API description file retrieval outline

Catalogue

1. Introduction to the RflySim fault injection architecture

Any unmanned system can be divided into several components (or subsystems), including physical components such as motors, propellers, gyroscopes, and virtual components such as wind, air pressure, and obstacles. As shown in Figure 6-1, any component can be considered to contain three types of models: energy consumption model, motion model, and fault model. First, the motion model is responsible for describing the transient law of the component, and then the energy consumption model is used to describe the long state law of the component. The motion model and the energy consumption model together describe the normal short - and long-term operation law of the component, while the fault model describes the rule of the component deviating from the normal operation state due to various internal and external factors.

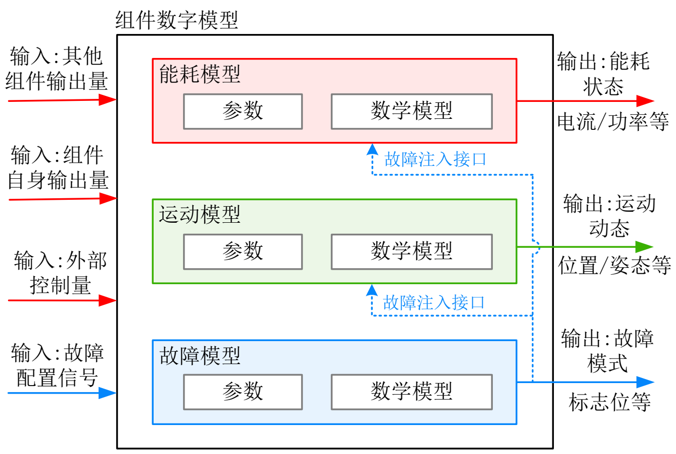


Figure 6-1 Unified Modeling framework for vehicle components

Each component may contain one or more of the three basic models, and a fault model is only necessary if you are considering fault testing. For example, batteries are normally described primarily by energy consumption models, while motors are described primarily by motion models. But when you need to consider fault injection, you need to embed the fault model into the component model in a suitable way. The battery model shown in Figure 6-2 is used as an example. The battery model mainly consists of an energy consumption model and a fault model. The energy consumption model mainly corresponds to the voltage drop curve of the battery, while the fault model mainly corresponds to the capacity loss curve. The capacity loss of the battery is mainly due to the aging of the electrolyte and the original caused by the increase of the number of battery charging and discharging, and the attenuation of the electricity brought by the battery is ultimately reflected in the voltage drop speed of the battery, and the decline of the voltage will eventually affect the overall movement law of the vehicle.

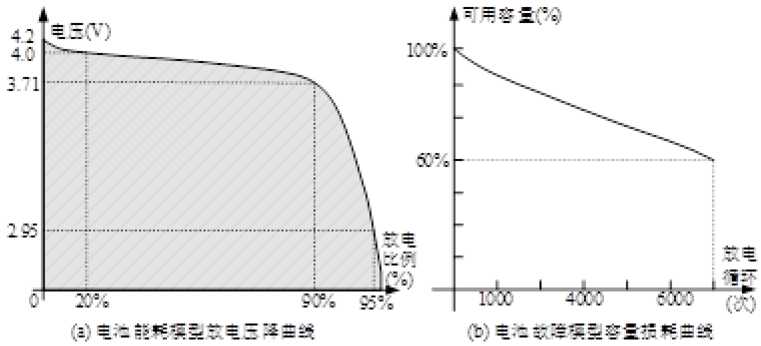


Figure 6-2 Battery model curve

In the unified modeling framework shown in Figure 6-1, the input of the fault model is mainly fault configuration signals (including whether the fault is enabled, triggered, and fault parameters). Of course, it also introduces information such as its own component state quantity, external component state quantity, and external control quantity as required. For example, the fault of the sensor high temperature failure requires the input of the fault configuration signal (whether to enable the fault and trigger temperature, etc.) and the external component state quantity (mainly the output of the temperature model component). The output of the fault model is mainly some indicators, such as whether the fault has been triggered, trigger time stamp, fault parameters, etc., which are used to feed back to the automatic test program. The fault model itself contains two parts: mathematical model and parameters. The fault model parameters are mainly to increase the extensibility of the model. The fault module can be applied to other systems by modifying the parameters in the future. The mathematical model describes the trigger mechanism of the failure and the influence law on other models. In addition, since each component may have a variety of different faults (high temperature failure, electromagnetic interference, etc.), then when the fault trigger signal is introduced, it needs to have a numbering function, and the fault model needs to first identify which kind of fault, and then generate the corresponding fault signal, which is injected into the energy consumption model or the motion model.

The fault model mainly identifies the fault source and trigger time, and then transmits the fault signal to the motion model or the energy consumption model, which can finally be reflected in the operation effect of the whole machine. The influence of fault model on motion model and energy consumption model is mainly reflected in three aspects: (1) parameter influence. Changing the parameters of the original motion model and energy consumption model, for example, when the sensor is faulty, the data noise may become larger, that is, the noise variance of the sensor output increases. (2) Modal influence. The mathematical expression of the original motion model and energy consumption model is directly changed, which can be considered as sending mode switching. For example, the fault of the sensor failure directly makes the sensor become an all-0 output mode; (3) Superposition effect. Interference is directly generated and superimposed on the semaphore of the motion model and the energy consumption model (which may be an input, intermediate state or output signal), such as external vibration failure, which may cause other noise to be superimposed on the sensor output.

In the simulation system, the fault trigger mechanism can also be divided into deterministic trigger and uncertain trigger (probabilistic trigger). Deterministic triggering refers to a clear known time or state in which the fault is triggered, which can be directly described by various logical judgment functions. Probabilistic (uncertain) triggering refers to triggering failure at a certain time or state with a certain probability, which usually needs to be described by combining probabilistic models of random processes. Uncertain fault triggering is usually very effective in the final safety test stage of the whole machine. It is necessary to set the failure probability of each component (in the simulation, the failure probability can be uniformly and reasonably increased to ensure that a failure can occur in as short a time as possible), and then conduct a large number of task simulation to test when a single or multiple failure bursts. Possible impact on the overall mission.

2. Development environment configuration

2.1 Windows development environment

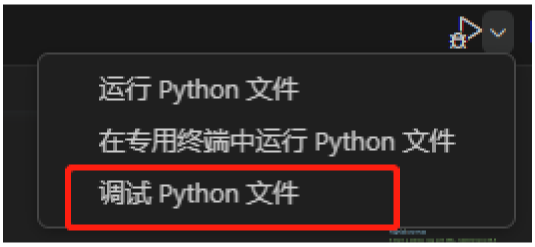
2.1.1 VS Code development tools

Open Visual Studio Code, select Open Folder and open the folder RflySimAPIs3.0\6.RflySimPHM\1.BasicExps\e4_FaultInjectAPITest_py.

The fault injection code in FaultInjectAPITest.py follows the FaultInjectAPITest_py in RflySimAPIs3.0\6.RflySimPHM\1.BasicExps\e4_FaultInjectAPITest_py The fault injection code is changed to propeller module fault (the propeller module fault injection code can be seen in the reference), and the fault parameters are modified.

```
silInt=np.zeros(8).astype(int).tolist()
silFloat=np.zeros(20).astype(float).tolist()
silInt[0:2]=[123450,123450]
silFloat[0:4]=[0,0,0,0]
# silInt[0:1]=[123540]
# silFloat[0:2]=[15,20]
mav1.sendSILIntFloat(silInt,silFloat)
print('Inject a fault, and start loging')
flag=2
```

Debug FaultInjectAPITest.py and watch the drone take off and malfunction in RflySim3D.



2.1.2 Matlab development tool

MATLAB installation package download path:https://ww2.mathworks.cn/products/matlab.html

3. Develop preparatory knowledge profiles

3.1 The Matlab-Simulink code generates the Visual studio compilation environment configuration

It is recommended to install Visual Studio 2017, online installation steps (Internet required) are as follows:

Double click "RflySimAPIs\SimulinkControlAPI\VS2017Installer\vs_community2017.exe"

For this course, just check "Desktop Development in C++" on the right.

Note: VS2019 can also be installed for advanced MATLAB versions, but MATLAB can only recognize Visual Studio versions below its own, so MATLAB 2017b cannot recognize VS2019.

Note: Please do not change the VS default installation directory (for example, to disk D), which will cause MATLAB to not recognize. You can't use the Mingw compiler, you need VS.

MATLAB compiler installation confirmation:

Enter the instruction "mex-setup" in the command line window of MATLAB.

In general, the VS 2017 compiler is automatically identified and installed, as shown in the picture on the right, "MEX configuration uses' Microsoft Visual C++ 2017 'for compilation" indicates that the installation is correct.

If there are other compilers, this page can also switch to select other compilers such as VS

2013/2015.

3.2 Matlab-Simulink code generation procedure

3.2.1 Model compilation parameter Settings

Common solver categories:

（1）Fixed step size and variable step size solvers

The simulation step size of the step solver is customized, and there is no error control mechanism. The variable step size solver needs to calculate the simulation step size in the simulation process, and satisfies the error tolerance by increasing/decreasing the step size. When generating the real-time operation code, you must use a step-size solver. If you do not intend to configure the model code generation, the choice of solver depends on building the model. Generally, the variable step size solver can reduce the simulation time, and the smaller the step size of the fixed step size solution, the higher the simulation accuracy. Therefore, under the same simulation accuracy requirements, when the fixed step size solver is used for simulation, the minimum step size of the variable step size solver must be used in the entire simulation process.

（2）Continuous and discrete solvers

There are continuous and discrete solvers in constant step size and variable step size solvers. Both continuous and discrete solvers rely on modules to compute all discrete state values. The module that defines the discrete state is responsible for calculating the discrete state value at each step time point, and the continuous solver calculates the state value of the module that defines the continuous state by numerical integration. When choosing a solver, you must first determine whether a discrete solver is needed in the model. If there is no continuous state module in the model, the solver can be continuous or discrete; if there is a continuous state model, the continuous solver must be used.

（3）Explicit and implicit solvers

The application of implicit solver is mainly to solve the rigid problem in the model, and the application of explicit solver is to solve the non-rigid problem. For example, in a control system, the control component is responsive and fast, with a small time constant, while the controlled object is generally inertial and slow, with a large time constant. Systems with very different time scales are often referred to as rigid systems, which, in popular terms, are systems that contain time-varying fast and slow-varying solution components (systems that contain both small time constants and large time constants). The rigid system has a very large recovery ability, so that the disturbance of the fast variable component is quickly attenuated. When numerically integrating such a system, it is expected to select an appropriate time step to calculate the slow variable component once the fast variable component disappears. Therefore, the essence of the rigid system is that the solution to be calculated is slow changing, but there is a rapidly decaying disturbance, which complicates the numerical calculation of the slow changing solution. Therefore, for the oscillation phenomenon in the system, the implicit solution is far more stable than the explicit solution, but the calculation cost is larger than the explicit solution. It needs to calculate the Jacobian matrix and algebraic equations generated by the Newton-like method at each step of the simulation. In order to reduce computational costs, Simulink provides parameters for calculating Jacobian methods to improve simulation performance.

（4）Single-step and multi-step solvers

Single-step and multi-step solvers are provided in the Simulink solution library. Single-step solution is to calculate the current time y(tn) of the system, and need to use the previous time y(TN-1) and the micro components of multiple time points between TN-1 and tn (these time points are called microsteps). The multistep solver uses the values of the previous times of the system to calculate the value of the current time. Simulink provides an explicit multistep solver, ode113, and an implicit multistep solver, ode15s, both of which are variable-step solvers.

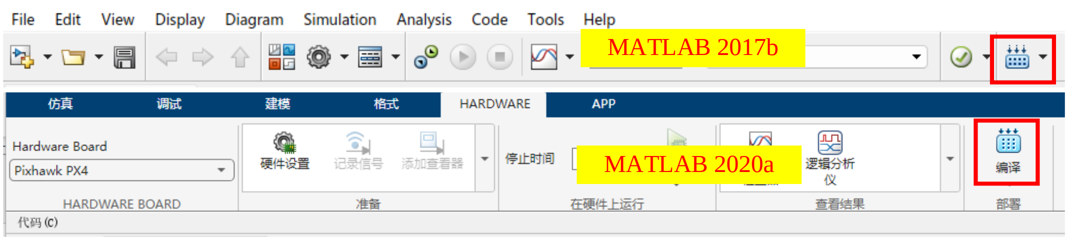
（5）Variable level solver

Simulink offers two kinds of variable level solvers. The ode15s solver uses 1 to 5 level simulation. ode113 applies orders 1 to 13. The highest level can be set for ode15s.

3.2.2 Generation of dynamic library files (use of GenerateModelDLLFile.p)

（1）First, use the initialization file in the routine and click Run.

(2) Open the file you want to use and compile it.



Upon completion of compilation, the.slxc Simulink component file, a compressed package file, and a series of other compiled files are generated.

(3) After that, we can click to run the GenerateModelDLLFile.p file, thus generating a.dll dynamic library file.

3.3 Matlab-Simulink common modeling modules (UDP, Goto -From tags)

3.3.1 UDP communication model used in Simulink

1. Create a UDP sender and receiver biock:

In the Simulink library, two blocks "UDP Send" and "UDP Receive" can be found. Add these blocks to the model and connect them for communication.

2. Configure UDP parameters:

In UDP Send and UDP Receive Block, you need to configure corresponding UDP parameters, such as the address, port number, and packet size. You can also choose to use a custom UDP header or payload.

3. Generate test data:

In a UDP Send Block, you can use the "Generate Test Data" feature to generate test data. This data will be sent to UDP Receive Block.

4. Verification results:

In a UDP Receive Block, the "Result" PORT can be used to verify the received data. You can also visualize Received data using the "Plot" Portail.

3.3.2 Goto-From tag model used in Simulink

The Goto-From tag is a pair of tags that specify a transition between two states in a state machine. The first tag, "Goto," indicates the target state the system should transition to, while the second tag, "From," specifies the current state from which it should transition. Together, they define a directed edge between two states, allowing the system to change its state based on certain conditions.

To use Goto-From tags in Simulink, perform the following steps:

1. Create a new State Machine: First create a new Simulink model and then select the "State Machine" module from the library. This creates a basic state machine with two states, "initial" and "final."

2. Add state: Click the state machine block, and then press the "Add State" button to add a new state to the computer. You can also delete or rename existing states as needed.

3. Add Transition: To add a transition between two states, click the Transition TAB in the state machine block, and then click the New Transition button. This creates a new transition arrow connecting the two states.

4. Assign the Goto-From label: Select the transition arrow, then click the Labels TAB in the Properties inspector. Here, you can assign a "Goto" tag to indicate the target state and a "From" tag

to indicate the current state. For example, if you want the system to transition From state A to state B, you should set the "Goto" tag to "B" and the "from" tag to "A".

5. Set conditions: You can also set the conditions of the conversion. Click the transition arrow, and then select the Conditions TAB in the Properties inspector. Here, you can specify a logical expression that must be true to perform the conversion.

6. Run the simulation: After defining the state machine and transformation, you can run the simulation by clicking the "Run" button in the Simulink toolbar. The system starts in its initial state and transitions between states according to specified conditions.

3.4 bat One-click start script modification and use (.bat file)

3.4.1 The PX4PSP boot path is modified

In the routine, we often use some software to start the script with one click. However, when the platform is installed, due to the different location of the installation, it may be found that the script cannot run, which is why we need to modify the path of PXP startup in the script.

We can right-click the script, select "Show more options", and click "Edit" to modify the script. We can see the following lines of code at the beginning:

REM Set the path of the RflySim tools

SET PSP_PATH=C:\PX4PSP

SET PSP_PATH_LINUX=/mnt/c/PX4PSP

C:

They are related to the path where the RflySim tool is set. C indicates that the PX4PSP is installed on the C disk, which can be modified according to the location of your platform installation.

3.4.2 Dynamic library load path modification

Dynamic library load path modification is also based on the bat script, after specifying the path of PX4PSP, in the subsequent bat script, we can see the following code:

REM Set use DLL model name or not, use number index or name string

REM This option is useful for simulation with other types of vehicles instead of multicopters QuadModel FaultModel QuadModelv

set DLLModel=MulticopterModel

REM Check if DLLModel is a name string, if yes, copy the DLL file to CopterSim folder

SET /A DLLModelVal=DLLModel

if %DLLModelVal% NEQ %DLLModel% (

REM Copy the latest dll file to CopterSim folder

Copy /Y

```
"%~dp0"\%DLLModel%.dll %PSP_PATH%\CopterSim\external\model\%DLLModel%.dll
)
```

The last line of code indicates that the dll file is copied from the current directory to the file in the specified C disk PX4PSP. The fourth line of code is the name of the dll file that needs to be copied, if the name of the dll file that we generate is inconsistent with the script, it cannot be copied, which requires us to modify it.

4. Construction and use of simulink faulty module encapsulation library (MulticopterModelLib.slx)

4.1 MotorFault module

```
% Define the 32-D ModelInParams vector for external modification
FaultParamAPI.FaultInParams = zeros(32,1);

MotorFaultTemp.FaultID=123450;
MotorFaultTemp.NoiseFaultID=111111;
MotorFaultTemp.MotorNum=int32(4);
```

4.1.1 Motor fault injection ID and parameter configuration

MotorFaultTemp.FaultID=123450;

字段	值
FaultID	123450
MotorNum	4

MotorFaultTemp.NoiseFaultID=111111;

字段	值
FaultID	123450
NoiseFaultID	111111
MotorNum	4

MotorFaultTemp.MotorNum=int32(4);

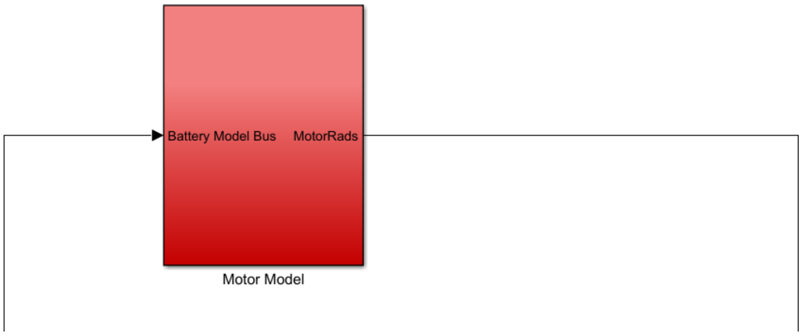
The number of motors here is four.

4.1.2 Encapsulation of fault parameters (FaultParamAPI) in the fault module

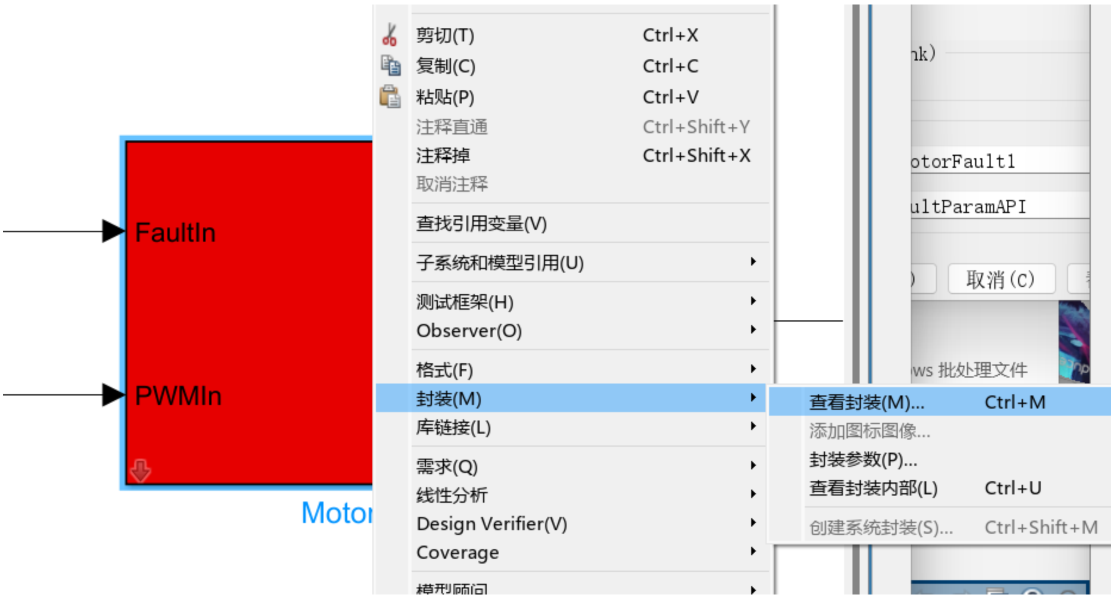
MotorFault	1x1 struct
MotorFault1	1x1 struct
MotorFaultTemp	1x1 struct
PropFault	1x1 struct
SensorFault	1x1 struct
WindFault	1x1 struct

We can view the encapsulation module reference parameters through the workspace.

Double-click to open the initialization script and run it on a single machine. By opening the routine file, we can view the encapsulated module.



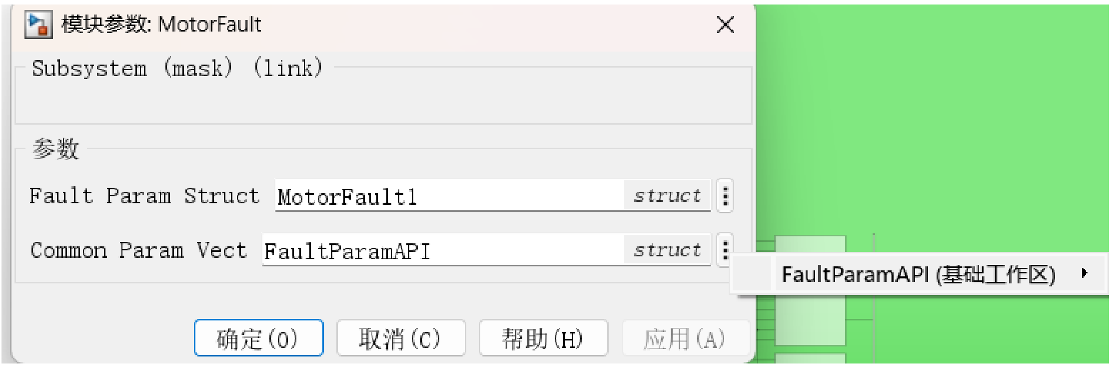
The parameters in this module are imported from the workspace. We can right click and view the package.

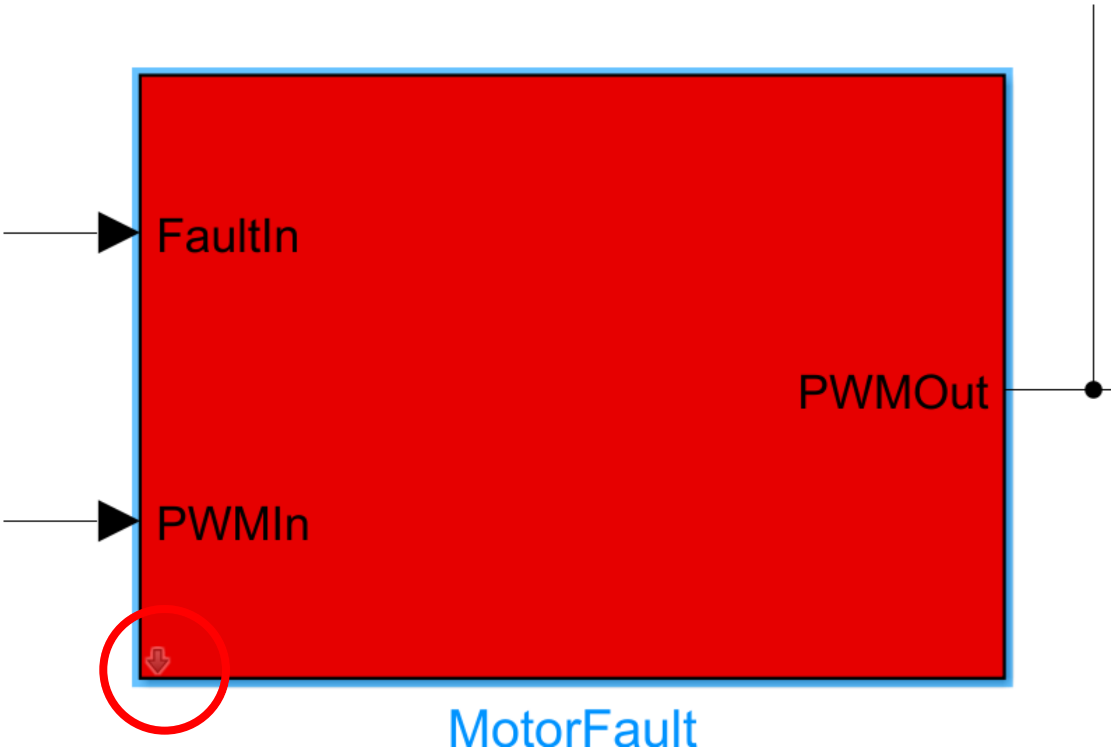


As you can see, we apply the same values as above, importing the values from the workspace, but naming them.

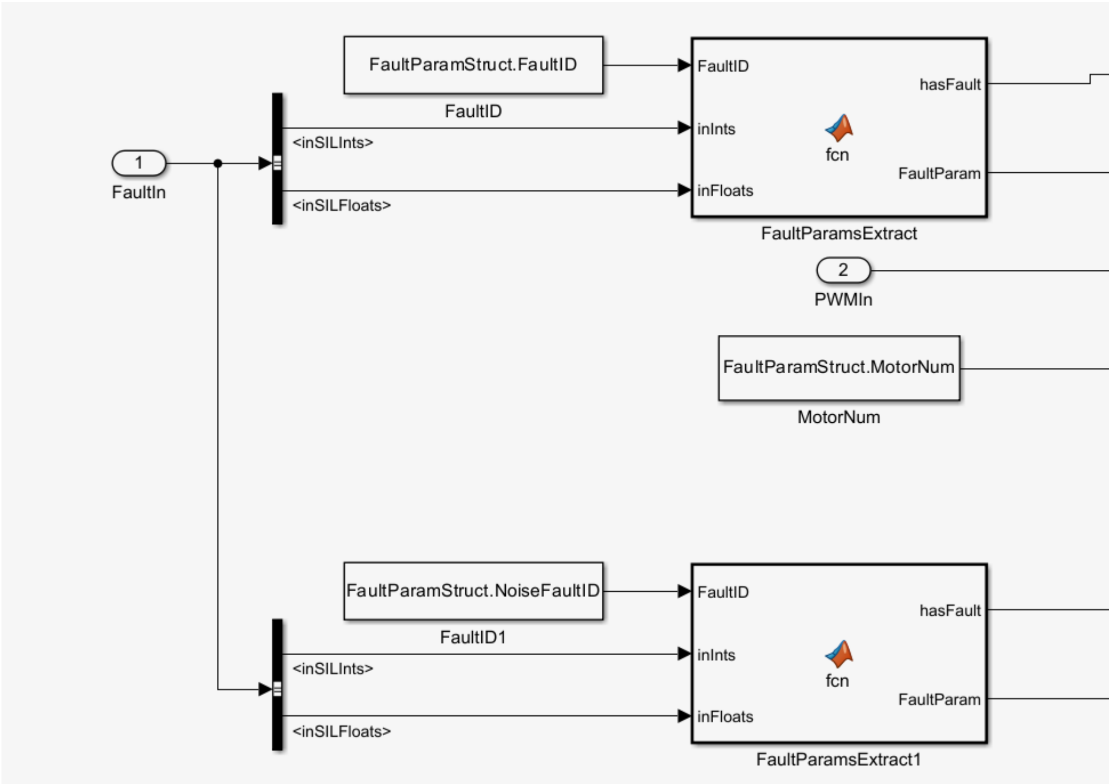
After that, we go inside the package and observe.

