API 说明文件检索大纲

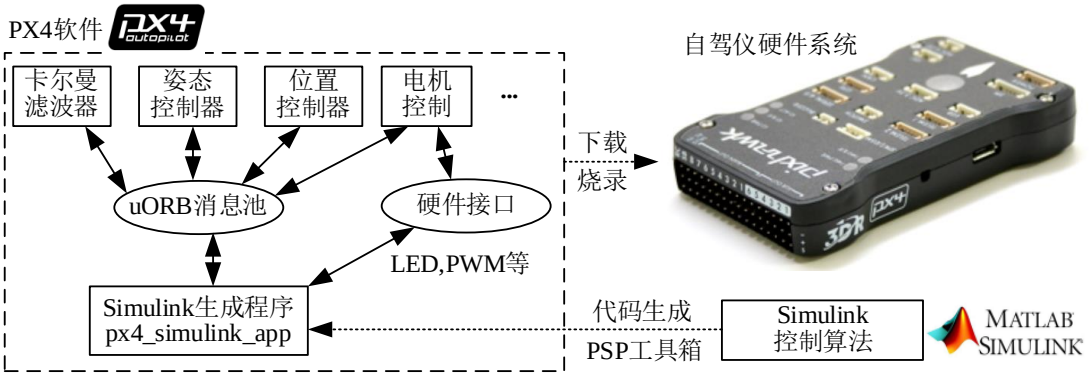
1. RflySim 自动代码生成 Simulink 配置接口

1.1. RflySim 自动代码生成系统的基本构架

Pixhawk Pilot Support Package (PSP,自驾仪支持包)工具箱是 Mathworks 公司官方为 Pixhawk 推出的一个工具箱。该工具箱能在 Simulink 中利用嵌入式代码产生器（Embedded Coder）将 Simulink 模型自驾仪算法自动编译和部署到 Pixhawk 硬件系统中。其主要功能有：

- 能在 Simulink 中对不同的飞机模型和自驾仪算法进行仿真和测试，并能自动将算法部署到自驾仪中；
- 工具箱提供了一些实用实例，包括灯光控制、遥控器数据处理和姿态控制器等；
- 工具箱中提供了很多接口模块，用于访问自驾仪的软硬件组件；
- 能自动记录传感器、执行机构以及自己部署进去的控制器的飞行数据；
- 能订阅和发布 uORB 话题消息。PX4 自驾仪软件的所有数据都暂存在一个 uORB 消息池中，通过 uORB 订阅功能可以从消息池中读取感兴趣的话题，通过 uORB 发布功能可以特定的话题发布到消息池中供其他模块使用。

PX4 软件系统可以分为若干个小模块，每个模块独立运行（多线程并行），各个模块通过 uORB 消息模块的订阅与发布功能实现数据的传输与交互。Simulink 生成的代码部署到 PX4 自驾仪软件之后，不会影响原生 PX4 自驾仪软件的运行，而是新增一个名为“px4_simulink_app”的独立模块（独立线程）并行于其他模块运行。由于原生 PX4 控制算法可能需要访问和“px4_simulink_app”同样的硬件输出资源，这会产生读写冲突。因此，平台一键部署脚本提供了自动屏蔽 PX4 原生固件对执行器的选项，以确保只有“px4_simulink_app”模块能够输出电机控制量，如下图所示。



PSP 工具箱将在 Simulink 中设计的控制算法生成 C 代码；将该代码导入到 PX4 软件系统的源代码中，生成一个“px4_simulink_app”独立运行的程序；工具箱调用编译工具将所有代码编译为“****.px4”的 PX4 软件固件文件；将得到的固件下载到飞控中并烧录，由飞控执行带有生成的算法代码的 PX4 软件。

1.2. PX4 软件系统内部通信

PX4 由两个主要部分组成：一是飞行控制栈（flight stack），该部分主要包括状态估计

和飞行控制系统；另一个是 中间件 ，该部分是一个通用的机器人应用层，可支持任意类型的自主机器人，主要负责机器人的内部/外部通讯和硬件整合。

所有的 PX4 支持的无人机机型（包括其他诸如无人船、无人车、无人水下航行器等平台）均共用同一个代码库。整个系统采用了 响应式（reactive） 设计，这意味着：

- 所有的功能都可以被分割成若干可替换、可重复使用的部件。
- 通过异步消息传递进行通信。
- 系统可以应对不同的工作负载。

PX4 系统通过一个名为 uORB 的发布-订阅消息总线实现模块之间的相互通讯。使用发布-订阅消息总线这个方案意味着：

- 系统是 “响应式” 的 — 系统异步运行，新数据抵达时系统立即进行更新。
- 系统所有的活动和通信都是完全并行的。
- 系统组件在任何地方都可以在保证线程安全的情况下使用数据。

uORB(Micro Object Request Broker,微对象请求代理器)是 PX4/Pixhawk 系统中非常重要且关键的一个模块，它肩负了整个系统的数据传输任务，所有的传感器数据、GPS、PPM 信号等都要从芯片获取后通过 uORB 进行传输到各个模块进行计算处理。实际上 uORB 是一套跨「进程」 的 IPC 通讯模块。在 Pixhawk 中，所有的功能被独立以进程模块为单位进行实现并工作。而进程间的数据交互就由为重要，必须要能够符合实时、有序的特点。

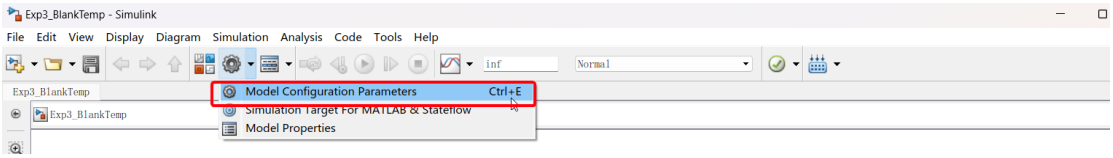
飞控内部使用 NuttX 实时 ARM 系统， 而 uORB 对于 NuttX 而言，它仅仅是一个普通的文件设备对象，这个设备支持 Open、Close、Read、Write、Ioctl 以及 Poll 机制。通过这些接口的实现，uORB 提供了一套“点对多”的跨进程广播通讯机制， “点”指的是通讯消息的“源”,“多”指的是一个源可以有多个用户来接收、处理。而“源”与“用户”的关系在于，源不需要去考虑用户是否可以收到某条被广播的消息或什么时候收到这条消息。它只需要单纯的把要广播的数据推送到 uORB 的消息“总线”上。对于用户而言，源推送了多少次的消息也不重要，重要的是取回最新的这条消息。也就是说在通讯过程中发送者只负责发送数据，而并不关心数据由谁接收，也不关心接收者是否能将所有的数据都接收到；而对于接收者来说并不关心数据是由谁发送的，也不关心在接收过程中是否将所有数据都接收到。

uORB 在数据发布与接收过程中并不保证发送者的所有数据都可以被接收者收到，而只保证接收者在想要接收时能收到最新的数据。而发送与接收的分离可以使飞行过程中各个模块相互独立，互不干扰。实际上一个 uORB 可以由多个发送者发布，也可以被多个接收者接收，也就是说他们之间是多对多的关系。发布者以一定频率更新发布数据到 uORB 平台上，不关心谁来接收。订阅者可以随时来获取数据。更多学习资料请参考：https://docs.px4.io/main/zh/middleware/uorb.html

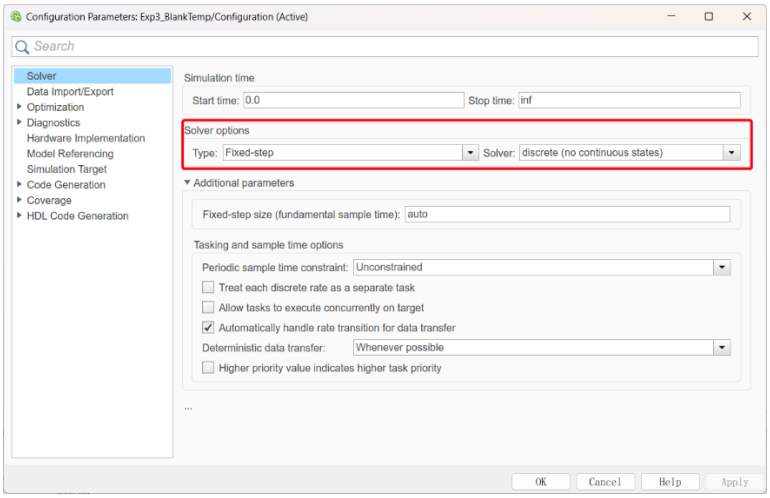
1.3. Simulink 配置接口

MATLAB/Simulink 的 Embedded Coder 模块可生成可读、紧凑且快速的 C 和 C++ 代码，以便用于大规模生产中使用的嵌入式处理器。它扩展了 MATLAB Coder 和 Simulink Coder 的功能，支持通过高级优化对生成的函数、文件和数据进行精确控制。这些优化可提高代码效率，并有助于与已有代码、数据类型和标定参数集成。可以集成第三方开发工具，以便为嵌入式系统或快速原型板上的全套部署构建可执行文件。

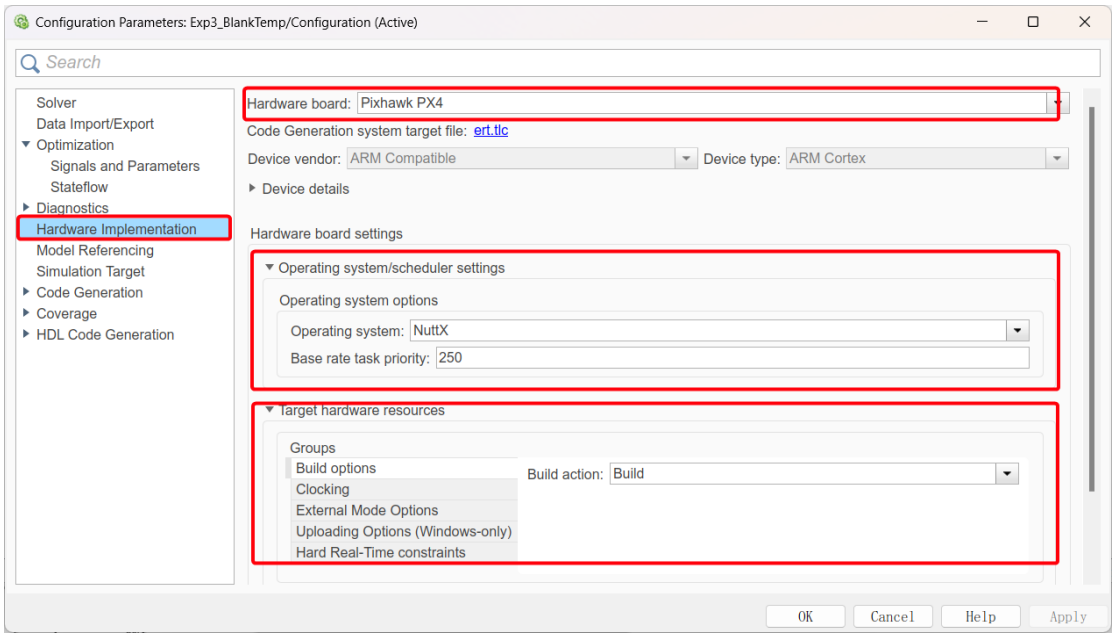
若要生成 Pixhawk 系列飞控硬件，在 Simulink 中搭建完成自己的模型之后，进入 Model Configuration Parameters 进行如下设置：



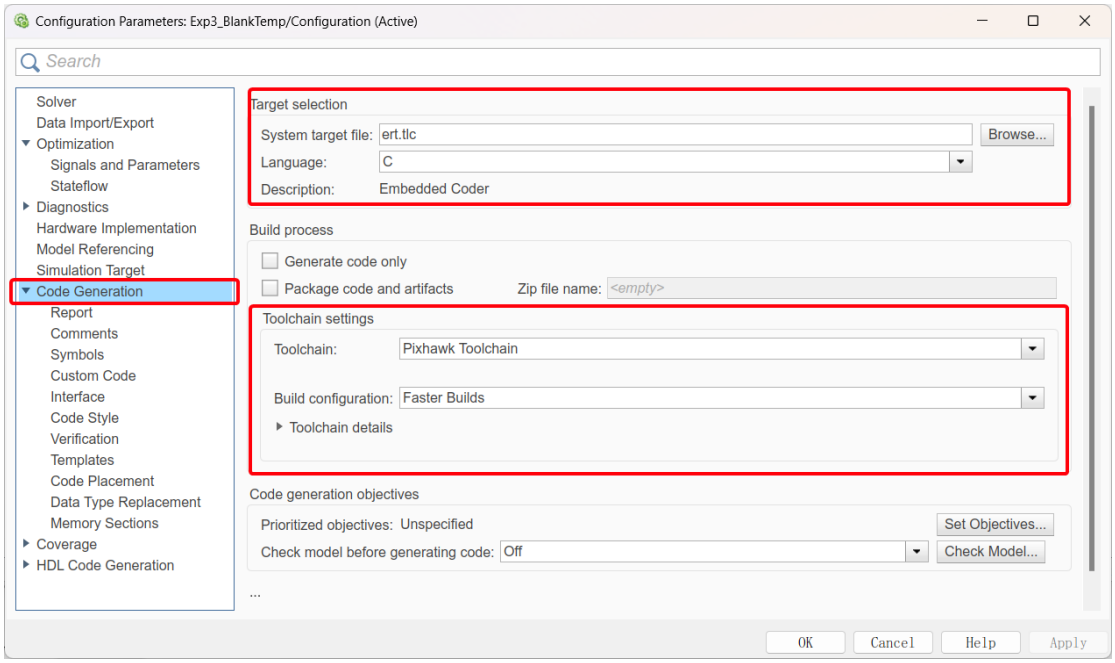
1) 设置求解器的类型为：Fixed-step，求解器选择：discrete。即选择离散的固定步长求解器。



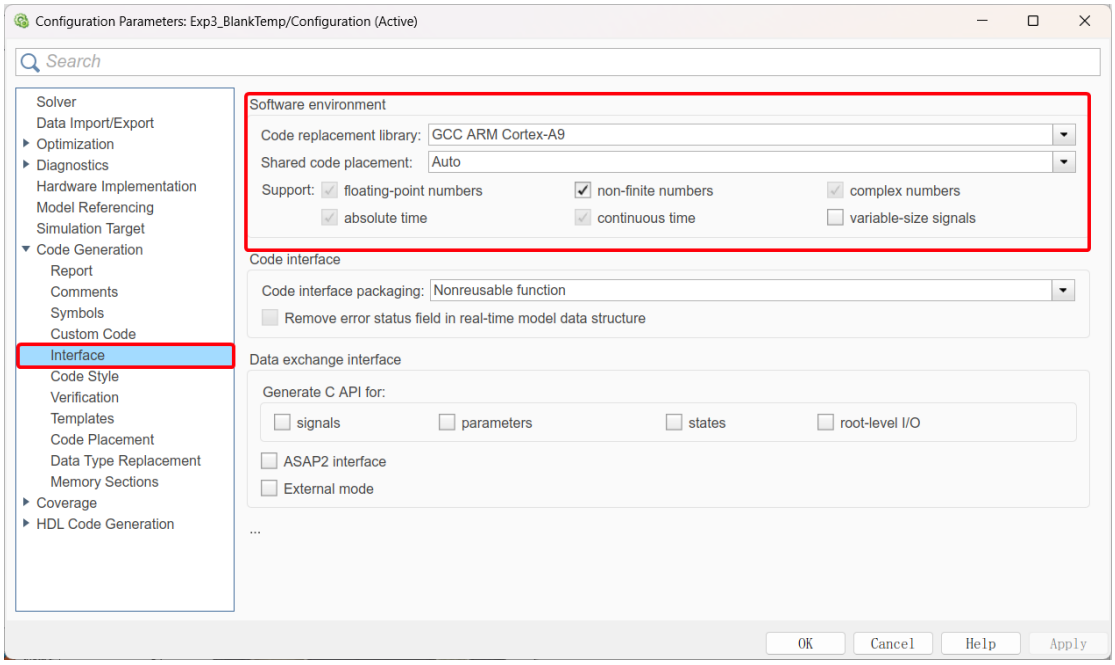
2) 进入 Hardware Implementation 中，设置 Hardware board 板载硬件为 Pixhawk PX4，运行系统选择为 PX4 软件系统中支持的 Nuttx 操作系统，主频设置为 250，Target hardware resource 的编译设置为仅 build 即可。



3) Code Generation 设置系统目标文件选择为 Embeded Coder 的 ert.tlc，语言为 C 语言。工具链设置为：Pixhawk Toolchain，编译配置为 Faster Builds



4) Code Generation 中的接口设置中，设置代码的依赖库为：GCC ARM Cortex-A9，support 中按照如下图中进行设置即可。



更多自动代码生成的选项设置请见文件：0.ApiExps\3.DesignExps\Readme.pdf

1.4. 代码修改接口

RflySim 平台的一键安装脚本，对官方的 PX4 源码进行了一些修改，以此才能更好的兼容平台。核心修改内容包括：

1) 在*\PX4PSP\Firmware\boards 找到对应的编译命令 cmake 文件，如：px4_fmu-v6x_default 是*\PX4PSP\Firmware\boards\px4\fmu-v6x\default.cmake，在其中添加 px4_simulink_app 的模块编译注册。
2) 在*\PX4PSP\Firmware\ROMFS\px4fmu_common\init.d\rcS 启动脚本中，添加 px4_simulink_app 的自启动
3) 在*\PX4PSP\Firmware\src\modules 目录下，创建 px4_simulink_app 文件夹，并生成满足编译需求的源码。
4) 修改 Firmware 下的源码，屏蔽 PX4 自己的电机输出，使得 px4_simulink_app 能够获得电机控制权限。

1.5. 自定义源码导入接口

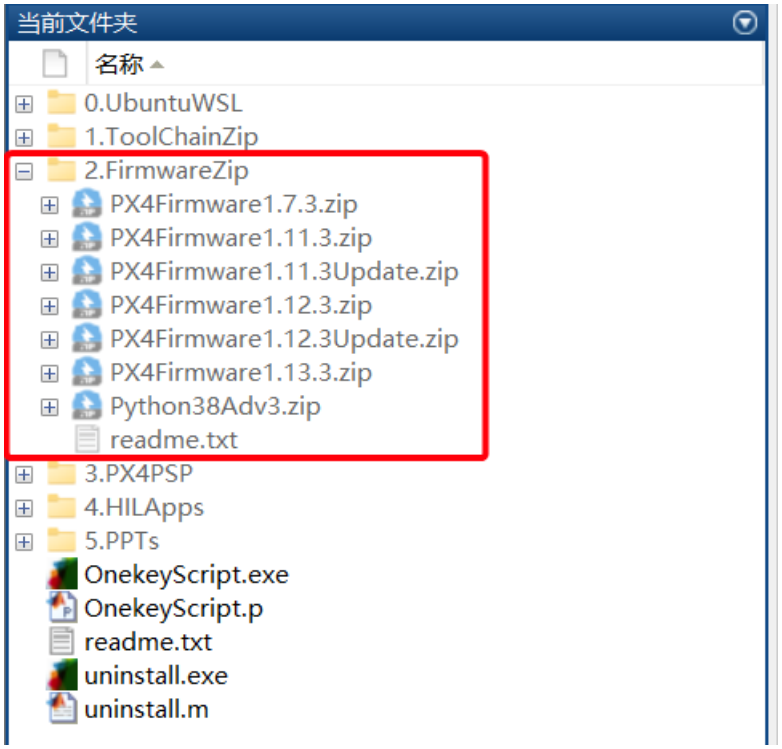
RflySim 平台的安装包中的 2.FirmwareZip 目录中存放了各种 PX4 源码固件，且支持导入自己开发的 PX4 源码。在一键安装脚本在重新部署固件时，会先删除*\PX4PSP\Firmware 文件夹；然后根据选项去解压“2.FirmwareZip\PX4Firmware***.zip”到*\PX4PSP 目录下；最后，解压 PX4Firmware***Update.zip 里面的内容，强制覆盖到 Firmware 里面。

其中 PX4Firmware***.zip 里面存放的是官方源码，从 Github 上下载，没有做任何改动。PX4Firmware***Update.zip 里面包含了我们修改的文件，会覆盖到 Firmware 目录。因此，进行自己的源码部署，可通过下面两种方式。

1) 直接将自己修改后的 Firmware 目录打包，根据自己的版本重命名为 PX4Firmware*****.zip 格式的名字（命名规则见 2.FirmwareZip\readme.txt），并删除 PX4Firmware****Update.zip 文件。这样一键安装脚本，会使用你自己的脚本进行部署。

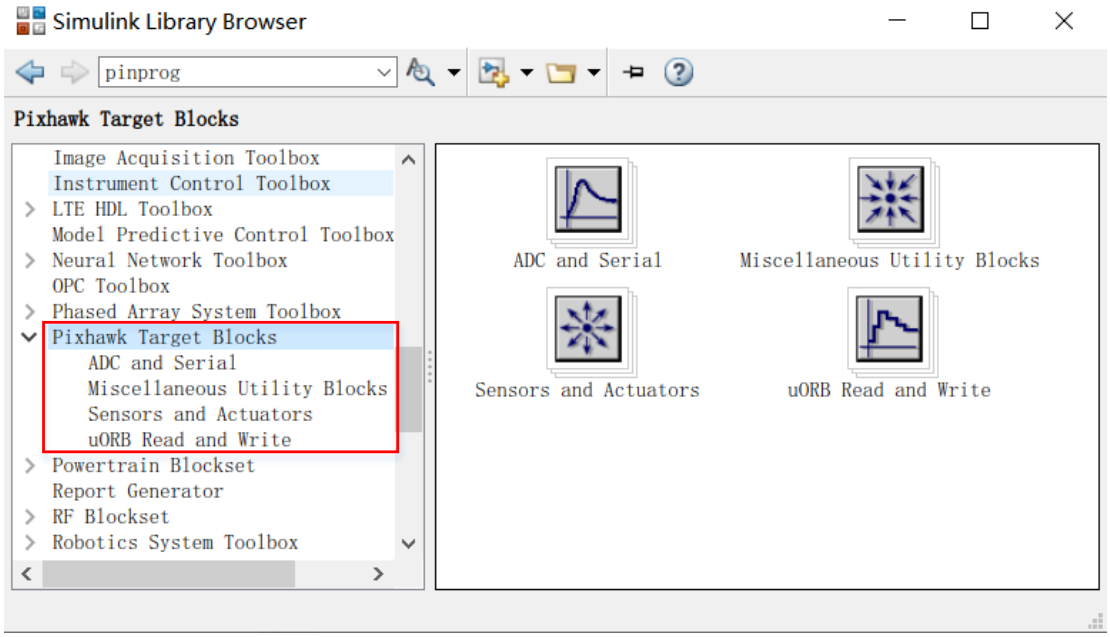
2) 也可以直接将自己修改的部分源码，按文件目录结构，直接存在 PX4Firmware***Update.zip 里面，在部署时会被复制进去，强制替换原来的代码。

RflySim 平台还支持其他的 PX4 固件版本，例如，1.9.2、1.10.2 等，见 2.FirmwareZip\readme.txt 文件，如下图所示。



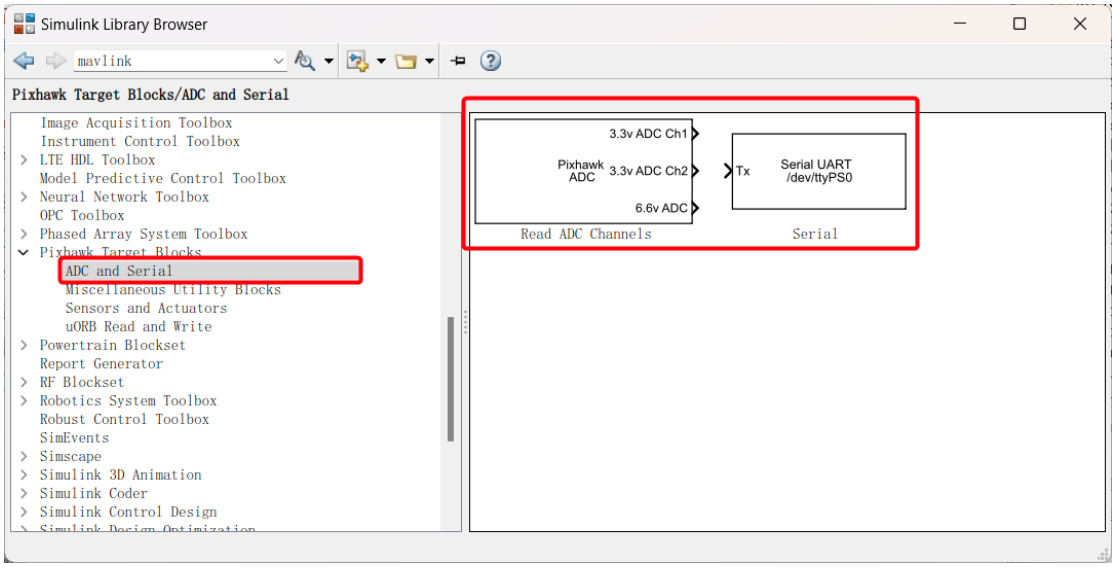
2. Simulink/PSP 工具箱模块接口

PSP 工具箱提供了一些 Simulink 模块作为 Pixhawk 的硬件接口，这些模块只负责生成相应的接口代码而不包含外围硬件的建模。自己搭的控制系统最好在仿真的时候就组织成一个 Simulink 模块，留出必要的接口，以便跟这些硬件接口模块连接，这样也方便重复地使用。这些硬件模块可以在 Simulink 的工具箱 Pixhawk Target Blocks 中查看，如下图所示，它由 5 个子库组成：ADC and Serial—ADC 和串口通信库、Miscellaneous Utility Blocks—其他库、Sensors and Actuators—传感器和执行器接口库、RflySim APIs—RflySim 平台自定义接口库、uORB Read and Write—uORB 消息读取和写入库。其中，RflySimAPIs 子库为 RflySim 平台自定义开发的子库，其中包含有针对底层开发的一些较为实用的接口模块。



2.1. ADC and Serial—ADC 和串口通信库

如下图所示，ADC 读模块可以获取 3 个外部 ADC 通道的数据，串口模块可以对指定串口进行读写操作。



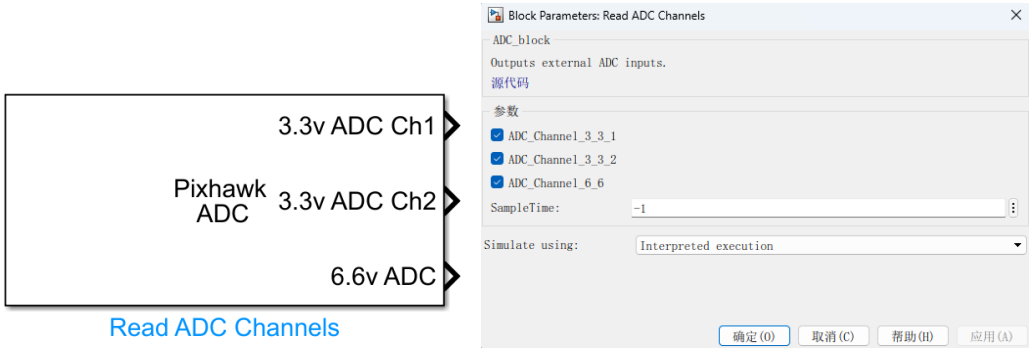
2.1.1. Read ADC Channels—输出外部 ADC 的输入

Read ADC Channels：如下图所示，从 3.3V 和 6.6V 模拟输入选择哪个 ADC 通道作为该模块的输出。请注意，通道 1 和 2 分别对应于 3.3V ADC 的引脚 14 和 15。

信号定义:

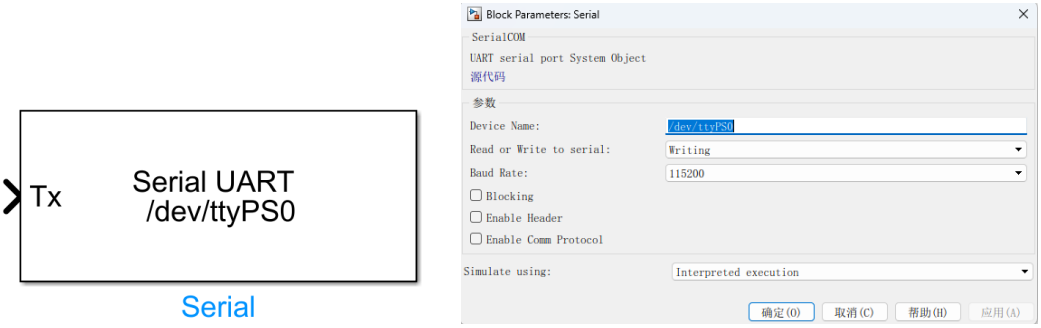
- 3.3V 模数转换(int32)
- 3.3V 模数转换(int32)
- 6.6V 模数转换(int32)

