

Chapter 7

Visual Basic Program: Subnetting Calculator

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**
- **Insert Command Buttons into a Form**
- **Adding a Copyright Statement to a Form**
- **Adding Comments in Visual Basic to Communicate the Copyright**
- **Declaring Variables in a Program with the Dimension Statement**
- **Setting Variables in a Program**
- **Using Condition Statements**
- **Writing Information to a Text File**
- **Resetting the Data**
- **Exiting the Program**
- **Adding a Browse and Open File Button**
- **Running the Program**

Subnetting a TCP/IP Network

Many times when we write a new program, we will want to output the data to a computer file such as a text document, spreadsheet or database. The Subnetting Calculator application in this chapter is a perfect example of a computer program that can create as few subnets as two or as many as 256, so we choose to send the IP addresses to a text file.

In this lesson, we need to utilize textboxes, labels and command buttons as we have in previous training sessions, but we will create an additional two command buttons to browse for the folder to place the newly created text file and another push button to open the computer file after we calculate the values of each subnet.

In our tutorial, we again use condition statements to make decisions, we utilize while loops to repeat repetitive expressions and we continue to concatenate strings to construct information that is useful to the computer user. Although this program is larger than some done before, many components of the code are rhythmic and we can quickly spot the changes to each segment of the software.

Open the Visual Basic Editor

In this session, 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.

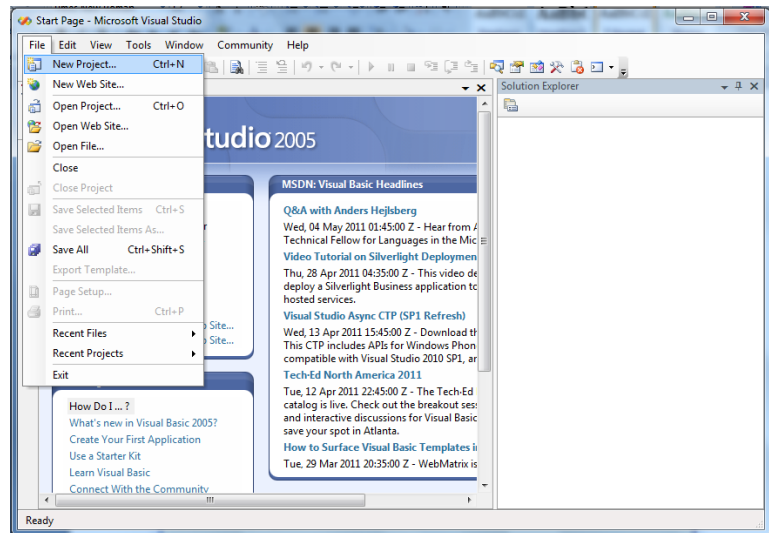


Figure 7.1 – The Start Page

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

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

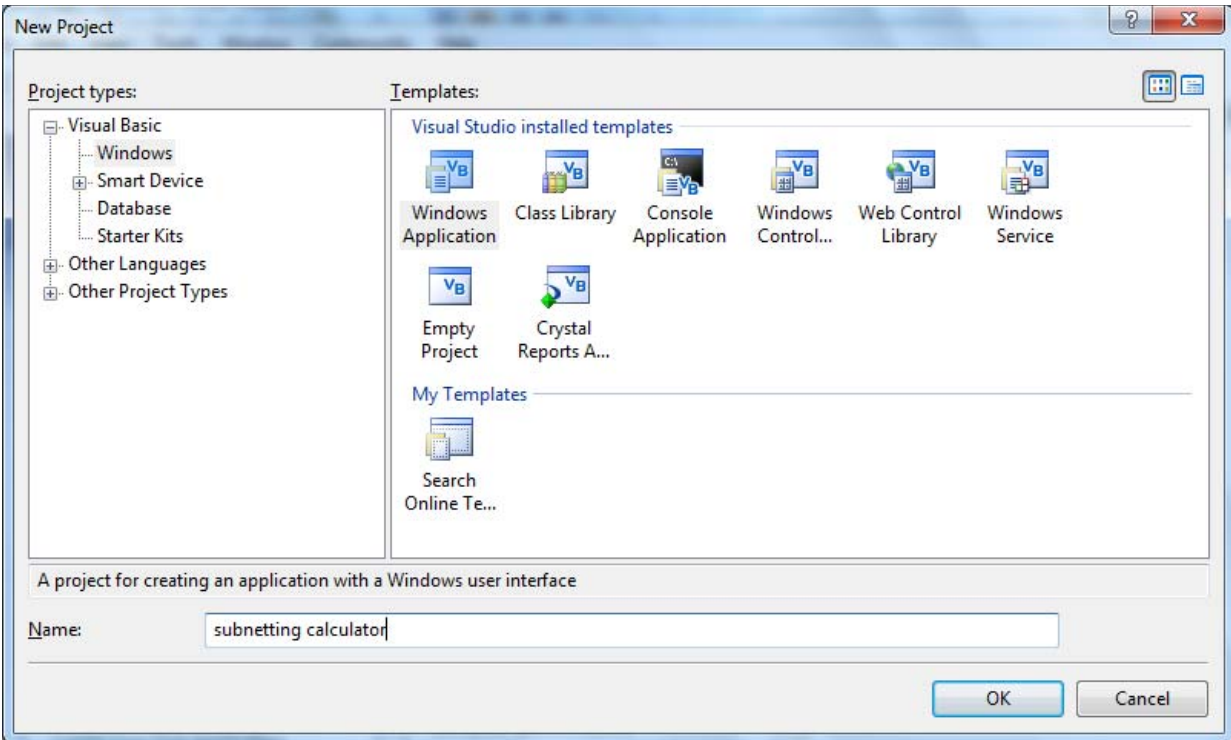


Figure 7.2 – New Project

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 “Subnetting Calculator”. A folder called “Subnetting Calculator” will be made and all the files for the program will be located in the folder.

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, **Subnetting Calculator**. We will have six textboxes to key in the network IP address, the number of subnet and the filename that will hold the calculated data. We will have two labels with borders to output the subnet mask and the number of devices on each subnet. There will be labels to identify the textboxes and labels withy borders. We will have five command buttons, **Browse, Calculate, Open File, 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 12 point Arial font. 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 7.3 – Sketch of the Resistor Sizing Form