Double-click to view module parameters.





Click the arrow in the red circle to enter the encapsulation.



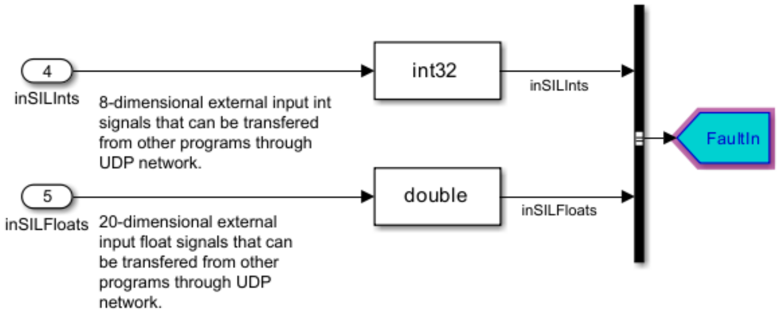
We can see the two motor modules, where the parameters used, we can see from the working area, the front fault ID module naming method is 32 bit fault parameter name plus fault ID composition. Double-click on the module and we can see the code logic.

```
function [hasFault, FaultParam] = fcn(FaultID,inInts,inFloats)
persistent hFault;
persistent fParam;

if isempty(hFault)
    hFault=false;
end
if isempty(fParam)
    fParam=zeros(20,1);
end
```

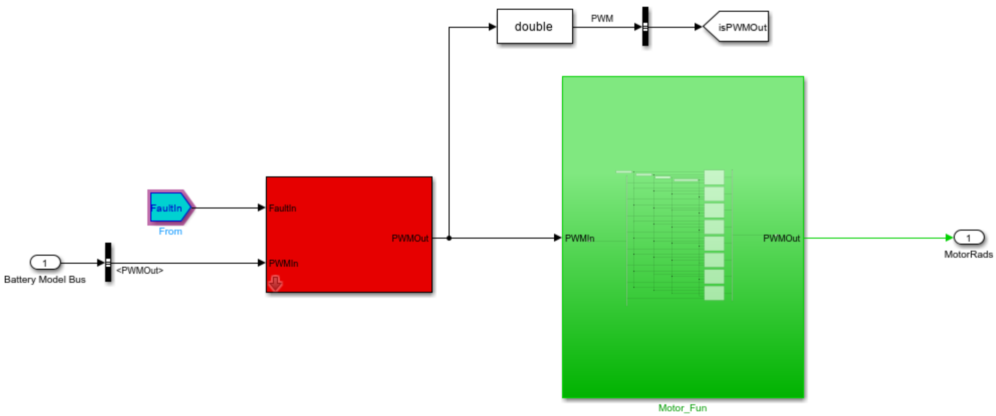
4.1.3 Subscription and triggering of fault messages

The Goto module FaultIn tag - publishes fault messages

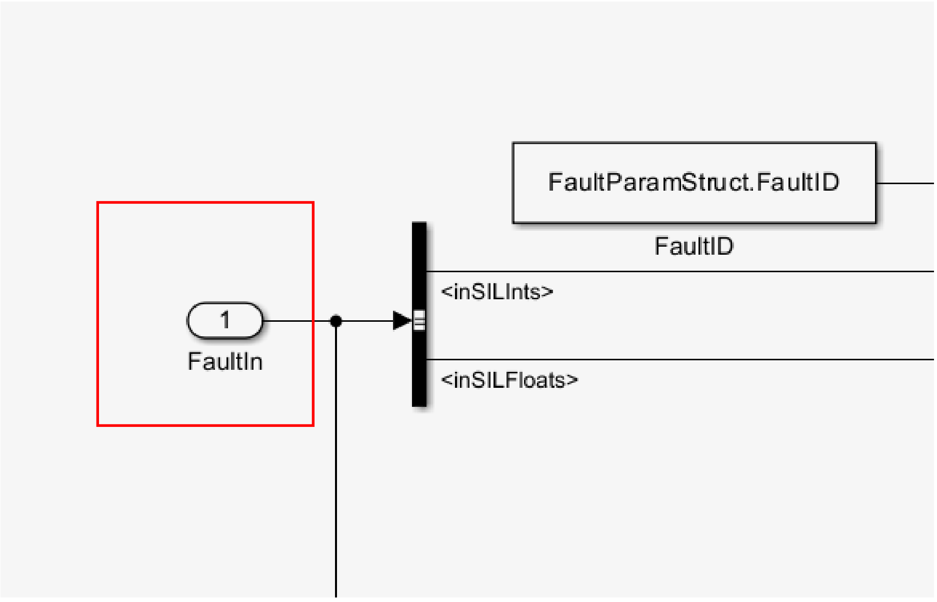
From module FaultIn flag - Subscribe to fault messages



The set parameters are imported through the initialization file, and then the data is transmitted through the Goto-from module.



Data imported from the workspace by the upper layer is transmitted here and injected into the fault module.



The fault is also injected FROM module. The fault is injected layer by layer through modules to achieve the effect of fault injection.

FaultParamsExtract 自定义模块-故障触发与处理

```
function [hasFault, FaultParam] = fcn(FaultID,inInts,inFloats)
persistent hFault;
persistent fParam;

if isempty(hFault)
    hFault=false;
end
if isempty(fParam)
    fParam=zeros(20,1);
end

hFaultTmp=false;
fParamTmp=zeros(20,1);
j=1;
for i=1:8
    if inInts(i) == FaultID
        hFaultTmp=true;
        fParamTmp(2*j-1)=inFloats(2*i-1);
        fParamTmp(2*j)=inFloats(2*i);
        j=j+1;
    end
end
if hFaultTmp
    hFault=hFaultTmp;
    fParamTmp(17:20) = inFloats(17:20);
    fParam=fParamTmp;
end

hasFault=hFault;
FaultParam=fParam;

```

Here the code shows the process of troubleshooting and triggering.

4.2 PropFault module

```
%Prop Fault Stuct
PropFault.FaultID = 123451;
PropFault.PropNum = int32(4);
```

4.2.1 Propeller fault injection ID and parameter configuration

PropFault.FaultID = 123451;

字段	值
FaultID	123451
PropNum	4

PropFault.PropNum = int32(4);

The number of propellers is 4.

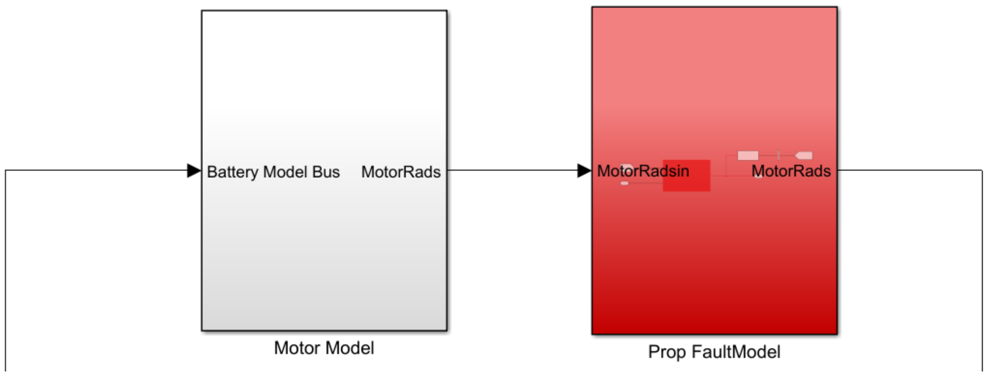
4.2.2 Encapsulation of fault parameters (FaultParamAPI) in the fault module

MotorFault	1x1 struct
MotorFault1	1x1 struct
MotorFaultTemp	1x1 struct
PropFault	1x1 struct
SensorFault	1x1 struct
WindFault	1x1 struct

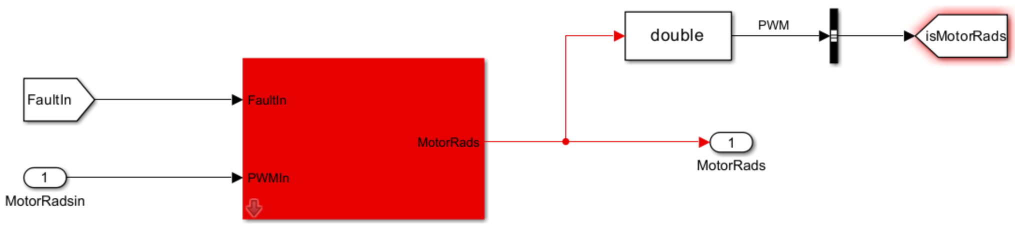
字段	值
FaultID	123451
PropNum	4

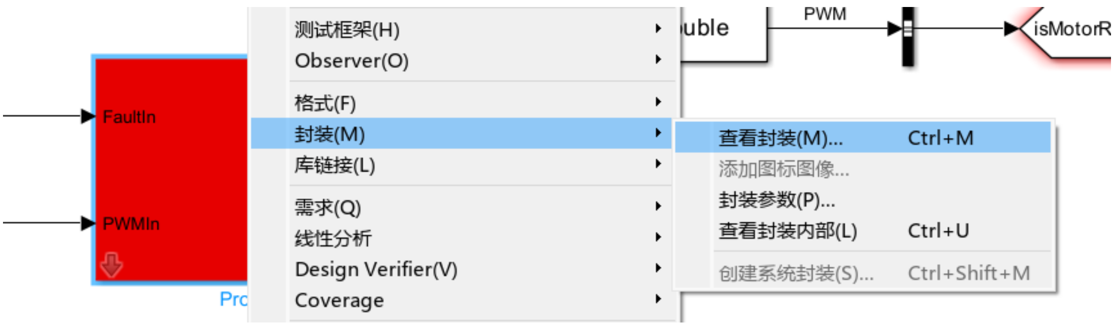
We can view the encapsulation module reference parameters through the workspace.

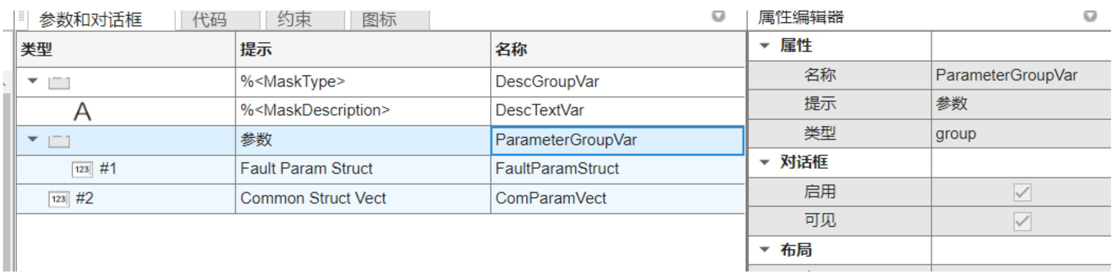
Double-click to open the initialization script and run it on a single machine. By opening the routine file, we can view the encapsulated module.



The parameters in this module are imported from the workspace. We can right click and view the package.



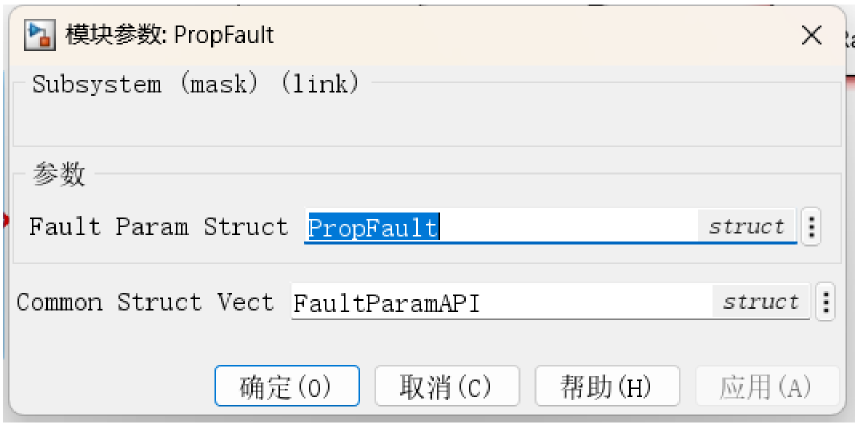


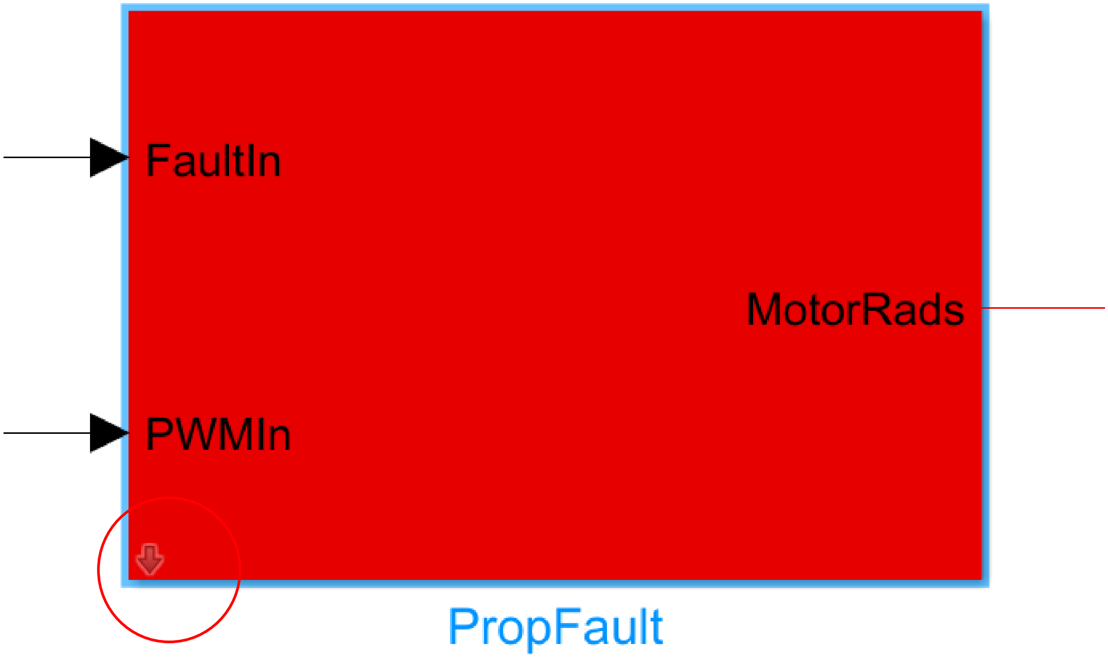


As you can see, we apply the same values as above, importing the values from the workspace, but naming them.

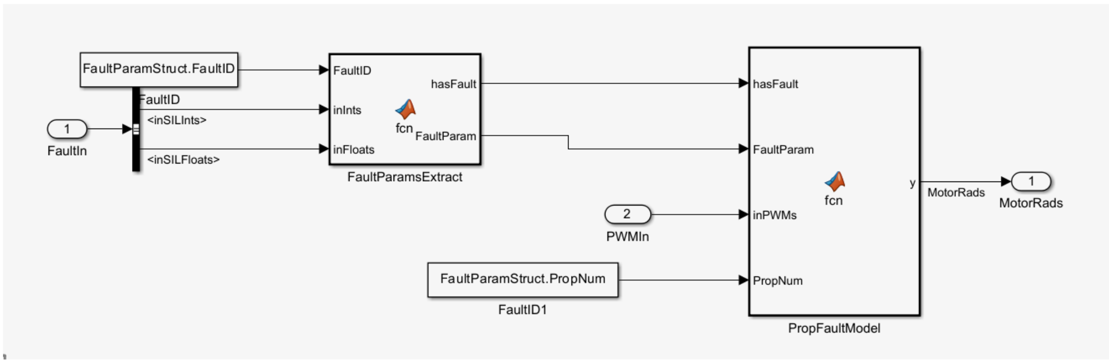
After that, we go inside the package and observe.

Double-click to view module parameters.





Click the arrow in the red circle to enter the encapsulation.

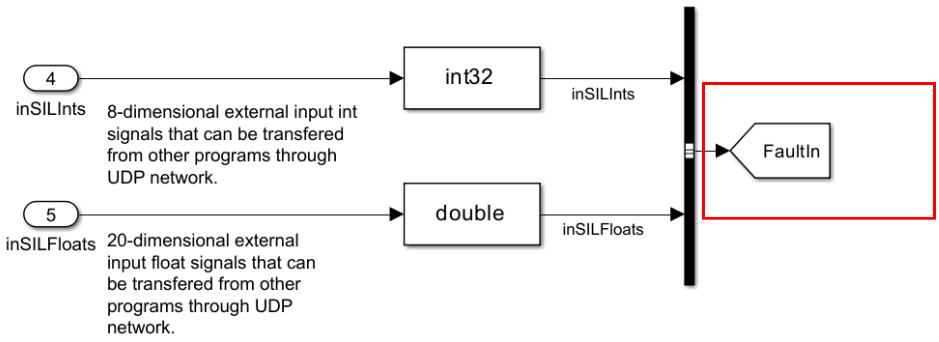


We can see the propeller fault injection module. Double-click FaultParamsExtract module, we can see its fault injection method.

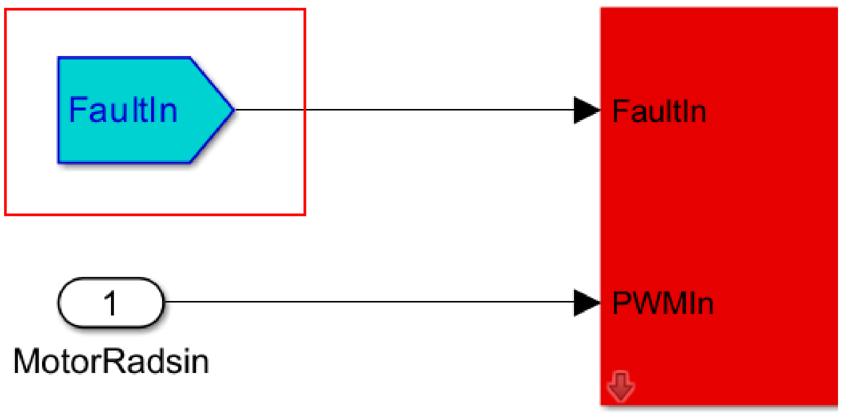
4.2.3 Subscription and triggering of fault messages

The Goto module FaultIn tag - publishes fault messages

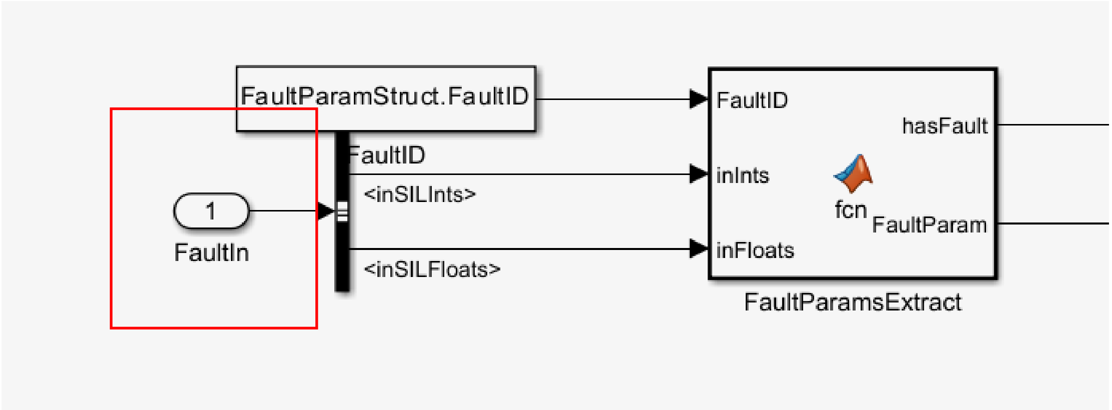
From module FaultIn flag - Subscribe to fault messages



Similarly, the routine file is transmitted to the propeller fault module by sending fault parameters through the GOTO module after reading the required parameters from the workspace.



The fault parameters are then transmitted to the encapsulation module in the same way.



FaultParamsExtract Custom module - fault triggering and handling

```matlab
function [hasFault, FaultParam] = fcn(FaultID,inInts,inFloats)
persistent hFault;
persistent fParam;

if isempty(hFault)
    hFault=false;
end
if isempty(fParam)
    fParam=zeros(20,1);
end

hFaultTmp=false;
fParamTmp=zeros(20,1);
j=1;
for i=1:8
    if inInts(i) == FaultID
        hFaultTmp=true;
        fParamTmp(2*j-1)=inFloats(2*i-1);
        fParamTmp(2*j)=inFloats(2*i);
        j=j+1;
    end
end
if hFaultTmp
    hFault=hFaultTmp;
    fParamTmp(17:20) = inFloats(17:20);
    fParam=fParamTmp;
end

hasFault=hFault;
FaultParam=fParam;

```

4.3 BatteryFault module

4.3.1 Set the battery fault injection ID and parameters

字段	值
PowOffFaultID	123452
LowVoltageFaultID	123453
LowCapacityFaultID	123454

BatteryFault.PowOffFaultID = 123452;

Battery failure The fault ID is 123452.

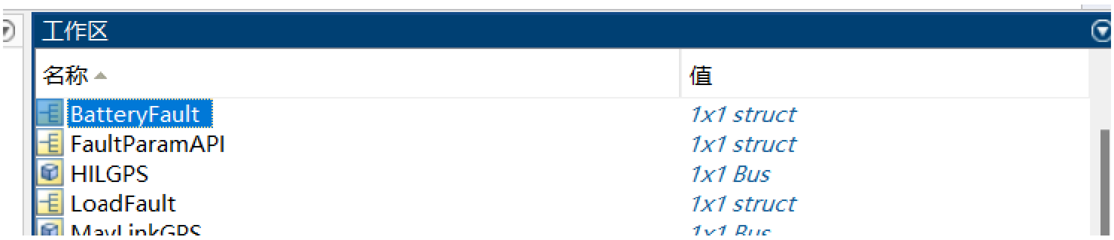
BatteryFault.LowVoltageFaultID = 123453;

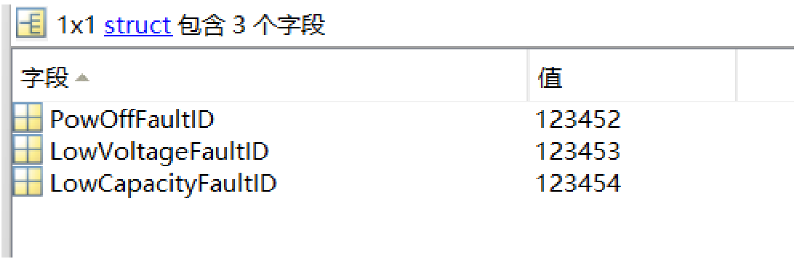
The undervoltage fault ID is 123453.

BatteryFault.LowCapacityFaultID = 123454;

The low-battery fault ID is 123454.

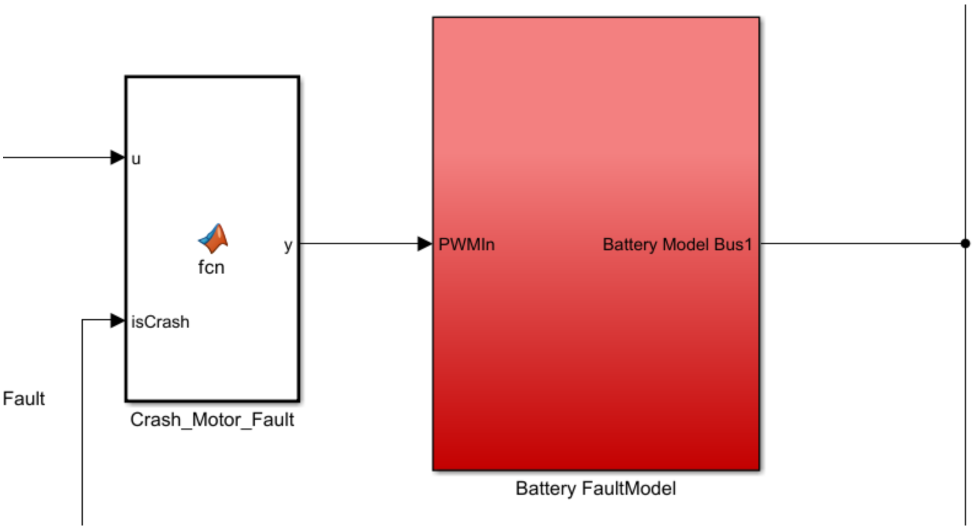
4.3.2 Encapsulation of fault parameters (FaultParamAPI) in the fault module



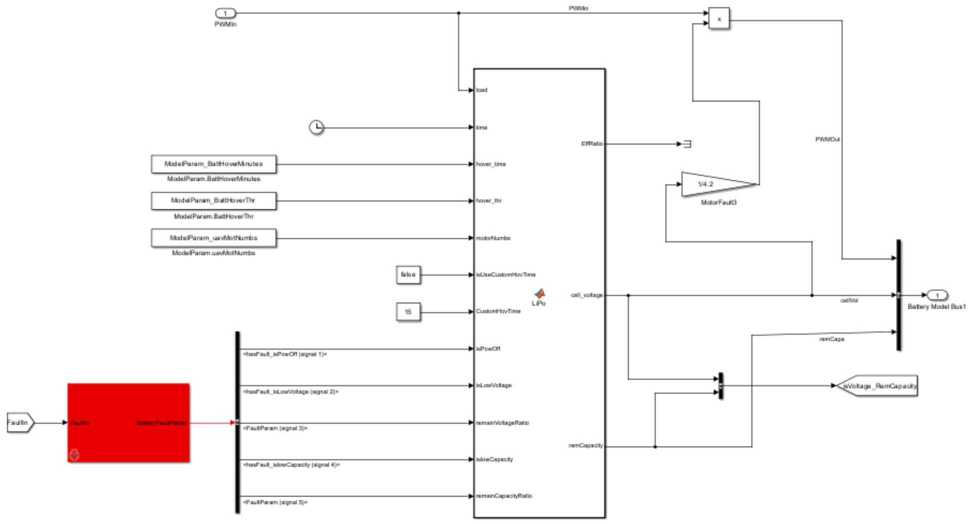


We can view the encapsulation module reference parameters through the workspace.

Double-click to open the initialization script and run it on a single machine. By opening the routine file, we can view the encapsulated module.



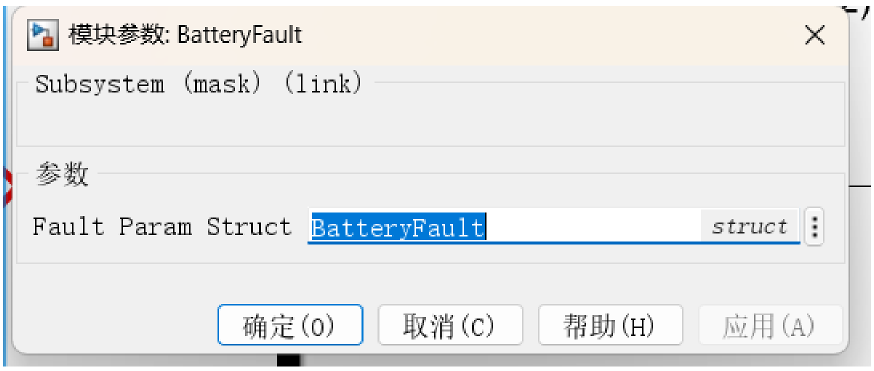
The parameters in this module are imported from the workspace. We can right click and view the package.

