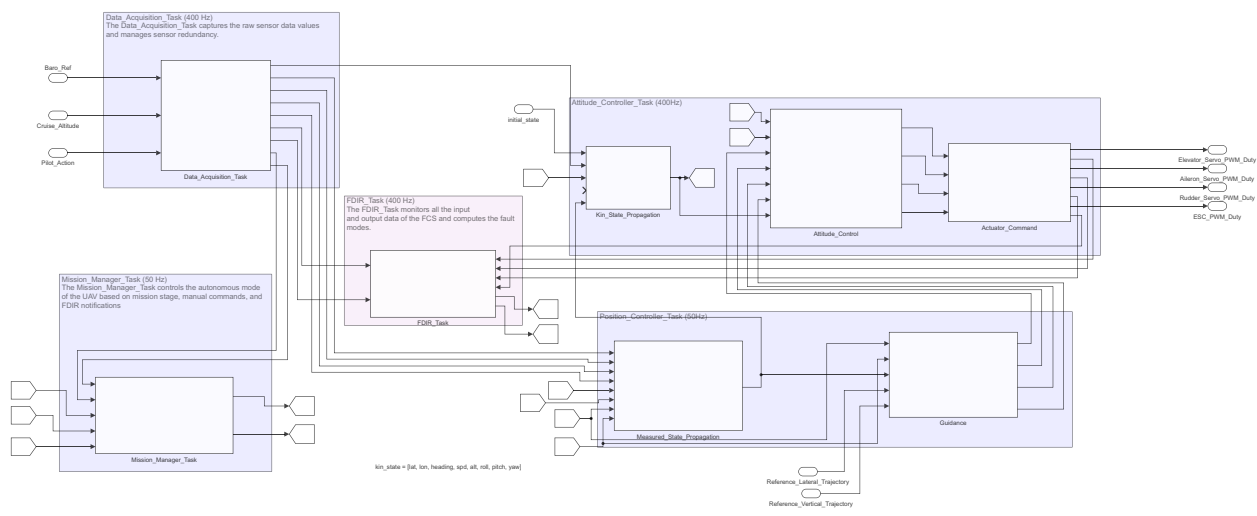
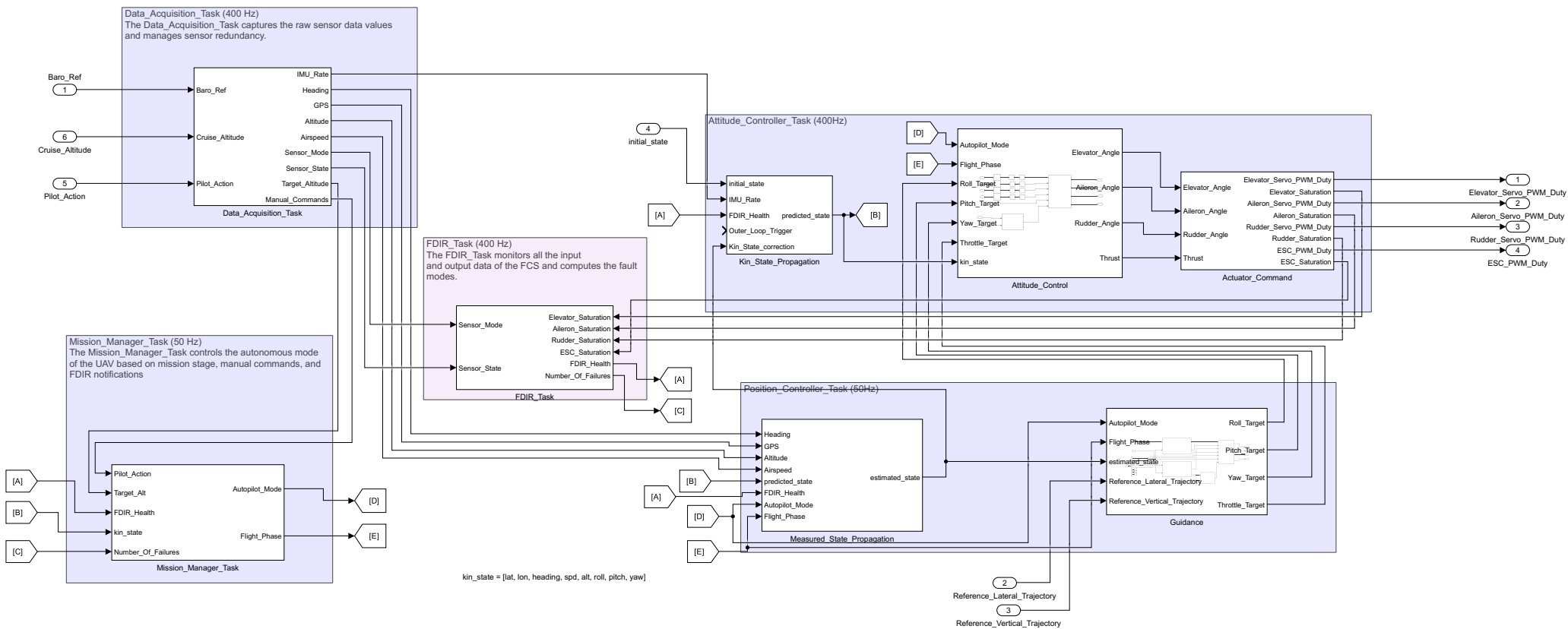


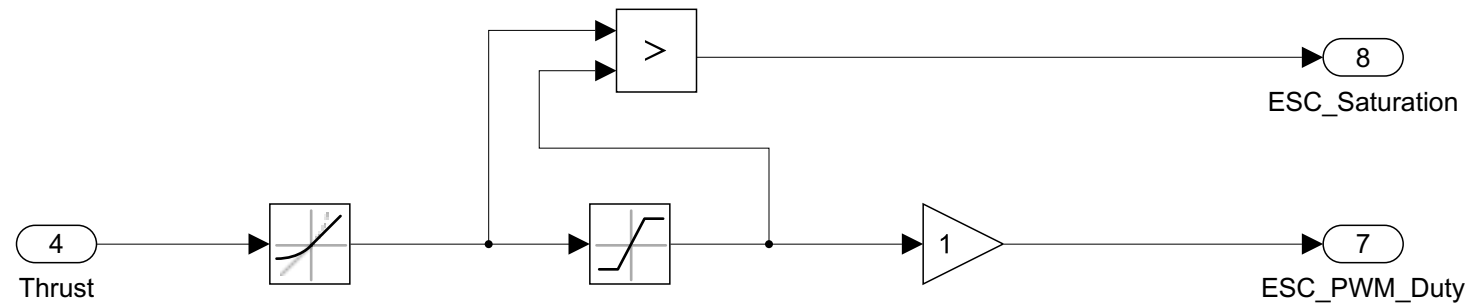
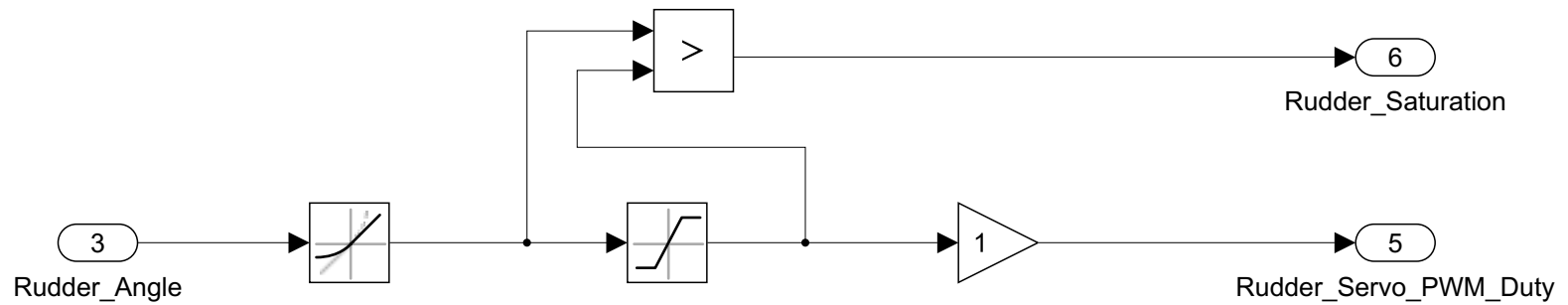
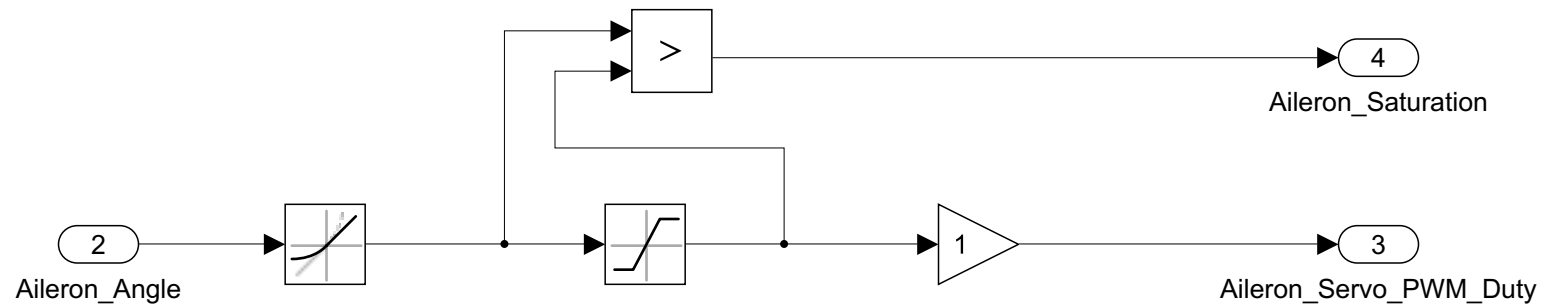
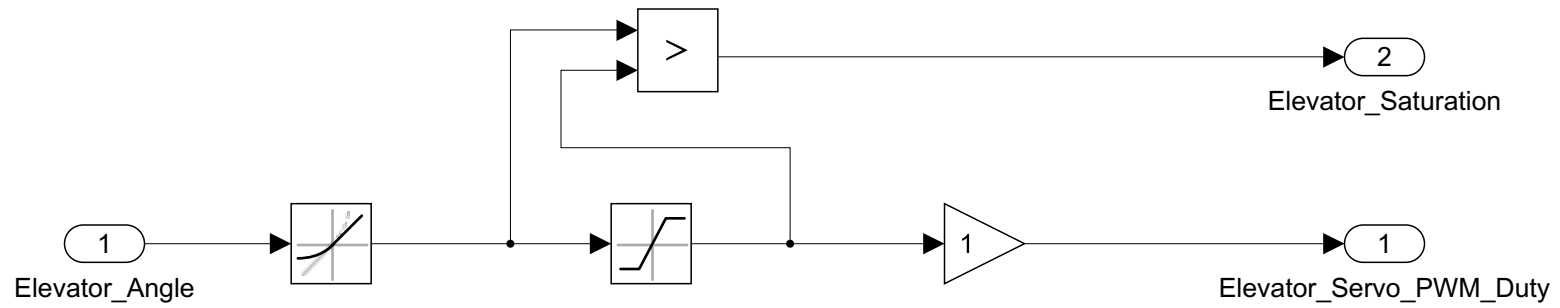
- > Baro_Ref
- > Reference_Lateral_Trajectory
- > Reference_Vertical_Trajectory
- > initial_state
- > Pilot_Action
- > Cruise_Altitude

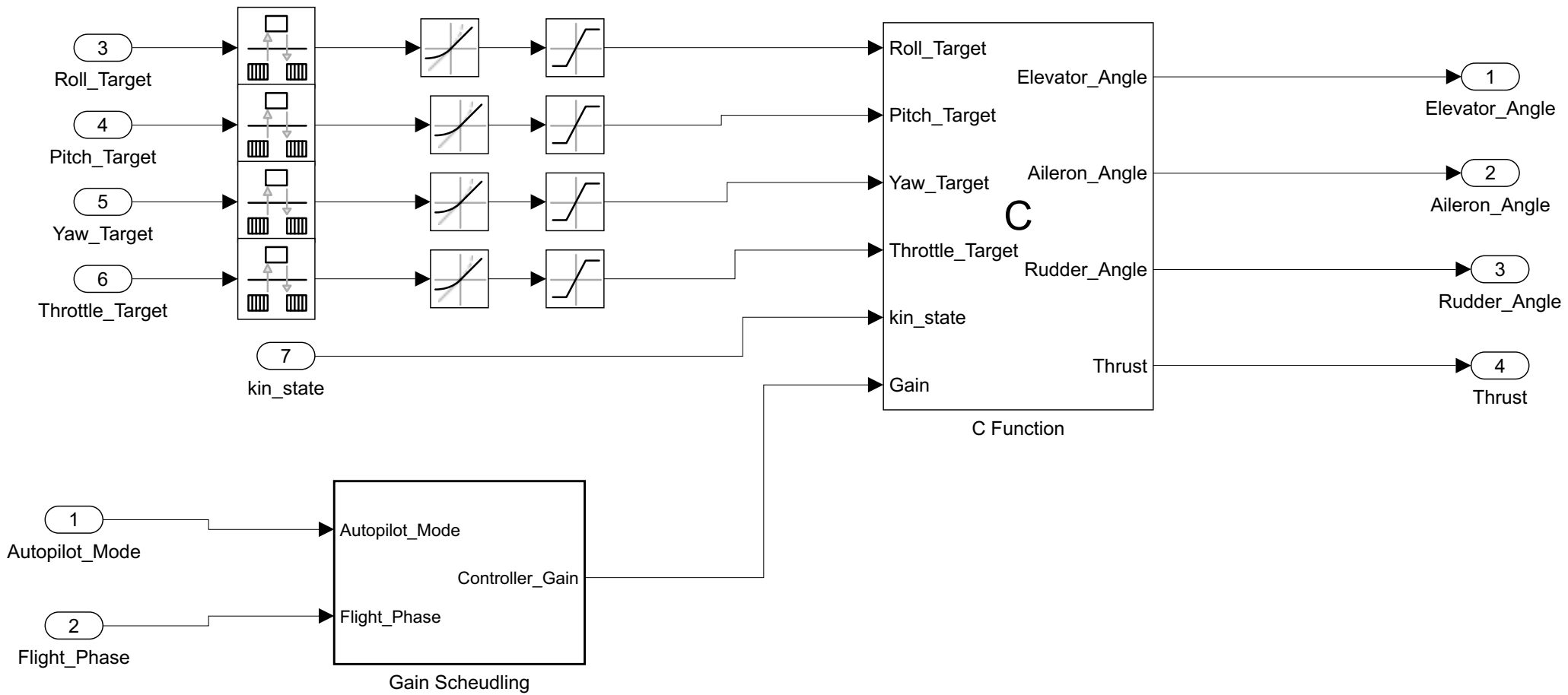


- Elevator_Servo_PWM_Duty >
- Aileron_Servo_PWM_Duty >
- Rudder_Servo_PWM_Duty >
- ESC_PWM_Duty >

