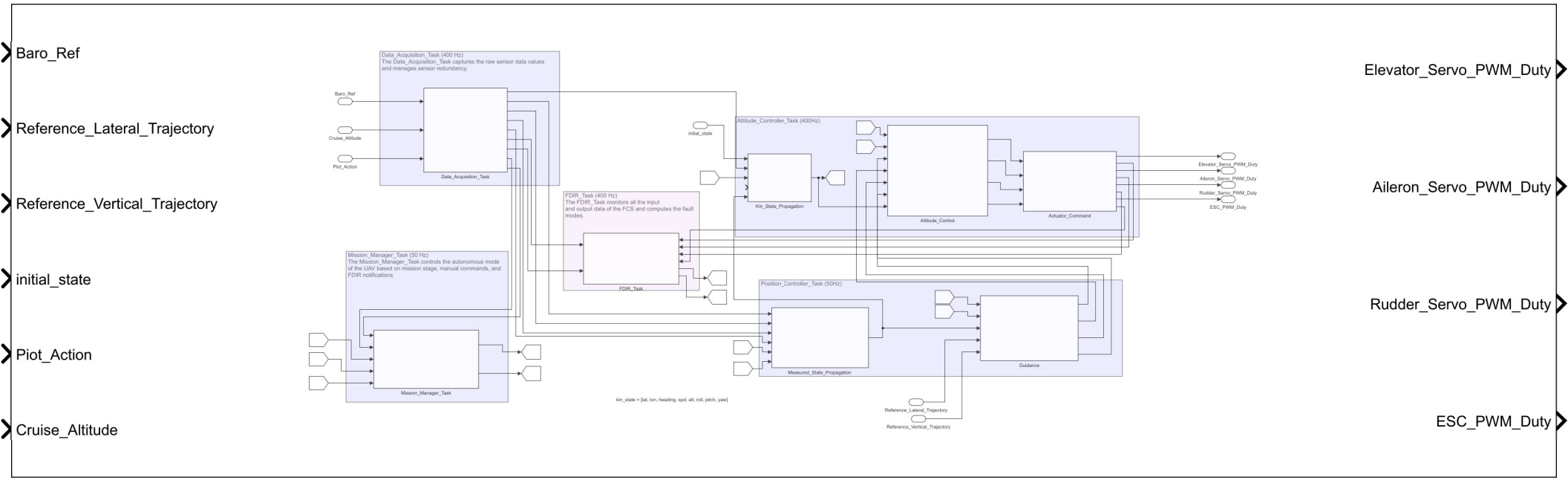
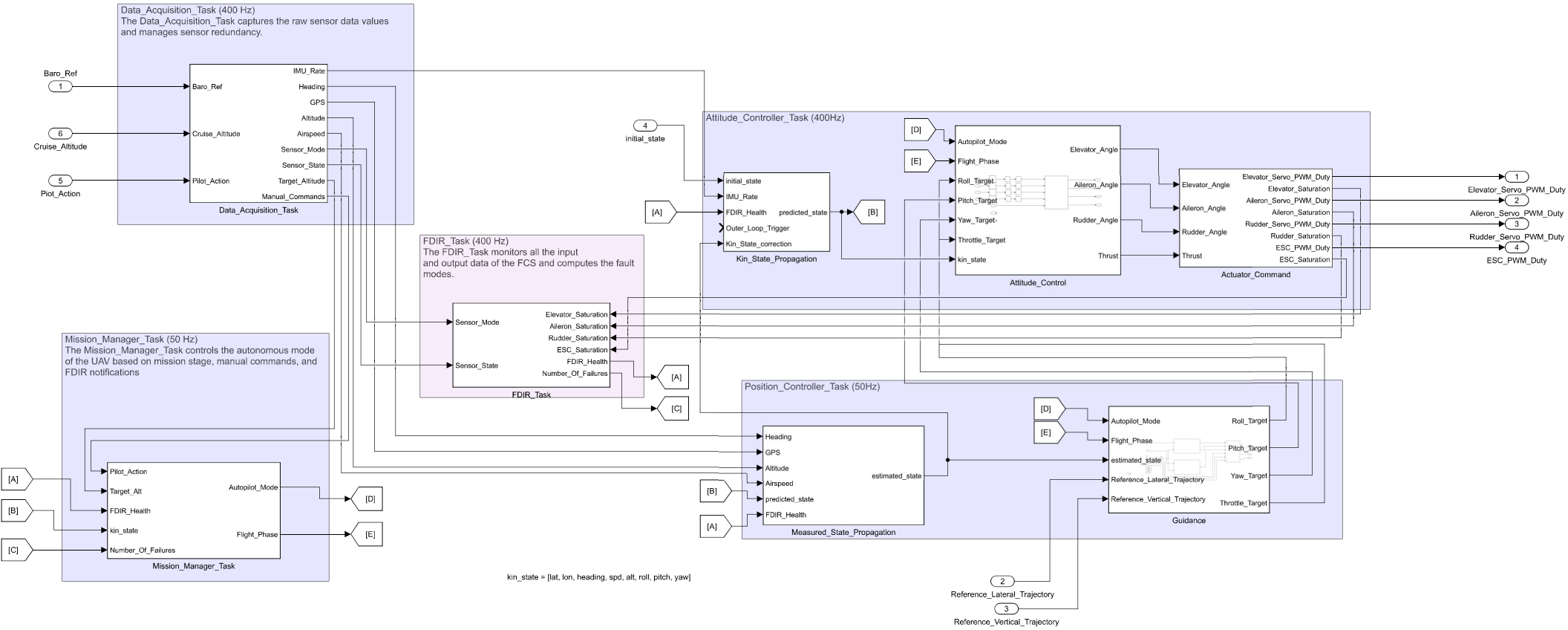
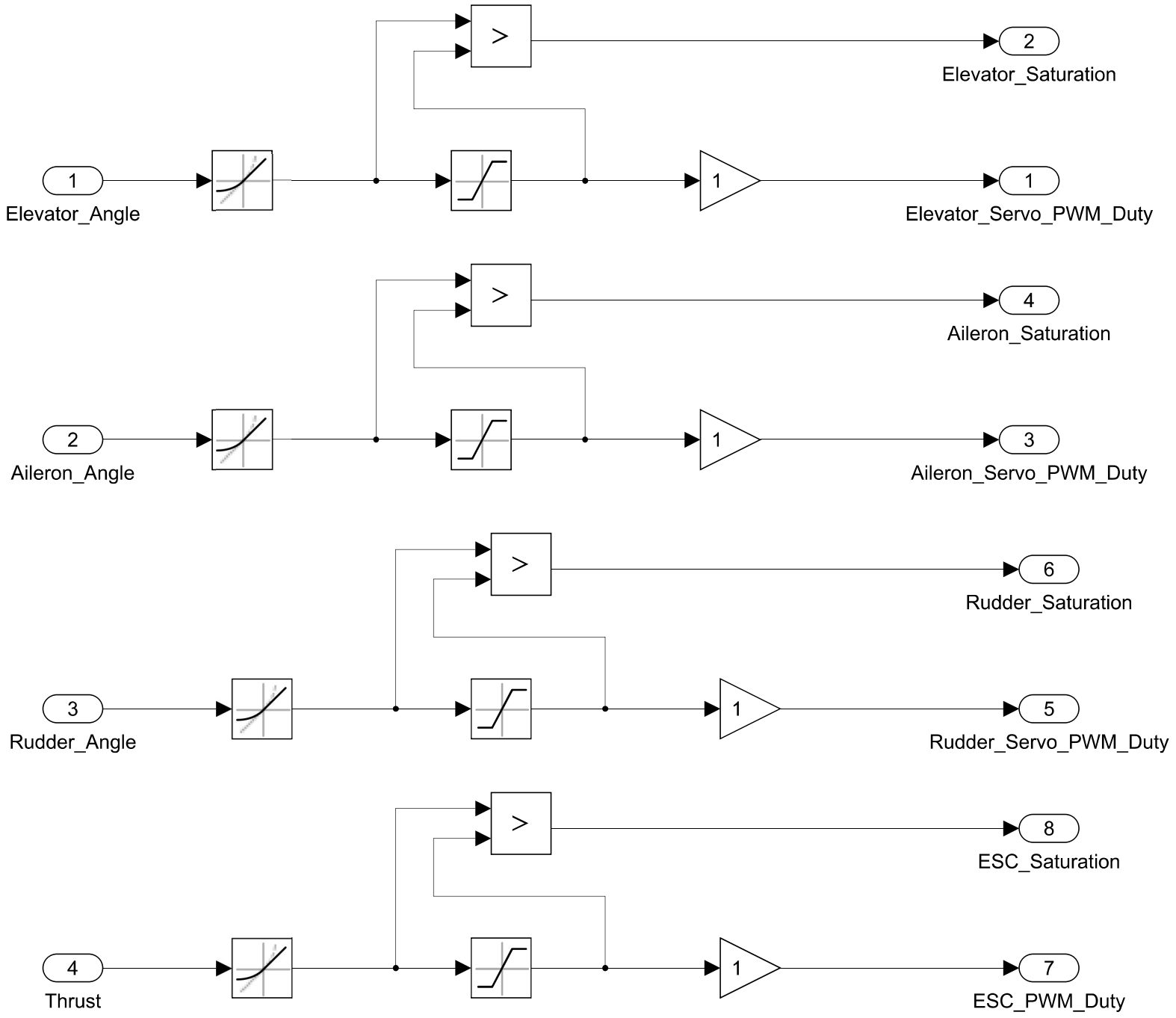
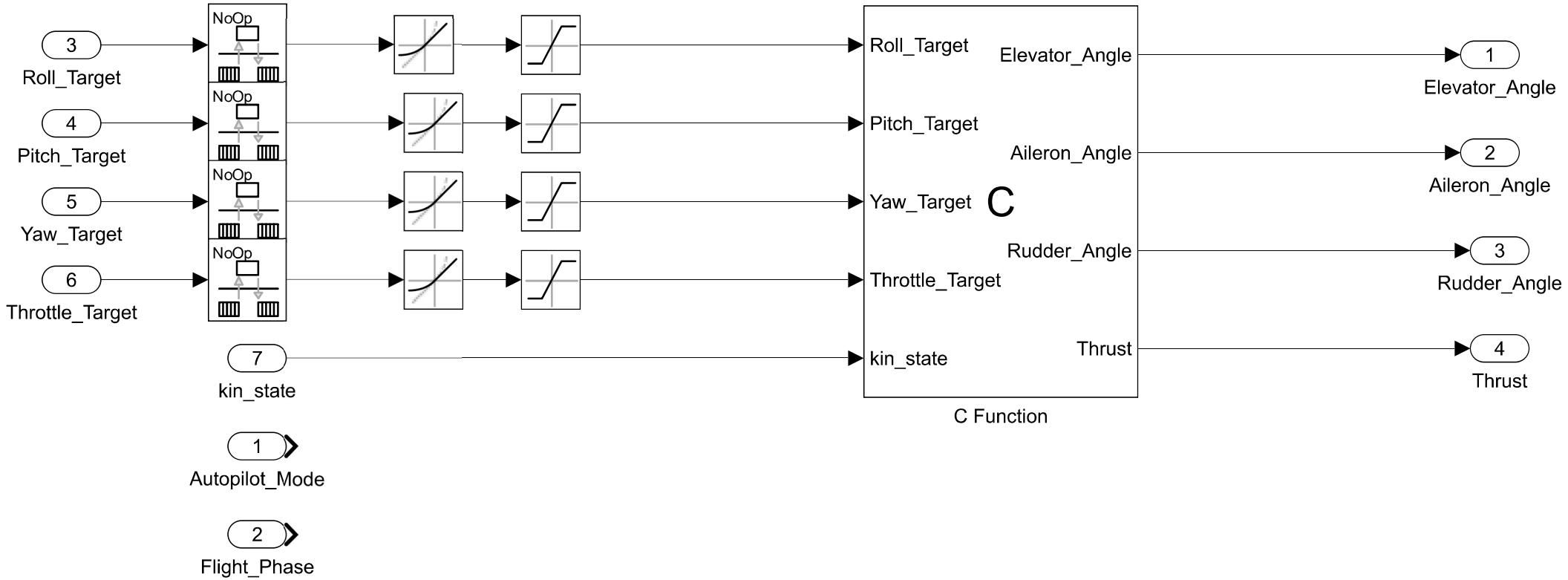
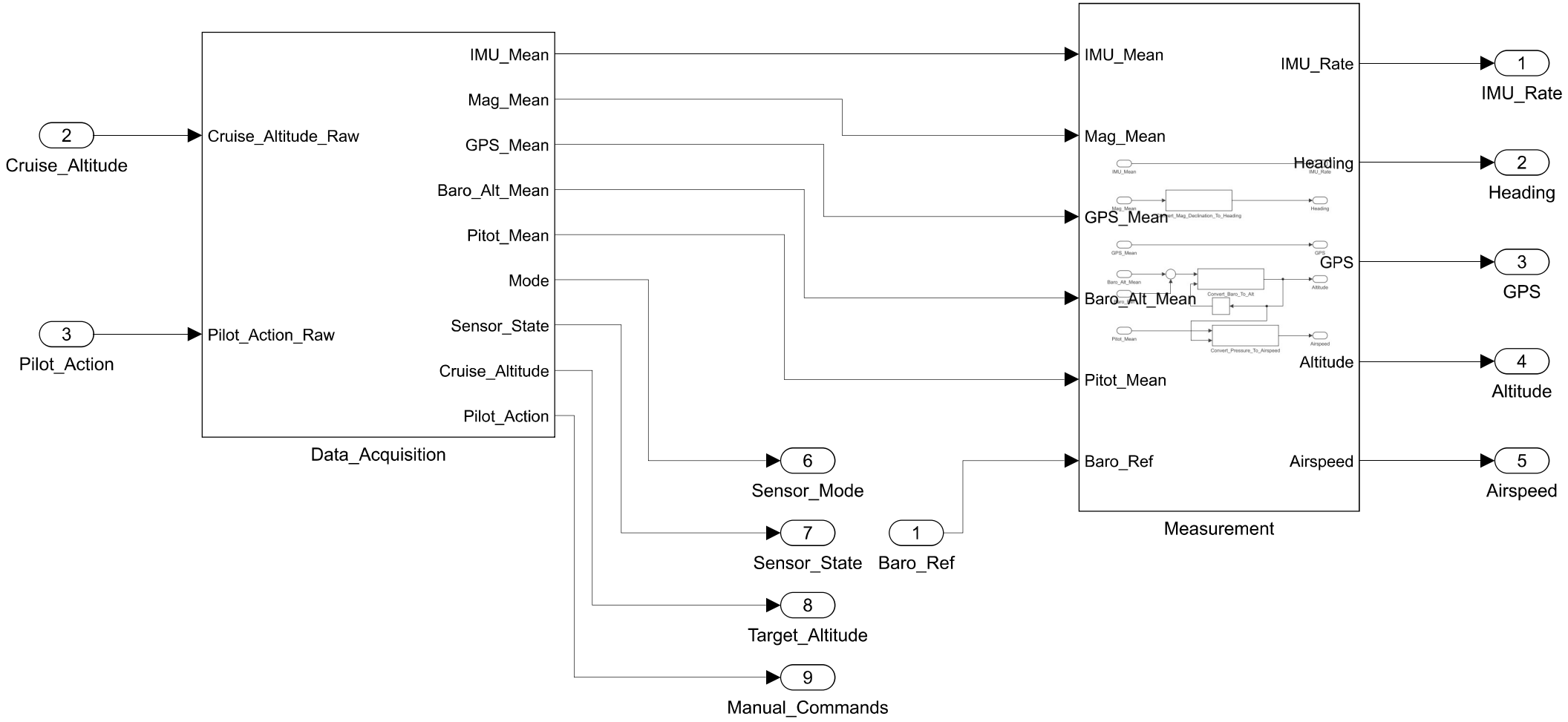

Baro_Ref
Reference_Lateral_Trajectory
Reference_Vertical_Trajectory
initial_state
Piot_Action
Cruise_Altitude
Elevator_Servo_PWM_Duty
Aileron_Servo_PWM_Duty
Rudder_Servo_PWM_Duty
ESC_PWM_Duty
Data_Acquisition_Task (400 Hz)
The Data_Acquisition_Task captures the raw sensor data values and manages sensor redundancy.
Data_Acquisition_Task
FDIR_Task (400 Hz)
The FDIR_Task monitors all the input and output data of the FCS and computes the fault modes.
FDIR_Task
Mission_Manager_Task (50 Hz)
The Mission_Manager_Task controls the autonomous mode of the UAV based on mission stage, manual commands, and FDIR notifications
Mission_Manager_Task
Attitude_Controller_Task (400Hz)
Kin_State_Propagation
Attitude_Control
Actuator_Command
Position_Controller_Task (50Hz)
Measured_State_Propagation
Guidance
kin_state = [lat, lon, heading, spd, alt, roll, pitch, yaw]

