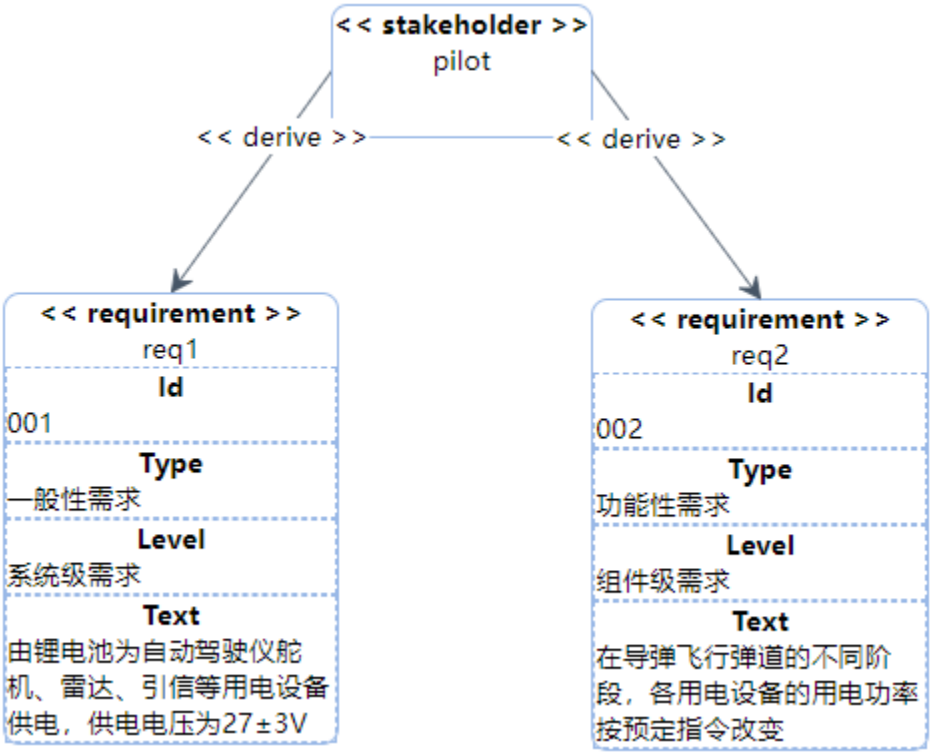
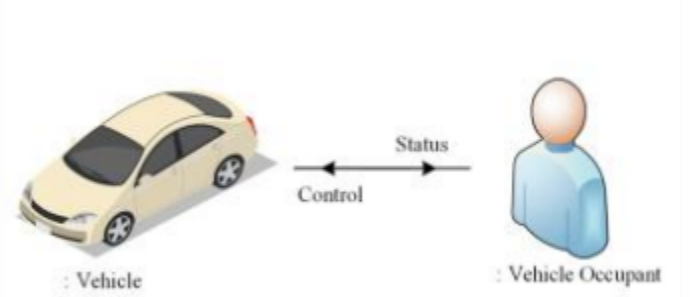
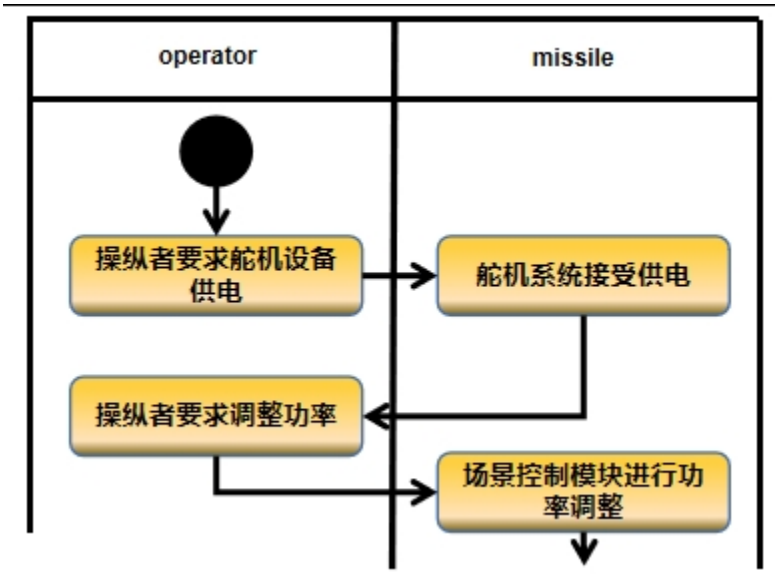