FCS



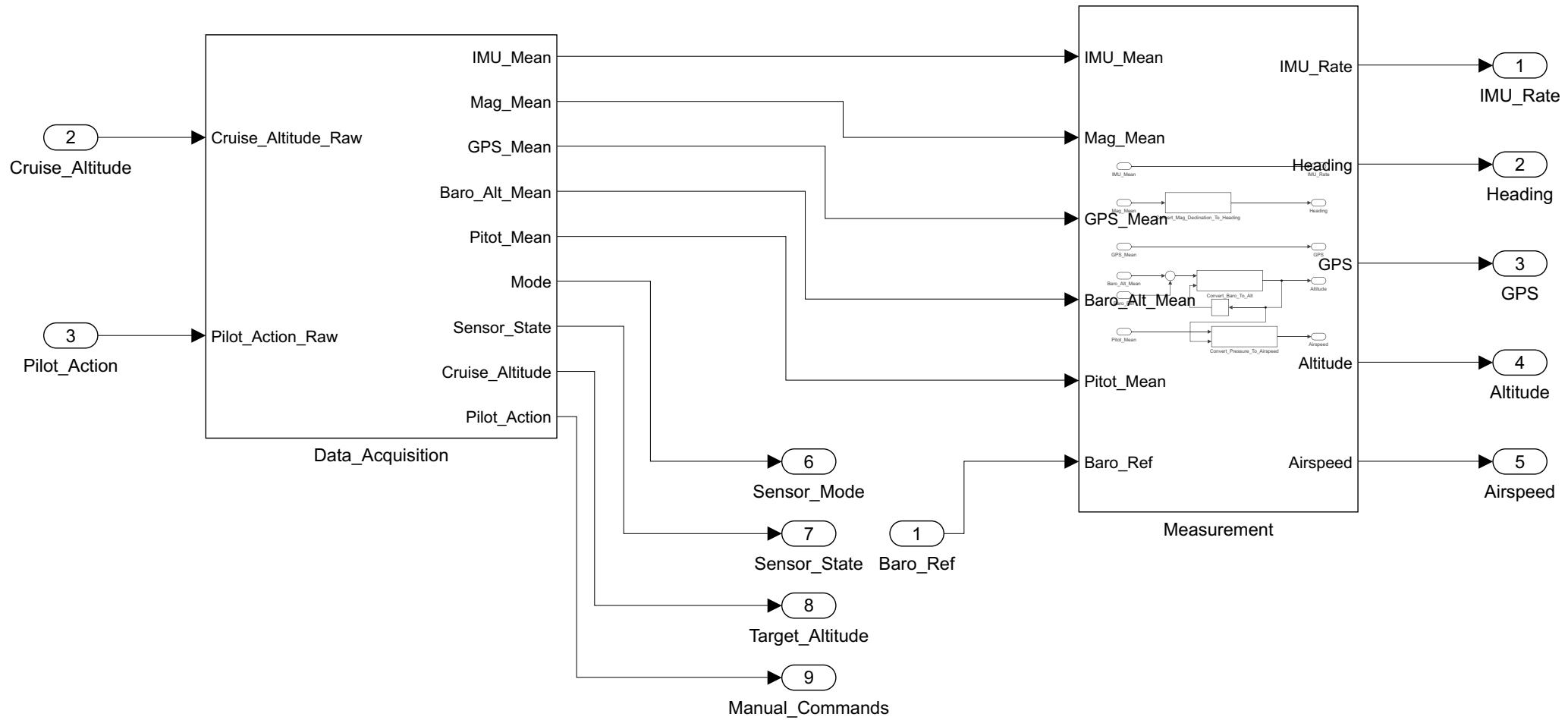


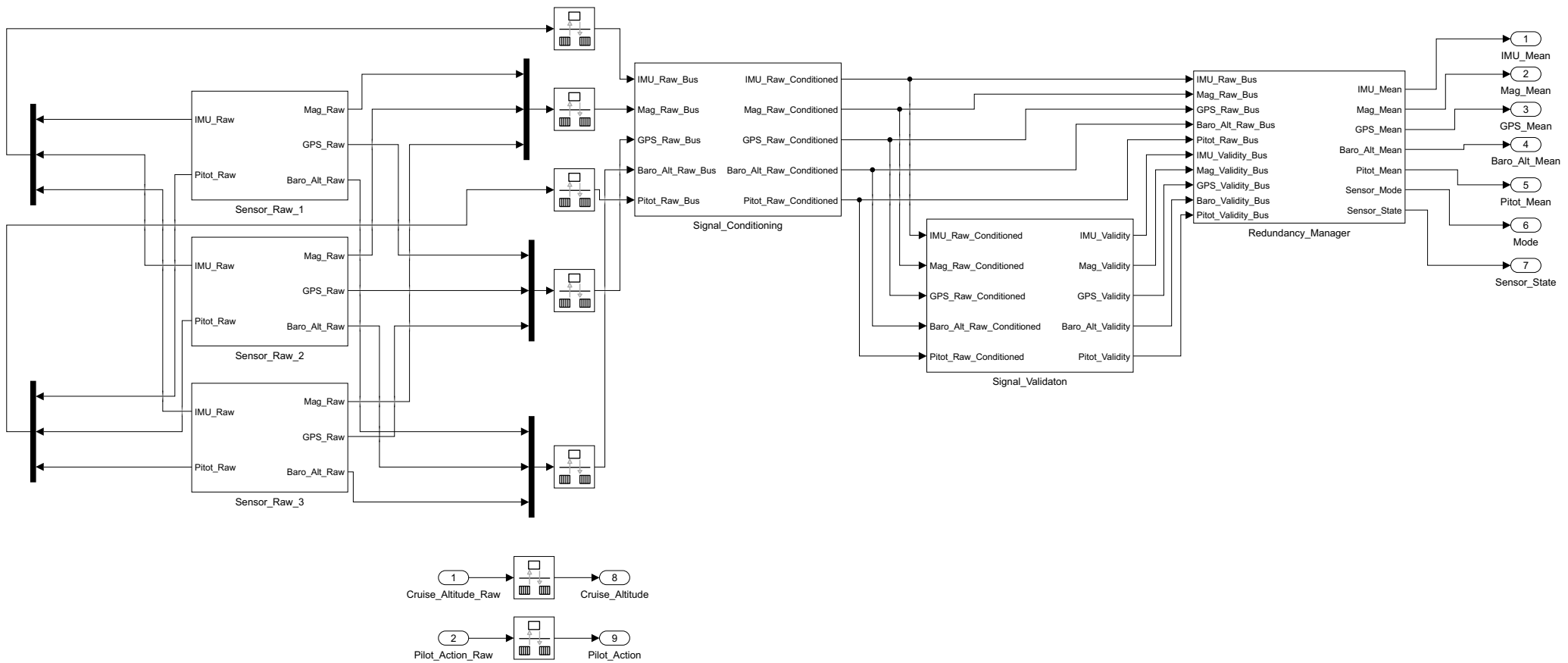


1
Autopilot_Mode

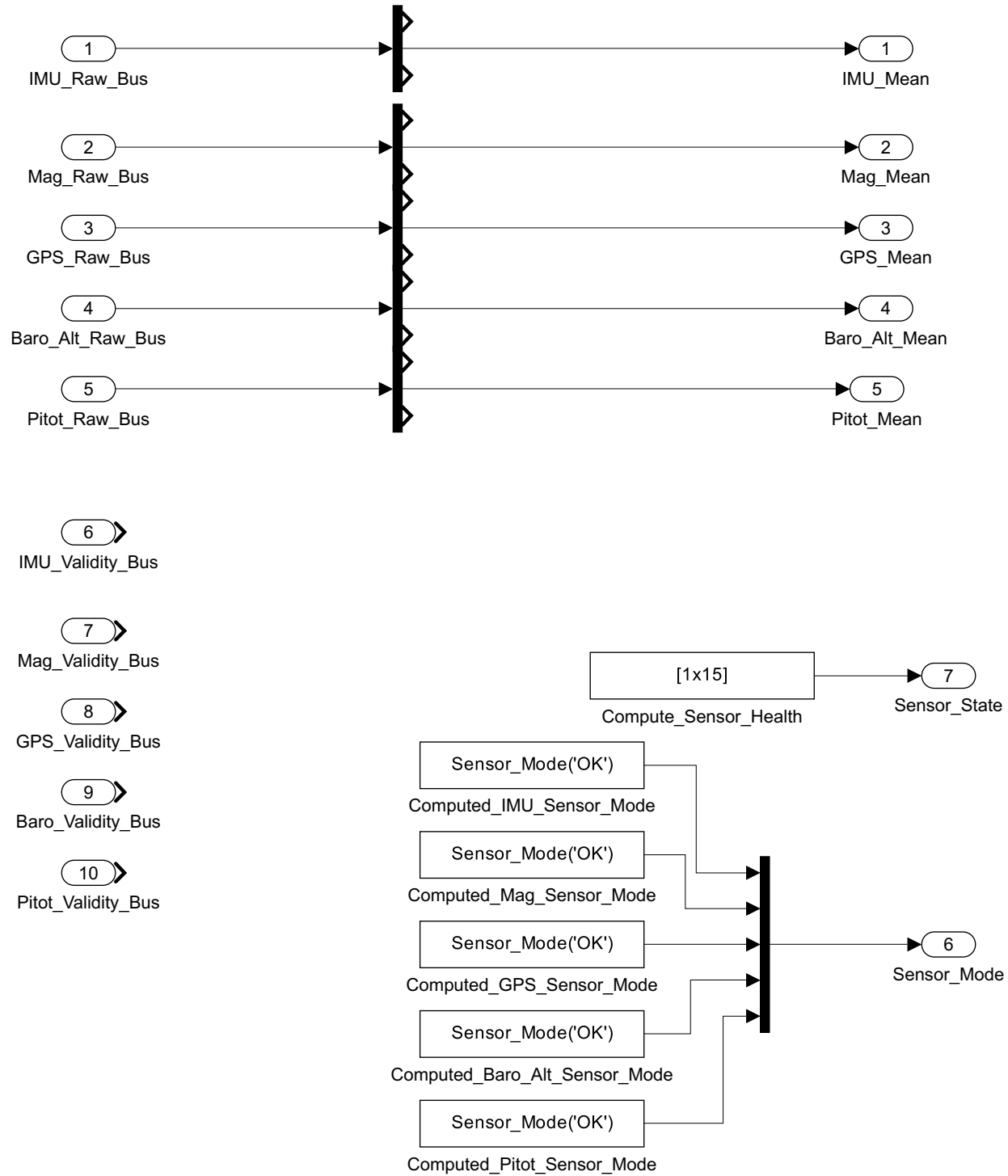
2
Flight_Phase

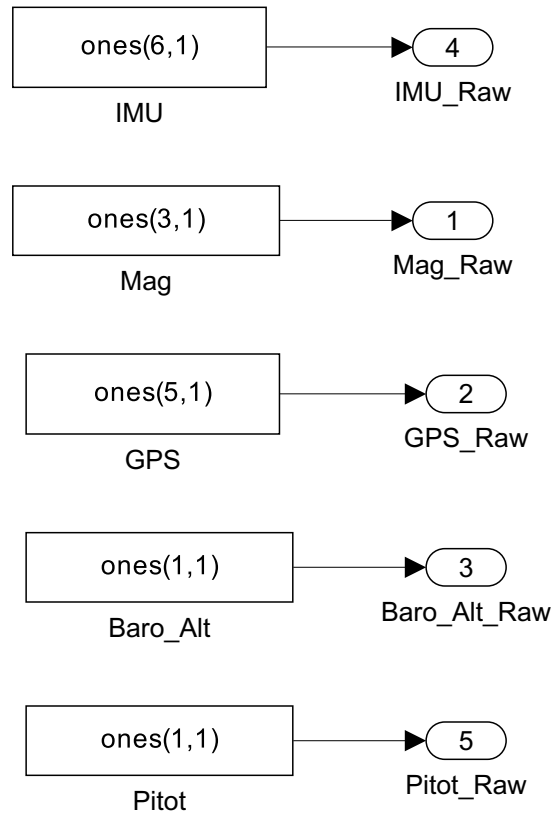
X 1
Controller_Gain

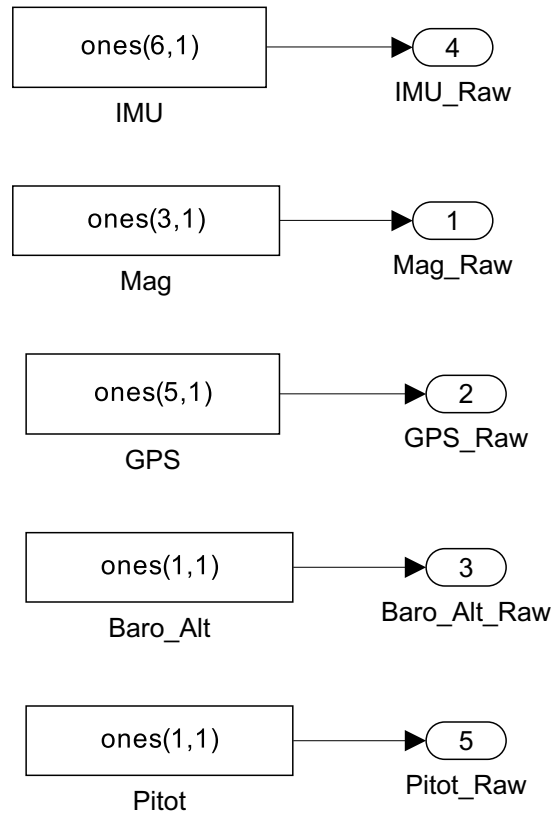


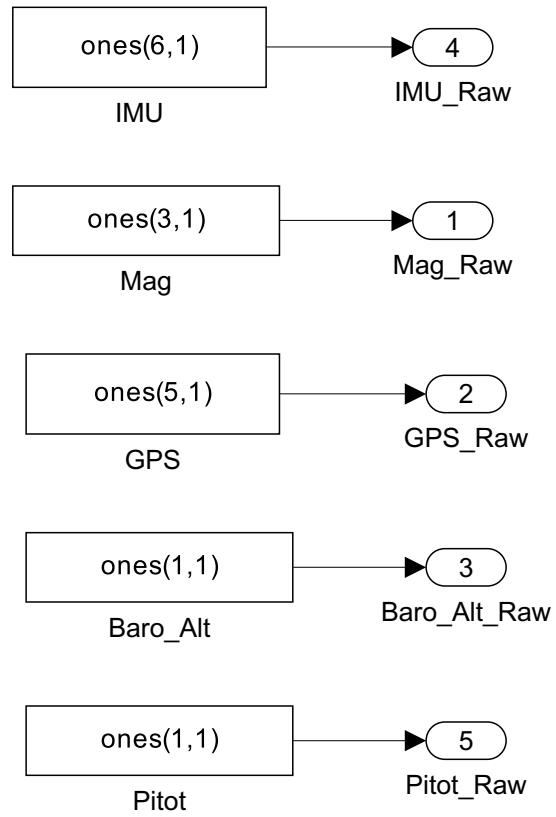


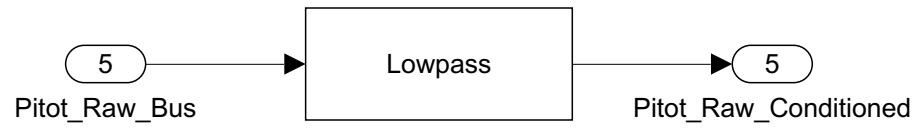
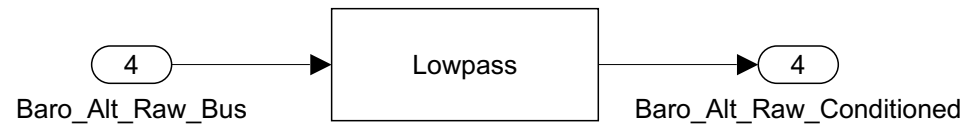
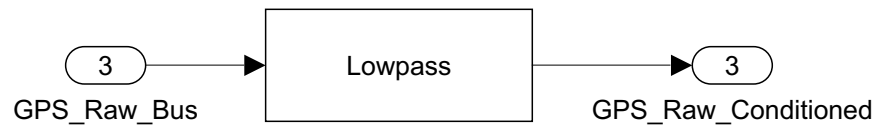
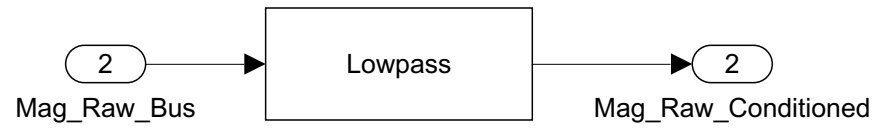
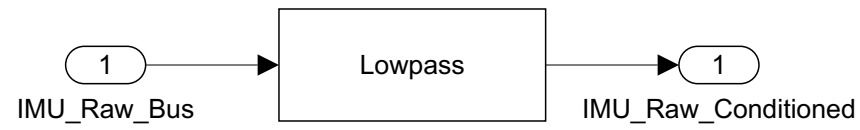
Dummy_Implementation

