Chapter 7

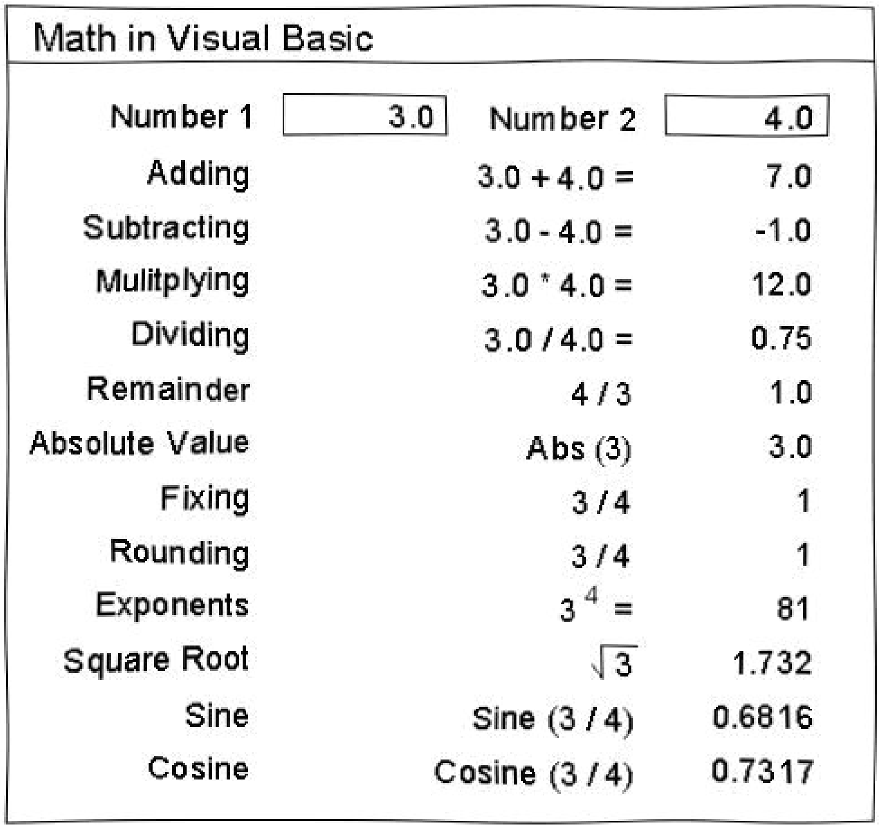
VBA: Math Functions

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

- **Writing Math Equations in Visual Basic**
- **Opening the Visual Basic Editor in AutoCAD**
- **Laying Out a User Input 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**
- **Declaring Variables in a Program with the Dimension Statement**
- **Setting Variables in a Program**
- **Adding Numbers in Visual Basic**
- **Subtracting Numbers in Visual Basic**
- **Multiplying Numbers in Visual Basic**
- **Dividing Numbers in Visual Basic**
- **Finding Remainders in Visual Basic**
- **Computing Absolute Values in Visual Basic**
- **Fixing Numbers in Visual Basic**
- **Rounding Numbers in Visual Basic**
- **Computing Exponents in Visual Basic**
- **Computing Square Roots in Visual Basic**
- **Computing Sine's in Visual Basic**
- **Computing Cosines in Visual Basic**
- **Resetting the Data with the cmdClear Command Button**
- **Exiting the Program with the cmdExit Command Button**
- **Executing a Subroutine with the cmdDraw Command Button**
- **Running the Program**

Writing Math Equations in Visual Basic

There are many arithmetic and geometric functions that we can utilize in programming a Visual Basic. Some arithmetic functions include adding, subtracting, multiplying and dividing, which are obvious for everyday work. The absolute value, exponent and square roots are for more serious technicians. Even greater code writers that are computing angles are using sine and cosine. These functions are easy to learn and there are examples in each section to practice every function in this chapter. Figure 7.1 shows a layout of the form we want to create to see how math functions operate in Visual Basic.


Math in Visual Basic

Number 1	3.0	Number 2	4.0
Adding		3.0 + 4.0 =	7.0
Subtracting		3.0 - 4.0 =	-1.0
Mulitplying		3.0 * 4.0 =	12.0
Dividing		3.0 / 4.0 =	0.75
Remainder		4 / 3	1.0
Absolute Value		Abs (3)	3.0
Fixing		3 / 4	1
Rounding		3 / 4	1
Exponents		3^4 =	81
Square Root		$\sqrt{3}$	1.732
Sine		Sine (3 / 4)	0.6816
Cosine		Cosine (3 / 4)	0.7317

Figure 7.1 – Sketch of the Visual Basic Form

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 2008, but we just finished using all the programs in this text with a group programming in AutoCAD 2000. Their drawings were automatically made just as efficiently as if they were using the most recent version of the Autodesk software.

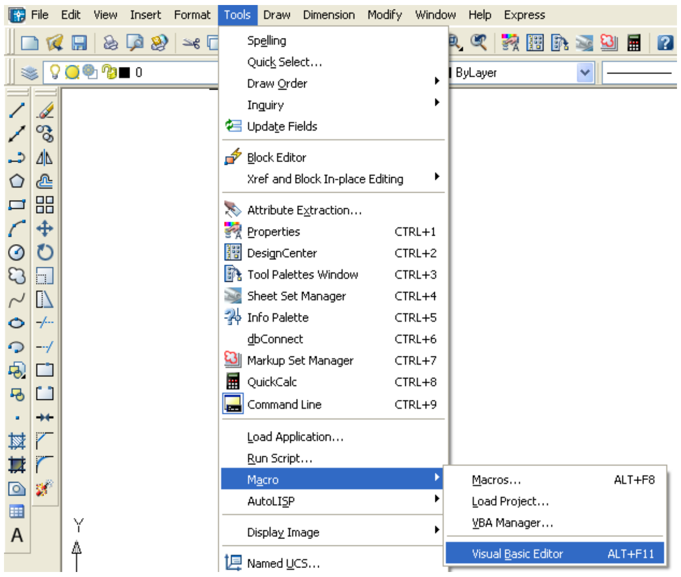


Figure 7.2 – Launching the Visual Basic Editor

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

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

For those individuals with previous Visual Basic experience, the Visual Basic Editor in AutoCAD has the same layout as in other VB programs. The Menu Bar contains tools for our use as well as the four toolbars, 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.

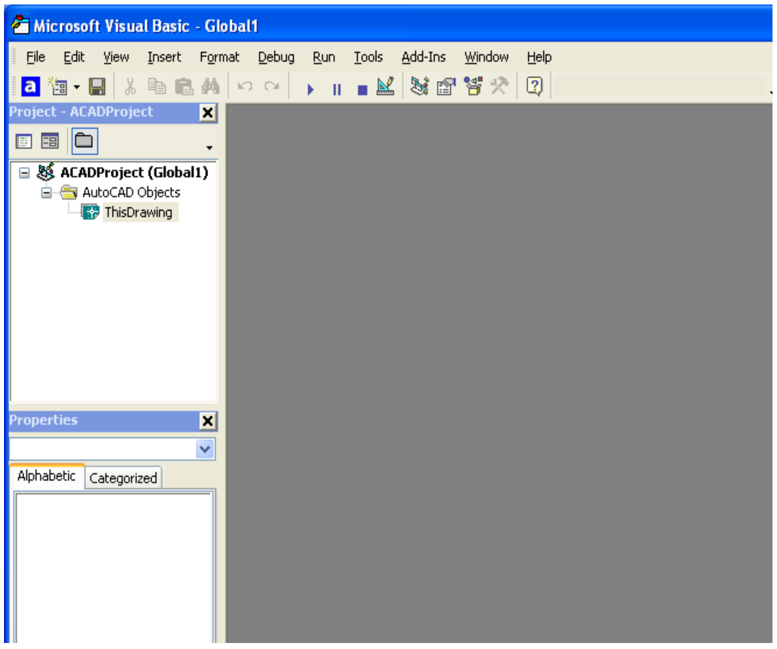


Figure 7.3 – The Visual Basic Editor

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 "Math Functions". The file has an extension called *dvb* which means DCL and Visual Basic programs as shown in Figure 7.4.

