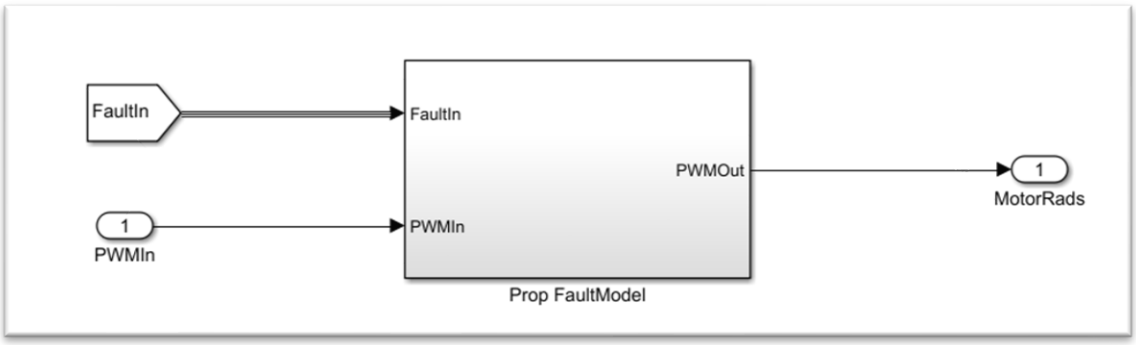
一、 故障注入标准化

协议说明：

1. 故障注入接口是一个 8 位的整型 inSILInts 和 20 位浮点型 inSILFloats，通过发布一个作用域为全局的 FaultIn 数据，可以在各个子模块订阅得到。**实现架构**



上图是一个标准化故障注入的结构。其中 FaultIn 是订阅了全局的 FaultIn 消息，下面是对应的信号输入（以 PWM 为例，将此替换成你所需要的信号），中间的模块为对应的自己开发的一个故障模块，输出位自己故障算法的期望输出。

2. 每个故障都需要命名自己的 ID 值（int32 型，-2147483648 到 2147483647 之间），作为一个密钥机制。
3. 故障目前都通过 inSILInts（8 维 Int32 型）和 inSILFloats（20 维 float 型）输入进来，为了实现：1）同时注入多个故障；2）所有故障采用同样构架；采用如下分布式机制

1） 所有模块都接收同样的数据，根据自己内部的故障 ID，来提取数据，并决定自己是否触发故障。

2）FaultIn 的标签是全局范围，子模块也可以直接用“From”模块订阅得到。

3）在没有故障注入时，每个故障模块不应该影响平台的正常输出。

4）**具体说明如下：**

1）inSILInts

inSILInts 相当于故障校验与匹配的作用。可以用于设定触发不同的故障类型。（即用 inSILInts 去匹配故障 ID）

比如，ID 为 123450 为电机故障，ID 为 123451 为螺旋桨故障，ID 为 123459 为常风故障 ,ID 为 123457 为负载漂移故障。则当一次触发一个故障时，inSILInts=[12345,0,0,0,0,0,0,0]，当一次触发两个故障时，inSILInts=[12345,12346,0,0,0,0,0,0]，当一次触发三个故障时，inSILInts=[12345,12346,12347,0,0,0,0,0].以此类推。

2） inSILFloats(一个 inSILInts 对应两个 inSILFloats 参数，多出的 4 位留作备用参数)

注：某种故障类型可能需要设计 1 个、2 个或 3 个以及更多的可能。由于一个 inSILInts 对应两个 inSILFloats 参数。当故障参数位 1 个时，多出的另一个置为 0；当为 3 个时，设置两次此种故障类型的 ID。

如：

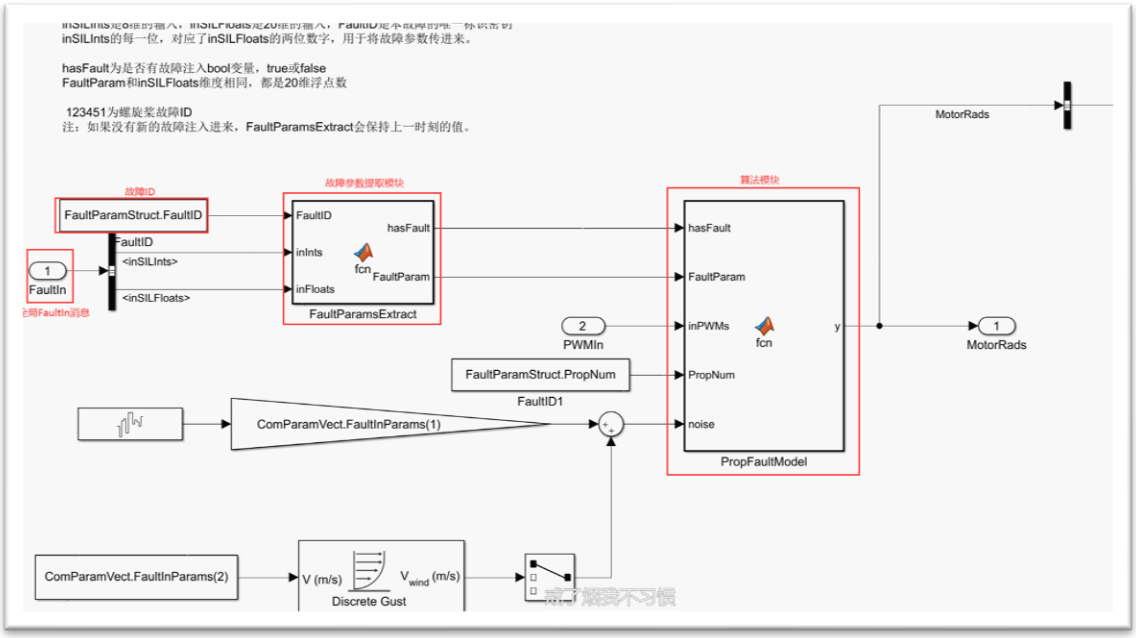
- 当故障参数个数为 1 位时，假设当前的故障情景为：触发一个故障，故障类型为电机故障，故障参数为 0.5。

