

Chapter 12

Visual Basic Applications (VBA) – The Next Step

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

- 1. Using Visual Basic Applications with Computer Aided Design**
- 2. Seven Steps to Computer Aided Design Customization**
- 3. The Fundamentals of Visual Basic Applications**
- 4. Creating Structures in Our Programs**
- 5. Programs Do It All**
- 6. Companies Using Custom Engineering Programs**
- 7. The Future of Engineering Programming**

Using Visual Basic Applications with Computer Aided Design

The next step in Engineering programming for computer aided designers is learning to write, troubleshoot and execute Visual Basic Applications (VBA) for CAD software. We might want to ask why someone would want to learn Visual Basic Applications versus Visual AutoLISP, and the answer would be that this technique of coding offers the user more flexibility when operating or applying the finished coded product. We will find that the VBA routines are much more complex in their structure than the LISP routines, but rather than the user receiving the queries on the Command line, the VBA routines use forms which we will launch as a window in a Microsoft product. Just having an appearance of a colorful window on our computer desktop, allows additional team members such as clients, salespeople, managers and technicians to create drawings in the computer aided design package. The routines permit the design group to become more diverse, allowing for many ideas to be entered into the drawings, implementing innovative thoughts into building the product or constructing the house. Having the client or their representatives enter figures into the drawing database keeps the key people involved in the process.

Where Visual AutoLISP code is typically one to two pages long, VBA programs can be four to six pages in length. One of the reasons the code is much longer is because the declaration of variables is a much more complex operation than in the LISP routine. Where in the Visual LISP coding, the variable is declared with the `setq` function and many times we are storing real numbers, integers, text strings or a list when using `setq`. But we declare a variable in Visual Basic by using the Dim statement. Numbers are set using an expression like *Dim dblWidth as double*. When we want to create a list of numbers which represents a point on the graphical display, we will use the dim statement but open the point to an array of P (0 to 2) to describe any coordinate using the X, Y and Z measurements. Lastly, text strings are declared as strings using the Dim statement so the variable can be properly utilized in the program. We will find the section of the code that we declare the variables possibly long in length but easy to write.

Another area in coding that is new to us is linking the form to the program. In Visual Lisp, we receive the majority of our information either from the user selecting with the mouse on the graphical display or inputting information at the Command line. In the VBA code, the users still can select a point on the monitor or more commonly type a measurement, quantity or textstring in the form which appears in a window on your Microsoft Operating System. Unlike LISP, our programs do not have to run in applications such as AutoCAD, but anyone on the team can launch the program from the Windows desktop anywhere in the world. VBA programs can receive their database information through the Internet and at which time the AutoCAD drawings can be produced. This is an extreme advantage over other programming languages. An example could be a salesperson with a custom door program meeting with the client many states away from the AutoCAD server. After many minutes of answering questions such as length, width, height, color, material and special features, the salesperson can press the Enter command button sending the data to our CAD server. Multiple documents such as assemblies, part drawings, elevations and even price quotes can be made using the information received from the client and salesperson along with costs stored by the sales manager and all the documentation can be sent back to the customer in minutes. Only a few companies have this capability at this time but with more VBA programmers graduating from college or completing

a certified course, small and medium size companies will have the same capabilities as the larger corporations.

As with Visual AutoLISP, we would find the programming tasks to be very difficult when trying to start too far along at the assembly or price quote level. Our first applications in this language should be creating individual part drawings or simple outline details displaying critical dimensions and notes. Just as we accomplished in Visual AutoLISP, we compile multiples subroutines which will lead to more complex subassemblies and finally to main assemblies. Returning to our door salesperson's code that was on the laptop, subroutines which will make various details such as doorknobs, hinges, windows and kick plates are required so that every possible option is available which may appear in the assembly drawing. Once we virtually assemble the door in the computer aided design software, each externally referenced component can be analyzed by our MRP software, building the price quote and delivery schedule for the customer's product. Building intelligence from all members of the manufacturing team into the product design software reduces the errors on the factory floor. Today, more than ever, consumers demand a higher percentage of custom products to be delivered, and any organization depending on antiquated documentation processes such as telephone order, sketches, and manual CAD drafting are bound to see slower delivery times with a higher percentage of faults. Capturing our human procedures in the software allows the engineering department to achieve a higher level customer satisfaction, better quality inspection, and engineering control since individuals have time to monitor their assigned processes.

So now we have the training and the expertise to automate the engineering documentation so where do we make those changes and how do avoid making serious mistakes in implementation. There are many professionals that have the basic software training in computer aided design customization but never make any headway in drawing efficiency or the quality of their product. Some of the many reasons Engineers and Architects get lost in the process is because they enter into the process at a higher level than they are ready to achieve. A great example of this would be for small company wanting to create 3D subroutines which will make all of their components but the group lacks even the two dimensional data which contains valuable information allowing us to efficiently create the model library. In companies that skip critical steps in the advancement of technology will see their Drafters, Architects and Engineers spend many hours researching basic dimensions and shapes for their products since this was not done in previous years. Managers will then blame their staff for the slowness of executing the automation procedures. Other organizations spend money and time investing in advanced software which is advertised to solve all of our problems. When the libraries and subroutines in these third party routines do not measure up to our company's needs, then the business which wrote the code will begin to use our organization as Alpha and Beta testers to correct problems in their software and to add additional capabilities into their libraries. Companies that invest into the third party software that creates drawings will begin a journey down a road where the software determines the technology capabilities of the business. So although we can use engineering software to promote efficiency and reduce errors, the company managers need to have open ended program which allow for freedom of shape, placement and overall design.

Managers and programmers need to remember to keep their departmental capabilities sharp and not to depend on the strength of outside organizations. When Architects, CAD Operators and Engineers no longer have the ability to control the design, the product begins to suffer. We have

seen organizations over the year explain to their customers that their request was out of the range of the design software application. This should never occur, since the coded routine is just automating human processes and our personnel should always be able to create the drawing files without the code to maintain their skills. So accessing the individual technical aptitude is part of the customization process.

