{ TC "'APPENDIX B: TRNSYS PV-SDHW COMPONENTS':" \l 1 \n }APPENDIX

B

TRNSYS PV-SDHW COMPONENTS

B.1 TRNSYS Component Model of a Photovoltaic Array{ TC "B.1 TRNSYS Component Model of a Photovoltaic Array" \l 2 }

```
C*************************************************************************
C                       TYPE62.FOR
          SUBROUTINE TYPE62(TIME,XIN,OUT,T,DTDT,PAR,INFO,ICONTROL,*)
C
C      This type simulates the electrical performance
C      of a photovoltaic array.
C
C      Originally developed    by      Y. Eckstien (1990)
C      Major modification      by      A. M. Al-Ibrahim (1995)
C
C      Convergence promoting feature added by B. Flake (1995)
C*************************************************************************
C
C**** declaration of the variables:

          implicit none

          Integer   INFO, ICONTROL
          Integer   N_in, N_out, N_parm, Mode_pv
          Integer   count1, count2, count3, Fail1, Fail2, Fail3

          PARAMETER (N_in = 6, N_out = 19, N_parm = 18)

          Real      PAR, TIME, T, DTDT
          Real*8    XIN, OUT

          Real*8   G_t, G_t_ref, G_t_limit, DINRG
          Real*8   pi, eps
          Real*8   T_amb, T_cell_ref, T_cell, T_cell_1, T_cell_0
          Real*8   E_q, MVoc, MIsc
          Real*8   U_L, tau_alfa
          Real*8   Isc_ref, Voc_ref, Vmp_ref, Imp_ref
          Real*8   Io_ref, A_ref, Rs_ref, IL_ref
          Real*8   Eff_mp_pv, Eff_pv, Eff_pv_ref
          Real*8   Area_module, Area_array, length, width
          Real*8   I, V, Io, Rs, P, Imp, Vmp, Pmax
```

```
2
          Real*8   A, IL, Isc, Voc
          Real*8   UTILIZ, FF
          Real*8   Imp0, Imp1, F, FP, X, W, M, IscVoc
          Real*8   i_0, i_1, f_i_0, df_i_0
          Real*8   NCS, Np_pv, Ns_pv
c
          double precision I_old, V_old, alpha

     common/lunits/lur,luw,iform,luk
     integer lur,luw,iform,luk
          Dimension  XIN(6), OUT(19), PAR(18), INFO(15)

C*************************************************************

C    In this section a couple of checks on the inputs are
C    performed. This is done at the second and following
C    calls in timestep

C*************************************************************

C****   initial call in simulation

          if (INFO(7) .eq. -1) then

C***    Conveying to TRNSYS the number of outputs expected

                    INFO(6) = 19

C***    Conveying to TRNSYS that this type does not depend upon the passage of time

                    INFO(9) = 1

C****   parameters setting ***************************

                    Mode_pv = NINT(PAR(1))

                    if (Mode_pv .eq. 1) then
C***    Direct Coupling the PV to the Load

                      CALL TYPECK(1, INFO, 6, N_parm, 0)

                    else if (Mode_pv .eq. 2) then
C***    PV is connceted to a Maximum Power Point Tracker

                      CALL TYPECK(1, INFO, 3, N_parm, 0)

                    else
               write(luw,*)'The paramter, Mode_pv, has been'
               write(luw,*)'assigned an unacceptable value '
               write(luw,*)'Mode_pv = ',Mode_pv,'   (not 1 or 2)'
               write(luw,*)'Please check the value of the paramter.'
               write(luw,*)'PROGRAM WILL TERMINATE'

               call mystop(1001)
               return 1
```

```
                    end if

C***    Parameters related to the PV panel

                    G_t_ref      = PAR(2)
                    T_cell_ref   = PAR(3)
                    Isc_ref      = PAR(4)
                    Voc_ref      = PAR(5)
                    Imp_ref      = PAR(6)
                    Vmp_ref      = PAR(7)
                    MIsc         = PAR(8)
                    MVoc         = PAR(9)
                    tau_alfa     = PAR(10)
                    E_q          = PAR(11)
                    NCS          = PAR(12)
                    width        = PAR(13)
                    length       = PAR(14)
                    Ns_pv        = PAR(15)
                    Np_pv        = PAR(16)
                    DINRG        = PAR(17)
c
                    alpha        = par(18)

C***    constants
                    eps = 1.e-4
                    pi = 4 * atan(1.)


C***    given the above parameters, further parameters need to be evaluated for each module

                    Area_module = width * length


                    Eff_pv_ref = Imp_ref * Vmp_ref / (G_t_ref * Area_module)

C****   evaluation of the four unknowns at the reference condition

                    IL_ref = Isc_ref

                    A_ref  = ((MVoc * T_cell_ref) - Voc_ref + E_q * NCS) /
   .            (((MIsc * T_cell_ref) / IL_ref) - 3.)

                    Io_ref = IL_ref / (exp(Voc_ref / A_ref) - 1.)

                    Rs_ref = (A_ref * log(1. - Imp_ref / IL_ref) -
   .               Vmp_ref + Voc_ref) / Imp_ref


C****   set up parameters for the entire array

                    IL_ref = Np_pv * IL_ref
```

```
                Io_ref = Np_pv * Io_ref

                A_ref  = Ns_pv * A_ref

                Rs_ref = (Ns_pv / Np_pv) * Rs_ref

                G_t_limit = G_t_ref / DINRG

                Area_array = Area_module * Ns_pv * Np_pv


                Return 1

        end if

C---------------------------------------------------------------

C****   first call in timestep

C****   set inputs

        G_t     = XIN(1) * 1000. / 3600. ! to convert from kj/hr-m2 to W/m2
        T_amb   = XIN(2) + 273.15        ! to convert from C to K
        U_L     = XIN(3)

        if (Mode_pv .eq. 1) then

          V     = XIN(4)

        end if


        Fail1 = 0
        Fail2 = 0
        Fail3 = 0


        if (G_t .le. G_t_limit) then

C***    Abort all the PV part, to avoid log of zeros or negative #s

          I = 0.0
          V = 0.0
          P = 0.0
          Imp = 0.0
          Vmp = 0.0
          Pmax = 0.0
          Voc  = 0.0
          Isc  = 0.0
          FF   = 0.0
          UTILIZ = 0.0
