Chapter

3

Basic Math Functions

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

- **Opening Visual Basic Editor**
- **Beginning a New Visual Basic Project**
- **Laying Out a User Input Form in Visual Basic**
- **Insert a Label into a Form**
- **Insert a Textbox into a Form**
- **Insert a Label into a Form to Post an Output**
- **Adding More Labels, Textboxes and Answer Labels in the Form**
- **Insert Command Buttons into a Form**
- **Adding a Copyright Statement to a Form**
- **Insert a Picture into a Form**
- **Adding Comments in Visual Basic to Communicate the Copyright**
- **Declaring Variables in a Program with the Dimension Statement**
- **Setting Variables in a Program**
- **Using a Label to Communicate with Variables**
- **Ending the Program**
- **Running the Program**

Open the Visual Basic Editor

_____

In our second lesson, we will step through each procedure in adding labels, textboxes and command buttons and we will integrate into the tutorial the methods to add, subtract, multiply and divide numbers. We will also include formatting the answers as they are shown in the answer labels. As in every project, we will create variable, set their values, execute mathematical equations and output data. In this lesson, we revisit the procedure to add the computer date and time to the form.

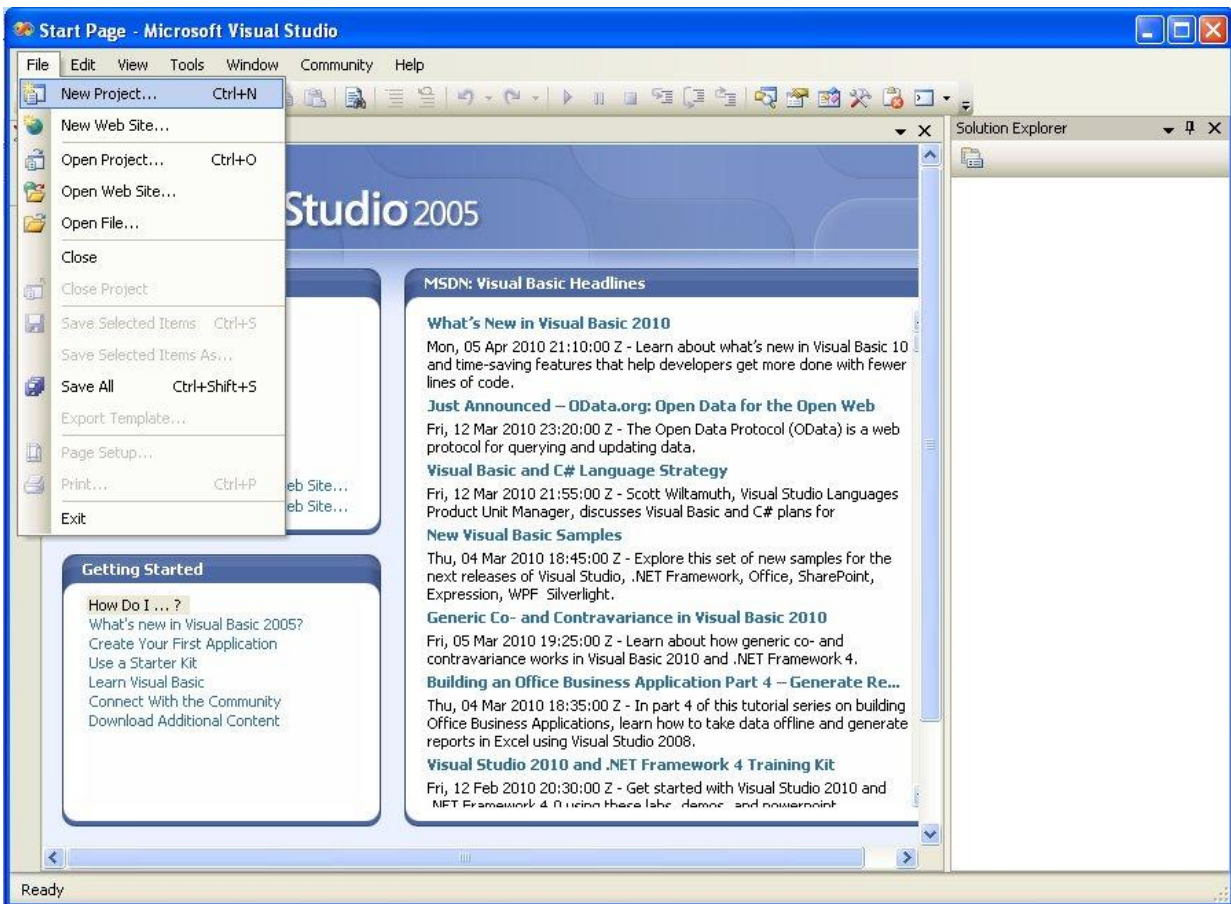To open a new project, we select File on the Menu Bar and New Project.

Figure 3.1 – The Start Page

We start a new Windows Application by picking the Windows Application icon from the installed templates list on the New Project window.
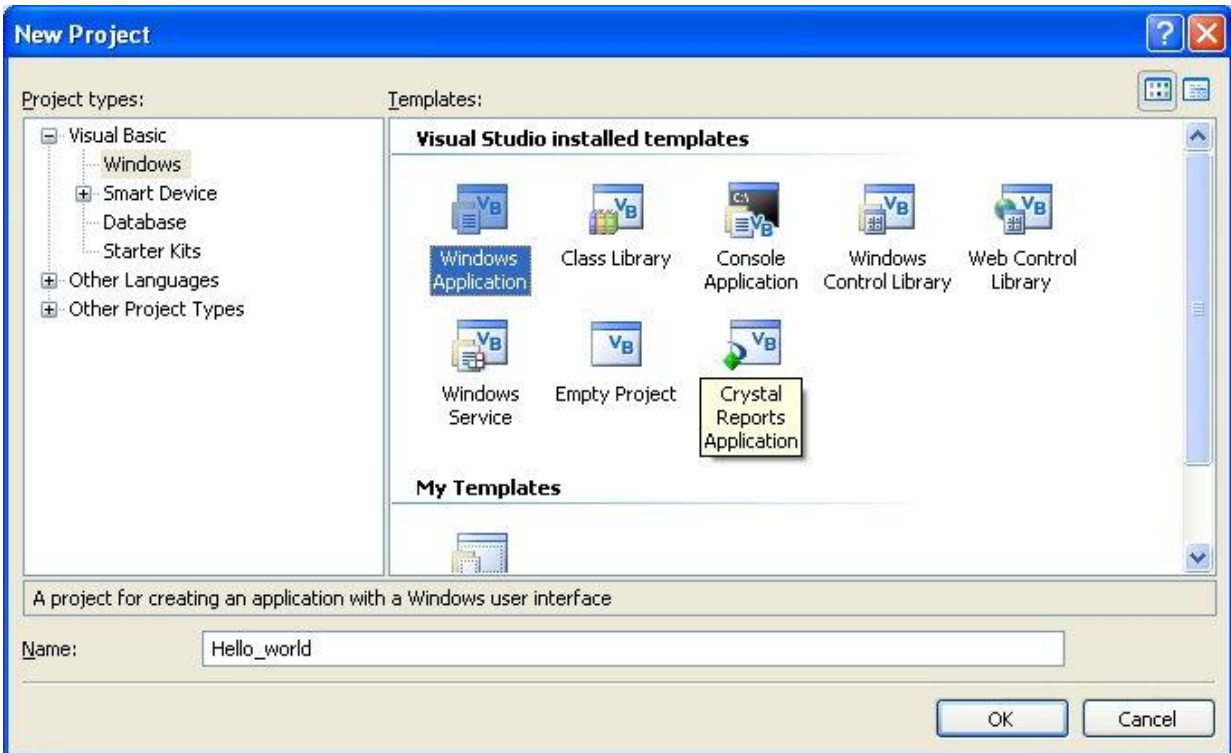
Figure 3.2 – New Project

