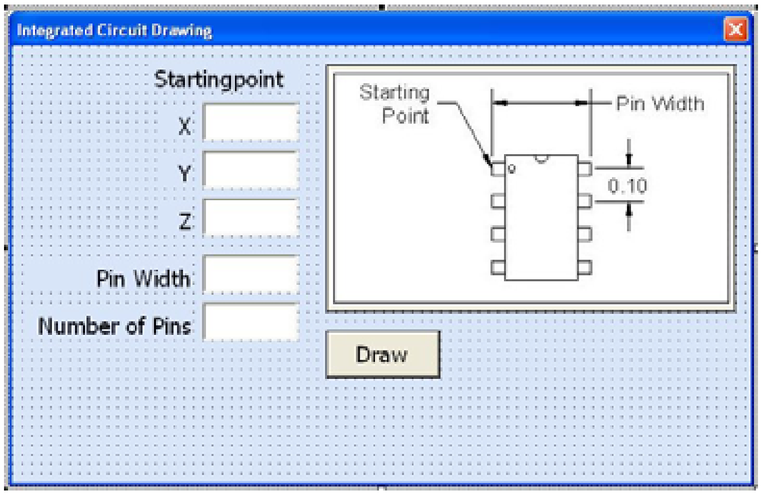
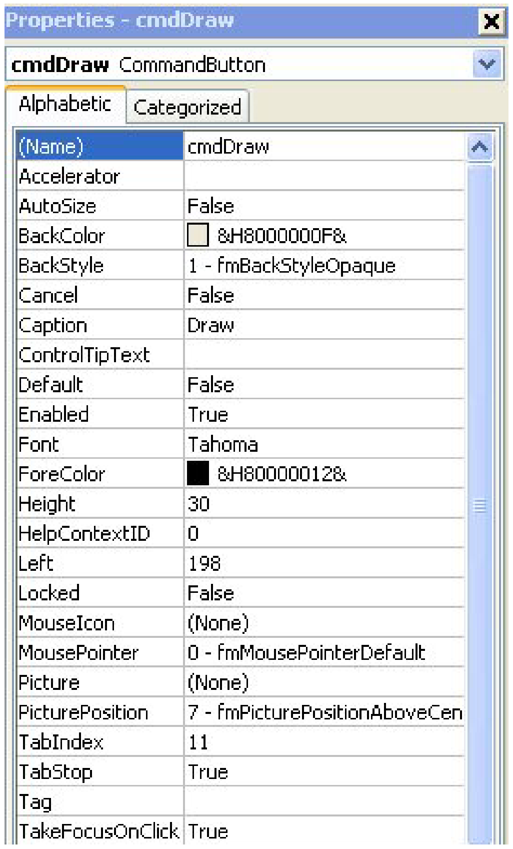


Figure C.20 – Insert a Command Button onto a Form

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

Alphabetic	
(Name)	cmdDraw
Caption	Draw
Font	Tahoma
Height	30
Width	72

The font we want for the Command Button is 18 point, Tahoma. When highlighting the row for Font, a small command button with three small dots appears to the right of the font name of Arial. 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.



(Name)	cmdDraw
Accelerator	
AutoSize	False
BackColor	⬜ &H8000000F&
BackStyle	1 - fmBackStyleOpaque
Cancel	False
Caption	Draw
ControlTipText	
Default	False
Enabled	True
Font	Tahoma
ForeColor	⬛ &H80000012&
Height	30
HelpContextID	0
Left	198
Locked	False
MouseIcon	(None)
MousePointer	0 - fmMousePointerDefault
Picture	(None)
PicturePosition	7 - fmPicturePositionAboveCen
TabIndex	11
TabStop	True
Tag	
TakeFocusOnClick	True

Figure C.21 – Changing the (Name) to cmdDraw

Add a second Command button; named cmdClear is for clearing the Starting point's X, Y, Z coordinates, Pin Width and Number of Pins textboxes. 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.