          Eff_pv = 0.0
          Eff_mp_pv = 0.0
          T_cell = T_amb
```

```
            Fail1 = -1

C****   Jump to the outputs part

            go to 2000

           end if

C***    Assuming a suitable value for the cell efficiency.  The assumed
C       value of the cell efficiency will be compared to a better guess
C       value until the difference is less than an allowable tolerance; 0.01.

           T_cell_0 = 0.0
           T_cell_1 = T_amb + (G_t * tau_alfa / U_L) *
     .              (1. - Eff_pv_ref / tau_alfa)


           count1 = 0

           do while ( abs(T_cell_1 - T_cell_0) .gt. eps*100)

            count1 = count1 + 1

            T_cell_0 = T_cell_1

C****   this part calculates how IL, Io, and A vary with T_cell and G_t

            IL = (G_t / G_t_ref) * (IL_ref + MIsc * Np_pv *
     .         (T_cell_0 - T_cell_ref))


            if (IL .lt. 0.0) then
              IL=0.0
            end if

            Io = Io_ref * ((T_cell_0 / T_cell_ref)**3) *
     .         exp(((NCS * Ns_pv * E_q) / A_ref) *
     .         (1. - (T_cell_ref/T_cell_0)))

            A  = A_ref * (T_cell_0 / T_cell_ref)

C****   Assuming that Rs is independent of solar radiation and cell temperature

            Rs = Rs_ref


C****   Calculating the open circuit voltage

            Voc = A * log(IL / Io)



C****   Calculating the short circuit current

            Isc = IL
```

```
C***    Computation of the current and voltage if the PV is directly coupled to a load.
C       However, if the PV is connected to a maximum power point tracker, it skips this
C       part and jumps to the maximum power point calculations part.

            if (Mode_pv .eq. 2) then
              I = 0.0
              V = 0.0
              go to 500
            end if

C****   Check if voltage is greater than open circuit voltage or smaller than 0.0


            if (V .gt. Voc) then

              V = Voc
              I = 0.0

            else if (V .le. 0.0) then

              V = 0.0
              I = Isc

            else

              count2 = 0
              i_0 = 0.0
              i_1 = IL

              do while (abs(i_1-i_0).gt.eps)

                    count2 = count2 + 1

                    i_0=i_1

                    f_i_0 = i_0 - IL + Io * (exp((V + i_0 * Rs) / A) - 1.)

                    df_i_0 = 1 + (Io * Rs/A) * exp(((V+i_0*Rs)/A))

                    i_1 = i_0 - (f_i_0/df_i_0)


              end do

              I = i_1

            end if

            P = I * V

 500    continue

C****   Maximum Power Point Calculation
```

```
C****   First guess

              count3 = 0
              Imp0 = 0.0

              Imp1 = G_t / G_t_ref * Np_pv * (Imp_ref + MIsc *
     .       (T_cell_0 - T_cell_ref))


              do while (abs(Imp1-Imp0) .gt. eps)

                count3 = count3 + 1

                Imp0 = Imp1

                M   = IL - Imp0 + Io
                W   = A * log(M / Io) - Imp0 * Rs
                X   = A + M * Rs

                F   = Imp0 - M * W / X

                FP  = 2. + W / X - M * Rs * W / X**2

                Imp1 = Imp0 - F/FP

              end do

              Imp = Imp1

C****   Now the maximum current is found; compute the voltage
C       that corresponds to this current

              Vmp = A * log((IL - Imp + Io) / Io) - Imp * Rs


              Pmax = Imp * Vmp

C***    if the PV is attached to a maximum power point tracker then the operating current
C       and voltage are identical to the maximum power current and voltage respectively.

              if (Mode_pv .eq. 2) then
                I = Imp
                V = Vmp
                P = Pmax
              end if

C*****  Computing the Fill Factor only if Voc AND/OR Isc are greater than 0.0

              IscVoc = Isc * Voc

              if (IscVoc .GT. EPS) then
                FF = Pmax / (Isc * Voc)
              else
                FF = 0.0
              end if
```

```
C*****  Computing the system utilizablity at this time of operation
C*****  defined as the ratio of the drawn power to the maximum power

              if (Pmax .gt. EPS) then

                UTILIZ = P / Pmax

              else

                UTILIZ = 0.0

              end if


C***  1) Efficiency for the direct-coupled PV to another load

              Eff_pv = P / (G_t * Area_array)


C***  2) Efficiency of the panel if operating at the maximum power point

              Eff_mp_pv = Pmax / (G_t * Area_array)


C***  Computing the cell temperature

              T_cell_1 = T_amb + (G_t * tau_alfa / U_L) *
     1                (1. - Eff_pv / tau_alfa)


             if (count1 .ge. 50) then
                     if (abs(T_cell_1-T_cell_0) .lt. 5.0) then
                             T_cell_1 = (T_cell_1 + T_cell_0)/2.0
                             FAIL2 = -1
                             go to 4000
                     else
                             T_cell_1 = 1.25 * T_amb

                             FAIL3 = -2
                     end if

             end if

            end do

4000      continue

            T_cell = T_cell_1

2000      continue

C***    DONE  DONE  DONE  DONE  DONE   ***
C****   Setting the outputs  ***********************************
```

```
c
c
          I_old = xin(5)
          V_old = xin(6)
c

          OUT(1)  = alpha*I + (1. - alpha)*I_old
          OUT(2)  = alpha*V + (1. - alpha)*V_old
          OUT(3)  = P
          OUT(4)  = Imp
          OUT(5)  = Vmp
          OUT(6)  = Pmax
          OUT(7)  = Voc
          OUT(8)  = Isc
          OUT(9)  = FF
          OUT(10) = UTILIZ
          OUT(11) = Eff_pv
          OUT(12) = Eff_mp_pv
          OUT(13) = T_cell    - 273.15   ! to convert from K to C
          OUT(14) = REAL(Fail1)
          OUT(15) = REAL(FAIL2)
          OUT(16) = REAL(FAIL3)
          OUT(17) = REAL(COUNT1)
          OUT(18) = REAL(COUNT2)
          OUT(19) = REAL(COUNT3)

          RETURN 1

          END
```

