My First Polymer VB Program: Plot ΔS_{mix}

Assumption: you did OK in Qbasic or QB programming in Chem 4010.

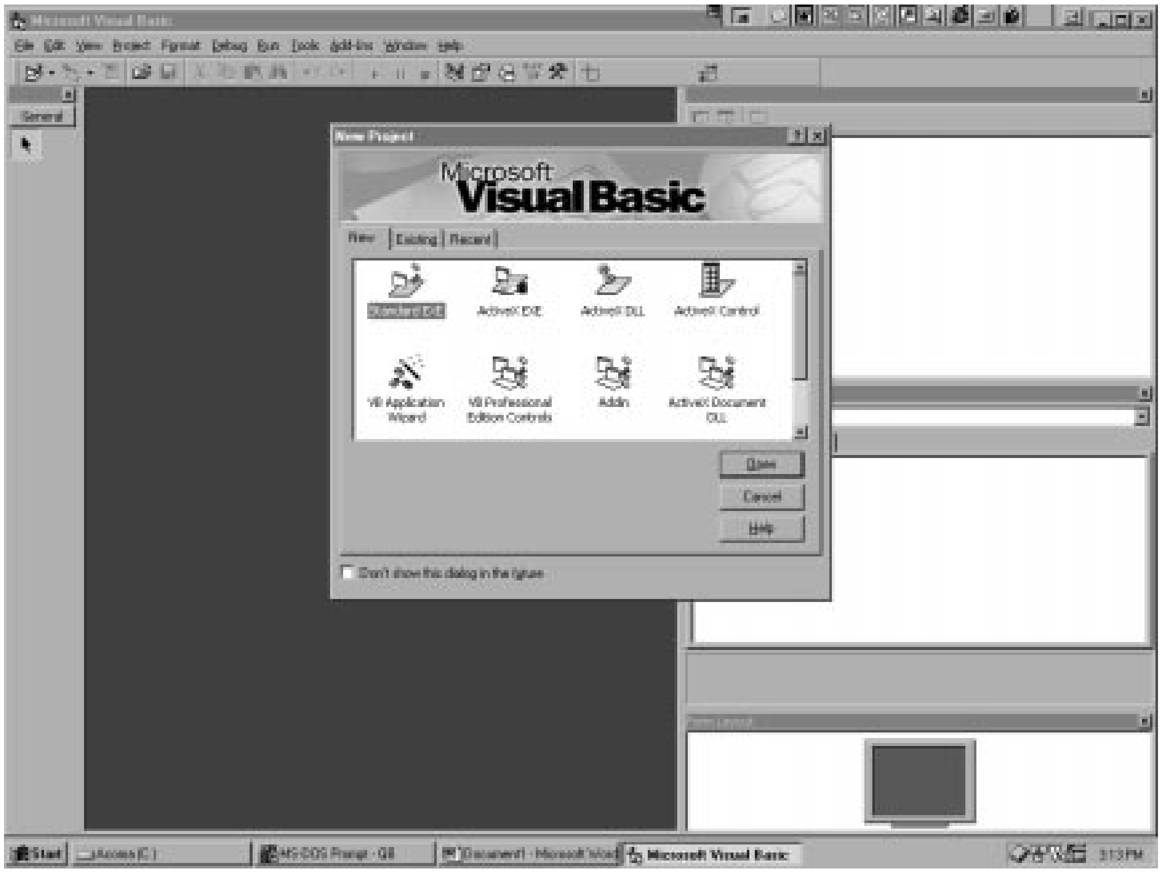
First install VB on your computer. Version 4.0 or higher is OK; earlier versions are a little goofy. As a Chemistry employee, you have a legal license to the latest version (yay!) so you can see Bob Zinn. If you install from my Microsoft Developer Network Toolkit disks, make sure you install the MSDN support. Otherwise, you will not have Help. Writing VB code is difficult without Help.

Next, go to Barnes & Noble, Books-a-Million or similar and get any book (e.g., VB for Dummies or Teach Yourself in 21 Days). Since I am not exactly a power user, I sometimes buy the book for the previous version (e.g., a 5.0 book for 6.0). These are heavily discounted. A VB5 book should be fine for your purposes.

Next, visit PAWs. Click on Student Services, then CBT. This gives you LSU's Computer Based Training, which has sections for VB and other cool stuff.

Launch the program (Start, Program Files, blah-blah-blah).