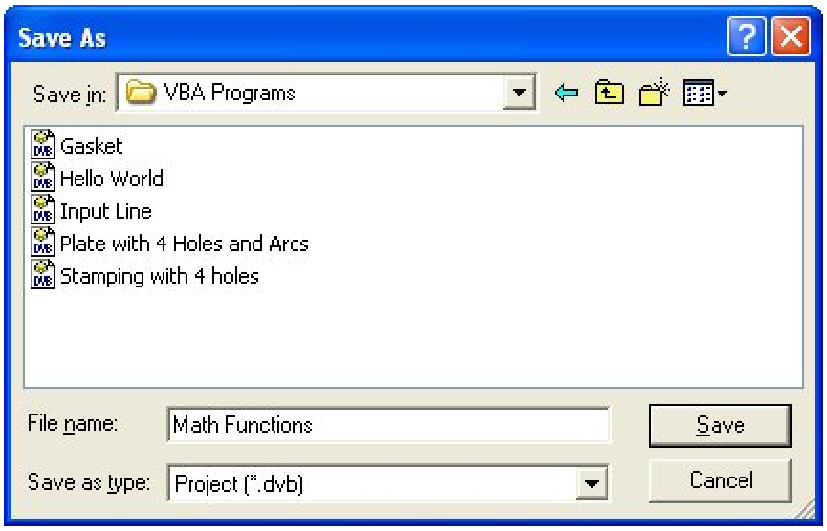


Figure 7.4 – Saving the Math Functions Program

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 7.5. 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.

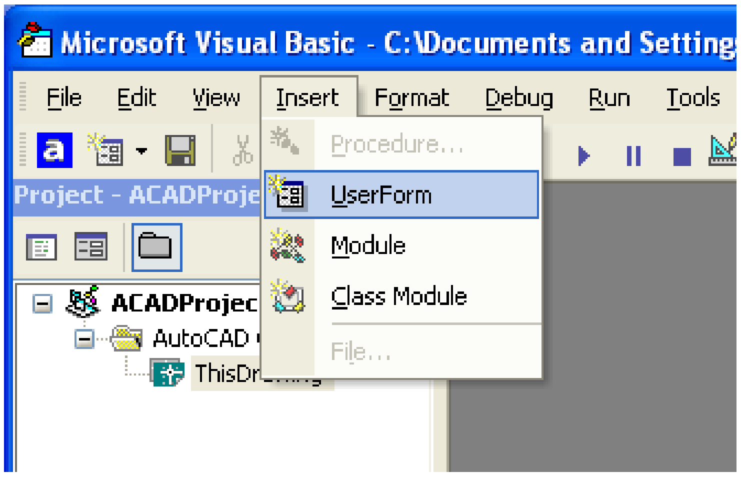
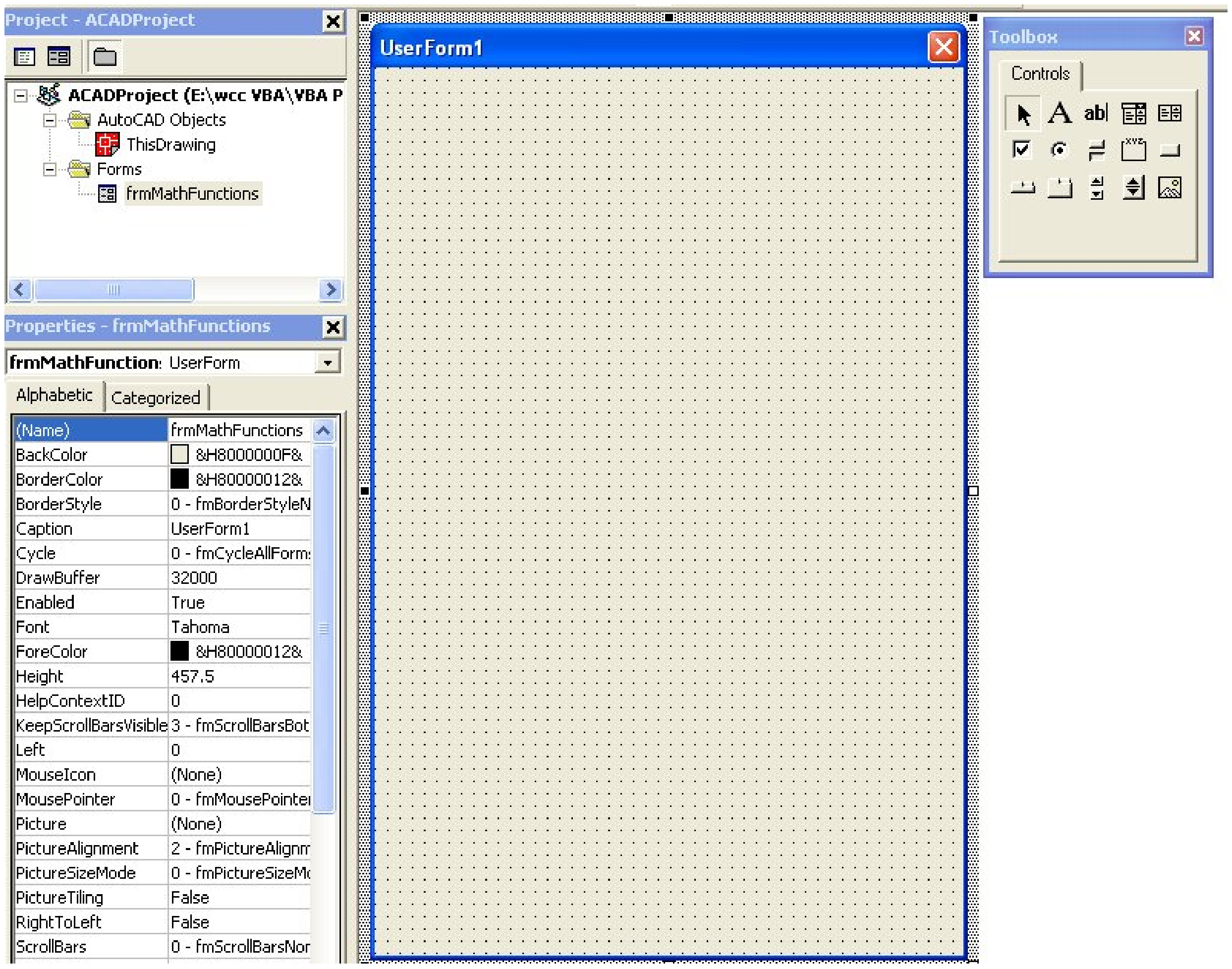


Figure 7.5 – Inserting a User Form

Change the name of the user form to frmMathFunctions. 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&".



