

# Using Sub Procedures and Variables to Create Your Own Calculator

DocumentID : Calculator.PDF  
Author : Michele Harris  
Version : 1.0  
Date : 07-02-2009

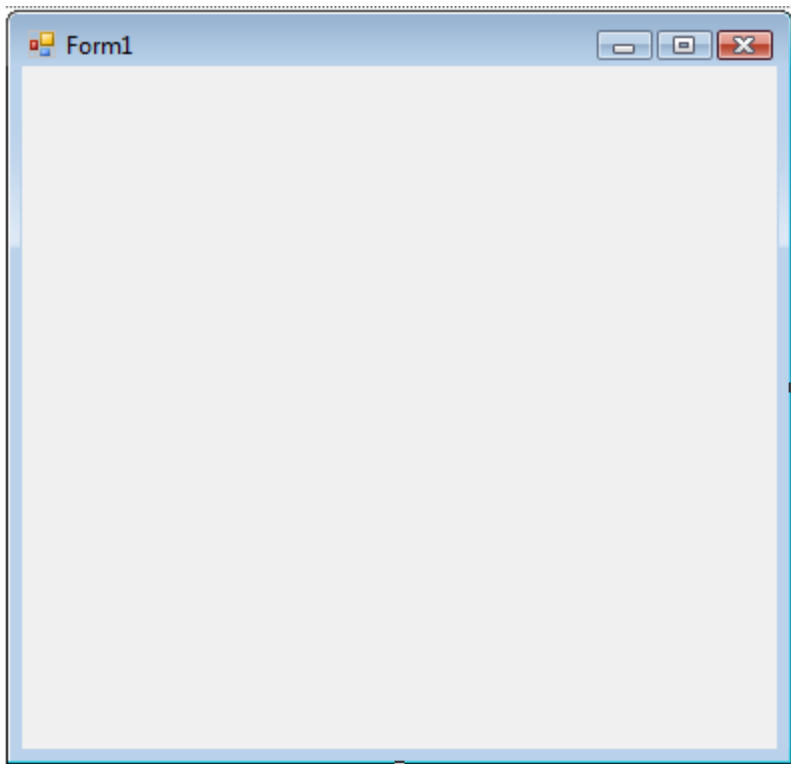
## Contents

Introduction .....	2
Adding and Formatting Objects .....	3
Declaring Our Variables .....	5
Programming Our Form Load Procedure .....	6
Programming Our Number Buttons .....	6
Programming the Clear Button .....	8
Programming the Operations Sub Procedures.....	9
Programming the Operations Buttons .....	12
Debugging Your Calculator.....	13
Our Calculator’s Limitations.....	15

---

## Introduction

In this example, we're going to create our own calculator! But before we can enter any code, we'll need to configure all the necessary objects on our form. Go to File: New Project in Visual Studio. Select the Windows Form option, then name your project "My Calculator." Hit enter, and your new form should appear.



## Adding and Formatting Objects

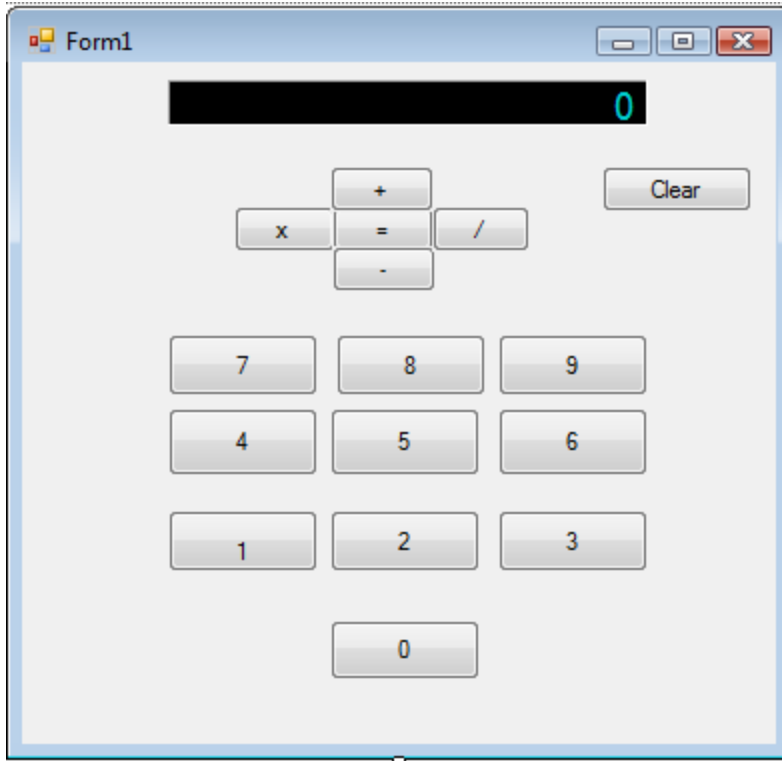
Next, we're going to need to add some controls to our form by Using the Toolbox.