  设置 inSILInts=[123450,0,0,0,0,0,0,0]

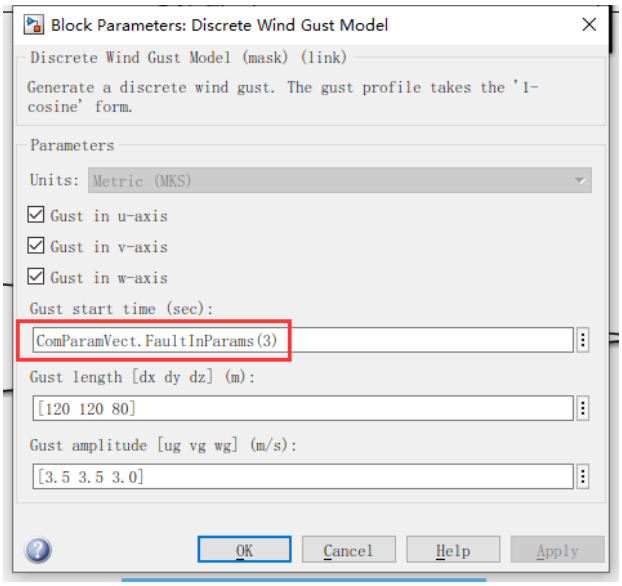
  设置 inSILFloats=[0.5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,17,18,19,20],

- 当故障参数个数为 3 位时，假设当前的故障情景为：触发一个故障，故障类型为常风故障，故障参数为[3,4,5]。
  设置 inSILInts=[123459,123459,0,0,0,0,0,0]（设置两次 123452）
  设置 inSILFloats=[3,4,5,0,0,0,0,0,0,0,0,0,0,0,0,0,17,18,19,20]

4. 每个子故障模块的实现机制如下：



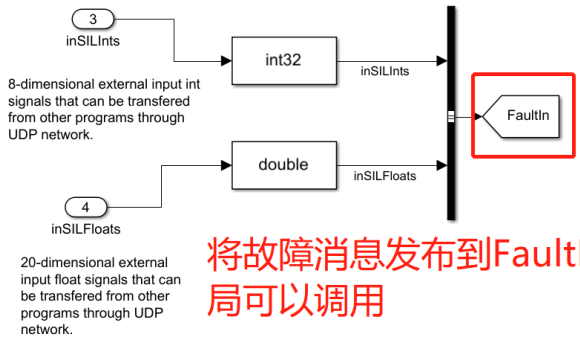
5. 目前参数输入 FaultParamAPI.FaultInParams（32 维 double 型数据，取代原来的 ModelInParams），作为公共参数集，用来实现无法通过 inSILInts 和 inSILFloats 注入的故障参数
6. 原则上：
   a) 为保证平台的通用性，尽量只使用 inSILInts 和 inSILFloats，不要使用 ModelInParams。
   b) 如果，必须使用 ModelInParams，请先发设置 ModelInParams 的消息，再发设置 inSILInts 和 inSILFloats 的消息，这个顺序不可以弄错，不然可能影响故障注入效果。
   c) 如果只使用 inSILInts 和 inSILFloats 来注入故障，那么 inSILInts 内的值可以灵活调整。例如，下面两种情况的效果应该是完全一样的。
   情况一：
   inSILInts=[123450 0 123450 0 0 0 0 0]
   inSILFloats=[1 2 0 0 5 6 …]
   情况二：
   inSILInts=[123450 123450 0 0 0 0 0 0]
   inSILFloats=[1 2 5 6 0 0 …]
   d) 使用 FaultInParams 时，需要注意多个模块共用次参数时，需满足所有模块的约束（大小和数据类型）。
   e) FaultInParams 在很多场景中是需要的，因为有些 Simulink 模块，需要配置参数来输出合适值。如下图的 ComParamVect.FaultInParams(3)（ComParamVect 为 FaultParamAPI 的别名）



