## Adapting DOE-2 to Meet the Needs of the Egyptian Housing and Building Research Center

**Joe Huang**

For the past two years, the author has been working with Joe Deringer of the Deringer Group in Berkeley and Prof. Moncef Krarti of the University of Colorado in providing technical assistance to the Housing and Building Research Center (HBRC) of Egypt in developing building energy codes for that country (see http://www.icbec.org/Egypt/EgyptHome.htm for details).

The international team recommended DOE-2 as the simulation tool and provided training in its use both in Egypt and the U.S. The DOE-2 training was done first using a user-friendly version with a graphic interface (*VisualDOE*) and then migrated to a standard version of DOE-2.1E as the HBRC staff gained more familiarity with the text-based input and output. To facilitate running parametric DOE-2 simulations, the Deringer Group developed a simple but effective Windows-based procedure called *DOE2Parm* (see http://www.icbec.org/Software/DOE2Parm.htm for details). The Egyptian team also added a post-processor using MS Excel macros to assist in analyzing results for the Egyptian application

The following two articles describe specialized applications of DOE-2 developed in response to the needs of the project that may be of interest to others working in building energy simulation.

### Modeling Thermal Comfort Using the DOE-2 Program

Since many residential buildings in Egypt might not have mechanical air-conditioning, HBRC wanted to know what were the indoor thermal conditions when only natural ventilation or ceiling fans were available, and what improvements could be expected if building standards were adopted.

To evaluate indoor thermal comfort conditions, a DOE-2 Function was written to derive the Predicted Percent Dissatisfied (PPD) and Predicted Mean Value (PMV) based on the Fanger Model. The calculations in this Function follow the derivation given in a paper by one of the team members (Holz et al. 1996). The Fanger Model, like other comfort models, requires as input the Mean Radiant Temperature of the space. DOE-2, however, calculates only the indoor air temperature and not the inside surface temperatures on which the MRT is based. To address this deficiency, a researcher from EMPA, the Swiss Federal Laboratories for Materials Testing and Research, in 1999 added a module to DOE-2.1E that back-calculated the inside surface temperatures and then, from these, calculated the Mean Radiant and Effective Temperatures. This modification was described in the Summer 1999 issue of the User News (Koschenz 1999).

One of the most problematic aspects in writing a DOE-2 Function is determining where a program variable of interest can be accessed. This problem is especially tricky when the variable, i.e., Mean Radiant Temperature, is a later addition that is calculated after the DOE-2 SYSTEM calculations for each hourly time step have been completed. Since there are no Function access points within the EMPA modifications, the only way to access the Mean Radiant and Effective Temperatures is to write them to a SYSTEMS hourly-report, and have the function loop through the HOURLY-REPORT array at the conclusion of each time step to pick up the temperatures.

Therefore, to run properly the PPMV Function shown below requires the two following input changes :

1. Option for calculating Inside Surface Temperatures is turned on, i.e., under the BUILDING-LOCATION command, add SURF-TEMP-CALC = YES
2. The Mean Radiant Temperature (Variable 91) and Effective Temperature (Variable 92) for each zone of interest are written to an hourly-report file. A sample hourly-report input for a two-zone building is shown below:

```
 HRSCH  SCHEDULE  THRU DEC 31 (ALL) (1,24) VALUES=(1)  ..
RB1 REPORT-BLOCK VARIABLE-TYPE=ZONE_1 VARIABLE-LIST=(91,92) ..
RB2 REPORT-BLOCK VARIABLE-TYPE=ZONE_2 VARIABLE-LIST=(91,92) ..
SHR HOURLY-REPORT REPORT-SCHEDULE=HRSCH
REPORT-BLOCK=(RB1,RB2) ..
```

Like other functions, the PPMV.func function should be inserted in the SYSTEMS input after the END command but before the COMPUTE SYSTEM command. If everything is set up correctly, the DOE-2 output will contain the PPD and PMV every hour for each zone, so users should be prepared for rather large output files. In addition, whenever DOE-2 is used with SURF-TEMP-CALC=YES, it generates a huge fort.15 file and a smaller fort.16 file. These should be deleted as they are temporary binary file with no further use.

