

SOME NOTES ON PROGRAMMING FOR DELTA-CAD and JustBASIC

This project was a cube dial, on an 8 by 8 by 8 cinder block, with the faces in clay.

This dial will be for Silver City, NM, whose coordinates are:

location lat:	32.75° N
location long:	108.2° W
magnetic declination:	10.6° E

Because a lot of dialists have been using Delta CAD, this project was going to use Delta CAD macros and clay for the dial plates of which there were to be four. One would be a horizontal dial, one a vertical, with the two remaining being meridian dials. These would be non decliners so there would be no alignment issues, however these dials would be longitude corrected. The reason for using Delta CAD was because it is becoming popular, second because the printed dial plates can be cut and used over clay to mark hour or calendar lines.

The author mostly uses glass and copper on his outside dials, however he was trained in sculpture and pottery in the days of his youth, and he had some kilns lying around the place.

The author's first career before winding up as an airline pilot was in programming computers. Some commercial, but mostly operating system software. The first system he programmed was an IBM 1401 in Autocoder. Then an IBM 360 in BAL (Basic Assembler Language), with a bit of PL/I, and FORTRAN. All his work on the IBM 370 and later machines was in BAL and on some machines in C and C++. Operating systems used were BOS, MFT, and MVT on the IBM 360s, and VS1 and MVS on the IBM 370s, and GCS under VM also.

One pet peeve the author has is that documentation for languages is drawn up by programmers as language specifications, and when the human interface is covered, it is always somewhat academic. This tendency became worse when object oriented programming techniques became the standard, and the novice is faced with a bit of a struggle. Thus the examples in this section for the BASIC macro language for DeltaCAD are intended to be "conversational" as opposed to transaction oriented. In other words less object oriented and more of a natural flow.

As background, the author designed FIDO and PATCHES which were early spooling systems on IBM 360 mainframes in the 1970s, and TOTO and later SHADOW which were teleprocessing programs running under DOS, MFT, and MVS on the IBM 370 and later ranges which were sold worldwide from 1972 until 1997. SHADOW in particular made over \$55m in sales before the author lost track and interest.

USING A CAD SYSTEM and USING CAD MACROS

A CAD system is a computer program that draws, usually rather better than the average human. The 2d systems such as Delta CAD are simple to use and provide professional draftings. The author prefers TurboCAD deluxe which is a full 3d modeling system and which was used for most of the pictorials in this series of books.

However, Delta CAD is clearly better than TurboCAD if one wishes to have the computer do all of the work, including calculating angles and then drawing them. However, computers do what they are told and thus special techniques are needed to do simple things a human automatically can do, such as constraining a line to the boundaries of a box.

This not intended to be a definitive work on Delta CAD and its associated macro language, but rather a few pointers to explain some of quirks. The author spent a total of less than a day reading the Delta CAD manual and the associated macro manual, and then proceeded forth. So, writing macros for Delta CAD is obviously within the reach of a computer literate person.

This section extracts some of the code and explains what it is doing, and then final dial plates are shown. There are many macros available for Delta CAD, they are well worth exploring. The author has his own versions of Delta CAD macros on the web site for those who are interested, and these complement the spreadsheets which are also made available.

Programs begin with initial setup, then they define variables to hold information being worked on, and also ask humans what they want, and then they produce the results.

```
' *****  
' A horizontal dial macro for Delta CAD but in conversational mode as  
' opposed to the more modern object oriented mode, but with notes  
' page numbers refer to Manual.pdf or Basic.pdf provided with delta cad  
' *****  
  
Sub Main() ' main procedure is required  
  
' *****  
' Initial house keeping - clear the screen - set the drafting area unit  
' *****  
  
' select all objects that may exist on the screen - p223 of Manual  
' then erase them all - page 189 of Manual  
  
If (dcSelectAll) Then  
    dcEraseSelObjs  
End If  
  
' set the entire future drafting area to inches or centimeters, etc  
' page 43 of the Manual: 1.0 generates inches, and 2.54 is cm  
  
dcSetDrawingScale 0.80
```

At this point, a few thoughts on hard coded numbers, specific dimensions and programming practice are relevant.

