

Bonus Chapter 5: Visual Basic for AutoCAD

In This Chapter

- ✓ **Accessing the VBA environment with commands in AutoCAD**
- ✓ **Using the Integrated Development Environment (IDE)**
- ✓ **Understanding a VBA project**
- ✓ **Introducing the AutoCAD Object Model**
- ✓ **Creating a basic project**

Visual Basic for Applications (VBA), an extension of the popular programming language Visual Basic (VB), has been around for some time. VBA is defined as an object-oriented programming (OOP) language. The concept behind OOP is that a computer program is developed by using a collection of individual units, or objects, as opposed to a listing of instructions. Each one of the objects has the capability to receive messages, process data, and transmit messages to other objects. The objects are self-contained units that can be easily reused and modified for many purposes, thus saving time and effort.



This chapter is aimed at AutoCAD users only. Sorry, AutoCAD LT users — the programming interfaces are limited to AutoCAD.VBA is part of the programming and development tools provided by Microsoft. Its origins date back to QBasic and MS-Basic, which were both introduced during and prior to the MS-DOS era. VB has been around much longer than VBA and, unlike VBA, VB can be used to build standalone applications that are not dependent on a host application.

A *host application* is a program that allows the VBA environment to run inside it; AutoCAD is one example. Many other popular Windows-based programs have VBA technology built into them. Some of these other applications are Autodesk Inventor, AutoCAD-based vertical products such as AutoCAD Architecture (formally known as Autodesk Architectural Desktop), and Microsoft Word and Excel.

VBA in AutoCAD has been a feature welcomed by both the nondevelopment and development communities. This programming option allows for the integration of business applications directly into the AutoCAD environment, enabling companies to tap into an existing development community that already knows VB/VBA.

This chapter only scratches the surface of what is possible with VBA for AutoCAD. VBA for AutoCAD goes beyond just drawing simple objects, such as lines and circles. You can use VBA to do all sorts of things, from importing data from a corporate system to doing drawing setups when starting a new design. The power that VBA has is marvelous, especially when you consider how easy it is to use and the depth of capabilities it has compared to other languages.

The benefits of using VBA range from reducing repetitive tasks in AutoCAD to linking to external databases and files to help improve downstream information sharing. The sky is truly the limit, so keep on pushing the boundaries of VBA and AutoCAD.

AutoCAD Commands for VBA

Just like you use commands to draw a line in AutoCAD, you need to know and understand the different commands that are used to access the VBA environment from inside AutoCAD. You should know five core commands to work with VBA projects:

- ◆ VBAIDE
- ◆ VBALOAD
- ◆ VBAUNLOAD
- ◆ VBARUN
- ◆ VBAMAN

Some other commands are used to interact with the VBA environment, but the functionality that these commands provide can be found in the five commands just listed.

VBAIDE

VBAIDE is used to load the Visual Basic for Applications Integrated Development Environment (VBAIDE). This is where UserForms and code modules are added and organized into a project. The IDE provides tools that help during the coding process and to debug the procedures in a project. I discuss the IDE in greater depth in the section “Working with the IDE.”

VBALOAD

VBALOAD is used to load an existing project from disk into AutoCAD. When you load the project into AutoCAD, it becomes available for both execution and editing.



VBALOAD can be used two ways, which are controlled by whether the command's name is prefixed with a hyphen. Using VBALOAD with no hyphen launches the Open VBA project dialog box. If -VBALOAD is used, the prompt Open VBA Project is displayed at the dynamic input tooltip or command line. This can be useful when loading VBA projects with the AutoLISP programming language or a script file.

If you're working with an existing project or want to view a sample project, you must first load it into AutoCAD. Here's how:

1. At the command prompt, type VBALOAD (or from the menu browser or menu bar, click Tools menu → Macro → Load Project).

The Open VBA Project dialog box (see Figure BC5-1) appears.

2. Use the Look In drop-down list to specify the location for the VBA project.

AutoCAD comes with some sample project files that can be found in the folder C:\Program Files\AutoCAD 2009\Sample\VBA by default.

3. In the list box below the Look In drop-down list, select the project.

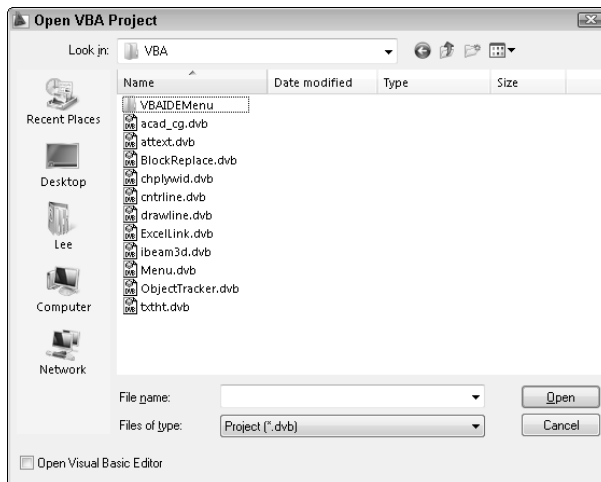
All the available project files in the specified folder are displayed in the list box. VBA project files have a unique file extension of .DVB.