Below is a table stating how many buttons and labels we'll need, as well as the names and properties for each.

Object to Add	Object Name	Object Text	Additional Properties
Label	lbl_calc	0	Font: Sans Serif 14 BackColor: Black ForeColor: Dark Turquoise TextAlign: Top Right
Button	btn_plus	+	
Button	btn_minus	-	
Button	btn_times	x	
Button	btn_divide	/	
Button	btn_equal	=	
Button	btn_clear	Clear	
Button	btn_1	1	
Button	btn_2	2	
Button	btn_3	3	
Button	btn_4	4	
Button	btn_5	5	
Button	btn_6	6	
Button	btn_7	7	
Button	btn_8	8	
Button	btn_9	9	
Button	btn_0	0	

Whew! Now that you've completed the laborious task of creating all your form's objects, you'll need to align them in a presentable way for your calculator. There's no right way to do this, but I'm sure that in your years of life, you've encountered enough calculators to have the format burnt on your eyelids. This might require you to resize

some of the buttons. Feel free to play with the font and colors, if you'd like. Here's how I arranged mine.



It needn't be a work of art, but feel free to make yours aesthetically pleasing, if you'd like. It's always good practice to figure out how to add background images, etc. So now that we finally have our design in place, let's get to work on the code!

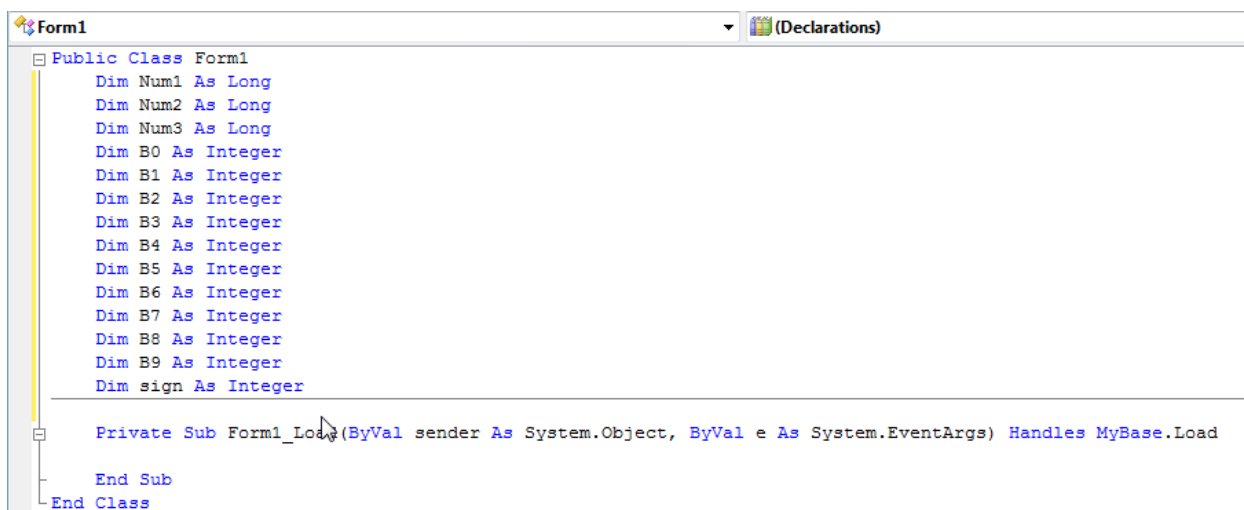
Double click your form (anywhere aside from the label and the buttons) to arrive at the Code Editor. You should see the following code already in place.

```
(Form1 Events) Load
Public Class Form1
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    End Sub
End Class
```

## Declaring Our Variables

Place your cursor right after “Public Class Form1.” Hit enter, and input the following code into the lines below.

```
Dim Num1 As Long
Dim Num2 As Long
Dim Num3 As Long
Dim B0 As Integer
Dim B1 As Integer
Dim B2 As Integer
Dim B3 As Integer
Dim B4 As Integer
Dim B5 As Integer
Dim B6 As Integer
Dim B7 As Integer
Dim B8 As Integer
Dim B9 As Integer
Dim sign As Integer
```



```
Public Class Form1
    Dim Num1 As Long
    Dim Num2 As Long
    Dim Num3 As Long
    Dim B0 As Integer
    Dim B1 As Integer
    Dim B2 As Integer
    Dim B3 As Integer
    Dim B4 As Integer
    Dim B5 As Integer
    Dim B6 As Integer
    Dim B7 As Integer
    Dim B8 As Integer
    Dim B9 As Integer
    Dim sign As Integer

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    End Sub
End Class
```

