```
Subroutine Type936
!------------------------------------------------------------------------------
! This subroutine models a refrigerator.
!------------------------------------------------------------------------------
! Copyright © 2011 Thermal Energy System Specialists, LLC. All rights reserved.

!export this subroutine for its use in external DLLs.
!DEC$ATTRIBUTES DLLEXPORT :: TYPE936

!Use Statements
Use TrnsysConstants
Use TrnsysFunctions

!Variable Declarations
Implicit None !force explicit declaration of local variables
Double Precision Time,Timestep
Integer CurrentUnit,CurrentType
Double Precision aa,bb,capacitance,u_value,Ti,Tf,Tave,area,Q_skin,Q_stored,fvol_fridge,fvol_freezer,Tset_fridge, &
Tset_freezer,deadband,cap_rated,COP_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(2),delt_now,delt_tot,Ti_now,Tave_tot,x_tot, &
        Q_evap,P_comp,control_prev
Integer n_temps_r,lu_data,n_temps_z,nx(2)
Logical found_end

!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
!------------------------------------------------------------------------------

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

```

```
44  !------------------------------------------------------------------------------
      ----------------------------------------
45  !Perform Any "After Convergence" Manipulations That May Be Required
46   If (getIsEndOfTimestep()) Then
47      Call SetStaticArrayValue(1,getStaticArrayValue(2))
48      Call SetStaticArrayValue(3,getStaticArrayValue(4))
49      Return
50   Endif
51
52  !------------------------------------------------------------------------------
      ----------------------------------------
53  !Do All of the "Very First Call of the Simulation Manipulations" Here
54   If (getIsFirstCallofSimulation()) Then
55
56  !  Tell the TRNSYS Engine How This Type Works
57      Call SetNumberofParameters(13)
58      Call SetNumberofInputs(2)
59      Call SetNumberofDerivatives(0)
60      Call SetNumberofOutputs(5)
61      Call SetIterationMode(1)
62      Call SetNumberStoredVariables(4,0)
63
64  !  Set the Correct Input and Output Variable Types
65      Call SetInputUnits(1,'TE1')
66      Call SetInputUnits(2,'HT1')
67      Call SetOutputUnits(1,'TE1')
68      Call SetOutputUnits(2,'PW1')
69      Call SetOutputUnits(3,'PW1')
70      Call SetOutputUnits(4,'PW1')
71      Call SetOutputUnits(5,'DM1')
72
73      Return
74   EndIf
75
76  !------------------------------------------------------------------------------
      ----------------------------------------
77  !Do All of the First Timestep Manipulations Here - There Are No Iterations at the
       Intial Time
78   If (getIsStartTime()) Then
79
80  ! Read in the Values of the Parameters from the Input File
81      capacitance = getParameterValue(1)
82      area = getParameterValue(2)
83      fvol_fridge = getParameterValue(3)
84      fvol_freezer = 1.-getParameterValue(3)
85      Tset_fridge = getParameterValue(4)
86      Tset_freezer = getParameterValue(5)
87      deadband = getParameterValue(6)
88      lu_data = JFIX(getParameterValue(7)+0.5)
89      n_temps_z = JFIX(getParameterValue(8)+0.5)
90      n_temps_r = JFIX(getParameterValue(9)+0.5)
91      cap_rated = getParameterValue(10)
```

```
 92      COP_rated = getParameterValue(11)
 93      P_condfan = getParameterValue(12)
 94      P_evapfan = getParameterValue(13)
 95
 96  !  Check the Parameters for Problems
 97      If (capacitance <= 0.) Call FoundBadParameter(1,'Fatal','The thermal
           capacitance must be greater than 0.')
 98      If (area < 0.) Call FoundBadParameter(2,'Fatal','The surface area cannot be
           negative.')
 99      If (fvol_fridge < 0.) Call FoundBadParameter(3,'Fatal','The volume fraction
           for the refrigerator cannot be negative.')
100      If (fvol_fridge > 1.) Call FoundBadParameter(3,'Fatal','The volume fraction
           for the refrigerator cannot be greater than 1.')
101      If (deadband < 0.) Call FoundBadParameter(6,'Fatal','The temperature deadband
            for control cannot be negative.')
102      If (lu_data < 10) Call FoundBadParameter(7,'Fatal','The logical unit number
           for the file with the refrigerator performance data cannot be less than
           10.')
