APPENDIX A

FLAT-PLATE COLLECTOR DESIGN PROGRAM (CODEPRO)

The flat-plate collector design program (CoDePro) is a program that can help solar engineers design flat-plate solar collectors. It has been developed so that most details of the collector configuration can be specified. The program has been developed with the professional version of EES and beta-tested from its development level by solar engineers.

CoDePro calculates the instantaneous efficiency, incident angle modifier coefficient, and stagnation temperatures. These calculations are related flat-plate solar collector tests. The test methodology is based on the standard test methods provided by ASHRAE Standard 93-86 [1] and SRCC Document RM-1 [17] that are described in the Chapter 2. This program can be used to investigate the thermal effect of the choices of geometric parameters and materials.

The employed models for the collector instantaneous efficiency are linear and 2nd-order polynomial equations (Equation 2.7.2 to 2.7.4). Once the program calculates the collector efficiency for different values of $(T_i\text{-}T_a)/G_T$, the coefficients of these equations are determined by linear regression. As specified in ASHRAE Standard 93-86 [1] CoDePro calculate the incident angle modifier, K_{ta}, at various angles between 0 to 60^o and then determines the incidence angle modifier coefficient b_0 of Equation 2.7.5 by

using linear regression.

Features

Some major features of the program are as follows:

1. Users can select the working fluid of the collector among water, ethylene glycol/water solution, and propylene glycol/water solution.
2. To get various values of $(T_i\text{-}T_a)/G_T$ the inlet temperature is varied from the ambient temperature to the circlating fluid saturation temperature.
3. To compare various collector configurations up to 5 sets of configuration and results data can be stored.
4. The ranges of design variables are confined to reasonable values
5. The value of ground reflectance is fixed at 0.4 to include the effect of ground-reflected radiation.

It should be noted that many functions used in the program have been developed by Solar Energy Laboratory, University of Wisconsin-Madison. It takes up to 10 seconds to evaluate the solar collector performance in Pentium II 350 MHz personal computers.

Main Window

Figure A.1 shows the main window of CoDePro. It has several child windows and drop-down menus for choosing the unit system, setting the test conditions, specifying the collector configuration, and displaying the results. The unit system can be selected from drop-down menu at the upper-right corner of the window. Up to 5 sets of configuration and calculation data can be saved by using the drop down menu named 'Save Calculation as'. Using this feature of the program the effect of parameters can be investigated very simply. Clicking the Calculate button will start calculation with the given test conditions and collector configurations. 'Save User Input Data' and 'Load saved data' buttons make it possible to save and load configuration and test data. The test results section in the middle of main window shows the Collector Efficiency Equations and Incident Angle Modifier Equation.

Child Window 1 – Test Conditions

In this window, shown in Figure A.2, the test conditions to calculate collector thermal performance are set. Input variables for test conditions are

- Intensity of incident solar radiation
- Diffuse radiation proportion
- Incident angle of beam radiation
- Collector slope
- Wind speed

- Relative humidity (for sky temperature calculation)

The European practice employs the temperature difference $T_{f,av}$-T_a as a base of collector efficiency while the Unite States uses T_i-T_a. Users can change the reference temperature difference in this window according to their standard.

Child window 2 – Collector Dimensions

Figure A.3 shows the window for collector dimensions where the overall and aperture collector dimensions are entered. It is assumed that the area of absorber is the same as the aperture area, i.e., the frontal transparent area. Input variables are

- Overall Dimensions: Length, width, and thickness
- Absorber Dimensions: Length and width

Child Window 3 – Cover and Plate

The cover and plate window shown in Figure A.4 needs the values of the optical and thermal properties of the covers and the absorber plate. The collector can have up to two covers. If it has only one cover, its properties are entered in Cover 1 section. It should be noted that the calculation results of CoDePro for two different plastic covers have not been verified against collector tests. Users can select the material or enter the properties of his/her own material by selecting ‘user defined’ from the drop down menu. Input variables in this window are

- Cover
    - Number of Covers: 0,1,2
    - Cover material
    - Refractive Index & transmittance (Solar spectrum)
    - Absorptance & Transmittance (long-wave spectrum)
- Cover-Plate Air Spacing
- Cover1-Cover2 Air Spacing
- Plate
    - Material (Conductivity)
    - Thickness
    - Absorptance for solar spectrum
    - Emittance for long-wave radiation

Child window 4 – Back and Edge Insulation

To calculate the heat loss coefficient, it is necessary for the program to be provided with information about the edge and back insulation. This information is provided through a window shown in Figure A.5.

- Thickness and conductivity of back Insulation
- Thickness and conductivity of edge Insulation

Child Window 5 – Tube and Fluid

Through the window, shown in Figure A.6, information about tubes and the working fluid is entered. The working fluid is selected among water, ethylene glycol/water and propylene glycol/water. The bond conductivity represents the contact resistance between the tube and absorber plate. If the thermal conductance between them is high, its value should be larger than 400 W/mK. Input variables in this window are

- Tube
    - Number of Tubes
    - Inner & outer diameter
- Fluid
    - Material
    - Percent Composition (for ethylene glycol/water and propylene glycol/water)
    - Volumetric flow rate
- Inlet Pressure
- Plate-Tube Bond Conductivity

Child Window 6 – Equations from Test

Figure A.7 shows the window where the results of the experimental test (if available) are entered. To compare the experimental results with the calculated results, the coefficients of the equations should be filled out and ‘Yes’ should be chosen from

the drop down menu at the lower-right corner.

Child Window 7 - Results

The test result window is shown in Figure A.8. This window summarizes the results calculated by CoDePro with the given test conditions and collector configuration. It shows the efficiency equations with calculated coefficients and the incident angle modifier equation with calculated incident angle modifier coefficient. There are buttons for plot windows and stagnation temperatures. Plot windows display collector performance and incident angle modifier as shown in Figure A.9 and A.10. The graph for each data set is distinguished by color. The stagnation temperature window (Figure A. 11) shows the conditions that affect the stagnation temperatures and stagnation temperatures of the cover and the plate.

