**实验一：最大模板模型的使用（多旋翼为例）**

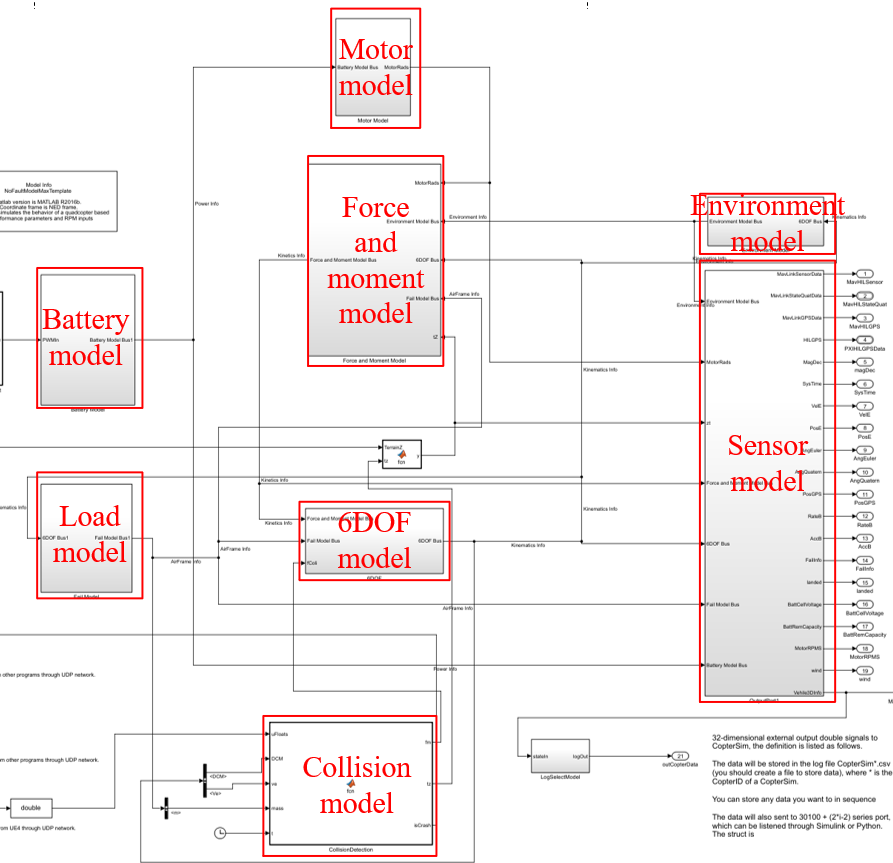
**实验目的：掌握无故障最大模板模型的使用**，了解无故障最大模板的基本模块和关系，为后续的故障建模与注入打好基础 。

**实验步骤：**

1. 本实验例程在文件src/examples/1.NoFaultModelMaxTemplate内，文件结构如下所示：

|  |  |
| --- | --- |
| File | Meaning |
| Init.m | Model initialization file |
| MavLinkStruct.mat | Mavlink structure file data |
| MulticopterModelLib.slx | Multi-rotor model library file |
| NoFaultModelMaxTemplate.slx | Maximum fault-free template model |
| GenerateModelDLLFile.p | Dll model automatically generates files |
| NoFaultModelMaxTemplateHITL.bat | Maximum fault-free template model hardware-in-the-loop one-click script |
| NoFaultModelMaxTemplateSITL.bat | Maximum fault-free template model software-in-the-loop one-click script |

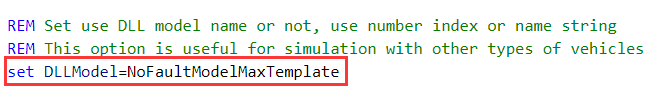
1. 打开src/examples/1.NoFaultModelMaxTemplate/NoFaultModelMaxTemplate文件如下：感兴趣的用户可以学习在线文档来进一步了解多旋翼以及其他载具的建模过程和原理，学习文档可以参考PX4PSP\RflySimAPIs\1.RflySimIntro\1.RflySim\_SupportBook，建模例程可以参考PX4PSP\RflySimAPIs\4.RflySimModel文件。