Good program practice is not to code numbers inline in a program, rather, those numbers should be in a data definition area. This makes the program easier to change, however, it also makes the program a little harder to understand.

It is similarly bad practice to define display areas, as this program does, of say 0,0 to 1,1 however, if the end result is scalable there is little harm, however good practice would still use symbolic numbers, with those defined in the data definition area.

Programming practice has been for many years to use structured coding techniques, that is IF-END IF, DO-END DO, FOR NEXT, and the like, and never to use the GOTO statements.

Another more recent architecture of programming systems has been to move towards object oriented methods, that is, where an object not only holds the data, but also the methods for altering or reading it, and further, all objects have inter-relationships, so if something changes that might affect any object, then that object will find out and act accordingly. For example, in Windows bring up two displays of the same folder, and in one display, delete an entry. Object oriented methods are what cause the other folder display to update itself and reflect the first display's changed status.

While BASIC as supplied with Delta CAD is not truly object oriented, it does use some of the data definition concepts, for example the structures needed to talk with humans. Next some programming structures are needed for human interaction. These are not difficult, nor are they as simple as simple BASIC.

```
' *****
' A generic definition is required for a screen input area
' *****
'
' Here a box on the screen for user dialog is structurally defined,
' it is only a definition of the generic area, it does not create it
' ..... Dialog aaaaa
'
' To create the area, there must be a Dim statement making a label
' relate to this definition
' ..... Dim bbbbb as aaaaa
'
' To use bbbbb there must be a ..... yyy = Dialog(xxxxx) which causes
' human interaction. So...
'
' create an area on the screen starting at x=20, y=20
' whose size is 200 left to right and 100 top to bottom
' whose title is "Location" - where 0,0 is top left

Begin Dialog aaaaa 20, 20, 200,100, "Location"
' the first text string starts at x=5,y=15 on the screen
' and the text string itself starts at x=60 for a height of 10
Text      5, 15,  60,10,  "Enter latitude"

' the input area starts at x=65 (further right) y=15 (same height) for a
' size of x=50, y=10
TextBox   65, 15,  50, 10, .mylat

' the second text string starts at x=5 but now y=30, i.e. lower down
' and the text string itself starts at x=60 for a height of 10
Text      5, 30,  60, 10,  "Enter longitude"

' and the input area starts at x=65 (further right) y=30
' (same height) for a size of x=50, y=10
TextBox   65, 30,  50, 10, .mylng

' the third text string starts at x=5 y=45
' and the text string itself starts at x=60 for a height of 10
Text      5, 45,  60, 10,  "Enter ref longitude"

' and the input area starts at x=65 y=30 for a size of x=50, y=10
TextBox   65, 45,  50, 10, .myref

' and two buttons for what the user means, location first, button size
' next - and all such boxes must have at least one button by the way
OKButton  65, 65,  40, 10
CancelButton 65, 85,  40, 10
End Dialog

' *****
' The generic definition must then be generated with a name
' *****
'
' this defines "bbbbb" as an instance of aaaaa dialog
Dim bbbbb As aaaaa
```

At this point, a few comments might be helpful. The Begin Dialog has nothing to do with a dialog. It is an encyclopedia definition of what you might wish to actually create.

It is created with the Dim statement.