更多信息可点击模块对话框的“帮助”按钮查看。

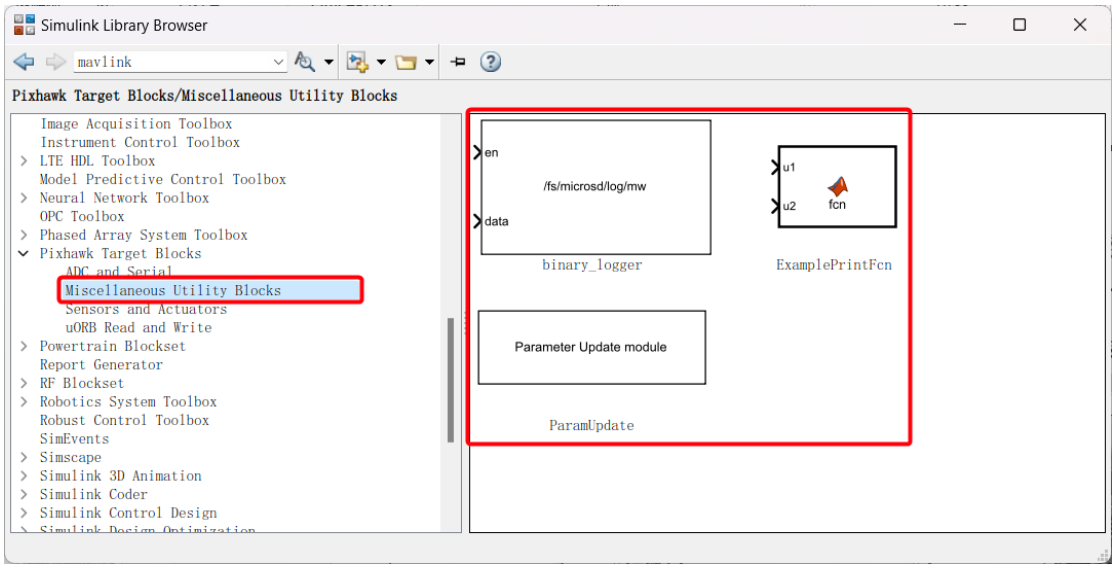


2.1.2. Serial—串行通信模块

Serial: 如下图所示，串行通信块，允许读取和写入设备，指定设备名称和读或写选项。具体用法可点击模块对话框的“帮助”按钮查看。



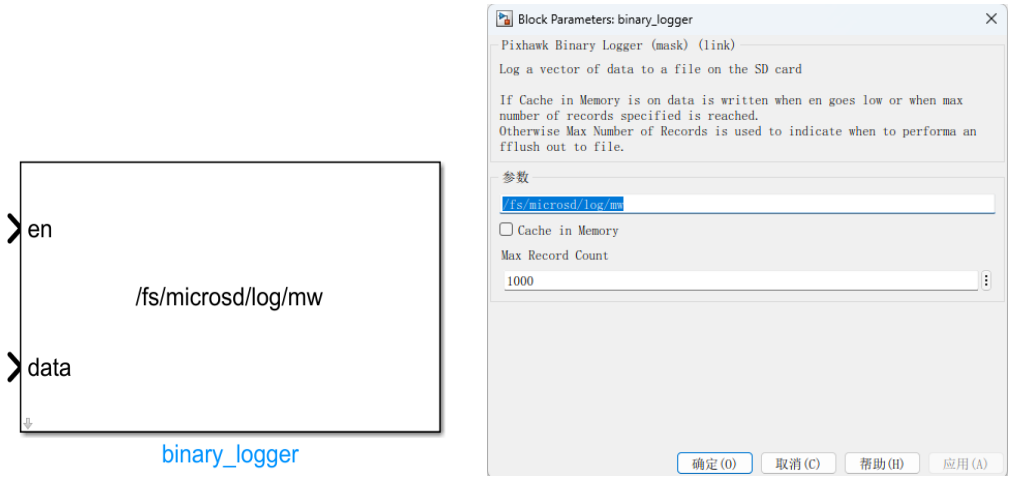
2.2. Miscellaneous Utility Blocks—其他库



2.2.1. binary_logger—数据记录模块

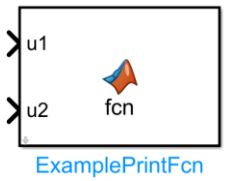
这个模块将数据记录为 double 型存储到 SD 卡中，如果在内存中缓存打开，则当 en 变低或达到指定的最大记录数时写入数据。在程序执行过程中第一个使能输入信号必须先触发高电平再降为低电平才能成功地记录数据，如果在代码结束运行之前没有降为低电平，那么记录文件将被认为处于打开状态，不可访问。此外，日志必须存储在目录“/fs/microsd/”下。如下图所示，可以设置日志存储路径及最大记录条数。具体用法可点击模块对话框的

“帮助”按钮查看，具体例程实验见文件：0.ApiExps\5.Log-Write-Read\Readme.pdf



2.2.2. ExamplePrintFcn—打印函数示例

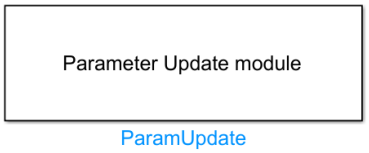
如下图所示，这个模块将信号数据内容打印到 PX4 Nuttx 控制台终端。这个块应该被视为一个“示例”块，打印消息可以以用户认为合适的任何方式构造。Coder.ceval()是一个 MATLABCoder 函数，用于计算 printf()语句以传递两个值。需要注意的是，在 Nuttx 中有一个错误，有时浮点数无法正确显示。请使用 warnx()而不是 printf()作为解决方法。

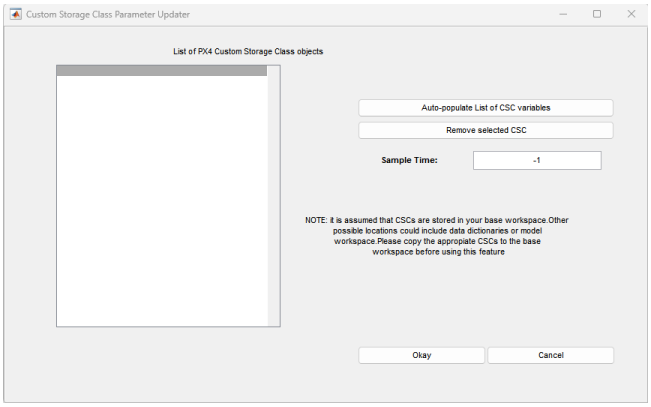



```
function fcn(u1, u2)
%#codegen

if strcmp(coder.target, 'rtw') == true
    coder.ceval('printf','%d %d %c', int32(u1), int32(u2), int8(10))
end
```

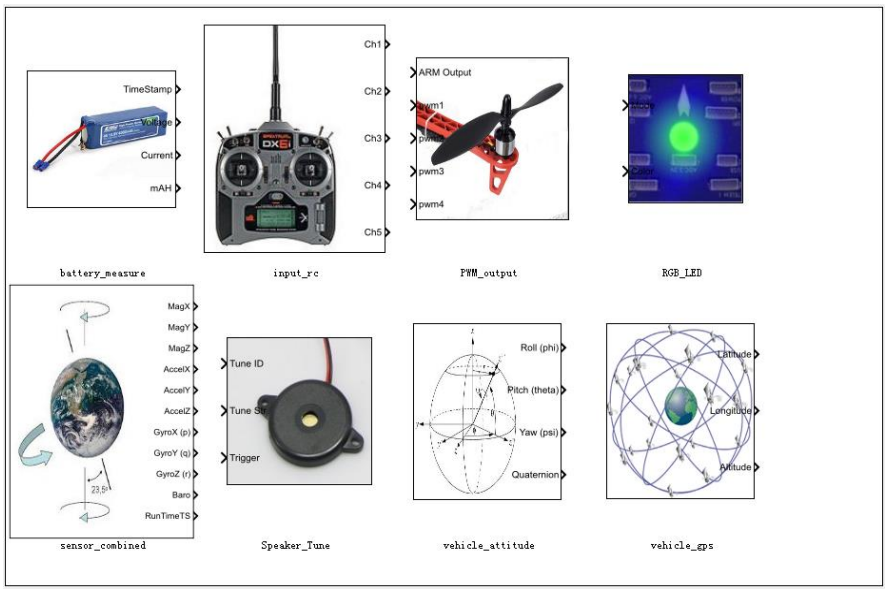
2.2.3. ParamUpdate—自定义存储类参数更新模块

如下图所示，该模块为更新自定义的 PX4 软件参数。在使用的过程中需要注意:假定 csc 存储在你的基本工作区中，其他可能的位置包括数据字典或模型工作区。在使用此功能之前，需要将适当的 csc 复制到基本工作区。





2.3. Sensors and Actuators—传感器和执行器接口库



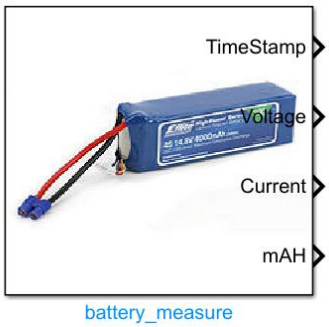
2.3.1. Battery_measure—电池数据模块

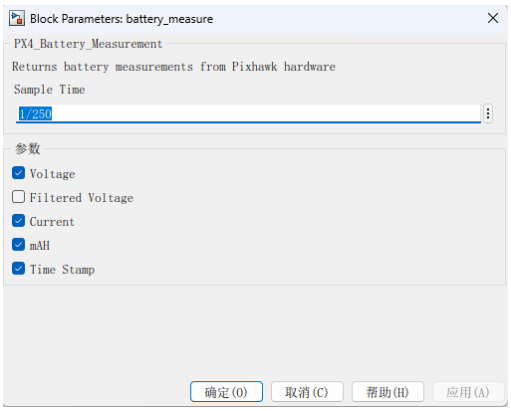
通过这个模块可以获取电池的健康状态，它是通过订阅 uORB 话题发布器 battery_status 发布的消息实现的，所以在实际运行时需要保证 Pixhawk 上插入电源模块才能获取正确数据。

信号定义:

- Voltage：(double)电池电压，单位为伏特。
- Filtered Voltage：(single) 过滤电池电压，单位为伏特。
- Current：(single)电池电流，以安培为单位。
- mAH：(single)以 mAH 为单位的放电量。
- Timestamp：(int32)测量的时间戳。

如下图所示，模块提供了采样率的设置框，以及可选择的电池状态选项。更多信息可点击对话框的“帮助”按钮查看。





2.3.2. Input_rc—遥控器输入模块

该模块允许用户访问来自 RC 发射机的信号， 通过这个模块可以选择输出的信号，包括多个遥控器通道的值，以及其他的一些信息。如下图所示，这些包括：

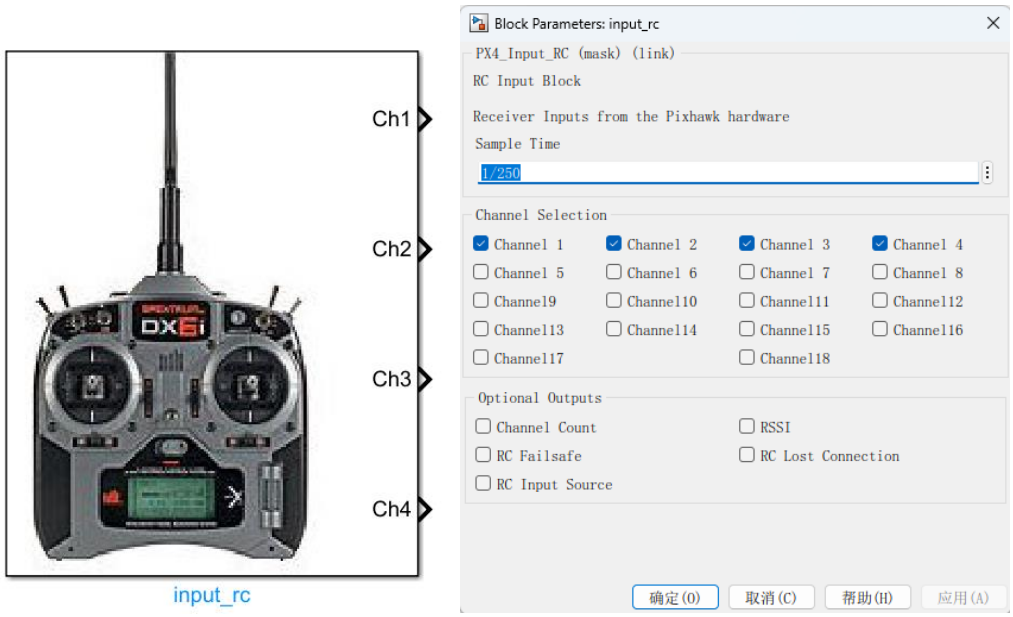
1. Channel Selection—通道选择
   a) uint16 数据类型，表示来自控制器的 PWM(在使用中)值。
   b) 测量每个支持 通道的脉冲宽度。
2. Channel Count—通道数
   a) Uint32 位数据类型，被 PX4 检测器检测的通道数。
3. RC Failsafe—遥控器信号失效保护
   a) 布尔数据类型，指示 RC Tx 正在发送 FailSafe 信号（如果设置正确）
   b) 显示 failsafe 标志：在 Tx 失败或者 Tx 超出范围时为 true,否则为 false。
   c) 只有真实状态是可靠的，因为市场上有一些（PPM）接收器在没有明确告诉我们的情况下进入故障安全。
4. RC Input Source—遥控器信号输入源
   a) 枚举数据类型，指示 RC 输入来自哪个源。
   b) 在 ENUM 文件中找到有效值：

   RC_INPUT_SOURCE_ENUM.m

```
RCINPUT_SOURCE_UNKNOWN          (0)
RCINPUT_SOURCE_PX4FMU_PPM       (1)
RCINPUT_SOURCE_PX4IO_PPM        (2)
RCINPUT_SOURCE_PX4IO_SPEKTRUM   (3)
RCINPUT_SOURCE_PX4IO_SBUS       (4)
```

5. RSSI—接收信号强度指标
   a) 接收信号强度指标（RSSI）：<0：未定义；0：无信号；255：全接收。
6. RC Lost Connection—遥控器信号丢失连接
   a) 指示 RC 接收器连接状态的布尔数据类型。
   b) 如果没有帧在预期时间内到达，则为 True,否则为 false。
   c) True 通常意味着接收器已断开连接，但也可以表示在“愚蠢的”系统上无线电链路丢失。
   d) 如果带有 failsafe 选项的 RX 在链路丢失后继续传输帧，则保持 false。

更多信息可点击对话框的“帮助”按钮查看或者查阅官方 PDF 文档。具体例程实验见文件：0.ApiExps\2.PSPOfficialExps\Readme.pdf



2.3.3. PWM_output—电机 PWM 模块

通过这个模块可以发送 PWM 信号到 PX4IO 的输出端口以控制电机转动，可以选择 PWM 的更新率及输入通道。

为了使飞行控制臂(使能)从软件端输出，arm 输出输入必须保持高(布尔值 TRUE)。只有这样，PWM 值才会发送到 PX4 硬件端口。这通常是一个功能的 RC Tx 结合其他飞行模式编程在 Simulink 模型由用户。

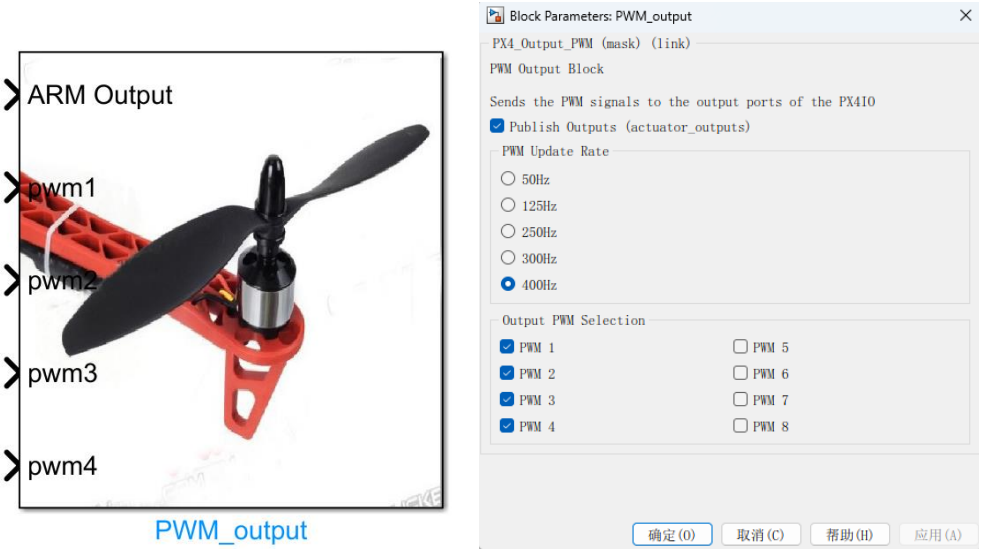
该块有 8 个可用端口(数据类型 uint16)，可以选择性地选择。这些对应于 px4fmu 硬件上的 8 个 PWM 输出端口。

PWM 的单位值是微秒(usec)，对应于脉冲宽度(1500 是 1500usec 或 1.5 毫秒)。

PWM 更新速率在 px4simulinkapp 启动时设置为 400Hz(或用户在块对话框中设置的内容)。可用的 PWM 更新速率为:50、125、250、300 和 400Hz。

px4_simulink_app 还将在启动时和 ARM 端口设置为低(布尔 FALSE)时设置 900usec 的空闲值，以便 ESC 控制器不会超时。

如下图所示，更多信息点击对话框的“帮助”按钮查看。



需要注意的是：

1. RflySim 版本低于 v3.00 的 PWM_output 模块输出，只支持 PWM 格式的电调协议，不支持 DShot，这个需要在文档中强调。

2. RflySim 版本高于 v3.00 的 PWM_output 模块输出，已经支持真机和硬件在环仿真，不需要进行替换（目前，仅支持多旋翼）。

在控制器设计时，推荐使用 TorqueThrustCtrls—力和力矩控制信号模块，这个模块能同时支持真机和仿真，且支持 DShot 等特殊电调模式。

2.3.4. RGB_LED—LED 灯

通过这个模块可以控制 LED 灯闪烁的模式和颜色。如下图所示，模块接收两个输入，一个是模式（Mode），另一个是颜色（Color），这两个输入是枚举数据类型。可以通过输入以下命令来查找 MATLAB 命令窗口中有效的值:

>>:enumeration (“RGBLED COLOR ENUM”)

‘RGBLEDCOLORENUM’类的枚举成员:

COLOR_OFF

COLOR_RED

COLOR_YELLOW

COLOR_PURPLE

COLOR_GREEN

COLOR_BLUE

COLOR_WHITE

COLOR_AMBER

COLOR_DIM_RED

COLOR_DIM_YELLOW

COLOR_DIM_PURPLE

COLOR_DIM_GREEN

COLOR_DIM_BLUE

COLOR_DIM_WHITE

COLOR_DIM_AMBER

'RGBLED_MODE_ENUM'类的枚举成员:

MODE_OFF

MODE_ON

MODE_BLINK_SLOW

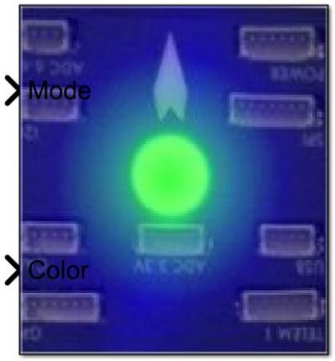
MODE_BLINK_NORMAL

MODE_BLINK_FAST

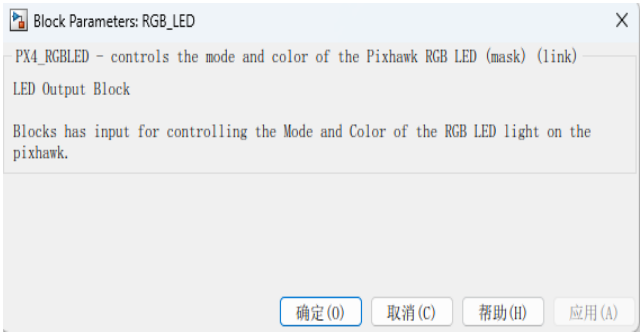
MODE_BREATHE

MODE_PATTERN

更多信息可以点击对话框的“帮助”按钮查看。具体例程实验见文件：0.ApiExps\2.PSP OfficialExps\Readme.pdf



RGB_LED



2.3.5. sensor_combined—传感器组合模块

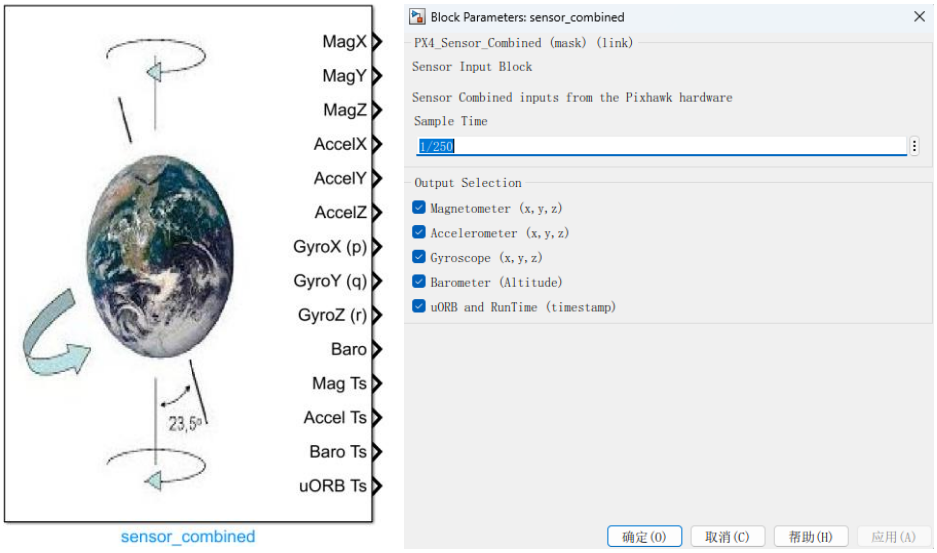
通过这个模块可以获取 Pixhawk 中可用的传感器数据，然后这些数据可以用于控制模型的设计。可获取的数据包括磁力计、加速度计、陀螺仪、气压计和时间戳。

信号定义:

- Magnetometer(x,y,z):(single)磁强计(x,y,z),NED 下的磁场，高斯单位。
- Accelerometer(x,y,z): (single)加速度计(x,y,z),加速度在 NED 下的框架，单位 m/s^2。
- Gyroscope(x,y,z): (single)陀螺仪(p,q,r),角速度单位弧度每秒。
- Barometer(Altitude): (single)气压计(海拔),已补偿温度的气压(毫巴)。
- uORB and RunTime(timestamp):(double)从陀螺仪启动以来以微秒为单位的时间戳。

sensor_combined 块需要在 PX4 硬件上运行 px4io 服务，以便获得有效的信号值。

如下图所示，更多信息可点击对话框的“帮助”按钮查看。具体例程实验见文件：0.Api Exps\11.SenorDataGet\Readme.pdf



2.3.6. Speaker_Tune—蜂鸣器模块

通过这个模块可以控制蜂鸣器在特定的事件发出特定的音调。如下图所示，这个模块接受 3 个输入信号。

- Tune ID:这是可以播放的预定义“曲调”的枚举数据类型。类’PX4 TUNE ENUM’的枚举成员:

  STOP TUNE

  STARTUP TUNE

  ERROR TUNE

  NOTIFY_POSITIVE_TUNE

  NOTIFY_NEUTRAL_TUNE

  NOTIFY_NEGATIVE_TUNE

  ARMING_WARNING_TUNE

  BATTERY_WARNING_SLOW_TUNE

  BATTERY_WARNING_FAST_TUNE

  GPS_WARNING_TUNE

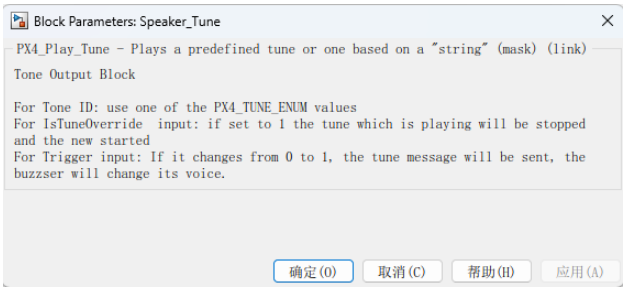
  ARMING_FAILURE_TUNE

  PARACHUTE_RELEASE_TUNE

- IsTuneOverride:用于定义要播放的自定义曲调。用户可以使用一个常量块来定义要播放的字符串。
- Trigger:这是一个触发信号，指示何时播放预定义的曲调(如果触发值变为 1)或自定义曲调(如果触发值变为 2)。曲调的变化将仅由该值的变化触发。

更多信息可点击对话框的“帮助”按钮查看。

Speaker_Tune



2.3.7. vehicleattitude—姿态数据模块

该模块提供了经过滤波的姿态数据（欧拉角和四元数）并且提供对正在运行的服务的访问，该服务计算无人机的姿态。必须运行一个 uORB 主题(vehicle attitude(姿态测量))发布者，以便该块提供有效的信号值。

为了使该块返回有效值，必须在 px4fmu 上运行其中一个。例如:

px4fmu-v2 ekf2:用于姿态估计的 ekf 扩展卡尔曼滤波器

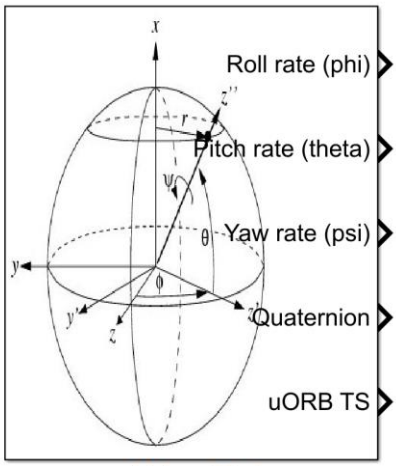
px4fmu-v2 default: SO(3)-利用加速度计、陀螺仪和磁力计进行姿态估计

如下图所示，模块提供了采样率的设置框，以及可选择的姿态选项。

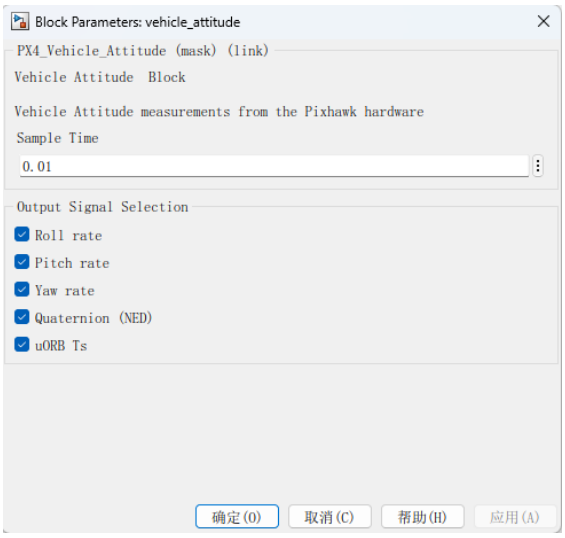
信号定义:

- Roll rate:(single)滚转率，单位为度/秒或者弧度/秒（NED）。
- Pitch rate: (single)俯仰率，单位为度/秒或者弧度/秒（NED）。
- Yaw rate: (single)偏航率，单位为度/秒或者弧度/秒（NED）。
- Quaternion(NED): (single)根据 uORB 发布者可选（NED）。
- uORB Ts：用于实现高效的数据交换和时间同步。

更多信息点击对话框的“帮助”按钮查看。



vehicle_attitude



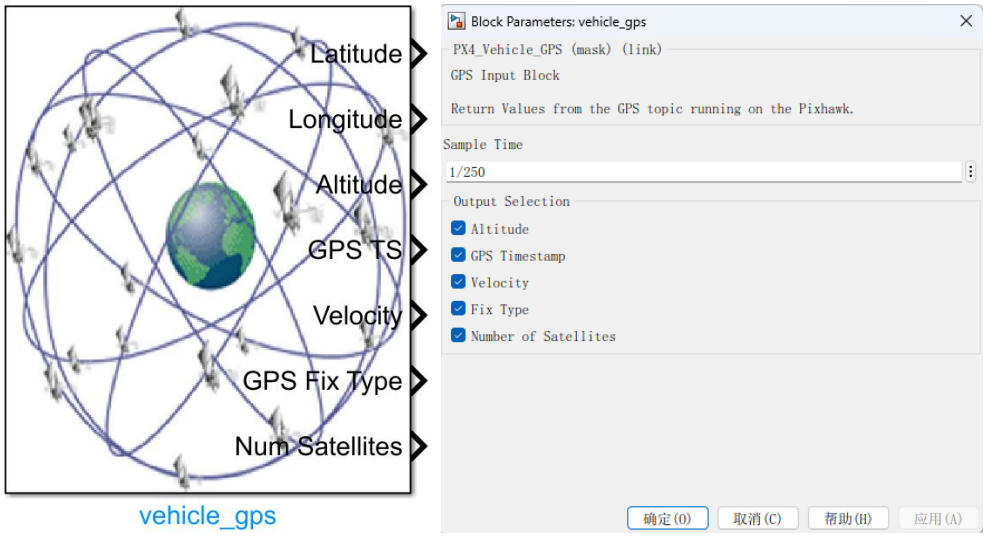
2.3.8. vehiclegps—GPS 数据模块

通过这个模块可以获取 Pixhawk 的 GPS 数据，它是通过订阅 uORB 话题“vehicle_gps”

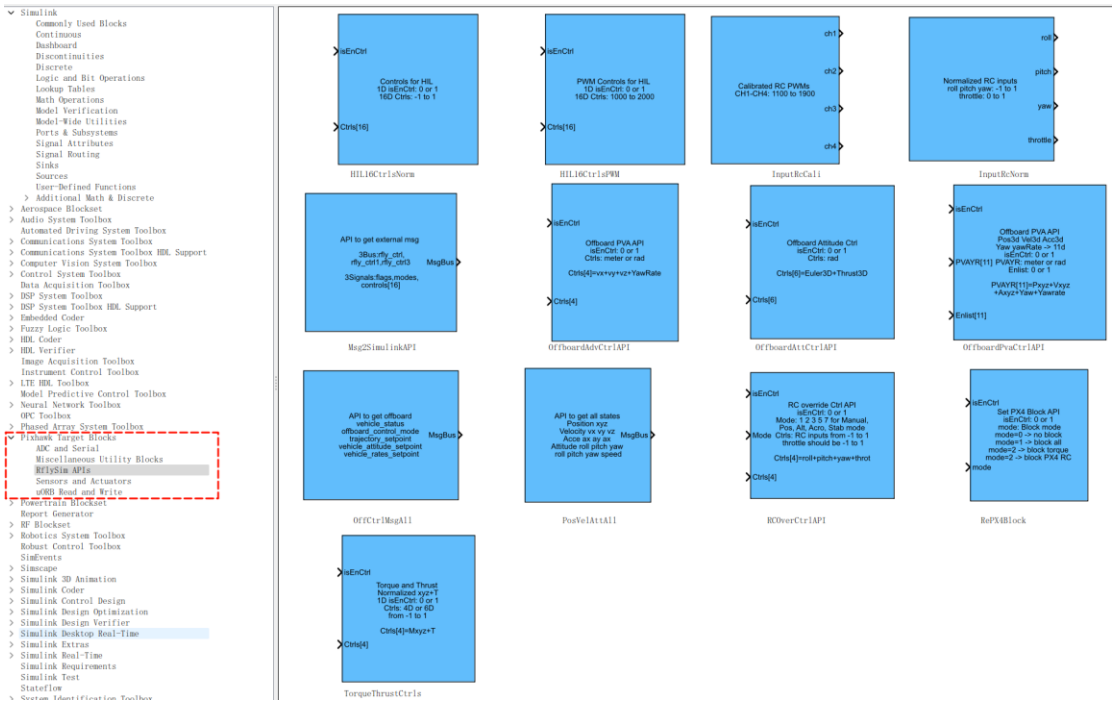
实现的，所以在实际运行时需要保证 Pixhawk 上插入 GPS 模块才能获取正确数据。如下图所示，模块提供了采样率的设置框，以及可选择的姿态选项。各选项的含义如下：