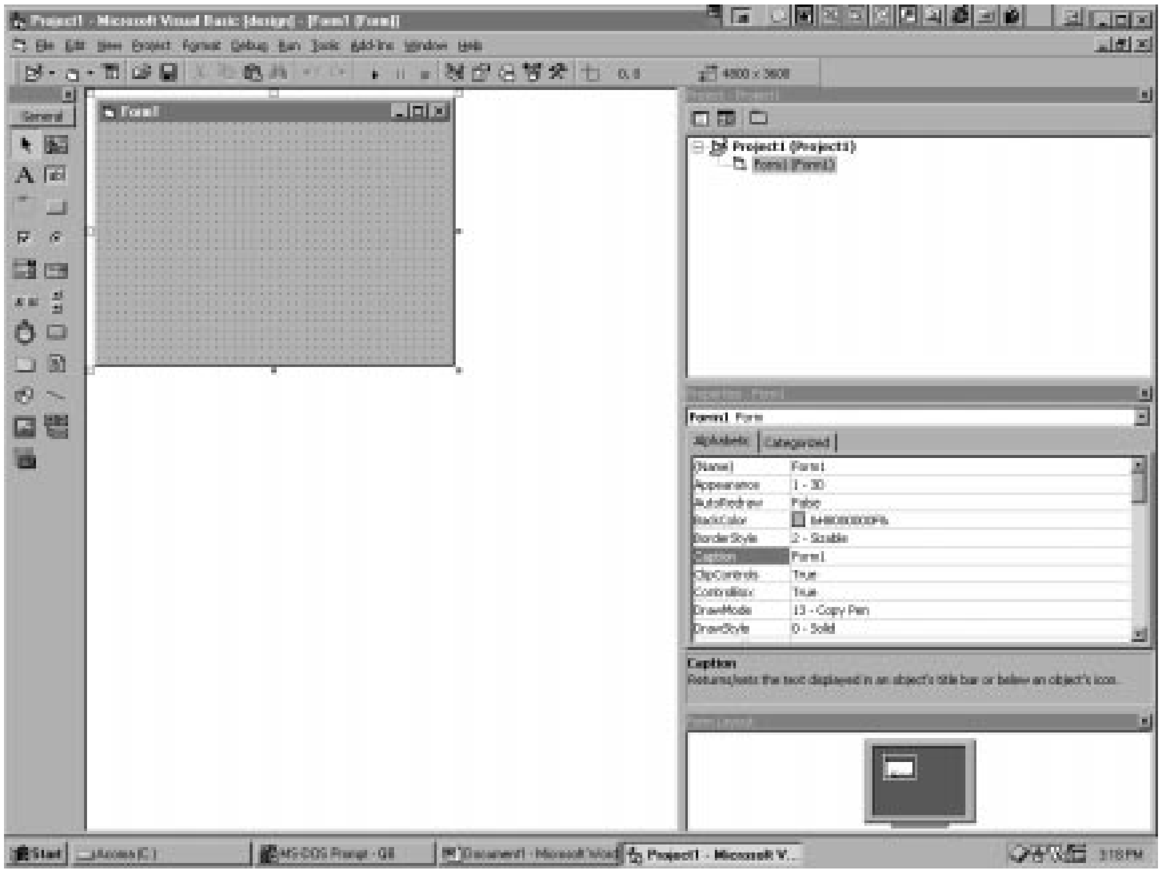
The screen will look like this



There are 9 options here because VB is a very versatile tool. For example, Active Document lets you make stuff for deployment on a web. Never used it. Should try someday. Need more time and better book.

We can just choose Standard EXE, which makes files that run directly on any Windows PC. You will be able to launch your program just like Word, Excel. It will also install the same way—i.e., by writing a lot of stuff no one understands to your PC's registry. A simpler install is one of the big luxuries of old-fashioned programs like Quick Basic that we used previously: just copy it to your drive and it's "installed." Delete it and it's really gone. That doesn't work in Windows because there are so many common features (perhaps a good example would be the "Browse" capability) shared by different programs. The software for such functions is stored in the C:\Windows\System directory. When you install the fruits of your VB labor, you will add some components, if they are not already present.

OK, so double click Standard EXE to get this screen:


Project1 - Microsoft Visual Basic [design] - [Form1 (Form)]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
General
Form1
Project1 (Project1)
Form1 (Form1)
Form1 Form
Alphabetic
Categorized
Start
Project1 - Microsoft V...

(Name)	Form1
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid

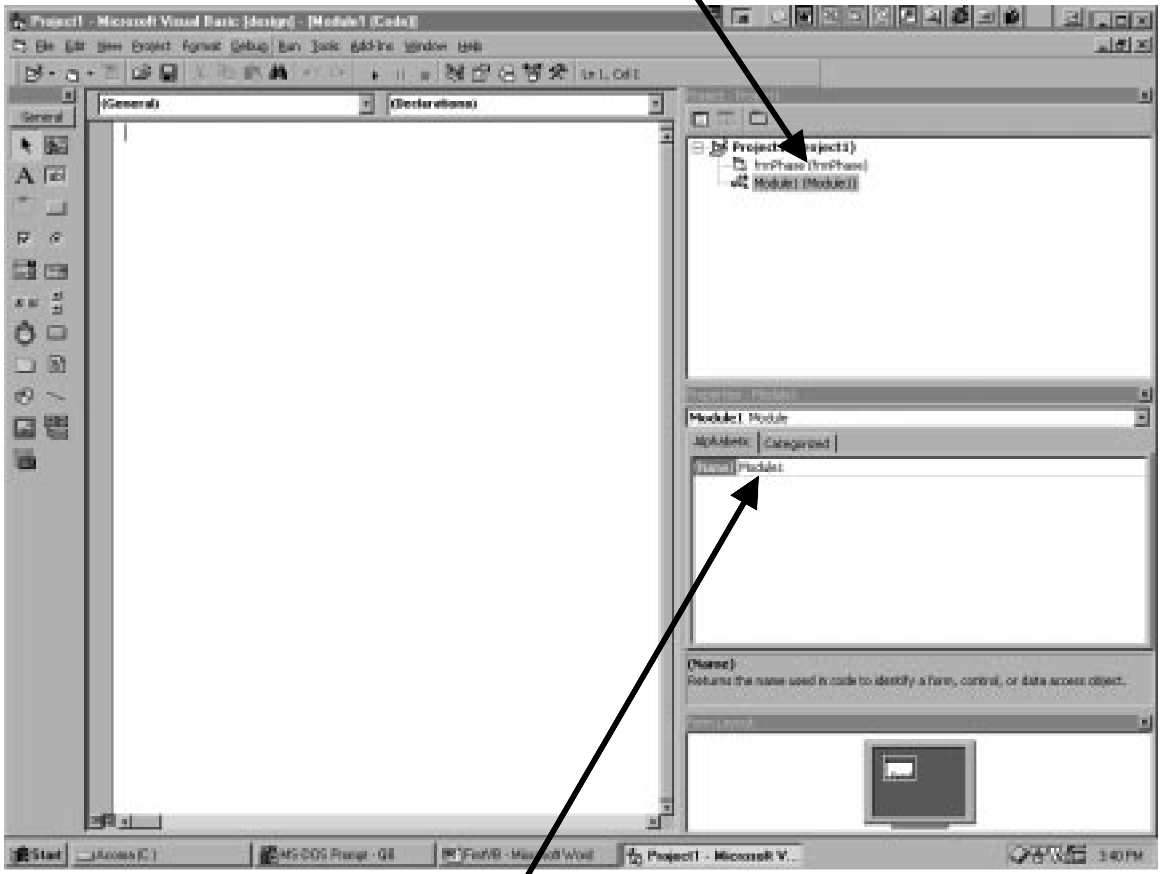
Caption
Returns/sets the text displayed in an object's title bar or below an object's icon.

The most prominent feature is the blank form, Form1. To the left are arranged controls you can drag onto the form. To the upper right is a Project Box, which shows all the components in your program….er….project. At middle right is the Properties Box, which describes all the characteristics of your form. You can organize these alphabetically or by function.

Change its name by typing in the Name Box. I recommend frmPhase. The first three letters indicate that this object is a form. The "Phase" indicates its ultimate function—plotting a phase diagram. Just try goofing with the other properties—e.g., backcolor, which has a drop-down menu that appears when you click on the line. Try changing your backcolor property and see what happens.

The next order of business is to add a module to store source code. I don't know why, but that's what I always do. This code in this module can be called by the controls or by the form itself. It is not the only place such code is stored, as you will see. But it is handy to have a module for "spare code".

