

PROGRAMMING IN VISUAL BASIC (Envelop)

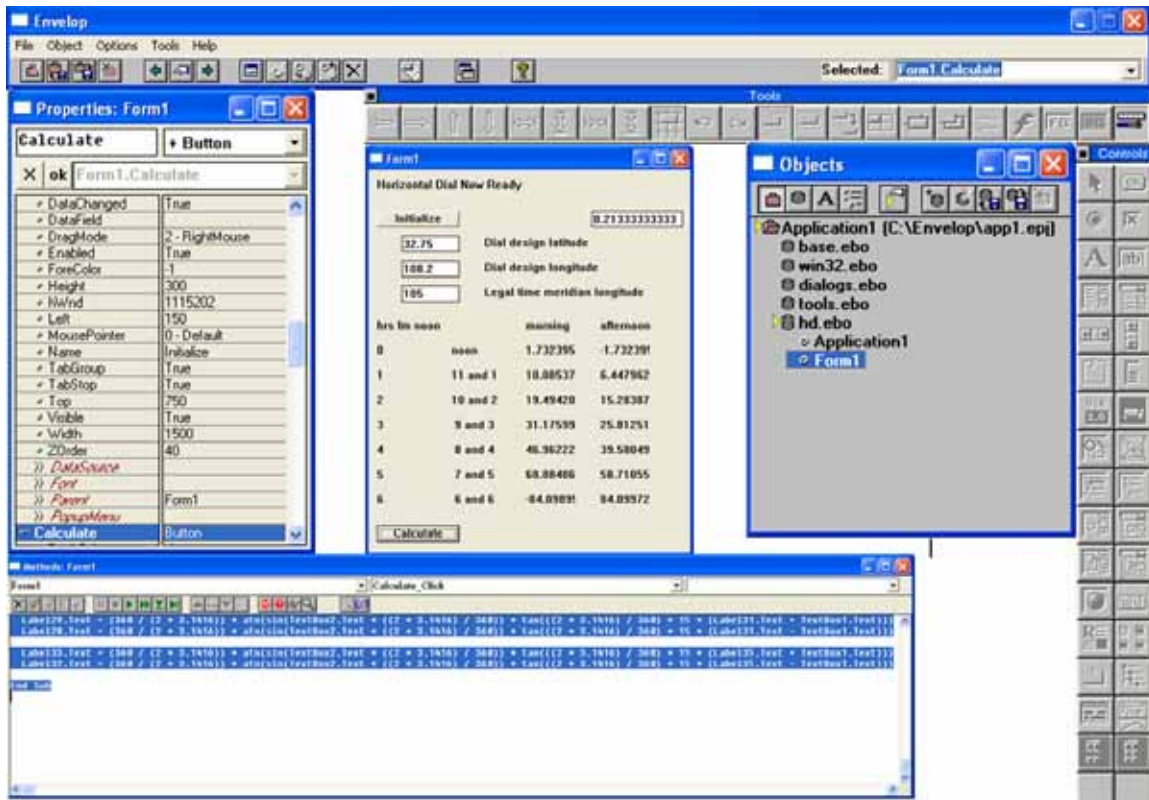
A free version of Visual Basic is available from:-

<http://www.freebyte.com/programming/compilers/envelop.html>

this is an excellent albeit unsupported product, downloaded as an INSTALL file and six parts, totaling just over 7mb. Once downloaded, each zip file is expanded into a single common folder, and the SETUP program run. Designed for Windows 95 or later, this runs on Windows XP service packs 1 and 2, Saving project source files is explained in HELP. The FILE/SAVE PROJECT is used to save a program suite. More importantly, to restore them, the OBJECTS form for the application must be clicked, as well as its subsequent FORM and APPLICATION entries.

Visual Basic is object oriented, thus the "screen" or form is designed first, fixed data entered next, and finally the code (methods) is entered for each button (object). When a button (object) is clicked, then its program (method) is invoked. It is thus event driven. The following example is more event driven than truly object oriented although the distinction is somewhat arbitrary.