B.2 TRNSYS Component Model of a Multiple Resistor/Controller{ TC "B.2 TRNSYS Component Model of a Multiple Resistor/Controller" \l 2 }

```
C**********************************************************

      SUBROUTINE TYPE98(TIME,XIN,OUT,T,DTDT,PAR,INFO,ICNTRL,*)

C     This subroutine models the function of a resistive element
C     set and irradiance level controller for use in PV-SDHW
C     systems.  Either six resistors connected in six parallel
C     combinations or three resistors connected in seven parallel
C     combinations are possible.


C**** TRNSYS specific variables:
```

```
C    XIN == input array
C    OUT == output array
C    PAR == parameters
C    TIME == simulation time
C    T,DTDT == not used in this component
C    INFO == array to use TRNSYS internal information

C**** component specific variables:
C    I = current [Amps]
C    V = voltage [Volts]
C    P = power [Watts]
C    R = resistance [Ohms]
C    MODE_A = signal determining operating mode
C         MODE_A=1: six resistors (in two clusters) in six
C                                                  parallel combinations
C         MODE_A=2: three resistors (in one cluster) in
C                                                  seven parallel combinations
C    MODE_B == signal determining operating mode
C         MODE_B=1: current is significant input
C         MODE_B=2: voltage is signifianct input
C*****************************************************************

C**** declaration of variables:
*     IMPLICIT NONE
     INTEGER INFO,MODE_A,N_PARM,N_RES
     REAL ROPT,RUSED,IMAX,VMAX,V,I,P,TIME,T,DTDT,PAR,MODE_B
     REAL RUPPER,RLOWER,PUPPER,PLOWER,CURR
     DOUBLE PRECISION XIN,OUT
     LOGICAL RES
     DIMENSION XIN(6),OUT(14),PAR(12),INFO(15),R(7),SW(6),RES(7)
     DIMENSION CURR(7)

      INFO(6)=14


C---------------------------------------------------------------


C**** initial call of component

     IF(INFO(7).LT.0)THEN

          MODE_A=NINT(PAR(1))

          IF (MODE_A.EQ.1) THEN
               N_PARM=12
               N_RES=6
          ELSE
               N_PARM=10
               N_RES=7
          ENDIF

          CALL TYPECK(1,INFO,6,N_PARM,0)

C****   set parameters
```

```
                IF (MODE_A.EQ.1) THEN
                R(1)=PAR(2)
                R(2)=1/(1/PAR(2)+1/PAR(3))
                R(3)=1/(1/PAR(2)+1/PAR(3)+1/PAR(4))
                R(4)=1/(1/PAR(2)+1/PAR(3)+1/PAR(4)+1/PAR(5))
                R(5)=1/(1/PAR(2)+1/PAR(3)+1/PAR(4)+1/PAR(5)+1/PAR(6))
                R(6)=1/(1/PAR(2)+1/PAR(3)+1/PAR(4)+1/PAR(5)+1/PAR(6)
     +                          +1/PAR(7))
                SW(1)=PAR(8)
                SW(2)=PAR(9)
                SW(3)=PAR(10)
                SW(4)=PAR(11)
                SW(5)=PAR(12)
                ELSE
                R(1)=PAR(2)
                R(2)=PAR(3)
                R(3)=1/(1/PAR(2)+1/PAR(3))
                R(4)=PAR(4)
                R(5)=1/(1/PAR(2)+1/PAR(4))
                R(6)=1/(1/PAR(3)+1/PAR(4))
                R(7)=1/(1/PAR(2)+1/PAR(3)+1/PAR(4))
                CALL SORT(N_RES,R)
                SW(1)=PAR(5)
                SW(2)=PAR(6)
                SW(3)=PAR(7)
                SW(4)=PAR(8)
                SW(5)=PAR(9)
                SW(6)=PAR(10)
                ENDIF
    ENDIF
C-----------------------------------------------------------------

C**** set inputs

    I=XIN(1)
    V=XIN(2)
    MODE_B=XIN(3)
    IMAX=XIN(4)
    VMAX=XIN(5)
    IT=XIN(6)

C**** DETERMINATION OF OPTIMUM RESISTANCE VALUE BASED ON IMAX AND VMAX

    IF (IMAX.GT.0.)THEN
    ROPT=VMAX/IMAX
    ELSE
    ROPT=999
    ENDIF

C**** DETERMINATION OF RESISTANCE USED BASED ON SOLAR IRRADIANCE

        DO K = 1,7
                RES(K) = .FALSE.
     END DO
```

```
IF (MODE_A.EQ.1) THEN
        IF (IT.GE.0.0.AND.IT.LE.SW(1)) THEN
        RUSED = R(1)
        RUPPER = R(1)
        RLOWER = 0
        RES(1) = .TRUE.
        ENDIF
        IF (IT.GT.SW(1).AND.IT.LE.SW(2)) THEN
        RUSED = R(2)
        RUPPER = PAR(2)
        RLOWER = PAR(3)
        RES(1) = .TRUE.
        RES(2) = .TRUE.
        ENDIF
        IF (IT.GT.SW(2).AND.IT.LE.SW(3)) THEN
        RUSED = R(3)
        RUPPER = PAR(2)
        RLOWER = 1/(1/PAR(3)+1/PAR(4))
        RES(1) = .TRUE.
        RES(2) = .TRUE.
        RES(3) = .TRUE.
        ENDIF
        IF (IT.GT.SW(3).AND.IT.LE.SW(4)) THEN
        RUSED = R(4)
        RUPPER = PAR(2)
        RLOWER = 1/(1/PAR(3)+1/PAR(4)+1/PAR(5))
        RES(1) = .TRUE.
        RES(2) = .TRUE.
        RES(3) = .TRUE.
        RES(4) = .TRUE.
        ENDIF
        IF (IT.GT.SW(4).AND.IT.LE.SW(5)) THEN
        RUSED = R(5)
        RUPPER = 1/(1/PAR(2)+1/PAR(6))
        RLOWER = 1/(1/PAR(3)+1/PAR(4)+1/PAR(5))
        RES(1) = .TRUE.