Properties - frmMathFunctions

frmMathFunction: UserForm

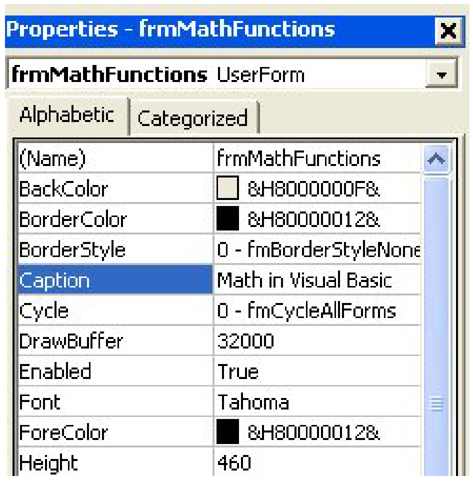
Alphabetic | Categorized

(Name)	frmMathFunctions
BackColor	☐ &H8000000F&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleN
Caption	UserForm1
Cycle	0 - fmCycleAllForm:
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	457.5
HelpContextID	0
KeepScrollBarsVisible	3 - fmScrollBarsBot
Left	0
MouseIcon	(None)
MousePointer	0 - fmMousePointer
Picture	(None)
PictureAlignment	2 - fmPictureAlignm
PictureSizeMode	0 - fmPictureSizeM
PictureTiling	False
RightToLeft	False
ScrollBars	0 - fmScrollBarsNor

Figure 7.6 – Designing the Math Function Form in Visual Basic

Next, we will change the **Caption** in the Properties pane to **Math in Visual Basic** to agree with the sketch in Figure 7.1. Go ahead and change the form in two other aspects, Height and Width.

Alphabetic	
(Name)	frmMathFunctions
BackColor	&H80000013&
Caption	Math in Visual Basic
Height	460
Width	300


Properties - frmMathFunctions

frmMathFunctions UserForm

Alphabetic | Categorized

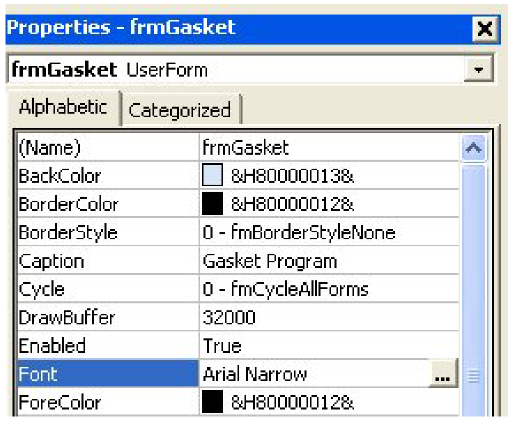
(Name)	frmMathFunctions
BackColor	☐ &H8000000F&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	Math in Visual Basic
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Tahoma
ForeColor	■ &H80000012&
Height	460

Figure 7.7 – 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 Arial Narrow and the Font size to 16 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.


Properties - frmGasket

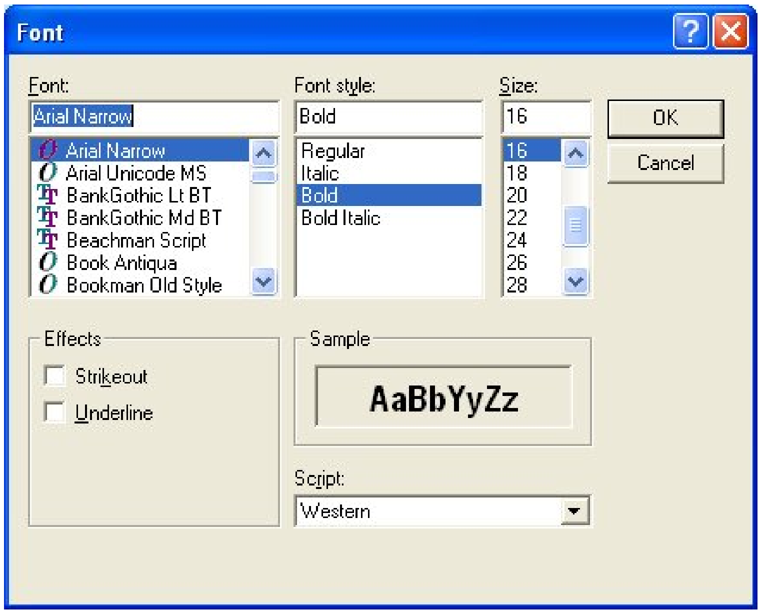
frmGasket UserForm

Alphabetic | Categorized

(Name)	frmGasket
BackColor	⬜ &H80000013&
BorderColor	■ &H80000012&
BorderStyle	0 - fmBorderStyleNone
Caption	Gasket Program
Cycle	0 - fmCycleAllForms
DrawBuffer	32000
Enabled	True
Font	Arial Narrow ...
ForeColor	■ &H80000012&

Figure 7.8 – Changing the Font to Tahoma

We will select the Arial Narrow font, Bold font style and 16 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

Font: Arial Narrow

Arial Narrow, Arial Unicode MS, BankGothic Lt BT, BankGothic Md BT, Beachman Script, Book Antiqua, Bookman Old Style

Font style: Bold

Regular, Italic, Bold, Bold Italic

Size: 16

16, 18, 20, 22, 24, 26, 28

OK

Cancel

Effects: Strikeout ☐ Underline ☐

Sample: AaBbYyZz

Script: Western

Figure 7.9 – The Font Window in Visual Basic

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.

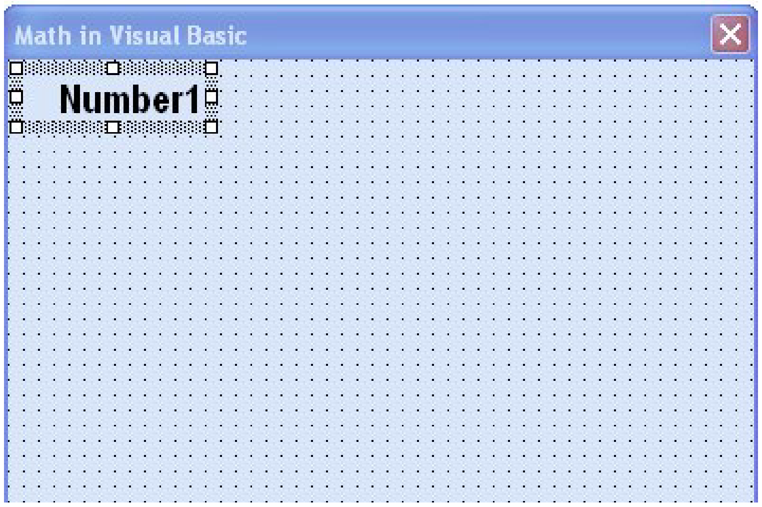


Figure 7.10 – The Finished Label on the Form

For the first label, set the name as **lblNumber1** and the caption as **Number 1**. The width of the textbox is 70 and the height is 18. For labels on the left side of the textbox, set the TextAlign attribute to right 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 7.11.