With the Visual Basic Editor open, select **File** on the Menu Bar and select **Save All**. For the location, we will browse to the folder "Visual Basic Projects" that we made in Chapter 2. We will name this project "Miles per gallon Calculator". A folder called "Miles per gallon calculator" will be made and all the files for the program will be located in the folder.
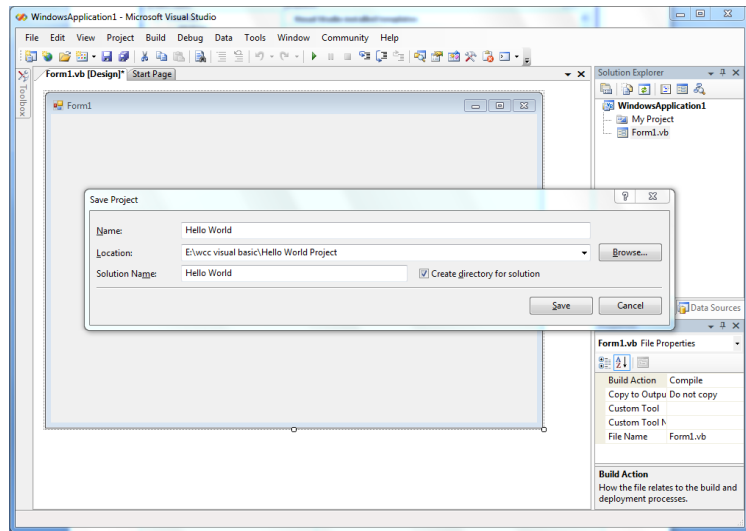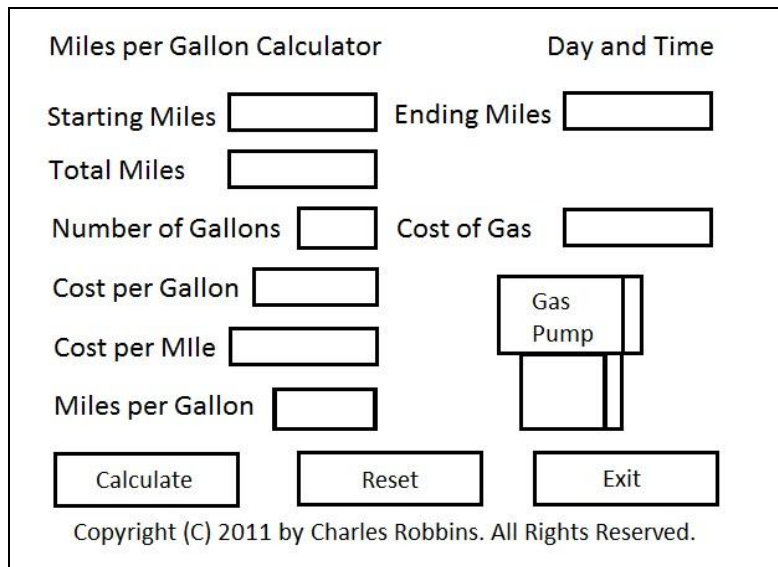

Figure 3.3 – Saving the Miles per Gallon Program

Beginning a New Visual Basic Application
_____

Remember, that all programming projects begin with one or more sketches. The sketch will show labels, textboxes, and command buttons. In this project, we will name the input form, **Miles per Gallon Calculator**. We will place a textbox close to the top of the form to type the starting miles. To the right of the textbox, we will insert the label, "**Starting Miles**". We will also have textboxes and labels for Ending Miles, Number of Gallons, and Cost of Gas. The Total Miles, Cost per Gallon and Cost per Mile will be labels. We will have three command buttons, **Calculate**, **Reset** and **Exit**. On the bottom of the form, we will write the copyright statement using another label. On this presentation, we can help ourselves by being as accurate as possible, by displaying sizes, fonts, colors and any other specific details which will enable us to quickly create the form. On this form, we will use a 15.75 point Arial font. We will have a graphic of a gasoline pump above the Exit button. From the beginning of inserting the form into the project, we need to refer to our sketch.

We should train new programmers initially in the art of form building. When using the editor, we insert and size the form, and selecting the Controls Toolbox, we will place all the various input tools and properly label them. Whenever we place an input tool, the properties window will display a list of every attribute associated with the tool, and we will take every effort to arrange the tool by performing such actions as naming, labeling and sizing the visual input device.
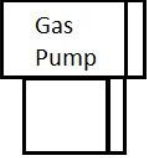
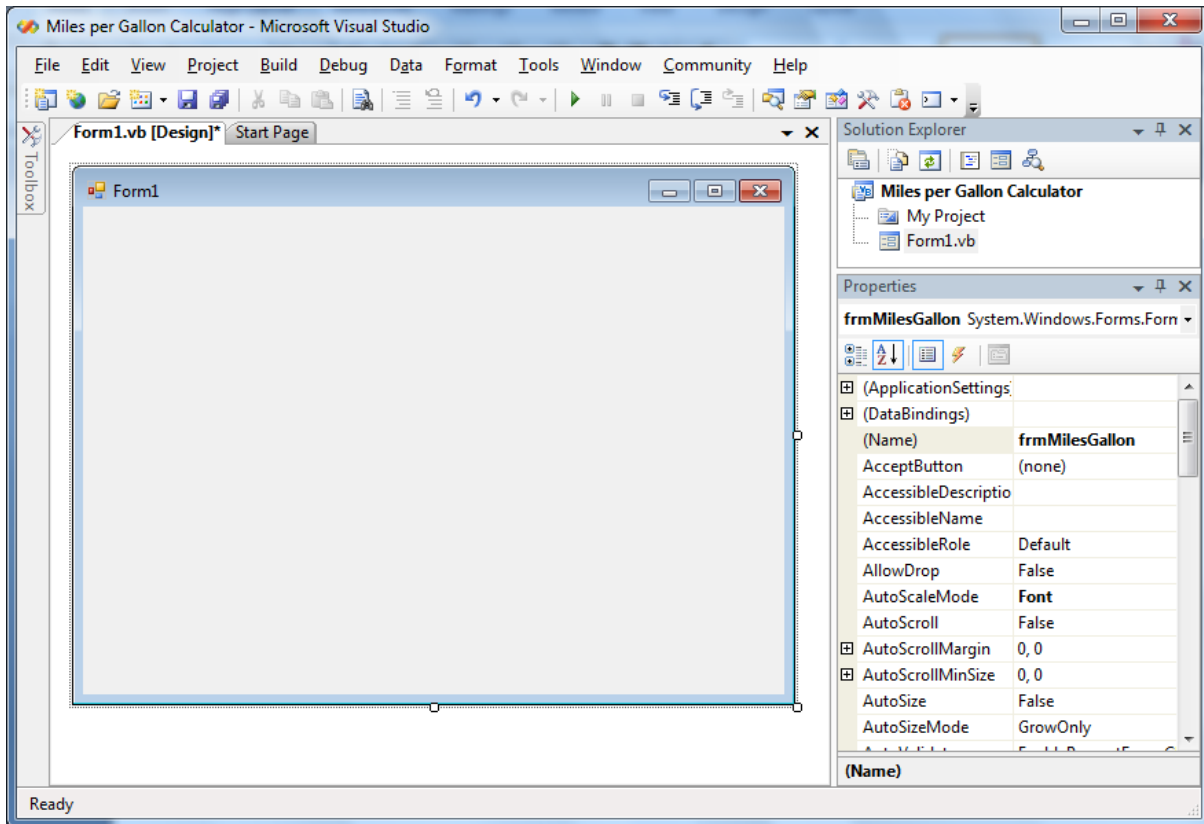**Figure 3.4 – Sketch of the Gas Mileage Form**

Figure 3.5 – Designing the Miles per Gallon Calculator Form in Visual Basic

Laying Out a User Input Form in Visual Basic

We will change the **Text** in the Properties pane to Miles per Gallon Calculator to agree with the sketch in Figure 3.4. Go ahead and change the form in two other aspects, BackColor and Size.