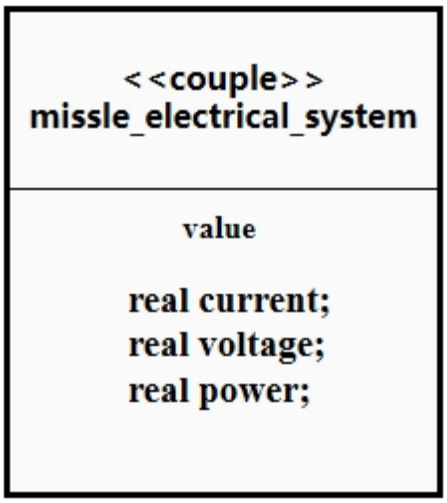
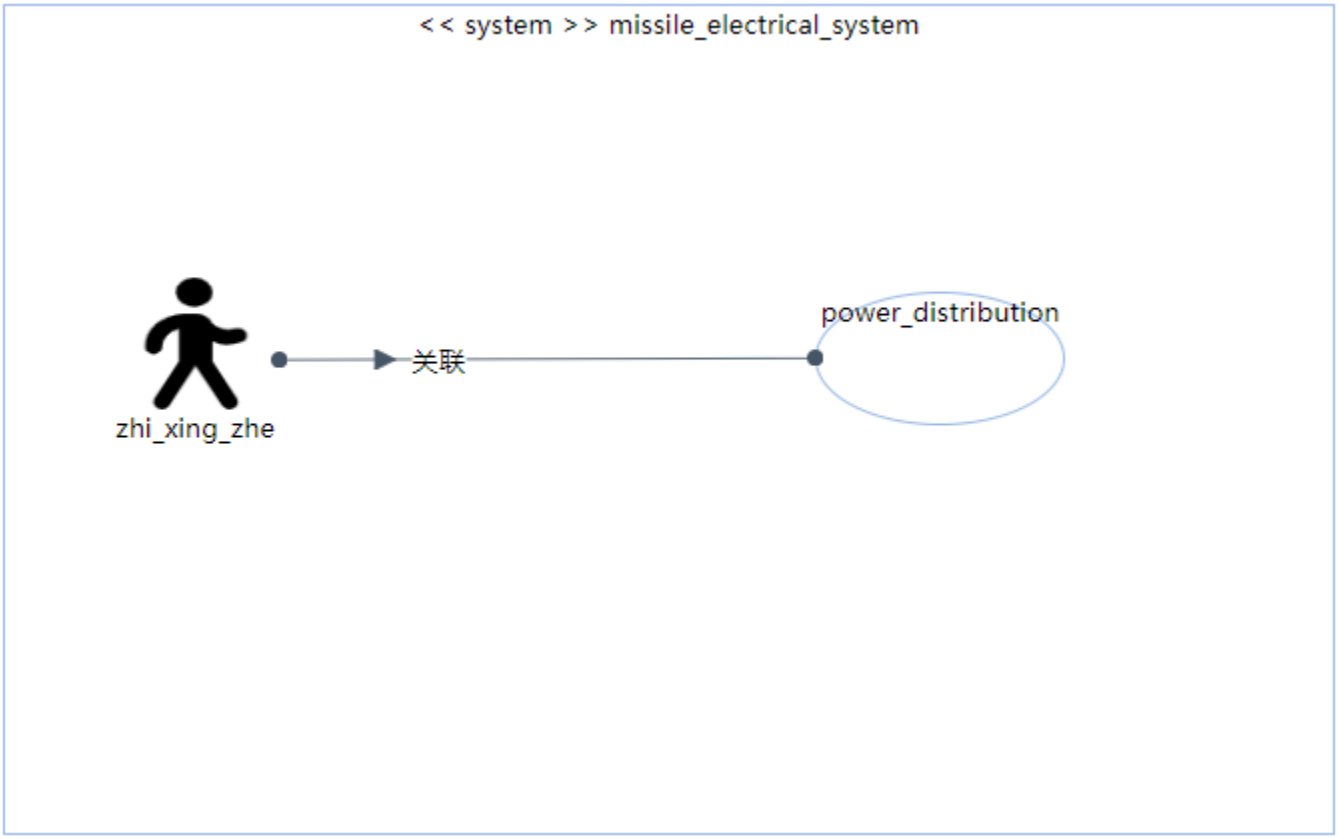
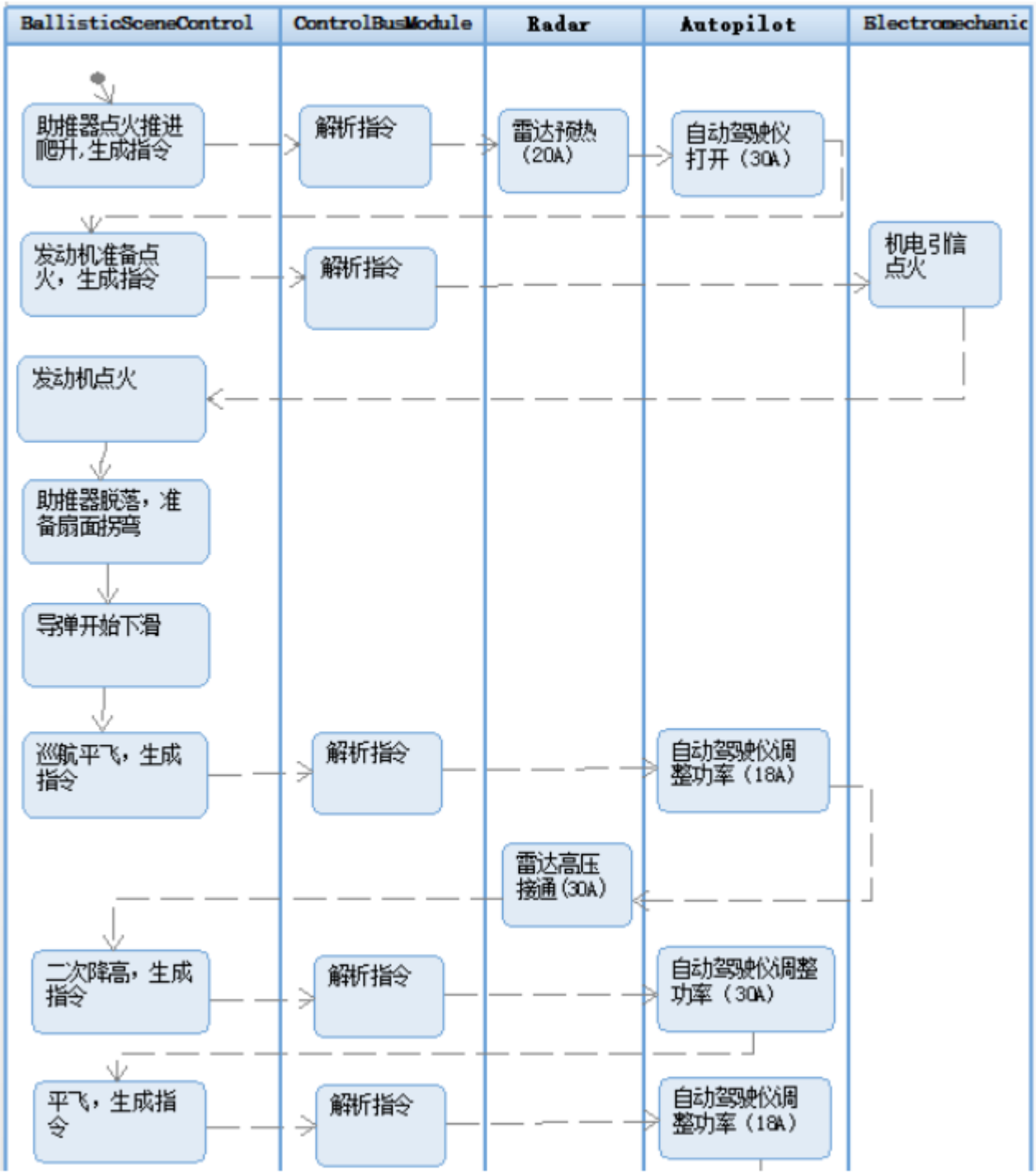