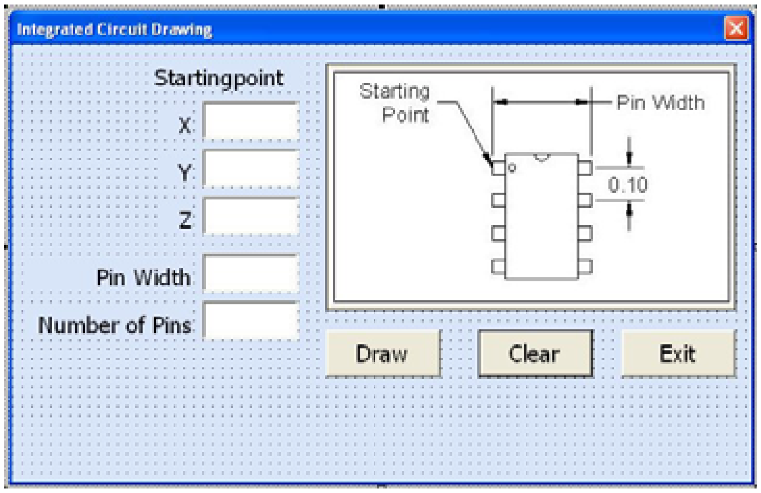


Figure C.22 – Insert Two More Command Buttons

The fourth command button is Pick Point, which we will name cmdPickPoint. We draw the button as shown in figure C.23 and after typing Pick for the caption, press Shift Enter and type Point on the second line. Center the text. When we code for this command button, we will allow the user to select a point on the graphical display and the X, Y and Z coordinates will appear in their specific textboxes.

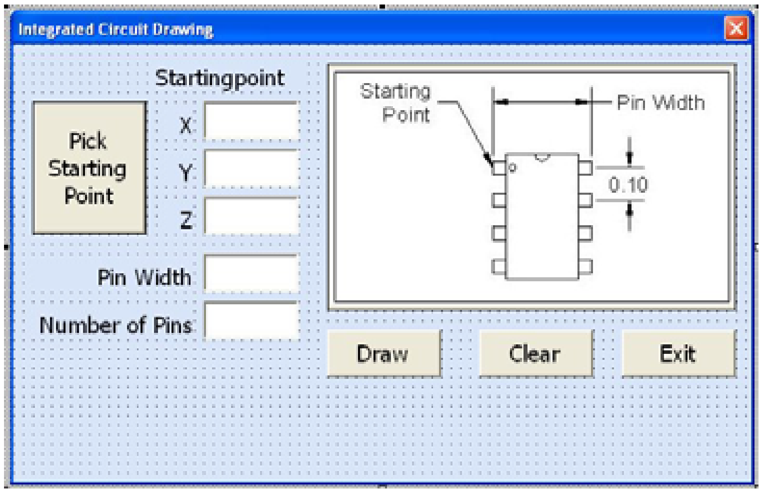


Figure C.23 – Insert the Fourth Command Button

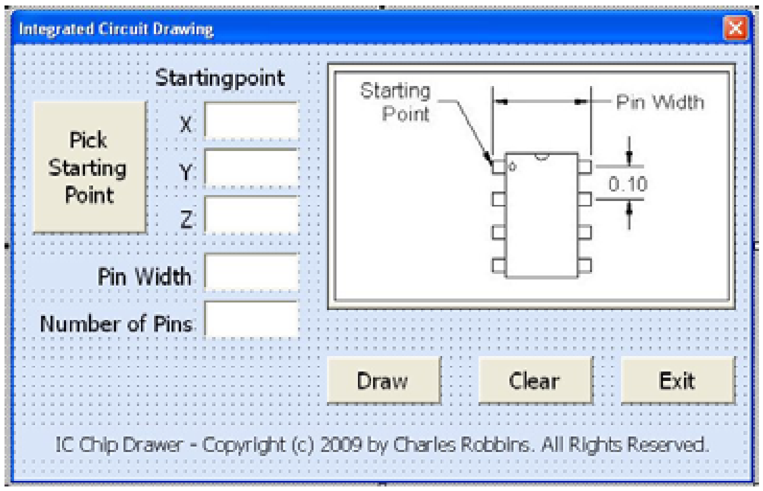
Adding a Copyright Statement to a Form

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.

IC Chip Drawer - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.

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



IC Chip Drawer - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.

Figure C.24 – 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 integrated circuit chip 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

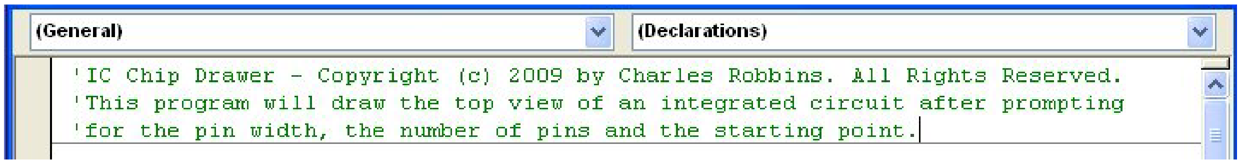
The comments we placed in the first three lines of the program will inform the individual opening and reading the code, but those user that may run the application without checking, the label on the bottom of the form with the copyright information is a great tool to alert the client to the rules of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

'IC Chip Drawer - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.
'This program will draw the top view of an integrated circuit after prompting
'for the pin width, the number of pins and the starting point.
```