103      If (n_temps_z < 1) Call FoundBadParameter(8,'Fatal','The number of unique
           zone temperatures for which there is performance data cannot be less than
           1.')
104      If (n_temps_r < 1) Call FoundBadParameter(9,'Fatal','The number of unique
           evaporator temperatures for which there is performance data cannot be less
           than 1.')
105      If (cap_rated <= 0.) Call FoundBadParameter(10,'Fatal','The rated cooling
           capacity must be greater than 0.')
106      If (COP_rated <= 0.) Call FoundBadParameter(11,'Fatal','The rated COP must be
            greater than 0.')
107      If (P_condfan < 0.) Call FoundBadParameter(12,'Fatal','The condenser fan
           power cannot be negative.')
108      If (P_evapfan < 0.) Call FoundBadParameter(13,'Fatal','The evaporator fan
           power cannot be negative.')
109
110  ! Set the outputs to initial values.
111      Call SetOutputValue(1,fvol_freezer*Tset_freezer+fvol_fridge*Tset_fridge)
112      Call SetOutputValue(2,0.d0)
113      Call SetOutputValue(3,0.d0)
114      Call SetOutputValue(4,0.d0)
115      Call SetOutputValue(5,0.d0)
116
117  !  Set the initial storage variables
118      Call SetStaticArrayValue(1,fvol_freezer*Tset_freezer+fvol_fridge*Tset_fridge)
119      Call SetStaticArrayValue(2,fvol_freezer*Tset_freezer+fvol_fridge*Tset_fridge)
120      Call SetStaticArrayValue(3,0.d0)
121      Call SetStaticArrayValue(4,0.d0)
122
123      Return
124   EndIf
125  !------------------------------------------------------------------------------
       ---------------------------------------
126
127  !------------------------------------------------------------------------------
```

```
      ---------------------------------------
!ReRead the Parameters if Another Unit of This Type Has Been Called Last
 If (getIsReReadParameters()) Then
    capacitance = getParameterValue(1)
    area = getParameterValue(2)
    fvol_fridge = getParameterValue(3)
    fvol_freezer = 1.-getParameterValue(3)
    Tset_fridge = getParameterValue(4)
    Tset_freezer = getParameterValue(5)
    deadband = getParameterValue(6)
    lu_data = JFIX(getParameterValue(7)+0.5)
    n_temps_z = JFIX(getParameterValue(8)+0.5)
    n_temps_r = JFIX(getParameterValue(9)+0.5)
    cap_rated = getParameterValue(10)
    COP_rated = getParameterValue(11)
    P_condfan = getParameterValue(12)
    P_evapfan = getParameterValue(13)
 EndIf
!---------------------------------------------------------------------------------------
  ---------------------------------------

!---------------------------------------------------------------------------------------
  ---------------------------------------
!Get the Current Inputs to the Model
 T_zone = getInputValue(1)
 u_value = getInputValue(2)

 If (u_value < 0.) Call FoundBadInput(2,'Fatal','The heat transfer coefficient is
    negative.')
 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
 Else
    T_evap = Tset_fridge
 EndIf
! Get the performance of the device at the current conditions
 nx(1) = n_temps_r
 nx(2) = n_temps_z
```

```
173   x(1) = T_evap
174   x(2) = T_zone
175   Call InterpolateData(lu_data,2,nx,2,x,y)
176   If (ErrorFound()) Return
177   capacity = cap_rated*y(1)
178   COP = COP_rated*y(2)
179  ! Set the new control signal to the value at the end of the timestep
180   If (Ti >= (T_control+deadband/2.)) Then
181      control_now = 1.
182   ElseIf (Ti <= (T_control-deadband/2.)) Then
183      control_now = 0.
184   Else
185      control_now = control_last
186   EndIf
187   delt_now = Timestep
188   Ti_now = Ti
189   found_end = .false.
190   Tave_tot = 0.
191   x_tot = 0.
192   delt_tot = 0.
193
194  ! The fridge is running
195   30 If (control_now > 0.5) Then
196  ! Run for the full timestep and see what happens
197      control_prev=1.