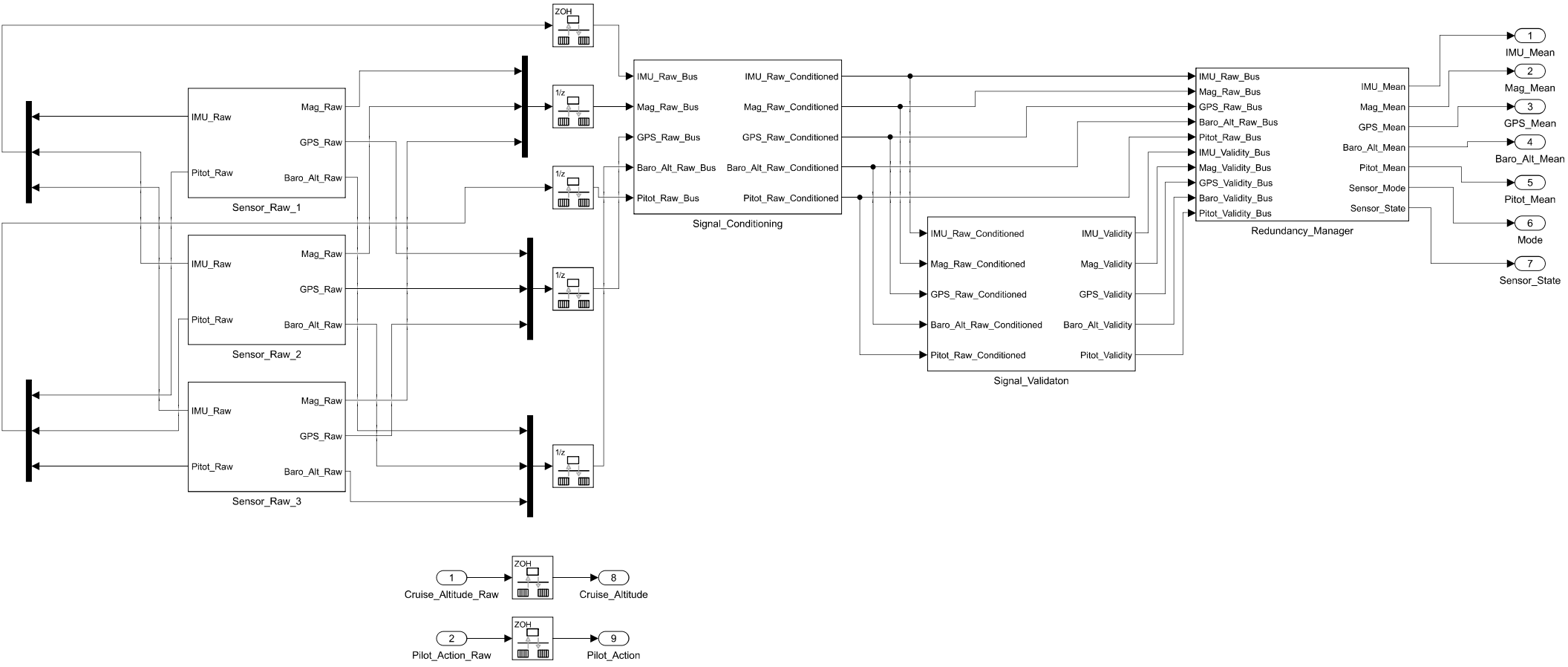
FCS


Data_Acquisition_Task (400 Hz)
The Data_Acquisition_Task captures the raw sensor data values and manages sensor redundancy.
Baro_Ref
1
6
Cruise_Altitude
5
Piot_Action
Baro_Ref
Cruise_Altitude
Pilot_Action
IMU_Rate
Heading
GPS
Altitude
Airspeed
Sensor_Mode
Sensor_State
Target_Altitude
Manual_Commands
Data_Acquisition_Task
4
initial_state
[A]
Attitude_Controller_Task (400Hz)
initial_state
IMU_Rate
FDIR_Health
predicted_state
Outer_Loop_Trigger
Kin_State_correction
Kin_State_Propagation
[B]
[D]
[E]
Autopilot_Mode
Flight_Phase
Roll_Target
Pitch_Target
Yaw_Target
Throttle_Target
kin_state
Elevator_Angle
Aileron_Angle
Rudder_Angle
Thrust
Attitude_Control
Elevator_Angle
Aileron_Angle
Rudder_Angle
Thrust
Elevator_Servo_PWM_Duty
Elevator_Saturation
Aileron_Servo_PWM_Duty
Aileron_Saturation
Rudder_Servo_PWM_Duty
Rudder_Saturation
ESC_PWM_Duty
ESC_Saturation
Actuator_Command
1
Elevator_Servo_PWM_Duty
2
Aileron_Servo_PWM_Duty
3
Rudder_Servo_PWM_Duty
4
ESC_PWM_Duty
FDIR_Task (400 Hz)
The FDIR_Task monitors all the input and output data of the FCS and computes the fault modes.
Sensor_Mode
Sensor_State
Elevator_Saturation
Aileron_Saturation
Rudder_Saturation
ESC_Saturation
FDIR_Health
Number_Of_Failures
FDIR_Task
[A]
[C]
Mission_Manager_Task (50 Hz)
The Mission_Manager_Task controls the autonomous mode of the UAV based on mission stage, manual commands, and FDIR notifications
[A]
[B]
[C]
Pilot_Action
Target_Alt
FDIR_Health
kin_state
Number_Of_Failures
Autopilot_Mode
Flight_Phase
Mission_Manager_Task
[D]
[E]
Position_Controller_Task (50Hz)
Heading
GPS
Altitude
Airspeed
predicted_state
FDIR_Health
estimated_state
Measured_State_Propagation
[B]
[A]
[D]
[E]
Autopilot_Mode
Flight_Phase
estimated_state
Reference_Lateral_Trajectory
Reference_Vertical_Trajectory
Roll_Target
Pitch_Target
Yaw_Target
Throttle_Target
Guidance
kin_state = [lat, lon, heading, spd, alt, roll, pitch, yaw]
2
Reference_Lateral_Trajectory
3
Reference_Vertical_Trajectory

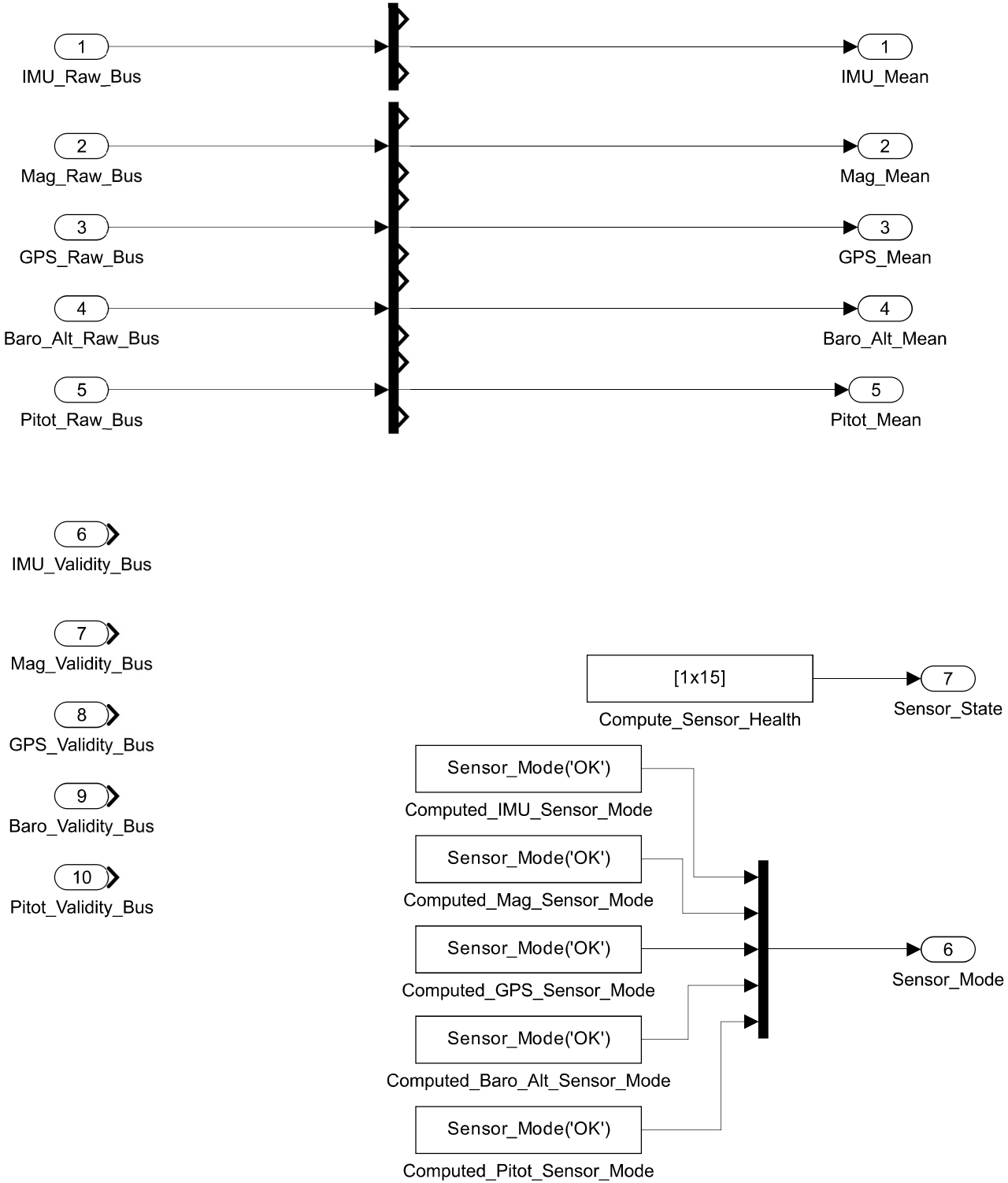

1
Elevator_Angle
>
2
Elevator_Saturation
1
1
Elevator_Servo_PWM_Duty
2
Aileron_Angle
>
4
Aileron_Saturation
1
3
Aileron_Servo_PWM_Duty
3
Rudder_Angle
>
6
Rudder_Saturation
1
5
Rudder_Servo_PWM_Duty
4
Thrust
>
8
ESC_Saturation
1
7
ESC_PWM_Duty

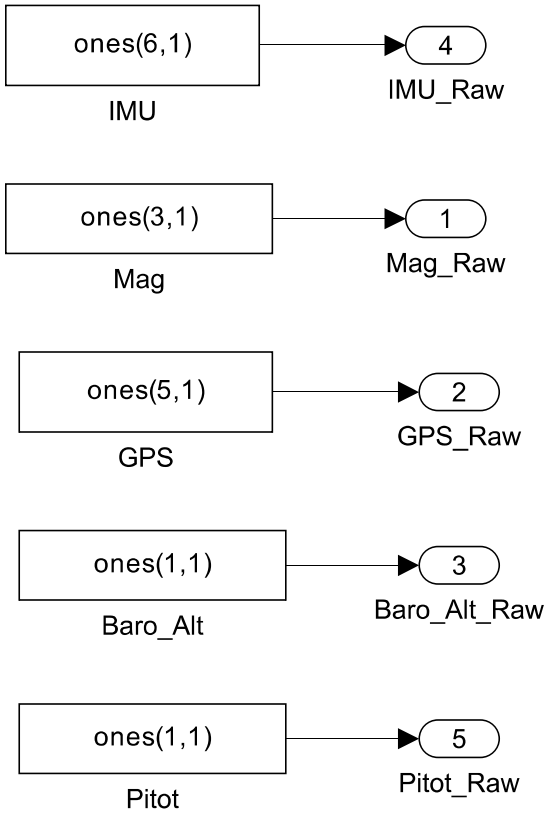

3
Roll_Target
4
Pitch_Target
5
Yaw_Target
6
Throttle_Target
NoOp
NoOp
NoOp
NoOp
7
kin_state
1
Autopilot_Mode
2
Flight_Phase
Roll_Target
Pitch_Target
Yaw_Target
Throttle_Target
kin_state
C
Elevator_Angle
Aileron_Angle
Rudder_Angle
Thrust
C Function
1
Elevator_Angle
2
Aileron_Angle
3
Rudder_Angle
4
Thrust

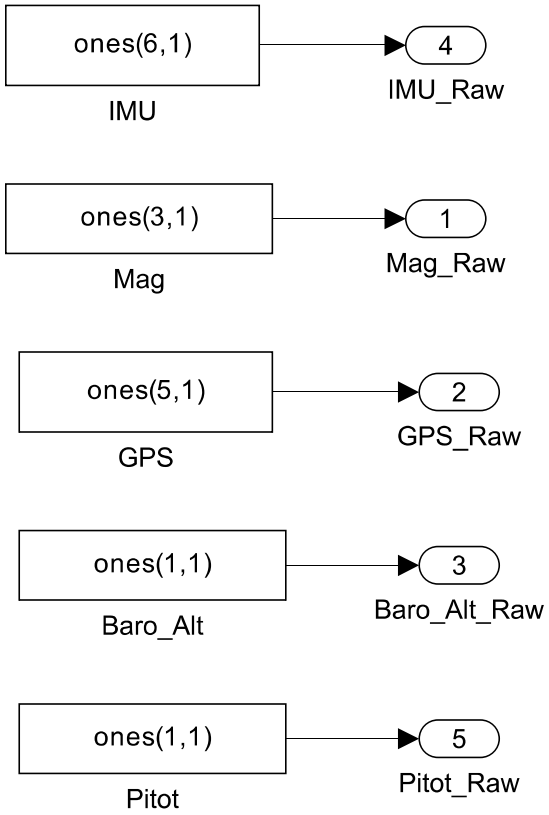

2
Cruise_Altitude
3
Pilot_Action
Cruise_Altitude_Raw
Pilot_Action_Raw
IMU_Mean
Mag_Mean
GPS_Mean
Baro_Alt_Mean
Pitot_Mean
Mode
Sensor_State
Cruise_Altitude
Pilot_Action
Data_Acquisition
6
Sensor_Mode
7
Sensor_State
8
Target_Altitude
9
Manual_Commands
1
Baro_Ref
IMU_Mean
Mag_Mean
GPS_Mean
Baro_Alt_Mean
Pitot_Mean
Baro_Ref
IMU_Rate
Heading
GPS
Altitude
Airspeed
Measurement
1
IMU_Rate
2
Heading
3
GPS
4
Altitude
5
Airspeed

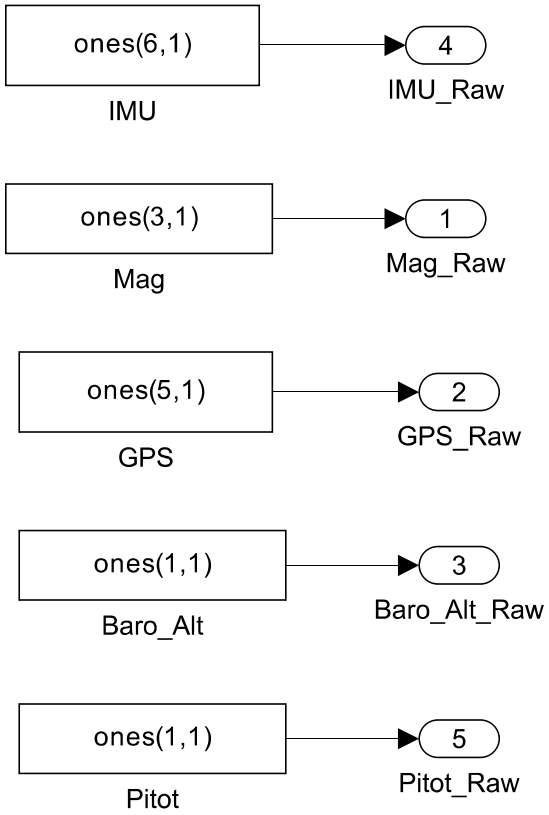

ZOH
1/z
IMU_Raw
Mag_Raw
GPS_Raw
Pitot_Raw
Baro_Alt_Raw
Sensor_Raw_1
Sensor_Raw_2
Sensor_Raw_3
IMU_Raw_Bus
Mag_Raw_Bus
GPS_Raw_Bus
Baro_Alt_Raw_Bus
Pitot_Raw_Bus
IMU_Raw_Conditioned
Mag_Raw_Conditioned
GPS_Raw_Conditioned
Baro_Alt_Raw_Conditioned
Pitot_Raw_Conditioned
Signal_Conditioning
IMU_Validity
Mag_Validity
GPS_Validity
Baro_Alt_Validity
Pitot_Validity
Signal_Validaton
IMU_Validity_Bus
Mag_Validity_Bus
GPS_Validity_Bus
Baro_Validity_Bus
Pitot_Validity_Bus
IMU_Mean
Mag_Mean
GPS_Mean
Baro_Alt_Mean
Pitot_Mean
Sensor_Mode
Sensor_State
Redundancy_Manager
1
IMU_Mean
2
Mag_Mean
3
GPS_Mean
4
Baro_Alt_Mean
5
Pitot_Mean
6
Mode
7
Sensor_State
Cruise_Altitude_Raw
8
Cruise_Altitude
Pilot_Action_Raw
9
Pilot_Action