Alphabetic	
BackColor	Light Steel Blue
Font	Arial, 15.75 pt
Size	526, 393

The first number is the width and the second number is the height. The form will change in shape to the size measurement.
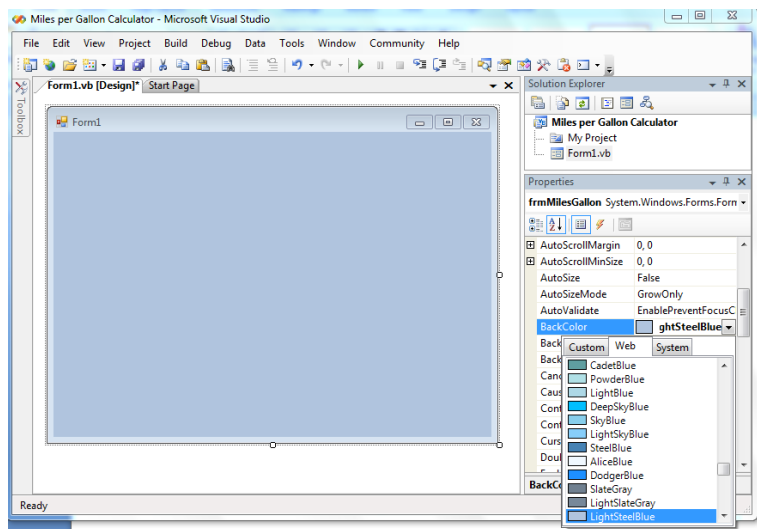

Figure 3.6 – Setting the Caption and other Properties

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 this project, we will select the font in the form. By selecting the font, font style and size for the form, each label, textbox and command button we insert will have these settings for their font.

When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual Basic Font window.
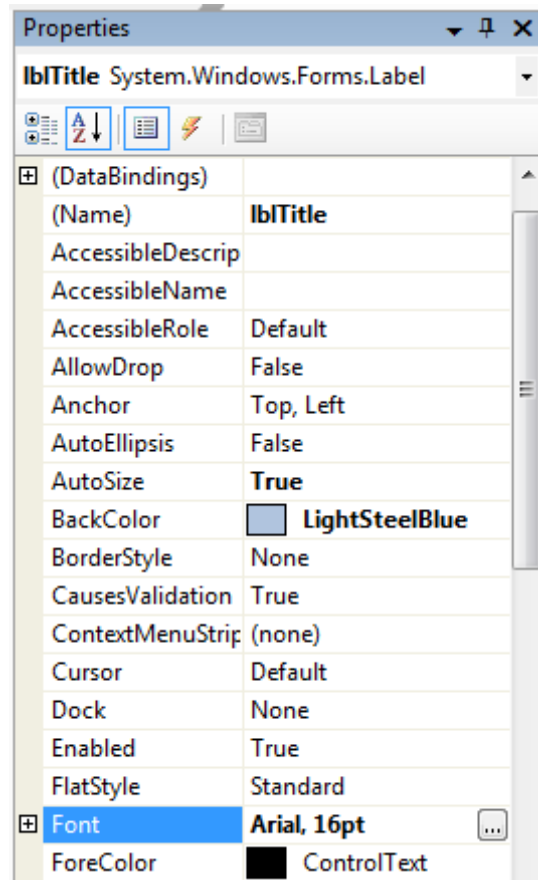

Figure 3.7 – The Font Window in Visual Basic

We will select the Arial font, Regular font style and 15.75 size for this project to agree with the initial sketch if the user input form. If we wish to underline the text or phrase in the label, add a check to the Underline checkbox in the Effects section of the Font window. When we finish making changes to the font property, select the OK command button to return to the work area.
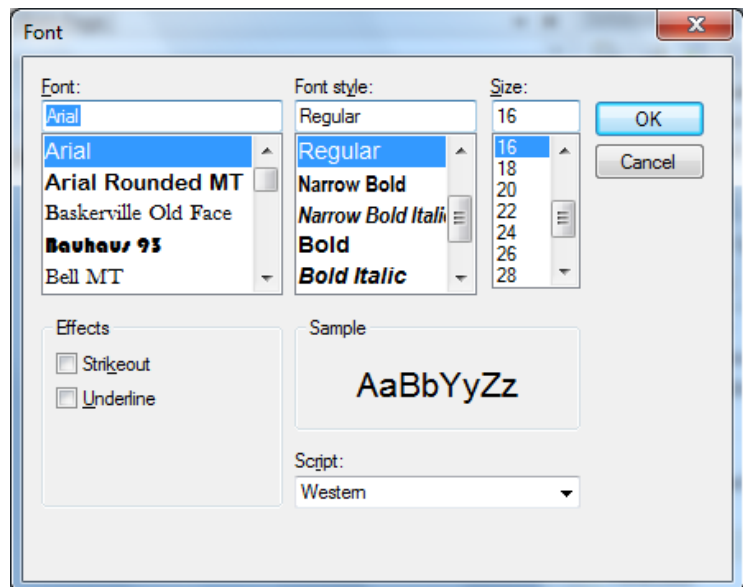

Figure 3.8 – Changing the Font to Arial

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 Windows; 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.
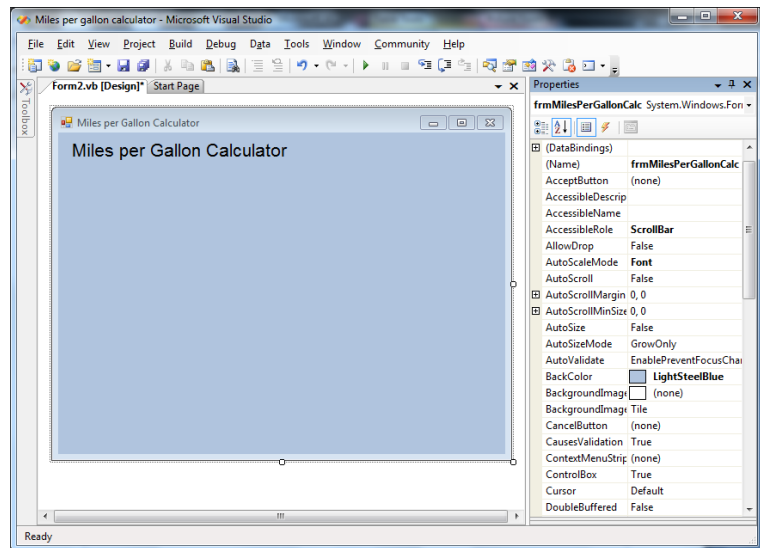

Figure 3.9 – Placing a Label on the Form

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.

