APPENDIX C

TYPE 67 (Simple Model): Description, TRNSYS Decks and Code

DESCRIPTION:

TYPE 67 models a NCHE water loop based upon provided experimental data. TYPE 67 takes into account both the heat transfer and pressure effects in order to predict the water flow rate, the HX outlet water temperatures, and the heat transfer in the heat exchanger. TYPE 67 was written for manufacturers and researchers, so that given pressure drop and heat transfer data on a particular NCHE, a SDHW system's performance can be predicted.

Requirements:

In order to run TYPE 67, the following inputs must be specified:

1) Type of and percent glycol solution used in collector loop (either propylene glycol or ethylene glycol solutions may be used.)

2) Dimensions of all piping in water loop: length, diameter, change in elevation.

3) Good estimate of minor losses for piping fittings, valves and entrance, exit conditions.

4) HX pressure drop as a function of water flow rate data file (curve).

5) HX modified effectiveness as a function of water flow rate data file (curve).

1) Dimensions of all piping in water loop

In describing the physical aspects of the SDHW system, the following diagram should make clear the various parameter inputs:

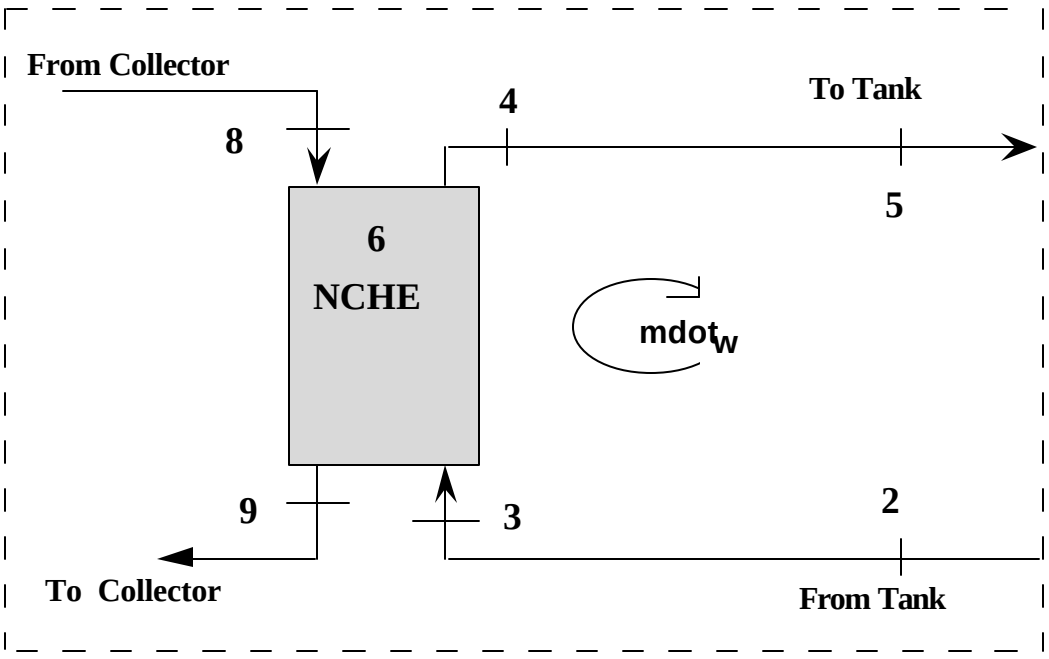

Figure C.1 Diagram of NCHE and associated piping.

TYPE 67 allows for two different diameter pipes in the piping runs from tank to NCHE and from NCHE to the tank. If only one diameter pipe is used from the tank to the NCHE, then input 0's into the TRNSYS deck for the respective positions. The list of parameter inputs follows.

Care must be taken when specifying *Dz* values as parameters. If $\sum_{water\ loop} \Delta z \neq 0$ then TRNSYS will deliver strange answers. That is, the following equation should hold:

$$\Delta z_2 + \Delta z_3 + \Delta z_4 + \Delta z_5 + \Delta z_{NCHE} + \Delta z_{Tank} = 0 \qquad \text{(C.1)}$$

where Δz_{Tank} is the distance from the tank inlet piping to the tank outlet piping.

2) Minor loss coefficients for water loop fittings

The minor losses can be found either experimentally, with Hooper's correlations for fitting pressure losses (Kakic 1987), or from table values.

3) HX pressure drop and modified effectivness curves

TYPE 67 necessitates some experimental work in order to find the pressure drop and modified effectiveness as a function of water flow rate. The procedure for finding these curves is outlined in Fraser (1992). The pressure drop and modified effectiveness curves must then be placed into seperate data files in the following forms. The pressure drop data should be written with flow rates in the first column, and the pressure drops in the following column:

Table C.1 Sample NCHE Shear Pressure Drop vs. Water Flow Rate Data File

0	0
0.005	0.00298
0.01	0.0084
0.015	0.01627
0.02	0.0266
0.025	0.03938
0.03	0.0546
0.035	0.07228
0.04	0.0924
0.045	0.11498
0.05	0.14
2000	188000540

It is necessary to put in as a top row the case for zero flow rate, as well, it is necessary to put in as the bottom row, a very large number for the flow rate so that TYPE 67 can interpolate. It may be necessary to find a function from the experimental pressure drop results and to extend the function to a high flow rate value for the last data file row, (providing it is smooth and exponential or linear in nature--not bumpy). TYPE 67 cannot extrapolate.

The form for the effectiveness data file is different as modified effectiveness is a function of two variables, the glycol and water flow rates, whereas the NCHE pressure drop is a function of only one variable, the water flow rate. TYPE 67 can read in modified effectiveness results for several glycol flow rate tests. The data file, should read like Table C.2. The top row should have a zero first, then the glycol flow rates at which each test was done. The first column starts with the zero previously mentioned, and lists water flow rates in ascending order. The last entry in the first column should be a very large number, as TYPE 67 cannot extrapolate. The rest of the table values should be modified effectiveness values for the corresponding water and glycol flow rates.

Table C.2 Chart Showing Data Placement for Modified Effectiveness Data File

0	1st test mdot_g	2nd test mdot_g	...	nth test mdot_g
mdot_w	eff'	eff'	eff'	eff'
mdot_w	eff'	eff'	eff'	eff'
mdot_w	eff'	eff'	eff'	eff'
mdot_w	eff'	eff'	eff'	eff'
mdot_w	eff'	eff'	eff'	eff'
mdot_w	eff'	eff'	eff'	eff'
mdot_w	eff'	eff'	eff'	eff'
mdot_w	eff'	eff'	eff'	eff'
9999999999	eff'	eff'	eff'	eff'

The table values used in writing and evaluating the model are shown below in Table C.3. The modified effectiveness values shown in Table C.3 come from Fraser's experimental work.

Table C.3 Sample Modified Effectiveness Data File

0	0.01	0.02	0.03
0	0	0	0
0.005	0.44	0.265	0.18
0.01	0.82	0.53	0.335
0.015	0.89	0.695	0.49
0.02	0.925	0.78	0.61
0.025	0.94	0.83	0.69
0.03	0.94	0.84	0.73
0.035	0.94	0.84	0.75
9999999999	0.94	0.84	0.75

Units for the data files should be:

flow rate	-	kg/s
pressure loss	-	kPa

The data files are assigned (linked) at the top of the TRNSYS deck.