- Latitude：(int32) 全局坐标给出在 1e-7 度。
- Longitude：(int32) 全局坐标给出在 1e-7 度。
- Altitude：(int32)在 MSL（平均海平面）以上 1e-3 米（毫米）。
- GPS TS：(double)时间戳（GPS 格式的微秒），这是来自 GPS 模块的时间戳。
- Velocity：(single)GPS 地面速度，单位为米/秒。
- GPS Fix Type：(uint8)0-1=NO fix,2=2D fix,3=3D fix。
- Num Satellites：(uint8)用于计算的卫星数量。

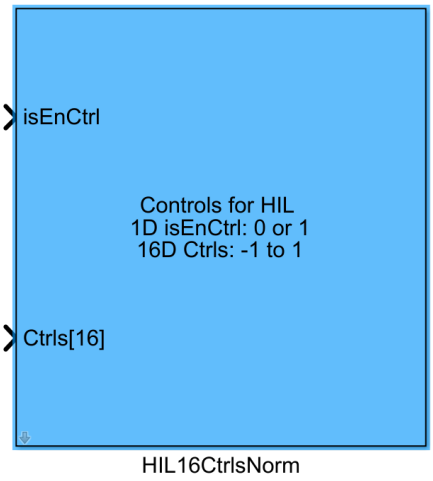
更多信息可点击对话框的“帮助”按钮查看。



2.4. RflySim APIs—RflySim 平台自定义接口库



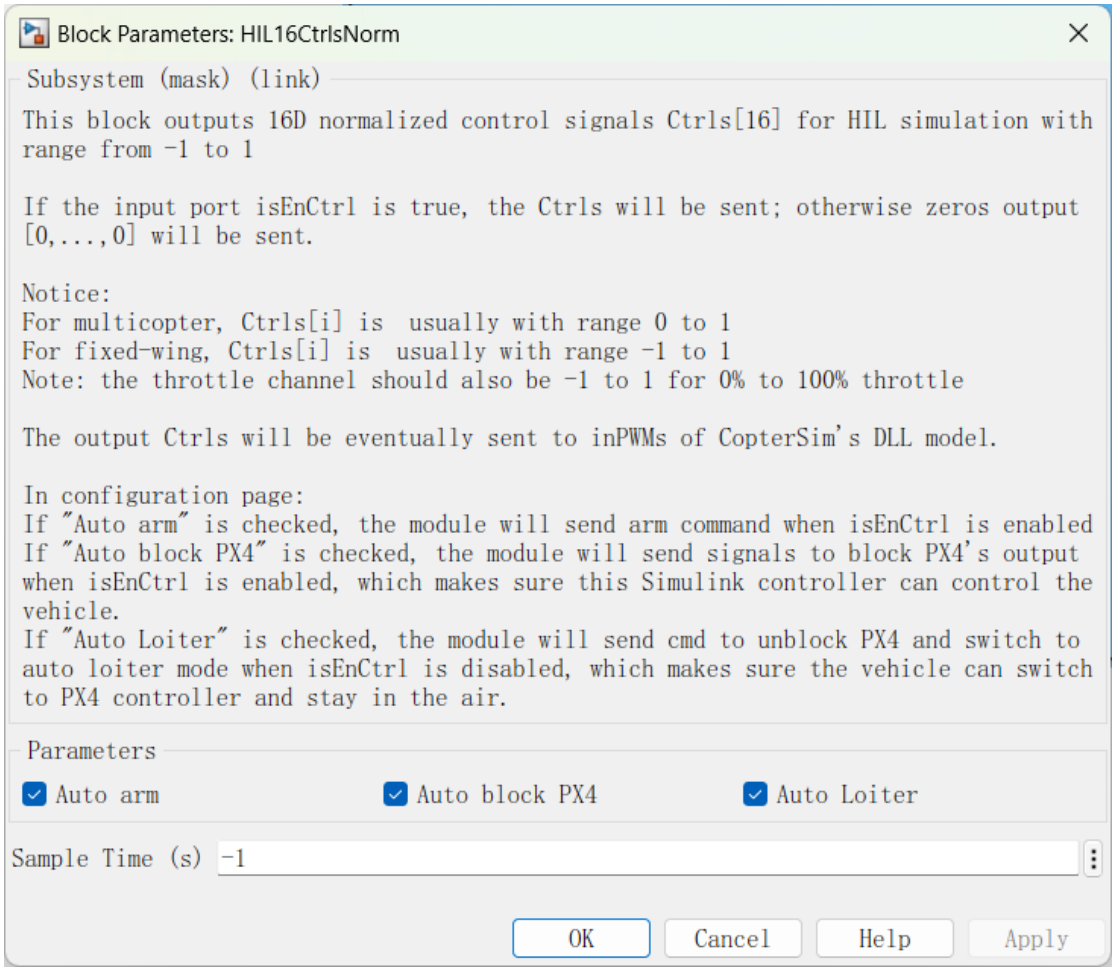
2.4.1. HIL16CtrlsNorm—硬件在环 16 维归一化控制信号模块



isEnCtr：当 isEnCtrl 端口输入为 true 时，则发送 CtrlS[16]端口数据；否则将发送[0，...，0]。

CtrlS[16]：为该模块输入 16 维归一化控制信号，用于 HIL 仿真，范围为-1 到 1。对于多旋翼模型，CtrlS[i]通常在 0 到 1 的范围内；对于固定翼模型，CtrlS[i]通常是-1 到 1 的范围。其中 0%至 100%的油门，油门通道也应为-1 至 1。

输出的 CtrlS[16]最终将被发送到 CopterSim 的 DLL 模型的 inPWM 接口中。

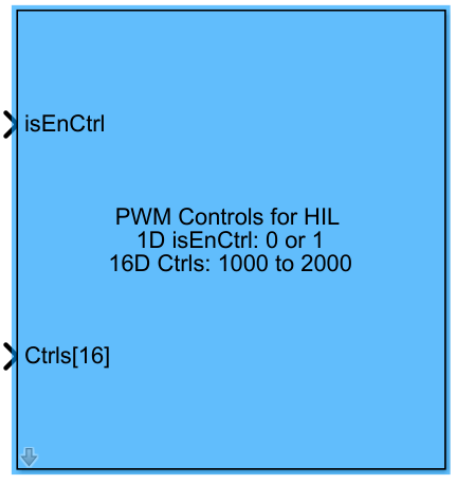


双击打开本模型的配置页面后，其具体定义如下：

- 若勾选 Auto arm，则启用 isEnCtrl 时，模块将发送解锁指令。
- 若勾选 Auto block PX4，则模块将发送信号屏蔽 PX4 的输出。当 isEnCtrl 接口输入为 true 时，将使用 Simulink 控制器控制载具。
- 若勾选 Auto Loiter，模块将发送命令解除对 PX4 输出的屏蔽，并切换到自动 Loiter 模式，当 isEnCtrl 接口输入为 false 时切换到 Loiter 模式，从而确保载具可以切换到 PX4 控制器并保持在空中。
- Sample Time(s)：采样时间。

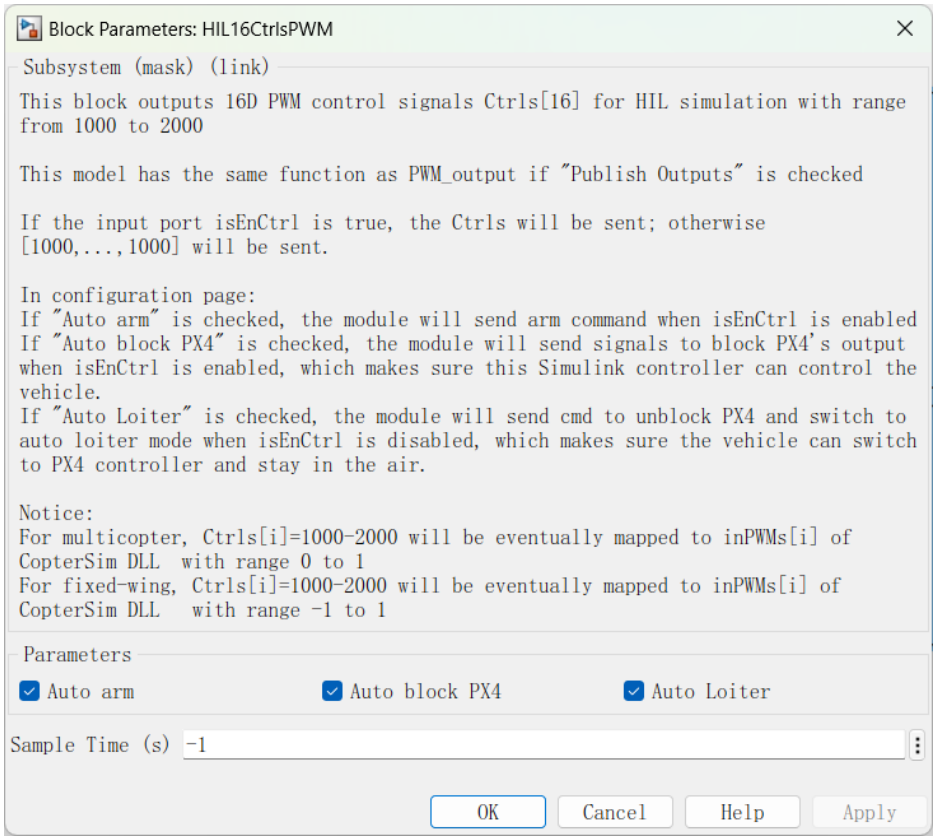
本模块相关例程可见：0.ApiExps\16.CtrlsSingalsAPI\Readme.pdf

2.4.2. HIL16CtrlsPWM—硬件在环 16 维 PWM 控制信号模块



isEnCtrl：此端口输入为 true 时，则发送 CtrlS[16]端口数据；否则将发送[1000，...，1000]。

CtrlS[16]：为该模块输入 16 维 PWM 控制信号，用于 HIL 仿真，范围为 1000 到 2000。该模块功能等同于将 PWM_output—电机 PWM 模块中的“PWM_output”勾选。对于多旋翼模型，CtrlS[i]=1000~2000 将对应匹配到 CopterSim 中 DLL 文件的 0~1；对于固定翼模型，CtrlS[i] =1000~2000 将对应匹配到 CopterSim 中 DLL 文件的-1~1。



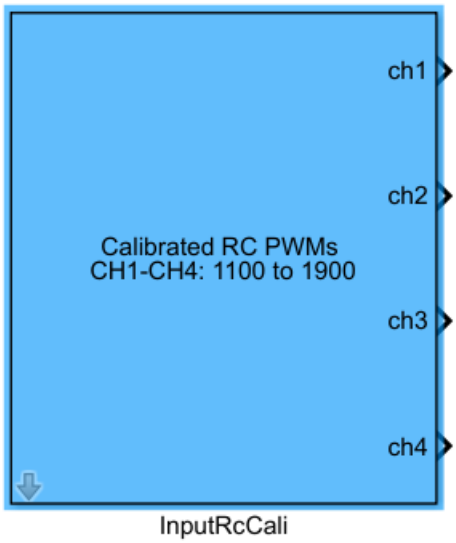
双击打开本模型的配置页面后，其具体定义如下：

- 若勾选 Auto arm，当 isEnCtrl 端口输入为 true 时，模块将发送解锁指令。
- 若勾选 Auto block PX4，则模块将发送信号屏蔽 PX4 的输出。当 isEnCtrl 接口输入为 true 时，将使用 Simulink 控制器控制载具。
- 若勾选 Auto Loiter，模块将发送命令解除对 PX4 输出的屏蔽，并切换到自动 Loiter 模式，当 isEnCtrl 接口输入为 false 时切换到 Loiter 模式，从而确保载具可以切换到 PX4 控制器并保持在空中。
- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\16.CtrlsSingalsAPI\Readme.pdf

2.4.3. InputRcCali—校准遥控器 PWM 模块

将遥控器的数据映射到 1100~1900 数据范围中。



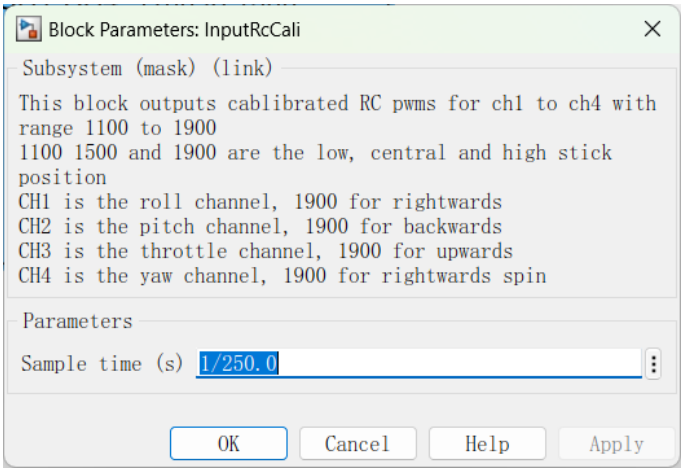
CH1：为输出滚转通道，范围为：1100~1900，1900 表示向最右方飞行；

CH2：为输出俯仰通道，范围为：1100~1900，1900 表示向最下方飞行；

CH3：为输出油门通道，范围为：1100~1900，1900 表示向最上方飞行；

CH4：为输出偏航通道，范围为：1100~1900，1900 表示向右方转动；

双击打开本模型的配置页面后，其具体定义如下：



- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\15.InputSourceAPI\Readme.pdf

2.4.4. InputRcNorm—遥控器信号归一化模块

将遥控器的数据通过归一化处理映射到的-1~1 范围数据中。

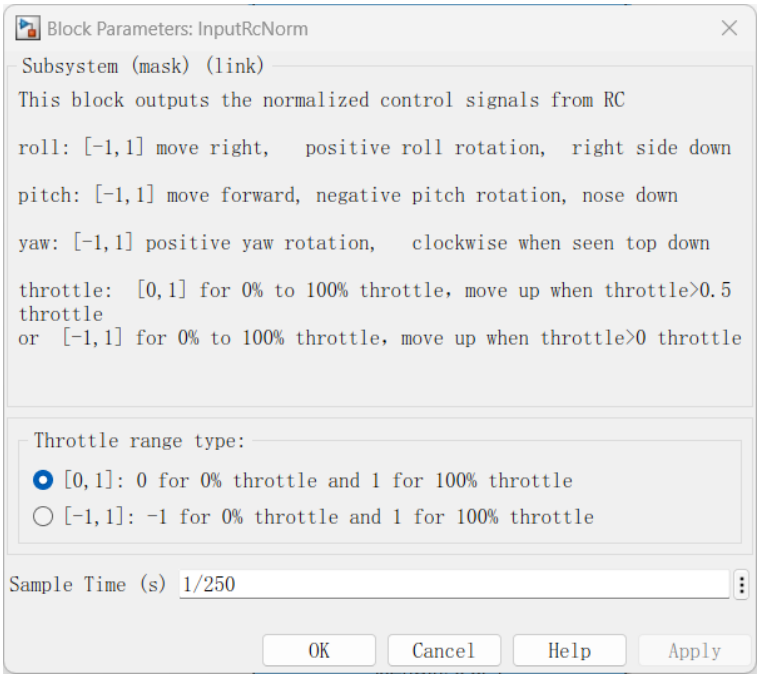
CH1：为输出滚转通道，范围为：-1~1，1 表示向最右方飞行；

CH2：为输出俯仰通道，范围为：-1~1，1 表示向最下方飞行；

CH3：为输出油门通道，范围为：0~1 或-1~1，1 表示向最上方飞行；

CH4：为输出偏航通道，范围为：-1~1，1 表示向右方转动；

双击打开本模型的配置页面后，其具体定义如下：



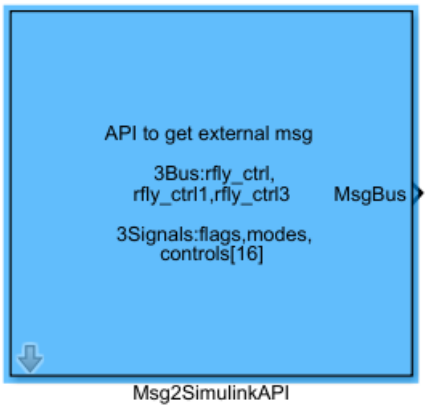
- 若勾选[0,1]：则 CH3 端口输出数据范围为 0~1。
- 若勾选[-1,1]：则 CH3 端口输出数据范围为-1~1。
- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\15.InputSourceAPI\Readme.pdf

2.4.5. Msg2SimulinkAPI—rfly_ctrl 消息不同 ID 数据输出模块

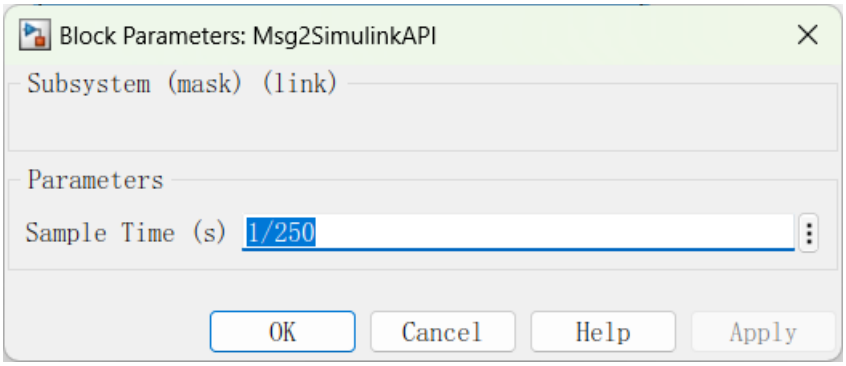
本模块中是将 rfly_ctrl 消息的不同 ID 输出的数据进行封装输出，rfly_ctrl 消息具体定义

可查看：rfly_ctrl.msg。



MsgBus：输出 48 维数据，分别为 rfly_ctrl 消息 ID 为 rfly_ctrl、rfly_ctrl1、rfly_ctrl2 的数据。

双击打开本模型的配置页面后，其具体定义如下：

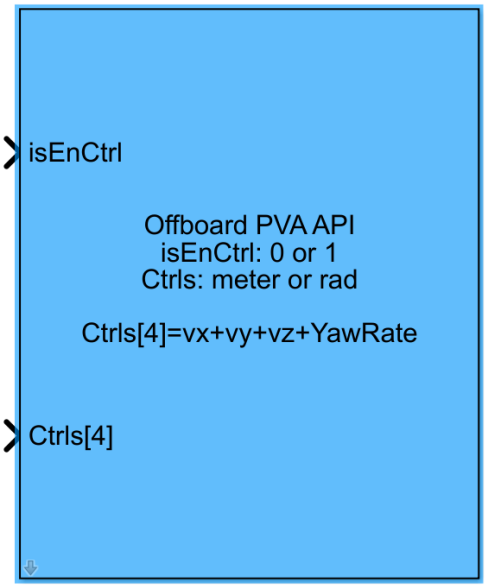


- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\9.PX4CtrlExternalTune\Readme.pdf

2.4.6. OffboardAdvCtrlAPI—Offboard 模式高级控制模块

本模块可使能载具进入 Offboard 模式，通过发送的指令(可以是：x、y、z、vx、vy、vz、ax、ay、az、yaw、yawrate)控制载具在 Offboard 模式下运动。



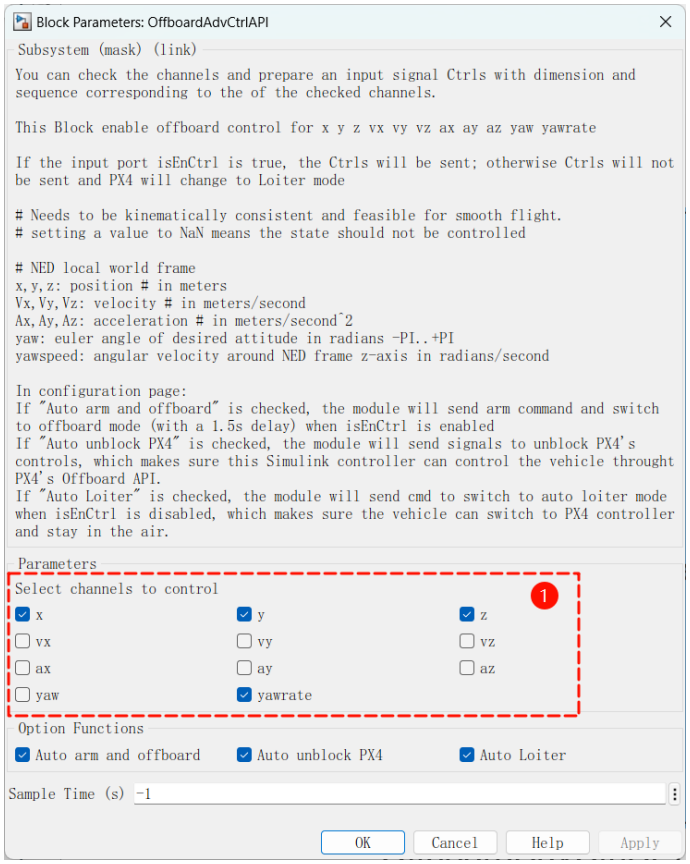
OffboardAdvCtrlAPI

isEnCtr：当 isEnCtrl 端口输入为 true 时，则发送 CtrlS[*]端口控制数据；否则不发送且 PX4 将进入 Loiter 模式。

Ctrls[*]：为该模块输入*维归一化控制信号，用于 Offboard 模式的控制指令。具体协议如下：

```
NED 坐标系下：
x、y、z：位置控制（m）
Vx、Vy、Vz：速度控制（m/s）
Ax、Ay、Az：加速度控制（m/s^2）
Yaw：偏航控制（rad，范围：-pi~pi）
Yawrate：偏航速度控制（rad/s）

% 为了平稳飞行，需要保持运动学上的一致性和可行性。
% 将值设置为 NaN 表示不应控制状态
```

双击打开本模型的配置页面后，其具体定义如下：

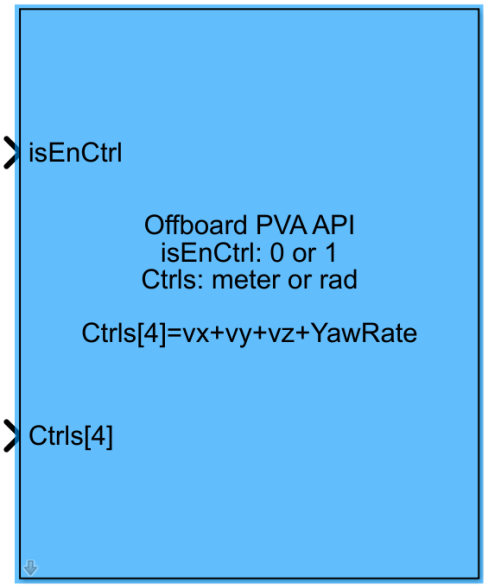


- ①：控制通道选择区域可以根据个人需求自定义选择不同的 Offboard 模式的控制指令。
- 若勾选 Auto arm，当 isEnCtrl 端口输入为 true 时，模块将发送解锁指令。
- 若勾选 Auto block PX4，则模块将发送信号屏蔽 PX4 的输出。当 isEnCtrl 接口输入为 true 时，将使用 Simulink 控制器控制载具。
- 若勾选 Auto Loiter，模块将发送命令解除对 PX4 输出的屏蔽，并切换到自动 Loiter 模式，当 isEnCtrl 接口输入为 false 时切换到 Loiter 模式，从而确保载具可以切换到 PX4 控制器并保持在空中。
- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\17.OffboardCtrlsAPI\Readme.pdf

2.4.7. OffboardAttCtrlAPI—Offboard 模式姿态控制模块

本模块可使能载具进入 Offboard 模式，通过发送的姿态指令(可以是：欧拉角度和四元数)控制载具在 Offboard 模式下运动。

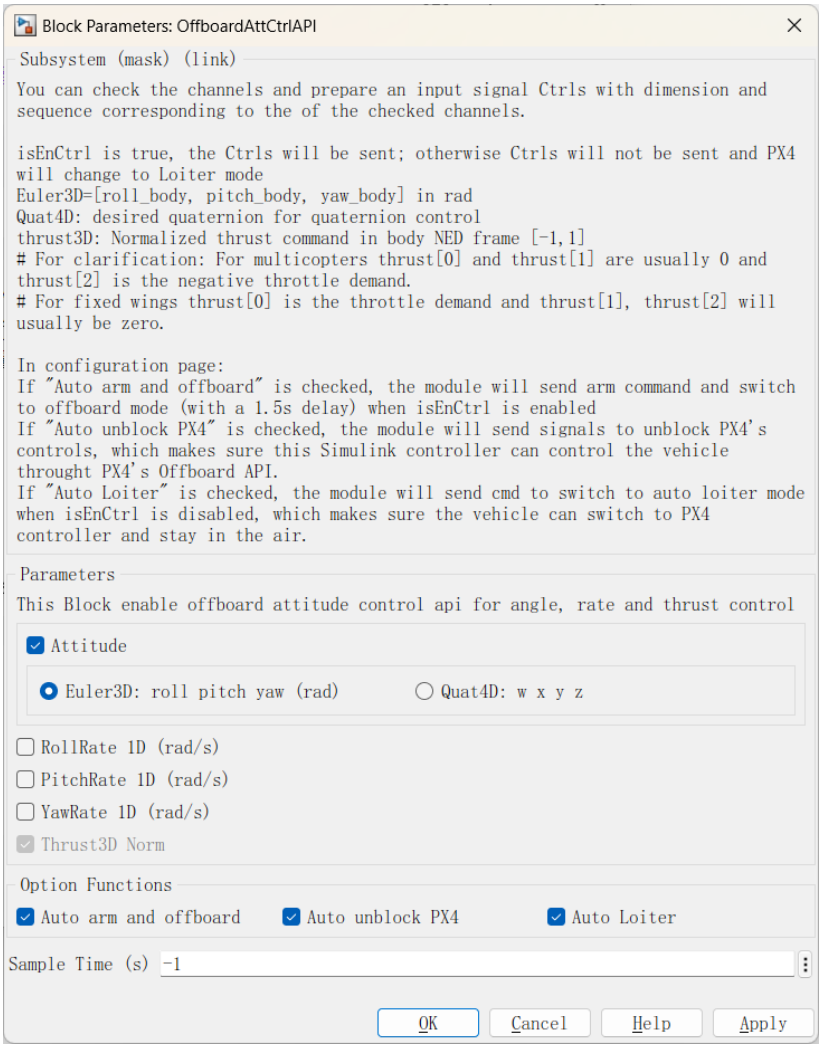


isEnCtr：当 isEnCtrl 端口输入为 true 时，则发送 CtrlS[*]端口控制数据；否则不发送且 PX4 将进入 Loiter 模式。对于多旋翼模型，CtrlS[i]=1000~2000 将对应匹配到 CopterSim 中 DLL 文件的 0~1；对于固定翼模型，CtrlS[i]=1000~2000 将对应匹配到 CopterSim 中 DLL 文件的-1~1。

Ctrls[*]：为该模块输入*维控制信号，用于 Offboard 模式的姿态控制指令。该控制信号可以是：

```
机体坐标系下：
三维欧拉角：[roll_body, pitch_body, yaw_body]，单位 rad
或
四元数：[w, x, y, z]
或
RollRate 1D (rad/s)
PitchRate 1D (rad/s)
YawRate 1D (rad/s)

Thrust3D Norm 默认选择，但：对于多旋翼，Thrust[0]和 Thrust[1]通常为 0，Thrust[2]是负油门请求。
对于固定翼，Thrust[0]是油门需求，Thrust[1]、Thrust[2]将通常为 0。
```

双击打开本模型的配置页面后，其具体定义如下：



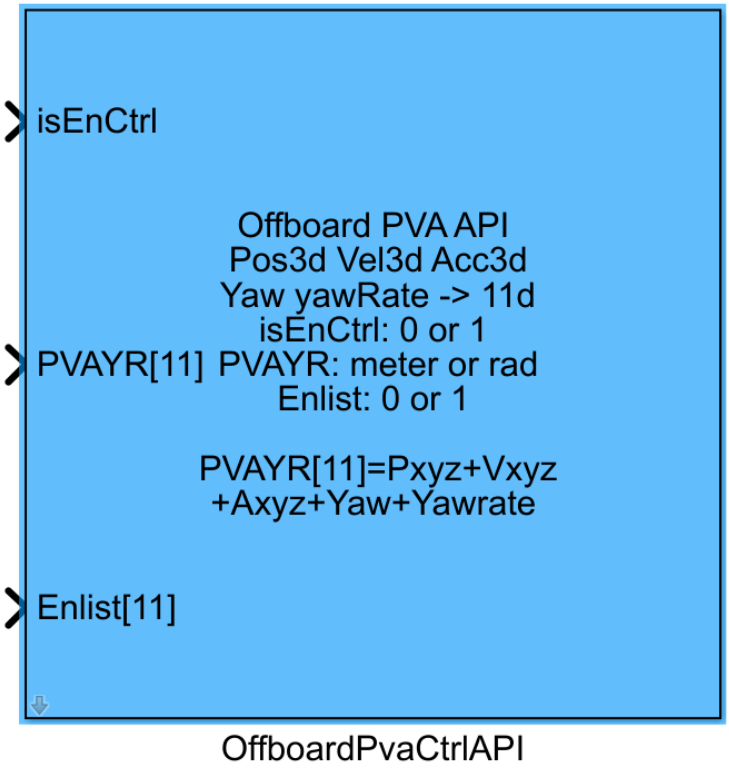
- Attitude 区域：该区域可以选择控制指令为欧拉角或四元数。
- RollRate：若勾选则表示通过滚转角速率(rad/s)进行控制。
- PitchRate：若勾选则表示通过俯仰角速率(rad/s)进行控制。
- YawRate：若勾选则表示通过偏航角速率(rad/s)进行控制。
- 若勾选 Auto arm，当 isEnCtrl 端口输入为 true 时，模块将发送解锁指令。
- 若勾选 Auto block PX4，则模块将发送信号屏蔽 PX4 的输出。当 isEnCtrl 接口输入为 true 时，将使用 Simulink 控制器控制载具。
- 若勾选 Auto Loiter，模块将发送命令解除对 PX4 输出的屏蔽，并切换到自动 Loiter 模式，当 isEnCtrl 接口输入为 false 时切换到 Loiter 模式，从而确保载具可以切换到 PX4 控制器并保持在空中。
- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\17.OffboardCtrlsAPI\Readme.pdf