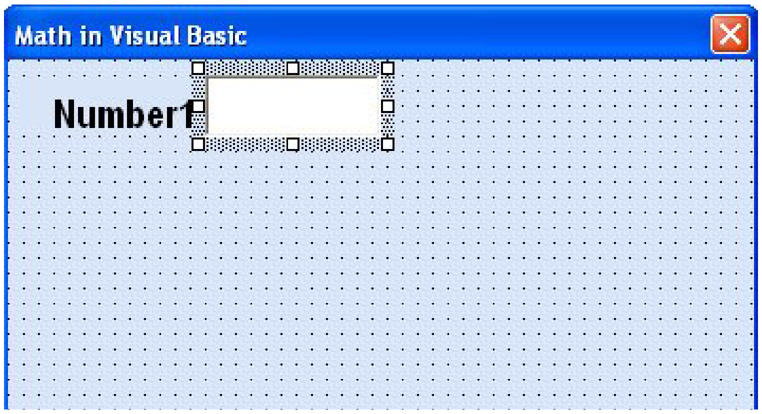
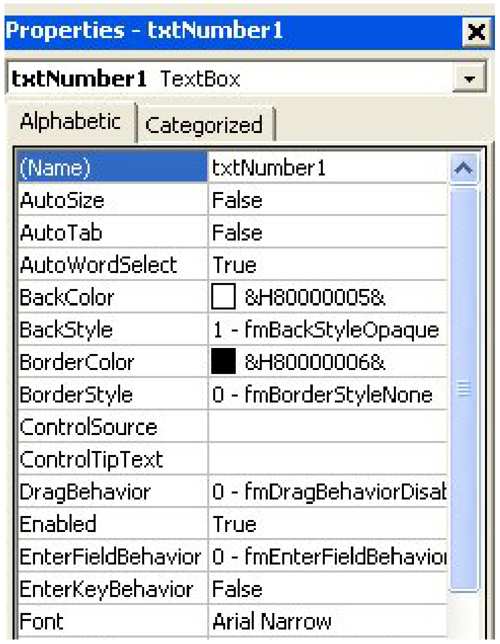


Figure 7.11 – Placing a TextBox on the Form

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 **txtNumber1.**

Alphabetic	
(Name)	txtNumber1
Height	24
Width	70
TextAlign	Right

On all of the textboxes, we will align the text to the right by setting the **TextAlign** property to right align.



Properties - txtNumber1

txtNumber1 TextBox

Alphabetic | Categorized

(Name)	txtNumber1
AutoSize	False
AutoTab	False
AutoWordSelect	True
BackColor	□ &H80000005&
BackStyle	1 - fmBackStyleOpaque
BorderColor	■ &H80000006&
BorderStyle	0 - fmBorderStyleNone
ControlSource	
ControlTipText	
DragBehavior	0 - fmDragBehaviorDisat
Enabled	True
EnterFieldBehavior	0 - fmEnterFieldBehavioı
EnterKeyBehavior	False
Font	Arial Narrow

Figure 7.12 – Changing the (Name) to txtNumber1

Place another label called lblAdding below the Number 1 label with the caption “Adding”. Add an additional label to the right of the Adding label that will hold the addition problem containing number 1 and number 2. This label will not contain a caption as shown in figure 7.18. Right align the text in both labels.

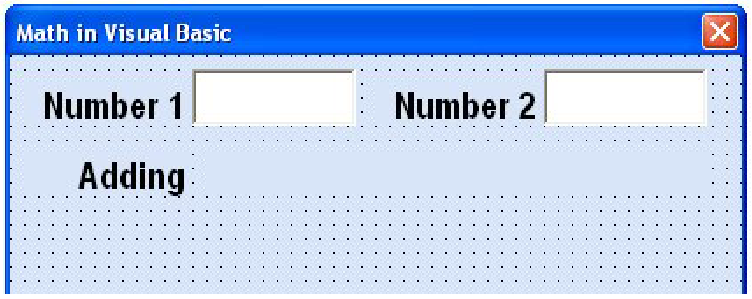


Figure 7.13 – Adding 2 Labels for Showing Adding

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 7.19.

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

Alphabetic	
(Name)	cmdCalculate
Caption	Calculate
Font	Arial Narrow
Height	42
Width	78

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

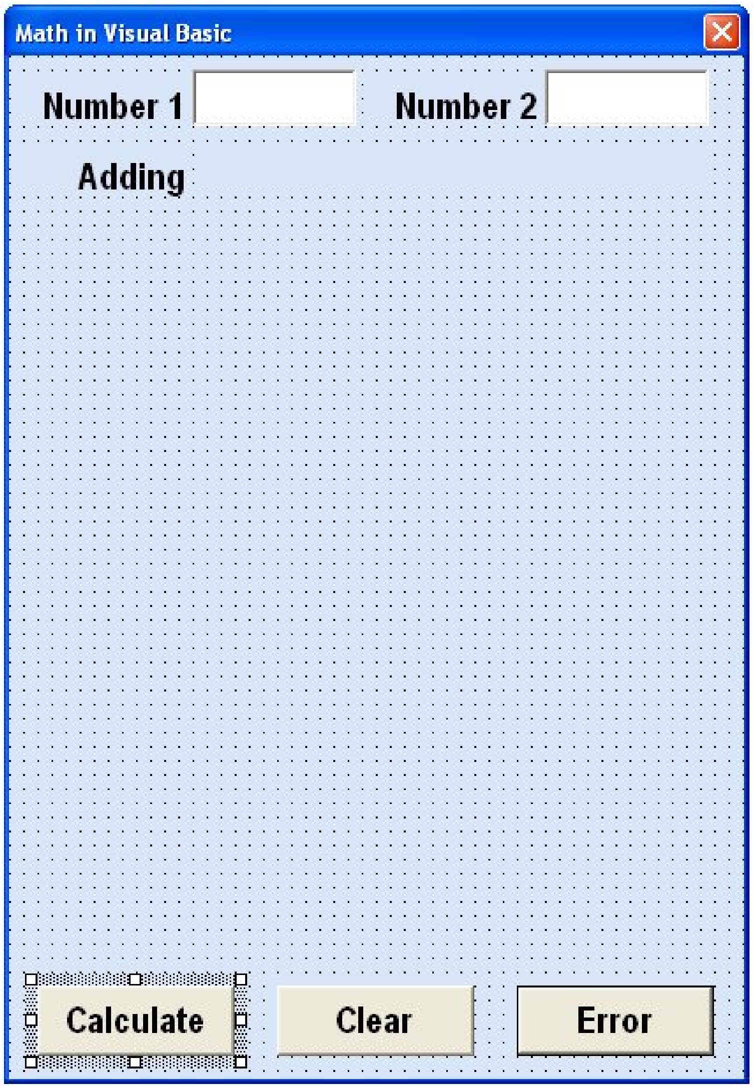


Figure 7.14 – Insert a Command Button onto a Form

Add a second Command button; named cmdClear is for clearing all the textboxes and problem labels. 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.

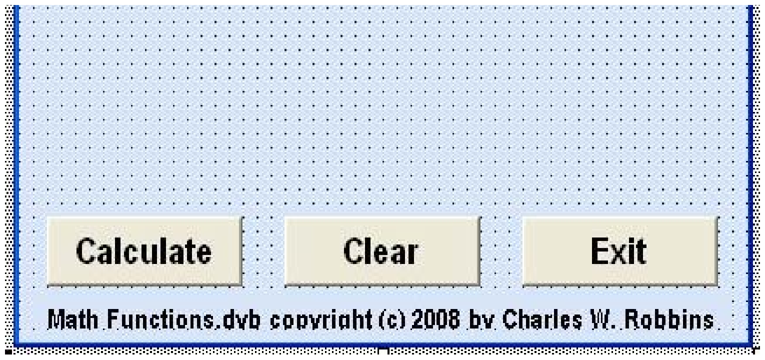
Adding a Copyright Statement to a Form

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.

'Math Functions Program.dvb - copyright (c) 2008 by Charles Robbins.