4. (Optional.) Select the Open Visual Basic Editor check box in the lower-left corner.

If the check box is selected, the Visual Basic Editor is launched if it isn't already. If the Visual Basic editor is minimized in the taskbar, it is restored to a normal display state.

5. Click the Open button to load the project.

Figure BC5-1:
The Open VBA Project dialog box loads a saved VBA project file into AutoCAD.



6. In the AutoCAD message box, click Enable Macros.

By default, an AutoCAD message box (see Figure BC5-2) is displayed when you load a VBA project. This message box informs you that the project could contain viruses or unsafe code. Clicking More Info displays the AutoCAD Online Help system.

The other two options at the bottom of the message box — Disable Macros and Do Not Load — can be helpful if you're not sure who created the macros contained in the project. Disable Macros allows you to view the project in the editor, but code that would normally be run automatically is disabled. Do Not Load keeps the project from being loaded into AutoCAD and the editor altogether.

If Open Visual Basic Editor was selected in the Open VBA Project dialog box, the Visual Basic Editor (see Figure BC5-3) is displayed.

Figure BC5-2: Warning message about potentially unsafe macros.

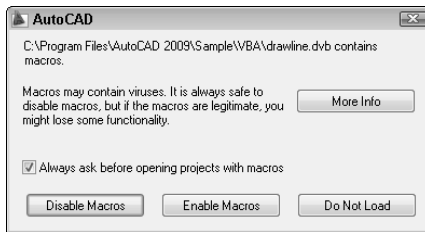
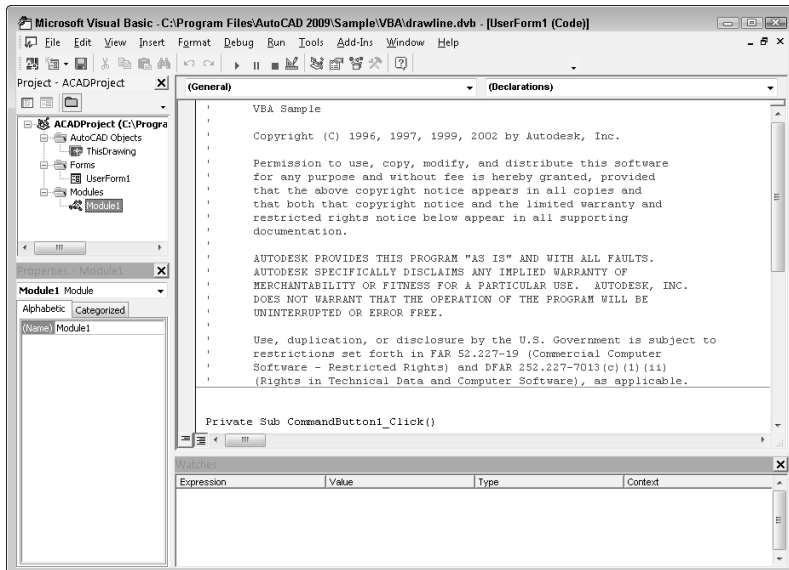


Figure BC5-3: You use the Visual Basic Editor to create and modify a VBA project.



VBAUNLOAD

VBAUNLOAD is used to unload a VBA project from AutoCAD that was previously loaded and is no longer needed. This can help to free up system resources for other tasks. Unlike VBALOAD, which displays a dialog box or can be run from the command line, VBAUNLOAD can be run only from the command line.

After you have finished working with or using a project, you can unload it without exiting AutoCAD. The following procedure describes how:

1. **At the command prompt, type VBAUNLOAD and press Enter.**
2. **At the Unload VBA Project prompt, type the full path to the project (the path from where it was loaded, plus the file name).**

As an example, if the project Drawline.DVB was loaded from the folder C:\Program Files\AutoCAD 2009\Sample\VBA, you would enter the following at the prompt:

```
“C:\Program Files\AutoCAD 2009\Sample\VBA\Drawline.DVB”
```

If the path to the project file contains spaces, double quotation marks must be entered before and after the path for the path to be valid.