2.4.8. OffboardPvaCtrlAPI—Offboard 模式下控制方式切换控制模块

本模块可使能载具进入 Offboard 模式，通过发送的指令(可以是：位置控制、速度控制、

加速度控制)控制载具在 Offboard 模式下运动，并且可在位置控制、速度控制、加速度控制中切换不同的控制方式。



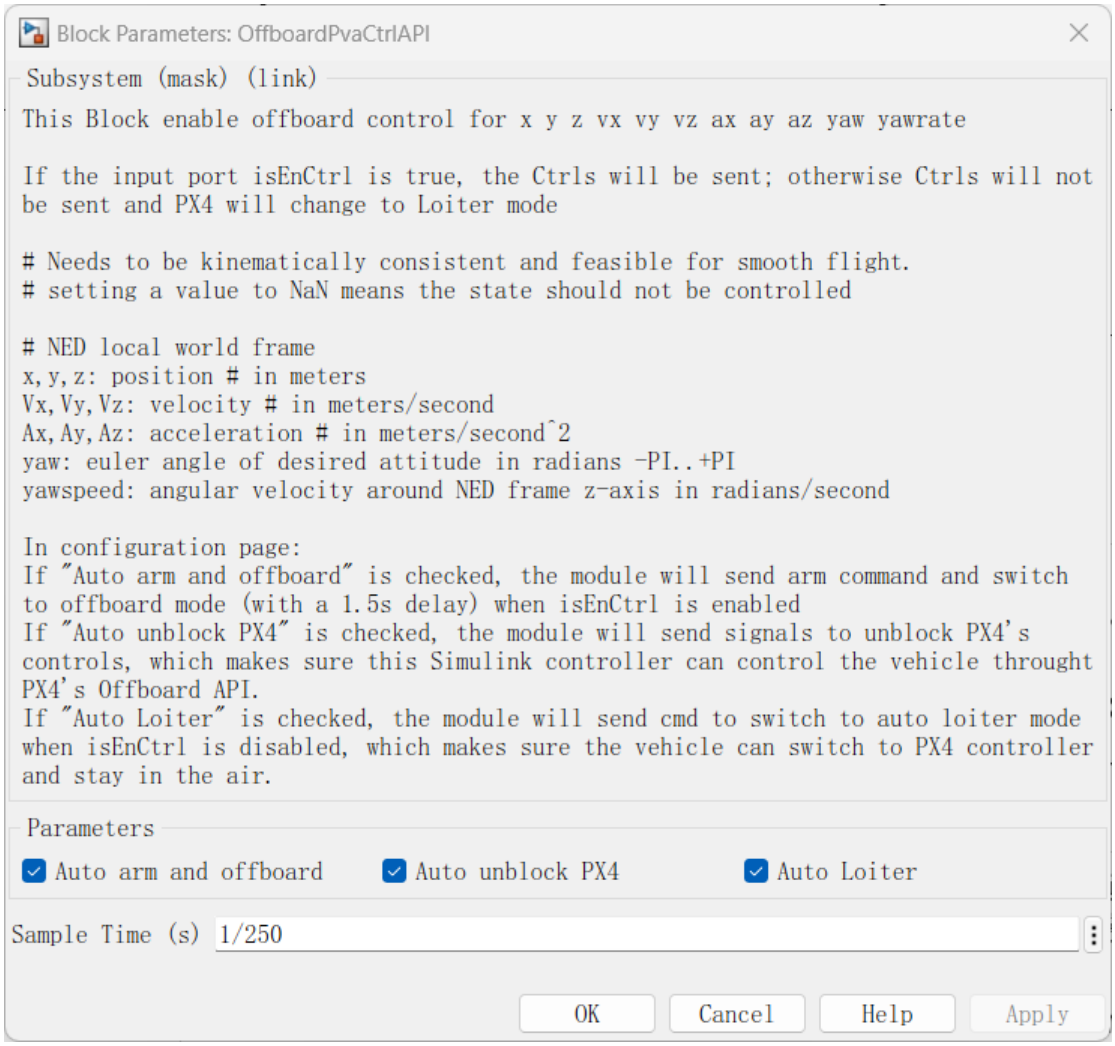
isEnCtr：当 isEnCtrl 端口输入为 true 时，则发送控制数据并进入 Offboard；否则不发送且 PX4 将进入 Loiter 模式。当指令从 false 到 true 时，会自动解锁并进入 Offboard。

PVAYR[11]：可输入 11 维 single 类型控制信号，用于 Offboard 模式的控制指令。具体协议如下：

```
NED 坐标系下：
x、y、z：位置控制（m）
Vx、Vy、Vz：速度控制（m/s）
Ax、Ay、Az：加速度控制（m/s^2）
Yaw：偏航控制（rad，范围：-pi~pi）
Yawrate：偏航速度控制（rad/s）

% 为了平稳飞行，需要保持运动学上的一致性和可行性。
% 将值设置为 NaN 表示不应控制状态
```

EnList[11]：可输入 11 维 boolean 类型的向量，每一维对应上述的具体的控制信号，当某一维置于 true，则 PX4 会响应本通道控制量。例如，Enlist11d=[true,true,true, 0,...]表示使能 xyz 的 pos 位置控制量。

双击打开本模型的配置页面后，其具体定义如下：

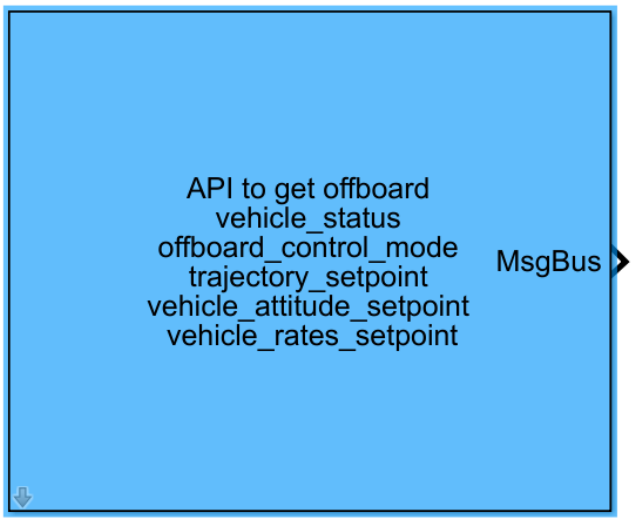


- 若勾选 Auto arm，当 isEnCtrl 端口输入为 true 时，模块将发送解锁指令。
- 若勾选 Auto block PX4，则模块将发送信号屏蔽 PX4 的输出。当 isEnCtrl 接口输入为 true 时，将使用 Simulink 控制器控制载具。
- 若勾选 Auto Loiter，模块将发送命令解除对 PX4 输出的屏蔽，并切换到自动 Loiter 模式，当 isEnCtrl 接口输入为 false 时切换到 Loiter 模式，从而确保载具可以切换到 PX4 控制器并保持在空中。
- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\17.OffboardCtrlsAPI\Readme.pdf

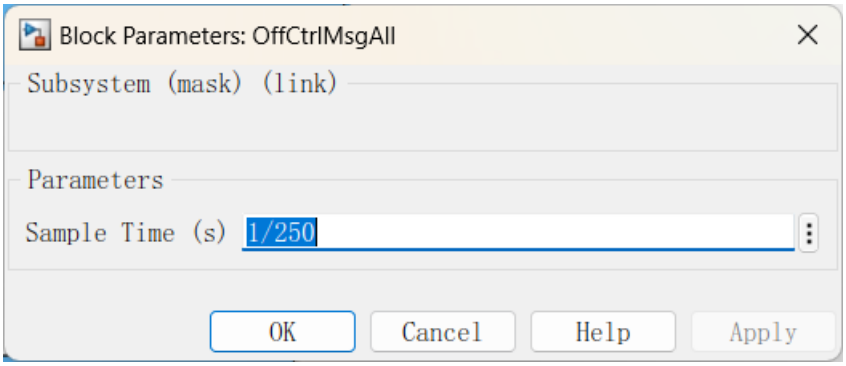
2.4.9. OffCtrlMsgAll—Offboard 模式相关控制消息

本模块中订阅了关于 Offboard 控制相关的数据，主要包括以下 uORB 消息：vehicle_status、offboard_control_mode、trajectory_setpoint、vehicle_attitude_setpoint、vehicle_rates_setpoint。每个消息内字段的详细定义，可以去【RflySim 安装目录】\Firmware\msg 目录下查看。



MsgBus：输出 vehicle_status、offboard_control_mode、trajectory_setpoint、vehicle_attitude_setpoint、vehicle_rates_setpoint 的 uORB 消息中定义的数据，

双击打开本模型的配置页面后，其具体定义如下：

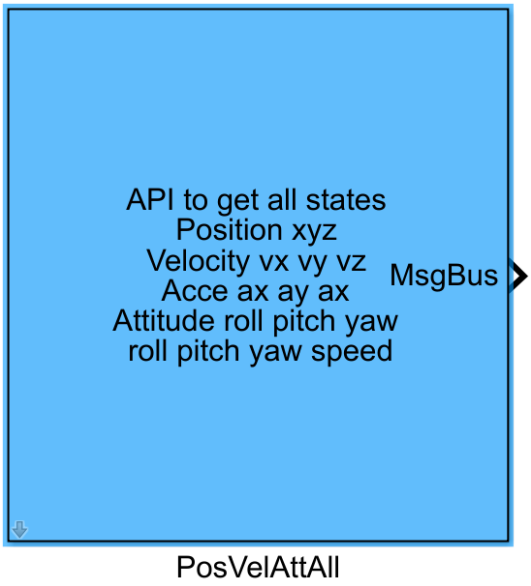


- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\17.OffboardCtrlsAPI\Readme.pdf

2.4.10. PosVelAttAll—载具状态量获取模块

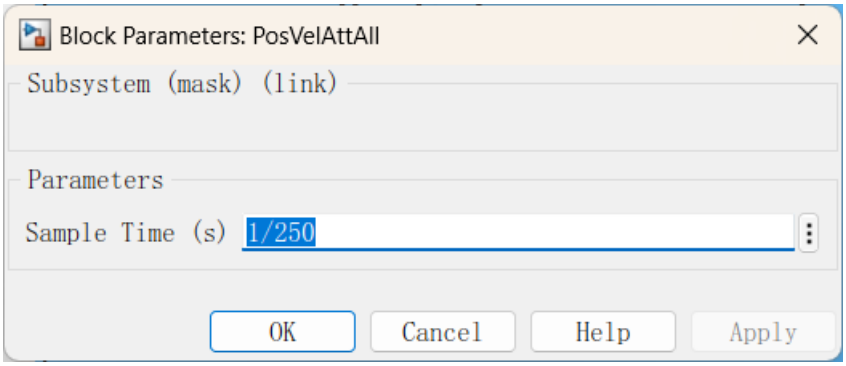
该模块订阅了 vehicle_local_position、vehicle_attitude、vehicle_angular_velocity 三个 uORB 消息，这三个消息中的数据均为 PX4 中滤波之后的数据，每个消息内字段的详细定义，可以去【RflySim 安装目录】\Firmware\msg 目录下查看。可获取到载具的各种运动状态量。



MsgBus：输出载具滤波后的状态量，具体协议如下：

```
    # Acceleration in NED frame
    float32 ax        # North velocity derivative in NED earth-fixed frame, (metres/sec^2)
    float32 ay        # East velocity derivative in NED earth-fixed frame, (metres/sec^2)
    float32 az        # Down velocity derivative in NED earth-fixed frame, (metres/sec^2)
    uint64 timestamp            # time since system start (microseconds)
    float32[4] q                # Quaternion rotation from the FRD body frame to the NED earth
 frame
    float32 Roll     # Roll in FRD coordinate system (obtained by quaternion conversion, unit:
 rad)
    float32 Pitch    # Pitch in FRD coordinate system (obtained by quaternion conversion, uni
t: rad)
    float32 Yaw     # Yaw in FRD coordinate system (obtained by quaternion conversion, unit: r
ad)
    float32[3] xyz     # Bias corrected angular velocity about the FRD body frame XYZ-axis in
 rad/s
```

双击打开本模型的配置页面后，其具体定义如下：



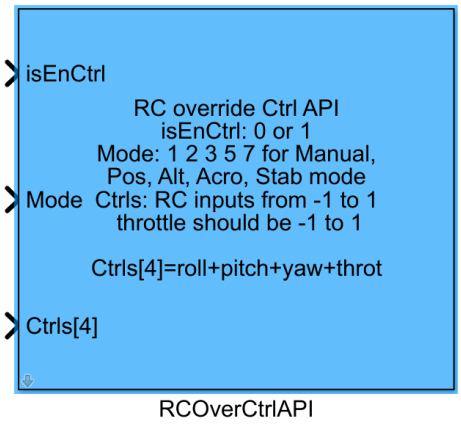
- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\11.StateDataGatAPI\Readme.pdf

2.4.11. RCOverCtrlAPI—遥控器手动控制信号的覆盖模块

本模块通过重发 manual_control_setpoint 的消息，来实现 RC 遥控器手动控制信号的覆

盖功能。



isEnCtr：当 isEnCtrl 端口输入为 true 时，则发送控制数据并进入 Offboard；否则不发送且 PX4 将进入 Loiter 模式。当指令从 false 到 true 时，会自动解锁并进入 Offboard。

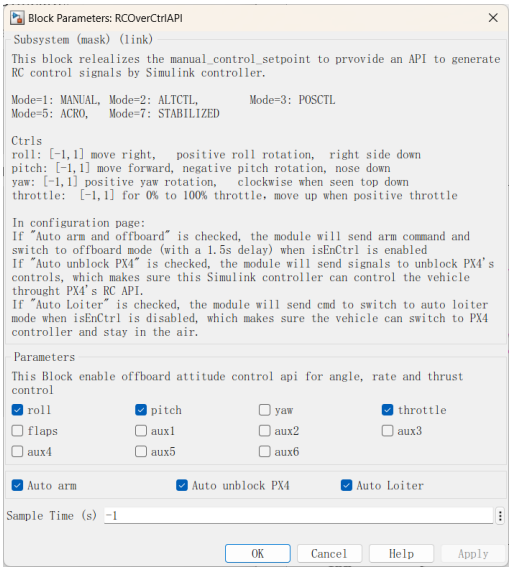
Mode：可输入 uint8 类型数据实现不同的控制模式，具体协议如下：

```
Mode=1：手动模式
Mode=2：定高模式
Mode=3：定点模式
Mode=5：特技模式
Mode=6：自稳模式
```

Ctrls[*]：可输入*维 single 类型的向量，对应的控制信号，具体协议如下：

```
roll：滚转通道，数据范围为[-1，1]向右移动，正向滚动旋转，右侧向下。
pitch：俯仰通道，数据范围为[-1，1]向前移动，负俯仰旋转，机头向下。
yaw：偏航通道，数据范围为[-1，1]正偏航旋转，从上往下看是顺时针方向。
throttle：油门通道，数据范围为[-1，1]，对应 0%到 100%的油门，当正油门时向上移动
flaps：襟翼的位置，数据范围[-1，1]，开关/旋钮/摇杆
aux1：辅助通道 1
aux2：辅助通道 1
aux3：辅助通道 1
aux4：辅助通道 1
aux5：辅助通道 1
aux6：辅助通道 1
```

双击打开本模型的配置页面后，其具体定义如下：

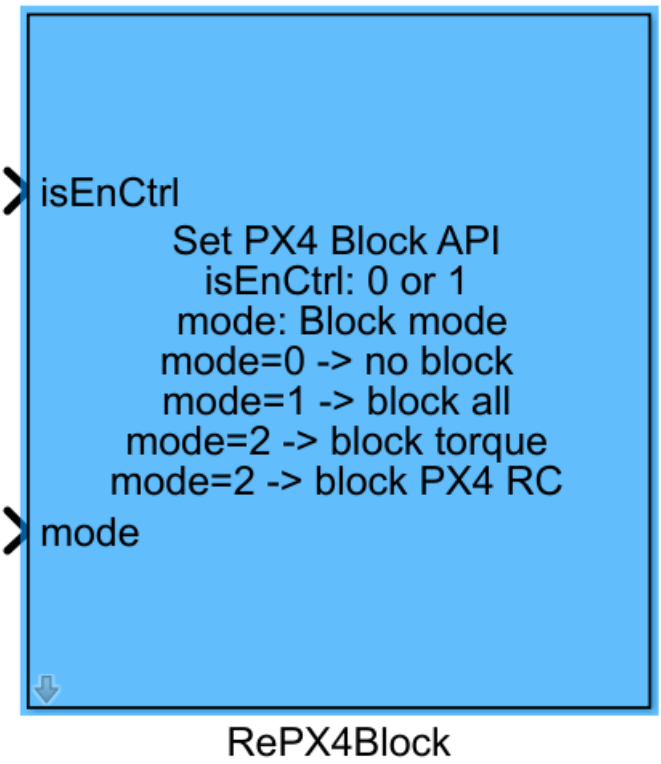


- Parameters：可自定义选择 Offboard 模式下姿态控制的不同通道，具体协议见上方。
- 若勾选 Auto arm，当 isEnCtrl 端口输入为 true 时，模块将发送解锁指令。
- 若勾选 Auto block PX4，则模块将发送信号屏蔽 PX4 的输出。当 isEnCtrl 接口输入为 true 时，将使用 Simulink 控制器控制载具。
- 若勾选 Auto Loiter，模块将发送命令解除对 PX4 输出的屏蔽，并切换到自动 Loiter 模式，当 isEnCtrl 接口输入为 false 时切换到 Loiter 模式，从而确保载具可以切换到 PX4 控制器并保持在空中。
- Sample Time(s)：采样时间。

本模块相关例程可见：0.ApiExps\15.InputSourceAPI\Readme.pdf

2.4.12. RePX4Block—在线屏蔽 PX4 输出模块

本模块可实现在线屏蔽 PX4 输出的，该控制切换功能，是一键安装脚本在部署固件时，自动新增的接口。



isEnCtr：当 isEnCtrl 端口输入为 true 时，则发送 Mode 中的数据进行自定义屏蔽输出，反则否。

mode：可输入 uint8 类型数据实现不同的控制模式，具体协议如下：

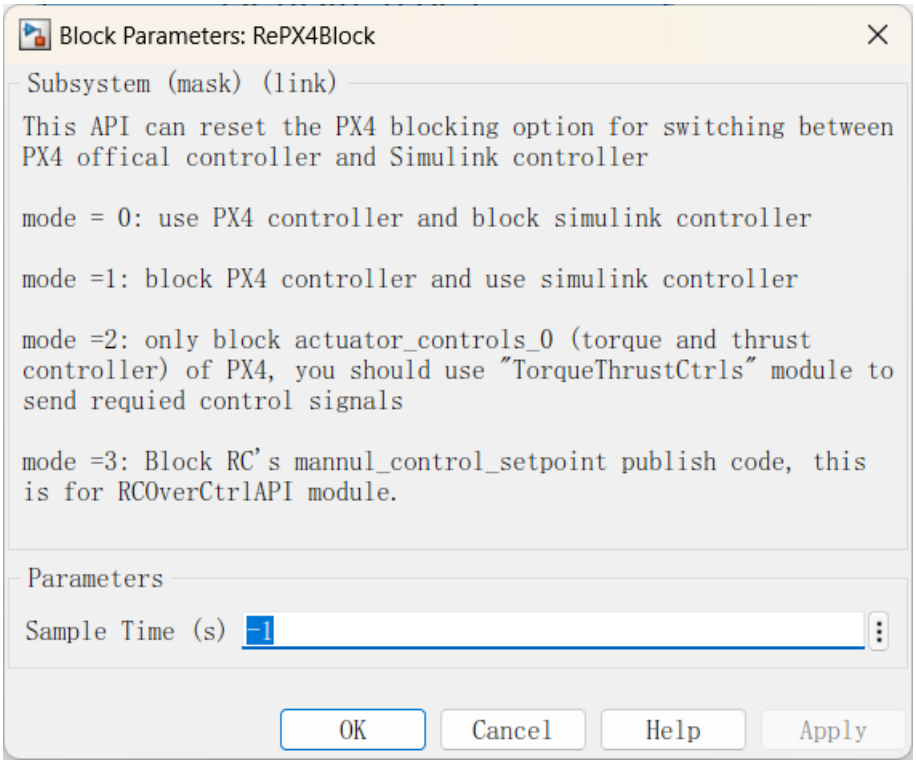
Mode0：不屏蔽模式，**PX4** 正常接收：**actuator_outputs**（软硬件在环仿真）、**actuator_controls_0**（力和力矩接口）、正常电机控制量

Mode1：完全屏蔽模式，**SIL**、**HIL** 仿真时：接收 **actuator_outputs_rfly** 消息（不再接收 **PX4** 的 **actuator_outputs**），需搭配 **HIL16CtrlsPWM** 或 **HIL16CtrlsNorm** 来发布消息；实飞时： 接收 **PWM_output** 和 **Aux_output** 模块控制（注意，不支持 **DShot** 等协议），屏蔽 **PX4** 底层电机舵机控制代码。

Mode2：中间力和力矩层屏蔽模式,，**SIL**、**HIL** 仿真+真机：接收 **actuator_controls_rfly** 的力和力矩控制量（不再接收 **PX4** 的 **actuator_controls_0**），需要搭配 **TorqueThrustCtrls** 模块来发送力和力矩控制量。本模块同时支持真机或仿真，也支持 **DShot**、**UartESC** 等特殊电调协议，兼容多旋翼、固定翼、无人车等多种载具。

Mode3：屏蔽 **PX4** 遥控器发出的 **manual_control_setpoint** 消息，通过 **RCOverCtrlAPI** 接口可以替换遥控器信号，实现更多控制功能。这个屏蔽接口，能够防止 **Simulink** 发送的遥控器信号和 **PX4** 发出的相冲突，导致控制抖动。

双击打开本模型的配置页面后，其具体定义如下：

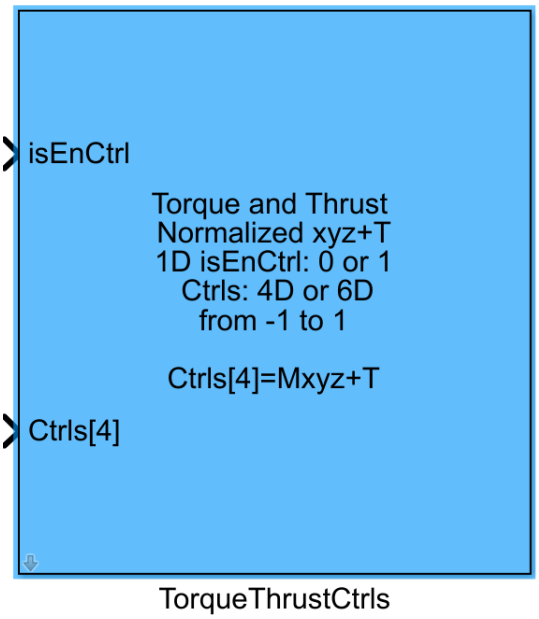


- Sample Time(s)：采样时间。

本模块相关例程可见：2.AdvExps\e0_AdvApiExps\5.RepPX4Func\Readme.pdf

2.4.13. TorqueThrustCtrls—力和力矩控制信号模块

发送力和力矩（归一化后的）控制量 actuator_controls_0 消息（或 PX4 v1.14 版本中的 vehicle_torque/thrust_setpoint），并经过 PX4 自带的 Mixer 混控器计算电机转速后，驱动给 CopterSim 的 inPWMs 输入，控制无人机的运动。真机实飞时，则直接驱动飞机进行飞行。



isEnCtrl：若输入为 true，则发送 Ctrls 控制量，反则发送[0,0,…0,0]。

Ctrls[*]：输入 single 数据类型的控制量，具体协议如下：

第 1 控制通道：对于 PX4 版本为 v1.7~v1.13 版本,该模块将发送 actuator_controls_0 的 uORB 消息。对于 PX4 版本为 v1.14 版本，该模块将发送 vehicle_torque_setpoint 和 vehicle_torque_setpoint 的 uORB 消息。

方式一：Ctrls[4]

X：机体坐标系下绕 X 轴力矩

Y：机体坐标系下绕 Y 轴力矩

Z：机体坐标系下绕 Z 轴力矩

Thrust 1D：机体坐标系下沿 Z 轴推力，大于 0 表示向上飞行。

方式二：Ctrls[6]

X：机体坐标系下绕 X 轴力矩

Y：机体坐标系下绕 Y 轴力矩

Z：机体坐标系下绕 Z 轴力矩

Thrust 3D：机体坐标系下沿 X、Y、Z 轴推力，沿 Z 轴大于 0 表示向上飞行。

使能第 2 通道（针对于 VOTLs）：对于 PX4 版本为 v1.7~v1.13 版本,该模块将发送 actuator_controls_1 的 uORB 消息。对于 PX4 版本为 v1.14 版本，该模块将发送 vehicle_torque_setpoint1 和 vehicle_torque_setpoint1 的 uORB 消息。

方式一：Ctrls[8]

X：机体坐标系下绕 X 轴力矩

Y：机体坐标系下绕 Y 轴力矩

Z：机体坐标系下绕 Z 轴力矩

Thrust 1D：机体坐标系下沿 Z 轴推力，大于 0 表示向上飞行。

X1：机体坐标系下绕 X1 轴力矩

Y1：机体坐标系下绕 Y1 轴力矩

Z1：机体坐标系下绕 Z1 轴力矩

Thrust1 1D：机体坐标系下沿 Z1 轴推力，大于 0 表示向上飞行。

方式二：Ctrls[12]

X：机体坐标系下绕 X 轴力矩

Y：机体坐标系下绕 Y 轴力矩

Z：机体坐标系下绕 Z 轴力矩

Thrust 3D：机体坐标系下沿 X、Y、Z 轴推力，沿 Z 轴大于 0 表示向上飞行。

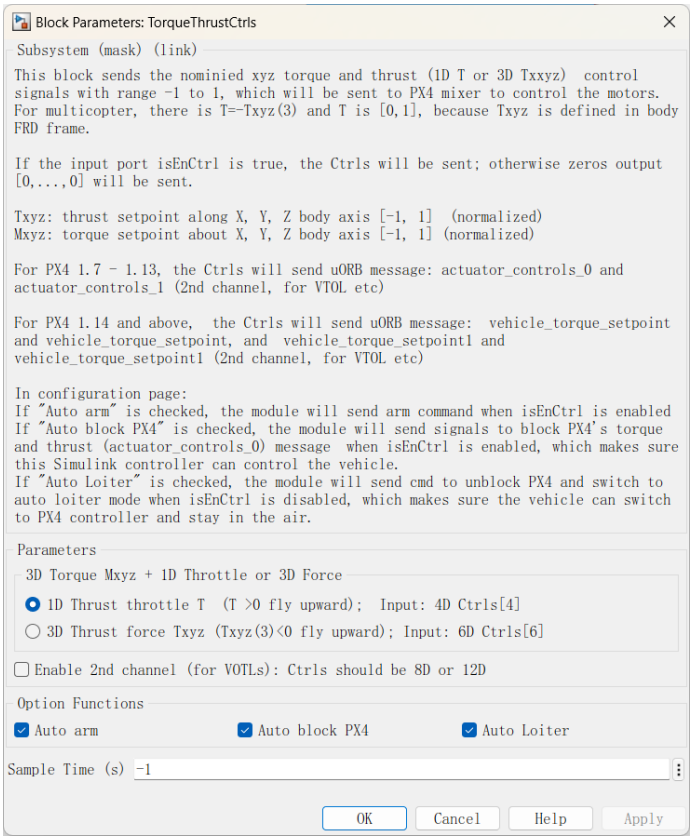
X1：机体坐标系下绕 X1 轴力矩

Y1：机体坐标系下绕 Y1 轴力矩

Z1：机体坐标系下绕 Z1 轴力矩

Thrust1 3D：机体坐标系下沿 X1、Y1、Z1 轴推力，沿 Z1 轴大于 0 表示向上飞行。

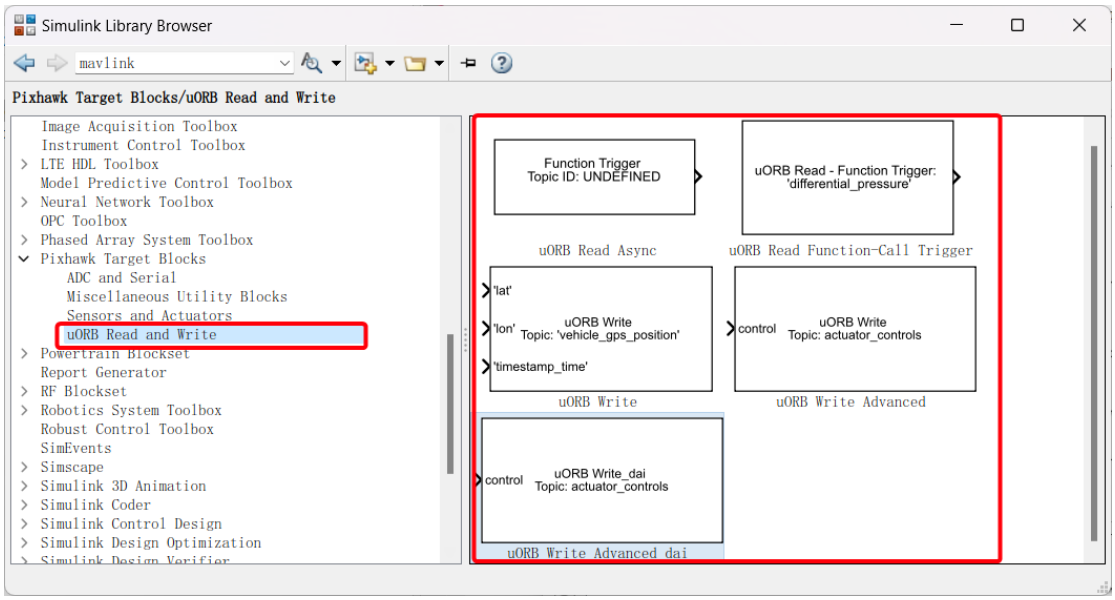
双击打开本模型的配置页面后，其具体定义如下：



- Parameters：可自定义选择控制信号中力矩的维度，具体定义见上述。
- Enable 2nd channel：若勾选，则表示开启第 2 控制通道。
- 若勾选 Auto arm，当 isEnCtrl 端口输入为 true 时，模块将发送解锁指令。
- 若勾选 Auto block PX4，则模块将发送信号屏蔽 PX4 的输出。当 isEnCtrl 接口输入为 true 时，将使用 Simulink 控制器控制载具。
- 若勾选 Auto Loiter，模块将发送命令解除对 PX4 输出的屏蔽，并切换到自动 Loiter 模式，当 isEnCtrl 接口输入为 false 时切换到 Loiter 模式，从而确保载具可以切换到 PX4 控制器并保持在空中。
- Sample Time(s)：采样时间。

