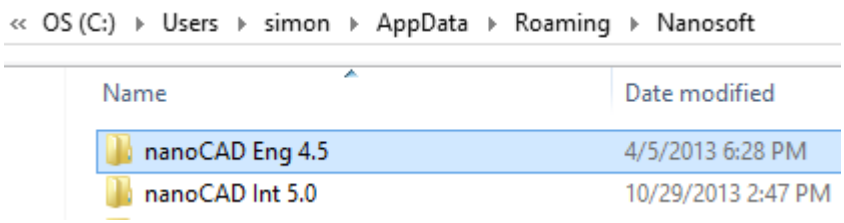
INSTALLING NANOCAD

NanoCAD 5 ~ ~ ~ October 29, 2013

The download, install, userid, and registration code process are simpler . If you used 4.5 then you must copy over the JS and VBS scripts to the newer installation.

NOTE: In NanoCAD 5 you must do FILE/NEW or NEW at the cmmand line, otherwise TOOLS/SCRIPTS will have JS and VBS greyed out.

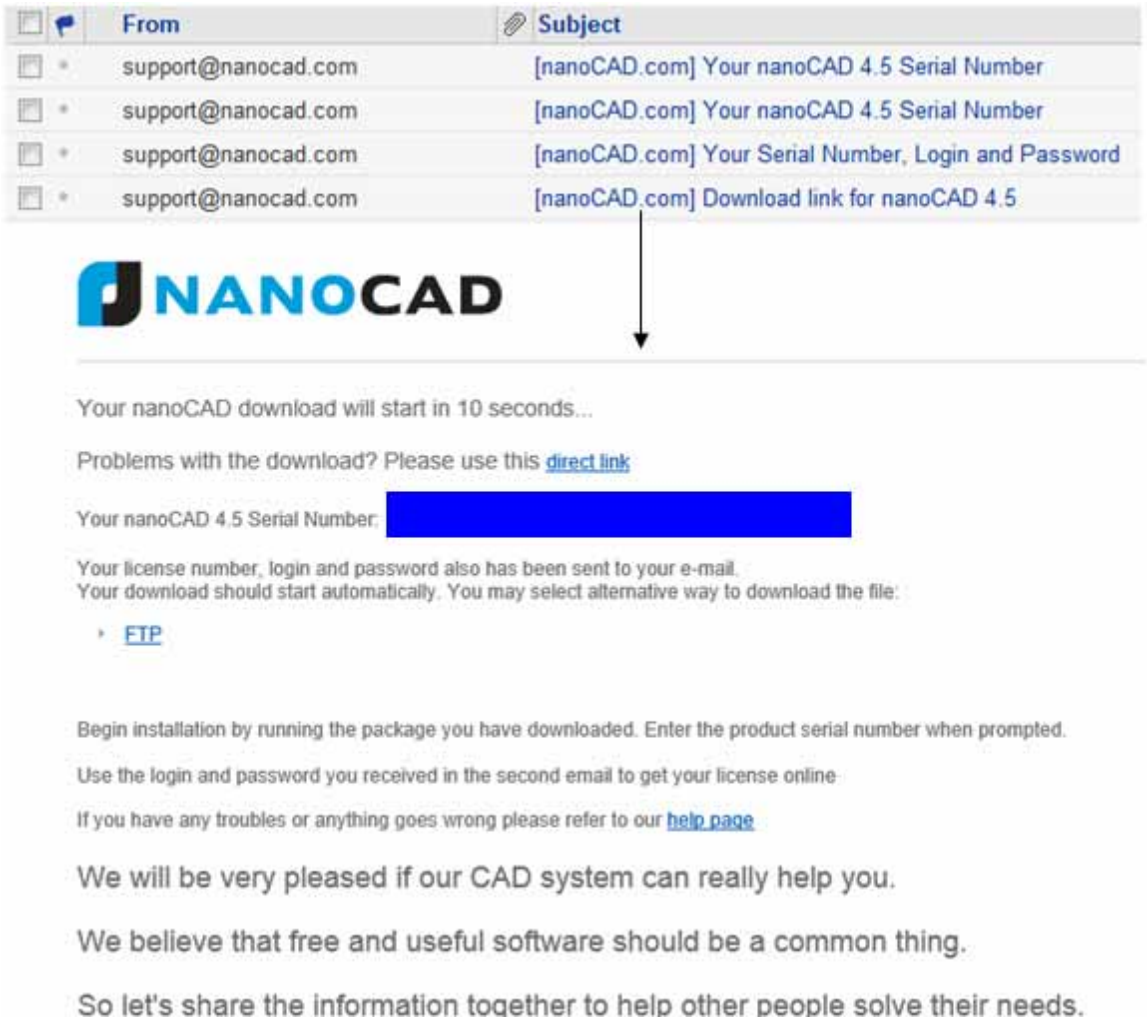


NanoCAD 4.5 ~ ~ ~ April 2, 2013 (NOTE: NanoVAD 5 is simpler to install)

1. www.nanocad.com
2. wait a long time
3. download is a multi step process

after the initial registration, go to your email and click on the download link

FIRST ESSENTIAL EMAIL ~ this has the serial number:-



then begin the download which you would SAVE AS somewhere that you can find it

SECOND ESSENTIAL EMAIL ~ this gives you a userid (email) and password:-

```
Your nanoCAD 4.5 serial number: [redacted]

Your nanoCAD.com login: anyuseron@yahoo.com
Password:  sdfghjklrtyui

Thank you for using our software!

--
nanoCAD.com Administration
```

Then you install the software, it will ask for the serial number from the first email:-

[redacted]