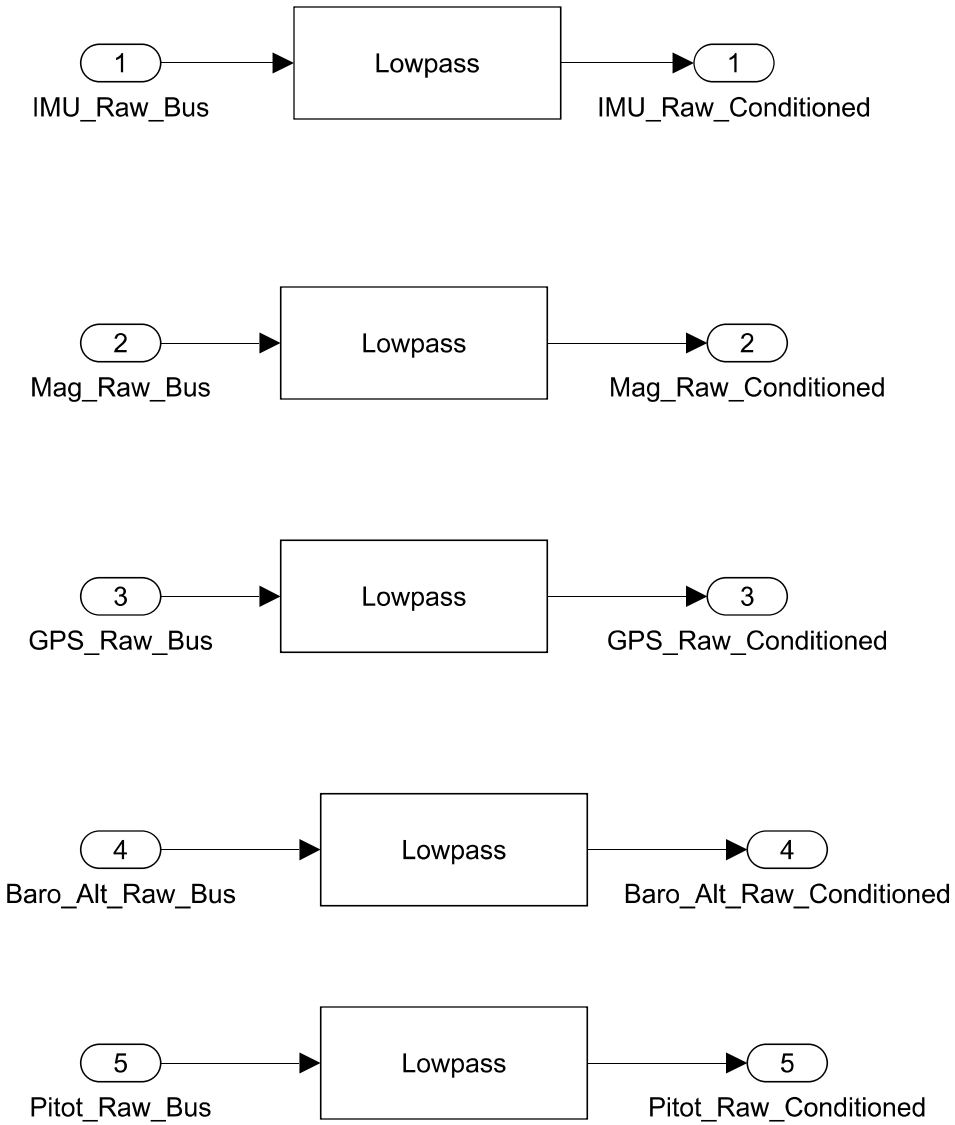
Dummy_Implementation

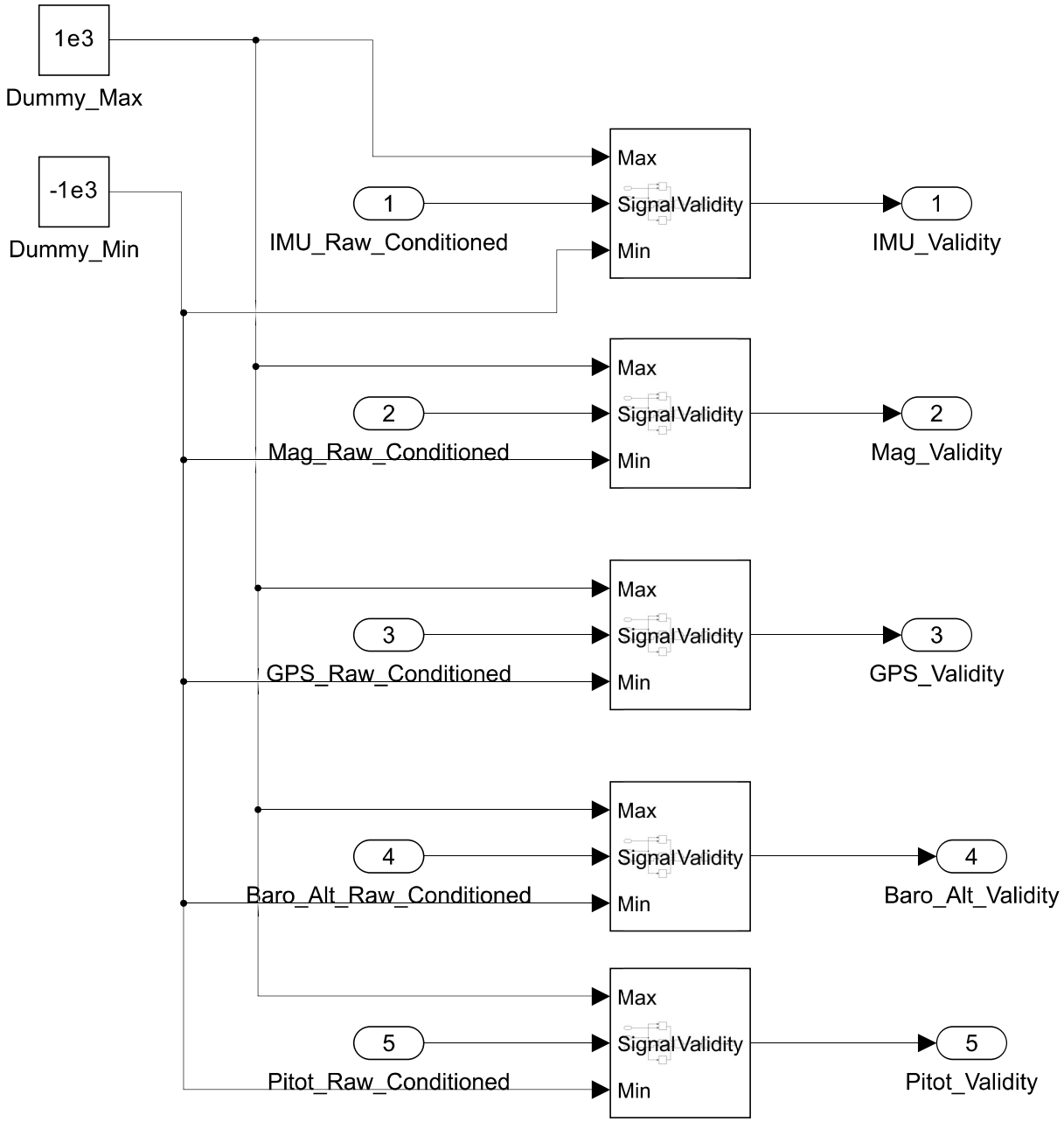


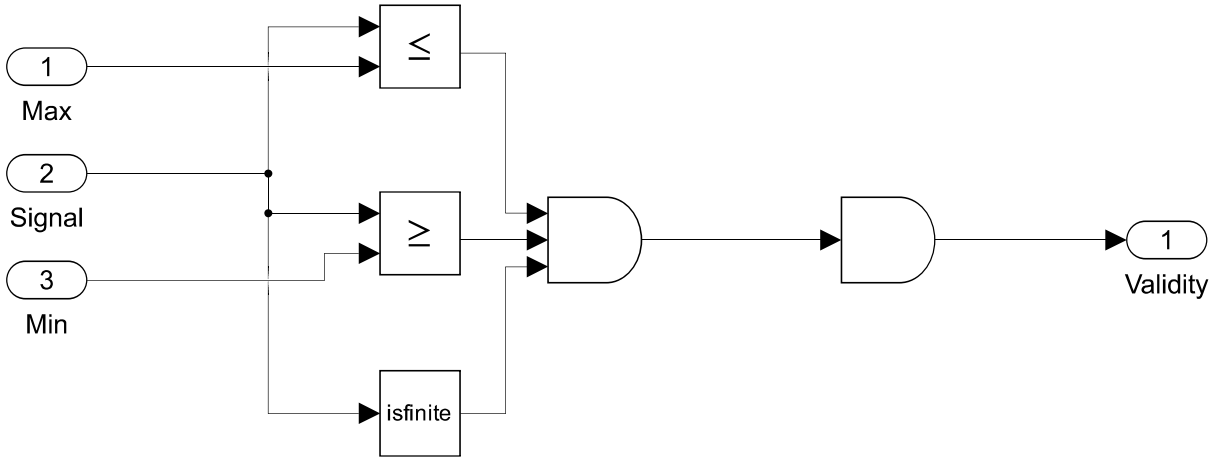

ones(6,1)
4
IMU_Raw
IMU
ones(3,1)
1
Mag_Raw
Mag
ones(5,1)
2
GPS_Raw
GPS
ones(1,1)
3
Baro_Alt_Raw
Baro_Alt
ones(1,1)
5
Pitot_Raw
Pitot

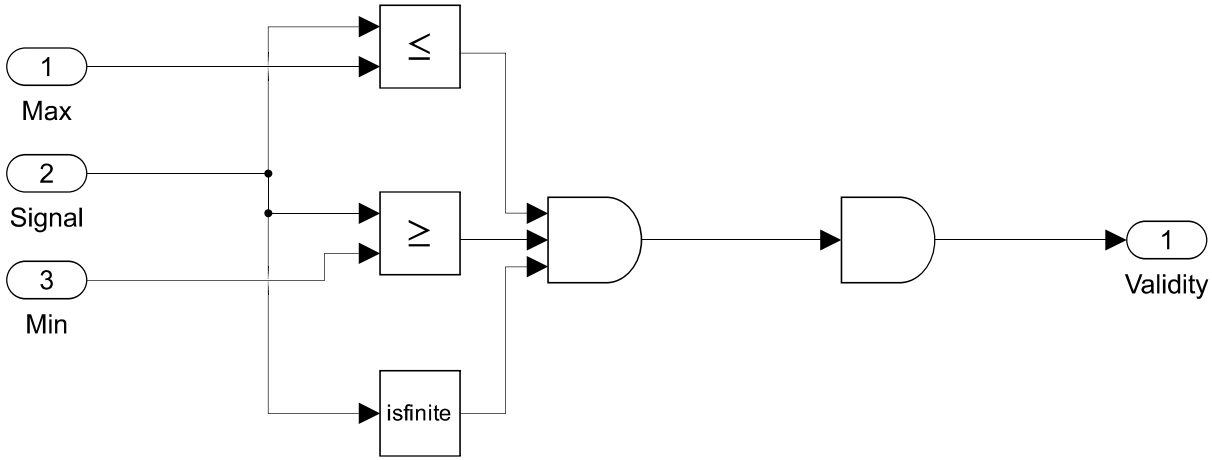

ones(6,1)
4
IMU_Raw
IMU
ones(3,1)
1
Mag_Raw
Mag
ones(5,1)
2
GPS_Raw
GPS
ones(1,1)
3
Baro_Alt_Raw
Baro_Alt
ones(1,1)
5
Pitot_Raw
Pitot

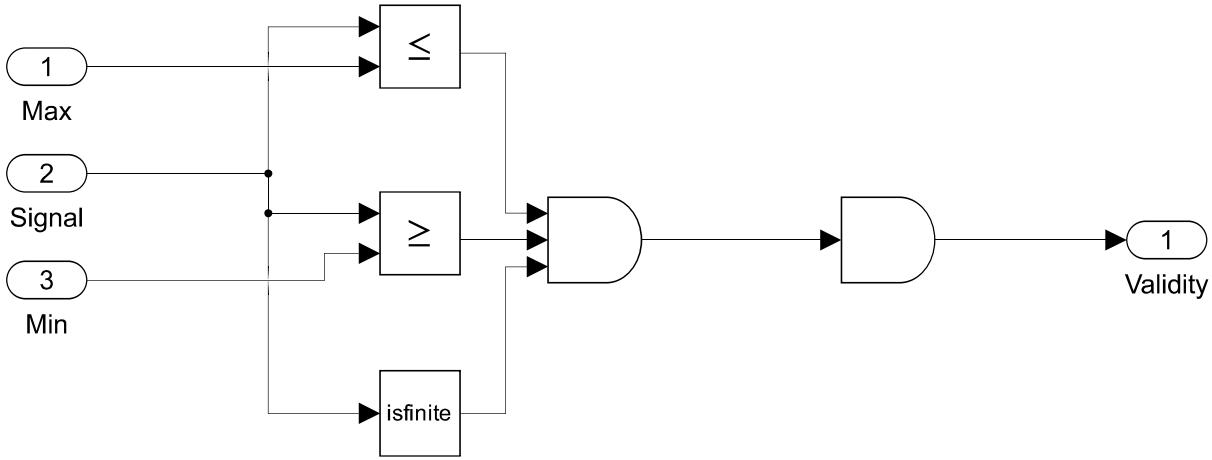

ones(6,1)
4
IMU_Raw
IMU
ones(3,1)
1
Mag_Raw
Mag
ones(5,1)
2
GPS_Raw
GPS
ones(1,1)
3
Baro_Alt_Raw
Baro_Alt
ones(1,1)
5
Pitot_Raw
Pitot

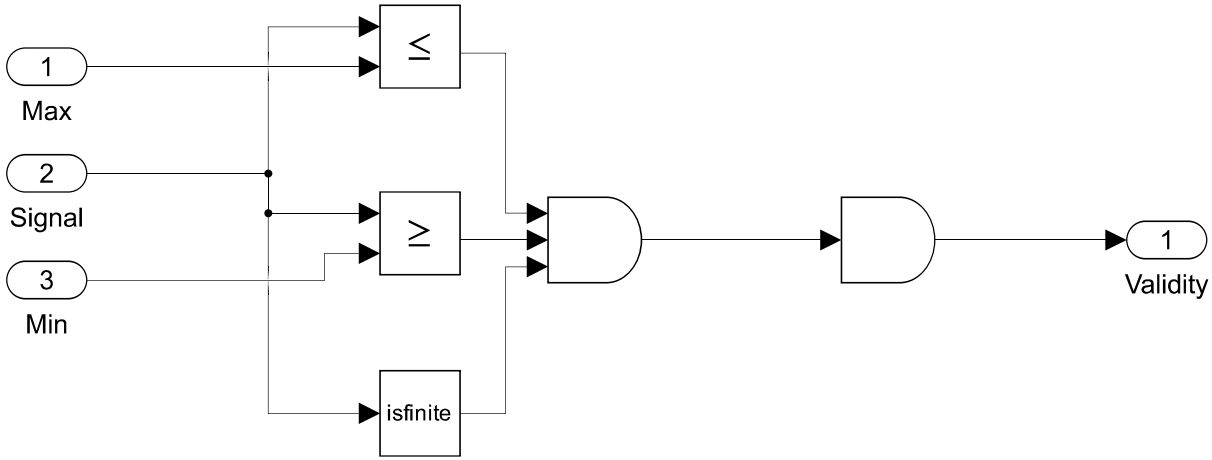

1
IMU_Raw_Bus
Lowpass
1
IMU_Raw_Conditioned
2
Mag_Raw_Bus
Lowpass
2
Mag_Raw_Conditioned
3
GPS_Raw_Bus
Lowpass
3
GPS_Raw_Conditioned
4
Baro_Alt_Raw_Bus
Lowpass
4
Baro_Alt_Raw_Conditioned
5
Pitot_Raw_Bus
Lowpass
5
Pitot_Raw_Conditioned

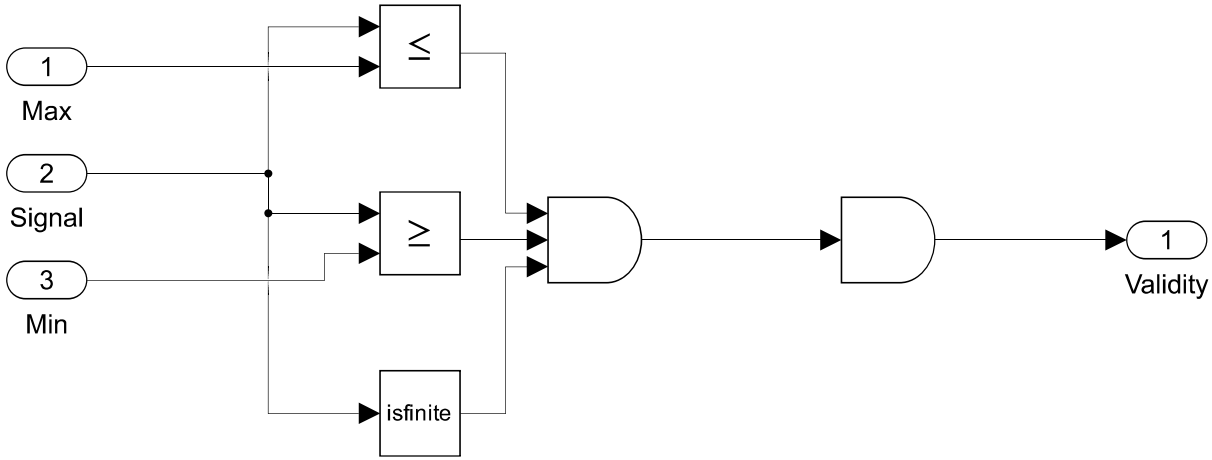

1e3
Dummy_Max
-1e3
Dummy_Min
1
IMU_Raw_Conditioned
Max
Signal Validity
Min
1
IMU_Validity
2
Mag_Raw_Conditioned
2
Mag_Validity
3
GPS_Raw_Conditioned
3
GPS_Validity
4
Baro_Alt_Raw_Conditioned
4
Baro_Alt_Validity
5
Pitot_Raw_Conditioned
5
Pitot_Validity

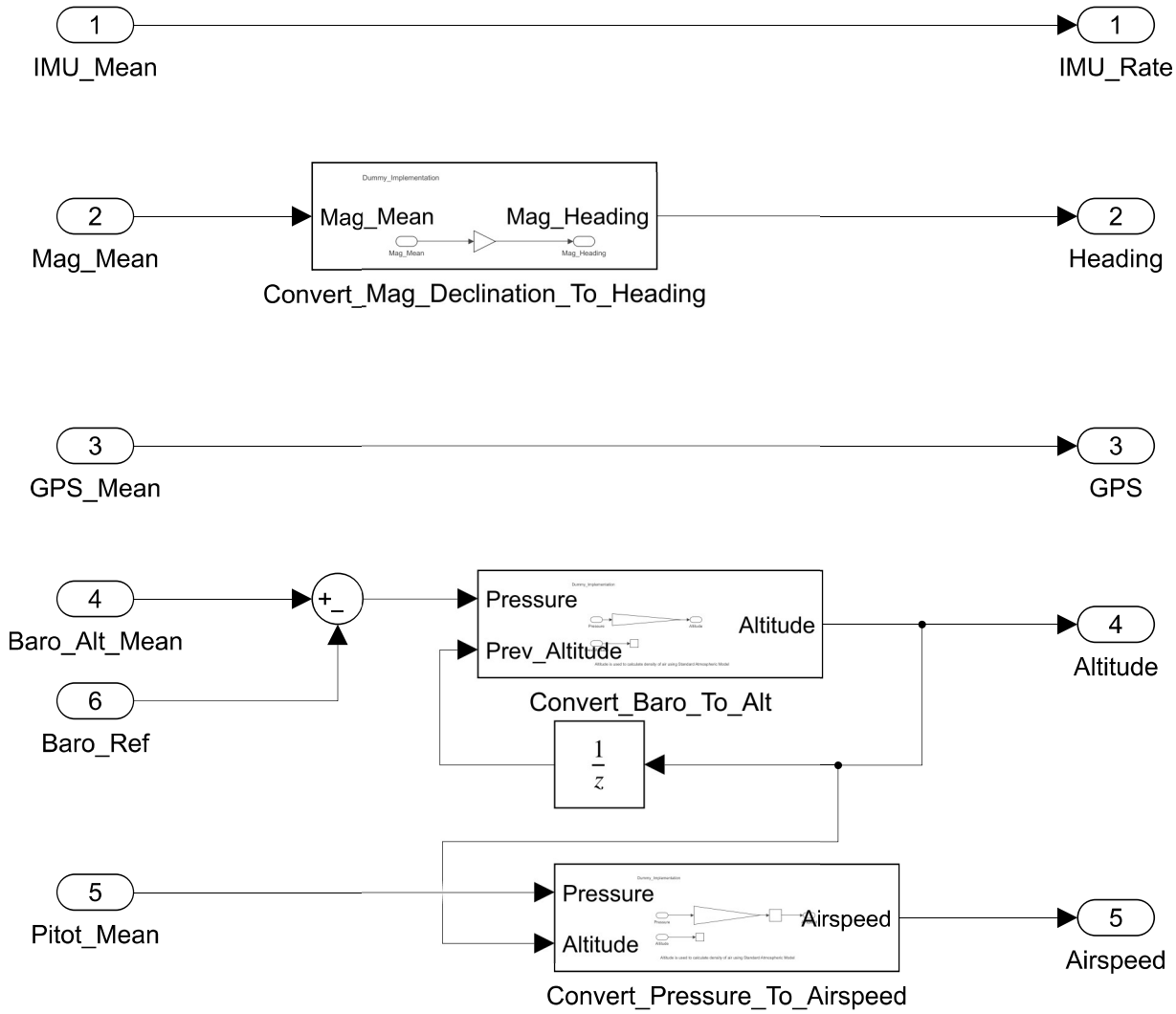

1
Max
2
Signal
3
Min
≤
≥
isfinite
1
Validity


1
Max
2
Signal
3
Min
≤
≥
isfinite
1
Validity

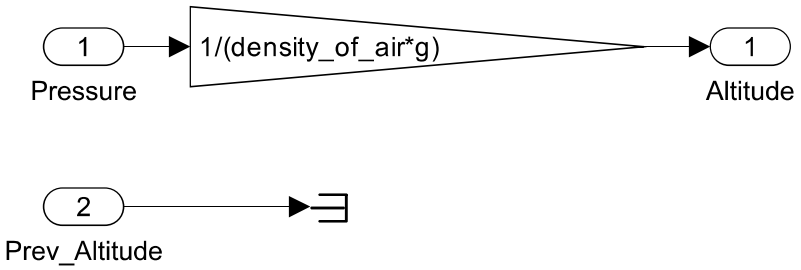

1
Max
2
Signal
3
Min
≤
≥
isfinite
1
Validity


1
Max
2
Signal
3
Min
≤
≥
isfinite
1
Validity


1
Max
2
Signal
3
Min
≤
≥
isfinite
1
Validity

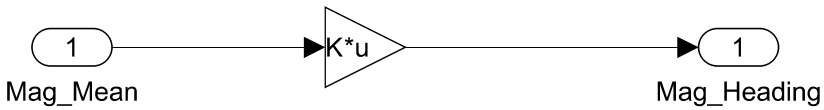

1
IMU_Mean
1
IMU_Rate
2
Mag_Mean
Dummy_Implementation
Mag_Mean
Mag_Heading
Mag_Mean
Mag_Heading
Convert_Mag_Declination_To_Heading
2
Heading
3
GPS_Mean
3
GPS
4
Baro_Alt_Mean
+_
6
Baro_Ref
Pressure
Prev_Altitude
Altitude
Convert_Baro_To_Alt
4
Altitude
1/z
5
Pitot_Mean
Pressure
Altitude
Airspeed
Convert_Pressure_To_Airspeed
5
Airspeed