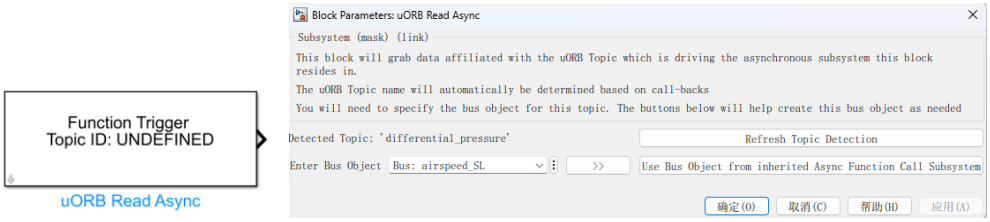
本模块相关例程可见：0.ApiExps\17.OffboardCtrlsAPI\Readme.pdf

2.5. uORB Read and Write—uORB 消息读取和写入库



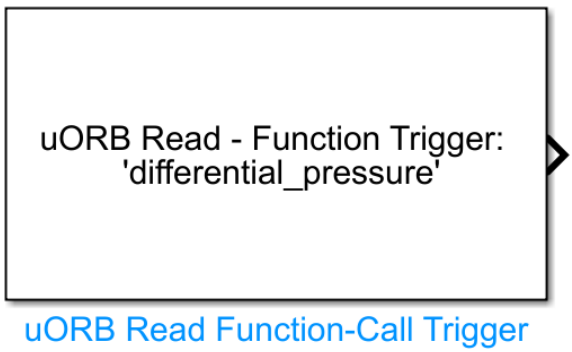
2.5.1. uORB Read Async—获取与 uORB Topic 相关的数据

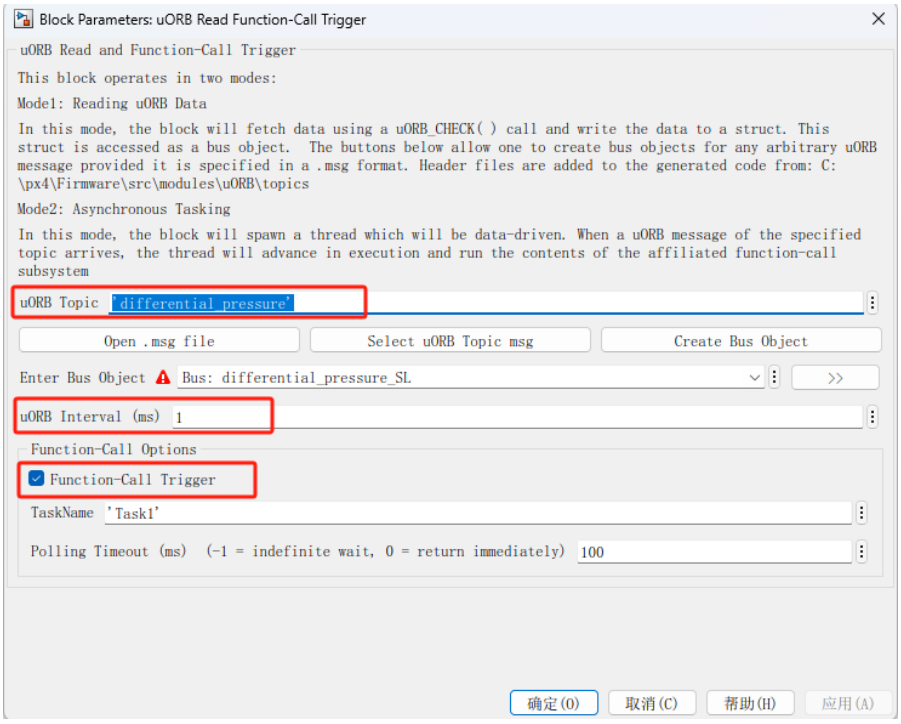
如下图所示，该块将获取与 uORB 主题相关的数据，该主题驱动该块所在的异步子系统。主题名称将根据回调自动确定，并且需要为该主题指定总线对象。更多信息可点击对话框的“帮助”按钮查看。



2.5.2. uORB Read Function-Call Trigger—uORB 消息读取回调函数触发模块

这个模块提供了两个功能，第一个是从某个 uORB 话题订阅对应的消息。第二个是对异步事件，采用触发函数调用信号的方法订阅话题上的消息数据。





如上图所示，第一个功能的使用步骤为：

- 选择一个已定义的话题

点击按钮“Select uORB Topic msg”可以打开话题列表以供选择，只支持非 C++对象的话题。

- 创建总线（bus）对象

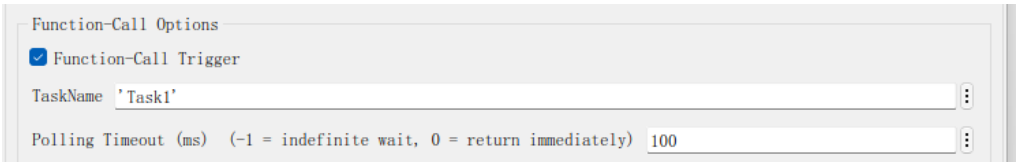
Simulink 的总线对象用来接收 uORB 的消息，点击按钮“Create Bus Object”，Simulink 将从.msg 文件夹中找到对应的消息文件，并将其映射到 MATLAB 工作空间生成总线对象。

- 设置 uORB 读间隔

非异步模式下，需要设置查询频率，单位是毫秒，某些话题设置了最高数据更新速率，设置的频率不要超过这个最大值。

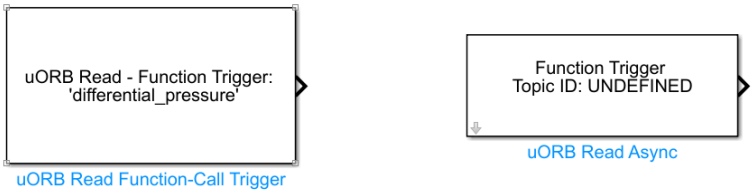
第二个功能的使用步骤为：

- 选择函数调用触发器



- 设置查询超时和任务名

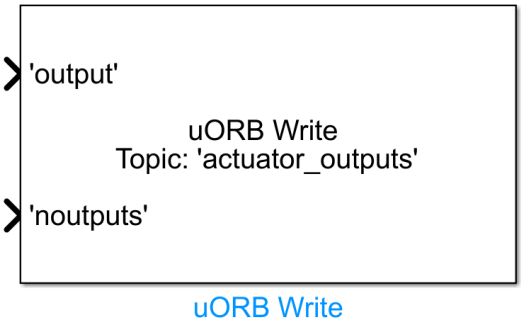
当选择了异步功能，采样时间设置框就消失了，这个时候需要设置查询超时参数和任务名。异步功能会衍生出一个新的线程，负责运行和函数触发信号相关的代码，它会通过查询来等待话题上新的数据的到来。此时需要另一个模块来读取话题上的数据，即“读 uORB 函数触发器数据模块”，如下图所示。

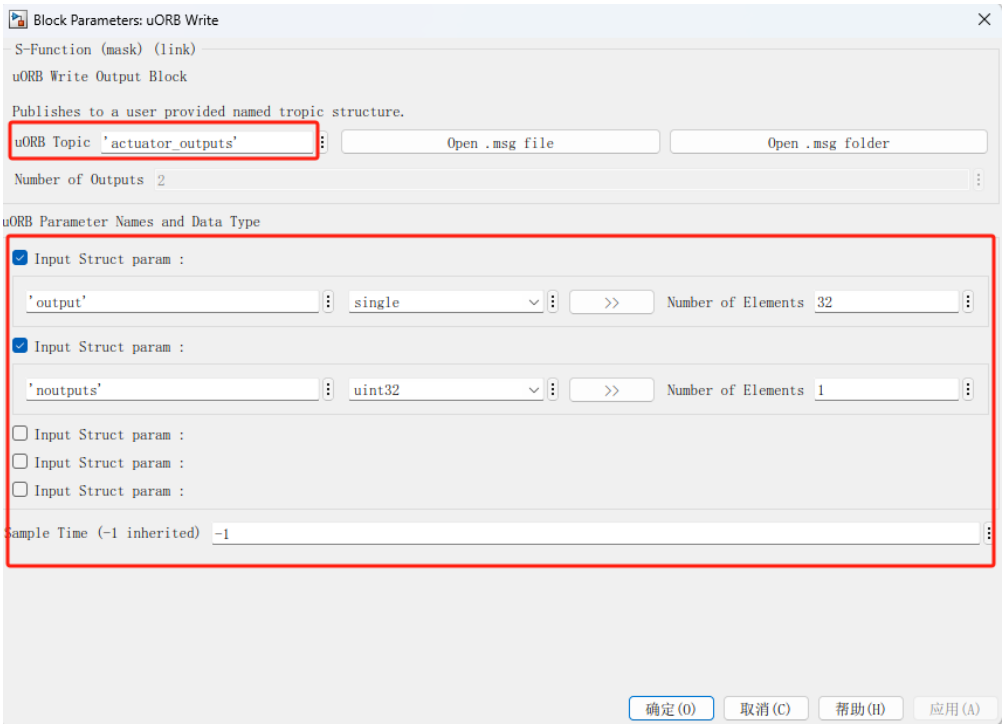


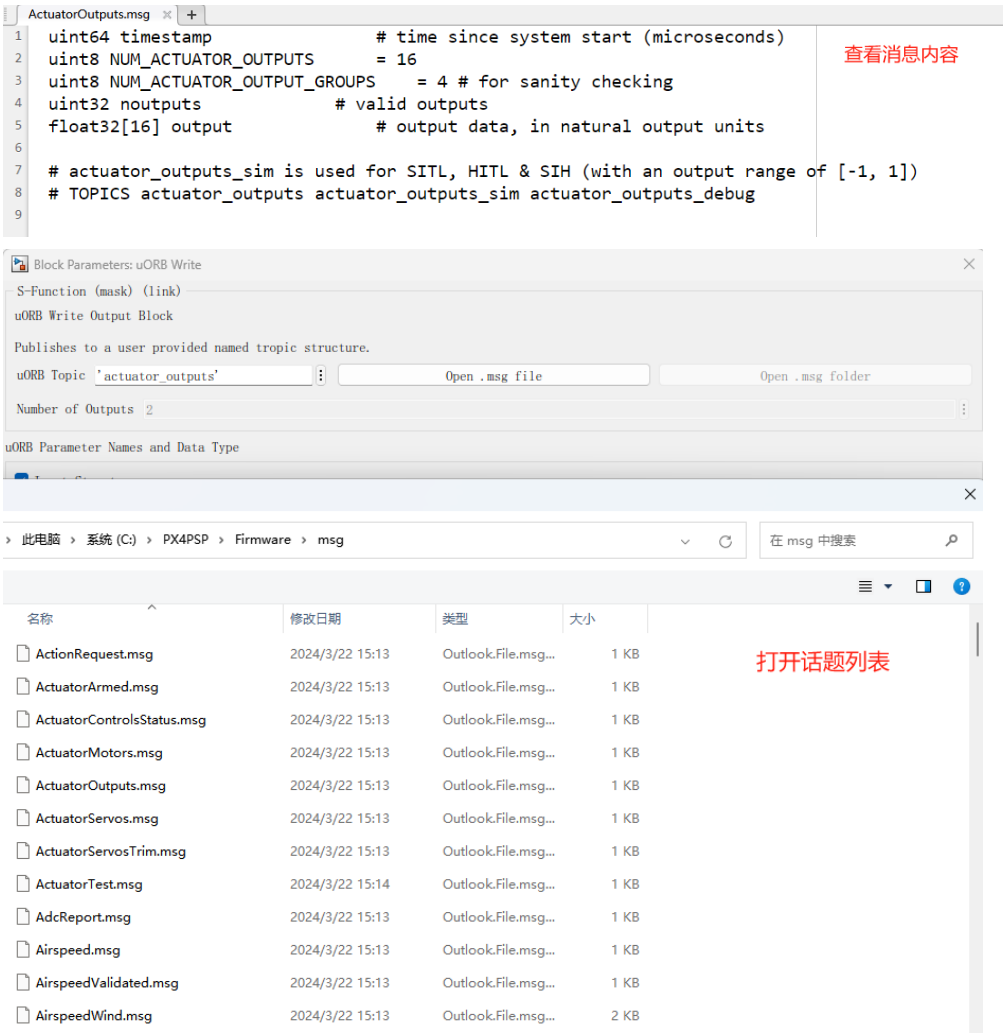
2.5.3. uORB Write—uORB 消息数据发布接口模块。

该接口允许用户向 uORB 话题发布指定的值或结构体，通过这个模块可以向某个 uORB 话题发布对应的消息，话题必须经过正确的定义，一些已定义的话题放在目录 C:\PX4PSP\Firmware\msg 下，在生成的代码中会自动包含话题的定义文件。

如下图所示，可以输入话题名，点击按钮“Open.msg file”打开对应的消息内容，点击按钮“Open.msg folder”打开话题列表，设置输入端口名及数据类型以对应话题消息。

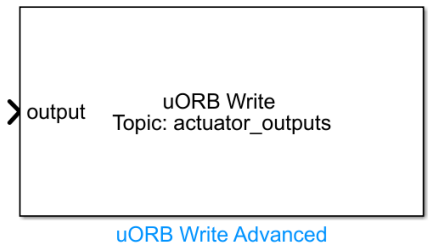






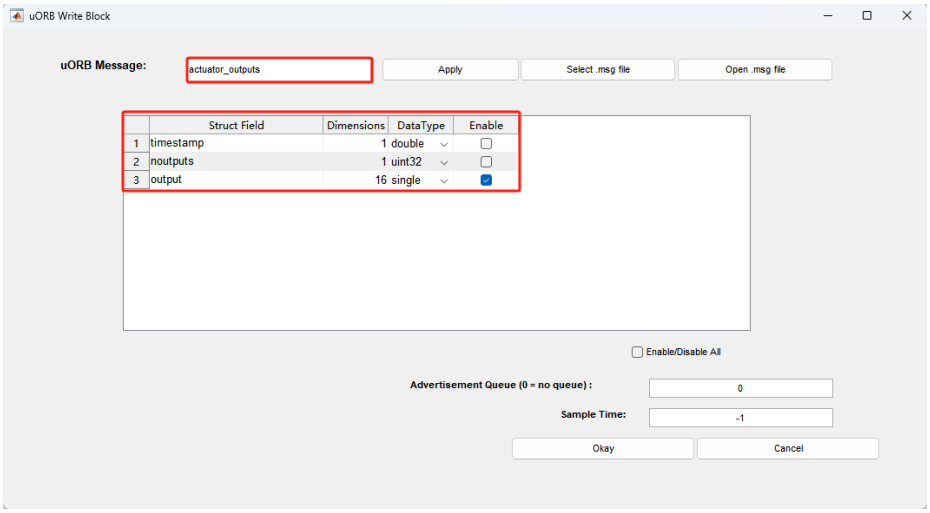
2.5.4. uORB Write Advanced—uORB 消息数据发布接口高级模块

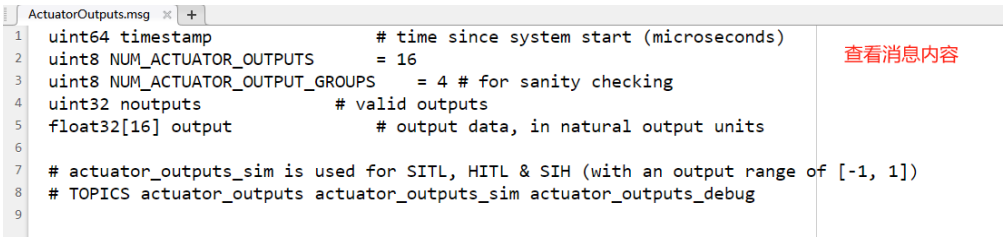
该接口可允许用户对其发布的数据进行更灵活的控制。在 Simulink 中使用 uORB Write Advanced 接口，可以实现更加复杂和精确的消息发布方式。可选择要写入的消息文件和一个消息 ID。此外，还可以设置消息的优先级、队列大小等高级选项。

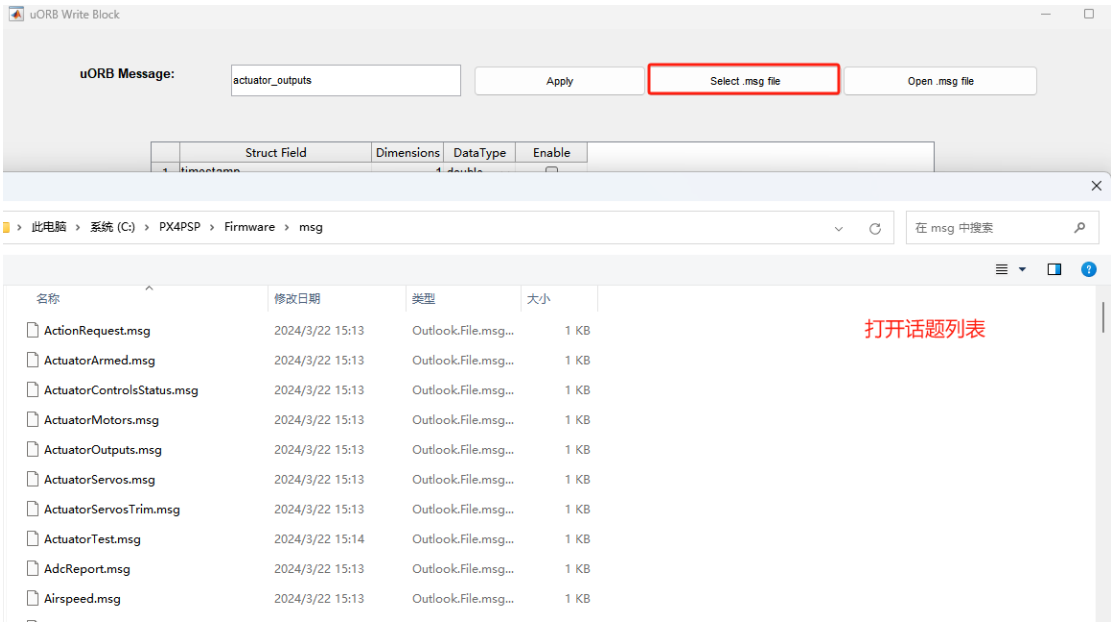


uORB Write Advanced

如下图所示，可以看到当前话题名，点击按钮“Open.msg file”打开对应的消息内容，点击按钮“Select.msg file”打开话题列表，并且可以设置输入端口名及数据类型以对应话题消息。

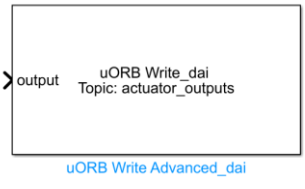


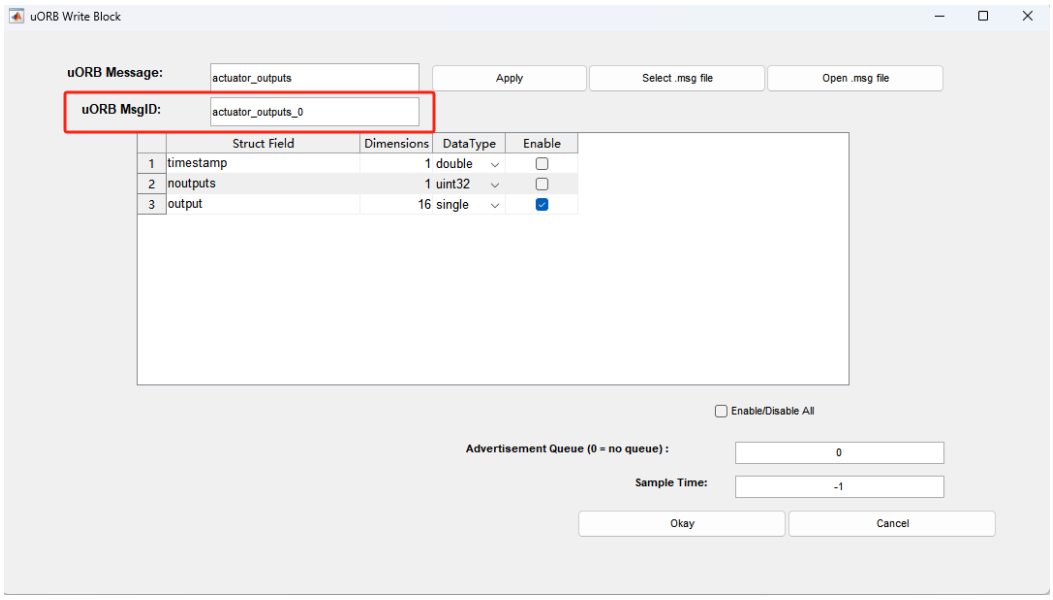




2.5.5. uORB Write Advanced_dai—uORB 消息数据发布接口进阶模块

如下图所示相较于 uORB Write Advanced，uORB Write Advanced_dai 新增自定义 uORB MsgID 功能。





具体例程实验见文件：

- 0.ApiExps\5.Log-Write-Read\Readme.pdf
- 0.ApiExps\6.uORB-Read-Write\Readme.pdf

3. MATLAB 命令行接口

RflySim 平台也支持 MATLAB 命令行窗口运行相关命令，具体包括有：

3.1. PX4Upload

可一键上传 PX4 固件到飞控中，此时上传的固件为：*\PX4PSP\Firmware\build\[编译命令]\[编译命令].px4（例如，[编译命令]可以为 px4_fmu-v6x_default）。

```
PX4Upload
```

执行完上述命令后，会弹出黑色窗口，提示用户插拔飞控，并显示上传进度条。

3.2. PX4CMD

固件编译选项替换，如需切换成 Pixhawk6C 飞控的编译环境，可进行如下操作：

```
PX4CMD('px4_fmu-v6c_default')
或
PX4CMD'px4_fmu-v6c_default'
```

3.3. PX4Build

可进行固件编译。

3.4. PX4AppName

重命名 PX4 软件中‘px4_simulink_app’模块名称，使得支持多个自动代码生成程序，具体使用请见：

```
PX4AppName('rfly_simulink_app')
%或
PX4AppName'rfly_simulink_app'
```

相关例程可见：2.AdvExps\e0_AdvApiExps\1.CusMaskPX4Code\Readme.pdf、2.AdvExps\e0_AdvApiExps\2.RenamePX4App\Readme.pdf

3.5. PX4AppLoad

加载重命名的 PX4 软件的 App，用于导入之前开发的 APP 程序，使用方法如下：

```
PX4AppLoad('C:\PX4PSP\rfly_simulink_app')
或
PX4AppLoad'C:\PX4PSP\rfly_simulink_app'
```

相关例程可见：2.AdvExps\e0_AdvApiExps\1.CusMaskPX4Code\Readme.pdf、2.AdvExps\e0_AdvApiExps\3.LoadPX4App\Readme.pdf

3.6. PX4ModiFile

通过 Excel 方式进行替换 PX4 软件中的部分代码，使用方法如下：

```
PX4ModiFile('C;\Users\dream\Desktop\自定义屏蔽 UORB 消息的例子 px4Block.xlsx')
```

相关例程可见：2.AdvExps\e0_AdvApiExps\1.CusMaskPX4Code\Readme.pdf、2.AdvExps\e0_AdvApiExps\2.RenamePX4App\Readme.pdf

3.7. PX4Official

通过执行命令，可以直接生成官方固件（未带输出屏蔽），可以用于还原飞控进行 HITL 的外部控制，或修复有问题的飞控。使用方法如下：

```
PX4Official
执行完上述命令后，再输入下面指令，可以将官方固件上传到飞控中：
PX4Upload
```

3.8. PX4SitlSet

使得当前自动代码生成的控制器 px4_simulink_app，支持 SITL 仿真。使用方法：Simulink 程序，点击 Build 生成硬件在环的.px4 文件后，直接运行 PX4SitlSet，然后运行 SITLRun（常规四旋翼），或其他由 DLL 模型驱动的 SITL 仿真脚本，即可对自动代码生成的算法硬件软件在环仿真。命令格式：

```
PX4SitlSet
```

相关例程可见：0.ApiExps\14.SITLVeriGenCodeFirm\Readme.pdf

3.9. PX4SitlRec

在 SITL 仿真的代码中，剔除掉自动代码生成控制器的 px4_simulink_app，回归到正常的软件在环仿真模式，重新支持 QGC 控制以及 Offboard 外部控制。注意：运行 PX4SitlSet 测试完 Simulink 的控制器，如果再要运行平台的官方视觉或外部控制等例程，请先使用 PX4SitlRec 还原环境。命令格式：

```
PX4SitlRec
```

相关例程可见：0.ApiExps\14.SITLVeriGenCodeFirm\Readme.pdf

4. 自动代码生成外部通信接口

RflySim 在运行一键安装脚本时，平台会对 Firmware 目录下的源码进行修改，在其中增加了 4 个 uORB 消息，并在*\PX4PSP\Firmware\msg\CMakeLists.txt 中已进行注册，详细介绍和使用细则如下：

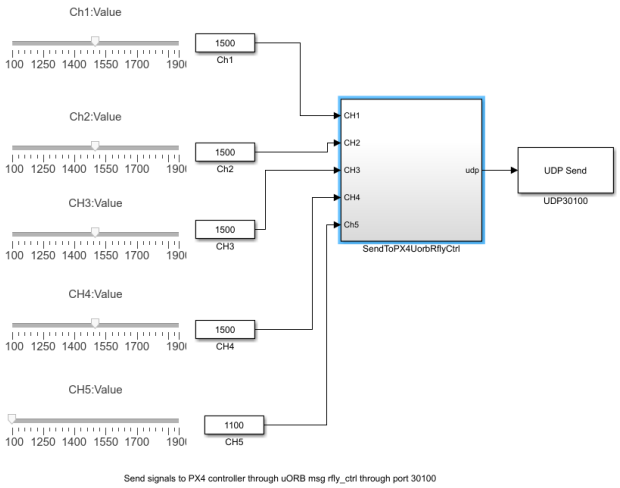
4.1. rfly_ctrl.msg

从外部通过 UDP 或 mavlink 协议传输进到 PX4 内部，消息定义格式：

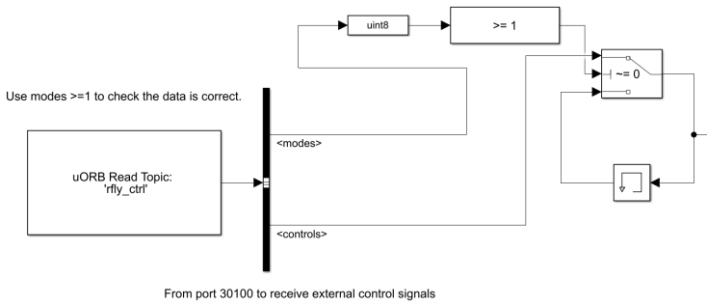
```
uint64 timestamp            # time since system start (microseconds)
uint32 flags                # control flag
uint8 modes                 # mode flag
float32[16] controls            # 16D control signals

# TOPICS rfly_ctrl rfly_ctrl1 rfly_ctrl2
```

Simulink 外部传数到 PX4 内部 uORB 消息：

发送端：见例程“0.ApiExps\9.PX4CtrlExternalTune\Readme.pdf”中的“SendToPX4UorbRflyCtrl”模块，它可以将消息通过 30100 系列端口发送给 CopterSim，然后通过 MAVLink 协议转发到 PX4 内部的 uORB 消息 rfly_ctrl



接收端：在 Simulink 进行底层控制器设计时，直接订阅 rfly_ctrl 消息，就能收到数据。



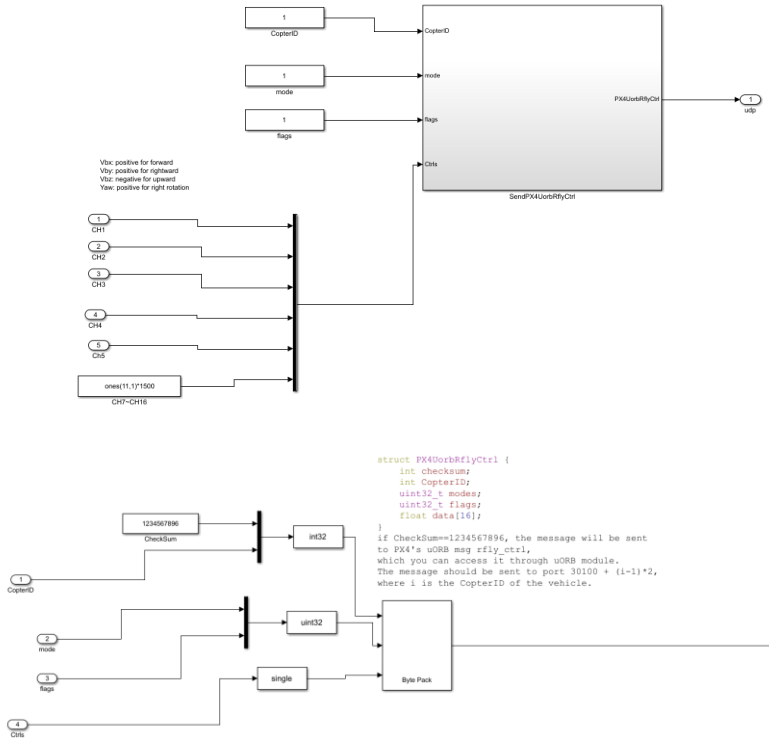
例程原理：PX4ExtMsgReceiver.slx 收到 rfly_ctrl 消息后，将 controls 的前 5 位，模拟成遥控器的输入，给到了之前例程的姿态控制器中，进行姿态控制。因此，PX4ExtMsgSender.slx 可以模拟发送遥控器的输入，来测试姿态算法。

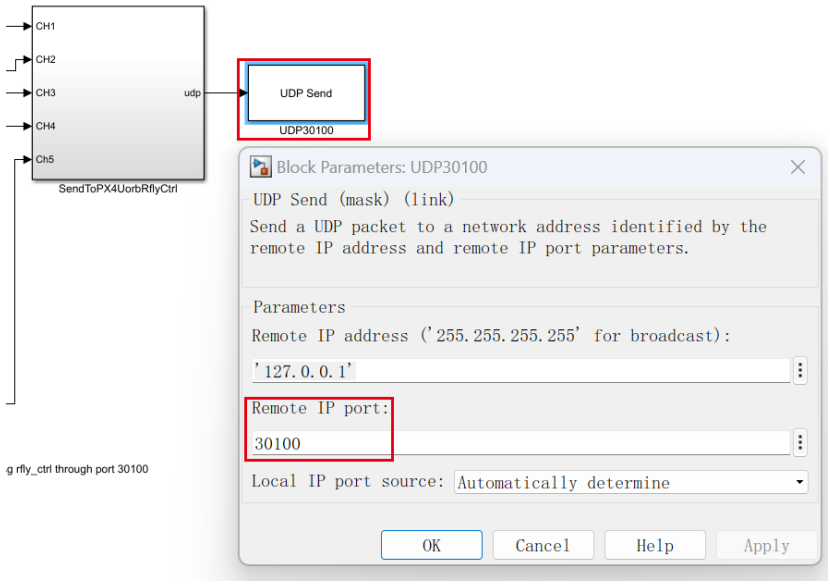
实验流程：PX4ExtMsgReceiver.slx 通过自动代码生成，烧入飞控之中，并启动硬件在环仿真，然后通过 Simulink 打开 PX4ExtMsgSender.slx，拖动“SendToPX4UorbRflyCtrl”左侧的滑块，可以模拟遥控器的各通道数据，来控制无人机。

Simulink 到 CopterSim 的通信原理：层层剖析“SendToPX4UorbRflyCtrl”模块可见：本 Simulink 模块通过 UDP 发送了下面结构体到本机 127.0.0.1 的 30100 端口

```
structPX4UorbRflyCtrl{
intchecksum;
intCopterID;
uint32_tmodes;
uint32_tflags;
floatdata[16];
}
```

注意：这里 checksum 必须设置成 1234567896 才能被 CopterSim 校验通过





CopterSim 到 PX4 的通信原理：CopterSim 在收到 PX4UorbRflyCtrl 消息后，会再转发为“hil_actuator_controls”（https://mavlink.io/en/messages/common.html#HIL_ACTUATOR_CONTROLS）的 mavlink 消息，然后转发到 PX4 飞控中。本消息的定义，如下图

HIL_ACTUATOR_CONTROLS (#93)

[Message] Sent from autopilot to simulation. Hardware in the loop control outputs (replacement for HIL_CONTROLS)

Field Name	Type	Units	Values	Description
time_usec	uint64_t	us		Timestamp (UNIX Epoch time or time since system boot). The receiving end can infer timestamp format (since 1.1.1970 or since system boot) by checking for the magnitude of the number.
controls	float[16]			Control outputs -1 .. 1. Channel assignment depends on the simulated hardware.
mode	uint8_t		MAV_MODE_FLAG	System mode. Includes arming state.
flags	uint64_t			Flags as bitfield, 1: indicate simulation using lockstep.