标准化开发过程（故障订阅与触发机制）：

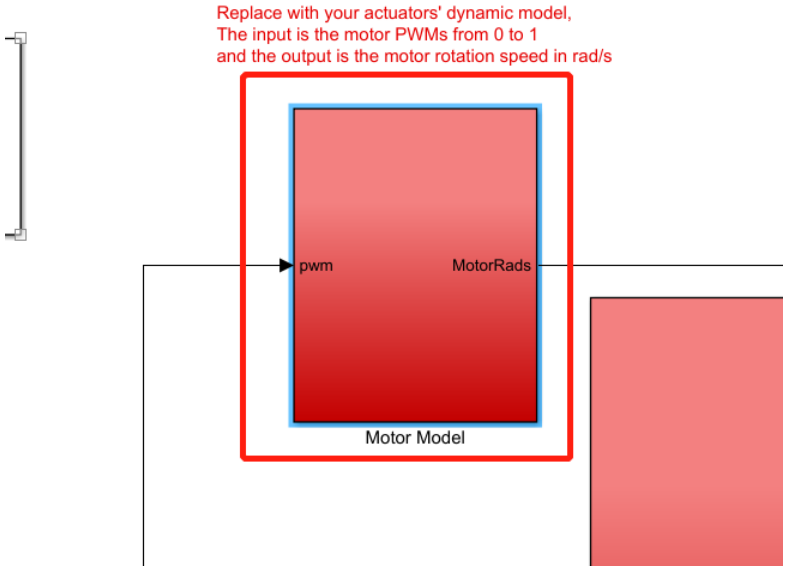
为了实现分布式模型设计的构想，采用了标准化的故障注入接口来辅助实现。具体方法是通过在顶层模型发布标准化故障注入信息 FaultIn，在各个子模块订阅 FaultIn 消息并提取本模块的故障数据来实现。提升了分布式开发的效率。以下以电机模块为例进行展示：

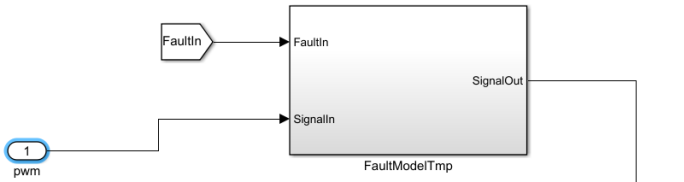
1. 订阅故障注入消息，并发布为 bus 结构体



将故障消息发布到FaultIn中，全局可以调用

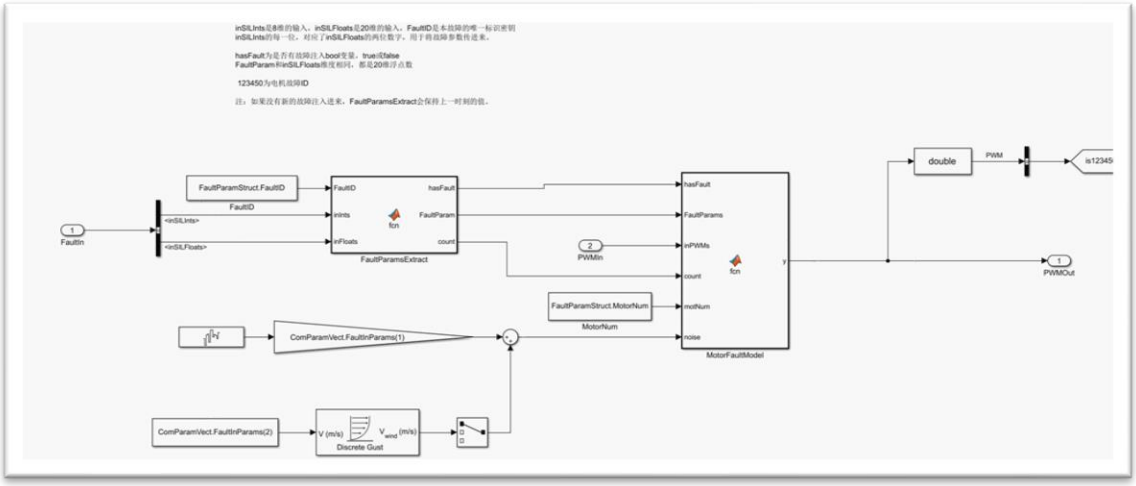
2. 电机模型内，导入故障注入模版





订阅故障消息，根据故障模型处理pwm数据，然后将处理后数据输出

3. 开始编写故障模型



4. 上述故障模型主要在 MotorFaultModel 中实现，可以自行查看代码理解实现机理。本故障模型设置故障 ID 为 123450，需要 4 位 inSILFloats（对应 4 位 inSILInts）用于传入 4 个电机的效率系数（0 到 1 之间），如果系数为 0 则这个电机完全故障，直接输出 0。
5. Python 中实现故障注入的例子（自动测试平台 v5\AutoTestAPI \FaultInjectAPITest.py）