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



Math Functions.dvb copyright (c) 2008 by Charles W. Robbins

Figure 7.15 – Adding a Copyright Statement

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

Adding Comments in Visual Basic to Communicate the Copyright

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

To begin the actual coding of the program, double click on the Calculate command button to enter the programming list. At the top of the program and after the line of code with **Private Sub cmdCalculate_click ()**, 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:

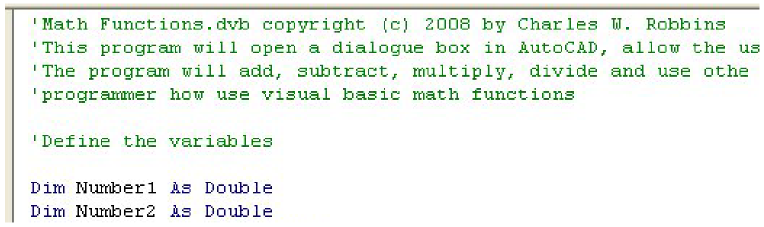
```
Private Sub cmdCalculate_click ()

'Math Functions.dvb copyright (c) 2008 by Charles W. Robbins
```

```
'This program will open a dialogue box in AutoCAD, allow the user to enter a two numbers
'The program will add, subtract, multiply, divide and use other math functions to train the
'programmer how use visual basic math functions
```

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 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.

In our program, we will set two variables to experiment with different math functions and one variable to hold the answer to the problem.



```
'Math Functions.dvb copyright (c) 2008 by Charles W. Robbins
'This program will open a dialogue box in AutoCAD, allow the us
'The program will add, subtract, multiply, divide and use othe
'programmer how use visual basic math functions

'Define the variables

Dim Number1 As Double
Dim Number2 As Double
```

Type the following code:

```
'Define the variables
    Dim Number1 As Double
    Dim Number2 As Double
```

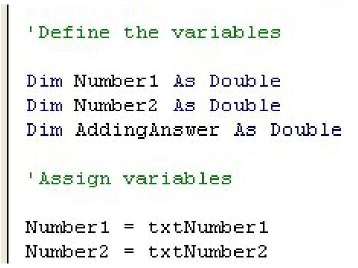
Figure 7.16 – Declaring Variables

Notice that the variable name 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.

Setting Variables in a Program

Next, we will set the variables using the equal function (=).

```
'Assign variables

Number1 = txtNumber1
Number2 = txtNumber2
```



```
'Define the variables

Dim Number1 As Double
Dim Number2 As Double
Dim AddingAnswer As Double

'Assign variables

Number1 = txtNumber1
Number2 = txtNumber2
```

Figure 7.17 – Setting the Variables

Adding Two Numbers in a Visual Basic

The first arithmetic function we will address is one of the most common, the adding function which is displayed by the icon +. The addition function allows us to add two or more numbers. The values of the numbers can be whole like 1,2,3… or decimals, positive or negative. Remember we can have more than two numbers like 2 + 3 + 7 + 4.

In this program, we will add the two variables that are holding the numbers, **Number1** and **Number2**. The variable **AddingAnswer** will equal the sum of the value in variable **number1** and with the value in variable **number2**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim AddingAnswer As Double
```

Then assign the variable **AddingAnswer** the results of adding Number1 plus Number2.

```
`Assign Variables
AddingAnswer = Number1 + Number2
```

By writing the addition problem to a label caption, we will display the answer to the problem when adding the numbers together. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
`Adding the numbers
lblAddingFormula.Caption = Number1 & " + " & Number2 & " = " & AddingAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Addition function.

Function	Name	Description
+	**Adding**	**The addition function will add two or more numbers**
Examples		
Using integers	**answer = 4 + 6**	Answers **10**
Using decimals	**answer = 2.3 + 5.1**	Answers **7.4**
Using negatives	**answer = 3 + -7**	Answers **-4**

Subtracting Numbers in a Visual Basic

The subtraction function is similar to the adding function, but the number following the first argument is subtracted from the first number. Again there is not a limit to the number of arguments in the math statement. An easy error to make is mistaking the negative sign attached to the front of a number constructing a negative value with the minus sign in the subtraction function. Be aware of these two symbols are their meaning to prevent a troubleshooting dilemma

In this program, we will subtract the second variable **Number2** from the first variable, **Number1** to equal the variable **SubtractingAnswer**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim SubtractingAnswer As Double
```

Then assign the variable **SubtractingAnswer** the results of subtracting Number2 from Number1.

```
`Assign Variables
SubtractingAnswer = Number1 - Number2
```

By writing the subtraction problem to a label caption, we will display the answer to the problem when subtracting the numbers. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
`Subtracting the numbers
lblSubtractingFormula.Caption = Number1 & " - " & Number2 & " = " & SubtractingAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Subtraction function.

Function	Name	Description
-	**Subtracting**	**The subtraction function will subtract a number from the preceding number**
Examples		
Using integers	**answer = 4 - 6**	Answers **-2**
Using decimals	**answer = 2.3 - 5.1**	Answers **-2.8**
Using negatives	**answer = 3 - -7**	Answers **10**

Multiplying Numbers in a Visual Basic

We can use the multiplying function to multiply two or more numbers together using the asterisk icon. After practicing the simple examples in the multiplying table, we can be experimenting with compound expressions on the AutoCAD Command line like:

answer = (4 + 3) * (7 – 3)

After computing 4 + 3 = 7 and 7 -3 = 4, the computer calculates 7 * 4 = 28. Very neat.

Now, we will multiply the first and second numbers.

In this program, we will multiply the two variables that are holding the numbers, **Number1** and **Number2**. The variable **AddingAnswer** will equal the product of the value in variable **number1** and the value in variable **number2**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim MultiplyingAnswer As Double
```

Then assign the variable **MultiplyAnswer** the results of Number1 times Number2.

```
`Assign Variables
MultiplyingAnswer = Number1 * Number2
```

By writing the addition problem to a label caption, we will display the answer to the problem when multiplying the numbers together. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
`Adding the numbers
    lblMultiplyingFormula.Caption = Number1 & " * " & Number2 & " = " & MultiplyingAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Multiplication function.

Function	Name	Description
*	**Multiplying**	**The multiplication function will multiply two or more numbers**
Examples		
Using integers	**answer = 4 * 6**	Answers **24**
Using decimals	**answer = 2.3 * 5.1**	Answers **11.73**
Using negatives	**answer = 3 * -7**	Answers **-21**

Dividing Numbers in a Visual Basic

As the first three arithmetic functions were very similar in there handling of integers, decimals and negatives, the division function will cause some problems depending on the direction of the division symbol. Also, we can divide multiple numbers where the first argument is divided by the second and if there is a third, the process will continue until there are no more arguments, but the danger in this function exists when we change the direction of the division symbol. In any Visual Basic, type the expression **3 / 2** and the return is **1.5**, and there is no problem, but type in **3 \ 2** and the answer will be **1**. When we only want the whole number in a division problem to determine the number of loops in the program, the integer division function **** will return an answer with the decimal remainder left off, so the **3 \ 2** is **1** instead of **1.5**. Some programmers make the error thinking the Visual Basic code will round the number up if the remainder is **5** or greater, but not so, the decimal component is just removed.

Now, we will divide the first number by the second number.

In this program, we will divide the two variables that are holding the numbers, **Number1** by **Number2**. The variable **Dividing1Answer** will equal the variable **number1** divided by the value in variable **number2**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim Dividing1Answer As Double
```

Then assign the variable **Dividing1Answer** the results of Number1 divided by Number2.

```
`Assign Variables
Dividing1Answer = Number1 / Number2
```

By writing the division problem to a label caption, we will display the answer to the problem when dividing the numbers. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
`Dividing the numbers
lblDividing1Formula.Caption = Number1 & " / " & Number2 & " = " & Dividing1Answer
```

The following is an extract from the Visual Basic Quick Reference for the Division function.

Function	Name	Description
/	**Division**	**The division function will divide the first number by the second number**
Examples		
Using integers	**answer = 4 / 6**	Answers **0.666**
Using decimals	**answer = 2.3 / 5.1**	Answers **0.450980**
Using negatives	**answer = 3 / -7**	Answers **-0.428571428**

Now we divide numbers when we only want the whole number in a division problem.

Now, we will divide the first number by the second number with the integer division symbol, \.

In this program, we will divide the two variables that are holding the numbers, **Number1** by **Number2**. The variable **Dividing2Answer** will equal the variable **number1** divided by the value in variable **number2**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim Dividing2Answer As Double
```