The actual computer desktop area looks something like the below.



The Microsoft Visual Basic link is:

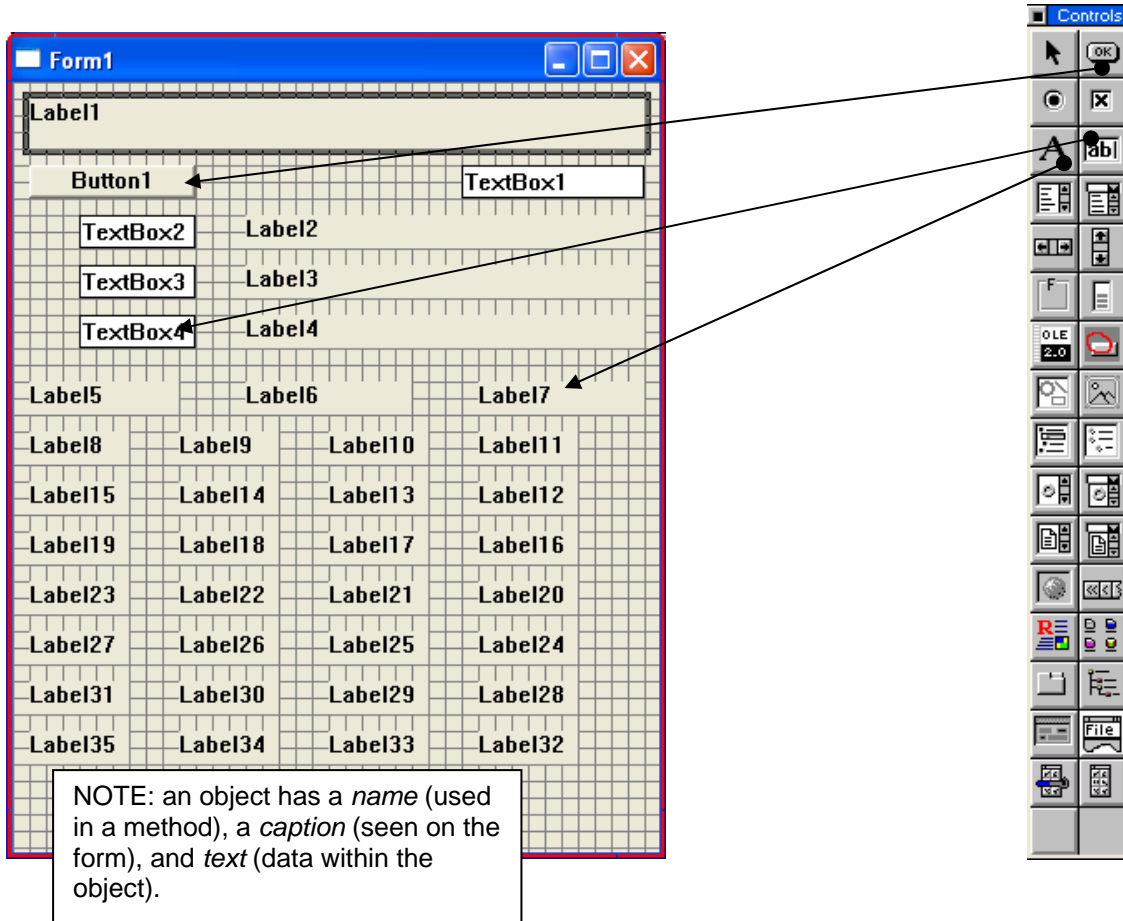
<http://msdn2.microsoft.com/en-us/vbasic/default.aspx>

Microsoft's Visual Basic 2005 Studio Express for Windows XP service pack 2 is about 60mb after an initial 3mb download and is downloadable for free from their website:

<http://msdn.microsoft.com/vstudio/express/downloads/default.aspx>

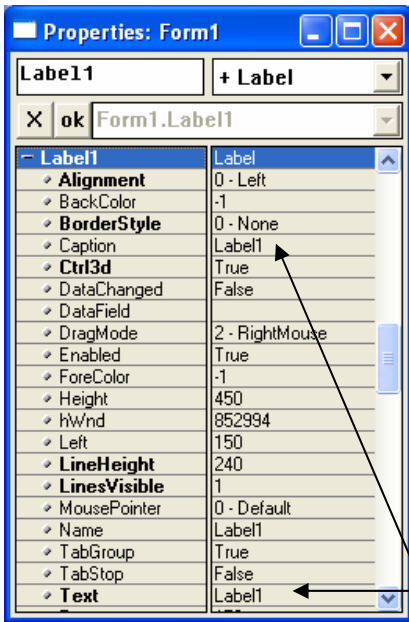
First, a form was established with some "labels" which hold resultant data, some "text boxes" to hold user variable data, and a couple of clickable buttons. This was done by selecting the appropriate tool from the "controls" menu and placing those tools on the form.

The program was developed in a matter of minutes, and some small clarifying changes made as work proceeded. However, the following screens are very close to the product that was designed and which is on the CD associated with Illustrating Shadows.



As you develop the "form", some logical sequence of adding objects will simplify their names, names which will be used in methods. If you enter data into the caption field, the form has that caption displayed and not the name of the object itself, and it is the name that is needed in the methods (programming associated with that object).

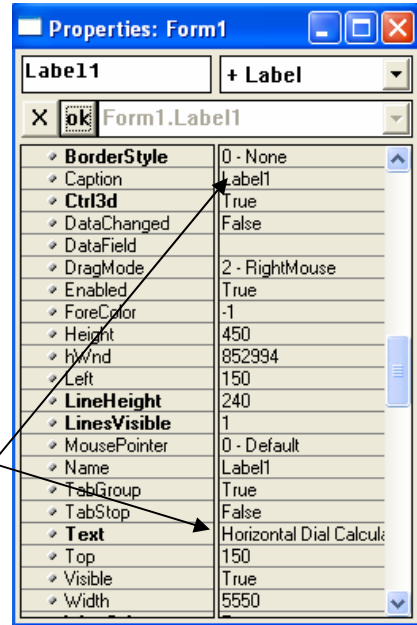
However, if you forget an object's name, then while building a method, one can click on an object in the form and then identify its label. Then one goes back to the method editor and continues.



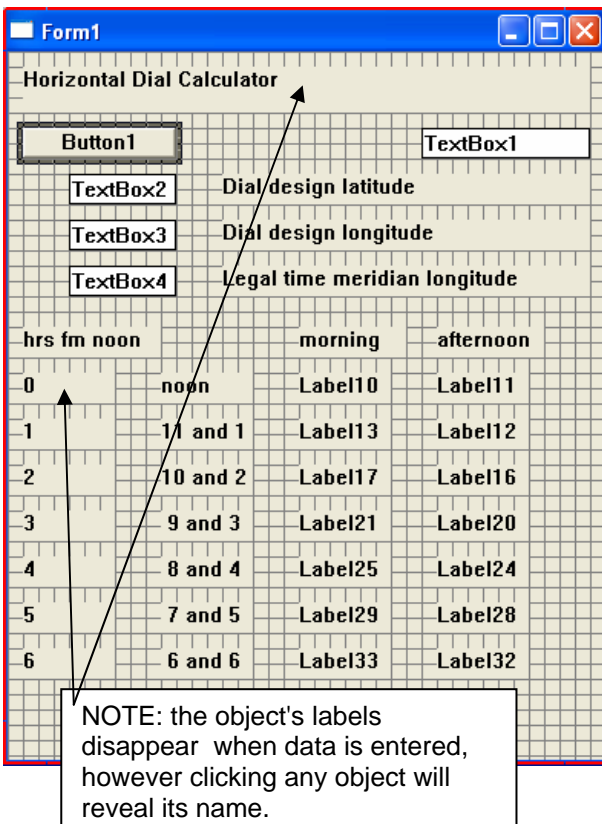
At this point, a form has been generated. In this design phase, some data may be entered by the designer, and the rest entered or calculated, by the "click" "move" action of a button.