发送一条消息

inSILInts=[12345,12345,0,0, 0, 0,0,0] （12345 号故障，2*4=8 位参数输入）

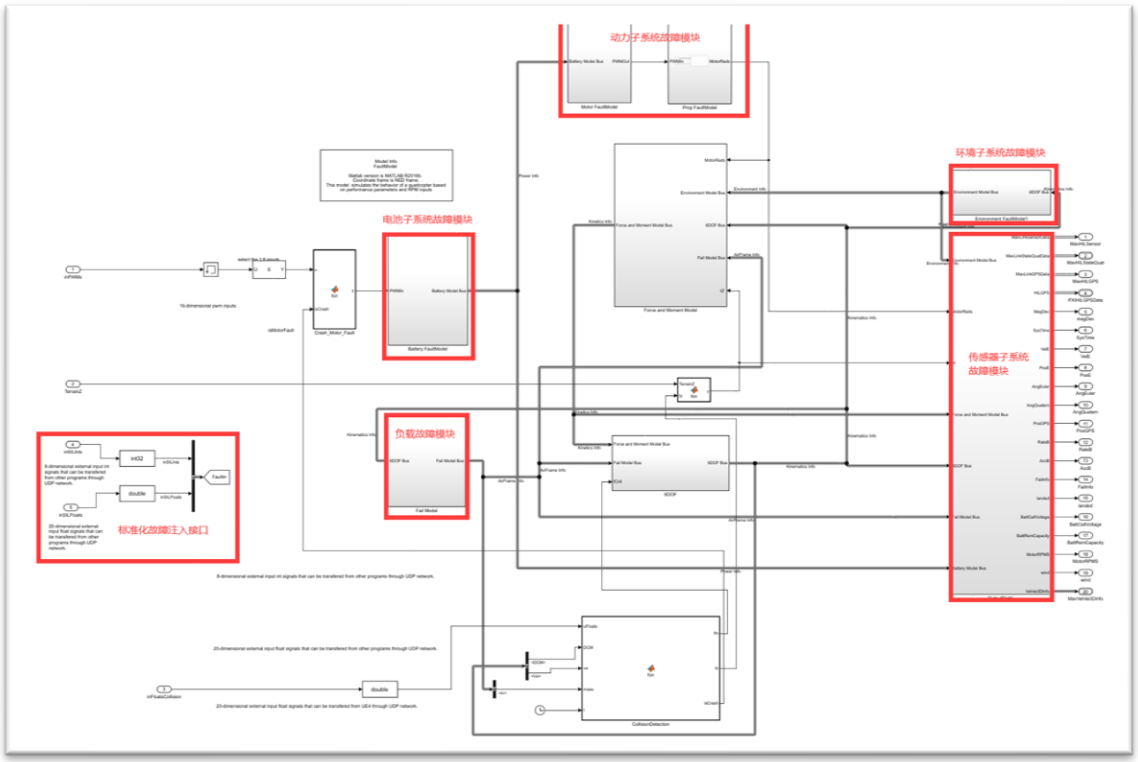
inSILFloats=[1,1,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]（3 号电机停转，其他电机正常）

```
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]=[1,1,0,1]
    mav1.sendSILIntFloat(silInt,silFloat)
    print('Inject a fault, and start loging')
    flag=2
```

6. 运行 bat 脚本（自动测试平台 v5\bat），fixed 后运行 py 脚本，可以看到飞机起飞，悬停，然后注入电机故障，飞机坠毁。

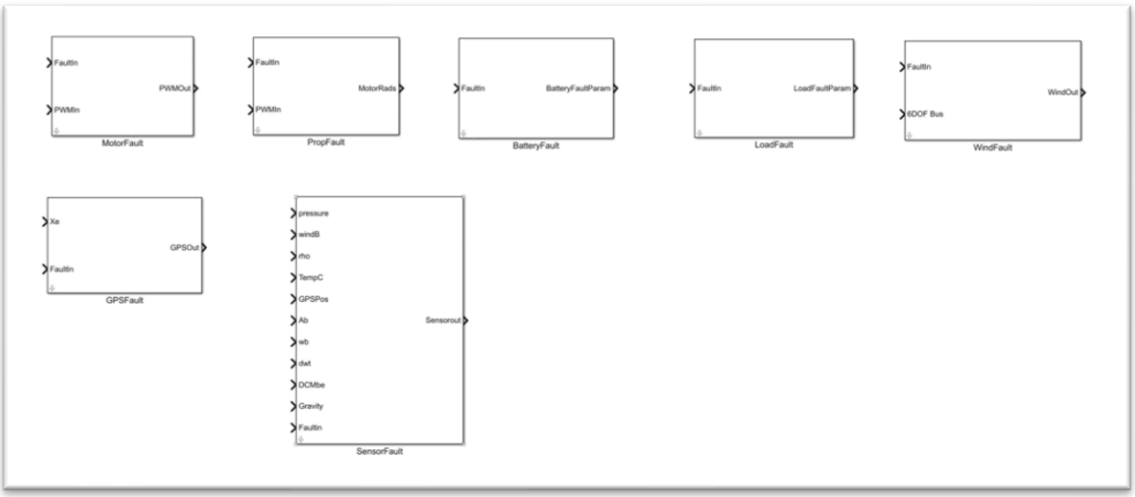
二、 故障模型库标准化

平台现有的故障模型如下：



其中故障模块包含电池子系统、负载子系统、动力子系统、环境子系统、传感器子系统。

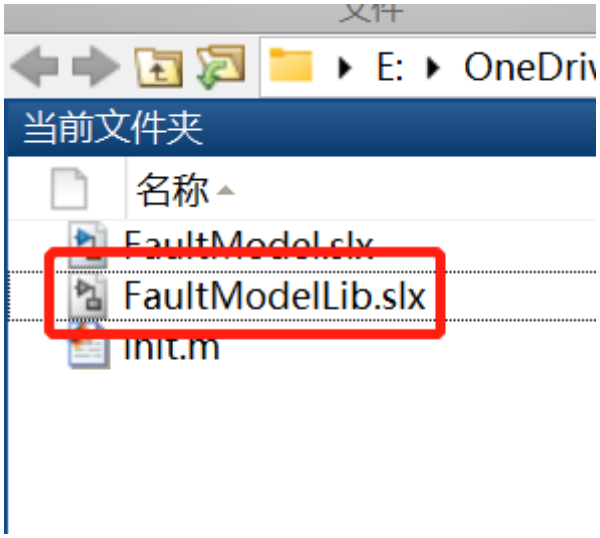
为了实现模块共享与高效开发，将现有的故障模型封装形成了一个故障模块库，通过修改故障模块库即可同步到现有的故障模型。故障模块库的结构如下：