<< stakeholder >>
pilot
<< derive >>
<< derive >>
<< requirement >>
req1
Id
001
Type
一般性需求
Level
系统级需求
Text
由锂电池为自动驾驶仪舵机、雷达、引信等用电设备供电，供电电压为27±3V
<< requirement >>
req2
Id
002
Type
功能性需求
Level
组件级需求
Text
在导弹飞行弹道的不同阶段，各用电设备的用电功率按预定指令改变

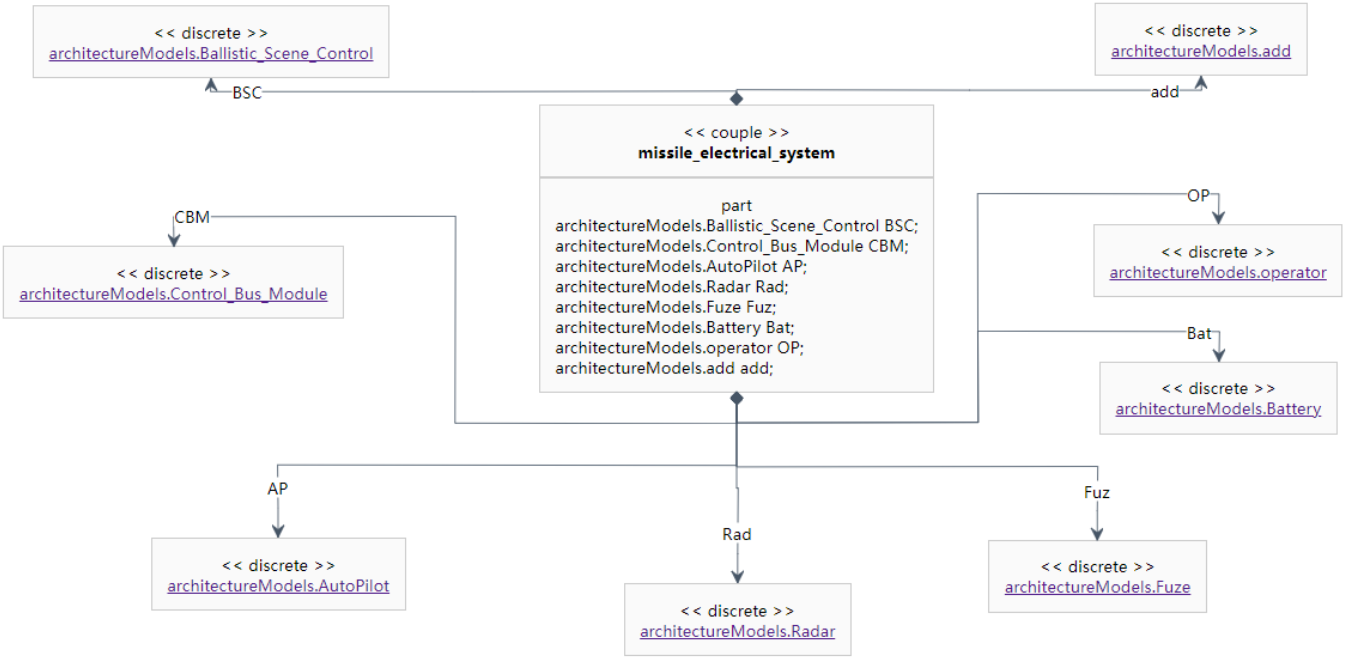

Status
Control
: Vehicle
: Vehicle Occupant

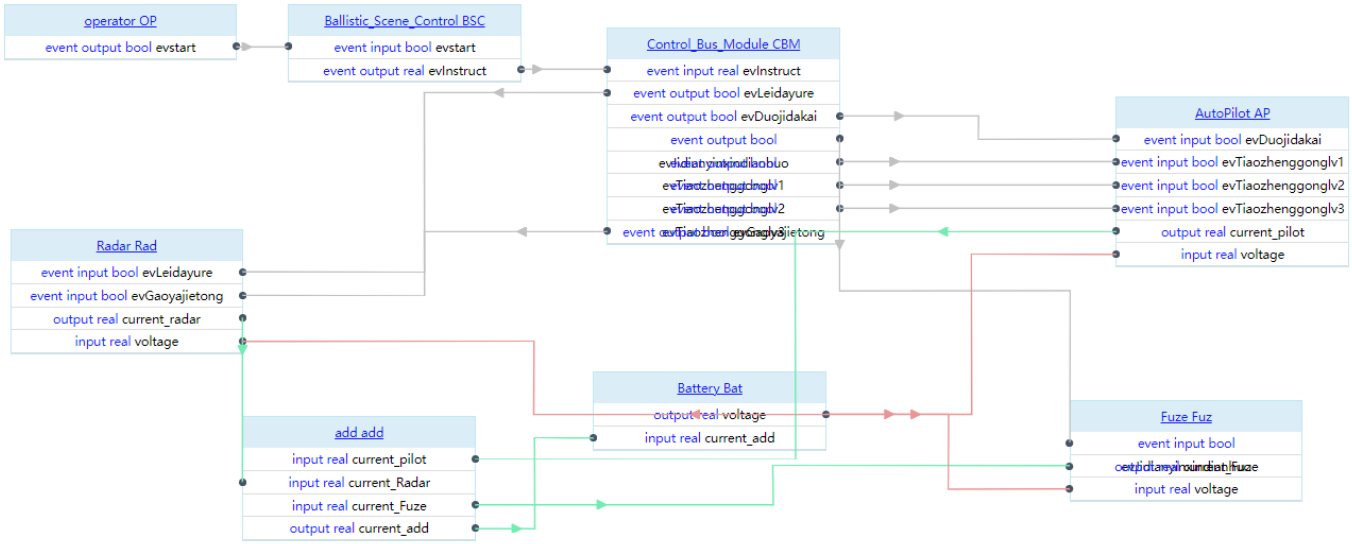

operator
missile
操纵者要求舵机设备供电
舵机系统接受供电
操纵者要求调整功率
场景控制模块进行功率调整


<<couple>>
missle_electrical_system
value
real current;
real voltage;
real power;

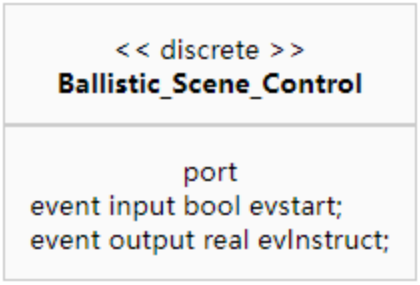

<< system >> missile_electrical_system
power_distribution
关联
zhi_xing_zhe


BallisticSceneControl
ControlBusModule
Radar
Autopilot
Electromechanic
助推器点火推进爬升，生成指令
解析指令
雷达预热（20A）
自动驾驶仪打开（30A）
发动机准备点火，生成指令
解析指令
机电引信点火
发动机点火
助推器脱落，准备扇面拐弯
导弹开始下滑
巡航平飞，生成指令
解析指令
自动驾驶仪调整功率（18A）
雷达高压接通(30A)
二次降高，生成指令
解析指令
自动驾驶仪调整功率（30A）
平飞，生成指令
解析指令
自动驾驶仪调整功率（18A）