First, "Label1.Text" was changed to "Horizontal Dial Calculator".

The caption can be changed instead, and that changes the text.



Then, other fixed fields were tailored in the same way. When this is done, the labels "Label1" etc are deleted, and replaced by the entered text. Since some of those labels will be used in formulae references, such as the left column of "hrs fm noon", it is important to document the label names. Should this not be done, then the label can still be found by clicking the object in the form display.



At this point, fixed data is entered, what is needed now is the actions to be taken when a button is clicked. The button is an "object", and when clicked, it invokes a "method".

In this case, the objects had a "move" and "click" associated with them.

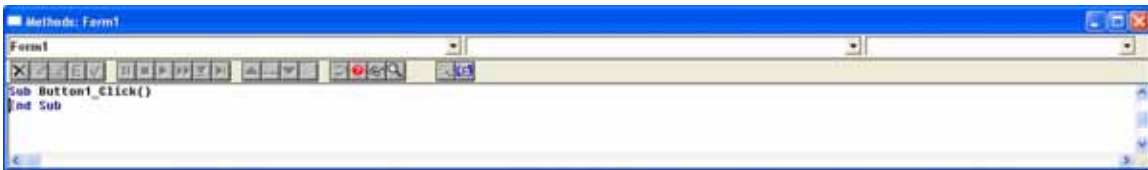
NOTE: Should you go back and re-edit the form, you may find that some of the text in the label fields may be missing. This can be mitigated by having the "initialize" button set field and label text defaults.

First, Button1 was programmed. The Button1 was selected and this then made active the method editor.



select CLICK in this box...

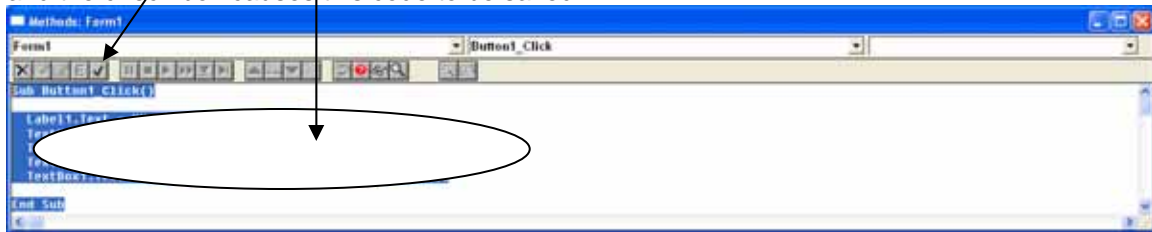
...which clears the two right hand boxes and generates some skeleton code.



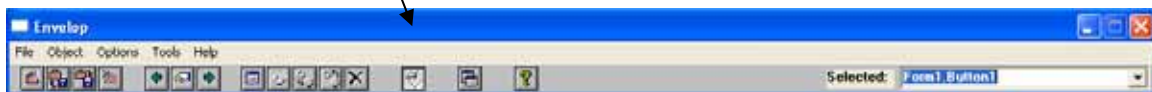
The following code is typed in to the skeleton code. This code ensures default text in Label fields in case the Envelop compiler loses the default data.

```
Sub Initialize_Click()  
    Label1.Text = "Horizontal Dial Now Ready"  
    TextBox2.Text = "32.75"  
    TextBox3.Text = "108.2"  
    TextBox4.Text = "105"  
    TextBox1.Text = (TextBox3.Text - TextBox4.Text) * 4 / 60  
    Label8.Text = 0  
    Label15.Text = 1  
    Label19.Text = 2  
    Label23.Text = 3  
    Label27.Text = 4  
    Label31.Text = 5  
    Label35.Text = 6  
    Label5.Text = "hrs of noon"  
    Label6.Text = "morning"  
    Label7.Text = "afternoon"  
    Label2.Text = "Design latitude"  
    Label3.Text = "Design longitude"  
    Label4.Text = "Legal time meridian's longitude"  
End Sub
```

and the check box causes this code to be saved.