注意：实际上在硬件在环仿真时，本消息是 PX4 传输给 CopterSim 电机控制指令的消息，这里是借用了这条消息，反向传数给 PX4。

RflySim 平台修改了 C:\PX4PSP\Firmware\src\modules\mavlink\mavlink_receiver.cpp 源码，增加了对 hil_actuator_controls 消息的支持。

PX4 内部对 rfly_ctrl 消息的解析：从下图源码可见，rfly_ctrl 和 rfly_ext 都是借用了 hil_actuator_controls 消息进行数据传输，当 mode 为 123 时，会通过 rfly_ext 发送给 Simulink 控制器，而其他情况则会通过 rfly_ctrl 发送给控制器。

C: > PX4PSP > Firmware > src > modules > mavlink > mavlink_receiver.cpp

```
109  MavlinkReceiver::handle_message(mavlink_message_t *msg)
110  {
111      switch (msg->msgid) {
112      case MAVLINK_MSG_ID_HIL_ACTUATOR_CONTROLS:{
113          mavlink_hil_actuator_controls_t hil_actuator_control;
114          mavlink_msg_hil_actuator_controls_decode(msg, &hil_actuator_control);
115          if(hil_actuator_control.mode==123){
116              rfly_ext_s re{};
117              re.timestamp = hrt_absolute_time();
118              re.modes = 1;
119              for(int i=0;i<16;i++){
120                  re.controls[i]=hil_actuator_control.controls[i];
121              }
122              _rfly_ext_pub.publish(re);
123          }else{
124              rfly_ctrl_s rc{};
125              rc.timestamp = hrt_absolute_time();
126              rc.modes = hil_actuator_control.mode;
127              rc.flags = hil_actuator_control.flags;
128              for(int i=0;i<16;i++){
129                  rc.controls[i]=hil_actuator_control.controls[i];
130              }
131              _rfly_ctrl_pub.publish(rc);
132          }
133          break;
134      }
```

Python 接口使用：根据 CopterSim 的转发原理，我们可以发送 PX4UorbRflyCtrl 结构体到 30100 端口，或直接发送 MAVLink 消息到 PX4 飞控，两种方式来传入 rfly_ctrl 消息。

第一种方式：使用了 PX4MavCtrlV4.py 接口的 sendPX4UorbRflyCtrl 函数进行数据发送。

```
23  ctrls=[1500,1500,1900,1500,1900,1100,1
24  mav.sendPX4UorbRflyCtrl(ctrls)
25  # 发送SendToPX4UorbRflyCtrl数据，在PX4内
26  # 本消息主要设置油门通道（3通道）置于最高，
```

sendPX4UorbRflyCtrl 内部发送了 PX4UorbRflyCtrl 结构体到 30100 系列端口，再转发到 PX4 飞控之中。

```
1258      # }2i2I16f
1259      def sendPX4UorbRflyCtrl(self,data=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],modes=1,flags=1):
1260          checksum=1234567896
1261          buf = struct.pack("2i2I16f",checksum,self.CopterID,modes,flags,*data)
1262          self.udp_socket.sendto(buf, (self.ip, self.port+10000))
```

既然知道 rfly_ctrl 和 rfly_ext 都是来自于 hil_actuator_controls 的 Mavlink 消息，来传递数据。当然，我们也可以直接用 Python 发送 Mavlink 消息的形式来传递数据。

```
22
23  ctrls=[1500,1500,1900,1500,1900,1100,1100,1100,1100,1100,1100,1100,1100,1100,1100,1100]
24  mav.SendHILCtrlMsg(ctrls)
25  # 发送SendToPX4UorbRflyCtrl数据，在PX4内部产生rfly_ctrl的uORB消息
26  # 本消息主要设置油门通道（3通道）置于最高，飞机起飞。同时5通道置于最高，控制器解锁。
```

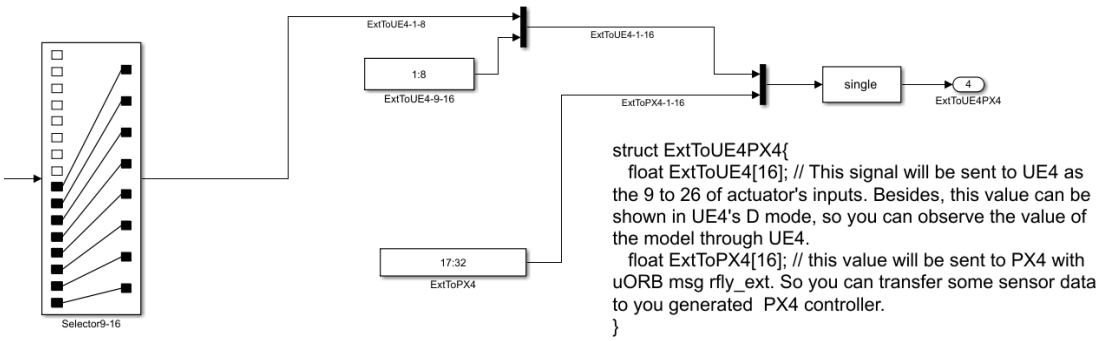
```
2213    # send hil_actuator_controls message to Pixhawk (for rfly_ctrl uORB message)
2214    def SendHILCtrlMsg(self,ctrls):
2215        """ Send hil_actuator_controls command to PX4, which will be transferred to uORB message rfly_ctrl
2216        https://mavlink.io/en/messages/common.html#HIL_ACTUATOR_CONTROLS
2217        """
2218        time_boot_ms = int((time.time()-self.startTime)*1000)
2219        controls = [1500,1500,1100,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500,1500]
2220        for i in range(len(ctrls)):
2221            if i<len(controls):
2222                controls[i]=ctrls[i]
2223        if self.isCom or self.isRealFly:
2224            self.the_connection.mav.hil_actuator_controls_send(time_boot_ms,controls,1,1)
2225        else:
2226            buf = self.mav0.hil_actuator_controls_encode(time_boot_ms,controls,1,1).pack(self.mav0)
2227            self.udp_socket.sendto(buf, (self.ip, self.port))
2228        #print("Msg Send.")
2229
```

4.2. rfly_ext.msg

从 DLL 模型直接传输进到 PX4 内部，消息定义格式：

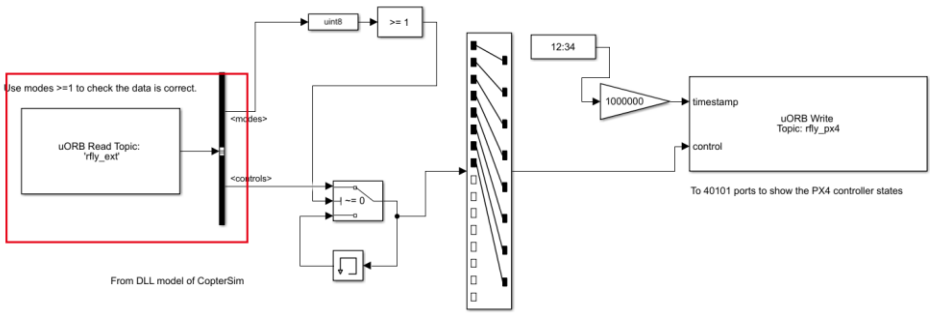
```
uint64timestamp             #timesincesystemstart(microseconds)
uint8modes#modeflag
float32[16]controls         #16Dcontrolsignals
```

1）在进行 DLL 模型开发时，引入 ExtToUE4PX4 的输出口，接口的前 16 维数据会直接发送给 UE4 用于执行器控制，而后 16 维数据，则会直接发送给 PX4，可以在底层控制器中通过 rfly_ext 消息接收到。



2）CopterSim 在进行 DLL 模型加载时，会将 ExtToUE4PX4 的 32 维数据，拆分成 16 维的 ExtToUE4 和 16 维的 ExtToPX4 结构体，后者会通过 MAVLink 消息 hil_actuator_controls 传入 PX4 飞控。注意，这里传输时强行设置了暗号 mode=123，因此，如上文所示 mavlink_receiver.cpp 会将其解析为 rfly_ext 再转发出去。

3）在 Simulink 底层控制器中，订阅“rfly_ext”即可得到来自 DLL 模型的数据。本接口可以用于传递目前 CopterSim 不支持的一些传感器数据到 PX4，用于加速硬件在环仿真环境的调试速度。



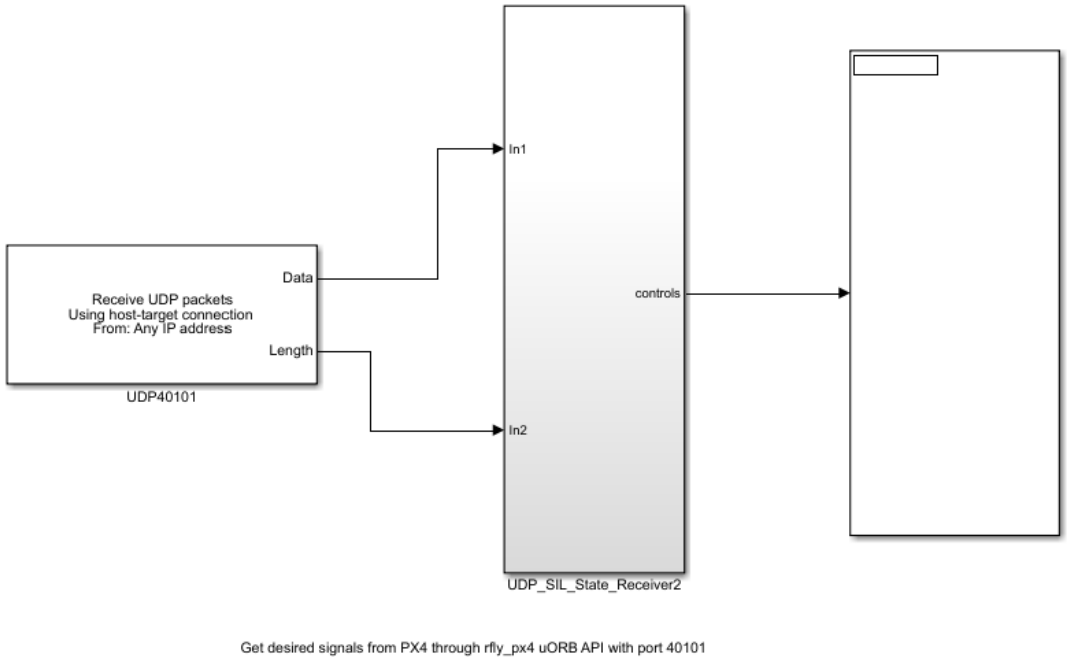
4）测试方法：PX4ExtMsgReceiver.slx 自动代码生成并烧入飞控，Exp2_MaxModelTemp.slx 生成 DLL 模型并启用硬件在环仿真。DLL 模型中，发送 1718…32 这几个数到了 rlfy_ext，然后底层控制器又将它以 rfly_px4 的 UORB 消息转发出去，因此可以用后文的方法，在 QGC 中查看数据的方式，查看 DLL 模型的消息是否正确传入。

4.3. rfly_px4.msg

将 PX4 内部数据，通过 UDP 或 mavlink 协议传输外部，消息定义格式：

```
uint64timestamp              #timesincesystemstart(microseconds)
float32[8]control            #8Dcontrolsignals
```

1）在底层控制器中发布 rfly_px4 消息，然后在 Simulink 中就能通过 40100 端口订阅得到数据，例程见：[0.ApiExps\9.PX4CtrlExternalTune\Readme.pdf](0.ApiExps\9.PX4CtrlExternalTune\Readme.pdf) 中的 UDP_SIL_State_Receiver2 模块



2）RflySim 平台修改了“Firmware\src\modules\mavlink\streams\ACTUATOR_CONTROL_TARGET.hpp”文件会订阅 rfly_px4 并转发为 ACTUATOR_CONTROL_TARGET 的 MAVLink 消息，并被 CopterSim 接收

```
109         actuator_controls_s act_ctrl;
110         if(N==0){
111             rfly_px4_s rf_px;
112             if (_rfly_px4_sub && _rfly_px4_sub->update(&rf_px)) {
113                 mavlink_actuator_control_target_t msg{};
114 
115                 msg.time_usec = rf_px.timestamp;
116                 msg.group_mlx = 123;
117 
118                 for (unsigned i = 0; i < sizeof(msg.controls) / sizeof(msg.controls[0]); i++) {
119                     msg.controls[i] = rf_px.control[i];
120                 }
121 
122                 mavlink_msg_actuator_control_target_send_struct(_mavlink->get_channel(), &msg);
123 
124                 return true;
125             }
```

消息详细定义见：https://mavlink.io/en/messages/common.html#ACTUATOR_CONTROL_TARGET

ACTUATOR_CONTROL_TARGET (#140)

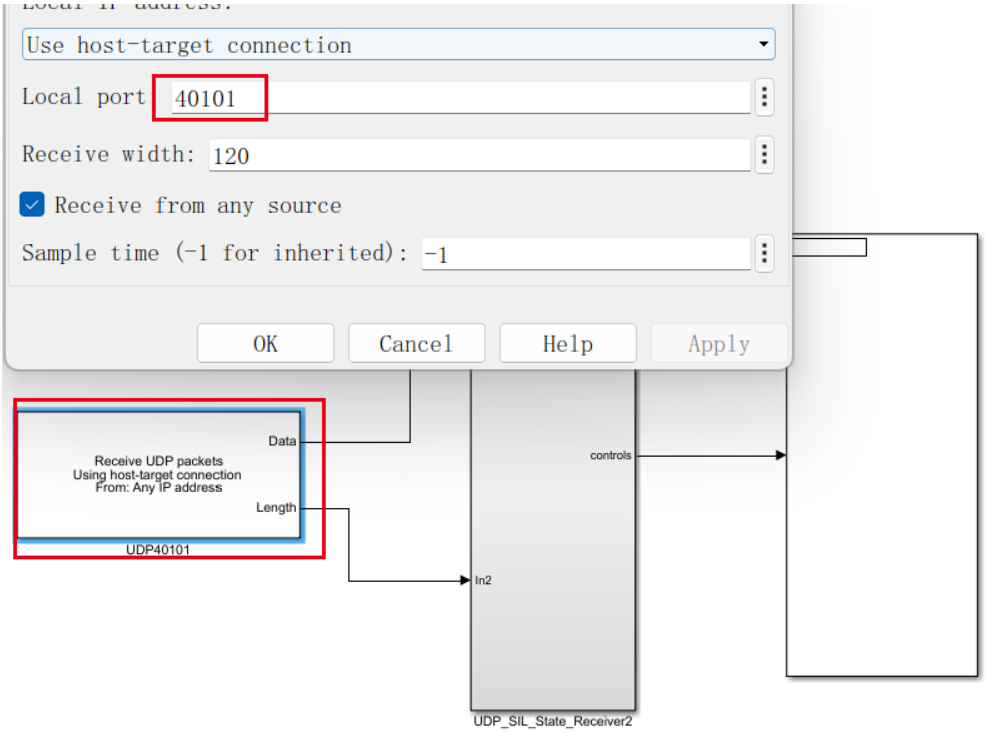
[Message] Set the vehicle attitude and body angular rates.

Field Name	Type	Units	Description
time_usec	uint64_t	us	Timestamp (UNIX Epoch time or time since system boot). The receiving end can infer timestamp format (since 1.1.1970 or since system boot) by checking for the magnitude of the number.
group_mlx	uint8_t		Actuator group. The "_mlx" indicates this is a multi-instance message and a MAVLink parser should use this field to difference between instances.
controls	float[8]		Actuator controls. Normed to -1..+1 where 0 is neutral position. Throttle for single rotation direction motors is 0..1, negative range for reverse direction. Standard mapping for attitude controls (group 0): (index 0-7): roll, pitch, yaw, throttle, flaps, spoilers, airbrakes, landing gear. Load a pass-through mixer to repurpose them as generic outputs.

3）CopterSim 接收到 ACTUATOR_CONTROL_TARGET 消息后，会判断 group_mlx 是否等于暗号 123，如果是则转发如下结构体到 40100 系列端口。

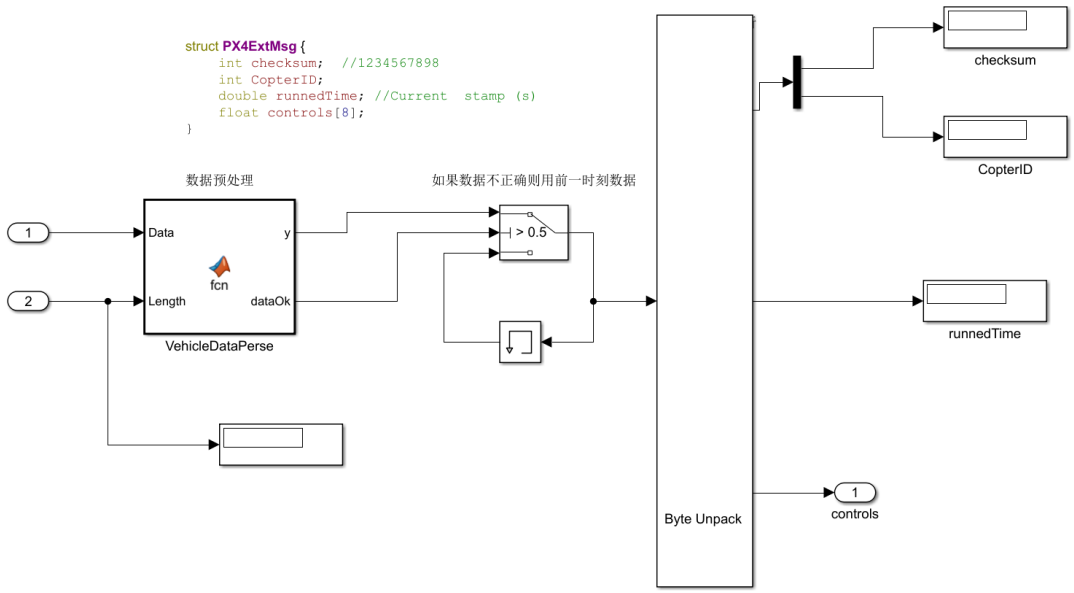
```
structPX4ExtMsg{
intchecksum;//1234567898
intCopterID;
doublerunnedTime;//Currentstamp(s)
floatcontrols[8];
}
```

注意：CopterSim 向外发的端口是奇数号，收的端口是偶数号，因此 1 号飞机，向外发布 PX4ExtMsg 消息对应的 40100 系列端口实际上是 40101。



Get desired signals from PX4 through rfly_px4 uORB API with port 40101

经过 UDP 模块监听 40101 端口，得到数据后，进行解析并输出



在解析函数中，对长度和 checksum 标志位进行了校验，防止读取到错误的数据，同时如果没有收到正确的数据，那么就会保持上述的数据。

编辑器 - Block: PX4ExtMsgSender/UDP_SIL_State_Receiver2/V

UDP_SIL_State_Receiver2/VehicleDataPerse

```
function [y,dataOk] = fcn(Data,Length)
y=uint8(zeros(48,1));
dataOk = 0;
if Length ~= 48
    return;
end
checksum=typecast(Data(1:4), 'int32');
if checksum ~= 1234567898
    return;
end

y = Data(1:48);
dataOk = 1;
```

同样的道理，通过 Python 或 C 语言，监听 MAVLink 的 ACTUATOR_CONTROL_TARGET 消息，也可以获取得到“rfly_ext”，读者可以自行尝试。

4.4. rfly_insils.msg

用于 PX4 内部数据传输使用，消息定义格式：

```
uint64timestamp             #timesincesystemstart(microseconds)
uint32checksum              #checksum/flag
int32[8]in_sil_ints         #8Dintsignals
float32[20]in_sil_floats        #20Dfloatsignals
```

本消息的例程在持续完善中，大体分为四个方面的使用：

1) 直接修改 PX4 的源码，增加 rfly_insils 的订阅代码，然后通过 Simulink 发布本消息，实现 Simulink 控制器与自己修改代码接口的通信。
2) 直接修改 PX4 源码，增加 rfly_insils 的发布代码，然后通过 Simulink 订阅本消息。
3) 生成两个自动代码的 APP，用于两者之间订阅发布沟通。
4) 在两处修改 PX4 的源码，相互收发，以连接两处代码。

5. 飞行日志记录与外部数据通信接口

5.1. 仿真真值数据分析

5.1.1. 离线获取方式

对于第 i 号飞机，只需要在 PX4PSP\CopterSim 下新建一个 CopterSim+i+.csv 的文件（例如，CopterSim1.csv），然后每次仿真后会记录仿真真值数据（同 RflySim3D 接收数据，包含了位置、速度、电机转速等信息）。详细操作步骤请见：*\PX4PSP\RflySimAPIs\2.RflySi

mUsage\1.BasicExps\e14_Log-Get\Readme.pdf。

5.1.2. 在线获取方式

RflySim 提供两种方式的仿真数据获取和分析，

- 方式一：MATLAB/Simulink 版的详细操作步骤见：*\PX4PSP\RflySimAPIs\10.RflySimSwarm\0.ApiExps\7.DataAnalysis_Mat\Readme.pdf。
- 方式二：Python 版的详细操作步骤见：*\PX4PSP\RflySimAPIs\10.RflySimSwarm\0.ApiExps\8.DataAnalysis_Py\Readme.pdf。

5.2. 飞行数据分析

5.2.1. 离线 Log 分析

离线 Log 日志分析详细操作步骤见：*\PX4PSP\RflySimAPIs\2.RflySimUsage\1.BasicExps\e4_Log-Reads-Python38Env\Readme.pdf。

5.2.2. 在线 Log 分析

访问 https://logs.px4.io/上传 ulog 文件，即可分析。

5.3. 控制器与外部数据通信接口

5.3.1. actuator_output 消息—HIL 仿真

actuator_output.msg 文件内容：

```
uint64timestamp                  #timesincesystemstart(microseconds)
uint8NUM_ACTUATOR_OUTPUTS        =16
uint8NUM_ACTUATOR_OUTPUT_GROUPS  =4   #forsanitychecking
uint32noutputs              #validoutputs
float32[16]output                #outputdata,innaturaloutputunits
```

该消息仅用于硬件在环仿真时使用。

5.3.2. pwm_output 消息—HIL&实飞

pwm_output.msg 消息文件内容：

```
uint64timestamp    #Timesincesystemstart(microseconds)
uint64error_count  #Timerovercaptureerrorflag(AUX5orMAIN5)
uint32pulse_width  #Pulsewidth,timercounts
uint32period       #Period,timercounts
```

该消息仅用于 HIL&实飞时使用，而且必须飞控的输出支持 px4io。

5.3.3. actuator_control_0—HIL&实飞

actuator_controls.msg 消息文件内容：

```
uint64timestamp             #timesincesystemstart(microseconds)
uint8NUM_ACTUATOR_CONTROLS=8
uint8NUM_ACTUATOR_CONTROL_GROUPS=6
uint8INDEX_ROLL=0
uint8INDEX_PITCH=1
uint8INDEX_YAW=2
uint8INDEX_THROTTLE=3
uint8INDEX_FLAPS=4
```

```
    uint8INDEX_SPOILERS=5
    uint8INDEX_AIRBRAKES=6
    uint8INDEX_LANDING_GEAR=7
    uint8INDEX_GIMBAL_SHUTTER=3
    uint8INDEX_CAMERA_ZOOM=4

    uint8GROUP_INDEX_ATTITUDE=0
    uint8GROUP_INDEX_ATTITUDE_ALTERNATE=1
    uint8GROUP_INDEX_GIMBAL=2
    uint8GROUP_INDEX_MANUAL_PASSTHROUGH=3
    uint8GROUP_INDEX_ALLOCATED_PART1=4
    uint8GROUP_INDEX_ALLOCATED_PART2=5
    uint8GROUP_INDEX_PAYLOAD=6

    uint64timestamp_sample  #thetimestampthedatathiscontrolresponseisbasedonwassampled
    float32[8]control

    #TOPICSactuator_controlsactuator_controls_0actuator_controls_1actuator_controls_2actuator
_controls_3
    #TOPICSactuator_controls_4actuator_controls_5
    #TOPICSactuator_controls_virtual_fwactuator_controls_virtual_mc
```

该消息主要传递驱动器控制指令，且不同的消息 ID 对用不同的控制组，详见 8.1 小节中 PX4 的控制组定义，如：actuator_control_0 系列消息。对应多旋翼的控制组，既支持多旋翼的硬件在环仿真又支持真机实飞。注：该消息使用时，需要在 RflySim 安装脚本中的第 10 项选择屏蔽 actuator_control_0 的消息，同时不能直接映射到电机，需要使用 PX4 的混控器规则来设置输出量。

5.3.4. TorqueThrustCtrls—力和力矩控制信号模块

见 TorqueThrustCtrls—力和力矩控制信号模块

5.4. 飞控与外部数据通信接口

注：本小节对应例程请见：*\PX4PSP\RflySimAPIs\5.RflySimFlyCtrl\0.ApiExps\9.PX4CtrlExternalTune。

5.4.1. 20100 系列端口—接收 PX4 内部状态估计值

20100 系列端口主要接收 PX4 内部状态估计值，其接收的数据包括：

```
    structoutHILStateData{//mavlinkdataforwardfromPixhawk
    uint32_ttime_boot_ms;//Timestampofthemessage
    uint32_tcopterID;//CopterIDstartfrom1
    int32_tGpsPos[3];//EstimatedGPSposition，lat&long:deg*1e7,alt:m*1e3andupispositive
    int32_tGpsVel[3];//EstimatedGPSvelocity,NED,m/s*1e2->cm/s
    int32_tgpsHome[3];//HomeGPSposition,lat&long:deg*1e7,alt:m*1e3andupispositive
    int32_trelative_alt;//alt:m*1e3andupispositive
    int32_thdg;//Courseangle,NED,deg*1000,0~360
    int32_tsatellites_visible;//GPSRawdata,sumofsatellite
    int32_tfix_type;//GPSRawdata,Fixedtype,3forfixed(goodprecision)
    int32_tresrveInit;//Int,reserveforthefutureuse
    floatAngEular[3];//EstimatedEulerangle,unit:rad/s
    floatlocalPos[3];//Estimatedlocoalposition,NED,unit:m
    floatlocalVel[3];//Estimatedlocoalvelocity,NED,unit:m/s
```

```
floatpos_horiz_accuracy;//GPShorizontalaccuracy,unit:m
floatpos_vert_accuracy;//GPSverticalaccuracy,unit:m
floatresrveFloat;//float,reserveforthefutureuse
}
```