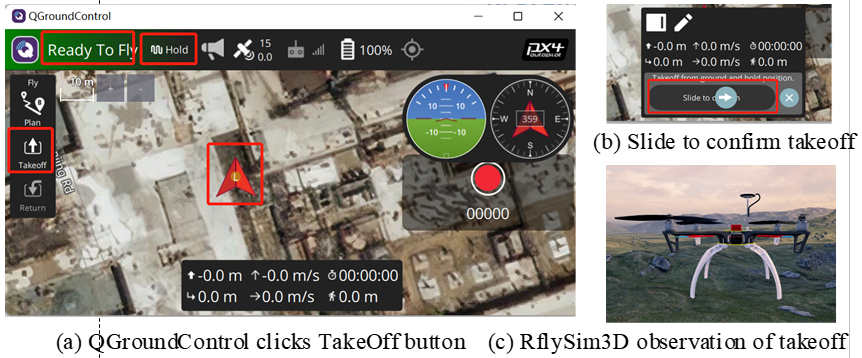
1. 编译模型，自动生成代码，生成MulticopterModel.zip包



1. 右键运行GenerateModelDLLFile.p，生成动态链接库NoFaultModelMaxTemplate.dll文件
2. 将生成的NoFaultModelMaxTemplate.dll拷贝到PX4PSP\CopterSim\external\model文件夹中
3. 打开NoFaultModelMaxTemplateSITL.bat和NoFaultModelMaxTemplateHITL.bat文件，将“set DLLModel”改为该文件名称，右键管理员模式运行（这样coptersim会加载该载具模型进行仿真）



1. 可以看到如下效果

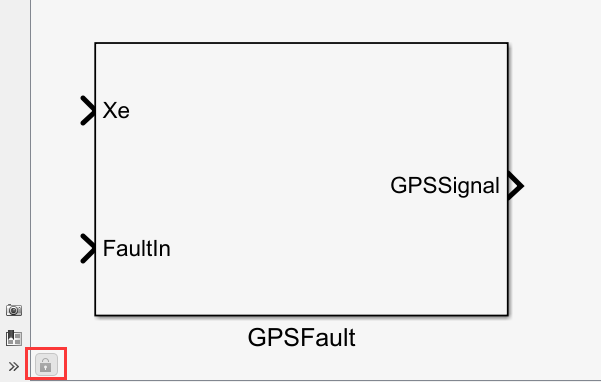


**实验二：故障模型公共库的使用（多旋翼为例）**

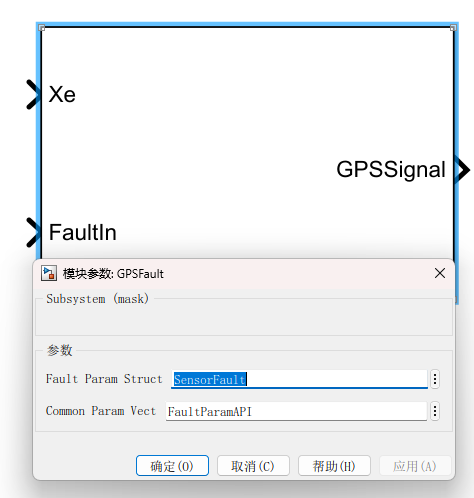
**实验目的：掌握故障库模型公共库的使用**，了解基本的故障库创建与使用方法，为后续的故障建模与注入打好基础 。

**实验步骤：**

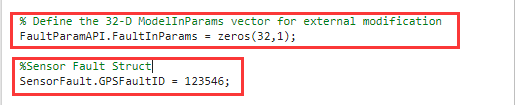
1. 本实验例程在文件src/examples/1.NoFaultModelMaxTemplate内，打开src/examples/1.NoFaultModelMaxTemplate/ MulticopterModelLib.slx文件如下：此处以GPS故障模块的建立与使用为例，注意，所有故障模块需要存放在“MulticopterModelLib.slx”这个Simulink库文件中，作为唯一模块源。后续，需要使用多个故障模块时，可以从该模块引用。