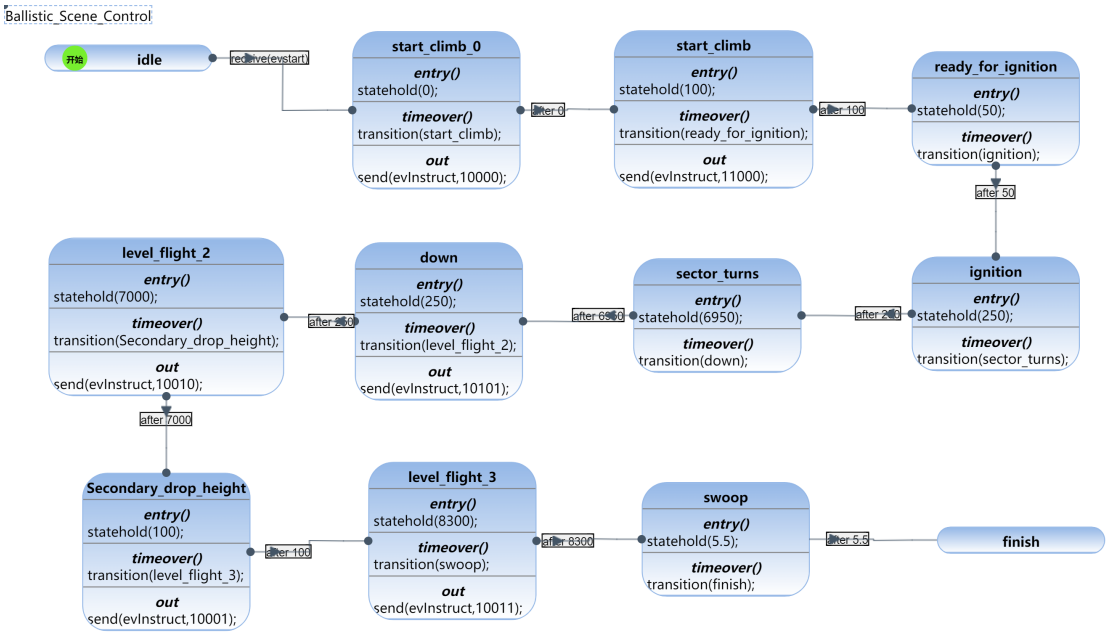

<< discrete >>
architectureModels.Ballistic_Scene_Control
BSC
<< discrete >>
architectureModels.add
add
<< couple >>
missile_electrical_system
part
architectureModels.Ballistic_Scene_Control BSC;
architectureModels.Control_Bus_Module CBM;
architectureModels.AutoPilot AP;
architectureModels.Radar Rad;
architectureModels.Fuze Fuz;
architectureModels.Battery Bat;
architectureModels.operator OP;
architectureModels.add add;
CBM
<< discrete >>
architectureModels.Control_Bus_Module
OP
<< discrete >>
architectureModels.operator
Bat
<< discrete >>
architectureModels.Battery
AP
<< discrete >>
architectureModels.AutoPilot
Rad
<< discrete >>
architectureModels.Radar
Fuz
<< discrete >>
architectureModels.Fuze

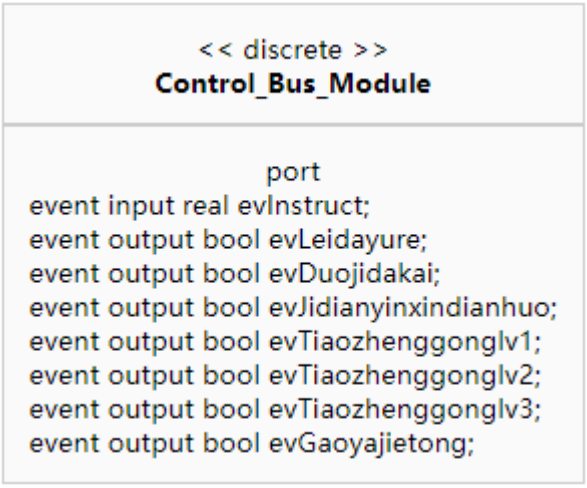

operator OP
event output bool evstart
Ballistic_Scene_Control BSC
event input bool evstart
event output real evInstruct
Control_Bus_Module CBM
event input real evInstruct
event output bool evLeidayure
event output bool evDuojidakai
event output bool
AutoPilot AP
event input bool evDuojidakai
event input bool evTiaozhenggonglv1
event input bool evTiaozhenggonglv2
event input bool evTiaozhenggonglv3
output real current_pilot
input real voltage
Radar Rad
event input bool evLeidayure
event input bool evGaoyajietong
output real current_radar
input real voltage
Battery Bat
output real voltage
input real current_add
Fuze Fuz
event input bool
input real voltage
add add
input real current_pilot
input real current_Radar
input real current_Fuze
output real current_add

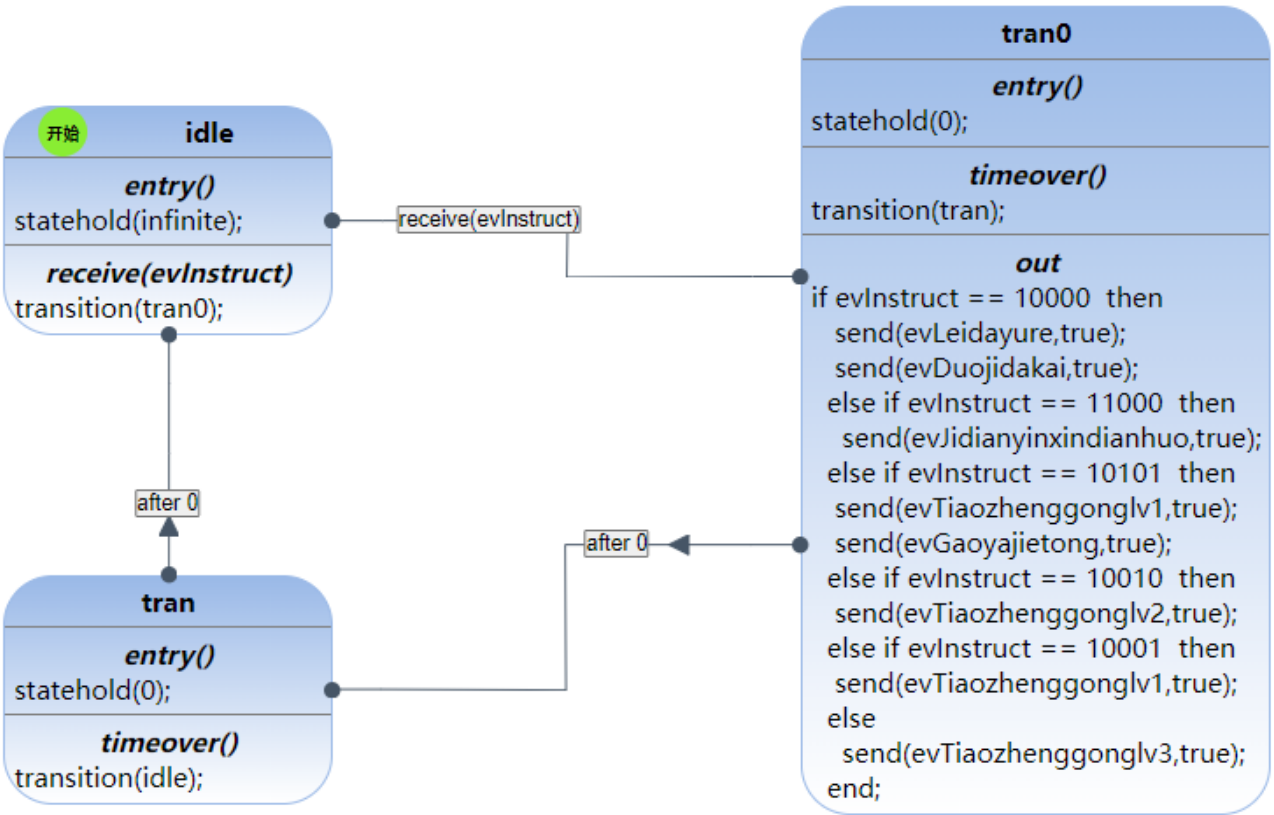
```
couple missile_electrical_system
import architectureModels.Ballistic_Scene_Control as
BSC;
import architectureModels.Control_Bus_Module as
CBM;
...
part:
  BSC BSC;
  CBM CBM;
...
connection:
  connect(OP.evstart,BSC.evstart);
  connect(BSC.evInstruct,CBM.evInstruct);
  connect(CBM.evDuojidakai,AP.evDuojidakai);
  connect(CBM.evLeidayure,Rad.evLeidayure);
...
end;
```