Then assign the variable **Dividing2Answer** the results of Number1 divided by Number2.

```
`Assign Variables
```

```
Dividing2Answer = Number1 / Number2
```

By writing the division problem to a label caption, we will display the answer to the problem when dividing the numbers. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
`Dividing the numbers
lblDividing2Formula.Caption = Number1 & " \ " & Number2 & " = " & Dividing2Answer
```

The following is an extract from the Visual Basic Quick Reference for the Integer Division function.

Function	Name	Description
\	**Integer Division**	**The division function will divide the first number by the second number**
Examples		
Using integers	answer = 15 \ 6	Answers **2**
Using decimals	answer = 3 \ 2	Answers **1**
Using negatives	answer = 13 \ -7	Answers **-1**

Finding Remainders with Visual Basic

The Modulus Division function will return a number displaying the remainder after the second number is divided into the first number. This is a useful tool when determining whether a counter number even or odd. If you write the following expression to set the counter.

```
counter = 4
```

And the even and odd checker to determine the state of the counter.

```
EvenOrOdd = counter mod  2
```

The variable **EvenOrOdd** will be a **0** for even or a **1** for odd.

In this program, we will divide the two variables that are holding the numbers, **Number1** by **Number2** and place the remainder in the variable **RemainderAnswer**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim RemainderAnswer As Double
```

Then assign the variable **RemainderAnswer** the results of Number1 divided by Number2 using the **mod** function.

```
`Assign Variables
RemainderAnswer = Number1 mod Number2
```

By writing the remainder problem to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Finding the Remainder
lblRemainderFormula.Caption = "The remainder of " & Number1 & " / " & Number2 & " is " & RemainderAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Integer Division function.

Function	Name	Description
mod	**Modulus Division**	**The Modulus Division function will return a number displaying the remainder after the second number is divided into the first number**
Examples		
Using integers	**answer = 15 mod 6**	Answer **3**
Using decimals	**answer = 3 mod 2.1**	Answer **0.9**
Using negatives	**answer = 13 mod -7**	Answer **0**

Computing Absolute Values with Visual Basic

The absolute value function, a function that will only allow a single case to follow the syntax **abs** will return with a positive number. This function is useful in formula writing when deciphering positive distances. In our career, we will discover that graphics program uses a starting point and ending point to explain the construction of a line. When subtracting the starting point x-value and ending point x-value depending on how the line is drawn, our answer can be a negative number. Placing the absolute value function in front of any answer will result in a positive number.

Now, we will find the absolute value of a number.

In this program, we will change the figure in variable **Number1** to its absolute value and place the answer in the variable **AbsoluteAnswer**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim AbsoluteAnswer As Double
```

Then assign the variable **AbsoluteAnswer** the results of Number1 using the **abs** function.

```
`Assign Variables
AbsoluteAnswer = abs(Number1)
```

By writing the absolute value to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Finding the absolute value
lblAbsoluteFormula.Caption = "The absolute value of " & Number1 & " is " & AbsoluteAnswer
```

The variable **AbsoluteAnswer** will equal the positive value of the variable **number1.**

The following is an extract from the Visual Basic Quick Reference for the Absolute Value function.

Function	Name	Description
abs	**Absolute Value**	**The absolute value function will return the positive value of a number**
Examples		
Using integers	**answer = abs (15)**	Answers **15**
Using decimals	**answer = abs (3.1)**	Answers **3.1**
Using negatives	**answer = abs (-7)**	Answers **7**

Fixing Numbers in Visual Basic

The fix function is in our list of number modifiers for arithmetic functions, but we will visit the tool again in the conversion function list. This contrivance will take a real number and remove the decimal places leaving a whole number or integer. There is not any rounding, but the numbers to the right of the decimal place are just removed. This coding method will be useful when computing an array, where we want a whole number response.

In this program, we will fix the variable **Number1**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim FixAnswer As Double
```

Then assign the variable **FixAnswer** the results of **Number1** using the **fix** function.

```
`Assign Variables
FixAnswer = Fix(Number1)
```

By writing the fixing problem to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Fixing the number
lblFixingFormula.Caption = "Fixing " & Number1 & " to " & FixAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Fix function.

Function	Name	Description
fix	**Fix**	**The fix function will return a whole number of a value by removing the number's decimal places**
Examples		
Using integers	**answer = fix (15)**	Answer **15**
Using decimals	**answer = fix (3.1)**	Answer **3**
Using negatives	**answer = fix (-7.5)**	Answer **-7**

Rounding Numbers in a Visual Basic

The round function will take a single number and round the number to the integer value or if we set the placeholder value will round the decimal places to the placeholder value. Numbers to the right of the rounding placeholder will cause that value to increase by one if the number is 5 or above. If the number to the right of the rounding placeholder is 4 or less, the value at the precise decimal place being requested will remain the same.

Type the following code in the program to set the variables for the rounding function.

In this program, we will round the variable **Number1**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim RoundingAnswer As Double
```

Then assign the variable **RoundingAnswer** the results of **Number1** using the **Round** function.

```
`Assign Variables
RoundingAnswer = Round(Number1,2)
```

By writing the rounding problem to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Rounding Answer
lblRoundingFormula.Caption = "Rounding " & Number1 & " 2 places to " & RoundingAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Round function.

Function	Name	Description
Round	**Round**	**The round function will return a number set to the requested decimal place.**
Examples		
Using integers	**answer = round (15,0)**	Answer **15**
Using decimals	**answer = fix (3.125,2)**	Answer **3.13**
Using negatives	**answer = fix (-7.523,1)**	Answer **-7.5**

Computing an Exponent in Visual Basic

The exponent function is used when applying formulas like the area of a circle, which is

$$A = \pi r^2$$

Setting the radius of the circle to 3 type:

r = 3

The value of **π** in AutoLISP is already assigned with syntax, **pi**. Remember we can type !**pi** to check any variable, so try check the value of **pi**. Now to find the area of the radius 3 circle, enter the code shown.

a = pi * r ^ 2

Type the following code in the program to set the variables for the exponent function.

In this program, we will take the variable **Number1** to the power of the value in the variable **Number2**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim ExponentAnswer As Double
```

Then assign the variable **ExponentAnswer** the results of **Number1** to the **Number2** power.

```
`Assign Variables
ExponentAnswer = Number1 ^ Number2
```

By writing the expediential problem to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Finding the Exponent
lblExponentFormula.Caption = Number1 & " to the " & Number2 & " power equals " & ExponentAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Exponent (^) function.

Function	Name	Description
^	**Exponent**	**The exponent function will raise the first number to the power of the second number**
Examples		
Using integers	**answer = 4 ^ 3**	Answers **64**
Using decimals	**answer = 5.5 ^ 0.2**	Answers **1.40628**
Using negatives	**answer = -2.0 ^ 4**	Answers **16.0**

Computing a Square Root in Visual Basic

The square root function will return a decimal number representing the side of a square area. This function only expects a single argument after the function **sqr**. Remember no negative numbers are permissible with the square root function. A complex expression that would be common place would be finding the hypotenuse of a triangle using the variables **a** and **b** that are already defined:

```
c = sqr ((a ^ 2) + (b ^ 2))
```

In this program, we will take the square root of the variable **Number1**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim SquareRootAnswer As Double
```

Then assign the variable **SquareRootAnswer** the results of the square root of **Number1**.

```
`Assign Variables
SquareRootAnswer = Number1 ^ Number2
```

By writing the square root problem to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Finding the square root
lblSquareRootFormula.Caption = "The square root of " & Number1 & " is " & SquareRootAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Square Root function.