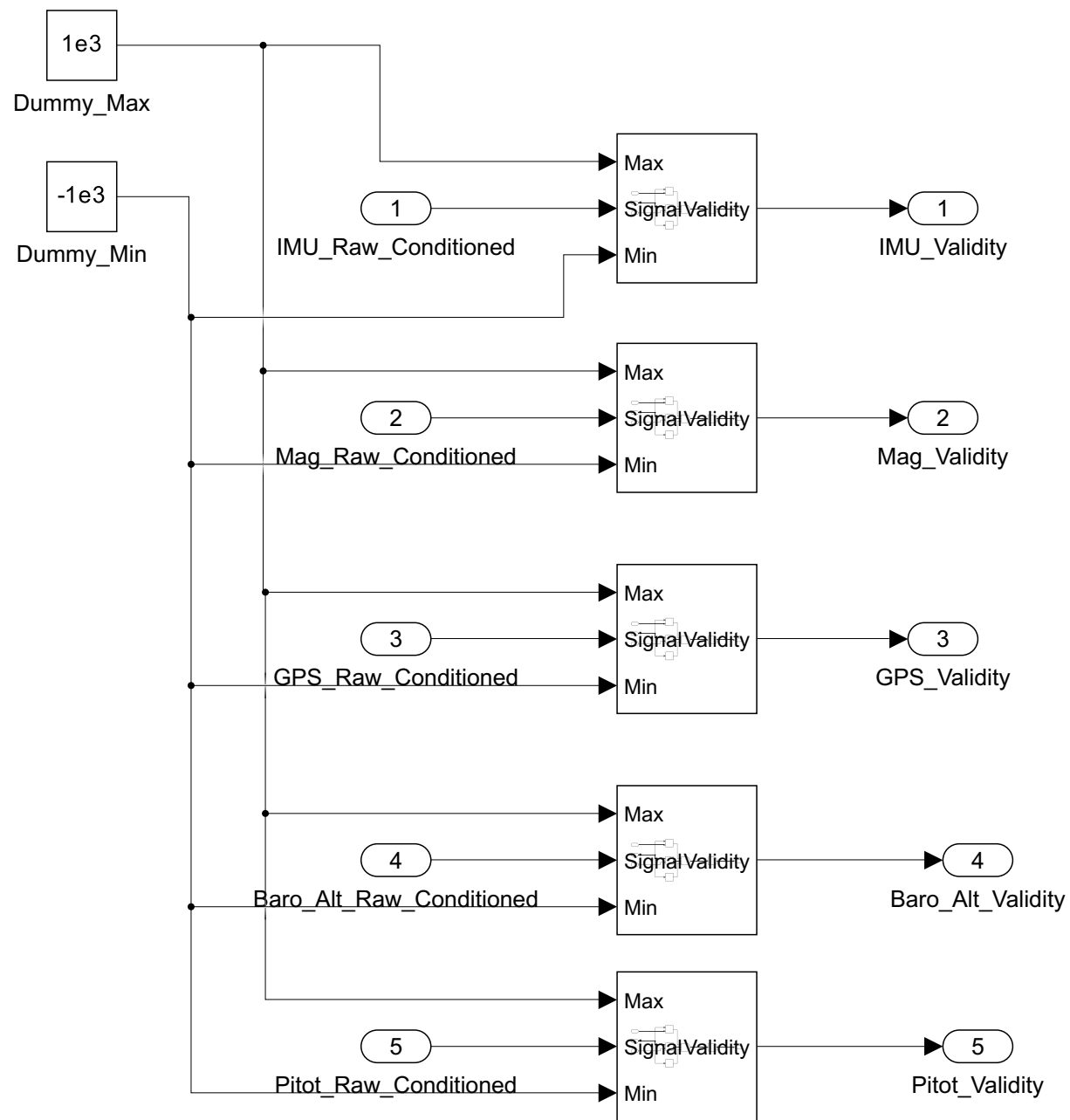


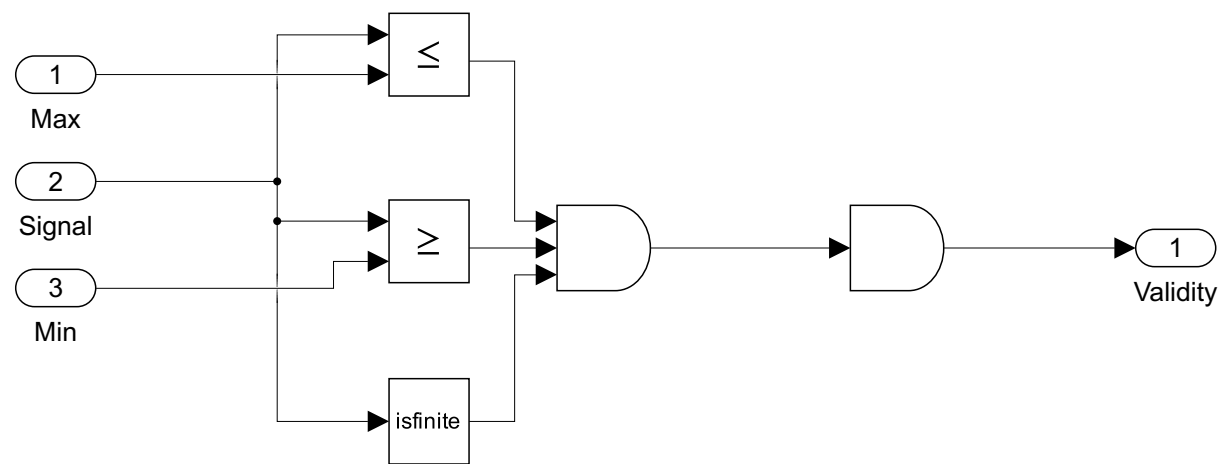


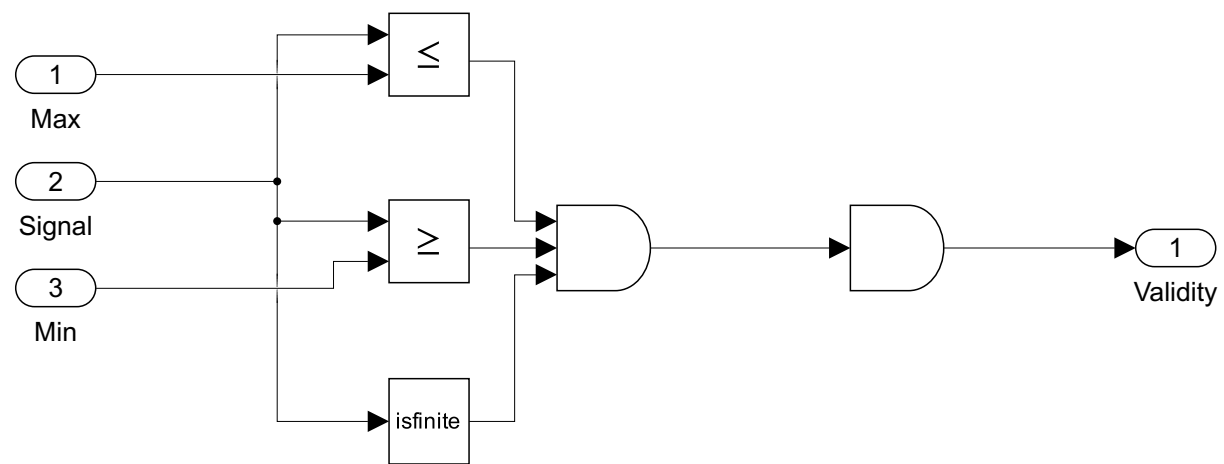


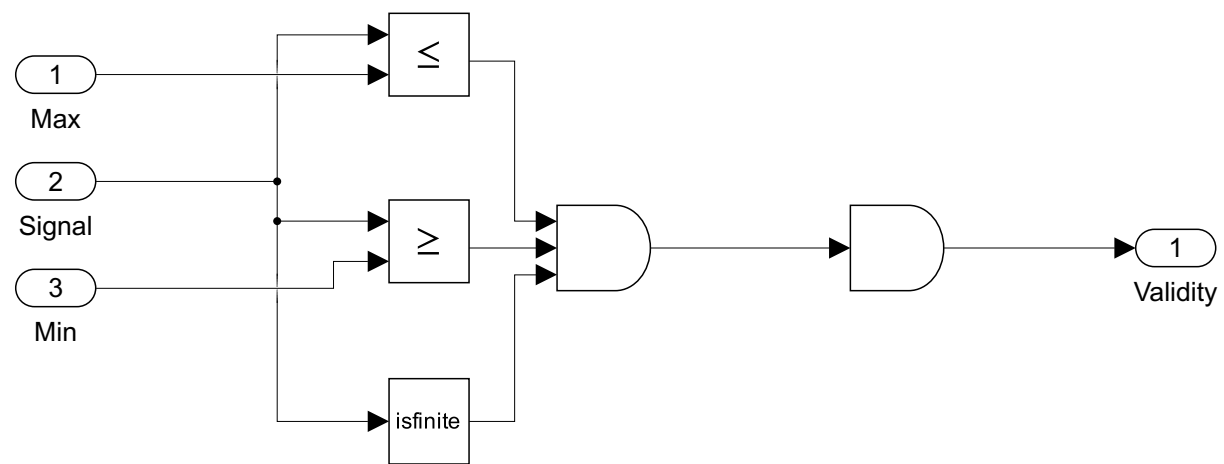


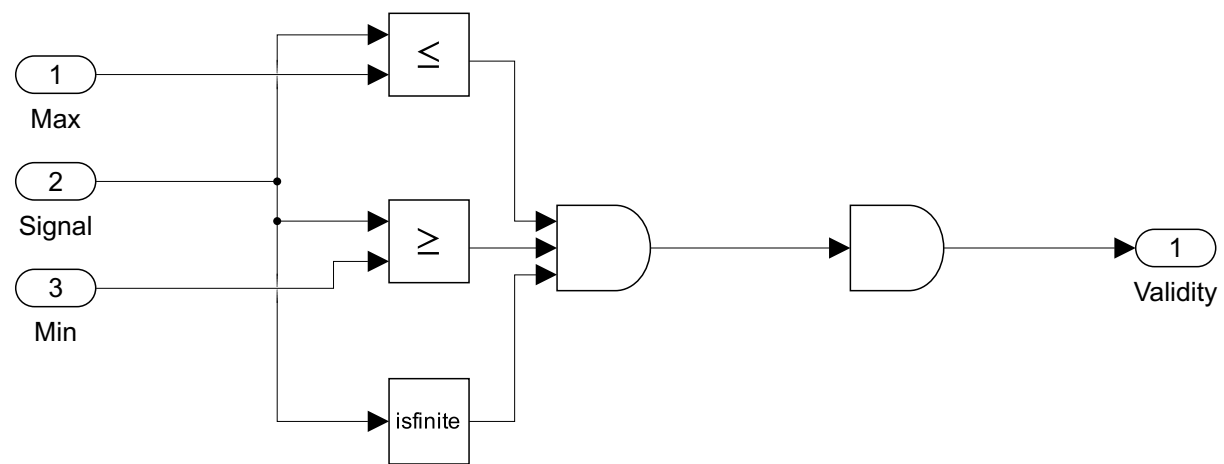


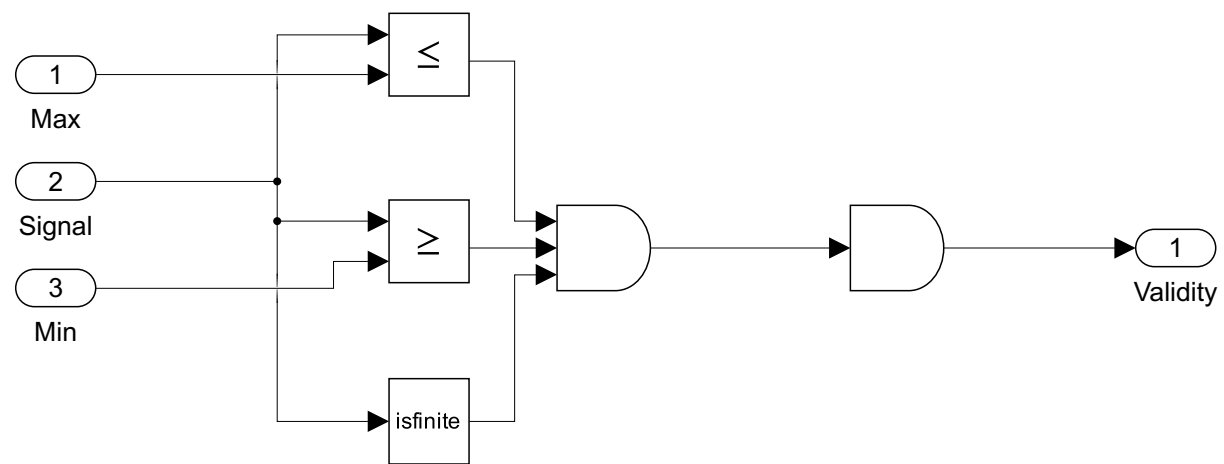


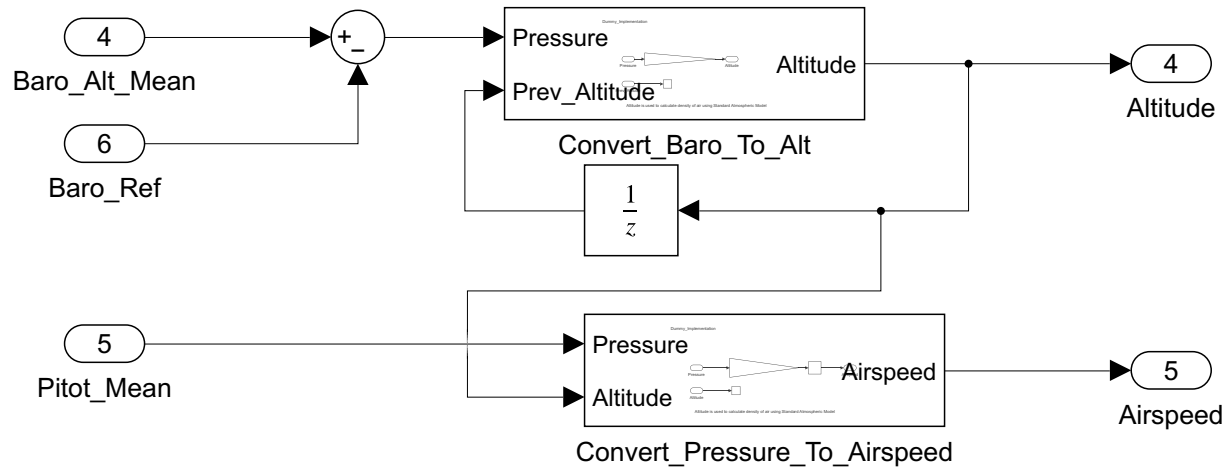
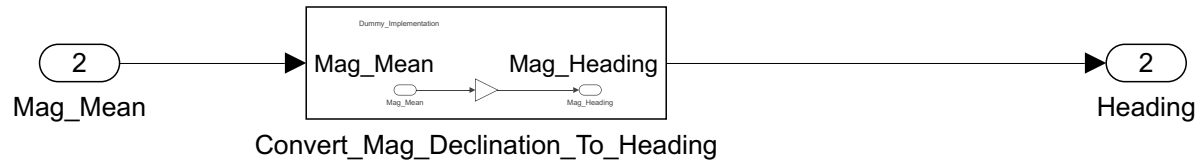