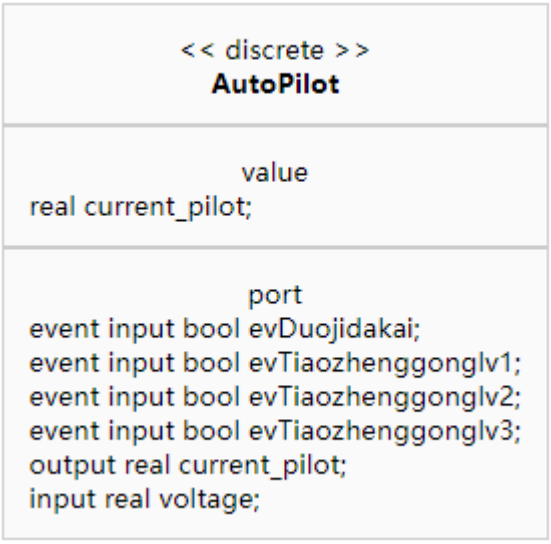

<< discrete >>
Ballistic_Scene_Control
port
event input bool evstart;
event output real evInstruct;

```
discrete Ballistic_Scene_Control
port:
  event input bool evstart;
  event output real evInstruct;
state:
initial state idle
when entry() then
  statehold(infinite);
end;
when receive(evstart) then
  transition(start_climb_0);
end;
end;
state start_climb_0
when entry() then
  statehold(0);
end;
when timeover() then
  transition(start_climb);
end;
end;
state start_climb
when entry() then
  statehold(100);
end;
when timeover() then
  transition(ready_for_ignition);
end;
end;
...
state finish
end;
end;
```

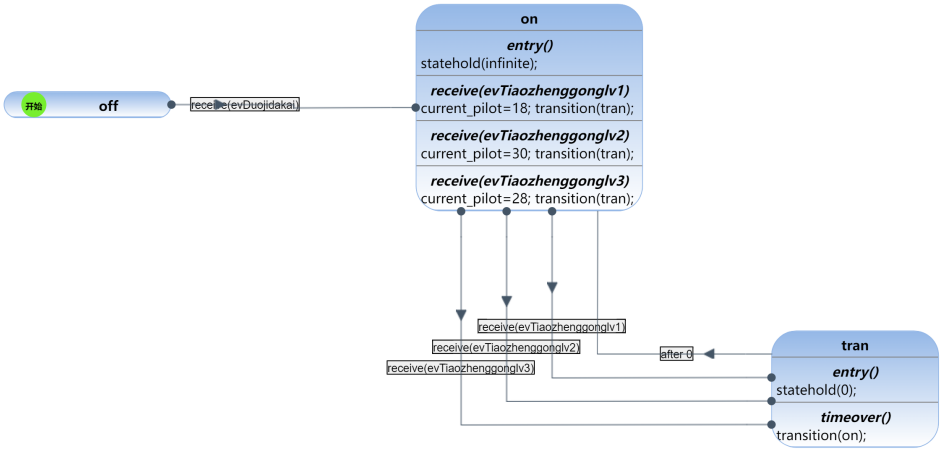

Ballistic_Scene_Control
开始
idle
receive(evstart)
start_climb_0
entry()
statehold(0);
timeover()
transition(start_climb);
out
send(evInstruct,10000);
after 0
start_climb
entry()
statehold(100);
timeover()
transition(ready_for_ignition);
out
send(evInstruct,11000);
after 100
ready_for_ignition
entry()
statehold(50);
timeover()
transition(ignition);
after 50
ignition
entry()
statehold(250);
timeover()
transition(sector_turns);
after 200
sector_turns
entry()
statehold(6950);
timeover()
transition(down);
after 6950
down
entry()
statehold(250);
timeover()
transition(level_flight_2);
out
send(evInstruct,10101);
after 250
level_flight_2
entry()
statehold(7000);
timeover()
transition(Secondary_drop_height);
out
send(evInstruct,10010);
after 7000
Secondary_drop_height
entry()
statehold(100);
timeover()
transition(level_flight_3);
out
send(evInstruct,10001);
after 100
level_flight_3
entry()
statehold(8300);
timeover()
transition(swoop);
out
send(evInstruct,10011);
after 8300
swoop
entry()
statehold(5.5);
timeover()
transition(finish);
after 5.5
finish

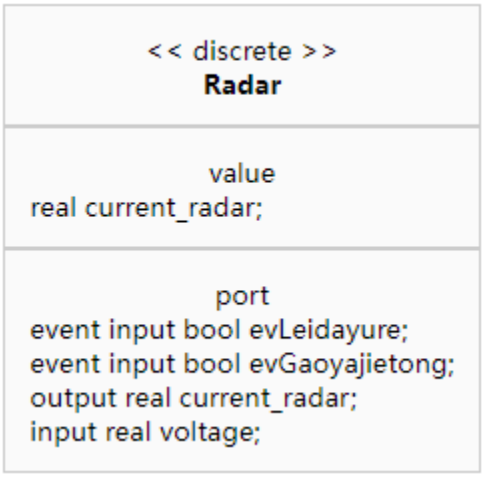

<< discrete >>
Control_Bus_Module
port
event input real evInstruct;
event output bool evLeidayure;
event output bool evDuojidakai;
event output bool evJidianyinxindianhuo;
event output bool evTiaozhenggonglv1;
event output bool evTiaozhenggonglv2;
event output bool evTiaozhenggonglv3;
event output bool evGaoyajietong;