后续开发的故障模块放在此标准化库中，在需要用到的时候，引用标准化库中的模型即可实现高效开发。

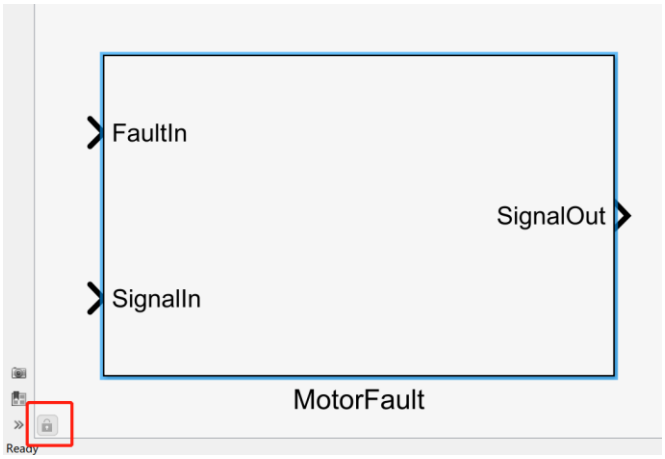
封装方法（电机模型为例）：

1. 所有故障模块需要存放在“FaultModelLib.slx”这个 Simulink 库文件中，作为唯一模块源。后续，需要使用多个本故障模块时，相当与使用的是本库模块的引用。



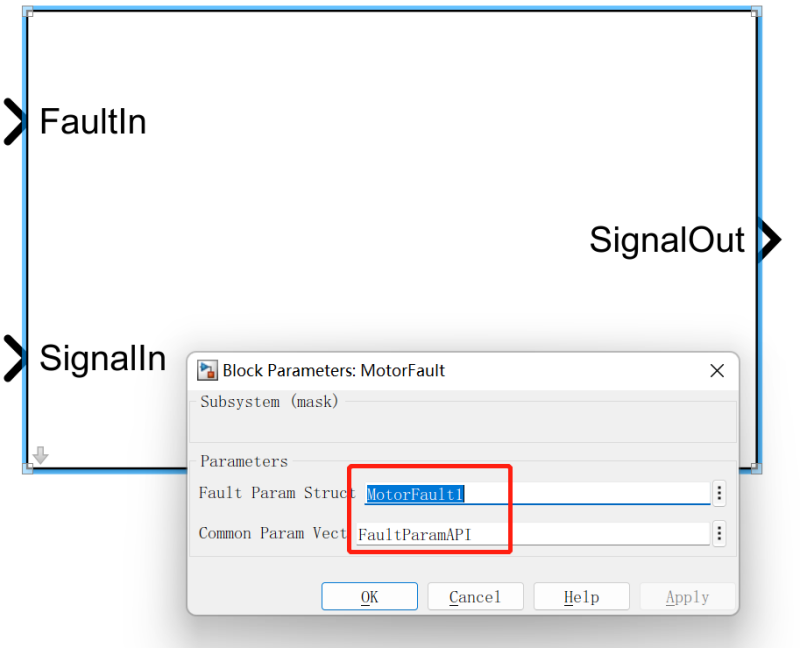
使用库引用方式的好处是，只需要修改“FaultModelLib.slx”中的库模型，所有引用本模型的故障测试用例程序都会跟着改变，便于多人协作，和模块化分享。

2. 其中一个电机故障模块“MotorFault”，见下图所示。



本模块默认是锁定状态，在本库文件中，以及所有引用本模块的 Simulink 程序中都无法改动。如果要改动，需要打开库文件，然后点击上图左下角的锁图标进行解锁。

3. 双击本模块可以看到两个参数输入框，需要分别输入两个变量（可以是标量，向量，或者结构体，这里使用的是结构体的形式）。其中，MotorFault 栏需要输入一个结构体，这个结构体用于存放本故障模块的私有参数（仿真时不可变）；FaultParamAPI 存放的是可以从 Python 发送的 32 维参数，作为可变的公共参数（仿真时可变）。