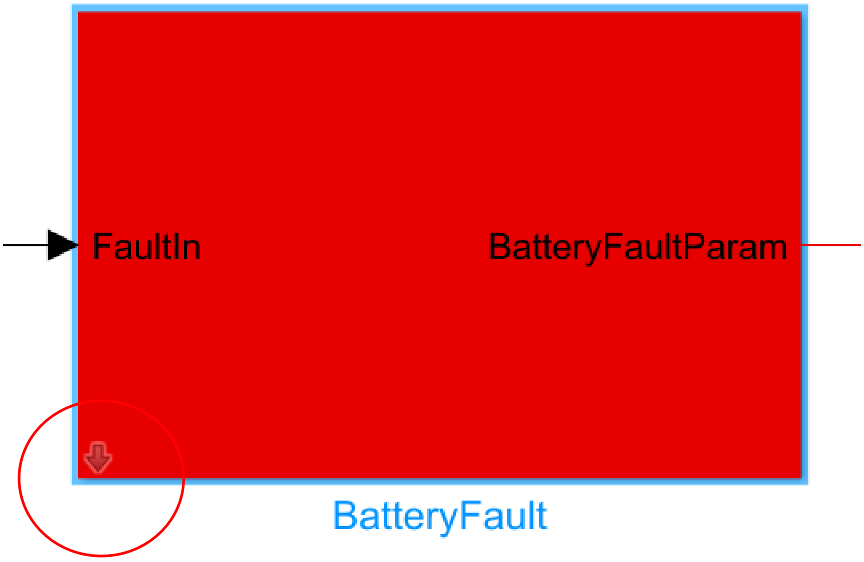


As you can see, we apply the same values as above, importing the values from the workspace, but naming them.

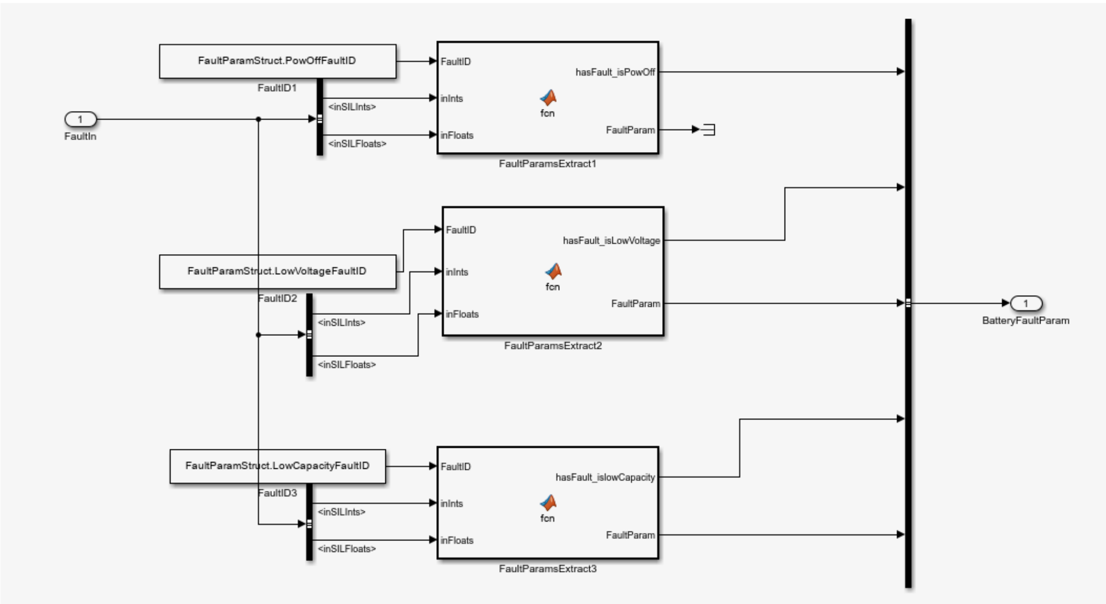
After that, we go inside the package and observe.

Double-click to view module parameters.





Click the arrow in the red circle to enter the encapsulation.

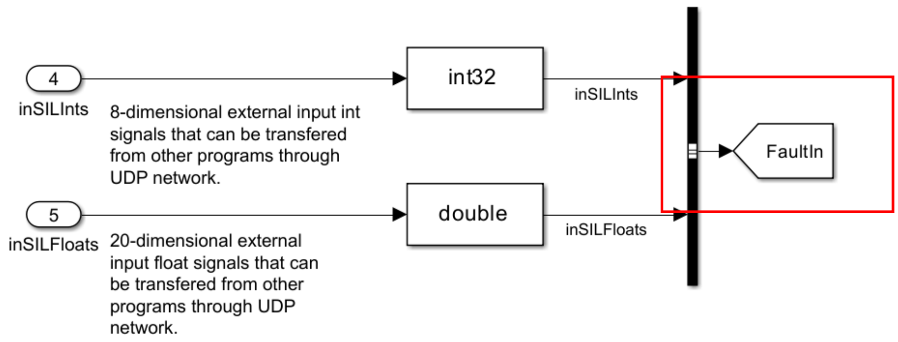


We can see three battery fault injection modules, this is because there are three kinds of faults can be injected, according to the injection fault ID is different, according to the internal code logic, the correct module will be selected for fault injection.

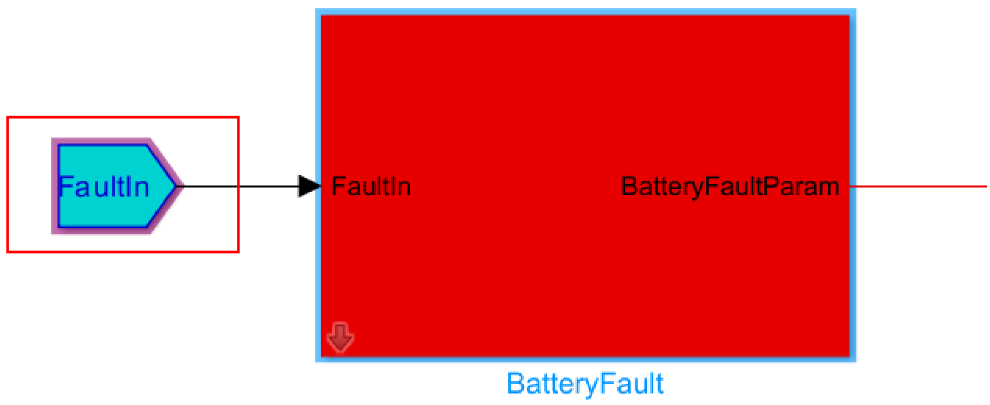
4.3.3 Subscription and triggering of fault messages

The Goto module FaultIn tag - publishes fault messages

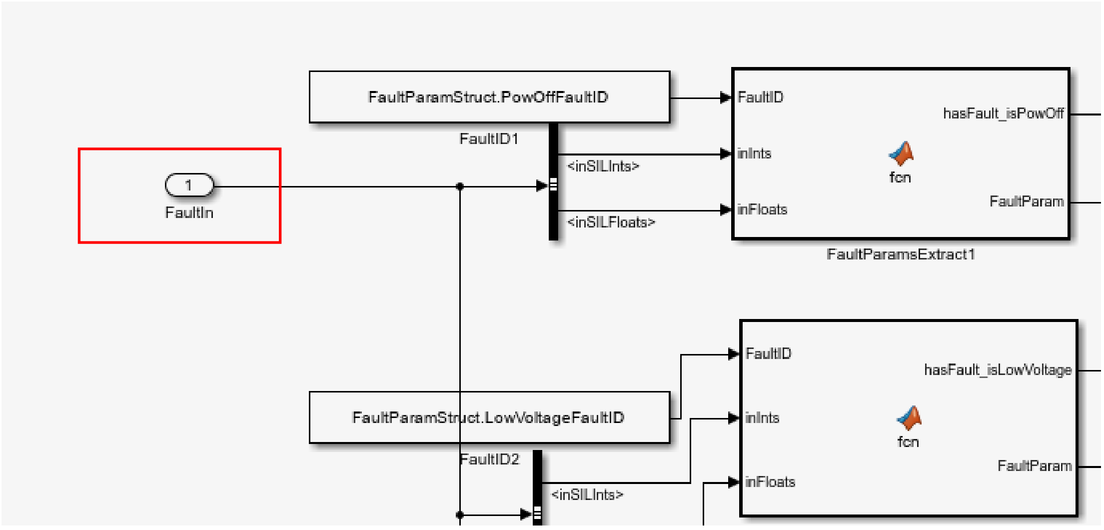
From module FaultIn flag - Subscribe to fault messages



Similarly, the routine file is transmitted to the battery fault module by sending the fault parameters through the GOTO module after reading the required parameters from the workspace.



The fault parameters are then transmitted to the encapsulation module in the same way.



FaultParamsExtract Custom module - fault triggering and handlin g

```
if isempty(hFault)
    hFault=false;
end
if isempty(fParam)
    fParam=zeros(20,1);
end

hFaultTmp=false;
fParamTmp=zeros(20,1);
j=1;
for i=1:8
    if inInts(i) == FaultID
        hFaultTmp=true;
        fParamTmp(2*j-1)=inFloats(2*i-1);
        fParamTmp(2*j)=inFloats(2*i);
        j=j+1;
    end
```

4.4 LoadFault module

4.4.1 Load fault injection ID and parameter Settings

字段	值
LoadFallFaultID	123455
LoadShiftFaultID	123456
LoadLeakFaultID	123457

LoadFault.LoadFallFaultID = 123455;

Load fault The fault ID is 123455.

LoadFault.LoadShiftFaultID = 123456;

Load drift fault The fault ID is 123456.

LoadFault.LoadLeakFaultID = 123457;

Load leakage fault The fault ID is 123457.

4.4.2 Encapsulation of fault parameters (FaultParamAPI) in the fault module

The encapsulation and transmission of fault parameters can be referred to above 4.1.2 故障模块中故障参数（FaultParamAPI）的封装传递

4.4.3 Subscription and triggering of fault messages

The Goto module FaultIn tag - publishes fault messages

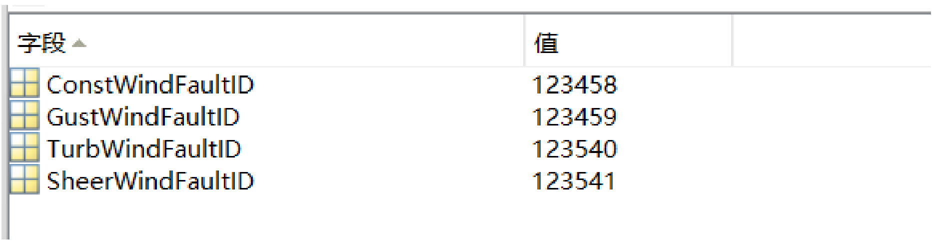
From module FaultIn flag - Subscribe to fault messages

FaultParamsExtract Custom module - fault triggering and handling

Refer to above here 4.1.3 故障消息的订阅与触发

4.5 The module is WindFault. Procedure

4.5.1 Environment Air fault injection ID and parameters



WindFault.ConstWindFaultID = 123458;

The ID of the fault is 123458.

WindFault.GustWindFaultID = 123459;

The ID of the gust fault is 123459.

WindFault.TurbWindFaultID = 123540;

Turbulent wind fault The fault ID is 123540.

WindFault.SheerWindFaultID = 123541;

Tangential wind fault The fault ID is 123541.

4.5.2 Encapsulation of fault parameters (FaultParamAPI) in the fault module

The encapsulation and transmission of fault parameters can be referred to above 4.1.2 故障模块中故障参数（FaultParamAPI）的封装传递

4.5.3 Subscription and triggering of fault messages

The Goto module FaultIn tag - publishes fault messages

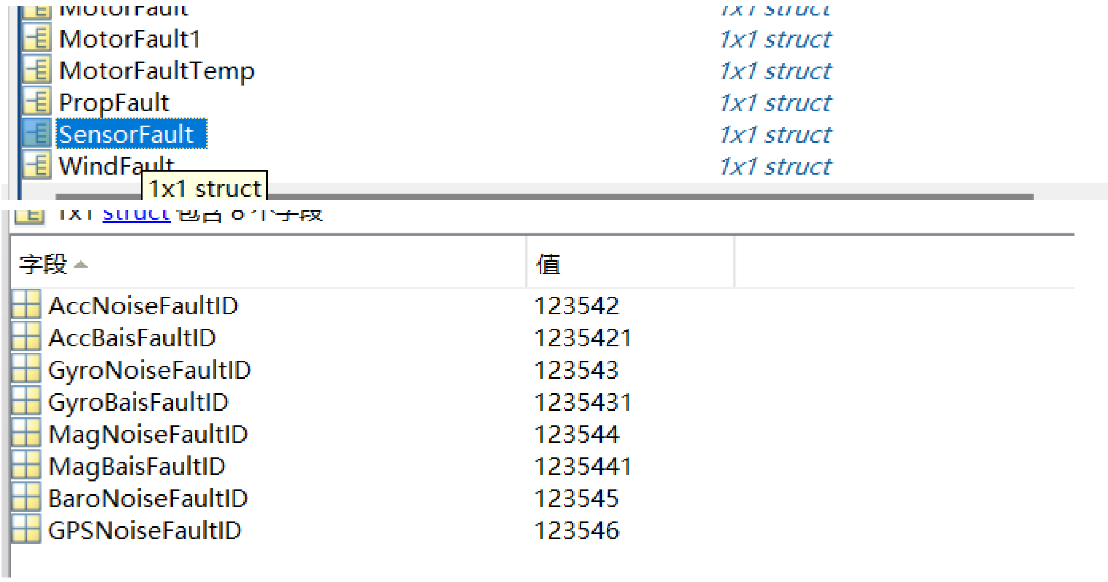
From module FaultIn flag - Subscribe to fault messages

FaultParamsExtract Custom module - fault triggering and handling

Refer to above here 4.1.3 故障消息的订阅与触发

4.6 The sensor is faulty (SensorFault)

4.6.1 Sensor fault injection ID and parameter Settings



The accelerometer SensorFault. AccNoiseFaultID = 123542;

The accelerometer noise interference fault ID is 123542.

gyroscope SensorFault.GyroNoiseFaultID = 123543;

The gyroscope noise interference fault ID is 123543.

magnetic compass SensorFault.MagNoiseFaultID = 123544;

The magnetometer noise interference fault ID is 123544.

barometer SensorFault.BaroNoiseFaultID = 123545;

The barometer noise interference fault ID is 123545.

GPS SensorFault.GPSNoiseFaultID = 123546;

GPS fault The fault ID is 123546.

4.6.2 Encapsulation of fault parameters (FaultParamAPI) in the fault module

The encapsulation and transmission of fault parameters can be referred to above 4.1.2 故障模块中故障参数（FaultParamAPI）的封装传递

4.6.3 Subscription and triggering of fault messages

The Goto module FaultIn tag - publishes fault messages

From module FaultIn flag - Subscribe to fault messages

FaultParamsExtract Custom module - fault triggering and handling

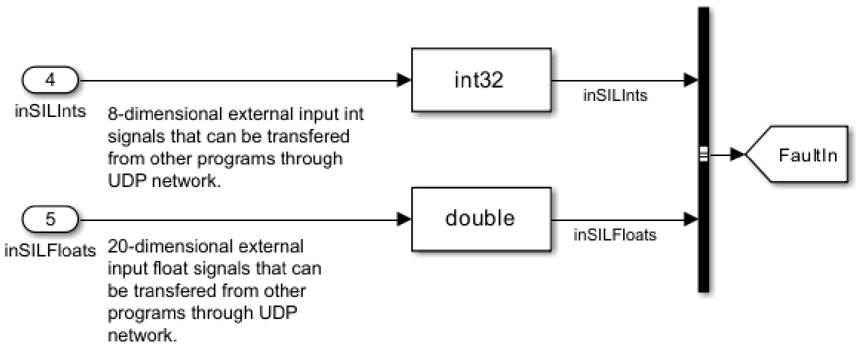
Refer to above here 4.1.3 故障消息的订阅与触发

5. simulink Model message interface

5.1 CopterSim Input/output interface

5.1.1 Message output interface

Output DLL model messages via udp module (30101) port and receive using a 32-dimensional array



5.1.2 Message input interface

Receive external input messages via udp module (30100) port

```python
# import required libraries
import time
import math
import numpy as np
# import RflySim APIs
import PX4MavCtrlV4 as PX4MavCtrl

# Create MAVLink control API instance
mav1 = PX4MavCtrl.PX4MavCtrler(1)
# mav2 = PX4MavCtrl.PX4MavCtrler(2)
# mav2 = PX4MavCtrl.PX4MavCtrler(3)
# mavN --> 20100 + (N-1)*2

# Init MAVLink data receiving loop
mav1.InitMavLoop()
#mav2.InitMavLoop(), ...

time.sleep(0.5)
mav1.InitTrueDataLoop()
time.sleep(0.5)

mav1.initOffboard()

lastTime = time.time()
startTime = time.time()
# time interval of the timer
timeInterval = 1/30.0 #here is 0.0333s (30Hz)
```

5.2 Rflysim interface protocol file Python-PX4MavCtrlV4.py

5.2.1 udp fault injection interface

aultInjectAPITest.py > ...

```python
# import required libraries
import time
import math
import numpy as np
# import RflySim APIs
import PX4MavCtrlV4 as PX4MavCtrl

# Create MAVLink control API instance
mav1 = PX4MavCtrl.PX4MavCtrler(1)
# mav2 = PX4MavCtrl.PX4MavCtrler(2)
# mav2 = PX4MavCtrl.PX4MavCtrler(3)
# mavN --> 20100 + (N-1)*2
```

```python
        self.port = 20100+self.CopterID*2-2

        # UDP模式解析
        if (Com=='udp' or Com=='UDP' or Com=='Udp') and ID>10000: # 如果是UDP通信模式
            # 兼容旧版协议，如果ID是20100等端口输入，则自动计算CopterID
            self.port=ID
            self.CopterID = int((ID-20100)/2)+1
```

5.2.2 Fault injection interface based on serial port

```python
import time
import math
import sys

import PX4MavCtrlV4 as PX4MavCtrl

# For hardware connection
#Windows use format PX4MavCtrler(ID,ip,'COM3',baud) for Pixhawk USB port connection
#Windows use format 'COM4' for Pixhawk serial port connection
#Linux use format '/dev/ttyUSB0' for USB, or '/dev/ttyAMA0' for Serial port (RaspberryPi
# PX4MavCtrler(1,'127.0.0.1','COM3',57600)
# PX4MavCtrler(1,'127.0.0.1','/dev/ttyS0',57600)
# constructor function
mav = PX4MavCtrl.PX4MavCtrler(Com = 'COM10:57600')

#mav.InitMavLoop(UDPMode), where UDPMode=0,1,2,3,4
```

```python
    # constructor function
    def __init__(self, ID=1, ip='127.0.0.1',Com='udp',port=0, simulinkDLL=False):
        global isEnableRdis
        self.isInPointMode = False
        self.isCom = False
        self.Com = Com
        self.baud = 115200
        self.isRealFly = 0
        self.ip = ip
        self.isRedis = False
        self.simulinkDLL = simulinkDLL

        # 这里是为了兼容之前的PX4MavCtrler('COM3:115200')串口协议，将来会取消
```

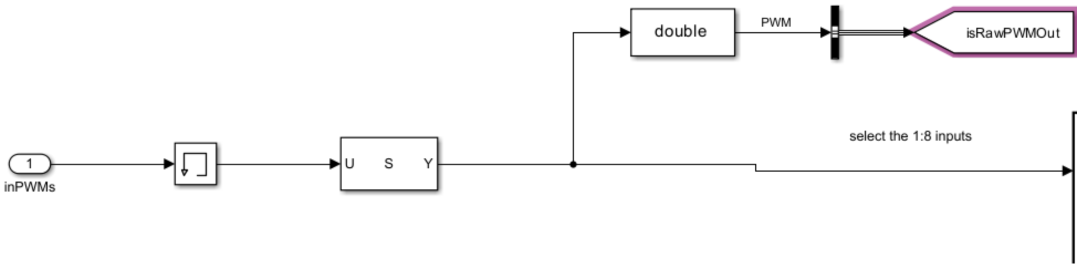
```python
        if Com[0:3]=='COM' or Com[0:3]=='com'  or Com[0:3]=='Com' or Com[0:3]=='/de': # 如果是串口连接方
            self.isCom = True # 串口通信模式
            strlist = Com.split(':')
            if port==0: # 默认值57600
                self.baud = 57600
            if(len(strlist) >= 2): # 串口号:波特率 协议解析，为了兼容旧接口
                if strlist[1].isdigit():
                    self.baud = int(strlist[1])
            self.ComName = strlist[0] # 串口名字
```

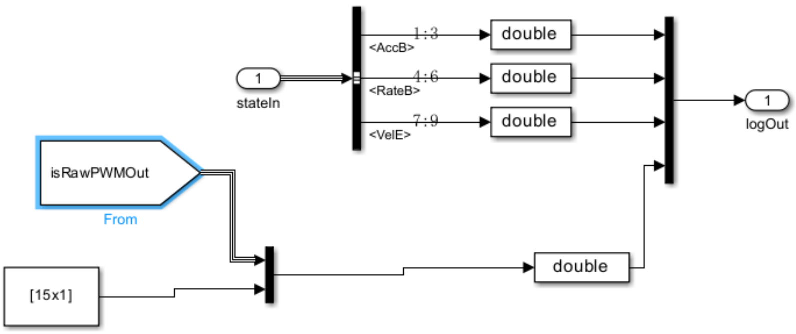
There are two types of baud rate used for serial port fault injection: 115200 and 57600. If the baud rate is set to 0, 57600 is the default.

5.3 DLL model internal status message input and output interface

5.3.1 Message construction

Collected by goto from tag (total 32 dimensional array)

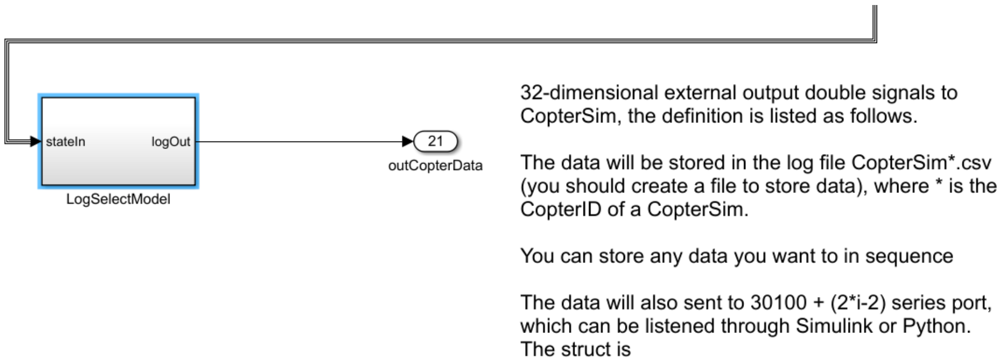




Here through the goto from label for 8-bit data (this data can be motor output, PWM output, etc.) output, the total output data is a 32-bit array, from label below the module is the output of excess data, to ensure that the final output of the total data is 32 bits.

5.3.2 Message output

Output through the outCopterData output port



5.4 Python- Interactive input and output interface for flight control hardware information

5.4.1 Serial transmission based on serial connection

```python
# import required libraries
import time
import math
import numpy as np
# import RflySim APIs
import PX4MavCtrlV4 as PX4MavCtrl

# Create MAVLink control API instance
mav1 = PX4MavCtrl.PX4MavCtrler(1)
# mav2 = PX4MavCtrl.PX4MavCtrler(2)
# mav2 = PX4MavCtrl.PX4MavCtrler(3)
# mavN --> 20100 + (N-1)*2
```

```
FaultInjectAPITest.py    PX4MavCtrlV4.py
C: > PX4PSP > RflySimAPIs > RflySimSDK > ctrl > PX4MavCtrlV4.py > PX4MavCtrler > _init_
        self.port = 20100+self.CopterID*2-2


        # UDP模式解析
        if (Com=='udp' or Com=='UDP' or Com=='Udp') and ID>10000: # 如果是UDP通信模式
            # 兼容旧版协议，如果ID是20100等端口输入，则自动计算CopterID
            self.port=ID
            self.CopterID = int((ID-20100)/2)+1
```

5.4.2 udp transfer based on usb connection

```python
import time
import math
import sys

import PX4MavCtrlV4 as PX4MavCtrl

# For hardware connection
#Windows use format PX4MavCtrler(ID,ip,'COM3',baud) for Pixhawk USB port connection
#Windows use format 'COM4' for Pixhawk serial port connection
#Linux use format '/dev/ttyUSB0' for USB, or '/dev/ttyAMA0' for Serial port (RaspberryPi
# PX4MavCtrler(1,'127.0.0.1','COM3',57600)
# PX4MavCtrler(1,'127.0.0.1','/dev/ttyS0',57600)
# constructor function
mav = PX4MavCtrl.PX4MavCtrler(Com = 'COM10:57600')

#mav.InitMavLoop(UDPMode), where UDPMode=0,1,2,3,4
```

```python
    # constructor function
    def __init__(self, ID=1, ip='127.0.0.1',Com='udp',port=0, simulinkDLL=False):
        global isEnableRdis
        self.isInPointMode = False
        self.isCom = False
        self.Com = Com
        self.baud = 115200
        self.isRealFly = 0
        self.ip = ip
        self.isRedis = False
        self.simulinkDLL = simulinkDLL

        # 这里是为了兼容之前的PX4MavCtrler('COM3:115200')串口协议，将来会取消
```

```python
        self.ComName = 'COM3' # 默认值，串口名字

        if Com[0:3]=='COM' or Com[0:3]=='com'  or Com[0:3]=='Com' or Com[0:3]=='/de': # 如果是串口连接方
            self.isCom = True # 串口通信模式
            strlist = Com.split(':')
            if port==0: # 默认值57600
                self.baud = 57600
            if(len(strlist) >= 2): # 串口号:波特率 协议解析，为了兼容旧接口
                if strlist[1].isdigit():
                    self.baud = int(strlist[1])
            self.ComName = strlist[0] # 串口名字
```

6. The establishment and use of automatic fault injection platform

6.1 Platform profile

6.1.1 The test case configures db.json

```json
{
    "faultcase": [
        {
            "CaseID": "RT01",
            "Subsystem": "Power",
            "Component": "Motor",
            "FaultID": "123450",
            "FaultType": "Elevator Steering Fault",
            "FaultMode": "Decreased efficiency of actuator execution",
            "CaseDescription": "Faulty steering gear for fixed-point navigation",
            "FaultParams": "FaultParam",
            "ControlSequence": "2,1;1,1,5;2,3,100,0,0;1,1,15;2,6,123540,5,10;1,1,10",
            "DataRequired": "Simulator ground truth data (pose and pose output)",
            "TestStatus": "Finished"
        }
    ],
    "testcase": "all",
    "Vision": "off"
}
```

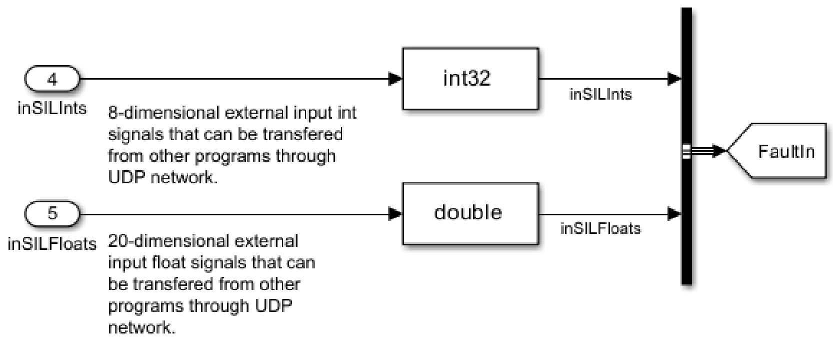
6.1.2 Configure the pod parameters Config.json

```json

{
    "VisionSensors":[
        {
            "SeqID":0,
            "TypeID":1,
            "TargetCopter":1,
            "TargetMountType":0,
            "DataWidth":640,
            "DataHeight":480,
            "DataCheckFreq":200,
            "SendProtocol":[0,127,0,0,1,9999,0,0],
            "CameraFOV":90,
            "SensorPosXYZ":[7,0,-2.8],
            "SensorAngEular":[0,0,0],
            "otherParams":[0,0,0,0,0,0,0,0]
        }
    ]
}
```

6.2 Rflysim interface protocol file PX4MavCtrlV4.py

6.2.1 Fault injection protocol class PX4SILIntFloat

```
# //输出到CopterSim DLL模型的SILints和SILFloats数据
# struct PX4SILIntFloat{
#     int checksum;//1234567897
#     int CopterID;
#     int inSILInts[8];
#     float inSILFLoats[20];
# };
#struct.pack 10i20f
```



Here is fault injection, which is divided into 8-bit fault ID injection and 20-bit fault parameter injection.

