```fortran
Subroutine Type9367
!-----------------------------------------------------------------------------------------------------------------------
! This subroutine models a refrigerator. This version of the code has several changes compared with the original Type936 to accomodate new things such as PCM heat storage.
!-----------------------------------------------------------------------------------------------------------------------
! Copyright � 2011 Thermal Energy System Specialists, LLC. All rights reserved. 2024 Daniel Lemos Marques, University of Aveiro, DEM, TEMA.

!export this subroutine for its use in external DLLs.
!DEC$ATTRIBUTES DLLEXPORT :: TYPE9367

!Use Statements
Use TrnsysConstants
Use TrnsysFunctions

!Variable Declarations
Implicit None !force explicit declaration of local variables
Double Precision Time,Timestep
Integer CurrentUnit,CurrentType,j,ninp,jj,nparua,nparmcp,simulation_mode
Double Precision aa,bb,capacitance,u_value,Ti,Tf,Tave,area,Q_skin,Q_stored,fvol_fridge,fvol_freezer,Tset_fridge, &
                 Tset_freezer,deadbandup,deadbanddown,cap_rated,Power_rated,capacity,COP,P_condfan,P_evapfan,Power,Q_rejected,Q_cond, &
        T_zone,control_now,control_last,T_control,T_evap,x(2),y(3),delt_now,delt_tot,Ti_now,Tave_tot,x_tot, &
                 Q_evap,P_comp,control_prev,mass,specific_heat,UA,Tzero,Q_PCM,RPM,UAref_max,Mref_max,Mref,Q_evap_theory,Power_theory,COP_datasheet,COP_datasheet_1,COP_carnot_1,COP_carnot_2
Integer n_temps_evap,lu_data,n_levels_rpm,nx(2)
Logical found_end
Logical InitializationDone
Data InitializationDone /.false./

!Get the Global Trnsys Simulation Variables
 Time=getSimulationTime()
 Timestep=getSimulationTimeStep()
 CurrentUnit = getCurrentUnit()
 CurrentType = getCurrentType()

!Set the Version Number for This Type
 If (getIsVersionSigningTime()) Then
     Call SetTypeVersion(17)
     Return
 Endif
!-----------------------------------------------------------------------------------------------------------------------
```

```fortran

!--------------------------------------------------------------------------------------------------------------------
!Do All of the Last Call Manipulations Here
 If (getIsLastCallofSimulation()) Then
    Return
 Endif

!--------------------------------------------------------------------------------------------------------------------
!Perform Any "After Convergence" Manipulations That May Be Required
 If (getIsEndOfTimestep()) Then
    Call SetStaticArrayValue(1,getStaticArrayValue(2))
    Call SetStaticArrayValue(3,getStaticArrayValue(4))
    Return
 Endif

!--------------------------------------------------------------------------------------------------------------------
!Do All of the "Very First Call of the Simulation Manipulations" Here
 If (getIsFirstCallofSimulation()) Then

    !Open(19, File="debug_freezer.txt", status='replace')
    !Write(19, *) "Entered Type936_Freezer Subroutine"
    !Close(19)

    ninp = getNumberOfInputs() !set the number of INPUTS to the number found in the deck
    nparua = getParameterValue(1) !get the number of pairs the user wants of U*A

!  Tell the TRNSYS Engine How This Type Works
    Call SetNumberofParameters(16)
    Call SetNumberofInputs(ninp)
    Call SetNumberofDerivatives(0)
    Call SetNumberofOutputs(9)
    Call SetIterationMode(1)
    Call SetNumberStoredVariables(4,0)

!  Set the Correct Input and Output Variable Types
    Call SetInputUnits(1,'TE1')
    Call SetInputUnits(2,'PW1')
    Do jj=5,(nparua*2+2),2
    Call SetInputUnits(jj-1,'HT1')
    Call SetInputUnits(jj,'AR1')
    EndDo
    Do j=(nparua*2+4),ninp-3,2
    Call SetInputUnits(j-1,'CP1')
    Call SetInputUnits(j,'MA1')
    EndDo
    Call SetInputUnits(ninp-2,'DM1')
    Call SetInputUnits(ninp-1,'PW1')
    Call SetInputUnits(ninp,'PW1')
```

```fortran

     Call SetOutputUnits(1,'TE1')
     Call SetOutputUnits(2,'PW1')
     Call SetOutputUnits(3,'PW1')
     Call SetOutputUnits(4,'PW1')
     Call SetOutputUnits(5,'DM1')
     Call SetOutputUnits(6,'PW1')
     Call SetOutputUnits(7,'DM1')
     Call SetOutputUnits(8,'DM1')
     Call SetOutputUnits(9,'DM1')

     Return
  EndIf

!-----------------------------------------------------------------------------------------------------------------
!Do All of the First Timestep Manipulations Here - There Are No Iterations at the Intial Time
  If (getIsStartTime()) Then

      If (.not. InitializationDone) Then
         InitializationDone = .true.