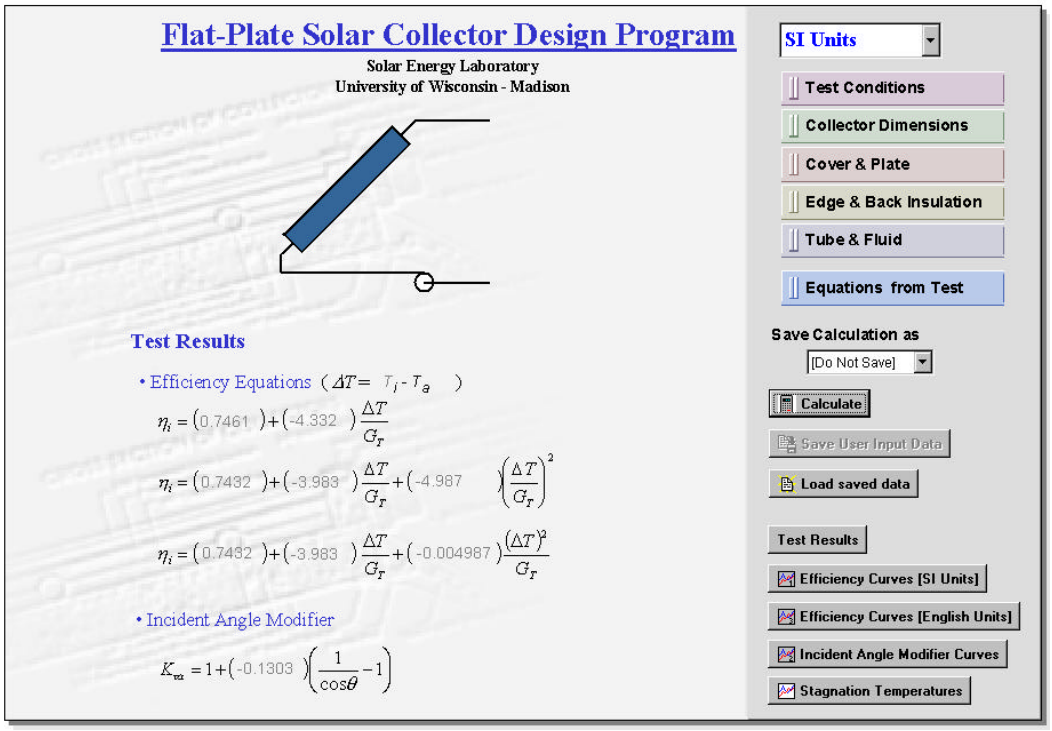

Figure A.1 Main Diagram Window

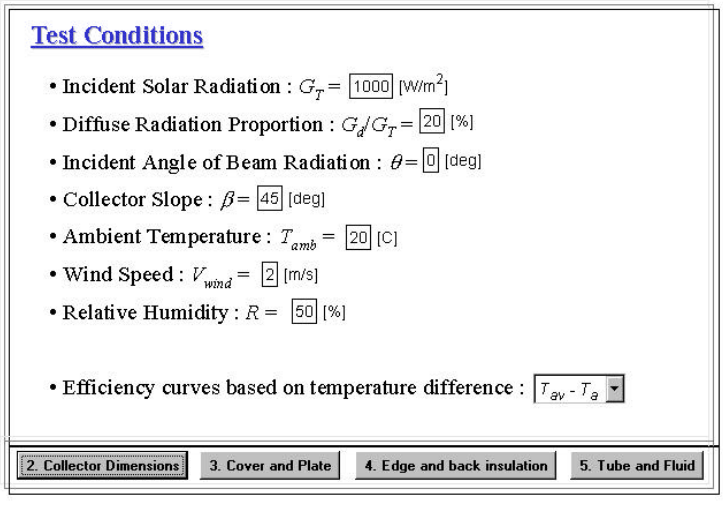

Figure A.2 Child window 1 - Test conditions

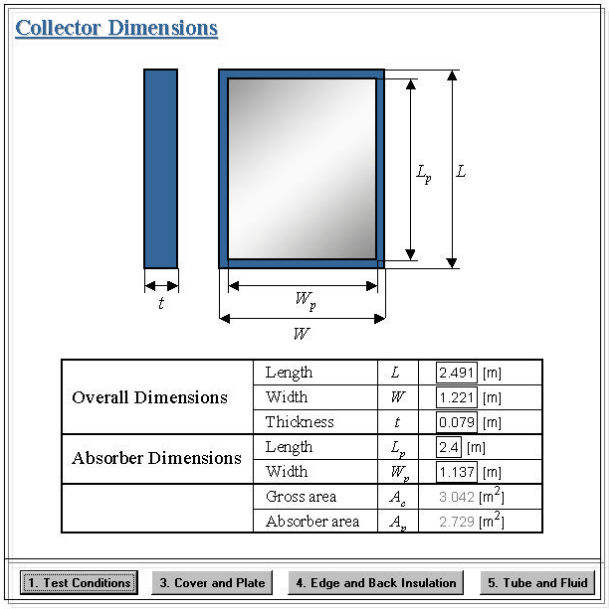

Figure A.3 Child window 2 - Collector dimensions

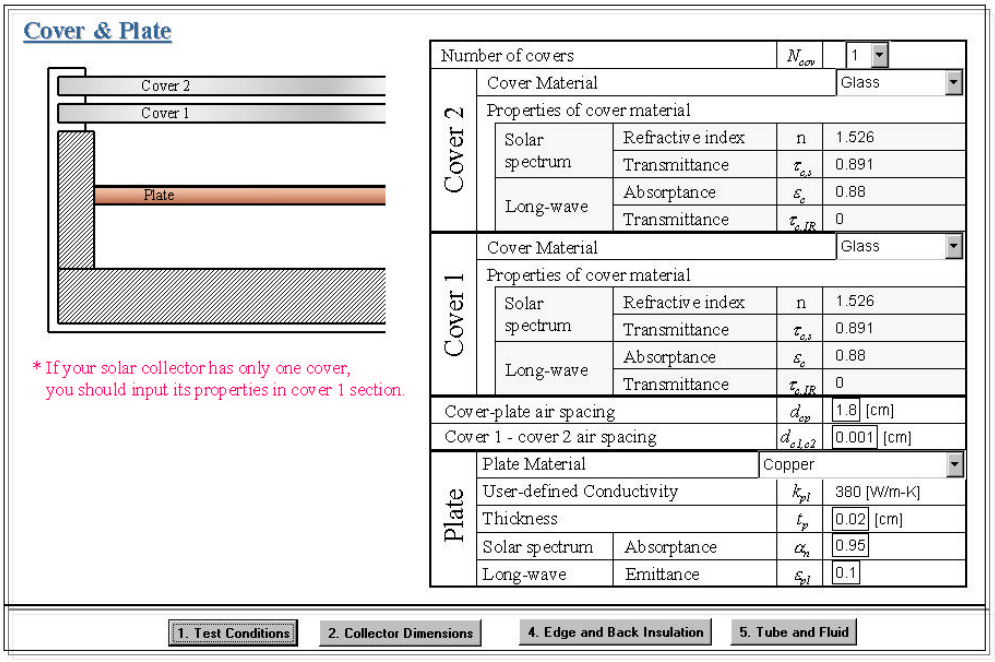

Figure A.4 Child window 3 - Cover and Plate

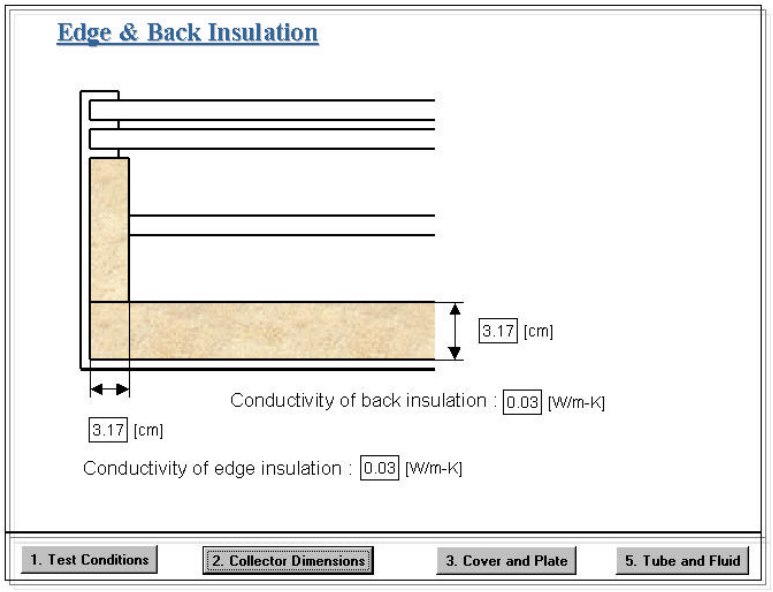

Figure A.5 Child window 4 – Edge and back insulation

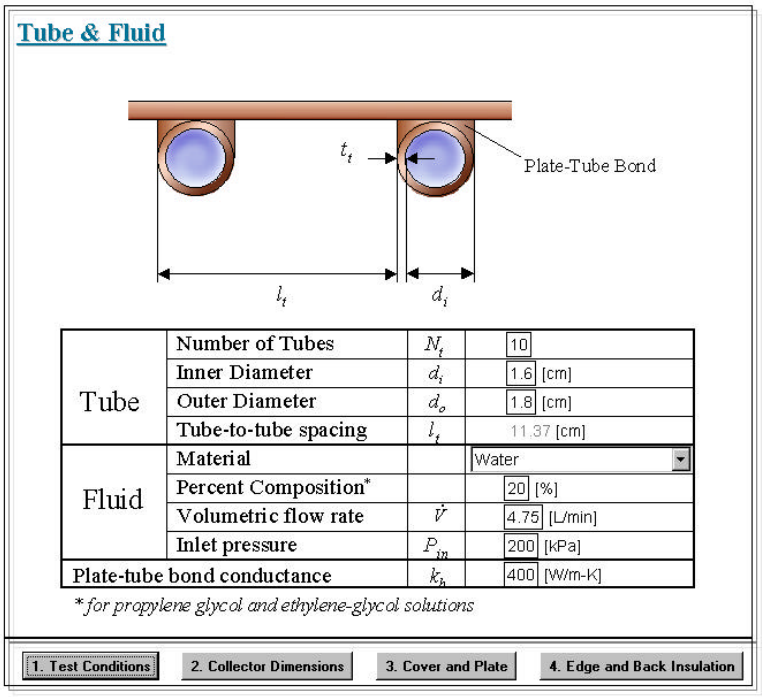

Figure A.6 Child window 5 – Tube and fluid

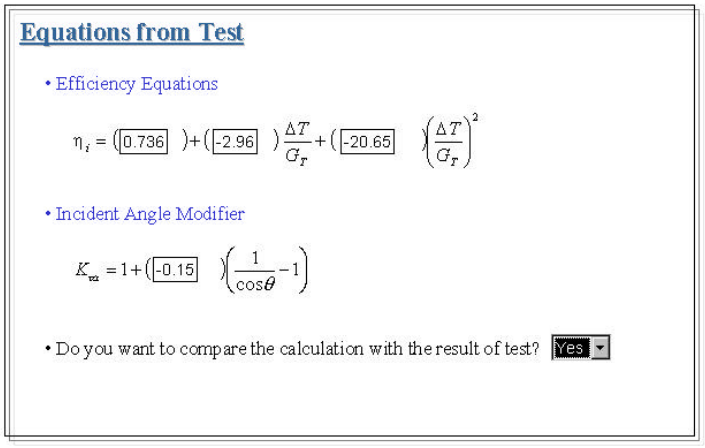

Figure A.7 Child window 6 – Equations from test

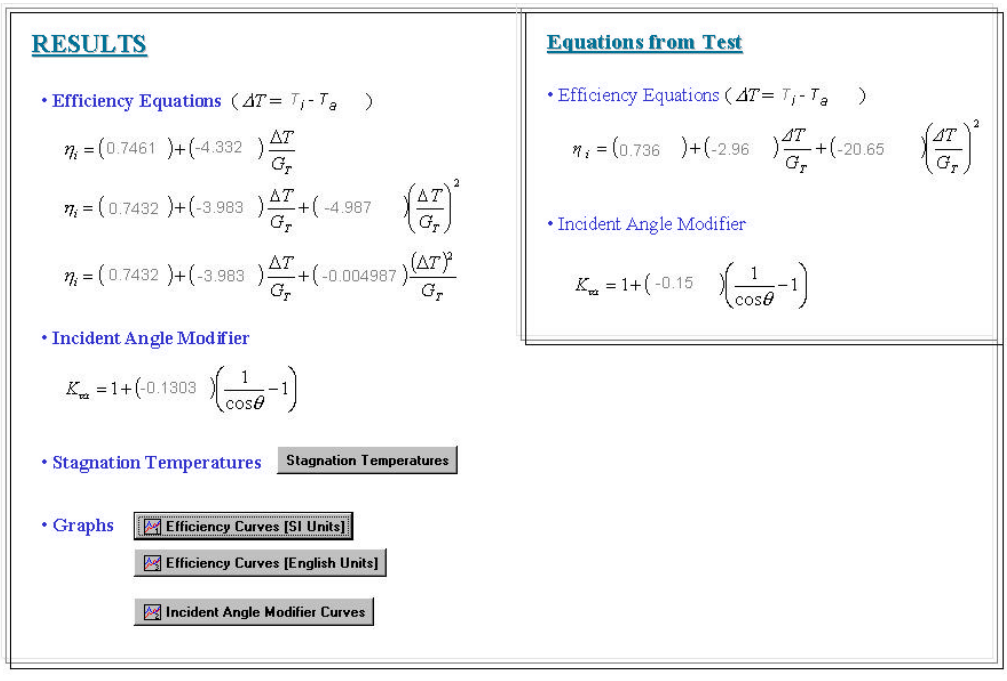

Figure A.8 Child window 7 –Results

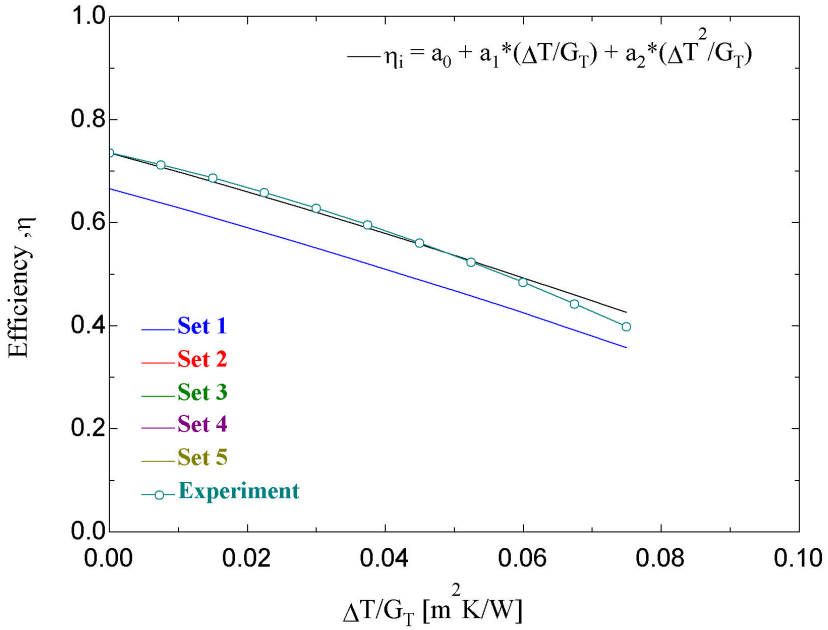

Figure A.9 Plot window – Instantaneous Efficiency

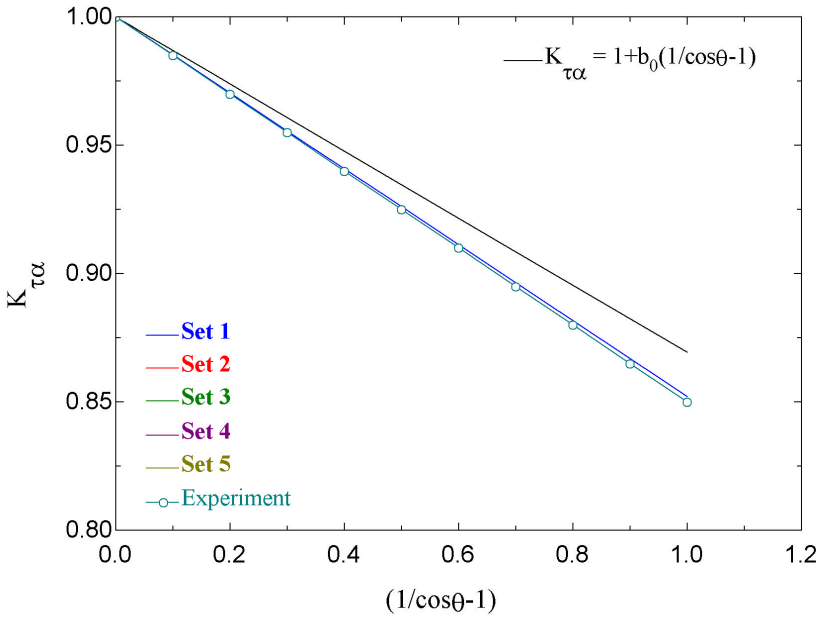

Figure A.10 Plot window - Incidence angle modifier

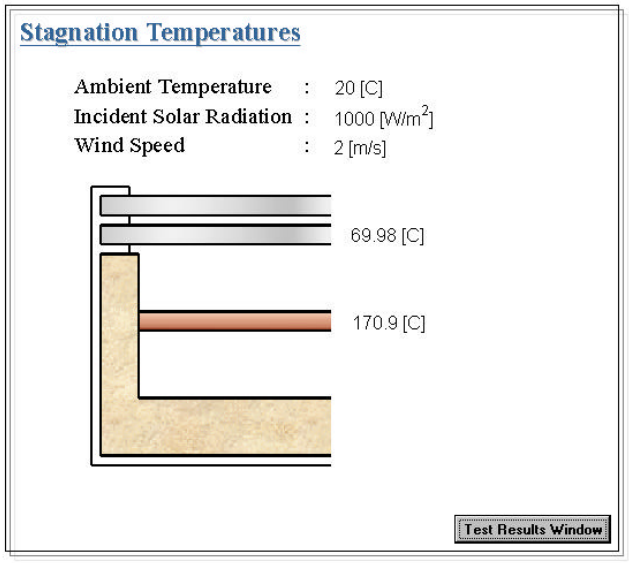

Figure A.11 Child window 8 - Stagnation temperatures

CODEPRO EES CODE

```
"CoDePro : Flat-Plate Solar Collector Design Program"

"Jae-Mo Koo, Research Assistant (koo@sel.me.wisc.edu)
William A. Beckman, Professor (beckman@me.engr.wisc.edu)
Solar Energy Laboratory
University of Wisconsin - Madison
CoDePro Ver. 0725, 12/16/1999
"

"! Unit Conversion ================================================="

Procedure Unit_Conv_11_(Unit_Sys$, G_T_inp, T_amb_inp, WindSp_inp : G_T, T_amb,
WindSp)
     If (Unit_Sys$ = 'SI Units') then
          G_T = G_T_inp
          T_amb = T_amb_inp
          WindSp = WindSp_inp
     Endif
     If (Unit_Sys$ = 'English Units') then
          G_T = G_T_inp*convert(Btu/hr-ft^2, W/m^2)
          T_amb = ConvertTemp('F', 'C', T_amb_inp)
          WindSp = WindSp_inp*convert(ft/s,m/s)
     Endif
End

Procedure Unit_Conv_12_(Unit_Sys$,Length_inp,Width_inp, Thickness_inp, Length_p_inp,
Width_p_inp: Length, Width, Thickness, Length_p, Width_p)
     If (Unit_Sys$ = 'SI Units') then
          Length = Length_inp
          Width = Width_inp
          Thickness = Thickness_inp
          Length_p = Length_p_inp
          Width_p = Width_p_inp
     Endif
     If (Unit_Sys$ = 'English Units') then
          Length = Length_inp*convert(ft,m)
          Width = Width_inp*convert(ft,m)
          thickness = thickness_inp*convert(ft,m)
          Length_p = Length_p_inp*convert(ft,m)
          Width_p = Width_p_inp*convert(ft,m)
     Endif
End

Procedure  Unit_Conv_13_(Unit_Sys$, d_cp_inp, d_c1c2_inp, cond_pl_inp, thick_pl_inp :
d_cp, d_c1c2, cond_pl, thick_pl)
     If (Unit_Sys$ = 'SI Units') then
          d_cp = d_cp_inp*convert(cm,m)
          d_c1c2 = d_c1c2_inp*convert(cm,m)
          cond_pl = cond_pl_inp
          thick_pl = thick_pl_inp*convert(cm,m)
```

```
    Endif
    If (Unit_Sys$ = 'English Units') then
        d_cp = d_cp_inp*convert(in, m)
        d_c1c2 = d_c1c2_inp*convert(in, m)
        Cond_pl = Cond_pl_inp*convert(Btu/hr-ft-F, W/m-K)
        Thick_pl = Thick_pl_inp*convert(in, m)
    Endif
End

Procedure Unit_Conv_14_(Unit_Sys$,t_e_inp,t_b_inp,k_e_inp,k_b_inp: t_e,t_b,k_e,k_b)
    If (Unit_Sys$ = 'SI Units') then
        t_e = t_e_inp*convert(cm,m)
        t_b = t_b_inp*convert(cm,m)
        k_e = k_e_inp
        k_b = k_b_inp
    Endif
    If (Unit_Sys$ = 'English Units') then
        t_b = t_b_inp*convert(in, m)
        t_e = t_e_inp*convert(in,m)
        k_b = k_b_inp*convert(Btu/hr-ft-F, W/m-K)
        k_e = k_e_inp*convert(Btu/hr-ft-F, W/m-K)
    Endif
End

Procedure Unit_Conv_15_(Unit_Sys$, TubeDiam_i_inp, TubeDiam_o_inp, Q_dot_inp,
P_in_inp, BondCond_inp :  TubeDiam_i, TubeDiam_o, Q_dot, P_in, BondCond)
    If (Unit_Sys$ = 'SI Units') then
        TubeDiam_i = TubeDiam_i_inp*convert(cm,m)
        TubeDiam_o = TubeDiam_o_inp*convert(cm,m)
        Q_dot = Q_dot_inp*convert(L/min, m^3/s)
        P_in = P_in_inp
        BondCond = BondCond_inp
    Endif
    If (Unit_Sys$ = 'English Units') then
        TubeDiam_i = TubeDiam_i_inp*convert(in, m)
        TubeDiam_o = TubeDiam_o_inp*convert(in, m)
        Q_dot = Q_dot_inp*convert(Gal/min, m^3/s)
        P_in = P_in_inp*convert(psi,kPa)
        BondCond = BondCond_inp*convert(Btu/hr-ft-F, W/m-K)
    Endif
End

Procedure Unit_Conv_22_(Unit_Sys$, A_p, A_c : A_p_out, A_c_out)
    If (Unit_Sys$ = 'SI Units') then
        A_p_out = A_p
        A_c_out = A_c
    Endif
    If (Unit_Sys$ = 'English Units') then
        A_c_out = A_c*convert(m^2, ft^2)
        A_p_out = A_p*convert(m^2, ft^2)
    Endif
End
```

```
Procedure Unit_Conv_25_(Unit_Sys$, TubeSp : TubeSp_out)
    If (Unit_Sys$ = 'SI Units') then
        TubeSp_out = TubeSp*convert(m,cm)
    Endif
    If (Unit_Sys$ = 'English Units') then
        TubeSp_out = TubeSp*convert(m, in)
    Endif
End


"! Preparation ============================================================="

"
Function to check the collector dimension"
Function check_1_(Width,Length, Thickness, Width_p, Length_p, d_cp, d_c1c2, thick_pl, TubeDiam_i, TubeDiam_o, t_b)
    if (Width<Width_p) then CALL ERROR('Your width of absorber (Width_p) is greater than the width of collector (Width).',Width_p)
    if (Length<Length_p) then CALL ERROR('Your length of absorber (Length_p) is greater than the length of collector (Legnth).',Length_p)
    thick_tot = d_cp + d_c1c2 + thick_pl + TubeDiam_i + t_b
    if (Thickness<thick_tot) then CALL ERROR('Your collector thickness (Thickness) is less than the thickness calculated from your data (d_cp + d_c1c2 + thick_pl + TubeDiam_i + t_b).',Thickness)
    if (TubeDiam_o<TubeDiam_i) then CALL ERROR('Outer diameter of tubes should be larger than inner diameter.',TubeDiam_o)
    check_1_ = 0
End

"
Function to check optical properties"
Function check_2_(epsilon_IR_c1, tau_IR_c1, epsilon_IR_c2, tau_c_IR_2)
    Dum1 = epsilon_IR_c1+tau_IR_c1
    Dum2 = epsilon_IR_c2+tau_c_IR_2
    if (Dum1 > 1) then CALL ERROR('Emittance + transmittance of Cover1 should be less than or equal to 1.',epsilon_IR_c1)
    if (Dum2 > 1) then CALL ERROR('Emittance + transmittance of Cover2 should be less than or equal to 1.',epsilon_IR_c2)
    check_2_ = 0
End

"
Procedure to set the temperature range for simulation"
Procedure T_bound_(T_amb, G_T, Fluid$, P_in : T_in_max, T_in_min)
    T_max = T_amb+0.1*G_T
    T_sat1 = 100.0
    T_in_max = min(99.9, T_max)
    T_in_min = max(0,T_amb)
END


"! Thermodynamic Properties ============================================="
```

```
"
Property Procedures for Propylene glycol and Ethylene glycol solution: From BRINEPROP"
Procedure Prop_Coeff_(Fl, Pr: c[1..18])
      if (Fl = 1) then
            If (Pr=1) then
                  c[1] = -2.196E+0001;c[2] =  9.186E-0006 ;c[3] = -1.136E-0006 ;c[4] =
1.798E-0008 ;c[5] = -1.015E+0000 ;c[6] = -3.391E-0006 ;c[7] =  7.548E-0008 ;c[8] =
8.493E-0010 ;c[9] = -1.414E-0002
                  c[10]=  2.355E-0007;c[11]=  2.159E-0009;c[12]= -1.311E-0010;c[13]= -
3.727E-0005;c[14]=  6.091E-0010;c[15]= -1.465E-0010;c[16]= -1.398E-0007;c[17]= -
2.269E-0010;c[18]= -2.512E-0008
            Endif
            If (Pr=2) then
                  c[1] =  1.056E+0003;c[2] = -3.987E-0001;c[3] = -3.068E-0003;c[4] =  1.233E-
0005;c[5] =  1.505E+0000;c[6] = -8.953E-0003;c[7] =  6.378E-0005;c[8] = -1.152E-
0007;c[9] = -1.634E-0003
                  c[10]=  1.541E-0004;c[11]= -1.874E-0006;c[12]= -9.809E-0009;c[13]= -
2.317E-0004;c[14]=  2.549E-0006;c[15]= -5.523E-0008;c[16]= -8.510E-0006;c[17]= -
3.848E-0008;c[18]= -1.128E-0007
            Endif
            If (Pr=3) then
                  c[1] =  3.501E+0003;c[2] =  3.954E+0000;c[3] =  6.065E-0005;c[4] = -5.979E-
0006;c[5] = -2.419E+0001;c[6] =  1.031E-0001;c[7] =  4.312E-0005;c[8] =  5.168E-
0006;c[9] =  4.613E-0003
                  c[10]= -6.595E-0005;c[11]=  1.620E-0005;c[12]= -3.250E-0007;c[13]=
6.028E-0003;c[14]=  5.642E-0005;c[15]= -7.777E-0007;c[16]= -7.977E-0005;c[17]=
5.190E-0007;c[18]= -3.380E-0006
            Endif
            If (Pr=4) then
                  c[1] =  4.211E-0001;c[2] =  7.995E-0004;c[3] = -5.509E-0008;c[4] = -1.460E-
0008;c[5] = -3.694E-0003;c[6] = -1.751E-0005;c[7] =  6.656E-0008;c[8] =  2.017E-0009;c[9]
=  2.095E-0005
                  c[10]=  2.078E-0007;c[11]= -2.394E-0009;c[12]= -6.772E-0011;c[13]=
3.663E-0007;c[14]= -5.272E-0009;c[15]= -1.126E-0010;c[16]= -6.389E-0009;c[17]= -
1.112E-0010;c[18]= -1.820E-0010
            Endif
            If (Pr=5) then
                  c[1] =  1.453E+0000;c[2] = -3.747E-0002;c[3] =  2.842E-0004;c[4] = -8.025E-
0007;c[5] =  2.920E-0002;c[6] = -1.131E-0004;c[7] =  1.729E-0006;c[8] = -5.073E-0008;c[9]
=  1.264E-0004
                  c[10]=  6.785E-0009;c[11]= -1.685E-0008;c[12]= -1.082E-0009;c[13]=
4.386E-0006;c[14]= -2.191E-0007;c[15]= -9.117E-0011;c[16]= -9.223E-0008;c[17]= -
4.294E-0009;c[18]= -3.655E-0009
            Endif
      Endif

      if (Fl=2) then
            if (Pr=1) then
                  c[1] = -2.360E+0001 ;c[2] = -2.970E-0004 ;c[3] =  1.871E-0007 ;c[4] =
1.590E-0007 ;c[5] = -1.088E+0000 ;c[6] =  4.693E-0005 ;c[7] = -9.089E-0007 ;c[8] = -
9.845E-0010 ;c[9] = -2.815E-0002
```

```
            c[10]=  1.613E-0006 ;c[11]=  5.066E-0008 ;c[12]= -1.251E-0009 ;c[13]= -
5.285E-0004 ;c[14]= -1.786E-0007 ;c[15]=  2.051E-0009 ;c[16]=  9.794E-0006 ;c[17]= -
5.668E-0009 ;c[18]=  4.270E-0007
        Endif
        If (Pr=2) then
            c[1] =  1.042E+0003;c[2] = -4.907E-0001;c[3] = -2.819E-0003;c[4] = -5.895E-
0007;c[5] =  8.081E-0001;c[6] = -9.652E-0003;c[7] =  7.168E-0005;c[8] =  2.404E-0007;c[9]
= -7.156E-0003
            c[10]=  1.088E-0004;c[11]= -3.328E-0006;c[12]=  1.153E-0007;c[13]=
1.190E-0004;c[14]= -6.226E-0006;c[15]= -3.026E-0008;c[16]= -1.170E-0005;c[17]= -
2.915E-0007;c[18]= -6.033E-0007
        Endif
        If (Pr=3) then
            c[1] =  3.679E+0003;c[2] =  1.571E+0000;c[3] =  1.331E-0002;c[4] =  1.975E-
0007;c[5] = -1.933E+0001;c[6] =  1.118E-0001;c[7] = -1.108E-0003;c[8] =  4.924E-
0006;c[9] = -4.879E-0002
            c[10]= -2.338E-0004;c[11]=  2.753E-0005;c[12]= -3.148E-0007;c[13]=
4.749E-0003;c[14]= -2.621E-0005;c[15]=  1.286E-0006;c[16]= -2.871E-0004;c[17]= -
9.050E-0008;c[18]= -1.068E-0005
        Endif
        if (Pr=4) then
            c[1] =  3.806E-0001;c[2] =  5.765E-0004;c[3] = -3.477E-0007;c[4] = -6.041E-
0009;c[5] = -3.815E-0003;c[6] = -1.423E-0005;c[7] = -1.203E-0008;c[8] = -5.854E-0010;c[9]
=  8.420E-0006
            c[10]=  1.081E-0007;c[11]=  1.959E-0009;c[12]=  1.271E-0010;c[13]= -
1.110E-0006;c[14]= -1.612E-0009;c[15]=  3.005E-0010;c[16]=  5.503E-0009;c[17]=
1.437E-0010;c[18]=  1.290E-0009
        Endif
        if (Pr=5) then
            c[1] =  2.274E+0000;c[2] = -5.342E-0002;c[3] =  5.372E-0004;c[4] = -4.955E-
0006;c[5] =  4.500E-0002;c[6] = -5.488E-0004;c[7] =  1.845E-0006;c[8] =  1.192E-0007;c[9]
= -7.808E-0005
            c[10]=  1.453E-0006;c[11]= -2.816E-0007;c[12]=  8.562E-0009;c[13]=
6.565E-0006;c[14]= -4.032E-0007;c[15]= -1.212E-0009;c[16]=  6.441E-0007;c[17]= -
1.430E-0008;c[18]=  1.092E-0008
        Endif
    Endif
End

Procedure Prop_Mean_(Fl:xm, ym)
    If (Fl=1) then
        xm = 38.1615
        ym = 6.3333
    Endif
    If (Fl=2) then
        xm = 42.7686
        ym = 5.3571
    Endif
End


Function BRINEPROP_(Fl$,Pr$,Conc,Temp)
```

```
    IF (Fl$='EG') AND (Conc<0) THEN
        Call ERROR('Concentration must be equal to or larger than 0.0%.',Conc)
        ELSE
        IF (Fl$='EG') AND (Conc>56.1) THEN
            Call ERROR('Concentration must be equal to or smaller than 56.1%.',Conc)
        ELSE
        IF (Fl$='PG') AND (Conc<15.2) THEN
            Call ERROR('Concentration must be equal to or larger than 15.2%.',Conc)
        ELSE
        IF (Fl$='PG') AND (Conc>57.0) THEN
            Call ERROR('Concentration must be equal to or smaller than 57.0%.',Conc)
    ENDIF;ENDIF;ENDIF;ENDIF

    "Sets property parameter"
    Freeze=1;Density=2;SpecHeat=3;ThermalC=4;DynVisc=5
    IF (Pr$='Freeze') THEN Pro:=Freeze ELSE
        IF (Pr$='Density') THEN Pro:=Density ELSE
            IF (Pr$='SpecHeat') THEN Pro:=SpecHeat ELSE
                IF (Pr$='ThermalC') THEN Pro:=ThermalC ELSE
                    IF (Pr$='DynVisc') THEN Pro:=DynVisc
    ENDIF;ENDIF;ENDIF;ENDIF;
    Pr:=Pro

    "Sets fluid parameter"
    EG=1;PG=2
    IF (Fl$='EG') THEN Flu:=EG ELSE
    IF (Fl$='PG') THEN Flu:=PG ELSE
    ENDIF;ENDIF
    Fl:=Flu

    "Coefficients"
    call Prop_Coeff_(Fl, Pr: c[1..18])

    "Mean values"
    call Prop_Mean_(Fl:xm, ym)

    x=Conc-xm
    y=Temp-ym

    "Equation, f(temp,conc)"
   Funkt=c[1]+(c[2]*y)+(c[3]*y^2)+(c[4]*y^3)+(c[5]*x)+(c[6]*x*y)+(c[7]*x*y^2)+(c[8]*x*y^3)+
(c[9]*x^2)+(c[10]*x^2*y)+(c[11]*x^2*y^2)+(c[12]*x^2*y^3)+(c[13]*x^3)+(c[14]*x^3*y)+(c[15]*x
^3*y^2)+(c[16]*x^4)+(c[17]*x^4*y)+(c[18]*x^5)

    "Conditions
    Mode=mu   =>  output = exp(f)
    Mode<>mu =>  output = f"

    IF (Pr=5) THEN
        BRINEPROP_=exp(Funkt)
    ELSE
    IF (Pr=3) THEN
        BRINEPROP_=Funkt/1000
```

```
    ELSE
        BRINEPROP_=Funkt
    ENDIF;ENDIF
END


"
User External Functions to calculate properties of fluid"
Function C_p_(Fluid$, T, P, Percent)
    If (Fluid$ = 'Water') then C_p_ = SPECHEAT(Water,T=T,P=P)
    If (Fluid$ = 'Propylene Glycol Solution') then C_p_ =
BRINEPROP_('PG','SpecHeat',Percent,T)
    If (Fluid$ = 'Ethylene-Glycol Solution') then C_p_ =
BRINEPROP_('EG','SpecHeat',Percent,T)
End

Function mu_(Fluid$, T, P, Percent)
    If (Fluid$ = 'Water') then mu_ = VISCOSITY(Water,T=T,P=P)
    If (Fluid$ = 'Propylene Glycol Solution') then mu_ =
BRINEPROP_('PG','DynVisc',Percent,T)*1.0e-3
    If (Fluid$ = 'Ethylene-Glycol Solution') then mu_ =
BRINEPROP_('EG','DynVisc',Percent,T)*1.0e-3
End

Function k_(Fluid$, T, P, Percent)
    If (Fluid$ = 'Water') then k_ = CONDUCTIVITY(Water,T=T,P=P)
    If (Fluid$ = 'Propylene Glycol Solution') then k_ =
BRINEPROP_('PG','ThermalC',Percent,T)
    If (Fluid$ = 'Ethylene-Glycol Solution') then k_ =
BRINEPROP_('EG','ThermalC',Percent,T)
End

Function rho_(Fluid$, T, P, Percent)
    If (Fluid$ = 'Water') then rho_ = DENSITY(Water,T=T,P=P)
    If (Fluid$ = 'Propylene Glycol Solution') then rho_ =
BRINEPROP_('PG','Density',Percent,T)
    If (Fluid$ = 'Ethylene-Glycol Solution') then rho_ =
BRINEPROP_('EG','Density',Percent,T)
End


"! Heat Transfer Relations ======================================="

"
Heat transfer coefficient and pressure drop in circular tubes"
Procedure Circular_tube_(Q_dot, D, L, Fluid$, T, P, SolPer, Num_tube : m_dot, C_p, h,
DELTAP, Re, DELTAP_1, DELTAP_2)
    T = min(100, T)
    T = max(0, T)
    rho = rho_(Fluid$, T, P, SolPer)
    m_dot = rho*Q_dot    "Mass flow rate"
    C_p = C_p_(Fluid$, T, P, SolPer)*convert(kJ, J)
    If (C_p  < 0.01) then C_p = 4000
```

```
    mu = mu_(Fluid$, T, P, SolPer)
    k = k_(Fluid$, T, P, SolPer)
    Re = (4*m_dot/Num_tube)/(pi*D*mu)
    Pr = mu*C_p/k
    A_inner = pi*D^2/4*Num_tube
    u = Q_dot/A_inner

    if (Re > 2300) then "for turbulent flow"
        fricfac = (0.79*ln(Re)-1.64)^(-2)
        Nusselt_long = ((fricfac/8)*(Re-1000)*Pr)/(1+12.7*sqrt(fricfac/8)*(Pr^(2/3)-1))
    "Gnielinski"
        Nusselt = Nusselt_long*(1+(D/L)^0.7)
    else "For laminar flow"
        fricfac = 64/Re
    "for laminar flow (assumption : constant heat flux condition and fully developed flow)"
        a = 0.00172
        b = 0.00281
        m = 1.66
        n = 1.29
        Nusselt = 4.4+a*(Re*Pr*D/L)^m/(1+b*(Re*Pr*D/L)^n)
    Endif

    "Heat Transfer Coefficient"
    h = Nusselt*k/D
    if (h <0) then h = 5

    "Pressure Drop"
    DELTAP_1 = fricfac*(rho*u^2)/(2*D)*L   "Pressure drop of fully developed flow"
    K = 0.5  "Loss Coefficient for Sharp edge"
    DELTAP_2 = K*D*(rho*u^2)/(2*D)  "Pressure drop due at pipe entrance : Sharp
Edge"
    DELTAP = DELTAP_1+DELTAP_2
END

"
Function for calculating Nusselt number for natural convection between parallel plates"
Function NuFlatPlate_(Slope,T1,T2,PlateSp,WLDP1,WLDP2)