If there are any questions in the use of this function, please e-mail me at YJHuang@lbl.gov.

**Function ppmv.func follows ----------**

```
FUNCTION NAME=PPMV ..
ASSIGN MON=IMO DAY=IDAY HR=IHR
       INILZE=INILZE
       IP=IP
       NPLSYS=NPLSYS
       IPLSYS=IPLSYS
       ISZONES=ISZONES
       NZONES=NZONES
       NPL=NPL
       NSP=NSP
       ZP1=ZP1
       ZP2=ZP2
       ZP2_EDTT=ZP2--EDTT
       IZNBUFP=IZNBUFP
       NZD=NZD
       ZNAM1=ZONE-NAME
       ta=TNOW
       VNTCFM=VNTCFM
       CFMZ=CFMZ
       Wa=HUMRAT
       Patm=PATM
    $  VS=WNDSPD     using outside wind speed as interim air speed
       VS=0
       Msi=metabolic_rate[] $ W/m2 range 57-95
       Wsi=work_rate[]      $ W/m2 generally 10% of above
       IPFLAG=PPMV_print[]  $ 0=detailed, 1=hourly
       IclWIN=clo_win[]     $ fraction
       IclSUM=clo_sum[]     $ fraction
       fanon=ceilfan_stp[]  $ ceil fan on temp (C)
       acon=aircond_stp[]   $ A/C on temp (C)
       ..
CALCULATE ..
      NPL=IP
      NS=0
100   NS=NS+1
      IF (NS.GT.NPLSYS) GO TO 199
      NSP=IACCESS(IPLSYS+NS-1)
      ZP1=ISZONES
      I=0
200   I=I+1
      IF (I.GT.NZONES) GO TO 299
      ZP2=ZP2_EDTT
      IPTR=IZNBUFP-1
      TMRX=ACCESS(IPTR+91)
      WZONEX=ACCESS(IPTR+93)
C     PRINT 5,CFMZ,VNTCFM
5     FORMAT(2F10.2)
C     ZP1=ZP1+NZD
C
C   PPMV follows
```

```
C
C  set Metabolic and clothing values
C
      IF (MON.LT.5) Icl=IclWin
      IF (MON.GE.5.AND.MON.LE.9) Icl=IclSum
      IF (MON.GT.9) Icl=IClWin

C  convert units from SI to IP
      M = Msi/3.1536
      W = Wsi/3.1536
      tac = (ta-32)/1.8
      tmrc = (tmrx-32)/1.8
      pasi = 689.5*Patm

C  assume ceiling fans are used when air temperature is
C  between ceilfan (fanon) and a/c (acon) setpoints
C  and provides 0.25 m/sec (50 fpm) air motion

      VS = 0
      if (tac.gt.fanon.and.tac.le.acon) VS = 50
      vssi = VS*0.005

C  calculate (M-W) and Rcl
      MW = M - W
      Rcl = 0.88*Icl
C  convert inHg to PSI and calculate Pa
      Patmx=Patm*0.49116
      Pa = Patmx/(1.0+0.622/Wa)
C  calculate tcl
      tcl = MW - 0.97*(5.73-0.22*MW-6.9*Pa) - 0.42*(MW-18.43)
      tcl = tcl - 0.0173*M*(5.87-6.9*Pa) - 0.00077*M*(93.2-ta)
      tcl = 96.3 - 0.156*MW - Rcl*tcl
C  calculate (tcl+460)**4
      tcl4 = 4*alog(tcl+460)
      tcl4 = exp(tcl4)
C  calculate (tmrx+460)**4
      tmrx4 = 4*alog(tmrx+460)
      tmrx4 = exp(tmrx4)
C  calculate hc
      tcl_ta = 0.25*alog(abs(tcl-ta))
      tcl_ta = 0.361*exp(tcl_ta)
      if (VS.GT.0) vsqrt = 0.151*sqrt(VS)
      if (VS.EQ.0) vsqrt = 0
      if (tcl_ta.gt.vsqrt) hc = tcl_ta
      if (tcl_ta.le.vsqrt) hc = vsqrt
C  calculate fcl
      if (Icl.le.0.5) fcl = 1.0 + 0.2*Icl
      if (Icl.gt.0.5) fcl = 1.05 + 0.1*Icl
C  calculate L
      L = MW - 1.196*0.000000001*fcl*(tcl4 - tmrx4)
      L = L - fcl*hc*(tcl-ta)-0.97*(5.73-0.022*MW) - 6.9*Pa
      L = L - 0.42*(MW-18.43) - 0.0173*M*(5.87-6.9*Pa)
      L = L - 0.00077*M*(93.2-ta)
C  convert to SI
      tclc = (tcl-32)/1.8
      Lsi = L*3.1536
C  calculate PMV and PPD
      PMV = (0.303*exp(-0.036*M) + 0.028)*L
      PMV2 = PMV*PMV
      PMV4 = PMV2*PMV2
      PPD = 100 - 95*exp( -1.0*(0.03353*PMV4 + 0.2179*PMV2) )
      IF (INILZE.LT.8) RETURN
      ICOUNT = ICOUNT + 1
      IF (IPFLAG.GT.0) GO TO 205
      IF (ICOUNT.EQ.1) PRINT 1
  1   FORMAT(/13X,4HZone,4x,2HIS,1x,2HIZ,3x,2HTa,2x,3HTmr,
     +        3x,2HWa,4x,2HWS,3x,2HPa,4x,3Htcl,4x,
     +        1HL,5x,3HPMV,3x,3HPPD)
      PRINT 20,MON,DAY,HR,ACCESS(ZP2),ACCESS(ZP2+1),NS,I,
```

```
     +         tac,tmrc,Wa,vssi,pasi,tclc,Lsi,PMV,PPD
 20   FORMAT(3F4.0,1X,2A4,F3.0,F3.0,2F5.1,F6.3,F5.2,
     +       f7.0,f5.1,f7.1,f6.1,f6.1)
      GO TO 210
205   CONTINUE
      IF (IPFLAG.GT.1) GO TO 207
      IF (ICOUNT.EQ.1) PRINT 2
  2   FORMAT(/14X,4HZone,5x,2HIS,2x,2HIZ,2x,2HTa,4x,3HTmr,
     +        5x,3HPMV,3x,3HPPD)
      PRINT 22,MON,DAY,HR,ACCESS(ZP2),ACCESS(ZP2+1),NS,I,
     +         tac,tmrc,PMV,PPD
 22   FORMAT(1X,3F4.0,1X,2A4,1x,F2.0,1x,F3.0,2F6.1,f7.2,f6.1)
      GO TO 210
207   CONTINUE
C     monthly hourly print not implemented !
C     store monthly hourly values
C     data(HR) = data(HR) + PMV
210   CONTINUE
      ZP1=ZP1+NZD
      GOTO 200
299   CONTINUE
      GOTO 100
199   CONTINUE
      END
```