4. 打开“Init.m”，可以看到定义这两个结构体的代码。

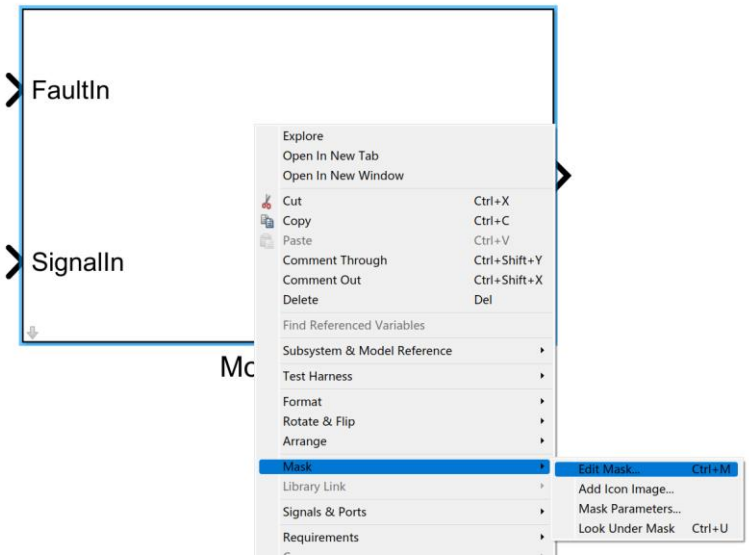
```
% Define the 32-D ModelInParams vector for external modification
FaultParamAPI.FaultInParams = zeros(32,1);

MotorFaultTemp.FaultID=12345;
MotorFaultTemp.MotorNum=int32(4);

MotorFault1=MotorFaultTemp;

MotorFault2=MotorFaultTemp;
MotorFault2.FaultID=12346;

FaultParamAPI.FaultInParams(3)=1;
FaultParamAPI.FaultInParams(4)=2;
FaultParamAPI.FaultInParams(5)=2;
```

电机1结构体

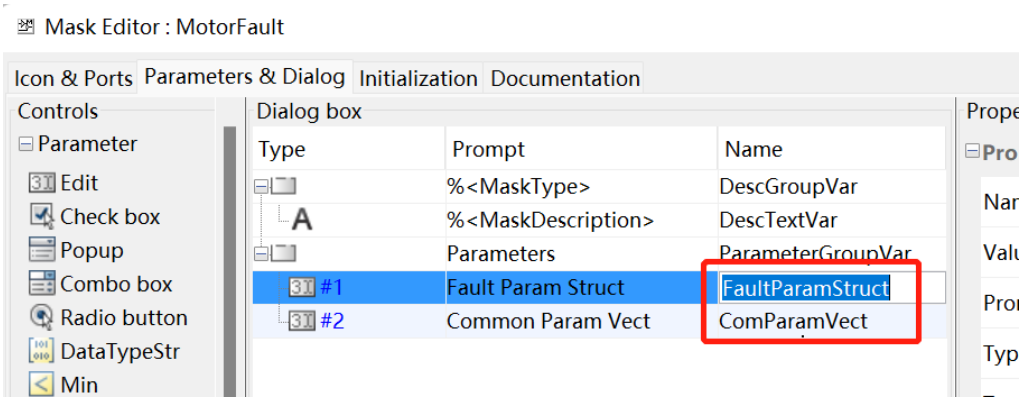
电机3结构体

公共结构体

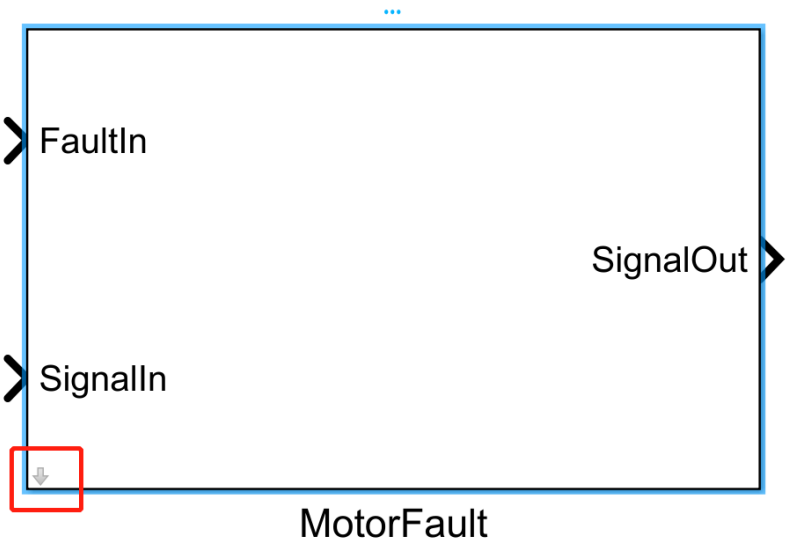
5. 上述的参数输入栏，是我们通过编程模块的 Mask 来实现的。如下图所示，右键可以选择编辑 Mask。注：如果是创建自己的模块，这里应该是创建 Mask。



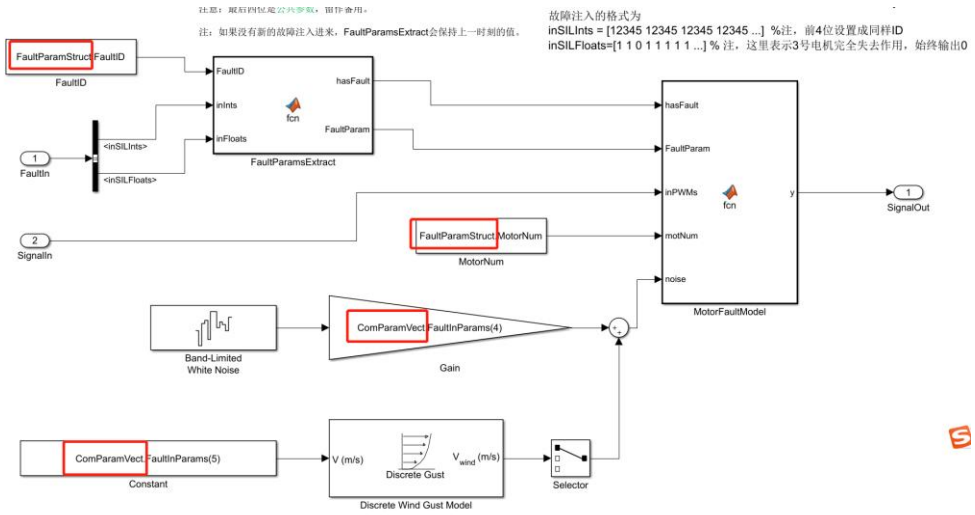
6. 如下图所示，我们创建了两个栏目，这两个栏目的输入，分别用别名 FaultParamStruct 和 ComParamVect 传入到内层模块中。