Dummy_Implementation

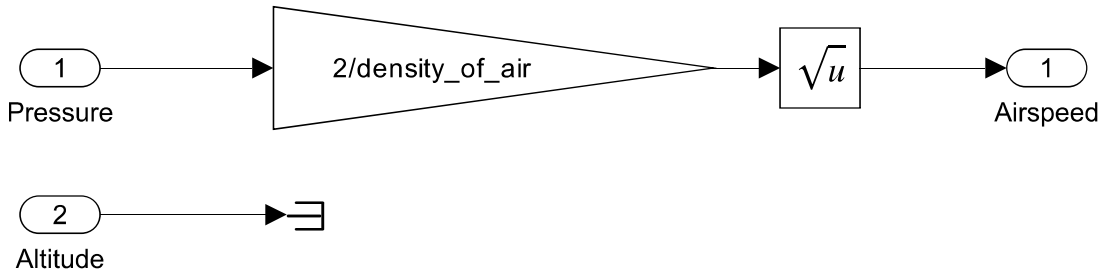


Altitude is used to calculate density of air using Standard Atmospheric Model

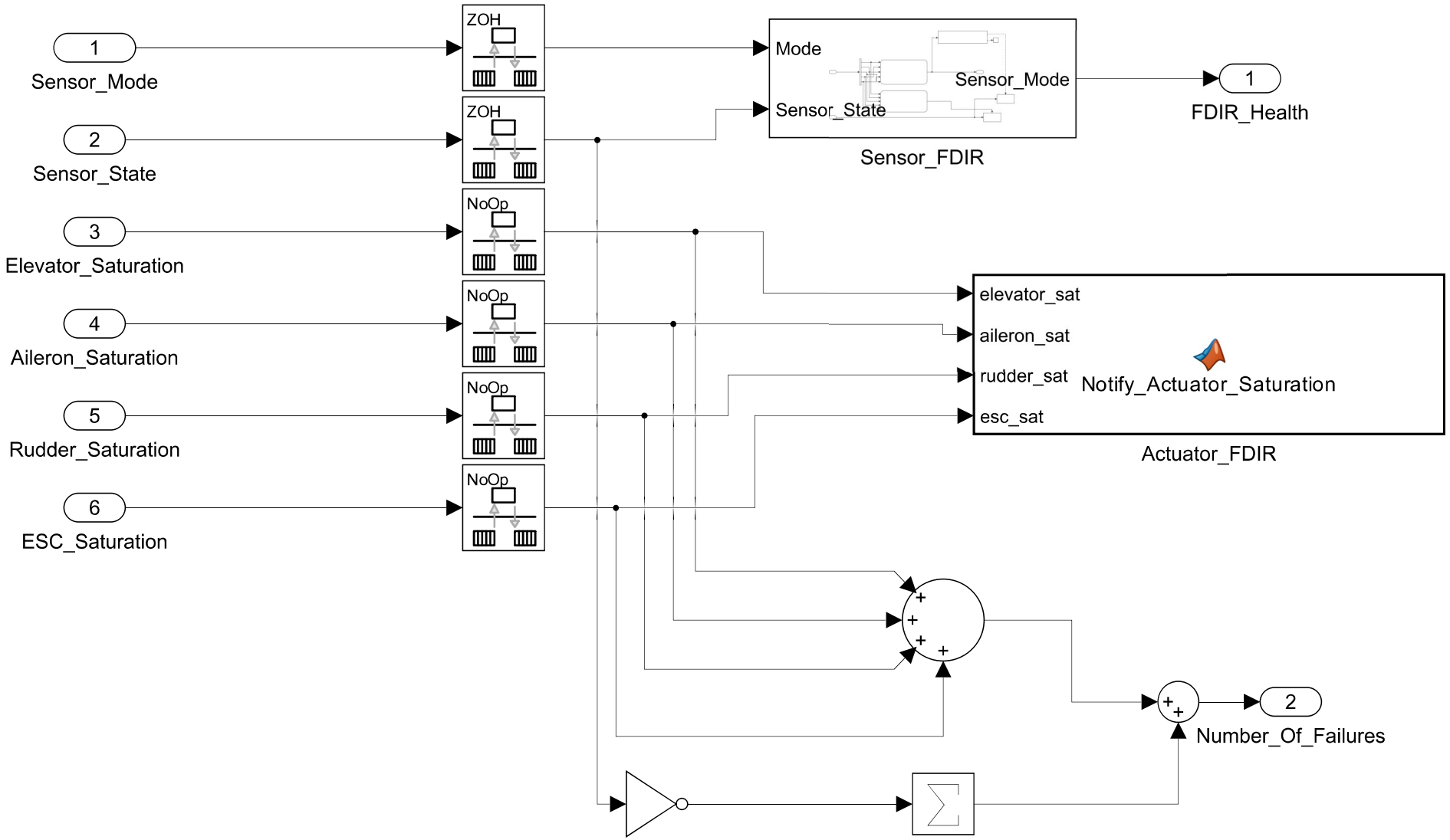
Dummy_Implementation



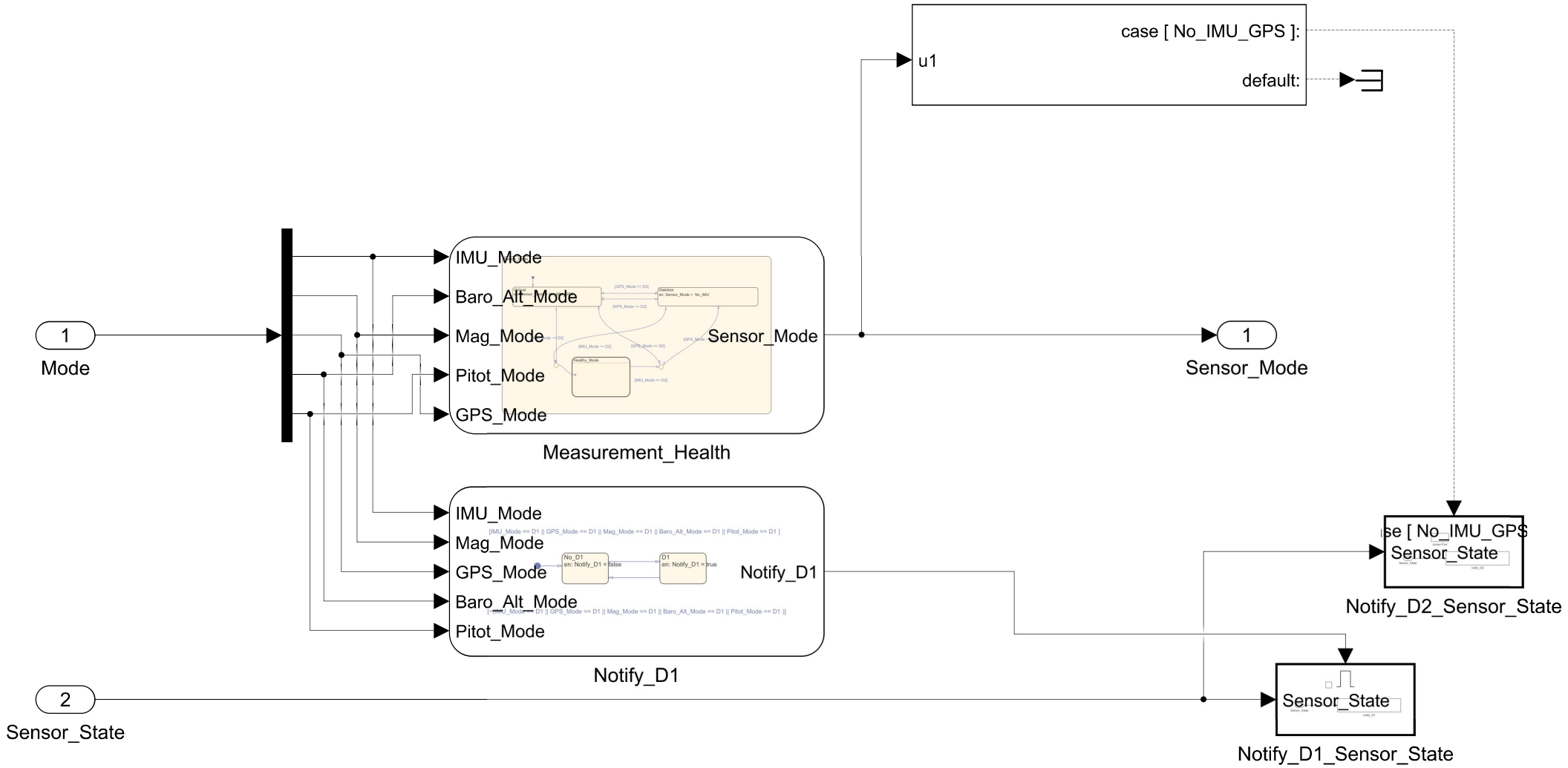
Dummy_Implementation

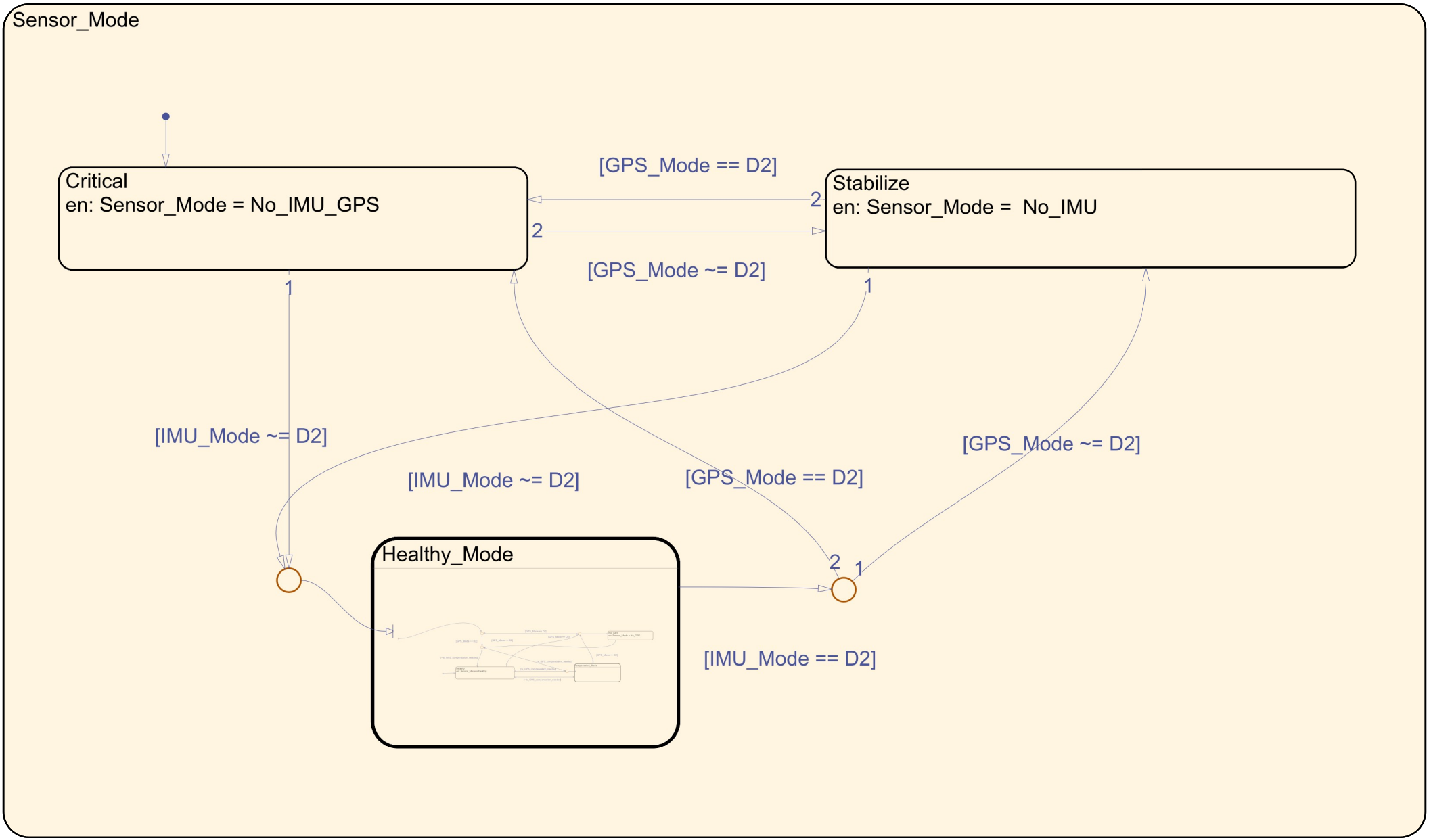


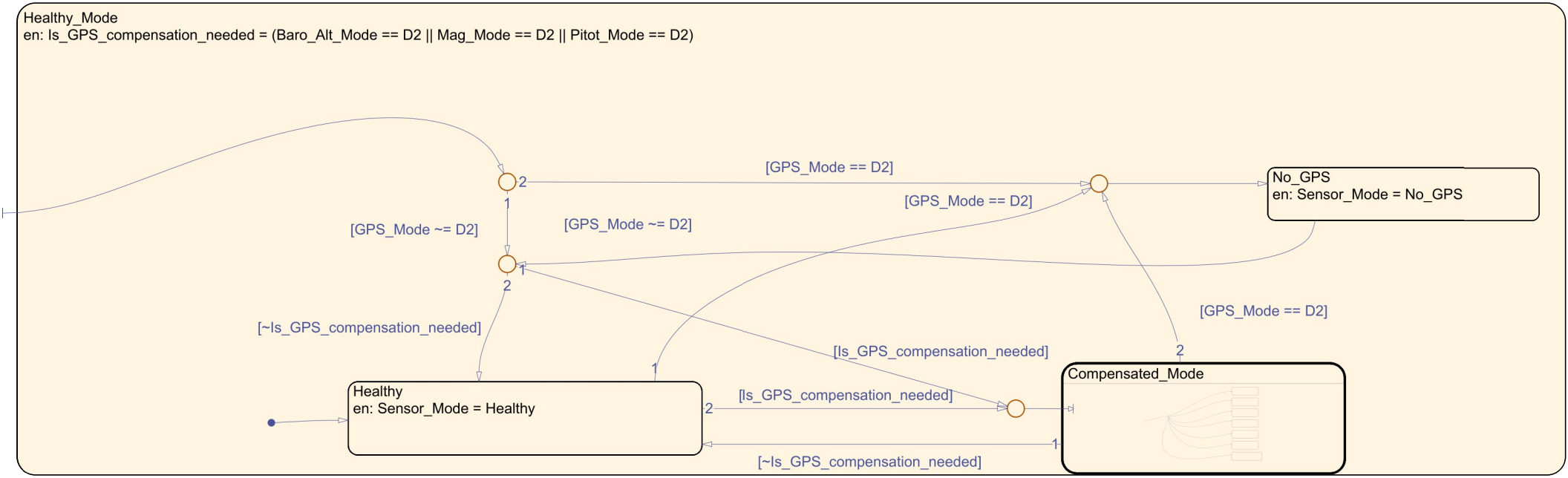
Altitude is used to calculate density of air using Standard Atmospheric Model

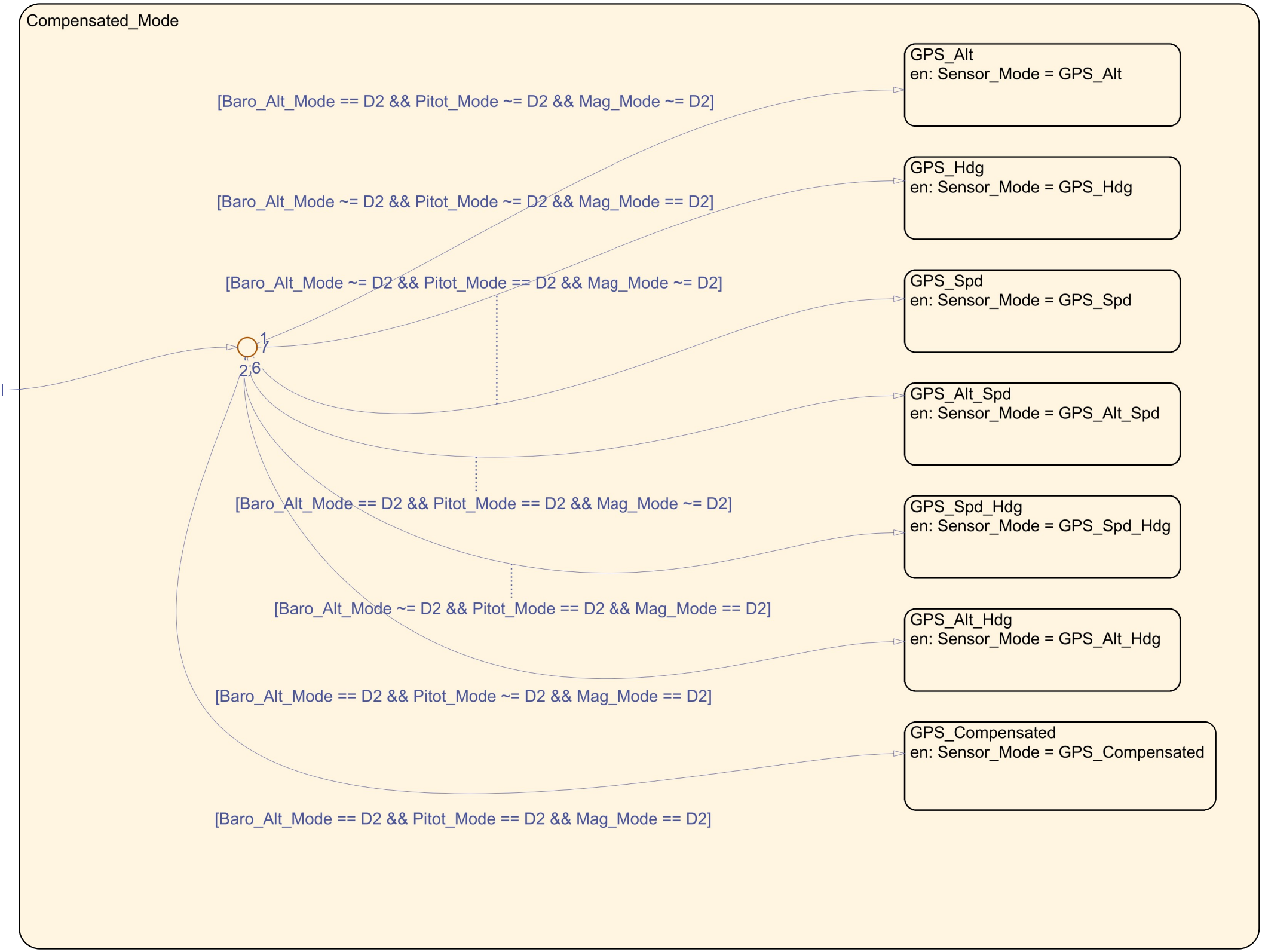

1
Sensor_Mode
2
Sensor_State
3
Elevator_Saturation
4
Aileron_Saturation
5
Rudder_Saturation
6
ESC_Saturation
ZOH
ZOH
NoOp
NoOp
NoOp
NoOp
Mode
Sensor_State
Sensor_Mode
Sensor_FDIR
1
FDIR_Health
elevator_sat
aileron_sat
rudder_sat
esc_sat
Notify_Actuator_Saturation
Actuator_FDIR
2
Number_Of_Failures

```
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
```

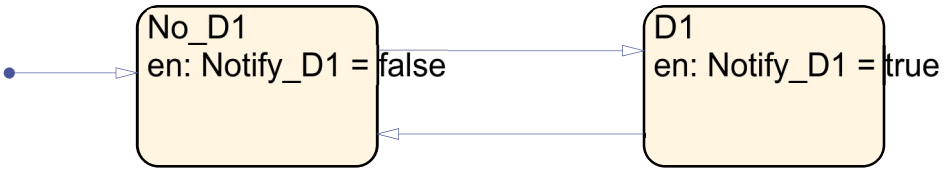

case [No_IMU_GPS]:
u1
default:
1
Mode
IMU_Mode
Baro_Alt_Mode
Mag_Mode
Pitot_Mode
GPS_Mode
Sensor_Mode
Measurement_Health
1
Sensor_Mode
IMU_Mode
Mag_Mode
GPS_Mode
Baro_Alt_Mode
Pitot_Mode
Notify_D1
Notify_D1
se [No_IMU_GPS
Sensor_State
Notify_D2_Sensor_State
2
Sensor_State
Sensor_State
Notify_D1_Sensor_State