**Modeling domes in DOE-2**

In another application, the Egyptian researchers expressed the need to model domes and vaults, which are very prevalent in Egypt. In response, I wrote an AWK script that generates DOE-2 BDL input lines for a dome composed of A rings, each containing A*4 quadrilateral polygons, topped of by one flat polygon with A*4 sides. The AWK script also writes BDL input lines for two horizontal polygons making up the remaining flat portion of a square roof of length B on each side surrounding the base, which I've called the "skirt". (For more information on the AWK programming language, see Aho, Kernighan, and Weinberger 1988)

The AWK script reads a small input file (dome.dat) that contains one line per dome with the following 5 numbers: dome radius, number of rings (A), length on each side of the square roof, and the XY coordinates for the dome center. For example, Figure 1 shows the result for a dome with a radius of 5 made up of 3 rings, a flat roof of length 15 on each side, and located at the origin, i.e., input as 5 3 15 0 0. Figure 2 shows a more detailed model of the same dome made up of 8 rings, i.e., input as 5 8 15 0 0, clearly a case of modeling overkill.

The syntax to generate the DOE-2 BDL input lines is

awk -f dome.awk dome.dat > dome.bdl

where dome.awk is the AWK file, dome.dat the one-line input file, and dome.bdl the output BDL snippet.. Dummy "INPUT LOADS ..", "END ..", and "STOP.." commands have been added at the beginning and end of the files so that they can be displayed using the *DrawBDL 3.0* program.