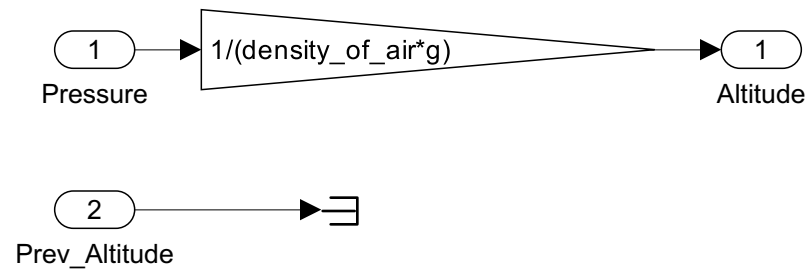






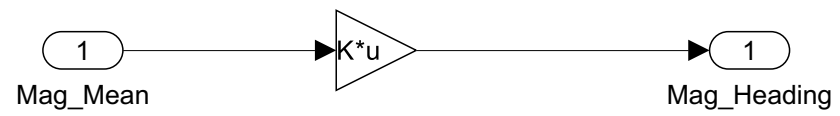


Dummy_Implementation

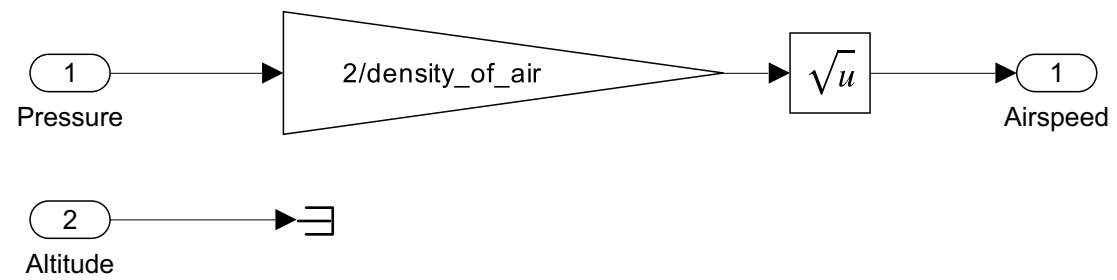


Altitude is used to calculate density of air using Standard Atmospheric Model

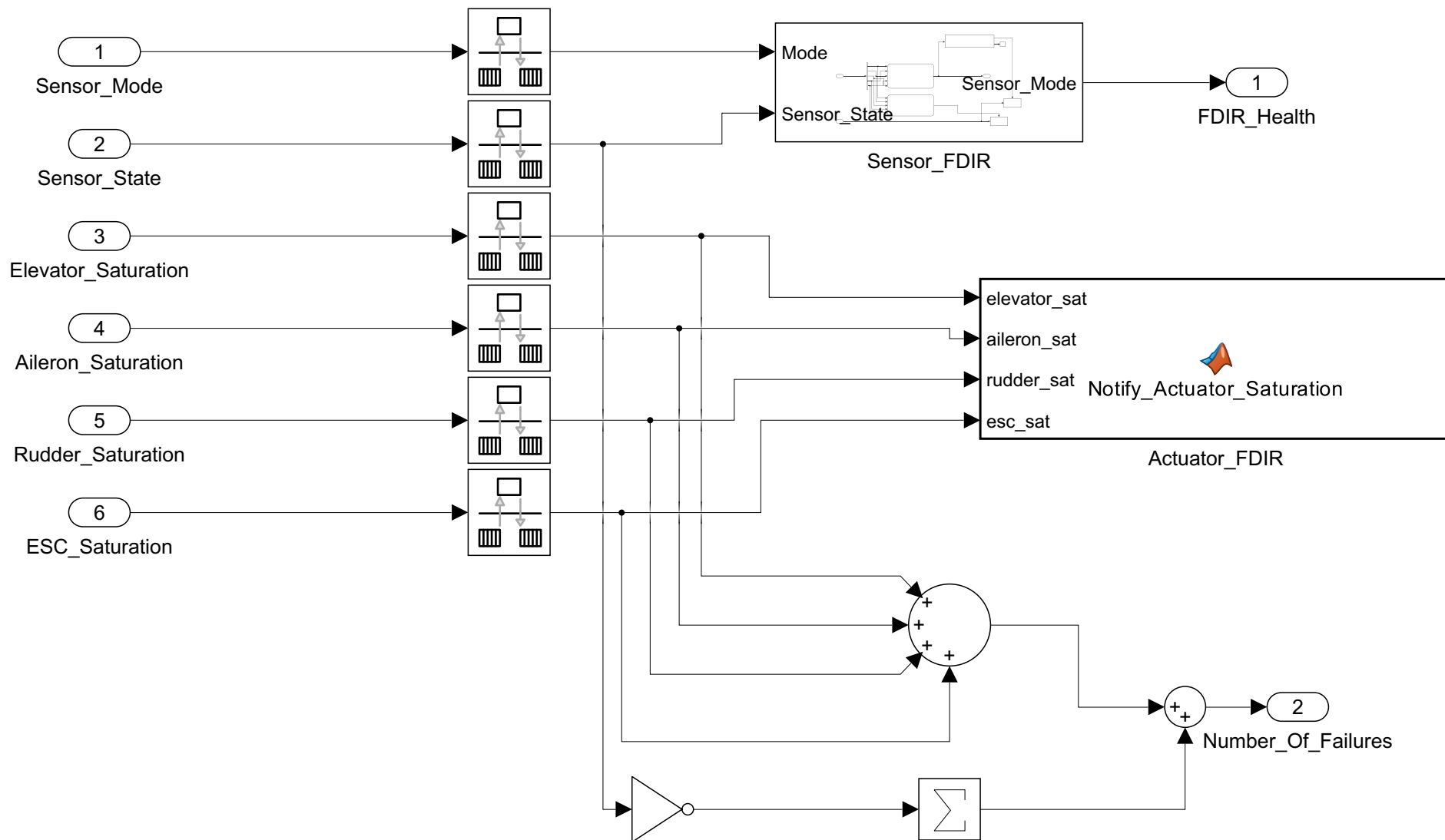
Dummy_Implementation



Dummy_Implementation



Altitude is used to calculate density of air using Standard Atmospheric Model



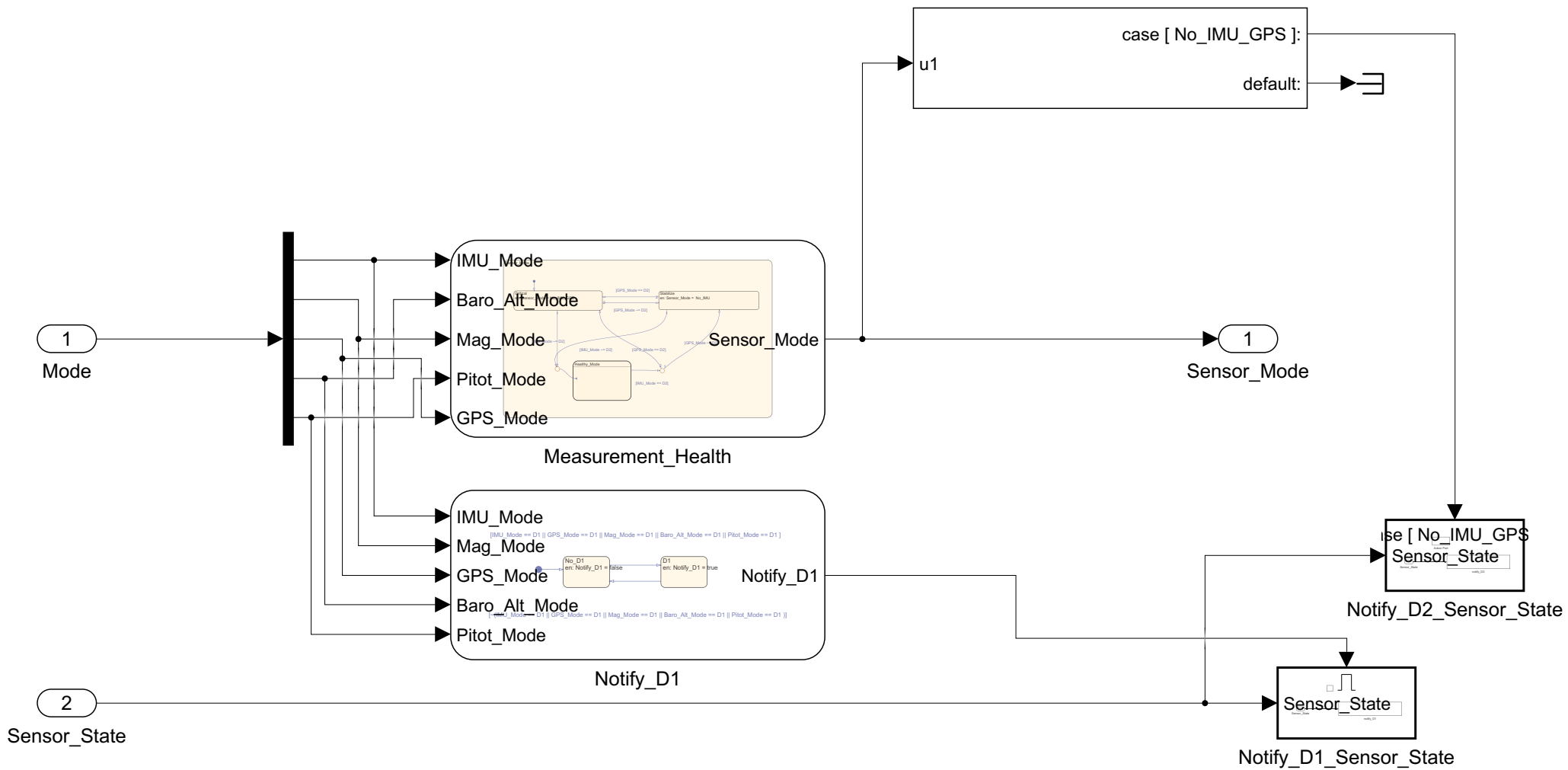
```
function Notify_Actuator_Saturation(elevator_sat, aileron_sat, rudder_sat, esc_sat)
    if elevator_sat
        disp('Elevator Servo saturated')
    end

    if aileron_sat
        disp('Aileron Servo saturated')
    end

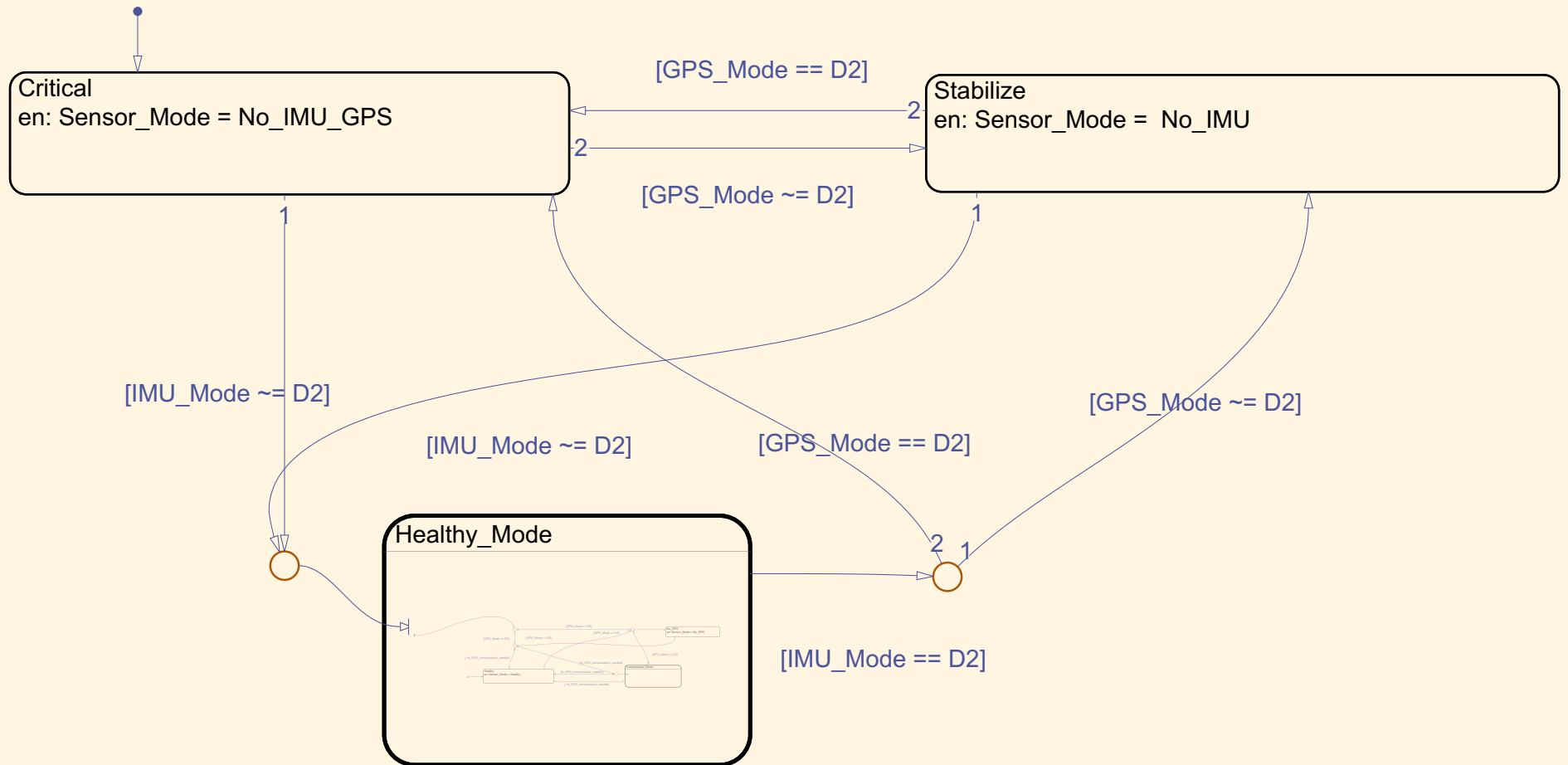
    if rudder_sat
        disp('Rudder Servo saturated')
    end

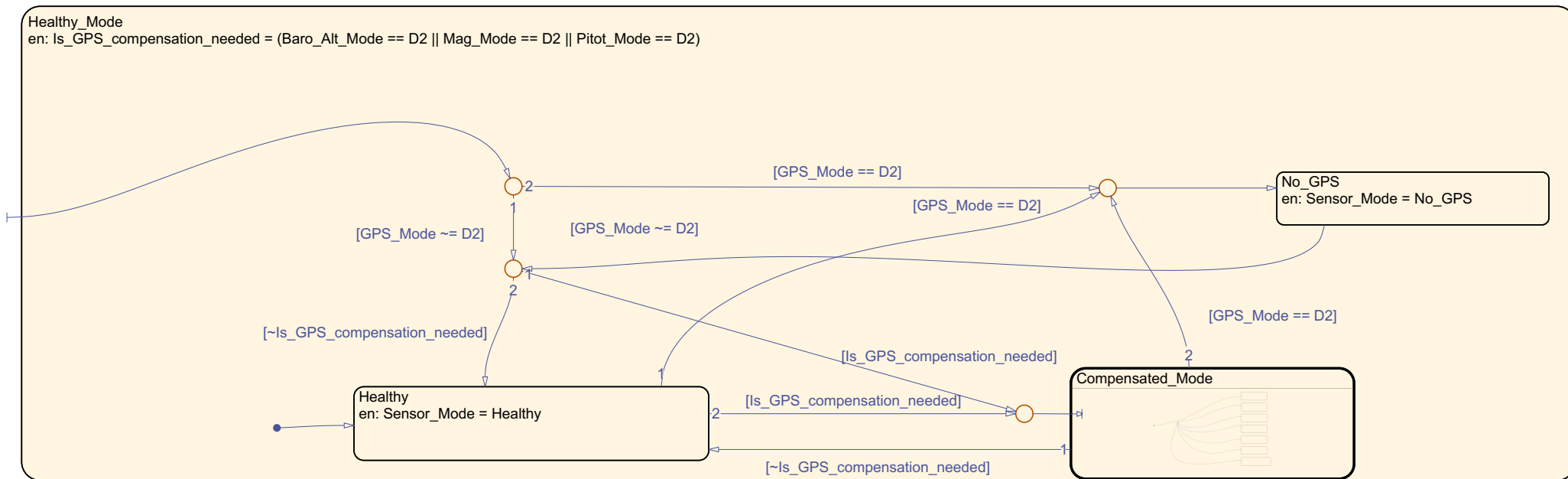
    if esc_sat
        disp('ESC saturated')
    end
end

end
```