VBARUN

VBARUN is used to execute a publicly defined procedure that is contained in a code module found in a loaded VBA project. The command launches the Macro dialog box, which allows you to perform a variety of tasks, from running a macro to changing the options of the VBA environment and even accessing the Visual Basic Editor. After a procedure has been executed, you follow any prompts or respond to any dialog boxes that are displayed.



VBARUN can be used two ways. The two different ways are controlled by whether the command's name is prefixed with a hyphen. Using VBARUN with no hyphen launches the Macros dialog box. If -VBARUN is used, you will be prompted for the name of the macro to run at the command prompt.

The -VBARUN command can be used to run a macro from a pull-down menu or a custom button on a toolbar or on the ribbon. The following format shows how to run a macro from a specific module by using the -VBARUN command:

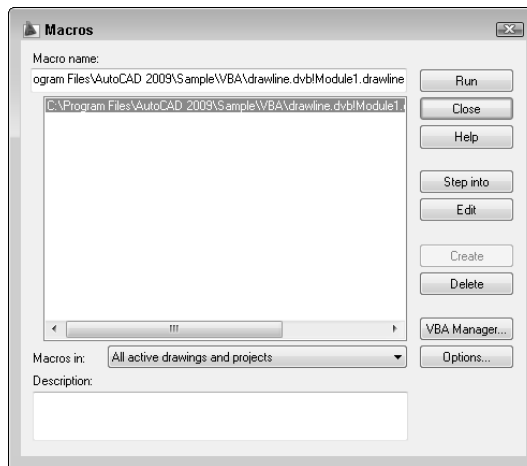
```
Macro name: <ACADProjectName>!<ModuleName>.<ProcedureName>  
"C:/Program Files/AutoCAD  
2009/Sample/VBA/drawline.dvb!Module1.drawline"
```

After a project has been loaded, you can execute one of the publicly defined procedures in a loaded project from AutoCAD, but not one that is defined as a private procedure. Here's how:

1. **At the command prompt, type VBARUN (or from the menu browser or menu bar, click Tools menu⇨Macro⇨Macros).**

The Macros dialog box (see Figure BC5-4) appears.

Figure BC5-4:
The Macros dialog box allows you to run a macro from a loaded project.



2. **In the list box in the middle of the dialog box, select a macro (or procedure) name.**

The list box displays all the available macros from the loaded projects. This list may be very large based on the number of projects that are loaded into AutoCAD at a given time. You can use the Macros In dropdown list to filter which macros are displayed in the list box.

3. **Click the Run button on the right side of the dialog box.**

VBAMAN

VBAMAN is used to launch the VBA Manager dialog box (see Figure BC5-5). From this dialog box, you can load or unload a project, run a macro, and much more. Much of the functionality in the VBA Macro dialog box is also found in the VBAIDE, VBALOAD, VBAUNLOAD, and VBARUN commands.

VBA Manager does offer some unique features, such as being able to save a loaded VBA project to a different file name, create a new project, and embed a project into a drawing file. Embedding a project into a drawing file has a couple of effects. One is that the drawing file now contains macros, so the

individual receiving the file might not feel comfortable opening the file due to the Macros warning message box being displayed. The recipient of the message might not be sure what will happen if the macros are enabled.

Also, having the code embedded in a drawing file means that the code is much harder to update because it is not in one centralized location. On the other hand, one of the benefits of having the macros embedded in a drawing file is that they are available to the individual that opens the drawing. By embedding a macro, you can provide utilities to help view your drawings or control layer display settings.

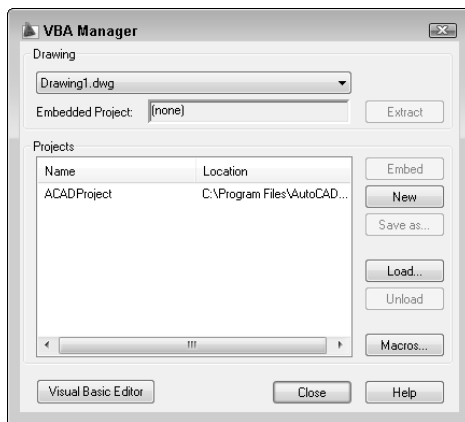


Figure BC5-5:
The VBA Manager dialog box is the hub for working with VBA projects.

Other commands

Some less commonly used commands are available in AutoCAD for use with the VBA environment. The functionality of these commands is a subset of the functionality that can be found in the VBA Manager dialog box. These commands are

- ◆ **VBAPREF:** Used to access settings that affect the VBA environment. This is the same as clicking the Options button in the Macros dialog box of the VBARUN command.
- ◆ **VBANEW:** Used to create a new and empty VBA project. This is the same as clicking the New button located in the VBA Manager dialog box of the VBAMAN command.
- ◆ **VBASTMT:** Allows for executing a single line of VBA code at the command prompt. The command can be used to integrate simple lines of VBA code into AutoLISP if needed. This is a great way to add message boxes and some light functionality that is not part of AutoLISP.