! Read in the Values of the Parameters from the Input File
!     capacitance = getParameterValue(1)
!     area = getParameterValue(1)
     nparua = getParameterValue(1) !get the number of pairs the user wants of U*A
     fvol_fridge = getParameterValue(2)
     fvol_freezer = 1.-getParameterValue(2)
     Tset_fridge = getParameterValue(3)
     Tset_freezer = getParameterValue(4)
     deadbandup = getParameterValue(5)
     deadbanddown = getParameterValue(6)
     lu_data = JFIX(getParameterValue(7)+0.5)
     n_levels_rpm = JFIX(getParameterValue(8)+0.5)
     n_temps_evap = JFIX(getParameterValue(9)+0.5)
     Tzero = getParameterValue(10)
     Power_rated = getParameterValue(11)
     P_condfan = getParameterValue(12)
     P_evapfan = getParameterValue(13)
     nparmcp = getParameterValue(14)
     cap_rated = getParameterValue(15)
     simulation_mode = getParameterValue(16)

!  Check the Parameters for Problems
!     If (capacitance <= 0.) Call FoundBadParameter(1,'Fatal','The thermal capacitance must be greater than 0.')
!     If (area < 0.) Call FoundBadParameter(1,'Fatal','The surface area cannot be negative.')
     If (nparua < 0.) Call FoundBadParameter(1,'Fatal','The user has to specify at least one pair of surface area and one U value.')
     If (fvol_fridge < 0.) Call FoundBadParameter(2,'Fatal','The volume fraction
```

```
for the refrigerator cannot be negative.')
    If (fvol_fridge > 1.) Call FoundBadParameter(2,'Fatal','The volume fraction for the refrigerator cannot be greater than 1.')
    If (deadbandup < 0.) Call FoundBadParameter(5,'Fatal','The temperature deadband for control cannot be negative.')
    If (deadbanddown < 0.) Call FoundBadParameter(5,'Fatal','The temperature deadband for control cannot be negative.')
    If (lu_data < 10) Call FoundBadParameter(6,'Fatal','The logical unit number for the file with the refrigerator performance data cannot be less than 10.')
    If (n_levels_rpm < 1) Call FoundBadParameter(7,'Fatal','The number of unique zone temperatures for which there is performance data cannot be less than 1.')
    If (n_temps_evap < 1) Call FoundBadParameter(8,'Fatal','The number of unique evaporator temperatures for which there is performance data cannot be less than 1.')
    If (Tzero <= -253) Call FoundBadParameter(9,'Fatal','The initial temperature is below the absolute 0.')
    If (Power_rated <= 0.) Call FoundBadParameter(10,'Fatal','The rated COP must be greater than 0.')
    If (P_condfan < 0.) Call FoundBadParameter(11,'Fatal','The condenser fan power cannot be negative.')
    If (P_evapfan < 0.) Call FoundBadParameter(12,'Fatal','The evaporator fan power cannot be negative.')
    If (nparmcp < 0.) Call FoundBadParameter(13,'Fatal','The user has to specify at least one pair of mass and specific heat.')
    If (cap_rated < 0.) Call FoundBadParameter(14,'Fatal','The user has to specify a cap_rated of at least zero.')