接收端口第 i 架飞机命名格式为：20100+(2*i-1)，如：第 1 架飞机该端口为 20101，第 2 架为 20103，……，以此类推。

5.4.2. 30100 系列端口—接收 CopterSim 飞行仿真值并向飞控发送 rfly_ctrl 消息

30100 系列端口接收 CopterSim 飞行仿真值并向飞控发送 rfly_ctrl 消息，其接收的数据包括：

```
structSOut2Simulator
{
intcopterID;//飞机 ID
intvehicleType;//飞机构型
doublerunnedTime;//仿真时间
floatVelE[3];//NED 地球系速度
floatPosE[3];//NED 地球系位置
floatAngEuler[3];//欧拉角
floatAngQuatern[4];//四元数
floatMotorRPMS[8];//电机转速 RPM
floatAccB[3];//机体轴加速度
floatRateB[3];//机体轴角速度
doublePosGPS[3];//地球 GPS 经纬高
}

//SOut2Simulator 的定义与 Simulink 里面新加的 MavVehileInfo 结构体完全一致
//可以直接对新的 simulink 模型的 MavVehile3DInfo 输出口数据进行打包
//两个结构体的长度都是 152
sizeof(SOut2Simulator)=sizeof(MavVehileInfo)=152

打包封装的结构体为
typedefstruct_netDataShort
{
TargetTypetg;//这里目前随便填，写 1 即可 uint32
intlen;//这个长度为传输结构体长度，目前是 152
charpayload[192];//这里面前 152 位存放了 SOut2Simulator 结构体数据，后面的 40 位保留
}netDataShort;

//UDP 包的总长度为 200
sizeof(netDataShort)=200


//发送的 UDP 端口号为：20010
```

接收端口第 i 架飞机命名格式为：30100+(2*i-1)，如：第 1 架飞机该端口为 30101，第 2 架为 30103，……，以此类推。

其发送的 uORB 消息数据格式为：

```
uint64timestamp             #timesincesystemstart(microseconds)
uint32flags                 #controlflag
uint8modes#modeflag
float32[16]controls         #16Dcontrolsignals
```

发送端口第 i 架飞机命名格式为：30100+(2*i-2)，如：第 1 架飞机该端口为 30100，第

2 架为 30102，……，以此类推。

5.4.3. 40100 系统端口—接收飞控内部 rfly_px4 消息

40100 系统端口接收飞控内部 rfly_px4 消息，接收的数据包括：

```
structPX4ExtMsg{
intchecksum;//1234567898
intCopterID;
doublerunnedTime;//Currentstamp(s)
floatcontrols[8];
}
```

接收端口第 i 架飞机命名格式为：40100+(2*i-1)，如：第 1 架飞机该端口为 40101，第 2 架为 40103，……，以此类推。

6. 代码屏蔽与替换接口

基于 RflySim 底层控制算法开发时，为了验证所开发的控制算法，我们需要屏蔽掉 PX4 软件中的输出，在大多数情况下，我们只需要直接屏蔽掉 PX4 软件系统的中的电机输出即可。但是，某些特定开发任务需要屏蔽的是 PX4 软件系统中某个模块的某个中间量，以此满足开发需求。例如：我们需要屏蔽 PX4 软件系统中的姿态角速率环控制器的模块(该位置为 PX4-1.12.3 版本，其他版本请查看 PX4 官方帮助文件)在：*\PX4PSP\Firmware\src\modules\mc_rate_control。打开该文件夹中的“MulticopterRateControl.cpp”文件，根据 px4 的源码构架可知，姿态角速率环的输出 uORB 消息是“actuator_controls_0”(该消息详细定义可以参考 https://docs.px4.io/v1.12/en/concept/mixing.html)。通过查阅代码可得，发布“actuator_controls_0”消息的代码如下(也可通过搜索：“_actuators_0_pub.publish(actuators);”找到)：

```
253                 }
254             }
255
256             actuators.timestamp = hrt_absolute_time();
257             _actuators_0_pub.publish(actuators);
258
259         } else if (_v_control_mode.flag_control_termination_enabled) {
260             if (!_vehicle_status.is_vtol) {
261                 // publish actuator controls
262                 actuator_controls_s actuators{};
263                 actuators.timestamp = hrt_absolute_time();
264                 _actuators_0_pub.publish(actuators);
265             }
266         }
267     }
268
```

为了屏蔽上述上述两行代码，有两种方法可以进行屏蔽。

方法一：我们只需要将其删除、注释掉、或者替换成其他无效代码就行。由于 PX4 的编译检查非常严格，再上面两行代码处直接注释，可能会导致 actuators 定义了但是未被使用，从而出现编译错误，因此这里需要用 UNUSED 宏来实现屏蔽。代码的屏蔽可以根据情况采用如下规则中的一种。

a) “_actuators_0_pub.publish(actuators);”替换为→“”。(这里是空字符，相当于删除。注意，这种方式仅限于不会出现变量未使用报错的情形)

b) “_actuators_0_pub.publish(actuators);”替换为→“//_actuators_0_pub.publish(actuators);”

这里相当于注释掉行，注意事项同上）

c) “_actuators_0_pub.publish(actuators);”替换为→“UNUSED(actuators);”。（这里相当于替换为无效语句。注意，这种方式适用于直接删除 actuators 变量会报错的情形，且适用于 PX41.12 及以下版本，因为 1.13 固件开始 UNUSED 宏被取消了）

d) “_actuators_0_pub.publish(actuators);”替换为→“(void)(actuators);”。（这里相当于替换为无效语句。注意，这种方式适用于直接删除 actuators 变量会报错的情形，且适用于包括 1.13 版本固件在内的所有版本）

方法二：可以直接用一个修改好的文件替换掉待修改的文件。平台提供的接口能够满足上面的需求。RflySim 平台的一键安装脚本提供了源码文件替换的功能，核心思想是按照给定的 excel 文件模版，将要替换的文件和内容进行描述，在运行安装脚本时填入 excel 文件地址，即可实现平台安装时，自动进行文件的替换或文件内容的修改。具体屏蔽方式详情可见例程文件：2.AdvExps\e0_AdvApiExps\1.CusMaskPX4Code\Readme.pdf

7. 多模块并行开发接口

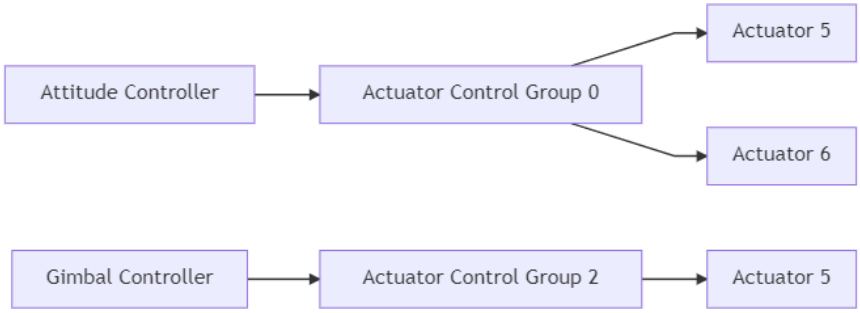
RflySim 平台最新版支持快速创建多个模块并行开发的功能，基于 PX4 软件系统中的多进程运行状态，MATLAB 自动代码生成的 PX4 应用名称为：px4_simulink_app，可通过“MATLAB 命令行接口”中的 PX4 应用重命名的方式，将 px4_simulink_app 改名，这样就可以继续通过 Simulink 搭建模型生成另一个 px4_simulink_app 名称的应用，若要再次新增应用可继续进行名称修改，以此类推，理论上可实现众多 PX4 的应用创建，以满足开发需求。具体接口使用详见实验文件：*\RflySimAPIs\5.RflySimFlyCtrl\2.AdvExps\e0_Basic-Interface\4.MultPX4App\Readme.pdf。

8. 不同机型开发接口

8.1. PX4 混控器介绍

PX4 架构保证了核心控制器中不需要针对机身布局做特别处理。混控指的是把输入指令（例如：遥控器打右转）分配到电机以及舵机的执行器（如电调或舵机 PWM）指令。对于固定翼的副翼控制而言，每个副翼由一个舵机控制，那么混控的意义就是控制其中一个副翼抬起而另一个副翼落下。同样的，对多旋翼而言，俯仰操作需要改变所有电机的转速。将混控逻辑从实际姿态控制器中分离出来可以大大提高复用性。

特定的控制器发送一个特定的归一化的力或力矩指令（缩放至-1..+1）给混控器，混控器则相应地去设置每个单独的执行器。控制量输出驱动程序（比如：UART,UAVCAN 或者 PWM）则将混控器的输出所放为执行器实际运行时的原生单位，例如输出一个值为 1300 的 PWM 指令。



PX4 的控制通道输出主要有 4 个控制组(ControlGroup)，分别为：

- actuator_controls_0：飞控的主要控制通道，用于输出俯仰、滚转、偏航、油门等各个通道的控制量。具体定义如下：

```
ControlGroup#0(FlightControl)
•    0:roll(-1..1)
•    1:pitch(-1..1)
•    2:yaw(-1..1)
•    3:throttle(0..1normalrange,-1..1forvariablepitch/thrustreversers)
•    4:flaps(-1..1)
•    5:spoilers(-1..1)
•    6:airbrakes(-1..1)
•    7:landinggear(-1..1)
```

- actuator_controls_1：备用控制通道，在 VTOL 中用于输出固定翼模式的控制输出。具体定义如下：

```
ControlGroup#1(FlightControlVTOL/Alternate)
•    0:rollALT(-1..1)
•    1:pitchALT(-1..1)
•    2:yawALT(-1..1)
•    3:throttleALT(0..1normalrange,-1..1forvariablepitch/thrustreversers)
•    4:reserved/aux0
•    5:reserved/aux1
•    6:reserved/aux2
•    7:reserved/aux3
```

- actuator_controls_2：云台控制通道。具体定义如下：

```
ControlGroup#2(Gimbal)
•    0:gimbalroll
•    1:gimbalpitch
•    2:gimbalyaw
•    3:gimbalshutter
•    4:reserved
•    5:reserved
•    6:reserved
•    7:reserved(parachute,-1..1)
```

- actuator_controls_3：遥控映射通道。具体定义如下：

```
ControlGroup#3(ManualPassthrough)
•    0:RCroll
•    1:RCpitch
•    2:RCyaw
•    3:RCthrottle
•    4:RCmodeswitch
•    5:RCaux1
```

- 6:RCaux2
- 7:RCaux3

8.2. PX4 混控器定义

PX4 中的混控器的其中一个作用是连接上层应用模块（例如姿态控制器解算的输出量）与底层硬件的执行机构输出（对应电机、舵机的 PWM 值）。混控器模块相当于隔离了上层应用模块与底层的硬件接口，在编写上层应用模块时不用担心控制量是输出给哪个电机，同时更改不同的混控器配置文件可以适应不同的机型而无需更改上层应用模块的代码。具体代码见：*\PX4PSP\Firmware\ROMFS\px4fmu_common\mixers

PX4 软件系统中混控器文件的默认设置可在文件夹*\PX4PSP\Firmware\ROMFS\px4fmu_common\init.d\airframes 中查阅。同时，也可在 SD 卡的*/etc/mixers/目录中同名的混控器文件进行覆盖。PX4 将混控器文件中被命名为 xxxx.main.mix 的文件映射到主(MAIN)输出，被命名为 yyyy.aux.mix 的文件映射到辅助(AUX)输出，其中前缀(xxxx/yyyy)取决于机架和机架的配置。通常，MAIN 和 AUX 输出对应于 MAIN 和 AUXPWM 输出，但当启用时，这些可能被加载到 UAVCAN(或其他)总线中。

主混合器文件名(前缀 xxxx)在机身配置中使用 setmixerxxxx 设置，如：airframes/4011_dji_f450 调用 setmixerquad_x 来加载主混控器文件 quad_x.MAIN.mix)。AUX 混控器文件(前缀 yyyy)取决于机身设置或默认值，如:

- `MIXER_AUX` 可用于设置加载哪个 yyyy.aux.mix 文件。如：*\PX4PSP\Firmware\ROMFS\px4fmu_common\init.d\airframes\13006_vtol_standard_delta 文件中，`setMIXER_AUXvtol_delta` 设置为该机架后，加载的混控器文件为：*\PX4PSP\Firmware\ROMFS\px4fmu_common\mixers\vtol_delta.aux.mix。
- 对于多旋翼和固定翼机型，若未设置 `MIXER_AUX` 选项，则默认加载的混控器文件为：*\PX4PSP\Firmware\ROMFS\px4fmu_common\mixers\pass.aux.mix。（注：pass.aux.mix 是遥控器的直通混控器文件，它将 4 个用户自定义的遥控器通道的值(使用 RC_MAP_AUXx/RC_MAP_FLAPS 参数设置)传递到 AUX 输出的前四个输出。）
- 垂直起降无人机（VOTL）若进行相关设置则加载所设置的 yyyy.aux.mix；若未进行设置，则只加载 `MIXER` 设置的混控器文件。
- 通用控制模式启用(且输出模式设置为 AUX)将覆盖机架中 `MIXER_AUX` 设置混控器文件，并加载*\PX4PSP\Firmware\ROMFS\px4fmu_common\mixers\mount.aux.mix 上的 AUX 输出。

注：上述所述的混控文件加载通过文件：*\PX4PSP\Firmware\ROMFS\px4fmu_common\init.d/rc.interface 实现。

8.3. PX4 混控器文件语法

Mixer 文件是定义一个或多个 Mixer 定义的文本文件:一个或多个输入与一个或多个输出之间的映射。主要有四种类型的定义:multirotormixer,helicoptermixer,summingmixer,andnul

lmixer。

- multirotormixer：定义输出 4、6 或 8 的+型或 x 型旋翼载具。
- helicoptermixer：定义直升机斜盘伺服器和主电机 ESCs 的输出(尾桨是一个单独的混合器)。
- summingmixer：将零或多个控制输入组合成单个执行器输出。输入被缩放，混合函数在应用输出缩放器之前对结果求和。
- summingmixer：产生一个输出为零的执行器输出(当不处于故障安全模式时)。

每个混控器产生的输出量取决于混控器的类型和配置。如：multirotormixer 根据其机型可有 4、6 或 8 个输出，而 summingmixer 或 summingmixer 只产生一个输出。可以在每个文件中指定多个混控器。输出顺序(将混合器分配给执行器)特定于读取混控器定义的设备，对于 PWM，输出顺序与声明顺序相匹配。

每个混控器文件最开始定义的语句为：

```
<tag>:<mixerarguments>
```

其中 tag 表示所选择的混控器类型，如下：

```
R:Multirotormixer
H:Helicoptermixer
M:Summingmixer
Z:Nullmixer
```

8.4. SummingMixer—加法混控器

SummingMixer 用于控制无人机执行器和伺服。它将零或多个控制输入组合成单个执行器输出。输入被缩放，混合函数在应用输出缩放器之前对结果求和。最小执行器遍历时间限制也可以在输出标量中指定(回转率的逆)。简单的混合器定义如下:

```
M:<controlcount>
O:<-vescale><+vescale><offset><lowerlimit><upperlimit><traversaltime>
```

若`<controlcount>`为零，则总和有效为零，混合器将输出一个固定值，该值受`<lowerlimit>`和`<upperlimit>`约束。

第二行用上述的标量参数定义输出标量。虽然计算作为浮点操作执行，但存储在定义文件中的值按 10000 倍缩放；即-0.5 的偏移量被编码为-5000。输出标度上的`<traversaltime>`(可选)用于执行器，若数值过大，可能会损坏飞行器—如：倾转旋翼垂直起降飞行器上的倾斜执行器。可以用来限制执行器的变化速率(如果没有指定，则不应用速率限制)。例如：`<traversaltime>`值 20000 将限制执行器的变化率，使得从`<lowerlimit>`和`<upperlimit>`至少需要 2 秒，反之亦然。

注 1：`<traversaltime>`应该只在硬件需要时使用！

注 2：不要对控制车辆姿态的致动器(如用于气动表面的伺服器)施加任何限制，因为这很容易导致控制器不稳定。

继续定义`<controlcount>`的输入及其缩放，形式为:

```
S:<group><index><-vescale><+vescale><offset><lowerlimit><upperlimit>
```

注 3：`S:`必须在`O:`的下面。

注 4：任何具有油门输入的混控器输出(`S:`中的`<group>=0`和`<index>=3`)均无法在解锁或预解锁状态下工作。如：有四个输入(滚转、俯仰、偏航和油门)的伺服器，即使有滚转/俯仰/偏航信号也不会在解锁状态下运动。

`<group>`值标识标量器要读取的控制组，`<index>`值表示该组中的偏移量。这些值特定于读取混控器定义的设备。当用于混合载具控制时，混控器组 0 为载具姿态控制组，而 0~3 通常分别为滚转、俯仰、偏航和推力。其余字段使用上面讨论的参数配置控制缩放器。当计算作为浮点运算执行时，存储在定义文件中的值按 10000 倍缩放；即-0.5 的偏移量被编码为-5000。下面解释一个典型的混合器文件示例。详细示例解析请见：https://docs.px4.io/v1.13/en/dev_airframes/adding_a_new_frame.html#mixer-file。

8.5. NullMixer—空混控器

该混控器不消耗任何控制通道，生成一个值始终为零的单个执行器输出。通常，Null Mixer用作混频器集合中的占位符，以实现执行器输出的特定模式。它也可以用来控制用于故障安全装置的输出值(正常使用时输出为 0；在故障安全期间，混控器被忽略，而使用故障安全值代替)。定义如下:

```
Z:
```

8.6. MultirotorMixer—多旋翼混控器

MultirotorMixer 将四个控制输入(滚转、俯仰、偏航、推力)组合成一组执行器输出，用于驱动电机速度控制器。定义如下：

```
R:<geometry><rollscale><pitchscale><yawscale><idlespeed>
```

支持的机型有：

- 4x-四旋翼 x 型配置
- 4+-四旋翼+型配置
- 6x-六旋翼 x 型配置
- 6+-六旋翼+型配置
- 8x-八旋翼 x 型配置
- 8+-八旋翼+型配置

横滚、俯仰和偏航比例值决定了相对于推力控制的横滚、俯仰和偏航控制的比例。当计算作为浮点运算执行时，存储在定义文件中的值按 10000 倍缩放；如：0.5 则被编码为 5000。横滚、俯仰和偏航输入的范围从-1.0 到 1.0，而推力输入的范围从 0.0 到 1.0。每个执行器的输出范围为-1.0 至 1.0。

空转速度的范围从 0.0 到 1.0。空转速度是相对于电机的最大速度，它是当所有控制输入为零时，电机被命令旋转的速度。在执行器饱和的情况下，所有执行器的值被重新调整，使饱和执行器限制为 1.0。

8.7. HelicopterMixer—直升机混控器

HelicopterMixer 将三个控制输入(滚转、俯仰、推力)组合成四个输出(旋转斜盘和主电机 ESC 设置)。直升机混合器的第一个输出是主马达的油门设置。随后的输出是旋转斜盘的伺服器。尾桨可以通过添加一个简单的混合器来控制。推力控制输入用于主电机设置以及斜盘的集体螺距。它使用了一个油门曲线和一个俯仰曲线，都由五个点组成。

注：油门和俯仰曲线将“推力”杆输入位置映射到油门值和俯仰值(分别)。这允许飞行特性调整为不同类型的飞行。

HelicopterMixer 定义如下：

```
H:<numberofswash-plateservos,either3or4>
T:<throttlesettingatthrust:0%><25%><50%><75%><100%>
P:<collectivepitchatthrust:0%><25%><50%><75%><100%>
```

`T:`定义油门曲线的点。`P:`定义俯仰曲线的点。两条曲线都包含 0 到 10000 之间的 5 个点。对于简单的线性变化，曲线的五个值应该是 0、2500、5000、7500、10000。

每个旋转斜盘的伺服器(3 或 4)的定义如下:

```
S:<angle><armlength><scale><offset><lowerlimit><upperlimit>
```

`<angle>`以度为单位，0 度为机头方向。正角度是顺时针。`<armlength>`是标准化的长度，即 10000 等于 1。如果所有的伺服臂都是相同的长度，值应该都是 10000。较大的臂长会减少伺服偏转的量，而较短的臂长会增加伺服偏转。伺服输出按`<scale>/10000`缩放。缩放后，`<offset>`被应用且它值应该在-10000 和+10000 之间。在全伺服范围内，`<lowerlimit>`和`<upperlimit>`应为-10000 和+10000。

尾桨可以通过加一个 SummingMixer 来控制:

```
M:1
S:0210000100000-1000010000
```

通过将尾桨直接映射到偏航命令。这适用于两个伺服控制的尾翼，以及尾翼与专用电机。

130 叶片直升机混合器文件如下所示：

```
H:3
T:030006000800010000
P:5001500250035004500
#Swashplateservos:
S:010000100000-80008000
S:140130541000000-80008000
S:220130541000000-80008000

#Tailservo:
M:1
S:0210000100000-1000010000
```

- 在 50%推力时，油门曲线的斜率略陡，达到 6000(0.6)。
- 在 100%推力的情况下，以较小的坡度达到 10000(1.0)。
- 俯仰曲线是线性的，但不会使用其整个范围。
- 在 0%油门时，总距操纵杆设置已经在 500(0.05)。

- 在最大油门下，总距操纵杆仅为 4500(0.45)。
- 对这种类型的直升机使用更高的数值会使叶片失速。
- 这架直升机的旋转斜盘系统位于 0、140 和 220 度的角度。
- 伺服臂长不相等。
- 与第一个伺服系统相比，第二个和第三个伺服系统的手臂长度为 1.3054。
- 伺服器被限制在-8000 和 8000，因为它们是机械约束。

8.8. VTOLMixer—垂直起降无人机混控器

垂直起降系统使用多旋翼混合器作为多旋翼模式下的输出，总和混合器作为固定翼模式下的输出。垂直起降无人机的混控器系统既可以组合成一个单独的混控器，其中所有的执行器都连接到 IO 或 FMU 端口，也可以分成单独的混控器文件用于 IO 和 AUX。

9. PX4 部分原生接口

PX4 具有高度可移植性，且是一个不依赖于操作系统的无人机解决方案。它由来自工业界和学术界的世界级开发人员开发，并得到活跃的全球社区的支持，为从赛车和货运无人机到地面车辆和潜水器的各种车辆提供动力。官网链接：https://docs.px4.io/main/en/。如下将介绍一些 PX4 软件系统中常用 uORB 消息、模块、参数，

9.1. PX4 软件系统入门

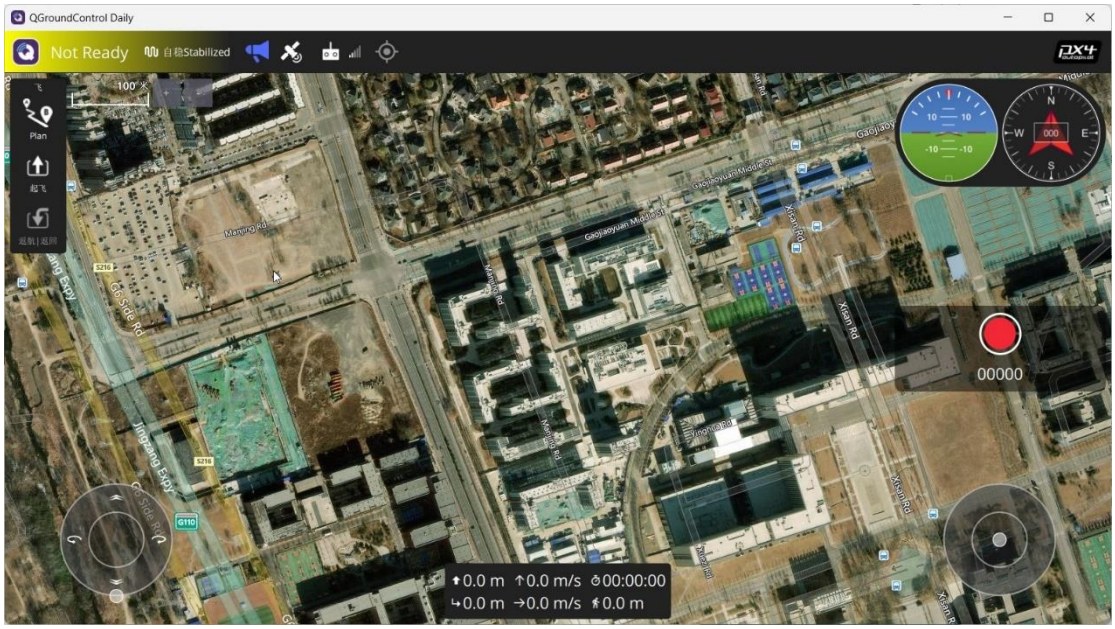
无人机是一种无人驾驶的“机器人”车辆，可以通过遥控或自主控制进行操作，被广泛应用于许多消费、工业、政府和军事领域。这些应用包括但不限于：空中拍摄/录像、运输货物、赛车、搜索和勘测等。不同类型的无人机用于空中、地面、海洋和水下。这些无人机被称为无人驾驶飞行器（UnmannedAerialVehicles，UAV）、无人驾驶系统（Unmanned AerialSystems，UAS）、无人驾驶地面车辆（UnmannedGroundVehicles，UGV）、无人驾驶水面车辆（UnmannedSurfaceVehicles，USV）和无人驾驶水下车辆（UnmannedUnderwater Vehicles，UUV）。无人机的“大脑”被称为自动驾驶系统。它由飞行堆栈软件运行在车辆控制器（“飞行控制器”）硬件上。一些无人机还配备了一台独立的机载计算机。这些计算机为网络、计算机视觉和许多其他任务提供了强大的通用计算平台。

PX4 是一款强大的开源无人机飞行控制栈，一些关键功能包括：

- 控制多种不同的车辆框架/类型，包括：无人机（多旋翼、固定翼飞机和垂直起降飞机）、地面车辆和水下车辆。
- 车辆控制器、传感器和其他外围设备的强大选择。
- 飞行模式和安全功能的灵活性和强大性。
- 与机载计算机和机器人 API（如：ROS2，MAVSDK）的深度集成。

PX4 是更广泛的无人机平台的核心部分，该平台包括基于 QGroundControl 的地面站、Pixhawk 硬件（在新窗口中打开）和 MAVSDK（在新窗口中打开），使用 MAVLink 协议与伙伴计算机、相机和其他硬件进行集成。

Dronecode 地面控制站叫做 QGroundControl。可以使用 QGroundControl 将 PX4 烧录到车辆控制硬件中，设置车辆、更改不同的参数、获取实时飞行信息并创建和执行完全自主的任务。QGroundControl 在 Windows、Android、MacOS 或 Linux 上运行。从这里下载和安装它（在新窗口中打开）。



PX4 支持空中、地面和水下车辆。可在这里查看 PX4 测试/调优的所有车辆类型和变种("框架")：空中架构参考。根据需要使用它的类型来选择不同得载具：

多旋翼：提供精确的悬停和垂直起降，但航程短，通常飞行速度较慢。PX4 有使它们易于飞行的模式，它们是最受欢迎的飞行车辆类型。

直升机：类似于多旋翼，机械上更复杂，但更有效。

固定翼：提供更长的飞行时间和更快的飞行速度，因此对地表调查等应用程序的覆盖面更好。但它们比多旋翼飞行更困难和降落，如果您需要悬停或以非常缓慢的速度飞行（例如，在调查垂直建筑物时），则不适用。

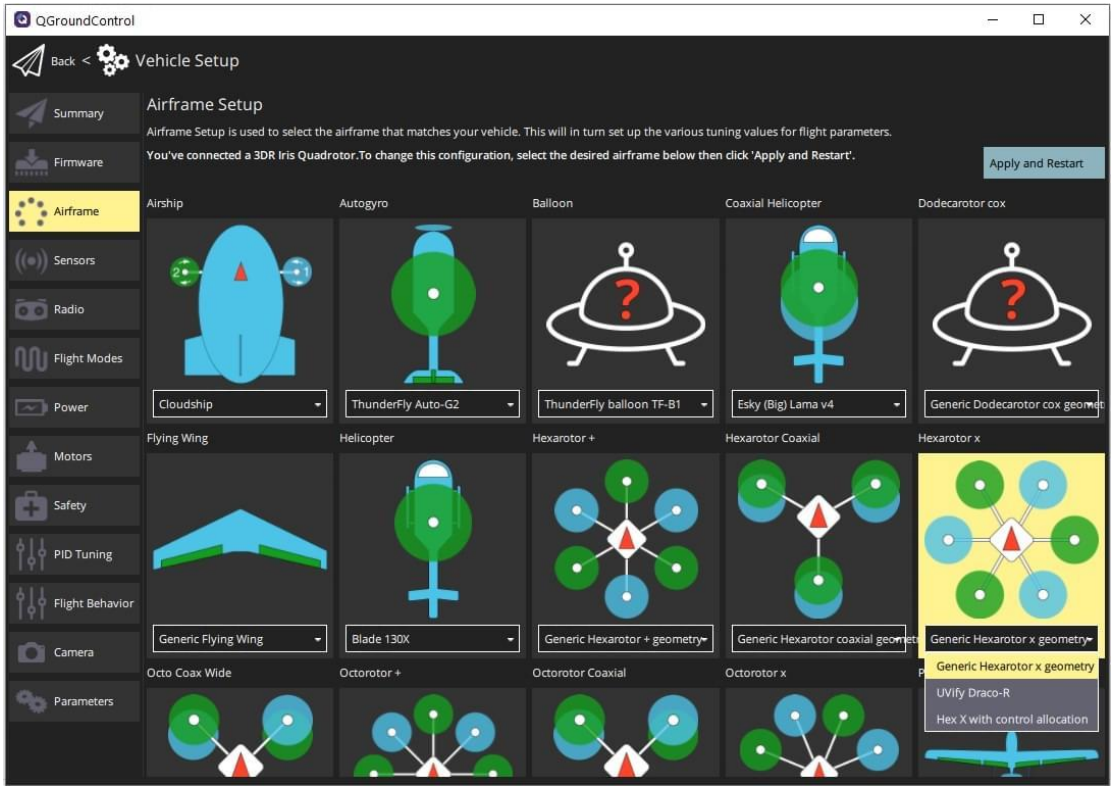
垂直起降（VTOL）无人机：有多种类型：倾斜旋翼、倒置式座位、四旋翼等。它们提供了双重优势：像多旋翼一样垂直起飞，然后像飞机一样在前进飞行中过渡。它们通常比多旋翼和固定翼飞机更昂贵，更难制造和调整。