And used later.....

```
' *****  
' Now define the initial general working variables  
' *****  
'  
' define a lat and a long, and a reference longitude  
Dim lat As single  
Dim lng As single  
Dim ref As single  
  
' *****  
' Now get the lat, long, and reference longitude  
' *****  
  
' first set the defaults - here bbbbb.mylat uses the structure  
' from aaaaa  
bbbbb.mylat = "32.75"  
bbbbb.mylng = "108.2"  
bbbbb.myref = "105.0"
```

....., in fact here it is being used!

```
' here the dialog is invoked and the button results returned to ccccc  
' page 20 and 24 etc of Basic discusses the Dialog function  
cccc = Dialog(bbbbb)  
' which causes the answer to be returned  
lat = bbbbb.mylat  
lng = bbbbb.mylng  
ref = bbbbb.myref  
' CANCEL button returns 0  
' OK button returns -1  
' you can determine the button with - Print ccccc, lat, lng, ref
```

The rest of the program is straight forward.

```
' *****  
' ok, what was returned? if "ok button" then do the program itself  
' *****  
  
' ccccc = -1 means the ok button was used and not the cancel button  
If ccccc = -1 Then  
  
' *****  
' this is the main program to draw the horizontal dial itself  
' *****
```

```
' calculate hour line angles next, but first define them
Dim h, hx, hy As Single ' Delta CAD is fussy about data attributes

' the formula is... hourlineangle = atan ( sin(lat) * tan (lha) )
' almost all systems us radians
' the formula also needs adjustment for longitude displacement

' line color is 0 is black
' line type is dcsolid
' line weight is dcnormal

' set the text color, font, size, etc also
dcSetTextParms dcBLACK,"Ariel","Bold",0,8, 20,0,0 ' p231 of Manual
dccreatetext -1.25, -0.3, 0, "Hour and hour line angle H-DIAL"
dccreatetext -1.25, -0.9, 0, "Lat: "
dccreatetext -0.8, -0.9, 0, Int(lat)
dccreatetext 0.0, -0.9, 0, "Long: "
dccreatetext 0.3, -0.9, 0, Int(lng)

For hr = 6 To 18 Step 1
' =====
' for the hour (hr) calculate the hour line angle (h)
h = deg(Atn(Sin(rad(lat))*Tan(rad((hr*15) +(ref-lng))))))

' show the time in hours
dcSetTextParms dcBLACK,"Ariel","Bold",0,8, 21,0,0 ' p231 Manual
dccreatetext (-1.2+((hr-6)/5)), -0.5, 0, Abs(hr)
' show the angle
dcSetTextParms dcBLACK,"Ariel","Bold",0,6, 21,0,0 ' p231 Manual
dccreatetext (-1.2+((hr-6)/5)), -0.7, 0, Int(h)

If hr < 12 Then
' -----
' morning hours ~ NOTE code for keeping lines in a boxed area
' -----
dcsetlineparms dcblue,dcsolid,dcthin ' page 228 Manual
If Abs(h) < 45 Then ' lines touch top of box
dcSetTextParms dcBLACK,"Ariel","Bold",0,8,21,0,0 ' p231
hx = Tan(rad((h)))
dccreateline 0, 0, hx, 1
dccreatetext (hx), 1.1, 0, Abs(hr) ' page 187 of Manual
Else ' lines touch side of box
dcSetTextParms dcBLACK,"Ariel","Bold",0,8, 20,0,0 ' p231
hy = Tan(rad((90-h)))
dccreateline 0, 0, -1, -hy
dccreatetext -1.1, -hy, 0, Abs(hr) ' page 187 of Manual
End If

ElseIf hr = 12 Then
' -----
' noon hours
' -----
... similar code which is straight forward

Else
' -----
' afternoon hours
' -----
... similar code except the sign of the angle goes -ve if 90

End If
' =====
Next h
```

The program concludes with drawing a couple of boxes. And after the end of the program, there are the DEGREES and RADIANS functions.

```
' draw a box around everything
dcreatebox  -1,    0,    1,    1    ' page 184 Manual
dcreatebox  -1.2, -.2,  1.2,  1.2  ' page 184 Manual
dcviewbox   -1.1, -1.1, 1.1,  1.3  ' page 225 Manual
End If

' *****
' *** this ends the entire program
' *****
End Sub

'
*****
' Useful routines or functions - Functions must be defined at the end
' after the main program which is sub(xx) ... end sub
' *****

' Convert degrees to radians
Function Rad ( n As single ) As single
' page 83 basic.pdf for functions
  Rad = (n * 2 * 3.14162) / 360
End Function

' Convert radians to degrees
Function Deg ( n As single ) As single
' page 83 basic.pdf for functions
  Deg = (360 * n) / (2 * 3.14162)
End Function
```