Working with the IDE

The tool that is used to develop VBA projects is referred to as the Integrated Development Environment, or IDE. The IDE is loaded by running the VBAIDE command, by selecting the Open Visual Basic Editor option in the Open VBA Project dialog box, by clicking the Visual Basic Editor button in the VBA Manager dialog box, or by clicking the Edit button in the Macros dialog box.

The IDE is where you'll spend the majority of your time coding, testing, and debugging any procedures and UserForms in a project. A VBA project is a group of modules organized into a specific file structure. These modules include UserForms, class, and standard modules. The projects created in the IDE work only in AutoCAD and many of the AutoCAD-based verticals such as AutoCAD Architecture. So a project created for AutoCAD can't be used in an application such as Microsoft Word or Excel, although they are also host applications for VBA projects.

To load the Visual Basic Editor from the menu browser or menu bar, click Tools menu → Macro → Visual Basic Editor.

Exploring the IDE

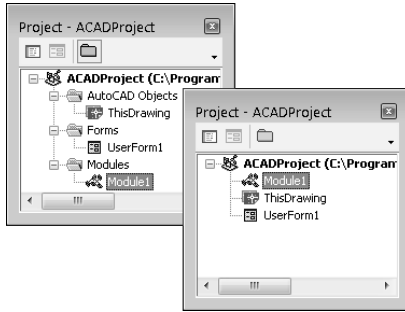
The Visual Basic Editor is broken up into many panes and areas; each of these provides a set of functions that makes many programming tasks much easier. By default, three panes and areas are visible at once. The four that you will most often use and should get comfortable with are

- ◆ Project Explorer
- ◆ Properties Window
- ◆ Code and UserForm Windows
- ◆ Object Browser

Project Explorer

The Project Explorer (see Figure BC5-6) is used to access the properties and the different modules that make up a VBA project. It provides an organizational view of the modules in the loaded projects, and allows you to open and view the contents of a module in its proper editing window within the Visual Basic Editor.

Figure BC5-6:
The Project Explorer from the Visual Basic Editor.



Properties window

The Properties window (see Figure BC5-7) is used to access the properties of a selected item in either the UserForm editor or the Project Explorer. The properties that can be set here range from controlling the visibility of a control on a UserForm to the name of the project or module. The Properties window helps to speed up the creation of UserForms by enabling you to change properties at design time and not just at run time.



Design time is when properties of a UserForm, a control, or an object are changed by using the Visual Basic Editor with one of the provided utilities, such as the Properties or editor windows. Run time is when properties of a UserForm, a control, or an object are changed during the execution of a procedure.

Figure BC5-7:
The Properties window from the Visual Basic Editor.

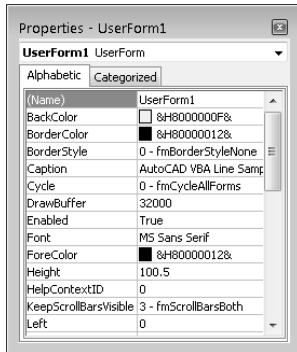
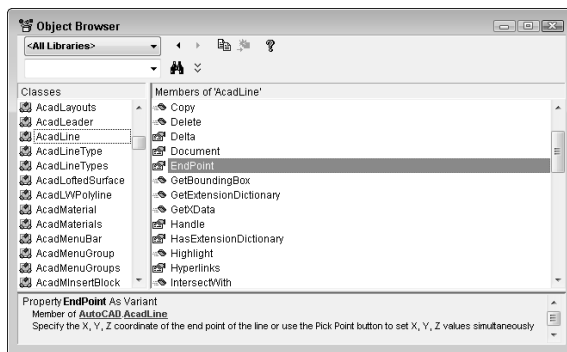


Figure BC5-9: The Object Browser in the Visual Basic Editor allows you to see which objects are in the loaded libraries.



Along with allowing you to see what objects are in a loaded library, the Object Browser provides access to the corresponding Help files and topics for the objects contained in the loaded libraries. This feature makes it easier to master new objects because you can see the different attributes they support right in the browser. Help can be accessed for a selected attribute by highlighting the attribute and then pressing F1. Many of the AutoCAD Help topics for VBA have a code example associated with them. This allows you to copy and paste code from the Help topic directly into the Visual Basic Editor and run it to see how it behaves.