Seven Steps to Computer Aided Design Customization

When we take a company which uses computer aided design software to the next level, the intent is not to create for this company another problem just over the horizon in the next few years. There are organizations which have already taken their departments to higher levels, so we can learn from the mistakes and the successes of the programmers and managers who have proceeded up the technology ladder. As shown in Figure 12.1, there are seven levels in which Architectural and Engineering departments can be transitioning through; Drafting Boards, 2D CAD, Custom Blocks, Custom Programs for 2D CAD, 3D CAD, Custom Programs for 3D CAD and 4D CAD (Finite Element Analysis).

Level	Technology	Output
1	Drafting Boards	Paper Drawings using pencils, erasers and drafting machine
2	2D CAD	CAD drawings with lines, circles, arcs, and text
3	Blocks	Block libraries with windows, doors, fasteners
4	Custom Programs for 2D	LISP or VBA code to draw components or assemblies
5	3D CAD	CAD drawings with virtual parts and externally referenced assemblies
6	Custom Programs for 3D	LISP or VBA code to draw components or assemblies
7	4D CAD	Finite Element Analysis (FEA) software which can measure mass, heat and more

Figure 12.1 – Levels of Engineering Documentation

Throughout the world and in the every industrial zone, we will still find individuals using drafting boards, pencils, erasers and drafting machines to draw two dimensional orthographic drawings on paper. These professionals use exact hand and eye coordination to create precision part drawings, assemblies and Bill of Materials for their organizations. Drawings are made at a scale such as ¼ inch equals a foot, so the drafter can fit the entire multi-view drawing onto the industry prescribed piece of paper. Training for these individuals who display great manual dexterity range in the hundreds of the hours for learning lettering, pencil pressure for line weight and quick geometric calculations allowing them to construct complex shapes. As we entered into the age of the microcomputer from the 1970's to the 1980's, 2D CAD software applications became powerful enough to create orthographic views. Now complex shapes such

as screw threads and all their intricacies can be planned swiftly, so no wonder technicians promoted the benefit of two dimensional computer aided design to their supervisors.

2D drafting on a computer essentially became the electronic substitute for the drafting board. The earliest versions of computer aided design software had commands such as line, circle, arc, and text. Although we could add entities into drawing file representing any shape very quickly, the biggest advantage was with commands such as Move, Copy, Erase and Array. Modifications of existing entities were simpler for a drafter than making the pencil drawing. Another marvelous addition was the dimensioning tools, where horizontal and vertical measurements were instantly recorded in the file when selecting three points. Although the processors in the earliest computers had limited capability to compile the drawing file, 2D CAD eventually became the standard in the industry by the 1990's.

Now many departments needed to increase the pace of diagramming floor plans and assemblies by using formerly drawn details which are called blocks. Libraries can be built and categorized by labels such as windows, doors and sinks. Inserting these pre-constructed views into a larger drawing reduce drawing times and errors. The next step was to write custom programs which allow for more details and to have them appear quicker. Department managers ordered LISP routines written which would give more details and blocks.

Then with 3D CAD applications, we had the advantage of reading a single three dimensional solid over deciphering various 2D entities such as lines, circles, arcs and text on the computer aided design graphical display. In most cases a solitary entity representing a complex shape has attributes such as mass and can be identified with a single part number. Some companies use blocks or regions so that a single part number can be used to identify the component, but in a plan set, more than one orthographic view of the same part can be shown and in these cases the programmer has to make exceptions when counting for the bill material. Drawings that contain three dimensional parts do not have this problem and the engineering software is capable of accurately calculating all quantities. The most current analysis shows that numerous industries are not capable of supporting 2D to 3D transition, since none of their sub assemblies are single solid entities. In our many years of experience, suppliers will only provide the three dimensional drawings of their products if the end manufacturer or construction company requires more technology, so typically the end user must stipulate the creation of the 3D model.

There are 3D routines that will create 2D orthographic views from the 3D part or assembly. Programs like Tenview and TenviewAssembly will create orthographic views of any solid part or externally referenced assembly. Other 3D programs will turn 2D floor plans into 3D houses. A few companies have provided the code to create their products by answering a few questions. This is one of the newest arenas in the world of custom engineering, so the groups that are venturing into this region of customization should expect some investment in research and development where there will not be any return except experiencing the hard lessons of learning from one's mistakes. Of course when our discoveries separate us from the competition then we will also reap from those benefits.

4D software which is far away from many organizations will allow the computer specialist to test their assemblies in a virtual environment. The file can start out as a 3D part or assembly drawing, but with the addition of a Finite Element Analysis program, we can test the product with interactions such as with forces and temperature. If the virtual product fails the test then

changes can be made to the part's wall thickness or shape. When the virtual invention passes the examination then we will be far more certain that the real structure is capable of surviving. These types of long calculations were done by hand in earlier years and the person who did them needed to know more advanced math than a typical Architect or Designer.

The Fundamentals of Visual Basic Applications

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. If we were making a sketch of a stamping as shown in Figure 12.2, then we would label each point as before, the label the X grid, the Y grid and possibly the Z grid, any constant or variable dimensions, and notes. This may require more than one visual rendering of the product. But unlike Visual AutoLISP, Visual Basic projects also require the sketch of the form as shown in Figure 12.3, which will show the labels, textbox, check boxes, radio boxes, images and command buttons. 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.