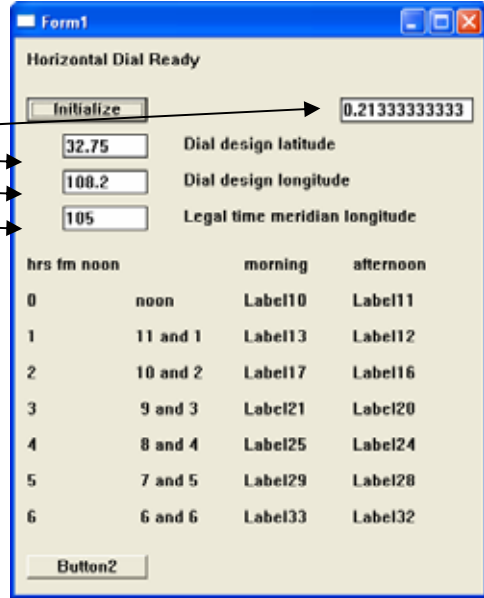
Then, click the button on the main editor to go from edit to run mode.



This tests the button's event driven method. In other words, it runs the code.

The result was that the header title was changed, and some default values generated.

The resulting field is a TextBox as opposed to a more appropriate Label object. This was only to make object numbering easier.



While at it, the word "Button1" was changed to the word "Initialize" so that this button would be more meaningful. Visual Basic is case sensitive, so when writing the event driven code, "TextBox" is not the same as "textbox" nor "Textbox".

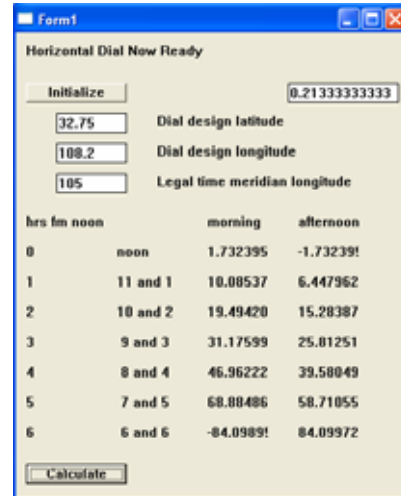
At this point, the fixed data has been built into the objects in the form, now the "Calculate" code must be built. This will use Button2, so that was renamed to Calculate. The object's CAPTION appears on the form, the object's NAME is what is used in program references. These are massaged by first clicking the object on the form, which in turn located that object in the PROPERTIES panel, and from there things can be changed. The final step is to click on the "Calculate" button so that the Methods Editor is highlighted, and the code for the hour lines then coded. The Method Editor "move" and "click" is used to identify the correct code for a button click for this object.

```
Sub Calculate_Click()
    TextBox1.Text = (TextBox3.Text - TextBox4.Text) * 4 / 60

    Label10.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label8.Text + TextBox1.Text)))
    Label11.Text = -Label10.Text
    Label13.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label15.Text + TextBox1.Text)))
    Label12.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label15.Text - TextBox1.Text)))
    Label17.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label19.Text + TextBox1.Text)))
    Label16.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label19.Text - TextBox1.Text)))
    Label21.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label23.Text + TextBox1.Text)))
    Label20.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label23.Text - TextBox1.Text)))
    Label25.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label27.Text + TextBox1.Text)))
    Label24.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label27.Text - TextBox1.Text)))
    Label29.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label31.Text + TextBox1.Text)))
    Label28.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label31.Text - TextBox1.Text)))
    Label33.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label35.Text + TextBox1.Text)))
    Label32.Text = (360 / (2 * 3.1416)) * atn(sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
tan(((2 * 3.1416) / 360) * 15 * (Label35.Text - TextBox1.Text)))
    Label5.Text = "hrs of noon"
    Label6.Text = "morning"
    Label7.Text = "afternoon"
    Label2.Text = "Design latitude"
    Label3.Text = "Design longitude"
    Label4.Text = "Legal time meridian's longitude"
End Sub
```

The code is inserted in the method editor, and the system switched from edit to execute mode and thus tested.