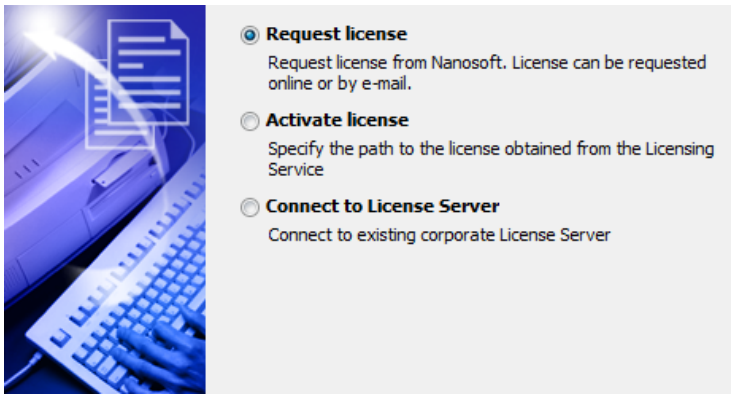
then it will need the license, which is done online with the logon id (your email you gave them) and a password they emailed to you:-

nanoCAD.com login: anyuseron@yahoo.com

Password: sdfghjklrtyuio

The next couple of pages show the screens you will see during the process:-

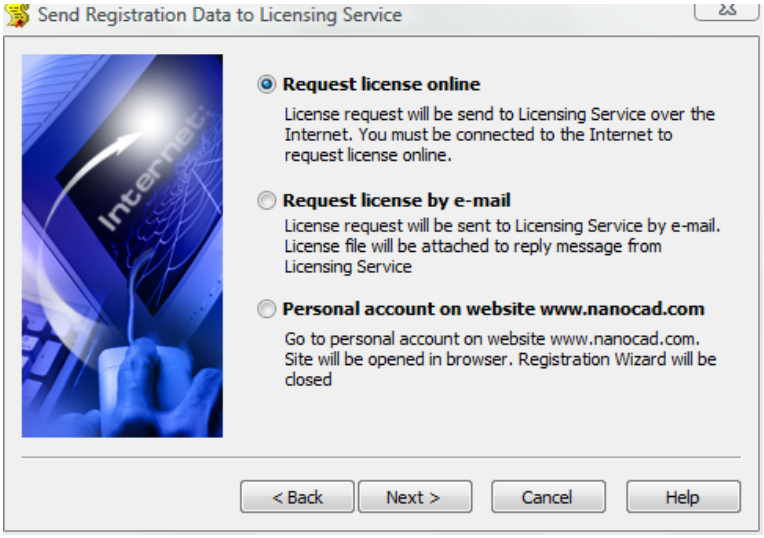
After the install then it requires registration:-

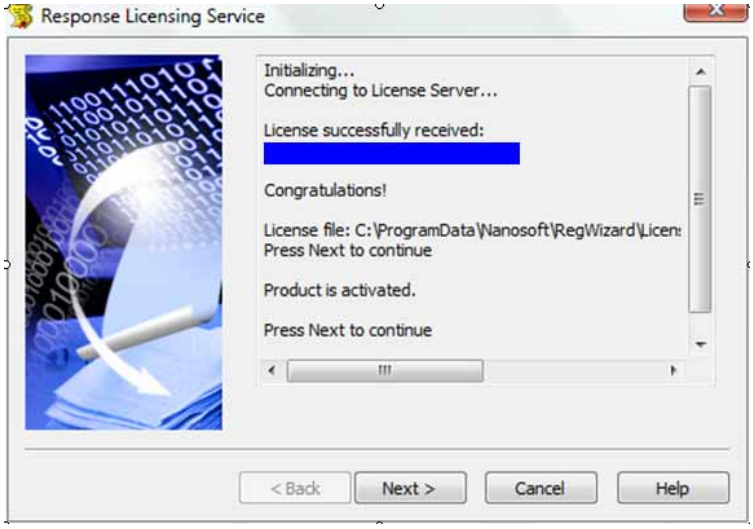


then

then


CDKey:
Login:
Password:
ID1:
ID2:
Customer:
Company:
Country:
Build:
RWB:


Send Registration Data to Licensing Service
Request license online
License request will be send to Licensing Service over the Internet. You must be connected to the Internet to request license online.
Request license by e-mail
License request will be sent to Licensing Service by e-mail. License file will be attached to reply message from Licensing Service
Personal account on website www.nanocad.com
Go to personal account on website www.nanocad.com. Site will be opened in browser. Registration Wizard will be closed
< Back
Next >
Cancel
Help


Response Licensing Service
Initializing...
Connecting to License Server...
License successfully received:
Congratulations!
License file: C:\ProgramData\Nanosoft\RegWizard\Licen
Press Next to continue
Product is activated.
Press Next to continue
< Back
Next >
Cancel
Help

NANOCAD FEATURES

Русская версия

NANOCAD

Why nanoCAD
Support
Developers
Company

http://nanocad.com/page/KeyFeatures

nanoCAD Key Features

nanoCAD is a multi-purpose free drafting software

nanoCAD is easy-to-use CAD application delivering great user experience by providing classic interface and native .dwg support. Being ultimate free drafting software nanoCAD has been built to deliver design and project documentation regardless of the industry an enterprise is operating at. nanoCAD includes all necessary tools required for basic design and allows creating and editing 2D and 3D vector objects. nanoCAD provides innovative, collaborative and customizable features to enhance your efficiency. nanoCAD is also free CAD platform with several API's serving different goals from routine task automation to complex CAD application development.

Industry-standard user interface

nanoCAD implements a very common approach to the CAD user interface. It offers typical command set, UI elements appearance as any other design and drafting CAD solution. The drawing space, command line, position of the menu items and icons on the toolbars are readily recognizable. Any engineer with CAD experience will use nanoCAD effectively from the very first minute

Clean and native *.dwg support

nanoCAD uses native *.dwg format, one of the most widely used design data formats. Once drawings are made in nanoCAD, they can be used in any other popular CAD systems without conversion or data loss. If you can't open a document due to damage or you want to audit an opened document for errors and correct some of them or you need to reduce file size, use Audit, Recover and Purge commands. Be assured you will not lose document data by software or hardware failures thanks to autosaving and backup functions.