3) Regarding the Tank Model

As TYPE 67 requires as an input the static pressure gain in the tank, the TYPE 4 tank model must be ammended or an alternate tank model (that outputs the static tank pressure gain) must be used in place of TYPE 4. The changes that must be made to the TYPE 4 code are listed below. Changes or additions to the code are in boldface.

```
310   DTDT(K) = (S(KF) - S(KI))/DELT
      TF = TF/HIGH

320   QTANK = FLWL*(S(AVG+1) - TL)
      QIN=FLWS*(TIN-S(AVG+NEQ))
      DELAU=(TF-S(IS+1))*MSCP
C-------------------------------------------------
C  COMPUTE STATIC HEAD LOSS
C-------------------------------------------------
        G=9.806
      if (n.lt.2) then
         Rho=(999.8396 + 18.224944*TF
     @              - 0.00792221*TF**2
     @              -55.44846e-6*TF**3
     @              +149.7562e-9*TF**4
     @              -393.2952e-12*TF**5)
     @              /(1.d0+18.159725e-3*TF)

        else
           rhot=0.d0
           DO  I=2,N
              Rhot=rhot+(999.8396 + 18.224944*s(avg+i)
     @              - 0.00792221*s(avg+i)**2
     @              -55.44846e-6*s(avg+i)**3
     @              +149.7562e-9*s(avg+i)**4
     @              -393.2952e-12*s(avg+i)**5)
     @              /(1.d0+18.159725e-3*s(avg+i))

           enddo
```

```
          rho=rhot/(n-1)
        endif
          STHEAD=RHO*G*HIGH*.001
C-----------------------------------------------

C  SET OUTPUTS
 330  OUT(1)=S(AVG+NEQ)
      OUT(2)=FLWSS
      OUT(3)=S(AVG+1)
      OUT(4)=FLWLL
      OUT(5)=QENV+QVENT
      OUT(6)=QTANK
      OUT(7)=DELAU
      OUT(8)=QBTOT(1)+QBTOT(2)
      OUT(9)=QBTOT(1)
      OUT(10)=QBTOT(2)
      OUT(11)=QIN
      OUT(12)=TF
     OUT(13)=STHEAD


      N=NEQ-1
      IF(N.LT.2) RETURN 1
      DO 350 I=2,N
350   OUT(12+I)=S(AVG+I)
      RETURN 1

1001  FORMAT(//2X,'***** WARNING *****',/2X,'THE SET POINT TEMERATURE OF
     1 THE UNIT ',I2,' TYPE 4 STORAGE TANK IS HIGHER THAN'/2X,'THE BOILI
     1NG TEMPERATURE.  THE BOILING TEMPERATURE WILL BE USED AS THE SET',
     1/2X,'POINT.')

      END
```

4) The Convergence Promoter

TYPE 67 requires a TYPE 44 convergence promoter in order to reach convergence at some time steps. The convergence promoter should be set to vary the glycol inlet temperature of the NCHE, (the first input to NCHE).

TRNSYS Component Configuration

PARAMETER NO.		DESCRIPTION
1	NX_{hx}	Number of nodes for HX temperature distribution
2	m_g	Flow rate in collector loop [kg/s]
3	H_{hx}	Height of HX [m]
Length of Pipe [m]:		
4	L(2)	of pipe section 2
5	L(3)	of pipe section 3
6	L(4)	of pipe section 4
7	L(5)	of pipe section 5
ΔZ of Pipe [m]:		
8	dz(2)	of pipe section 2
9	dz(3)	of pipe section 3
10	dz(4)	of pipe section 4
11	dz(5)	of pipe section 5
12	dz(6)	of water storage tank
Diameter of Pipe [m]:		
13	D(2)	of pipe section 2
14	D(3)	of pipe section 3
15	D(4)	of pipe section 4
16	D(5)	of pipe section 5
Associated Minor Loss K Values [-]:		
17	K(2)	of pipe section 2
18	K(3)	of pipe section 3
19	K(4)	of pipe section 4
20	K(5)	of pipe section 5
Lookup Tables of Effectiveness, HX Pressure Drop vs. Water Flow Rate, Glycol Type		
21	$N_{pts,eff}$	# points in modified effectiveness vs m_w curve for a given glycol flow rate

22	$N_{pts,dphx}$	# points in HX pressure drop vs m_w curve
23	%	% glycol solution [integer]
24	N_{tests}	# curves of modified effectiveness vs m_w (# glycol flow rates tested for)
25	Glycol	Glycol Type: 1) propylene glycol solution 2) ethylene glycol solution

INPUT NUMBER		**DESCRIPTION**
1	$T_{w,i}$	Temperature of inlet water stream to HX [°C]
2	$T_{g,i}$	Temperature of inlet glycol stream to HX [°C]
3	$\Delta P_{st,tank}$	Static pressure head in tank [kPa]
4	γ	Control Function from controller

OUTPUT NUMBER		**DESCRIPTION**
1	$T_{w,o}$	Water outlet temperature [°C]
2	$T_{g,o}$	Glycol outlet temperature [°C]
3	Q_{hx}	Energy transfer rate in HX [kJ/hr]
4	e'	HX modified effectiveness
5	m_w	Water flow rate [kg/hr]
6	$\sum \Delta P_{static}$	Total static presure drop around water loop [kPa]
7	$\sum \Delta P_{shear}$	Total shear pressure drop around water loop [kPa]
8	Error	Error = $\sum \Delta P_{static} - \sum \Delta P_{shear}$

Information Flow Diagram

Inputs	-	4
Outputs	-	8
Parameters	-	25
Derivatives	-	0

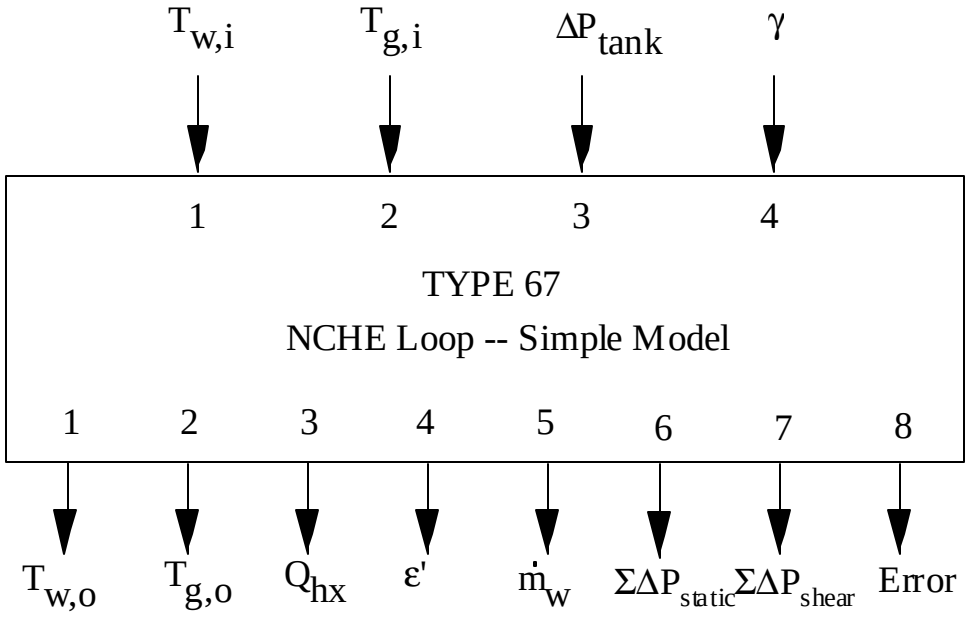

Figure C.2 TRNSYS component diagram for simple model.

DECKS:

1) Steady State Deck

```
ASSIGN STEADYSTATE.OUT   14
ASSIGN LOOKUP.DPHX       15
ASSIGN LOOKUP.EFF        16
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                         *
*                           steady state                                  *
*               NATURAL CONVECTION HEAT EXCHANGER SETUP                   *
*                            MARCH 1994                                   *
*                            simple deck                                  *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
limits 100 100
SIMULATION 25. 90. 1
WIDTH 72
tol -0.0001 0.0001
```

```
equations 3
THXWI=19
THXGI=TIME
MWS = [68,2]/3600

Equations 25 ------------System Parameters-----------------------------
mdotg=.02*3600
*
* vertical rise in piping loop
*
dz2=0
dz3=0.064
dz4=0
dz5=0.965
hthx=16*2.54/100
httank=-(dz2+dz3+dz4+dz5+hthx)
*
* length of pipes in piping loop
*
L=0.5
L2=L
L3=L
L4=L
L5=L
*
* diameter of pipes in piping loop
*
D=.75*2.54/100
DIA2=D
DIA3=D
DIA4=D
DIA5=D
*
* minor loss coefficients in piping loop
*
K=3/2
K2=K
K3=K
K4=K
K5=K
*
* type of glycol solution: 1=propylene glycol
*                          2=ethylene glycol
glycol=1
*
* percent propylene glycol solution
*
percent=40
*
* number of effectiveness curves
*
```

```
Ntests=3

UNIT 67 TYPE 67 ------NATURAL CONVECTION HEAT EXCHANGER ----------------
PARAMETERS 25
40 mdotg hthx
L2  L3  L4  L5
dz2 dz3 dz4 dz5 httank
DIA2  DIA3  DIA4  DIA5
K2  K3  K4  K5
9  39 percent Ntests glycol
INPUTS 4
*T2    T8     head     control
THXWI  THXGI  68,13    0,0
25     25     14.      1
*outputs:1)Two,(2)Tgo,(3)Qhx,(4)eff,(5)mdotw,(6)dpstatic,(7)-dpshear

EQUATIONS 1
qhx=[67,3]/3.6

UNIT 68 TYPE 68 ------- TANK ----------------------------------------
PARAMETERS 20
*mode vol    cp    rho     U_t    ht of node
1  0.454    4.18   993     1.44    httank        1
1 1 55 2 0
1 1 55 2 0
*          bp
0.0  20.0  100.0
INPUTS 5
*T5   mdotw  Tmains  mdotload  Tenv
0,0   67,5   0,0     0,0       0,0
19    72     19      0         19.0
DERIVATIVES 1
19
*output: 1)Tout 2)mdot out 3)Tload 4)mdotLoad 5)Qenv ...12)Tave

UNIT 25 TYPE 25 -------- PRINTER -------------------------------------
PARAMETERS 5
*DTIME    t_START    t_END   LOGICAL UNIT#
1.        25.        90.         14           1
INPUTS 6
  THXGI  MWS   67,1   67,2   67,4     qhx
  Tgin  MDOTW  THXWO  THXGO  HXEFF    QHX
    C     KG/S   C      C      -      J/S

end

************************************************************************
```

2) Transient Deck

```
assign WDATA.DAT         24
ASSIGN LOOKUP.DPHX       15
ASSIGN LOOKUP.EFF        16
assign out.hourly.simple 51
assign out.month.simple  32
assign out.year.simple   33
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                           *
*                            TRANSIENT                                      *
*            NATURAL CONVECTION HOT WATER HEATING SYSTEM                    *
*                           December 1994                                   *
*                            simple deck                                    *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
limits 100 100
WIDTH 72
tol -.001 .001

constants 5 --------------------- TIME -----------------------------
tbegin=90*24
tend=120*24
*tbegin=0
*tend = 8670
timest=1
firstday=int(tbegin/24)
pstop=91*24

SIMULATION tbegin tend timest

equations 4 --------- PARAMETERS COMMON TO BOTH MODES---------------
*
* type of glycol solution: 1=propylene glycol
*                          2=ethylene glycol
glycol=1
*
* percent propylene glycol solution
*
percent=40
*
* HX inlet, outlet minor loss coefficients
*
Kinlet=0.4
Kexit=1

Equations 29 ------------System Parameters-------------------------
Ntests=3
city=127
lat=43.14
slope=lat
roeg=0.2
cpg=3.82
mdotg=.03*3600
```

```
areac=4.5
*
* vertical rise in piping loop
*
dz2=0
dz3=0.064
dz4=0
dz5=0.965
hthx=16*2.54/100
httank=-(dz2+dz3+dz4+dz5+hthx)
*
* length of pipes in piping loop
*
L=0.5
L2=L
L3=L
L4=L
L5=L
*
* diameter of pipes in piping loop
*
D=.75*2.54/100
DIA2=D
DIA3=D
DIA4=D
DIA5=D
*
* minor loss coefficients in piping loop
*
K=3/2
K2=K
K3=K
K4=K
K5=K

unit 54 type 54 ---  WEATHER DATA GENERATER -  ------------------------
parameters 6
*si units LU city#
  1       24  city     1 1 1
*outputs:1)month,(2)dayofmonth,(3)hourofday,(4)T_db,(5)T_dewpoint
*        (6)humidity,(7)total radiation on horizontal
*        (8)Direct normal radiation,(9)Diffuse radiation,(10)Windspeed

UNIT 16 TYPE 16 ------   RADIATION PROCESSOR  ------------------------
PARAMETERS 9
*MODE  TRACK-MODE  SURF-MODE  DAY       LAT    SC  SHFT  SMOOTH  IE
  8        1           1      firstday  lat   4871  0       1    -1
INPUTS 9
* I  IDN  TD1  TD2  RHOG  SLOPE  AZIMUTH
*54,7 54,8 54,19 54,20 0,0  0,0   0,0   54,27 54,28
*0.0  0.0   0.0   1.0 roeg  slope  0.0  0.0   0.0
* I  Idiff   TD1    TD2   RHOG  SLOPE  AZIMUTH  Inext   Idiff_next
```

```
54,7  54,9  54,19  54,20  0,0    0,0   0,0      54,27  54,29
0.0   0      0     1      roeg   slope  0        0      0

equation 1
fcol=[2,1]*mdotg

UNIT 1 TYPE 1 ------------  SOLAR COLLECTOR  -------------------------
* this mode allows for a constant effectiveness HX attached
* as this is not appropriate in this deck:
*  1) set parameter 11 = parameter 4