Sensor_Mode





Compensated_Mode

[Baro_Alt_Mode == D2 && Pitot_Mode ~= D2 && Mag_Mode ~= D2]

[Baro_Alt_Mode ~= D2 && Pitot_Mode ~= D2 && Mag_Mode == D2]

[Baro_Alt_Mode ~= D2 && Pitot_Mode == D2 && Mag_Mode ~= D2]

[Baro_Alt_Mode == D2 && Pitot_Mode == D2 && Mag_Mode ~= D2]

[Baro_Alt_Mode ~= D2 && Pitot_Mode == D2 && Mag_Mode == D2]

[Baro_Alt_Mode == D2 && Pitot_Mode ~= D2 && Mag_Mode == D2]

[Baro_Alt_Mode == D2 && Pitot_Mode == D2 && Mag_Mode == D2]

GPS_Alt
en: Sensor_Mode = GPS_Alt

GPS_Hdg
en: Sensor_Mode = GPS_Hdg

GPS_Spd
en: Sensor_Mode = GPS_Spd

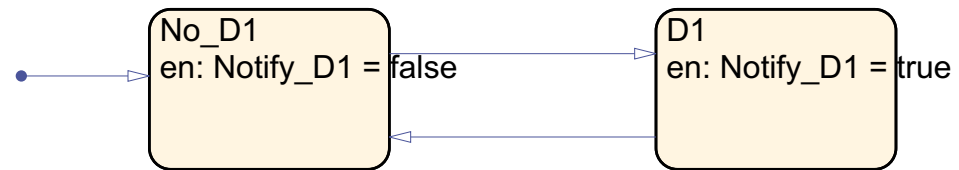
GPS_Alt_Spd
en: Sensor_Mode = GPS_Alt_Spd

GPS_Spd_Hdg
en: Sensor_Mode = GPS_Spd_Hdg

GPS_Alt_Hdg
en: Sensor_Mode = GPS_Alt_Hdg

GPS_Compensated
en: Sensor_Mode = GPS_Compensated

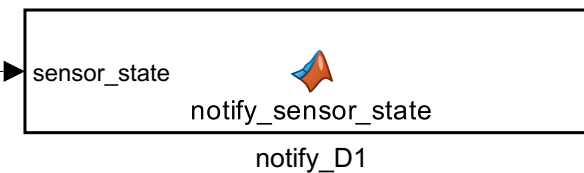
[IMU_Mode == D1 || GPS_Mode == D1 || Mag_Mode == D1 || Baro_Alt_Mode == D1 || Pitot_Mode == D1]



[~(IMU_Mode == D1 || GPS_Mode == D1 || Mag_Mode == D1 || Baro_Alt_Mode == D1 || Pitot_Mode == D1)]

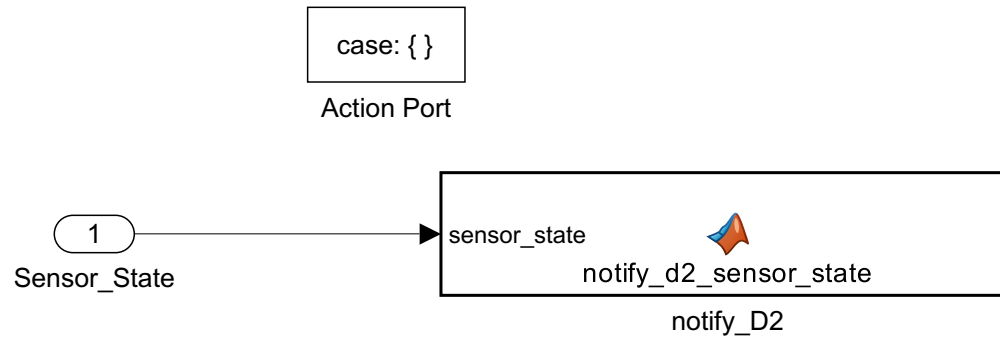


1
Sensor_State



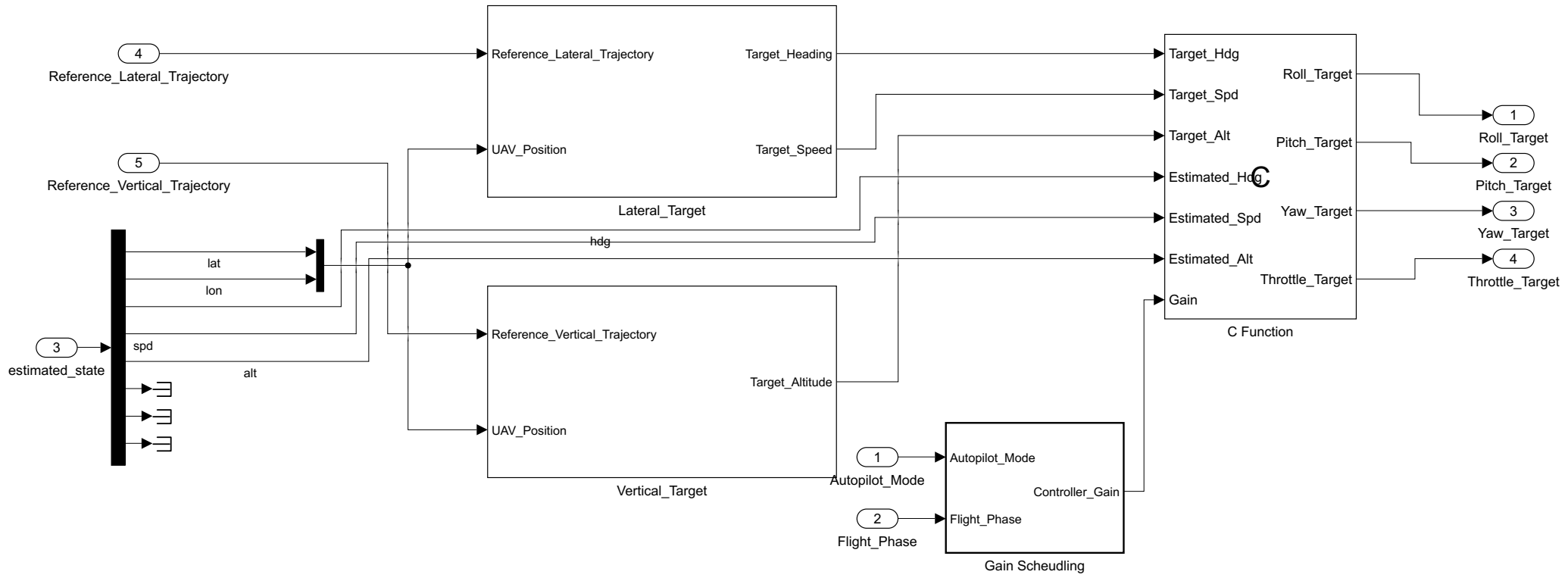
```
function notify_sensor_state(sensor_state)

    for i = 1:length(sensor_state)
        if sensor_state(i) ~= true
            if i >= 1 && i <= 3
                disp('IMU_%d degraded', i)
            elseif i >= 4 && i <= 6
                disp('Mag_%d degraded', i)
            elseif i >= 7 && i <= 9
                disp('GPS_%d degraded', i)
            elseif i >= 10 && i <= 12
                disp('Baro_Alt_%d degraded', i)
            else
                disp('Pitot_%d degraded', i)
            end
        end
    end
end
```



```
function notify_d2_sensor_state(sensor_state)

    for i = 1:3:length(sensor_state)
        if sensor_state(i) ~= true
            if i == 1
                disp('All IMU degraded')
            elseif i == 4
                disp('All Mag degraded')
            elseif i == 7
                disp('All GPS degraded')
            elseif i == 10
                disp('All Baro_Alt degraded')
            else
                disp('All Pitot degraded')
            end
        end
    end
end
```



1
Autopilot_Mode

2
Flight_Phase

X 1
Controller_Gain

1
Reference_Lateral_Trajectory

2
UAV_Position

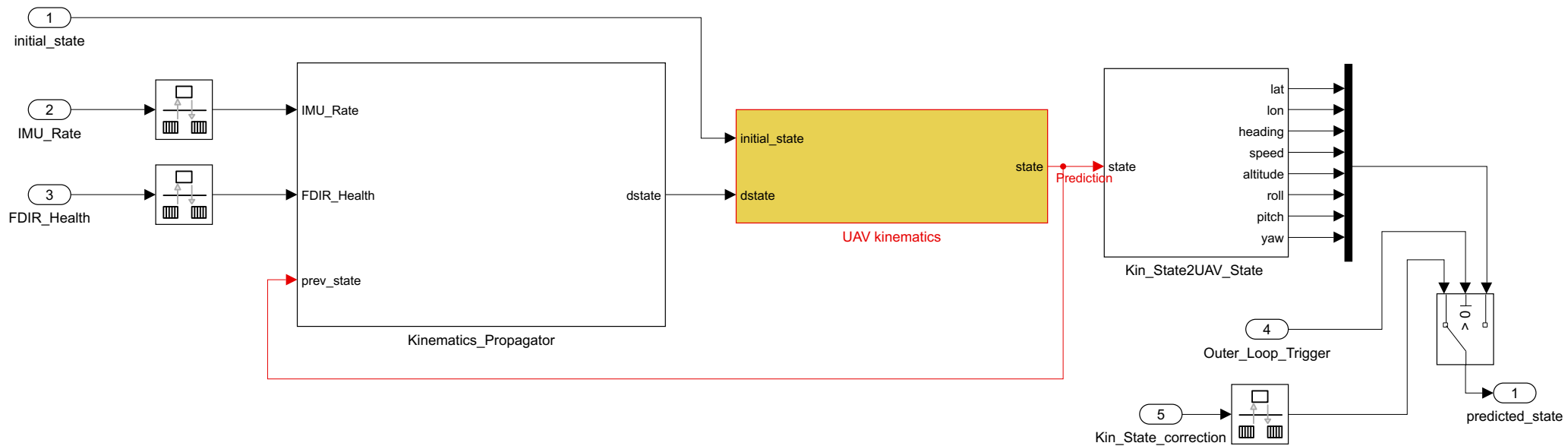
1
Target_Heading

2
Target_Speed

1
Reference_Vertical_Trajectory

X 1
Target_Altitude

2
UAV_Position



1
state

1
lat

2
lon

3
heading

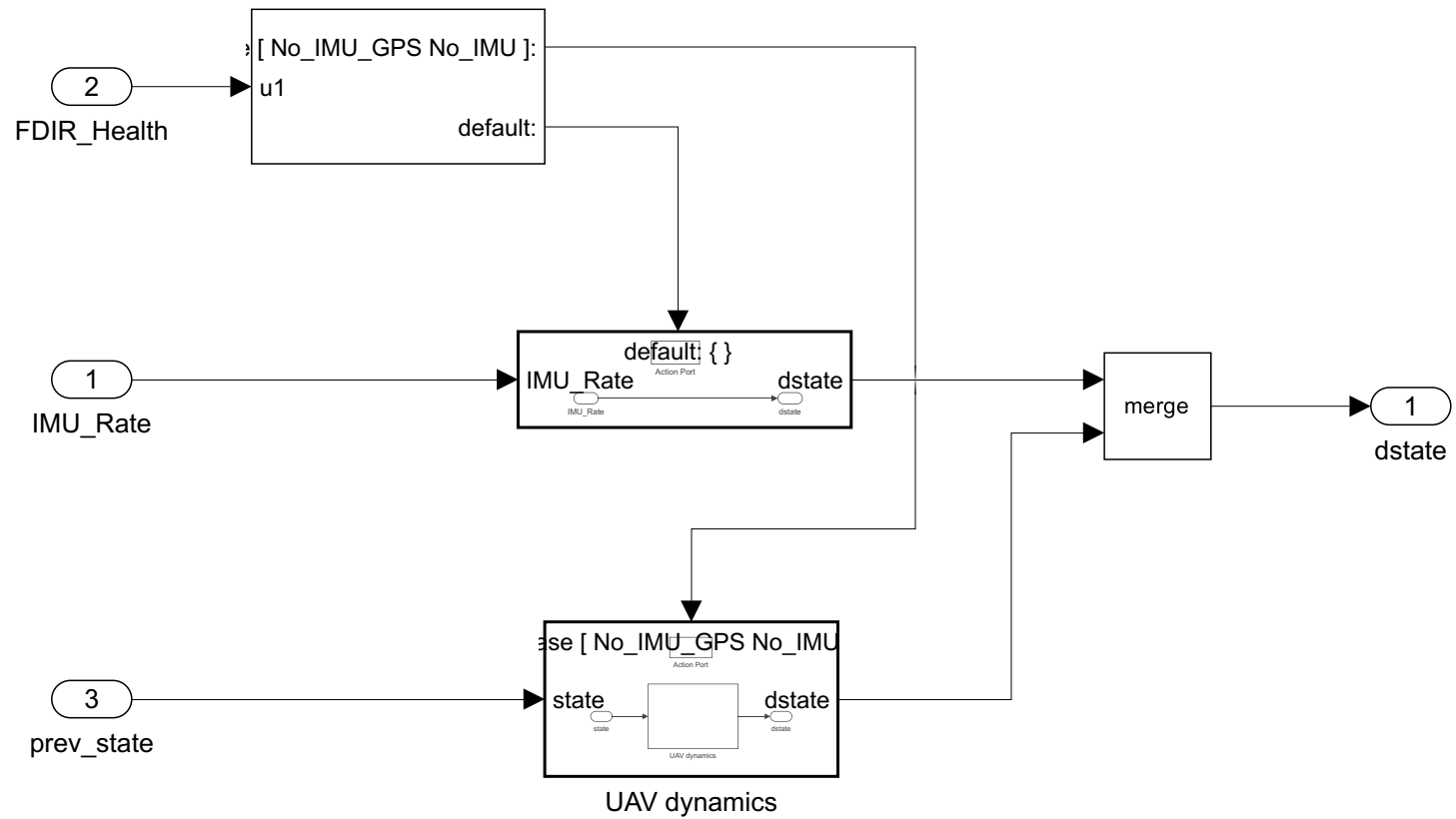
4
speed

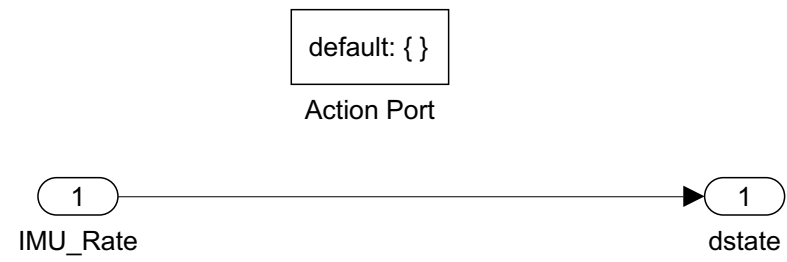
5
altitude

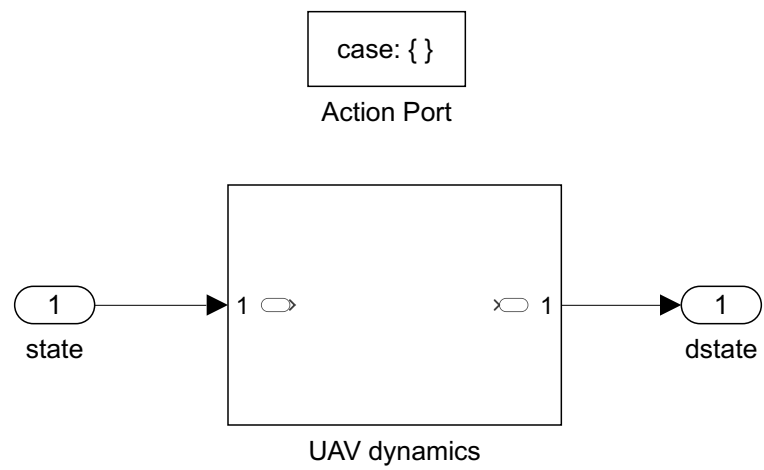
6
roll

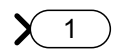
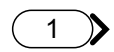
7
pitch