Sensor_Mode
Critical
en: Sensor_Mode = No_IMU_GPS
Stabilize
en: Sensor_Mode = No_IMU
[GPS_Mode == D2]
[GPS_Mode ~= D2]
2
2
1
1
[IMU_Mode ~= D2]
[IMU_Mode ~= D2]
[GPS_Mode == D2]
[GPS_Mode ~= D2]
Healthy_Mode
2
1
[IMU_Mode == D2]

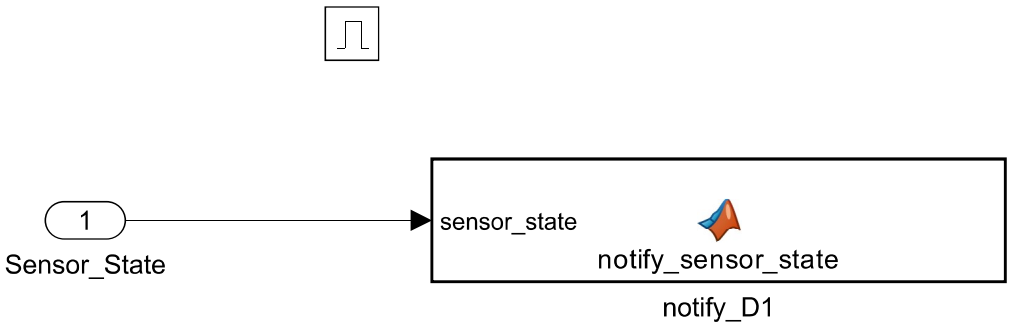

Healthy_Mode
en: Is_GPS_compensation_needed = (Baro_Alt_Mode == D2 || Mag_Mode == D2 || Pitot_Mode == D2)
[GPS_Mode == D2]
[GPS_Mode == D2]
[GPS_Mode ~= D2]
[GPS_Mode ~= D2]
No_GPS
en: Sensor_Mode = No_GPS
[GPS_Mode == D2]
[~Is_GPS_compensation_needed]
[Is_GPS_compensation_needed]
Compensated_Mode
Healthy
en: Sensor_Mode = Healthy
[Is_GPS_compensation_needed]
[~Is_GPS_compensation_needed]


Compensated_Mode
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
[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]
1
2
6
7

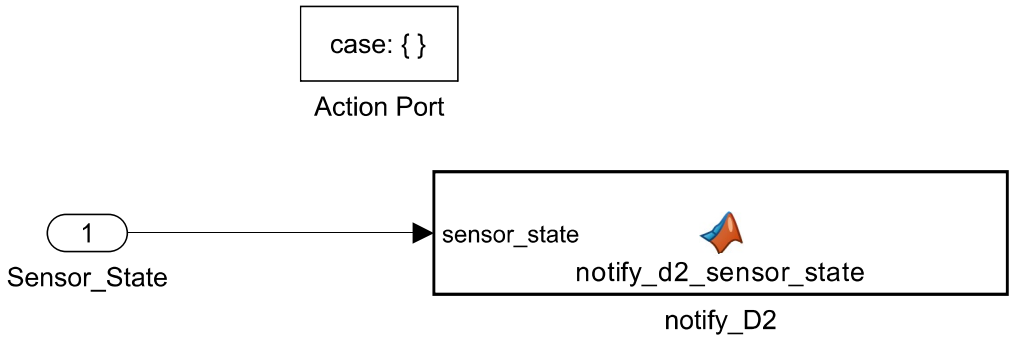
[IMU_Mode == D1 || GPS_Mode == D1 || Mag_Mode == D1 || Baro_Alt_Mode == D1 || Pitot_Mode == D1]



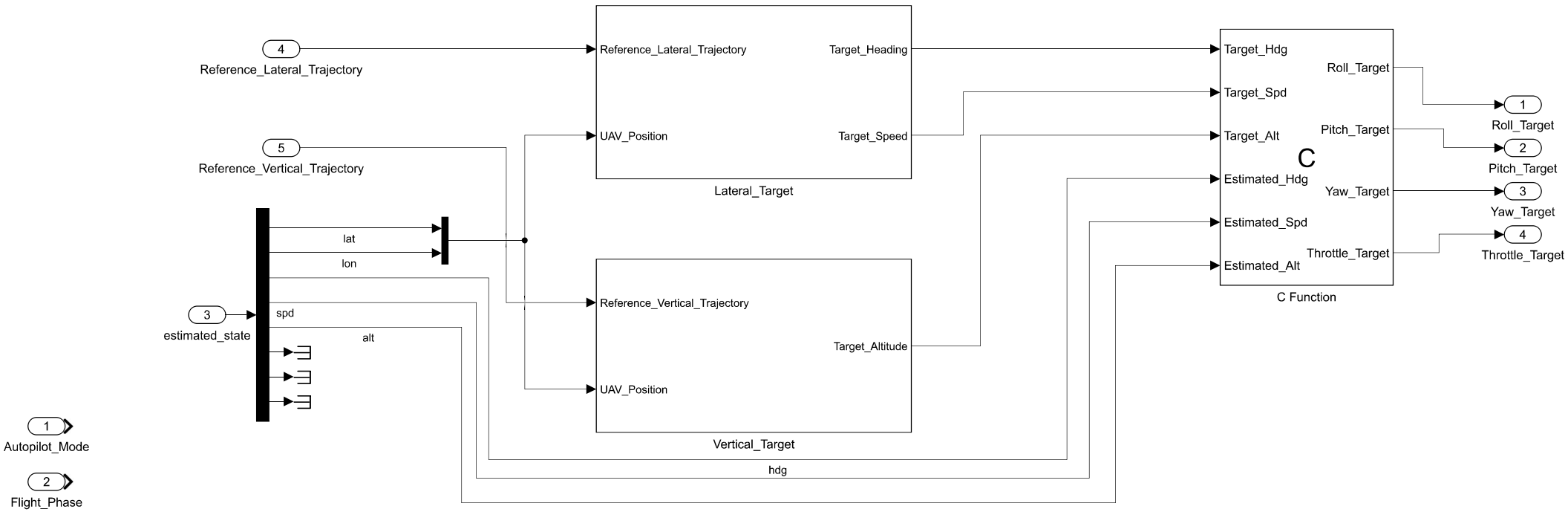
[~(IMU_Mode == D1 || GPS_Mode == D1 || Mag_Mode == D1 || Baro_Alt_Mode == D1 || Pitot_Mode == D1)]

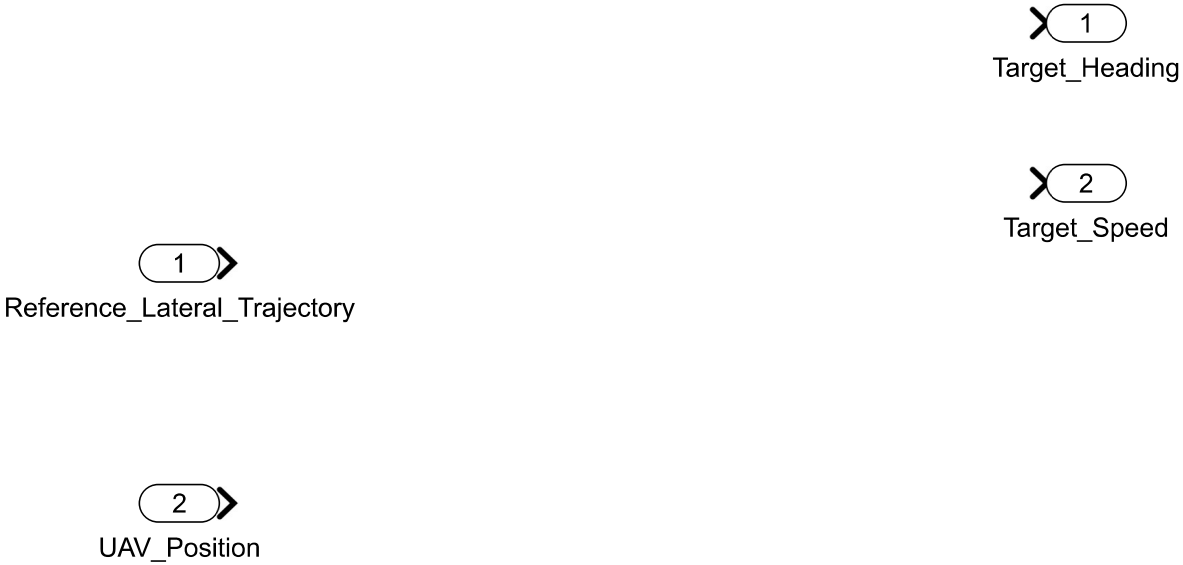

1
Sensor_State
sensor_state
notify_sensor_state
notify_D1

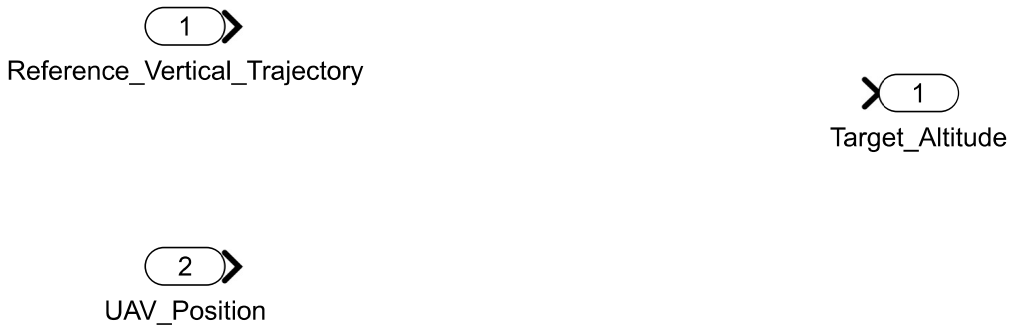
```
function notify_sensor_state(sensor_state)

    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
```

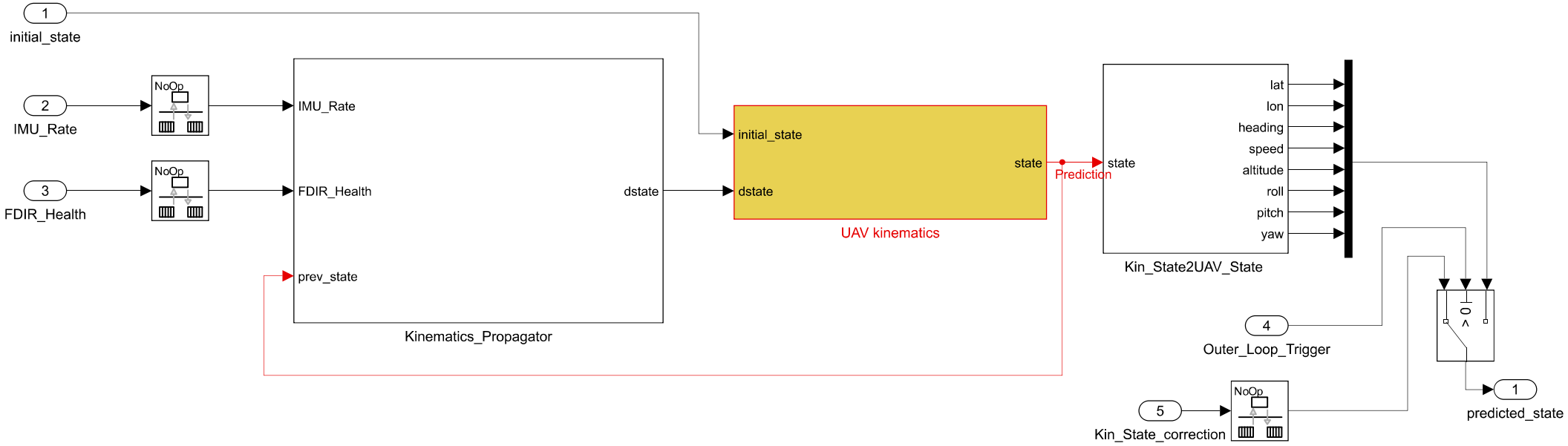

case: { }
Action Port
1
Sensor_State
sensor_state
notify_d2_sensor_state
notify_D2

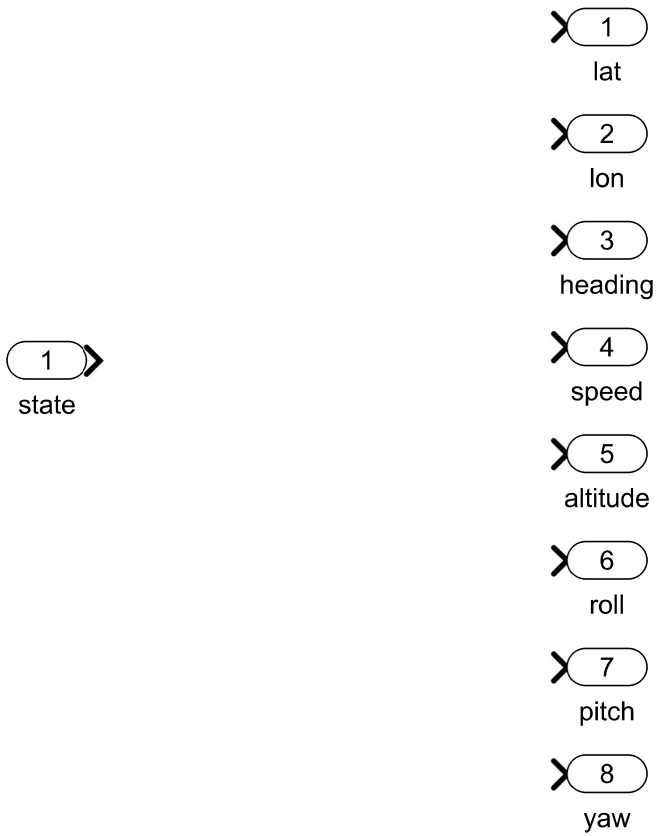
```
function notify_d2_sensor_state(sensor_state)

    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
```

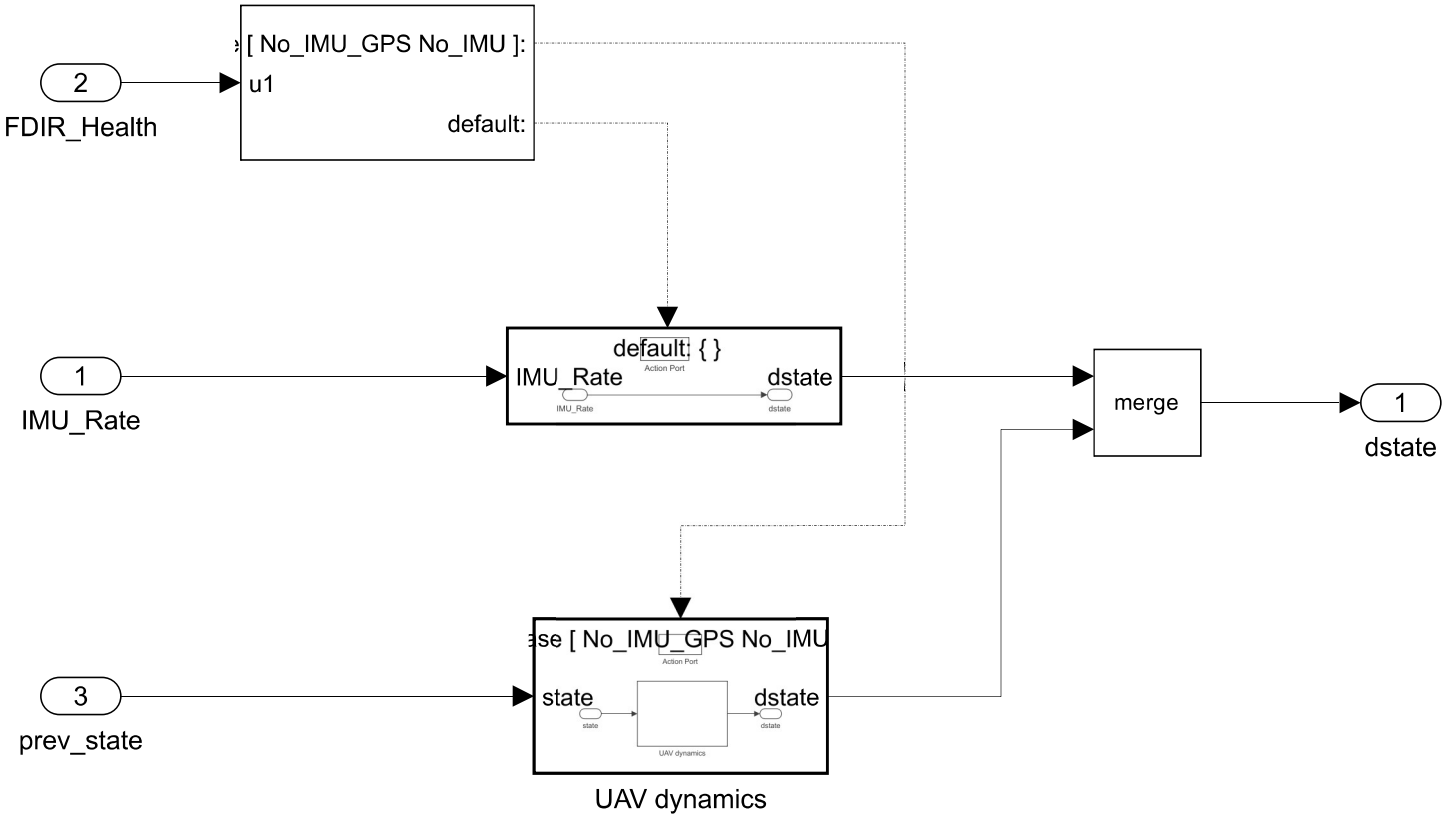

4
Reference_Lateral_Trajectory
5
Reference_Vertical_Trajectory
3
estimated_state
lat
lon
spd
alt
Reference_Lateral_Trajectory
UAV_Position
Target_Heading
Target_Speed
Lateral_Target
Reference_Vertical_Trajectory
UAV_Position
Target_Altitude
Vertical_Target
hdg
Target_Hdg
Target_Spd
Target_Alt
Estimated_Hdg
Estimated_Spd
Estimated_Alt
C
Roll_Target
Pitch_Target
Yaw_Target
Throttle_Target
C Function
1
Roll_Target
2
Pitch_Target
3
Yaw_Target
4
Throttle_Target
1
Autopilot_Mode
2
Flight_Phase