6.2.2 Unlocked/unlocked interface SendMavArm

```
# send MAVLink command to Pixhawk to Arm/Disarm the drone
def SendMavArm(self, isArm=0):
    """ Send command to PX4 to arm or disarm the drone
    """
    if self.UDPMode>1.5:
        if (isArm):
            self.SendMavCmdLong(mavlink2.MAV_CMD_COMPONENT_ARM_DISARM, 1)
        else:
            self.SendMavCmdLong(mavlink2.MAV_CMD_COMPONENT_ARM_DISARM, 0, 21196.0)
    else:
        ctrls=[isArm,0,0,0]
        self.sendUDPSimpData(9,ctrls)
```

6.2.3 The target location interface of the drone SendPosNED

```python
# send target position in earth NED frame
def SendPosNED(self,x=0,y=0,z=0,yaw=0):
    """ Send vehicle targe position (m) to PX4 in the earth north-east-down (NED) frame with yaw control (rad)
    when the vehicle fly above the ground, then z < 0
    """
    self.offMode=0 # SET_POSITION_TARGET_LOCAL_NED
    self.ctrlMode=2 #地球位置控制模式
    self.EnList = [1,0,0,0,1,0]
    self.type_mask=self.TypeMask(self.EnList)
    self.coordinate_frame = mavlink2.MAV_FRAME_LOCAL_NED
    self.pos=[x,y,z]
    self.vel = [0,0,0]
    self.acc = [0, 0, 0]
    self.yawrate = 0
    self.yaw = yaw

# send target position in earth NED frame
def SendVelYawAlt(self,vel=10,yaw=6.28,alt=-100):
    """ Send vehicle targe position (m) to PX4 in the earth north-east-down (NED) frame with yaw control (rad)
    when the vehicle fly above the ground, then z < 0
    """

    if abs(yaw)<0.00001:
        yaw = 6.28
    self.offMode=0 # SET_POSITION_TARGET_LOCAL_NED
    self.ctrlMode=13 #速度高度偏航控制模式
    self.type_mask=int("000111000000", 2)
    self.coordinate_frame = 1
    self.pos=[0,0,alt]
    self.vel = [yaw,vel,0]
    self.acc = [0, 0, 0]
    self.yawrate = 0
    self.yaw = yaw
```

```python
# send target position in earth NED frame
def SendPosNEDNoYaw(self,x=0,y=0,z=0):
    """ Send vehicle targe position (m) to PX4 in the earth north-east-down (NED) frame without yaw control (rad)
    when the vehicle fly above the ground, then z < 0
    """
    self.offMode=0 # SET_POSITION_TARGET_LOCAL_NED
    self.ctrlMode=2 #地球位置控制模式
    self.EnList = [1,0,0,0,0,0]
    self.type_mask=self.TypeMask(self.EnList)
    self.coordinate_frame = mavlink2.MAV_FRAME_LOCAL_NED
    self.pos=[x,y,z]
    self.vel = [0,0,0]
    self.acc = [0, 0, 0]
    self.yawrate = 0
    self.yaw = 0
```

```python
# send target position in body FRD frame
def SendPosFRD(self,x=0,y=0,z=0,yaw=0):
    """ Send vehicle targe position (m) to PX4 in the body forward-rightward-downward (FRD) frame with yaw control (rad)
    when the vehicle fly above the ground, then z < 0
    """
    self.offMode=0 # SET_POSITION_TARGET_LOCAL_NED
    self.ctrlMode=3 #机体位置控制模式
    self.EnList = [1,0,0,0,1,0]
    self.type_mask=self.TypeMask(self.EnList)
    self.coordinate_frame = mavlink2.MAV_FRAME_BODY_NED
    self.pos=[x,y,z]
    self.vel = [0,0,0]
    self.acc = [0, 0, 0]
    self.yawrate = 0
    self.yaw = yaw

# send target position in body FRD frame
def SendPosFRDNoYaw(self,x=0,y=0,z=0):
    """ Send vehicle targe position (m) to PX4 in the body forward-rightward-downward (FRD) frame without yaw control (rad)
    when the vehicle fly above the ground, then z < 0
    """
    self.offMode=0 # SET_POSITION_TARGET_LOCAL_NED
    self.ctrlMode=3 #机体位置控制模式
    self.EnList = [1,0,0,0,0,0]
    self.type_mask=self.TypeMask(self.EnList)
    self.coordinate_frame = mavlink2.MAV_FRAME_BODY_NED
    self.pos=[x,y,z]
    self.vel = [0,0,0]
    self.acc = [0, 0, 0]
    self.yawrate = 0
    self.yaw = 0
```

```python
def SendPosNEDExt(self,x=0,y=0,z=0,mode=3,isNED=True):
    """ Send vehicle targe position (m) to PX4
    when the vehicle fly above the ground, then z < 0
    """
    self.offMode=0 # SET_POSITION_TARGET_LOCAL_NED
    self.EnList = [1,0,0,0,1,0]
    self.type_mask=self.TypeMask(self.EnList)
    if mode==0:
        # Gliding setpoint
        self.type_mask=int(292) # only for fixed Wing
    elif mode==1:
        # Takeoff setpoints
        self.type_mask=int(4096) # only for fixed Wing
    elif mode==2:
        # Land setpoints
        self.type_mask=int(8192) # only for fixed Wing
    elif mode==3:
        # Loiter setpoints
        # for Rover:  Loiter setpoint (vehicle stops when close enough to setpoint).
        # for fixed wing:  Loiter setpoint (fly a circle centred on setpoint).
        self.type_mask=int(12288)
    elif mode==4:
        # Idle setpoint
        # only for fixed wing
        # Idle setpoint (zero throttle, zero roll / pitch).
        self.type_mask=int(16384)
    if isNED:
        self.coordinate_frame = mavlink2.MAV_FRAME_LOCAL_NED
    else:
        self.coordinate_frame = mavlink2.MAV_FRAME_BODY_NED
    self.pos=[x,y,z]
    self.vel = [0,0,0]
    self.acc = [0, 0, 0]
    self.yawrate = 0
    self.yaw = 0
```

6.2.4 Drone flight speed interface FlyVel

```
self.uavVelNED = [0, 0, 0] # Estimated local velocity from PX4 in NED frame
        self.trueVelNED = [0, 0, 0] # True simulated speed from CopterSim's DLL model  in NED frame
```

uavVel runs the synthetic flight speed for the routine, trueVel runs the real flight speed without the synthesis.

6.3 Platform command control interface command.py

6.3.1 Description of the database fault command protocol

```
def __init__(self,mav):
    self.CID = 2
    self.mav = mav
    self.isArm = 0 #  解锁标志(解锁意味着开始记录数据)
    self.isDone = 0 #  任务完成标志
    self.LandFlag = 0
    self.LandFlagtag = 0
    #  初始化故障注入参数
    self.silInt = np.zeros(8).astype(int).tolist()
    self.silFloats = np.zeros(20).astype(float).tolist()
    self.isRecord = 0
        self.isInject = 0
```

6.3.2 Ununlocked command interface DisArm(self)

```
def DisArm(self): # ID = 2
    self.isDone = 0
    self.mav.SendMavArm(0)
    print('DisArmed')
        self.isDone = 1
```

6.3.3 Unlock command interface Arm(self)

```
def Arm(self): # ID = 1
    self.isDone = 0
    self.mav.SendMavArm(1)
    print('Armed')
    self.isArm = 1
        self.isDone = 1
```

6.3.4 FlyPos(self,pos)

```
def FlyPos(self,pos): # ID = 3
    self.isDone = 0
```

```
        self.mav.SendPosNED(pos[0],pos[1],pos[2])
        print('Send Pos {}'.format(pos))
            self.isDone = 1
```

6.3.5 FlyVel(self,vel)

```
def FlyVel(self,vel): # ID = 4
        self.isDone = 0
        self.mav.SendVelNED(vel[0],vel[1],vel[2])
        print('Send Vel {}'.format(vel))
            self.isDone = 1
```

6.3.6 Land(self)

```
def Land(self): # ID = 5
        self.isDone = 0
        self.LandFlag = 1
        if self.LandFlagtag == 0:
            self.mav.SendVelNED(0,0,2)
            print('Start Landing')
            self.LandFlagtag = 1
        if abs(self.mav.truePosNED[2]) < 1.5:
            print('Landed')
            self.isDone = 1
            # 复位成功后重置 LandFlagtag 为 0
                self.LandFlagtag = 0
```

6.3.7 FaultInject(self,param)

```
def FaultInject(self,param): # ID = 6
        self.isDone = 0
        inInts = np.array([])
        inFloats = np.array([])
        for i in range(len(param)):
            if param[i] >= 123450:
                inInts = np.append(inInts,param[i])
            else:
                inFloats = np.append(inFloats,param[i])

        for i in range(len(inInts)):
            self.silInt[i] = inInts[i].astype(int)
        for i in range(len(inFloats)):
            self.silFloats[i] = inFloats[i].astype(np.double)

        print('Start Inject Fault')
        self.mav.SendMavCmdLong(183,16,568,1,1,1,1,1)
        # 作为一个故障输入的标志，方便后续对故障输入时间的判断
```

```
        self.mav.sendSILIntFloat(self.silInt,self.silFloats)
        self.isDone = 1
        self.isRecord = 1 #  故障注入之后开始收集数据
            self.isInject = 1
```

6.4 Pod Vision API VisionCaptureApi.py

6.4.1 Start visual image capture with startImgCap

```
def startImgCap(self, isRemoteSend=False):
        """start loop to receive image from UE4,
        isRemoteSend=true will forward image from memory to UDP port
        """
        self.isRemoteSend = isRemoteSend
        global isEnableRosTrans
        memList = []
        udpList = []
        if isEnableRosTrans:
            self.time_record = np.zeros(len(self.VisSensor))
            if is_use_ros1:
                self.rostime = np.ndarray(len(self.time_record), dtype=rospy.Time)
            else:
                self.rostime = np.ndarray(len(self.time_record), dtype=rclpy.time.Time)

        for i in range(len(self.VisSensor)):
            self.Img = self.Img + [0]
            self.Img_lock = self.Img_lock + [
                threading.Lock()
            ]  #  每个传感器都是一个独立的线程，应时使用独立的锁
            self.ImgData = self.ImgData + [0]
            self.hasData = self.hasData + [False]
            self.timeStmp = self.timeStmp + [0]
            self.imgStmp = self.imgStmp + [0]

            TarCopt = self.VisSensor[i].TargetCopter
            starTime=0
            for j in range(len(self.RflyTimeVect)):
                if self.RflyTimeVect[j].copterID == TarCopt:
                    if isEnableRosTrans:
                        starTime=self.RflyTimeVect[j].rosStartTimeStmp
                    else:
                        starTime=self.RflyTimeVect[j].pyStartTimeStmp
                    print('Got start time for SeqID #',self.VisSensor[i].SeqID)
            self.rflyStartStmp = self.rflyStartStmp + [starTime]
```

```
            IP = (
                str(self.VisSensor[i].SendProtocol[1])
                + "."
                + str(self.VisSensor[i].SendProtocol[2])
                + "."
                + str(self.VisSensor[i].SendProtocol[3])
                + "."
                + str(self.VisSensor[i].SendProtocol[4])
            )
            if IP == "0.0.0.0":
                IP = "127.0.0.1"
            if self.RemotSendIP != "":
                IP = self.RemotSendIP
            self.IpList = self.IpList + [IP]
            self.portList = self.portList + [self.VisSensor[i].SendProtocol[5]]
            if self.VisSensor[i].SendProtocol[0] == 0:
                memList = memList + [i]
            else:
                udpList = udpList + [i]

        if len(memList) > 0:
            self.t_menRec = threading.Thread(target=self.img_mem_thrd, args=(memList,))
            self.t_menRec.start()

        if len(udpList) > 0:
            # print('Enter UDP capture')
            for i in range(len(udpList)):
                udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                udp.setsockopt(socket.SOL_SOCKET, socket.SO_RCVBUF, 60000 * 100)
                udp.bind(("0.0.0.0", self.portList[udpList[i]]))
                typeID = self.VisSensor[udpList[i]].TypeID
                t_udpRec = threading.Thread(
                    target=self.img_udp_thrdNew,
                    args=(
                        udp,
                        udpList[i],
                        typeID,
                    ),
                )
                    t_udpRec.start()
```

6.4.2 Update visual image sendUpdateUEImage

```
def sendUpdateUEImage(self, vs=VisionSensorReq(), windID=0, IP="127.0.0.1"):
```

```
        if not isinstance(vs, VisionSensorReq):
            raise Exception("Wrong data input to addVisSensor()")
        intValue = [
            vs.checksum,
            vs.SeqID,
            vs.TypeID,
            vs.TargetCopter,
            vs.TargetMountType,
            vs.DataWidth,
            vs.DataHeight,
            vs.DataCheckFreq,
        ] + vs.SendProtocol
        if self.isNewJson: # 使用新版协议发送
            floValue = [vs.CameraFOV] + vs.SensorPosXYZ + [vs.EularOrQuat] +
vs.SensorAngEular + vs.SensorAngQuat + vs.otherParams
            buf = struct.pack("16H28f", *intValue, *floValue)
        else:  # 使用旧版协议发送
            floValue = [vs.CameraFOV] + vs.SensorPosXYZ + vs.SensorAngEular +
vs.otherParams[0:8]
            buf = struct.pack("16H15f", *intValue, *floValue)

        self.udp_socket.sendto(buf, (IP, 20010 + windID))
        if self.RemotSendIP != "" and self.RemotSendIP != "127.0.0.1":
                self.udp_socket.sendto(buf, (self.RemotSendIP, 20010 + windID))
```

6.4.3 The pod parameter configuration file loads the interface jsonLoad

```
def jsonLoad(self, ChangeMode=-1, jsonPath=""):
        """load config.json file to create camera list for image capture,
        if ChangeMode>=0, then the SendProtocol[0] will be set to ChangeMode to change the
transfer style
        """
        #print(sys.path[0])
        if os.path.isabs(jsonPath):
            print('Json use absolute path mode')
        else:
            print('Json use relative path mode')
            if len(jsonPath) == 0:
                jsonPath = sys.path[0] + "/Config.json"
            else:
                jsonPath = sys.path[0] + "/" + jsonPath
```

```
    print("jsonPath=", jsonPath)

    if not os.path.exists(jsonPath):
        print("The json file does not exist!")
        return False
    self.isNewJson=False
    with open(jsonPath, "r", encoding="utf-8") as f:
        jsData = json.loads(f.read())
        if len(jsData["VisionSensors"]) <= 0:
            print("No sensor data is found!")
            return False
        for i in range(len(jsData["VisionSensors"])):
            visSenStruct = VisionSensorReq()
            if isinstance(jsData["VisionSensors"][i]["SeqID"], int):
                visSenStruct.SeqID = jsData["VisionSensors"][i]["SeqID"]
            else:
                print("Json data format is wrong!")
                continue

            if isinstance(jsData["VisionSensors"][i]["TypeID"], int):
                visSenStruct.TypeID = jsData["VisionSensors"][i]["TypeID"]
            else:
                print("Json data format is wrong!")
                continue

            if isinstance(jsData["VisionSensors"][i]["TargetCopter"], int):
                visSenStruct.TargetCopter = jsData["VisionSensors"][i][
                    "TargetCopter"
                ]
            else:
                print("Json data format is wrong!")
                continue

            if isinstance(jsData["VisionSensors"][i]["TargetMountType"], int):
                visSenStruct.TargetMountType = jsData["VisionSensors"][i][
                    "TargetMountType"
                ]
            else:
                print("Json data format is wrong!")
                continue

            if isinstance(jsData["VisionSensors"][i]["DataWidth"], int):
                visSenStruct.DataWidth = jsData["VisionSensors"][i]["DataWidth"]
            else:
```

```
            print("Json data format is wrong!")
            continue

        if isinstance(jsData["VisionSensors"][i]["DataHeight"], int):
            visSenStruct.DataHeight = jsData["VisionSensors"][i]["DataHeight"]
        else:
            print("Json data format is wrong!")
            continue

        if isinstance(jsData["VisionSensors"][i]["DataCheckFreq"], int):
            visSenStruct.DataCheckFreq = jsData["VisionSensors"][i][
                "DataCheckFreq"
            ]
        else:
            print("Json data format is wrong!")
            continue

        if isinstance(
            jsData["VisionSensors"][i]["CameraFOV"], float
        ) or isinstance(jsData["VisionSensors"][i]["CameraFOV"], int):
            visSenStruct.CameraFOV = jsData["VisionSensors"][i]["CameraFOV"]
        else:
            print("Json data format is wrong!")
            continue

        if len(jsData["VisionSensors"][i]["SendProtocol"]) == 8:
            visSenStruct.SendProtocol = jsData["VisionSensors"][i][
                "SendProtocol"
            ]
            if ChangeMode != -1:
                # 如果是远程接收模式，那么读图这里需要配置为 UDP 接收
                visSenStruct.SendProtocol[0] = ChangeMode
        else:
            print("Json data format is wrong!")
            continue

        if len(jsData["VisionSensors"][i]["SensorPosXYZ"]) == 3:
            visSenStruct.SensorPosXYZ = jsData["VisionSensors"][i][
                "SensorPosXYZ"
            ]
        else:
            print("Json data format is wrong!")
            continue
```

```
            isNewProt=False

            if 'EularOrQuat' in jsData["VisionSensors"][i]:
                isNewProt=True
                visSenStruct.EularOrQuat = jsData["VisionSensors"][i][
                    "EularOrQuat"
                ]
            else:
                visSenStruct.EularOrQuat=0

            if len(jsData["VisionSensors"][i]["SensorAngEular"]) == 3:
                visSenStruct.SensorAngEular = jsData["VisionSensors"][i][
                    "SensorAngEular"
                ]
            else:
                print("Json data format is wrong!")
                continue

            if isNewProt:
                if len(jsData["VisionSensors"][i]["SensorAngQuat"]) == 4:
                    visSenStruct.SensorAngQuat = jsData["VisionSensors"][i][
                        "SensorAngQuat"
                    ]
                else:
                    print("Json data format is wrong!")
                    continue


            if isNewProt: #  新协议使用 16 维的 otherParams
                if len(jsData["VisionSensors"][i]["otherParams"]) == 16:
                    visSenStruct.otherParams = jsData["VisionSensors"][i]["otherParams"]
                else:
                    print("Json data format is wrong!")
                    continue
            else:
                if len(jsData["VisionSensors"][i]["otherParams"]) == 8:
                    visSenStruct.otherParams =
jsData["VisionSensors"][i]["otherParams"]+[0]*8 #  扩展到 16 维
                else:
                    print("Json data format is wrong!")
                    continue
            self.VisSensor = self.VisSensor + [visSenStruct]

            if ~self.isNewJson and isNewProt:
                self.isNewJson=True
```

```
        if (len(self.VisSensor)) <= 0:
            print("No sensor is obtained.")
            return False
        print("Got", len(self.VisSensor), "vision sensors from json")

        if len(self.RflyTimeVect)==0 and ~self.tTimeStmpFlag:
            #print('Start listening CopterSim time Data')
            self.StartTimeStmplisten()
            time.sleep(2)
            self.endTimeStmplisten()
        if len(self.RflyTimeVect)>0:
            print('Got CopterSim time Data for img')
        else:
            print('No CopterSim time Data for img')

            return True
```

6.4.4 Add vision Sensor addVisSensor

```
def addVisSensor(self, vsr=VisionSensorReq()):
        """Add a new VisionSensorReq struct to the list"""
        if isinstance(vsr, VisionSensorReq):
            self.VisSensor = self.VisSensor + [copy.deepcopy(vsr)]
        else:
                raise Exception("Wrong data input to addVisSensor()")
```

6.5 Platform automation test API AutoTest.py

6.5.1 Automated Test TestcasePro()

```
def TestcasePro():
    path = sys.path[0]+'\db.json'

    with open(path, "r",encoding='utf-8') as f:
        db_data = json.load(f)

    global caselist
    if db_data.get('testcase') == 'all':
        caselist = []
        # 提取 faultcase 的 CaseID
        casedata = db
        for data in casedata:
            ID = data.get('CaseID')
```

```
            caselist.append(ID)
    else:
        caselist = re.split(',',db_data.get('testcase')) # ['3', '1', '30']
             caselist = [int(val) for val in caselist] # [3, 1, 30]
```

6.5.2 Control Instruction Interface DoCmd(ctrlseq)

```
def DoCmd(ctrlseq):
    cmdseq = ctrlseq # '2,3,0,0,-20'
    cmdseq = re.findall(r'-?\d+\.?[0-9]*',cmdseq) # ['2', '3', '0', '0', '-20']
    cmdCID = cmdseq[0]
    if   cmdCID in CID:
        FID = FIDPro(cmdCID)
        # 有参数输入
        if len(cmdseq) > 2:
            # 提取参数
            param = cmdseq[2:len(cmdseq)]
            param = [float(val) for val in param]
            FID[cmdseq[1]](param)

        else:
            FID[cmdseq[1]]()
    else:
             print('Command input error, please re-enter')
```

6.5.3 Get Instruction Interface FIDPro(cmdCID)

```
def FIDPro(cmdCID):
    if cmdCID == '1':
        FID = {
            '1':CID1obj.Wait,
            '2':CID1obj.WaitReset
        }
    elif cmdCID == '2':
        FID = {
            '1':CID2obj.Arm,
            '2':CID2obj.DisArm,
            '3':CID2obj.FlyPos,
            '4':CID2obj.FlyVel,
            '5':CID2obj.Land,
            '6':CID2obj.FaultInject
        }
         return FID
```

6.5.4 Command Sequence Control Interface CmdPro(seq)

```
def CmdPro(seq):
```

```
    case = re.split(';',seq)
    cmd = np.array([])
    for i in range(len(case)):
        cmd = np.append(cmd,case[i])
         return cmd
```

6.6 Database failure use case read and write mavdb.py

6.6.1 get cursor(self) Get cursor(self)

```
def get_cursor(self):
        self.mydb = sqlite3.connect(".\\fault.db")
        self.mydb.row_factory = dict_factory
            self.cursor=self.mydb.cursor()
```

6.6.2 get fault case(self) Get fault case(self)

```
# 获取故障测试用例
    def get_fault_case(self):
        # 获取游标
        mavdb.get_cursor(self)
        sql='''
        select    *
        from       faultcase
        '''
        self.cursor.execute(sql)
        result=self.cursor.fetchall()
            return result
```

6.7 The writing and use of fault injection test case set

```
# import required libraries
import time
import math
import numpy as np
# import RflySim APIs
import PX4MavCtrlV4 as PX4MavCtrl

# Create MAVLink control API instance
mav1 = PX4MavCtrl.PX4MavCtrler(1)
# mav2 = PX4MavCtrl.PX4MavCtrler(2)
# mav2 = PX4MavCtrl.PX4MavCtrler(3)
# mavN --> 20100 + (N-1)*2

# Init MAVLink data receiving loop
mav1.InitMavLoop()
#mav2.InitMavLoop(), ...
```

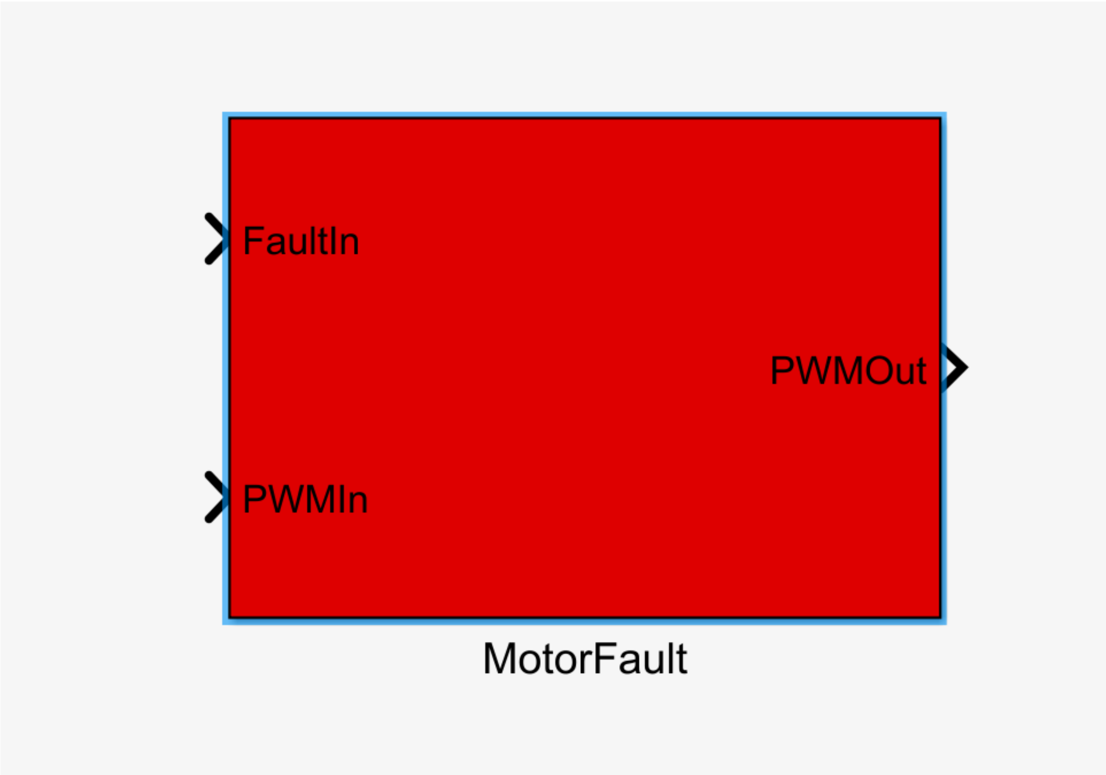
```
time.sleep(0.5)
mav1.InitTrueDataLoop()
time.sleep(0.5)

mav1.initOffboard()

lastTime = time.time()
startTime = time.time()
# time interval of the timer
timeInterval = 1/30.0 #here is 0.0333s (30Hz)

# flags for vehicle 1
flag = 0
flagTime=startTime

# Start a endless loop with 30Hz, timeInterval=1/30.0
while True:
    lastTime = lastTime + timeInterval
    sleepTime = lastTime - time.time()
    if sleepTime > 0:
        time.sleep(sleepTime) # sleep until the desired clock
    else:
        lastTime = time.time()
    # The following code will be executed 30Hz (0.0333s)

    # Create the mission to vehicle 1
    if time.time() - startTime > 5 and flag == 0:
        # The following code will be executed at 5s
        print("5s, Arm the drone")
        flag = 1
        flagTime=time.time()
        mav1.SendMavArm(True) # Arm the drone
        #mav2.SendMavArm(True), ...

        print("Arm the drone!")

        # Takeoff to ten meters height
        mav1.SendPosNED(0,0,-10)
        time.sleep(0.5)
        print("开始起飞")
        mav1.SendMavArm(True) # Arm the drone

    if time.time() - startTime > 20 and flag==1:
```