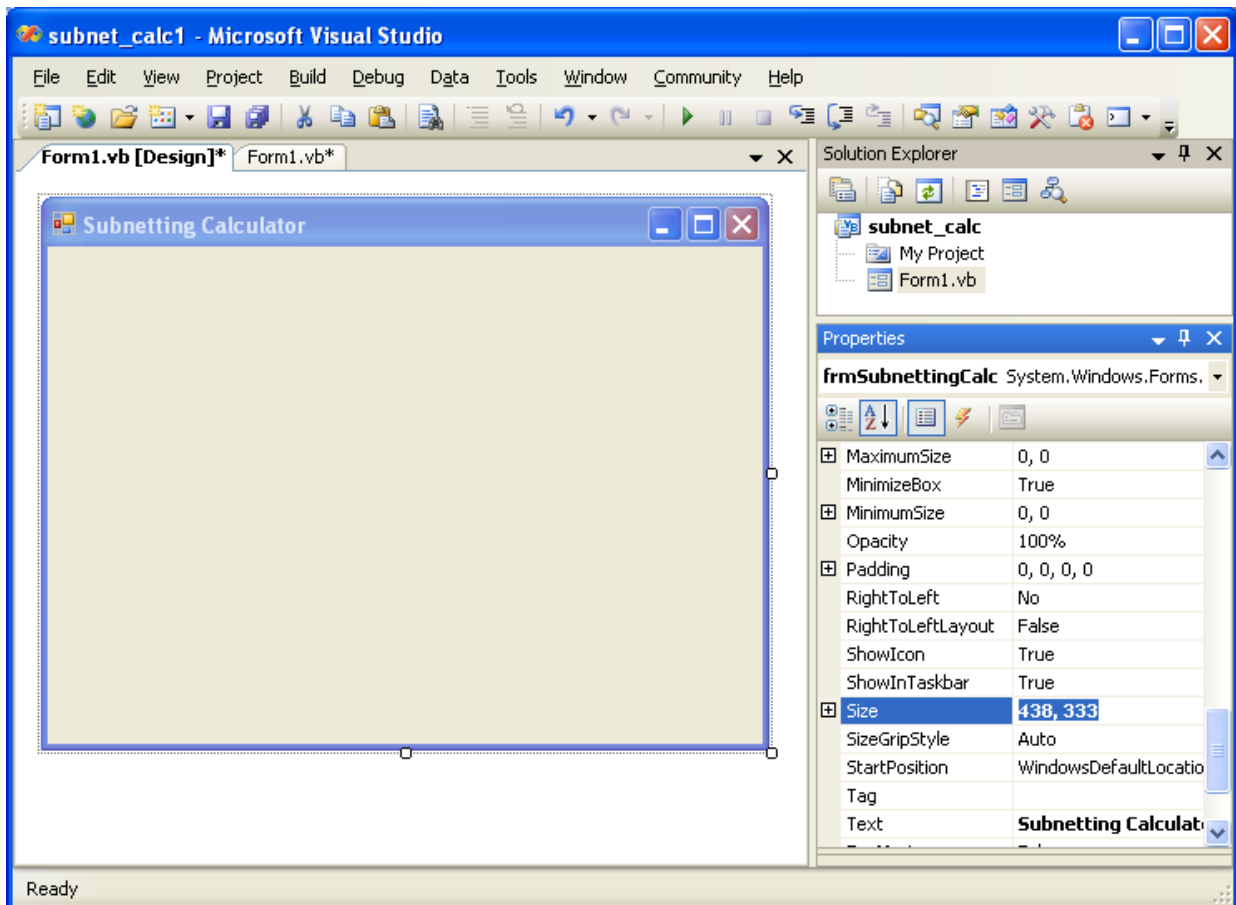


Figure 7.4 – Designing the Subnetting Calculator Form in Visual Basic

Laying Out a User Input Form in Visual Basic

We will change the **Text** in the Properties pane to Subnetting Calculator to agree with the sketch in Figure 7.3. Go ahead and change the form in two other aspects, BackColor and Size.

Alphabetic	
(Name)	frmSubnettingCalc
Size	446, 337
Text	Subnetting Calculator

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

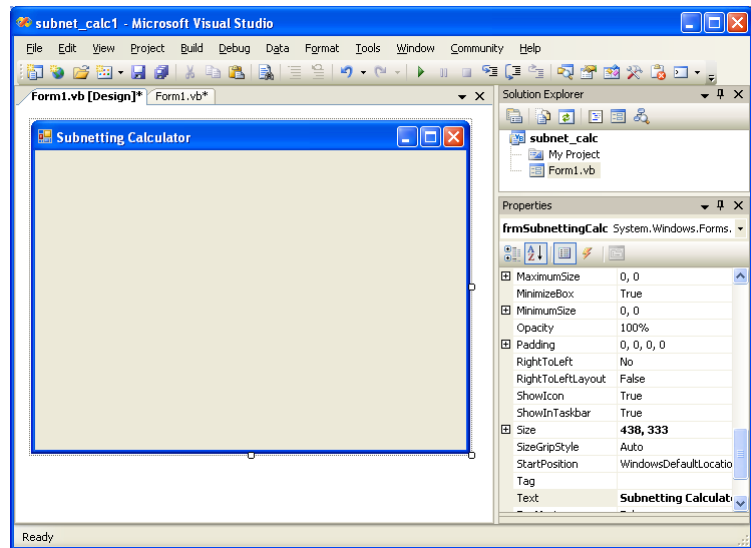


Figure 7.5 – Setting the Form Properties

The background color will change to a white. 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.

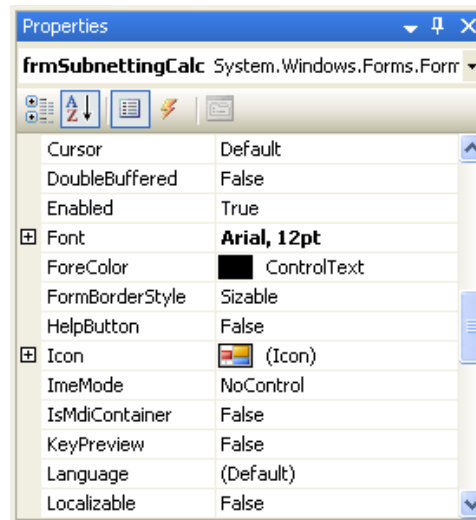


Figure 7.6 – The Font Window in Visual Basic

We will select the Arial font, Regular font style and 12 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.

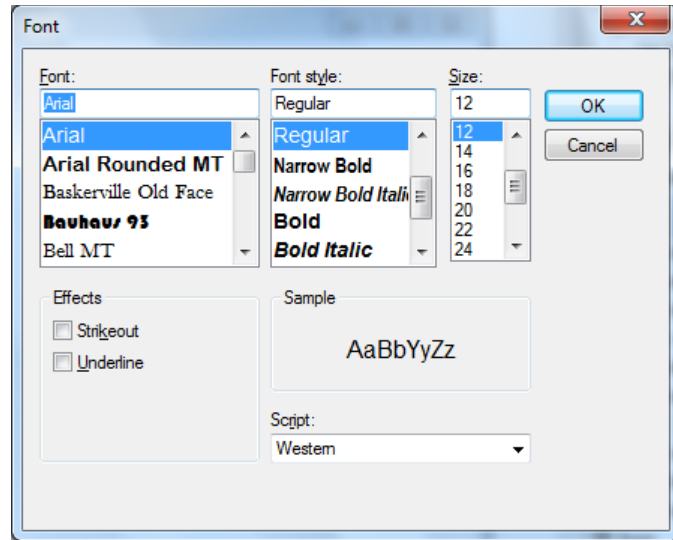


Figure 7.7 – Changing the Font to Arial

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.

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 **lblNetworkAddress**.

Alphabetic	
(Name)	lblNetworkAddress
Text	Network Address

Since the font is already set, we just type “Network Address” at the text attribute.

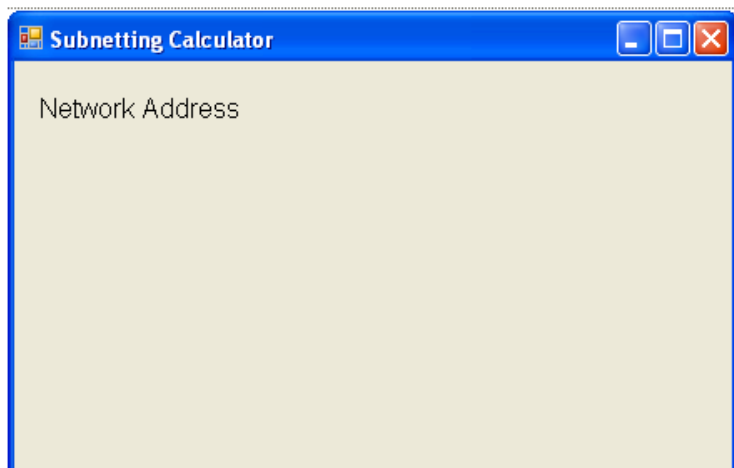


Figure 7.8 – The Finished Label on the Form

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.