```
'IC Chip Drawer - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.
'This program will draw the top view of an integrated circuit after prompting
'for the pin width, the number of pins and the starting point.
```

Figure C.25 – 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.

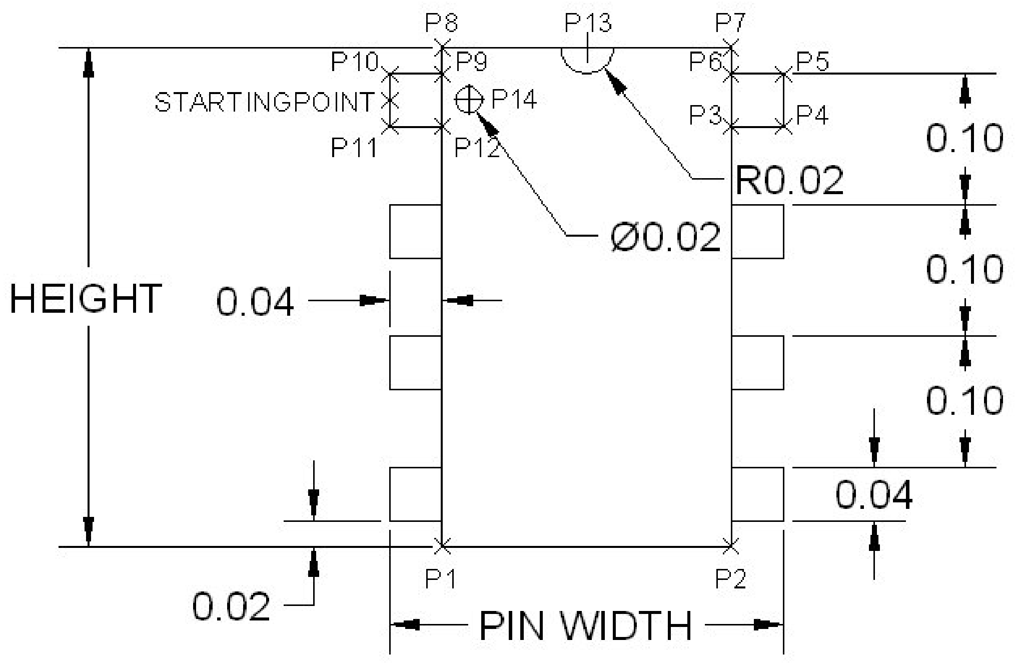


Figure C.26 – Identifying the Variables for the Integrated Circuit Chip Program

Type the following lines of code after the comment.

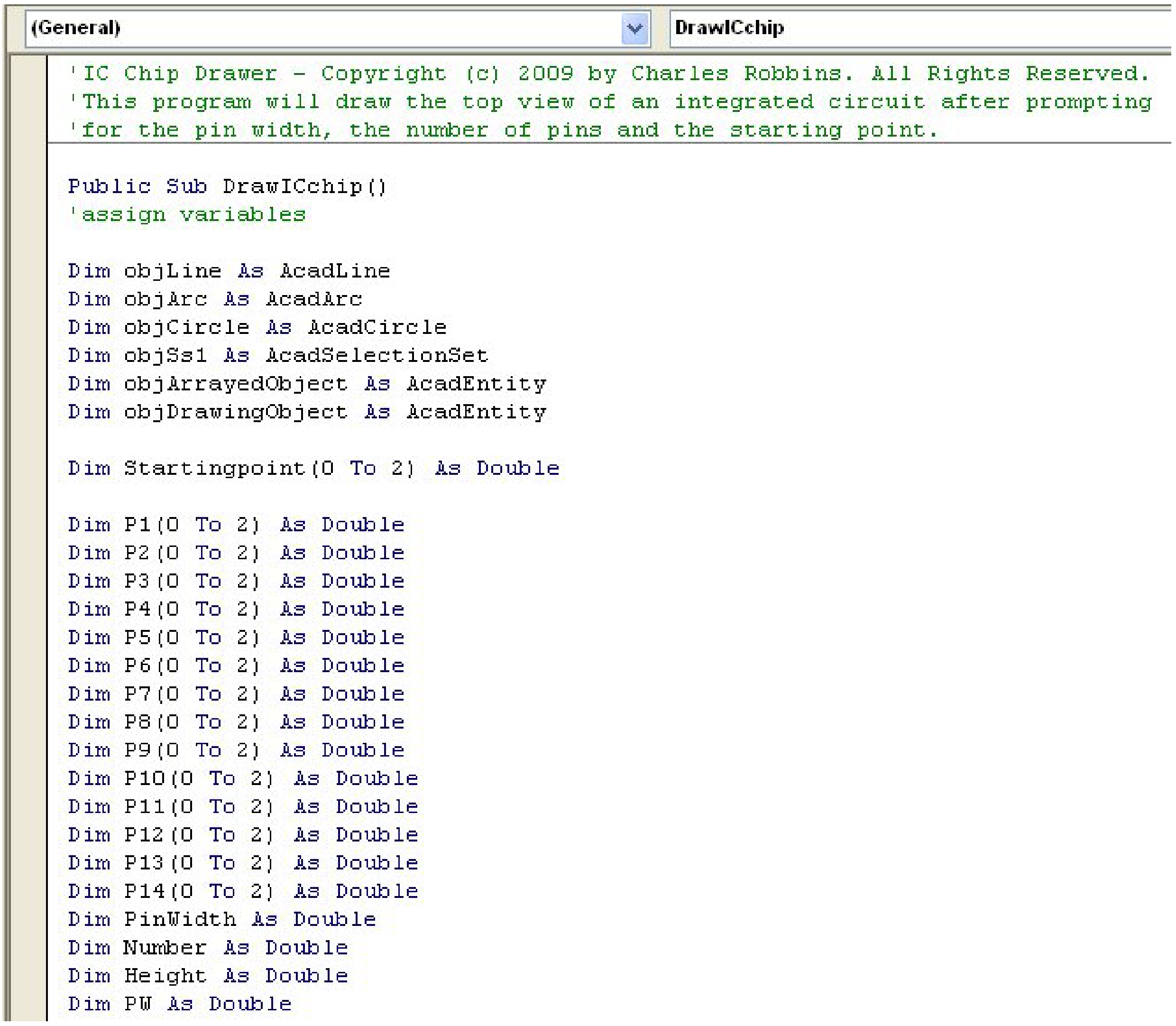
```
'assign variables

