

Version 2.0

AirConditioner ::= DEVICE || HeatRegular || Conn_sys

HeatRegular ::= HeatCPU || HeatRAM || 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;

(tranR_Reagular??prior;
 (run_prior<prior)->(BUSY; CHANGE(Regular,prior););
 (run_prior>prior) -> INSERT(Regular ,prior); ready_num := ready_num+1
|| tranR_HeaterCooler??prior;
 (run_prior<prior)->(BUSY;CHANGE(HeaterCooler,prior););
 (run_prior>prior) -> INSERT(HeaterCooler ,prior); ready_num := ready_num+1
|| tranR_Sensor?? prior;
 (run_prior<prior)->(BUSY; CHANGE(Sensor,prior););
 (run_prior>prior) -> INSERT(Sensor ,prior); ready_num := ready_num+1
)*; DISPATCH*

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

CHANGE(thread,prior) ::=

 (thread ="Regular") -> (run_now:="Regular"; run_prior:=prior; run_Regular!!);
 || (thread ="HeaterCooler") ->
 (run_now:="HeaterCooler"; run_prior:=prior; run_HeaterCooler!!);
 || (thread ="Sensor") -> (run_now:="Sensor"; run_prior:=prior; run_Sensor!!);

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

DISPATCH ::= free??;

 (ready_num>0) -> (CHANGE(first_ready, first_prior);ready_num :=ready_num-1 ;);
 (ready_num=0) -> (run_now:=0;run_prior := 0;)

####Schedule2 非抢占式优先级调度策略 ####

```
SCHEDULE2 ::= run_now:=0; run_prior:=0; ready_num:=0;
  ( tranR_Reagular??prior; INSERT(Regular ,prior); ready_num := ready_num+1
  || tranR_HeaterCooler??prior; INSERT(HeaterCooler ,prior); ready_num := ready_num+1
  || tranR_Sensor?? prior; INSERT(Sensor ,prior); ready_num := ready_num+1
  )*; DISPATCH*
```

```
CHANGE(thread,prior) ::=
  (thread ="Regular") -> (run_now:="Regular"; run_prior:=prior; run_Regular!!);
  || (thread ="HeaterCooler") ->
    (run_now:="HeaterCooler"; run_prior:=prior; run_HeaterCooler!!);
  || (thread ="Sensor") -> (run_now:="Sensor"; run_prior:=prior; run_Sensor!!);
```

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

```
DISPATCH ::= free??
  (ready_num>0) -> (CHANGE(first_ready, first_prior);ready_num :=ready_num-1 );
  (ready_num=0) -> (run_now:=0;run_prior := 0; )
```

THREAD Regular

Regular(period, deadline, priority, dispatch_protocol) ::= ACT-Regular* || COM-Regular*

ACT-Regular ::= act-Regular !! ; wait period ;

```
COM-Regular ::= act-Regular ??
  ( t:=0; isReady :=1 ; ( isReady -> Ready-Regular)*
  || c:=0; Running-Regular*
  || Await-Regular*
  )
```

```
Ready-Regular ::= (unblock-Regular ?? ct) -> (t :=ct) ; tranR-Regular !! priority ;
  { DOT(t) = 1; DOMAIN( t< deadline)
    INTERUPET ( run-Regular ?? -> resume-Regular !! t ; )
  };
  t=deadline -> isReady := 0 ;
```

```
Running-Regular ::= resume-Regular ?? t ;
  GetData(desiredTemp); GetData(measuredTemp);
  { DOT(t) = 1; DOT(c) =1; ANNEX-Regular ; DOMAIN( t< deadline)
    INTERUPET ( needResource-Regular ?? -> block-Regular!! t;free!!)
  AND
```

```

INTERUPET ( complete_Annex-Regular ?? -> SetData(command) ; free!! )
AND
INTERUPET ( busy-Regular ?? -> unblock-Regular!!t ; )
};

t=deadline -> free!!

Await-Regular ::= block-Regular?? t ;
{ DOT(t) = 1; DOMAIN( t < deadline )
INTERUPET ( haveResource-Regular ?? -> unblock-Regular !! t )
};

####THREAD HeaterCooler ####
HeatCooler( period, deadline, priority, dispatch_protocol )
::= ACT_HeaterCooler* || COM_HeaterCooler*

ACT_HeaterCooler ::= act_HeaterCooler !! ; wait period ;

COM_HeaterCooler ::= act_HeaterCooler ?? ;
( t:=0; isReady :=1 ; ( isReady -> Ready_HeaterCooler )*
|| c:=0; Running_HeaterCooler*
|| Await_HeaterCooler*
)

Ready_HeaterCooler ::=
(unblock_HeaterCooler ?? ct) -> (t :=ct) ;
tranR_HeaterCooler !! priority ;
{ DOT(t) = 1; DOMAIN( t < deadline )
INTERUPET ( run_HeaterCooler ?? -> resume_HeaterCooler !! t ) ;
};

t=deadline -> isReady := 0 ;

Running_HeaterCooler ::=
resume-Regular ?? t ;
GetData(command);
{ DOT(t) = 1; DOT(c) =1; ANNEX_HeaterCooler ; DOMAIN( t < deadline )
INTERUPET ( needResource_HeaterCooler ?? -> block_HeaterCooler!! t; free!! )
AND
INTERUPET(complete_Annex_HeaterCooler??->SetEvent(heating);
SetEvent(cooling); free!! )
AND
INTERUPET ( busy_HeaterCooler ?? -> unblock_HeaterCooler!! t ) ;
};

t=deadline -> free!!