1. 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。
2. 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，SensorFault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）。



1. 打开“Init.m”，可以看到定义这两个结构体的代码



1. 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。

图形用户界面

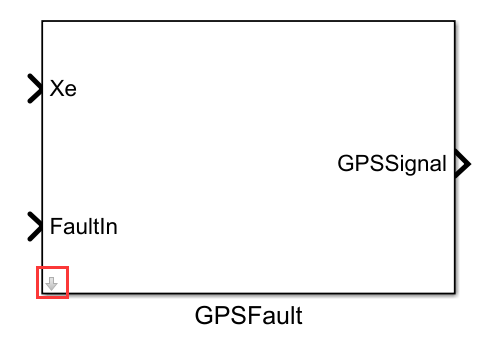
描述已自动生成

1. 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

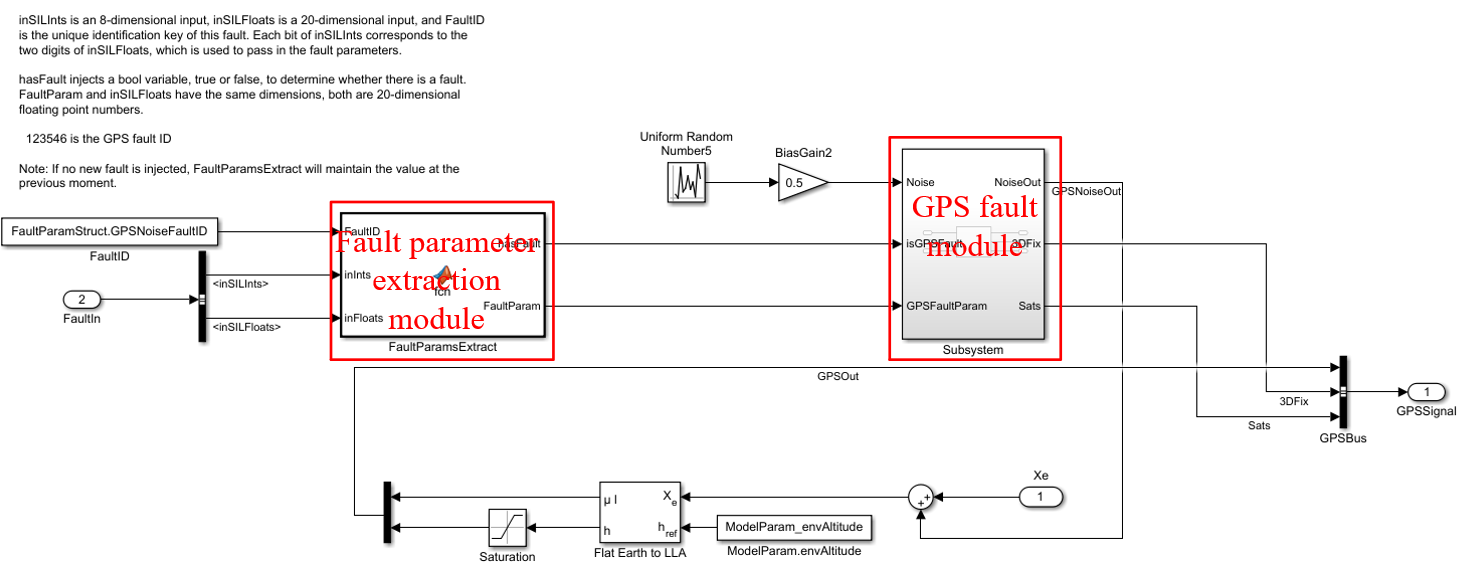
图形用户界面, 表格

描述已自动生成

1. 点击左下角的下箭头，可以进入模块内部



1. 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）



1. 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

至此，实验二全部完成，用户可以根据自己的兴趣搭建杂技的故障模块，并在需要的地方引用即可。