  Dim ObjArc As AcadArc
  Dim ObjCircle As AcadCircle
  Dim ObjLine As AcadLine
  Dim ObjSs1 As AcadSelectionSet
  Dim ObjArrayedObject As AcadEntity
  Dim ObjDrawingObject As AcadEntity
  Dim Startingpoint(0 To 2) As Double
  Dim P1(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 P5(0 To 2) As Double
  Dim P6(0 To 2) As Double
  Dim P7(0 To 2) As Double
  Dim P8(0 To 2) As Double
  Dim P9(0 To 2) As Double
  Dim P10(0 To 2) As Double
  Dim P11(0 To 2) As Double
  Dim P12(0 To 2) As Double
  Dim P13(0 To 2) As Double
  Dim P14(0 To 2) As Double
  Dim PinWidth As Double
  Dim Number As Double
  Dim Height As Double
  Dim PW As Double
```

In our program, we will declare a variable to enable us to draw circles, arcs and lines, a variable for each vertex and a variable for the pin width, number of pins, height and pw. PW is the variable for the number of rows in the array. 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 ArcObject is an AutoCAD Arc.

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.

We will declare points P1 through P14 for the vertices in the drawing in Figure C.26. Lastly, we declare PinWidth, Number, Height and PW as double integers (As Double).



```
'IC Chip Drawer - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.
'This program will draw the top view of an integrated circuit after prompting
'for the pin width, the number of pins and the starting point.

Public Sub DrawICchip()
'assign variables

Dim objLine As AcadLine
Dim objArc As AcadArc
Dim objCircle As AcadCircle
Dim objSs1 As AcadSelectionSet
Dim objArrayedObject As AcadEntity
Dim objDrawingObject As AcadEntity

Dim Startingpoint(0 To 2) As Double

Dim P1(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 P5(0 To 2) As Double
Dim P6(0 To 2) As Double
Dim P7(0 To 2) As Double
Dim P8(0 To 2) As Double
Dim P9(0 To 2) As Double
Dim P10(0 To 2) As Double
Dim P11(0 To 2) As Double
Dim P12(0 To 2) As Double
Dim P13(0 To 2) As Double
Dim P14(0 To 2) As Double
Dim PinWidth As Double
Dim Number As Double
Dim Height As Double
Dim PW As Double
```

Figure C.27 – 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.

```
'set variables

   Startingpoint(0) = txtSpX.Text
   Startingpoint(1) = txtSpY.Text
   Startingpoint(2) = txtSpZ.Text
   PinWidth = txtPinWidth.Text
   Number = txtNumber.Text
   Height = (Number - 2) / 2 * 0.1 + 0.08
   PW = Number / 2