We will insert our first Label on the upper left corner of the form and call the entity **lblTitle.**

Alphabetic	
(Name)	lblTitle
BackColor	Light Steel Blue
Text	Miles per Gallon Calculator
Font	Arial, 15.75 pt

Since the backcolor and font are already set, we just type "Miles per Gallon Calculator" at the text attribute.
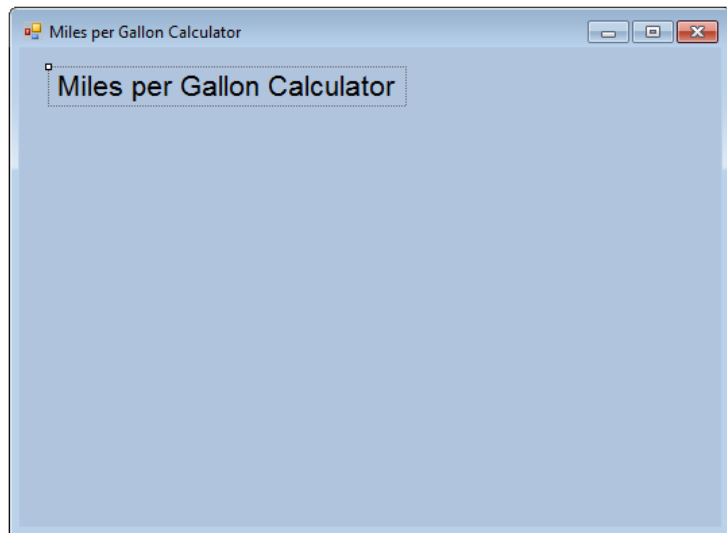

Figure 3.10 – The Finished Label on the Form

3-7

We will insert another Label to the right **lblTitle** and call the entity **lblDateTime.** When the program is running, we will place the computer date and time in the form.

Alphabetic	
(Name)	lblDateTime
BackColor	Light Steel Blue
Text	Date and Time
Font	Arial, 15.75 pt

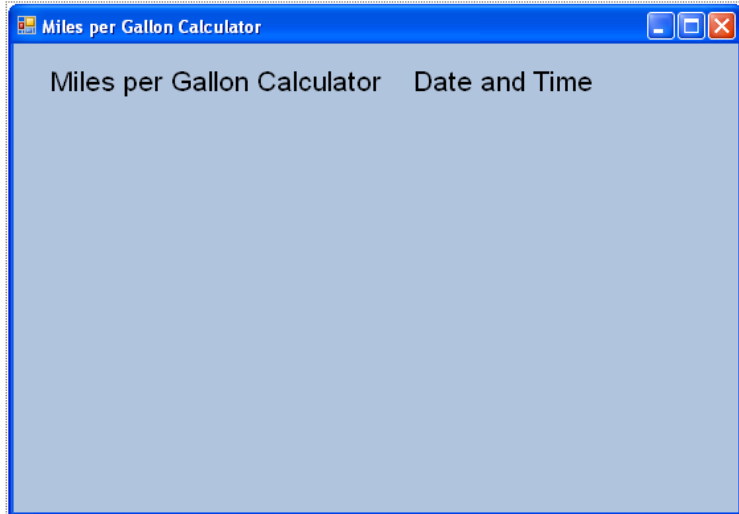The label's text is Date and Time**.** The font on the sketch is 15.75 point, Arial.

Figure 3.11 – The DateTime Label on the Form

We will insert a third Label under the **lblTitle** and call the entity **lblStartingMiles.**

Alphabetic	
(Name)	lblStartingMiles
BackColor	Light Steel Blue
Text	Starting Miles
Font	Arial, 15.75 pt

The label's text is Starting Miles**.** The font on the sketch is 15.75 point, Arial.
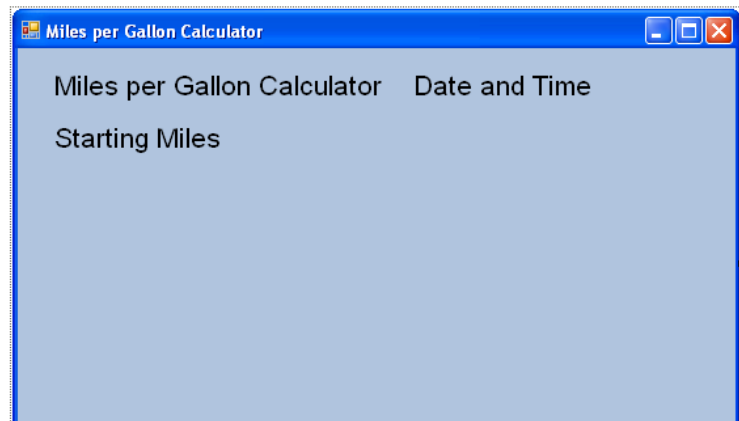

Figure 3.12 – The Starting Miles Label on the Form

Inserting a Textbox into a Form

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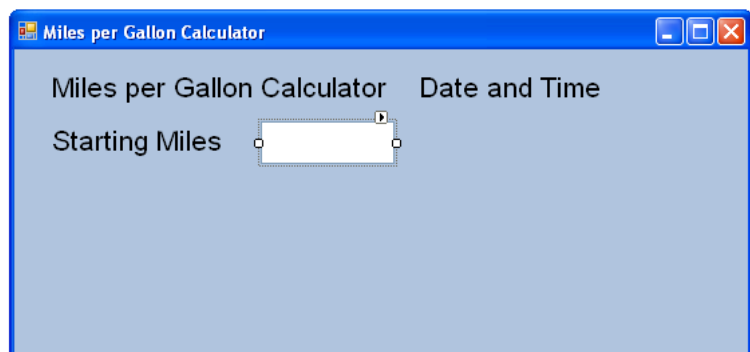

Figure 3.13 – Placing a TextBox on the Form

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 **txtStartingMiles.**

Alphabetic	
(Name)	txtStartingMiles
Size	100, 32
TextAlign	Right

The size of the textbox will be 100 wide and 32 tall and the characters inside the textbox will be aligned to the right.
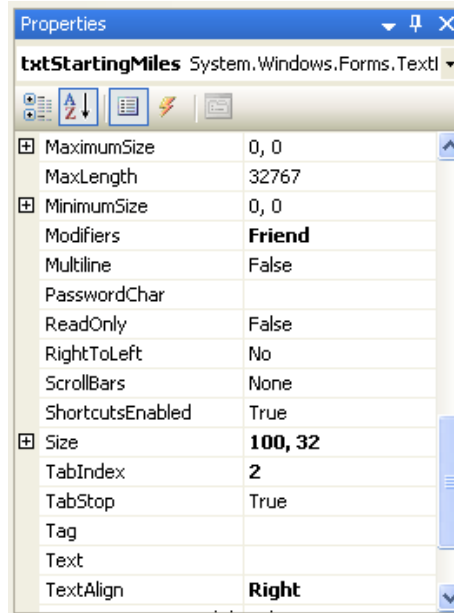

Figure 3.14 – Changing the (Name) to txtName

We will place a label to the right of the **txtStartingMiles** textbox and call it **lblEndingMiles.** We will make the label text Ending Miles. The key attributes for the label are:

Alphabetic	
(Name)	lblEndingMiles
BackColor	Light Steel Blue
Text	Starting Miles
Font	Arial, 15.75 pt

We will insert a second textbox in the form to the right of the **lblEndingMiles** and call the textbox **txtEndingMiles.**

Alphabetic	
(Name)	txtEndingMiles
Size	100, 32
TextAlign	Right

The size of the textbox will be 100 wide and 32 tall and the characters inside the textbox will be aligned to the right.
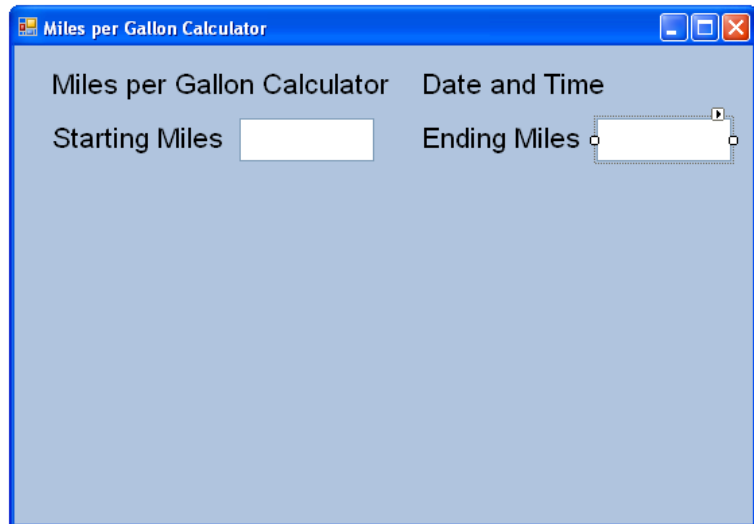