*  2) set input 2 = input 3
PARAMETERS 14
*mode #series Area Cp_g  effmode mtest[kg/hr-m^2]
1      1      Areac  cpg    1        11.86
*FRTalpha  FRUL-1  FRUL-2  NO HX
0.634      14.49  0.026    -1
*Cp_g  opticalmode  bo-1  bo-2
cpg     1           0.448 -0.234
INPUTS 10
*Tin  mdotg mdotg  Tamb  I_T   I_horiz I_d  roeg inc.ang. slope
67,2  fcol   fcol  54,4  16,6  16,4    16,5  0,0  16,9    16,10
20.0   72     72   15.6  0.0   0.0     0.0   roeg  0.0     40.0
* outputs: 1)Tout  2)mdot_g  3)Qu[kJ/hr]

unit 44 type 44 -------convergence promoter--------------------------
parameters 1
3
inputs 1
1,1
25

unit 2 type 2 -------------  CONTROLLER  ----------------------------
param 4
*   HDB  LDB  cutoff
7    0   0     100    96
input 4
*hi T  lo T   cutoff T  inputcontrolfunction
1,1    68,1    0,0      2,1
1      30       0       1

UNIT 67 TYPE 67 ------NATURAL CONVECTION HEAT EXCHANGER --------------
PARAMETERS 25
40 mdotg hthx
L2  L3  L4  L5
dz2 dz3 dz4 dz5 httank
DIA2  DIA3  DIA4  DIA5
K2  K3  K4  K5
9  39 percent Ntests glycol
INPUTS 4
*T2     T8   head     control
68,1   44,1  68,13    2,1
25     25    14.      1
```

```
*outputs:1)Two,(2)Tgo,(3)Qhx,(4)eff,(5)mdotw,(6)dpstatic,(7)-dpshear

nocheck 4
2,1 2,2 2,4 67,4

UNIT 14 TYPE 14 ------------LOAD--------------------------------------
PARAMETERS 72
0.0,   0.0    7.0, 0.0
7.0,  85.86   8.0,85.86
8.0,   4.64   9.0, 4.64
9.0,   4.18  10.0, 4.18
10.0,  4.08  11.0, 4.08
11.0,  0.0   12.0, 0.0
12.0,  4.18  13.0, 4.18
13.0,  2.23  14.0, 2.23
14.0,  3.25  15.0, 3.25
15.0,  4.64  16.0, 4.64
16.0,  1.39  17.0, 1.39
17.0,  0.0   18.0, 0.0
18.0,142.77  19.0,142.77
19.0,  0.0   20.0, 0.0
20.0,  1.39  21.0, 1.39
21.0,  0.0   22.0, 0.0
22.0,  1.39  23.0, 1.39
23.0,  0.0   24.0, 0.0

unit 6 type 6 -----------AUXILLIARY HEATER----------------------------
parameters 5
*max heating rate, Tset, Cp, UAheater, efficiency of heater
1000000             60  4.19    0           1
inputs 4
*Tin  mdoti control Tamb for loss calcs
11,1   11,2     0,0      0,0
60      1        1       20
*output:(1)Tout,(2)mdotout,(3)required heating rate[kj/hr]

unit 10 type 11-- tempering valve located in upstream of tank----------
par 2
5  5
inputs 4
*Tinlet  minlet  Theatsource  Tset
0,0      14,1     68,3        0,0
8.0      0.0      15.0        60.0
*output: (1)(3)Toutlet,(2)mw to tank,(4)mw to heater

equations 3 ----------------TEMPERING VALVE FLOW EQNS-----------------
* mdraw is the mass flow rate of draw
* mtempr is the mass flow rate of mains diverted to heater
* mtk is the mass flow rate of mains that flows into tank
mdraw=[14,1]
mtempr=[10,4]
mtk=mdraw-mtempr
```

```
unit 11 type 11---- T Piece, located in front of aux heater------------
par 1
1
inputs 4
*Tinlet1  minlet1  Tinlet2    minlet2
68,3     mtk     10,3        10,4
8.0      1.0      8.0        0.0
*output: (1)Toutlet to aux heater,(2)mw to aux heater
*inlet 1 is tank, inlet 2 is from tempering valve


UNIT 68 TYPE 68---------------------TANK-------------------------------
PARAMETERS 20
*mode vol    cp     roe   U_t    ht of node
1  0.454    4.18   993     1.44    httank        1
1 1 55 2 0
1 1 55 2 0
*          bp
0.0  20.0  100.0
INPUTS 7
*T5   mdotw    Tmains mdotload  Tamb
67,1  67,5      0,0       mtk      0,0   0,0  0,0
25    30          8       0        20     0.0  0.0
DeRIVATIVES 5
19 19 19 19 19
*output: 1)Tout 2)mdot out 3)Tload 4)mdotLoad 5)Qenv 6)rate energy to load
*         7)DUtank...12)tave

equations 4  ---------------EQNS FOR INTEGRATER-----------------------
month=8670/12
year=8670
*[Qtoload (from tank) + Qaux = Qrequired]
*[Q aux = heat needed to supplement solar heating (aux heater)]
Qreq=max([68,6]+[6,3],.0001)
Qaux=[6,3]

UNIT 30 TYPE 24------------------- INTEGRATOR-------------------------
PARAMETERS 1
month
INPUTS 5
*           Qtoenv  Qtoload QintoTK
Qaux Qreq   68,5     68,6    68,11
0      0     0         0       0

UNIT 24 TYPE 24------------------- INTEGRATOR-------------------------
PARAMETERS 1
year
INPUTS 5
*           Qtoenv  Qtoload QintoTK
Qaux Qreq   68,5     68,6    68,11
0      0     0         0       0
```

```
equations 11 -----------------EQNS FOR PRINTERS-----------------------
mws=[67,5]/(3600)
Qin=[30,5]
Qout=[30,4]+[30,3]+[68,7]
enbal=200*(Qin-Qout)/(max(0.001,Qin+Qout))
SF=([30,2]-[30,1])/max(0.001,[30,2])
Qinyr=[24,5]
Qoutyr=[24,4]+[24,3]+[68,7]
enbalyr=200*(Qinyr-Qoutyr)/(max(0.001,Qinyr+Qoutyr))
SFyr=([24,2]-[24,1])/max(0.001,[24,2])

UNIT 49 TYPE 25 ----------------------PRINTER-----------------------
*PRINT ENERGY TERMS
PARAMETERS 4
*DTIME    t_START    t_END   LOGICAL UNIT#
month      tbegin     tend       32
inputs 6
  Qin Qout 30,1  30,2 sf  enbal
  Qin Qout Qaux  Qreq sf  enbal%

UNIT 48 TYPE 25 ----------------------PRINTER-----------------------
*PRINT ENERGY TERMS
PARAMETERS 4
*DTIME    t_START    t_END   LOGICAL UNIT#
year      tbegin     tend       33
inputs 6
 Qinyr Qoutyr 24,1 24,2 sfyr enbalyr
 Qinyr Qoutyr Qaux Qreq sf   enbal%

UNIT 51 TYPE 25 ----------------------PRINTER-----------------------
*PRINT ENERGY TERMS
PARAMETERS 4
*DTIME    t_START    t_END   LOGICAL UNIT#
timest    tbegin     pstop       51
inputs 5
67,3  mwave  67,4   67,1 68,12
Q       mw   modeff two  Ttkave

end
************************************************************************
```