'point assignments and math

   P1(0) = Startingpoint(0) + 0.04
   P1(1) = Startingpoint(1) + 0.04 - Height
   P1(2) = Startingpoint(2)
   P2(0) = Startingpoint(0) + PinWidth - 0.04
   P2(1) = Startingpoint(1) + 0.04 - Height
   P2(2) = Startingpoint(2)
   P3(0) = Startingpoint(0) + PinWidth - 0.04
   P3(1) = Startingpoint(1) - 0.02
   P3(2) = Startingpoint(2)
   P4(0) = Startingpoint(0) + PinWidth
   P4(1) = Startingpoint(1) - 0.02
   P4(2) = Startingpoint(2)
   P5(0) = Startingpoint(0) + PinWidth
   P5(1) = Startingpoint(1) + 0.02
   P5(2) = Startingpoint(2)
   P6(0) = Startingpoint(0) + PinWidth - 0.04
   P6(1) = Startingpoint(1) + 0.02
   P6(2) = Startingpoint(2)
   P7(0) = Startingpoint(0) + PinWidth - 0.04
   P7(1) = Startingpoint(1) + 0.04
   P7(2) = Startingpoint(2)
   P8(0) = Startingpoint(0) + 0.04
   P8(1) = Startingpoint(1) + 0.04
   P8(2) = Startingpoint(2)
   P9(0) = Startingpoint(0) + 0.04
   P9(1) = Startingpoint(1) + 0.02
   P9(2) = Startingpoint(2)
   P10(0) = Startingpoint(0)
   P10(1) = Startingpoint(1) + 0.02
   P10(2) = Startingpoint(2)
   P11(0) = Startingpoint(0)
   P11(1) = Startingpoint(1) - 0.02
   P11(2) = Startingpoint(2)
   P12(0) = Startingpoint(0) + 0.04
```

```
P12(1) = Startingpoint(1) - 0.02
P12(2) = Startingpoint(2)
P13(0) = Startingpoint(0) + PinWidth / 2
P13(1) = Startingpoint(1) + 0.04
P13(2) = Startingpoint(2)
P14(0) = Startingpoint(0) + 0.06
P14(1) = Startingpoint(1)
P14(2) = Startingpoint(2)
```

As we can see below, the first five variables equal the values from the textbox. After that, we assign each point's X, Y and Z coordinates a number either from the variable or from a mathematical calculation that we arrive from the sketch in Figure C.26. We use the variables PinWidth and Height to measure the distance from one point to another.

Inputting the Code to Draw in Visual Basic

Now we want to enter the code that will actually draw lines, circles and arcs in the AutoCAD Model Space. We use the Set function to draw a line by typing **Set ObjLine** and then we tell the computer that it will draw in Modelspace by adding a line from point P9 to point P10.

Go ahead and type the following comments and drawing code:

```
'draw the pins

  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P9, P10)
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P10, P11)
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P11, P12)

  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P3, P4)
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P4, P5)
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P5, P6)
```

We draw five more lines from P10 to P11, P11 to P12, P3 to P4, P4 to P5 and finally from P5 to P6.

Automatically Selecting and Arraying Drawing Entities

We array entities by selecting the object. We zoom all to have all the entities appear in the graphical display. We delete the TempSS selection set in case a this temporary file is open from a previous program. Then we create a selection set by using **acSelectionSetWindow, P11, P5** to retrieve all six objects inside the window of those two points. When we array the pins, we type:

```
Set objArrayedObject = objDrawingObject.ArrayRectangular(PW, 1, 1, -0.1, 1, 0)
```

Where the PW is the number of rows in the array, the 1 is the number of columns in the array, the next 1 is the number of levels for 3D arrays. Next we input 8 for the distance between rows, 1 for the distance between columns and 1 for the distance between levels. After the array, we delete the selection set objSs1.

```
'Array the Pins

ThisDrawing.Application.ZoomAll

On Error Resume Next
ThisDrawing.SelectionSets("TempSS").Delete

Set objSs1 = ThisDrawing.SelectionSets.Add("TempSS")
   objSs1.Select acSelectionSetWindow, P11, P5

For Each objDrawingObject In objSs1
Set objArrayedObject = objDrawingObject.ArrayRectangular(PW, 1, 1, -0.1, 1, 0)
   objArrayedObject.Update
Next
objSs1.Delete
```

We use the Set function to draw a circle by typing **Set ObjArc** and then we tell the computer that it will draw in Modelspace by adding an arc from the center point number 13 with a radius of 0.02. The arc will draw counterclockwise from the value of pi (3.14159) radians to zero radians.

```
'Draw arc
   Set ObjArc = ThisDrawing.ModelSpace.AddArc(P13, 0.02, 3.14159, 0)
```

We use the Set function to draw a circle by typing **Set ObjCircle** and then we tell the computer that it will draw in Modelspace by adding a circle from the center point number 14 with a radius of 0.01.

```
'Draw circle
   Set ObjCircle = ThisDrawing.ModelSpace.AddCircle(P14, 0.01)
```

We end the construction of the integrated circuit chip with the four lines of the chip’s body.

```
'Draw the lines
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P1, P2)
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P2, P7)
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P7, P8)
  Set ObjLine = ThisDrawing.ModelSpace.AddLine(P8, P1)
```

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:

```
'Unload and End the program
        Unload Me
        End
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 txtPinWidth, txtPinWidth.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.