The actual code on the web site may differ, however, the objective has been to step through most of the procedural coding. There are some coding violations, however they have been somewhat intentional in order to make the process of a complete macro, when accompanied by DeltaCAD's two books, MANUAL.PDF and BASIC.PDF, more understandable.

The horizontal dial and vertical dial methods are simple and straight forward. However, the meridian dial, uses distances along an equinox line rather than angles. And just as the horizontal and vertical dial have commonalities and differences, so does the east and west non declining meridian dial.

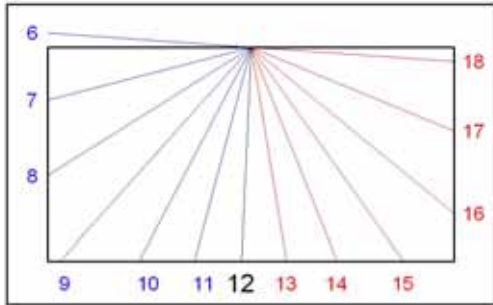
Distance calculations now need a gnomon linear height.

Things increase in complexity if calendar lines are added, and a bit more complex if those calendar lines are to terminate the hour lines.

While the horizontal and vertical dials had rigid code to constrain the hour lines within a boundary box, this was less of a concern in the meridian dial. For no good reason, the meridian dial logic allows the user to specify a gnomon linear height, and since it does, it is very easy for the hour lines to be far away, and exceed a boundary box. So, the meridian dial logic will not focus on constraining those hour lines and calendar lines, but instead will focus on the conversion from trigonometry to serial programming, and how the hour lines are constrained to the calendar lines.

The process is straightforward.

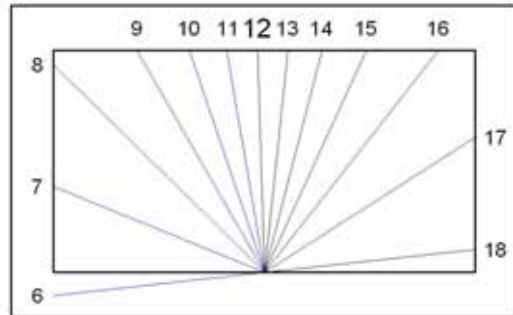
Four dial plates are produced with the DeltaCAD macros available on the web site.