    Slope`=if(Slope,0,.001,.001,Slope)
    Slope``=if(Slope`,75,Slope`,Slope`,75)
    Tmean_1=(T1+T2)/2
    Tmean =  min(500, Tmean_1)
    Tmean = max(-5, Tmean)                    "For convergence"
    TmeanK=Tmean+273*unitsystem('C')
    k=Conductivity(Air,T=Tmean)
    Rho=1/Volume(Air,T=Tmean,P=101.3)
    Nu=Viscosity(Air,T=Tmean)/Rho
    Cp=SpecHeat(Air,T=Tmean)*1000
    Alpha=k/(Rho*Cp)
    Ra=max(10,9.81*abs(T2-T1)*PlateSp^3/(TmeanK*Nu*Alpha))
    if (WLDP1 < 0.5) then
        a=1.44*(1-1708*(sin(1.8*Slope``)^1.6)/(Ra*cos(Slope``)))*max(0,1-
(1708/(Ra*cos(Slope``))))  "Hollands et al., 1976"
    else
```

```
        a=1.77*(1-1296*(sin(1.8*Slope``)^1.6)/(Ra*cos(Slope``)))*max(0,1-
(1296/(Ra*cos(Slope``))))  "Yiqin et al., 1991 for plastic cover 1 and glass cover 2"
    Endif
    b=max(0,(Ra*cos(Slope``)/5830)^(1/3)-1)
    NuFlatPlate_=1+a+b
END


"! Absorbed Radiation ====================================="

"
Function to calculate Sky temperature"
Function SkyTemp_JK_(T_amb,T_dp,HrAng)
    Time=(180-HrAng) "sky emittance always greater at night"
    T_amb`=T_amb+273.15
    T_dp`=T_dp
    SkyTemp_JK_=(T_amb`*(.711+.0056*T_dp`+.000073*T_dp`^2+.013*cos(Time))^.25)
END

"
Function to evaluate the angluar dependence of absorptance of the absorber plate"
Function Abs\Abs_n_(IncAng)
    Abs\Abs_n_=1.0000-1.5879e-3*IncAng+2.7314e-4*IncAng^2-2.3026e-5*IncAng^3
+9.0244e-7*IncAng^4-1.8000e-8*IncAng^5+1.7734e-10*IncAng^6-6.9937e-13*IncAng^7
END

"
Function to calculate perpendicular and parallel components of optical properties"
Procedure OpticalProp1_(IncAng,KL,n : TRANSperp, TRANSpara, RHOperp, RHOpara,
ABSperp, ABSpara)
    IncAng = max(IncAng, 0.0001)      "To make possible 0 as an input of incident angle"
    RefrAng = arcsin(sin(IncAng)/n)
    REFperp=(sin(RefrAng-IncAng)/sin(RefrAng+IncAng))^2
    REFpara=(tan(RefrAng-IncAng)/tan(RefrAng+IncAng))^2
    TRANSabs=exp(-KL/cos(RefrAng))
    TRANSperp = TRANSabs*(1-REFperp)^2/(1-(REFperp*TRANSabs)^2)
    TRANSpara = TRANSabs*(1-REFpara)^2/(1-(REFpara*TRANSabs)^2)
    RHOperp = REFperp*(1+TRANSabs*TRANSperp)
    RHOpara = REFpara*(1+TRANSabs*TRANSpara)
    ABSperp = (1-TRANSabs)*(1-REFperp)/(1-REFperp*TRANSabs)
    ABSpara = (1-TRANSabs)*(1-REFpara)/(1-REFpara*TRANSabs)
END

"
Function to classify the kind of covers"
Procedure OpticalProp3_(tau_IR, epsilon_IR: WLDP, TRANS_IR, Emitt_IR)
    if (tau_IR=0) then WLDP = 0
    if (tau_IR>0) then WLDP = 1
    TRANS_IR=tau_IR
    Emitt_IR = epsilon_IR
End

"
```

```
Function calculating transmittance-absorptance product"
Function TauAlpha_(Ncov,IncAng, KL_c1, KL_c2, n_c1, n_c2, Alpha_n)
    if (Ncov = 0) then
        Tau = 1
        rho_d = 0
    Endif

    if (Ncov = 1) then
        Call OpticalProp1_(IncAng,KL_c1,n_c1 : TRANSperp, TRANSpara,a, b,c,d)
        tau = (TRANSperp+TRANSpara)/2
        Call OpticalProp1_(60,KL_c1,n_c1 : a, b, RHOperp, RHOpara,c,d)
        rho_d = (RHOperp+RHOpara)/2
    Endif

    if (Ncov = 2) then
        Call OpticalProp1_(IncAng,KL_c1,n_c1: TRANSperp1,TRANSpara1,RHOperp1,
RHOpara1,c,d)
        Call OpticalProp1_(IncAng,KL_c2,n_c2: TRANSperp2,TRANSpara2,RHOperp2,
RHOpara2,c,d)
        tau=((TRANSperp2*TRANSperp1)/(1-RHOperp2*RHOperp1)+(TRANSpara2
*TRANSpara1)/(1-RHOpara2*RHOpara1))/2
        Call OpticalProp1_(60,KL_c1,n_c1:TRANSperp1d,TRANSpara1d,RHOperp1d,
RHOpara1d,c,d)
        Call OpticalProp1_(60,KL_c2,n_c2:TRANSperp2d,TRANSpara2d,RHOperp2d,
RHOpara2d,c,d)
        tau_d=((TRANSperp2d*TRANSperp1d)/(1-RHOperp2d*RHOperp1d)+
(TRANSpara2d*TRANSpara1d)/(1-RHOpara2d*RHOpara1d))/2
        rho_d=((RHOperp2d+(tau_d*RHOperp1d*TRANSperp2d)/TRANSperp1d)
+(RHOpara1d+(tau_d*RHOpara1d*TRANSpara2d)/TRANSpara1d))/2
    Endif
    alpha = Abs\Abs_n_(IncAng)*Alpha_n
    TauAlpha_ = (tau*alpha)/(1-(1-Alpha)*rho_d)
END


"! Loss Coefficient ======================================="

"
Function to calculate the wind coefficient"
Function WindCoeff`FS_(WindSp,Length,Width,T_amb,Pr)
    CharLength=4*Length*Width/(2*(Length+Width))
    Density=1/Volume(Air, T=T_amb, P=101.3)
    Visc=Viscosity(Air, T=T_amb)
    Re=CharLength*WindSp*Density/Visc
    Nu=.86*Re^.5*Pr^.333
    WindCoeff`FS_=max(5,Nu*Conductivity(Air, T=T_amb)/CharLength)
END


"
Calculation of the Exact Top Loss Coefficient"
"No Cover"
Procedure U_T_0_(T_pl,T_amb,T_s`,Slope,WindCoeff,epsilon_IR_p: U_t, T_c1, T_c2)
```

```
      T_pl`=T_pl+273
      q_r_pa = epsilon_IR_p*5.67e-8*(T_pl`^4-T_s`^4)
      q_c_pa = WindCoeff*(T_pl-T_amb)
      q_loss_t = q_r_pa + q_c_pa
      U_t = q_loss_t/(T_pl-T_amb)
      T_c1 = 0
      T_c2 = 0
END

"One Cover"
Procedure U_T_1_(T_pl,T_amb,T_s`,Slope,WindCoeff,Emitt_IR_p,Emitt_IR_c1,d_cp,
WLDP1, TRANS_IR_c1: U_t, T_c1, T_c2)
      REF_IR_p  =1-Emitt_IR_p
      REF_IR_c1 = 1-(Emitt_IR_c1+TRANS_IR_c1)
      T_amb` = T_amb+273.15
      T_pl`=T_pl+273.15
      T_c`_new = (T_pl` + T_s`)/2
      q_1_o = 400
      REPEAT
            T_c` = T_c`_new
            T_c = T_c`-273.15
            Nu = NuFlatPlate_(Slope,T_c,T_pl,d_cp,0,0)
            T_mean = Max(-10, (T_pl+T_c)/2)
            T_mean = Min(500, T_mean)
            h_c_pc = Nu*Conductivity(Air,T=T_mean)/d_cp
            q_c_pc = h_c_pc*(T_pl-T_c)
            q_2_i = 5.67e-8*T_s`^4
            q_1_i = REF_IR_p*q_1_o+Emitt_IR_p*5.67e-8*T_pl`^4
            q_1_o = TRANS_IR_c1*q_2_i+REF_IR_c1*q_1_i+Emitt_IR_c1*5.67e-8*T_c`^4
            q_2_o = TRANS_IR_c1*q_1_i+REF_IR_c1*q_2_i+Emitt_IR_c1*5.67e-8*T_c`^4
            T_c`_new=T_c`+((q_1_i-q_1_o-q_2_o+q_2_i+h_c_pc*T_pl`+WindCoeff*T_amb`)
/(WindCoeff+h_c_pc)-T_c`)*0.3
            Error = abs(T_c`_new-T_c`)
      UNTIL (Error < 1.0e-4)
      q``_t = q_1_i-q_1_o+q_c_pc
      U_t = q``_t/(T_pl-T_amb)
      T_c1 = T_c`-273.15
      T_c2 = 0
End


"Two Covers"
Procedure U_T_2_(T_pl, T_amb, T_s`, Slope, WindCoeff, Emitt_IR_p, Emitt_IR_c1,
Emitt_IR_c2, Ncov,d_cp,d_c1c2, WLDP1, WLDP2, TRANS_IR_c1, TRANS_IR_c2,
xfraccov: U_t, T_c1, T_c2)
      REF_IR_p  =1-Emitt_IR_p
      REF_IR_c1 = 1-(Emitt_IR_c1+TRANS_IR_c1)
      REF_IR_c2 = 1-(Emitt_IR_c2+TRANS_IR_c2)
      T_amb` = T_amb+273.15
      T_pl`=T_pl+273.15
      T_c1`_new = (T_pl` + T_s`)/2
      T_c2`_new = T_c1`_new - 1
      q_1_o = 400; q_2_o = 400; q_3_o = 400
```

```
    REPEAT
        T_c1` = T_c1`_new
        T_c1 = T_c1`-273.15
        T_c2` = T_c2`_new
        T_c2 = T_c2`-273.15
        Nu_1 = NuFlatPlate_(Slope,T_c1,T_pl,d_cp,WLDP1,WLDP2)
        T_mean1 = Max(-10, (T_pl+T_c1)/2)
        T_mean1 = Min(500, T_mean1)
        h_c_pc1 = Nu_1*Conductivity(Air,T=T_mean1)/d_cp
        q_c_pc1 = h_c_pc1*(T_pl-T_c1)
        Nu_2 = NuFlatPlate_(Slope,T_c2,T_c1,d_c1c2,WLDP1,WLDP2)
        T_mean2 = Max(-10, (T_c1+T_c2)/2)
        T_mean2 = Min(500, T_mean2)
        h_c_c1c2 = Nu_2*Conductivity(Air,T=T_mean2)/d_c1c2
        q_c_c1c2 = h_c_c1c2*(T_c1-T_c2)
        q_1_i = REF_IR_p*q_1_o+Emitt_IR_p*5.67e-8*T_pl`^4
        q_2_i = q_3_o
        q_3_i = q_2_o
        q_4_i = 5.67e-8*T_s`^4
        q_1_o = TRANS_IR_c1*q_2_i+REF_IR_c1*q_1_i+Emitt_IR_c1*5.67e-8*T_c1`^4
        q_2_o = TRANS_IR_c1*q_1_i+REF_IR_c1*q_2_i+Emitt_IR_c1*5.67e-8*T_c1`^4
        q_3_o = TRANS_IR_c2*q_4_i+REF_IR_c2*q_3_i+Emitt_IR_c2*5.67e-8*T_c2`^4
        q_4_o = TRANS_IR_c2*q_3_i+REF_IR_c2*q_4_i+Emitt_IR_c2*5.67e-8*T_c2`^4
        T_c1`_new=T_c1`+((q_1_i-q_1_o-q_2_o+q_2_i+h_c_pc1*T_pl`+h_c_c1c2*T_c2`)
/(h_c_pc1+h_c_c1c2)-T_c1`)*0.2
        T_c2`_new=T_c2`+((q_3_i-q_3_o-q_4_o+q_4_i+h_c_c1c2*T_c1`+WindCoeff*
T_amb`)/(h_c_c1c2+WindCoeff)-T_c2`)*0.2
        Error = abs(T_c1`_new-T_c1`)+abs(T_c2`_new-T_c2`)
    UNTIL (Error < 1.0e-4)
    T_c1 = T_c1`-273.15
    T_c2 = T_c2`-273.15
    q``_t = q_1_i-q_1_o+q_c_pc1
    U_t = q``_t/(T_pl-T_amb)
End

"
Procedure to evaluate overall heat loss coefficient"
Procedure U_L_(T_pl,T_amb,T_s`,Slope,WindCoeff,Emitt_IR_p,Emitt_IR_c1,Emitt_IR_c2,
Ncov,U_b,U_e,d_cp,d_c1c2, WLDP1, WLDP2, TRANS_IR_c1, TRANS_IR_c2, xfraccov :
U_L,T_c1, T_c2 )

    If (Ncov = 0) then call U_T_0_(T_pl,T_amb,T_s`,Slope,WindCoeff,Emitt_IR_p: U_t,
T_c1, T_c2)
    if (Ncov=1) then call U_T_1_(T_pl,T_amb,T_s`,Slope,WindCoeff,Emitt_IR_p,
Emitt_IR_c1,d_cp, WLDP1, TRANS_IR_c1: U_t, T_c1, T_c2)
    if (Ncov = 2) then call U_T_2_(T_pl,T_amb,T_s`,Slope,WindCoeff,Emitt_IR_p,
Emitt_IR_c1,Emitt_IR_c2, Ncov,d_cp,d_c1c2, WLDP1, WLDP2, TRANS_IR_c1,
TRANS_IR_c2, xfraccov: U_t, T_c1, T_c2)

    U_L = max((U_t + U_b + U_e), U_b+U_e)
End

"! Solar Collector Gain ======================================="
```

```
"
Mean fluid temperature"
Function T_Fl`M_(T_in,Q_u,A,F_R,F``,U_L)
      T_Fl`M_=T_in+Q_u/(A*F_R*U_L)*(1-F``)
END

"
Mean plate temperature"
Function T_Pl`M_(T_in,Q_u,A,F_R,U_L)
      T_Pl`M_=T_in+Q_u/(A*F_R*U_L)*(1-F_R)
END

"
Collector efficiency factor"
Function CollEffFact_(U_L,Cond_pl,Thick_pl,TubeSp,TubeDiam_i,TubeDiam_o,BondCond,
h_fi)
      m=(U_L/(Cond_pl*Thick_pl))^.5
      F=tanh(m*(TubeSp-TubeDiam_o)/2)/(m*(TubeSp-TubeDiam_o)/2)
      CollEffFact_=(1/U_L)/(TubeSp*(1/(U_L*(TubeDiam_o+(TubeSp-TubeDiam_o)*F))+
1/BondCond+1/(pi*TubeDiam_i*h_fi)))
END

"
Collector heat removal factor"
Function F_R_(MassFR,C_P,A,U_L,CollEffFact)
      DimFlowRate=MassFR*C_P/(A*U_L*CollEffFact)
      F``=DimFlowRate*(1-exp(-1/DimFlowRate))
      F_R_=CollEffFact*F``
END


"!                                    Post                                    processing
============================================================"

"Procedure for the output in the stagnation temperature window"
Procedure Stag_disp(Unit_Sys$, Ncov, T_stag_p, T_stag_c1, T_stag_c2: T_stag_p$,
T_stag_c1$, T_stag_c2$)
      If (Unit_Sys$ = 'SI Units') then
            T_stag_p_out = T_stag_p
            T_stag_c1_out = T_stag_c1
            T_stag_c2_out = T_stag_c2
            if (Ncov = 0) then
                  T_stag_p$=CONCAT$(String$(T_stag_p_out), ' [C]')
                  T_stag_c1$='   '
                  T_stag_c2$='   '
            Endif
            if (Ncov = 1) then
                  T_stag_p$=CONCAT$(String$(T_stag_p_out), ' [C]')
                  T_stag_c1$=CONCAT$(String$(T_stag_c1_out), ' [C]')
                  T_stag_c2$='   '
            Endif
            if (Ncov = 2) then
```

```
            T_stag_p$=CONCAT$(String$(T_stag_p_out), ' [C]')
            T_stag_c1$=CONCAT$(String$(T_stag_c1_out), ' [C]')
            T_stag_c2$=CONCAT$(String$(T_stag_c2_out), ' [C]')
        Endif
    Endif

    If (Unit_Sys$ = 'English Units') then
        T_stag_p_out = ConvertTemp('C', 'F', T_stag_p)
        T_stag_c1_out = ConvertTemp('C', 'F', T_stag_c1)
        T_stag_c2_out = ConvertTemp('C', 'F', T_stag_c2)
        if (Ncov = 0) then
            T_stag_p$=CONCAT$(String$(T_stag_p_out), ' [F]')
            T_stag_c1$='   '
            T_stag_c2$='   '
        Endif
        if (Ncov = 1) then
            T_stag_p$=CONCAT$(String$(T_stag_p_out), ' [F]')
            T_stag_c1$=CONCAT$(String$(T_stag_c1_out), ' [F]')
            T_stag_c2$='   '
        Endif
        if (Ncov = 2) then
            T_stag_p$=CONCAT$(String$(T_stag_p_out), ' [F]')
            T_stag_c1$=CONCAT$(String$(T_stag_c1_out), ' [F]')
            T_stag_c2$=CONCAT$(String$(T_stag_c2_out), ' [F]')
        Endif
    Endif
End

"
Procedure to plot efficiency curves in both English and SI units"
Procedure Efficiency_curve_(Unit_Sys$, Eff_temp_diff$, T_in[0..10], T_av[0..10], T_amb,
G_T : DELTAT\G_T[0..10], DELTAT2\G_T[0..10])
    If (Unit_Sys$ = 'SI Units') then
        if (Eff_temp_diff$ = 'T_i - T_a') then
            Duplicate jj = 0, 10
                DELTAT\G_T[jj] = (T_in[jj] - T_amb)/G_T
                DELTAT2\G_T[jj] = (T_in[jj]-T_amb)^2/G_T
            End
        else
            Duplicate jj = 0, 10
                DELTAT\G_T[jj] = (T_av[jj] - T_amb)/G_T
                DELTAT2\G_T[jj] = (T_av[jj]-T_amb)^2/G_T
            End
        Endif
    Endif

    If (Unit_Sys$ = 'English Units') then
        if (Eff_temp_diff$ = 'T_i - T_a') then
            Duplicate jj = 0, 10
                DELTAT\G_T[jj] = (T_in[jj] - T_amb)/G_T*Convert(C,F)/Convert(W/m^2,
Btu/hr-ft^2)
                DELTAT2\G_T[jj] = (T_in[jj]-T_amb)^2/G_T*Convert(C,F)^2/
Convert(W/m^2, Btu/hr-ft^2)
```

```
            End
        else
            Duplicate jj = 0, 10
                DELTAT\G_T[jj] = (T_av[jj] - T_amb)/G_T*Convert(C,F)/Convert(W/m^2,
Btu/hr-ft^2)
                DELTAT2\G_T[jj]=(T_av[jj]-T_amb)^2/G_T*Convert(C,F)^2/
Convert(W/m^2, Btu/hr-ft^2)
            End
        Endif
    Endif
End


"! Comparison with the experimental equations ========================"

"
Procedure to draw a graph of the experimental test results"
Procedure comp_exp_(Unit_Sys$, Comp_exp$, a_0_exp, a_1_exp, a_2_exp, b_0_exp,
DELTAT\G_T_max:
x_i_exp_SI[0..10],x_i_exp_English[0..10],eta_i_exp[0..10],x_j_exp[0..10], K_ta_exp[0..10])
    If (Comp_exp$ = 'Yes') then
        Stepsize = DELTAT\G_T_max/10
        duplicate i=0,10
            if (Unit_Sys$ = 'SI Units') then
                x_i_exp_SI[i] = stepsize*i
                x_i_exp_English[i] = x_i_exp_SI[i]*convert(m^2-K/W,F-hr-ft^2/Btu)
                eta_i_exp[i] = a_0_exp+a_1_exp*x_i_exp_SI[i]+a_2_exp*x_i_exp_SI[i]^2
            Endif
            if (Unit_Sys$ = 'English Units') then
                x_i_exp_English[i] = stepsize*i
                x_i_exp_SI[i] = x_i_exp_English[i]*convert(F-hr-ft^2/Btu, m^2-K/W)
                eta_i_exp[i]                                                    =
a_0_exp+a_1_exp*x_i_exp_English[i]+a_2_exp*x_i_exp_English[i]^2
            Endif
            x_j_exp[i] = 0.1*i
            K_ta_exp[i] = 1+b_0_exp*x_j_exp[i]
        End
    Endif
    If (Comp_exp$ = 'No') then
        duplicate i=0,10
            x_i_exp_SI[i] = 0; x_i_exp_English[i]=0;eta_i_exp[i]=0;x_j_exp[i]=0;
K_ta_exp[i] = 0
        End
    Endif
End

"! Data Handling ================================================="

"
Subroutines to set the value of a variable used as an index"
Function Set_i_backup_(i_backup_1$)
    if (i_backup_1$ = '[Erase All Data]') then Set_i_backup_ = -1
    if (i_backup_1$ = '[Do Not Save]') then Set_i_backup_ = 0
```

```
    if (i_backup_1$ = 'Set 1') then Set_i_backup_ = 1
    if (i_backup_1$ = 'Set 2') then Set_i_backup_ = 2
    if (i_backup_1$ = 'Set 3') then Set_i_backup_ = 3
    if (i_backup_1$ = 'Set 4') then Set_i_backup_ = 4
    if (i_backup_1$ = 'Set 5') then Set_i_backup_ = 5
end

"Subroutine to store results in lookup table"
Function backup_(i_backup, A_c, A_p, TubeSp, DELTAT\G_T[0..10], eta_i_fit3[0..10],
index1[0..10], k_ta_fit[0..10], a_3_0,a_3_1, a_3_2,b_0, T_stag_p, T_stag_c1, T_stag_c2)
$COMMON
Unit_Sys,G_T,I_d\I_T,IncAng,Slope,T_amb,WindSp,Rel_hum,Eff_temp_diff$,Length,Width,
thickness,Length_p,Width_p
$COMMON
Ncov,Coverkind2$,n_c2,tau_s_c2,epsilon_IR_c2,tau_IR_c2,Coverkind1$,n_c1,tau_s_c1,ep
silon_IR_c1,tau_IR_c1,d_cp,d_c1c2,Thick_pl,alpha_n,epsilon_IR_p,Cond_pl
$COMMON
t_b,t_e,k_b,k_e,N_Tube,TubeDiam_i,TubeDiam_o,Fluid$,SolPer,Q_dot_inp,P_in,BondCon
d, a_0_exp, a_1_exp, a_2_exp, b_0_exp
    if (i_backup = -1) then
        duplicate ii=2,11
            duplicate jj=1,97
                lookup(jj,ii)=0
            End
        End
        duplicate ii=13,22
            duplicate jj=2,24
                lookup(jj,ii)=0
            End
        End
        backup_ = -1
    else
    if (i_backup = 0) then
        backup_ =0
    else
        i_backup_1 = i_backup*2
        i_backup_2 = i_backup*2+1

        lookup(1,i_backup_2) = Unit_Sys

        Duplicate jj = 2, 12
            jjm = jj-2
            lookup(jj,i_backup_1) = DELTAT\G_T[jjm]
            lookup(jj,i_backup_2) = eta_i_fit3[jjm]
        End
        Duplicate jj=13,23
            jjm = jj-13
            lookup(jj,i_backup_1) = index1[jjm]
            lookup(jj,i_backup_2) = k_ta_fit[jjm]
        End

        lookup(25,i_backup_2) = a_3_0
        lookup(26,i_backup_2) = a_3_1
```