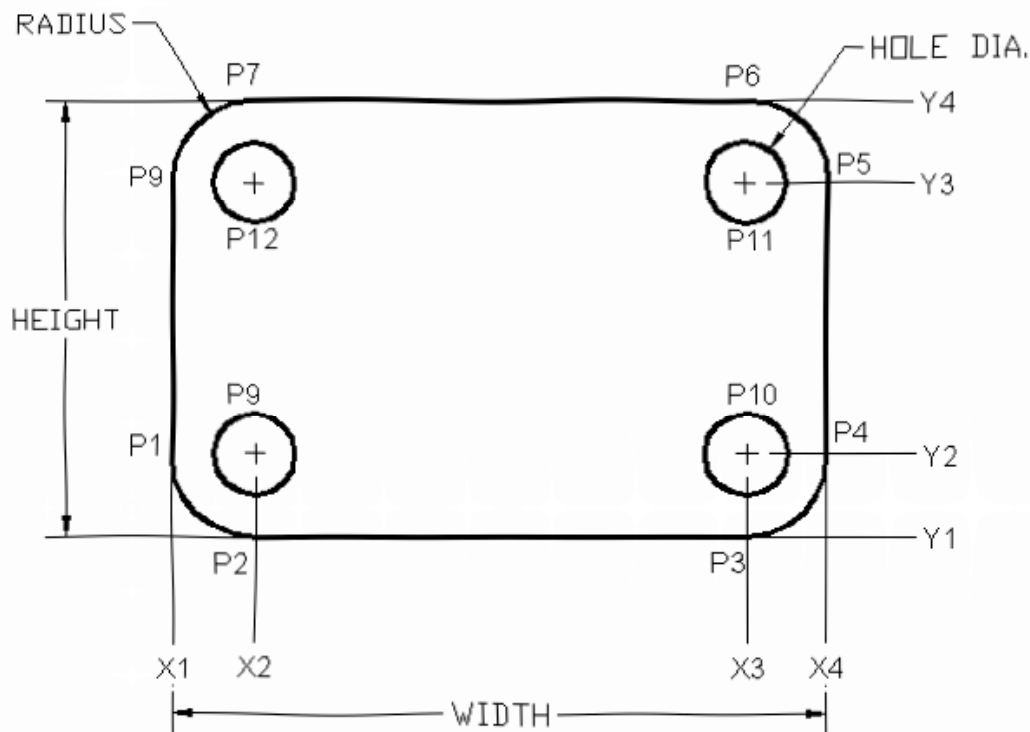


Figure 12.2 – Sketch of a Part Ready to Program

If we have found the Visual AutoLISP editor to be useful then we will discover that the Visual Basic Editor provides similar capabilities. Unlike LISP, the Visual Basic Editor has many more features since we will link the code to a Microsoft Window when we build a form, which we will do right from the beginning. From the beginning of inserting the form when building a

project, immediately we need to refer to our sketch. In any department, we would train new programmers initially in the art of form building. Using the editor, we insert and size the form, and selecting the Toolbox, we will place all the various input tools and properly label them. Whenever we place input tool, the properties window will display a list of every attribute associated with the tool, and we will take every effort arrange the tool by performing such actions as naming, labeling and sizing the visual input device.

