Version 3.0 ##### 2019-7-12

AirConditioner ::= HeatRegular || Conn_sys

HeatRegular ::= HeatSW || Conn_pro

HeatSW ::= Regular || HeaterCooler || Sensor || Conn_thr || SCHEDULE

###Schedule 抢占式优先级调度策略 ###

```
SCHEDULE ::= run_now:=0; run_prior:=0; ready_num:=0;
    (   SELECT {
            tran_Reagular??prior;
            (run_prior<prior)->(BUSY; run_now:=”Regular”; run_prior:=prior; run_Regular!! ) ;
            (run_prior>=prior) -> INSERT(Regular ,prior);    ready_num := ready_num+1;
        |   tran_HeaterCooler??prior;
            (run_prior<prior)->(BUSY;run_now:=”HeaterCooler”;run_prior:=prior;
                                                                    run_HeaterCooler!! );
            (run_prior>=prior) -> INSERT(HeaterCooler ,prior); ready_num := ready_num+1;
        |   tran_Sensor?? prior;
            (run_prior<prior)->(BUSY; run_now:=”Sensor”; run_prior:=prior; run_Sensor!!);
            (run_prior>=prior) -> INSERT(Sensor ,prior); ready_num := ready_num+1;
        |   free??;
              (ready_num>0) -> (CHANGE; ready_num :=ready_num-1 ; );
              (ready_num=0) -> (run_now:=0;run_prior := 0; )
        } ; )*

BUSY ::=    (run_now=”Regular”) -> (busy_Regular!!);
              (run_now=”HeaterCooler”) -> (busy_HeaterCooler!!);
             (run_now=”Sensor”) -> (busy_Sensor!!);

INSERT (thread , prior) ::= ...
CHANGE ::= ...
```