lookup(27,i_backup_2) = a_3_2

lookup(29,i_backup_2) = b_0

lookup(31,i_backup_2) = a_0_exp
lookup(32,i_backup_2) = a_1_exp
lookup(33,i_backup_2) = a_2_exp
lookup(34,i_backup_2) = b_0_exp

lookup(37,i_backup_2) = T_stag_p
lookup(38,i_backup_2) = T_stag_c1
lookup(39,i_backup_2) = T_stag_c2

lookup(43,i_backup_2) = G_T
lookup(44,i_backup_2) = I_d\I_T
lookup(45,i_backup_2) = IncAng
lookup(46,i_backup_2) = Slope
lookup(47,i_backup_2) = T_amb
lookup(48,i_backup_2) = WindSp
lookup(49,i_backup_2) = Rel_hum
{lookup(50,i_backup_2) = Reserved row}
lookup(51,i_backup_2) = Eff_temp_diff$

lookup(53,i_backup_2) = Length
lookup(54,i_backup_2) = Width
lookup(55,i_backup_2) = thickness
lookup(56,i_backup_2) = A_c

lookup(58,i_backup_2) = Length_p
lookup(59,i_backup_2) = Width_p
lookup(60,i_backup_2) = A_p

lookup(62,i_backup_2) = Ncov
lookup(63,i_backup_2) = Coverkind2$
lookup(64,i_backup_2) = n_c2
lookup(65,i_backup_2) = tau_s_c2
lookup(66,i_backup_2) = epsilon_IR_c2
lookup(67,i_backup_2) = tau_IR_c2
lookup(68,i_backup_2) = Coverkind1$
lookup(69,i_backup_2) = n_c1
lookup(70,i_backup_2) = tau_s_c2
lookup(71,i_backup_2) = epsilon_IR_c1
lookup(72,i_backup_2) = tau_IR_c1

lookup(74,i_backup_2) = d_cp
lookup(75,i_backup_2) = d_c1c2

lookup(77,i_backup_2) = alpha_n
lookup(78,i_backup_2) = epsilon_IR_p
lookup(79,i_backup_2) = Thick_pl
lookup(80,i_backup_2) = Cond_pl