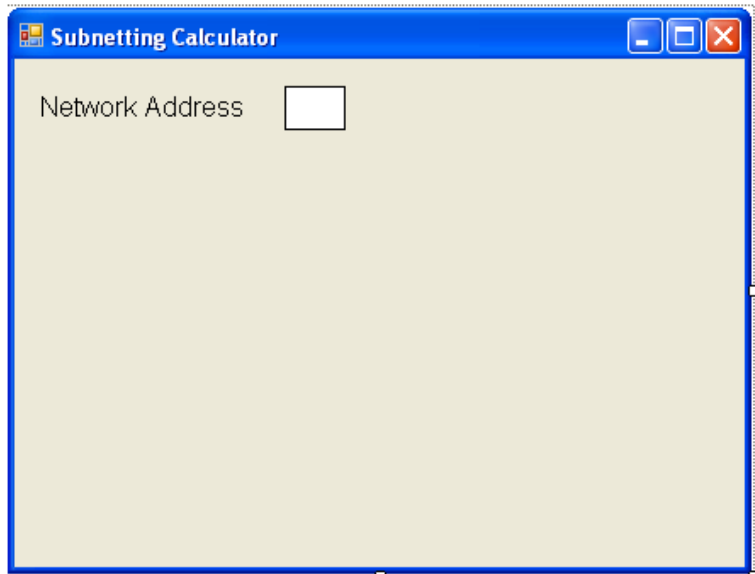


Figure 7.9 – Placing a TextBox on the Form

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 **txtOctet1**.

Alphabetic	
(Name)	txtOctet1
BackColor	White
Size	36, 26
TextAlign	Center

The size of the textbox will be 36 wide and 26 tall and the characters inside the textbox will be center aligned. The BackColor will be white.

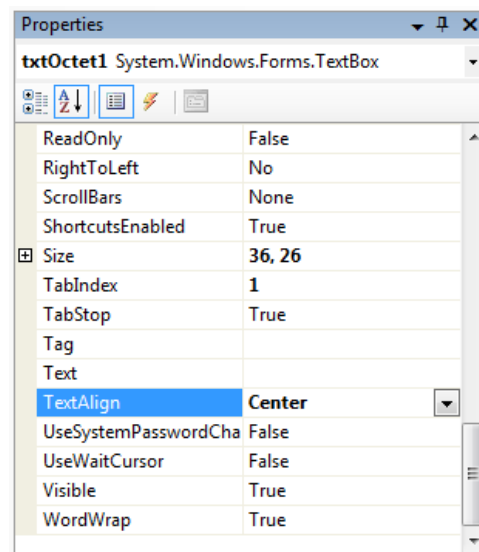


Figure 7.10 – Setting the Size of the Textbox

We will insert three labels named lblDot1, 2 and 3 that display a period between each octet textbox. We will insert three more textboxes named **Octet2**, **Octet3** and **Octet4** as shown in figure 7.11.

We will modify the properties for the textboxes as shown below.

Alphabetic	
(Name)	txtOctet2
BackColor	White
Size	36, 26
TextAlign	Center

Alphabetic	
(Name)	txtOctet3
BackColor	White
Size	36, 26
TextAlign	Center

Alphabetic	
(Name)	txtOctet4
BackColor	White
Size	36, 26
TextAlign	Center

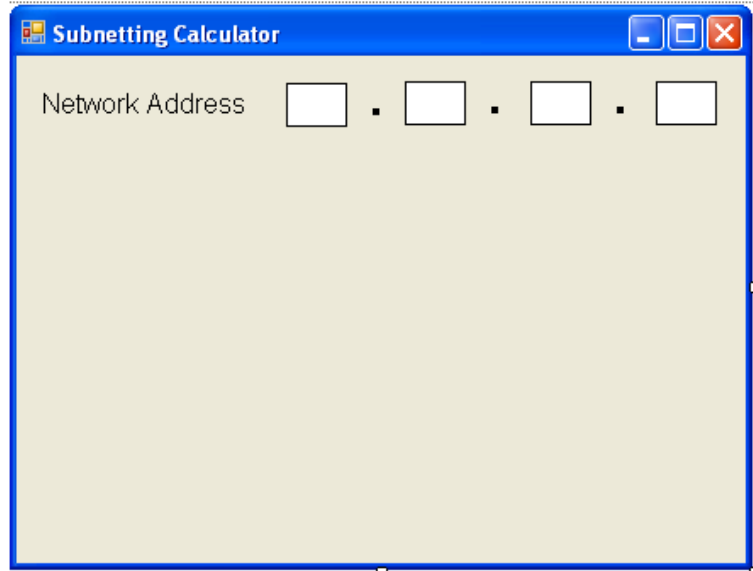


Figure 7.11 – Adding another Label

We add another row of label and a textbox. The following are the key properties.

Alphabetic	
(Name)	lblSubnetQty
Text	Number of Subnets

Alphabetic	
(Name)	txtSubnetQty
BackColor	White
Size	36, 26

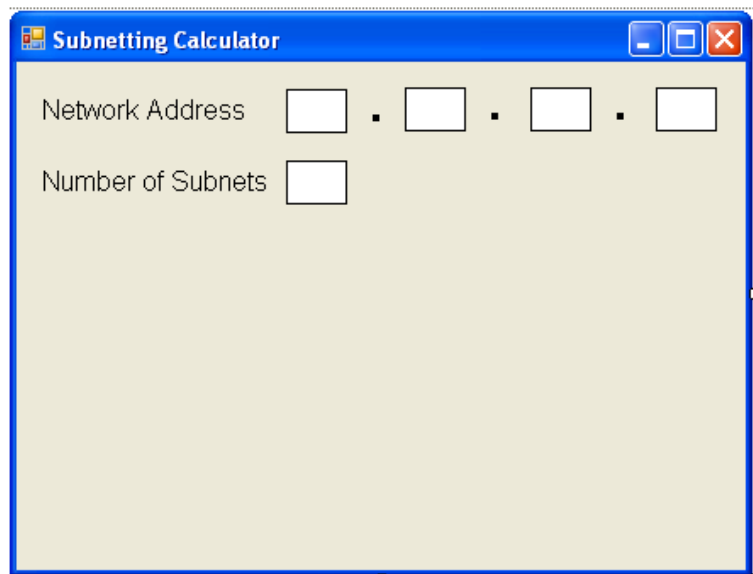


Figure 7.12 – Second Row of Label and Textbox

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.

We will place a label under **lblSubnetQty** label and call it **lblSubnetMask**. We will make the label text Subnet Mask. The key attributes for the label are:

Alphabetic	
(Name)	lblSubnetMask
Text	Subnet Mask

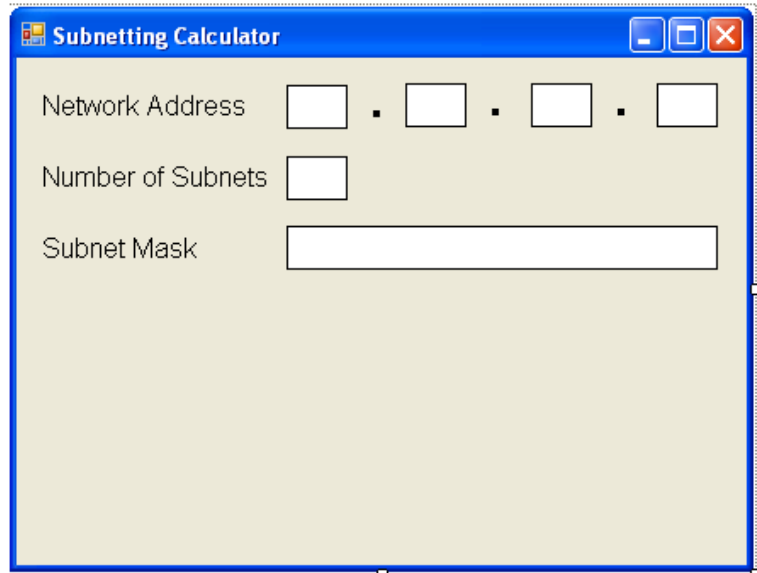


Figure 7.13 – Placing a Label for the Answer

We will insert the label for the answer to the right of **lblSubnetMask** and name the label **lblSubnetMaskAnswer**.