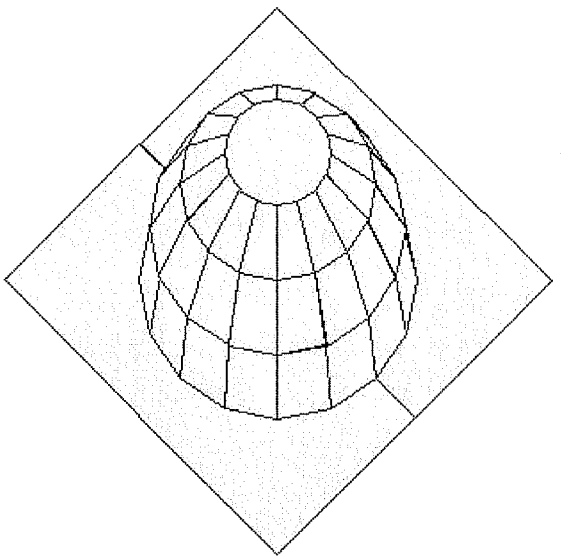

Figure 1

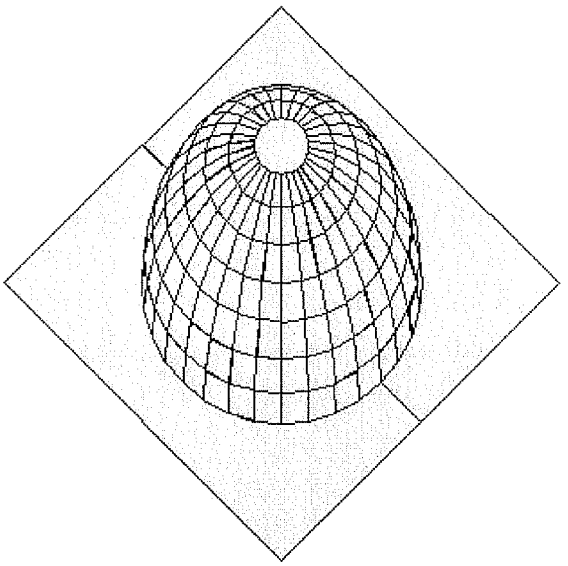

Figure 2

## Listing for dome.awk ----------

```
#
# this awk script generates DOE-2 BDL input for a dome composed of "arcs" x "rows"
# of quadrilateral polygons and one polygon with "arcs" sides on the top .
# syntax is awk -f dome.awk dome.dat > dome.bdl where dome.dat is a file
# with 1 line per dome, each line containing 5 numbers for the dome radius, number
# of rings (integer) for dome, length on each side of square roof, and XY coordinates
# at the dome center.
#
#                                done by Joe Huang 22 July 02
{angle = 2*3.14159263/360.}
{r = $1 ; rows = $2; rflgth = $3}
{arcs = $2*4 ; ang1 = 360/arcs ; ang2 = ang1/2 }
{if (NR==1) printf("INPUT LOADS ..\nBUILDING-LOCATION  AZ 0 ..\n") }
{printf("DOME%d SPACE X %1.2f Y %1.2f ..\n",NR,$4,$5) }
{for (i=1;i<=arcs;i++) az[i]=ang1*i - ang2 }
{rowlgth =2*r*sin(360*angle/(arcs*2) )}
{for (i=1;i<=rows;i++)
  {
   rowtilt[i] = (90+ang2)-(ang1*i)
   rowht[i] = r*sin(rowtilt[i]*angle)
   rowz[i] = r*sin((i-1)*ang1*angle)
   rowwd[i] = 2*r*sin(ang2*angle)*cos((i-1)*ang1*angle)
  }
}
{for (i=1;i<=rows;i++)
  {
   j=i+1
   rowstart[i] = (rowwd[i] + rowwd[j])/2
   rowend[i] = (rowwd[i] - rowwd[j])/2
   printf("SET-DEFAULT FOR ROOF Z %1.2f TILT %d ..\n",rowz[i],rowtilt[i])
   if (i < rows )
     {
      for (k=1;k<=arcs;k++)
        {
         rowx[i,k] = r*sin(ang1*k*angle)*cos((i-1)*ang1*angle)
         rowy[i,k] = r*cos(ang1*k*angle)*cos((i-1)*ang1*angle)
         if (k == 1)
           {
            printf("DOME%d%d-%dP POLYGON (0,0) (%1.2f,0) (%1.2f,%1.2f) (%1.2f,%1.2f)
..\n",NR,i,k,rowwd[i],rowstart[i],rowlgth,rowend[i],rowlgth)
```

```