lookup(82,i_backup_2) = t_b

```
            lookup(83,i_backup_2) = k_b

            lookup(85,i_backup_2) = t_e
            lookup(86,i_backup_2) = k_e

            lookup(87,i_backup_2) = N_Tube
            lookup(88,i_backup_2) = TubeDiam_i
            lookup(90,i_backup_2) = TubeDiam_o
            lookup(91,i_backup_2) = TubeSp

            lookup(93,i_backup_2) = Fluid$
            lookup(94,i_backup_2) = SolPer
            lookup(95,i_backup_2) = Q_dot_inp
            lookup(96,i_backup_2) = P_in
            lookup(97,i_backup_2) = BondCond

            "For Efficiency graph for both SI and English Units"
            i_backup_3 = i_backup*2+11
            i_backup_4 = i_backup*2+12

            if (Unit_Sys <0.5) then
                Duplicate jj = 2, 12
                    jjm = jj-2
                    lookup(jj,i_backup_3) = DELTAT\G_T[jjm]
                    lookup(jj,i_backup_4) = eta_i_fit3[jjm]
                End
                Duplicate jj = 14, 24
                    jjm = jj-14
                    lookup(jj,i_backup_3)= DELTAT\G_T[jjm]*convert(m^2-K/W,F-hr-ft^2/Btu)
                    lookup(jj,i_backup_4)=eta_i_fit3[jjm]
                End
            else
                Duplicate jj = 2, 12
                    jjm = jj-2
                    lookup(jj,i_backup_3)=DELTAT\G_T[jjm]*convert(F-hr-ft^2/Btu,m^2-K/W)
                    lookup(jj,i_backup_4)=eta_i_fit3[jjm]
                End
                Duplicate jj = 14, 24
                    jjm = jj-14
                    lookup(jj,i_backup_3)=DELTAT\G_T[jjm]
                    lookup(jj,i_backup_4)=eta_i_fit3[jjm]
                End
            Endif
            backup_ = 1
        Endif
        Endif
End


"!Modules
============================================================"

"
```

```
Calculation of product of Extincton coefficient and thickness of a cover"
Module KL_s_(n, tau:KL)
      REF =((n-1)/(n+1))^2  "For normal incidenct angle"
      TRANSabs = exp(-KL)
      tau = TRANSabs*(1-REF)^2/(1-(REF*TRANSabs)^2)
End


"!=============================================================="
"

"
"!Main Routine"

"
Unit Conversion"
Call Unit_Conv_11_(Unit_Sys$,G_T_inp,T_amb_inp,WindSp_inp: G_T,T_amb,WindSp)
Call Unit_Conv_12_(Unit_Sys$, Length_inp, Width_inp, Thickness_inp, Length_p_inp,
Width_p_inp: Length, Width, Thickness, Length_p, Width_p)
Call Unit_Conv_13_(Unit_Sys$, d_cp_inp, d_c1c2_inp, cond_pl_inp, thick_pl_inp :  d_cp,
d_c1c2, cond_pl, thick_pl)
Call Unit_Conv_14_(Unit_Sys$, t_e_inp, t_b_inp, k_e_inp, k_b_inp:  t_e, t_b, k_e, k_b)
Call Unit_Conv_15_(Unit_Sys$, TubeDiam_i_inp,  TubeDiam_o_inp, Q_dot_inp, P_in_inp,
BondCond_inp :  TubeDiam_i, TubeDiam_o, Q_dot, P_in, BondCond)

"
Set the value of a variable related to save option"
i_backup =  Set_i_backup_(i_backup_1$)

"
Check the collector dimension"
dumdim_1 = check_1_(Width,Length,Thickness, Width_p, Length_p, d_cp, d_c1c2,
thick_pl, TubeDiam_i, TubeDiam_o, t_b)
dumchk_1 = check_2_(epsilon_IR_c1, tau_IR_c1, epsilon_IR_c2, tau_IR_c2)

"
Evaluation of the optical characteristics of cover and absorber plate"
Call KL_s_(n_c1, tau_s_c1:KL_c1)
Call KL_s_(n_c2, tau_s_c2:KL_c2)

Call OpticalProp3_(tau_IR_c1, epsilon_IR_c1: WLDP1, TRANS_IR_c1, Emitt_IR_c1)
Call OpticalProp3_(tau_IR_c2, epsilon_IR_c2: WLDP2, TRANS_IR_c2, Emitt_IR_c2)

xfraccov = epsilon_IR_c1

Emitt_IR_p = epsilon_IR_p

"
a] Calculation of Instantaneous Efficiency"

"Areas"
A_c = Width*Length
A_p = Width_p*Length_p
```

```
A_edge = thickness*2*(Length+Width)

"Tube Spacing"
N_Tube = Width_p/TubeSp

"Sky Temperature"
T_dp = DEWPOINT(AirH2O,T=T_amb,P=101.3,R = Rel_Hum/100)
T_s` = SkyTemp_JK_(T_amb,T_dp,0)    "Assumption : Hour Angle is zero."

"Absorbed radiation"
I_dT = G_T*I_d\I_T/100                  "Diffuse Radiation"
I_d = I_dT/(1+cos(Slope))/2
rho_g = 0.4                             "Ground Reflectance is assumed to be 0.4"
I_gT = (I_bT+I_d)*rho_g*(1-cos(Slope))/2      "Ground-reflected radiation"
G_T = I_bT`+I_dT+I_gT
I_bT = max(0,I_bT`)                     "Beam Radiation"

taualpha_b = TauAlpha_(Ncov,IncAng, KL_c1, KL_c2, n_c1, n_c2, Alpha_n)
IncAng_d = 59.7-0.1388*slope+0.001497*slope^2      "Equivalent angle of diffuse radiation"
taualpha_d = TauAlpha_(Ncov,IncAng_d, KL_c1, KL_c2, n_c1, n_c2, Alpha_n)
IncAng_g = 90-0.5788*slope+0.002693*slope^2        "Equivalent angle of diffuse radiation"
taualpha_g = TauAlpha_(Ncov,IncAng_g, KL_c1, KL_c2, n_c1, n_c2, Alpha_n)

S = I_bT*taualpha_b+I_dT*taualpha_d+I_gT*taualpha_g            "Isotropic sky model"

"Loss Coefficients"
WindCoeff =WindCoeff`FS_(WindSp,Length,Width,T_amb,0.7)
U_b = k_b/t_b*(A_c/A_p)
U_e = (k_e/t_e)*(A_edge/A_p)

"Loop"
jj_max1 = 10        "Number of iteration"

Call T_bound_(T_amb, G_T, Fluid$, P_in : T_in_max, T_in_min) "Maximum&Minimum Inlet
Temperature"

Duplicate jj = 0, jj_max1
      T_in[jj] = (T_amb-5)+jj*(T_in_max-T_in_min)/jj_max1
      F``[jj] = F_R[jj]/F`[jj]
      T_fl[jj] = T_Fl`M_(T_in[jj],Q_u[jj],A_p,F_R[jj],F``[jj],U_L[jj])
      Call Circular_tube_(Q_dot, TubeDiam_i, Length_p, Fluid$, T_fl[jj], P_in, SolPer,
N_Tube : m_dot[jj], C_p[jj], h_fi[jj], DELTAP[jj], Re[jj], DELTAP_1[jj], DELTAP_2[jj])
      Call U_L_(T_pl[jj],T_amb,T_s`,Slope,WindCoeff,Emitt_IR_p,Emitt_IR_c1,Emitt_IR_c2,
Ncov,U_b,U_e,d_cp,d_c1c2, WLDP1, WLDP2, TRANS_IR_c1, TRANS_IR_c2, xfraccov :
U_L[jj],T_c1[jj], T_c2[jj])
      T_pl[jj] = T_Pl`M_(T_in[jj],Q_u[jj],A_p,F_R[jj],U_L[jj])
      F`[jj]          =          CollEffFact_(U_L[jj],Cond_pl,Thick_pl,TubeSp,TubeDiam_i,
TubeDiam_o,BondCond,h_fi[jj])
      F_R[jj] = F_R_(m_dot[jj],C_p[jj],A_p,U_L[jj],F`[jj])
      "Useful gain from the Collector"
      {Q_u[jj] = max(F_R[jj]*(S*A_p-U_L[jj]*A_p*(T_in[jj]-T_amb)),0) }
      Q_u[jj] = max((S*A_p-U_L[jj]*A_p*(T_pl[jj]-T_amb)),0)
      "Exit Fluid Temperature"
```

```
      Q_u[jj] = m_dot[jj]*C_p[jj]*(T_out[jj] - T_in[jj])
      "Average Temperature for European practice"
      T_av[jj] = (T_in[jj]+T_out[jj])/2

      "Calculate Efficiency of Solar Collector"
      eta_i[jj] = Q_u[jj]/(A_p*G_T)

      "Linear Regression curve fitting"
      eta_i_fit1[jj] = a_1_0 + a_1_1*(DELTAT\G_T[jj])
      eta_i_fit2[jj] = a_2_0 + a_2_1*(DELTAT\G_T[jj]) + a_2_2*DELTAT\G_T[jj]^2
      eta_i_fit3[jj] = a_3_0 + a_3_1*(DELTAT\G_T[jj]) + a_3_2*DELTAT2\G_T[jj]
End

"
b] Calculation of incidence angle modifier, b_0"

"transmittance-absorptance product at normal incidence"
taualpha_n = TauAlpha_(Ncov,0, KL_c1, KL_c2, n_c1, n_c2, Alpha_n)

jj_max2 = 10        "Number of anlges to be calculated"
Maxtheta = 60       "Maximum angle of incidence"

DELTAtheta = Maxtheta/(JJ_max2)
Duplicate jj = 0, jj_max2
      IncAng[jj] = DELTAtheta*jj
      ta[jj] = TauAlpha_(Ncov,IncAng[jj], KL_c1, KL_c2, n_c1, n_c2, Alpha_n)
      K_ta[jj] = ta[jj]/taualpha_n
      index1[jj] = 1/cos(incAng[jj])-1
      K_ta_fit[jj] = 1+ b_0*index1[jj]
End

"
Curve Fitting by Linear Regression"
bias=sum(eta_i_fit1[jj]-eta_i[jj],jj=0,jj_max1)/(jj_max1+1)+sum(eta_i_fit2[jj]-eta_i[jj],jj=0,
jj_max1)/(jj_max1+1)+sum(eta_i_fit3[jj]-eta_i[jj],jj=0,jj_max1)/(jj_max1+1)+sum(K_ta_fit[jj]-
K_ta[jj],jj=0,jj_max2)/(jj_max2+1)    {bias error}
sigma=sum((eta_i_fit1[jj]-eta_i[jj])^2,jj=0,jj_max1)+sum((eta_i_fit2[jj]-eta_i[jj])^2,jj=0,
jj_max1)+sum((eta_i_fit3[jj]-eta_i[jj])^2,jj=0,jj_max1)+sum((K_ta_fit[jj]-K_ta[jj])^2,jj=0,
jj_max2)       {sum of square errors}

"Regression Formulae to find a_0, a_1, a_2, b_0"
0 = sum(eta_i_fit1[jj]-eta_i[jj],jj=0,jj_max1)
0 = sum((eta_i_fit1[jj]-eta_i[jj])*(DELTAT\G_T[jj]),jj=0,jj_max1)

0 = sum(eta_i_fit2[jj]-eta_i[jj],jj=0,jj_max1)
0 = sum((eta_i_fit2[jj]-eta_i[jj])*(DELTAT\G_T[jj]),jj=0,jj_max1)
0 = sum((eta_i_fit2[jj]-eta_i[jj])*(DELTAT\G_T[jj])^2,jj=0,jj_max1)

0 = sum(eta_i_fit3[jj]-eta_i[jj],jj=0,jj_max1)
0 = sum((eta_i_fit3[jj]-eta_i[jj])*DELTAT\G_T[jj],jj=0,jj_max1)
0 = sum((eta_i_fit3[jj]-eta_i[jj])*DELTAT2\G_T[jj],jj=0,jj_max1)

0 = sum((K_ta_fit[jj]-K_ta[jj])*index1[jj],jj=0,jj_max2)
```

```
"
c] Stagnation Temperature"

S = U_L*(T_stag_p-T_amb)
Call U_L_(T_stag_p,T_amb,T_s`, Slope,WindCoeff,Emitt_IR_p,Emitt_IR_c1,Emitt_IR_c2,
Ncov,U_b,U_e,d_cp,d_c1c2, WLDP1, WLDP2, TRANS_IR_c1, TRANS_IR_c2, xfraccov :
U_L,T_stag_c1, T_stag_c2 )
Call Stag_disp(Unit_sys$,Ncov,T_stag_p,T_stag_c1,T_stag_c2:T_stag_p$,T_stag_c1$,
T_stag_c2$)

"
d] Comparison with the experimental equation"

call
comp_exp_(Unit_Sys$,Comp_exp$,a_0_exp,a_1_exp,a_2_exp,b_0_exp,DELTAT\G_T[10]:
x_i_exp_SI[0..10], x_i_exp_English[0..10],eta_i_exp[0..10],x_j_exp[0..10], K_ta_exp[0..10])

"
e] Post-Processing"

"Efficiency Curves : for SI and English units"
Call Efficiency_curve_(Unit_Sys$, Eff_temp_diff$, T_in[0..10], T_av[0..10], T_amb, G_T :
DELTAT\G_T[0..10], DELTAT2\G_T[0..10])
Call Efficiency_curve_('SI Units', Eff_temp_diff$, T_in[0..10], T_av[0..10], T_amb, G_T :
DELTAT\G_T_SI[0..10], DELTAT2\G_T_SI[0..10])
Call Efficiency_curve_('English Units', Eff_temp_diff$, T_in[0..10], T_av[0..10], T_amb, G_T
: DELTAT\G_T_English[0..10], DELTAT2\G_T_English[0..10])

"Put the results in Lookup table"
dum1= backup_(i_backup, A_c, A_p, TubeSp, DELTAT\G_T[0..10], eta_i_fit3[0..10],
index1[0..10], k_ta_fit[0..10], a_3_0,a_3_1, a_3_2,b_0,T_stag_p, T_stag_c1, T_stag_c2)

"Unit Conversion for Output"
Call Unit_Conv_22_(Unit_Sys$, A_p, A_c : A_p_out, A_c_out)
Call Unit_Conv_25_(Unit_Sys$, TubeSp : TubeSp_out)
```

APPENDIX B

HEAT EXCHANGER DESIGN PROGRAM FOR SDHW SYSTEM

B.1 Introduction

There are three types of heat exchangers frequently used in solar domestic heating water systems; shell-and-tube, immersed internal, and natural convection heat exchangers. In the configuration of SDHW systems, the addition of a heat exchanger reduces the overall performance of the system. However, using anti-freeze in collector loop in cold climates is an advantage. To calculate the characteristics of heat exchanger, design programs for shell-and-tube and immersed internal heat exchanger are developed so that details of heat exchanger configuration can be specified in an easy-to-use graphic interface. The calculation results from the programs can be used as input parameters for simulation programs like TRNSYS [7].

B.2 Shell-And-Tube Type Heat Exchanger

Figure B.1 shows a schematic diagram of a shell-and-tube heat exchanger with straight tubes. Water or an anti-freeze is pumped though the collector and heat exchanger shell while water circulates through the storage tank and heat exchanger tubes by mechanical pump.

With the assumption of no phase change and constant specific heats of the working fluids, the energy balance equations are given by

$$Q = C_s\left(T_{s,i} - T_{s,o}\right) \tag{B.2.1}$$

$$Q = C_t\left(T_{t,o} - T_{t,i}\right) \tag{B.2.2}$$

where C is heat capacity rate, the product of mass flow rate $\dot{m}$ and specific heat c_p. Subscripts s and t represent shell and tube side fluid while i and o means inlet and outlet temperatures, respectively. By applying log-mean-temperature-difference method the total heat transfer rate Q can be calculated by

$$Q = (UA)\cdot \Delta T_{lm} \tag{B.2.3}$$

where (UA) is the overall heat transfer coefficient-area product and DT_{lm} is log-mean-temperature-difference. Log-mean-temperature-difference is defined as

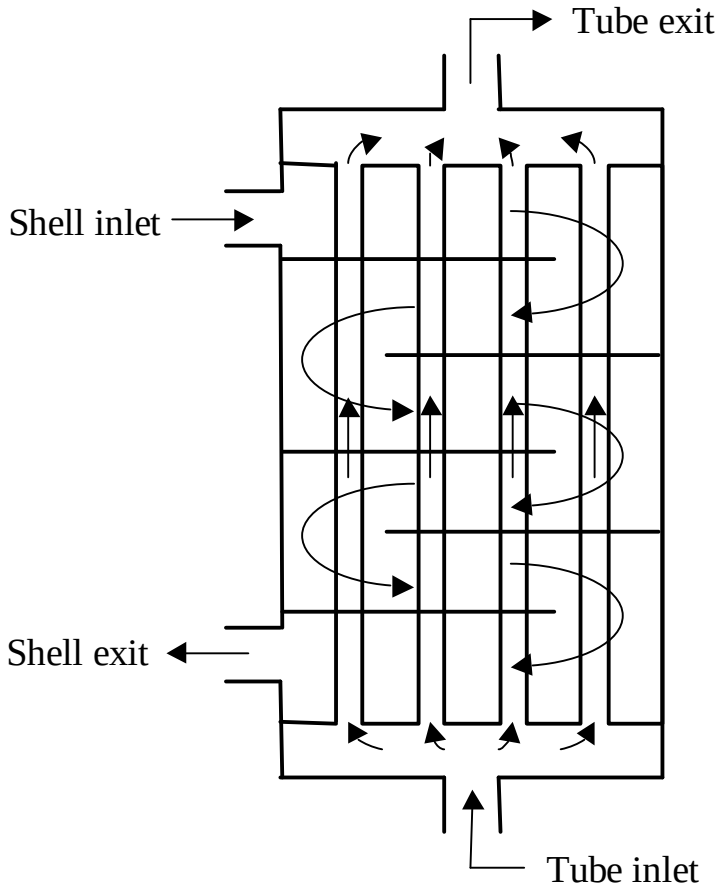

Figure B.2.1 Shell-and-tube type heat exchanger

$$\Delta T_{lm} = \frac{\Delta T_2 - \Delta T_1}{\ln\left(\Delta T_2 / \Delta T_1\right)} \qquad \text{(B.2.4)}$$

where for parallel-flow heat exchanger the endpoint temperature differences are defined as

$$\Delta T_1 = T_{s,i} - T_{t,i}, \quad \Delta T_2 = T_{s,o} - T_{t,o} \tag{B.2.5}$$

and for counter-flow exchanger

$$\Delta T_1 = T_{s,i} - T_{t,o}, \quad \Delta T_2 = T_{s,o} - T_{t,i} \tag{B.2.6}$$

The thermal performance of heat exchanger is determined by solving Equations B.2.1 – B.2.4 simultaneously. With the assumption of negligible fouling factors, the overall heat transfer coefficient-area product is defined as

$$(UA) = \frac{n_t}{\frac{1}{h_i A_i} + \frac{\ln(D_o/D_i)}{2\pi kL} + \frac{1}{h_o A_o}} \tag{B.2.7}$$

where subscripts *i* and *o* represent inside and outside of the tube, respectively. n_t is the number of tubes, *h* is the convection heat transfer coefficient, and *k* is the thermal conductivity of tube material. *L*, *D* and *A* are the length, diameter, and heat transfer area of each tube, respectively. The heat transfer coefficient inside of tubes can be determined by Section 1.1.5. It is assumed that the shell-and-tube heat exchanger does not have bafflings in shell side. For this case Liu et al. [9] suggested that the heat transfer coefficients of shell side be calculated using the relation of Su and El-Genk [18]

:

$$Nu = (2.34 - 3.09e)\mathrm{Re}^{(0.89e-0.044)}\,\mathrm{Pr}^{0.33} \quad \text{(laminar flow)} \qquad \text{(B.2.8)}$$

$$Nu = 5.5\times10^{-3}\left(\frac{4.47}{\sqrt{1-e}}\right)\mathrm{Re}^{0.8}\,\mathrm{Pr}^{0.33} \quad \text{(turbulent flow)} \qquad \text{(B.2.9)}$$

where e is the tube-bundle porosity defined as the ratio of the actual shell side flow area to the shell inside cross-sectional area. The transition Reynolds number from laminar to turbulent flow is defined as

$$\mathrm{Re}_T = 1.23\times10^4\left(\frac{0.91}{\sqrt{1-e}} - 1\right). \qquad \text{(B.2.10)}$$

The thermal performance of the shell-and-tube type heat exchanger can be represented as the effectiveness e that is defined as the ratio of the actual heat transfer rate to the maximum possible heat transfer rate for a heat exchanger [5]:

$$e \equiv \frac{q_{act}}{q_{\max}} = \frac{C_s\left(T_{s,i} - T_{s,o}\right)}{C_{\min}\left(T_{s,o} - T_{t,i}\right)} = \frac{C_t\left(T_{t,o} - T_{t,i}\right)}{C_{\min}\left(T_{s,o} - T_{t,i}\right)} \qquad \text{(B.2.11)}$$

where C_{min} is minimum heat capacity rate of the heat exchanger.

B.3 Immersed Internal Heat Exchanger

Figure B.1 illustrates an immersed internal shell-and-tube heat exchanger used in unpressurized thermal storage tanks. The bundle of tubes is placed inside of storage tank where the local tank temperature is T_{tank}. It is a bundle of horizon tubes with or without fins outside of tubes. Mechanical pump circulates either water or an anti-freeze through the collector and inside of tubes while the heat transfer outside of tube occurs by natural convection in a water storage tank.

The thermal performance of the heat exchanger can be determined by the solution of the energy balance equations given by

$$Q = C_t\left(T_{t,i} - T_{t,o}\right) \qquad \text{(B.3.1)}$$

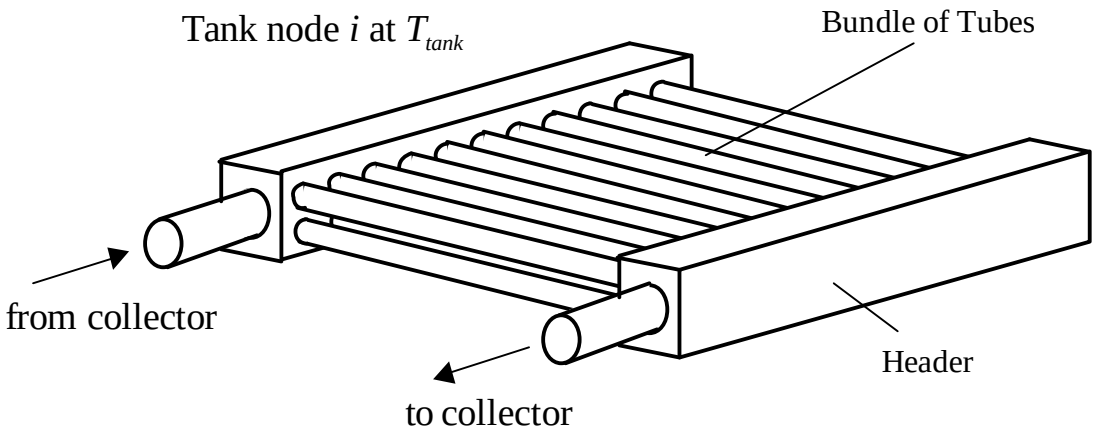

Figure B.3.1 Immersed internal heat exchanger inside of tank

$$Q = (UA) \cdot \Delta T_{lm} \tag{B.3.2}$$

where it is assumed that there is no phase change and the specific heats of the working fluids are constant. The log-mean-temperature-difference of an immersed internal heat exchanger is defined as

$$\Delta T_{lm} = \frac{T_{t,i} - T_{t,o}}{\ln\left(\frac{T_{t,i} - T_{\tan k}}{T_{t,o} - T_{\tan k}}\right)} \tag{B.3.3}$$

where subscript *tank* indicates the node of storage tank where the tube bundle is immersed.

The overall heat transfer coefficient-area product is calculated by

$$(UA) = \frac{n_t}{\frac{1}{h_i A_i} + \frac{\ln(D_o/D_i)}{2pkL} + \frac{1}{h_o\left(A_o + h_{fin} A_{fin}\right)}} \tag{B.3.4}$$

where subscripts *fin* represents the property of fins attached to outside of tube to enhance natural convection heat transfer. The natural convection heat transfer coefficient for the outside of the heat exchanger is calculated by

$$Nu = 0.513Ra^{0.25} \tag{B.3.5}$$

where

$$Ra = \frac{r^2 b c_p g D_o^3}{mk}\left(\Delta T_{lm,corr}\right). \tag{B.3.6}$$

The coefficients in Equation B.3.5 are determined to provide a good generalized correlation for immersed bodies [10]. The corrected log-mean-temperature-difference is defined as [10]

$$\Delta T_{lm,corr} = \Delta T_{lm} - \frac{Q}{n_t}\left[\frac{1}{A_i h_i} + \frac{\ln\left(D_0/D_i\right)}{2pkL}\right] \tag{B.3.7}$$

where n_t is number of tubes.

B.3.1 Fin Efficiency

The fins attached to the tubes are circular with a rectangular cross section as shown in Figure B.3.2. In this case, the fin efficiency is calculated using [5]

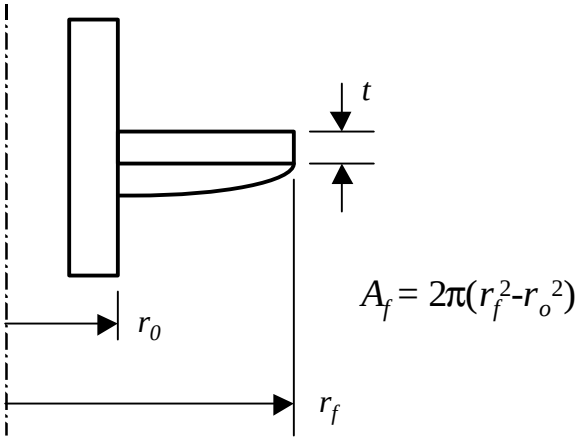

Figure B.3.2 Circular fin for Immersed tubes

$$h_f = \frac{(r_o/m)}{(r_c^2 - r_o^2)} \frac{K_1(mr_o)I_1(mr_c) - I_1(mr_o)K_1(mr_c)}{I_0(mr_o)K_1(mr_c) - K_0(mr_o)I_1(mr_c)} \tag{B.3.8}$$

where

$$m = \sqrt{\frac{2h}{kt}} \tag{B.3.9}$$

$$r_o = \frac{D_o}{2} \tag{B.3.10}$$

$$r_c = r_f + \frac{t}{2} \tag{B.3.11}$$

and K_0, K_1, I_0 and I_1 are zero-order and first-order modified Bessel functions and k is thermal conductivity of the fin.