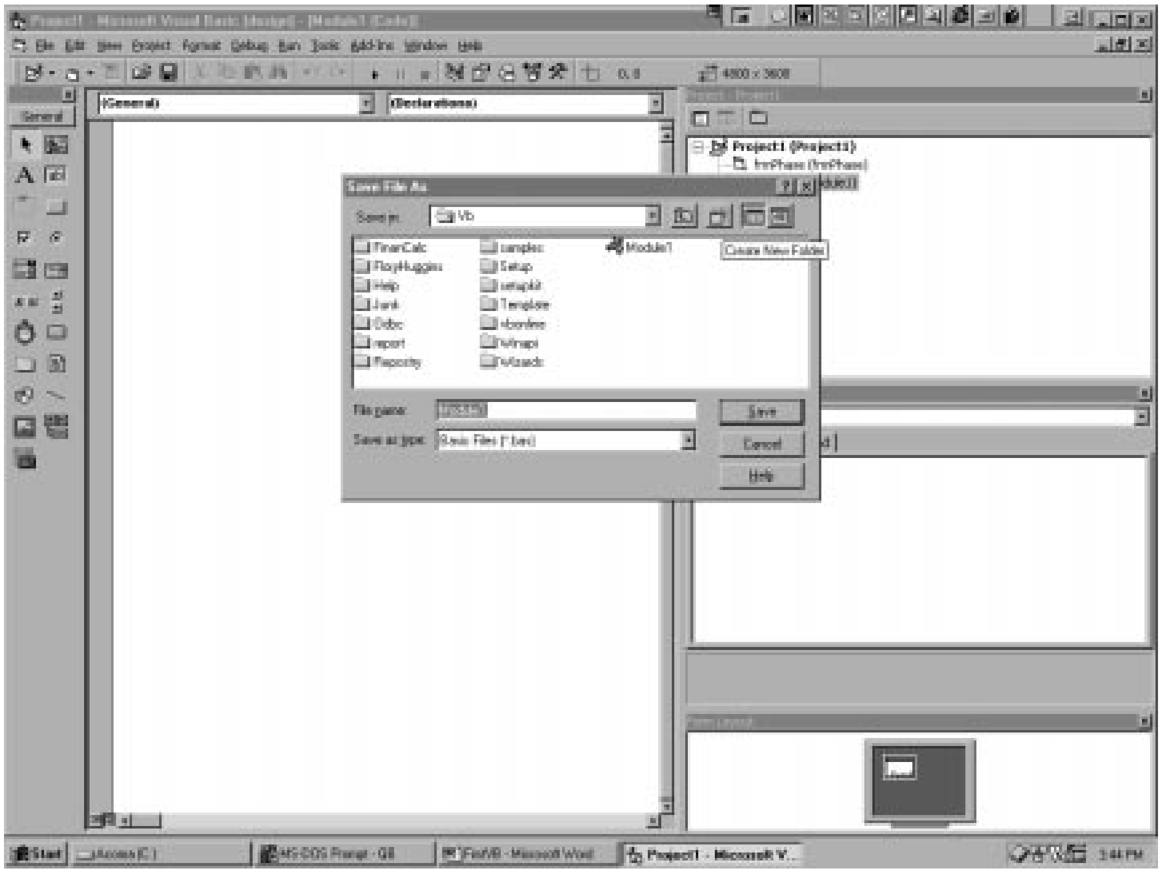
Click Project/Add Module and a new menu appears. Choose "Module" and "Open". See? It added a new module to your Project box:


Project1 - Microsoft Visual Basic [design] - [Module1 (Code)]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
General
(General)
(Declarations)
Project1 (Project1)
frmPhase (frmPhase)
Module1 (Module1)
Module1 Module
Alphabetic
Categorized
(Name) Module1
(Name)
Returns the name used in code to identify a form, control, or data access object.
Start
Project1 - Microsoft V...

You can change its name, if you wish: (I usually don't).

Now is a good time to **save this project**. Saving is confusing in VB.

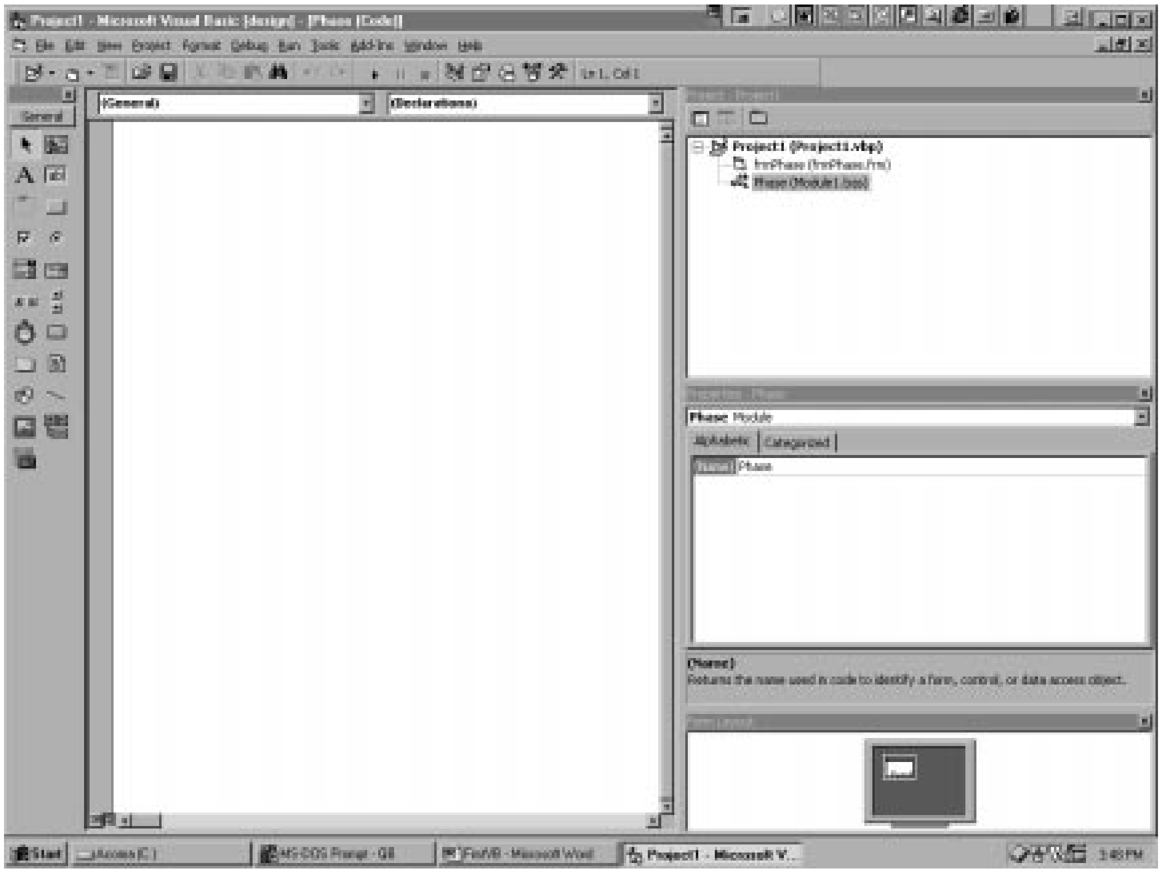
Click File/Save Project and get a window like this:



Note where my mouse pointer is pointing. See the “Create New Folder” icon? Well, click it and make a new folder to store your program, which will consist of several files. I used the name Phase. You have to double click the new folder once you’ve created it in order for stuff to get stored there. Be careful!