        RES(2) = .TRUE.
        RES(3) = .TRUE.
        RES(4) = .TRUE.
        RES(5) = .TRUE.
        ENDIF
        IF (IT.GT.SW(5)) THEN
        RUSED = R(6)
        RUPPER = 1/(1/PAR(2)+1/PAR(6)+1/PAR(7))
        RLOWER = 1/(1/PAR(3)+1/PAR(4)+1/PAR(5))
        RES(1) = .TRUE.
        RES(2) = .TRUE.
        RES(3) = .TRUE.
        RES(4) = .TRUE.
        RES(5) = .TRUE.
        RES(6) = .TRUE.
        ENDIF

ELSE
```

```
c               THIS IS THE CASE OF THREE RESISTORS BEING USED IN SEVEN COMBINATIONS
('UPPER' IS
c               TAKEN TO BE THE SINGLE RESISTOR CLUSTER)

                IF (IT.GE.0.0.AND.IT.LE.SW(1)) THEN
                RUSED = R(7)
                RUPPER = R(7)
                RLOWER = 0.
                RES(1) = .TRUE.
                ENDIF
                IF (IT.GT.SW(1).AND.IT.LE.SW(2)) THEN
                RUSED = R(6)
                RUPPER = R(6)
                RLOWER = 0.
                RES(2) = .TRUE.
                ENDIF
                IF (IT.GT.SW(2).AND.IT.LE.SW(3)) THEN
                RUSED = R(5)
                RUPPER = R(5)
                RLOWER = 0.
                RES(1) = .TRUE.
                RES(2) = .TRUE.
                ENDIF
                IF (IT.GT.SW(3).AND.IT.LE.SW(4)) THEN
                RUSED = R(4)
                RUPPER = R(4)
                RLOWER = 0.
                RES(3) = .TRUE.
                ENDIF
                IF (IT.GT.SW(4).AND.IT.LE.SW(5)) THEN
                RUSED = R(3)
                RUPPER = R(3)
                RLOWER = 0.
                RES(1) = .TRUE.
                RES(3) = .TRUE.
                ENDIF
                IF (IT.GT.SW(5).AND.IT.LE.SW(6)) THEN
                RUSED = R(2)
                RUPPER = R(2)
                RLOWER = 0.
                RES(2) = .TRUE.
                RES(3) = .TRUE.
                ENDIF
                IF (IT.GT.SW(6)) THEN
                RUSED = R(1)
                RUPPER = R(1)
                RLOWER = 0.
                RES(1) = .TRUE.
                RES(2) = .TRUE.
                RES(3) = .TRUE.
                ENDIF
        ENDIF
```

```
      IF (MODE_B.EQ.1.) THEN
        V=I*RUSED

      ELSE

C****   voltage guess

        I=V/RUSED
      ENDIF

        DO L=1,7
            IF (RES(L).EQV..TRUE.) THEN
            CURR(L)=V/PAR(L+1)
                        ELSE
            CURR(L)=0.
                        ENDIF
        END DO

C**** DEFINE OVERALL POWER AND POWER TO INDIVIDUAL RESISTOR CLUSTERS

      P=I*V

      PUPPER=V*V/RUPPER

      IF (RLOWER.GT.0) THEN
       PLOWER=V*V/RLOWER
      ELSE
       PLOWER=0
      ENDIF

C**** set outputs
      OUT(1)=I
      OUT(2)=V
      OUT(3)=P
      OUT(4)=ROPT
      OUT(5)=RUSED
      OUT(6)=PUPPER
      OUT(7)=PLOWER
            OUT(8)=CURR(1)
            OUT(9)=CURR(2)
            OUT(10)=CURR(3)
            OUT(11)=CURR(4)
            OUT(12)=CURR(5)
            OUT(13)=CURR(6)
            OUT(14)=CURR(7)

      RETURN 1
      END

C     ------------------------------------------------------------
C
C     This subroutine is employed to sort the seven overall
C     resistances in the three resistor case since the order of
C     connection is not specified when the three resistor values
C     are given.
```

```
      SUBROUTINE sort(n,arr)
    INTEGER n,M,NSTACK
    REAL arr(n)
    PARAMETER (M=7,NSTACK=50)
    INTEGER i,ir,j,jstack,k,l,istack(NSTACK)
    REAL a,temp
    jstack=0
    l=1
    ir=n
1   if(ir-l.lt.M)then
     do 12 j=l+1,ir
      a=arr(j)
      do 11 i=j-1,l,-1
       if(arr(i).le.a)goto 2
       arr(i+1)=arr(i)
11     continue
      i=l-1
2      arr(i+1)=a
12    continue
     if(jstack.eq.0)return
     ir=istack(jstack)
     l=istack(jstack-1)
     jstack=jstack-2
    else
     k=(l+ir)/2
     temp=arr(k)
     arr(k)=arr(l+1)
     arr(l+1)=temp
     if(arr(l).gt.arr(ir))then
      temp=arr(l)

      arr(l)=arr(ir)
      arr(ir)=temp
     endif
     if(arr(l+1).gt.arr(ir))then
      temp=arr(l+1)
      arr(l+1)=arr(ir)
      arr(ir)=temp
     endif
     if(arr(l).gt.arr(l+1))then
      temp=arr(l)
      arr(l)=arr(l+1)
      arr(l+1)=temp
     endif
     i=l+1
     j=ir
     a=arr(l+1)
3     continue
      i=i+1
     if(arr(i).lt.a)goto 3
4     continue
      j=j-1
     if(arr(j).gt.a)goto 4
     if(j.lt.i)goto 5
```

```
      temp=arr(i)
      arr(i)=arr(j)
      arr(j)=temp
      goto 3
5      arr(l+1)=arr(j)
      arr(j)=a
      jstack=jstack+2

      if(jstack.gt.NSTACK)pause 'NSTACK too small in sort'
      if(ir-i+1.ge.j-l)then
        istack(jstack)=ir
        istack(jstack-1)=i
        ir=j-1
      else
        istack(jstack)=j-1
        istack(jstack-1)=l
        l=i
      endif
    endif
    goto 1
          RETURN
    END
```