B.4 Heat Exchanger Design Program

Based on the theory in Appendix B, a heat exchanger design program for solar energy systems has been developed. In the calculation, the thermal conductivity of the tube and fin material is assumed to be constant and thermal properties of working fluids are evaluated at a mean fluid temperature. For tube material, properties of high temperature nylon and cross-linked polyethylene are built-in along with the conventional materials. The working fluid can be chosen among water, propylene glycol solution, and ethylene-glycol solution. The design program calculates the thermal performance of the heat exchanger: effectiveness, actual heat transfer rate, exit temperatures, and pressure drop of the tube side. Figure B.4.1 – B.4.6 show the main and child windows of the heat exchanger program.

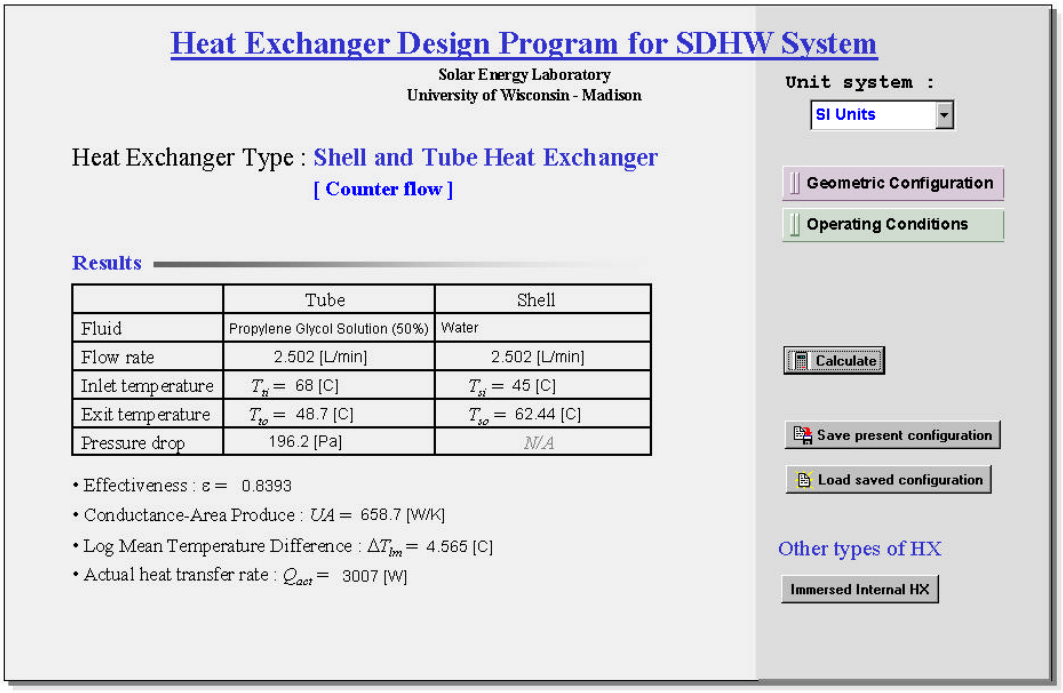

Figure B.4.1 Main diagram window of Shell-and tube heat exchanger design program

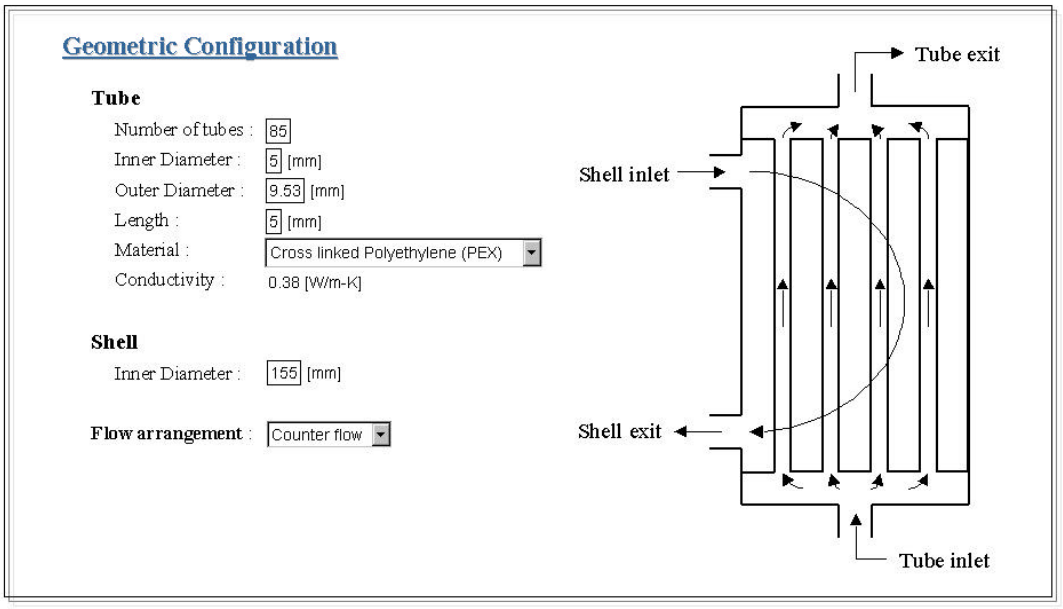

Figure B.4.2 Geometric configuration window for Shell-and tube heat exchanger

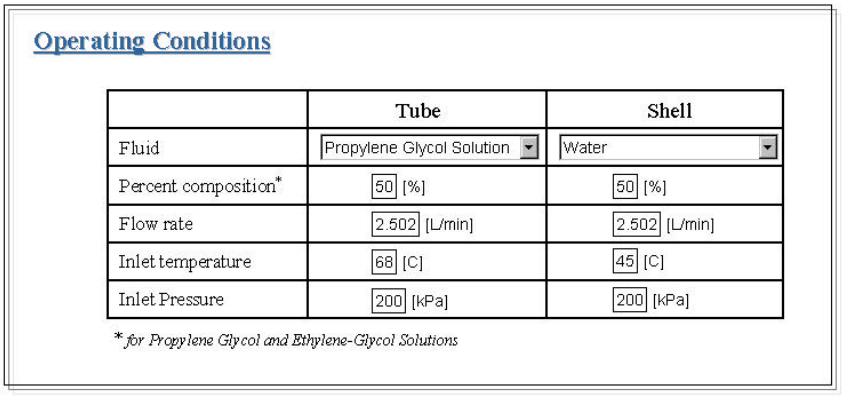

Figure B.4.3 Operating condition window for Shell-and tube heat exchanger

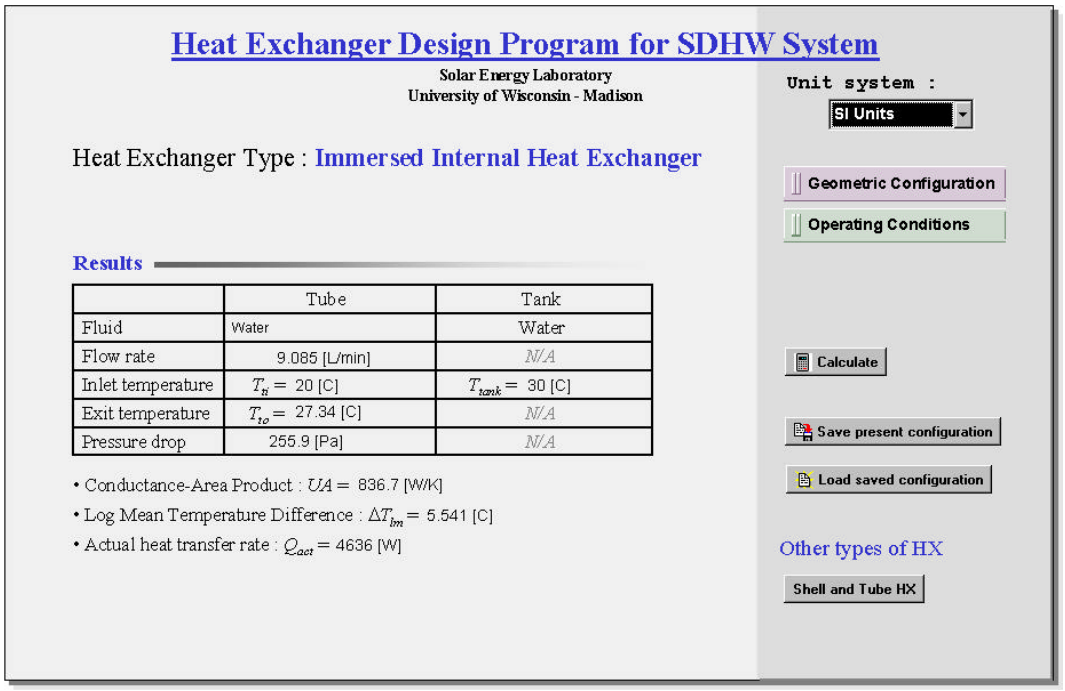

Figure B.4.4 Main diagram window of immersed internal heat exchanger design program