Hour and hour line angle VERTICAL NON DECLINER

6	7	8	9	10	11	12	13	14	15	16	17	18
86	-77	-60	-44	-29	-16	-3	9	23	36	52	68	86

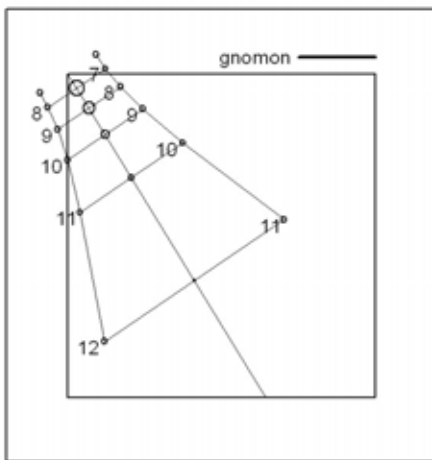
 Lat: 32 Long: 108



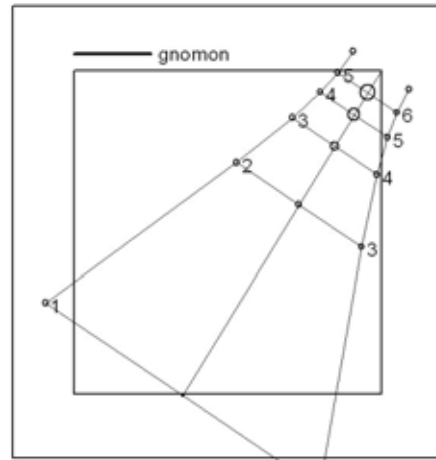
Hour and hour line angle H-DIAL

6	7	8	9	10	11	12	13	14	15	16	17	18
84	-69	-47	-32	-20	-11	-2	6	15	25	39	58	84

 Lat: 32 Long: 108



E meridian dial
 Lat: 32
 Long: 108



W meridian dial
 Lat: 32
 Long: 108



The CASE STUDY on clad dials shows a cube dial using these DeltaCAD macros, and the plat layouts above.

JustBASIC

Available at www.justbasic.com is a free BASIC true IDE (itegrated development environment). A more extensive version called Liberty Basic is availabel for a small fee.

This system is used in some of the programs on this web site as well as on the CD provided with these books.

The system is different from DeltaCAD, but very close. What follows is a simple program to display data for a simple horizontal dial.

```
' =====
' A simple horizontal dial with longitude correction
' by definition it can be a vertical dial for co-latitude
' =====

' Ask for longitude and latitude and the longitude for the legal time zone
print "-----H-DIAL-----"
print "If run as an .EXE file, please read [please Read JB.txt]"
print " sws-bas-h-dial.bas -or-- sws-bas-h-dial.tkn"
print " Last revision: Jan 27, 2007 0932 mst"
print "-----"
print " "
input "Enter design latitude: [eg 32.75] "; zla
input "Enter design longitude: [eg 108.2] "; zln
input "Enter legal meridian longitude: [eg 105] "; zle

' if null inut then setup defaults
if zla = 0 then zla=32.75
if zln = 0 then zln=108.2
if zle = 0 then zle=105

' Calculate the correction due to longitude, and display the correction
cor = zln-zle ' This is the correction in degrees
print "Lat: ", zla
print "long: ", zln, "Ref: ", zle
print "corr: ", 4*cor, "minutes" ' This is the correction in minutes
print "-----"

' Loop the range of hours and show the hour and its angle

print " "
print "If the sign for the earlier hours differ from the later hours"
print " then it means the hour line went past 90, adjust accordingly."
print " "
print "Hr: ", "no corr: ", "long corr: "
for n=05 to 19 step 1 ' Start with am on left, to pm on right
zhr = n - ( 4*cor/60) ' This is the correction in hours
' One could use IF, ELSEIF, etc however JustBASIC does not support ElseIf
if n < 12 then
if n = 11 then
print n, shrink(hdial(zla,zln,12-n)), shrink(hdial(zla,zln,12-zhr)), "+ve is to the left"
else
print n, shrink(hdial(zla,zln,12-n)), shrink(hdial(zla,zln,12-zhr))
end if
end if
```

```

'
if n = 12 then
    print "- - - -"
    if zln >= zle then
        print n, shrink(hdial(zla,zln,12-n)), shrink(hdial(zla,zln,12-zhr)), "to the left of sub style"
    else
        print n, shrink(hdial(zla,zln,12-n)), shrink(hdial(zla,zln,12-zhr)), "to the right of sub style"
    end if
    print "- - - -"
end if
'

if n > 12 then
    print n, hdial(zla,zln,n-12), hdial(zla,zln,zhr-12)
    if n = 13 then
        print n, shrink(hdial(zla,zln,12-n)), shrink(hdial(zla,zln,12-zhr)), "-ve is to the right"
    else
        print n, shrink(hdial(zla,zln,12-n)), shrink(hdial(zla,zln,12-zhr))
    end if
end if
' repeat the loop
next n

print "Hr: ", "no corr: ", "long corr: "
print " "
print "If the sign for the later hours differ from the earlier hours"
print " then it means the hour line went past 90, adjust accordingly."

[dial]
print "-----"
input "end - hit ENTER to terminate"; zzz
end

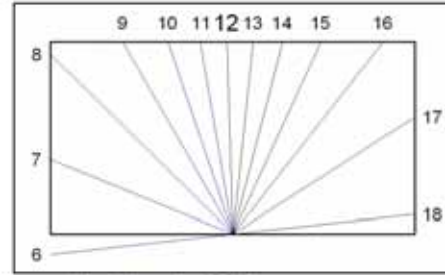
'define a function for returning n hour angle
'H = atan ( sin(lat) * tan (lha) )
function hdial(lat,long,hr)
    hdial = atn((sin(lat*2*3.1416/360) * tan(hr*15*2*3.1416/360)))*360/(2*3.1416)
end function

' Shrink to 2 significant digits
function shrink(a)
    shrink = int(a*100)/100
end function

' *** END ***
    
```