The Object Browser will quickly become your new best friend when you are getting started with VBA. It helps you get in touch with information quickly, which can help reduce the learning curve. The following steps describe how to navigate the Object Browser:

- 1. In the Visual Basic Editor, click View⇨Object Browser.**
- 2. Click the down arrow in the top drop-down list to specify which of the loaded libraries you want to look in.**

By default, <All Libraries> is selected. This allows you to view all the objects and attributes in all the loaded libraries. The objects associated with AutoCAD are in the AutoCAD library. The objects associated with the Visual Basic for Applications programming language are in the VBA library.

- 3. In the Classes list box, select an object.**

The Classes list box displays the top level of the objects contained in a loaded library. The types of items that are listed are Classes, Collections, Enumerators, Types, and Modules. Notice that a description of the selected item is provided at the bottom of the Object Browser.

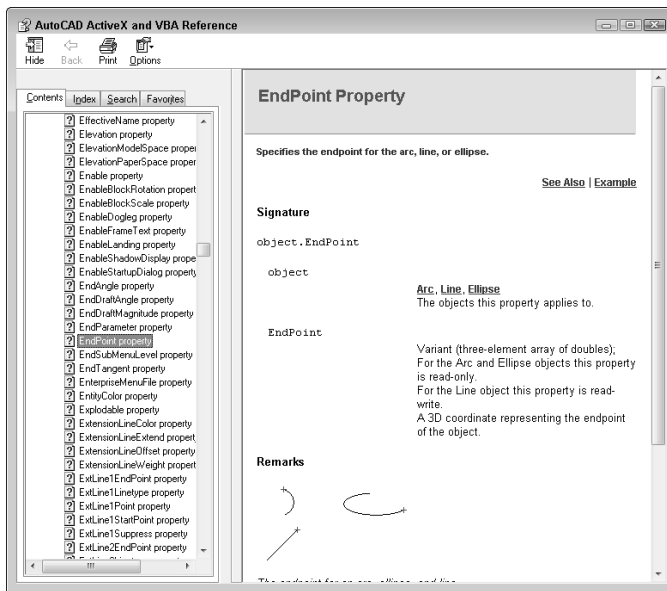
4. In the Members Of list box, select an attribute.

The Members Of list box displays the constants, properties, methods, events, and objects related to the class selected in the Classes list box. Notice that information about the selected member is displayed at the bottom of the Object Browser. The information displayed might include a brief description of the member, the syntax for a method, or an indicator that a property is read-only.

5. If you want additional information about the selected item in the Members Of list box or the Classes list box, press F1.

This launches the AutoCAD ActiveX and VBA Reference Online Help file (see Figure BC5-10).

Figure BC5-10: Help about classes and members in the AutoCAD library is just a click away with the AutoCAD ActiveX and VBA Reference Online Help file.



Parts of a VBA Project

A VBA project holds the code that is used to automate a task. A project can be as simple as one class module called ThisDrawing and three lines of code, or as complex as a project that might include UserForms and many different code modules. The tasks being automated by the VBA project determine the type of modules that are used as part of the final project. This section outlines the modules that can be found in a project and what is contained in each module.

Standard code module

Standard code modules are used to store procedures, data types, variables, and functions that are accessed by other modules in the project. Code modules should be used to hold reusable code that can be used in future projects or called from multiple places in a project. The concept of making code reusable is called *modular programming* and helps to avoid redundant code.

Class code module

Class code modules are used to store procedures and properties for user-defined classes. A user-defined class allows you to create your own object for storing information or manipulating other objects, similar to the way AutoCAD uses objects. All AutoCAD VBA projects contain a special class module called *ThisDrawing*. The *ThisDrawing* class module represents the object that defines the current drawing in AutoCAD. Although you can create your own classes with VBA, it is not something that is often needed or done.

Procedure (subroutine and function)

A procedure is a block, or group, of code that is given a name so it can be referenced easily in a project. There are two types of procedures: sub and function. A sub, or subroutine, is a type of procedure that does not return a result; a function returns a single result after the procedure has completed all tasks and computations. To return a value for a function, you use a variable with the same name as the procedure. Procedures that are used to manipulate objects or return information about an object can be found throughout the AutoCAD Object Library. Some examples of procedures follow:

```
Function Add2WholeNumbers (Num1 as Integer, Num2 as Integer)
    As Integer
    Add2WholeNumbers = Num1 + Num2
End Function
```

```
Sub EchoValue (Value as Variant)
    MsgBox Cstr(Value)
End Function
```

The following shows how to use the two preceding custom procedures, *Add2WholeNumbers* and *EchoValue*:

```
EchoValue Add2WholeNumbers(1, 3)
```

This code displays a message box containing the text 4.

Declaring variables

A variable is a way to store data in memory for use later, to hold the value of an argument that is passed into a procedure, or to capture the returning value

of a function. To use a variable, you should first use the Dim statement to dimension (or declare) the variable as a specific data type. Declaring the variable as a specific type helps the editor to check the data integrity of the code when it is being executed and debugged.