7. 点击左下角的下箭头，可以进入模块内部。



8. 利用上面的两个传入变量，进行内部编程。注：在本例中，我们将模块的故障 ID 也存储到了 FaultParamStruct.FaultID 中（对应外层传入结构体的 FaultID 元素）



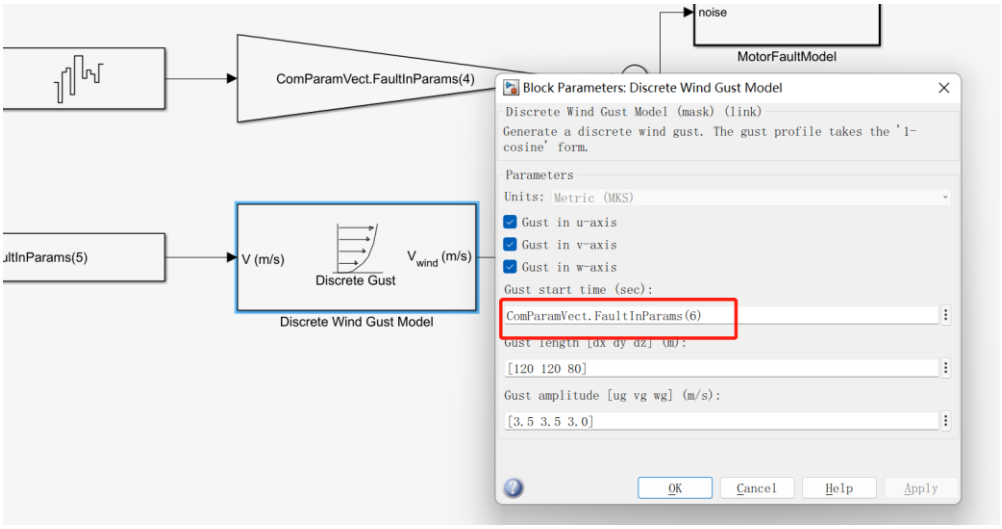
9. 通过本构架，可以将私有参数和共有参数传入故障模块。有些参数须在 init.m 中提前声明

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

MotorFaultTemp.FaultID=123450;
MotorFaultTemp.MotorNum=int32(4);

% Motor Fault Struct
MotorFault1=MotorFaultTemp;
%Prop Fault Stuct
PropFault.FaultID = 123451;
PropFault.PropNum = int32(4);
%Battery Fault Struct
BatteryFault.UseCusTomHoverTimeFaultID = 123452;
BatteryFault.PowOffFaultID = 123453;
BatteryFault.LowVoltageFaultID = 123454;
BatteryFault.LowCapacityFaultID = 123455;
```