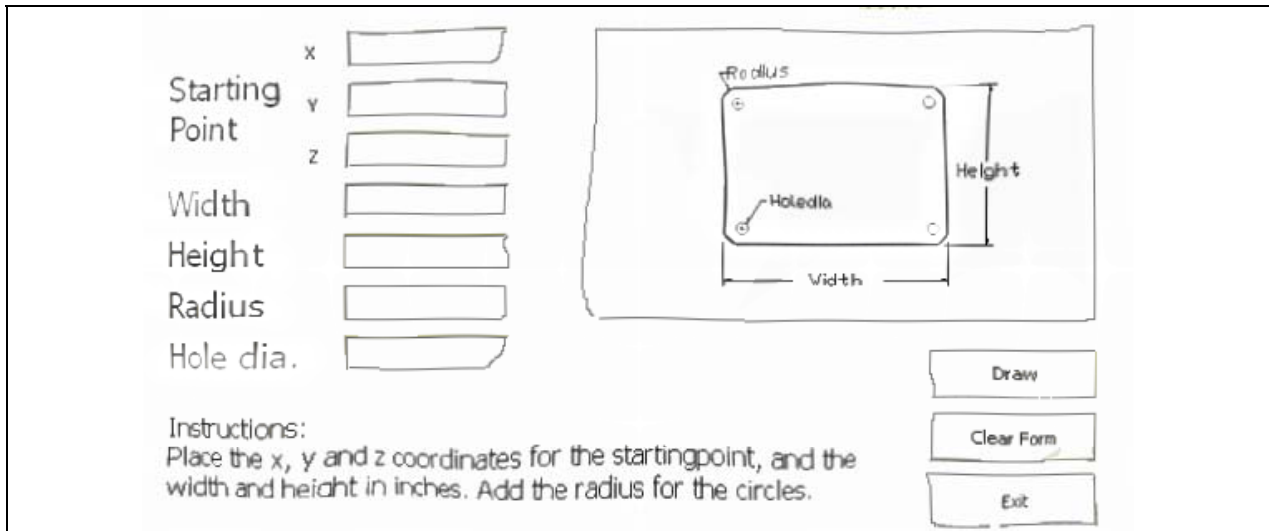


Figure 12.3 – Sketch of the Form

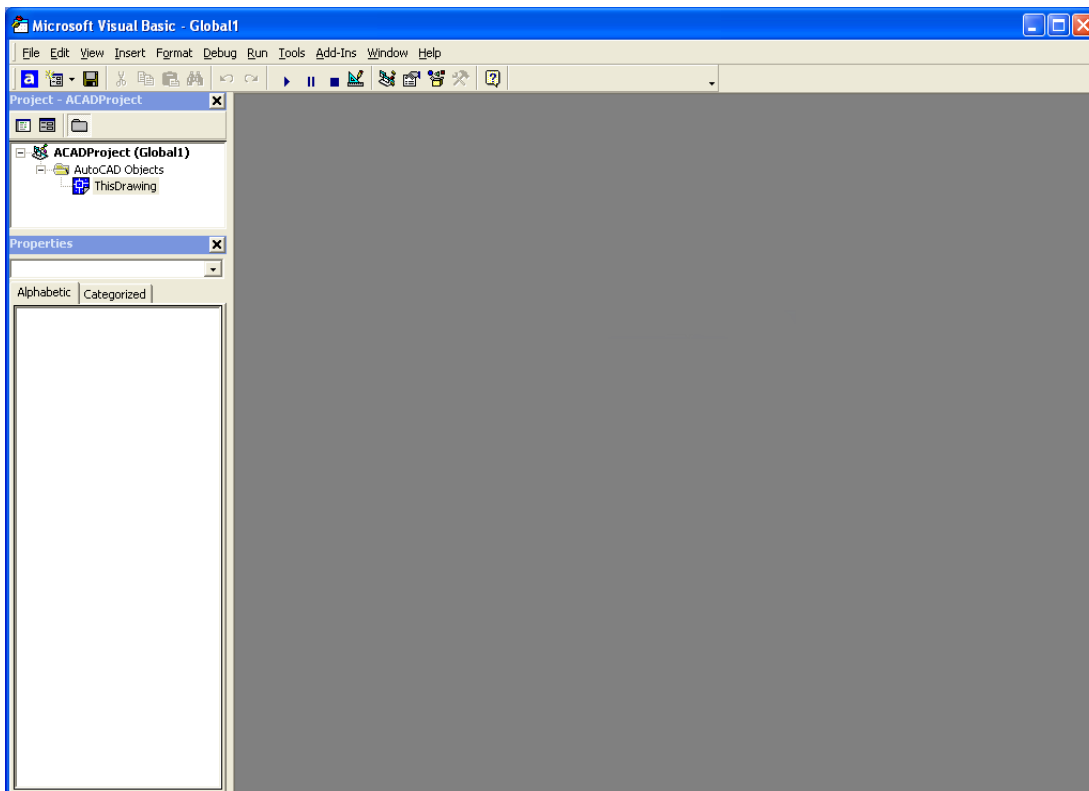


Figure 12.4 – The Visual Basic Editor

Now that we have an idea what the form will look like, we can start building the form in AutoCAD by selecting Tools, then Macro, and Visual Basic Editor. A new project will start in the editor and shown in Figure 12.4.

In the Visual Basic Editor, select Insert and then Userform to start to build an input window for the program. We can adjust the size of the form by placing the mouse on one of the grips in resizing the dialog box. When the form appears in the editor, the Visual Basic Toolbox will also come into view. When we pick any of the Control tools, at that time we can place the object on the form. Or finish form is shown in Figure 12.5.