Variables can be defined locally within a given procedure or module and globally at a module level. For most data types, you only need to dimension the variable and assign the value that you want stored with it. Some examples of variables are

```
Dim strValue as String
Dim vVar
```

Data types

Data types are used to describe what type of data a variable holds; only a single data type can be assigned to a variable at one time. By default, when a variable is not declared with the Dim statement, or when it is defined with the statement but no data type, it is declared as a *variant*. Variants are often used to capture return values and as arguments for a procedure, in some cases based on the type of information that is being passed in.

Table BC5-1 shows some of the commonly used data types in VBA.

Table BC5-1	Common Data Types
<i>Data Type</i>	<i>Description and Example</i>
Boolean	A true or false value. Example: True
Integer	Any whole number without decimal places from -32,768 to 32,767. Example: 10
Double	Any large number with decimal places, accurate up to 16 digits. The range is $1.80 \times 10E308$ to $-4.94 \times 10E-324$ for negative values; $4.94 \times 10E-324$ to $1.80 \times 10E308$ for positive values. Example: 1.0, 0.23451, .5
String	Any number of alphanumeric characters enclosed in double quotes. The range is from 0 to approximately 2 billion characters. Example: "Hello"
Object	A valid COM object. Example: objLine
Variant	Any numeric range up to a double, a text string, or a COM object. Example: 183.59

Assigning a value to a variable

The equal sign (=) is used to assign a value to a variable. The value on the right is assigned to the variable on the left. If an object is being returned and will be assigned to a variable, the Set statement must be used with the equal (=) sign. Some examples of value assignment are

```
' String value assignment
Dim strValue as String
strValue = "Hello out there"
' Object value assignment
Dim acObjLine as AcadLine
Dim dPT1(0 to 2) as Double, dPT2(0 to 2) as Double
dPT1(0) = 0: dPT1(1) = 0: dPT1(2) = 0
dPT2(0) = 5: dPT2(1) = 5: dPT2(2) = 0
Set acObjLine = ThisDrawing.ModelSpace.AddLine(dPT1, dPT2)
```

The basics of working with objects

You may encounter objects throughout the use of the AutoCAD Object Library and when working with UserForms and controls. Objects come in two distinct types: objects and collections.

Objects exist as individuals and as part of a group containing many objects of the same kind. A group of objects is referred to as a *collection*. The first object that you encounter when you open the Visual Basic Editor is This Drawing, which is an example of an individual object. ThisDrawing represents an object of the data type AcadDocument. ThisDrawing is an object that represents the current drawing and is part of a collection that contains all open drawings in AutoCAD called AcadDocuments, which is part of the Acad Application object.

After you have a reference to an object, you use a dot to access its properties or methods. At times, you can access the object above or below the current object that you are referencing. The following syntax represents how to access the properties and methods of an individual object:

```
Object.<Object or Property or Method>
```

The following two examples show how to reference the full name of the current drawing and add a line object to model space:

```
ThisDrawing.FullName
ThisDrawing.ModelSpace.AddLine dPT1, dPT2
```

BC94 Introducing the AutoCAD Object Model

If you want to access an object that is part of a collection, you need to do a little more work because objects in a collection are accessed in a slightly different way. Objects contained in a collection are indexed and are accessible by using their unique names or a number in most cases. An index is used for each object added to a collection and represents the order in which items are added to the collection. An object's index is similar to a street address for a house. All houses have unique street numbers so they can get their mail; indexes for collections work the same way. Most collections begin with an index of 0; however, some start with an index of 1. A collection works the same whether the first item has an index of 0 or 1. (Really it's what the programmer felt like using that day for the first item's index.) Here are a few examples of how to access a layer from the Layers collection in the current drawing:

```
ThisDrawing.ActiveLayer = ThisDrawing.Layers("0")
ThisDrawing.ActiveLayer = ThisDrawing.Layers(1)
```

As mentioned, the Set statement must be used to store a reference to the object in a variable:

```
Dim objLayer As AcadLayer
Set objLayer = ThisDrawing.Layers("0")
```

Adding comments

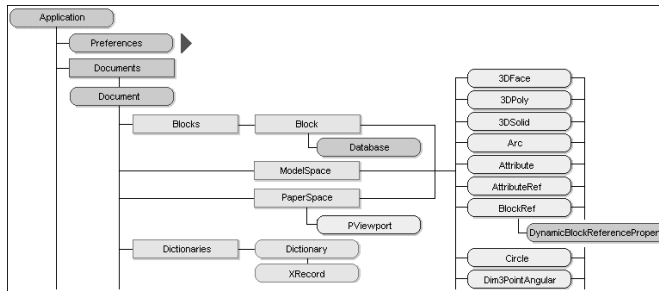
A *comment* is a line or series of lines in a code module that does not get executed. A comment can be used to identify what is taking place in a procedure or may consist of a general note at the top of the module that includes general revision and author information. An apostrophe (') or single quote mark is placed at the front of a text string to designate it as a comment. By default, a comment appears in green in the Visual Basic Editor. An example comment is

```
' Close application
```

Introducing the AutoCAD Object Model

The AutoCAD Object Model is the organizational structure that is used to navigate the AutoCAD Object Library. The AutoCAD Object Model (see Figure BC5-11) is very large; only a small portion is shown here.