CODE:

```
      SUBROUTINE TYPE67(time,xin,out,t,dtdt,par,info,icntrl,*)
************************************************************
* This subroutine represents the water piping loop for a
* natural convection SDHW system.  The NCHE is located in
* subroutine NCHE, this subroutine is just a regula falsi
```

```
* solver.
*
* This model is termed the simple deck.
* Last revised: Jan 17, 1994
************************************************************
      implicit none

      integer iter,fix,seed
      real*8 mhi,mlo,errhi,errlo,dm,de,m,err,toler
      real*8 m11,m22,two1,two2,tgo0,tgo1,tgo2
      real*8 two3,tgo3,m33
      real*8 two4,tgo4,m44
      real*8 two5,two6,tgo5,tgo6,m55,m66
      integer*4 info(15)
      integer i,j,icount,ni,nd,np,icntrl(2)
      character*3 ycheck(4),ocheck(8)
      real*4 t,dtdt,par(25),time
      real*8 xin(4),out(8)
      real*8 toler2,factor
      real*8 eeeold,eeealt,deeold,deesum,deealt,eee
      integer control
      real*8 tgocp,twocp,mcp
      real*4 timeold
      real*8 fac
      integer ct,Nconv
      integer isum,jsum,fiter
*--------------------------------
*  first call, info array stuff
*--------------------------------
      if (info(7).ge.0) go to 1

       np=info(4)            ! # of parameters
       info(6)=8             ! # of outputs
       info(9)=1             ! call type 67 every timestep
       icount=0
       ni=4
       nd=0

       call typeck(1,info,ni,np,nd)
*--------------------------------
*   set variable types
*--------------------------------
      data ycheck/'TE1','TE1','PR2','DM1'/
      data ocheck/'TE1','TE1','PW1','DM1','MF1',
     @            'PR2','PR2','PR2'/
      call rcheck(info,ycheck,ocheck)

      fac=0.d0
      call NCHE(fac,fac,TIME,XIN,OUT,PAR,INFO)

      tgocp=0
      twocp=0
```

```
      mcp=0
      isum=0
      jsum=0
      toler2=0.005d0
      Nconv=0
      timeold=time

      return1
*-------------------------------------------------------------
* Program beginning
*-------------------------------------------------------------
 1    if ((time-timeold).gt.0.01) then
         fiter=0
         mcp=0
         twocp=0
         tgocp=0
         isum=0
         jsum=0
         ct=0
         fix=0

       endif
*---------------------
      ct=ct+1
      icount=icount+1
*----------------------------------------
* if controller is off, or T_g,i < T_w,i
* then set outlet temps to inlet temps, exit subroutine.
*----------------------------------------
      control=xin(4)
      if (control.lt.0.5.or.xin(1).gt.xin(2)) then
         out(2)=xin(2)                    !Tgo=Tgi
         out(1)=xin(1)                    !Two=Twi
         out(3)=0.d0                      !Q_hx=0
         out(4)=0.d0                      !eff=0
         out(5)=0.d0                      !mdotw=0
         out(6)=0.d0                      !dpstto=0
         out(7)=0.d0                      !dpshto=0
         out(8)=out(6)-out(7)             !error=0

         m=out(5)
         err=out(8)
         timeold=time
         iter=1
         return1
      endif

      iter=1
      toler=0.001d0
*----------
* this finds a good starting mhi point, cuts down on iterations
*----------
```

```
 3    mhi=0.16d0*3600.d0

      j=1
      errhi=1
      do while (errhi.gt.0.d0)
         mhi=0.5d0*mhi
         call NCHE(mhi,errhi,TIME,XIN,OUT,PAR,INFO)

         j=j+1
         if (j.gt.9) then
            mlo=0.d0
            goto 4
         endif
       enddo
       mlo=mhi                !as the last mhi was negative

 4     mhi=2.d0*mhi
*-------------------------------------
 5    call NCHE(mhi,errhi,TIME,XIN,OUT,PAR,INFO)
      call NCHE(mlo,errlo,TIME,XIN,OUT,PAR,INFO)

 10   dm=mhi-mlo
      de=errhi-errlo
*-------------------
* If, as happens once every 600,000 timesteps or so
* (I've seen it a few times), the process is
* forced out of ncheloop again and again, and the dm is allowed
* to converge to zero without a solution, then the program will
* die.  To avoid this, the following statement kicks it back out
* to TRNSYS to change the inputs, so hopefully this won't
* reoccur.
*-------------------
      if (abs(dm).lt.0.000001d0.and.abs(de).gt.0.005d0) then
         goto 2000
      endif
*--------------------
* I've also seen it iterate within TYPE67 for 12000 iterations
* before being sent out.  Solution: send it out at 500 iterations,
* let TRNSYS return with better inputs. Also rare ocurrence.
*-------------------
      if (iter.gt.500) then
         goto 2000
      endif
*-----------------------
* this is the line function used to find new guess point
*----------------------
      m=dm/de*(mlo*(de/dm)-errlo)

      call NCHE(m,err,TIME,XIN,OUT,PAR,INFO)
*---------------------
* The following says if m<0, call NCHE with m=0.
* If err>0, the solution for mdotw is negative,
```

```
*  => so send the solution out for mdotw=0.
* If err<0, then continue on as before.
*--------------------
 48   if (m.lt.0.d0) then
             m=0.0000d0
             call NCHE(m,err,TIME,XIN,OUT,PAR,INFO)
             if (err.gt.0.d0) then
                goto 2000
             elseif (err.le.0.d0) then
                 goto 1000
             endif
      endif
*--------
* This is for if NCHE keeps returning the same error value
* and keeps getting sent the same m value.
* The following code will sent it out of this type for TRNSYS to
* give it new values to work with.
*-----------------------
      eeealt=eeeold             !eee,eeeold,eeealt are error values
      eeeold=eee                !eee at current timestep,
      eee=err                   !eeeold at previous
      deeold=abs(eeeold-err)    !eeealt at timestep before eeeold
      deealt=abs(eeealt-eeeold)
      deesum=deeold+deealt
      if (deesum.lt.0.0000001d0) then
         goto 2000
      endif
*--------------------------------
*  The meat of the regula falsi, if err<>=toler then do this or that.
*--------------------------------
 1000 if (abs(err).lt.toler) then
         goto 2000
      endif
      if (err.lt.0.d0) then
         mlo=m
         errlo=err
      elseif (err.gt.0.d0) then
         mhi=m
         errhi=err
      endif
      iter=iter+1
      goto 10
*---------------------------
* convergence promotion: If there is excessive cycling between
* TRNSYS and this TYPE, if TYPE 44 won't resolve the cycling,
* this internal convergence promoter will.  Most likely this is
* not needed, but to insure convergence at every step, I left it
* in.  It's function is to guess a good starting point for
* the successive substitution iterating process, as at times,
* successive substution cannot converge.  It also helps the
* secant method.
*---------------------------
```

```
 2000 if (ct.gt.3) then
         Two6=Two5
         Two5=Two4
         Two4=Two3
         Two3=Two2
         Two2=Two1
         Two1=out(1)
         Tgo6=Tgo5
         Tgo5=Tgo4
         Tgo4=Tgo3
         Tgo3=Tgo2
         Tgo2=Tgo1
         Tgo1=out(2)
         m66=m55
         m55=m44
         m44=m33
         m33=m22
         m22=m11
         m11=out(5)
*----------
* The following are different cases of cycling--that is, when the
* outputs cycle between two sets of outputs, as do the inputs.
*---------
         if (abs(Two3-Two1) .lt.toler2.
     @   and.abs(Tgo3-Tgo1) .lt.toler2.
     @   and.abs(m33-m11)   .lt.toler2.

                           !ct is #times in 67 at timestep
                           !fiter is ct at last convergence promotion

     @   or.abs(twocp-two2) .lt.0.1d0.
     @   and.abs(tgocp-tgo2).lt.0.1d0.
     @   and.abs(mcp-m22)   .lt.0.1d0.
     @   and.(ct-fiter).gt.5.

     @   or.abs(m44-m22).lt.0.0000001d0.
     @   and.abs(m33-m11).lt.20.d0.
     @   and.abs(m33-m11).gt.0.0000001d0.
     @   and.abs(two3-two1).lt.1.d0.
     @   and.abs(tgo3-tgo1).lt.1.d0.
     @   and.(ct-fiter).gt.5.

     @   or.abs(m55-m33-m11).lt.0.0000001d0.
     @   and.abs(m66-m22).lt.20.d0.
     @   and.abs(m66-m22).gt.0.0000001d0.
     @   and.abs(two6-two2).lt.1.d0.
     @   and.abs(tgo6-tgo2).lt.1.d0
     @   .and.(ct-fiter).gt.5
     @   ) then

            Twocp=Two3
            Tgocp=Tgo3
```

```
            mcp=m33
            fix=fix+1
            Nconv=Nconv+1
            if (fix.eq.1) then
               tgo0=(tgo2+tgo3)/2.d0
                fiter=ct
            elseif (fix.eq.2) then
               tgo0=(tgo2+tgo3)/2.d0+abs(tgo2-tgo3)/4.d0
                fiter=ct
            elseif (fix.eq.3) then
               tgo0=(tgo2+tgo3)/2.d0-abs(tgo2-tgo3)/4.d0