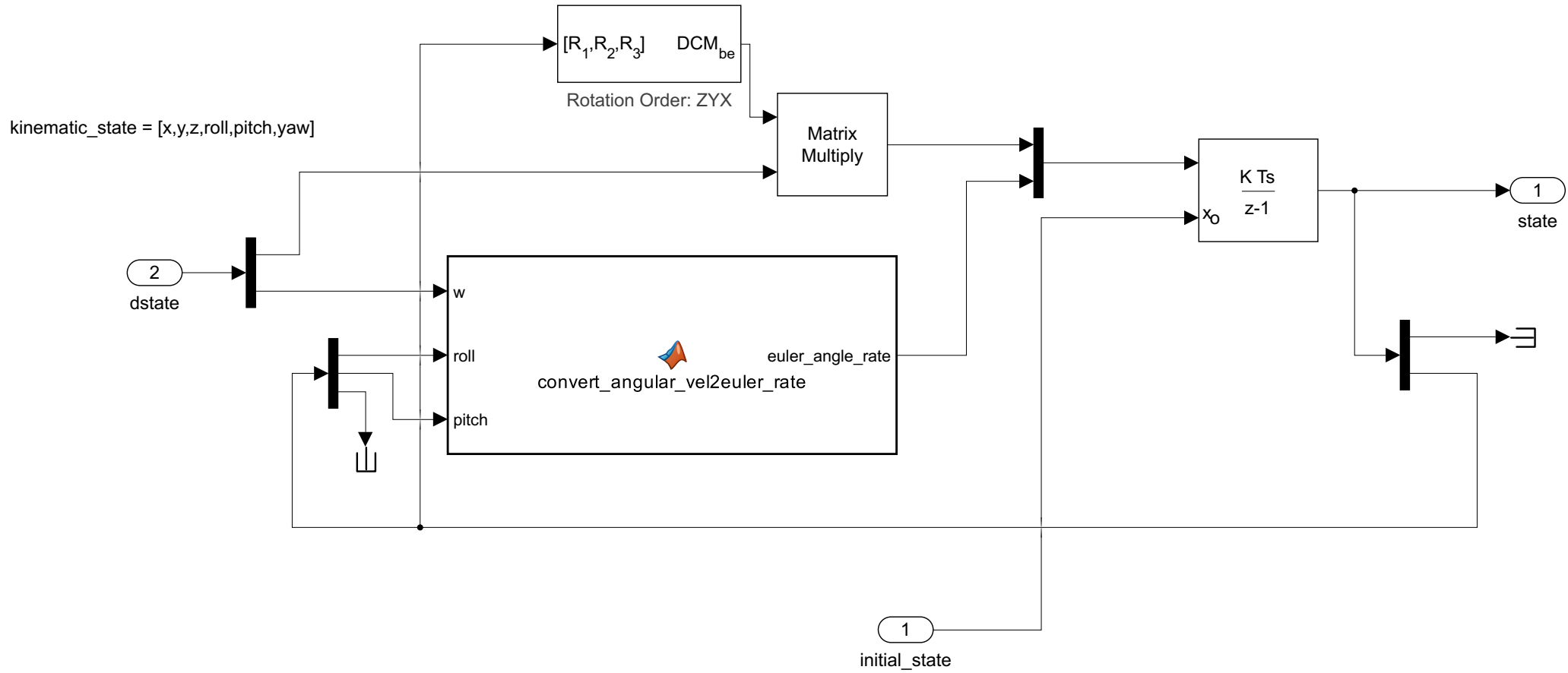
8
yaw

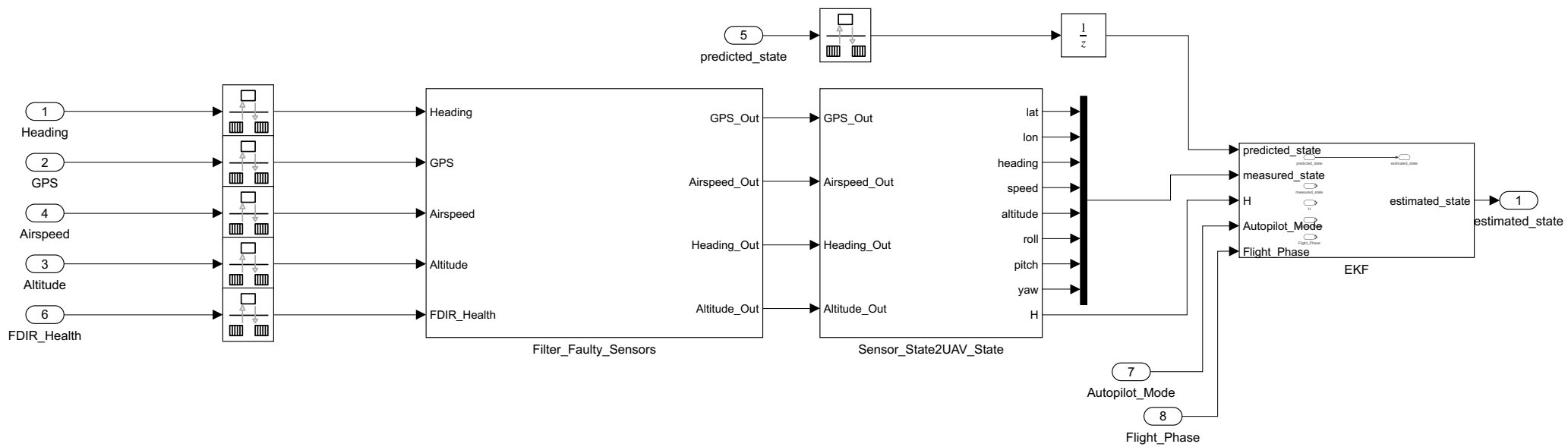


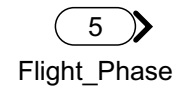
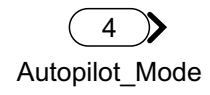
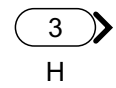
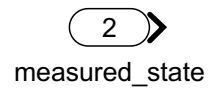
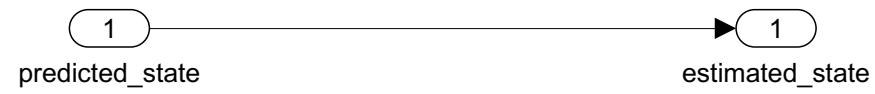