Function	Name	Description
sqr	**Square Root**	**The square root function will find the root of the square represented a number**
Examples		
Using integers	**answer = squ (4)**	Answers **2.0**
Using decimals	**answer = squ (5.5)**	Answers **2.34521**
Using negatives	**answer = squ (-155)**	**Negatives Not Allowed**

The sine, cosine and tangent functions use a angular measurement of radians. We are moist familiar with radians when we hear the word pi. Pi radians or 3.14159 is equal to 180 degrees. To convert degrees to radians, divide the angle in degrees by 57.29578.

Computing Sine in a Visual Basic

The sin function is a key tool in find the length of the side opposite the angle and in the case of the standard CAD drawing system using the Cartesian coordinate system, the sine of the angle times the hypotenuse of the triangle or length of the line will represent the delta-y of the line being examined.

```
delta_y = sin (angle / 57.29578)
```

Type the following code in the program to set the variables for the sine function.

In this program, we will take the Sine of the variable **Number1**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim SineAnswer As Double
```

Then assign the variable **SineAnswer** the results of the sine of **Number1**.

```
`Assign Variables
SineAnswer = Sin(Number1)
```

By writing the Sine problem to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Finding the Sine
lblSineFormula.Caption = "The sine of " & Number1 & " is " & SineAnswer
```

The following is an extract from the Visual Basic Quick Reference for the Sine function.

Function	Name	Description
sin	**Sine**	**The sine function will return the length of the side opposite the angle of a right sided triangle when the hypotenuse is 1**
Examples		
Using integers	**answer = sin (2)**	Answer **0.909297**
Using decimals	**answer = sin (0.5)**	Answer **0.479426**
Using negatives	**answer = sin (-0.2)**	Answer **-0.198669**

Computing Cosine in a Visual Basic

The **cos** function is likewise a tool in find the length of the side adjacent to the angle and in the case of the standard CAD drawing system using the Cartesian coordinate system, the cosine of the angle times the hypotenuse of the triangle or length of the line will represent the delta-x of the line being examined.

```
delta_x = cos (angle / 57.29578)
```

In this program, we will take the Cosine of the variable **Number1**.

Type the following code in the Define Variables Section.

```
`Define Variables
Dim CosineAnswer As Double
```

Then assign the variable **CosineAnswer** the results of the cosine of **Number1**.

```
`Assign Variables
CosineAnswer = cos(Number1)
```

By writing the Cosine problem to a label caption, we will display the answer to the problem. We will use the text concatenation function (&) to connect the text string together so the label can be easily inserted into the caption.

```
'Cosine
lblCosineFormula.Caption = "The cosine of " & Number1 & " is " & CosineAnswer
```

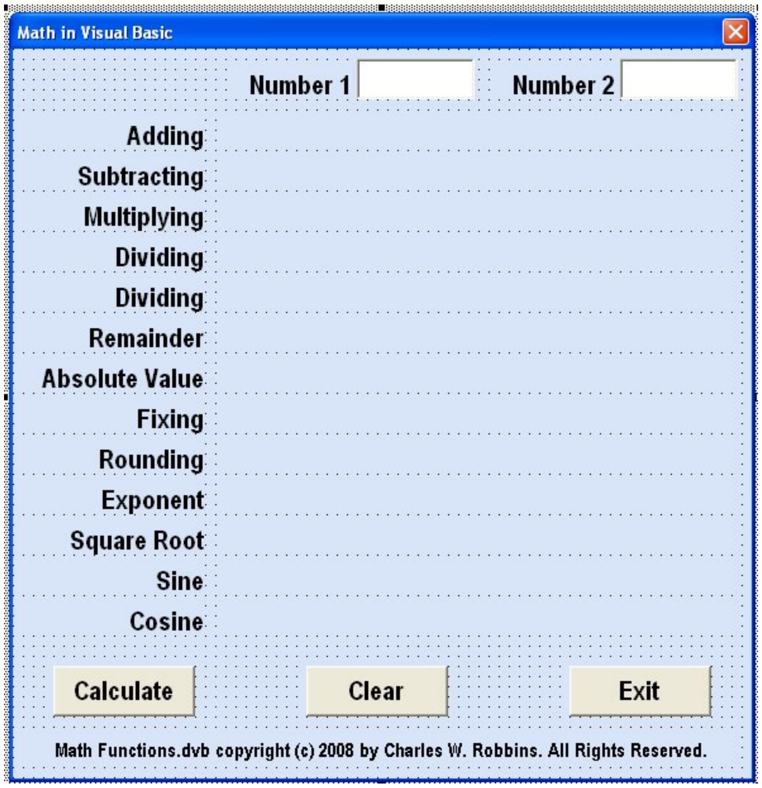
The following is an extract from the Visual Basic Quick Reference for the Cosine function.

Function	Name	Description
cos	**Cosine**	**The cosine function will return the length of the side adjacent to the angle of a right sided triangle when the hypotenuse is 1**
Examples		
Using integers	**answer = cos (2)**	Answer **-0.416147**
Using decimals	**answer = cos (0.5)**	Answer **0.877583**
Using negatives	**answer = cos (-0.2)**	Answer **0.980067**

We have now added a label that defines the category and another that holds the answer to the math problem in the caption. The last step in the program is to add the clear and exit code.

For each label, we have coded a simple math function using one or both of the numbers at the top of the form to get an answer described to the left of the form.

The following is the entire code to practice the Visual Basic math functions.



Math Functions.dvb copyright (c) 2008 by Charles W. Robbins. All Rights Reserved.

Figure 7.18 – The Final Form

```
Private Sub cmdCalculate_Click()
'Math Functions.dvb copyright (c) 2008 by Charles W. Robbins
'This program will open a dialogue box in AutoCAD, allow the user to enter a two numbers
'The program will add, subtract, multiply, divide and use othe math functions to train the
'programmer how use visual basic math functions
'Define the variables

Dim Number1 As Double
Dim Number2 As Double
Dim AddingAnswer As Double
Dim SubtractingAnswer As Double
Dim MultiplyingAnswer As Double
Dim Dividing1Answer As Double
Dim Dividing2Answer As Double
Dim RemainderAnswer As Double
Dim AbsoluteAnswer As Double
Dim FixAnswer As Double
Dim RoundingAnswer As Double
Dim ExponentAnswer As Double
Dim SquareRootAnswer As Double
Dim SineAnswer As Double
Dim CosineAnswer As Double

'Assign variables

Number1 = txtNumber1
Number2 = txtNumber2
AddingAnswer = Number1 + Number2
SubtractingAnswer = Number1 - Number2
MultiplyingAnswer = Number1 * Number2
Dividing1Answer = Number1 / Number2
Dividing2Answer = Number1 \ Number2
RemainderAnswer = Number1 Mod Number2
AbsoluteAnswer = Abs(Number1)
FixAnswer = Fix(mumber1)
RoundingAnswer = Round(Number1, 2)
ExponentAnswer = Number1 ^ Number2
SquareRootAnswer = Sqr(Number1)
SineAnswer = Sin(Number1)
CosineAnswer = Cos(Number1)

'Adding the numbers
lblAddingFormula.Caption = Number1 & " + " & Number2 & " = " & AddingAnswer

'Subtracting the numbers
lblSubtractingFormula.Caption = Number1 & " - " & Number2 & " = " & SubtractingAnswer

'Multiplying the numbers
lblMultiplyingFormula.Caption = Number1 & " * " & Number2 & " = " & MultiplyingAnswer

'Dividing the numbers
lblDividing1Formula.Caption = Number1 & " / " & Number2 & " = " & Dividing1Answer
lblDividing2Formula.Caption = Number1 & " \ " & Number2 & " = " & Dividing2Answer
'Finding the remainder
lblRemainderFormula.Caption = "The remainder of " & Number1 & " / " & Number2 & " is " &
RemainderAnswer
```

```
'Finding the absolute value
lblAbsoluteFormula.Caption = "The absolute value of " & Number1 & " is " & AbsoluteAnswer

'Fixing the number
lblFixingFormula.Caption = "Fixing " & Number1 & " to " & FixAnswer

'Rounding the number
lblRoundingFormula.Caption = "Rounding " & Number1 & " 2 places to " & RoundingAnswer

'Exponents
lblExponentFormula.Caption = Number1 & " to the " & Number2 & " power equals " &
ExponentAnswer

'Square root
lblSquareRootFormula.Caption = "The square root of " & Number1 & " is " & SquareRootAnswer

'Sine
lblSineFormula.Caption = "The sine of " & Number1 & " is " & SineAnswer

'Cosine
lblCosineFormula.Caption = "The cosine of " & Number1 & " is " & CosineAnswer

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 Number1 and Number2 property to a “” entry by using the equal sign “=”. We do the same to the captions of the labels holding the answers.

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