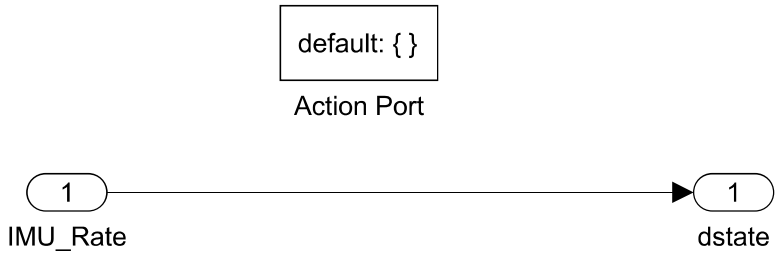

1
Target_Heading
2
Target_Speed
1
Reference_Lateral_Trajectory
2
UAV_Position

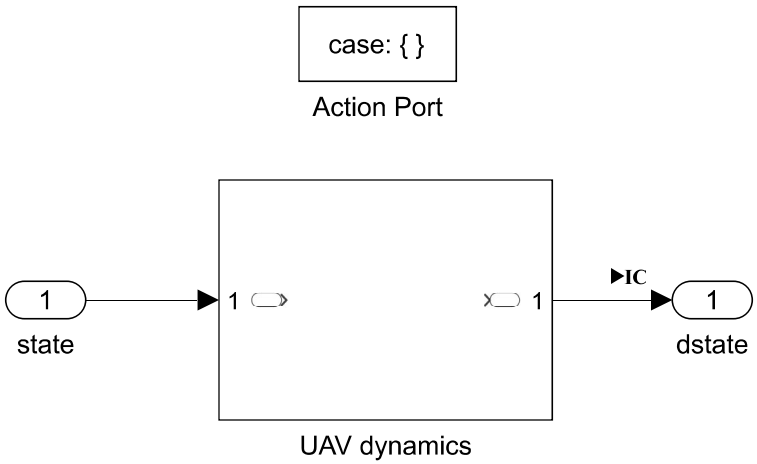

1
Reference_Vertical_Trajectory
1
Target_Altitude
2
UAV_Position

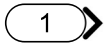

1
initial_state
2
IMU_Rate
NoOp
3
FDIR_Health
NoOp
IMU_Rate
FDIR_Health
prev_state
dstate
Kinematics_Propagator
initial_state
dstate
state
UAV kinematics
Prediction
state
lat
lon
heading
speed
altitude
roll
pitch
yaw
Kin_State2UAV_State
4
Outer_Loop_Trigger
0
5
Kin_State_correction
NoOp
1
predicted_state

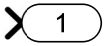

1
state
1
lat
2
lon
3
heading
4
speed
5
altitude
6
roll
7
pitch
8
yaw

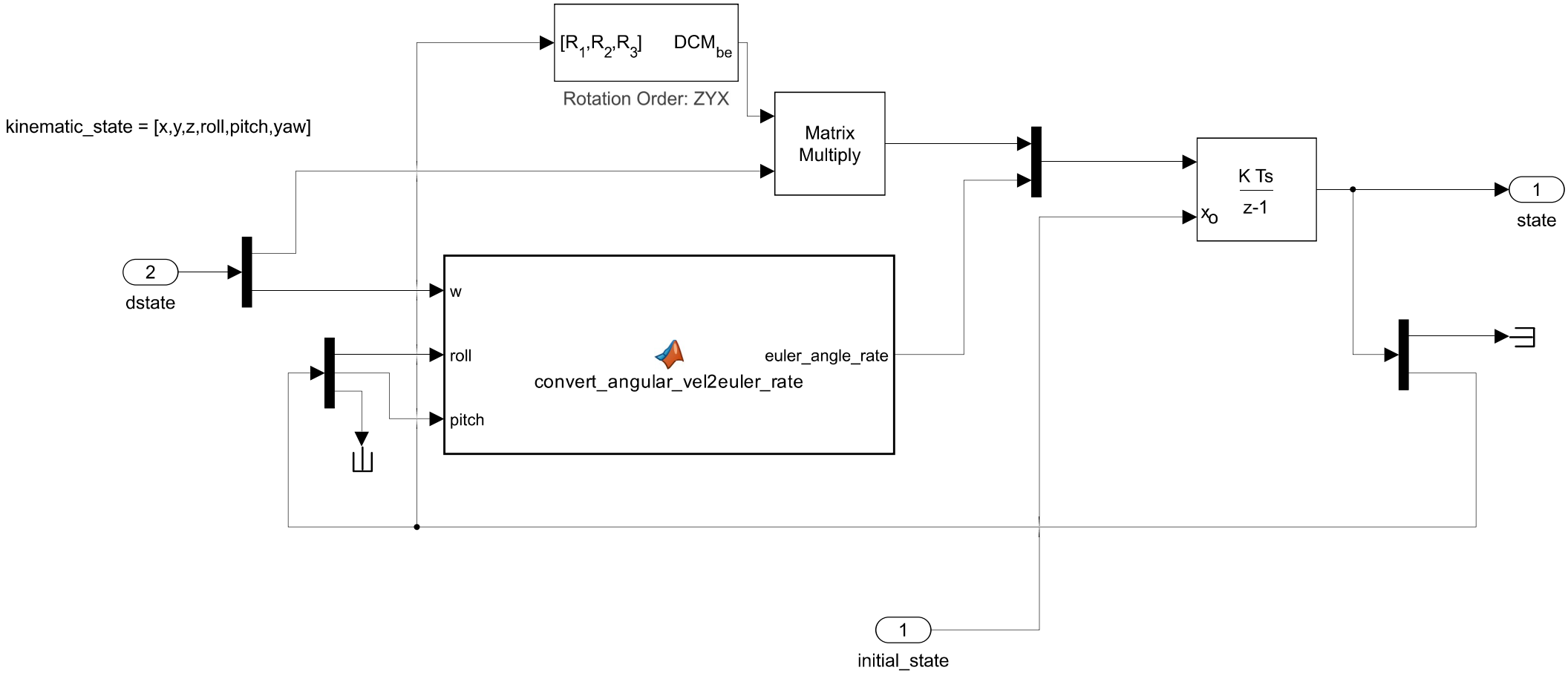

2
FDIR_Health
[No_IMU_GPS No_IMU]:
u1
default:
1
IMU_Rate
default: { }
Action Port
IMU_Rate
dstate
merge
1
dstate
3
prev_state
se [No_IMU_GPS No_IMU
Action Port
state
dstate
UAV dynamics
UAV dynamics


default: { }
Action Port
1
IMU_Rate
1
dstate

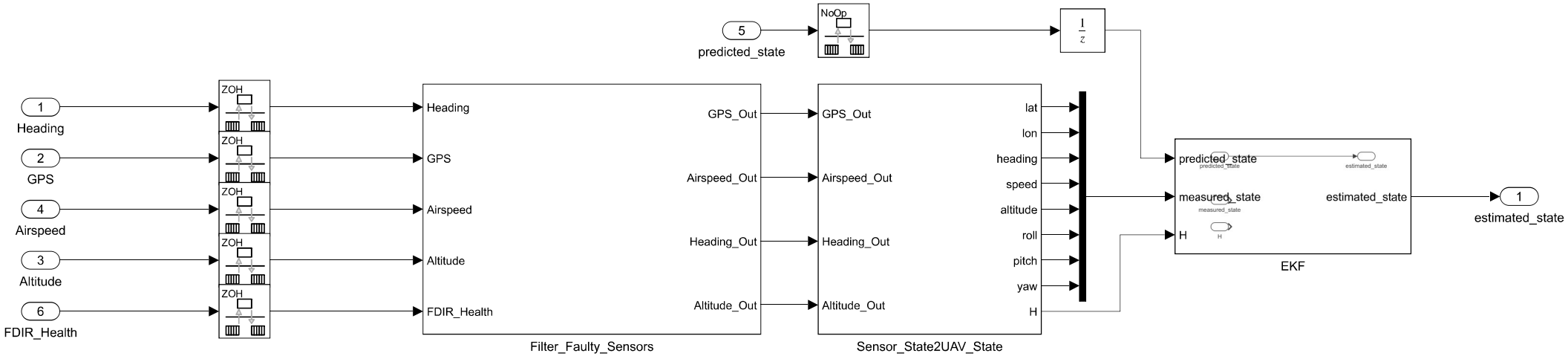

case: { }
Action Port
1
state
1
1
IC
1
dstate
UAV dynamics


1

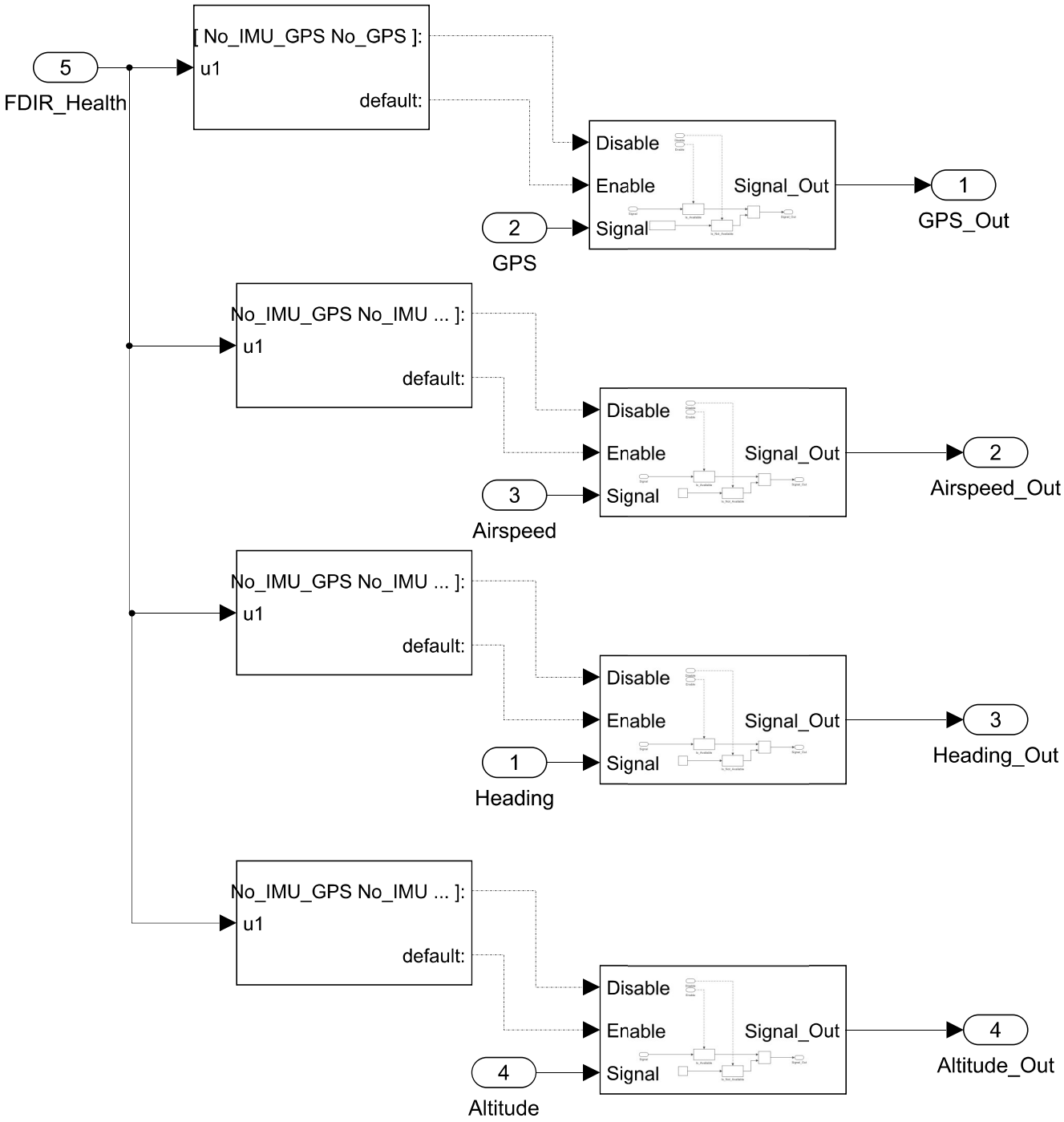

1

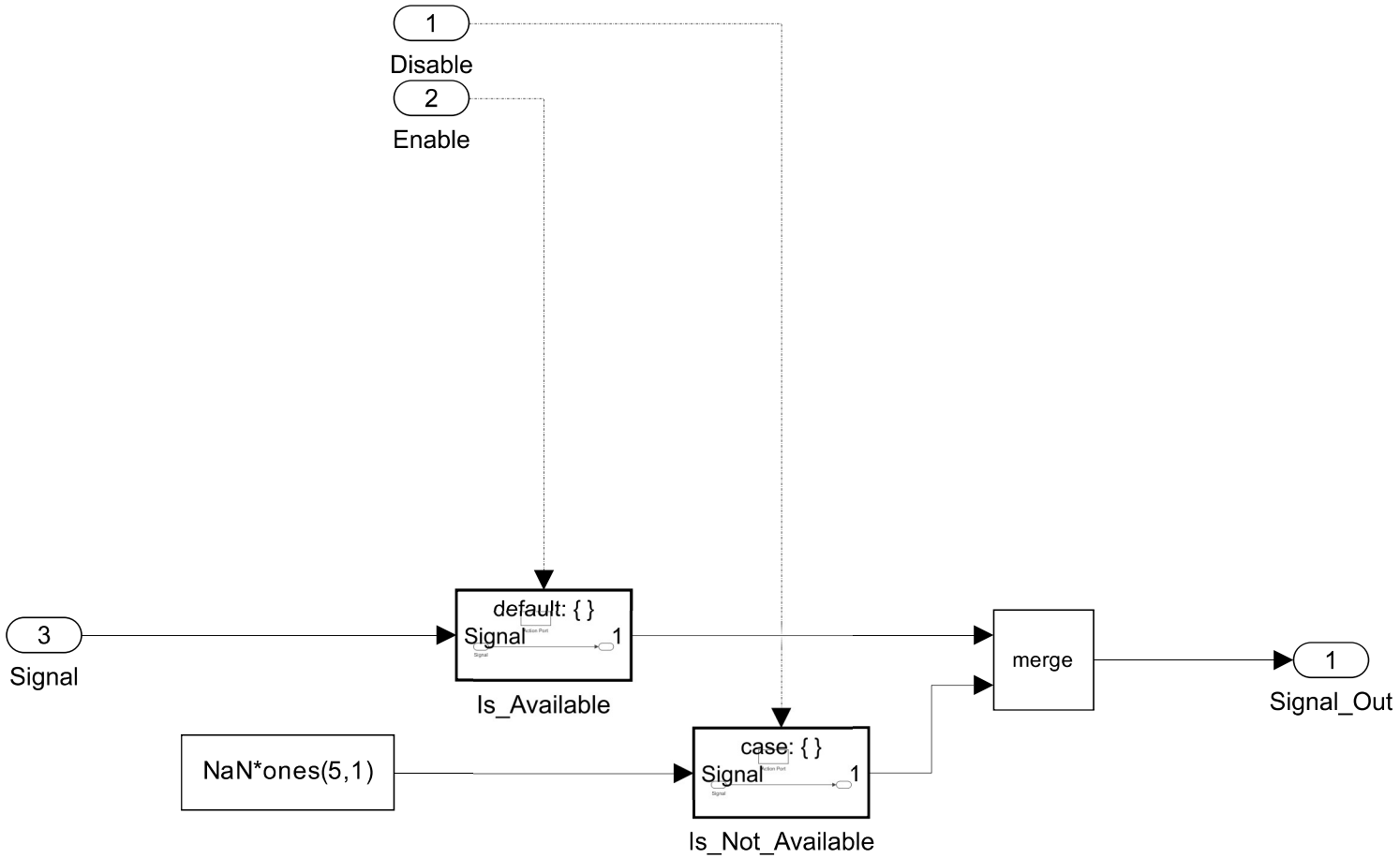

[R1,R2,R3]
DCMbe
Rotation Order: ZYX
kinematic_state = [x,y,z,roll,pitch,yaw]
Matrix
Multiply
K Ts
z-1
xo
1
state
2
dstate
w
roll
pitch
euler_angle_rate
convert_angular_vel2euler_rate
1
initial_state

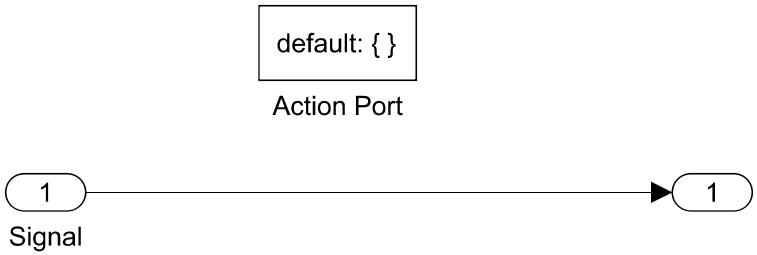
```
function euler_angle_rate = convert_angular_vel2euler_rate(w, roll, pitch)
p = w(1);
q = w(2);
r = w(3);

euler_angle_rate = [p + (q*sin(roll)+r*cos(roll))*tan(pitch),
                    q*cos(roll)-r*sin(roll),
                    (q*sin(roll)+r*cos(roll)/cos(pitch))];
```

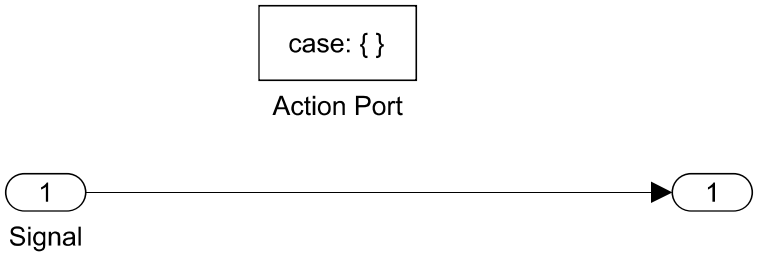

1
Heading
2
GPS
4
Airspeed
3
Altitude
6
FDIR_Health
ZOH
ZOH
ZOH
ZOH
ZOH
Heading
GPS
Airspeed
Altitude
FDIR_Health
GPS_Out
Airspeed_Out
Heading_Out
Altitude_Out
Filter_Faulty_Sensors
5
predicted_state
NoOp
1
z
GPS_Out
Airspeed_Out
Heading_Out
Altitude_Out
Sensor_State2UAV_State
lat
lon
heading
speed
altitude
roll
pitch
yaw
H
predicted_state
measured_state
H
estimated_state
EKF
1
estimated_state