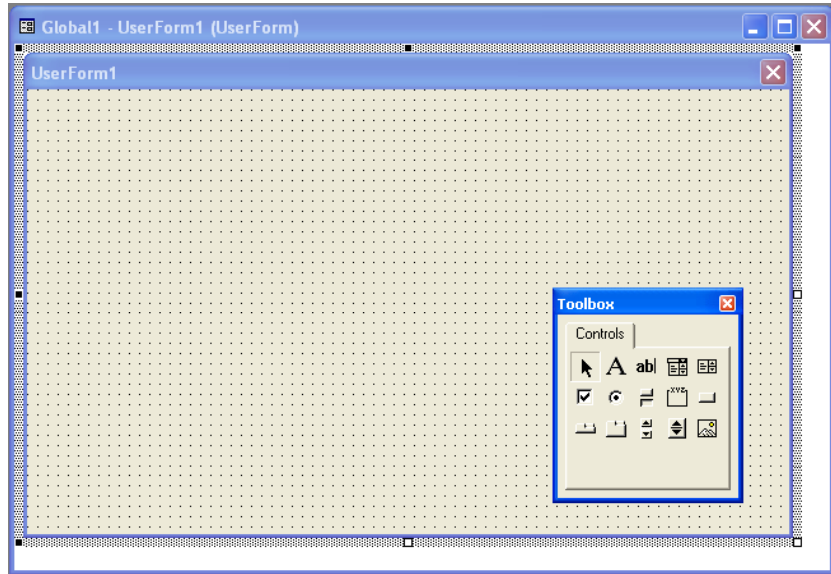


Figure 12.5 – A Blank Form and Toolbox

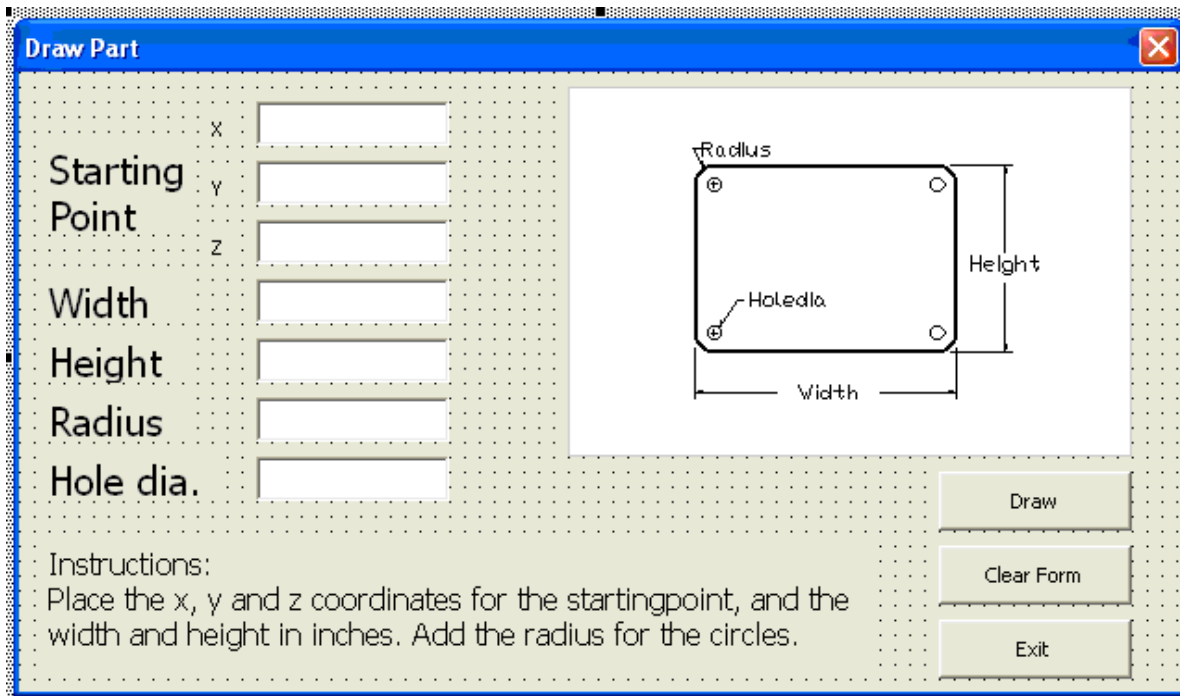


Figure 12.6 – Building a Visual Basic Form

Our next step after building the form and labeling the controls is to build a program. Many times we will pencil out the code to get our thoughts together. As with the Visual AutoLISP programming, we continue to use structure when building a routine. When starting the Visual

Basic code, now is a time to declare the variables using the Dim statement. We do this by referring to the sketch that shows the points and the grid. The variables such as X1 and Y1 contain a single integer, but we will proclaim the points with X, Y and Z coordinates with the letter P followed by 0, 1 or 2.

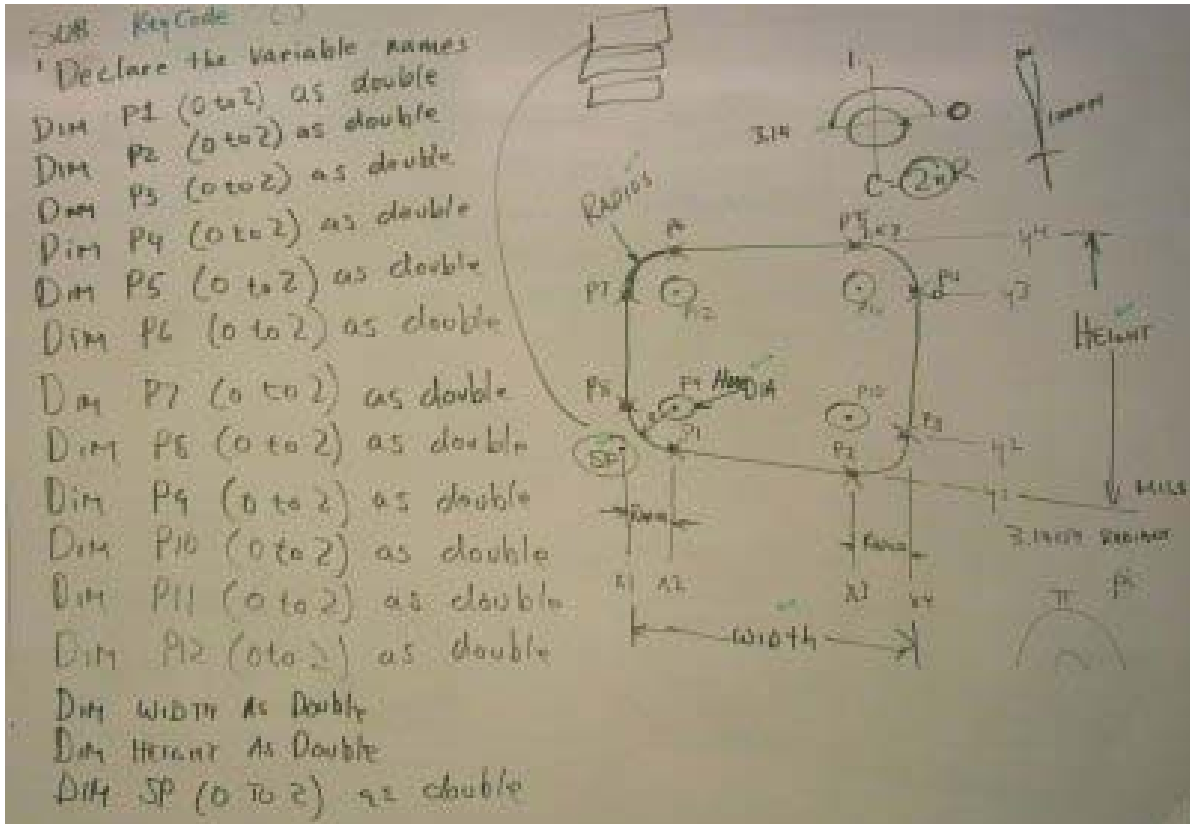


Figure 12.7 – Rough Notes Declaring the Variables

Programmers are able to write their routines in just about any location throughout the world by having a notebook and a pencil to record their ideas. In Figure 12.7, we see a small sketch and the beginning of the calling out what variables or needed. Sometimes we make programming pads that contain each step of the visual basic program such as sections showing the sketch, variable declarations, math, point assignments, and drawing. When a senior programmer has their thoughts on paper, they can allow anyone in the department familiar with the Visual Basic Editor to build a form and type the code.

Another area which we have not discussed is labeling the Input Controls. There are standard naming prefixes which we place in front of the labels. Forms have the prefix “frm” so our form would be named frmPart as shown in Figure 12.8. In just a few hours we can become very proficient at naming Controls by just looking at the name on the sketch.