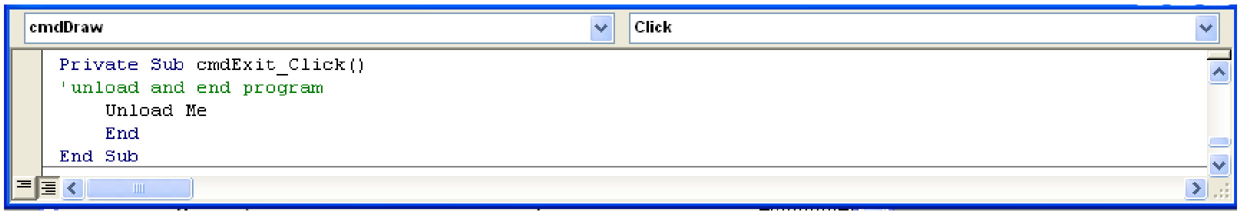
Key the following code as a new subroutine Private Sub cmdClear_Click().

```
Private Sub cmdClear_Click()
'clear the form
    txtSpX = ""
    txtSpY = ""
    txtSpZ = ""
    txtPinWidth = ""
End Sub
```

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
```



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

Figure C.28 – Coding the Exit Button

Executing a Subroutine with the cmdPickPoint Command Button

For the cmdPickPoint button, we will write SelectPoint which is a subroutine to allow the user to select a point on the graphical display and the X, Y and Z coordinates will be placed in the appropriate textboxes. A user can now type a starting point manually or has the choice to pick the initial point with their mouse.

```
Private Sub cmdPickPoint_Click()
   SelectPoint
End Sub
```

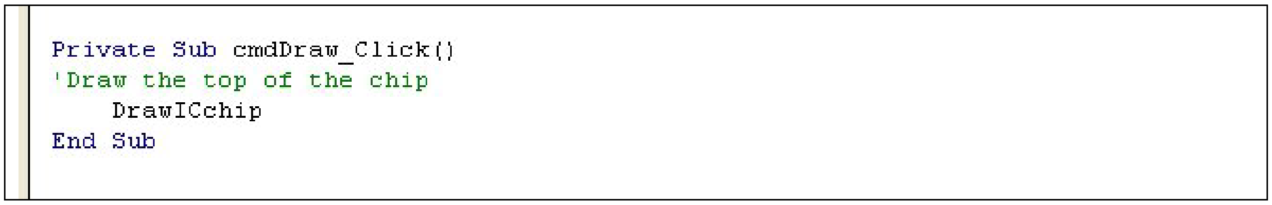
The SelectPoint subroutine starts with hiding the Foundation window and the prompt "Pick a Start Point: " is written on the command line. Once the single point is selected, the txtSpX textbox is given the first value of the starting point. After all three txtSP textboxes are filled, the Foundation window reappears.

```
Sub SelectPoint()
Me.hide
Dim StartPoint As Variant
   StartPoint = ThisDrawing.Utility.GetPoint(, vbCr & "Pick a Start Point: ")
   txtSpX = StartPoint(0)
   txtSpY = StartPoint(1)
   txtSpZ = StartPoint(2)
Me.Show
End Sub
```

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, DrawICchip.

```
Private Sub cmdDraw_Click()
'draw the top of the chip
   DrawICchip
End Sub
```



```
Private Sub cmdDraw_Click()
'Draw the top of the chip
    DrawICchip
End Sub
```

Figure C.29 – Coding the Draw Button

Written below is the entire program for creating the Integrated Circuit Chip. Next, we will insert a module to launch the form.

```
'IC Chip Drawer - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.
'This program will draw the top view of an integrated circuit after prompting
'for the pin width, the number of pins and the starting point.

Sub DrawICchip()
'assign variables

    Dim ObjArc As AcadArc
    Dim ObjCircle As AcadCircle
    Dim ObjLine As AcadLine
    Dim ObjSs1 As AcadSelectionSet
    Dim ObjArrayedObject As AcadEntity
    Dim ObjDrawingObject As AcadEntity
    Dim Startingpoint(0 To 2) As Double
    Dim P1(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 P5(0 To 2) As Double
    Dim P6(0 To 2) As Double
    Dim P7(0 To 2) As Double
    Dim P8(0 To 2) As Double
    Dim P9(0 To 2) As Double
    Dim P10(0 To 2) As Double
    Dim P11(0 To 2) As Double
    Dim P12(0 To 2) As Double
    Dim P13(0 To 2) As Double
    Dim P14(0 To 2) As Double
    Dim PinWidth As Double
    Dim Number As Double
    Dim Height As Double
    Dim PW As Double

'set variables

    Startingpoint(0) = txtSpX.Text
    Startingpoint(1) = txtSpY.Text
    Startingpoint(2) = txtSpZ.Text
    PinWidth = txtPinWidth.Text
    Number = txtNumber.Text
    Height = (Number - 2) / 2 * 0.1 + 0.08
    PW = Number / 2
```