Figure BC5-11:
A view of the AutoCAD Object Model.



At the top of the AutoCAD Object Model is the Application; below that are the Preferences and Documents collections. You can identify that they are collections because their names end with *s*. Just below the Documents collection is the Document object; this is the level that the object ThisDrawing represents in the object model. To give you an idea of what each Document object is made up of, I have included a few of the members that define a Document object:

- ◆ **ActiveLayer:** Property to set and return the current layer.
- ◆ **FullName:** Property to return the drawing's name with path.
- ◆ **Blocks:** Property that returns a reference to the collection that holds all the blocks defined in the drawing.
- ◆ **SaveAs:** Method used to save the drawing with a different name or location.

These steps show you how to open the AutoCAD Object Model Map in the AutoCAD Online Help system:

- 1. On the menu browser or menu bar in AutoCAD, click Help menu⇨Additional Resources⇨Developer Help.**
- 2. Click the Contents tab and expand ActiveX and VBA Reference.**
- 3. Select Object Model just below ActiveX and VBA Reference on the Contents tab.**

Creating a basic VBA project

You've come this far, so you should be ready to build a small VBA project to add a line in the current drawing.

The following steps describe how to create a new VBA project:

- 1. On the menu browser or menu bar in AutoCAD, click Tools menu⇨Macro⇨VBA Manager.**

The VBA Manager dialog box is displayed.

2. Click the New button on the right side of the dialog box.

If no projects were previously loaded, a new project named ACADProject is added to the Projects list box with the location of Global1.

3. In the Projects list box, select the new project.

4. Click the Save As button on the right side of the dialog box.

The Save As dialog box is displayed, allowing you to specify a file name and location for the new VBA project.

5. Using the Save In drop-down list or the Places list, browse to the My Documents folder.

6. In the File Name field of the Save As dialog box, enter FirstVBAProject.

7. Click the Save button.

The new VBA project is saved to the My Documents folder. The location and file name that the project is saved to is reflected in the VBA Manager dialog box in the Location column.

Working with the new VBA project in the editor

Here is how to launch the Visual Basic Editor and work in the new project:

1. In the Projects list box, select the new project.

2. Click the Visual Basic Editor button (at the bottom left of the VBA Manager dialog box).

The Visual Basic Editor should be displayed on-screen. If it is not displayed, check the Windows taskbar to see whether the Editor is minimized.

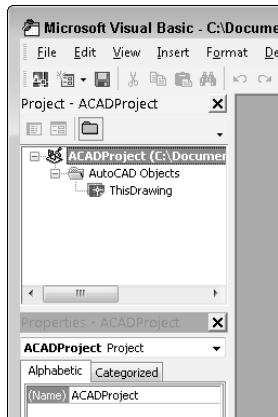
3. In Project Explorer, select the top node, ACADProject, which represents the project.

In the Properties window (see Figure BC5-12), the properties for the project are listed. A project has one property that can be modified from the Properties window, and that is Name.

4. In the Name field in the Properties window, highlight the current value of ACADProject and enter FirstVBAProject.

This changes the name that appears in the VBA Manager dialog box and the Macros dialog box. Using a unique name lets you distinguish the project from other loaded projects.

**Figure
BC5-12:**
Name that
project.



5. Right-click Project Explorer and click Insert→Module.

A new standard code module named *Module1* is added to the project. Note that a new folder named *Modules* is automatically created under the project. All standard code modules that are imported or created in a project are added to this folder.

6. Select Module1 from Project Explorer. Use the Properties window to change the value of the Name property for Module1 to basDraw.

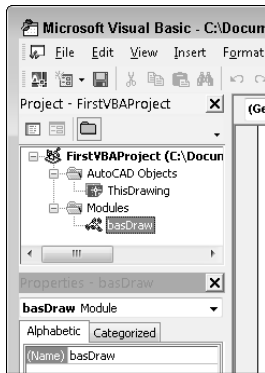
A project can have more than one module, so it's a good idea to give each module a unique name. A module can't be imported into another project if a module with the same name already exists. The project and standard code module should both be renamed now (see Figure BC5-13).