Alphabetic	
(Name)	lblSubnetMaskAnswer
BackColor	White
BorderStyle	FixedSingle
Size	254,26
TextAlign	Middle left

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

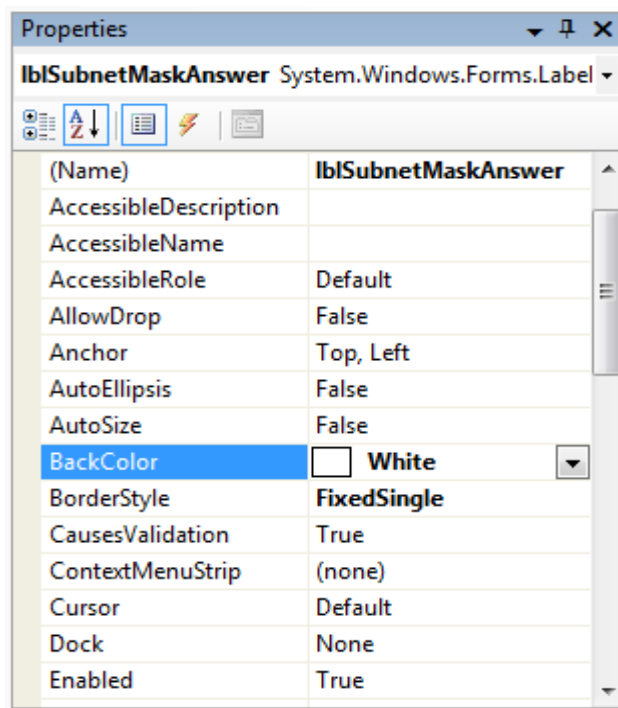


Figure 7.14 – BackColor is White

We will place another label under **lblSubnetMask** label and call it **lblHostQty**. We will make the label text Subnet Mask. The key attributes for the label are:

Alphabetic	
(Name)	lblHost Qty
Text	Subnet Mask

We will insert the label for the answer to the right of **lblHostQty** and name the label **lblHostQtyAnswer**.

Alphabetic	
(Name)	lblHostQtyAnswer
BackColor	White
BorderStyle	FixedSingle
Size	254,26
TextAlign	Middle left

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

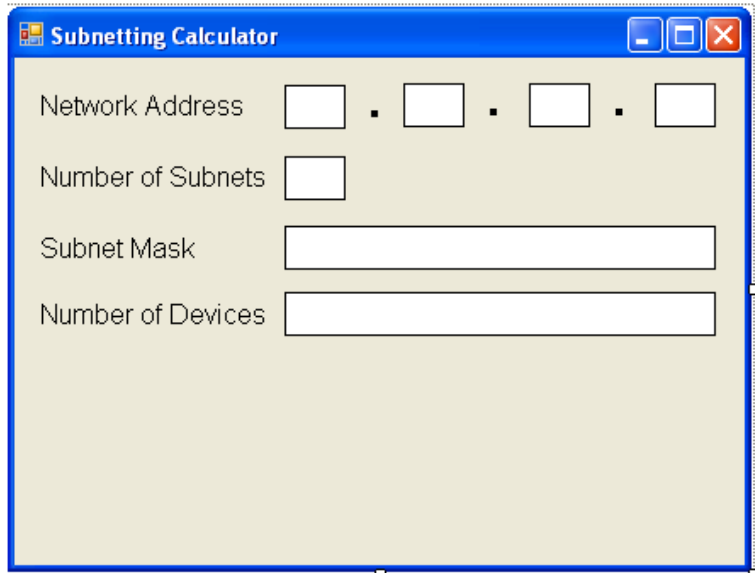


Figure 7.16 – Label Name is lblPower

We will place another label under **lblHostQty** label and call it **lblFilename**. We will make the label text File Name. The key attributes for the label are:

Alphabetic	
(Name)	lblFilename
Text	File Name

We will insert a textbox for the input to the right of **lblFilename** and name the textbox **txtFilename**.

Alphabetic	
(Name)	txtFilename
BackColor	White
BorderStyle	FixedSingle
Size	200,26
TextAlign	Center

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

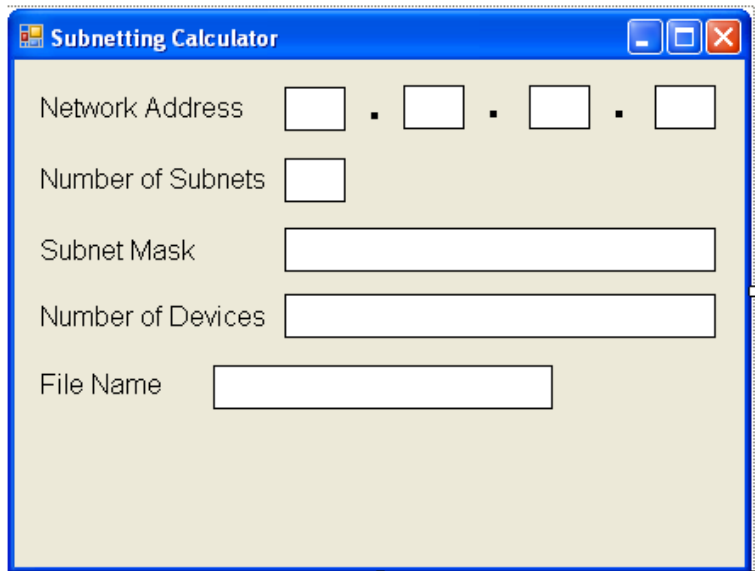


Figure 7.17 – Label Name is lblWattsAnswer

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 7.18.

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

Alphabetic	
(Name)	cmdBrowse
Caption	Browse
Size	90,29

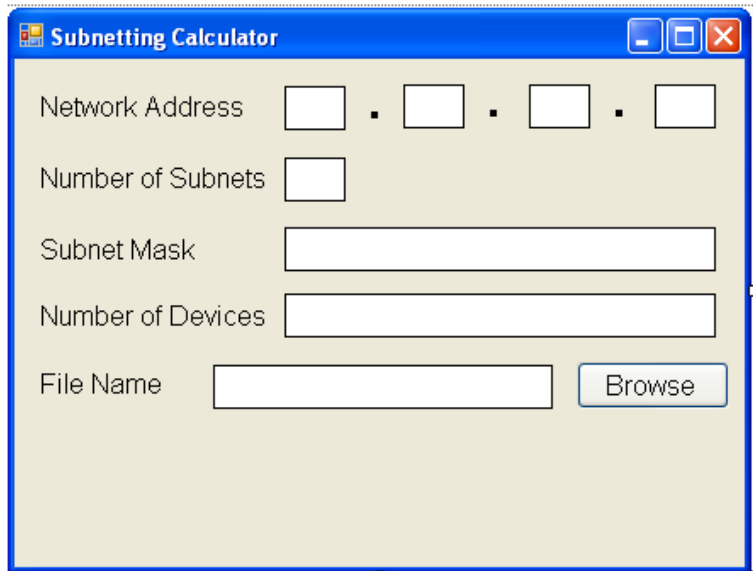


Figure 7.18 – The Command cmdCalculate Button

Add a second Command button, named cmdCalculate computing the subnets. Add a third Command button, named cmdOpenFile to view the output. The fourth Command button is for clearing the textboxes and output labels. The fifth 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.