Comprehensive command set

nanoCAD includes appropriate toolbox to create and edit 2D/3D objects. There are several drawing methods available for most of nanoCAD's geometric. Effective object editing commands allows you to modify drawings with minimal mouse clicks. Reusable blocks of elements and references to external drawings simplify and speed up the drawing process. Advanced dimensioning feature enables you to create any kind of dimensions by several means.

Powerful Table Editor

nanoCAD has Excel-style Table editor. It has unique features and very useful for creating tables with macros in their cells.

ActiveX Automation and LISP

Use nanoCAD scripting engine to automate everyday routine tasks. User can write macros using Visual Basic Script, Java Script or any other scripting language supported by Microsoft Windows as well as built-in LISP (since version 4.5).

C++/C# API

nanoCAD has several types of API for creating CAD applications using nanoCAD functionality. NRX is a C++ and .NET API very close to

AutoCAD ARX API. It allows the translation of AutoCAD applications to nanoCAD with ease. It is an object-oriented, compact and robust programming interface. It encapsulates CAD platform functionality. There is a new addition to nanoCAD API starting from version 4.5. MultiCAD API for C++ and .NET is a revolutionary development tool to create binary compatible applications for different CAD platforms. Application, developed with the use of a MultiCAD API will be able to run in nanoCAD as well in the other CAD, e.g AutoCAD.

Extended plotting

nanoCAD's plot settings allow you to set several plot areas and provide multipage plot; this can be useful to print large drawings on printers with smaller output format. Batch Plot command creates and prints a drawing set. It is convenient when it is needed to print out already finished project at once. Also it is possible to output drawings to a single or multi-sheet plot file.

USING NANOCAD

Some useful web links:-

http://nanocad.com/page/Programming1

http://nanocad.com/page/Programming2

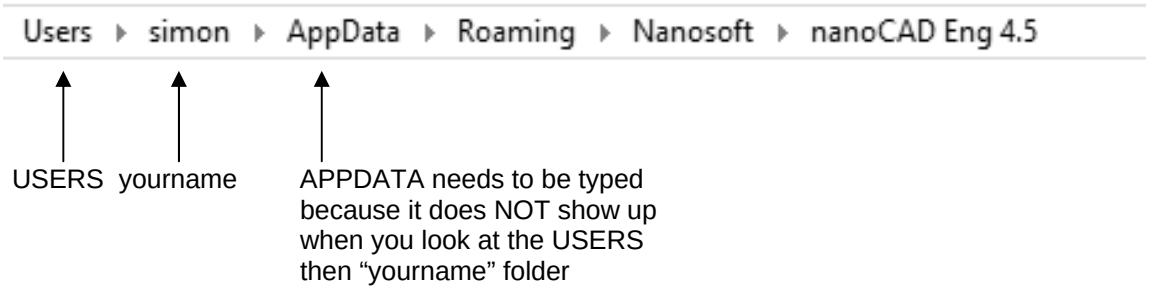
http://www.w3schools.com/vbScript/vbscript_ref_functions.asp

http://forum.nanocad.com/index.php?/topic/13-script-example/
has line draw example

Page 529 of “User guide.pdf”

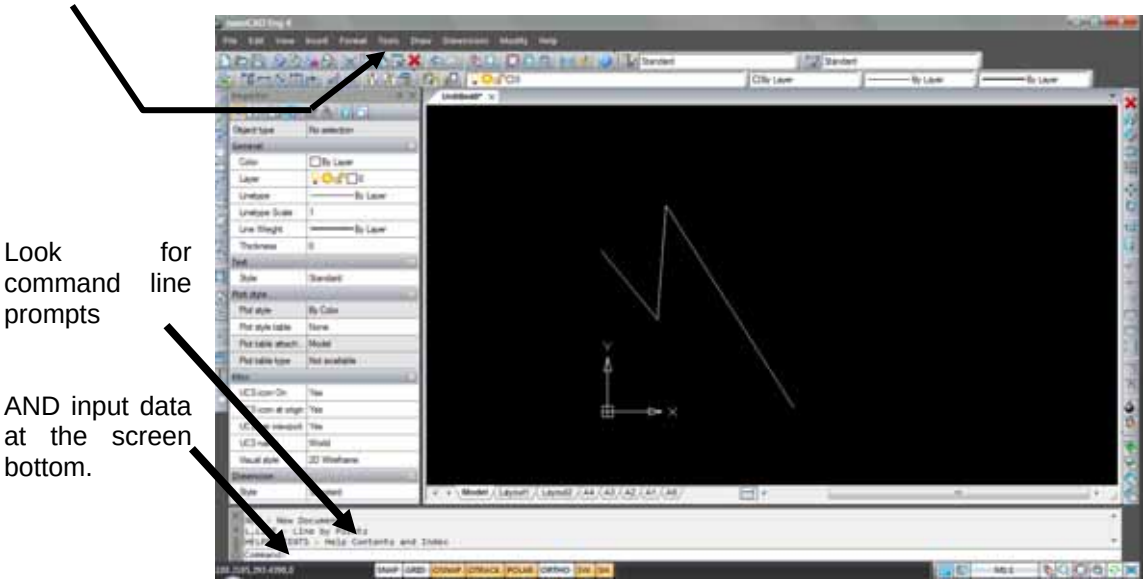
VBS ~ Visual basic Script

NanoCAD loads visual basic scripts from “appdata” in “users\yourname\xxx.” as shown below”-