Here we’ve declared a number of variables. As you’ve learned, to declare a variable, you first place the word “Dim” in front of your variable name, and then designate what type of variable it’s meant to be—whether that be an integer, a string, boolean, long, short, etc. Here, Num1 and Num2 will later serve as containers to hold the numbers our

user inputs into the calculator. Num3 will be the resulting addition, subtracting, multiplication, or division of Num1 and Num2.

Variables B0 through B9 merely represent the single digit numbers on the calculator.

Sign is the integer representation the operation our user chooses. For instance, in this program, if Sign = 1, then our program will *add* Num1 and Num2 to arrive at Num3. If

Sign = 3, our program will multiply Num1 and Num2. But we're not done programming the code just yet.

## Programming Our Form Load Procedure

Place your cursor right after "Handles MyBase.Load." Hit enter, and type the following on the new line.

```
lbl_calc.Text = Nothing
```

So why would we want our label text to load with nothing in it? This statement allows us to open our calculator with a clean slate. Setting the text property of an object equal to nothing merely clears out any previous data displayed in that object.

## Programming Our Number Buttons

Next, we'll want to program the code for our number buttons. On the line following our previous "End Sub" statement, type the following code. (Note: your buttons **must** have the same names as mine.)

```
Private Sub btn_0_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btn_0.Click
```

---

```
        B0 = 0
        lbl_calc.Text = lbl_calc.Text + B0
    End Sub

    Private Sub btn_1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_1.Click
        B1 = 1
        lbl_calc.Text = lbl_calc.Text & B1
    End Sub

    Private Sub btn_2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_2.Click
        B2 = 2
        lbl_calc.Text = lbl_calc.Text & B2
    End Sub

    Private Sub btn_3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_3.Click
        B3 = 3
        lbl_calc.Text = lbl_calc.Text & B3
    End Sub

    Private Sub btn_4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_4.Click
        B4 = 4
        lbl_calc.Text = lbl_calc.Text & B4
    End Sub

    Private Sub btn_5_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_5.Click
        B5 = 5
        lbl_calc.Text = lbl_calc.Text & B5
    End Sub

    Private Sub btn_6_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_6.Click
        B6 = 6
        lbl_calc.Text = lbl_calc.Text & B6
    End Sub

    Private Sub btn_7_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_7.Click
        B7 = 7
        lbl_calc.Text = lbl_calc.Text & B7
    End Sub

    Private Sub btn_8_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_8.Click
        B8 = 8
        lbl_calc.Text = lbl_calc.Text & B8
    End Sub

    Private Sub btn_9_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_9.Click
        B9 = 9
```

---

```
    lbl_calc.Text = lbl_calc.Text & B9  
End Sub
```

So what exactly does this code do? Well, anytime your user presses the button 5, that number will appear on the calculator's screen. We accomplish this by setting the B5 variable equal to 5, and then making the text of our calculator equal to its current value and 5.

The line `lbl_calc.Text = lbl_calc.Text & B6` might be confusing for anyone who tries to think of that statement mathematically. After all, how can something be equal to itself plus another number? My advice is not to think of this equation mathematically. What it's really saying is this:

The new value for `lbl_calc` = The old value of `lbl_calc` with 5 appended to its right side.

For instance, let's say the old value of `lbl_calc` was 3. Using this statement, we would append 5 to its right side, which equals 35—the new value of `lbl_calc`.

## Programming the Clear Button

Now we need to add a little more code. Under the "End Sub" of the `btn_9_click` procedure, enter the following.

```
Private Sub btn_clear_Click(ByVal sender As System.Object, ByVal e As  
System.EventArgs) Handles btn_clear.Click  
    lbl_calc.Text = Nothing  
End Sub
```

---

Remember the `lbl_calc.Text` statement we used earlier for the Form load procedure? It comes in especially handy when for programming a “Clear” button for your calculator.