            printf("DOME%d%d-%d ROOF POLYGON DOME%d%d-%dP X %1.2f Y %1.2f AZ %d
..\n",NR,i,k,NR,i,k,rowx[i,k],rowy[i,k],az[k])
            }
          else
            printf("DOME%d%d-%d ROOF LIKE DOME%d%d-1 X %1.2f Y %1.2f AZ %d
..\n",NR,i,k,NR,i,rowx[i,k],rowy[i,k],az[k])
        }
      }
    else
      {
       printf("DOME%d%d-1P POLYGON  ",NR,i)
    #  for (k=1;k<=arcs;k++)
       for (k=arcs;k>=1;k--)
         {
          rowx[i,k] = r*sin(ang1*k*angle)*cos((i-1)*ang1*angle)
          rowy[i,k] = r*cos(ang1*k*angle)*cos((i-1)*ang1*angle)
          l = k-1
          if (l%5 == 0) printf "\n  "
          printf("(%1.2f,%1.2f) ",rowx[i,k],rowy[i,k])
    #     if (k== arcs) printf " .. \n"
          if (k== 1) printf " .. \n"
         }
       printf("DOME%d%d-1 ROOF POLYGON DOME%d%d-1P X 0  Y 0  AZ 0 TILT 0 .. \n",NR,i,NR,i)
      }
  }
}
END {
printf("SKIRT-EP POLYGON (0,%d) (0,%d) (%d,%d) (%d,%d) (0,%d) (0,%d)\n",-r,-rflgth/2,rflgth/2,-
rflgth/2,rflgth/2,rflgth/2,rflgth/2,r)
halfarc=arcs/2 - 1
for (k=1;k<=halfarc;k++)
  {
   rowx[1,k] = r*sin(ang1*k*angle)
   rowy[1,k] = r*cos(ang1*k*angle)
   l = k-1
   if (l%5 == 0 && k > 1) printf "\n  "
   printf("(%1.2f,%1.2f) ",rowx[1,k],rowy[1,k])
   if (k == halfarc) printf " .. \n"
  }
printf("SKIRT-E ROOF POLYGON SKIRT-EP X 0 Y 0 Z 0 AZ 0 TILT 0 .. \n")
printf("SKIRT-WP POLYGON (0,%d) (0,%d) (%d,%d) (%d,%d) (0,%d) (0,%d)\n",r,rflgth/2,-
rflgth/2,rflgth/2,-rflgth/2,-rflgth/2,-rflgth/2,-r)
for (k=halfarc;k<=arcs;k++)
  {
   rowx[1,k] = r*sin(ang1*k*angle)
   rowy[1,k] = r*cos(ang1*k*angle)
   l = k-1
   if (l%5 == 0) printf "\n  "
   printf("(%1.2f,%1.2f) ",rowx[1,k],rowy[1,k])
   if (k== arcs) printf " .. \n"
  }
printf("SKIRT-W ROOF POLYGON SKIRT-WP X 0 Y 0 Z 0 AZ 0 TILT 0 .. \n")
print "END .."
print "STOP .."
}
```