    If (simulation_mode < 0.) Call FoundBadParameter(15,'Fatal','The user has to specify a simulation_mode of0 or an integer positive value.')

! Set the outputs to initial values.
    Call SetOutputValue(1,Tzero)
    Call SetOutputValue(2,0.d0)
    Call SetOutputValue(3,0.d0)
    Call SetOutputValue(4,0.0d0)
    Call SetOutputValue(5,0.d0)
    Call SetOutputValue(6,0.0d0)
    Call SetOutputValue(7,1.d0)
    Call SetOutputValue(8,2500.0d0)
    Call SetOutputValue(9,0.d0)

!  Set the initial storage variables
    Call SetStaticArrayValue(1,Tzero)   !by doing this for t=0, the T_i = Tzero defined by the user
    Call SetStaticArrayValue(2,Tzero)   !by doing this for t=0, the T_f = Tzero defined by the user
    Call SetStaticArrayValue(3,0.d0)    !by doing this for t=0, the compressor last mode was the OFF mode.
    Call SetStaticArrayValue(4,0.d0)

    Endif
```

```

    Return
 EndIf
!-------------------------------------------------------------------------------------------------------------------

!-------------------------------------------------------------------------------------------------------------------
!ReRead the Parameters if Another Unit of This Type Has Been Called Last
 If (getIsReReadParameters()) Then
 !   capacitance = getParameterValue(1)
 !   area = getParameterValue(1)
    nparua = getParameterValue(1) !get the number of pairs the user wants of U*A
    fvol_fridge = getParameterValue(2)
    fvol_freezer = 1.-getParameterValue(2)
    Tset_fridge = getParameterValue(3)
    Tset_freezer = getParameterValue(4)
    deadbandup = getParameterValue(5)
    deadbanddown = getParameterValue(6)
    lu_data = JFIX(getParameterValue(7)+0.5)
    n_levels_rpm = JFIX(getParameterValue(8)+0.5)
    n_temps_evap = JFIX(getParameterValue(9)+0.5)
    Tzero = getParameterValue(10)
    Power_rated = getParameterValue(11)
    P_condfan = getParameterValue(12)
    P_evapfan = getParameterValue(13)
    nparmcp = getParameterValue(14)
    cap_rated = getParameterValue(15)
    simulation_mode = getParameterValue(16)
 EndIf
!-------------------------------------------------------------------------------------------------------------------

!-------------------------------------------------------------------------------------------------------------------
!Get the Current Inputs to the Model
 T_zone = getInputValue(1)
 Q_PCM = getInputValue(2)
 RPM = getInputValue(ninp-2)
 Q_evap_theory = getInputValue(ninp-1)
 Power_theory = getInputValue(ninp)

 jj=4 !initialize jj again
 UA=0. !initialize UA=0

 Do While (jj<=(nparua*2+2))
 u_value = getInputValue(jj-1)
 area = getInputValue(jj)
     If (u_value <= 0.) Call FoundBadInput(jj-1,'Fatal','The Heat Transfer Coefficient must be greater than 0.')
     If (area <= 0.) Call FoundBadInput(jj,'Fatal','The area must be greater than 0.')
```

```
     UA=UA+(u_value*area) ! Calculate the global value of UA through the sum of u_values*areas (input values)
     jj=jj+2
    If (ErrorFound()) Return
 EndDo

 j=(nparua*2+4) !initialize j again
 capacitance=0. !initialize capacitance=0

 Do While (j<=ninp-3)
    specific_heat = getInputValue(j-1)
    mass = getInputValue(j)
     If (specific_heat <= 0.) Call FoundBadInput(j-1,'Fatal','The thermal specific_heat must be greater than 0.')
     If (mass <= 0.) Call FoundBadInput(j,'Fatal','The mass must be greater than 0.')
    capacitance=capacitance+(mass*specific_heat) ! Calculate the capacitance through the sum of masses*specific heats (input values)
    j=j+2
    If (ErrorFound()) Return
 EndDo

 !If (u_value < 0.) Call FoundBadInput(2,'Fatal','The heat transfer coefficient is negative.')