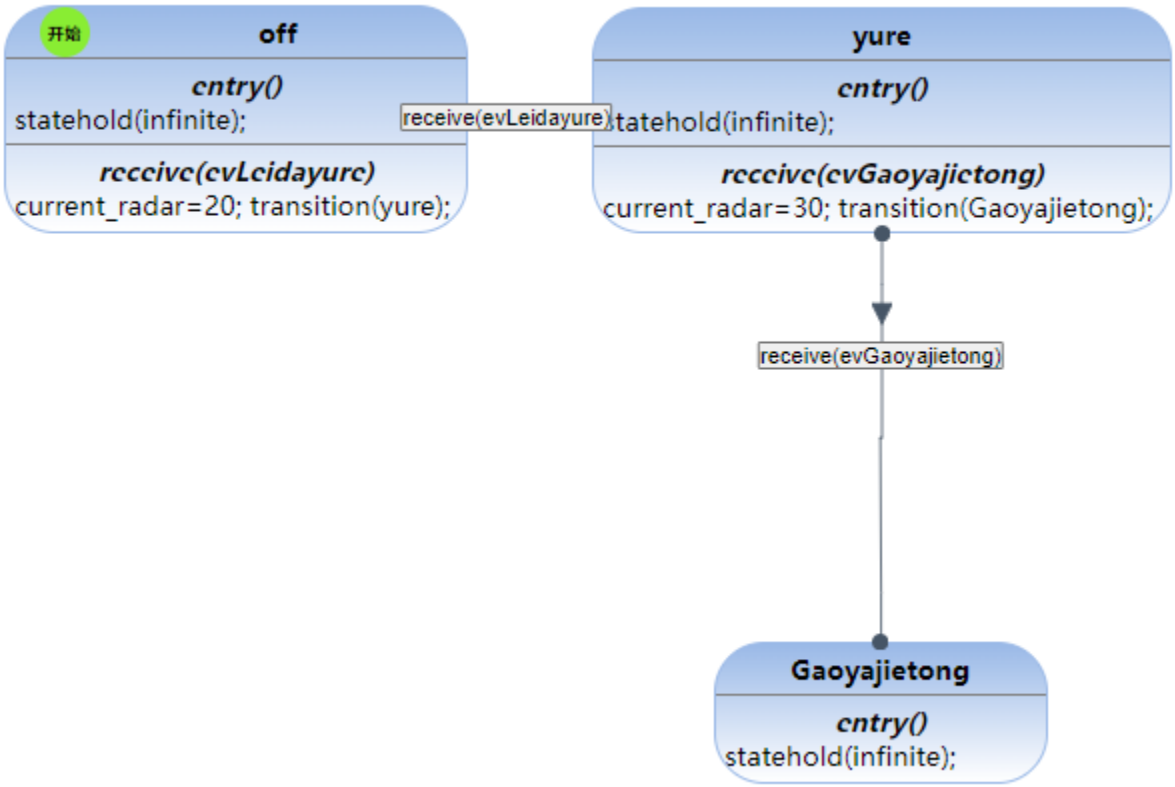

开始
idle
entry()
statehold(infinite);
receive(evInstruct)
transition(tran0);
receive(evInstruct)
tran0
entry()
statehold(0);
timeover()
transition(tran);
out
if evInstruct == 10000 then
send(evLeidayure,true);
send(evDuojidakai,true);
else if evInstruct == 11000 then
send(evJidianyinxindianhuo,true);
else if evInstruct == 10101 then
send(evTiaozhenggonglv1,true);
send(evGaoyajietong,true);
else if evInstruct == 10010 then
send(evTiaozhenggonglv2,true);
else if evInstruct == 10001 then
send(evTiaozhenggonglv1,true);
else
send(evTiaozhenggonglv3,true);
end;
after 0
after 0
tran
entry()
statehold(0);
timeover()
transition(idle);


<< discrete >>
AutoPilot
value
real current_pilot;
port
event input bool evDuojidakai;
event input bool evTiaozhenggonglv1;
event input bool evTiaozhenggonglv2;
event input bool evTiaozhenggonglv3;
output real current_pilot;
input real voltage;

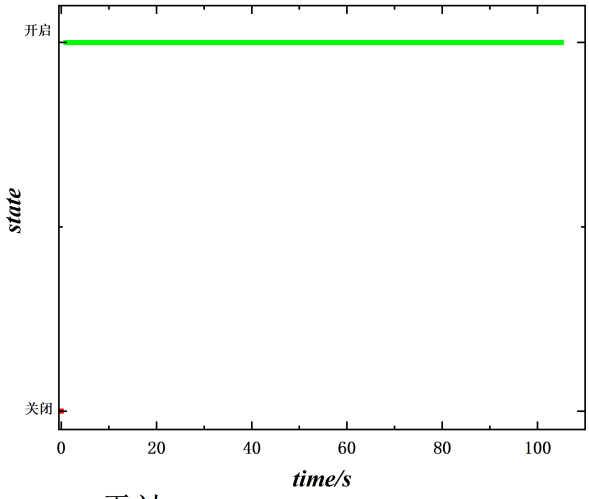
AutoPilot

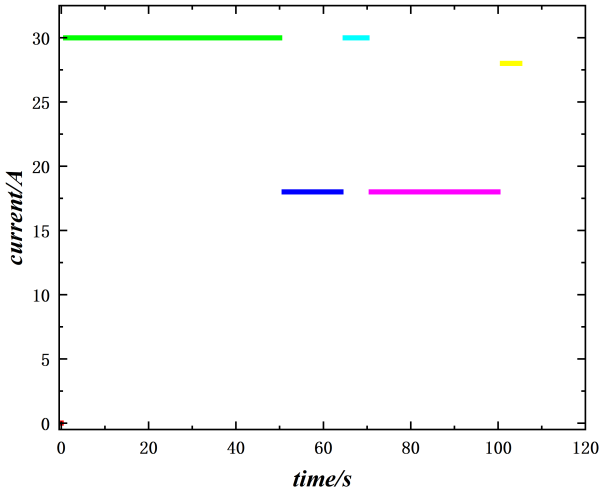

on
entry()
statehold(infinite);
receive(evTiaozhenggonglv1)
current_pilot=18; transition(tran);
receive(evTiaozhenggonglv2)
current_pilot=30; transition(tran);
receive(evTiaozhenggonglv3)
current_pilot=28; transition(tran);
开始
off
receive(evDuojidakai)
receive(evTiaozhenggonglv1)
receive(evTiaozhenggonglv2)
receive(evTiaozhenggonglv3)
after 0
tran
entry()
statehold(0);
timeover()
transition(on);


<< discrete >>
Radar
value
real current_radar;
port
event input bool evLeidayure;
event input bool evGaoyajietong;
output real current_radar;
input real voltage;

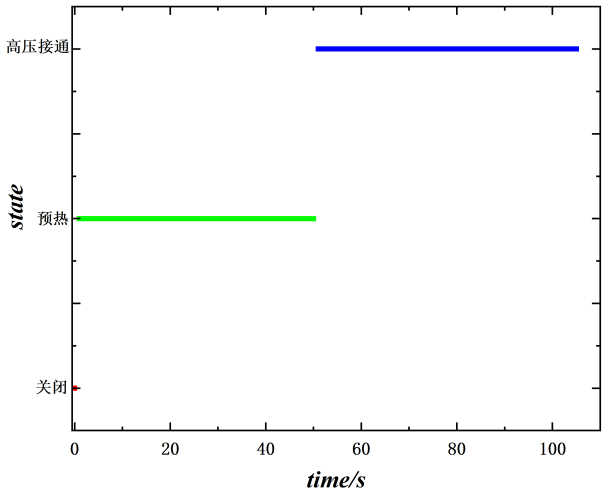

开始
off
cntry()
statehold(infinite);
rcccivc(cvLcidayurc)
current_radar=20; transition(yure);
receive(evLeidayure)
yure
cntry()
statehold(infinite);
rcccivc(cvGaoyajictong)
current_radar=30; transition(Gaoyajietong);
receive(evGaoyajietong)
Gaoyajietong
cntry()
statehold(infinite);