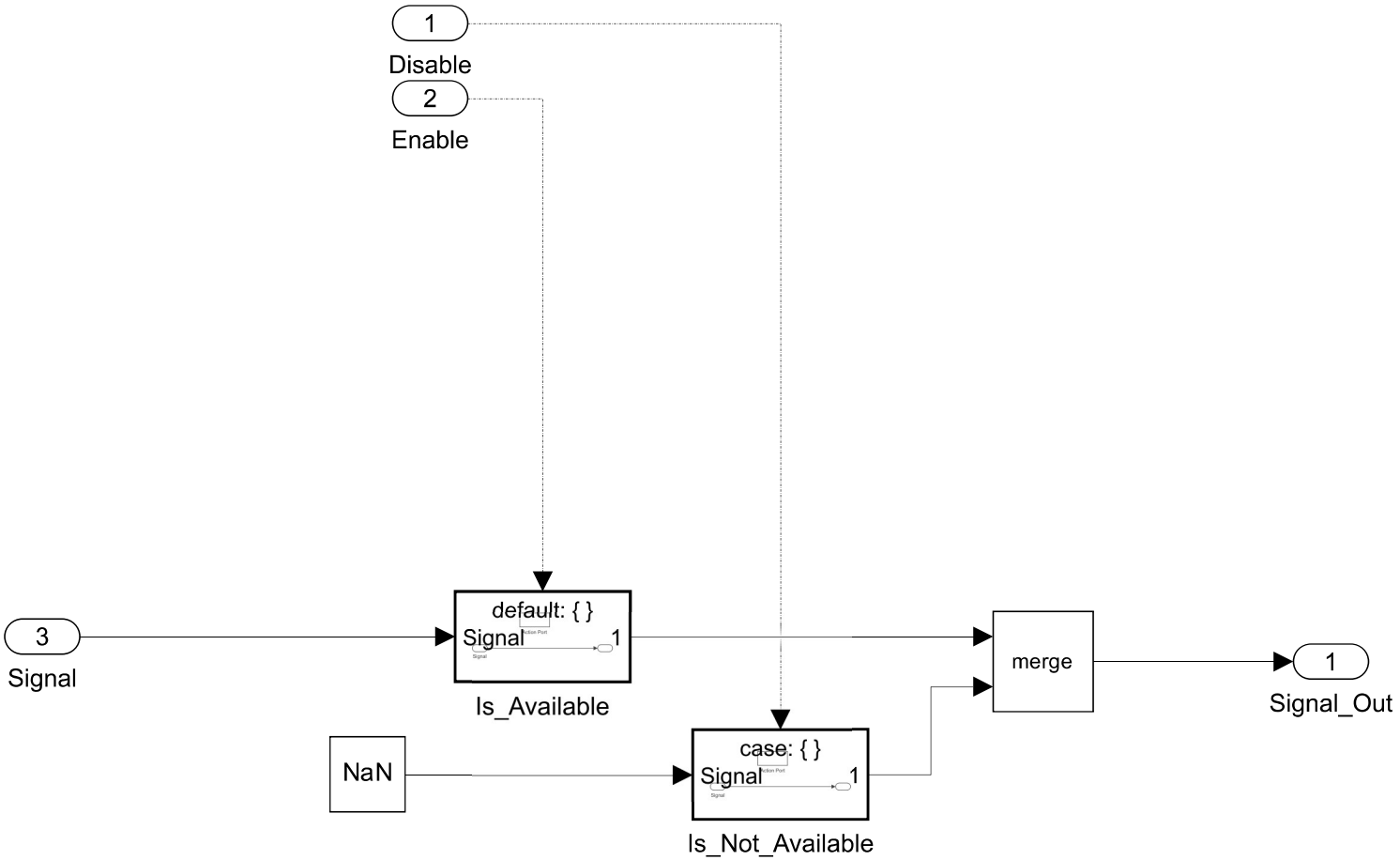

1
predicted_state
1
estimated_state
2
measured_state
3
H

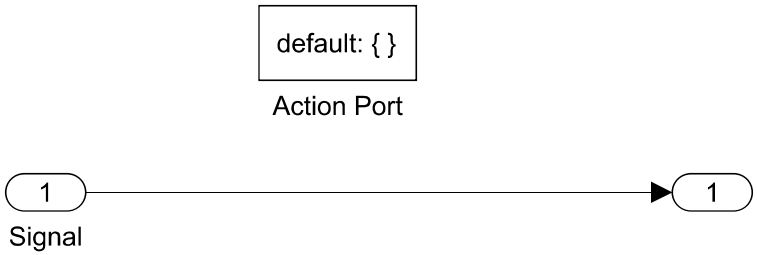

5
FDIR_Health
[No_IMU_GPS No_GPS]:
u1
default:
Disable
Enable
Signal
Signal_Out
2
GPS
1
GPS_Out
No_IMU_GPS No_IMU ...]:
u1
default:
Disable
Enable
Signal
Signal_Out
3
Airspeed
2
Airspeed_Out
No_IMU_GPS No_IMU ...]:
u1
default:
Disable
Enable
Signal
Signal_Out
1
Heading
3
Heading_Out
No_IMU_GPS No_IMU ...]:
u1
default:
Disable
Enable
Signal
Signal_Out
4
Altitude
4
Altitude_Out

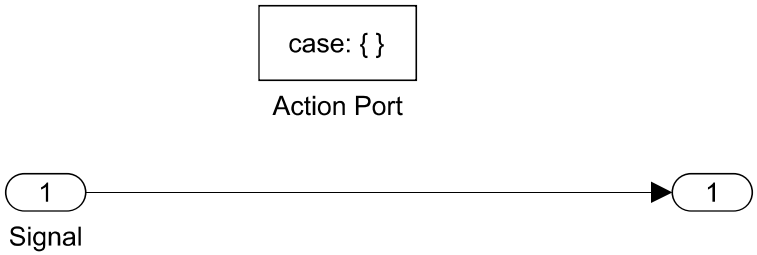

1
Disable
2
Enable
3
Signal
default: { }
Signal
1
Is_Available
NaN*ones(5,1)
case: { }
Signal
1
Is_Not_Available
merge
1
Signal_Out

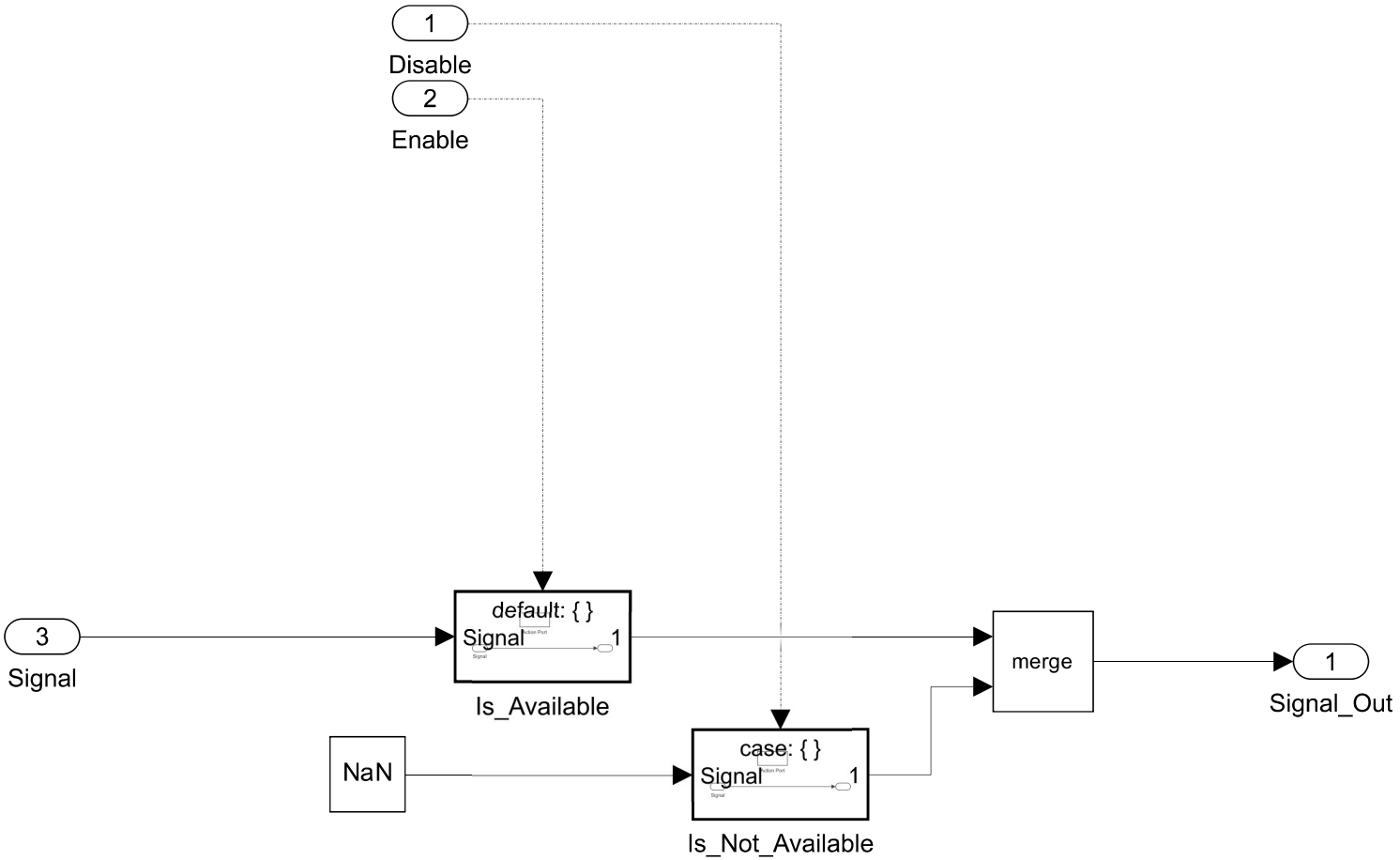

default: { }
Action Port
1
Signal
1

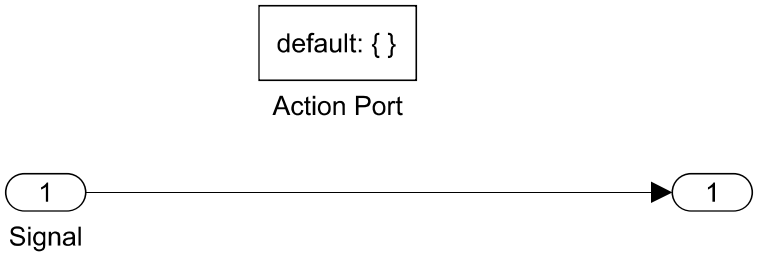

case: { }
Action Port
1
Signal
1

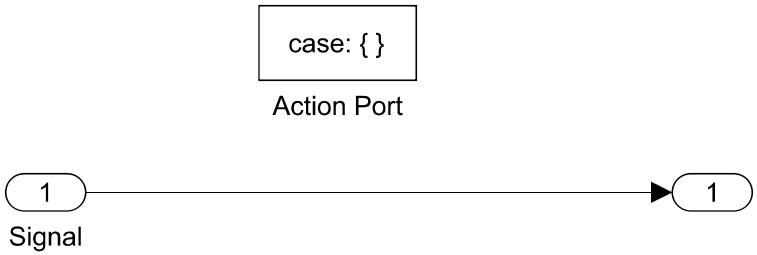

1
Disable
2
Enable
3
Signal
default: { }
Signal
1
Is_Available
NaN
case: { }
Signal
1
Is_Not_Available
merge
1
Signal_Out

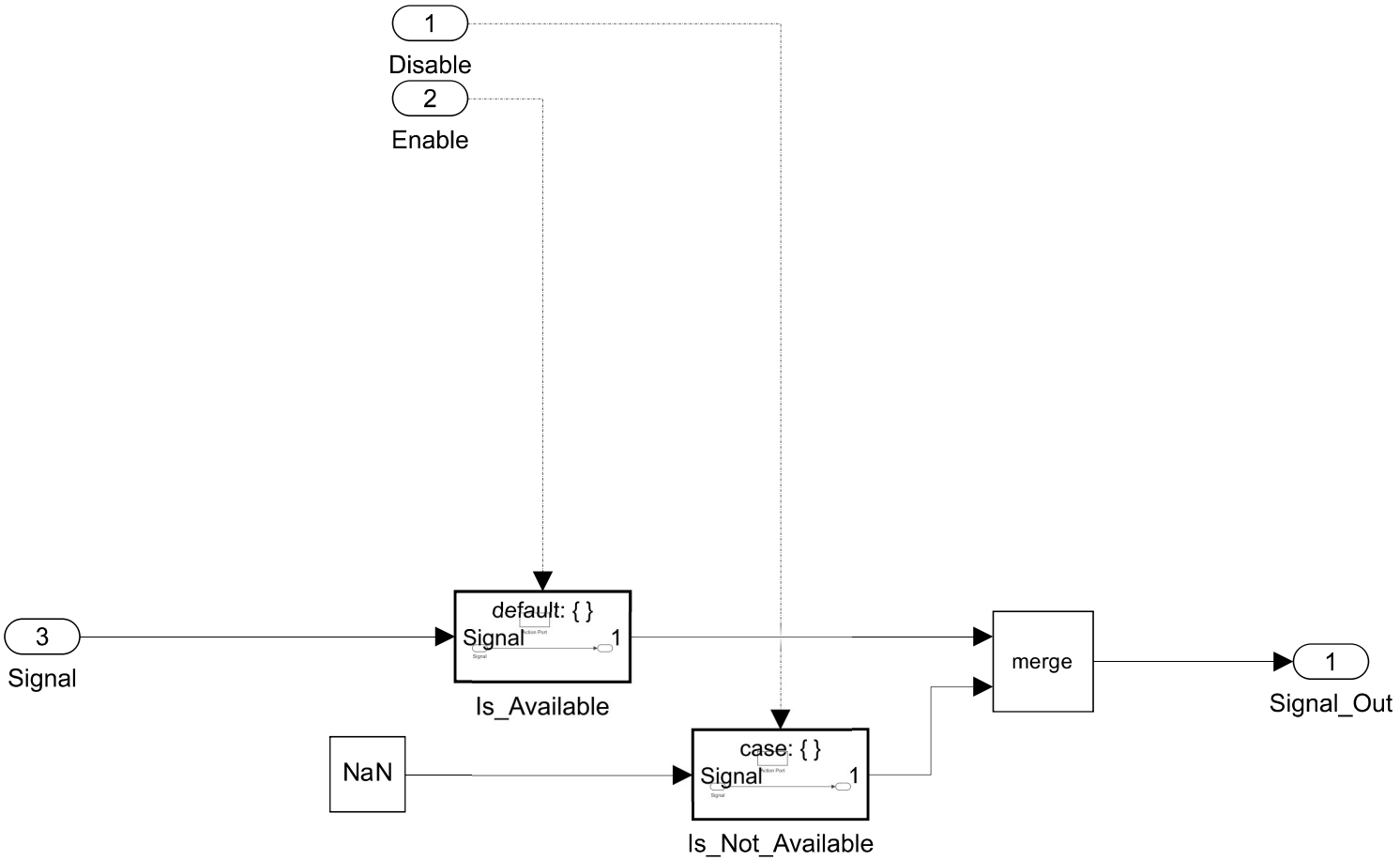

default: { }
Action Port
1
Signal
1

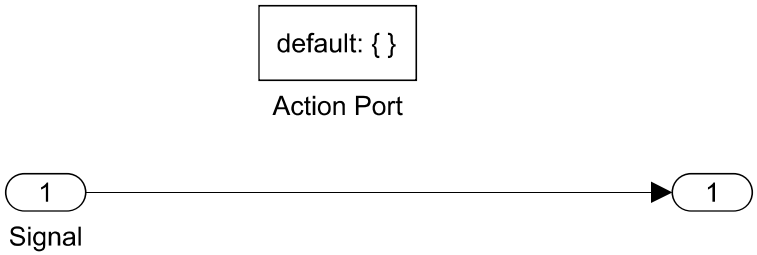

case: { }
Action Port
1
Signal
1

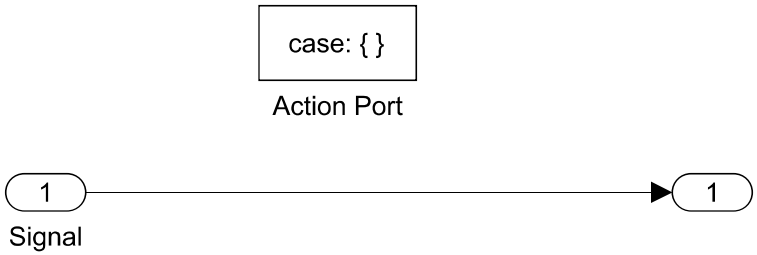

1
Disable
2
Enable
3
Signal
default: { }
Signal
1
Is_Available
NaN
case: { }
Signal
1
Is_Not_Available
merge
1
Signal_Out


default: { }
Action Port
1
Signal
1


case: { }
Action Port
1
Signal
1


1
Disable
2
Enable
3
Signal
default: { }
Signal
1
Is_Available
NaN
case: { }
Signal
1
Is_Not_Available
merge
1
Signal_Out