```
Private Sub cmdClear_Click()
'clear the form
   Number1 = ""
   Number2 = ""
   lblAddingFormula.Caption = ""
   lblSubtractingFormula.Caption = ""
   lblMultiplyingFormula.Caption = ""
   lblDividing1Formula.Caption = ""
   lblDividing2Formula.Caption = ""
   lblRemainderFormula.Caption = ""
   lblAbsoluteFormula.Caption = ""
   lblFixingFormula.Caption = ""
   lblRoundingFormula.Caption = ""
   lblExponentFormula.Caption = ""
```

```
    lblSquareRootFormula.Caption = ""
    lblSineFormula.Caption = ""
    lblCosineFormula.Caption = ""
End Sub
```

Exiting the Program with the cmdExit Command Button

To exit this program, we will unload the application and end the program.

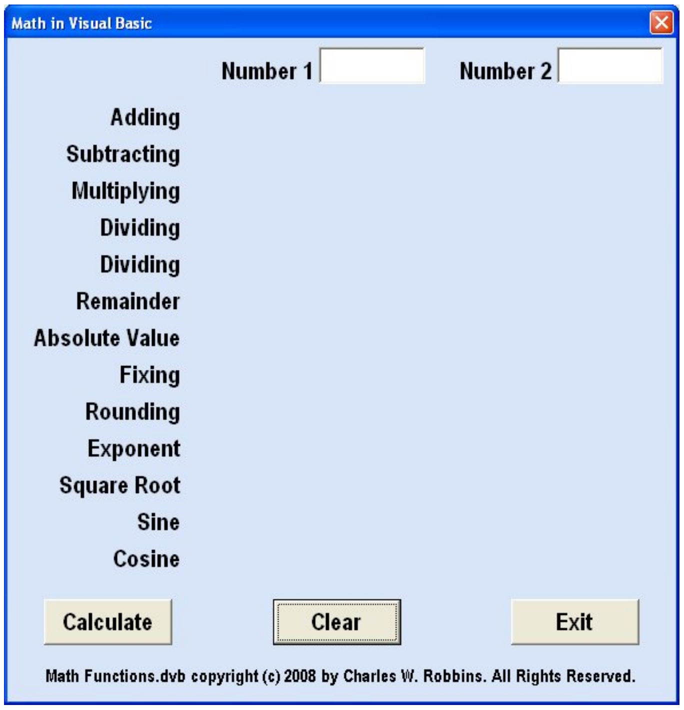
Type the following code:

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

Running the Program

After noting that the program is saved, press the F5 to run the Foundation application. The Foundation window will appear on the graphical display in AutoCAD as shown in Figure 7.19. Type the data shown below or something similar into the textboxes and select the Calculate Command Button to execute the program.

Number1	12
Number2	3

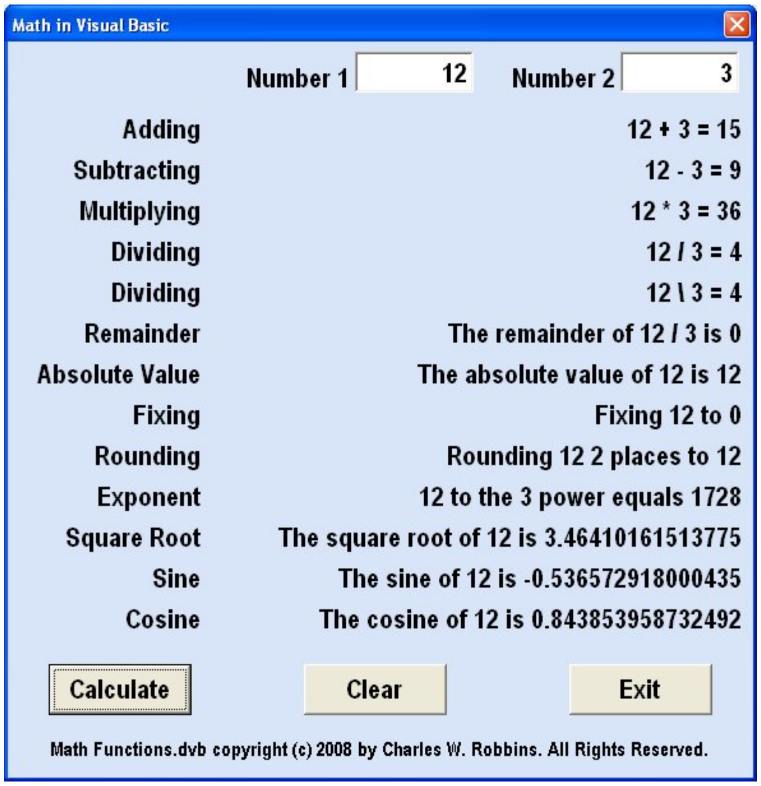


Math Functions.dvb copyright (c) 2008 by Charles W. Robbins. All Rights Reserved.

Figure 7.19 – Launching the Program

There are many variations of this Visual Basic Application we can practice. While we are practicing with forms, we can learn how to use variables, create programs that can compute complex equations, so we can design better products.

After viewing the answers, press the Exit command button on the Math Function Program window.



Number 1	12
Number 2	3
Adding	12 + 3 = 15
Subtracting	12 - 3 = 9
Multiplying	12 * 3 = 36
Dividing	12 / 3 = 4
Dividing	12 \ 3 = 4
Remainder	The remainder of 12 / 3 is 0
Absolute Value	The absolute value of 12 is 12
Fixing	Fixing 12 to 0
Rounding	Rounding 12 2 places to 12
Exponent	12 to the 3 power equals 1728
Square Root	The square root of 12 is 3.46410161513775
Sine	The sine of 12 is -0.536572918000435
Cosine	The cosine of 12 is 0.843853958732492

Math Functions.dvb copyright (c) 2008 by Charles W. Robbins. All Rights Reserved.

Figure 7.30 – The Finished Draw

*** World Class CAD Challenge 5-7 * - Write a Visual Basic Application that computes basic math functions by letting us input data in a form and the program gives us the answer in a label caption. Complete the program in less than 120 minutes to maintain your World Class ranking.**

Continue this drill four times making other math programs, each time completing the Visual Basic Application in less than 120 minutes to maintain your World Class ranking.