###Schedule2 非抢占式优先级调度策略 ###

```
SCHEDULE2 ::= run_now:=0; run_prior:=0; ready_num:=0;
( SELECT{
            tran_Reagular??prior; INSERT(Regular ,prior); ready_num := ready_num+1 ;
      |     tran_HeaterCooler??prior; INSERT(HeaterCooler ,prior); ready_num := ready_num+1 ;
      |     tran_Sensor?? prior; INSERT(Sensor ,prior); ready_num := ready_num+1 ;
      |     free??; (ready_num>0) -> (CHANGE;ready_num :=ready_num-1 );
                                              (ready_num=0) -> (run_now:=0;run_prior := 0; )
```

```
      }; ) *

  INSERT (thread , prior) ::= ...
CHANGE ::=

##### 线程周期式激活调度( periodic ) #####
### THREAD Regular ###
Regular( period, deadline, priority, dispatch_protocal )
                                        ::= ACT_Regular* || DIS_Regular* || COM_Regular*

ACT_Regular ::= act_Regular !!

DIS_Regular ::= act_Regular ?? wait period ; dispatch_Regular!! ;
                    GetData(desiredTemp); GetData(measuredTemp);
                    input_Regular!! (desiredTemp , measuredTemp)
                    ( complete_Regular?? || exit_Regular?? )

COM_Regular ::= dispatch_Regular ?? ; t:=0;    init_Regular !! t ;
               (    Ready_Regular*
                    || c:=0; Running_Regular*
                    || Await_Regular*
                    || Annex_Regular
               )
Ready_Regular ::=
                 SELECT {
                      init_Regular ?? t ;
                    |   unblock_Regular ?? t ;
                    |   preempt_Regular??t ;
                    };
                    tran_Regular !! priority ;
                    {   DOT(t) = 1; DOMAIN( t< deadline)
                            INTERUPET ( run_Regular ?? -> resume_Regular !! t ; )
                     } ;
                    t=deadline -> exit _Regular!!

Running_Regular ::= resume_Regular ?? t ; run_Annex_Regular !! ;
                    { DOT(t) = 1; DOT(c) =1;    DOMAIN( t< deadline)
                    INTERUPET ( needResource_Regular ?? -> (block_Regular!! t; free!! ) )
                    AND
                    INTERUPET ( complete_Annex_Regular?? ->
                                                ( SetData(command); free!! ;complete_Regular!! ) )
                    AND
                    INTERUPET ( busy_Regular ?? -> preempt_Regular!! t ; )
```

```
                } ;

                t= deadline ->( free!! ;exit_Regular!!)

Await_Regular ::=    block_Regular?? t ;
                      { DOT(t) = 1; DOMAIN( t< deadline )
                      INTERUPET ( haveResource_Regular ?? -> unblock_Regular !! t )
                     } ;
                t = deadline -> exit_Regular!!


###THREAD HeaterCooler ###

HeatCooler( period, deadline, priority, dispatch_protocal )
                  ::= ACT_HeaterCooler* || DIS_HeaterCooler* || COM_HeaterCooler*

ACT_HeaterCooler ::= act_HeaterCooler !! ;

DIS_HeaterCooler ::= act_HeaterCooler ?? wait period ; dispatch_HeaterCooler!! ;
                       GetData(command); input_HeaterCooler!! command ;
                       ( complete_HeaterCooler?? || exit_HeaterCooler?? )

COM_HeaterCooler ::= dispatch_HeaterCooler ?? ;    t:=0;    init_HeaterCooler !! t ;
                       (    Ready_HeaterCooler*
                       || c:=0; Running_HeaterCooler*
                       || Await_HeaterCooler*
                       || Annex_HeaterCooler
                       )

Ready_HeaterCooler ::=
              SELECT {
                    init_HeaterCooler ?? t ;
                |   unblock_HeaterCooler ?? t ;
                |   preempt_HeaterCooler?? t ;
                };
                tran_HeaterCooler !! priority ;
                {   DOT(t) = 1; DOMAIN( t< deadline)
                       INTERUPET ( run_HeaterCooler ?? -> resume_HeaterCooler !! t ; )
                 } ;
                t=deadline -> exit _HeaterCooler !!

Running_HeaterCooler ::=