## Programming the Operations Sub Procedures

Next, we’re going to want to program the Sub Procedures that will dictate what goes on when the plus, minus, multiplication, division, and equals buttons are clicked. On the line under the last Sub Procedure you entered, (but before the End Class line of code) type the following.

```
Sub Plus()  
    Num1 = Val(lbl_calc.Text)  
    lbl_calc.Text = Nothing  
    sign = 1  
End Sub  
Sub Minus()  
    Num1 = Val(lbl_calc.Text)  
    lbl_calc.Text = Nothing  
    sign = 2  
End Sub  
Sub Times()  
    Num1 = Val(lbl_calc.Text)  
    lbl_calc.Text = Nothing  
    sign = 3  
End Sub  
Sub Divide()  
    Num1 = Val(lbl_calc.Text)  
    lbl_calc.Text = Nothing  
    sign = 4  
End Sub  
Sub Equaled()  
    Num2 = Val(lbl_calc.Text)  
    lbl_calc.Text = Nothing  
    If sign = 1 Then  
        Num3 = Num1 + Num2  
    End If  
    If sign = 2 Then  
        Num3 = Num1 - Num2  
    End If  
    If sign = 3 Then  
        Num3 = Num1 * Num2  
    End If  
    If sign = 4 Then  
        Num3 = Num1 / Num2
```

```
End If  
    lbl_calc.Text = Num3  
End Sub
```

So let's look at the Sub Plus() routine. As you'll notice, it's in the format of a basic Sub Procedure without any optional arguments. It's public procedure, meaning that it can be accessed by any of the individual objects or events contained in Form1.

But let's just look at the code line by line.

```
Num1 = Val(lbl_calc.Text).
```

This line assigns a value to our variable. Whatever number our user enters into the calculator before pressing the plus sign is now the value for Num1. Assigning this value to a variable is a lot like storing it in a box; we don't necessarily need to look at it right now, but we want to keep it handy and easily accessible for later.

You're well familiar with the next line by now: `lbl_calc.Text = Nothing`. As you might guess, this line erases the numbers displayed in our calculator's screen, so that new numbers can be entered.

Our final step is assigning a value to our sign variable. Here, if someone clicks the plus button, the value for sign will be 1. Again, we don't need to use this number right now, but we need to have this value around for when our user clicks the equals sign.

---

The steps for Sub Minus(), Sub Times(), and Sub Divide() look nearly identical, so let's take a look at the Sub Equaled procedure.

Here, again, we have before us a simple Sub Procedure. At first, Sub Equaled() starts off similar to Sub Plus(); again, we're taking the newly entered value in the calculator and assigning its value to Num2. And once more, we're using the lbl\_calc.Text = nothing statement to clear all values from our calculator's screen.

Here's where our code differs, however. The next twelve lines of code essentially figure out which operation we need to perform on Num1 and Num2—whether that's adding, subtracting, multiplying, and dividing. Remember that the Sub Plus() procedure assigned a value for the sign variable? Well, we're using a series of If/Then statements to detect just what that number was, and then to perform the correct operation.

So if sign = 1, we want to add the two numbers because our user has previously clicked the plus sign. If sign = 2, we'll want to subtract Num2 from Num1, because the user has previously clicked the minus sign, and so on.

After our If/Then statements determine which operation to perform and then perform it, assigning the resultant value to Num3, we now want to make our calculator screen

---

display the result. We accomplish this by making the text property of lbl\_calc equal to Num3.

## Programming the Operations Buttons

Whew! Almost done. So we've programmed the proper sub routines for figuring out our calculations. Next, we'll want to call those sub routines any time those respective buttons are clicked. Finally, enter the following code on the line after the last Sub procedure you typed.

```
Private Sub btn_plus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_plus.Click
    Plus()
End Sub
Private Sub btn_minus_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_minus.Click
    Minus()
End Sub

Private Sub btn_times_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_times.Click
    Times()
End Sub

Private Sub btn_divide_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_divide.Click
    Divide()
End Sub

Private Sub btn_equal_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btn_equal.Click
    Equaled()
End Sub
```