## Listing for dome.dat ----------

```
5 3 15 0 0
```

## Listing for dome.bdl ----------

```
INPUT LOADS ..
BUILDING-LOCATION  AZ 0 ..
DOME1 SPACE X 0.00 Y 0.00 ..
SET-DEFAULT FOR ROOF Z 0.00 TILT 78 ..
DOME11-1P POLYGON (0,0) (1.95,0) (1.88,1.95) (0.07,1.95) ..
DOME11-1 ROOF POLYGON DOME11-1P X 1.91 Y 4.62 AZ 11 ..
DOME11-2 ROOF LIKE DOME11-1 X 3.54 Y 3.54 AZ 33 ..
DOME11-3 ROOF LIKE DOME11-1 X 4.62 Y 1.91 AZ 56 ..
DOME11-4 ROOF LIKE DOME11-1 X 5.00 Y 0.00 AZ 78 ..
```

```
DOME11-5 ROOF LIKE DOME11-1 X 4.62 Y -1.91 AZ 101 ..
DOME11-6 ROOF LIKE DOME11-1 X 3.54 Y -3.54 AZ 123 ..
DOME11-7 ROOF LIKE DOME11-1 X 1.91 Y -4.62 AZ 146 ..
DOME11-8 ROOF LIKE DOME11-1 X 0.00 Y -5.00 AZ 168 ..
DOME11-9 ROOF LIKE DOME11-1 X -1.91 Y -4.62 AZ 191 ..
DOME11-10 ROOF LIKE DOME11-1 X -3.54 Y -3.54 AZ 213 ..
DOME11-11 ROOF LIKE DOME11-1 X -4.62 Y -1.91 AZ 236 ..
DOME11-12 ROOF LIKE DOME11-1 X -5.00 Y -0.00 AZ 258 ..
DOME11-13 ROOF LIKE DOME11-1 X -4.62 Y 1.91 AZ 281 ..
DOME11-14 ROOF LIKE DOME11-1 X -3.54 Y 3.54 AZ 303 ..
DOME11-15 ROOF LIKE DOME11-1 X -1.91 Y 4.62 AZ 326 ..
DOME11-16 ROOF LIKE DOME11-1 X -0.00 Y 5.00 AZ 348 ..
SET-DEFAULT FOR ROOF Z 1.91 TILT 56 ..
DOME12-1P POLYGON (0,0) (1.80,0) (1.59,1.95) (0.21,1.95) ..
DOME12-1 ROOF POLYGON DOME12-1P X 1.77 Y 4.27 AZ 11 ..
DOME12-2 ROOF LIKE DOME12-1 X 3.27 Y 3.27 AZ 33 ..
DOME12-3 ROOF LIKE DOME12-1 X 4.27 Y 1.77 AZ 56 ..
DOME12-4 ROOF LIKE DOME12-1 X 4.62 Y 0.00 AZ 78 ..
DOME12-5 ROOF LIKE DOME12-1 X 4.27 Y -1.77 AZ 101 ..
DOME12-6 ROOF LIKE DOME12-1 X 3.27 Y -3.27 AZ 123 ..
DOME12-7 ROOF LIKE DOME12-1 X 1.77 Y -4.27 AZ 146 ..
DOME12-8 ROOF LIKE DOME12-1 X 0.00 Y -4.62 AZ 168 ..
DOME12-9 ROOF LIKE DOME12-1 X -1.77 Y -4.27 AZ 191 ..
DOME12-10 ROOF LIKE DOME12-1 X -3.27 Y -3.27 AZ 213 ..
DOME12-11 ROOF LIKE DOME12-1 X -4.27 Y -1.77 AZ 236 ..
DOME12-12 ROOF LIKE DOME12-1 X -4.62 Y -0.00 AZ 258 ..
DOME12-13 ROOF LIKE DOME12-1 X -4.27 Y 1.77 AZ 281 ..
DOME12-14 ROOF LIKE DOME12-1 X -3.27 Y 3.27 AZ 303 ..
DOME12-15 ROOF LIKE DOME12-1 X -1.77 Y 4.27 AZ 326 ..
DOME12-16 ROOF LIKE DOME12-1 X -0.00 Y 4.62 AZ 348 ..
SET-DEFAULT FOR ROOF Z 3.54 TILT 33 ..
DOME13-1P POLYGON (0,0) (1.38,0) (1.06,1.95) (0.32,1.95) ..