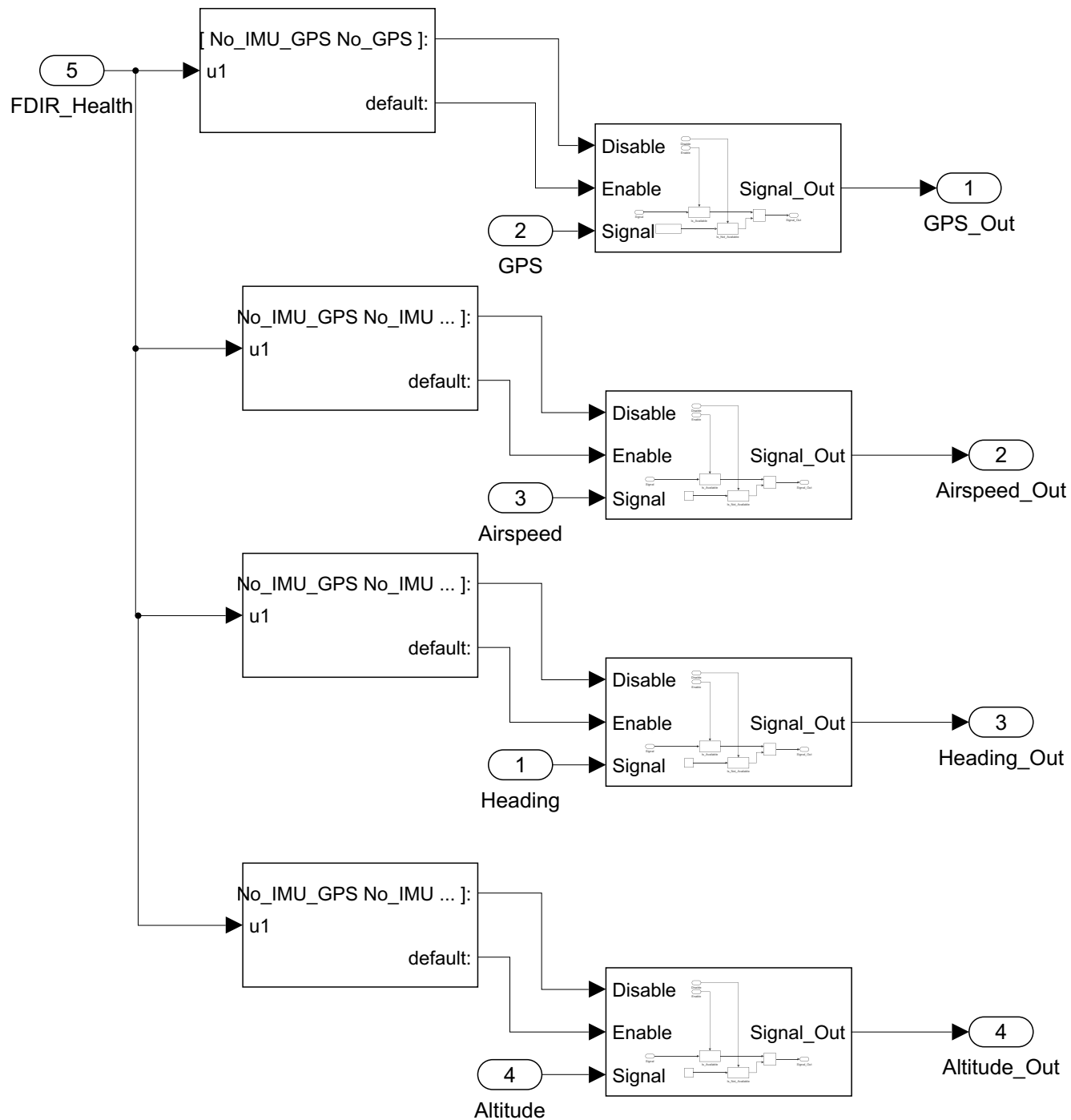


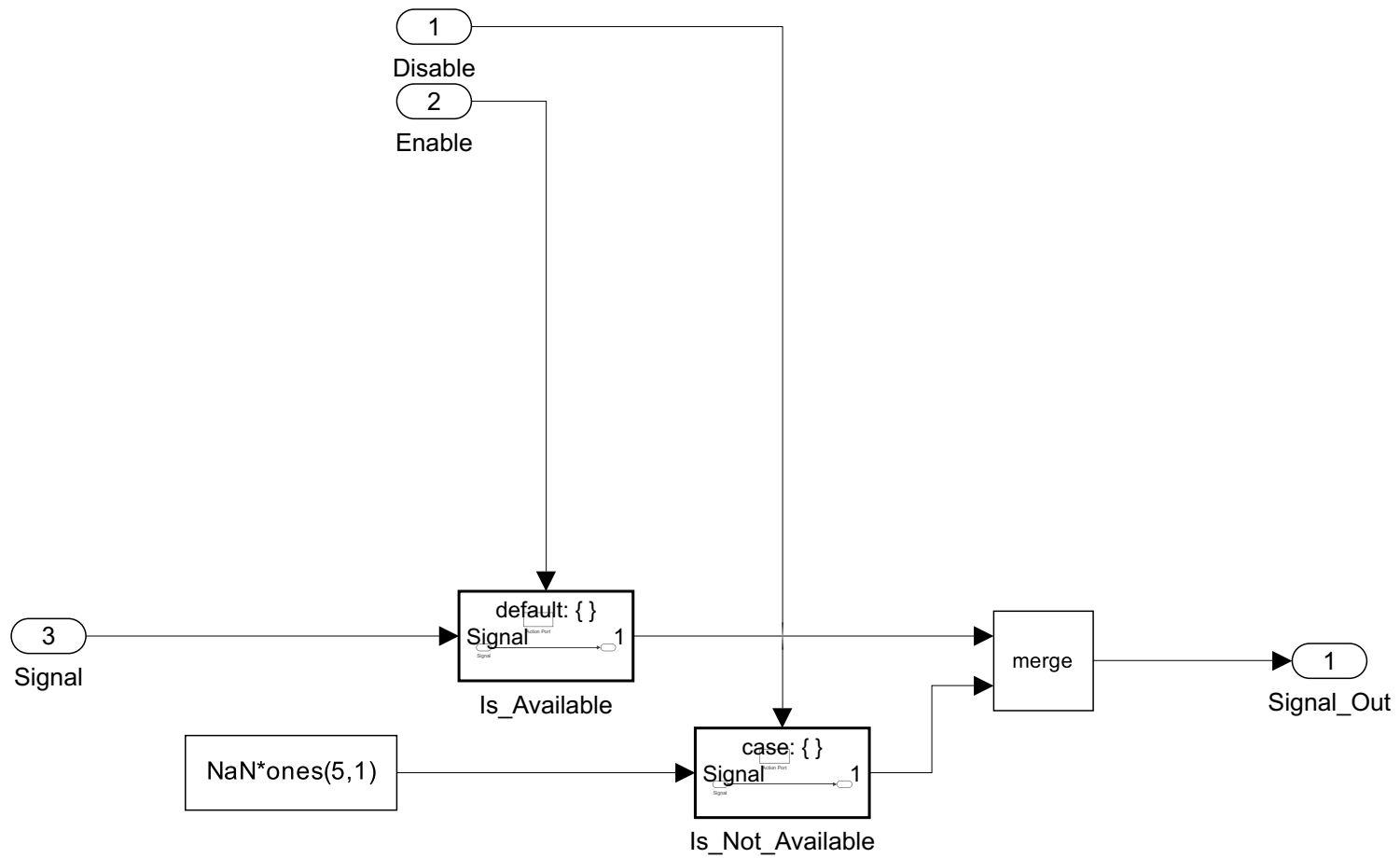


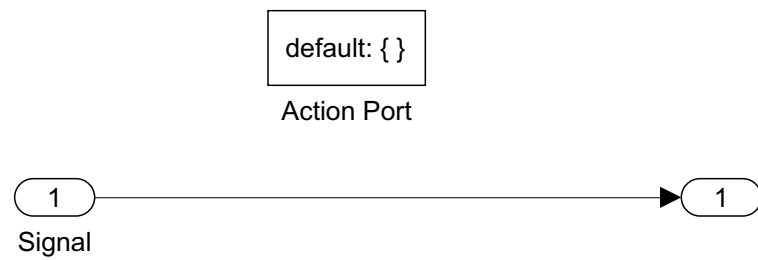


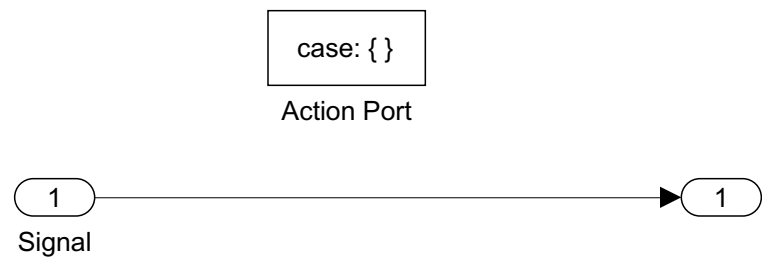


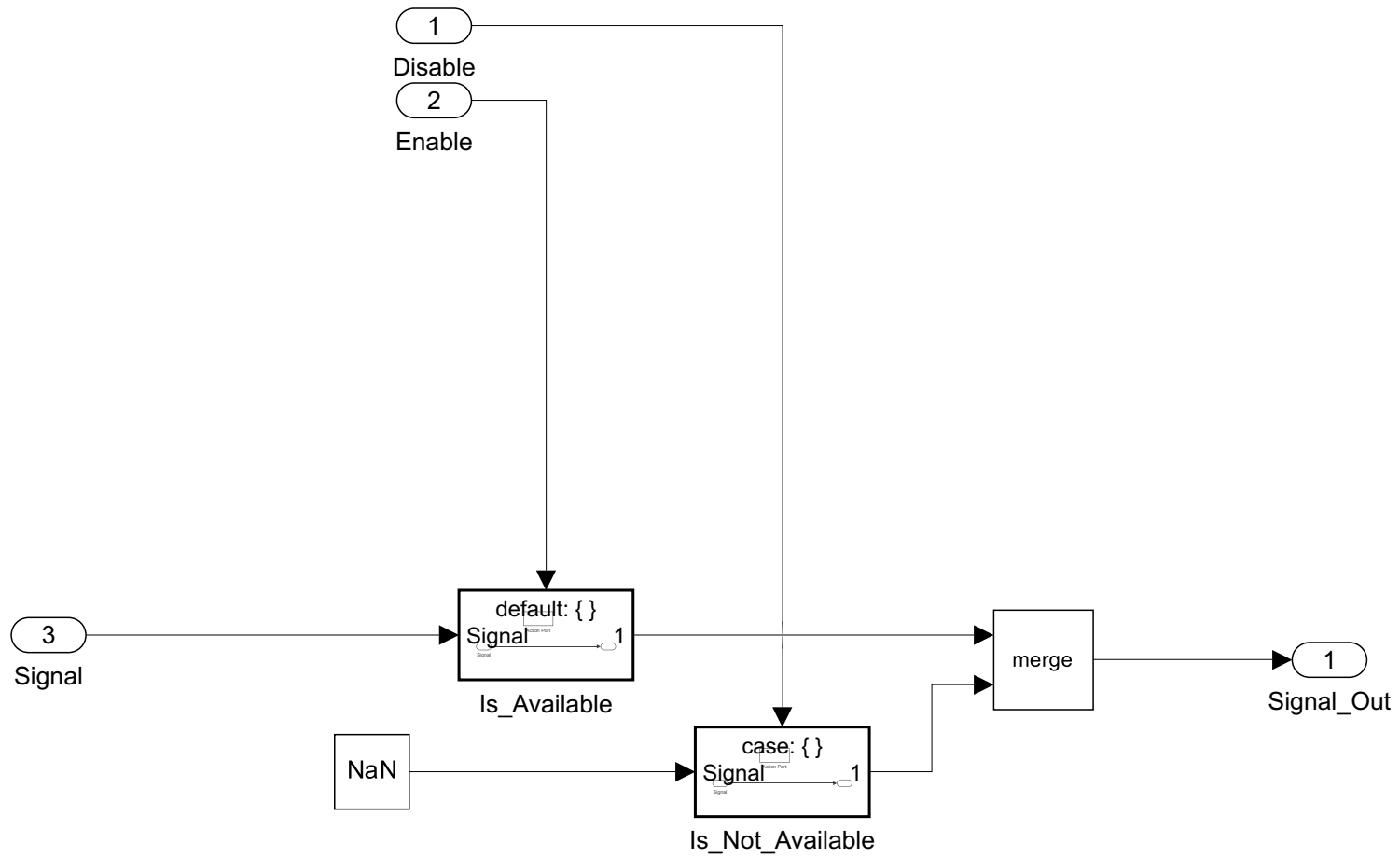


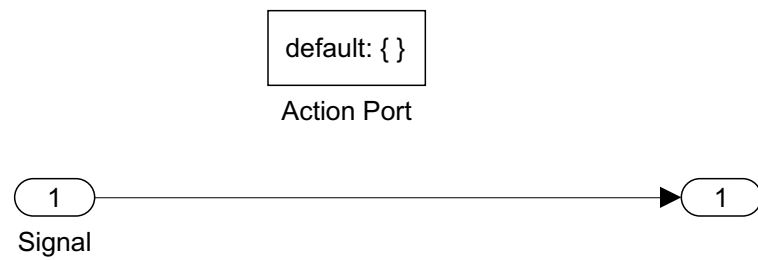


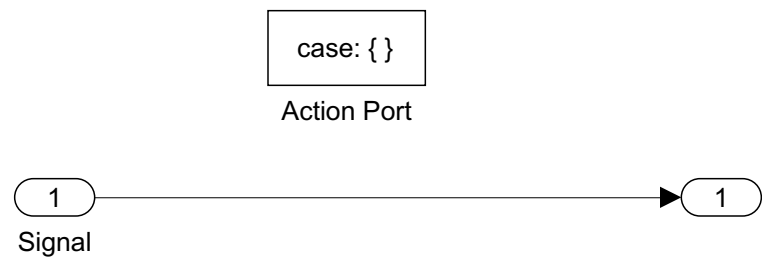


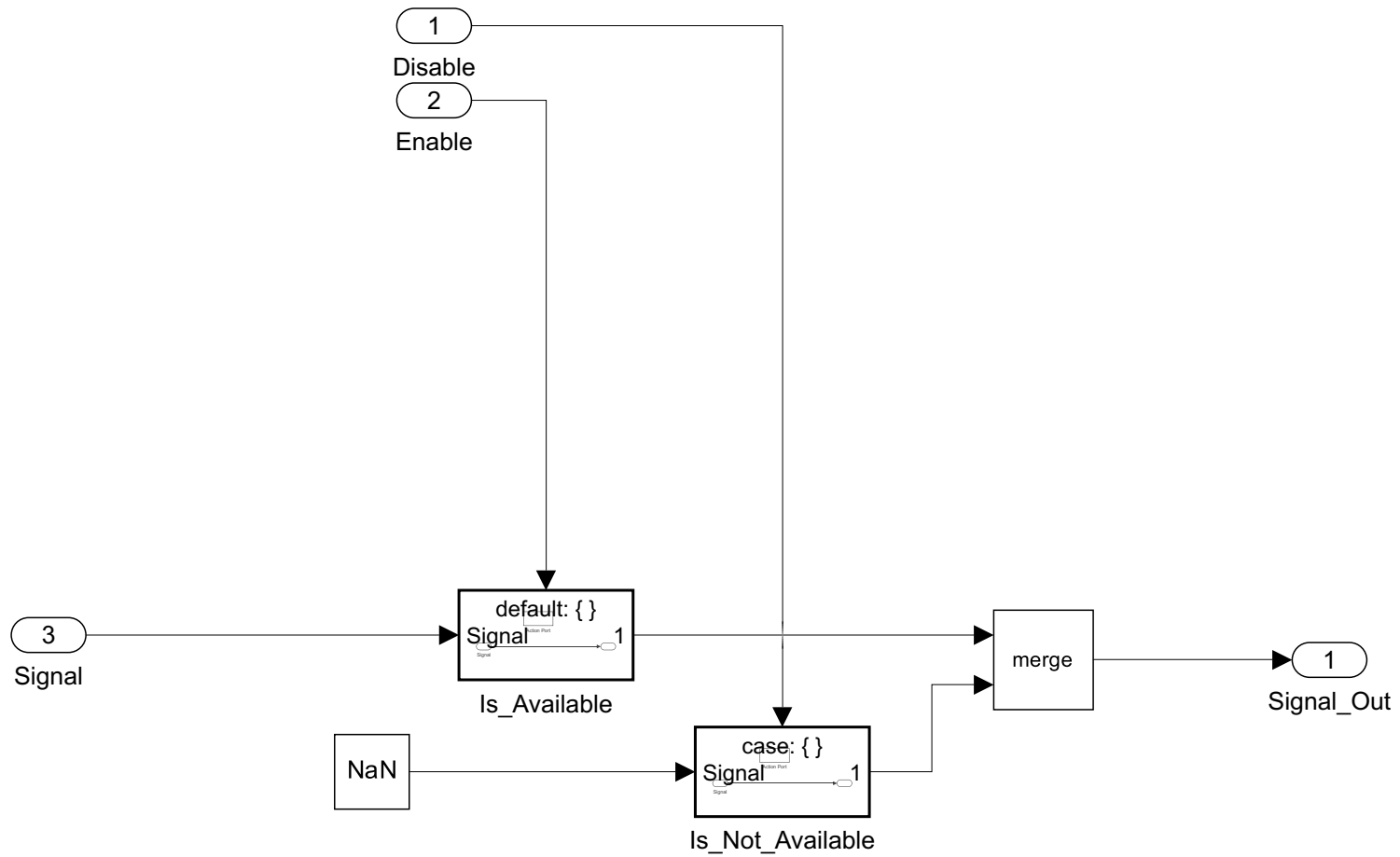


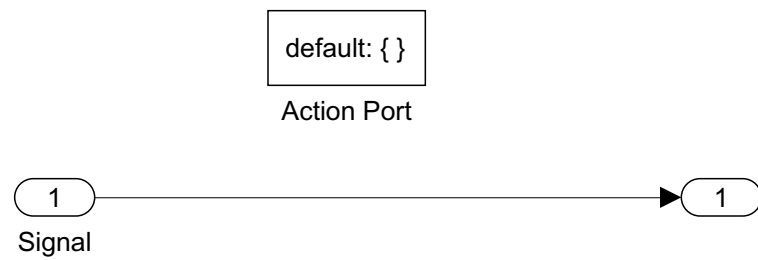


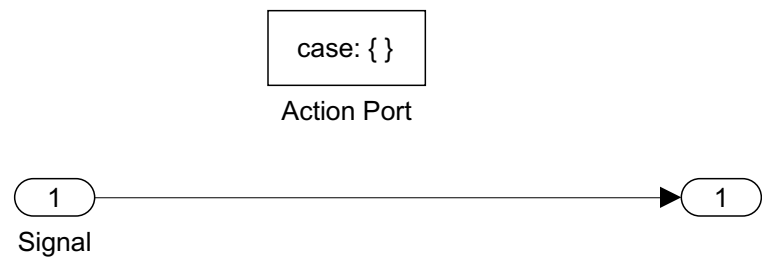


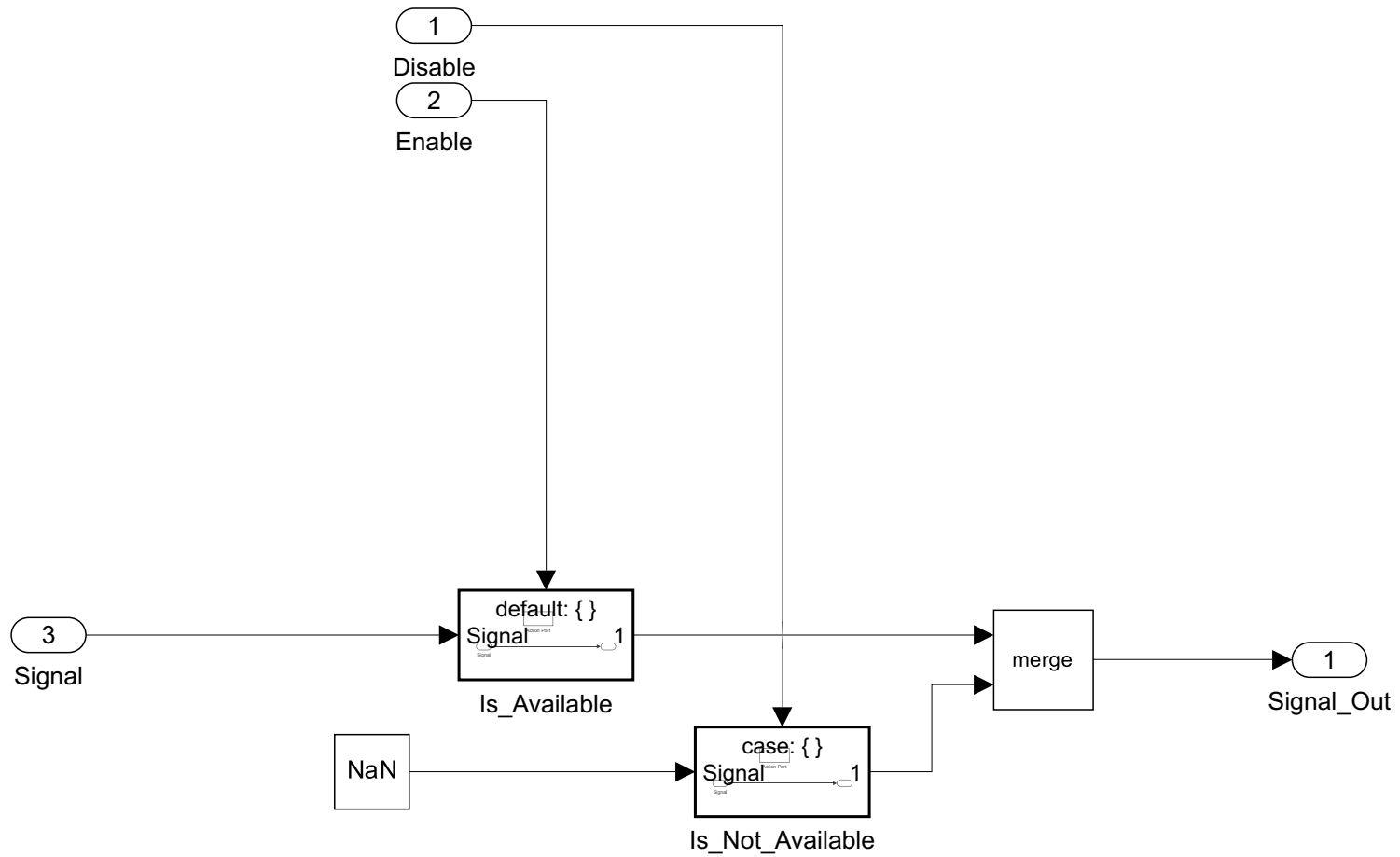


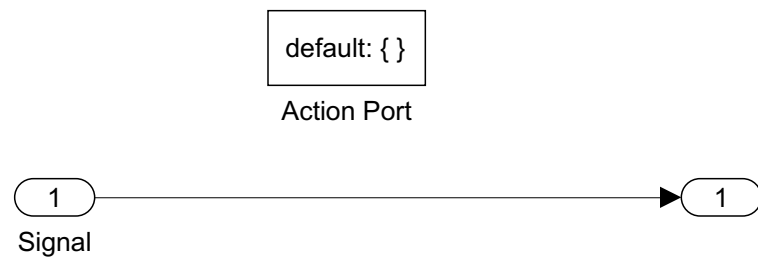


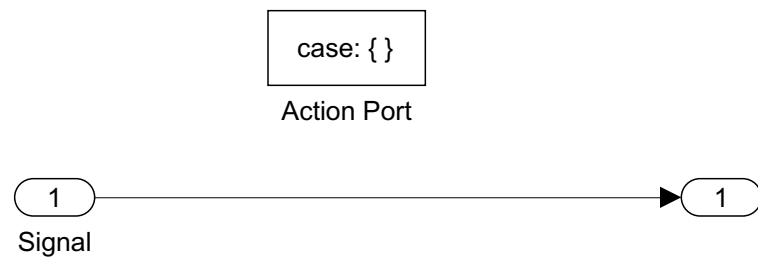








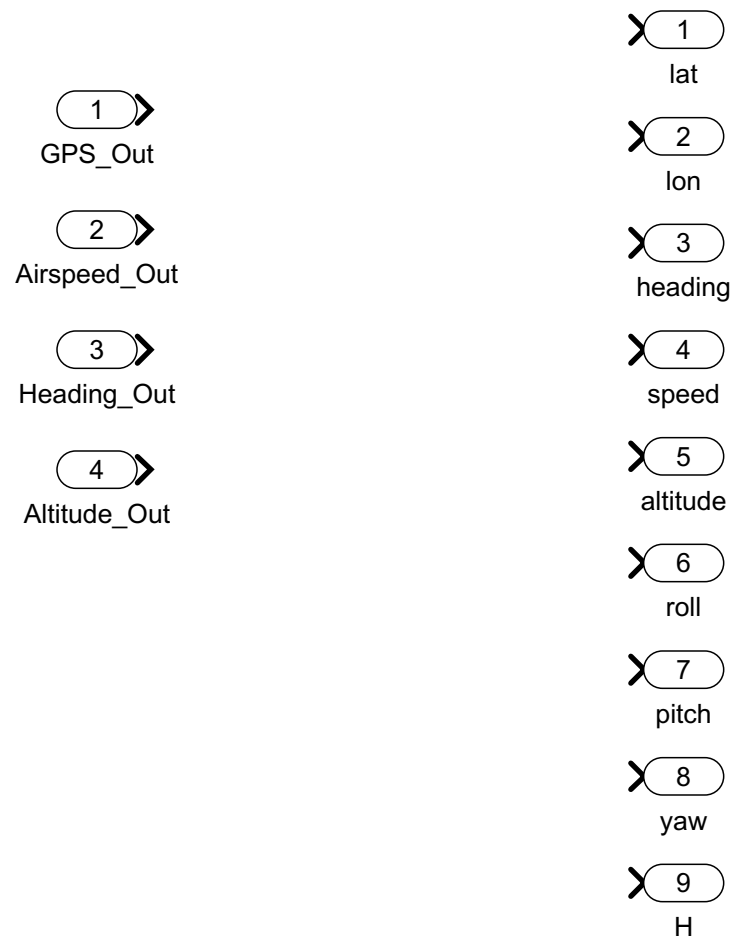