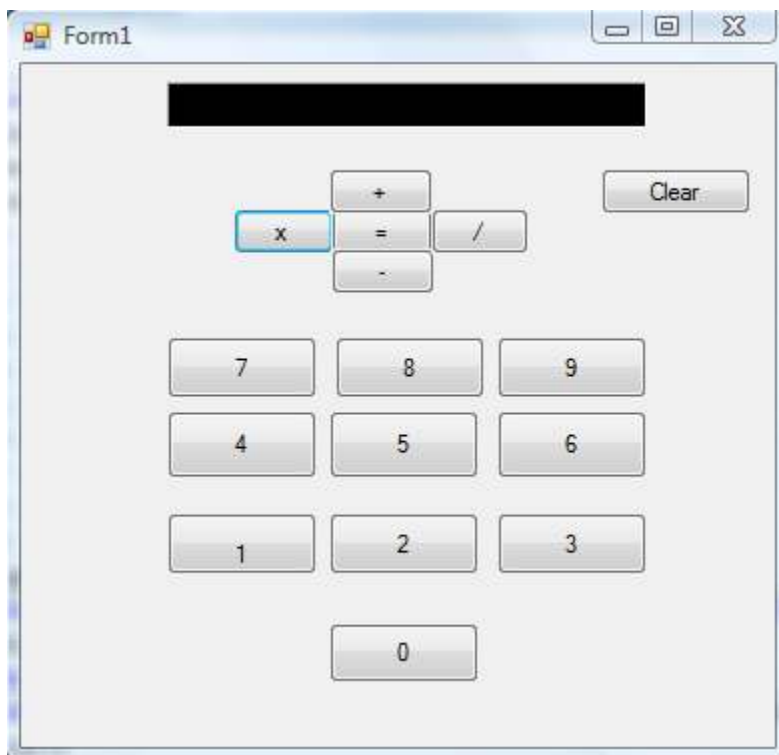
Now, before we begin the process of debugging, make sure there are no blue lines throughout your code indicating errors. If there are, carefully compare the code you typed to the code displayed here, and make sure it matches up perfectly. One tiny typo

---

can prevent our program from running. If you're still having trouble, make sure all your objects share the same name as mine; otherwise, your program will not function.

## Debugging Your Calculator

Press F5 to start debugging. Your calculator should appear in a new window. Notice that though we typed in "0" in the design view of the label, our `lbl_calc.Text = nothing` statement clears away any previous values.



Now that we have our newly created calculator open, feel free to experiment with it.

Let's say we want to divide 3594 by 2. Enter 3594 into the calculator using the number buttons.

Form1

3594

+

x = /

-

Clear

7 8 9

4 5 6

1 2 3

0

Now click the divide sign.

Form1

+

x = /

-

Clear

7 8 9

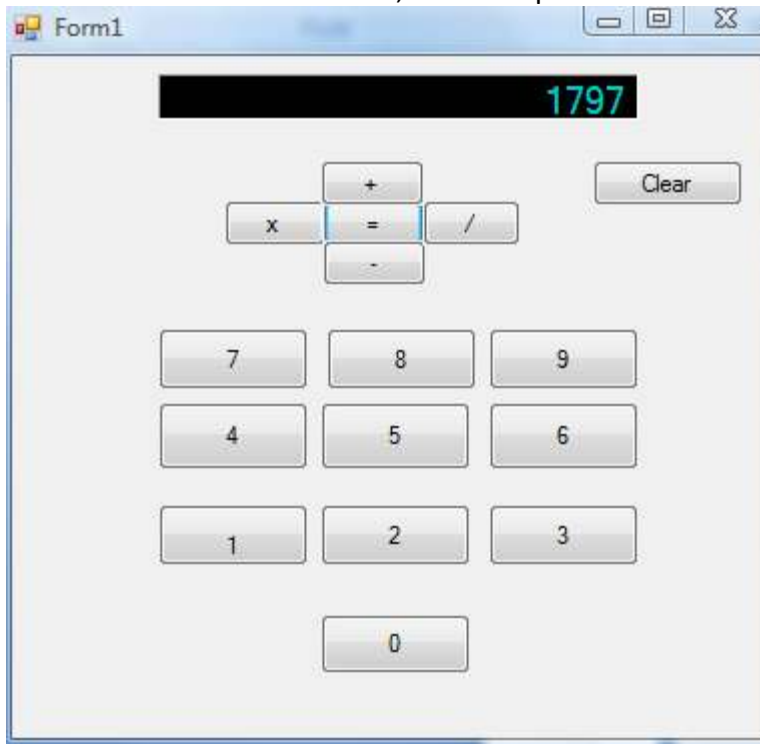
4 5 6

1 2 3

0

You'll notice that our number disappeared! No worries, though; the program we created will remember it for later.

Now click the 2 button, and hit equals.



Congratulations! You have a functional calculator!

## Our Calculator's Limitations

Now it's worth mentioning that this calculator has some limitations. For one, it can't return any decimal values (we'll leave that operation to another day). This means that any time you divide by a number that won't divide evenly, the calculator will display the rounded down calculation.

For instance, let's say we wanted to divide 9 by 4. The calculator will return 2—because 4 goes into 9 two whole times. Still, this isn't the precisely right value. Also, your calculator won't like it if you divide by 0. But what calculator does?

---