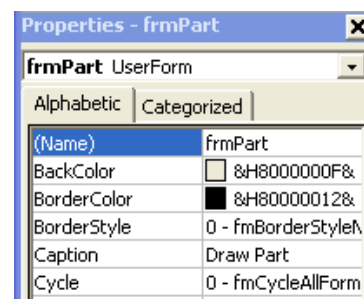
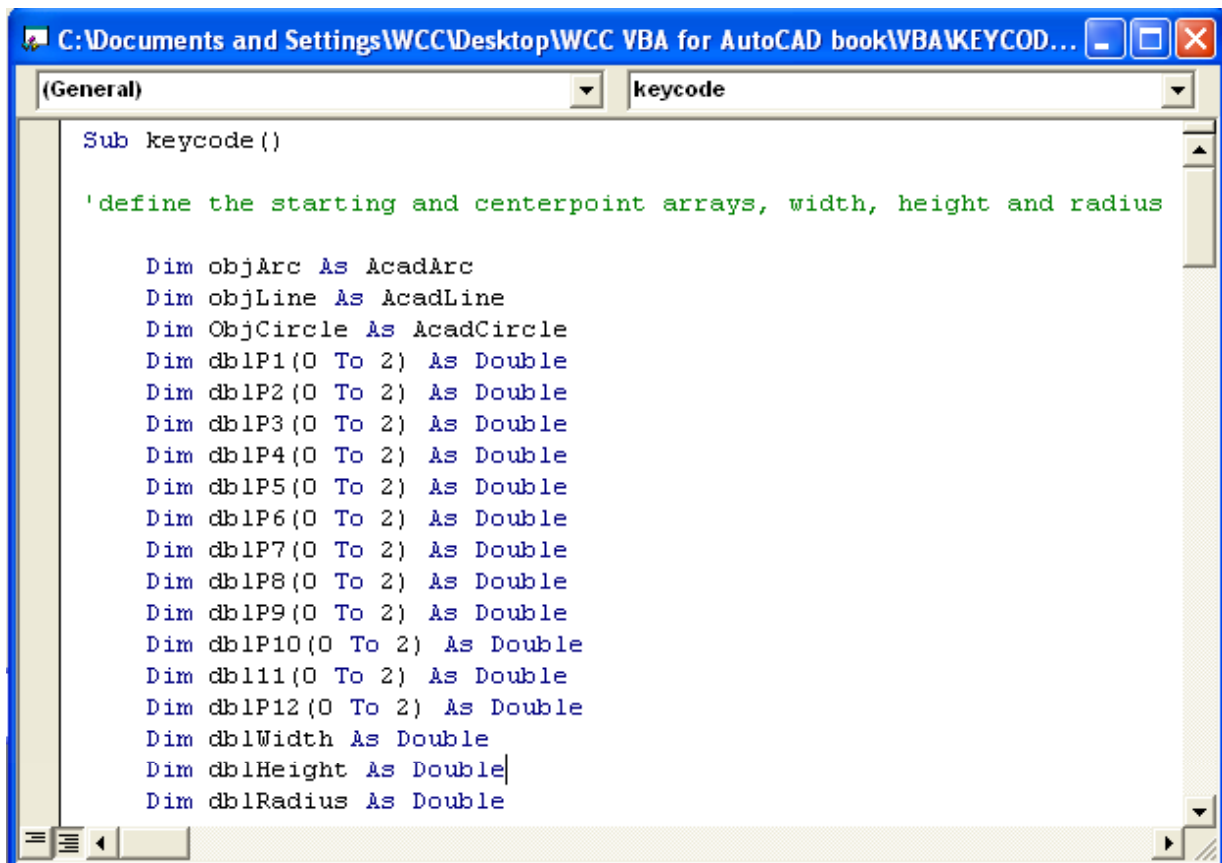


Figure 12.8 – The Visual Basic Properties

Other common Control prefixes are **cmd** for command buttons, **chk** for checkboxes, **cbo** for combo boxes, **img** for images, **lbl** for labels, **lst** for list box, **mnu** for menus, **opt** for option button and **txt** for textboxes.

Variables also have common naming conventions are using the following prefixes:

Cur for currency, **dtm** four date (time), **dbl** for double integer, **int** for integer, lng for a long integer, **obj** for object, **sng** for single integer, **str** for string. We can see in Figure 12.9, the beginning of our coded routine where we declare the variables that the names are preceded by the correct prefix. The coded names are a concatenation of the prefix and the tag from the sketch.



```
Sub keycode()  
  
'define the starting and centerpoint arrays, width, height and radius  
  
Dim objArc As AcadArc  
Dim objLine As AcadLine  
Dim ObjCircle As AcadCircle  
Dim dblP1(0 To 2) As Double  
Dim dblP2(0 To 2) As Double  
Dim dblP3(0 To 2) As Double  
Dim dblP4(0 To 2) As Double  
Dim dblP5(0 To 2) As Double  
Dim dblP6(0 To 2) As Double  
Dim dblP7(0 To 2) As Double  
Dim dblP8(0 To 2) As Double  
Dim dblP9(0 To 2) As Double  
Dim dblP10(0 To 2) As Double  
Dim dbl111(0 To 2) As Double  
Dim dblP12(0 To 2) As Double  
Dim dblWidth As Double  
Dim dblHeight As Double  
Dim dblRadius As Double
```

Figure 12.9 – The Visual Basic Code

After the code and is typed, we can launch the program by pressing the function key F5 on the keyboard. The program starts in AutoCAD with the dialog box to draw the part appearing on the graphical display. In our example we type the starting point in three textboxes, one each measurement on the X, Y and Z axis. Then we enter the width, height, radius and the hole diameter into the appropriate textboxes referring to the image or instructions if we need help. If we make a mistake, we can clear the input window using the “Clear Form” command button. If we did not want to make the part, we can press the “Exit” command button. But once we key the information shown in Figure 12.10, we can press the “Draw” command button to create the part. The finished drawing is shown in Figure 12.11 for the data that we typed into the dialog box.