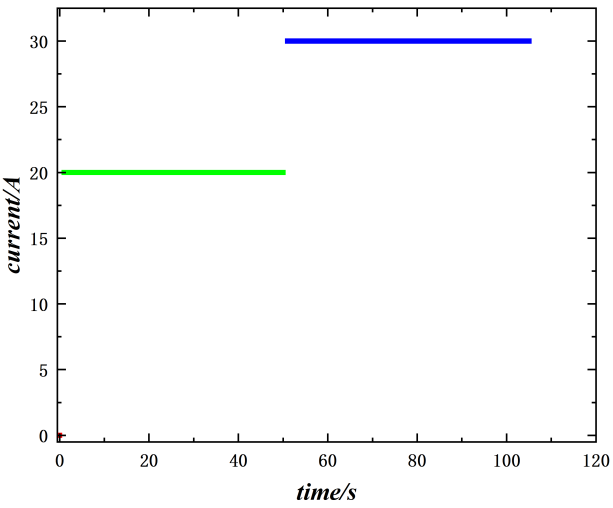
舵机


开启
关闭
state
0
20
40
60
80
100
time/s

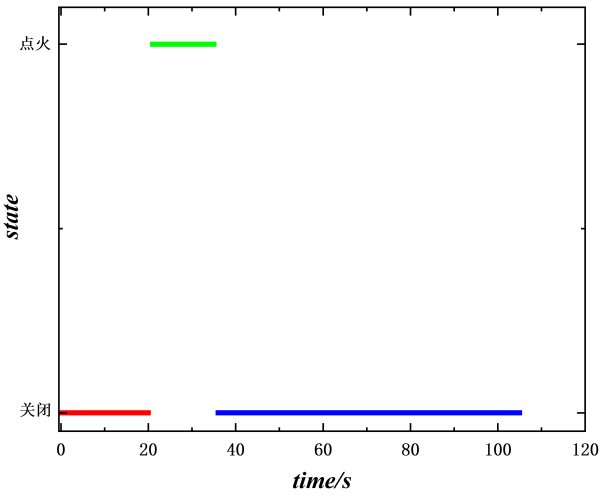

30
25
20
15
10
5
0
current/A
0
20
40
60
80
100
120
time/s

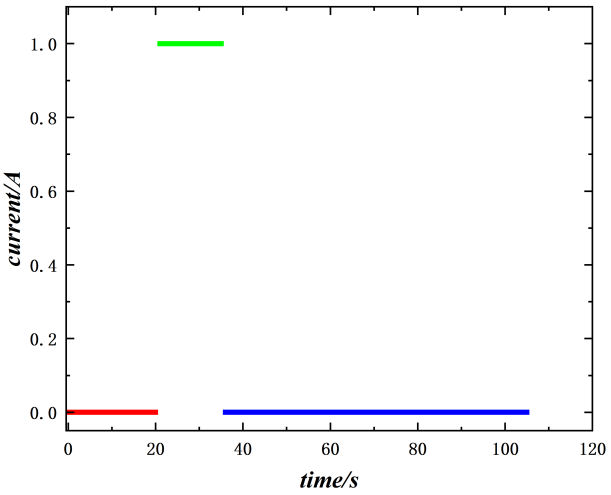
雷达


高压接通
预热
关闭
state
0
20
40
60
80
100
time/s

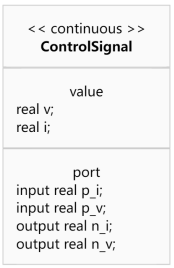

30
25
20
15
10
5
0
current/A
0
20
40
60
80
100
120
time/s

引信

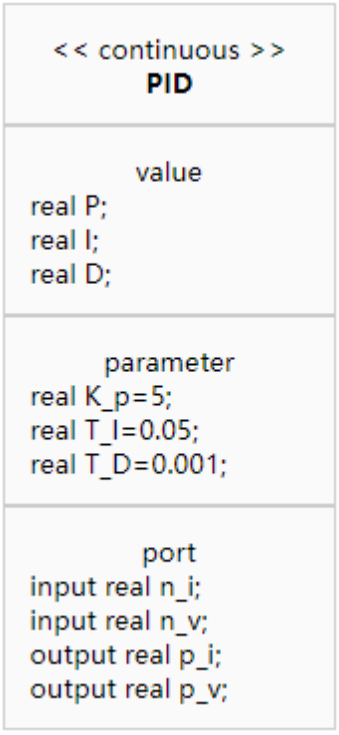

点火
关闭
state
0
20
40
60
80
100
120
time/s


1.0
0.8
0.6
0.4
0.2
0.0
current/A
0
20
40
60
80
100
120
time/s

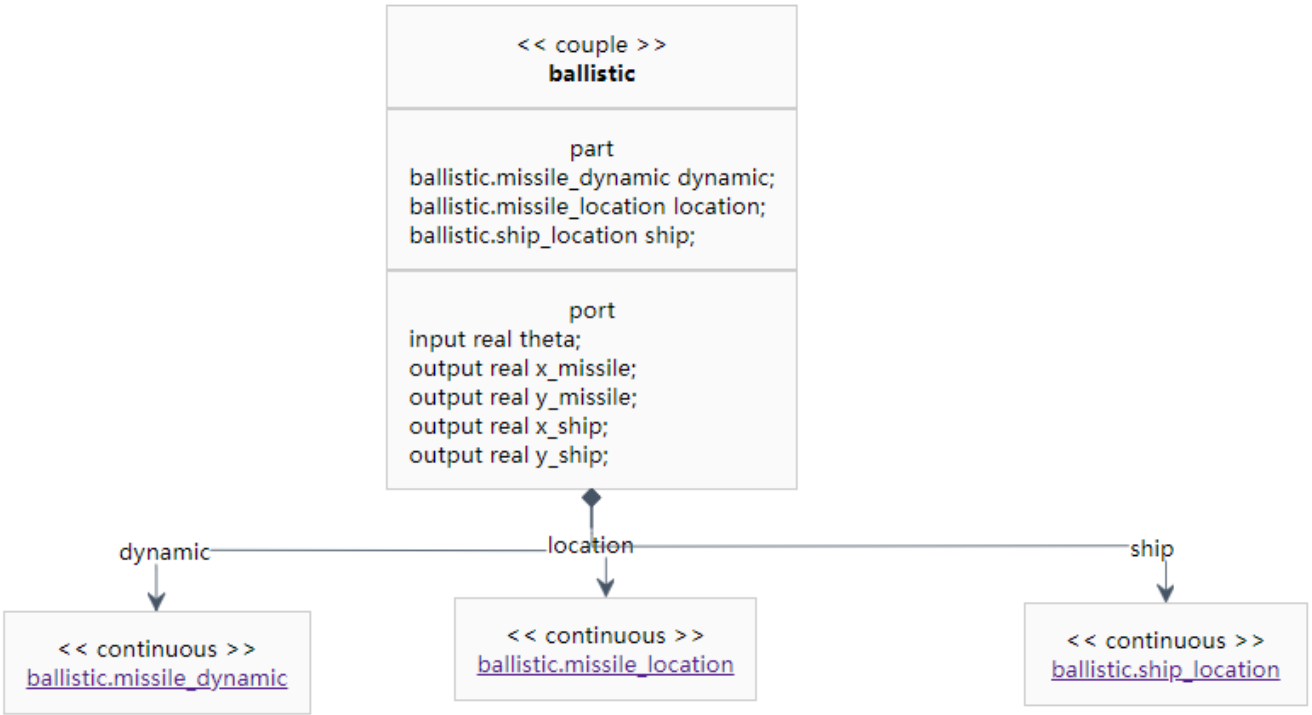
ControlSignal

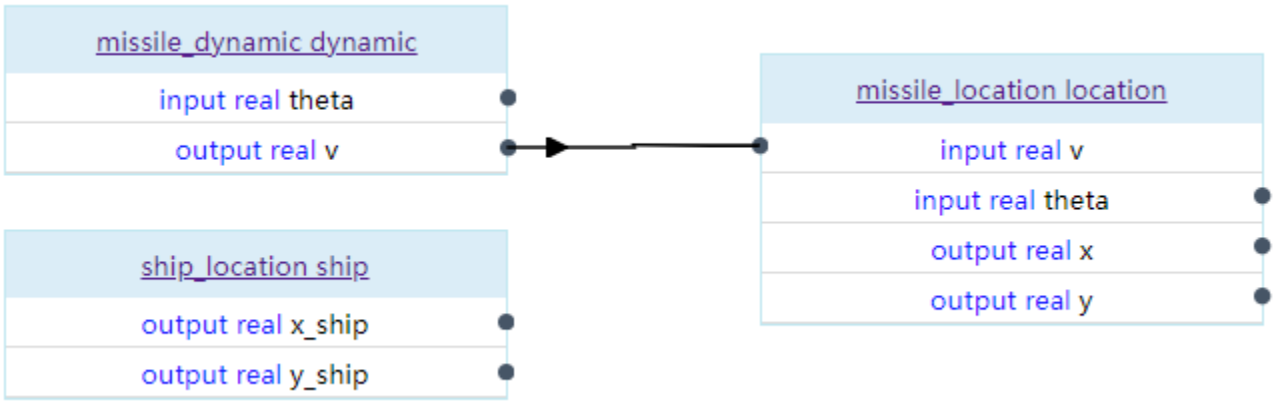


```
continuous ControlSignal
value:
   real v;
   flow real i;
port:
   input real n_i;
   input real n_v;
...
equation:
 ...
   if (n_v == 0) then       p_v=0.68*sin(2*pi*time);
   else if (n_v == 3) then
...
end;
```

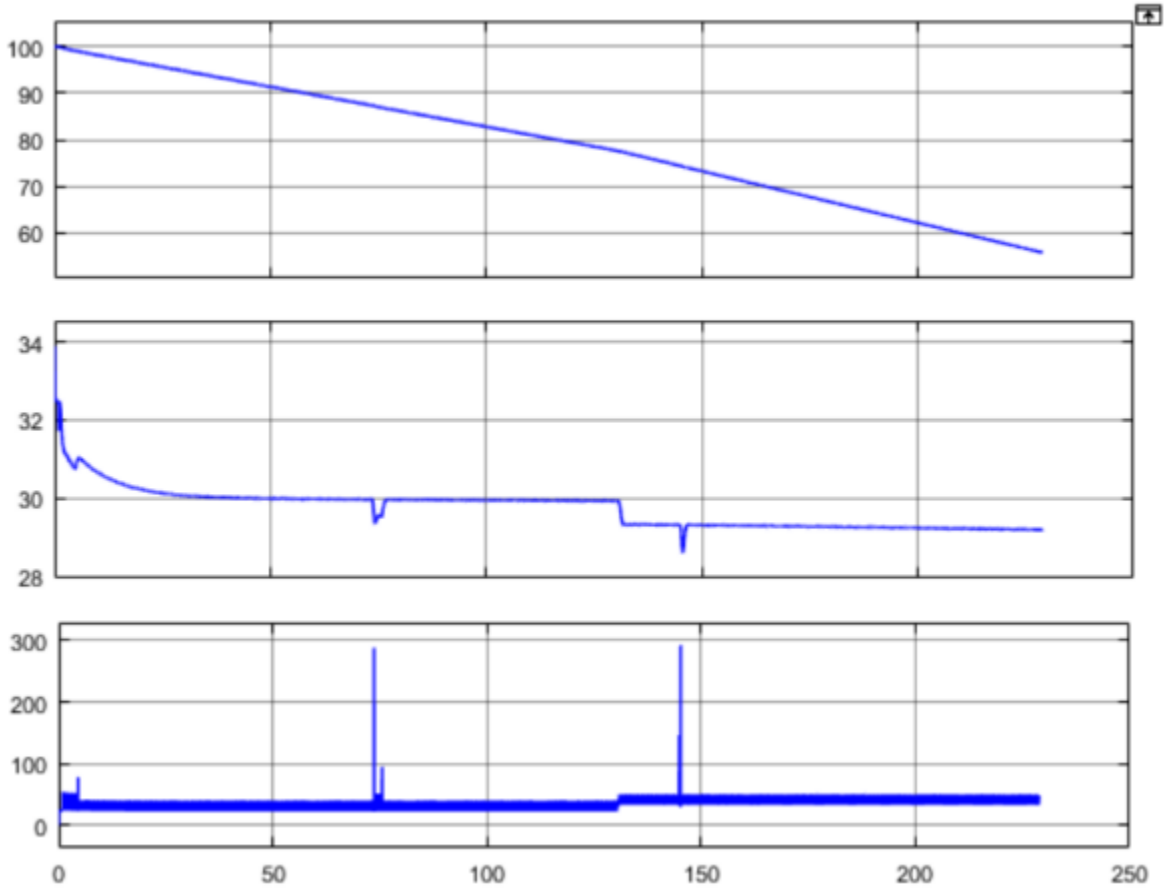

<< continuous >>
PID
value
real P;
real I;
real D;
parameter
real K_p=5;
real T_I=0.05;
real T_D=0.001;
port
input real n_i;
input real n_v;
output real p_i;
output real p_v;

```
continuous PID
parameter:
   real K_p=5;
   value:
   real P;
   real I;
   real D;
port:
   input real n_i;
....
  equation:
....
     p_v=K_p*(P+I+D);
 ....
end;
```


<< couple >>
ballistic
part
ballistic.missile_dynamic dynamic;
ballistic.missile_location location;
ballistic.ship_location ship;
port
input real theta;
output real x_missile;
output real y_missile;
output real x_ship;
output real y_ship;
dynamic
location
ship
<< continuous >>
ballistic.missile_dynamic
<< continuous >>
ballistic.missile_location
<< continuous >>
ballistic.ship_location


missile_dynamic dynamic
input real theta
output real v
ship_location ship
output real x_ship
output real y_ship
missile_location location
input real v
input real theta
output real x
output real y

```
couple ballistic
import ballistic.missile_dynamic as dynamic;
import ballistic.missile_location as location;
import ballistic.ship_location as ship;
port:
input real theta;
output real x_missile;
output real y_missile;
output real x_ship;
output real y_ship;
part:
  dynamic dynamic;
  location location;
  ship ship;
connection:
  connect(dynamic.v,location.v);
end;
```


100
90
80
70
60
34
32
30
28
300
200
100
0
0
50
100
150
200
250