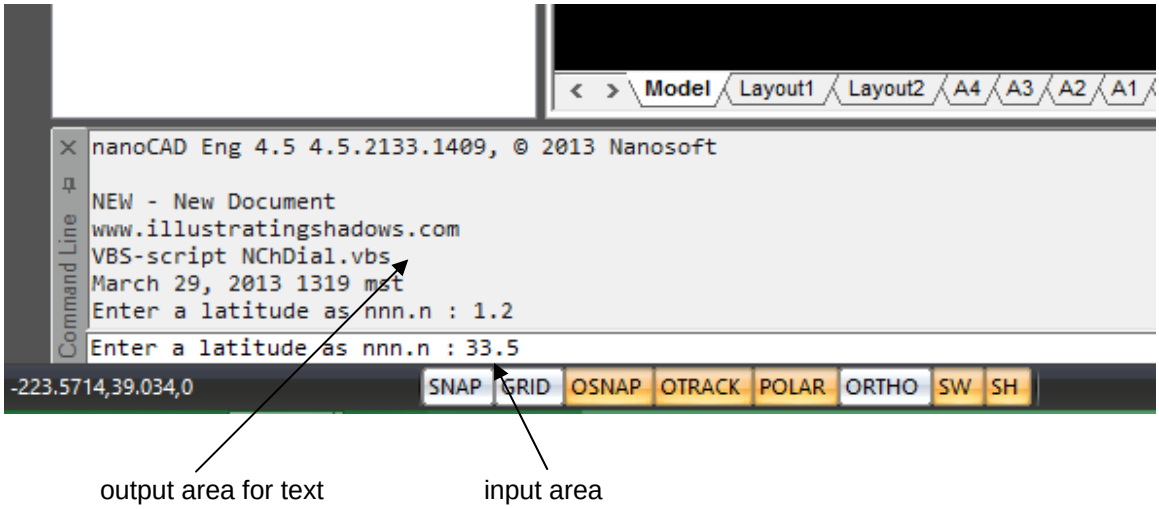
and using this folder saves lots of clicks. However, you are allowed to browse and load VBS programs from anywhere.

TOOLS, SCRIPTS, VBS SCRIPTS is how you load a VBS program which runs immediately.



As with many free systems, it is not overburdened by documentation. So these notes continue the PBE (programming by example) method of explaining what is happening.

The good news is that VBS is installed on your Windows system so VBS programs will run straight away once NanoCAD is installed.



The script “NChDial.vbs” can be loaded by placing it in the standard folder, or picked up from browsing in TOOLS, SCRIPTS, VBS SCRIPTS.
The plotting area is cleared of prior information with ctrl-A and delete, that simple.