Now that we know that the program is capable of making our parts, we have only made a single test of the code. You can get someone else in the department to test your program in multiple configurations many times looking for any errors. If our beta tester finds any errors, they can record them in an e-mail also including any comments about the form which may have caused them some confusion. We can make those changes are quickly and returning the next version of the code to the beta tester. Once we are confident that the program contains no errors, and then we can make a formal release within our group so everyone can utilize this time saving tool.

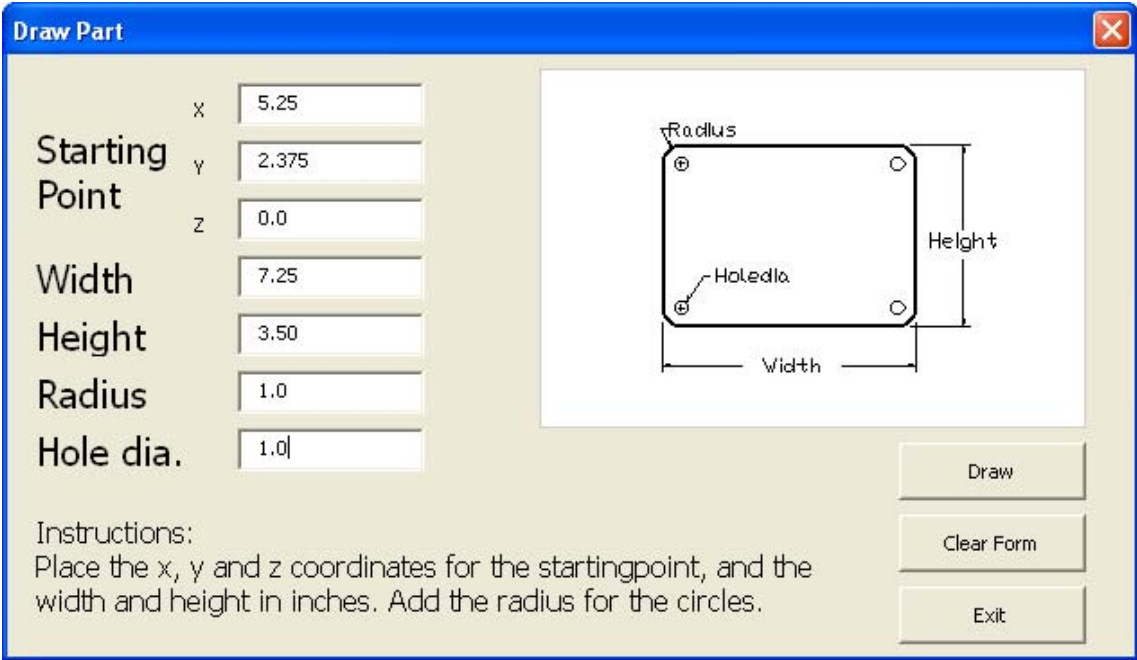


Figure 12.10 – Launching the Visual Basic Program

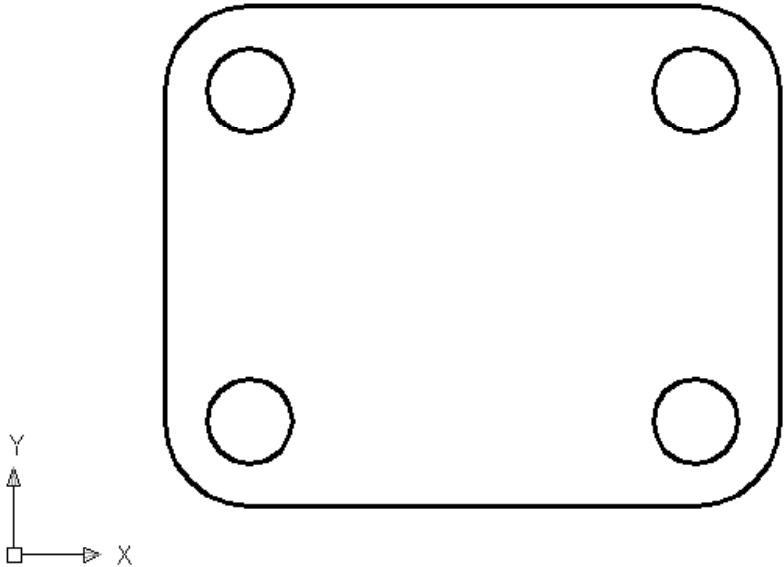


Figure 12.11 – The Finished Drawing

Creating Structure in Our Programs

In the last few pages, we have seen the very foundations of Visual Basic programming, but we do not have all the tools at our disposal to start making part and assembly drawings yet. The chapters in this textbook will begin with the simplest of problems in venture towards finished drawings. We will follow the steps shown in Figure 12.12, which are sketch the part and form, start the program, create the form, declare the variables, do the math, make the point assignments, draw the part and end the program. Learning to follow is structured path while we code just makes the process easier and more enjoyable.

Step 1	Sketch part and form
Step 2	Start the program
Step 3	Create the form
Step 4	Declare variables
Step 5	Do the math
Step 6	Point assignments
Step 7	Lets draw
Step 8	End the program

Figure 12.12 – Visual Basic Program Structure

Another factor for us will be the World Class CAD Challenge. Every exercise will confront us to increase our efficiency when sketching, coding and running the new Visual Basic programs. As in all of the other World Class textbooks, learning the organization skills will allow us to achieve finished code in record times and therefore give us new skills to complete orthographic views quicker. Worded below is a typical World Class CAD Challenge that we find with each drill.