Going further with AutoCAD and VBA

In addition to the information I cover in this chapter, you should explore the AutoCAD Object Model and take a look at the different objects and collections in the AutoCAD Object Library through the Object Browser in the Visual Basic Editor. AutoCAD also ships with a variety

of samples in the folder C:\Program Files\AutoCAD 2009\Sample\VBA. In addition, the Internet is a great tool for finding tutorials that demonstrate how to use VBA for AutoCAD as well as many other host applications.

Figure BC5-13: Visual Basic Editor with the new project and code module displayed.



Adding a new procedure to a code module

These steps describe how to create a new procedure in a code module:

1. Click in the Code window for the new code module.

A code window should have opened up automatically when the new standard code module was added. If not, double-click `basDraw` in Project Explorer for the project. Clicking in the code window sets the editing focus to the code window and at the desired location within the code window.

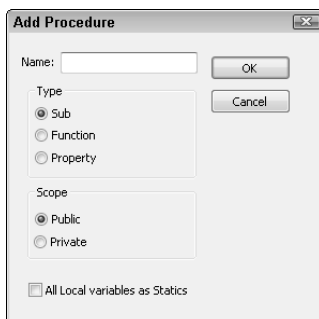
2. In the Visual Basic Editor, click `Insert`→`Procedure`.

This launches the Add Procedure dialog box (see Figure BC5-14).

3. In the Name text box, enter `DrawLine`.

The name entered is what will be displayed in the Macros dialog box or can be used with the `-VBARUN` command. The name of the procedure is also used when you want to call the procedure from another location in the project.

Figure BC5-14: The Add Procedure dialog box.



4. In the Type section, click Sub.

The new procedure will be defined as a subroutine because no return value is required.

5. In the Scope section, click Public.

The new procedure needs to be defined as public so it can be accessed from the Macros dialog box and outside the code window it is defined in.

6. Click OK.

The new procedure is added to the code window and looks like the following:

```
Public Sub DrawLine()  
End Sub
```

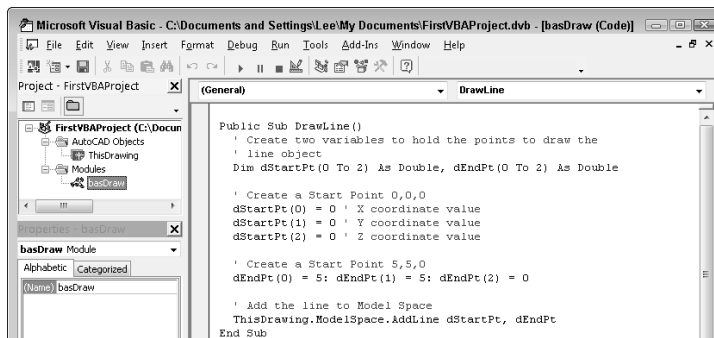
7. Add the following code between the two lines that were added by the Add Procedures dialog box:

```
' Create two variables to hold the points to draw  
' the line object  
Dim dStartPt(0 To 2) As Double, dEndPt(0 To 2) As Double  
    Create a Start Point of 0,0,0  
dStartPt(0) = 0 ' X coordinate value  
dStartPt(1) = 0 ' Y coordinate value  
dStartPt(2) = 0 ' Z coordinate value  
' Create an End Point of 5,5,0  
dEndPt(0) = 5: dEndPt(1) = 5: dEndPt(2) = 0  
' Add the line to Model Space  
ThisDrawing.ModelSpace.AddLine dStartPt, dEndPt
```

8. In the Visual Basic Editor, click Save on the Standard toolbar.

This saves the changes that have been made to the VBA project up to this point. The DrawLine procedure has now been defined (see Figure BC5-15) and is ready for use in AutoCAD.

Figure BC5-15:
The completed DrawLine procedure.



Running the new procedure

These steps describe how to run the DrawLine procedure:

- 1. In the Visual Basic Editor, click View⇨AutoCAD or switch to AutoCAD by clicking the icon on the Windows taskbar.**
- 2. On the menu browser or menu bar in AutoCAD, click Tools menu⇨Macro⇨Macros.**

The Macros dialog box is launched; it should list only one macro, DrawLine.

- 3. Select the macro from the list box (if it's not already selected) and click the Run button located on the right side of the Macros dialog box.**
- 4. On the menu browser or menu bar in AutoCAD, click View menu⇨Zoom⇨Extents.**

The new line is displayed in the center of the drawing window. If you list the line with the LIST command or use the Properties window in AutoCAD, you can see that the new line starts at 0,0,0 and ends at 5,5,0, which are the values defined in the DrawLine procedure.