                 resume_HeaterCooler ?? t ; run_Annex_HeaterCooler!! ;
               { DOT(t) = 1; DOT(c) =1; DOMAIN( t< deadline )
```

```
                INTERUPET ( needResource_HeaterCooler ?? -> block_HeaterCooler!! t; free!! )
                AND
                INTERUPET ( complete_Annex_HeaterCooler?? ->
                        ( SetEvent(heating); SetEvent(cooling); free!! ;complete_HeaterCooler !! ) )
                AND
                INTERUPET ( busy_HeaterCooler ?? -> preempt_HeaterCooler!! t; )
                 } ;
                t=deadline ->( free!! ; exit_HeaterCooler !!)

Await_HeaterCooler ::= block_HeaterCooler ?? t ;
                      { DOT(t) = 1; DOMAIN( t< deadline )
                       INTERUPET ( haveResource_HeaterCooler ?? -> unblock_HeaterCooler !! t )
                      } ;
                      t=deadline -> exit_HeaterCooler !!


###THREAD Sensor ###
Sensor( period, deadline, priority, dispatch_protocal )
                                              ::= ACT_Sensor* || DIS_Sensor* || COM_Sensor*

ACT_Sensor ::= act_Sensor !! ;

DIS_Sensor::= act_Sensor ?? wait period ; dispatch_Sensor!! ;
                GetData(heaterTemp) ; input_Sensor!! heaterTemp;
                ( complete_Sensor?? || exit_Sensor?? )

COM_Sensor ::= dispatch_Sensor ?? ;    t:=0;    init_Sensor !! t ;
              (     Ready_Sensor*
                || c:=0;Running_Sensor*
                || Await_Sensor*
                || Annex_Sensor
                )

Ready_Sensor ::=
               SELECT {
                     init_Sensor ?? t ;
                |    unblock_Sensor ?? t ;
                |    preempt_Sensor ?? t ;
                };
                tran_Sensor !! priority ;
                {     DOT(t) = 1; DOMAIN( t< deadline)
                       INTERUPET ( run_Sensor ?? -> resume_Sensor !! t ; )
                 } ;
                t=deadline -> exit _Sensor !!
```

```
Running_Sensor ::=
                resume_Sensor ?? t ; run_ANNEX_Sensor !!
                { DOT(t) = 1; DOT(c) =1;
                DOMAIN( t< deadline )
                INTERUPET ( needResource_Sensor ?? -> block_Sensor!! t; free!! )
                AND
                INTERUPET ( complete_Annex_Sensor?? ->
                                        ( SetData(measuredTemp); free!! ;complete_Sensor!! ) )
                AND
                INTERUPET ( busy_Sensor ?? -> preempt_Sensor!! t; )
                 } ;
                t=deadline ->( free!! ; exit_Sensor !!)

Await_Sensor ::= block_Sensor ?? t ;
                   { DOT(t) = 1; DOMAIN( t< deadline )
                      INTERUPET ( haveResource_Sensor ?? -> unblock_Sensor !! t )
                   } ;
                   t=deadline -> exit_Sensor !!


### 资源调度 ###

ResourceApplication ::=
        needResource_Regular??; GETRESOURCE; haveResorce_Regular!!;
        needResource_HeaterCooler??; GETRESOURCE; haveResorce_HeaterCooler!!;
        needResource_Sensor??; GETRESOURCE; haveResorce_Sensor!!;

### Connection ###
 Conn_sys ::= Settings?? x ; HeatRegulator_desiredTemp!! x ;
              | | HeatRegulator_currentTemp ??x ; Temperature!! x;
              | | HeatRegulator_heating ?? ; HeaterStatus_red!!;
              | | HeatRegulator.cooling ?? ; HeaterStatus.green!!;

Conn_pro ::=   HeatRegular_desiredTemp ??x; HeaterSW_desiredTemp!! x ;
              | | HeaterSW_heating ?? ; HeatRegular_heating !! ;
              | | HeaterSW_cooling ?? ; HeatRegular_cooling !! ;
              | | HeaterSW_measuredTemp ??x ; HeatRegular_currentTemp !!x ;

Conn_thr ::=   HeaterSW_desiredTemp ??x ; Regulator_desiredTemp!! x ;
              | | HeaterCooler_heating ?? ; HeaterSW_heating!! ;
              | | HeaterCooler_cooling ?? ; HeaterSW_cooling!! ;
              | | Sensor_measuredTemp ??x ; HeaterSW_measuredTemp !! x ;
              | | Regulator_heaterCommand??x ;HeaterCooler_command!! x ;
```