DOME13-1 ROOF POLYGON DOME13-1P X 1.35 Y 3.27 AZ 11 ..
DOME13-2 ROOF LIKE DOME13-1 X 2.50 Y 2.50 AZ 33 ..
DOME13-3 ROOF LIKE DOME13-1 X 3.27 Y 1.35 AZ 56 ..
DOME13-4 ROOF LIKE DOME13-1 X 3.54 Y 0.00 AZ 78 ..
DOME13-5 ROOF LIKE DOME13-1 X 3.27 Y -1.35 AZ 101 ..
DOME13-6 ROOF LIKE DOME13-1 X 2.50 Y -2.50 AZ 123 ..
DOME13-7 ROOF LIKE DOME13-1 X 1.35 Y -3.27 AZ 146 ..
DOME13-8 ROOF LIKE DOME13-1 X 0.00 Y -3.54 AZ 168 ..
DOME13-9 ROOF LIKE DOME13-1 X -1.35 Y -3.27 AZ 191 ..
DOME13-10 ROOF LIKE DOME13-1 X -2.50 Y -2.50 AZ 213 ..
DOME13-11 ROOF LIKE DOME13-1 X -3.27 Y -1.35 AZ 236 ..
DOME13-12 ROOF LIKE DOME13-1 X -3.54 Y -0.00 AZ 258 ..
DOME13-13 ROOF LIKE DOME13-1 X -3.27 Y 1.35 AZ 281 ..
DOME13-14 ROOF LIKE DOME13-1 X -2.50 Y 2.50 AZ 303 ..
DOME13-15 ROOF LIKE DOME13-1 X -1.35 Y 3.27 AZ 326 ..
DOME13-16 ROOF LIKE DOME13-1 X -0.00 Y 3.54 AZ 348 ..
SET-DEFAULT FOR ROOF Z 4.62 TILT 11 ..
DOME14-1P POLYGON
  (-0.00,1.91) (-0.73,1.77) (-1.35,1.35) (-1.77,0.73) (-1.91,-0.00)
  (-1.77,-0.73) (-1.35,-1.35) (-0.73,-1.77) (0.00,-1.91) (0.73,-1.77)
  (1.35,-1.35) (1.77,-0.73) (1.91,0.00) (1.77,0.73) (1.35,1.35)
  (0.73,1.77)  ..
DOME14-1 ROOF POLYGON DOME14-1P X 0  Y 0  AZ 0 TILT 0 ..
SKIRT-EP POLYGON (0,-5) (0,-7) (7,-7) (7,7) (0,7) (0,5)
(1.91,4.62) (3.54,3.54) (4.62,1.91) (5.00,0.00) (4.62,-1.91)
  (3.54,-3.54) (1.91,-4.62)  ..
SKIRT-E ROOF POLYGON SKIRT-EP X 0 Y 0 Z 0 AZ 0 TILT 0 ..
SKIRT-WP POLYGON (0,5) (0,7) (-7,7) (-7,-7) (0,-7) (0,-5)
(1.91,-4.62) (0.00,-5.00) (-1.91,-4.62) (-3.54,-3.54)
  (-4.62,-1.91) (-5.00,-0.00) (-4.62,1.91) (-3.54,3.54) (-1.91,4.62)
  (-0.00,5.00)  ..
SKIRT-W ROOF POLYGON SKIRT-WP X 0 Y 0 Z 0 AZ 0 TILT 0 ..
END ..
STOP ..
```

## References

Aho, A., Kernighan, B., and Weinberger, P. 1988. *The AWK Programming Language*, Addison-Wesley, Boston, MA.

Koschenz, Markus 1999. "Surface Temperature Calculation in DOE-2.1E", *Building Energy User News*, Vol. 20, No. 2 (Summer 1999), pp 4-7, Lawrence Berkeley National Laboratory, Berkeley CA.

Holz, R., A. Hourigan, R. Sloop, P. Monkman, and M. Krarti 1996. "Effects of Standard Energy Conserving Measures on Thermal Comfort", *Building and Environment*, Vol. 32, No. 1, pp. 31-43, Pergamon, London UK.