B.3 Life Cycle Economic Analysis for PV-SDHW Systems{ TC "B.3 Life Cycle Economic Analysis for PV-SDHW Systems" \l 2 }

```
    subroutine type89 (time,xin,out,t,dtdt,par,info,icntrl,*)

************************************************
* LIFE CYCLE ECONOMIC ANALYSIS FOR PV-SDHW SYSTEMS
* (modified from program by Trzesniewski (1995))
************************************************

    implicit none

*  Standard TRNSYS declarations
    COMMON /SIM/ TIME0,TFINAL,DELT,IWARN
    COMMON /LUNITS/ LUR,LUW,IFORM,LUK
    double precision xin,out
    real time,t,dtdt,par,tfinal,time0,delt,iwarn
    integer info,icntrl,lur,luk,luw,iform
    dimension xin(1),out(8),t(1),dtdt(1),par(18),info(15)

*  Economic variable declarations
    double precision N_e,i,d,m,N_L,N_D,t_bar,ptax,Down,M_s,V,
   .       i_f,C,C_mod,N_mod,C_other,N_min,N_min_b,R_v,costsav,
   .       maxdif,ror,PW,dPW,LCS_lev,
   .       P_1,P_2,LCS,P_1_ror,P_2_ror,dP_1_d,dP_2_d,PWF,dPWF_d,
   .       initcost,downpay,taxcred,txcredit
    integer  lc,found,runtime,simtime
************************************************************************
```

```
************************************************************************
      call typeck(1,info,1,18,0)

      simtime = int(time+.001)
      runtime = int(tfinal+.001)

c     read input variable: annual fuel cost savings

      costsav = xin(1)

c     read parameters

      N_e = par(1)        !period of economic analysis
      i = par(2)/100      !general inflation rate
      d = par(3)/100      !market discount rate
      m = par(4)/100      !annual mortgage interest rate
      N_L = par(5)        !term of loan
      N_D = par(6)        !depreciation lifetime
      t_bar = par(7)/100  !effective income tax rate
      ptax = par(8)/100   !property tax rate based on assessed value
      Down = par(9)/100   !ratio of down payment to initial investment
      M_s = par(10)/100   !ratio of year 1 misc. maint. costs to inv.
      V = par(11)/100     !ratio of assessed valuation in year 1 to inv.
      R_v = par(12)/100   !ratio of resale value to initial invest.
      i_f = par(13)/100   !electricity cost inflation rate
      C = par(14)         !1 for income-producing, 0 otherwise
      C_mod = par(15)     !cost of a PV module
      N_mod = par(16)     !number of PV modules used
      C_other = par(17)   !fixed cost of system
      taxcred = par(18)/100 !ratio of time 0 fed./state tax cred. to inv.

      info(6) = 8   !number of outputs
      info(9) = 3   !call routine at the end of each timestep
************************************************************************

************************************************************************

c     at end of annual simulation, determine P1, P2, LCS, etc.

      if (simtime .eq. runtime) then
            N_min = min(N_e,N_L)
            N_min_b = min(N_e,N_D)
            P_1 = (1-C*t_bar)*PWF(N_e,i_f,d)
            P_2 = Down+(1-Down)*PWF(N_min,0.d0,d)/PWF(N_L,0.d0,m)-
     .           t_bar*(1-Down)*(PWF(N_min,m,d)*(m-1/PWF(N_L,0.d0,m))+
     .      PWF(N_min,0.d0,d)/PWF(N_L,0.d0,m))+
     .      M_s*(1-C*t_bar)*PWF(N_e,i,d)+
     .      ptax*V*(1-t_bar)*PWF(N_e,i,d)-
     .      R_v*(1-C*t_bar)/(1+d)**N_e
            if (N_min_b.gt.0.d0) then
             P_2 = P_2-C*t_bar*PWF(N_min_b,0.d0,d)/N_D
            endif

        initcost = N_mod*C_mod+C_other
```

```
      downpay = Down*initcost
            txcredit = initcost*taxcred

           LCS = txcredit+P_1*costsav-P_2*(C_mod*N_mod+C_other)

           LCS_lev = LCS/PWF(N_e,0.d0,d)


c     calculate the Rate of Return (Return on Investment)

      if (lcs.gt.0) then
            maxdif = 0.0001
            lc = 0        !intitialize loop counter
            found = 0     !if found = 1 -> the solution has converged
            ror = d       !initial guess value for rate of return
10      lc = lc+1

           P_1_ror = (1-C*t_bar)*PWF(N_e,i_f,ror)
           P_2_ror = Down+(1-Down)*PWF(N_min,0.d0,ror)/PWF(N_L,0.d0,m)-
     .               t_bar*(1-Down)*(PWF(N_min,m,ror)*
     .          (m-1/PWF(N_L,0.d0,m))+
     .          PWF(N_min,0.d0,ror)/PWF(N_L,0.d0,m))+
     .          M_s*(1-C*t_bar)*PWF(N_e,i,ror)+
     .          ptax*V*(1-t_bar)*PWF(N_e,i,ror)-
     .          R_v*(1-C*t_bar)/(1+ror)**N_e
       if (N_min_b.gt.0.d0) then
        P_2_ror = P_2_ror-C*t_bar*PWF(N_min_b,0.d0,ror)/N_D
       endif

       PW = txcredit+P_1_ror*costsav-P_2_ror*(C_mod*N_mod+C_other)

           if (abs(PW) .lt. maxdif) then
            found = 1
       else
            dP_1_d = (1-C*t_bar)*dPWF_d(N_e,i_f,ror)
            dP_2_d = (1-Down)*dPWF_d(N_min,0.d0,ror)/PWF(N_L,0.d0,m)-
     .               t_bar*(1-Down)*(dPWF_d(N_min,m,ror)*
     .          (m-1/PWF(N_L,0.d0,m))+
     .          dPWF_d(N_min,0.d0,ror)/PWF(N_L,0.d0,m))+
     .          M_s*(1-C*t_bar)*dPWF_d(N_e,i,ror)+
     .          ptax*V*(1-t_bar)*dPWF_d(N_e,i,ror)+
     .          N_e*R_v*(1-C*t_bar)/(1+ror)**(2*N_e)
       if (N_min_b.gt.0.d0) then
        dP_2_d = dP_2_d-C*t_bar*dPWF_d(N_min_b,0.d0,ror)/N_D
       endif
       dPW = dP_1_d*costsav-dP_2_d*(C_mod*N_mod+C_other)
            ror = -PW/dPW + ror
            if (lc .eq. 10000) then
        WRITE(LUW,*) 'ROR has not converged'
        goto 20
            endif
       endif
            if (found .eq. 0) goto 10
      else
       ror = 0.d0
```

```
      endif
*************************************************************************

 20     continue

c     set outputs

      out(1) = initcost
            out(2) = txcredit
      out(3) = downpay
            out(4) = P_1
            out(5) = P_2
            out(6) = LCS
            out(7) = LCS_lev
            out(8) = ror*100

     endif

     return 1
     end
*************************************************************************

*************************************************************************
     function PWF(N,i,d)
     double precision i,d,N,PWF

     if (i.eq.d) then
            PWF = N/(1+i)
     else
            PWF = (1-((1+i)/(1+d))**N)/(d-i)
     endif

     return
     end
*************************************************************************

*************************************************************************
     function dPWF_d(N,i,d)
     double precision i,d,N,dPWF_d,PWF

     if (i.eq.d) then
            dPWF_d = -N/(d+1)**2
     else
            dPWF_d = (N/(1+d)*((1+i)/(1+d))**N-PWF(N,i,d))/(d-i)
     endif

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

B.4 System Performance Summary Generator{ TC "B.4 System