```
FaultParamAPI.FaultInParams(1)=1; % Band-Limited White Noise,FaultID = 123450
FaultParamAPI.FaultInParams(2)=2;
FaultParamAPI.FaultInParams(3)=2;
FaultParamAPI.FaultInParams(4)=3; %Wind speed at 6 m defines the low-altitude intensity (m/s),FaultID = 123541
FaultParamAPI.FaultInParams(5)=1; %Wind direction at 6 m (degrees clockwise from north), FaultID = 123541
```

10. 用途解析：
    a) 私有参数，主要用于区分不同电机模块的构架。例如，有一个垂起飞机，他有两种电机，要分别注入两种电机故障。那么需要新建两个电机故障模块，但是他们的故障 ID，和一些故障数据范围之类的参数是有区别的。这些参数区别，可以通过私有参数接口，在仿真前给定。也就是说，本私有参数接口，可以确保同一故障模块，通过修改参数，能适应更复杂的场景。
    私有参数只能在仿真前设定，无法在仿真过程中修改。

b) 公共参数，主要用于一些可调的公共参数。例如，有一个飞机的两个电机故障模块，存在同一个噪声控制模块（作为原始噪声源）。我们需要注入一个故障，能够实时调整所有电机故障模块的数据频率，就可以使用本接口。
注意：公共参数模块主要用于应对一些无法通过输入 inSILInts 和 inSILFloats 来注入的故障。例如，本例程中，下面模块只有参数接口，没有输入接口。



三、 控制序列标准化

为了实现控制序列的标准化，采用了特定数字字符串命令的形式来规范控制序列。数据库中的控制系列如下图所示:

	CaseID	Subsystem	FaultType	ControlSequence	TestStatus
▶	1	Sensor subsystem	123544	2,1;1,1,5;2,3,0,0,-10;1,1,10;2,6,123544,0,0;1,1,10	Not Finished
	2	Power subsystem	123450	2,1;1,1,5;2,3,0,0,-10;1,1,10;2,6,123450,0.4,1;1,1,10	Not Finished

测试脚本会解析数据库中的控制序列，从而实现一次测试的控制逻辑。
测试序列由不同的指令组成，其中，每一条指令由分号结束（;），每条指令又由不同的字符串组成。（所有的字符都是英文符号）
其中每个指令的第一位代表一个操控类。现有的操控类含义解析如下:
1：时间类（包含等待时间、等待复位功能）
2：控制类（包含解锁、上锁、位置控制、速度控制、降落、故障注功能）

每条指令的第二位代表具体的功能，后面的位数代表对应函数的参数。现有的功能含义解析如下:
1：时间类:
　1：等待时间：Wait（times）
　2：等待复位（四旋翼）：WaitReset（Pos）
　3: 等待复位（固定翼）：WaitResetForFixWing(targetpos)
2：控制类:
　1：解锁：Arm（void）
　2：上锁：DisArm（void）
　3：位置控制：FlyPos（pos）

4：速度控制：FlyVel（vel）
5：降落：Land（void）
6：故障注入：FaultInject（param）
7: 发送起飞命令（固定翼）：TakeOff(targetpos)
8: 设置盘旋半径（固定翼）:SetCuriseRadius(radius)
9: 降落（固定翼）:FixWingLand(pos)
10: 发送目标位置（固定翼）:FixWingFlyPos(pos)

一个四旋翼例子如下：
2,1;1,1,5;2,3,0,0,-20;1,2,0,0,-20;2,6,123450,123450,0.6,0.8,1,1;1,1,10;2,5
解析如下：（每条指令的第一位代表一个类，第二位代表此类对应的函数，之后的位数都是函数的参数，没有参数则不用设置）
解锁（2,1）->等待 5s（1,1,5）->发送位置控制，飞至[0,0,-20]处（2,3,0,0,-20）->等待复位，复位位置[0,0,-20]（1,2,0,0,-20）->故障注入，注入电机故障，故障注入参数[0.6,0.8,1,1]（2,6,123450,123450,0.6,0.8,1,1）->等待 10s（1,1,10）->降落（2,5）

一个固定翼例子如下：
2,1;1,1,5;2,7,100,0,-30;1,3,100,0,-30;2,10,500,100,-50;2,8,20;1,3,500,100,-50;1,1,10
解析如下：（每条指令的第一位代表一个类，第二位代表此类对应的函数，之后的位数都是函数的参数，没有参数则不用设置）
解锁（2,1）->等待 5s（1,1,5）->发送起飞命令，起飞目标位置，飞至[100,0,-30]处（以解锁位置为起点）（2,7,100,0,-30）->等待复位，复位位置[100,0,-30]（1,3,100,0,-30）->发送飞行的目标位置[500,100,-50](2,10,500,100,-50)->设置盘旋半径为 20 (2,8,20)->等待飞至目标位置[500,100,-50] (1,3,500,100,-50)->等待 10s（1,1,10）

用户可以根据自己的需要任意组合
注：在故障注入时，模型采用的是标准化故障注入模块实现，具体是由 8 维整型 InSILInts 和 20 维 InSILFloats 参数实现。在数据库的控制序列中，对于 8 位的 InSILInts 和 20 位的 InSILFloats，只需要设置对应的故障 ID 和参数即可，剩余的位数不用补齐。
在给出的例子中，其中的故障注入参数中，123450 为 inSILInts 的故障 ID（一位 ID 对应 2 位故障参数），由于四旋翼有 4 个电机，故设置了两个 123450 的故障，有 4 个故障参数[0.6,0.8,1,1]。其他的类似。

现有故障包括：
1、 电机故障（123450）
故障参数为 4 个，范围为[0,1]。0 为彻底损坏，1 为正常
2、 螺旋桨故障（123451）
故障参数为 4 个，范围为[0,1]。0 为彻底损坏，1 为正常
3、 用户自定义悬停时间（123452）
故障参数为 1 个，即自定义悬停时间
4、 电池失效故障（123453）
无故障参数，直接触发
5、 低电压故障（123454）

故障参数为 1 个，剩余电压比，范围[0,1]
6、 低电量故障（123455）
故障参数 1 个，剩余电量比，范围[0,1]
7、 负载掉落故障（123456）
故障参数 1 个，重量泄露比，范围[0,1]
8、 负载漂移故障（123457）
故障参数 4 个，重量泄露比，以及 x，y，z 轴的漂移因子，范围[0,1]
9、 负载泄露故障（123458）
故障参数 2 个，重量泄露比,以及泄露因子，范围[0,1]
10、 常风故障（123459）
故障参数 3 个，x,y,z 轴的风速
11、 阵风故障（123540）
故障参数 2 个，阵风强度（风速）以及阵风方向
12、紊流风故障（123542）
故障参数 1 个，风强度（风速）
12、 切向风故障（123543）
故障参数 1 个，风强度（风速）
13、 风噪声（123543）
故障参数 2 个，风振幅扰动因子（范围[0,1]）,风增益水平
14、 加速度计故障（123544）
故障参数 1 个，噪声增益水平
15、 陀螺仪故障（123545）
故障参数 1 个，噪声增益水平
16、 磁力计故障（123546）
故障参数 1 个，噪声增益水平
17、 气压计故障（123547）
故障参数 1 个，噪声增益水平
18、 GPS 故障（123548）
故障参数 1 个，噪声增益水平