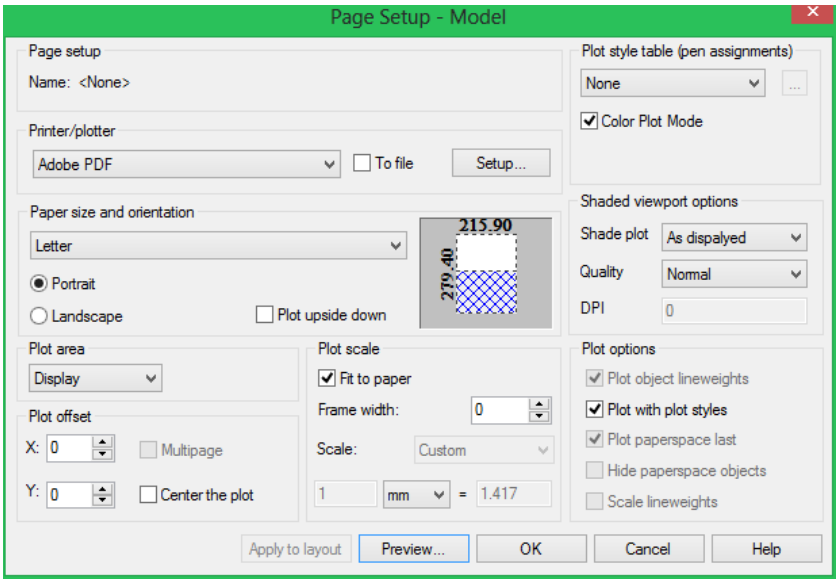
```
' ###########################################################################
' #  NChDial.vbs  #                Simon Wheaton-Smith march 29, 2013       #
' ###########################################################################
' #  a horizontal sundial program written in VBS for the NanoCAD software    #
' #  NanoCAD is available at:      http://nanocad.com/                       #
' #                                free but needs registering and a license  #
' #                                see "installUseNanoCAD.pdf"  on           #
' #                                www.illustratingshadows.com               #
' ###########################################################################

' http://www.w3schools.com/vbScript/vbscript_ref_functions.asp
' http://nanocad.com/page/Programming1
' http://forum.nanocad.com/index.php?/topic/13-script-example/
'                                    ' has line draw example

' The Dim for swsms simply creates a variable into which the Set
' swsms places the system thingy called ThisDrawing.ModelSpace
' this allows "swsms." to be used in place of ThisDrawing.ModelSpace
Dim swsms
Set swsms = ThisDrawing.ModelSpace

Dim ms
Set ms    = ThisDrawing.ModelSpace

' The Dim and Set below allow ThisDrawing.Utility to be replaced by
' the variable "ut" as shown in the "Prompt" statements below
Dim ut
Set ut    = ThisDrawing.Utility

ThisDrawing.Utility.Prompt("www.illustratingshadows.com")
ThisDrawing.Utility.Prompt("VBS-script NChDial.vbs")
ut.Prompt("March 29, 2013 1454 mst")
```

```
' ###############################################################################
' Ask for latitude, longitude, and legal meridian
' ###############################################################################

' Now predefine some variables, although vbs will define them on the fly if uses
Dim latr
Dim lat
Dim lats

latr = ThisDrawing.Utility.GetReal("Enter a latitude as nnn.n ")
'
' ThisDrawing.Utility.Prompt(latr)
lats = sin(2*3.14162*(latr)/360)
'ThisDrawing.Utility.Prompt(lats)
'
dim lng
lng = ThisDrawing.Utility.GetReal("Enter longitude as nnn.n ")
dim ref
ref = ThisDrawing.Utility.GetReal("Enter legal meridian as nnn.n ")


' ###############################################################################
' draw the hour lines
' ###############################################################################
dim x, y, z
x = y = z = 0
' establish a radius of 100 for arbitrary reasons
r = 100
for i = 5 to 19

    ha = (ref-lng + (i * 15) ) * 2 * 3.1416 / 360
    ' get an hour line angle in degrees
    hla = 360*atn(lats*tan(ha))/(2*3.1416)

    x=r*sin(2*3.1416*hla/360)
    y=r*cos(2*3.1416*hla/360)

    ' late afternoon code to stop it showing up as morning
    if i > 12 then
       if hla < 0 then
          x = -x
          y = -y
       end if
    end if
    ' early morning code to stop it showing up as afternoon
    if i < 12.01 then
       if hla > 0 then
          x = -x
          y = -y
       end if
    end if

    ' draw the hour line
    ptb = "0,0,0"
    pte = CStr(x)+","+CStr(y)+","+CStr(0)
    Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)

    ' name the hour line
    Set hourObj = ThisDrawing.ModelSpace.AddText(i,   pte , 5)

    hlap = int(hla*100) / 100
    ' show its angle also
    if i < 12 then
       ptq = CStr(x-10)+","+CStr(y+10)+","+CStr(0)
    else
       ptq = CStr(x+10)+","+CStr(y+10)+","+CStr(0)
    end if
    Set hourObj = ThisDrawing.ModelSpace.AddText(hlap, ptq , 3)

next
```

```
' ###############################################################################
' #  Draw the gnomon sub style                                                  #
' ###############################################################################

k = 3
' k is the divisor of the "R" value so things look reasonable
ptb = "0,0,0"
pte = CStr(0)+","+CStr(130)+","+CStr(0)
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)

x = r * tan(latr*2*3.1416/360)    /k
' use a sub style length.......  1/k ...... of the R setting
ptb = "0,0,0"
pte = CStr(x)+","+CStr(r/k)+","+CStr(0)
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)

ptb = CStr(0)+","+CStr(r/k)+","+CStr(0)
pte = CStr(x)+","+CStr(r/k)+","+CStr(0)
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)

' ###############################################################################
' #  Draw the equinox line                                                      #
' ###############################################################################

' k is the divisor of the "R" value so things look reasonable, must be same
' as was used when drawing the gnomon
x = r * tan(latr*2*3.1416/360)    /k
yd = (r/k)  +  x / tan((90-latr)*2*3.1416/360)
' yd is the displacement up the dial plate from the dial center

ptb = CStr(-50)+","+CStr(yd)+","+CStr(0)
pte = CStr( 50)+","+CStr(yd)+","+CStr(0)
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)

' ###############################################################################
' #  display key design data                                                    #
' ###############################################################################

ptq = "-140,110,0"
Set hourObj = ThisDrawing.ModelSpace.AddText("Latitude: " , ptq , 4)
ptq = "-100,110,0"
Set hourObj = ThisDrawing.ModelSpace.AddText( latr, ptq , 4)

ptq = "-140,100,0"
'Set hourObj = ThisDrawing.ModelSpace.AddText("Longitude: "+Cstr(lng) , ptq , 4)
' could do above but alignment is better if we do the below
Set hourObj = ThisDrawing.ModelSpace.AddText("Longitude: " , ptq , 4)
ptq = "-100,100,0"
Set hourObj = ThisDrawing.ModelSpace.AddText( lng, ptq , 4)

ptq = "-140, 90,0"
Set hourObj = ThisDrawing.ModelSpace.AddText("Legal: " , ptq , 4)
ptq = "-100, 90,0"
Set hourObj = ThisDrawing.ModelSpace.AddText( ref, ptq , 4)

' ###############################################################################
' #  Draw a box to bound the depiction                                          #
' ###############################################################################

ptb = "-150,-80,0"
pte = "+150,-80,0"
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)
ptb = "-150,130,0"
pte = "+150,130,0"
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)
ptb = "-150,-80,0"
pte = "-150,130,0"
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)
ptb = " 150,-80,0"
pte = " 150,130,0"
Set lineObj = ThisDrawing.ModelSpace.AddLine(ptb,pte)
```

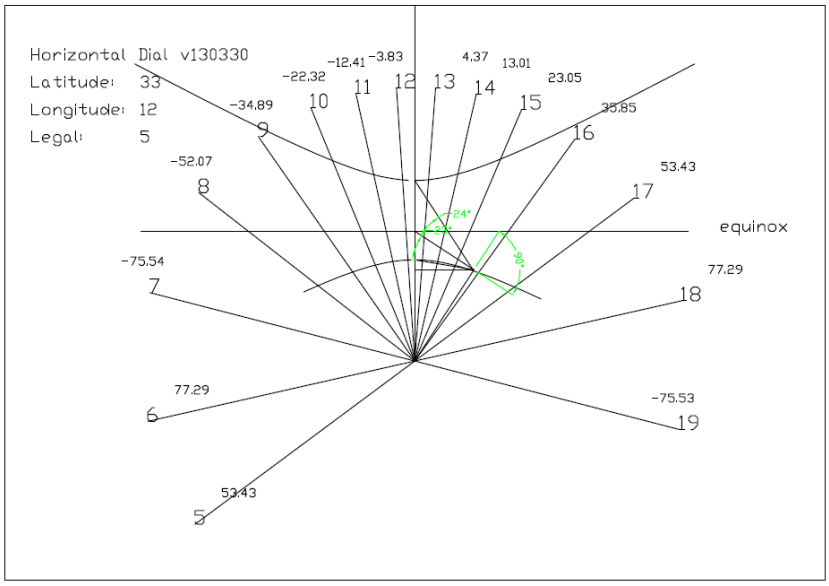
```
' ###############################################################################
' #  make sure you can find the darn thing                                      #
' ###############################################################################

ThisDrawing.Utility.Prompt "Perform a ZoomAll"
ThisDrawing.Application.ZoomAll



ThisDrawing.Utility.Prompt("*** END ***")
' #######
' # END #
' #######
```

Printing is done by the plotter. FILE, PAGE SETUP, EDIT will allow you to select any printer, which is simplicity itself, however if not done, PLOT PREVIEW and PLOT will not work.