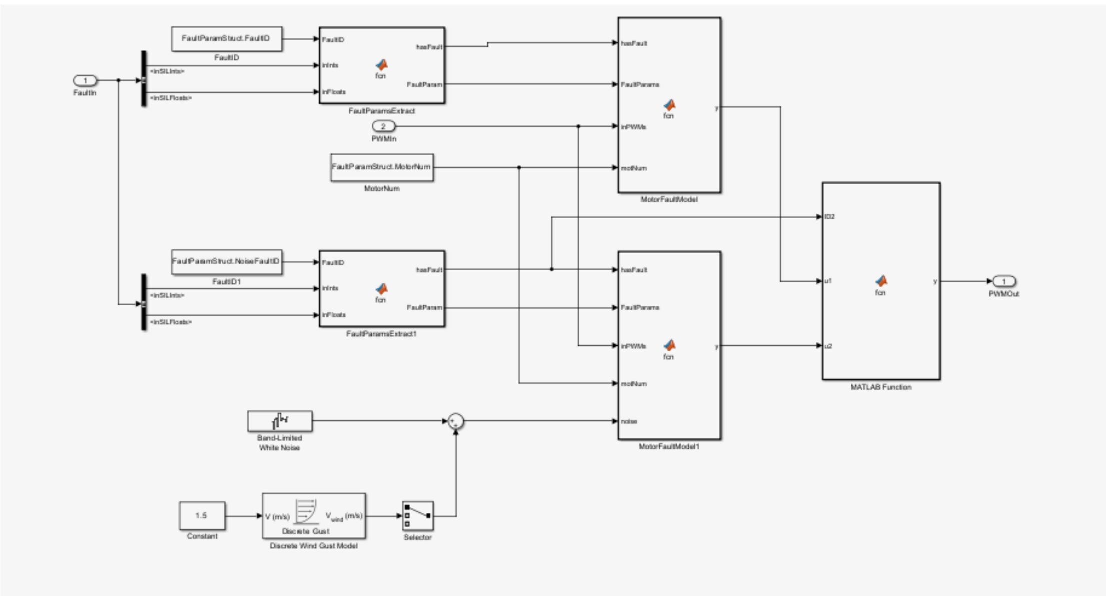
```
        #np.zeros()
        silInt=np.zeros(8).astype(int).tolist()
        silFloat=np.zeros(20).astype(float).tolist()
        silInt[0:2]=[123450,123450]
        silFloat[0:4]=[0,0,0,0]
        # silInt[0:1]=[123540]
        # silFloat[0:2]=[15,20]
        mav1.sendSILIntFloat(silInt,silFloat)
        print('Inject a fault, and start loging')
        flag=2

    if flag==2:
        print(mav1.uavPosNED,mav1.truePosNED)
        if time.time() - startTime > 50:
            break

print('Sim End')

mav1.endMavLoop()
     mav1.EndTrueDataLoop()
```

7. Software in the loop simulink model fault injection interface with use

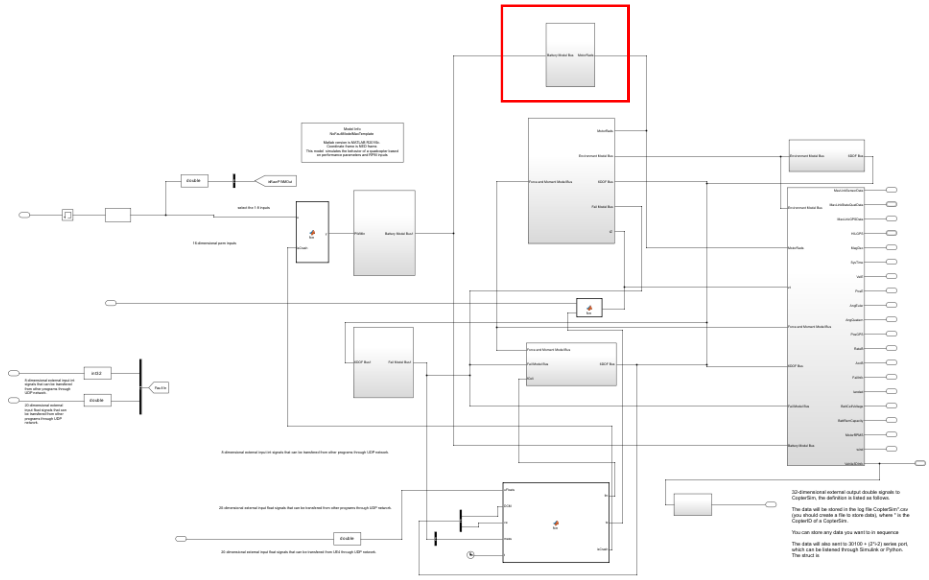
First, in most of the routines in this chapter, there is a fault module library for MulticopterModelLib.slx, which contains fault injection modules with various faults.

7.1 MotorFault injection and use

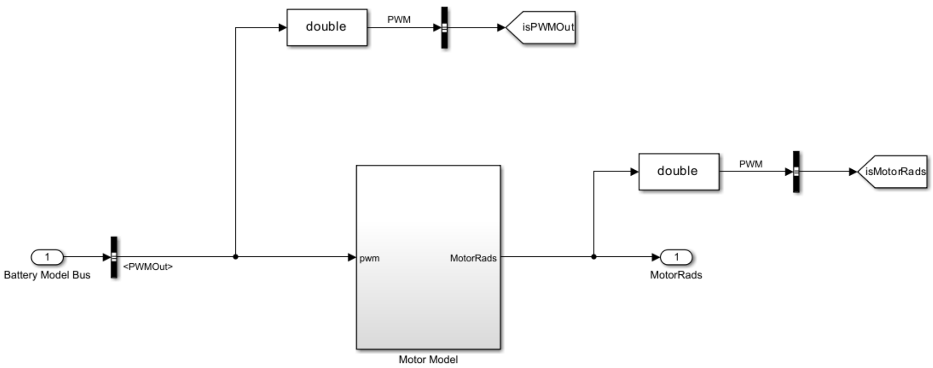




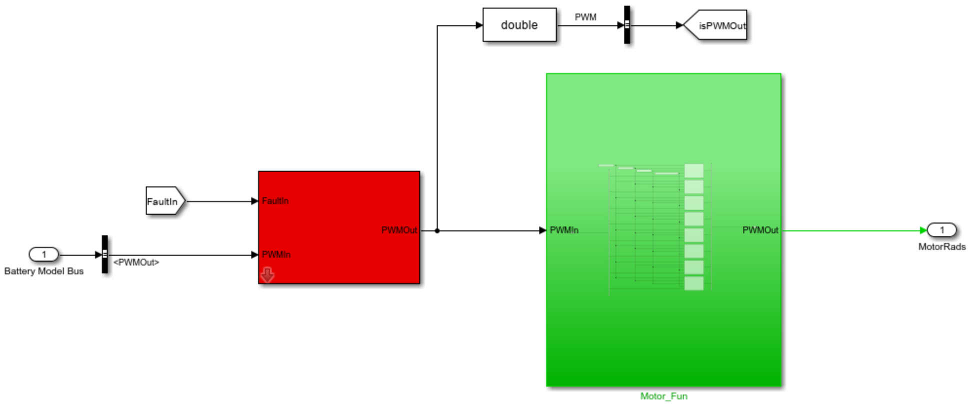
The MotorFault module in the routine file is the motor fault package library. We can check the inside of the package. This module needs to be used together with the maximum template without fault injection and the minimum template without fault injection.



This routine file is the largest template without fault injection, from which we need to find the corresponding position of the motor module.

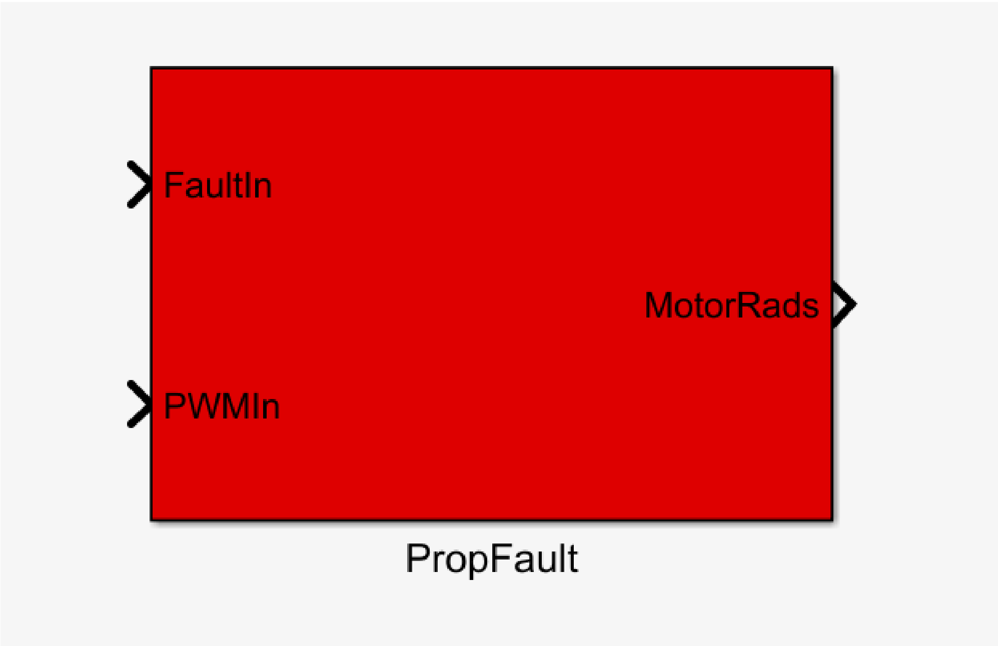


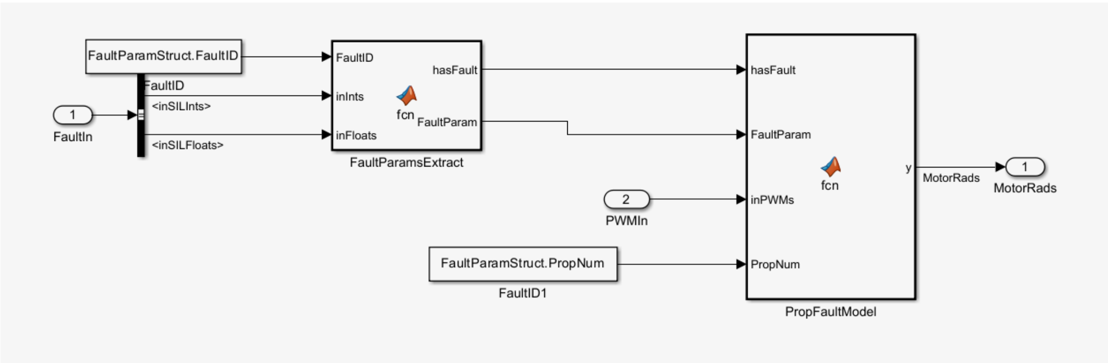
This is required to change the corresponding module from the wrapper library.



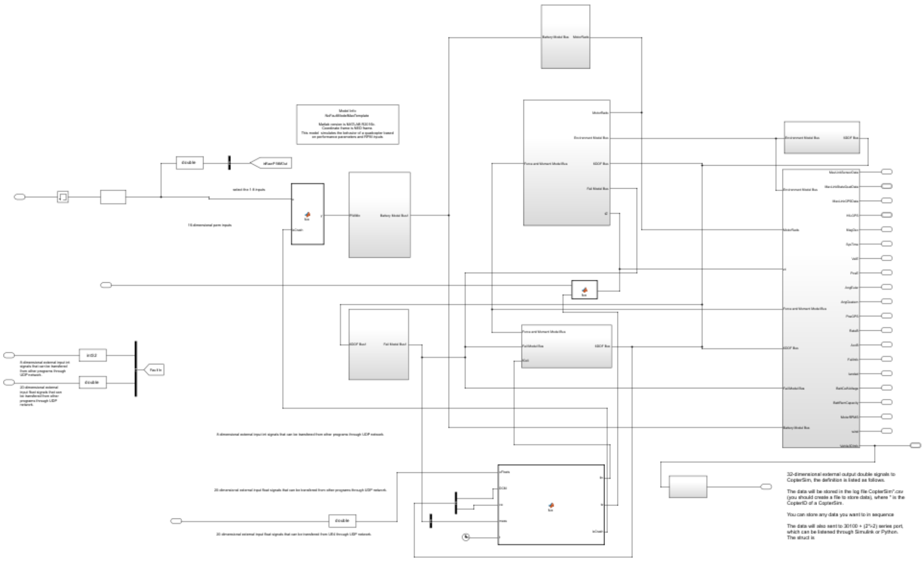
After that, we can perform motor fault injection experiments on the model.

7.2 PropFault injection and use

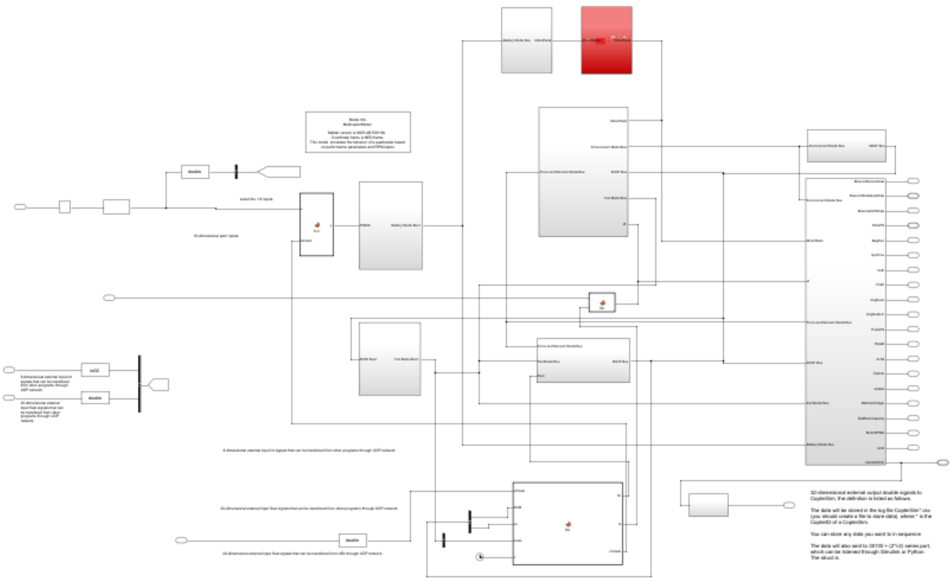


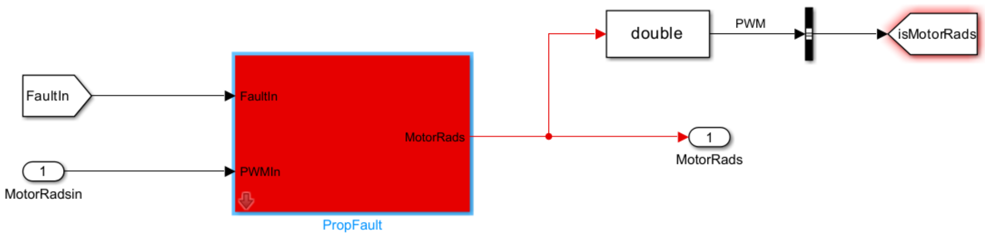


The PropFault module in the routine file is the propeller fault package library that we can look inside the package. This module needs to be used in conjunction with the largest template without fault injection and the smallest template without fault injection. The template with fault injection needs to be replaced with the non-fault module.



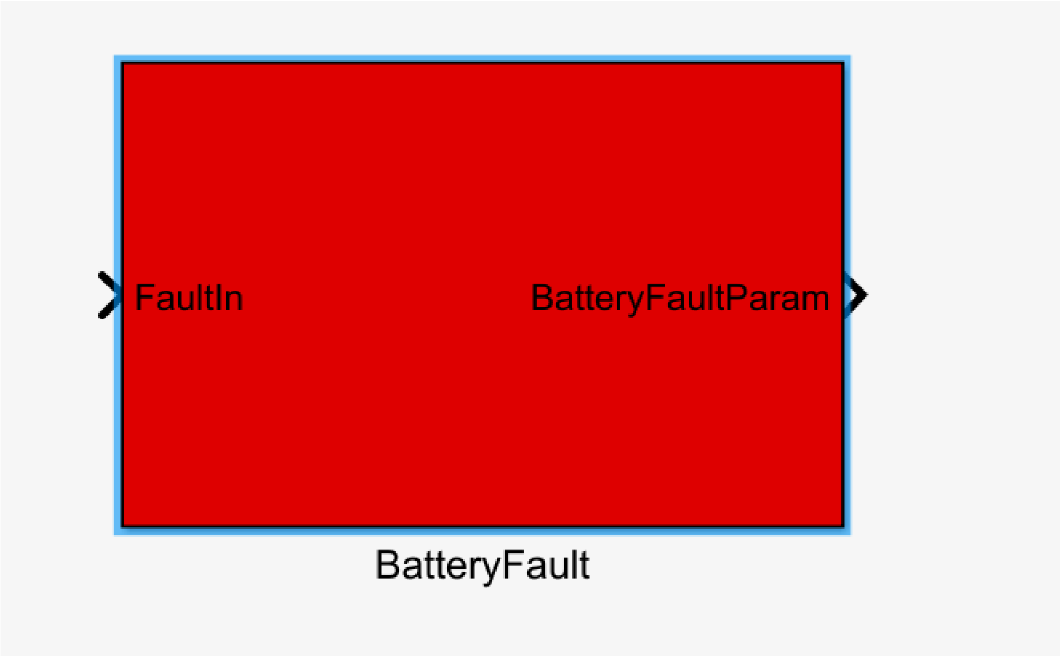
This routine file is the largest template without fault injection, and we need to add the previous propeller fault injection module to it.

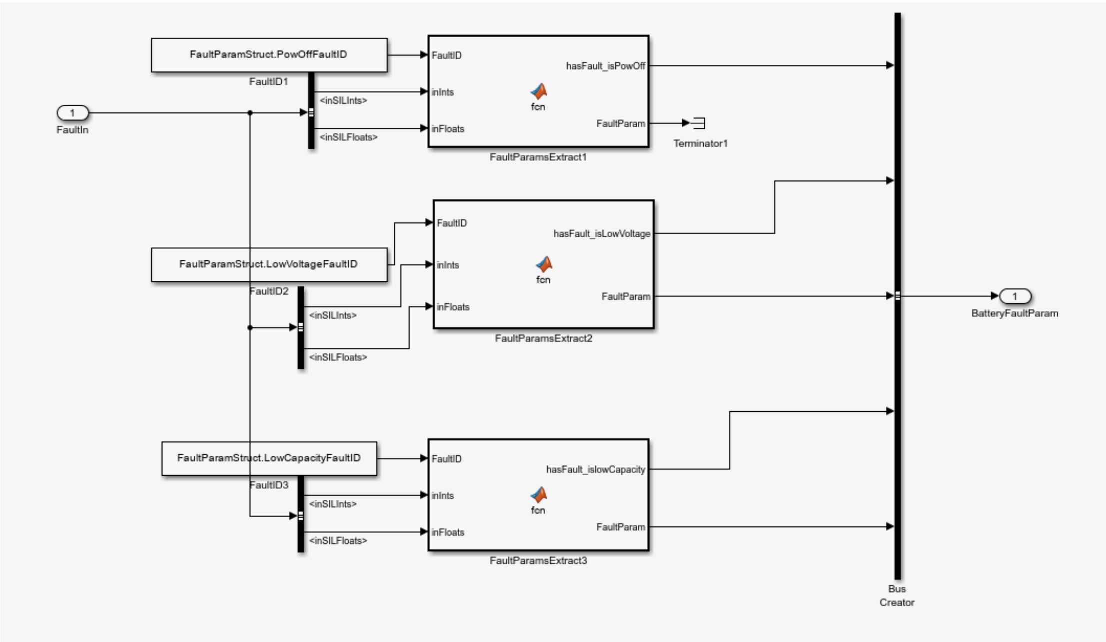




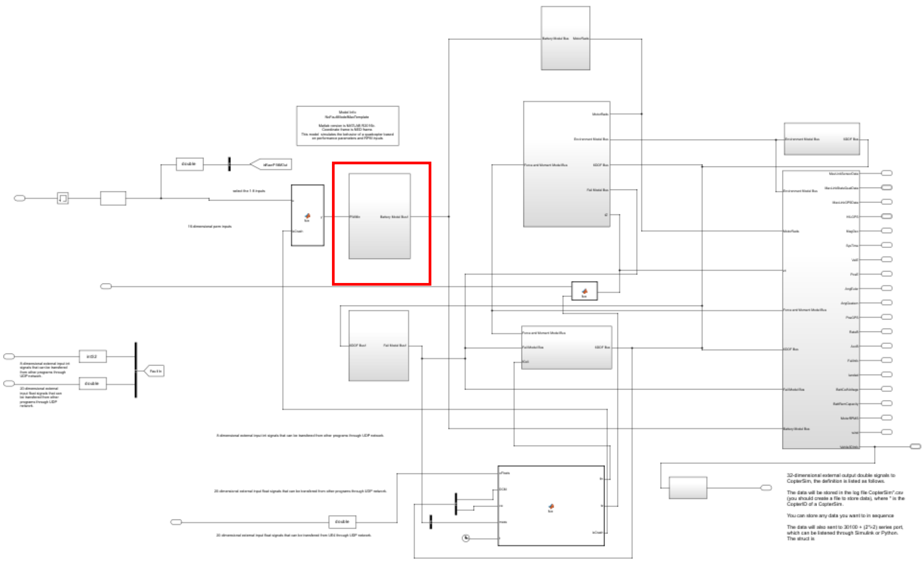
After that, we can perform propeller fault injection experiments on the model.

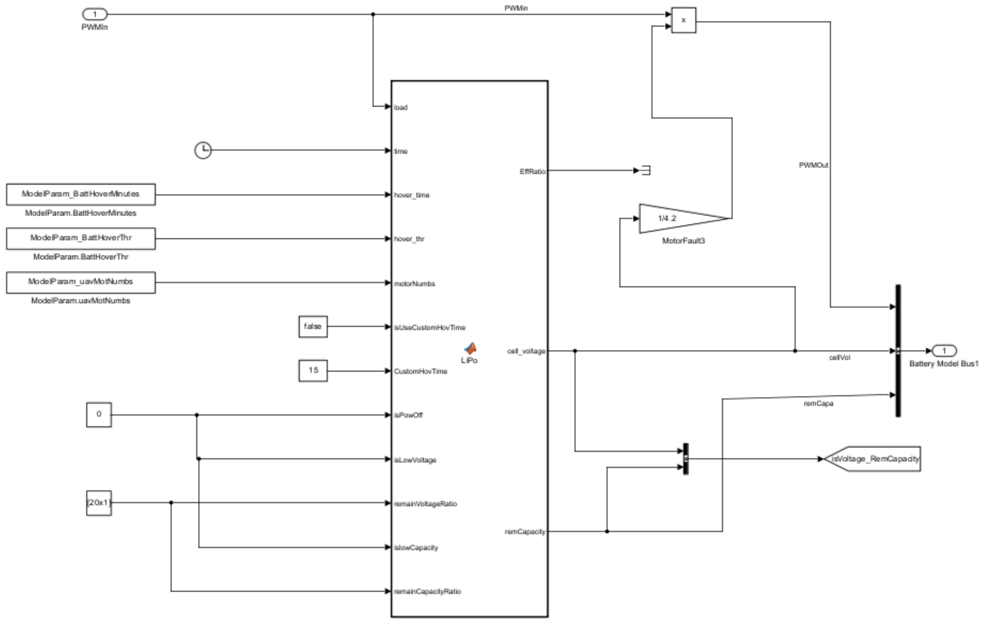
7.3 BatteryFault injection and use



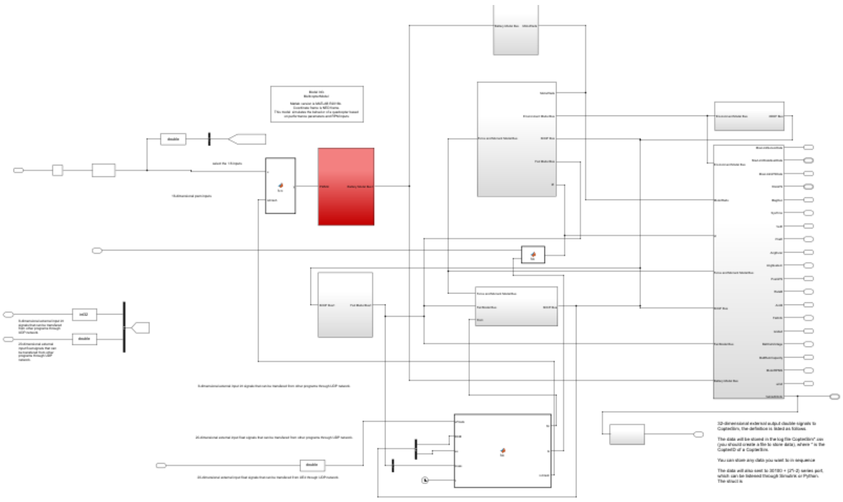


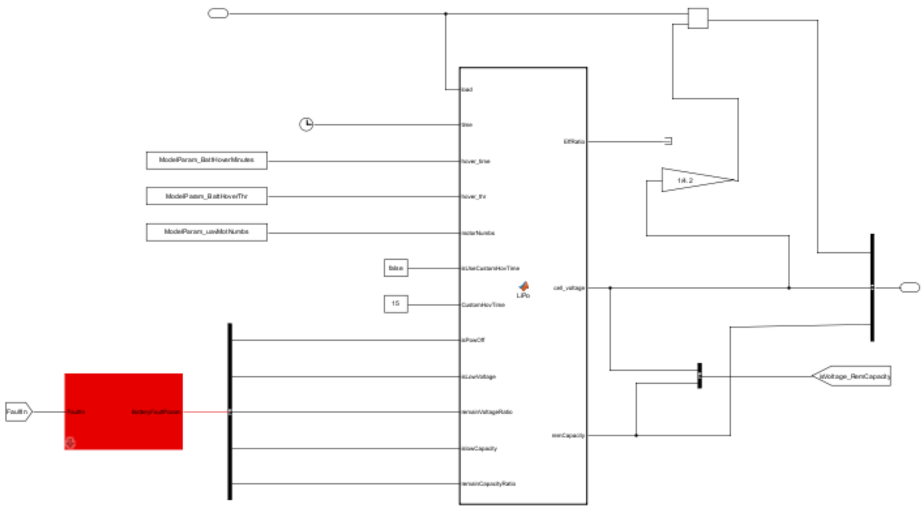
The BatteryFault module in the routine file is a battery fault encapsulation library. You can check the inside of the encapsulation. This module needs to be used together with the maximum template without fault injection and the minimum template without fault injection.