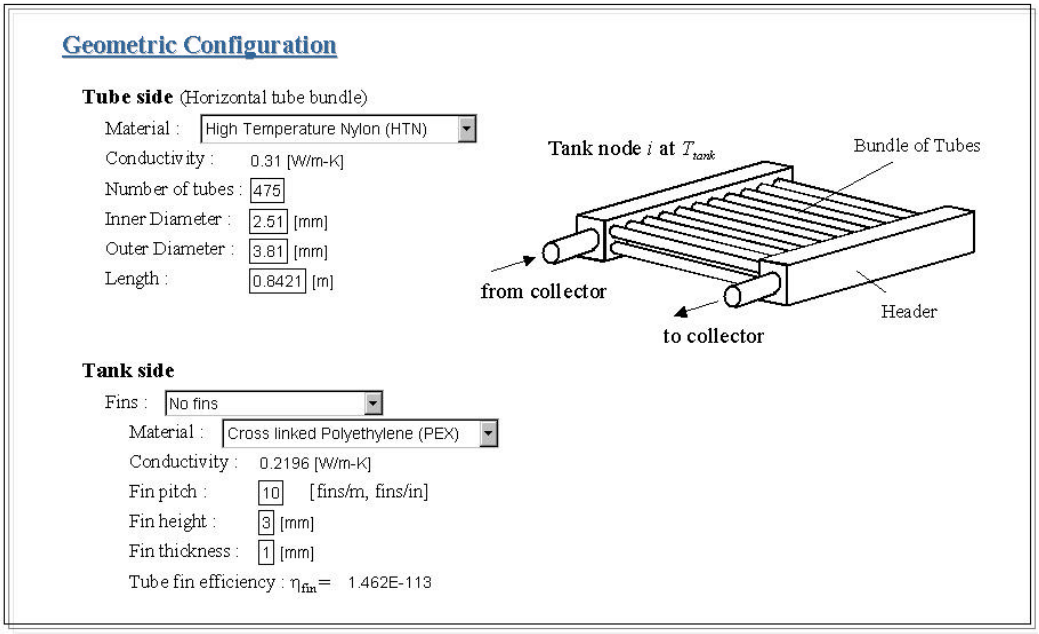

Figure B.4.5 Geometric configuration window for immersed heat exchanger

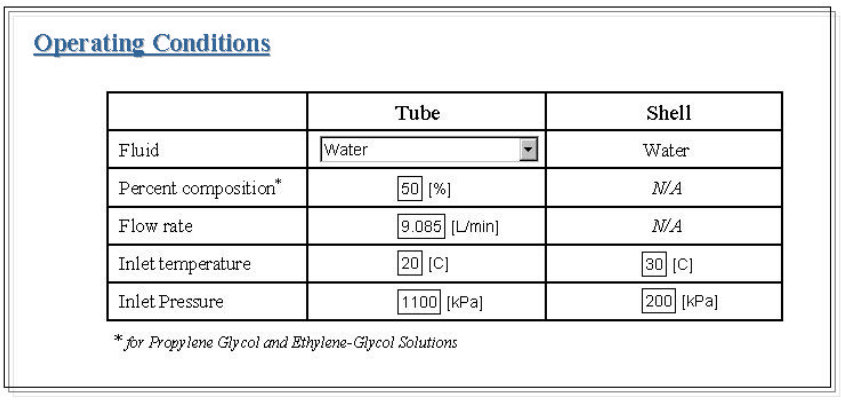

Figure B.4.6 Operating condition window for immersed heat exchanger

EES CODE OF HEAT EXCHANGER DESIGN PROGRAM

1. Sell –and-Tube Heat Exchanger

```
"Heat Exchanger Design Program for Solar Domestic Heating Water System"
"Shell-and-Tube Heat Exchanger"

"Jae-Mo Koo(koo@sel.me.wisc.edu)
William A. Beckman, Professor (beckman@me.engr.wisc.edu)
Solar Energy Laboratory
University of Wisconsin - Madison
SaThx050, 12/05/1999"

"Note : All the calculations are done with SI units."


$INCLUDE Therprop.LIB

"! Unit Conversion ====================================="

Procedure Unit_conv_11_(Unit_Sys$, D_t_inner_inp, D_t_outer_inp, D_s_inner_inp,
L_t_inp, k_tube_inp: D_t_inner, D_t_outer, D_s_inner, L_t, k_tube)
    If (Unit_Sys$ = 'SI Units') then
        D_t_inner = D_t_inner_inp*convert(mm,m)
        D_t_outer = D_t_outer_inp*convert(mm,m)
        D_s_inner = D_s_inner_inp*convert(mm,m)
        L_t = L_t_inp
        k_tube = k_tube_inp
    Endif
    If (Unit_Sys$ = 'English Units') then
        D_t_inner = D_t_inner_inp*convert(in,m)
        D_t_outer = D_t_outer_inp*convert(in,m)
        D_s_inner = D_s_inner_inp*convert(in,m)
        L_t = L_t_inp*convert(ft,m)
        k_tube = k_tube_inp*convert(Btu/hr-ft-F,W/m-K)
    Endif
End

Procedure Unit_conv_12_(Unit_Sys$, Q_dot_t_inp, Q_dot_s_inp, T_t_i_inp, T_s_i_inp,
P_t_i_inp, P_s_i_inp : Q_dot_t, Q_dot_s, T_t_i, T_s_i, P_t_i, P_s_i)
    If (Unit_Sys$ = 'SI Units') then
        Q_dot_t = Q_dot_t_inp*convert(L/min, m3/s)
        Q_dot_s = Q_dot_s_inp*convert(L/min, m3/s)      "[m3/s]"
        T_t_i = T_t_i_inp
        T_s_i = T_s_i_inp
        P_t_i = P_t_i_inp
```

```
        P_s_i = P_s_i_inp
    Endif
    If (Unit_Sys$ = 'English Units') then
        Q_dot_t = Q_dot_t_inp*convert(Gal/min, m3/s)
        Q_dot_s = Q_dot_s_inp*convert(Gal/min, m3/s)  "[m3/s]"
        T_t_i = ConvertTemp('F', 'C', T_t_i_inp)
        T_s_i = ConvertTemp('F', 'C', T_s_i_inp)
        P_t_i = P_t_i_inp*convert(psi,kPa)
        P_s_i = P_s_i_inp*convert(psi,kPa)
    Endif
End

Procedure Unit_conversion_21_(Unit_Sys$, Flow_arrange$, T_t_o, T_s_o, DELTAP_t, UA,
Q_tot, LMTD : Flow_arrange_out$,  T_t_o$, T_s_o$, DELTAP_t$, UA$, Q_act$,
DELTAT_lm$)
    "Flow arrangement"
    dumb_1$ = CONCAT$('[ ',Flow_arrange$)
    Flow_arrange_out$ =  CONCAT$(dumb_1$, ' ]')

    If (Unit_Sys$ = 'SI Units') then
        T_t_o$ = CONCAT$(STRING$(T_t_o), ' [C]')
        T_s_o$ = CONCAT$(STRING$(T_s_o), ' [C]')
        DELTAP_t$ = CONCAT$(STRING$(DELTAP_t), ' [Pa]')
        UA$ = CONCAT$(STRING$(UA), ' [W/K]')
        Q_act = abs(Q_tot)
        Q_act$ = CONCAT$(STRING$(Q_act), ' [W]')
        DELTAT_lm = abs(LMTD)
        DELTAT_lm$ = CONCAT$(STRING$(DELTAT_lm), ' [C]')
    Endif
    If (Unit_Sys$ = 'English Units') then
        T_t_o_out = ConvertTemp('C', 'F', T_t_o)
        T_t_o$ = CONCAT$(STRING$(T_t_o_out), ' [F]')
        T_s_o_out = ConvertTemp('C', 'F', T_s_o)
        T_s_o$ = CONCAT$(STRING$(T_s_o_out), ' [F]')
        DELTAP_t_out = DELTAP_t*convert(Pa, psi)
        DELTAP_t$ = CONCAT$(STRING$(DELTAP_t_out), ' [psi]')
        UA_out = UA*convert(W/K, Btu/h-F)
        UA$ = CONCAT$(STRING$(UA_out), ' [Btu/h-F]')
        Q_act = abs(Q_tot*convert(W, Btu/h))
        Q_act$ = CONCAT$(STRING$(Q_act), ' [Btu/h]')
        DELTAT_lm = abs(LMTD*convert(C, F))
        DELTAT_lm$ = CONCAT$(STRING$(DELTAT_lm), ' [F]')
    Endif
End

"! Preparation ====================================="

Procedure Check_range_1_(T_t_i, T_s_i, Num_tube, D_t_inner, D_t_outer, D_s_inner:
Check_range_1)
    If (D_t_inner > D_t_outer) then CALL ERROR('The Inner diameter of tubes (I.D.) is
greater than outer diameter of them (O.D.).',D_t_inner)
    A_s = pi*D_s_inner^2/4
    A_t = Num_tube*pi*D_t_inner^2/4
```

```
    if (A_t > A_s) then CALL ERROR('You cannot put tube bundle into your shell. Increase
Shell Diameter.',D_s_inner)
    if (T_t_i >100) then CALL ERROR('This is sensible heat exchanger. Tube inlet
temperature should be lower than boiling temperature.',T_t_i)
    if (T_s_i >100) then CALL ERROR('This is sensible heat exchanger. Shell inlet
temperature should be lower than boiling temperature.',T_s_i)
    if (T_t_i < 0) then CALL ERROR('This is sensible heat exchanger. Tube inlet
temperature should be higher than melting temperature.',T_t_i)
    if (T_s_i < 0) then CALL ERROR('This is sensible heat exchanger. Shell inlet
temperature should be higher than ice temperature.',T_s_i)
    if (T_s_i = T_t_i) then CALL ERROR('Since the inlet temperatures of Shell and tube
sides are the same, there is no heat transfer.',T_s_i)
    Check_range_1 = 0
End

"! For LMTD METHOD ====================================="

Procedure Hot_Cold_check_1_(T_t_i, T_s_i, C_t, C_s:Index_HC, T_h_i, T_c_i, C_h, C_c)
    if (T_t_i > T_s_i) then
        Index_HC = -1
        T_h_i = T_t_i
        T_c_i = T_s_i
        C_h = C_t
        C_c = C_s
    Endif
    if (T_t_i < T_s_i) then
        Index_HC = 1
        T_h_i = T_s_i
        T_c_i = T_t_i
        C_h = C_s
        C_c = C_t
    Endif
End

Procedure Tube_Shell_check_1_(Index_HC, T_h_o, T_c_o : T_t_o, T_s_o)
    if (Index_HC = -1) then
        T_t_o = T_h_o
        T_s_o = T_c_o
    Endif
    if (Index_HC = 1) then
        T_s_o = T_h_o
        T_t_o = T_c_o
    Endif
End

"! Calculation of Shell Side ============================================"

Procedure Shell_(m_dot, D_t, D_s, L, Fluid$, T, P, SolPer, Num_tube : h, Re)
    rho = rho_(Fluid$, T, P, SolPer)"[kg/m3]"
    mu = mu_(Fluid$, T, P, SolPer)"[kg/s-m]"
    C_p = C_p_(Fluid$, T, P, SolPer)*convert(kJ, J)  "[J/kg-K]"
    k = k_(Fluid$, T, P, SolPer)            "[W/m-K]"
    Pr = mu*C_p/k
```

```
      por = (D_s^2-Num_tube*D_t^2)/D_s^2
      A_ts = pi*D_s^2/4-Num_tube*pi*D_t^2/4
      P_ts = pi*D_s+Num_tube*pi*D_t
      D_ts = 4*A_ts/P_ts

      U = m_dot/rho/A_ts
      Re = (rho*U*D_ts)/mu
      Re_t = 1.23E4*(0.91/sqrt(1-por)-1)

      if (Re<Re_t) then
            A = 2.34-3.09*por
            B = 0.89*por-0.044
            Nusselt = A*Re^b*Pr^0.33
      Else
            C = 5.5E-3*(4.47/sqrt(1-por)-1)
            Nusselt = C*Re^0.8*Pr^0.33
      Endif

      h = Nusselt*k/D_ts            "[W/m^2-K]"
End

"! Calculation of LMTD ====================================="

Procedure LMTD_1_(Flow_arrange$, T_h_i, T_c_o, T_h_o,T_c_i: LMTD)
      if (Flow_arrange$ = 'Counter Flow') then
            If (T_h_o < T_c_i) then T_h_o = T_c_i
            If (T_c_o > T_h_i) then T_c_o = T_h_i
            DELTAT_1 = T_h_i - T_c_o
            DELTAT_2 = T_h_o - T_c_i
      Endif
      if (Flow_arrange$ = 'Parallel Flow') then
            If (T_h_o < T_c_o) then T_h_o = T_c_i
            DELTAT_1 = T_h_i - T_c_i
            DELTAT_2 = T_h_o - T_c_o
      Endif
      if (DELTAT_1 = 0) then DELTAT_1 = 1
      DELTAT_2_1 = DELTAT_2/DELTAT_1
      if (DELTAT_2_1 = 1) then DELTAT_2_1 = 2
      if (DELTAT_2_1 <= 0) then DELTAT_2_1 = 2
      LMTD=(DELTAT_2-DELTAT_1)/ln(DELTAT_2_1)
End


"! Main Program ============================================="

"
Pre-processing ============================================="

Call Unit_conv_11_(Unit_Sys$, D_t_inner_inp, D_t_outer_inp, D_s_inner_inp, L_t_inp,
k_tube_inp : D_t_inner, D_t_outer, D_s_inner, L_t, k_tube)
Call Unit_conv_12_(Unit_Sys$, Q_dot_t_inp, Q_dot_s_inp, T_t_i_inp, T_s_i_inp, P_t_i_inp,
P_s_i_inp : Q_dot_t, Q_dot_s, T_t_i, T_s_i, P_t_i, P_s_i)
```

```
Call Check_range_1_(T_t_i, T_s_i, Num_tube, D_t_inner, D_t_outer, D_s_inner:
Check_range_1)

"
Areas ================================================="
A_t_inner = L_t*pi*D_t_inner
A_t_outer = L_t*pi*D_t_outer

"
Calculation-Tube Side ================================================"
      T_t = T_t_i
      rho_t = rho_(Fluid_t$, T_t, P_t_i, SolPer_t)   "[kg/m3]"
      C_p_t = C_p_(Fluid_t$, T_t, P_t_i, SolPer_t)*convert(kJ, J)  "[J/kg-K]"

      m_dot_t = Q_dot_t*rho_t                                    "[kg/s]"
      C_t = m_dot_t*C_p_t
        Call Circular_tube_(m_dot_t, D_t_inner, L_t, Fluid_t$, T_t, P_t_i, SolPer_t, Num_tube
: h_t, DELTAP_t, Re_t, DELTAP_t_1, DELTAP_t_2)

      Power_t = DELTAP_t*Q_dot_t


"
Calculation-Shell Side ================================================"
      T_s =T_s_i
      rho_s = rho_(Fluid_s$, T_s, P_s_i, SolPer_s)"[kg/m3]"
      C_p_s = C_p_(Fluid_s$, T_s, P_s_i, SolPer_s)*convert(kJ, J)          "[J/kg-K]"
      m_dot_s = Q_dot_s*rho_s                      "[kg/s]"

      Call Shell_(m_dot_s, D_t_outer, D_s_inner, L_t, Fluid_s$, T_s, P_s_i, SolPer_s,
Num_tube : h_s, Re_s)
      C_s = m_dot_s*C_p_s

"
Switch from Tube-Shell to Hot-Cold ================================="
Call Hot_Cold_check_1_(T_t_i, T_s_i, C_t, C_s:Index_HC, T_h_i, T_c_i, C_h, C_c)

"
Overall Calculation ==========================================="
"Total heat transfer coefficient*Area"
      R_t_inner = 1/(h_t*A_t_inner)
      R_w = ln(D_t_outer/D_t_inner)/(2*pi*k_tube*L_t)
      R_t_outer = 1/(h_s*A_t_outer)

      UA = Num_tube/(R_t_inner+R_w+R_t_outer)

"Total Heat Transfer rate"
      Q_tot = C_h*(T_h_i-T_h_o)
      Q_tot = C_c*(T_c_o-T_c_i)
      Q_tot = UA*LMTD

"Log Mean Temperature Difference"
```

```
    Call LMTD_1_(Flow_arrange$, T_h_i, T_c_o, T_h_o,T_c_i: LMTD)

"Maximum heat transfer rate"
    C_min = Min(C_h, C_c)
    Q_max = C_min*(T_h_i-T_c_i)

"Effectiveness"
    epsilon = Q_tot/Q_max

    NTU = UA/C_min

"Switch from Hot-Cold to Tube-Shell"
Call Tube_Shell_check_1_(Index_HC, T_h_o, T_c_o : T_t_o, T_s_o)

"
Post-Processing for output ==================================="
Call Out_1_(Fluid_t$, SolPer_t: Fluid_t_out$)
Call Out_1_(Fluid_s$, SolPer_s: Fluid_s_out$)

Call Unit_conversion_21_(Unit_Sys$, Flow_arrange$, T_t_o, T_s_o, DELTAP_t, UA, Q_tot,
LMTD : Flow_arrange_out$,  T_t_o$, T_s_o$, DELTAP_t$, UA$, Q_act$, DELTAT_lm$)
```

2. Immersed Internal Heat Exchanger

```
"Heat Exchanger Design Program for Solar Domestic Heating Water System"
"Immersed Internal  Heat Exchanger"

"Jae-Mo Koo (koo@sel.me.wisc.edu)
William A. Beckman, Professor (beckman@me.engr.wisc.edu)
Solar Energy Laboratory
University of Wisconsin - Madison
ImmHX050, 12/05/1999"

"Note : All the calculations are done with SI units."


$INCLUDE Therprop.LIB

"! Unit Conversion ====================================="

Procedure Unit_conversion_21_(Unit_Sys$, k_tube_inp, D_t_inner_inp, D_t_outer_inp,
L_t_inp, k_fin_inp, H_fin_inp, t_fin_inp: k_tube, D_t_inner, D_t_outer, L_t, k_fin, H_fin,
t_fin)
      If (Unit_Sys$ = 'SI Units') then
            k_tube = k_tube_inp
            D_t_inner = D_t_inner_inp*convert(mm,m)
            D_t_outer = D_t_outer_inp*convert(mm,m)
            L_t = L_t_inp
            k_fin = k_fin_inp
            H_fin = H_fin_inp*convert(mm,m)
               t_fin = t_fin_inp*convert(mm,m)
      Endif
      If (Unit_Sys$ = 'English Units') then
            k_tube = k_tube_inp*convert(Btu/hr-ft-F,W/m-K)
            D_t_inner = D_t_inner_inp*convert(in,m)
            D_t_outer = D_t_outer_inp*convert(in,m)
            L_t = L_t_inp*convert(ft,m)
            k_fin = k_fin_inp**convert(Btu/hr-ft-F,W/m-K)
            H_fin = H_fin_inp*convert(in,m)
               t_fin = t_fin_inp*convert(in,m)
      Endif
End

Procedure Unit_conversion_22_(Unit_Sys$, Q_dot_t_inp, T_t_i_inp,  P_t_i_inp,
T_tank_inp, P_tank_inp : Q_dot_t, T_t_i, P_t_i, T_tank, P_tank)
      If (Unit_Sys$ = 'SI Units') then
            Q_dot_t = Q_dot_t_inp*convert(L/min, m3/s)
            T_t_i = T_t_i_inp
            T_tank = T_tank_inp
            P_t_i = P_t_i_inp
            P_tank = P_tank_inp
      Endif
```

```
    If (Unit_Sys$ = 'English Units') then
        Q_dot_t = Q_dot_t_inp*convert(Gal/min, m3/s)
        T_t_i = ConvertTemp('F', 'C', T_t_i_inp)
        T_tank = ConvertTemp('F', 'C', T_tank_inp)
        P_t_i = P_t_i_inp*convert(psi,kPa)
        P_tank = P_tank_inp*convert(psi,kPa)
    Endif
End

Procedure Unit_conversion_26_(Unit_Sys$, Q_dot_t_inp, T_t_i_inp, T_tank_inp, T_t_o,
DELTAP_t, UA, Q_tot, LMTD :  Q_dot_t$, T_t_i$, T_tank$, T_t_o$, DELTAP_t$, UA$,
Q_act$, DELTAT_lm$)

    If (Unit_Sys$ = 'SI Units') then
        Q_dot_t$ = CONCAT$(STRING$(Q_dot_t_inp), ' [L/min]')
        T_t_i$ = CONCAT$(STRING$(T_t_i_inp), ' [C]')
        T_tank$ = CONCAT$(STRING$(T_tank_inp), ' [C]')
        T_t_o$ = CONCAT$(STRING$(T_t_o), ' [C]')
        DELTAP_t$ = CONCAT$(STRING$(DELTAP_t), ' [Pa]')
        UA$ = CONCAT$(STRING$(UA), ' [W/K]')
        Q_act = abs(Q_tot)
        Q_act$ = CONCAT$(STRING$(Q_act), ' [W]')
        DELTAT_lm = abs(LMTD)
        DELTAT_lm$ = CONCAT$(STRING$(DELTAT_lm), ' [C]')
    Endif
    If (Unit_Sys$ = 'English Units') then
        Q_dot_t$ = CONCAT$(STRING$(Q_dot_t_inp), ' [G/min]')
        T_t_i$ = CONCAT$(STRING$(T_t_i_inp), ' [F]')
        T_tank$ = CONCAT$(STRING$(T_tank_inp), ' [F]')
        T_t_o_out = ConvertTemp('C', 'F', T_t_o)
        T_t_o$ = CONCAT$(STRING$(T_t_o_out), ' [F]')
        DELTAP_t_out = DELTAP_t*convert(Pa, psi)
        DELTAP_t$ = CONCAT$(STRING$(DELTAP_t_out), ' [psi]')
        UA_out = UA*convert(W/K, Btu/h-F)
        UA$ = CONCAT$(STRING$(UA_out), ' [Btu/h-F]')
        DELTAT_lm = abs(LMTD*convert(C,F))
        DELTAT_lm$ = CONCAT$(STRING$(DELTAT_lm), ' [F]')
        Q_act = abs(Q_tot*convert(W, Btu/h))
        Q_act$ = CONCAT$(STRING$(Q_act), ' [Btu/h]')
    Endif
End

"! Preparation ========================================="

Procedure Check_range_2_(T_t_i, T_tank,D_t_inner, D_t_outer: Check_range_2)
    If (D_t_inner > D_t_outer) then CALL ERROR('The Inner diameter of tubes (I.D.) is
greater than outer diameter of them (O.D.).',D_t_inner)
    if (T_t_i >100) then CALL ERROR('This is sensible heat exchanger. Tube inlet
temperature should be lower than boiling temperature.',T_t_i)
    if (T_tank >100) then CALL ERROR('This is sensible heat exchanger. Shell inlet
temperature should be lower than boiling temperature.',T_s_i)
    if (T_t_i < 0) then CALL ERROR('This is sensible heat exchanger. Tube inlet
temperature should be higher than melting temperature.',T_t_i)
```

```
      if (T_tank < 0) then CALL ERROR('This is sensible heat exchanger. Shell inlet
temperature should be higher than melting temperature.',T_s_i)
      if (T_tank = T_t_i) then CALL ERROR('Since the inlet temperatures of Shell and tube
sides are the same, there is no heat transfer.',T_s_i)
      Check_range_2 = 0
End

"! Tank Calculations ======================================="

Procedure h_tank_(Fluid_tank$, SolPer_tank, T_tank, P_tank, C1_tank,n_tank, D_t_outer,
LMTD_corr:h_tank)
      g = 9.81        "[m/s^2]"
      rho_tank = rho_(Fluid_tank$, T_tank, P_tank, SolPer_tank)  "[kg/m3]"
      C_p_tank = C_p_(Fluid_tank$, T_tank, P_tank, SolPer_tank)*convert(kJ, J)  "[J/kg-K]"
      mu_tank = mu_(Fluid_tank$, T_tank, P_tank, SolPer_tank)  "[kg/s-m]"
      beta_tank = beta_water_(T_tank)
      k_tank_fluid = k_(Fluid_tank$, T_tank, P_tank, SolPer_tank)            "[W/m-K]"
      Ra_tank =
(rho_tank^2*beta_tank*C_p_tank*g)/(mu_tank*k_tank_fluid)*D_t_outer^3*abs(LMTD_corr)
      "!Check!"
      if (Ra_tank <0) then Ra_tank = -Ra_tank
      Nusselt_tank = C1_tank*Ra_tank^n_tank
      h_tank = Nusselt_tank*k_tank_fluid/D_t_outer            "[W/m^2-K]"
End

"! Calculation of LMTD ====================================="

Procedure LMTD_2_(T_t_i, T_t_o, T_tank: LMTD)
      DELTAT_1 = T_t_o-T_tank
      DELTAT_2 = T_t_i-T_tank
      if (DELTAT_1 = 0) then DELTAT_1 = 1
      DELTAT_2_1 = DELTAT_2/DELTAT_1
      If (DELTAT_2_1 <= 0) then DELTAT_2_1 = 2
      if (DELTAT_2_1 = 1) then DELTAT_2_1 = 2
      LMTD=(DELTAT_2-DELTAT_1)/ln(DELTAT_2_1)
End

"! Main Program ============================================="

"
Pre-processing ============================================="
Call Unit_conversion_21_(Unit_Sys$, k_tube_inp, D_t_inner_inp, D_t_outer_inp, L_t_inp,
k_fin_inp, H_fin_inp, t_fin_inp: k_tube, D_t_inner, D_t_outer, L_t, k_fin, H_fin, t_fin)
Call Unit_conversion_22_(Unit_Sys$, Q_dot_t_inp, T_t_i_inp,  P_t_i_inp, T_tank_inp,
P_tank_inp : Q_dot_t, T_t_i, P_t_i, T_tank, P_tank)

Call Check_range_2_(T_t_i, T_tank,D_t_inner, D_t_outer: Check_range_2)

"
Heat transfer areas"
A_t_inner = L_t*pi*D_t_inner
A_t_outer = L_t*pi*D_t_outer
```

```
"
Calculation ======================================="
"
Tube Side"
      T_t = (T_t_i+T_t_o)/2
      rho_t = rho_(Fluid_t$, T_t, P_t_i, SolPer_t)   "[kg/m3]"
      C_p_t = C_p_(Fluid_t$, T_t, P_t_i, SolPer_t)*convert(kJ, J)  "[J/kg-K]"
      m_dot_t = Q_dot_t*rho_t   "[kg/s]"
      C_t = m_dot_t*C_p_t
      Call Circular_tube_(m_dot_t, D_t_inner, L_t, Fluid_t$, T_t, P_t_i, SolPer_t, Num_tube :
h_t, DELTAP_t, Re_t, DELTAP_t_1, DELTAP_t_2)
      Power_t = DELTAP_t*Q_dot_t

"
Tank Side "
      Fluid_tank$ = 'Water'
      SolPer_tank = 50

      C1_tank = 0.513
      n_tank = 0.25

      Call h_tank_(Fluid_tank$, SolPer_tank, T_tank, P_tank, C1_tank,n_tank,
D_t_outer,LMTD_corr:h_tank)

      N_fin = Fin_pitch*L_t*Num_tube            "Total number of fin"
      Call fin_annular_(fin_YN$, N_fin, h_fin, D_t_outer, t_fin, h_tank, k_fin : A_fin, eta_fin)

"
Overall Calculation ================================="
"Total heat transfer coefficient*Area"
      R_t_inner = 1/(h_t*A_t_inner)
      R_w = ln(D_t_outer/D_t_inner)/(2*pi*k_tube*L_t)
      R_t_outer = 1/(h_tank*(A_t_outer+eta_fin*A_fin))

      UA = Num_tube/(R_t_inner+R_w+R_t_outer)

"Total Heat Transfer rate"
      Q_tot = C_t*(T_t_i-T_t_o)
      Q_tot = UA*LMTD

"Log Mean Temperature Difference"
      Call LMTD_2_(T_t_i, T_t_o, T_tank: LMTD)
      LMTD_corr = LMTD-Q_tot*(R_t_inner+R_w)/Num_tube

"
Post-Processing for output ==================================="
Call Out_1_(Fluid_t$, SolPer_t: Fluid_t_out$)
DELTAT_lm = abs(LMTD)
Q_act = abs(Q_tot)
Call Unit_conversion_26_(Unit_Sys$, Q_dot_t_inp, T_t_i_inp, T_tank_inp, T_t_o,
DELTAP_t, UA, Q_tot, LMTD :  Q_dot_t$, T_t_i$, T_tank$, T_t_o$, DELTAP_t$, UA$,
Q_act$, DELTAT_lm$)
```