From now on, you should **save every five minutes or so** to keep your work secure.

Double Click on Module1 and the screen looks like this:


Project1 - Microsoft Visual Basic [design] - [Phase (Code)]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
(General)
(Declarations)
Project1 (Project1.vbp)
frmPhase (frmPhase.frm)
Phase (Module1.bas)
Phase Module
Alphabetic Categorized
(Name) Phase
(Name)
Returns the name used in code to identify a form, control, or data access object.
Start
Project1 - Microsoft V...

The form is gone! In the left panel, where the form used to be, is space to write code. The first thing to write is two words:

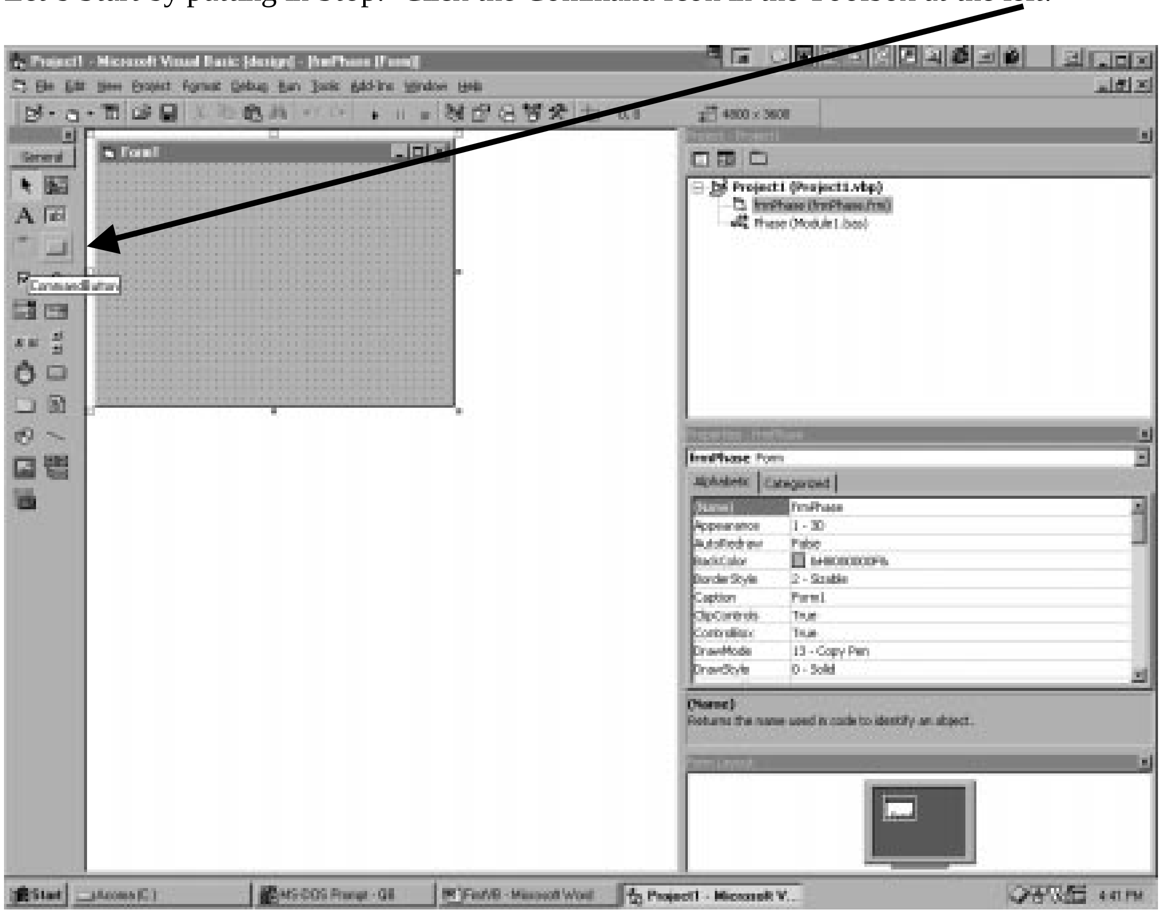
Option Explicit

Notice as you hit "Enter" the color of those words changes. Neat, eh? This is VB's way of telling you that it knows what the words, "Option Explicit", mean. If you had typed something wrong, like Optiun Explicit, the words would NOT have turned blue. As a good VB programmer, you learn to look for these things, which help you avoid typographical errors.

Avoiding typos is one purpose of the words Option Explicit in the first place. This command forces you to define every variable you will ever use. If you are writing a VERY SMALL VB program, you can often avoid this. In most cases, you are MUCH better off putting up with the nuisance of having to define all variables. (Users of C++ or Pascal will be used to this; to users of Qbasic or QB, it will seem horrible at first. But the first time that Option Explicit button saves you about five days of goofing around looking for some dumb error in some huge program, you will be very, very happy you optioned for Option Explicit). Just do it, OK? Trust me.

Now we can design the form. To see the form again, double click the frmPhase icon above Phase module in the Project box.