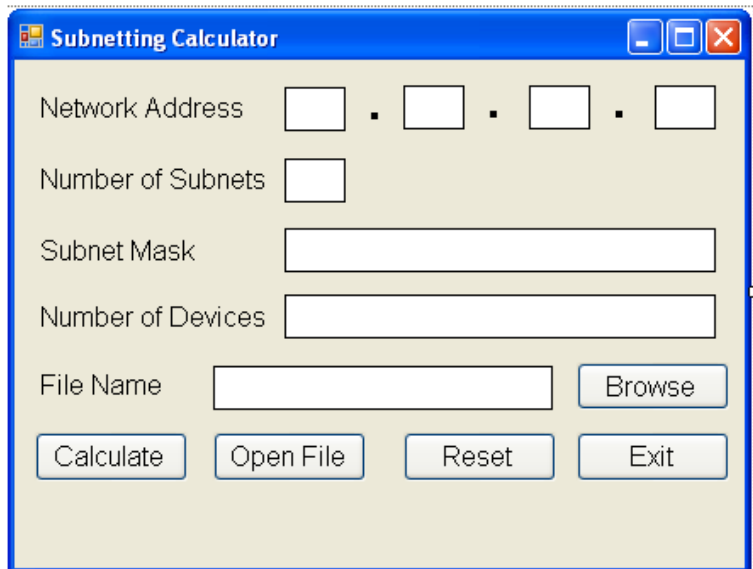


Figure 7.19 – Insert Two More Command Buttons

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.

Subnetting Calculator.vb 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.

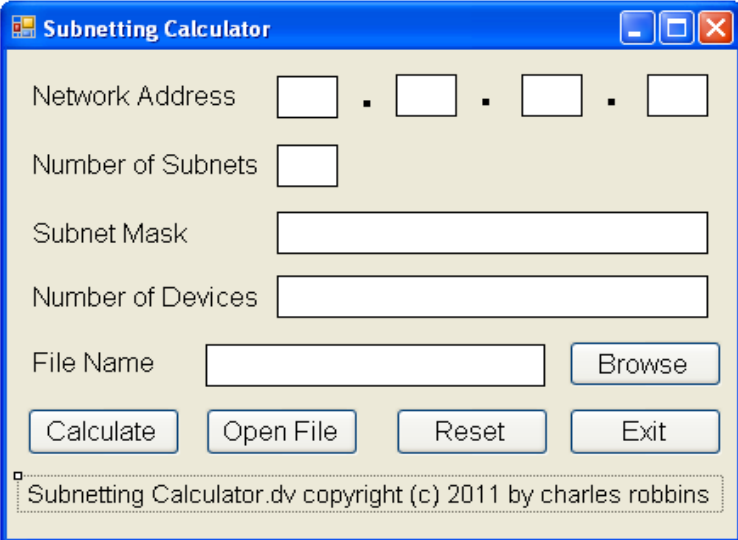


Figure 7.20 – Adding a Copyright Statement

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:

```
'Subnetting Calculator.vb copyright (c) 2011 by Charles W. Robbins  
'This program will open a dialogue box, allow the user to type the network address, the number of  
'subnets and the filename that will hold each subnet network address, IP address range and  
'broadcast address. When the user clicks on the Calculate button, all is calculated.
```

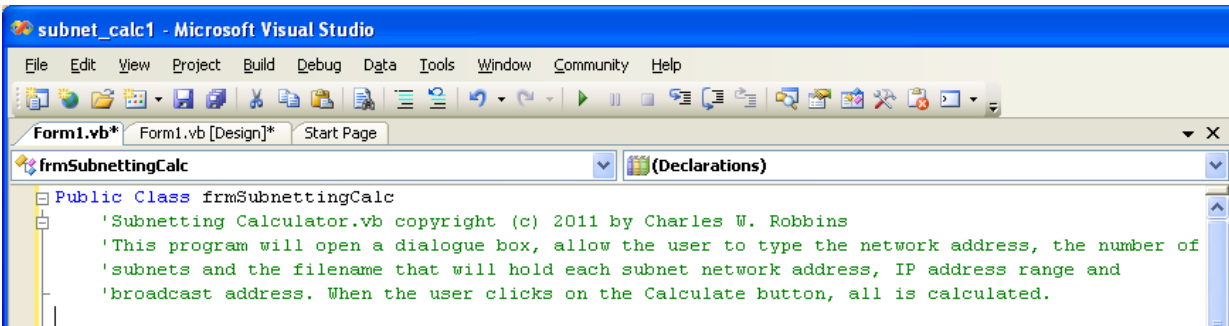


Figure 7.24 – Adding a Copyright Statement

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.

Under the Public Class we will declare two variable using public statements, which are **path** and **newfile**. Both of these variables are strings and the first hold the characters that describe the folder where the text file for the subnets. The second variable is the actual text file name.

```

Public Class frmSubnettingCalc
    Public path As String
    Public newfile As String

```

The next ten variables are the logical holders of data for the application. The first four **octet1** through **octet4** hold the TCP/IP version4 network address. The **subnet** variable holds the quantity of subnets we wish to compute. When the last octet for the subnet mask is computed it will be stored in the variable **subnet_last_octet**. The **netIP** variable will store the IP address for each subnet. The **IPrange** variable will store the IP address range for each subnet. The **Broadcast** variable will store the broadcast IP address for each subnet. Lastly, the **counter** variable holds the integer that advances by one on each cycle of the while loop.

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

```

Private Sub cmdCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles cmdCalculate.Click
    'declare variables
    Dim octet1 As Double
    Dim octet2 As Double
    Dim octet3 As Double
    Dim octet4 As Double
    Dim subnet As Double

```

```

Dim subnet_last_octet As String
Dim netIP As String
Dim IpRange As String
Dim Broadcast As String
Dim counter As Double

```

```

Form1.vb [Design]* Form1.vb*
cmdReset Click
Public Class frmSubnettingCalc
    'Subnetting Calculator.vb copyright (c) 2011 by Charles W. Robbins
    'This program will open a dialogue box, allow the user to type the network address, the number
    'of subnets and the filename that will hold each subnet network address, IP address range and
    'broadcast address. When the user clicks on the Calculate button, all is calculated.
    Public path As String
    Public newfile As String

    Private Sub cmdCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
        'declare variables
        Dim octet1 As Double
        Dim octet2 As Double
        Dim octet3 As Double
        Dim octet4 As Double
        Dim subnet As Double
        Dim subnet_last_octet As String
        Dim netIP As String
        Dim IpRange As String
        Dim Broadcast As String
        Dim counter As Double
    End Sub
End Class

```

Figure 7.25 – 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 octet variables to the numbers in the four textboxes. We capture the subnet quantity in the **subnet** variable and we store the file name in the variable **newfile**.

Type the following code under the “set variable” section of the cmdCalculate subroutine of the program.

```

'set variables
octet1 = txtOctet1.Text
octet2 = txtOctet2.Text
octet3 = txtOctet3.Text

```

```

octet4 = txtOctet4.Text
subnet = txtSubnetQty.Text
newfile = txtFileName.Text

```

```

' set variables
octet1 = txtOctet1.Text
octet2 = txtOctet2.Text
octet3 = txtOctet3.Text
octet4 = txtOctet4.Text
subnet = txtSubnetQty.Text
newfile = txtFileName.Text

```

Figure 7.26 – Setting the Variables in the VBA Code

Using Condition Statements

In this section of the code, we will use the condition statement to compute the subnet mask's last octet. If the subnet quantity is 1 then the `subnet_last_octet` variable will be 0 and the subnet quantity is still 1. We can see this in the If Then Else statement

```

If subnet = 1 Then
    subnet_last_octet = "0"
    subnet = 1
End If

```

However the next condition statement uses the And function to test for two conditions simultaneously which is subnet greater than and less or equal to two. If the subnet quantity meets these conditions then the `subnet_last_octet` variable will be 2 and the subnet quantity is still 2. We can copy each If-Then expression and change it for the progression:

{1,2,4,8,16,32,64,128,256}

```

' calculate mask
If subnet = 1 Then
    subnet_last_octet = "0"
    subnet = 1
End If
If subnet > 1 And subnet <= 2 Then
    subnet_last_octet = "128"
    subnet = 2
End If
If subnet > 2 And subnet <= 4 Then
    subnet_last_octet = "192"
    subnet = 4
End If
If subnet > 4 And subnet <= 8 Then
    subnet_last_octet = "224"
    subnet = 8
End If
If subnet > 8 And subnet <= 16 Then
    subnet_last_octet = "240"
    subnet = 16
End If
If subnet > 16 And subnet <= 32 Then
    subnet_last_octet = "248"
    subnet = 32
End If
If subnet > 32 And subnet <= 64 Then
    subnet_last_octet = "252"
    subnet = 64
End If
If subnet > 64 And subnet <= 128 Then
    subnet_last_octet = "254"
    subnet = 128
End If
If subnet > 128 And subnet <= 256 Then
    subnet_last_octet = "255"
    subnet = 256
End If

```

Figure 7.27 – Displaying the Answers

Writing Information to a Text File

The next three sections are also repetitive and somewhat easy to write. The first thing we do is check the first octet against the range for each class in the TCP/IP scheme. The following is the permissible range for each class.

Class	Range
A	1-126
B	128-191
C	192-223

Then we build the string to write to the label, **lblSubnetMaskAnswer.Text** by starting out with 225 and the **subnet_last_octet** variable and then **“.0.0”**. After that, we construct the number of devices on each subnet to write to the label, **lblHostQtyAnswer.Text** by computing 2 to the 24 power divided by the number of subnets and then subtracting two.

We open the newfile using **My.Computer.FileSystem.WriteAllText(path & newfile & ".txt"** and write the headers **"Subnet" & vbTab & "Network IP" & vbTab & vbTab & "IP Range" & vbTab & vbTab & vbTab & "Broadcast IP"** above our soon to be written table and add a newline using the **vbCrLf** expression. We use **vbTab** to tab in the text file between strings.

We set the variable counter to zero when we start the while loop. The loop will set the network IP address, the broadcast IP address and the IP range and then write the information to the text file. We add one to the counter before starting through the cycle again.

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

```
'set subnet mask for class A network
If octet1 >= 1 And octet1 <= 126 Then
    lblSubnetMaskAnswer.Text = "255." & subnet_last_octet & ".0.0"
    lblHostQtyAnswer.Text = 2 ^ 24 / subnet - 2
    My.Computer.FileSystem.WriteAllText(path & newfile & ".txt", "Subnet" & vbTab & "Network IP" & vbTab & vbTab &
    "IP Range" & vbTab & vbTab & vbTab & "Broadcast IP" & vbCrLf, True)
    counter = 0
    Do While counter < subnet
        netIP = Str(octet1) & "." & Str(octet2 + (256 / subnet) * counter) & "." & Str(octet3) & "." & Str(octet4)
        Broadcast = Str(octet1) & "." & Str(octet2 + (256 / subnet) * (counter + 1) - 1) & ".255" & ".255"
        IpRange = Str(octet1) & "." & Str(octet2 + (256 / subnet) * counter) & "." & Str(octet3) & "." & Str(octet4 + 1) & "-" &
        Str(octet1) & "." & Str(octet2 + (256 / subnet) * (counter + 1) - 1) & ".255" & ".254"
        My.Computer.FileSystem.WriteAllText(path & newfile & ".txt", " " & Str(counter + 1) & vbTab & netIP & vbTab &
        IpRange & vbTab & Broadcast & vbCrLf, True)
        counter = counter + 1
    Loop
End If
```

We can copy the class A section and make small changes to the program because the class B subnet mask makes it changes in the third octet.

'set subnet mask for class B network

If octet1 >= 128 And octet1 <= 191 Then

lblSubnetMaskAnswer.Text = "255.255." & subnet_last_octet & ".0"

lblHostQtyAnswer.Text = 2 ^ 16 / subnet - 2

counter = 0

My.Computer.FileSystem.WriteAllText(path & newfile & ".txt", "Subnet" & vbTab & "Network IP" & vbTab & vbTab & "IP Range" & vbTab & vbTab & vbTab & "Broadcast IP" & vbCrLf, True)

Do While counter < subnet

netIP = Str(octet1) & "." & Str(octet2) & "." & Str(octet3 + (256 / subnet) * counter) & "." & Str(octet4)

Broadcast = Str(octet1) & "." & Str(octet2) & "." & Str(octet3 + (256 / subnet) * (counter + 1) - 1) & ".255"

IpRange = Str(octet1) & "." & Str(octet2) & "." & Str(octet3 + (256 / subnet) * counter) & "." & Str(octet4 + 1) & "-" & Str(octet1) & "." & Str(octet2) & "." & Str(octet3 + (256 / subnet) * (counter + 1) - 1) & ".254"

My.Computer.FileSystem.WriteAllText(path & newfile & ".txt", " " & Str(counter + 1) & vbTab & netIP & vbTab & IpRange & vbTab & Broadcast & vbCrLf, True)

counter = counter + 1

Loop

End If

We can copy the class A section and make small changes to the program because the class C subnet mask makes it changes in the third octet.

'set subnet mask for class C network

If octet1 >= 192 And octet1 <= 223 Then

lblSubnetMaskAnswer.Text = "255.255.255." & subnet_last_octet

lblHostQtyAnswer.Text = 2 ^ 8 / subnet - 2

counter = 0

My.Computer.FileSystem.WriteAllText(path & newfile & ".txt", "Subnet" & vbTab & "Network IP" & vbTab & vbTab & "IP Range" & vbTab & vbTab & vbTab & "Broadcast IP" & vbCrLf, True)

Do While counter < subnet

netIP = Str(octet1) & "." & Str(octet2) & "." & Str(octet3) & "." & Str(octet4 + (256 / subnet) * counter)

Broadcast = Str(octet1) & "." & Str(octet2) & "." & Str(octet3) & "." & Str(octet4 + (256 / subnet) * (counter + 1) - 1)

IpRange = Str(octet1) & "." & Str(octet2) & "." & Str(octet3) & "." & Str(octet4 + (256 / subnet) * counter + 1) & "-" & Str(octet1) & "." & Str(octet2) & "." & Str(octet3) & "." & Str(octet4 + (256 / subnet) * (counter + 1) - 2)

My.Computer.FileSystem.WriteAllText(path & newfile & ".txt", " " & Str(counter + 1) & vbTab & netIP & vbTab & IpRange & vbTab & Broadcast & vbCrLf, True)

counter = counter + 1

Loop

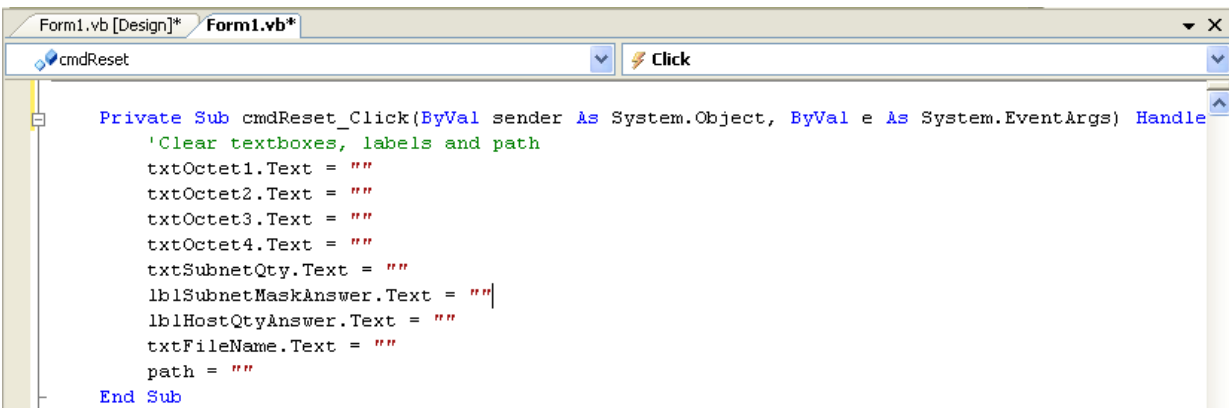
End If

Resetting the Data

To clear the textboxes or labels containing the data, we will clear the four octet textboxes, the subnet quantity textbox, the subnet mask answer textbox, the host quantity textbox, the text file textbox, and the path. Type the following code under the cmdReset subroutine of the program