Figure 3.15 – Changing the (Name) to txtName

3-9

Inserting a Label into a Form to Post the Output
_____

Some labels on a form are in a position to display an answer after the user inputs data and they press the command button to execute the application. To add this label, 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.
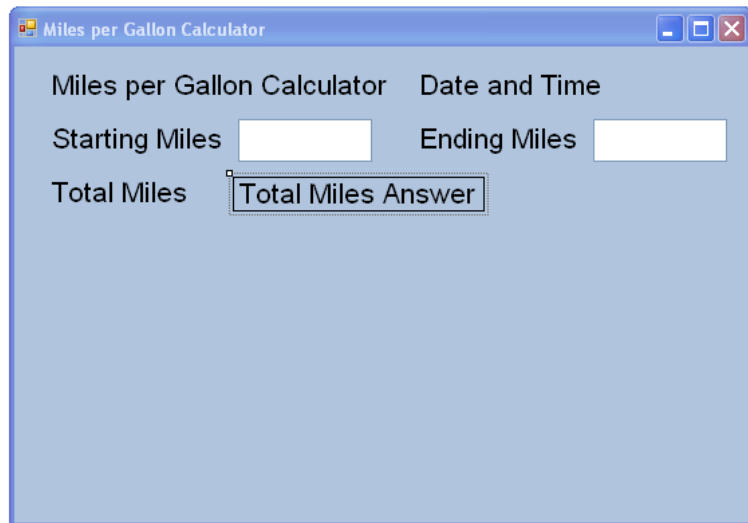

Figure 3.16 – Placing a Label for the Answer

We will place a label under **lblStartingMiles** label and call it **lblTotalMiles.** We will make the label text Total Miles. The key attributes for the label are:

Alphabetic	
(Name)	lblTotalMiles
BackColor	Light Steel Blue
Text	Total Miles
Font	Arial, 15.75 pt

We will insert the label for the answer to the right of **lblTotalMiles** and name the label **lblTotalMilesAnswer.**

Alphabetic	
(Name)	lblTotalMilesAnswer
BorderStyle	FixedSingle

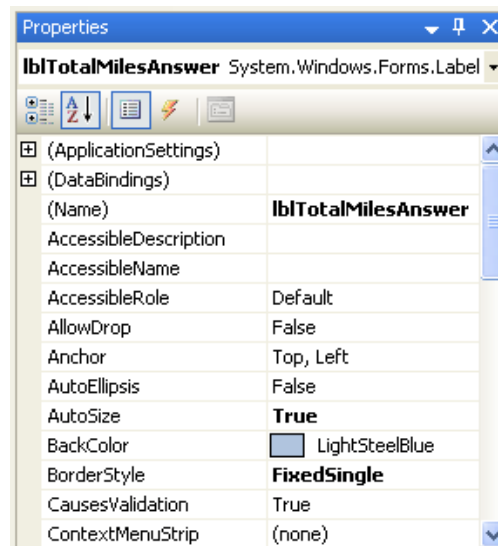We will make the boderstyle FixedSingle to place a line around the answer.

Figure 3.17 – Label Name is lblTotalMilesAnswer

Adding More Labels, Textboxes and Answer Labels in the Form

_____

Now that we have placed a number of labels, textboxes and answer labels on the form, we need to complete similar objects for Gallons, Cost of Gas, Cost per Gallon, Cost per Mile, and Miles per Gallon.

The key attributes for the Gallons label are:

Alphabetic	
(Name)	lblGallons
BackColor	Light Steel Blue
Text	Gallons
Font	Arial, 15.75 pt

Figure 3.18 – Label Name is lblGallons

The key attributes for the Gallons textbox are:

Alphabetic	
(Name)	txtGallons
Size	100, 32
TextAlign	Right

Figure 3.19 – Insert Textbox txtGallons

The key attributes for the Cost of Gas label are:

Alphabetic	
(Name)	lblGasCost
BackColor	Light Steel Blue
Text	Gallons
Font	Arial, 15.75 pt

Figure 3.20 – Label Name is lblGasCost

The key attributes for the Gallons textbox are:

Alphabetic	
(Name)	txtGasCost
Size	100, 32
TextAlign	Right

Figure 3.21 – Insert Textbox txtGasCost

3-11

Now all we have to do is add three simple labels called lblGalCost, lblCostperMile and lblMilesperGal on the left and three answer labels with single lined borders for a border style on the right.

Alphabetic	
(Name)	lblGalCost
BorderStyle	FixedSingle

Alphabetic	
(Name)	lblCostperMile
BorderStyle	FixedSingle

Alphabetic	
(Name)	lblMilesperGal
BorderStyle	FixedSingle
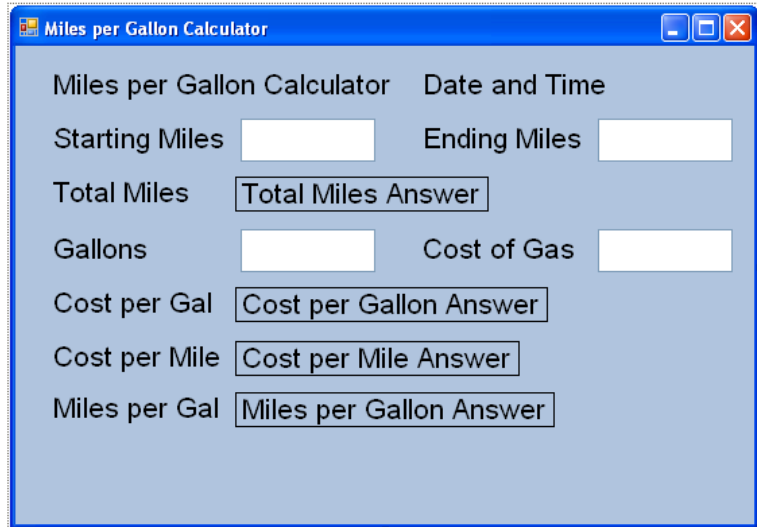
Figure 3.22 – Three Answer Labels

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 3.21.
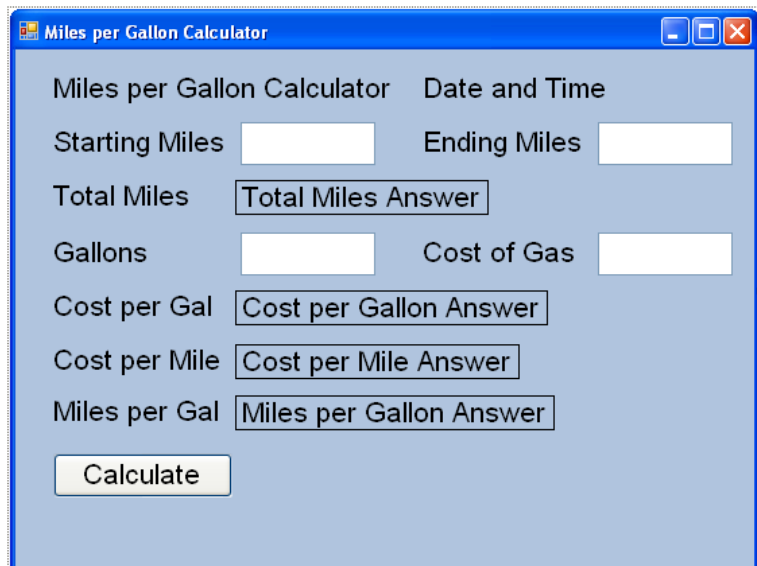
Figure 3.23 – Insert a Command Button onto a Form