Let's Start by putting in Stop. Click the Command Icon in the Toolbox at the left.



frmPhase Form

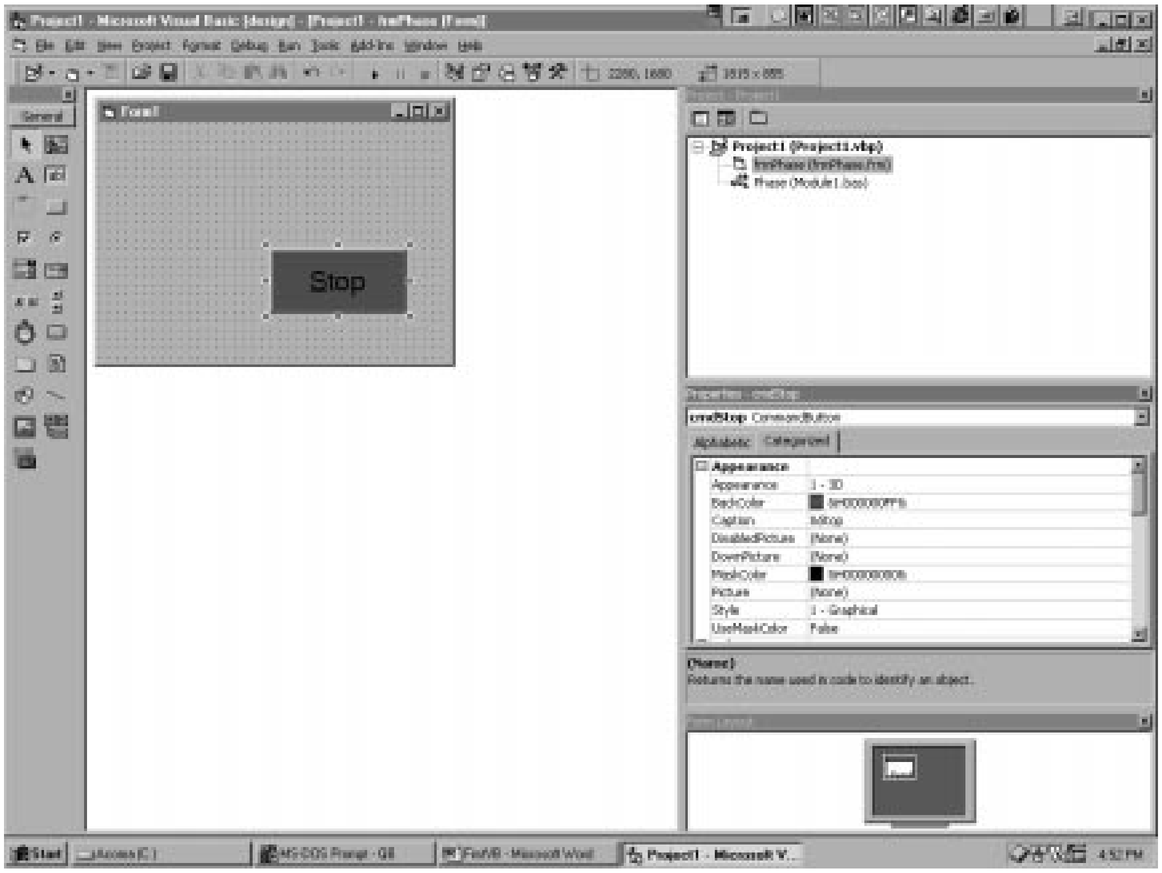
(Name)	frmPhase
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid

(Name)
Returns the name used in code to identify an object.

Draw (by depressing the mouse button and moving the mouse) a command button onto the form. Put it anywhere; you can move it later. Note the words on it: Command1. This is called the Caption. Locate Caption in the Properties Box and change it to &Stop. The ampersand (&) makes it possible to stop by typing Alt-S. Had we written Sto&p, the "hot key" would have been Alt-P. This is a very useful feature: all buttons (and other menus) should have hot keys. Makes it easy for those of us who live out of our Laptops.

Change the name of this button to cmdStop. (cmd = 3-letter shortcut for command)

A Stop sign should be red, don't you think? See if you can fiddle around with the properties until it looks like this. Note that little help text appears at the bottom of the property box as each property is selected. This tells you what that property does.

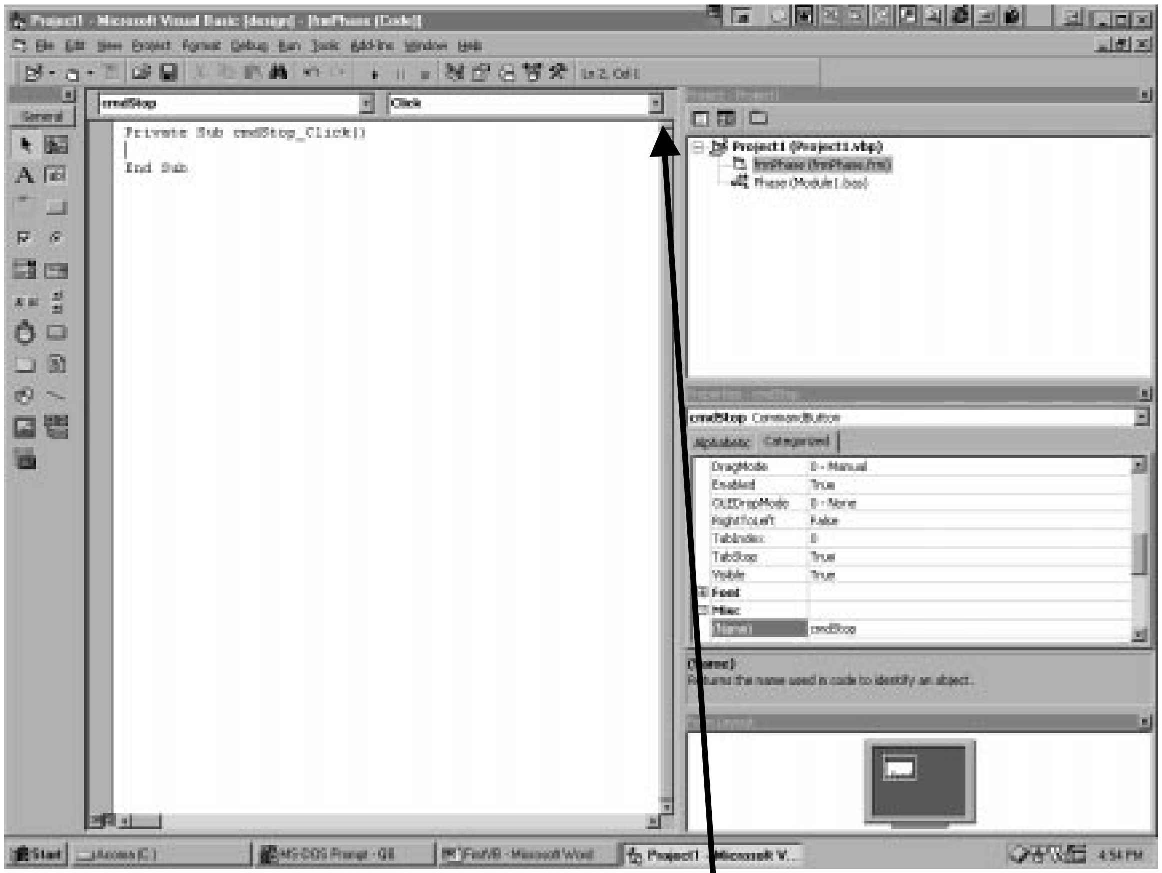


cmdStop CommandButton

Appearance	
Appearance	1 - 3D
BackColor	&H008080FF&
Caption	&Stop
DisabledPicture	(None)
DownPicture	(None)
MaskColor	&H00C0C0C0&
Picture	(None)
Style	1 - Graphical
UseMaskColor	False

(Name)
Returns the name used in code to identify an object.