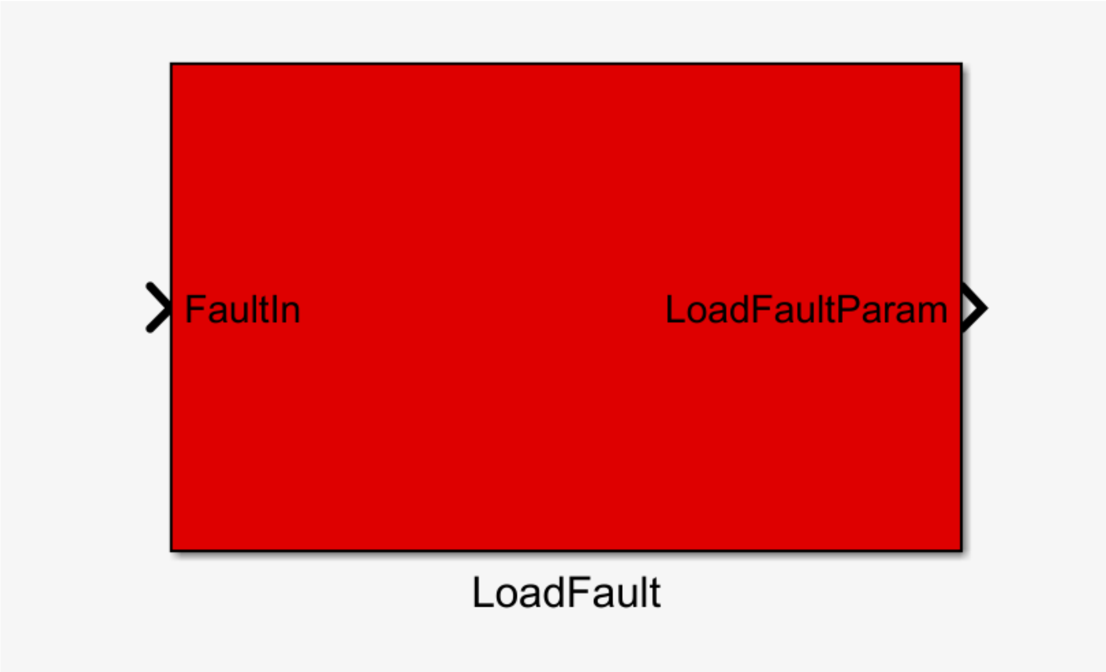
This routine file is the largest template without fault injection, and we need to find the corresponding location of the battery module from it.

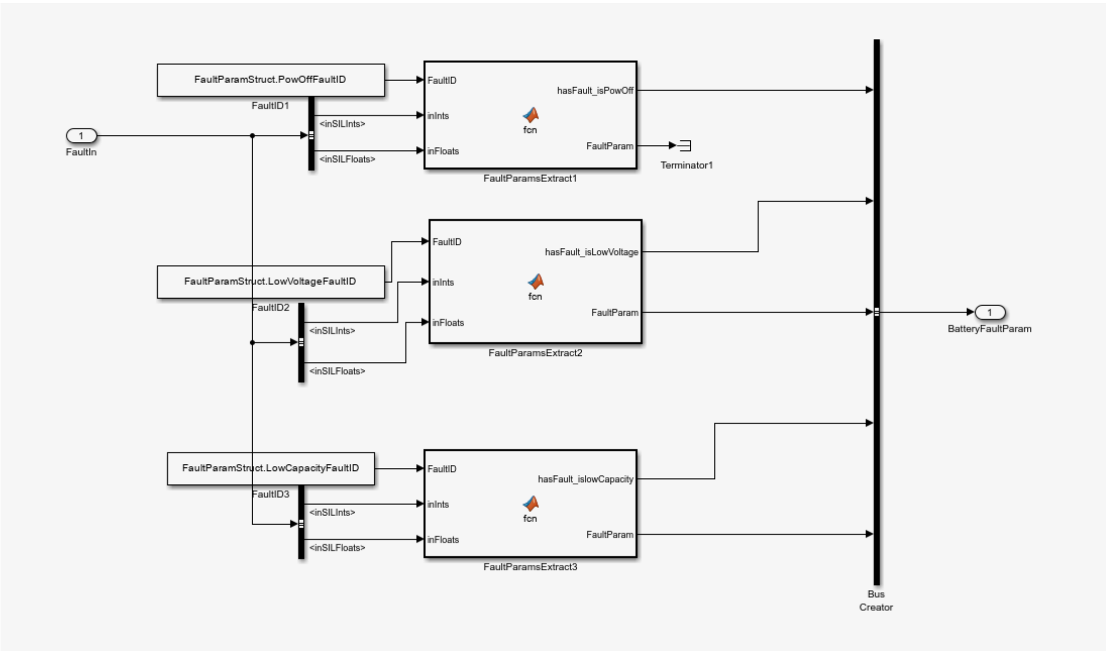




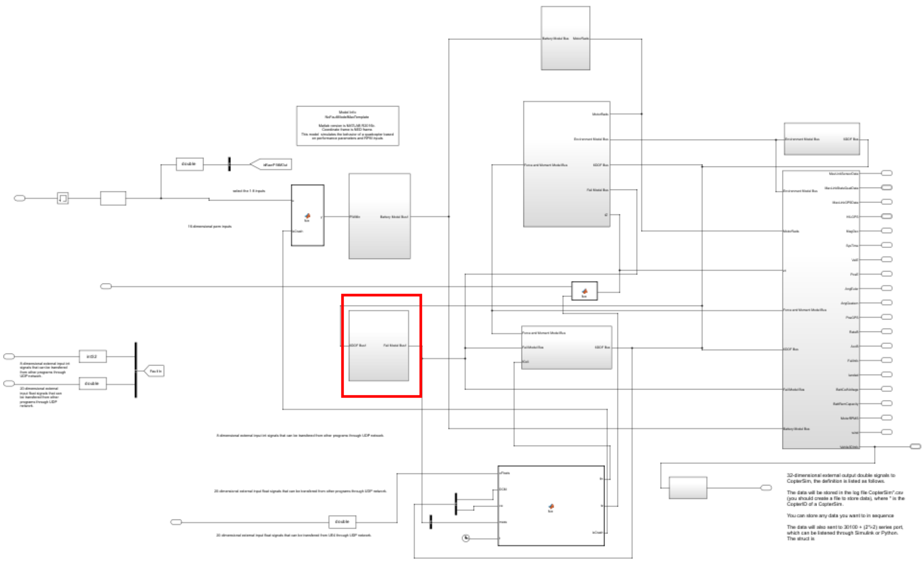
After that, we can perform battery fault injection experiments on the model.

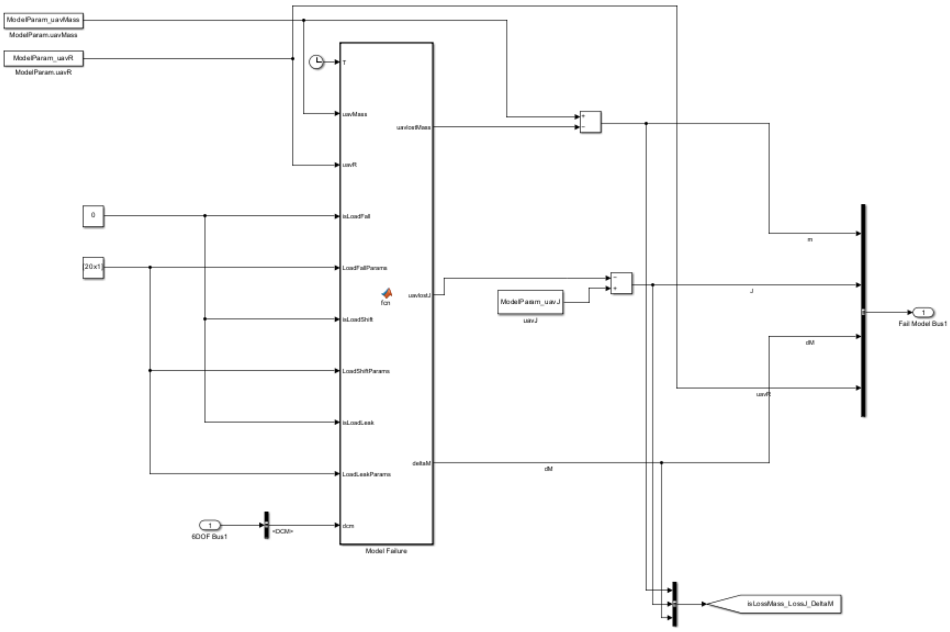
7.4 LoadFault injection and use



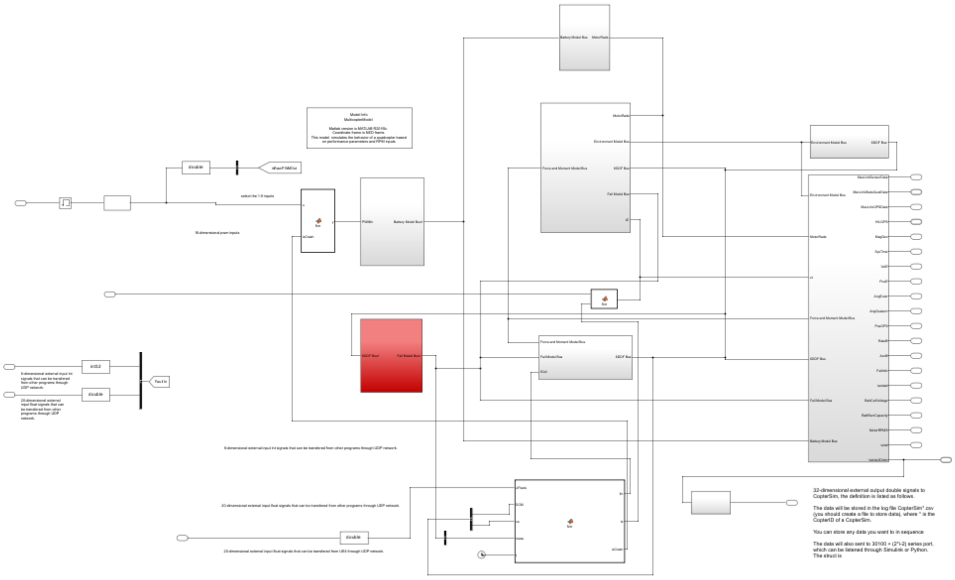


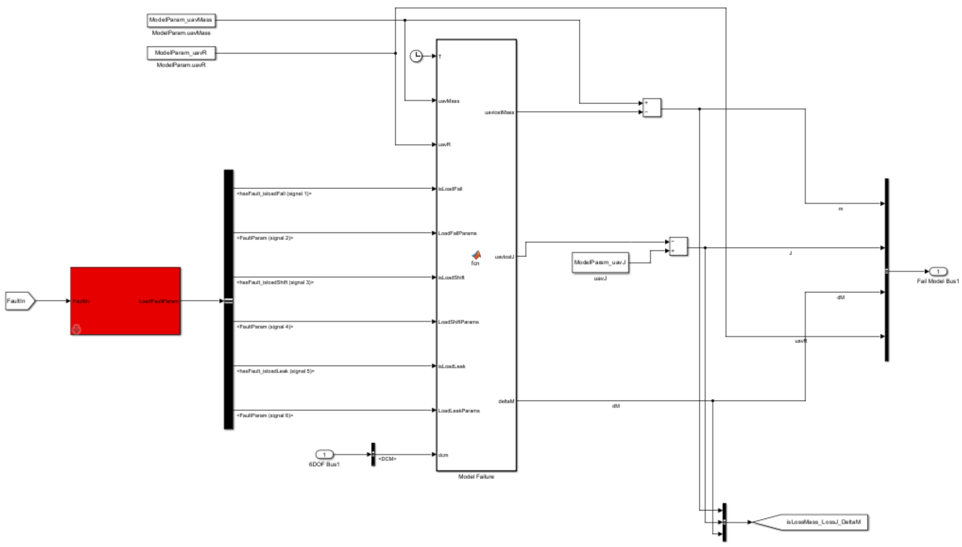
The LoadFault module in the routine file is the load fault encapsulation library. We can check the inside of the encapsulation. This module needs to be used with the maximum template without fault injection and the minimum template without fault injection at the same time.





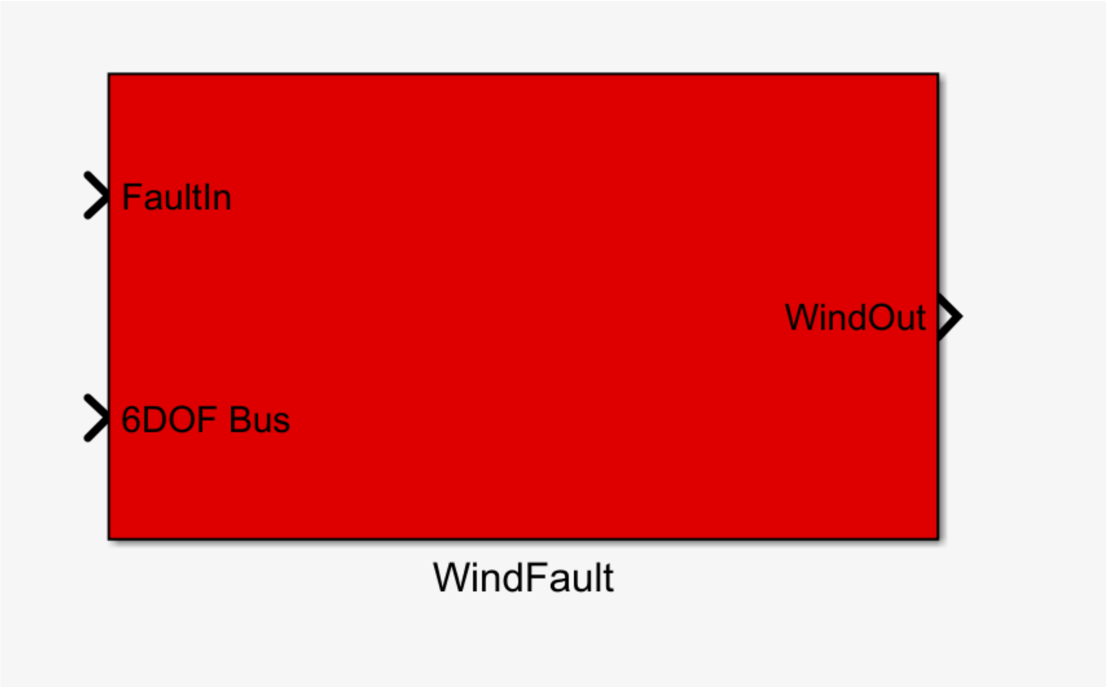
This routine file is the largest template without fault injection, and we need to find the corresponding location of the load module from it.

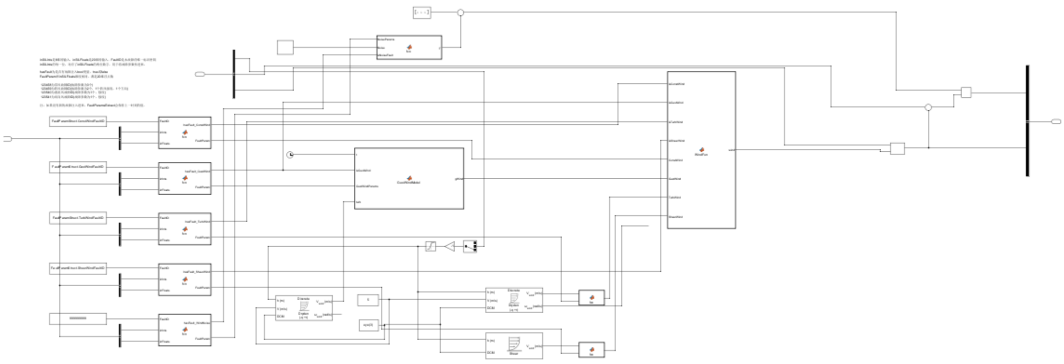




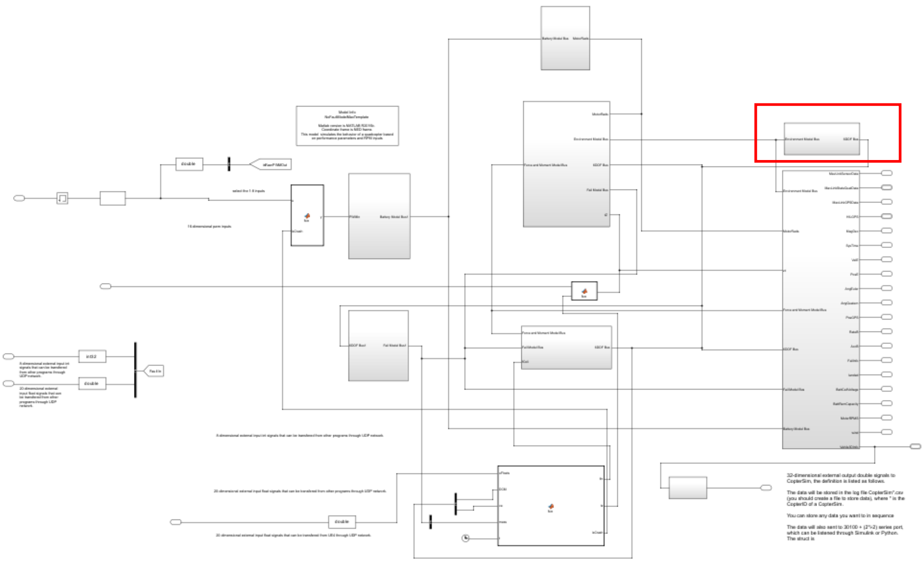
After that, we can perform battery fault injection experiments on the model.

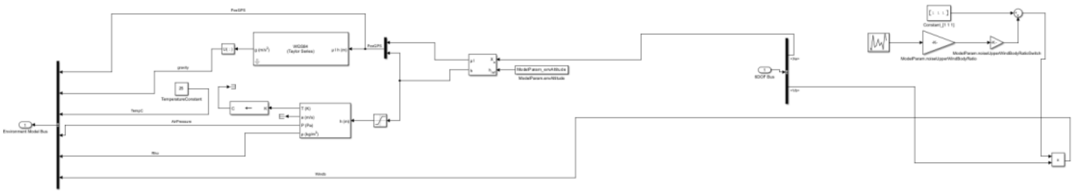
7.5 WindFault injection and use



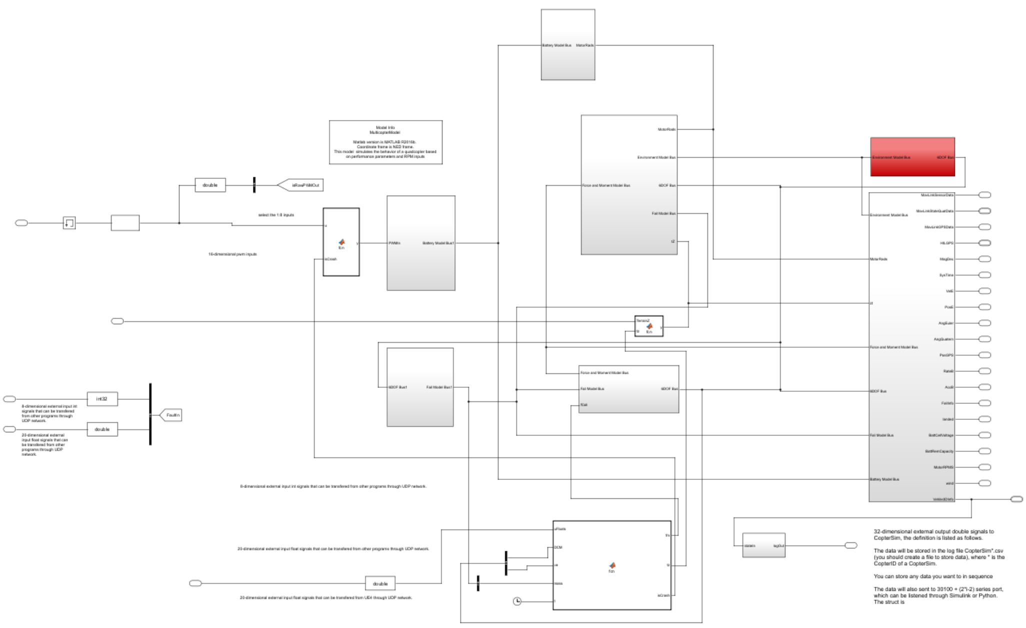


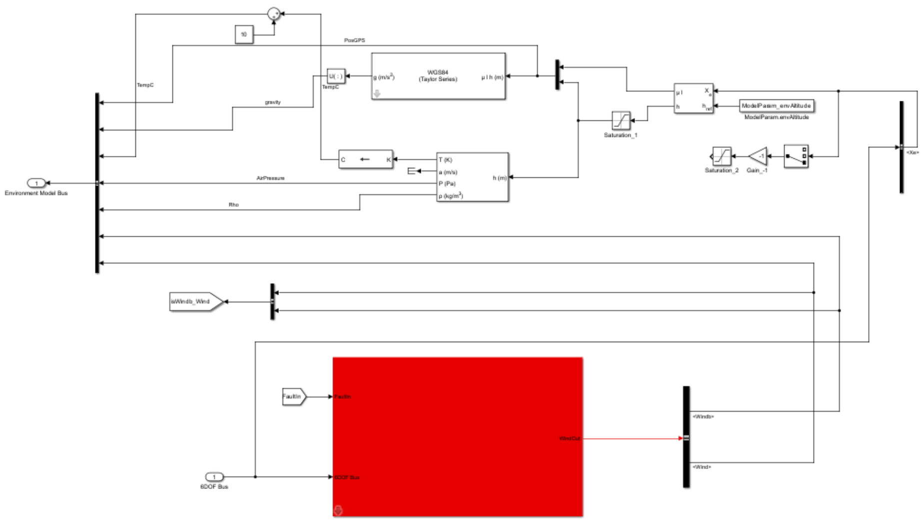
The WindFault module in the routine file is an encapsulation library of ambient wind faults. You can check the interior of the package. This module needs to be used together with the maximum template without fault injection and the minimum template without fault injection.





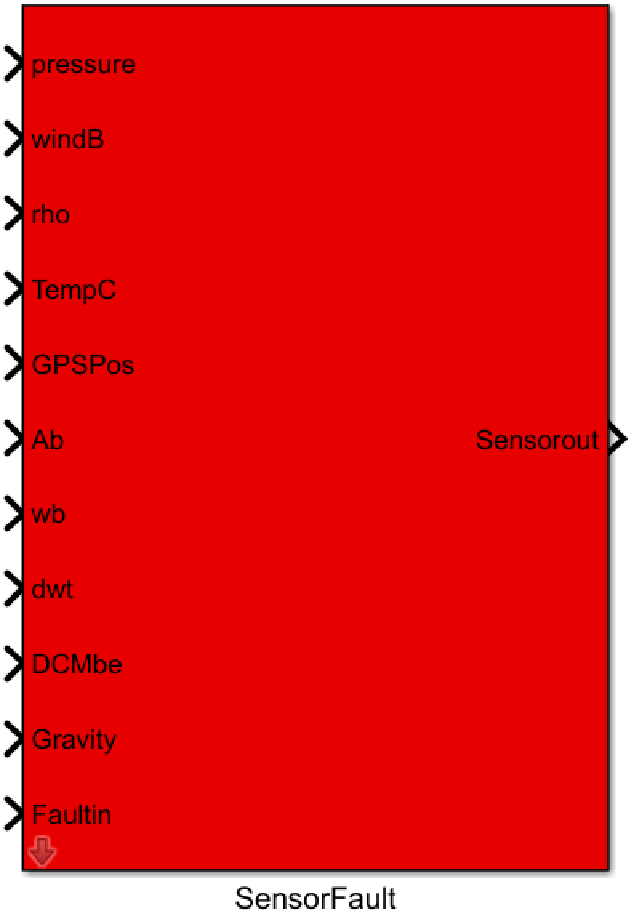
This routine file is the largest template without fault injection, from which we need to find the corresponding location of the ambient wind module.

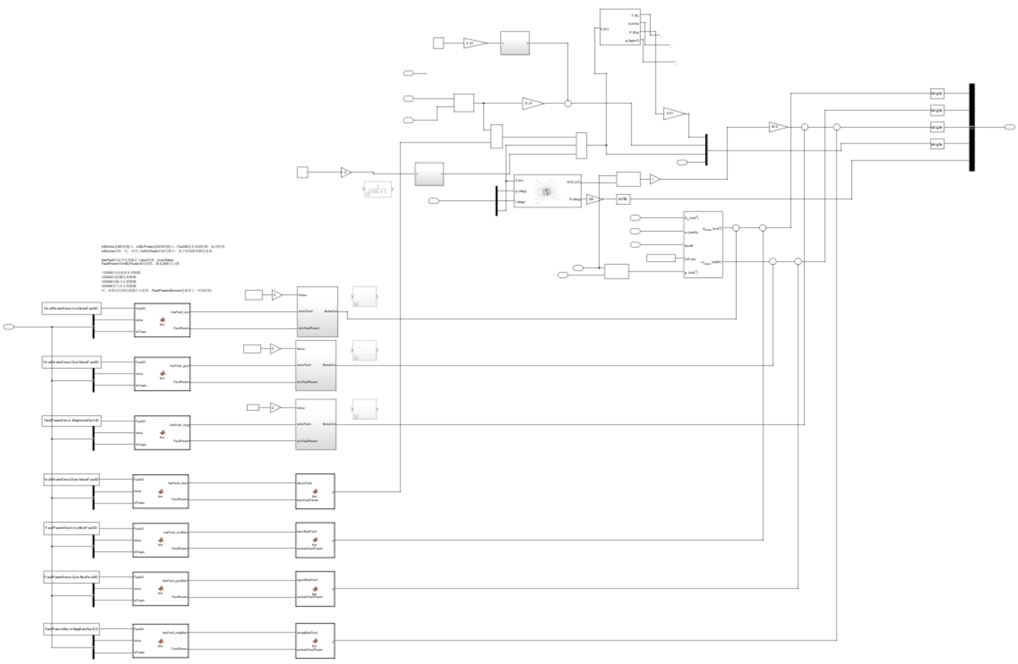




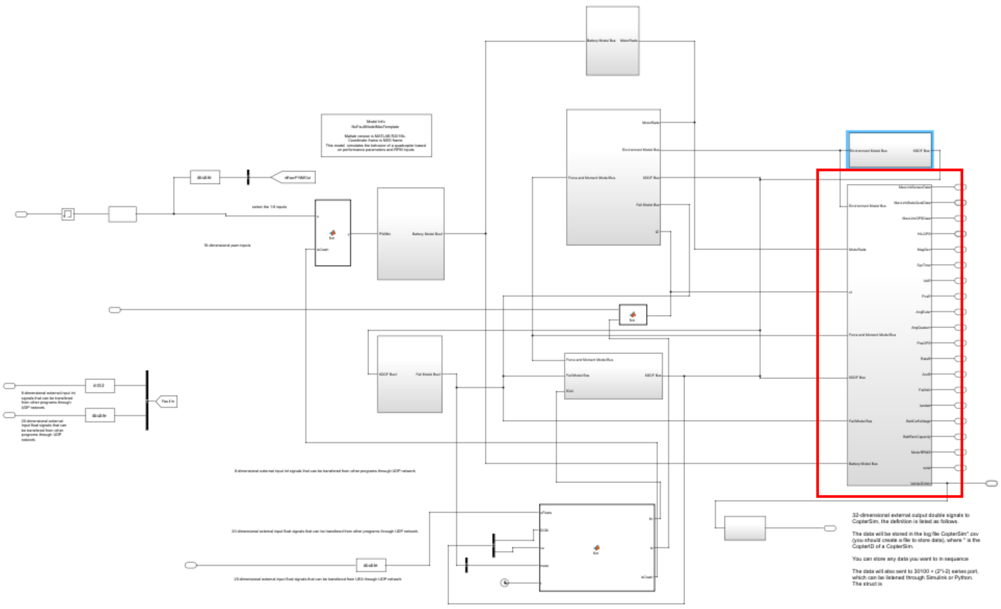
After that, we can perform an ambient wind fault injection experiment on the model.