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

Alphabetic	
(Name)	cmdCalculate
Caption	Calculate
Font	Arial, 15.75 pt
Size	133,33
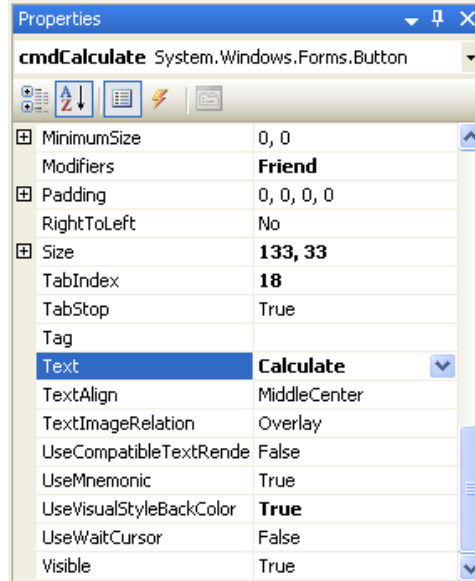
Figure 3.24 – Changing the (Name) to cmdCalculate

Add a second Command button, named cmdReset is for clearing the txtName and lblGreeting objects. The third command button is to exit the program. When the user presses the Exit command button, the application closes. Notice the equal spacing between the command buttons gives a visually friendly appearance.
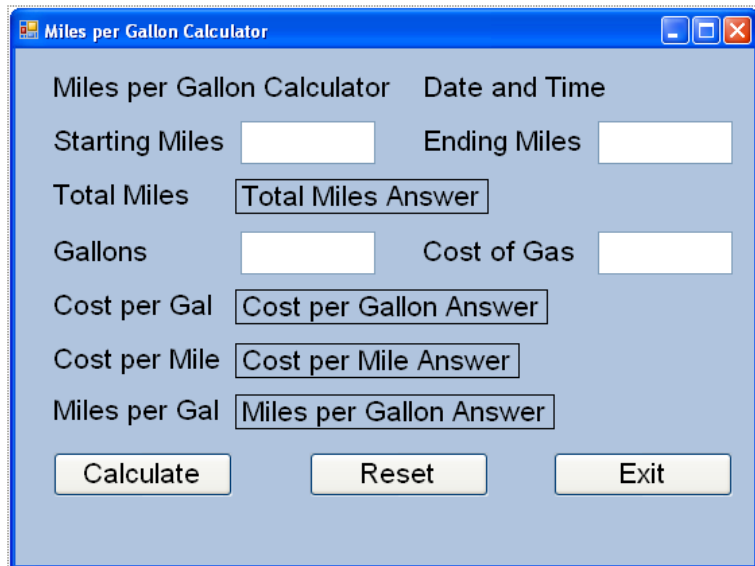
Figure 3.25 – Insert Two More Command Buttons

Adding a Copyright Statement to a Form
_____

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

Finish the form with the following copyright information.

'Miles per Gallon Calculator.dv copyright (c) 2011 by charles robbins

If there are special rules or instructions that the user needs to know, place that information on the bottom of the form.
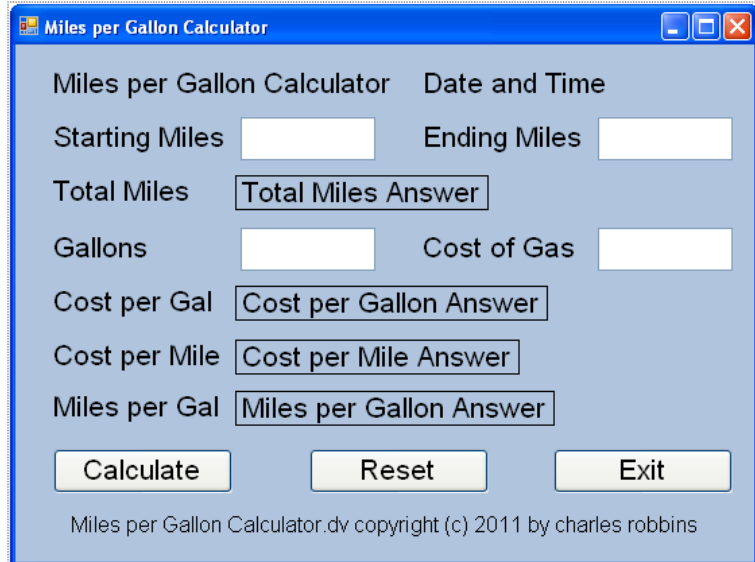
Figure 3.26 – Adding a Copyright Statement

Inserting a Picture into a Form

_____

We select the toolbox and Picturebox and we draw a box to the right of the labels that will contain the answers. We name the picturebox **imgGasPump**. We scroll down on the properties window and select the three dots button at the Image property and a Select Resource window will appear.
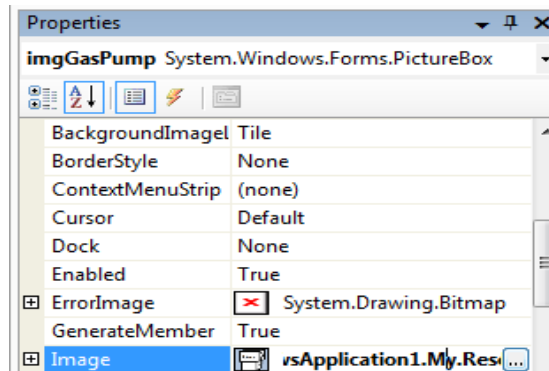
Figure 3.27 – Adding an Image

We then will import the graphic of our gas pump which we made in Microsoft Paint and saved as a bitmap image. We then press the OK button and the image will appear in the picturebox.
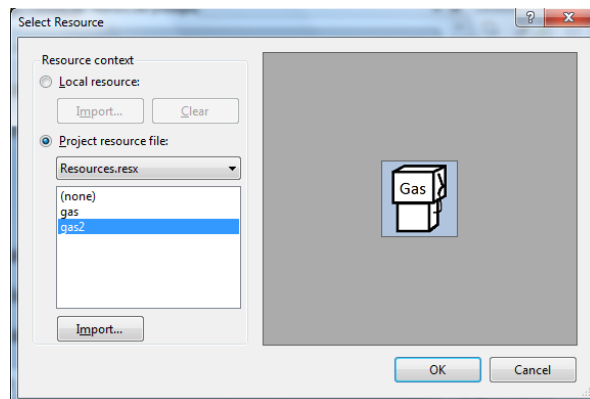
Figure 3.28 – Import an Image

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 Hello command button. At the top of the program and before 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:

'Miles per Gallon Calculator.vb copyright (c) 2011 by Charles W. Robbins
'This program will open a dialogue box, allow the user to type their starting and ending miles
'The user can add the number of gallons and the cost of gas for the fill up
'When the user clicks on the Calculate button, a the total miles, cost per gallon, cost per mile
'and miles per gallon is calculated.
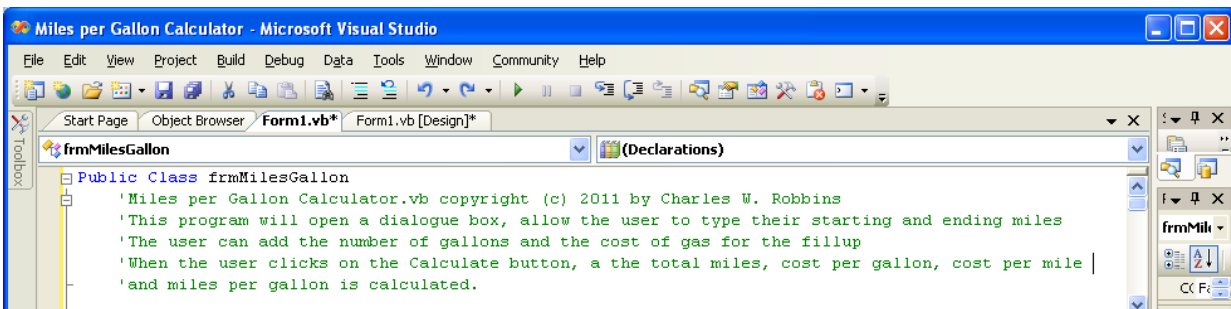
Figure 3.29 – Adding a Copyright Statement

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 procedure level. The other two ways are the Private and Public statements, which we will use in later chapters.