Double Click the Stop Button to get this:



```
Private Sub cmdStop_Click()

End Sub
```

DragMode	0 - Manual
Enabled	True
OLEDropMode	0 - None
RightToLeft	False
TabIndex	0
TabStop	True
Visible	True
Font	
Misc	
(Name)	cmdStop

(Name)
Returns the name used in code to identify an object.

Aha! Time to write a line of code. We now see a subroutine, not associated with the Phase module we created earlier, but attached instead to the form. The line says Private Sub cmdStop_Click(). Let's dissect this:

Private: this subroutine is not be visible throughout the program. You have to refer to it as frmPhase.cmdStop.

Click(): Click is one of the events associated with the command button. To see all the other events that cause something to happen, click here:

So the deal is: Private Sub cmdStop_Click() is a subroutine that contains the code that will execute if the user clicks on the button. If you wanted to have something happen when the mouse is over the button (e.g., you could change its color then) you would select that event. Click is, by far, the most common event.

So here's the big bit of code we have to type in: end

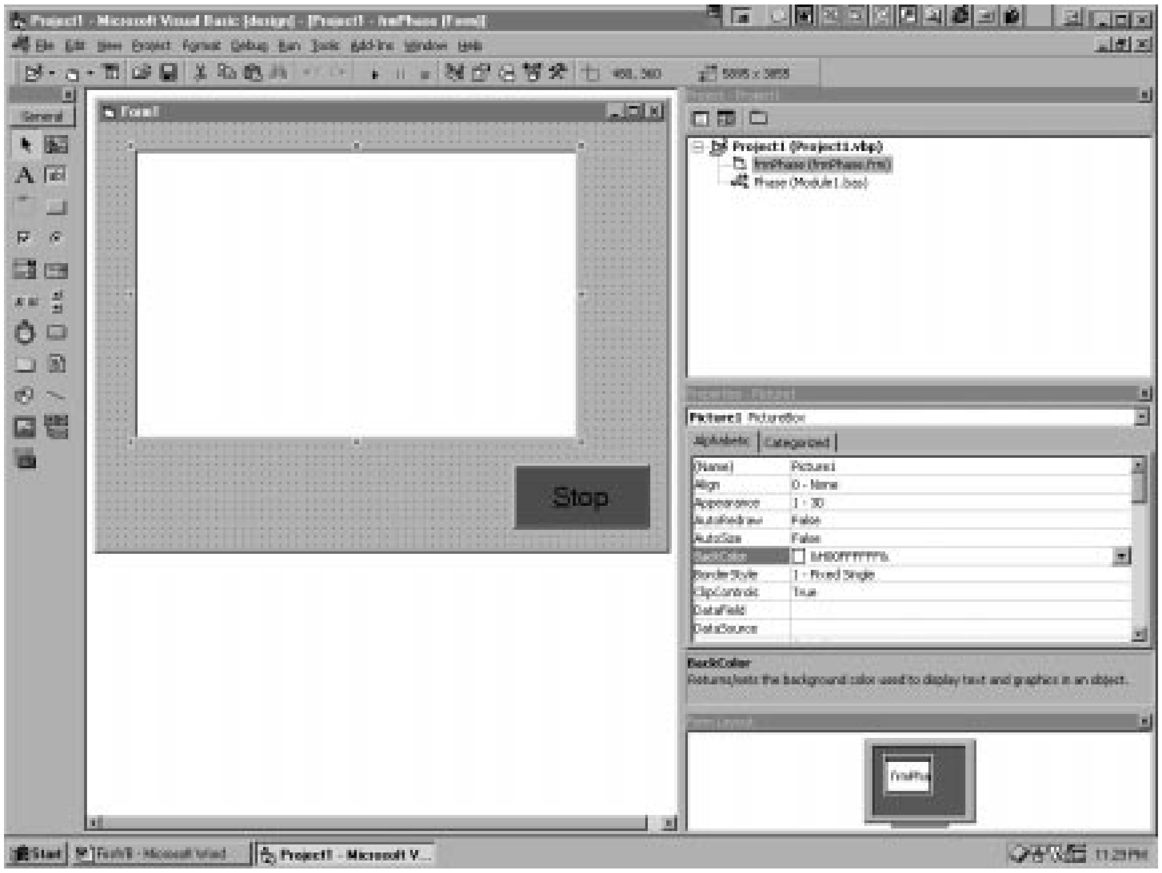
Just type it in, like this: end (note, VB converted it to blue and capitalized the E in End to let us know that End is a word with special meaning to VB, a so-called "reserved word").

```
Private Sub cmdStop_Click()
  End
End Sub
```

To see this "program" run, press function button F5 on your computer (or click the little arrow that looks like a CD player's "Run" button or go to the Run menu). The programming environment fades into the background, and the program runs. Press the Stop button (or Alt-S) to stop.

Next, **expand the form** by dragging its corner and move the Stop Command to the lower right.

Now highlight the Picture Tool (little cactus plant icon) and draw a picture box on the form. Change the background color to white (BackColor in "Properties" box). Change the name of the new picture picPlot. Change the DrawWidth property to 3.


Project1 - Microsoft Visual Basic [design] - [Project1 - frmPhase (Form)]
File Edit View Project Format Debug Run Tools Add-Ins Window Help
General
Form1
Stop
Project1 (Project1.vbp)
frmPhase (frmPhase.frm)
Phase (Module1.bas)
Alphabetic
Categorized
Start
Project1 - Microsoft V...