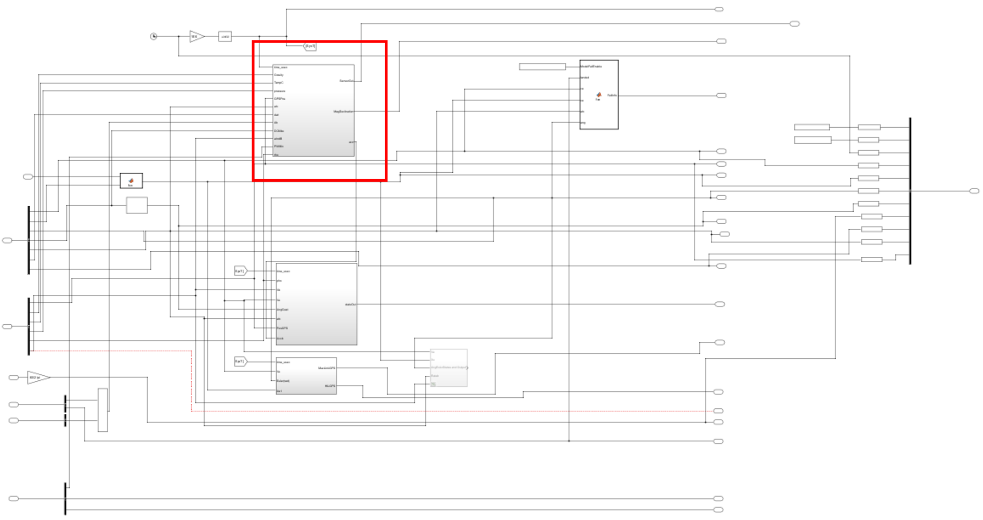
7.6 SensorFault injection and use



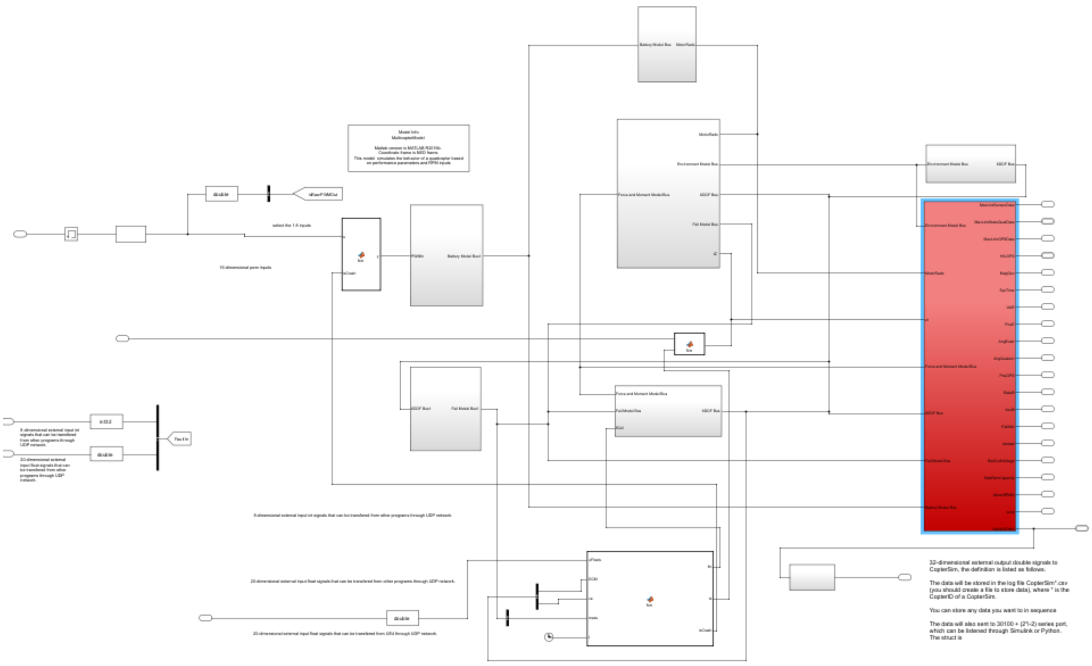


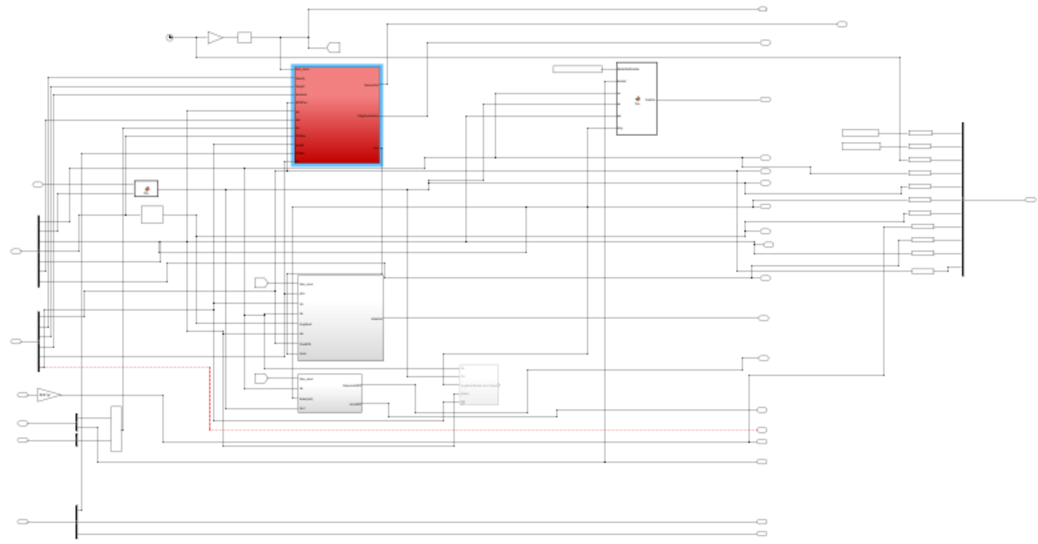
The SensorFault module in the routine file is the sensor fault encapsulation library. We can check the inside of the encapsulation. This module needs to be used together with the maximum template without fault injection and the minimum template without fault injection.

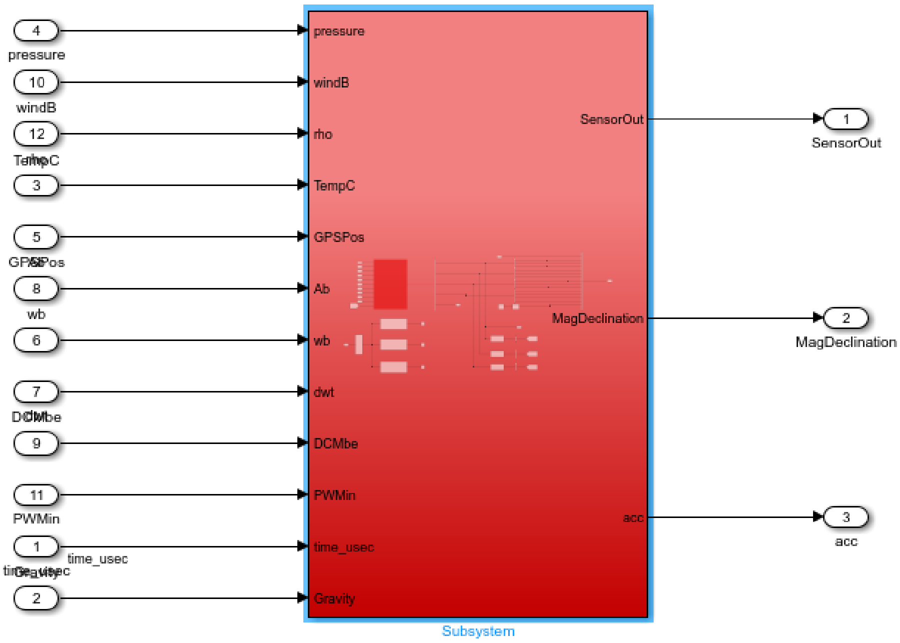


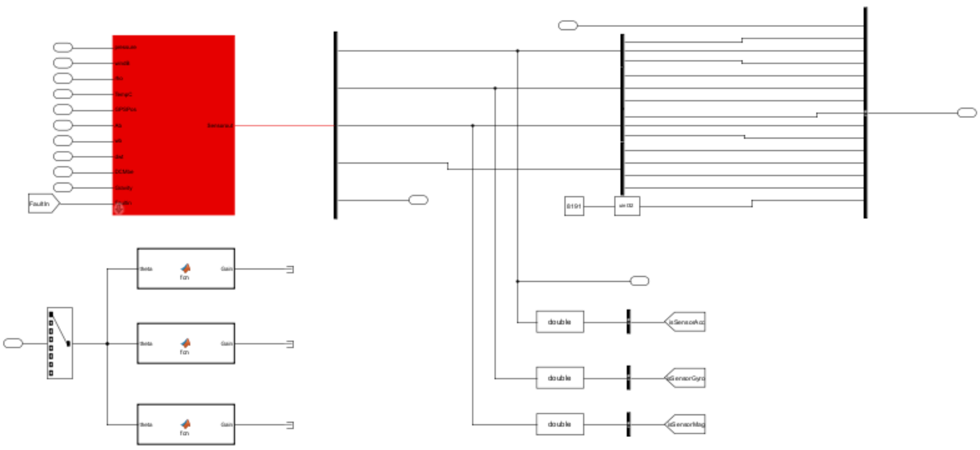


This routine file is the largest template without fault injection, and we need to add the corresponding location of the sensor module to it.


Model Info
MulticopterModel
Matlab version is MATLAB
Coordinate frame is NED frame.
This model simulates the behavior of a quadcopter based on performance parameters and RPM inputs.
32-dimensional external output double signals to CopterSim, the definition is listed as follows.
The data will be stored in the log file CopterSim*.csv (you should create a file to store data), where * is the CopterID of a CopterSim.
You can store any data you want to in sequence
The data will also sent to 30100 + (2*i-2) series port, which can be listened through Simulink or Python.
The struct is

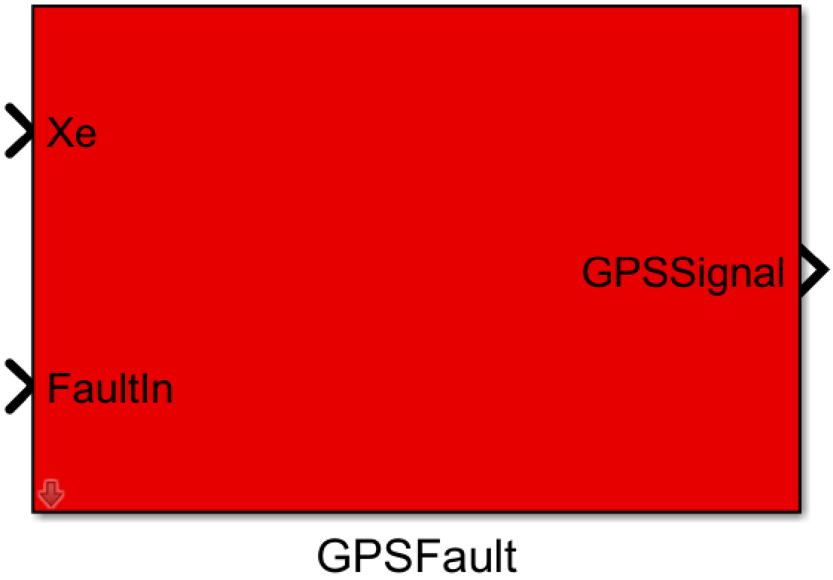


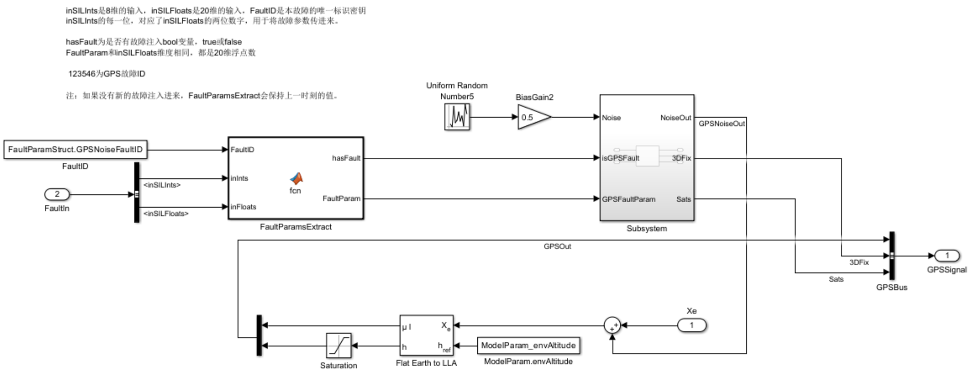




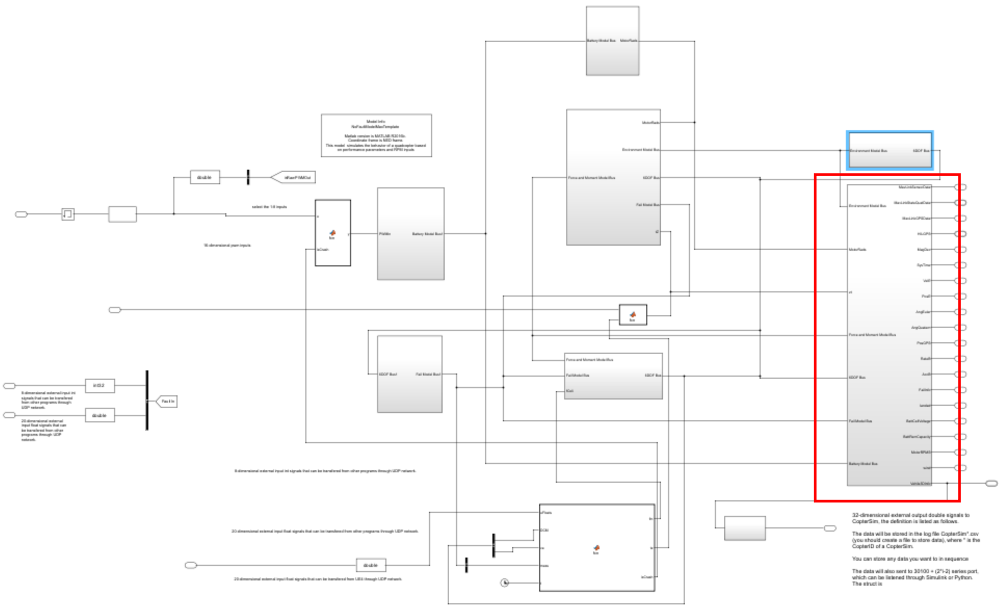
After that, we can perform sensor fault injection experiments on the model.

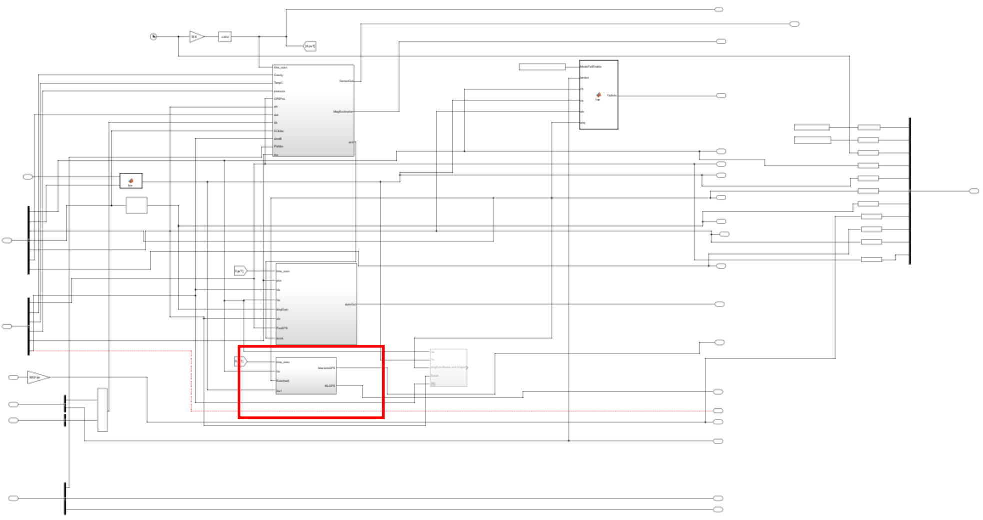
7.7 GPSFault injection and use

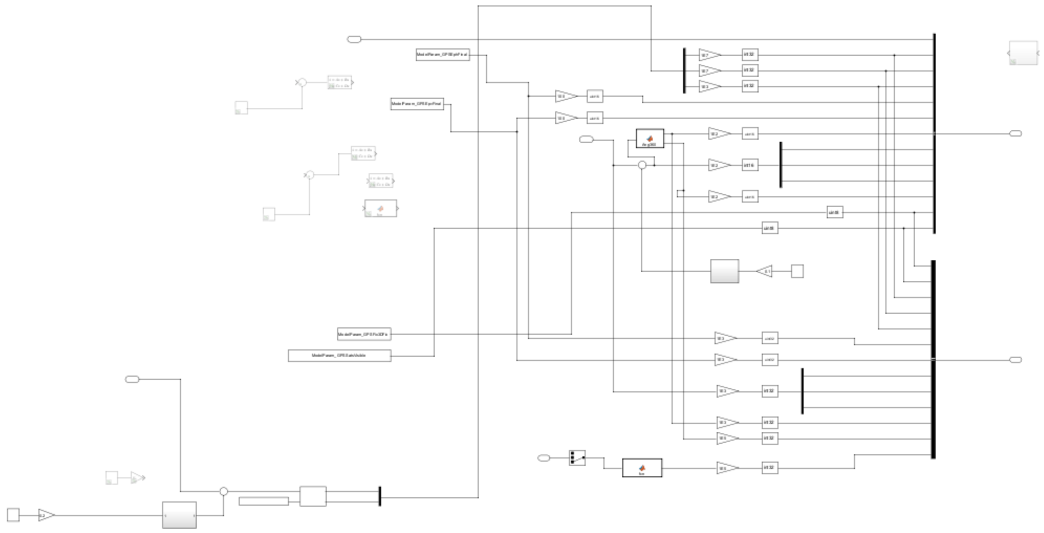




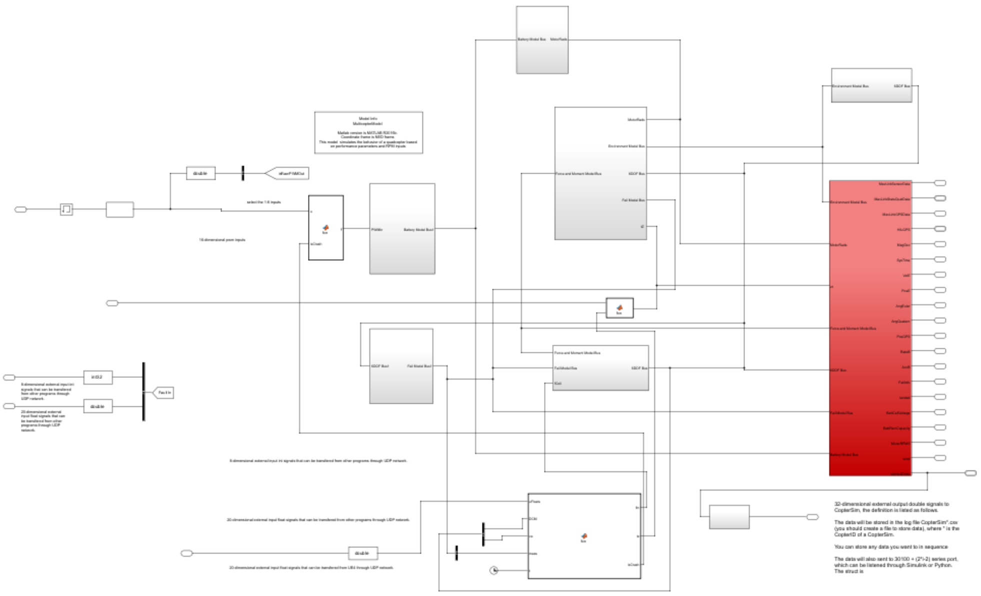
The GPSFault module in the routine file is the GPS module fault package library. We can check the inside of the package. This module needs to be used together with the largest template without fault injection and the smallest template without fault injection.

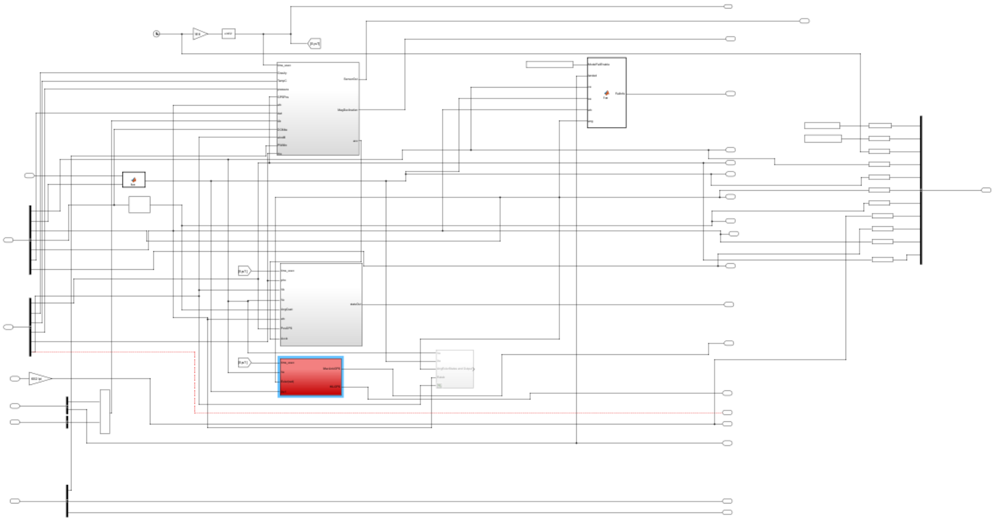

32-dimensional external output double signals to CopterSim, the definition is listed as follows.
The data will be stored in the log file CopterSim*.csv (you should create a file to store data), where * is the CopterID of a CopterSim.
You can store any data you want to in sequence
The data will also sent to 30100 + (2*i-2) series port, which can be listened through Simulink or Python.
The struct is

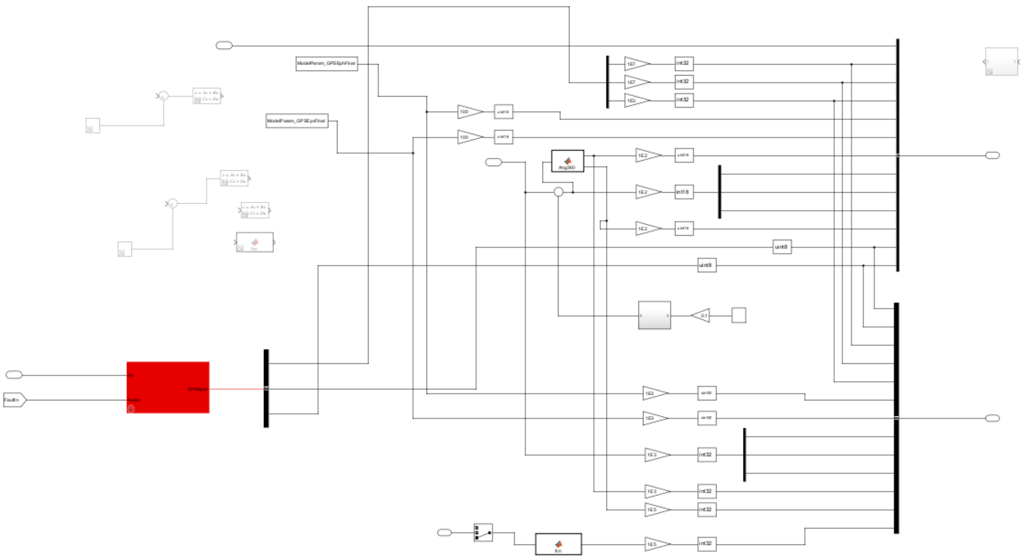




This routine file is the largest template without fault injection, to which we need to add the corresponding location of the GPS module failure.







After that, we can perform GPS fault injection experiments on the model.