The end result →



The developed system can be saved. Saving project source files are not well explained in HELP. The FILE/SAVE PROJECT as well as the prompted FILE/SAVE MODULE are used to save a program. More importantly, to restore them, the OBJECTS "form" for the application must be clicked, as well as its subsequent FORM and APPLICATION entries.

The end result is three files.

- .obj The project file, small, a sort of coordinating file.
- .ebo The objects file mostly, relates to the form and application.
- .exe The executable program.

While the .exe file can be executed as is, it uses .dll files that only exist if Envelop's Visual Basic are installed. So, exporting these Visual Basic programs means the end user also installing Envelop. An alternative is to click on START, PROGRAMS, ENVELOP, and select the Application Install Wizard. This should generate a fully executable program.



However, even this may not install all of the required .dll files.

Never the less, the Envelop Visual Basic system is a good package, easy to use, and complete with an extensive help system to facilitate a programmer new to object oriented techniques, and in an IDE (integrated development environment).

The Envelop implementation of Visual Basic runs on Windows XP both service pack's 1 and 2.

As will be seen, the Visual Basic IDEs are a good introduction to the JAVA NetBeans IDE.

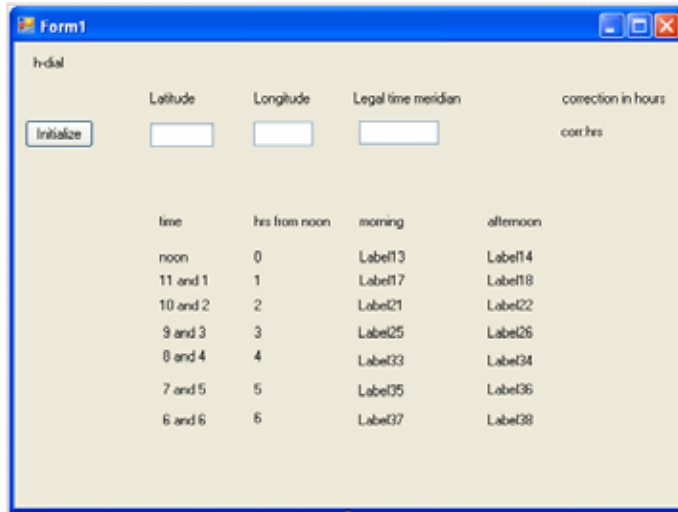
PROGRAMMING IN VISUAL BASIC EXPRESS (Microsoft)

Visual Basic Express is downloaded, first as a 3mb installer, then as 60mb of code, if the installer likes your system. And XP SP1 is not liked.

This is from Microsoft, and the registration process is cumbersome.

While the tool bars can get in the way of your work, the process is similar to the Envelop Visual Basic system.

The coding for the button clicks is similar but notice that a function's library name is used in the function calls, e.g.:- *System.Math.Tan* where "System.Math." is the library holding the "Tan" function.



The url for the Visual Basic Express, and other light weight Express products is:-

<http://msdn.microsoft.com/vstudio/express/downloads/default.aspx>

And Microsoft's main Visual Basic web page is:-

<http://msdn2.microsoft.com/en-us/vbasic/default.aspx>

Here is some Visual Basic Express code for the horizontal dial.

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Label11.Text = "horizontal dial ready"
        TextBox1.Text = 32.73
        TextBox2.Text = 108.2
        TextBox3.Text = 105.0
        Label16.Text = 4 * (TextBox2.Text - TextBox3.Text) / 60
        Button2.Text = "Calculate"
    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        Label11.Text = "horizontal dial data ready"
        Label16.Text = 4 * (TextBox2.Text - TextBox3.Text) / 60
        Label113.Text = (360 / (2 * 3.1416)) *
System.Math.Atan(System.Math.Sin(TextBox1.Text * ((2 * 3.1416) / 360)) *
System.Math.Tan(((2 * 3.1416) / 360) * 15 * (Label112.Text + Label16.Text)))
        Label114.Text = -Label113.Text
        Label117.Text = (360 / (2 * 3.1416)) *
System.Math.Atan(System.Math.Sin(TextBox1.Text * ((2 * 3.1416) / 360)) *
System.Math.Tan(((2 * 3.1416) / 360) * 15 * (1 + Label16.Text)))
        Label118.Text = (360 / (2 * 3.1416)) *
System.Math.Atan(System.Math.Sin(TextBox1.Text * ((2 * 3.1416) / 360)) *
System.Math.Tan(((2 * 3.1416) / 360) * 15 * (1 - Label16.Text)))