3. THERPROP.LIB

```
"Heat Exchanger Design Program for Solar Domestic Heating Water System"
"Common Library File, Therprop.lib"

"Jae-Mo Koo (koo@sel.me.wisc.edu)
William A. Beckman, Professor (beckman@me.engr.wisc.edu)
Solar Energy Laboratory
University of Wisconsin - Madison"

"This library file contains thermodynamic property functions and some other routines for
Heat Exchanger Design Program."


"! Thermodynamic Properties ====================================="

"
Property Procedure for Propylene glycol and Ethylene glycol solution"
Procedure Prop_Coeff_(Fl, Pr: c[1..18])
      if (Fl = 1) then
            If (Pr=1) then
                  c[1] = -2.196E+0001;c[2] =  9.186E-0006 ;c[3] = -1.136E-0006 ;c[4] =
1.798E-0008 ;c[5] = -1.015E+0000 ;c[6] = -3.391E-0006 ;c[7] =  7.548E-0008 ;c[8] =
8.493E-0010 ;c[9] = -1.414E-0002
                  c[10]=  2.355E-0007;c[11]=  2.159E-0009;c[12]= -1.311E-0010;c[13]= -
3.727E-0005;c[14]=  6.091E-0010;c[15]= -1.465E-0010;c[16]= -1.398E-0007;c[17]= -
2.269E-0010;c[18]= -2.512E-0008
            Endif
            If (Pr=2) then
                  c[1] =  1.056E+0003;c[2] = -3.987E-0001;c[3] = -3.068E-0003;c[4] =  1.233E-
0005;c[5] =  1.505E+0000;c[6] = -8.953E-0003;c[7] =  6.378E-0005;c[8] = -1.152E-
0007;c[9] = -1.634E-0003
                  c[10]=  1.541E-0004;c[11]= -1.874E-0006;c[12]= -9.809E-0009;c[13]= -
2.317E-0004;c[14]=  2.549E-0006;c[15]= -5.523E-0008;c[16]= -8.510E-0006;c[17]= -
3.848E-0008;c[18]= -1.128E-0007
            Endif
            If (Pr=3) then
                  c[1] =  3.501E+0003;c[2] =  3.954E+0000;c[3] =  6.065E-0005;c[4] = -5.979E-
0006;c[5] = -2.419E+0001;c[6] =  1.031E-0001;c[7] =  4.312E-0005;c[8] =  5.168E-
0006;c[9] =  4.613E-0003
                  c[10]= -6.595E-0005;c[11]=  1.620E-0005;c[12]= -3.250E-0007;c[13]=
6.028E-0003;c[14]=  5.642E-0005;c[15]= -7.777E-0007;c[16]= -7.977E-0005;c[17]=
5.190E-0007;c[18]= -3.380E-0006
            Endif
            If (Pr=4) then
                  c[1] =  4.211E-0001;c[2] =  7.995E-0004;c[3] = -5.509E-0008;c[4] = -1.460E-
0008;c[5] = -3.694E-0003;c[6] = -1.751E-0005;c[7] =  6.656E-0008;c[8] =  2.017E-0009;c[9]
=  2.095E-0005
```

```
                c[10]=  2.078E-0007;c[11]= -2.394E-0009;c[12]= -6.772E-0011;c[13]=
3.663E-0007;c[14]= -5.272E-0009;c[15]= -1.126E-0010;c[16]= -6.389E-0009;c[17]= -
1.112E-0010;c[18]= -1.820E-0010
            Endif
            If (Pr=5) then
                c[1] =  1.453E+0000;c[2] = -3.747E-0002;c[3] =  2.842E-0004;c[4] = -8.025E-
0007;c[5] =  2.920E-0002;c[6] = -1.131E-0004;c[7] =  1.729E-0006;c[8] = -5.073E-0008;c[9]
=  1.264E-0004
                c[10]=  6.785E-0009;c[11]= -1.685E-0008;c[12]= -1.082E-0009;c[13]=
4.386E-0006;c[14]= -2.191E-0007;c[15]= -9.117E-0011;c[16]= -9.223E-0008;c[17]= -
4.294E-0009;c[18]= -3.655E-0009
            Endif
      Endif

      if (Fl=2) then
            if (Pr=1) then
                c[1] = -2.360E+0001 ;c[2] = -2.970E-0004 ;c[3] =  1.871E-0007 ;c[4] =
1.590E-0007 ;c[5] = -1.088E+0000 ;c[6] =  4.693E-0005 ;c[7] = -9.089E-0007 ;c[8] = -
9.845E-0010 ;c[9] = -2.815E-0002
                c[10]=  1.613E-0006 ;c[11]=  5.066E-0008 ;c[12]= -1.251E-0009 ;c[13]= -
5.285E-0004 ;c[14]= -1.786E-0007 ;c[15]=  2.051E-0009 ;c[16]=  9.794E-0006 ;c[17]= -
5.668E-0009 ;c[18]=  4.270E-0007
            Endif
            If (Pr=2) then
                c[1] =  1.042E+0003;c[2] = -4.907E-0001;c[3] = -2.819E-0003;c[4] = -5.895E-
0007;c[5] =  8.081E-0001;c[6] = -9.652E-0003;c[7] =  7.168E-0005;c[8] =  2.404E-0007;c[9]
= -7.156E-0003
                c[10]=  1.088E-0004;c[11]= -3.328E-0006;c[12]=  1.153E-0007;c[13]=
1.190E-0004;c[14]= -6.226E-0006;c[15]= -3.026E-0008;c[16]= -1.170E-0005;c[17]= -
2.915E-0007;c[18]= -6.033E-0007
            Endif
            If (Pr=3) then
                c[1] =  3.679E+0003;c[2] =  1.571E+0000;c[3] =  1.331E-0002;c[4] =  1.975E-
0007;c[5] = -1.933E+0001;c[6] =  1.118E-0001;c[7] = -1.108E-0003;c[8] =  4.924E-
0006;c[9] = -4.879E-0002
                c[10]= -2.338E-0004;c[11]=  2.753E-0005;c[12]= -3.148E-0007;c[13]=
4.749E-0003;c[14]= -2.621E-0005;c[15]=  1.286E-0006;c[16]= -2.871E-0004;c[17]= -
9.050E-0008;c[18]= -1.068E-0005
            Endif
            if (Pr=4) then
                c[1] =  3.806E-0001;c[2] =  5.765E-0004;c[3] = -3.477E-0007;c[4] = -6.041E-
0009;c[5] = -3.815E-0003;c[6] = -1.423E-0005;c[7] = -1.203E-0008;c[8] = -5.854E-0010;c[9]
=  8.420E-0006
                c[10]=  1.081E-0007;c[11]=  1.959E-0009;c[12]=  1.271E-0010;c[13]= -
1.110E-0006;c[14]= -1.612E-0009;c[15]=  3.005E-0010;c[16]=  5.503E-0009;c[17]=
1.437E-0010;c[18]=  1.290E-0009
            Endif
            if (Pr=5) then
                c[1] =  2.274E+0000;c[2] = -5.342E-0002;c[3] =  5.372E-0004;c[4] = -4.955E-
0006;c[5] =  4.500E-0002;c[6] = -5.488E-0004;c[7] =  1.845E-0006;c[8] =  1.192E-0007;c[9]
= -7.808E-0005
```

```
                c[10]=  1.453E-0006;c[11]= -2.816E-0007;c[12]=  8.562E-0009;c[13]=
6.565E-0006;c[14]= -4.032E-0007;c[15]= -1.212E-0009;c[16]=  6.441E-0007;c[17]= -
1.430E-0008;c[18]=  1.092E-0008
            Endif
        Endif
End

Procedure Prop_Mean_(Fl:xm, ym)
        If (Fl=1) then
            xm = 38.1615
            ym = 6.3333
        Endif
        If (Fl=2) then
            xm = 42.7686
            ym = 5.3571
        Endif
End


Function BRINEPROP_(Fl$,Pr$,Conc,Temp)

        IF (Fl$='EG') AND (Conc<0) THEN
            Call ERROR('Concentration must be equal to or larger than 0.0%.',Conc)
            ELSE
            IF (Fl$='EG') AND (Conc>56.1) THEN
                Call ERROR('Concentration must be equal to or smaller than 56.1%.',Conc)
            ELSE
            IF (Fl$='PG') AND (Conc<15.2) THEN
                Call ERROR('Concentration must be equal to or larger than 15.2%.',Conc)
            ELSE
            IF (Fl$='PG') AND (Conc>57.0) THEN
                Call ERROR('Concentration must be equal to or smaller than 57.0%.',Conc)
        ENDIF;ENDIF;ENDIF;ENDIF

        "Sets property parameter"
        Freeze=1;Density=2;SpecHeat=3;ThermalC=4;DynVisc=5
        IF (Pr$='Freeze') THEN Pro:=Freeze ELSE
            IF (Pr$='Density') THEN Pro:=Density ELSE
                IF (Pr$='SpecHeat') THEN Pro:=SpecHeat ELSE
                    IF (Pr$='ThermalC') THEN Pro:=ThermalC ELSE
                        IF (Pr$='DynVisc') THEN Pro:=DynVisc
        ENDIF;ENDIF;ENDIF;ENDIF;
        Pr:=Pro

        "Sets fluid parameter"
        EG=1;PG=2
        IF (Fl$='EG') THEN Flu:=EG ELSE
        IF (Fl$='PG') THEN Flu:=PG ELSE
        ENDIF;ENDIF
        Fl:=Flu

        "Coefficients"
        call Prop_Coeff_(Fl, Pr: c[1..18])
```

```
      "Mean values"
      call Prop_Mean_(Fl:xm, ym)

      x=Conc-xm
      y=Temp-ym

      "Equation, f(temp,conc)"
     Funkt=c[1]+(c[2]*y)+(c[3]*y^2)+(c[4]*y^3)+(c[5]*x)+(c[6]*x*y)+(c[7]*x*y^2)+(c[8]*x*y^3)+
(c[9]*x^2)+(c[10]*x^2*y)+(c[11]*x^2*y^2)+(c[12]*x^2*y^3)+(c[13]*x^3)+(c[14]*x^3*y)+(c[15]*x
^3*y^2)+(c[16]*x^4)+(c[17]*x^4*y)+(c[18]*x^5)

      "
      Conditions
      Mode=mu   =>  output = exp(f)
      Mode<>mu =>  output = f"

      IF (Pr=5) THEN
           BRINEPROP_=exp(Funkt)
      ELSE
      IF (Pr=3) THEN
           BRINEPROP_=Funkt/1000
      ELSE
           BRINEPROP_=Funkt
      ENDIF;ENDIF
END

"
User External Functions"

"Functions to get properties of fluid"
Function C_p_(Fluid$, T, P, Percent)
      If (Fluid$ = 'Water') then C_p_ = SPECHEAT(Water,T=T,P=P)
      If (Fluid$ = 'Propylene Glycol Solution') then C_p_ =
BRINEPROP_('PG','SpecHeat',Percent,T)
      If (Fluid$ = 'Ethylene-Glycol Solution') then C_p_ =
BRINEPROP_('EG','SpecHeat',Percent,T)
End

Function mu_(Fluid$, T, P, Percent)
      If (Fluid$ = 'Water') then mu_ = VISCOSITY(Water,T=T,P=P)
      If (Fluid$ = 'Propylene Glycol Solution') then mu_ =
BRINEPROP_('PG','DynVisc',Percent,T)*1.0e-3
      If (Fluid$ = 'Ethylene-Glycol Solution') then mu_ =
BRINEPROP_('EG','DynVisc',Percent,T)*1.0e-3
End

Function k_(Fluid$, T, P, Percent)
      If (Fluid$ = 'Water') then k_ = CONDUCTIVITY(Water,T=T,P=P)
      If (Fluid$ = 'Propylene Glycol Solution') then k_ =
BRINEPROP_('PG','ThermalC',Percent,T)
      If (Fluid$ = 'Ethylene-Glycol Solution') then k_ =
BRINEPROP_('EG','ThermalC',Percent,T)
```

```
End


Function rho_(Fluid$, T, P, Percent)
      If (Fluid$ = 'Water') then rho_ = DENSITY(Water,T=T,P=P)
      If (Fluid$ = 'Propylene Glycol Solution') then rho_ =
BRINEPROP_('PG','Density',Percent,T)
      If (Fluid$ = 'Ethylene-Glycol Solution') then rho_ =
BRINEPROP_('EG','Density',Percent,T)
End

Function beta_water_(T)    "Temperature unit : C"
      if (T<0) then T = 0
      if (T>100) then T = 100
      beta=-64.99 + 17.236*T - 0.21274*T^2 + 0.0018762*T^3 - 0.0000066052*T^4
      beta_water_ = beta*1.0e-6
End


"!Heat transfer coefficient and pressure drop in circular tubes"

Procedure Circular_tube_(m_dot, D, L, Fluid$, T, P, SolPer, Num_tube : h, DELTAP, Re,
DELTAP_1, DELTAP_2)
      rho = rho_(Fluid$, T, P, SolPer)"[kg/m3]"
      C_p = C_p_(Fluid$, T, P, SolPer)*convert(kJ, J)  "[J/kg-K]"
      mu = mu_(Fluid$, T, P, SolPer)"[kg/s-m]"
      k = k_(Fluid$, T, P, SolPer)            "[W/m-K]"
      Re = 4*(m_dot/Num_tube)/(pi*D*mu)
      Pr = mu*C_p/k
      A_inner = pi*D^2/4*Num_tube
      u = m_dot/rho/A_inner

      if (Re > 2300) then "for turbulent flow"
            fricfac = (0.79*ln(Re)-1.64)^(-2)
            {Nusselt = 0.023*Re^0.8*Pr^0.4            "Dittus-Doelter"}
            Nusselt_long = ((fricfac/8)*(Re-1000)*Pr)/(1+12.7*sqrt(fricfac/8)*(Pr^(2/3)-1))
      "Gnielinski"
            Nusselt = Nusselt_long*(1+(D/L)^0.7)
      else "For laminar flow"
            fricfac = 64/Re
            {Nusselt = 4.364}
      {for laminar flow (assumption : constant heat flux condition and fully developed flow)}
            a = 0.00172                        {Table 3.14.1 from Duffie & Beckman}
            b = 0.00281
            m = 1.66
            n = 1.29
            Nusselt = 4.4+a*(Re*Pr*D/L)^m/(1+b*(Re*Pr*D/L)^n)  {Equation 3.14.5}
      Endif
      "Heat Transfer Coefficient"
      h = Nusselt*k/D          "[W/m^2-K]"
      "Pressure Drop"
      DELTAP_1 = fricfac*(rho*u^2)/(2*D)*L    "Pressure drop of fully developed flow"
      K = 0.5+1.0   "Loss Coefficient for Sharp edge at inlet and exit"
```

```
    DELTAP_2 = K*D*(rho*u^2)/(2*D)    "Pressure drop due at pipe entrance : Sharp
Edge"
    DELTAP = DELTAP_1+DELTAP_2
END

"Function for Fins around Tubes"
Procedure  fin_annular_(fin_YN$, N_fin, h_fin, D_tube, t_fin,h_o, k_fin : A_fin, eta_fin)
    if (fin_YN$ = 'Annular fins around tubes') then
        r_tube = D_tube/2
        r_fin = r_tube+h_fin
        r_fin_c = r_fin+(t_fin/2)      "Corrected length of fin"
        A_f_1 = 2*pi*(r_fin_c^2-r_tube^2)
        A_fin = N_fin*A_f_1
        m = sqrt(2*h_o/(k_fin*t_fin))
        C_1 = (2*r_tube/m)/(r_fin_c^2-r_tube^2)
        mr_1 = m*r_tube
        mr_2 = m*r_fin_c
        eta_fin = C_1*(Bessel_K1(mr_1)*Bessel_I1(mr_2)-
Bessel_I1(mr_1)*Bessel_K1(mr_2))/(Bessel_I0(mr_1)*Bessel_K1(mr_2)+Bessel_K0(mr_1)
*Bessel_I1(mr_2))
    else
        eta_fin = 0
          A_fin = 0
    Endif
End

"! Procedures for output ============================================="

Procedure Out_1_(Fluid$,SolPer : Fluid_out$)
    if (Fluid$ = 'Water') then Fluid_out$ = Fluid$
    if (Fluid$ = 'Propylene Glycol Solution') or (Fluid$ = 'Ethylene-Glycol Solution') then
        Dum1$ = CONCAT$(Fluid$,' (')
        Dum2$ = CONCAT$(Dum1$, String$(SolPer))
        Fluid_out$ = CONCAT$(Dum2$, '%)')
    Endif
End
```

REFERENCES

1. ASHRAE Standard 93-86, *Methods of Testing to Determine the Thermal Performance of Solar Collectors*, American Society of Heating, Refrigeration, and Air Conditioning Engineers, 1986.

2. Brandemuehl, M. J. and Beckman, W. A., "Transmission of Diffuse Radiation through CPC and Flat-Plate Collector Glazings," Solar Energy, Vol. 24, pp. 511, 1980.

3. Duffie, J. A. and Beckman W. A., *Solar Engineering of Thermal Processes*, 2nd Edition, John Wiley & Sons, New York, 1991.

4. Hollands, K. G. T., Unny, T. E., Raithby, G. D., and Lonicek, L, "Free Convection Heat Transfer Across Inclined Air Layers," Transactions of ASME Journal of Heat Transfer, Vol. 98, pp. 189, 1976.

5. Incropera, F. P. and DeWitt, D. P., *Fundamentals of Heat and Mass Transfer*, 4th Edition, John Wiley & Sons, New York, 1996.

6. Klein, S. A. and Alvarado, F. L., *EES: Engineering Equation Solver*, F-Chart Software, Middleton, WI, 1999.

7. Klein, S. A. et al., *TRNSYS 14 Manual*, Solar Energy Laboratory, University of Wisconsin-Madison, WI, 1994.

8. Liu, B. Y. H. and Jordan, R. C., "The Interrelationship and Characteristic Distribution of Direct, Diffuse, and Total Solar Radiation," *Solar Energy*, Vol 4, pp. 1, 1960.

9. Liu, W., Davidson, J., Raman, R., and Mantell, S., "Thermal and Economic Analysis of Plastic Heat Exchangers for Solar Water Heating," Proceedings of ASES Annual Conference, Portland, Maine, USA, June 12-16, pp. 101, 1999.

10. Newton, B. J., Schmid, M., Mitchell, J. W., and Beckman, W. A., "Storage Tank Models," Proceedings of the ASME International Solar Energy Conference, Vol. 2, pp. 1111, 1995.

11. Pettit, R. B. and Sowell, R. P., "Solar Absorptance and Emittance Properties," *Journal of Vacuum Science and Technology*, Vol. 13, No. 2, pp. 596, 1976.

12. Siegel, R. and Howell, J. R., Thermal Radiation Heat Transfer, 3rd Edition, Taylor & Francis, New York, 1992.

13. Solar Collector Test Report on MSC-32 American Energy Technologies Inc., Testing And Laboratories Division, Florida Solar Energy Center, 1988.

14. Solar Collector Test Report on STG-24 Sun Trapper Solar Systems, Inc., Testing And Laboratories Division, Florida Solar Energy Center, 1995.

15. Solar Collector Test Report on SX-600 Solmax Inc., Testing And Laboratories Division, Florida Solar Energy Center, 1993.

16. Souka, A. F. and Safwat, H. H, "Optimum Orientations for the Double Exposure Flat-Plate Collector and Its Reflector," Solar Energy, Vol. 10, pp. 170, 1966.

17. SRCC Document RM-1, *Methodology for Determining the Thermal Performance Rating for Solar Collectors*, Solar Rating and Certification Corporation, Florida, 1994.

18. Su, B. and El-Genk, M. S., "Forced Convection of Water in Rod-Bundles, Int. Comm. Heat and Mass Transfer, Vol. 20, pp. 295, 1993.

19. Whillier, A., "Prediction of Performance of Solar Collectors," Applications of Solar Energy for Heating and Cooling of Buildings, ASHRAE, New York, pp. VIII-1, 1977.

20. White, F. M, *Fluid Mechanics*, 2nd Edition, McGraw-Hill, New York, 1987.

21. Yiqin, Y, Hollands, K. G. T., Brunger, A. P., " Measured Top Heat Loss Coefficients for Flat Plate Collectors with Inner Teflon Covers," Proceedings of the Biennial Congress of the International Solar Energy Society, Denver, Colorado, USA, August 19-23, pp. 1200, 1991.