 !If (specific_heat <= 0.) Call FoundBadInput(3,'Fatal','The thermal specific_heat must be greater than 0.')
 !If (mass <= 0.) Call FoundBadInput(4,'Fatal','The mass must be greater than 0.')
 If (ErrorFound()) Return
!-----------------------------------------------------------------------------------------------------------------------

!-----------------------------------------------------------------------------------------------------------------------
!Retrieve the Values from Storage
 Ti = getStaticArrayValue(1)
 Tf = getStaticArrayValue(2)
 control_last = getStaticArrayValue(3)
!-----------------------------------------------------------------------------------------------------------------------

! Calculate the average temperature setting for the device
 T_control = fvol_freezer*Tset_freezer+fvol_fridge*Tset_fridge
 If (fvol_freezer > 0.) Then
    !T_evap = Tset_freezer
    T_evap = Ti
 Else
    T_evap = Tset_fridge
 EndIf

! Get the performance of the device at the current conditions
 nx(1) = n_temps_evap
```

```
 nx(2) = n_levels_rpm
 x(1) = T_evap
 x(2) = RPM
 Call InterpolateData(lu_data,2,nx,3,x,y)
 If (ErrorFound()) Return
 !capacity = UAref_max*Mref/Mref_max*(Ti-T_evap)
 !capacity = cap_rated*y(1)*3.6 !3.6 is used to convert the value of Qevap that comes from the data sheet in W to kJ/hr
 capacity = cap_rated*Q_evap_theory !to import the value coming from EES.

! Set the new control signal to the value at the end of the timestep
 If (simulation_mode == 0) Then
    control_now = 0.
    deadbandup = 300
 Else If (simulation_mode == 1) Then
     If (Ti >= (T_control+deadbandup)) Then
        control_now = 1.
    Else If (Ti <= (T_control-deadbanddown)) Then
        control_now = 0.
    Else
        control_now = control_last
    EndIf
 Else
  Call FoundBadParameter(99,'Fatal','Invalid simulation mode. Must be 0 or 1.')
!  STOP
 EndIf

  !Open(19, File="debug_freezer.txt", position='append')
  !Write(19, *) "at time", Time
  !Write(19, *) "data_Tfreezer=", Ti, "Tevap=",T_evap
  !Close(19)

 delt_now = Timestep
 Ti_now = Ti
 found_end = .false.
 Tave_tot = 0.
 x_tot = 0.
 delt_tot = 0.

! The fridge is running
 30 If (control_now > 0.5) Then
! Run for the full timestep and see what happens
    control_prev=1.
! Set up the governing differential equation in the form dT/dt=aT+b
    bb = (-Q_PCM-capacity+UA*T_zone)/capacitance ! previously was bb = (-capacity+u_value*area*T_zone)/capacitance
    aa = -UA/capacitance ! previously was aa = -u_value*area/capacitance
! Solve the diffeq analytically
    If (aa == 0.) Then
       Tf = Ti_now+bb*delt_now
       Tave = Ti_now+bb*delt_now/2.
    Else
```

```
       Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
       Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
    EndIf
! Check the resultant temperature
    If (Tf >= (T_control-deadbanddown)) Then
       delt_now = delt_now
       control_now = 1.
       found_end = .true.
    Else
! Calculate the time to get to the setpoint
       Tf = T_control-deadbanddown
       If (aa == 0.) Then
          delt_now = DMIN1(delt_now,((Tf-Ti_now)/bb))
          Tf = Ti_now+bb*delt_now
          Tave = Ti_now+bb*delt_now/2.
       Else
          delt_now = DMIN1(delt_now,(DLOG((Tf+bb/aa)/(Ti_now+bb/aa))/aa))
          Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
          Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
       EndIf
       control_now = 0.
    EndIf
 Else
! Set up the governing differential equation in the form dT/dt=aT+b
    bb = (-Q_PCM+UA*T_zone)/capacitance ! previously was  bb = (u_value*area*T_zone)/capacitance
    aa = -UA/capacitance ! previously was aa = -u_value*area/capacitance
    control_prev = 0.