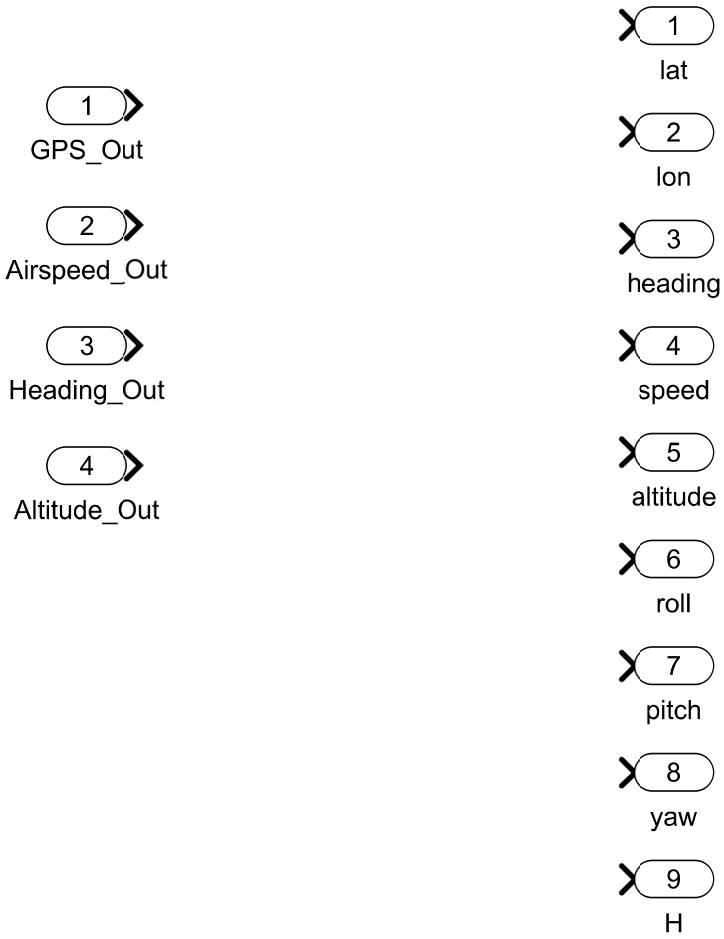

default: { }
Action Port
1
Signal
1

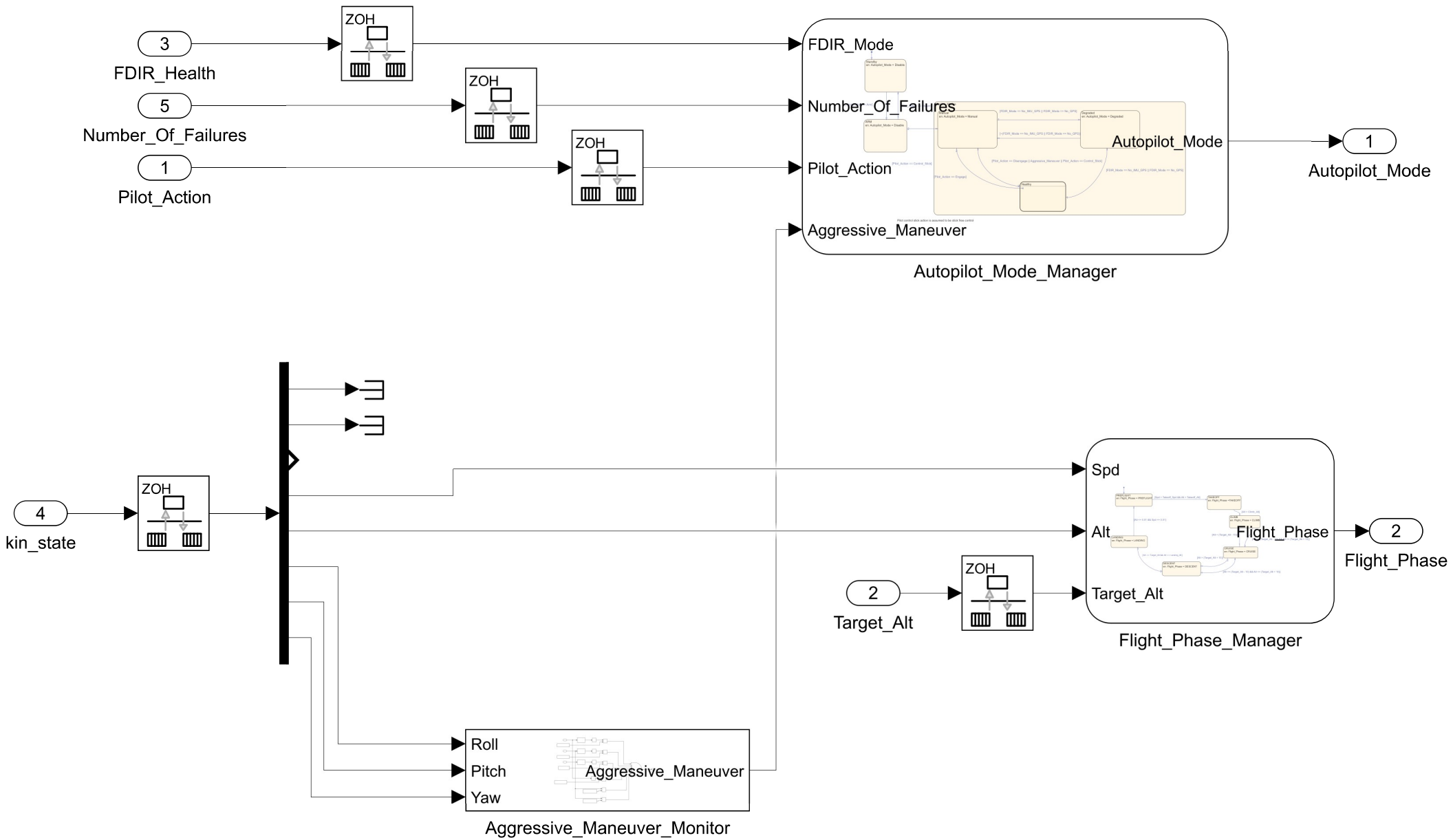

case: { }
Action Port
1
Signal
1

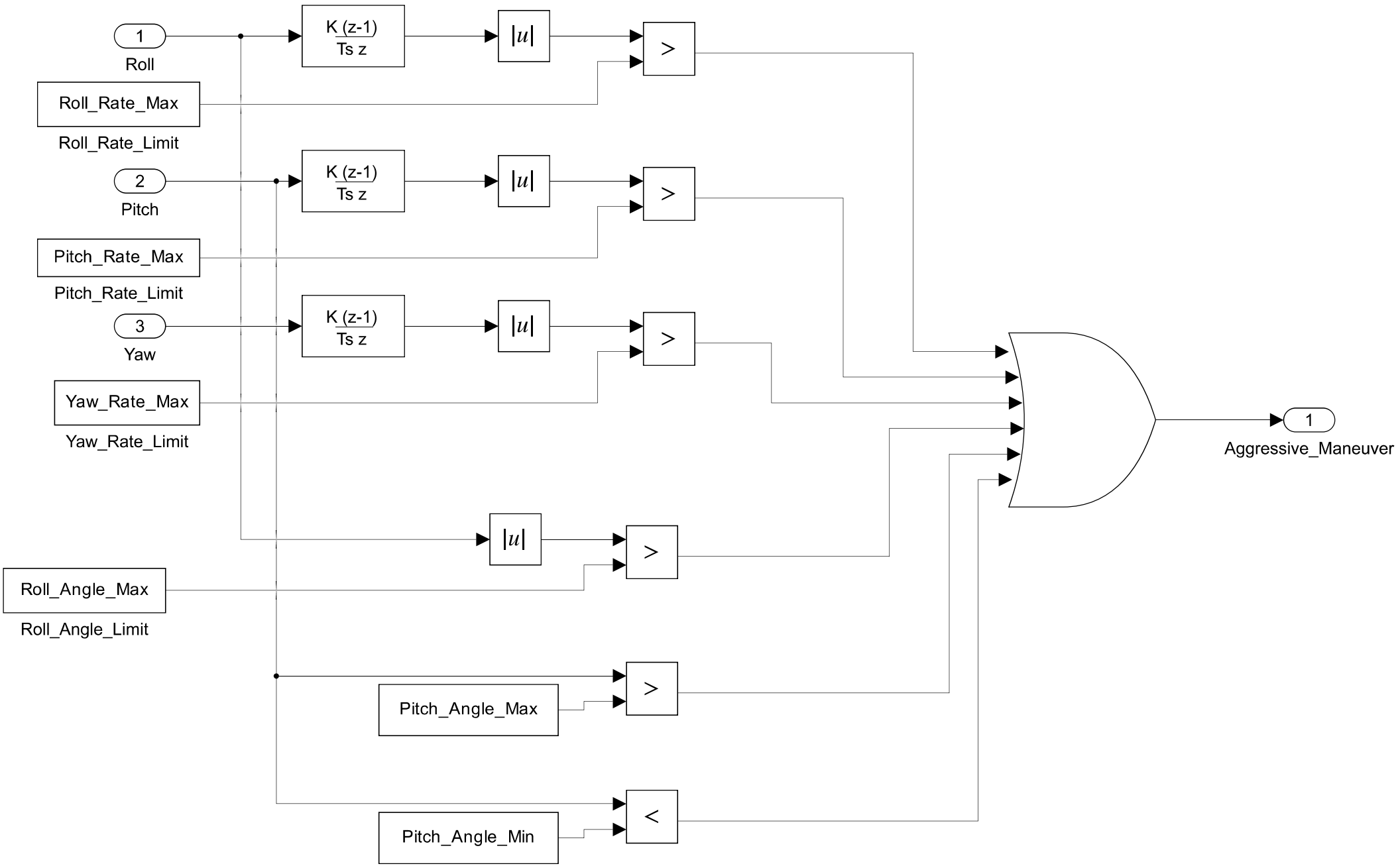
Dummy_Implementation

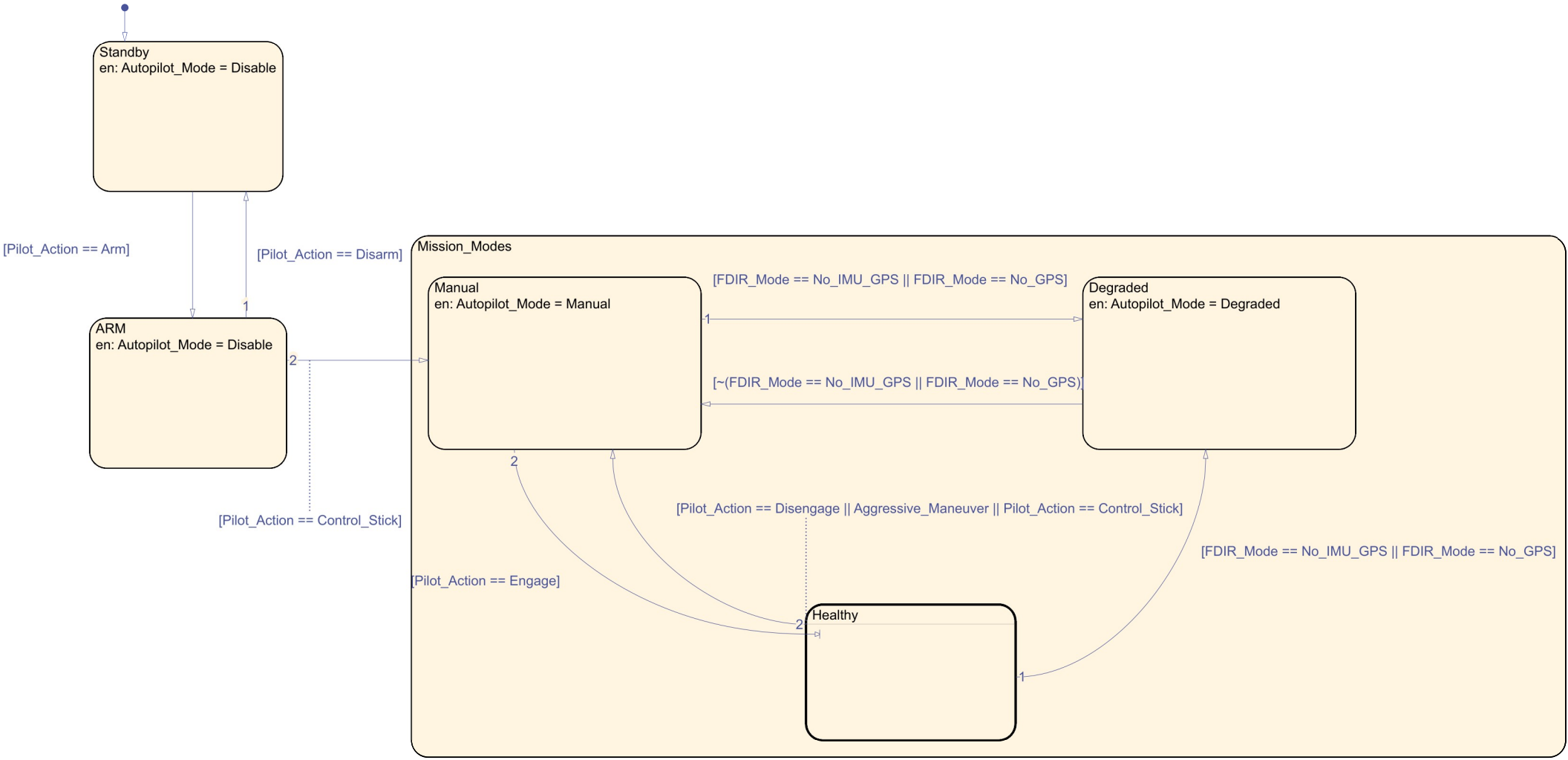
Logic needs to be implemented to pick the sensor fusion quantities based on quality of the data (Airspeed from pitot tube is more preferable than GPS speed when both are available) and filter out the NaN signals

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

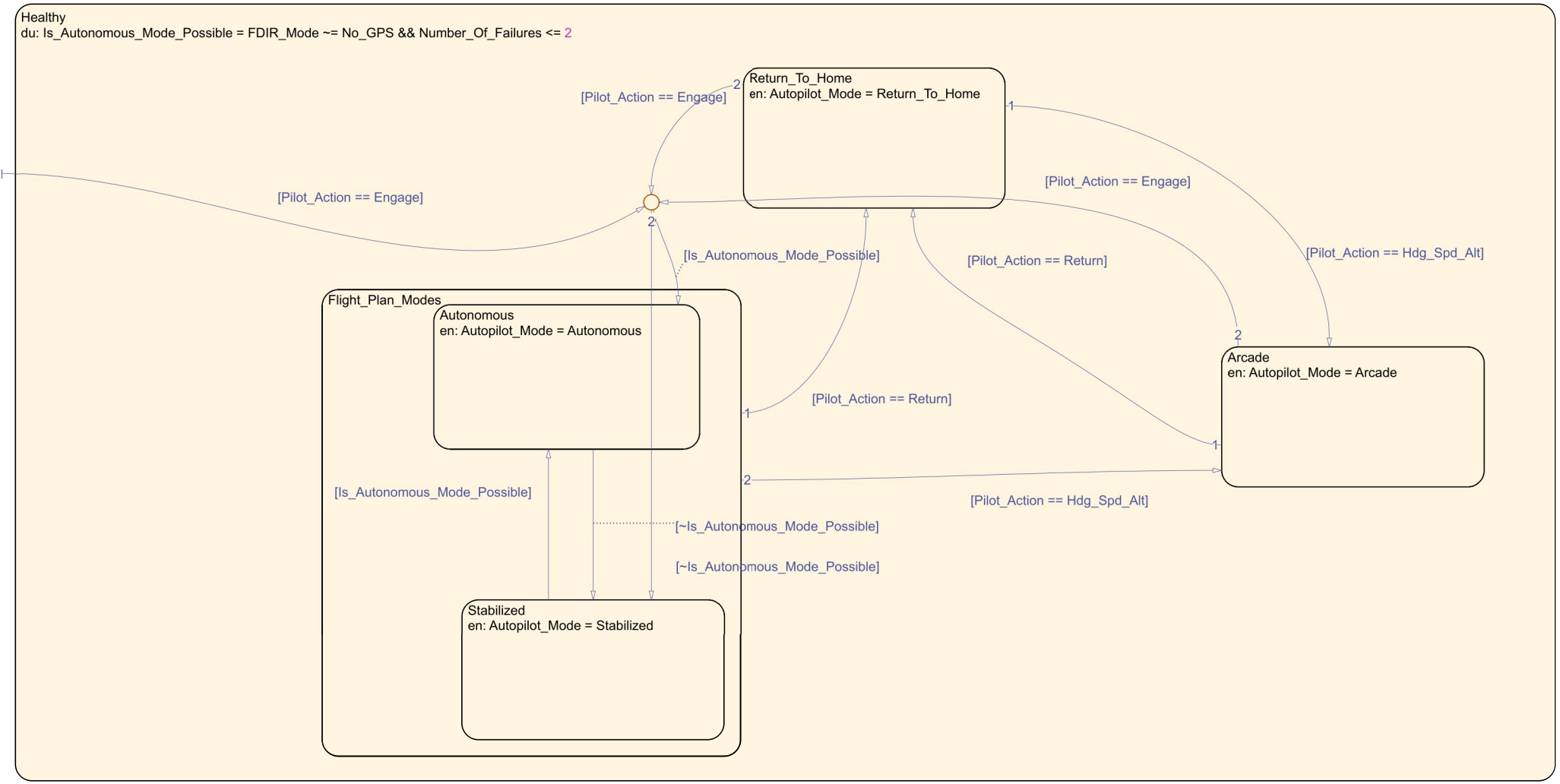


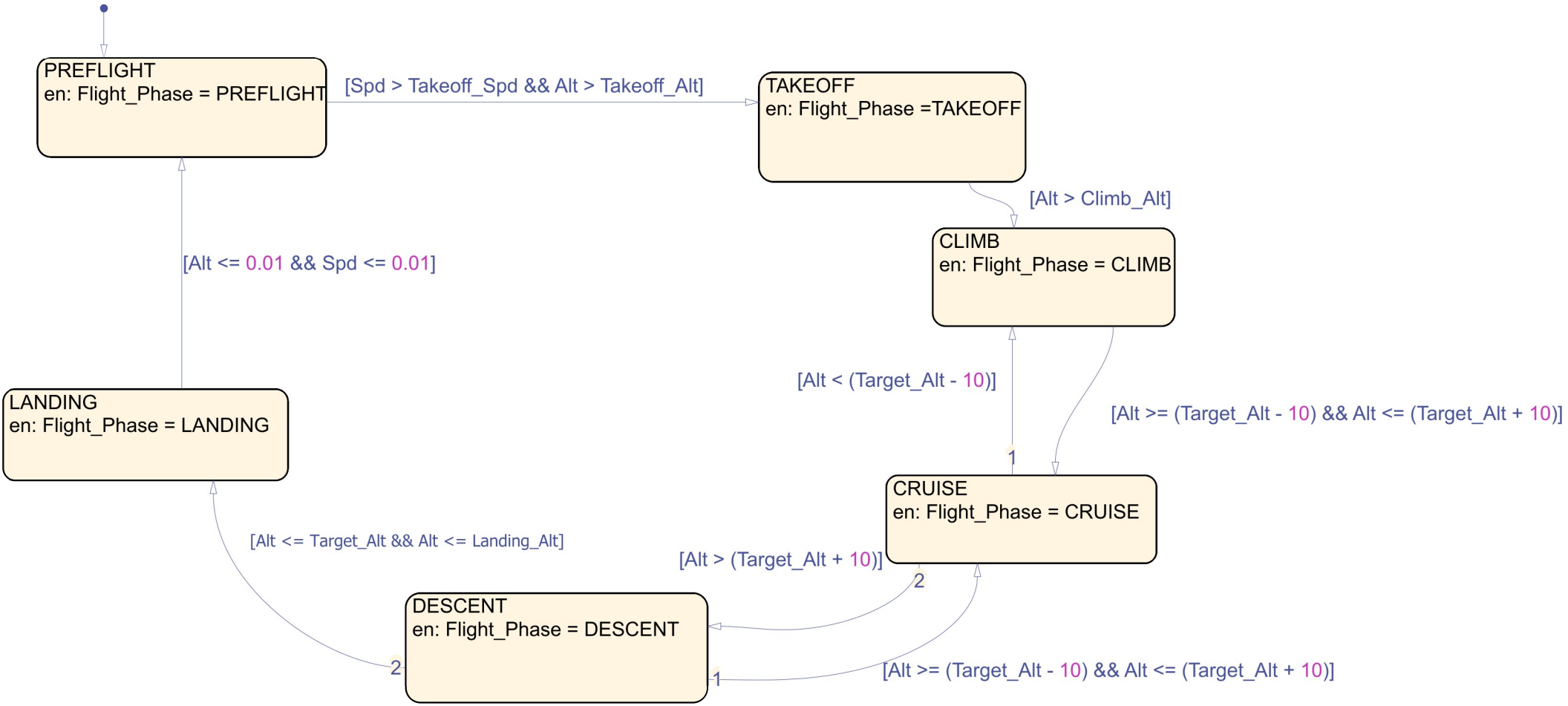

3
FDIR_Health
5
Number_Of_Failures
1
Pilot_Action
ZOH
ZOH
ZOH
FDIR_Mode
Number_Of_Failures
Pilot_Action
Aggressive_Maneuver
Autopilot_Mode
Autopilot_Mode_Manager
1
Autopilot_Mode
4
kin_state
ZOH
Spd
Alt
Target_Alt
Flight_Phase
Flight_Phase_Manager
2
Flight_Phase
2
Target_Alt
ZOH
Roll
Pitch
Yaw
Aggressive_Maneuver
Aggressive_Maneuver_Monitor


1
Roll
Roll_Rate_Max
Roll_Rate_Limit
2
Pitch
Pitch_Rate_Max
Pitch_Rate_Limit
3
Yaw
Yaw_Rate_Max
Yaw_Rate_Limit
Roll_Angle_Max
Roll_Angle_Limit
Pitch_Angle_Max
Pitch_Angle_Min
K (z-1)
Ts z
|u|
>
<
1
Aggressive_Maneuver



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
Return_To_Home
en: Autopilot_Mode = Return_To_Home
[Pilot_Action == Engage]
[Pilot_Action == Engage]
[Pilot_Action == Engage]
[Pilot_Action == Hdg_Spd_Alt]
[Pilot_Action == Return]
[Is_Autonomous_Mode_Possible]
Flight_Plan_Modes
Autonomous
en: Autopilot_Mode = Autonomous
Arcade
en: Autopilot_Mode = Arcade
[Pilot_Action == Return]
[Is_Autonomous_Mode_Possible]
[Pilot_Action == Hdg_Spd_Alt]
[~Is_Autonomous_Mode_Possible]
[~Is_Autonomous_Mode_Possible]
Stabilized
en: Autopilot_Mode = Stabilized


PREFLIGHT
en: Flight_Phase = PREFLIGHT
[Spd > Takeoff_Spd && Alt > Takeoff_Alt]
TAKEOFF
en: Flight_Phase =TAKEOFF
[Alt > Climb_Alt]
CLIMB
en: Flight_Phase = CLIMB
[Alt <= 0.01 && Spd <= 0.01]
[Alt < (Target_Alt - 10)]
[Alt >= (Target_Alt - 10) && Alt <= (Target_Alt + 10)]
LANDING
en: Flight_Phase = LANDING
1
CRUISE
en: Flight_Phase = CRUISE
[Alt <= Target_Alt && Alt <= Landing_Alt]
[Alt > (Target_Alt + 10)]
2
DESCENT
en: Flight_Phase = DESCENT
2
1
[Alt >= (Target_Alt - 10) && Alt <= (Target_Alt + 10)]