The results of this program for the vertical dial on the top left of page 8 are shown on the next page. The results of this program for the horizontal dial on the top right of page 8 are shown on the next page. For those with no Excel and no DeltaCAD, this JustBASIC system is an ideal option.

```
-----H-DIAL-----
Enter design latitude: [eg 32.75]
Enter design longitude: [eg 108.2]
Enter legal meridian longitude: [eg 105]
Lat:          32.75
long:         108.2      Ref:          105
corr:         12.8      minutes
-----
```



Hour and hour line angle H-DIAL

6	7	8	9	10	11	12	13	14	15	16	17	18
84	-69	-47	-32	-20	-11	-2	6	15	25	39	58	84

Lat: 32 Long: 108

If the sign for the earlier hours differ from the later hours then it means the hour line went past 90, adjust accordingly.

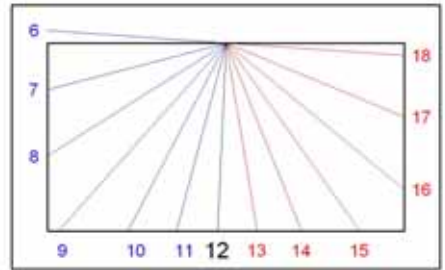
Hr:	no corr:	long corr:	
5	-63.64	-58.7	
6	-89.99	-84.09	
7	63.65	68.88	
8	43.13	46.96	
9	28.41	31.17	
10	17.34	19.49	
11	8.24	10.08	+ve is to
the left			

12	0	1.73	to the left of sub style

13	-8.24	-6.44	-ve is to the right
14	-17.34	-15.28	
15	-28.41	-25.81	
16	-43.13	-39.58	
17	-63.65	-58.71	
18	89.99	-84.09	
19	63.64	68.88	
Hr:	no corr:	long corr:	

The same program works for vertical dials by complementing the latitude.

```
-----H-DIAL-----
Enter design latitude: [eg 32.75]          57.25
Enter design longitude: [eg 108.2]        108.2
Enter legal meridian longitude: [eg 105]  105
Lat:          57.25
long:         108.2      Ref:          105
corr:         12.8      minutes
-----
```



Hour and hour line angle VERTICAL NON DECLINER

6	7	8	9	10	11	12	13	14	15	16	17	18
86	-77	-60	-44	-29	-16	-3	9	23	36	52	68	86

Lat: 32 Long: 108

If the sign for the earlier hours differ from the later hours then it means the hour line went past 90, adjust accordingly.

Hr:	no corr:	long corr:	
5	-72.32	-68.64	
6	-89.99	-86.19	
7	72.32	76.05	
8	55.53	59.01	
9	40.06	43.24	
10	25.9	28.82	
11	12.69	15.45	+ve is to the left

12	0	2.69	to the left of sub style

13	-12.69	-9.96	-ve is to the right
14	-25.9	-23.01	
15	-40.06	-36.94	
16	-55.53	-52.11	
17	-72.32	-68.64	
18	89.99	-86.19	
19	72.32	76.04	
Hr:	no corr:	long corr:	