|| HeaterCooler_temperature??x ; Sensor_heaterTemp !! x ;

|| Sensor_measuredTemp ??x ; Regulator_measuredTemp!! x;

附： AADL 代码

```
----------------------------------------
-- Air Conditioner
-- AADL Inspector
-- (c) Ellidiss Technologies
-- Updated: January 2017
----------------------------------------

PACKAGE AirConditioner_Pkg
PUBLIC
WITH Ellidiss::Math::Int;
RENAMES Ellidiss::Math::Int::ALL;
WITH Ellidiss::Gui;
RENAMES Ellidiss::Gui::ALL;
WITH AI;

SYSTEM AirConditioner
END AirConditioner;

SYSTEM IMPLEMENTATION AirConditioner.others
SUBCOMPONENTS
  Settings : DEVICE IntSelector;
  Temperature : DEVICE IntDisplay;
  HeaterStatus : DEVICE Light;
  HeatRegulator : SYSTEM HeatRegulator.others;
CONNECTIONS
  cnx_0 : PORT Settings.value -> HeatRegulator.desiredTemp;
  cnx_1 : PORT HeatRegulator.currentTemp -> Temperature.value;
  cnx_2 : PORT HeatRegulator.heating -> HeaterStatus.red;
  cnx_3 : PORT HeatRegulator.cooling -> HeaterStatus.green;
PROPERTIES
-- required by Ocarina
  AI::root_system => "SELECTED";
END AirConditioner.others;

SYSTEM HeatRegulator
FEATURES
  desiredTemp : IN DATA PORT int;
  heating : OUT EVENT PORT;
  cooling : OUT EVENT PORT;
  currentTemp : OUT DATA PORT int;
END HeatRegulator;

SYSTEM IMPLEMENTATION HeatRegulator.others
SUBCOMPONENTS
  HeaterSW : PROCESS HeaterSW.others;
  HeaterCPU : PROCESSOR HeaterCPU;
  HeaterRAM : MEMORY HeaterRAM;
CONNECTIONS
  cnx_0 : PORT desiredTemp -> HeaterSW.desiredTemp;
  cnx_1 : PORT HeaterSW.heating -> heating;
  cnx_2 : PORT HeaterSW.cooling -> cooling;
  cnx_3 : PORT HeaterSW.measuredTemp -> currentTemp;
PROPERTIES
  Actual_Processor_Binding => ( reference(HeaterCPU) ) applies to HeaterSW;
  Actual_Memory_Binding => ( reference(HeaterRAM) ) applies to HeaterSW;
END HeatRegulator.others;
```

```
57
58 PROCESS HeaterSW
59 FEATURES
60   desiredTemp : IN DATA PORT int;
61   heating : OUT EVENT PORT;
62   cooling : OUT EVENT PORT;
63   measuredTemp : OUT DATA PORT int;
64 END HeaterSW;
65
66 PROCESS IMPLEMENTATION HeaterSW.others
67 SUBCOMPONENTS
68   Regulator : THREAD Regulator.others;
69   HeaterCooler : THREAD HeaterCooler.others;
70   Sensor : THREAD Sensor.others;
71 CONNECTIONS
72   cnx_0 : PORT desiredTemp -> Regulator.desiredTemp;
73   cnx_1 : PORT HeaterCooler.heating -> heating;
74   cnx_2 : PORT HeaterCooler.cooling -> cooling;
75   cnx_3 : PORT Sensor.measuredTemp -> measuredTemp;
76 1  cnx_4 : PORT Regulator.heaterCommand -> HeaterCooler.command;
77   cnx_5 : PORT HeaterCooler.temperature -> Sensor.heaterTemp;
78   cnx_6 : PORT Sensor.measuredTemp -> Regulator.measuredTemp;
79 END HeaterSW.others;
80
81 THREAD Regulator
82 FEATURES
83   desiredTemp : IN DATA PORT int;
84   measuredTemp : IN DATA PORT int;
85   heaterCommand : OUT DATA PORT int;
86 END Regulator;
87
88 THREAD IMPLEMENTATION Regulator.others
89 PROPERTIES
90   Dispatch_Protocol => Periodic;
91   Priority => 8;
92   Deadline => 20ms;
93   Period => 20ms;
94 ANNEX Behavior_Specification {**
95 VARIABLES diff, gain : int;
96 STATES s : INITIAL COMPLETE FINAL STATE;
97 TRANSITIONS t : s -[ON DISPATCH]-> s
98   { gain := 2;
99     diff :=  desiredTemp - measuredTemp;
100    heaterCommand := diff * gain };
101 **};
102 END Regulator.others;
103
104 THREAD HeaterCooler
105 FEATURES
106   command : IN DATA PORT int;
107   temperature : OUT DATA PORT int;
108   heating : OUT EVENT PORT;
109   cooling : OUT EVENT PORT;
110 END HeaterCooler;
111
112 THREAD IMPLEMENTATION HeaterCooler.others
113 SUBCOMPONENTS
114   Temp : DATA int;
115 PROPERTIES
116   Dispatch_Protocol => Periodic;
117   Priority => 6;
118   Deadline => 20ms;
119   Period => 20ms;
120 ANNEX Behavior_Specification {**
121 STATES s : INITIAL COMPLETE FINAL STATE;
122 TRANSITIONS t : s -[ON DISPATCH]-> s
123   { if (command >= 0) heating!; Temp := Temp + 1 end if;
124     if (command < 0) cooling!; Temp := Temp - 1 end if;
125     temperature := Temp };
126 **};
127 END HeaterCooler.others;
```

```
128
129THREAD Sensor
130FEATURES
131  heaterTemp : IN DATA PORT int;
132  measuredTemp : OUT DATA PORT int;
133END Sensor;
134
135THREAD IMPLEMENTATION Sensor.others
136PROPERTIES
137  Dispatch_Protocol => Periodic;
138  Priority => 10;
139  Deadline => 20ms;
140  Period => 20ms;
141ANNEX Behavior_Specification {**
142VARIABLES e : int;
143STATES s : INITIAL COMPLETE FINAL STATE;
144TRANSITIONS t : s -[ON DISPATCH]-> s
145  { err!(2,e); measuredTemp := heaterTemp + e };
146**};
147END Sensor.others;
148
149PROCESSOR HeaterCPU
150PROPERTIES
151  Scheduling_Protocol => (HPF);
152END HeaterCPU;
153
154MEMORY HeaterRAM
155END HeaterRAM;
156
157END AirConditioner_Pkg;
```