        . . . . .

    End Sub
End Class
```

horizontal dial data ready

	Latitude	Longitude	Legal time meridian	correction in hours
Initialize	32.73	108.2	105	0.2133333333333334

Calculate

time	hrs from noon	morning	afternoon
noon	0	1.73145631609394	-1.73145631609394
11 and 1	1	10.0800111577901	6.44449280114107
10 and 2	2	19.4844173114913	15.2759642004791
9 and 3	3	31.1622155738264	25.8003232421829
8 and 4	4	46.9467128660803	39.5652248590156
7 and 5	5	68.8744101928061	58.6967507263195
6 and 6	6	-84.0957721434881	84.0965448085501

The final program can be published, and of course it asks where to. Selecting a CD does not eliminate the fact that when the program is installed elsewhere, it still needs to download all sorts of Microsoft run time facilities.

PROGRAMMING IN VISUAL BASIC .NET 2003 Learning Edition (Microsoft)

Visual Basic .net 2003 is available for less than \$100 including shipping. It arrives with a full book and a number of CDs. The installation process, while taking a lot of time, is simple. The system worked first time, and was easy to use. The Visual Basic Express code was ported pretty much as-is with the labels changed due to different labeling sequences.

However, the program was enhanced with a graphical depiction of the hour lines, this was simple trigonometry. The "hour label" was moved from a set of labels, and what was left was identified as non displayed hours. The code for moving the labels was not elegant since it was a series of relocations for each hour, as opposed to a loop. This was because no easy method was found to iterate through a set of labels.

Additionally, buttons were left blank until enabled. The following is the code that runs when the INITIALIZE button is clicked.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click

    REM *** initialization stuff ***

    Label1.Text = "horizontal dial setup"

    TextBox2.Text = 32.75
    TextBox3.Text = 108.2
    TextBox4.Text = 105

    Label3.Text = 4 * (TextBox3.Text - TextBox4.Text) / 60

    Button2.Text = "CALCULATE"

End Sub
```

The following is the code that runs when the CALCULATE button is clicked.

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click

    REM *** display hour line data ***
    Label1.Text = "Horizontal Dial figures ready"

    Label3.Text = 4 * (TextBox3.Text - TextBox4.Text) / 60

    Button2.Text = "calculated"

    Label10.Text = (360 / (2 * 3.1416)) *
System.Math.Atan(System.Math.Sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
System.Math.Tan(((2 * 3.1416) / 360) * 15 * (0 + Label3.Text)))
    Label11.Text = -Label10.Text

    Label14.Text = (360 / (2 * 3.1416)) *
System.Math.Atan(System.Math.Sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
System.Math.Tan(((2 * 3.1416) / 360) * 15 * (1 + Label3.Text)))
    Label15.Text = (360 / (2 * 3.1416)) *
System.Math.Atan(System.Math.Sin(TextBox2.Text * ((2 * 3.1416) / 360)) *
System.Math.Tan(((2 * 3.1416) / 360) * 15 * (1 - Label3.Text)))

    . . . . .

    Button3.Text = "DRAW"

End Sub
```

The following is the code that runs when the DRAW button is clicked.