```

```

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

####THREAD Sensor ####
Sensor( period, deadline, priority, dispatch_protocol ) ::= ACT_Sensor* || COM_Sensor*
ACT_Sensor ::= act_Sensor !! ; wait period ;

COM_Sensor ::= act_Sensor ?? ;
    (   t:=0; isReady :=1 ; ( isReady -> Ready_Sensor )*
        || c:=0; Running_Sensor*
        || Await_Sensor*
    )

Ready_Sensor ::=  (unblock_Sensor ?? ct) -> (t :=ct) ;
    tranR_Sensor !! priority ;
    {  DOT(t) = 1; DOMAIN( t< deadline )
        INTERUPET ( run_Sensor?? -> resume_Sensor !! t ; )
    } ;
    t=deadline -> isReady := 0 ;

Running_Sensor ::=
    resume_Sensor ?? t ;
    GetData(heaterTemp);
    { DOT(t) = 1; DOT(c) =1; ANNEX_Sensor ; DOMAIN( t< deadline )
        INTERUPET ( needResource_Sensor ?? -> block_Sensor !! t ;free!! )
        AND
        INTERUPET(complete_Annex_Sensor ?? -> SetData(measuredTemp); free!! )
        AND
        INTERUPET(busy_Sensor ?? -> unblock_Sensor!!t )
    } ;
    t=deadline -> free!!

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

### 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

```
1 -----  
2 -- Air Conditioner  
3 -- AADL Inspector  
4 -- (c) Ellidiss Technologies  
5 -- Updated: January 2017  
6 -----  
7  
8 PACKAGE AirConditioner_Pkg  
9 PUBLIC  
10 WITH Ellidiss::Math::Int;  
11 RENAMES Ellidiss::Math::Int::ALL;  
12 WITH Ellidiss::Gui;  
13 RENAMES Ellidiss::Gui::ALL;  
14 WITH AI;  
15  
16 SYSTEM AirConditioner  
17 END AirConditioner;  
18  
19 SYSTEM IMPLEMENTATION AirConditioner.others  
20 SUBCOMPONENTS  
21   Settings : DEVICE IntSelector;  
22   Temperature : DEVICE IntDisplay;  
23   HeaterStatus : DEVICE Light;  
24   HeatRegulator : SYSTEM HeatRegulator.others;  
25 CONNECTIONS  
26   cnx_0 : PORT Settings.value -> HeatRegulator.desiredTemp;  
27   cnx_1 : PORT HeatRegulator.currentTemp -> Temperature.value;  
28   cnx_2 : PORT HeatRegulator.heating -> HeaterStatus.red;  
29   cnx_3 : PORT HeatRegulator.cooling -> HeaterStatus.green;  
30 PROPERTIES  
31   -- required by Ocarina  
32   AI::root_system => "SELECTED";  
33 END AirConditioner.others;  
34  
35 SYSTEM HeatRegulator  
36 FEATURES  
37   desiredTemp : IN DATA PORT int;  
38   heating : OUT EVENT PORT;  
39   cooling : OUT EVENT PORT;  
40   currentTemp : OUT DATA PORT int;  
41 END HeatRegulator;  
42  
43 SYSTEM IMPLEMENTATION HeatRegulator.others  
44 SUBCOMPONENTS  
45   HeaterSW : PROCESS HeaterSW.others;  
46   HeaterCPU : PROCESSOR HeaterCPU;  
47   HeaterRAM : MEMORY HeaterRAM;  
48 CONNECTIONS  
49   cnx_0 : PORT desiredTemp -> HeaterSW.desiredTemp;  
50   cnx_1 : PORT HeaterSW.heating -> heating;  
51   cnx_2 : PORT HeaterSW.cooling -> cooling;  
52   cnx_3 : PORT HeaterSW.measuredTemp -> currentTemp;  
53 PROPERTIES  
54   Actual_Processor_Binding => ( reference(HeaterCPU) ) applies to HeaterSW;  
55   Actual_Memory_Binding => ( reference(HeaterRAM) ) applies to HeaterSW;  
56 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   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**};
102END 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|
129 THREAD Sensor
130 FEATURES
131   heaterTemp : IN DATA PORT int;
132   measuredTemp : OUT DATA PORT int;
133 END Sensor;
134|
135 THREAD IMPLEMENTATION Sensor.others
136 PROPERTIES
137   Dispatch_Protocol => Periodic;
138   Priority => 10;
139   Deadline => 20ms;
140   Period => 20ms;
141 ANNEX Behavior_Specification {**
142 VARIABLES e : int;
143 STATES s : INITIAL COMPLETE FINAL STATE;
144 TRANSITIONS t : s -[ON DISPATCH]-> s
145   { err!(2,e); measuredTemp := heaterTemp + e };
146 **};
147 END Sensor.others;
148|
149 PROCESSOR HeaterCPU
150 PROPERTIES
151   Scheduling_Protocol => (HPF);
152 END HeaterCPU;
153|
154 MEMORY HeaterRAM
155 END HeaterRAM;
156|
157 END AirConditioner_Pkg;

```