'Clear textboxes, labels and path

```
txtOctet1.Text = ""  
txtOctet2.Text = ""  
txtOctet3.Text = ""  
txtOctet4.Text = ""  
txtSubnetQty.Text = ""  
lblSubnetMaskAnswer.Text = ""  
lblHostQtyAnswer.Text = ""  
txtFileName.Text = ""  
path = ""
```

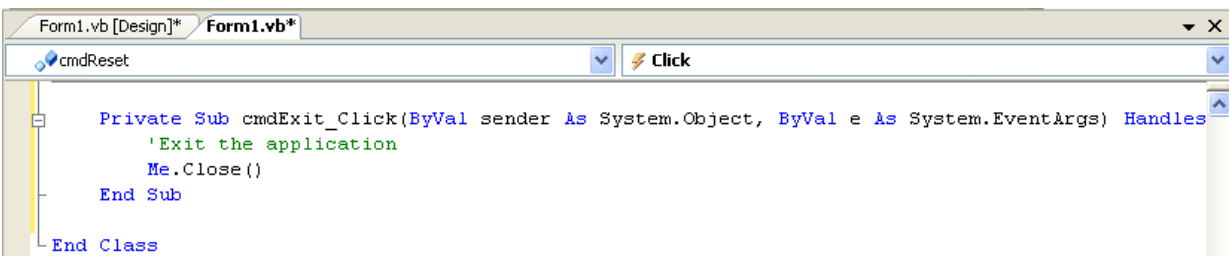


The screenshot shows a Visual Studio window with the code editor open. The code is for a Private Sub cmdReset_Click, which is triggered by a Click event. The code sets the Text property of several controls to an empty string. The controls are txtOctet1, txtOctet2, txtOctet3, txtOctet4, txtSubnetQty, lblSubnetMaskAnswer, lblHostQtyAnswer, txtFileName, and path. The code is as follows:

```
Private Sub cmdReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
    'Clear textboxes, labels and path  
    txtOctet1.Text = ""  
    txtOctet2.Text = ""  
    txtOctet3.Text = ""  
    txtOctet4.Text = ""  
    txtSubnetQty.Text = ""  
    lblSubnetMaskAnswer.Text = ""  
    lblHostQtyAnswer.Text = ""  
    txtFileName.Text = ""  
    path = ""  
End Sub
```

Figure 7.28 – Computing the Reset Button by Clearing a Textbox and Label Caption

Exiting the Program



The screenshot shows a Visual Studio window with the code editor open. The code is for a Private Sub cmdExit_Click, which is triggered by a Click event. The code calls Me.Close() to exit the application. The code is as follows:

```
Private Sub cmdExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles  
    'Exit the application  
    Me.Close()  
End Sub
```

End Class

Figure 7.29 – 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()

Programming a Browse and Open File Button

We found a way to add a Browse button from the Microsoft website¹. We wanted a way for the user to pick a folder that will hold the subnet text file. In this section of code, we place under the Browse command button.

'Find the folder to write the subnet file to

```
Dim FolderBrowser As New FolderBrowserDialog
```

```
FolderBrowser.ShowNewFolderButton = False
```

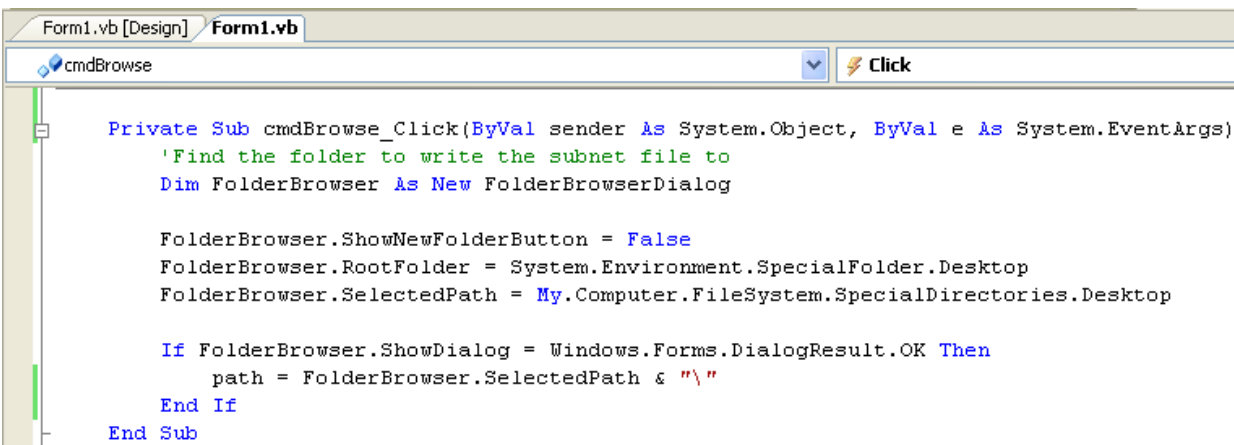
```
FolderBrowser.RootFolder = System.Environment.SpecialFolder.Desktop
```

```
FolderBrowser.SelectedPath = My.Computer.FileSystem.SpecialDirectories.Desktop
```

```
If FolderBrowser.ShowDialog = Windows.Forms.DialogResult.OK Then
```

```
    path = FolderBrowser.SelectedPath & "\"
```

```
End If
```



The screenshot shows the Visual Studio IDE with the code editor open to Form1.vb. The code is for the Click event of a button named cmdBrowse. The code is as follows:

```
Private Sub cmdBrowse_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    'Find the folder to write the subnet file to
    Dim FolderBrowser As New FolderBrowserDialog

    FolderBrowser.ShowNewFolderButton = False
    FolderBrowser.RootFolder = System.Environment.SpecialFolder.Desktop
    FolderBrowser.SelectedPath = My.Computer.FileSystem.SpecialDirectories.Desktop

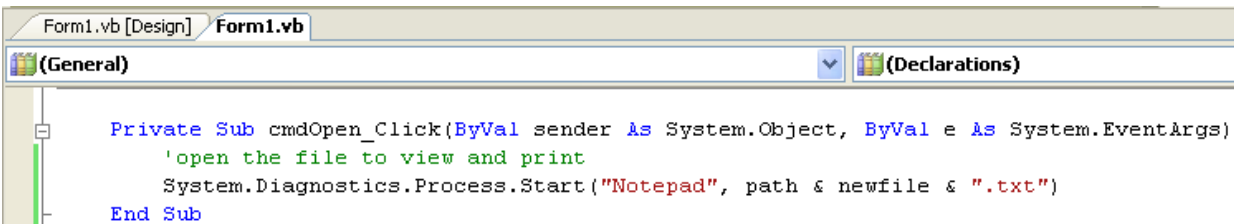
    If FolderBrowser.ShowDialog = Windows.Forms.DialogResult.OK Then
        path = FolderBrowser.SelectedPath & "\"
    End If
End Sub
```

Figure 7.30 – Launching the Program

Under the Open Command button we write this expression to open the text file.