```
      subroutine trnout

      implicit none

      real loadtot,cpi,solar,elecaux,elecsav,solfract,draw,actpow,
     .    maxpow,initcost,taxcred,downpay,p_1,p_2,lcs,lcslev,ror,
     .    pveffic,time,loadpre,temp,hotpre,hotaux,electot,pvfract,
     .    costsav,runout,filenum
      integer i
      dimension loadtot(13),cpi(13),solar(13),elecaux(13),elecsav(13),
     .         solfract(13),draw(13),actpow(13),maxpow(13),pveffic(13),
     .         loadpre(13),hotpre(13),hotaux(13),electot(13),
     .         pvfract(13),costsav(13),runout(13)

c     open simulation output data files

      open(unit=36,file='c:\pvsdhw\decks\pvsdhw1.dat',
     .status='unknown')
      open(unit=37,file='c:\pvsdhw\decks\pvsdhw2.dat',
     .status='unknown')
      open(unit=38,file='c:\pvsdhw\decks\pvsdhw3.dat',
     .status='unknown')
      open(unit=39,file='c:\pvsdhw\decks\pvsdhw4.dat',
     .status='unknown')

c     read economic results and system type

      read(39,*)
      read(39,*) time,initcost,taxcred,downpay,p_1,p_2,lcs,lcslev,ror,
     .filenum

c     determine which output file to write to based on system type

            if (filenum.eq.1) then
                  open(unit=35,file='c:\pvsdhw\decks\pv2t.out',
     .position='append',status='unknown')
            endif

            if (filenum.eq.2) then
                  open(unit=35,file='c:\pvsdhw\decks\pv1t_a.out',
     .position='append',status='unknown')
            endif

            if (filenum.eq.3) then
                  open(unit=35,file='c:\pvsdhw\decks\pv1t_b.out',
     .position='append',status='unknown')
            endif

            if (filenum.eq.4) then
                  open(unit=35,file='c:\pvsdhw\decks\thm2t.out',
     .position='append',status='unknown')
            endif
```

```
      if (filenum.eq.5) then
            open(unit=35,file='c:\pvsdhw\decks\thm1t.out',
.position='append',status='unknown')
      endif

c   initialize yearly total array positions

      loadtot(13) = 0
      loadpre(13) = 0
      draw(13) = 0
      solfract(13) = 0
      pvfract(13) = 0
      elecaux(13) = 0
      electot(13) = 0
      elecsav(13) = 0
      costsav(13) = 0
      solar(13) = 0
      actpow(13) = 0
      maxpow(13) = 0
      cpi(13) = 0
      pveffic(13) = 0
      hotpre(13) = 0
      hotaux(13) = 0
      runout(13) = 0

c   read loads, draw, electricity use, solar available, etc.