                fiter=ct
            elseif (fix.ge.4.and.fix.lt.13) then
               if (fix.eq.5) fix=6
               tgo0=min(tgo2,tgo3)+(fix-3)*abs(tgo2-tgo3)/10.d0
                fiter=ct
            elseif (fix.ge.13) then
                  if (fix.eq.13) seed=13
                  seed=2045*seed+1
                  seed=seed-(seed/1048576)*1048576
                  factor=dble(seed+1)/1048577.d0
               tgo0=min(tgo2,tgo3)+
     @                factor*abs(tgo2-tgo3)
               fiter=ct
             endif
              out(2)=tgo0
           endif
         endif

 2001 isum=isum+iter
      jsum=jsum+j
      timeold=time
*----------------------------
      return1
      end
*****************************************************************
      SUBROUTINE NCHE(m,error,time,XIN,OUT,PAR,INFO)
*---------------------------------------------------------
* This subroutine models the NCHE. It requires two data files:
* one for shear pressure drop in the NCHE, and the other for
* modified effectiveness, both as a function of water flow rate.
*---------------------------------------------------------
*  LOCATIONS:
*     (1)=NCHE average
*     (2)=h2o out of storage tank
*     (3)=cold h2o into NCHE
*     (4)=hot water out of NCHE
*     (5)=hot water into storage tank
*     (6)=storage tank average
*     (7)=hot solution out of collector
*     (8)=hot solution into NCHE
*     (9)=cool solution out of NCHE
```

```
*     (10)=cool solution into pump
*     (11)=cool solution out of pump
*     (12)=cool solution into collector
*-----------------------------------------------------------
      implicit none

* TRNSYS VARIABLES

      integer*4 info(15)
      integer i,j,icount
      real*4 par(25),time
      real*8 xin(4),out(8)

* TYPE 67 VARIABLES

      integer hxnx               ! # of nodes for HX temp. distribution
      integer percent            ! % of propylene glycol solution
      integer npts1              ! # of rows in data file: eff vs mdotw
      integer npts2              ! # of rows in data file: dphx vs mdotw
      integer Ntests             ! # of curves of eff for diff mdotg's
      integer glycol             ! type of antifreeze: 1)prop. gly
                                 !                     2)eth. gly
      real*8 temp(15),roe(15)    ! property arrays  [C],[kg/m^3]
      real*8 cp(15),mu(15)       ! property arrays  [J/kg-K],[Pa-s]
      real*8 hthx                ! height (length) of HX  [m]
      real*8 dia(15),L(15)       ! diameter, length of pipe sections  [m]
      real*8 dz(15)              ! change in elevation of pipe sections[m]
      real*8 k(15)               ! minor loss k values assoc.w.pipe
                                   sections
      real*8 cpgave              ! ave cp for glycol in HX  [J/kg-K]
      real*8 cw,cg               ! cw = mdotw * cpw, cg = ...  [W/K]
      real*8 mdotw,mdotg         ! flow rates of water, glycol  [kg/hr]
      real*8 mdotws,mdotgs       ! flow rates of water, glycol  [kg/s]
      real*8 mfirst,efirst       ! 1st non-zero values from data
                                   files,(row 2)
      real*8 dphx                ! shear pressure head in HX  [kPa]
      real*8 dptank              ! static pressure head in tank  [kPa]
      real*8 mpos,cwpos          ! abs. values of mdotw, cw
      real*8 effmod              ! modified effectiveness
      real*8 qhx                 ! heat transfer in HX  [W]
      real*8 dpstto              ! tot. static head around H20 loop  [kPa]
      real*8 dpshto              ! tot. shear press.loss around loop [kPa]
      real*8 m                   ! input into NCHE, water mdot [kg/hr]
      real*8 error               ! error:the diff. betw. shear and static
                                 ! pressure drops,output from NCHE
      real*8 errold              ! error from the last iter. within nche
      real*8 erralt              ! errold from the last iter. within nche
      real*8 derr                ! derr=abs(error-errold)
                                       +abs(error-erralt)
      integer run,zzz,zzzz       ! counters
      character*3 fluid          ! antifreeze fluid id: 1) prop. gly.
                                 !                      2) eth. gly.
```

```
*-----------------------------------
      if (info(7).ge.0) go to 1
*--------
* get parameters and inputs from arrays
*--------------------------------
      hxnx=par(1)
      mdotg=par(2)
      mdotg=dble(int(10000.d0*mdotg)/10000.d0)   !done to lose error due
                                                 !to par being real*4
      hthx=par(3)
      L(2)=par(4)
      L(3)=par(5)
      L(4)=par(6)
      L(5)=par(7)

      dz(1)=par(3)
      dz(2)=par(8)
      dz(3)=par(9)
      dz(4)=par(10)
      dz(5)=par(11)
      dz(6)=par(12)

      dia(2)=par(13)
      dia(3)=par(14)
      dia(4)=par(15)
      dia(5)=par(16)

      k(2)=par(17)
      k(3)=par(18)
      k(4)=par(19)
      k(5)=par(20)
      npts1=par(21)
      npts2=par(22)
      percent=par(23)
      Ntests=par(24)
      glycol=par(25)
*---------------------------------------
* determine type of glycol solution to be used
*-------------------------------------
        if (glycol.eq.1) then
           fluid='pro'
        else if (glycol.eq.2) then
           fluid='eth'
        else
           print*,'error in parameter 25'
           print*,'must be 1 or 2'
           stop
        endif
*------------------
*  set initial temp values based on temp2, temp7
*------------------
      temp(4)=1.d0/2.d0*(temp(7)-temp(2))+temp(2)
```

```
      temp(9)=temp(7)-1.d0/2.d0*(temp(7)-temp(2))
      temp(1)=1.d0/2.d0*(temp(4)-temp(2))+temp(2)

         call Lookuptables(mpos,mdotgs,effmod,npts1,npts2,
     @                     1,Ntests,dphx,mfirst,efirst)

      return
*=====================================================================
* Program beginning
*-------------------
 1    zzz=0
      zzzz=0
      icount=icount+1
      run=0

 2    temp(2)=xin(1)
      temp(7)=xin(2)
      dptank=xin(3)
      mdotw=m
*---------------------------------
*  change mdot from [kg/hr] to [kg/s]
*---------------------------------
      mdotgs=mdotg/3600.d0
      mdotws=mdotw/3600.d0
*---------------------------------
*  Assuming no heat losses in pipes:
*---------------------------------
      temp(3)=temp(2)
      temp(8)=temp(7)
*---------------------------------
*  get properties of both loops
*---------------------------------
 5      call propsw(roe,mu,cp,temp)
        call propsg(fluid,percent,cp,roe,temp)
*---------------------------------
*  find cpgave
*---------------------------------
        cpgave=(cp(8)+cp(9))/2.d0
*---------------------------------
*  find c_w, c_g
*---------------------------------
 60     cw = abs(mdotws) * cp(1)
        cg = mdotgs * cpgave

        run=run+1
*---------------------------------
*  find experimental effectiveness from mdot_w
*---------------------------------
       if (run.eq.1) then
           mpos=abs(mdotws)
           call Lookuptables(mpos,mdotgs,effmod,npts1,npts2,
     @                     45,Ntests,dphx,mfirst,efirst)
```

```
       endif
*-------------------------------------
*  find Temperature of water out of NCHE from eff_mod
*-------------------------------------
      if (abs(cw).ge.0.001d0) temp(4)=temp(3)+
     @                         effmod*cg*(temp(8)-temp(3))/cw
      if (abs(cw).lt.0.001d0) temp(4)=temp(3)+
     @                  efirst/mfirst*cg/cp(1)*(temp(8)-temp(3))
*-------------------------------------
*  find heat transfer in NCHE from t(4),mdot_w
*-------------------------------------
      qhx=cw*(temp(4)-temp(3))
*-------------------------------------
*  find temperature of antifreeze out ot NCHE from q_hx
*-------------------------------------
      temp(9)=temp(8)-qhx/cg
      temp(5)=temp(4)
*-------------------------------------
*  find hx average water temperature, ave hx density
*-------------------------------------
      cwpos=abs(cw)
      call xtdist(temp,hthx,hxnx,cwpos,cg,roe,mdotws)
*-------------------------------------
*  if this is the first time through, then get props again
*  do energy balances again
*-------------------------------------
      if (run.eq.1) goto 5
*-------------------------------------
*  find properties of water loop again
*-------------------------------------
      call propsw(roe,mu,cp,temp)
*-------------------------------------
*  find static pressure drop from densities around water loop