'open the file to view and print

```
System.Diagnostics.Process.Start("Notepad", path & newfile & ".txt")
```



The screenshot shows the Visual Studio IDE with the code editor open to Form1.vb. The code is for the Click event of a button named cmdOpen. The code is as follows:

```
Private Sub cmdOpen_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    'open the file to view and print
    System.Diagnostics.Process.Start("Notepad", path & newfile & ".txt")
End Sub
```

¹FolderBrowserDialog.RootFolder Property, 2011, Microsoft, June 5, 2011

<<http://msdn.microsoft.com/en-us/library/system.windows.forms.folderbrowserdialog.rootfolder.aspx>>

Running the Program

After noting that the program is saved, press the F5 to run the Subnetting Calculator application. The Subnetting Calculator window will appear on the graphical display as shown in Figure 7.30. Notice the professional appearance and presentation of information in a clean dialogue box.

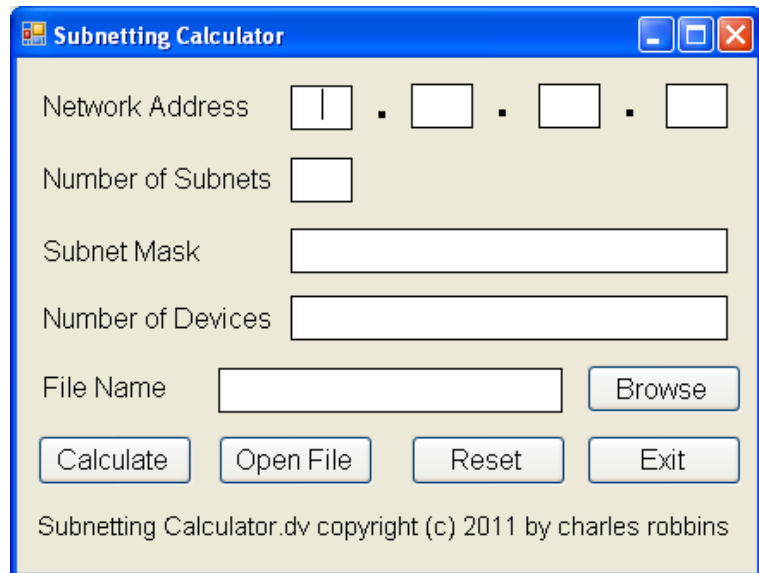


Figure 7.30 – Launching the Program

Type the network IP address and the number of subnets in the textboxes just as we typed as shown in Figure 7.31. If we make a mistake, we can type over the text entry or press the Reset command button to clear the textbox. Type the file name.

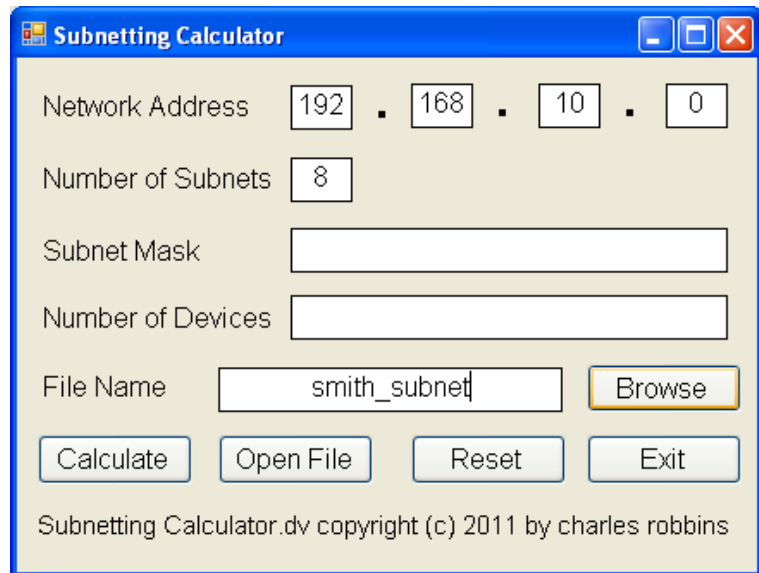


Figure 7.31 – Running the Program

Then click on the Browse button to locate a folder to hold the new text file. We will choose My Documents and then choose the OK button.

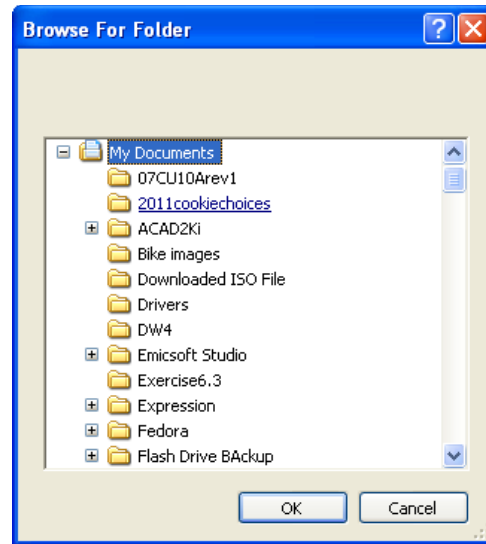


Figure 7.31 – Running the Program

Press the Calculate command button and the two answer labels will have the subnet mask and the number of devices.

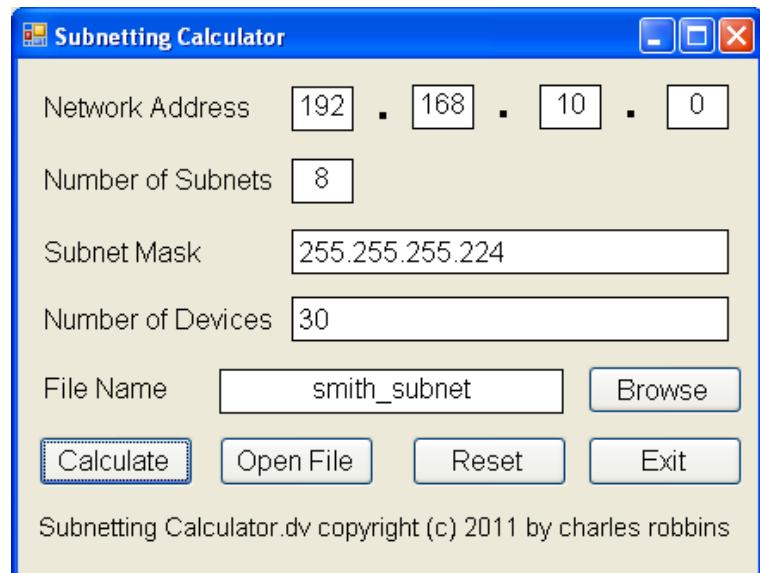


Figure 7.31 – Running the Program

We can pick the Open File command button and we can read each subnet mask. After experimenting with our program, press the Exit command button to exit the application.

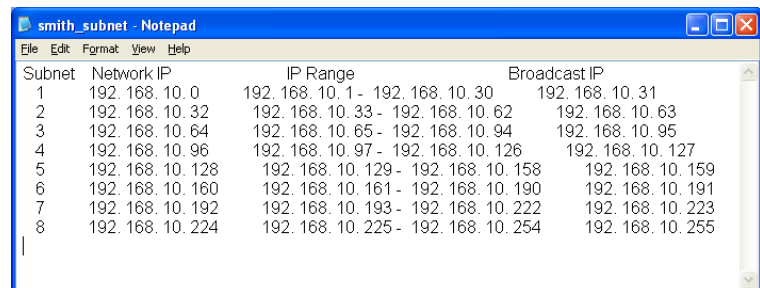


Figure 7.31 – Running the Program

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 90-6 * - Write a Visual Basic Application that writes a series of text strings to a text file. Incorporate the vbTab, and new line functions. Also, allow the user to create a file name and choose a path for the file using a Browse button. Complete the assignment in two hours to maintain your World Class CAD ranking.**

Continue this drill four times using some other form designs, each time completing the Visual Basic Project in less than 2 hour to maintain your World Class ranking.