      read(36,*)
      read(36,*)
      do i = 1,12
            read(36,*) time,loadtot(i),loadpre(i),draw(i),
.            elecaux(i),elecsav(i),solar(i),costsav(i)
        solfract(i) = loadpre(i)/(loadtot(i)+1e-10)
            loadtot(13) = loadtot(13) + loadtot(i)
        loadpre(13) = loadpre(13) + loadpre(i)
        draw(13) = draw(13) + draw(i)
        elecaux(13) = elecaux(13) + elecaux(i)
        elecsav(13) = elecsav(13) + elecsav(i)
        solar(13) = solar(13) + solar(i)
            costsav(13) = costsav(13) + costsav(i)
      enddo

c   read actual and maximum possible PV energy output

      read(37,*)
      read(37,*)
      do i = 1,12
            read(37,*) time,actpow(i),maxpow(i)
            cpi(i) = actpow(i)/(maxpow(i)+1e-10)
        pveffic(i) = (actpow(i)*3.6)/(solar(i)+1e-10)
            actpow(13) = actpow(13) + actpow(i)
        maxpow(13) = maxpow(13) + maxpow(i)
      enddo

      solfract(13) = loadpre(13)/(loadtot(13)+1e-10)
```

```
      cpi(13) = actpow(13)/(maxpow(13)+1e-10)
      pveffic(13) = (maxpow(13)*3.6)/(solar(13)+1e-10)

c     read scalding hours and runout hours

      do i = 1,12
        read(38,*) temp,hotpre(i)
        hotpre(13) = hotpre(13) + hotpre(i)
        read(38,*) temp,hotaux(i)
        hotaux(13) = hotaux(13) + hotaux(i)
        read(38,*) temp,runout(i)
        runout(13) = runout(13) + runout(i)
      enddo

c     unit conversions and formation of other results

      do i = 1,13
        cpi(i) = cpi(i)*100
        solfract(i) = solfract(i)*100
        pveffic(i) = pveffic(i)*100
        elecaux(i) = elecaux(i)*2.777777e-4
        elecsav(i) = elecsav(i)*2.777777e-4
              loadtot(i) = loadtot(i)*2.777777e-4
        actpow(i) = actpow(i)*1e-3
        maxpow(i) = maxpow(i)*1e-3
              electot(i) = elecaux(i) + actpow(i)
        pvfract(i) = (elecsav(i)/(elecsav(i)+elecaux(i)+1e-10))*100
      enddo

c     write output to file

      write(35,*) '************************************************'
      write(35,*)
      write(35,*) '       Solar Domestic Hot Water System'
      write(35,*) '      Overall System Performance Summary'
      write(35,*) '_________________________________________________'
      write(35,*)
      write(35,414)
      write(35,415)
      write(35,416)
      write(35,401) draw(1),loadtot(1),electot(1),elecaux(1),
     .           elecsav(1),costsav(1)
      write(35,402) draw(2),loadtot(2),electot(2),elecaux(2),
     .           elecsav(2),costsav(2)
      write(35,403) draw(3),loadtot(3),electot(3),elecaux(3),
     .           elecsav(3),costsav(3)
      write(35,404) draw(4),loadtot(4),electot(4),elecaux(4),
     .           elecsav(4),costsav(4)
      write(35,405) draw(5),loadtot(5),electot(5),elecaux(5),
     .           elecsav(5),costsav(5)
      write(35,406) draw(6),loadtot(6),electot(6),elecaux(6),
     .           elecsav(6),costsav(6)
      write(35,407) draw(7),loadtot(7),electot(7),elecaux(7),
     .           elecsav(7),costsav(7)
      write(35,408) draw(8),loadtot(8),electot(8),elecaux(8),
```

```
.           elecsav(8),costsav(8)
 write(35,409) draw(9),loadtot(9),electot(9),elecaux(9),
.           elecsav(9),costsav(9)
 write(35,435) draw(10),loadtot(10),electot(10),elecaux(10),
.           elecsav(10),costsav(10)
 write(35,411) draw(11),loadtot(11),electot(11),elecaux(11),
.           elecsav(11),costsav(11)
 write(35,412) draw(12),loadtot(12),electot(12),elecaux(12),
.           elecsav(12),costsav(12)
 write(35,*)
 write(35,413) draw(13),loadtot(13),electot(13),elecaux(13),
.           elecsav(13),costsav(13)
 write(35,*)
 write(35,*)
 write(35,*)
 write(35,*)

 write(35,*)
 write(35,*) '     Collector or PV Array/Controller'
 write(35,*) '         Performance Summary'
 write(35,*) '_____________________________________________________'
 write(35,*)
 write(35,514)
 write(35,515)
 write(35,516)
 write(35,501) actpow(1),pvfract(1),cpi(1),pveffic(1)
 write(35,502) actpow(2),pvfract(2),cpi(2),pveffic(2)
 write(35,503) actpow(3),pvfract(3),cpi(3),pveffic(3)
 write(35,504) actpow(4),pvfract(4),cpi(4),pveffic(4)
 write(35,505) actpow(5),pvfract(5),cpi(5),pveffic(5)
 write(35,506) actpow(6),pvfract(6),cpi(6),pveffic(6)
 write(35,507) actpow(7),pvfract(7),cpi(7),pveffic(7)
 write(35,508) actpow(8),pvfract(8),cpi(8),pveffic(8)
 write(35,509) actpow(9),pvfract(9),cpi(9),pveffic(9)
 write(35,535) actpow(10),pvfract(10),cpi(10),pveffic(10)
 write(35,511) actpow(11),pvfract(11),cpi(11),pveffic(11)
 write(35,512) actpow(12),pvfract(12),cpi(12),pveffic(12)
 write(35,*)
 write(35,513) actpow(13),pvfract(13),cpi(13),pveffic(13)
 write(35,*)
 write(35,*)
 write(35,*)
 write(35,*)

 write(35,*)
 write(35,*) '     Number of Hours of Possible Scalding'
 write(35,*) '           or Hot Water Runout'
 write(35,*) '_____________________________________________________'
 write(35,*)
 write(35,214)
 write(35,215)
 write(35,216)
 write(35,201) hotpre(1),hotaux(1),runout(1)
 write(35,202) hotpre(2),hotaux(2),runout(2)
 write(35,203) hotpre(3),hotaux(3),runout(3)
```