*** World Class CAD Challenge * - Create a New AutoCAD file and sketch the entire problem on the proper layers, using proper dimensions and finally placing the points and x and y grid on the drawing. Save your AutoCAD sketch. Open the Visual Basic Editor and code the problem using the structured coding method. Save the code as named on the problem sheet. Send your copy of your drawing and code for verification to the authors of these problems to have your name and location posted on the website.**

Figure 12.13 – Sample World Class CAD Challenge

Programs That Do It All

We are not mistaken that by the time we finish this one textbook, we will have every skill to draw an entire drawing after answering a series of questions as seen in an Input form. Our instructors have the experience from working on previous professional projects to create entire drawings and then after more work, we can make entire set of prints to finish a complete release of architectural or engineering documents. We have been in offices where our code made 400 drawings completely after answering 15 to 20 minutes of questions. The entire morning the

computers are running and sending files to the printers, so the design team and customer can review the work. When changes are made, either the input data can be reinserted and recompiled or simple changes can be made in the drawing set physically or by a smaller routine. Companies that utilize these capabilities can increase their percent of custom work since the design team has the luxury to improve on their products.

Companies using Custom Engineering Programs

Are companies using Visual Basic programs to create new designs? Yes, we have graduates that have automated the creation of drawings devising routines for title blocks, bill of materials, designing manufacturing robots, making production drawings for custom windows, detailing NEMA enclosures for electrical control panel and making electrical wiring diagrams for Programmable Logic Controllers.

When a person introduces automation in a department by writing a routine to create an entire drawing, the other individuals in the department will probably be somewhat skeptical thinking that only someone with special insight can accomplish such a task. The fact is that anyone who can organize their thoughts and use a structured process is capable of accomplishing the entire project unaccompanied. We have seen third party organizations attempt to write code for a business where they will slowly learn the procedures that the company goes through to make a drawing. Honestly, a properly trained programmer, who finishes the material in this textbook, will do a better and faster job than an outside organization. Remember, just like any other process that an architectural or engineering group attempts, the technology must be understood by the members of our department and revised periodically as the organization changes. Purchasing third party or in the box applications from outside suppliers usually does not meet most company's needs.

Company managers typically want a group of Computer Aided Design (CAD) programmers creating custom routines versus having one individual with the knowledge and capability to accelerate their drawing processes and give their customers choices. So when we learn to program engineering code, we want to develop a programming partner simultaneously in our organization. That individual will grow in the discipline with us, and the new skill set is now more likely to spread among others in the department. Next, document the code and place plenty of comments in the routines that are written in simple English so that anyone one in the work area can understand the phrase. Check the finished drawing as per our normal company standard operating procedure, and when the print returns from the plant floor or work site with any corrections, determine whether these were the result of a programming error. If the mistake was in the code, correct the infraction and retest the program. As in any successful area of learning, the more hands and eyes on the finished product, the better the machine or code will become. For years to come, training Engineering programmers will be imperative to the success of our organizations.

The Future of Engineering Programming

When we look back, our first large undertaking was taking a engineering group through the

process of writing routines that would make hundreds of outline drawings that were complete with a front and side orthographic view of an complex assembly, showing dimensions, notes, border and title block. On each assembly drawing, the salesperson could input custom text and the routine would refer to data from a federal standard to space the text across the face of the assembly. There were less training material for engineering programmers in those years, and we would telephone the experts in the field to get their opinions on different approaches. With a lot of hard work, we released the first code and the sales department made thousands of drawings in the first year of the program release, which was a higher production number of prints than coming out of our main engineering departments.

For five years after that initial set of routines, our group of coders released hundreds of programs that would make many of the drawings in the department. There are a few reasons which might push an architectural or engineering group into programming. In our case, productivity in the business was increasing faster than we could hire and train personnel, so we bet on this newer technology to counter for office space and availability of trained designers. Another way of thinking was that the routines would do repetitive types of drawings to support custom product lines that engineers and designers felt were unexciting to do. Also, the sales and marketing team in our region wanted quotes and drawings in 24 hours.

Our branch of a major corporation was growing exponentially and we were always just in time or slightly behind in our print release schedule. As the company was growing, especially in the area of custom work, the production of print sets was paramount to keep the business running. We could spend a few months to get new personnel acquainted with a certain product line and multiple years training them in all the aspects of computer aided design. We need to remember that most universities spend the smallest amount of time training students in computer aided design, so this is a very common problem in the workplace. One could have to say that writing custom code to create unique drawings was forced upon us. So a group of three designers and one electrical engineer became the core group in writing the routines.

One of the smartest steps taken early in the process was to separate ourselves from software development in the other departments. After many years of study in this technology, our group is convinced more than ever that engineering programmers need to be right in the mix of the other design disciplines in order to create high-quality code. We have seen other programming groups struggle when they are separated from the main engineering departments they support. Also, writing the routines today and having the code working in the department tomorrow is a real world solution to the demand of print output. When a professional receives the finished code that has nearly unlimited potential to make thousands of drawings, they can be successfully integrating the new routine in minutes. The difference is astounding and shows in the weekly engineering production report.

Sales departments need color prints with actual data for their customers and not just an advertising sheet with a table displaying possible variations. Many of the clients have a difficult time determining what they will be receiving and expect more service for their money. Our sales force wanted quotes and drawings in a single day. These days, programmers are using the Visual Basic Application routines to make that step simpler. Creating forms as we have seen resulting in an input method that would enable any salesperson to create a detail drawing for the customer that meets all of the organization's and industry's standards.

Finally, we would be remiss in our CAD training method not to train every student in a feature that is in every version of AutoCAD.

Welcome to the world of Visual Basic programming.

Charles Robbins

*** World Class CAD Challenge 04-00 * - Complete this textbook in 40 hours of classroom training. Pass your Programming Levels 1, 2 and 3 certifications to be ranked among the best in the world.**