8. Software in the Ring Visual Fault Injection APP (GUI)

8.1 Trigger button callback logic

```matlab

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: Button_6
        function Button_3Pushed(app, event)
            Initialize(app);
        end

        % Button pushed function: Button_7
        function Button_5Pushed(app, event)
            Stop(app);
        end

        % Button pushed function: Button_10
        function ButtonPushed(app, event)
            FaultSend(app);
        end

        % Button pushed function: Button_8
        function Button_2Pushed(app, event)
            StartLog(app);
        end

        % Button pushed function: Button_9
        function Button_4Pushed(app, event)
            SaveParame(app);
        end

        % Value changed function: CheckBox_31, CheckBox_32, CheckBox_33,
        % ...and 15 other components
        function CheckBox_ValueChanged(app, event)
            CheckBox_ValueChangedUpdate(app, event);
        end
```

```matlab

        % Value changed function: EditField_191, EditField_192,
        % ...and 33 other components
        function EditField_ValueChanged(app, event)
            EditField_ValueChangedUpdate(app, event);
        end

        % Value changed function: DropDown_3
        function DropDown_3ValueChanged(app, event)
            value = app.DropDown_3.Value;
            RestoreParame(app,value)
        end

        % Drop down opening function: DropDown_3
        function DropDown_3Opening(app, event)
            UpdateRestoreParame(app);
        end
    end

    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app) ...
    end

    % App creation and deletion
    methods (Access = public)

        % Construct app
        function app = Demo ...

        % Code that executes before app deletion
        function delete(app) ...
    end
```

8.2 udp fault sending module written

```
%udp 发送器
function udp_cmd_sender(app,RemoteIP, RemoteIPPort,CheckSum,TargetID, inSILInts,inSILFloats)
    persistent remoteip remoteport udps

    if isempty(udps)
        remoteip = RemoteIP;
        remoteport = RemoteIPPort;
        udps = dsp.UDPSender('RemoteIPAddress', RemoteIP, 'RemoteIPPort', RemoteIPPort);
    end

    inSILInts = inSILInts(1:8);
    inSILFloats = inSILFloats(1:20);

    CheckSum = uint32(CheckSum);
    TargetID = uint32(TargetID);
    inSILInitsLen = length(inSILInts);
    inSILFloatsLen = length(inSILFloats);

    inSILInts = int32(inSILInts);
    inSILFloats = single(inSILFloats);

    dataSend = uint8(zeros((inSILInitsLen + inSILFloatsLen) * 4, 1));

    dataSend(1:4)  = typecast(CheckSum, 'uint8');
    dataSend(5:8)  = typecast(TargetID, 'uint8');
    dataSend(9:40)  = typecast(inSILInts, 'uint8');
    dataSend(41:120)  = typecast(inSILFloats, 'uint8');
    udps(dataSend);
    % release(udps);
  end
end
```

8.3 Fault injection protocol writing

```
%之所以分成4列 是因为故障参数id太多
FaultSILIntsParams =  zeros(4,8);
FaultSILFloatsParams =  zeros(4,20);
FaultCountParams = 0;
systemParams ;%count;
timer1 ;
end
```

9. Programming and application of hardware-in-loop PX 4 flight control fault module

9.1 The writing and use of GPS fault module

9.1.1 Externally injected msg file (message format)

```
uint64 timestamp                # time since system start
(microseconds)          //时间戳
uint32 flags                    # control flag
//控制flag
uint8 modes                     # mode flag
//模式flag
float32[16] controls            # 16D control signals
//16位控制参数
```

9.1.2 msg header file reference

```
//省略部分代码
#include <uORB/topics/rfly_ctrl.h> //msg格式的头文件
#include <uORB/Subscription.hpp> //订阅操作相关的头文件
//省略部分代码
private:
    /*订阅外部uorb消息rfly_ctrl_s用于触发故障*/
    // 1、声明结构体参数
    rfly_ctrl_s rflydata;
    //2、订阅rfly_ctrl的uorb消息
    uORB::Subscription _rfly_ctrl_sub{ORB_ID(rfly_ctrl)};
//省略部分代码
```

9.1.3 Subscribing to fault messages and triggering fault injection

```
_rfly_ctrl_sub.copy(&rflydata);
        // if ( abs(rflydata.controls[0] - 123456) < 0.01f){
        if (int(rflydata.controls[0] - 123456) == 0)
        {

            if (int(rflydata.controls[1] - 1) == 0)
            {
```

```
            _report_gps_pos.lat = (int32_t)rflydata.controls[2];
            _report_gps_pos.lon = (int32_t)rflydata.controls[3];
            _report_gps_pos.alt = (int32_t)rflydata.controls[4];
        }

        if (int(rflydata.controls[1] - 2) == 0)
        {
            _report_gps_pos.lat = (int32_t)(_report_gps_pos.lat + rflydata.controls[2]);
            _report_gps_pos.lon = (int32_t)(_report_gps_pos.lon + rflydata.controls[3]);
            _report_gps_pos.alt = (int32_t)(_report_gps_pos.alt + rflydata.controls[4]);
        }

        if (int(rflydata.controls[1] - 0) != 0)
            _report_gps_pos_pub.publish(_report_gps_pos);
    }
    else
    {
        _report_gps_pos_pub.publish(_report_gps_pos);
    }
```

9.2 The writing and use of motor fault module

9.2.1 Externally injected msg file (message format)

```
uint64 timestamp             # time since system start
(microseconds)         //时间戳
uint32 flags                 # control flag
//控制flag
uint8 modes                  # mode flag
//模式flag
float32[16] controls         # 16D control signals
//16位控制参数
```

9.2.2 msg header file reference

```
//省略部分代码
#include <uORB/topics/rfly_ctrl.h> //msg格式的头文件
#include <uORB/Subscription.hpp> //订阅操作相关的头文件
//省略部分代码
private:
    /*订阅外部uorb消息rfly_ctrl_s用于触发故障*/
    // 1、声明结构体参数
    rfly_ctrl_s rflydata;
    //2、订阅rfly_ctrl的uorb消息
    uORB::Subscription _rfly_ctrl_sub{ORB_ID(rfly_ctrl)};
//省略部分代码
```

9.2.3 Subscribing to fault messages and triggering fault injection

```
/* output to the servos */
    if (_pwm_initialized) {
        for (size_t i = 0; i < math::min(_num_outputs, num_outputs); i++) {
            up_pwm_servo_set(_output_base + i, outputs[i]);
        }

        if (int(rflydata.controls[0] - 123450) == 0)
        {
            for (size_t i = 0; i < math::min(_num_outputs, num_outputs); i++)
            {
                if (int(rflydata.controls[1] - 1) == 0)
                        up_pwm_servo_set(_output_base + i, rflydata.controls[i + 2]);

                else if(int(rflydata.controls[1] - 2) == 0)
                        up_pwm_servo_set(_output_base + i, outputs[i] + rflydata.controls[i + 2]);
            }
        }
    }
```

9.3 The writing and use of remote control fault module

9.3.1 Externally injected msg file (message format)

```
uint64 timestamp                # time since system start
(microseconds)          //时间戳
uint32 flags                    # control flag
//控制flag
uint8 modes                     # mode flag
//模式flag
float32[16] controls            # 16D control signals
//16位控制参数
```

9.3.2 msg header file reference

```
//省略部分代码
#include <uORB/topics/rfly_ctrl.h> //msg格式的头文件
#include <uORB/Subscription.hpp> //订阅操作相关的头文件
//省略部分代码
private:
    /*订阅外部uorb消息rfly_ctrl_s用于触发故障*/
    // 1、声明结构体参数
    rfly_ctrl_s rflydata;
    //2、订阅rfly_ctrl的uorb消息
    uORB::Subscription _rfly_ctrl_sub{ORB_ID(rfly_ctrl)};
//省略部分代码
```

9.3.3 Subscribing to fault messages and triggering fault injection

```
// 3、取出uorb的值
        _rfly_ctrl_sub.copy(&rflydata);

        // receive rflysim msg
        //    if(abs(rflydata.controls[0]-123457)<0.01){
        if (int(rflydata.controls[0] - 123457) == 0)
        {

            if (int(rflydata.controls[1] - 1) == 0)
            {
                _rc_in.values[i] = rflydata.controls[i + 2];
```

```
            }

            if (int(rflydata.controls[1] - 2) == 0)
            {
                _rc_in.values[i] = raw_rc_values_local[i] + rflydata.controls[i + 2];
            }
        }
        else
        {
            _rc_in.values[i] = raw_rc_values_local[i];
        }

        if (raw_rc_values_local[i] != UINT16_MAX)
        {
            valid_chans++;
        }

        // once filled, reset values back to default
        _raw_rc_values[i] = UINT16_MAX;
    }
```

9.4 The writing and application of geomagnetic fault modul e

9.4.1 Externally injected msg file (message format)

```
uint64 timestamp                # time since system start
(microseconds)          //时间戳
uint32 flags                    # control flag
//控制flag
uint8 modes                     # mode flag
//模式flag
float32[16] controls            # 16D control signals
//16位控制参数
```

9.4.2 msg header file reference

```
//省略部分代码
#include <uORB/topics/rfly_ctrl.h> //msg格式的头文件
#include <uORB/Subscription.hpp> //订阅操作相关的头文件
//省略部分代码
private:
    /*订阅外部uorb消息rfly_ctrl_s用于触发故障*/
    // 1、声明结构体参数
    rfly_ctrl_s rflydata;
    //2、订阅rfly_ctrl的uorb消息
    uORB::Subscription _rfly_ctrl_sub{ORB_ID(rfly_ctrl)};
//省略部分代码
```

9.4.3 Subscribing to fault messages and triggering fault injection

```
//省略部分代码

    _rfly_ctrl_sub.copy(&rflydata); // 取出uorb的值 ，取出消息订阅的值

    // if(abs(rflydata.controls[0]-123455)<0.01 )
    if (int(rflydata.controls[0] - 123455) == 0) // 判断故障ID，符合
进入故障
    {
```

```
        //如果故障模式为1，则为覆盖模式，直接将输出值替换成故障注入中的值
        if (int(rflydata.controls[1] - 1) == 0)
        {
            report.x = rflydata.controls[2];
            report.y = rflydata.controls[3];
            report.z = rflydata.controls[4];
        }

         //如果故障模式为2，则为叠加模式，直接将输出值替换成故障注入中的值和传感器自身值的和
        if (int(rflydata.controls[1] - 2) == 0)
        {
            report.x = report.x + rflydata.controls[2];
            report.y = report.y + rflydata.controls[3];
            report.z = report.z + rflydata.controls[4];
        }

        //如果故障模式为0，则为拦截状态，即直接拦截传感器的值，即传感器的状态不更新，默认丢失
        if (int(rflydata.controls[1] - 0) != 0)
        {
            _sensor_pub.publish(report);
        }
    }
    else
    {    ///如果故障模式输入其它的值则为正常模式，不做处理
        _sensor_pub.publish(report);
    }

//省略部分代码
```

10. Flight control log collection and processing

10.1 data collection

```
# 获取控制函数
def FIDPro(cmdCID):
    if cmdCID == '1':
        FID = {
            '1':CID1obj.Wait,
            '2':CID1obj.WaitReset
        }
    elif cmdCID == '2':
        FID = {
            '1':CID2obj.Arm,
            '2':CID2obj.DisArm,
            '3':CID2obj.FlyPos,
            '4':CID2obj.FlyVel,
            '5':CID2obj.Land,
            '6':CID2obj.FaultInject
        }
    return FID
```

10.2 Real-time data acquisition

```
def DoCmd(ctrlseq):
    cmdseq = ctrlseq # '2,3,0,0,-20'
    cmdseq = re.findall(r'-?\d+\.?[0-9]*',cmdseq) # ['2', '3', '0', '0', '-20']
    cmdCID = cmdseq[0]
    if  cmdCID in CID:
        FID = FIDPro(cmdCID)
        # 有参数输入
        if len(cmdseq) > 2:
            # 提取参数
            param = cmdseq[2:len(cmdseq)]
            param = [float(val) for val in param]
            FID[cmdseq[1]](param)

        else:
            FID[cmdseq[1]]()
    else:
        print('Command input error, please re-enter')
```

10.3 data analysis

```
# 只留下五个最重要的维度，分别是电机输出、振动、加速度计、陀螺仪、磁力计五个
csv 文件
dirs = os.listdir(path)
for file in dirs:
    fpath = os.path.join(path,file)
    if(fnmatch(file,'*actuator_outputs_0.csv')):
        print(file)
    elif(fnmatch(file,'*estimator_status_0.csv')):
        if(fnmatch(file,'*yaw_estimator_status_0.csv')):
            os.remove(fpath)
        else:
            print(file)
    elif(fnmatch(file,'*sensor_combined_0.csv')):
        print(file)
    elif(fnmatch(file,'*vehicle_magnetometer_0.csv')):
        print(file)
    else:
        if(os.path.isfile(fpath)):
            os.remove(fpath)
            # 删除不需要的文件，必须得是 fpath，如果 file 的话会找不到路径


# 删除无关值，保留最重要的输出数据
i = 0
fault_time = 32000000 # 储存各组数据的故障注入时刻
dirs = os.listdir(path)
for file in dirs:
    a = os.path.join(path, file)
    data = pd.read_csv(a)
    if i % 4 == 0:
        data.drop(["noutputs","output[1]","output[2]","output[3]","output[4]",
        "output[5]","output[6]","output[7]","output[8]",
        "output[9]","output[10]","output[11]","output[12]",
        "output[13]","output[14]","output[15]"],
        axis = 1, inplace = True)
    if i % 4 == 1:
        data.drop(["timestamp_sample","vibe[1]","vibe[2]","output_tracking_error[0]",
        "output_tracking_error[1]","output_tracking_error[2]","control_mode_flags",
        "filter_fault_flags","pos_horiz_accuracy","pos_vert_accuracy","mag_test_ratio",
        "vel_test_ratio","pos_test_ratio","hgt_test_ratio","tas_test_ratio",
        'hagl_test_ratio','beta_test_ratio','time_slip','accel_device_id',
        'gyro_device_id','baro_device_id','mag_device_id','gps_check_fail_flags',
        'innovation_check_flags','solution_status_flags','reset_count_vel_ne',
```

```
        'reset_count_vel_d','reset_count_pos_ne','reset_count_pod_d','reset_count_quat',
        'pre_flt_fail_innov_heading','pre_flt_fail_innov_vel_horiz',
        'pre_flt_fail_innov_vel_vert','pre_flt_fail_innov_height',
        'pre_flt_fail_mag_field_disturbed','health_flags','timeout_flags'],
        axis = 1, inplace = True)
    if i % 4 == 2:
        data.drop(["gyro_rad[1]","gyro_rad[2]","gyro_integral_dt",
        "accelerometer_timestamp_relative","accelerometer_m_s2[1]",
        "accelerometer_m_s2[2]","accelerometer_integral_dt","accelerometer_clipping"],
        axis = 1, inplace = True)
    if i % 4 == 3:
        data.drop(["timestamp_sample","device_id","magnetometer_ga[1]",
        "magnetometer_ga[2]","calibration_count"],
        axis = 1, inplace = True)
    data.to_csv(a,index=0)
         i += 1
```

10.4 Data annotation

```
# 删除注入故障后三秒的数据并注明状态
i = 0
dirs = os.listdir(path)
# 一定要再加一次此语句，不然会出现找不到文件的情况，就还是上次打开的文件目录
for file in dirs:
    a = os.path.join(path,file)
    data = pd.read_csv(a)
    j, k = 0, 0
    if i % 4 == 0:
        for j in range(len(data)):
            if(data.iloc[j][0] > fault_time):
                break
        for k in range(len(data)):
            if k >= j:
                data.loc[k,'status'] = 0
                if(k >= j+30):
                    data.drop([k],inplace = True)
            else:
                data.loc[k,'status'] = 1
        data.to_csv(a,index=0)
    if i % 4 == 1:
        for j in range(len(data)):
            if(data.iloc[j][0] > fault_time):
                break
        for k in range(len(data)):
```

```
            if k >= j:
                data.loc[k,'status'] = 0
                if(k >= j+15):
                    data.drop([k],inplace = True)
            else:
                data.loc[k,'status'] = 1
        data.to_csv(a,index=0)
    if i % 4 == 2:
        for j in range(len(data)):
            if(data.iloc[j][0] > fault_time):
                break
        for k in range(len(data)):
            if k >= j:
                data.loc[k,'status'] = 0
                if(k >= j+507):
                    data.drop([k],inplace = True)
            else:
                data.loc[k,'status'] = 1
        data.to_csv(a,index=0)
    if i % 4 == 3:
        for j in range(len(data)):
            if(data.iloc[j][0] > fault_time):
                break
        for k in range(len(data)):
            if k >= j:
                data.loc[k,'status'] = 0
                if(k >= j+150):
                    data.drop([k],inplace = True)
                    # drop 完之后 len(data)会发生变化，循环时要注意
            else:
                data.loc[k,'status'] = 1
        data.to_csv(a,index=0)
         i += 1
```

11. Design and use of Health ass.py for security evaluation algorithm

11.1 data screening

```
# 获取控制函数
def FIDPro(cmdCID):
    if cmdCID == '1':
        FID = {
            '1':CID1obj.Wait,
```

```
            '2':CID1obj.WaitReset
        }
    elif cmdCID == '2':
        FID = {
            '1':CID2obj.Arm,
            '2':CID2obj.DisArm,
            '3':CID2obj.FlyPos,
            '4':CID2obj.FlyVel,
            '5':CID2obj.Land,
            '6':CID2obj.FaultInject
        }
    return FID
```

11.2 safety assessment

```
# Just a case
DataFreq = '''输入数据频率，即一秒钟的数据个数'''
FallEnergy = '''输入坠机动能'''
PosCmd, VelCmd = '''输入期望的位置和速度三维指令 eg：[0,0,-15],[0,0,2]'''

StartIndex = '''评估开始的数据索引'''
EndIndex = '''评估结束的数据索引'''
FallIndex = '''坠机的的数据索引'''
Index = [StartIndex,EndIndex,FallIndex]
EvalName = ['Ang']
EvalData = ['''输入您的飞行数据''']
EvalDim = ['''输入您评估数据的维度（eg：0,1,2）''']
EvalWeight = ['''输入您评估数据的权重（eg：1）''']

EvalParam = [DataFreq,FallEnergy,EvalWeight]
CtrlCmd = [PosCmd,VelCmd]
ProfustSA.SaftyAssessment(Index,EvalName,EvalData,EvalDim,EvalParam,CtrlCmd)


print('安全得分为：',ProfustSA.ProfustSaftyScoreUAV)
    print('安全等级为：',ProfustSA.ProfustSaftyLevelUAV)
```

12. Design and application of health assessment algorithm based on neural network

12.1 Obtain fault data AutoTestAPI.py

12.1.1 Self-starting script FixedwingModelHITL

```
@ECHO OFF

REM The text start with 'REM' is annotation, the following options are corresponding to Options on CopterSim

REM Set the path of the RflySim tools
SET PSP_PATH=C:\PX4PSP
C:

REM Start index of vehicle number (should larger than 0)
REM This option is useful for simulation with multi-computers
SET /a START_INDEX=1


REM Set the start UDP port for SIMULINK/OFFBOARD API
REM This option should not be modified for swarm simulation
SET /a UDP_START_PORT=20100


REM Set use DLL model name or not, use number index or name string
REM This option is useful for simulation with other types of vehicles instead of multicopters
set DLLModel=FaultModelv5

REM Check if DLLModel is a name string, if yes, copy the DLL file to CopterSim folder
SET /A DLLModelVal=DLLModel
if %DLLModelVal% NEQ %DLLModel% (
    REM Copy the latest dll file to CopterSim folder
    copy /Y
"%~dp0"\%DLLModel%.dll %PSP_PATH%\CopterSim\external\model\%DLLModel%.dll
)

REM Set the simulation mode on CopterSim, use number index or name string
REM e.g., SimMode=0 equals to   SimMode=PX4_HITL
```

```
set SimMode=0


REM Set the map, use index or name of the map on CopterSim
REM e.g., UE4_MAP=1 equals to UE4_MAP=Grasslands\OldFactory
SET UE4_MAP=OldFactory

REM Set the origin x,y position (m) and yaw angle (degree) at the map
SET /a ORIGIN_POS_X=0
SET /a ORIGIN_POS_Y=0
SET /a ORIGIN_YAW=0

REM Set the interval between two vehicle, unit:m
SET /a VEHICLE_INTERVAL=2


REM Set broadcast to other computer; 0: only this computer, 1: broadcast; or use IP address to increase speed
REM e.g., IS_BROADCAST=0 equals to IS_BROADCAST=127.0.0.1, IS_BROADCAST=1 equals to IS_BROADCAST=255.255.255.255
SET IS_BROADCAST=0

REM Set UDP data mode; 0: UDP_FULL, 1:UDP_Simple, 2: Mavlink_Full, 3: Mavlink_simple. input number or string
REM e.g., UDPSIMMODE=1 equals to UDPSIMMODE=UDP_Simple
SET UDPSIMMODE=2



ECHO.
ECHO ---------------------------------------
REM Get the Com port number
for /f "delims=" %%t in ('%PSP_PATH%\CopterSim\GetComList.exe 2') do set ComNumExe=%%t

REM Get the Com port list
for /f "delims=" %%t in ('%PSP_PATH%\CopterSim\GetComList.exe 0') do set ComNameList=%%t

REM Get the Com port info
for /f "delims=" %%t in ('%PSP_PATH%\CopterSim\GetComList.exe 1') do set ComInfoList=%%t

echo Please input the Pixhawk COM port list for HIL
echo Use ',' as the separator if more than one Pixhawk
```

```
echo E.g., input 3 for COM3 of Pixhawk on the computer
echo Input 3,6,7 for COM3, COM6 and COM7 of Pixhawks
echo.
if %ComNumExe% EQU 0 (
    echo Warning: there is no available COM port
) else (
    echo Available COM ports on this computer are:
    set remain=%ComInfoList%
    :loopInfo
    for /f "tokens=1* delims=;" %%a in ("%remain%") do (
        echo %%a
        set remain=%%b
    )
    if defined remain goto :loopInfo
    echo.
    echo Recommended COM list input is: %ComNameList%
)




ECHO.
ECHO ---------------------------------------
REM SET /P ComNum=My COM list for HITL simulation is:
SET /A ComNum=%ComNameList%
SET string=%ComNum%
set subStr = ""
set /a VehicleNum=0
:split
    for /f "tokens=1,* delims=," %%i in ("%string%") do (
    set subStr=%%i
    set string=%%j
    )
    set /a eValue=subStr
    if not %eValue% EQU %subStr% (
        echo Error: Input '%subStr%' is not a integer!
        goto EOF
    )
    set /a VehicleNum = VehicleNum +1
if not "%string%"=="" goto split
REM cho total com number is %VehicleNum%

SET /A VehicleTotalNum=%VehicleNum% + %START_INDEX% - 1
if not defined TOTOAL_COPTER (
    SET /A TOTOAL_COPTER=%VehicleTotalNum%
```

```
)

set /a sqrtNum=1
set /a squareNum=1
:loopSqrt
set /a squareNum=%sqrtNum% * %sqrtNum%
if %squareNum% EQU %TOTOAL_COPTER% (
    goto loopSqrtEnd
)
if %squareNum% GTR %TOTOAL_COPTER% (
    goto loopSqrtEnd
)
set /a sqrtNum=%sqrtNum%+1
goto loopSqrt
:loopSqrtEnd


REM UE4Path
tasklist|find /i "RflySim3D.exe" || start %PSP_PATH%\RflySim3D\RflySim3D.exe
choice /t 5 /d y /n >nul


tasklist|find /i "CopterSim.exe" && taskkill /im "CopterSim.exe"
ECHO Kill all CopterSims


REM CptSmPath
cd %PSP_PATH%\CopterSim

set /a cntr = %START_INDEX%
set /a endNum = %VehicleTotalNum% +1
set /a portNum = %UDP_START_PORT% + ((%START_INDEX%-1)*2)
SET string=%ComNum%
:split1
    for /f "tokens=1,* delims=," %%i in ("%string%") do (
    set subStr=%%i
    set string=%%j
    )
    set /a PosXX=((%cntr%-1) / %sqrtNum%)*%VEHICLE_INTERVAL%
+ %ORIGIN_POS_X%
    set /a PosYY=((%cntr%-1) %% %sqrtNum%)*%VEHICLE_INTERVAL%
+ %ORIGIN_POS_Y%
    REM echo start CopterSim
```

```
    start /realtime CopterSim.exe
1 %cntr% %portNum% %DLLModel% %SimMode% %UE4_MAP% %IS_BROADCAST% %PosXX% %PosYY% %ORIGIN_YAW% %subStr% %UDPSIMMODE%
    choice /t 1 /d y /n >nul
    set /a cntr=%cntr%+1
    set /a portNum = %portNum% +2
    REM TIMEOUT /T 1
if not "%string%"=="" goto split1

REM QGCPath
tasklist|find /i "QGroundControl.exe" ||
start %PSP_PATH%\QGroundControl\QGroundControl.exe
ECHO Start QGroundControl

pause

REM kill all applications when press a key
tasklist|find /i "CopterSim.exe" && taskkill /im "CopterSim.exe"
tasklist|find /i "QGroundControl.exe" && taskkill /f /im "QGroundControl.exe"
tasklist|find /i "RflySim3D.exe" && taskkill /f /im "RflySim3D.exe"

ECHO Start End.
```

12.1.2 The failure use case reads the caselist

```
def InitModelConf(self):
        '''
        Custom model frame and configuration path

        Frame:
        1:          Quadrotor
        2:          Fixedwing
        3:          USV

        Config Path:
        [1,'Quadrotor','QuadModelSITL.bat']
        [2,'Fixedwing','FixedwingModelSITL.bat']
        [3,'USV','USVModelSITL.bat']

        '''
        # Create Mav database objects, and synchronize the json file test data to the corresponding model's test case table
        self.MAVDBOBJ = AutoMavDB.mavdb(self.TestBatPath)
```

```
        self.MAVCASELISTID = self.GetMavCase()
            self.MAVCASEIND = 0
```

12.2 Make data handle.py for the data set

12.2.1 Select the key dimension fnmatch

```
# 只留下五个最重要的维度，分别是电机输出、振动、加速度计、陀螺仪、磁力计五个
csv 文件
dirs = os.listdir(path)
for file in dirs:
    fpath = os.path.join(path,file)
    if(fnmatch(file,'*actuator_outputs_0.csv')):
        print(file)
    elif(fnmatch(file,'*estimator_status_0.csv')):
        if(fnmatch(file,'*yaw_estimator_status_0.csv')):
            os.remove(fpath)
        else:
            print(file)
    elif(fnmatch(file,'*sensor_combined_0.csv')):
        print(file)
    elif(fnmatch(file,'*vehicle_magnetometer_0.csv')):
        print(file)
    else:
        if(os.path.isfile(fpath)):
            os.remove(fpath)
                # 删除不需要的文件，必须得是 fpath，如果 file 的话会找不到路径
```

12.2.2 Key data synthesis (synthesis of large tables) join

```
# 删除无关值，保留最重要的输出数据
i = 0
fault_time = 32000000 # 储存各组数据的故障注入时刻
dirs = os.listdir(path)
for file in dirs:
    a = os.path.join(path, file)
    data = pd.read_csv(a)
    if i % 4 == 0:
        data.drop(["noutputs","output[1]","output[2]","output[3]","output[4]",
        "output[5]","output[6]","output[7]","output[8]",
        "output[9]","output[10]","output[11]","output[12]",
        "output[13]","output[14]","output[15]"],
        axis = 1, inplace = True)
    if i % 4 == 1:
```

```
        data.drop(["timestamp_sample","vibe[1]","vibe[2]","output_tracking_error[0]",
        "output_tracking_error[1]","output_tracking_error[2]","control_mode_flags",
        "filter_fault_flags","pos_horiz_accuracy","pos_vert_accuracy","mag_test_ratio",
        "vel_test_ratio","pos_test_ratio","hgt_test_ratio","tas_test_ratio",
        'hagl_test_ratio','beta_test_ratio','time_slip','accel_device_id',
        'gyro_device_id','baro_device_id','mag_device_id','gps_check_fail_flags',
        'innovation_check_flags','solution_status_flags','reset_count_vel_ne',
        'reset_count_vel_d','reset_count_pos_ne','reset_count_pod_d','reset_count_quat',
        'pre_flt_fail_innov_heading','pre_flt_fail_innov_vel_horiz',
        'pre_flt_fail_innov_vel_vert','pre_flt_fail_innov_height',
        'pre_flt_fail_mag_field_disturbed','health_flags','timeout_flags'],
        axis = 1, inplace = True)
    if i % 4 == 2:
        data.drop(["gyro_rad[1]","gyro_rad[2]","gyro_integral_dt",
        "accelerometer_timestamp_relative","accelerometer_m_s2[1]",
        "accelerometer_m_s2[2]","accelerometer_integral_dt","accelerometer_clipping"],
        axis = 1, inplace = True)
    if i % 4 == 3:
        data.drop(["timestamp_sample","device_id","magnetometer_ga[1]",
        "magnetometer_ga[2]","calibration_count"],
        axis = 1, inplace = True)
    data.to_csv(a,index=0)
         i += 1
```

12.3 Model training train.py

12.3.1 Define the model DNN

```
#定义编译器 优化函数
model.compile(
    loss = keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer = keras.optimizers.RMSprop(),
    metrics = ["accuracy"],
     )
```

12.3.2train_accuracy

```
#模型训练
     train_info = model.fit(x_train,y_train,epochs = 10,validation_split = 0.2)
```

12.4 AutoTestAPI.py

12.4.1 load_model

```
#搭建模型
model = Sequential([
    layers.Dense(56,activation = 'relu'),
    layers.Dense(32,activation = 'relu'),
    layers.Dense(2,activation = 'softmax')
      ])
```

12.4.2 model.predict

```
model.save('mymodel.model')
```