*-------------------------------------
       call statdp(dptank,roe,dz,dpstto)
*-------------------------------------
* find dpshto from old mdot value, for berg deck, dont need to call
*-------------------------------------
       call findshear(k,L,dia,roe,mu,dpshto,mdotws,dphx)
*-------------------------------------
*  change mdot from [kg/s] to [kg/hr]
*-------------------------------------
 900  mdotg=mdotgs*3600.d0
      mdotw=mdotws*3600.d0
*-------------------------------------
*  write output variables to arrays
*-------------------------------------
       out(1)=temp(5)              ![C]
       out(2)=temp(9)              ![C]
       out(3)=qhx*3.6d0            ![kJ/hr]
       out(4)=effmod               ![-]
       out(5)=mdotw                ![kg/hr]
```

```
       out(6)=-dpstto              ![kPa]
       out(7)=dpshto               ![kPa]
       out(8)=out(7)-out(6)        ![kPa]
*-------------------------------------
* Counters, old m,error,time values
*-------------------------------------
        erralt=errold
        errold=error
        error=out(8)
        derr=abs(error-errold)+abs(error-erralt)
*------------------------------
* if three successive iterations in NCHE
* are the same, then send it out
*------------------------------
        if (derr.le.1.d-6) then
           return
        endif
*------------------------------
* if it has not yet converged then,
* increment number of iterations in NCHE
* for this call from Regula Falsi
*------------------------------
        if (derr.gt.1.0d-6) then
           zzz=zzz+1
*------------------------------
* If it hasn't converged in 10 iterations, it is cycling within NCHE.
* After 10 iterations in NCHE, shift Tgi over by 1%, try again.
*------------------------------
           if (zzz.gt.10) then              !if 10 iter in nche
               xin(2)=1.01d0*xin(2)
               zzz=0
               zzzz=zzzz+1                  !keep track of # shifts
           endif
*--------------------------
* If it still won't converge after 5 shifts, send it out to Regula Falsi
* solver as it is.
*--------------------------
           if (zzzz.gt.5) then
                 return
           endif

           goto 2                           !shift or no shift, try again
        endif
*-------------------------
       return
       end
*=======================================================
       Subroutine findshear(k,L,d,roe,mu,dpshto,
     @                      mdotws,dphx)
*-------------------------------------------------------
*  this subroutine will find the shear pressure drop in the
*  system using the inputted [old] water flow rate
```

```
*---------------------
      implicit none
      integer i,j
      real*8 k(15),L(15),d(15),roe(15),mu(15)
      real*8 dpshto,mdotws,pi
      real*8 var1,var2,var3,var4,dphx
*-----------------
      pi=3.14159d0
*-----------------
*  define function, which is the sum of the pressure drops around the
*  water loop, or in other words the shear pressur loss.
*-----------------
      var1=(16.d0*pi*mu(2)*L(2) + K(2)*abs(mdotws))/(D(2)**4*roe(2))
      var2=(16.d0*pi*mu(3)*L(3) + K(3)*abs(mdotws))/(D(3)**4*roe(3))
      var3=(16.d0*pi*mu(4)*L(4) + K(4)*abs(mdotws))/(D(4)**4*roe(4))
      var4=(16.d0*pi*mu(5)*L(5) + K(5)*abs(mdotws))/(D(5)**4*roe(5))

      dpshto= 1.d0/1000.d0*
     @          ( 8.d0*abs(mdotws)/(pi*pi)
     @                             *(var1+var2+var3+var4))
     @           +dphx
*-------------------
*  if flow is negative, then set dpshto to opposite sign
*-------------------
      if (mdotws.lt.0.d0) dpshto=-dpshto
      return
      end
*=======================================================
      Subroutine statdp(dptank,roe,dz,dpstto)
*-------------------------------------------------------
*  This subroutine will use densities around the secondary
*  loop to find static pressure drop
*-------------------------------------------------------
      integer i
      real*8 dptank,roe(15),g
      real*8 dz(15),dpstto,dpstat(15)

      g=9.806d0
      dpstat(6)=dptank

      dpstto=0.d0
      do i=1,5
       dpstat(i)=roe(i)*g*dz(i)*.001
       dpstto=dpstto+dpstat(i)
      enddo

      dpstto=dpstto-dpstat(6)
      return
      end
*=======================================================
      Subroutine xtdist(temp,hthx,hxnx,cw,cgly,roe,mdotw)
*-------------------------------------------------------
```

```
*  This subroutine will use the trapezoid rule to find the
*  temperature distribution of the heat exchanger.
*  This subroutine assumes:
*-------------------------------------------------------
*  SYMBOLS:
*     hthx=height of hx
*     hxnx=number of points used in integration
*     ax,bx,cx,dx,ex=variables of no great significance
*     dtw,dthot,dto=interim variables of little significance
*-------------------------------------------------------
      implicit none

      integer i,hxnx
      real*8 temp(15),var1,dtw,dthot,dto,cw,cgly
      real*8 ax,bx,cx,dx,ex,thx(100),tinteg,mdotw
      real*8 thxave,hthx,avdens,rhohx(100),roe(15)
*-------------------------w40r this takes care of no flow sit
      if (abs(temp(3)-temp(4)).lt.0.0010d0) then
        temp(4)=temp(3)+0.0001d0
      endif
*--------------------------
      var1=hthx/(2*hxnx)
      dtw=temp(4)-temp(3)
      dthot=temp(8)-temp(4)
*---------------
* negative dthot kills program
*--------------
      if (temp(4).gt.temp(8)) dthot=0.d0
*----------------
      dto=temp(8)-dtw*cw/cgly-temp(3)
*-------------------------------
      ax=temp(3)
      dx=2*var1
*---------------------------
      if (dthot.eq.dto) then
       bx=0.d0
       cx=0.d0
       ex=dtw/hthx
      else
       bx=dto*dtw/(dthot-dto)
       cx=dthot/dto
       ex=0.d0
      endif

        if (cx.lt.0.d0) cx=0.0001d0
*----------------------------
*  find temperature at each point in the NCHE
*----------------------------
      thx(0)=ax + bx*(cx**(0*dx/hthx)-1)

      do i=1,hxnx
       thx(i)=ax + bx*(cx**(i*dx/hthx)-1)
```

```
     @               +ex*i*dx
      enddo
*----------------------------
*  use Trapezoid Rule to integrate
*----------------------------
      tinteg=0.d0
      do i=1,hxnx-1
       tinteg=tinteg+2.d0*thx(i)
      enddo
      tinteg=var1*(tinteg+thx(0)+thx(hxnx))
*----------------------------
*  find the average temperature of hx, assign it to temp(1)
*----------------------------
      thxave=tinteg/hthx
      temp(1)=thxave
*----------------------------
*  find the density at each point corresponding to the temp
*----------------------------
             rhohx(0)=(999.8396 + 18.224944*thx(0)
     @              - 0.00792221*thx(0)**2
     @              -55.44846e-6*thx(0)**3
     @              +149.7562e-9*thx(0)**4
     @              -393.2952e-12*thx(0)**5)
     @              /(1.d0+18.159725e-3*thx(0))
      do i=1,hxnx
       rhohx(i)=(999.8396 + 18.224944*thx(i)
     @              - 0.00792221*thx(i)**2
     @              -55.44846e-6*thx(i)**3
     @              +149.7562e-9*thx(i)**4
     @              -393.2952e-12*thx(i)**5)
     @              /(1.d0+18.159725e-3*thx(i))

      enddo
*----------------------------
*  use Trapezoid Rule to integrate
*----------------------------
      tinteg=0.d0
      do i=1,hxnx-1
       tinteg=tinteg+2.d0*rhohx(i)
      enddo
      tinteg=var1*(tinteg+rhohx(0)+rhohx(hxnx))
*----------------------------
*  find the average temperature of hx, assign it to temp(1)
*----------------------------
      avdens=tinteg/hthx
      roe(1)=avdens

      return
      end
*=======================================================
      Subroutine propsg(fluid,percent,cp,roe,temp)
*-------------------------------------------------------
```

```
*  This subroutine will find properties of a propylene glycol
*  solution given the temperature in celsius and the percent
*  by volume of propylene glycol.  The properties found are:
*
*     1) density [kg/m^3]
*     2) specific heat [J/kg-C]
*
*  These property functions were written by Tim McDowell, 1994
*-------------------------------------------------------------
      integer i,j,percent
      character*3 fluid
      real*8 cp(15),roe(15),temp(15),tem
*-------------------------
      if (fluid.eq.'pro') then
        do i=7,9

          tem       =temp(i)+273.15d0

          cp(i)      = ((3.8649883866 - 0.023691954902*percent -
     *        0.00011278222908*percent**2) + (0.001023655712 +
     *        5.6633876714e-5*percent)*Tem)*1000