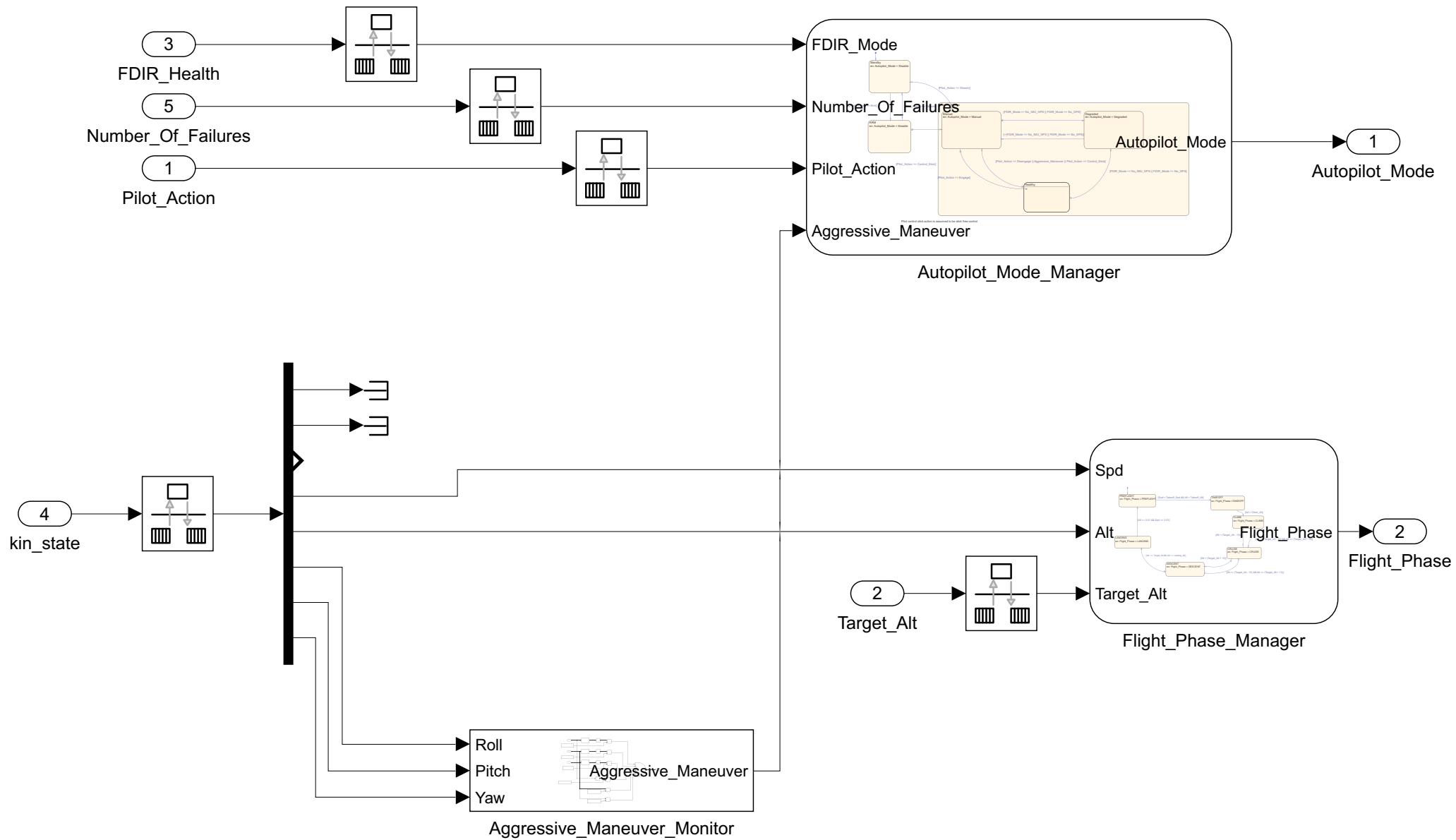


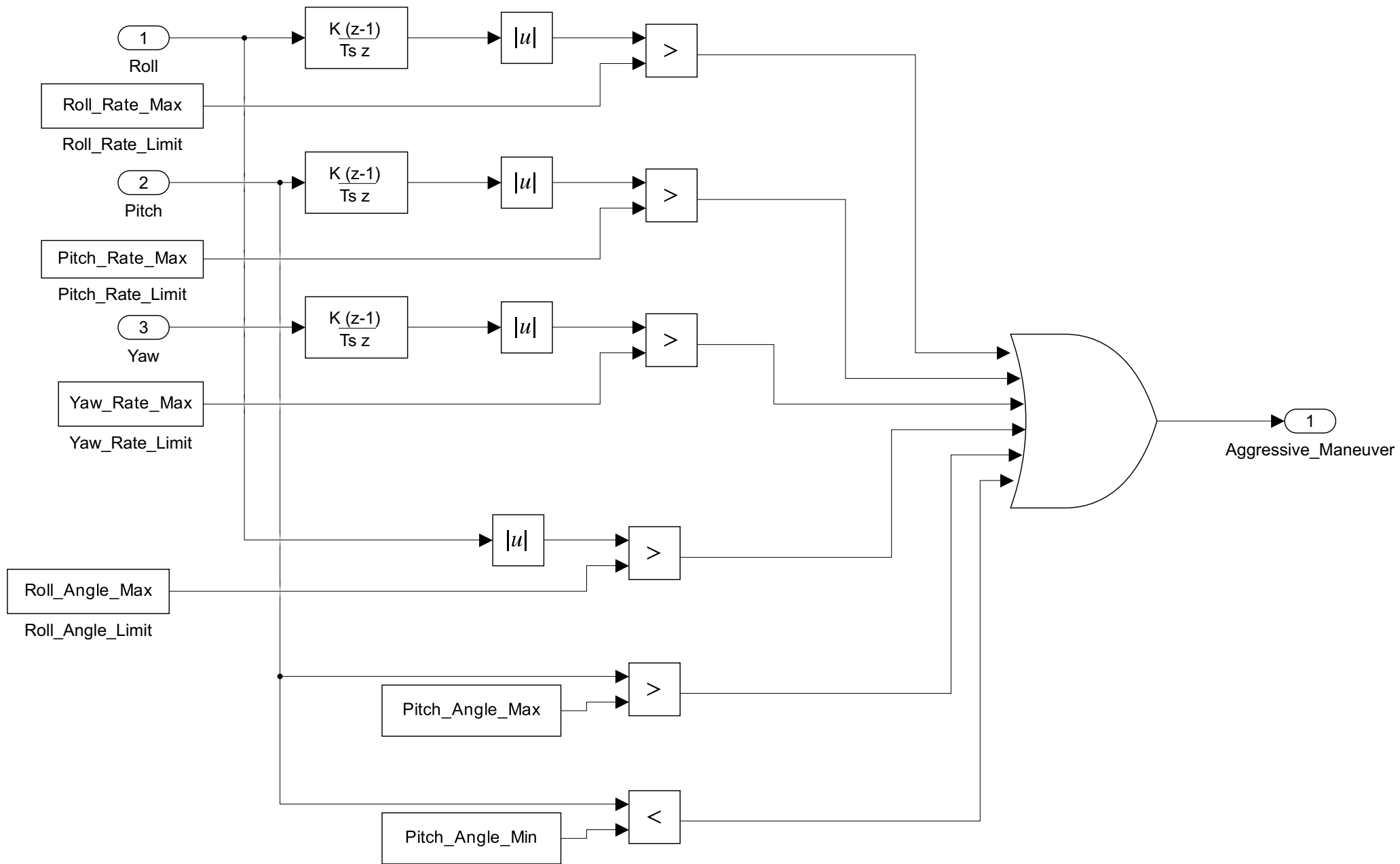
Dummy_Implementation

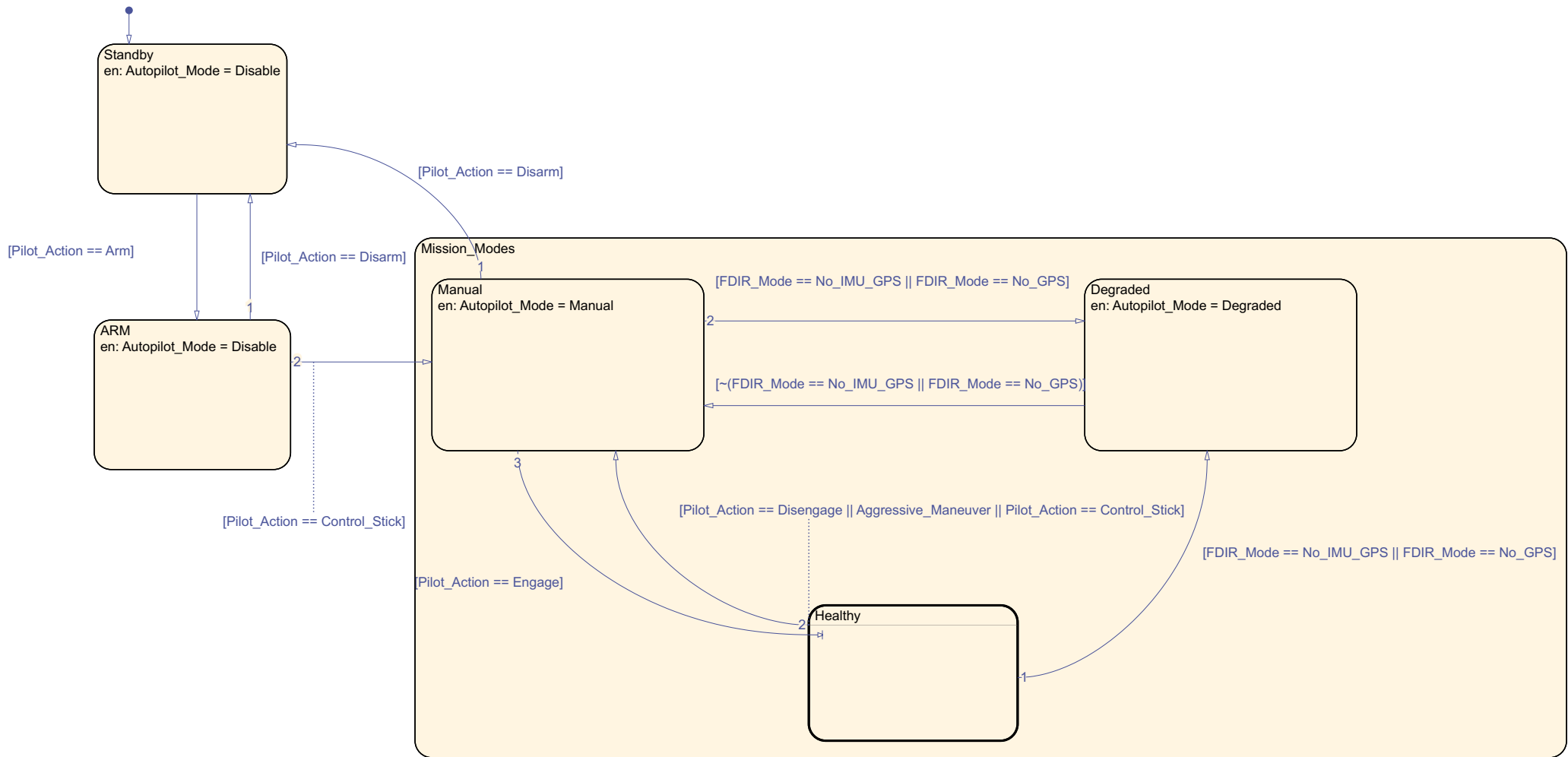
Logic needs to be implemented to pick the sensor fusion quantities based on quality of the data (Airspeed from pitot tube is more preferable than GPS speed when both are available) and filter out the NaN signals

Entries of H matrix for measurement equation can be made zero to not consider them for measurement update









Pilot control stick action is assumed to be stick free control

Healthy

du: Is_Autonomous_Mode_Possible = FDIR_Mode == No_GPS && Number_Of_Failures < 2