```
'point assignments and math

    P1(0) = Startingpoint(0) + 0.04
    P1(1) = Startingpoint(1) + 0.04 - Height
    P1(2) = Startingpoint(2)
    P2(0) = Startingpoint(0) + PinWidth - 0.04
    P2(1) = Startingpoint(1) + 0.04 - Height
    P2(2) = Startingpoint(2)
    P3(0) = Startingpoint(0) + PinWidth - 0.04
    P3(1) = Startingpoint(1) - 0.02
    P3(2) = Startingpoint(2)
    P4(0) = Startingpoint(0) + PinWidth
    P4(1) = Startingpoint(1) - 0.02
    P4(2) = Startingpoint(2)
    P5(0) = Startingpoint(0) + PinWidth
    P5(1) = Startingpoint(1) + 0.02
    P5(2) = Startingpoint(2)
    P6(0) = Startingpoint(0) + PinWidth - 0.04
    P6(1) = Startingpoint(1) + 0.02
    P6(2) = Startingpoint(2)
    P7(0) = Startingpoint(0) + PinWidth - 0.04
    P7(1) = Startingpoint(1) + 0.04
    P7(2) = Startingpoint(2)
    P8(0) = Startingpoint(0) + 0.04
    P8(1) = Startingpoint(1) + 0.04
    P8(2) = Startingpoint(2)
    P9(0) = Startingpoint(0) + 0.04
    P9(1) = Startingpoint(1) + 0.02
    P9(2) = Startingpoint(2)
    P10(0) = Startingpoint(0)
    P10(1) = Startingpoint(1) + 0.02
    P10(2) = Startingpoint(2)
    P11(0) = Startingpoint(0)
    P11(1) = Startingpoint(1) - 0.02
    P11(2) = Startingpoint(2)
    P12(0) = Startingpoint(0) + 0.04
    P12(1) = Startingpoint(1) - 0.02
    P12(2) = Startingpoint(2)
    P13(0) = Startingpoint(0) + PinWidth / 2
    P13(1) = Startingpoint(1) + 0.04
    P13(2) = Startingpoint(2)
    P14(0) = Startingpoint(0) + 0.06
    P14(1) = Startingpoint(1)
    P14(2) = Startingpoint(2)
```

```
'draw the pins

   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P9, P10)
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P10, P11)
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P11, P12)

   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P3, P4)
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P4, P5)
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P5, P6)

   'Array the Pins

   ThisDrawing.Application.ZoomAll

   On Error Resume Next
   ThisDrawing.SelectionSets("TempSS").Delete

   Set objSs1 = ThisDrawing.SelectionSets.Add("TempSS")
      objSs1.Select acSelectionSetWindow, P11, P5

   For Each objDrawingObject In objSs1
   Set objArrayedObject = objDrawingObject.ArrayRectangular(PW, 1, 1, -0.1, 1, 0)
      objArrayedObject.Update
   Next
   objSs1.Delete

'Draw arc
    Set ObjArc = ThisDrawing.ModelSpace.AddArc(P13, 0.02, 3.14159, 0)

'Draw circle
    Set ObjCircle = ThisDrawing.ModelSpace.AddCircle(P14, 0.01)

'Draw the lines
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P1, P2)
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P2, P7)
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P7, P8)
   Set ObjLine = ThisDrawing.ModelSpace.AddLine(P8, P1)

'Unload and End the program
       Unload Me
       End
End Sub

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

```
Private Sub cmdClear_Click()
'clear the form
   txtSpX = ""
   txtSpY = ""
   txtSpZ = ""
   txtPinWidth = ""
End Sub

Private Sub cmdPickPoint_Click()
   SelectPoint
End Sub

Sub SelectPoint()
Me.hide
Dim StartPoint As Variant
   StartPoint = ThisDrawing.Utility.GetPoint(, vbCr & "Pick a Start Point: ")
   txtSpX = StartPoint(0)
   txtSpY = StartPoint(1)
   txtSpZ = StartPoint(2)
Me.Show
End Sub

Private Sub cmdDraw_Click()
'draw the top of the chip
   DrawICchip
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 C.30. In the Project Menu, double click on the Module and type the following code.