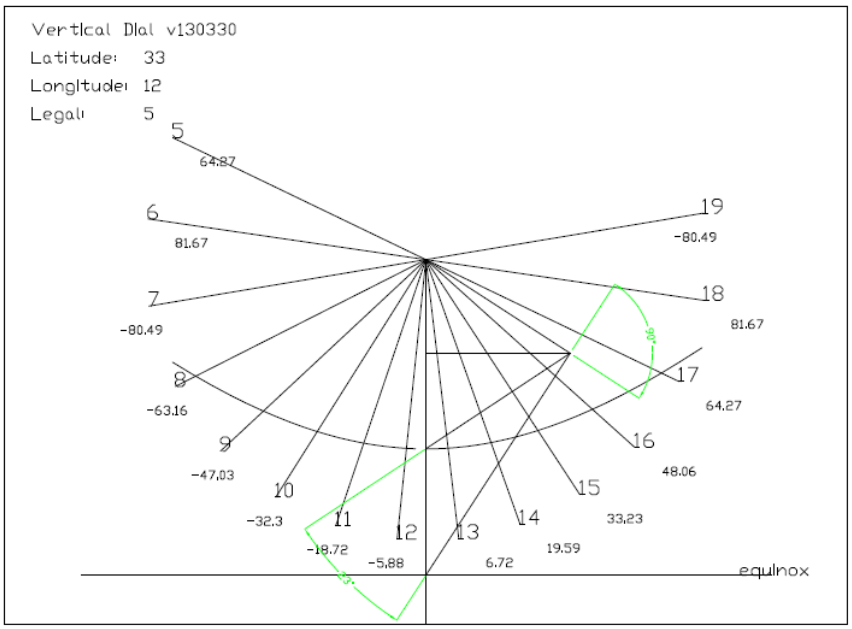
Horizontal dial: NChDial.vbs

The display has a black background, but when printed it is white. However, a screen capture would capture it as black with white lines. The result looks something like:-



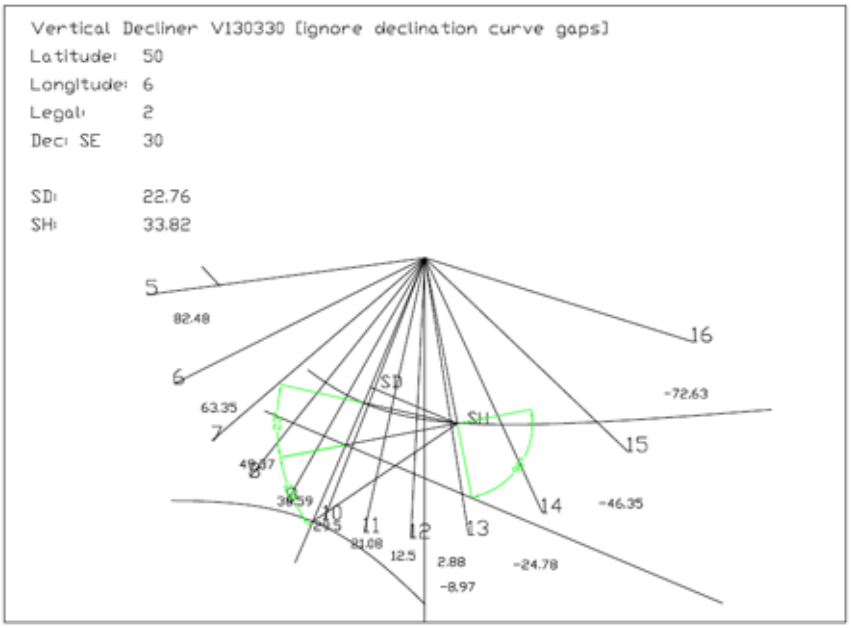
Vertical dial:

NCvDial.vbs



Vertical decliner:

NCvDec.vbs



JS ~Java Script

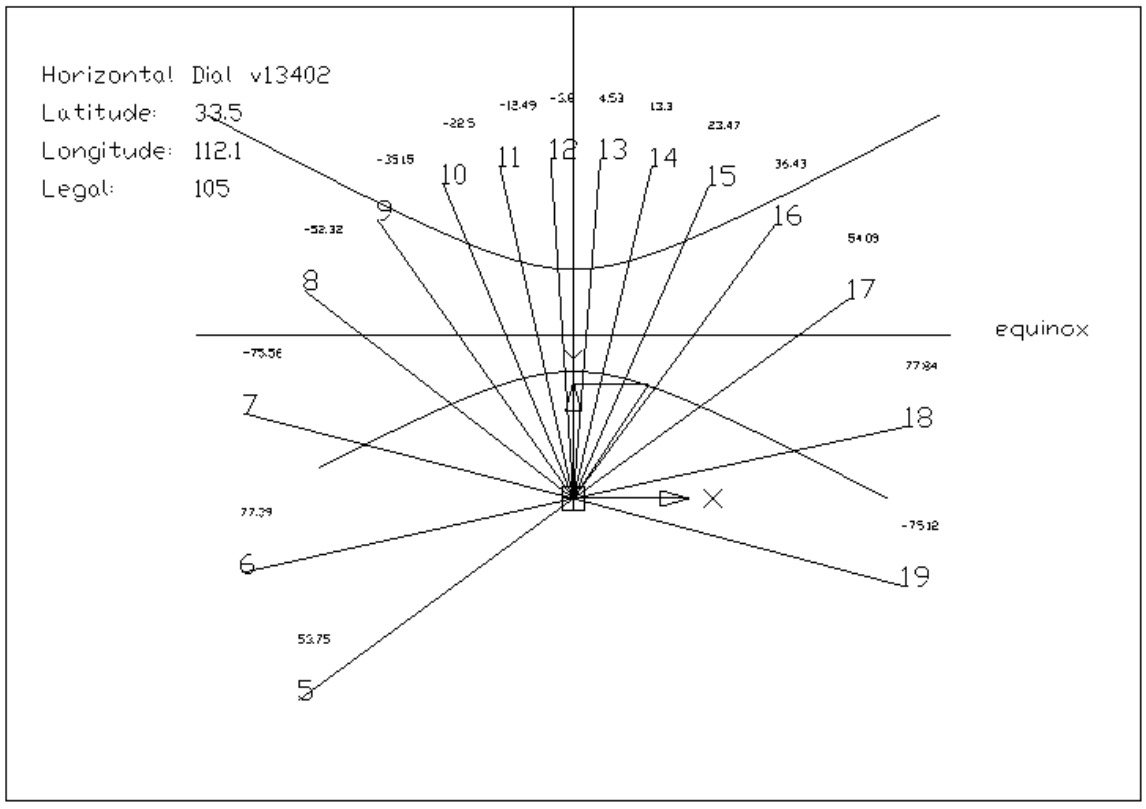
```
/* ********************************************************************************
   * NanoCAD   Java Script as opposed to VBS *     horizontal dial                  *
   ********************************************************************************
*/

function nnXnnn (q) {
  n   = Math.round(q*1000)/1000                        /* Math.round = int */
  return n
}

function nnXnn (q) {
  n   = Math.round(q*100)/100                          /* Math.round = int */
  return n
}

function arcSin(no1) {
   asno1 = Math.asin(no1)
   return asno1*360/(2*3.1416)
}

function azimuth(hr, lat, decl)  {
   th = hr
   if (th > 12) {        /* ensure hour in formula below is hours from noon */
      th = th - 12
      }
   else {
      th = 12 - th
   }
   Rlat   = lat  * 2  *    3.1416 / 360
   Rdecl  = decl * 2  *    3.1416 / 360
   Rth15  = th   * 15 * 2* 3.1416 / 360

   tz = 360*(Math.atan(Math.sin(Rth15)/(Math.sin(Rlat)*Math.cos(Rth15)-
Math.tan(Rdecl)*Math.cos(Rlat))))/(2*3.1416)
                        /* atn is in radians, so we make the end result degrees */
   if (tz > 90) {         /* ensure that negative azimuths are handles correctly  */
         tz = 180 - tz
   }
   if (tz < 0) {          /* test this second */
         tz = 180 + tz
   }
   if (hr >= 12) {
      tz = -Math.abs(tz)
      }
   else {
      tz = Math.abs(tz)
   }
   return tz
}

function altitude(hr, lat, decl)  { /* ensure hour in formula below is from noon */
   th = hr
   if (th > 12) {
      th = th - 12
      }
   else {
      th = 12 - th
   }
   Rlat   = lat  * 2  *    3.1416 / 360
   Rdecl  = decl * 2  *    3.1416 / 360
   Rth15  = th   * 15 * 2* 3.1416 / 360

   ta1 = Math.sin(Rdecl)*Math.sin(Rlat)+Math.cos(Rdecl)*Math.cos(Rlat)*Math.cos(Rth15)
   ta  = arcSin( ta1 )       /* arcSin returns degrees */
   if (ta < 0) {             /* If below the horizon */
      ta = ta
   }
   return ta
}
```

```
function hlaH (hr, lat, lng, ref) {             /* hour can be 0 to 23.59 */
   ha    = hr*15 - (lng-ref)
   hlat  = Math.sin(2*3.146*lat/360)*Math.tan(2*3.146*ha/360)
   hla1  = 360*(Math.atan(hlat))/(2*3.146)
   hla   = nnXnn(hla1)
   return hla1
}

function makeText(  i,    x,y   ,sz ) {
   var ptt
   ptt = String(x)+","+String(y)+","+String(0)
   ThisDrawing.ModelSpace.AddText(i , ptt , sz)
   return 0
}

function makeLine(x1,y1   ,   x2, y2) {
   var ptb
   var pte
   ptb = String(x1)+","+String(y1)+","+String(0)
   pte = String(x2)+","+String(y2)+","+String(0)
   ThisDrawing.ModelSpace.AddLine(ptb,pte)
   return 0
}



/*##########################################################################################
'#  E N D    F U N C T I O N S  #                                                          #
'##########################################################################################
*/



/* ***************************************************************************
   * NanoCAD   Java Script as opposed to VBS *     horizontal dial                *
   ***************************************************************************/

var ptb
var pte

ThisDrawing.Utility.Prompt(" H   H   EEEEE   RRRR     EEEEE")
ThisDrawing.Utility.Prompt(" H   H   E       R   R    E    ")
ThisDrawing.Utility.Prompt(" HHHHH   EEE     RRRR     EEE  ")
ThisDrawing.Utility.Prompt(" H   H   E       R   R    E    ")
ThisDrawing.Utility.Prompt(" H   H   EEEEE   R    R   EEEEE")
ThisDrawing.Utility.Prompt("  ")

ThisDrawing.Utility.Prompt(" Java Script: NanoCAD: v" + 130402)

lat = ThisDrawing.Utility.GetReal("Enter latitude  as nnn.n ")
lng = ThisDrawing.Utility.GetReal("Enter longitude as nnn.n ")
ref = ThisDrawing.Utility.GetReal("Enter legal     as nnn.n ")

var h
var lat
var r

/*###########################################################################
  #  display the hour lines                                                 #
  ###########################################################################
*/
for ( h = 5; h<20; h++)
{
      hla = hlaH (h, lat, lng, ref)

      x=r*Math.sin(2*3.1416*hla/360)
      y=r*Math.cos(2*3.1416*hla/360)

      /* late afternoon code to stop it showing up as morning */
      if (h > 12) {
         if (hla < 0) {
            x = -x
            y = -y
```

```
         }
      }

      /* early morning code to stop it showing up as afternoon*/
      if (h < 12) {
         if (hla > 0) {
            x = -x
            y = -y
         }
      }

      ptb = "0,0,0"
      pte = String(x)+","+String(y)+","+String(0)
      makeLine(0,0,  x,y)
      makeText(h,   x,y,     5)

      /*pte = String(x)+","+String(y+15)+","+String(0)*/
      n   = nnXnn(hla)
      makeText( n ,  x,y+15 , 2)
 }


/*##############################################################################
  #  Draw the gnomon sub style                                                 #
  ##############################################################################