In our program, we will retrieve the data from the textboxes and also we will create data from mathematical computations. We will place the values in variables called StartingMiles, EndingMiles, TotalMiles, Gallons, GasCost, CostPerGal, CostPerMile, and MilesPerGal. These variables will hold numbers for calculations so we will declare them as Double Integers.

Type the following code under the cmdCalculate subroutine of the program.

```
' declare variables
      Dim StartingMiles As Double
      Dim EndingMiles As Double
      Dim TotalMiles As Double
      Dim Gallons As Double
      Dim GasCost As Double
      Dim CostPerGal As Double
      Dim CostPerMile As Double
      Dim MilesPerGal As Double
```
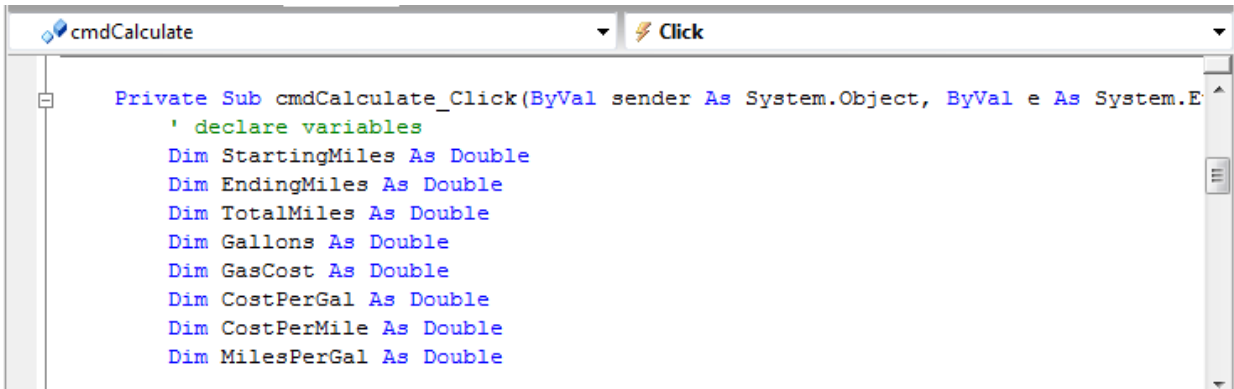

Figure 3.30 – Declaring Variables with Dim Statements
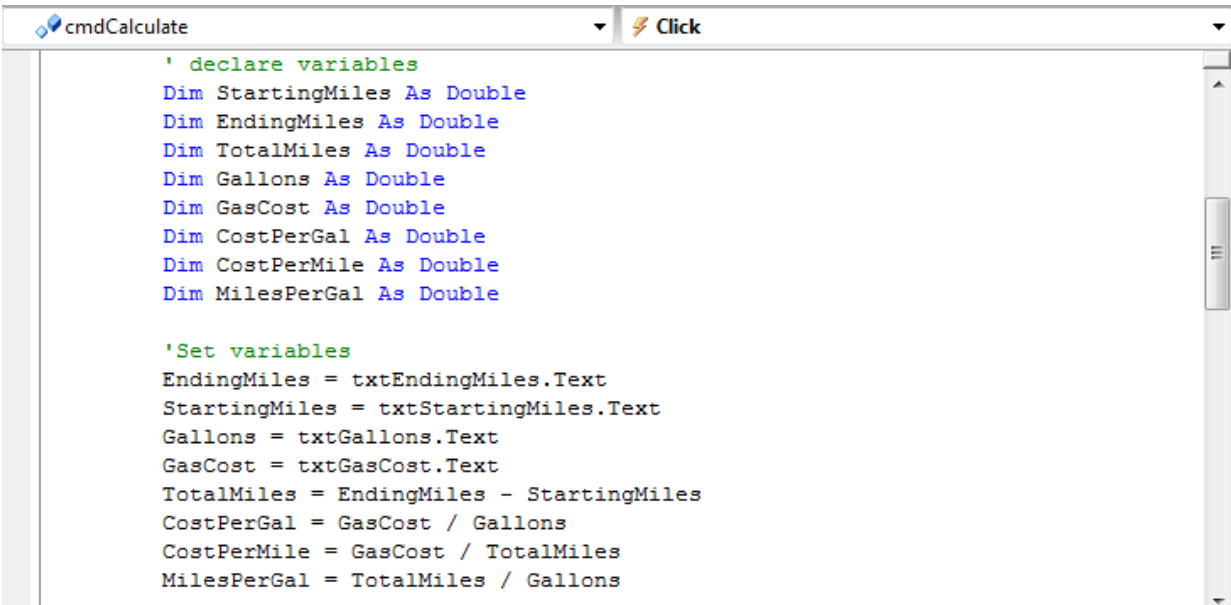
Notice that the variable name should be a word or a phrase without spaces that represents the value that the variable contains. If we want to hold a value of one's date of birth, we can call the variable, DateofBirth. The keywords Date and Birth are in sentence case with the first letter capitalized. 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. We will set the numbers in the four textboxes to their variable and we compute total miles by using subtraction and the cost per gallon, cost per mile and miles per gallon by using division.

Type the following code under the "declare variable" section of the cmdCalculate subroutine of the program.

```
'Set variables
      EndingMiles = txtEndingMiles.Text
      StartingMiles = txtStartingMiles.Text
      Gallons = txtGallons.Text
      GasCost = txtGasCost.Text
      TotalMiles = EndingMiles - StartingMiles
      CostPerGal = GasCost / Gallons
      CostPerMile = GasCost / TotalMiles
      MilesPerGal = TotalMiles / Gallons
```

```
cmdCalculate                        ▼   ⚡ Click                        ▼
        ' declare variables
        Dim StartingMiles As Double
        Dim EndingMiles As Double
        Dim TotalMiles As Double
        Dim Gallons As Double
        Dim GasCost As Double
        Dim CostPerGal As Double
        Dim CostPerMile As Double
        Dim MilesPerGal As Double

        'Set variables
        EndingMiles = txtEndingMiles.Text
        StartingMiles = txtStartingMiles.Text
        Gallons = txtGallons.Text
        GasCost = txtGasCost.Text
        TotalMiles = EndingMiles - StartingMiles
        CostPerGal = GasCost / Gallons
        CostPerMile = GasCost / TotalMiles
        MilesPerGal = TotalMiles / Gallons
```

Figure 3.31 – Setting the Variables in the VBA Code

Using a Label to Communicate with Variables
_____

The numbers that were set to the variables can now be assigned to the answer labels using the equal sign such as **lblTotalMilesAnswer.Text = TotalMiles** and the answer will appear in the form with a single border around the numbers.