          roe(i)     = (875.54696219 + 2.151387542*percent) +
     *        (1.1191046068 - 0.0007599907262*percent -
     *        4.9236799989e-5*percent**2)*Tem + (-0.002377960199 -
     *        9.1377252136e-6*percent + 1.0872237562e-7*percent**2)
     *        * Tem**2

       enddo
*---------------------------------------
      elseif (fluid.eq.'eth') then

           if (percent.gt.85.d0.or.percent.lt.55.d0) then
              print*,'ethylene properties only calculated'
              print*,' for 55% to 85% by volume.'
              stop
           endif

        do i=7,9
             Tem=temp(i)+273.15d0

             cp(i)=3.9189-.035267*percent+
     @           (.0014555+4.8423d-5*percent)*Tem

              B=884.53+2.1741*percent
              C=1.1613-0.0033403*percent
              D=-0.0024393+2.994d-8*percent
              roe(i)=B+C*Tem+D*Tem**2

           enddo
        endif
      return
```

```
      end
*=========================================================
      Subroutine propsw(roe,mu,cp,temp)
*---------------------------------------------------------
*  This subrountine will find:
*     1) the density of water [kg/m^3]
*     2) the viscosity of water [kg/m-s]
*     3) the specific heat of water [J/kg-C]
*  at the different locations specified using equations given
*  in Gebhart [1988].
*---------------------------------------------------------
      integer  i,j
      real*8  roe(15),mu(15),cp(15),temp(15)
*------------------------
      do i=1,6

       roe(i)=(999.8396 + 18.224944*temp(i)
     @              - 0.00792221*temp(i)**2
     @              -55.44846e-6*temp(i)**3
     @              +149.7562e-9*temp(i)**4
     @              -393.2952e-12*temp(i)**5)
     @              /(1.d0+18.159725e-3*temp(i))

       mu(i)=2.414e-5*10**(247.8/(temp(i)+133.15))

*      cp(i) = 4193 - 0.79411* temp(i) + 0.012545*temp(i)**2
*     @            -3.9665e-5*temp(i)**3 + 1.7441e-7*temp(i)**4

         cp(i)=4209.4- 1.824*temp(i)
     @               + 3.0525d-2*temp(i)**2
     @               - 1.2388d-4*temp(i)**3
     @               + 1.2774d-7*temp(i)**4

      enddo

      return
      end
*=========================================================
      Subroutine Lookuptables(mdotw,mdotg,eff,npts1,npts2,
     @                        init,Ntests,dphx,mfirst,efirst)
*---------------------------------------------------------------
*  This subroutine will:
*     1)put given data file into arrays
*     2)given a independent variable, mdotw
*     3)identify appropriate range for interpolation
*     4)interpolate and give appropriate dependent variable, eff
*---------------------------------------------------------------
      implicit none
      integer i,j,Ntests,npts1,npts2,init
      real*8 mp(40),m(40),e(40,40),mhi,mlo,ehi,elo,mdotw,eff
      real*8 mfirst,efirst,dphx,p(40),phi,plo
      real*8 mdotg,el,zero
```

```
      real*8 mg(40),mghi,mglo
      real*4 time

*      open (unit=11,file='lookup.manyeff',status='unknown')
*      open (unit=21,file='lookup.dphx',status='unknown')

      i=1
      j=1
*---------------------------
* 1)  FOR HX HEAT TRANSFER EFFECTIVENESS
*-----------------------
* read in arrays from data file
*-----------------------
      if (init.eq.1) then

           if (Ntests.eq.1) then
              read(11,*) zero,mg(1)        !reads 1st row for mg values
              do i=1,Npts1
                 read(11,*) m(i),e(i,1)    !reads remaining rows for
mdotw,eff
              enddo
              efirst=e(2,1)
              mfirst=m(2)

           elseif (Ntests.eq.2) then
              read(11,*) zero,mg(1),mg(2)
              do i=1,Npts1
                 read(11,*) m(i),e(i,1),e(i,2)
              enddo
           elseif (Ntests.eq.3) then
              read(11,*) zero,mg(1),mg(2),mg(3)
              do i=1,Npts1
                 read(11,*) m(i),e(i,1),e(i,2),e(i,3)
              enddo
           elseif (Ntests.eq.4) then
              read(11,*) zero,mg(1),mg(2),mg(3),mg(4)
              do i=1,Npts1
                 read(11,*) m(i),e(i,1),e(i,2),e(i,3),e(i,4)
              enddo
           elseif (Ntests.eq.5) then
              read(11,*) zero,mg(1),mg(2),mg(3),mg(4),mg(5)
              do i=1,Npts1
                 read(11,*) m(i),e(i,1),e(i,2),e(i,3),e(i,4),e(i,5)
              enddo
           elseif  (Ntests.eq.6) then
              read(11,*) zero,mg(1),mg(2),mg(3),mg(4),mg(5),mg(6)
              do i=1,Npts1
                 read(11,*) m(i),e(i,1),e(i,2),e(i,3),e(i,4),
     @                     e(i,5),e(i,6)
              enddo
           elseif (Ntests.gt.6) then
              print*,'only six glycol flow rates allowed in data file'
```

```
               stop
            endif

      mfirst=m(2)

        goto 100
        endif
*------------
*  for case of just one test
*-----------
        if (ntests.eq.1) then

           if (mdotw.eq.m(1)) then
              eff=e(1,1)
              goto 100
           endif

           i=2

 5           if (mdotw.le.m(i)) then
                mhi=m(i)
                mlo=m(i-1)
                ehi=e(i,1)
                elo=e(i-1,1)
             elseif (mdotw.gt.m(i)) then
                i=i+1
                goto 5
             endif

          eff=elo+(ehi-elo)*(mdotw-mlo)/(mhi-mlo)

          goto 100
      endif
*----------------------
* find range for interpolation
*---------------------
      if (mdotw.eq.m(1)) then

         if (mdotg.lt.mg(1)) then
            print*,'mdotg out of range--mdotw=m(1),mdotg<mg(1)'
            print*,'this program does not extrapolate'
            print*,'mdotw,mdotg',mdotw,mdotg
            stop
         endif

      if (mdotg.gt.mg(Ntests)) then
         print*,'mdotg out of range'
         print*,'this program does not extrapolate'
         print*,'mdotg',mdotg
         print*,'last mdotg column in lookup table',mg(Ntests)
         stop
      endif
```

```
         j=2
 8       if (mdotg.le.mg(j)) then
            mghi=mg(j)
            mglo=mg(j-1)
         elseif (mdotg.gt.mg(j)) then
            j=j+1
            goto 8
         endif

         elo=e(1,j-1)
         ehi=e(1,j)
         eff=elo+(ehi-elo)*(mdotg-mglo)/(mghi-mglo)

         elo=e(2,j-1)
         ehi=e(2,j)
         efirst=elo+(ehi-elo)*(mdotg-mglo)/(mghi-mglo)

         goto 100
      endif
*-------------------------------------
      if (mdotg.gt.mg(Ntests)) then
         print*,'mdotg out of range'
         print*,'this program does not extrapolate'
         print*,'mdotg',mdotg
         print*,'last mdotg in file',mg(Ntests)

         stop
      endif

      i=2
 10   if (mdotw.le.m(i)) then
         mhi=m(i)
         mlo=m(i-1)
      elseif (mdotw.gt.m(i)) then
         i=i+1
         goto 10
      endif

      if (mdotg.lt.mg(1)) then
         print*,'mdotg out of range'
         print*,'this program does not extrapolate'
         stop
      endif

      j=2
 20   if (mdotg.le.mg(j)) then
         mghi=mg(j)
         mglo=mg(j-1)
      elseif (mdotg.gt.mg(j)) then
         j=j+1
         goto 20
```

```
      endif
*---------------------------
* interpolate first time
*---------------------------
      ehi=e(i-1,j)
      elo=e(i-1,j-1)
      el=elo+(ehi-elo)*(mdotg-mglo)/(mghi-mglo)

      ehi=e(i,j)
      elo=e(i,j-1)
      ehi=elo+(ehi-elo)*(mdotg-mglo)/(mghi-mglo)

      elo=el
*-----------------------------
* interpolate again
*-----------------------------
      eff=elo+(ehi-elo)*(mdotw-mlo)/(mhi-mlo)
*---------------------------
* interpolate to find efirst
*---------------------------
      elo=e(2,j-1)
      ehi=e(2,j)
      efirst=elo+(ehi-elo)*(mdotg-mglo)/(mghi-mglo)
*-----------------------
* 2) FOR PRESSURE LOSSES ACROSS HX
*    (NOT COUNTING STATIC HEAD)
*-----------------------
* read in arrays from data file
*-----------------------
 100  if (init.eq.1) then
       do i=1,npts2
          read(21,*) mp(i),p(i)
       enddo
       rewind 21

       return
      endif
*----------------------
* find range for interpolation
*----------------------
      if (mdotw.eq.mp(1)) then
         dphx=p(1)
      endif

      i=2

 200  if (mdotw.le.mp(i)) then
         mhi=mp(i)
         mlo=mp(i-1)
      elseif (mdotw.gt.mp(i)) then
         i=i+1
         goto 200
```

```
      endif
*------------------------------
* interpolate
*------------------------------
      phi=p(i)
      plo=p(i-1)

      dphx=plo+(phi-plo)*(mdotw-mlo)/(mhi-mlo)

      return
      end
*=======================================================
```