! Solve the diffeq analytically
    If (aa == 0.) Then
       Tf = Ti_now+bb*delt_now
       Tave = Ti_now+bb*delt_now/2.
    Else
       Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
       Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
    EndIf
! Check the resultant temperature
    If (Tf <= (T_control+deadbandup)) Then
       delt_now = delt_now
       control_now = 0.
       found_end = .true.
    Else
! Calculate the time to get to the setpoint
       Tf = T_control+deadbandup
       If (aa == 0.) Then
          delt_now = DMIN1(delt_now,((Tf-Ti_now)/bb))
          Tf = Ti_now+bb*delt_now
          Tave = Ti_now+bb*delt_now/2.
       Else
          delt_now = DMIN1(delt_now,(DLOG((Tf+bb/aa)/(Ti_now+bb/aa))/aa))
          Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
          Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
```

```
          EndIf
          control_now = 1.
       EndIf
    EndIf
   ! Update the temperatures
    Tave_tot = Tave_tot+Tave*delt_now/Timestep
    Ti_now = Tf
   ! Update the run-time counter
    x_tot = x_tot+control_prev*delt_now/Timestep
   ! Set the remaining time
    delt_tot = delt_tot+delt_now
    delt_now = Timestep-delt_tot
   ! Check to see if we should run again
    If (.not.found_end) Goto 30
   ! Calculate the energy flows from the refrigerator
    Q_skin = UA*(T_zone-Tave_tot) ! previously was Q_skin = u_value*area*(T_zone-Tave_tot)
    Q_stored = capacitance*(Tf-Ti)/Timestep
   ! Calculate the energy removed from the space
    Q_evap = capacity*x_tot
   ! Calculate the power of the compressor
   ! P_comp = y(3)*x_tot*3.6 !3.6 is to convert the value of W that comes from the comrpessor data sheet to kJ/hr
   P_comp = x_tot*Power_theory*Power_rated !to use the value from EES
    ! Previously, P_comp was calculated like this: P_comp = DMAX1(0.,(Q_evap/COP-P_evapfan*x_tot-P_condfan*x_tot))
   ! Calculate the total heat rejection
    Q_cond = Q_evap+P_comp
    Q_rejected = Q_cond+P_condfan*x_tot
    !Calculate the total power considering the power in the compressor and the power in the fans
    Power = P_condfan*x_tot+P_evapfan*x_tot+P_comp

    !Calculating the COP - firts it reads the COP from the data_sheet of the compressor for a given T_evap and 45ºC of condenser temperature.
    COP_datasheet_1 = y(2)*x_tot
    !then it calculates the COP of the reverse Carnot cycle in ideal situations for both condensing temepratures
    COP_carnot_1 = T_evap/(45-T_evap)
    COP_carnot_2 = T_evap/(T_zone-T_evap)
    !then it uses these results to conver the COP from the datasheet to a theoretic COP at the same T_evap but for ambiente temperature as the condensing temperature.
    COP_datasheet = COP_datasheet_1*COP_carnot_2/COP_carnot_1

    !Calculating the final COP of the real refrigeration system
    COP = Q_evap/(Power+0.00001)

   !------------------------------------------------------------------------------------------------------------------------------
   !Set the values in storage
    Call SetStaticArrayValue(1,Ti)
```

```
 Call SetStaticArrayValue(2,Tf)
 Call SetStaticArrayValue(3,control_last)
 Call SetStaticArrayValue(4,control_now)

!--------------------------------------------------------------------------------
  -------------------------------------
! Set outputs
 Call SetOutputValue(1,Tave_tot)
 Call SetOutputValue(2,Q_skin)
 Call SetOutputValue(3,Q_rejected)
 Call SetOutputValue(4,Power)
 Call SetOutputValue(5,x_tot)
 Call SetOutputValue(6,Q_evap)
 Call SetOutputValue(7,COP_datasheet)
 Call SetOutputValue(8,RPM)
 Call SetOutputValue(9,COP)


  !Open(19, File="debug_freezer.txt", position='append')
  !Write(19, *) "at time", Time
  !Write(19, *) "outputs", Tave_tot, Power, x_tot, Q_evap, RPM
  !Close(19)

 Return
End
```