*/

k = 3          /* k is the divisor of the "R" value so things look reasonable */
ptb = "0,0,0"
pte = String(0)+","+String(130)+","+String(0)
ThisDrawing.ModelSpace.AddLine(ptb,pte)

x = r * Math.tan(lat*2*3.1416/360) /k
/* use a sub style length.......  1/k ...... of the R setting */

/* or use our own function rather than the native stuff */
makeLine(0,0,   x, r/k)

/* or use our own function rather than the native stuff */
makeLine(0,r/k,   x, r/k)

/* save the point on the dial plate relating to the nodus */
nodusx = 0
nodusy = r/k
/* as well as the gnomon linear height, needed for calendar curves */
glh    = (r/k) * Math.tan(lat*2*3.1416/360)



/*##############################################################################
  #  Draw the equinox line                                                     #
  ##############################################################################
*/

/* k is the divisor of the "R" value so things look reasonable, must be same
   as was used when drawing the gnomon */
x  = r * Math.tan(lat*2*3.1416/360)/k
yd = (r/k)  +  x / Math.tan((90-lat)*2*3.1416/360)
/* yd is the displacement up the dial plate from the dial center */

ptb = String(-100)+","+String(yd)+","+String(0)
pte = String( 100)+","+String(yd)+","+String(0)
ThisDrawing.ModelSpace.AddLine(ptb,pte)
ThisDrawing.ModelSpace.AddText("   equinox " , pte , 4)
```

```
/*#############################################################################
' #  MAIN LOOP   #   solstice declinations                                     #
' #############################################################################
*/

/* this uses NODUSX,Y from the gnomon drawing earlier */
wx = 9999                     /* these, when 0, tell the calendar */
wy = 9999                     /* line draw to draw nothing */

for (hr=7;  hr < 17;  hr=hr+0.25) {
   /* get an altitude and azimuth for this one hour at this winter SOLAR declination */
   al = altitude( hr, lat, -23.44 )      /* degrees altitude */
   d  = glh / Math.tan(al*2*3.1416/360)  /* radial distance fr nodus base to point on cal
curve */
   az =  azimuth( hr, lat, -23.44 )      /* degrees azimuth */
   xxx=  d * Math.sin(2*3.1416*az/360)
   yyy=  d * Math.cos(2*3.1416*az/360)
   if (  wx<9999 && wy<9999 ) {
         if ( Math.abs(nodusx+wx)<150 && Math.abs(nodusx+xxx)<150 ) {
            if ( Math.abs(nodusx+wx)<130 && Math.abs(nodusx+xxx)<130 ) {
              makeLine(  nodusx + wx,nodusy + wy   ,   nodusx + xxx, nodusy + yyy  )
            }
         }
   }
   wx = xxx
   wy = yyy
}

wx = 9999                     /* these, when 0, tell the calendar */
wy = 9999                     /* line draw to draw nothing */

for (hr=6;  hr < 18;  hr=hr+0.25) {
   /* get an altitude and azimuth for this one hour at this winter SOLAR declination */
   al = altitude( hr, lat,  23.44 )      /* degrees altitude */
   d  = glh / Math.tan(al*2*3.1416/360)  /* radial distance fr nodus base to point on cal
curve */
   az =  azimuth( hr, lat,  23.44 )      /* degrees azimuth */
   xxx=  d * Math.sin(2*3.1416*az/360)
   yyy=  d * Math.cos(2*3.1416*az/360)
   if (  wx<9999 && wy<9999 ) {
         if ( Math.abs(nodusx+wx)<150 && Math.abs(nodusx+xxx)<150 ) {
            if ( Math.abs(nodusx+wx)<130 && Math.abs(nodusx+xxx)<130 ) {
              makeLine(  nodusx + wx,nodusy + wy   ,   nodusx + xxx, nodusy + yyy  )
            }
         }
   }
   wx = xxx
   wy = yyy
}



/*#############################################################################
  #  display key design data                                                  #
  #############################################################################
*/

makeText("Horizontal Dial v13402" , -140,110 , 4)

makeText("Latitude: " , -140,100 , 4)
makeText( lat, -100,100 , 4)

/*makeText("Longitude: "+String(lng) , -140,90 , 4) */
makeText("Longitude: " , -140,90 , 4)
makeText( lng, -100,90 , 4)

makeText("Legal: " , -140, 80 , 4)
makeText( ref, -100, 80 , 4)
```

```
/*##########################################################################################
  #  Draw a box to bound the depiction                                                     #
  ##########################################################################################
*/

/* for no good reason, using NanoCAD native javascript commands here, could have used
"makeLine"*/
ptb = "-150,-80,0"
pte = "+150,-80,0"
ThisDrawing.ModelSpace.AddLine(ptb,pte)
ptb = "-150,130,0"
pte = "+150,130,0"
ThisDrawing.ModelSpace.AddLine(ptb,pte)
ptb = "-150,-80,0"
pte = "-150,130,0"
ThisDrawing.ModelSpace.AddLine(ptb,pte)
ptb = " 150,-80,0"
pte = " 150,130,0"
ThisDrawing.ModelSpace.AddLine(ptb,pte)

ThisDrawing.Application.ZoomAll()
ThisDrawing.Utility.Prompt("*** END ***")

/*
 END
 */
```



Measuring an angle is simple. DIMENSIONS on the top row, then ANGLE DIMENSION and then move the mouse to a line, click, and to the next line, click, and the angle is displayed.