```
Sub DrawICChip ()
'draw the chip
   ICchip.Show
End Sub
```

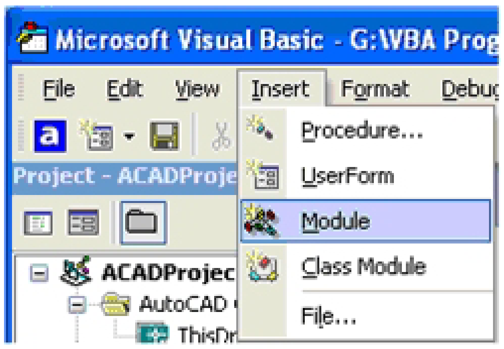
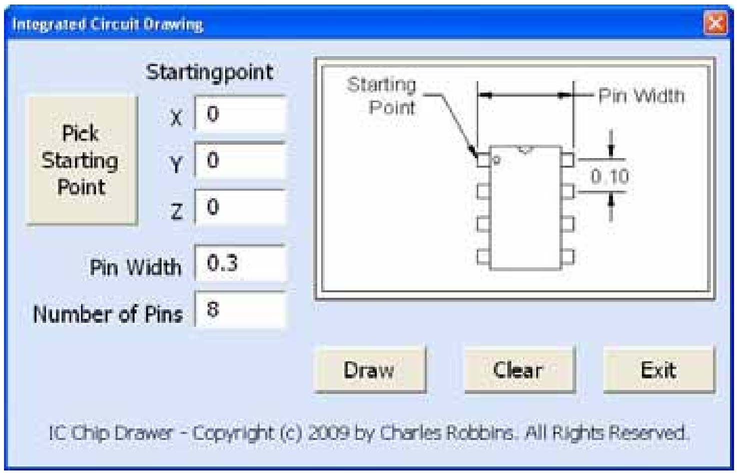


Figure C.30 – Inserting a Module

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

Running the Program

After noting that the program is saved, press the F5 to run the Integrated Circuit Chip application. The Integrated Circuit Chip window will appear on the graphical display in AutoCAD as shown in Figure C.31.



X: 0
Y: 0
Z: 0
Pin Width: 0.3
Number of Pins: 8

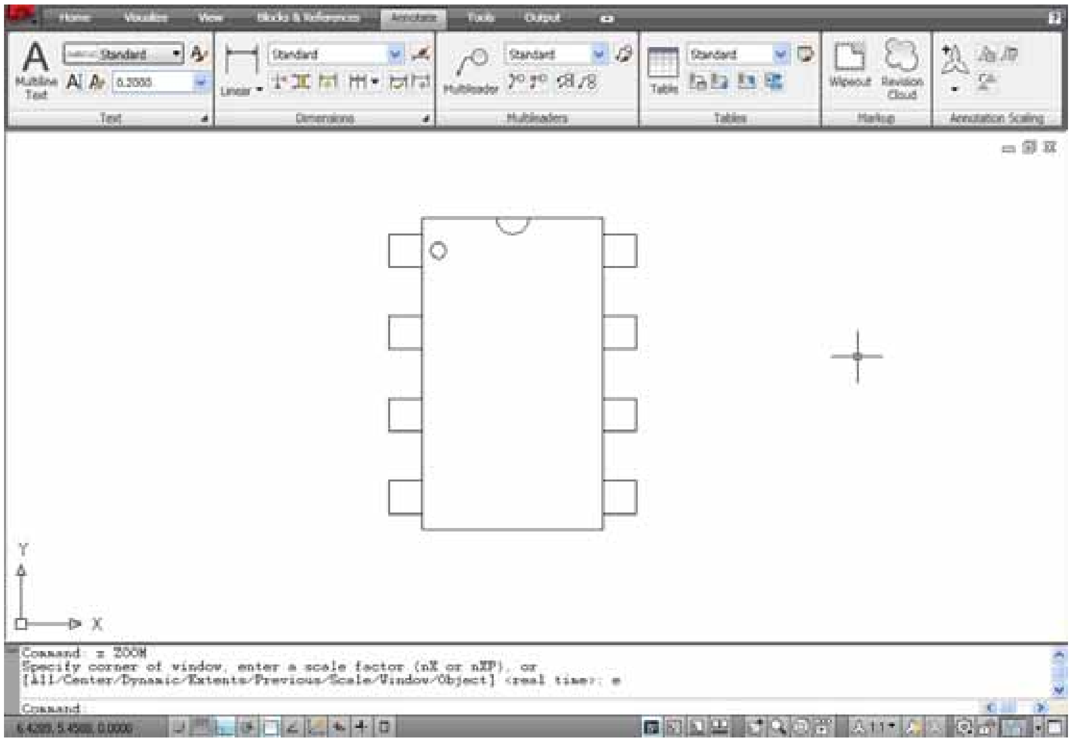
IC Chip Drawer - Copyright (c) 2009 by Charles Robbins. All Rights Reserved.

Figure C.31 – 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
Pin Width	0.3
Number of Pins	8

Figure C.32 – Input Data



```
Command: _ZOOM
Specify corner of window, enter a scale factor (nX or nXP), or
[All/Center/Dynamic/Extents/Previous/Scale/Window/Object] <real time>: e
Command:
```

6.4289, 5.4588, 0.0000

Figure C.36 – 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 5-12 * - Write a Visual Basic Application that draws any integrated circuit lines, circles and arcs by a inputting data into 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.