Picture1 PictureBox

(Name)	Picture1
Align	0 - None
Appearance	1 - 3D
AutoRedraw	False
AutoSize	False
BackColor	&H80FFFFFF&
BorderStyle	1 - Fixed Single
ClipControls	True
DataField	
DataSource	

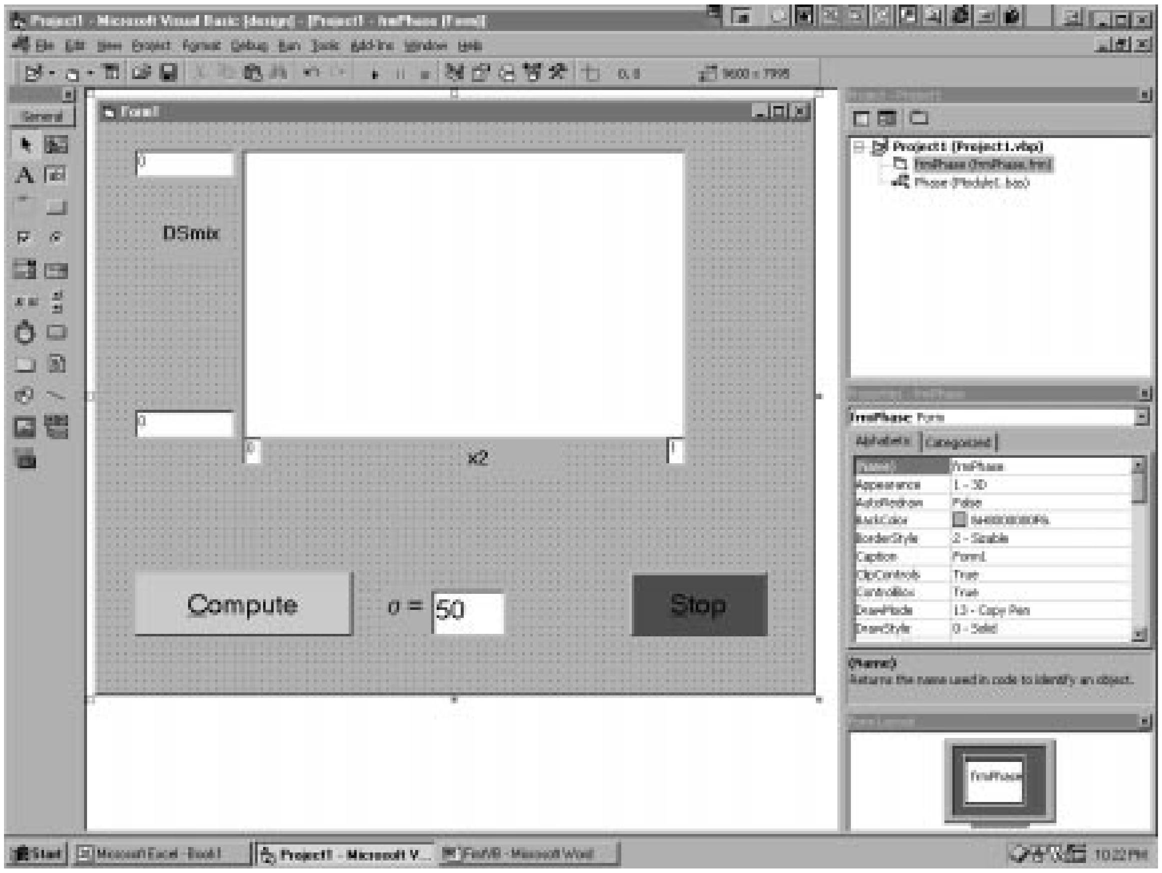
BackColor
Returns/sets the background color used to display text and graphics in an object.

Add a new button, and name it cmdCompute. Caption it &Compute.

Add a text box (the tool that says ab|) and name it txtSigma. Set its "text" property to 100 and boost the font size to match the others (18 point).

Choose the Label tool (it has the letter A) and make a label that says "Sigma = " or, better yet, "σ = " for its caption. Change its name to lblSigma.

Now add two text boxes, named txtXmin and txtXmax, with initial values 0 and 1. **Add a label**, called lblXaxis, with caption phi2. **Do similar things for the y-axis** to make the form look like this:

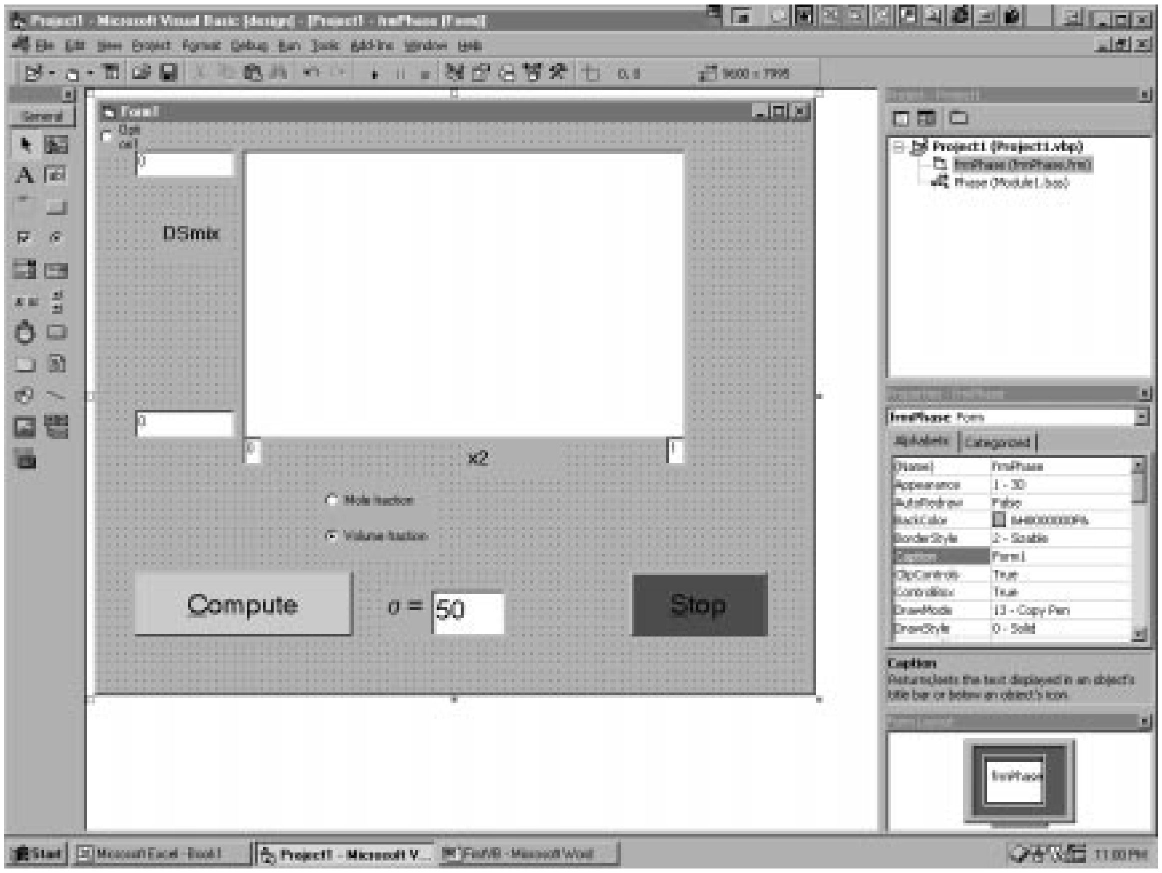


frmPhase Form

(Name)	frmPhase
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid

(Name)
Returns the name used in code to identify an object.