```

Private Sub draw_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click

    REM *** DRAW ***

    Dim darea As System.Drawing.Graphics
    darea = Me.CreateGraphics

    Dim pcolor As New System.Drawing.Pen(System.Drawing.Color.Red)
    Dim pcolorb As New System.Drawing.Pen(System.Drawing.Color.Blue)
    Dim pcolorg As New System.Drawing.Pen(System.Drawing.Color.Green)

    REM *** Define top left right bottom of the drawable area
    Dim lx, rx, xtox, xhalf As Integer
    Dim xc As Integer

    Dim ty, by, ytoy As Integer

    REM *** set drawing area coordinates - X is 300+300 wide - Y is 300
    lx = 10
    xtox = 600
    xhalf = xtox / 2
    rx = lx + xtox
    xc = (lx + rx) / 2

    REM *** thus we have two half boxes each is 300 by 300
    ty = 350
    ytoy = 300
    by = ty + ytoy

    REM *** draw a boundary area and two 45 degree lines
    darea.DrawRectangle(pcolor, lx, ty, xtox, ytoy)
    darea.DrawLine(pcolor, xc, by, xc + xhalf, by - ytoy)
    darea.DrawLine(pcolor, xc, by, xc - xhalf, by - ytoy)
    darea.DrawLine(pcolor, xc, by, xc + 0, by - ytoy)

    REM *** now draw hour lines
    Dim i As Integer
    Dim ii As Short

    For i = -6 To +6 Step 1
        Dim ang As Short
        Dim xxx, yyy As Short
        ii = i

        REM *** derive the hour line angle
        ang = (360 / (2 * 3.1416)) * System.Math.Atan(System.Math.Sin(TextBox2.Text *
((2 * 3.1416) / 360)) * System.Math.Tan(((2 * 3.1416) / 360) * 15 * (ii - Label3.Text)))

        REM *** work the coordinates - this is regardless of am or pm
        If System.Math.Abs(ang) > 45 Then
            xxx = xtox / 2
            yyy = xxx / System.Math.Tan(((2 * 3.1416) / 360) * ang)
        End If
        If System.Math.Abs(ang) < 45 Then
            yyy = -1 * (xtox / 2)
            xxx = yyy * System.Math.Tan(((2 * 3.1416) / 360) * ang)
        End If
        If ang = 45 Then
            xxx = xtox / 2
            yyy = ytoy
        End If

        REM *** do scaling for whatever reason
        xxx = 0.89 * xxx
        yyy = 0.89 * yyy
        REM that is poor coding, the 0.89 should be in a constant, not entered twice

        REM *** draw the lines
        REM if lines below the border we drop them
        If i <= 0 Then
            REM *** morning hours
            If (by + yyy) <= by Then
                REM This line is above the border
                darea.DrawLine(pcolorb, xc, by, xc - xxx, by + yyy)
            End If
        Else
            REM *** afternoon hours
    
```

```

If System.Math.Abs(ang) > 45 Then
    REM not elegant but works
    xxx = -1 * xxx
    yyy = -1 * yyy
End If
If (by + yyy) <= by Then
    REM This line is above the border
    darea.DrawLine(pcolg, xc, by, xc - xxx, by + yyy)
End If
End If
End If

REM *** at this point, ( xc - xxx, by + yyy ) are the line end points
If (by + yyy) <= by Then
    REM *** this was a line within the boxed border
    REM *** this code is not very elegant but it works
    If i = -6 Then
        Label40.Location = New Point(xc - xxx, by + yyy)
    End If
    If i = -5 Then
        Label41.Location = New Point(xc - xxx, by + yyy)
    End If
    If i = -4 Then
        Label42.Location = New Point(xc - xxx, by + yyy)
    End If
    If i = -3 Then
        Label43.Location = New Point(xc - xxx, by + yyy)
    End If
    . . . . .
    End If
    If i = 6 Then
        Label52.Location = New Point(xc - xxx, by + yyy)
    End If
End If
Next

Label2.Text = "hours NOT depicted"

End Sub

```

To the right is a depiction of the tabular as well as graphical dial display.

This code is not elegant in that iteration is not used. The code is structured, and is event driven.

This code is a good model for additional work, and the logic in the DeltaCAD "macros" can easily be ported to this Visual Basic.