Go ahead and type the following code below the set variables section.

```
'display answers
lblTotalMilesAnswer.Text = TotalMiles
lblGalCostAnswer.Text = FormatCurrency(CostPerGal)
lblCostperMileAnswer.Text = Format(CostPerMile, "#.###")
lblMilesPerGalAnswer.Text = Format(MilesPerGal, "#.##")
```

We have added a new feature to the output by using the FormatCurrency() function that puts the answer in dollar and cents. We also are using the Format() function with the variable first then a comma and the format of how many decimals we would like to see in the answer. "#.##" will give us 2 decimals and "#.###"will give us 3 decimals in the answer.

```
cmdCalculate                           ▼    ⚡ Click                              ▼

              'Set variables
              EndingMiles = txtEndingMiles.Text
              StartingMiles = txtStartingMiles.Text
              Gallons = txtGallons.Text
              GasCost = txtGasCost.Text
              TotalMiles = EndingMiles - StartingMiles
              CostPerGal = GasCost / Gallons
              CostPerMile = GasCost / TotalMiles
              MilesPerGal = TotalMiles / Gallons

              'display answers
              lblTotalMilesAnswer.Text = TotalMiles
              lblGalCostAnswer.Text = FormatCurrency(CostPerGal)
              lblCostperMileAnswer.Text = Format(CostPerMile, "#.###")
              lblMilesPerGalAnswer.Text = Format(MilesPerGal, "#.##")
```

Figure 3.32 – Displaying the Answers

Resetting the Data

_____

To clear the textboxes or labels containing the data, we will replace the date with blank strings
and the date and time with the current day and time setting.

Type the following code under the cmdReset subroutine of the program

```
    'Reset date and time, textboxes and labels with answers
   lblDateTime.Text = DateString() & " at " & TimeString()
   txtStartingMiles.Text = ""
   txtEndingMiles.Text = ""
   txtGallons.Text = ""
   txtGasCost.Text = ""
   lblTotalMilesAnswer.Text = "              "
   lblGalCostAnswer.Text = "            "
   lblCostperMileAnswer.Text = "             "
   lblMilesPerGalAnswer.Text = "             "
```

```
(General)                               ▼    (Declarations)                        ▼

       Private Sub cmdReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
           'Reset date and time, textboxes and labels with answers
           lblDateTime.Text = DateString() & " at " & TimeString()
           txtStartingMiles.Text = ""
           txtEndingMiles.Text = ""
           txtGallons.Text = ""
           txtGasCost.Text = ""
           lblTotalMilesAnswer.Text = "              "
           lblGalCostAnswer.Text = "             "
           lblCostperMileAnswer.Text = "             "
           lblMilesPerGalAnswer.Text = "             "
       End Sub
```
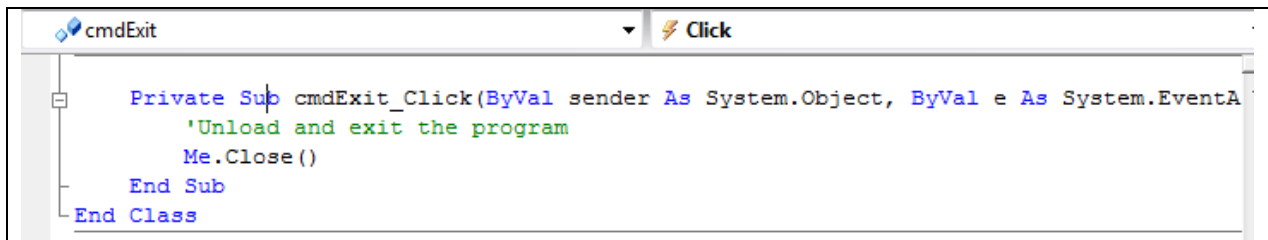
Figure 3.33 – Computing the Reset Button by Clearing a Textbox and Label Caption

Exiting the Program

_____

Figure 3.34 – Exiting the Program

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

"Unload and exit the program
   Me.Close()

Running the Program

_____

After noting that the program is saved, press the F5 to run the Miles per Gallon Calculator application. The Miles per Gallon Calculator window will appear on the graphical display as shown in Figure 3.34. Notice the professional appearance and presentation of information in a clean dialogue box. The color of the background is neither black nor white, which would match the normal graphical display colors used by designers.
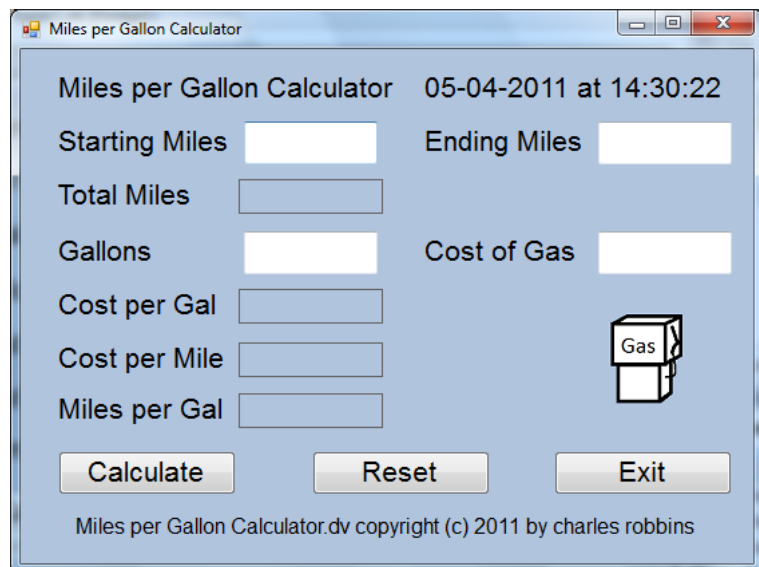

Figure 3.35 – Launching the Program

Type the starting and ending miles in the textbox just as we typed. Input the gallons and the cost of the gasoline as shown in Figure 3.35. If we make a mistake, we can type over the text entry or press the Reset command button to clear the textbox. Press the Calculate command button and the four answer labels will have the total miles, cost per gallon, cost per mile and miles per gallon for this fill up. After experimenting with our program, press the Exit command button to exit the application.
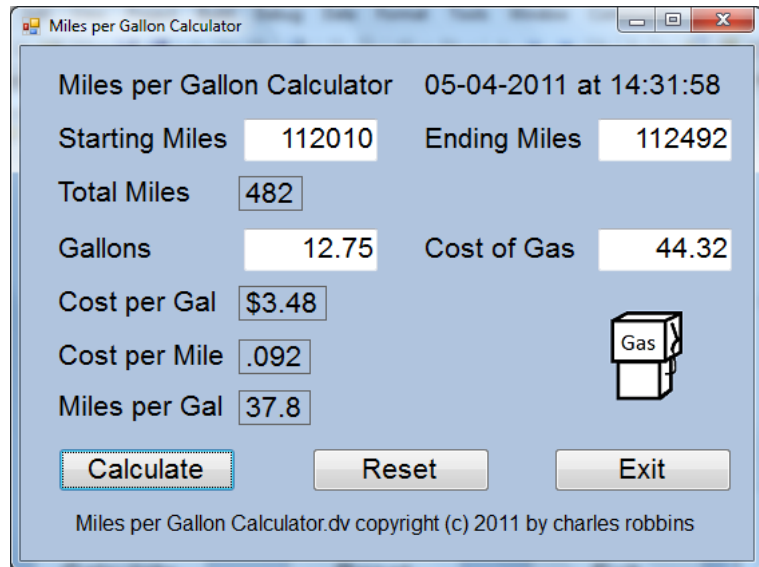

Figure 3.36 – Running the Program

If our program does not function correctly, go back to the code and check the syntax against the program shown in previous sections. Repeat any processes to check or Beta test the program. When the program is working perfectly, save and close the project.

There are many variations of this Visual Basic Application we can practice and obtain information from a personal computer. While we are practicing with forms, we can learn how to use variables, strings and comments. These are skills that we want to commit to memory.

* World Class CAD Challenge 90-2 * - Write a Visual Basic Application that displays a single input form, allows the user to type in their data, and when executed, the program will give the user information obtained from the computer and from mathematical computations.

Continue this drill four times using some other form designs, each time completing the Visual Basic Project in less than 1 hour to maintain your World Class ranking.