We want the plot's abscissa to be able to switch from mole fraction to volume fraction. Radio buttons can be useful for this. **Click the radio button icon** (next to the checkmark) and draw a radio button on the form. Make its name optXchoice and its caption Mole Fraction. **Click Ctrl-C** to copy it and **Ctrl-V** to paste it. You will be asked if you want to make a control array. In this case, **say yes**. Move the pasted button (near the upper left) close to the first one. Change its caption to Volume Fraction. Set its value property to True. The form will look like this:



frmPhase Form

(Name)	frmPhase
Appearance	1 - 3D
AutoRedraw	False
BackColor	&H8000000F&
BorderStyle	2 - Sizable
Caption	Form1
ClipControls	True
ControlBox	True
DrawMode	13 - Copy Pen
DrawStyle	0 - Solid

Caption
Returns/sets the text displayed in an object's title bar or below an object's icon.

Let's program this last element, **the radio buttons**, first because it's fun. Double click either the "Volume fraction" or "Mole fraction" buttons. As with the Stop button, this will open up a new subroutine in the form. Add the code below until it looks like this:

```
Private Sub optXChoice_Click(Index As Integer)
  If Index = 0 Then lblXaxis.Caption = "x2"
  If Index = 1 Then lblXaxis.Caption = "phi2"
  frmPhase.picPlot.Cls
End Sub
```

VB knows the Radio Button needs an argument-- "Index as Integer"-- while the Stop command did not. We can use the index to set the label on the x-axis. This accounts for the first two lines of code "If Index = 0…" **Run the program** by hitting F5 or clicking the arrow that looks like the one on any CD player. Change the radio buttons and see what happens! (The third line clears the plot: after there *is* a plot, we don't want any leftover data being left on it if we change the axis).

Now Put in the Code! This will be done in two parts—code for Module1 and code for frmPhase.

Module1 Code. Double click the Phase (Module1.bas) line in the Project Box. This will open the page that holds the code for Module1. Make it look like this. The best thing is to TYPE IT ALL IN. Won't take that long, and you'll learn a lot about the VB programming environment.

```
Option Explicit

Public n1 'number of solvent molecules
Public n2 'number of polymer molecules
Public sigma  'degree of polymerization
Public no 'no = n1 + sigma*n2
Public R 'gas constant

Public x2(200000) 'polymer mole fraction array (0.01 to 0.99)
Public phi2(200000) 'volume fraction array (

Public DSmix(200000) 'entropy of mixing array
Public Npts 'number of data points generated
'
'




Sub DoThePlot()
'MsgBox "Entered DoThePlot OK."

  Dim i 'loop index
  Dim Ymin, Ymax 'hold smallest & largest values of Y




  Ymin = 10000000000#  'set a ridiculously high minimum
  Ymax = -10000000000#  'set a ridiculously low maximum
  For i = 1 To Npts
    If DSmix(i) > Ymax Then Ymax = DSmix(i)  'replace with better Ymax
    If DSmix(i) < Ymin Then Ymin = DSmix(i)  'replace with better Ymin

  Next i
  Ymin = Ymin - 0.05 * Abs(Ymax - Ymin) 'provide a buffer zone
  Ymax = Ymax + 0.05 * Abs(Ymax - Ymin)
  'MsgBox "Ymin:  " & Ymin & "  Ymax:  " & Ymax
  frmPhase.picPlot.Cls  'clear any old plots off the form

  frmPhase.picPlot.Scale (0, Ymax)-(1, Ymin)
  frmPhase.txtYmin.Text = Ymin
  frmPhase.txtYmax.Text = Ymax
  DoEvents  'this forces VB to do stuff it is holding in its "to do" list.
            'add this when things seem to be happening too slowly.

  If frmPhase.lblXaxis.Caption = "x2" Then
  For i = 1 To Npts
    frmPhase.picPlot.PSet (x2(i), DSmix(i))
  Next i

  ElseIf frmPhase.lblXaxis.Caption = "phi2" Then
  For i = 1 To Npts
   frmPhase.picPlot.PSet (phi2(i), DSmix(i))
  Next i
  End If









End Sub
```

frmPhase Code. Make it look like the following. There are two choices: type it all in, or just click the objects on the form and type in just the stuff that isn't Sub Something….End. The following is more realistic—more like how you actually program.

```
Private Sub cmdStop_Click()

 End

End Sub

Private Sub cmdCompute_Click()
'the general strategy is to let n1=1000 (constant), then increment
'n2 until we reach a maximum x2 of 0.99


 Dim i 'loop index
 Dim x2max 'the largest x2 generated
 sigma = txtSigma.Text  'set the variable sigma to the contents of the textbox
 n1 = 10000
 R = 8.314
 While x2max < 0.99
   i = i + 1
   n2 = 10 * i
   x2(i) = n2 / (n2 + n1)
   x2max = x2(i)
   phi2(i) = (sigma * n2) / (sigma * n2 + n1)
   DSmix(i) = R * (n1 * Log(1 - phi2(i)) + n2 * Log(phi2(i))) / (n1 + sigma * n2)
 Wend
 Npts = i
 'MsgBox "Number of points:  " & Npts

 DoThePlot


 End Sub



Private Sub Form_Load()

End Sub

Private Sub optXChoice_Click(Index As Integer)
  If Index = 0 Then lblXaxis.Caption = "x2"
  If Index = 1 Then lblXaxis.Caption = "phi2"
  frmPhase.picPlot.Cls


End Sub

Private Sub ProgressBar1_MouseDown(Button As Integer, Shift As Integer, x As Single,
y As Single)

End Sub

Private Sub ProgressBar1_OLEGiveFeedback(Effect As Long, DefaultCursors As
Boolean)

End Sub
```