198  ! Set up the governing differential equation in the form dT/dt=aT+b
199      bb = (-capacity+u_value*area*T_zone)/capacitance
200      aa = -u_value*area/capacitance
201  ! Solve the diffeq analytically
202      If (aa == 0.) Then
203         Tf = Ti_now+bb*delt_now
204         Tave = Ti_now+bb*delt_now/2.
205      Else
206         Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
207         Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
208      EndIf
209  ! Check the resultant temperature
210      If (Tf >= (T_control-deadband/2.)) Then
211         delt_now = delt_now
212         control_now = 1.
213         found_end = .true.
214      Else
215  ! Calculate the time to get to the setpoint
216         Tf = T_control-deadband/2.
217         If (aa == 0.) Then
218            delt_now = DMIN1(delt_now,((Tf-Ti_now)/bb))
219            Tf = Ti_now+bb*delt_now
220            Tave = Ti_now+bb*delt_now/2.
221         Else
222            delt_now = DMIN1(delt_now,(DLOG((Tf+bb/aa)/(Ti_now+bb/aa))/aa))
223            Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
224            Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
```

```
225         EndIf
226         control_now = 0.
227      EndIf
228   Else
229  ! Set up the governing differential equation in the form dT/dt=aT+b
230      bb = (u_value*area*T_zone)/capacitance
231      aa = -u_value*area/capacitance
232      control_prev = 0.
233  ! Solve the diffeq analytically
234      If (aa == 0.) Then
235         Tf = Ti_now+bb*delt_now
236         Tave = Ti_now+bb*delt_now/2.
237      Else
238         Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
239         Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
240      EndIf
241  ! Check the resultant temperature
242      If (Tf <= (T_control+deadband/2.)) Then
243         delt_now = delt_now
244         control_now = 0.
245         found_end = .true.
246      Else
247  ! Calculate the time to get to the setpoint
248         Tf = T_control+deadband/2.
249         If (aa == 0.) Then
250            delt_now = DMIN1(delt_now,((Tf-Ti_now)/bb))
251            Tf = Ti_now+bb*delt_now
252            Tave = Ti_now+bb*delt_now/2.
253         Else
254            delt_now = DMIN1(delt_now,(DLOG((Tf+bb/aa)/(Ti_now+bb/aa))/aa))
255            Tf = Ti_now*(DEXP(aa*delt_now))+bb/aa*(DEXP(aa*delt_now))-bb/aa
256            Tave = 1./aa/delt_now*(Ti_now+bb/aa)*((DEXP(aa*delt_now))-1.)-bb/aa
257         EndIf
258         control_now = 1.
259      EndIf
260   EndIf
261  ! Update the temperatures
262   Tave_tot = Tave_tot+Tave*delt_now/Timestep
263   Ti_now = Tf
264  ! Update the run-time counter
265   x_tot = x_tot+control_prev*delt_now/Timestep
266  ! Set the remaining time
267   delt_tot = delt_tot+delt_now
268   delt_now = Timestep-delt_tot
269  ! Check to see if we should run again
270   If (.not.found_end) Goto 30
271  ! Calculate the energy flows from the refrigerator
272   Q_skin = u_value*area*(T_zone-Tave_tot)
273   Q_stored = capacitance*(Tf-Ti)/Timestep
274  ! Calculate the energy removed from the space
275   Q_evap = capacity*x_tot
276  ! Calculate the power of the compressor
```

```
277   P_comp = DMAX1(0.,(Q_evap/COP-P_evapfan*x_tot-P_condfan*x_tot))
278  ! Calculate the total heat rejection
279   Q_cond = Q_evap+P_comp
280   Q_rejected = Q_cond+P_condfan*x_tot
281   Power = P_condfan*x_tot+P_evapfan*x_tot+P_comp
282
283  !------------------------------------------------------------------------------- ↵
       ---------------------------------------
284  !Set the values in storage
285   Call SetStaticArrayValue(1,Ti)
286   Call SetStaticArrayValue(2,Tf)
287   Call SetStaticArrayValue(3,control_last)
288   Call SetStaticArrayValue(4,control_now)
289
290  !------------------------------------------------------------------------------- ↵
       ---------------------------------------
291  ! Set outputs
292   Call SetOutputValue(1,Tave_tot)
293   Call SetOutputValue(2,Q_skin)
294   Call SetOutputValue(3,Q_rejected)
295   Call SetOutputValue(4,Power)
296   Call SetOutputValue(5,x_tot)
297
298   Return
299  End
300
301
```