```
      write(35,204) hotpre(4),hotaux(4),runout(4)
      write(35,205) hotpre(5),hotaux(5),runout(5)
      write(35,206) hotpre(6),hotaux(6),runout(6)
      write(35,207) hotpre(7),hotaux(7),runout(7)
      write(35,208) hotpre(8),hotaux(8),runout(8)
      write(35,209) hotpre(9),hotaux(9),runout(9)
      write(35,235) hotpre(10),hotaux(10),runout(10)
      write(35,211) hotpre(11),hotaux(11),runout(11)
      write(35,212) hotpre(12),hotaux(12),runout(12)
      write(35,*)
      write(35,213) hotpre(13),hotaux(13),runout(13)
      write(35,*)
      write(35,*)
      write(35,*)
      write(35,*)

      write(35,*)
      write(35,*) '      Life Cycle Economics Results'
      write(35,*) '_____'
      write(35,*)
      write(35,300) initcost
      write(35,301) taxcred
      write(35,302) downpay
      write(35,303) lcs
      write(35,304) lcslev
      write(35,305) ror
      write(35,306) p_1,p_2
      write(35,*)
      write(35,*)
      write(35,*)
      write(35,*) '**************************************************'

  201  FORMAT(T2,'Jan',T15,f11.0,T30,f11.0,T45,f11.0)
  202  FORMAT(T2,'Feb',T15,f11.0,T30,f11.0,T45,f11.0)
  203  FORMAT(T2,'Mar',T15,f11.0,T30,f11.0,T45,f11.0)
  204  FORMAT(T2,'Apr',T15,f11.0,T30,f11.0,T45,f11.0)
  205  FORMAT(T2,'May',T15,f11.0,T30,f11.0,T45,f11.0)
  206  FORMAT(T2,'Jun',T15,f11.0,T30,f11.0,T45,f11.0)
  207  FORMAT(T2,'Jul',T15,f11.0,T30,f11.0,T45,f11.0)
  208  FORMAT(T2,'Aug',T15,f11.0,T30,f11.0,T45,f11.0)
  209  FORMAT(T2,'Sep',T15,f11.0,T30,f11.0,T45,f11.0)
  235  FORMAT(T2,'Oct',T15,f11.0,T30,f11.0,T45,f11.0)
  211  FORMAT(T2,'Nov',T15,f11.0,T30,f11.0,T45,f11.0)
  212  FORMAT(T2,'Dec',T15,f11.0,T30,f11.0,T45,f11.0)
  213  FORMAT(T2,'Year',T15,f11.0,T30,f11.0,T45,f11.0)

  214  FORMAT(T2,'Period',T20,'Pre. Tank',T35,'Aux. Tank',T50,'System')
  215  FORMAT(T20,'Scalding',T35,'Scalding',T50,'Runout')
  216  FORMAT(T20,'(hours)',T35,'(hours)',T50,'(hours)')


  300  FORMAT(T2,'Initial Cost of System              $',F9.2)
  301  FORMAT(T2,'Tax Credit upon Installation        $',F9.2)
  302  FORMAT(T2,'Down Payment                        $',F9.2)
  303  FORMAT(T2,'Life Cycle Savings                  $',F9.2)
```

```
304  FORMAT(T2,'Levelized Life Cycle Savings          $',F9.2)
305  FORMAT(T2,'Rate of Return on Investment           ',F4.1,'%')
306  FORMAT(T7,'(P1 = ',f5.2,' and P2 = ',f4.2,')')

401  FORMAT(T2,'Jan',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
402  FORMAT(T2,'Feb',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
403  FORMAT(T2,'Mar',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
404  FORMAT(T2,'Apr',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
405  FORMAT(T2,'May',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
406  FORMAT(T2,'Jun',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
407  FORMAT(T2,'Jul',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
408  FORMAT(T2,'Aug',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
409  FORMAT(T2,'Sep',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
435  FORMAT(T2,'Oct',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
411  FORMAT(T2,'Nov',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
412  FORMAT(T2,'Dec',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)
413  FORMAT(T2,'Year',T15,f11.0,T30,f11.0,T45,f11.0,T60,f11.0,T75,
   .f11.0,T90,f11.2)

414  FORMAT(T2,'Period',T20,'Hot Water',T35,'System',T50,'Total Energy',
   .T65,'Auxiliary',T80,'Elec. Energy',T95,'Elec. Cost')
415  FORMAT(T20,'Draw',T35,'Load',T50,'Used',
   .T65,'Elec. Energy',T80,'Savings',T95,'Savings')
416  FORMAT(T20,'(L)',T35,'(kWh)',T50,'(kWh)',
   .T65,'(kWh)',T80,'(kWh)',T95,'($)')

501  FORMAT(T2,'Jan',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
502  FORMAT(T2,'Feb',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
503  FORMAT(T2,'Mar',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
504  FORMAT(T2,'Apr',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
505  FORMAT(T2,'May',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
506  FORMAT(T2,'Jun',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
507  FORMAT(T2,'Jul',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
508  FORMAT(T2,'Aug',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
509  FORMAT(T2,'Sep',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
535  FORMAT(T2,'Oct',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
511  FORMAT(T2,'Nov',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
512  FORMAT(T2,'Dec',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)
513  FORMAT(T2,'Year',T15,f11.0,T30,f11.1,T45,f11.1,T60,f11.1)

514  FORMAT(T2,'Period',T20,'Useful',
   .T35,'Solar Energy',T50,'CPI',T65,'Solar Device')
515  FORMAT(T20,'Solar Energy',
```

```
     .T35,'Fraction',T65,'Efficiency')
 516  FORMAT(T20,'(kWh)',
     .T35,'(%)',T50,'(%)',T65,'(%)')

      close(35)
      close(36)
      close(37)
      close(38)
      close(39)

      return
      end
```