气球/气垫车辆：是轻于空气的飞行车辆，通常提供高海拔长时间飞行，通常在航程和速度控制方面存在限制（或没有限制）。

越野车是像汽车一样的地面车辆。它们易于控制，通常很有趣。

无人船：是水面车辆。

水下无人车：是水下车辆。

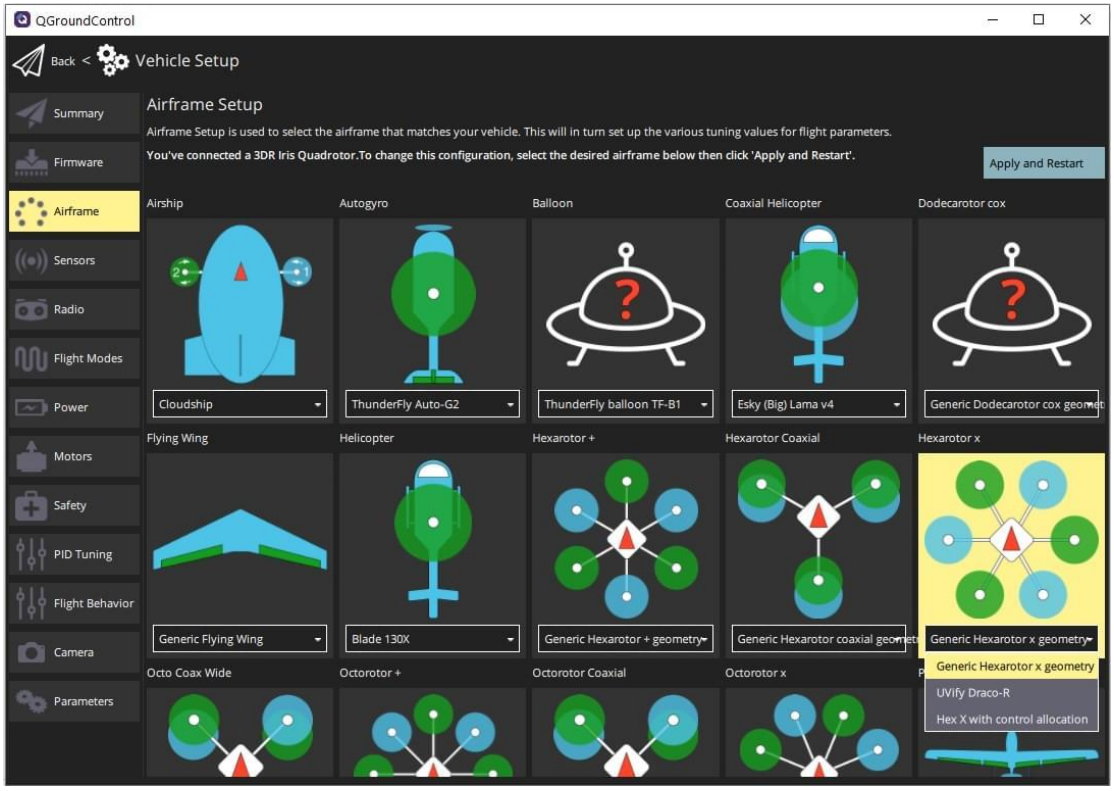


9.1.1. 固件下载

安装完成 RflySim 平台后，可在路径：*\PX4PSP\Firmware，查看到完整得 PX4 源码，打开桌面得 WSL 子系统(*\桌面\RflyTools\Win10WSL.lnk)即可进行编译，编译完成后，打开 QGC 软件进行固件下载，注：也可在网络链接得情况下，直接通过 QGC 下载固件。

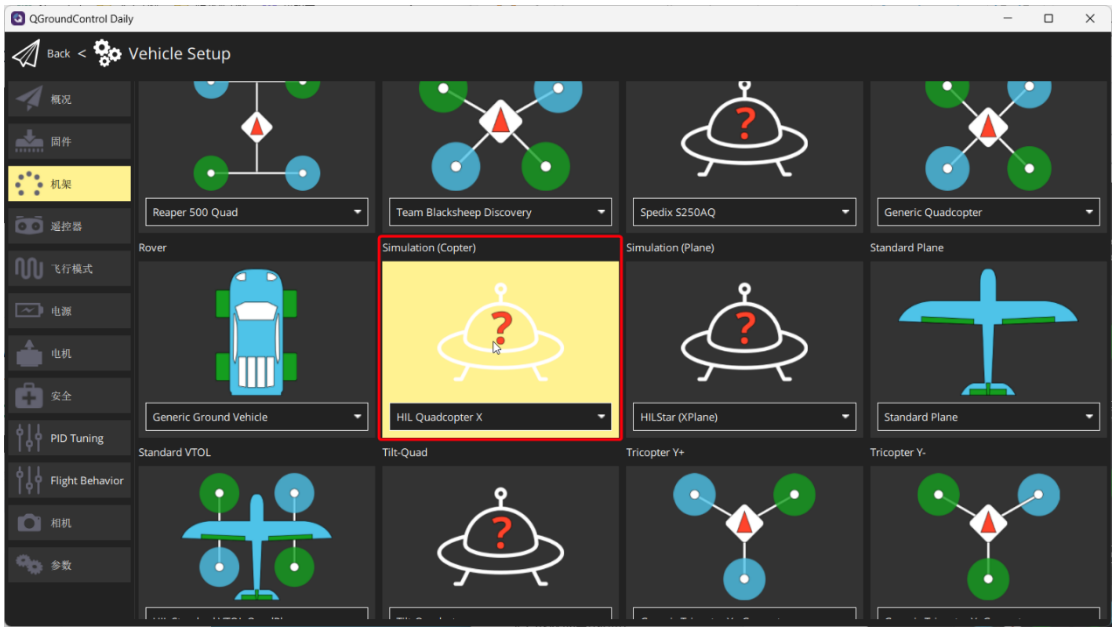
9.1.2. 机型设置

固件烧录完成之后，启动 QGroundControl 软件，连接飞控后，选择"Q"图标>车辆配置>车架（侧栏）以打开车架配置。选择与你的车架匹配的的车辆组/类型，然后在组内使用下拉菜单选择最适合您的车辆的车架。如：进行四旋翼硬件在环仿真，可选择机架为：Simulation->HILQuadcopterX。

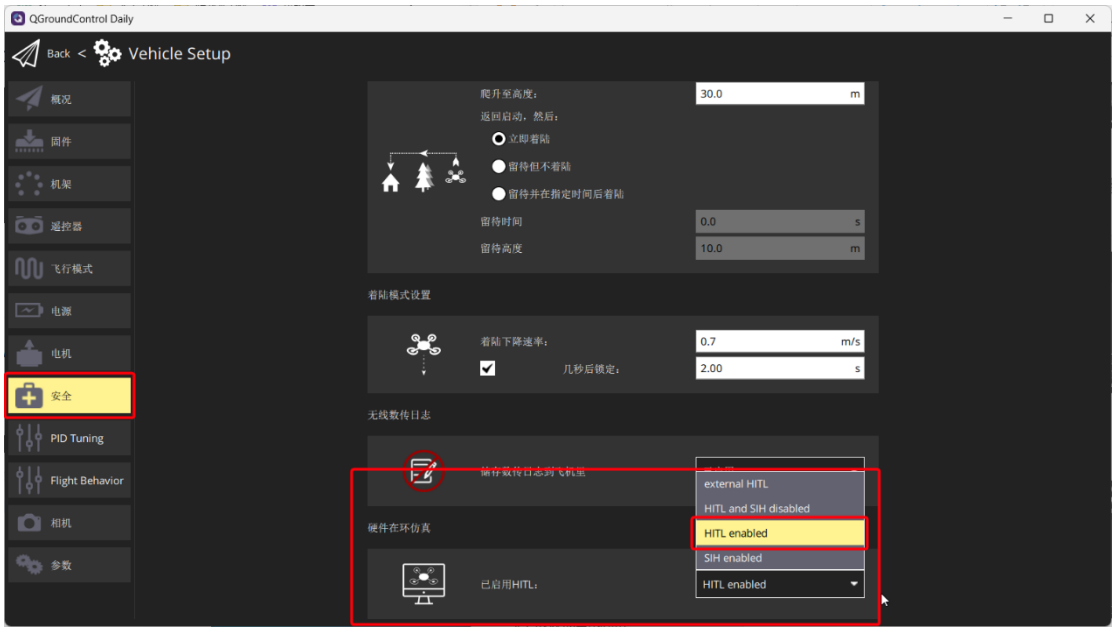


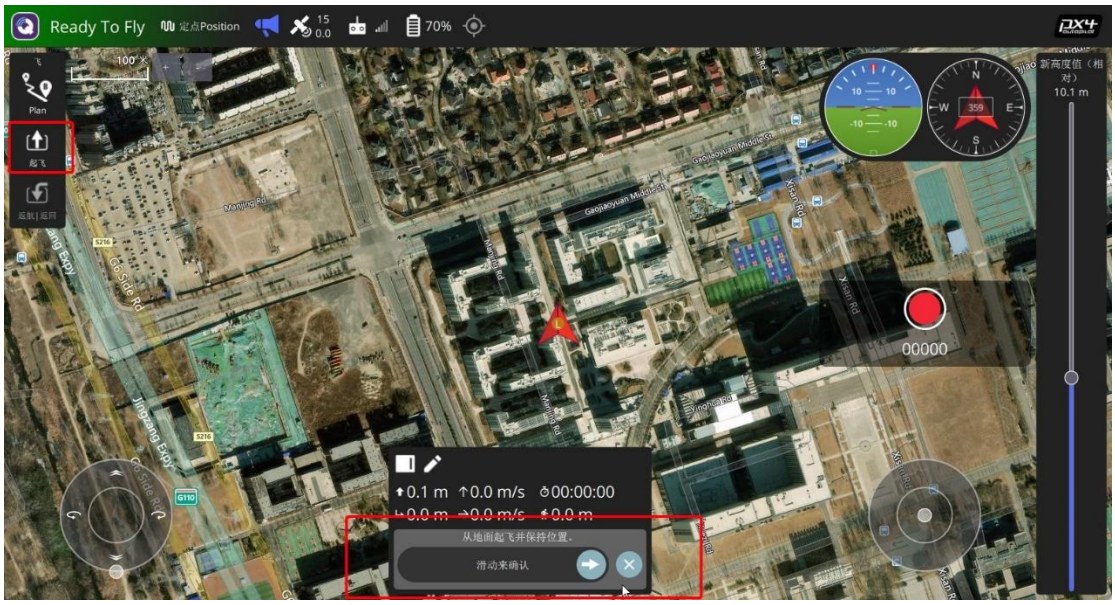
9.1.3. 硬件在环仿真

飞控链接电脑，打开 QGC 地面站，选择机架为：Simulation->HILQuadcopterX。在安全选项中设置：安全->硬件在环仿真->HITLenabled。



打开“*\桌面\RflyTools\HITLRun.lnk”一键启动硬件在环仿真脚本，输入飞控得端口号，等待 CopterSim 左下方的消息栏中显示：PX4:GPS3Dfixed&EKFinitializationfinished。即可在 QGC 中解锁起飞飞机。


QGroundControl Daily
Back
Vehicle Setup
概况
固件
机架
遥控器
飞行模式
电源
电机
安全
PID Tuning
Flight Behavior
相机
参数
爬升至高度：
30.0
m
返回启动，然后：
立即着陆
留待但不着陆
留待并在指定时间后着陆
留待时间
0.0
s
留待高度
10.0
m
着陆模式设置
着陆下降速率：
0.7
m/s
几秒后锁定：
2.00
s
无线数传日志
硬件在环仿真
已启用HITL：
external HITL
HITL and SIH disabled
HITL enabled
SIH enabled
HITL enabled


Ready To Fly
定点Position
70%
Plan
起飞
新高度值（相对）
10.1 m
00000
0.1 m
0.0 m/s
00:00:00
0.0 m/s
0.0 m
从地面起飞并保持位置。
滑动来确认

9.1.4. 飞机实飞

选择自己的载具的机架，如：四旋翼可选择机架为：QuadrotorX->DJIF450w/DJIESCs。在安全选项中设置：安全->硬件在环仿真->externalHITL。更加详细的实飞步骤请见实验：1.BasicExps\e5-AttitudeCtrl\e5.4\Readme.pdf。

9.2. PX4 官方支持飞控简介

PX4 官方支持的飞控均由 PX4-Autopilot 维护人员和 Dronecode 团队维护，符合 Pixhawk 标准，更多最新信息请见：https://docs.px4.io/main/en/flight_controller/。下表为目前部分常用飞控硬件及编译命令：

序号	板载信息	硬件名称	编译命令
1	FMUv6XandFMUv6C	CUAVPixahwkV6X(FMUv6X)	px4_fmu-v6x_default
2		HolybroPixhawk6X(FMUv6X)	
3		HolybroPixhawk6C(FMUv6C)	px4_fmu-v6c_default
4		HolybroPix32v6(FMUv6C)	
5	FMUv5andFMUv5X(STM32F7,2019/20)	Pixhawk4(FMUv5)	px4_fmu-v5_default
6		Pixhawk4mini(FMUv5)	
7		CUAVV5+(FMUv5)	
8		CUAVV5nano(FMUv5)	
9	FMUv4(STM32F4,2015)	Pixracer	px4_fmu-v4_default
10		Pixhawk3Pro	px4_fmu-v4pro_default
11	FMUv3(STM32F4,2014)	Pixhawk2	px4_fmu-v3_default
12		PixhawkMini	
13		CUAVPixhackv3	
14	FMUv2(STM32F4,2013)	Pixhawk	px4_fmu-v2_default

9.3. PX4 版本迭代更新说明

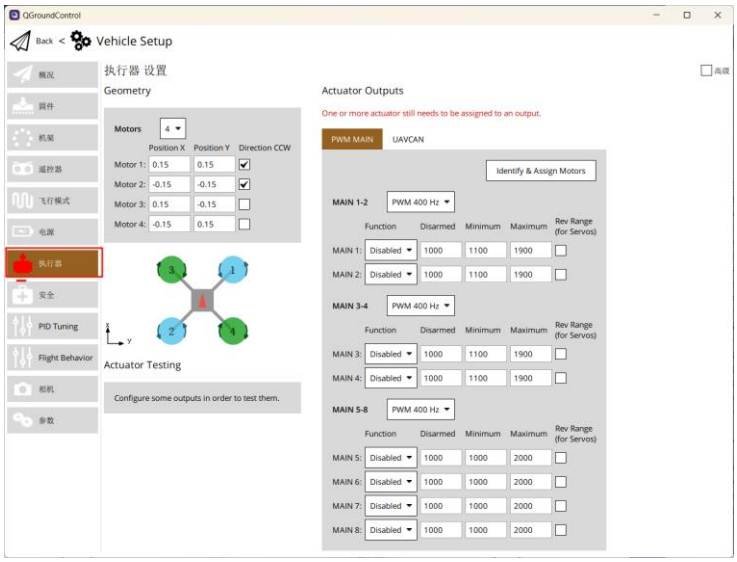
根据 PX4 官方近几年的发布节奏，基本上是一年发布一次新版本，而且新版本的迭代差距较大，RflySim 平台也已经适配最新稳定版 1.14.2 版本，可在一键安装脚本中自行选择安装。下面将详细介绍 PX4 软件最几次版本更新说明。

9.3.1. PX4-1.14 版本更新说明

v1.14 版本的更新具有一些突破性的变化，特别是通过 mixer 文件配置载具类型、电机以及驱动器的方式。此外，相较于之前的版本，v1.14 弃用了 ROS 2 使用的 Fast-RTPS 接口，转而使用更简洁的解决方案，该解决方案不需要自定义构建目标，并且省去了额外的消息生成步骤。

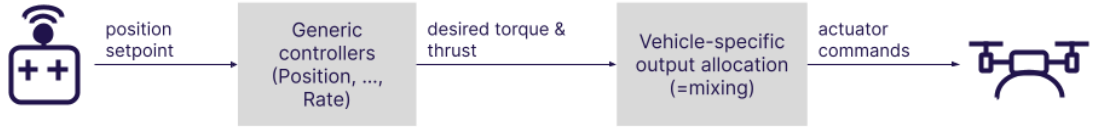
1. 动态控制分配

v1.14 版本默认启用了新的动态控制分配功能，用户能够在运行时定义载具配置，而无需使用混合器文件，RflySim 平台的 v3.02 版本及以上均已支持，可在平台自带的 QGround Control 中设置全新的车辆仪表板（飞控需烧录 v1.14.0 及以上版本）。

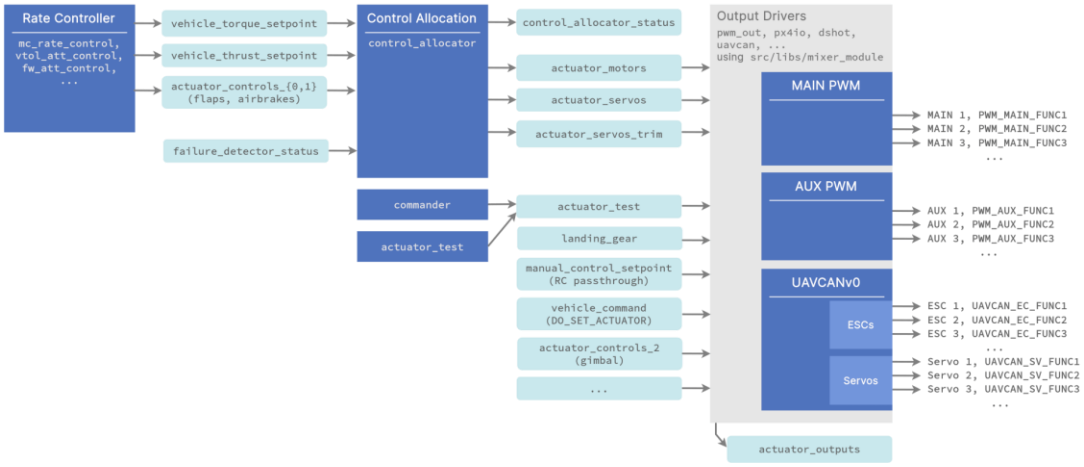


PX4 从核心控制器获取所需的扭矩和推力指令，并将其转换为控制电机或伺服器的执行器指令。转换取决于载具的类型。如：给定一个“向右转”的扭矩指令时，对于固定翼飞机来说，每副翼一个伺服器的飞机将发出一个伺服器高的指令，另一个伺服器则为低的指令；对于多旋翼飞行器，则会通过改变所有电机的速度来向右偏航。PX4 将这种被称为"混合 "的转换逻辑与姿态/速度控制器分开。这确保了核心控制器无需针对每个不同的载具进行特殊处理，并大大提高了可重用性。

此外，PX4 还将输出功能抽象为特定的硬件输出。这意味着任何电机或伺服都可以分配给几乎任何物理输出。



按模块和 uORB 主题分类列出的混控通道概览如下图所示。



从上图中可以看出：

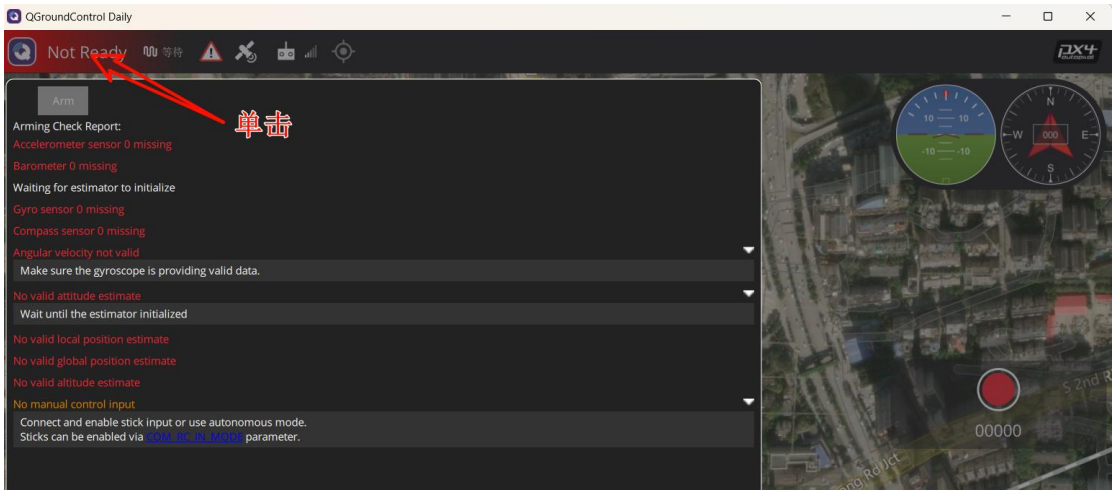
- 速率控制器输出扭矩和推力设定值
- 控制分配器模块功能：
  - 根据配置参数处理不同载具的几何形状
  - 进行混合
  - 处理电机故障
  - 发布电机和伺服控制信号
  - 单独发布伺服微调信号，以便在测试执行器（使用测试滑块）时将其作为偏移量添加。
- 输出驱动模块功能：
  - 处理硬件初始化和更新
  - 使用共享库【RflySim 安装目录】\Firmware\src\lib\mixer_module。驱动程序定义了参数前缀，如：PWM_MAIN，然后库就使用该前缀进行配置。它的主要任务是从输入话题中进行选择，并根据用户设置的`<param_prefix>_FUNCx`参数值为输出分配正确的数据。如：`PWM_MAIN_FUNC3`设置为电机 2，则第 3 个输出将被设置为执行器_电机中的第 2 个电机。
  - 输出函数定义在【RflySim 安装目录】\Firmware\src\lib\mixer_module\output_functions.yaml。
- 若要通过 MAVLink 控制输出，可将相关输出函数设置为“Offboard Actuator Set x”，然后发送 MAV_CMD_DO_SET_ACTUATOR 命令即可。

更多本模块的相关学习资料可见：https://docs.px4.io/v1.14/en/concept/control_allocation.

html

2. 改进飞行前故障检查报告

v1.14 通过事件界面改进了飞行前故障报告功能。如果飞行器无法解锁，用户可在 QGC 解锁检查用户界面中更轻松地找到原因。不用再纠结是安全开关出了问题、校准不良还是估算器内部出了问题！

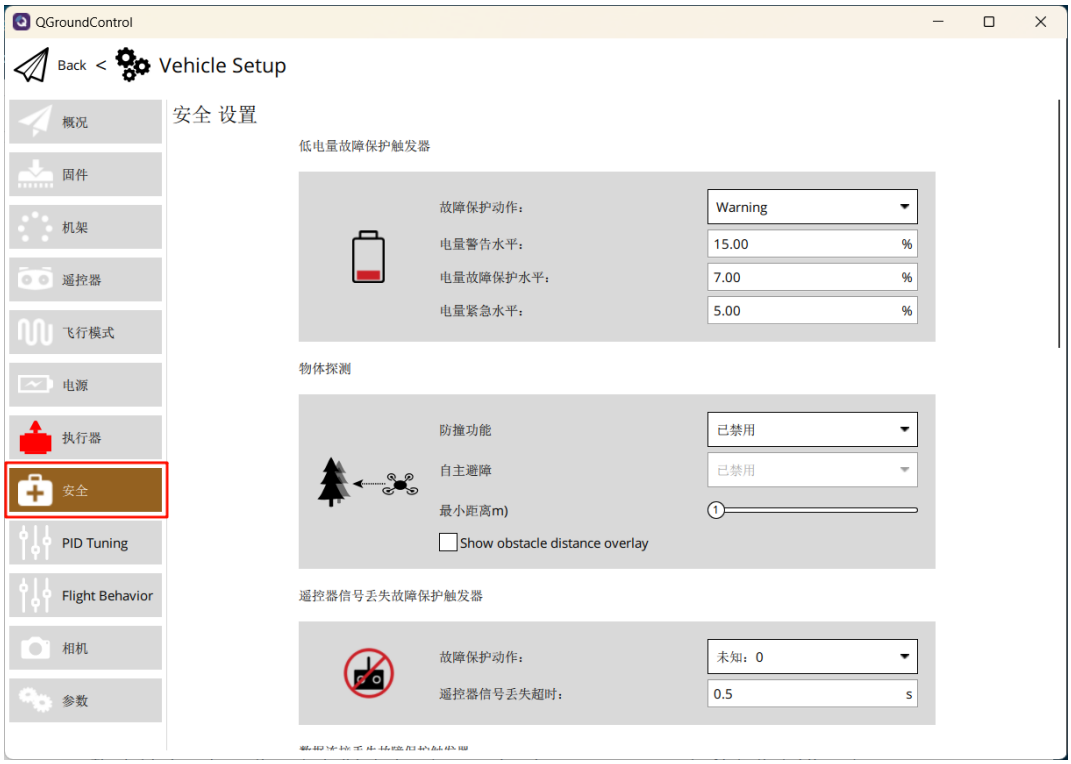


作为此更改的一部分，现在可以在解除警报时切换到任何模式（以前，如果您没有良好的位置估计，则无法切换到需要 GPS 的模式）。PX4 只有在满足当前模式的条件时才允许解锁，并将与当前模式无关的故障检查报告为警告。

更多本模块的相关学习资料可见：https://docs.qgroundcontrol.com/master/en/qgc-user-guide/fly_view/fly_view.html#arm

3. 安全配置简化与仿真

PX4 具有许多安全功能，可在出现故障时保护和恢复飞行器：故障安全保护允许用户指定可以安全飞行的区域和条件，以及触发故障安全保护后将执行的操作（例如着陆、保持位置或返回指定点）。最重要的故障安全设置可在 QGroundControl 安全设置页面中配置。其他设置必须通过参数进行配置。遥控器上的安全开关可用于在出现问题时立即停止电机或返回飞行器。



v1.14 版本中的安全失控保护处理已得到简化，特别是当一个失控保护触发时，另一个失控保护已在进行中。在执行操作前会有一个保持延迟，以便用户在需要时有时间覆盖故障安全保护。如果触发了多个故障保护，则会采取更严重的操作。如，当遥控器和 GPS 信号均已丢失时，手动失控被设置为返回模式，GCS 链路丢失被设置为着陆，则会执行着陆。通过新的故障安全状态机模拟，用户可以在所有可能的配置和条件下测试故障安全行为。

更多本模块的相关学习资料可见：https://docs.px4.io/v1.14/en/config/safety.html

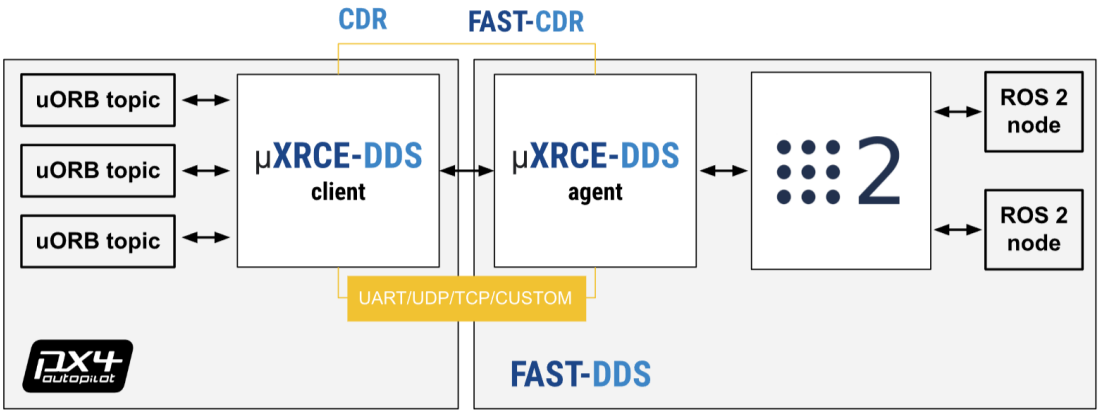
4. 默认仿真适配新版 Gazebo

RflySim 平台中内置有基于 UnrealEngine（虚幻引擎）开发的 RflySim3D 软件，该软件具有高品质的图形渲染、强大的物理模拟、灵活的蓝图系统等等优势，实现高逼真的无人系统仿真。更多本模块的相关学习资料可见：..\3.RflySim3DUE\API.pdf

5. uXRCE-DDS 改进 ROS 2 接口

v1.14 版本用 uXRCE-DDS 取代了 Fast-RTPS，从而全面改善了使用体验。这一改动还避免了`_rtps`构建目标的需要，从而在更多目标上默认启用了该界面。

ROS 2-PX4 架构提供了 ROS 2 和 PX4 之间的深度集成，允许 ROS 2 订阅者或发布者节点与 PX4 uORB 主题直接对接。由于使用了 uXRCE-DDS 通信中间件，ROS 2 的应用通道更加简单。



uXRCE-DDS 中间件包括在 PX4 上运行的客户端和在配套计算机上运行的代理，两者之间通过串行、UDP、TCP 或自定义链路进行双向数据交换。代理作为客户端的代理，负责发布和订阅全局 DDS 数据空间中的主题。

PX4`uxrce_dds_client`在构建时生成，并默认包含在 PX4 固件中。它既包括“generic”微型 XRCE-DDS 客户端代码，也包括 PX4 专用的翻译代码，用于向 uOR 话题发布或从 uORB 话题发布。生成到客户端的 uORB 消息子集在【安装目录】\Firmware\src\modules\uxrce_dds_client\dds_topics.yaml 中列出。生成器使用源代码树中的 uORB 消息定义【安装目录】\Firmware\msg 中的 uORB 消息定义来创建发送 ROS 2 消息的代码。

ROS 2 应用程序需要在具有相同消息定义的工作区中创建，这些消息定义用于在 PX4 固件中创建 uXRCE-DDS 客户端模块。用户可以将接口包 PX4/px4_msgs 克隆到 ROS 2 工作区中（软件仓库中的分支与不同 PX4 版本的信息相对应）。

更多本模块的相关学习资料可见：https://docs.px4.io/v1.14/en/ros/ros2_comm.html

关于 PX4 v1.14 版本的更多更新说明请见：https://docs.px4.io/v1.14/en/releases/1.14.html

9.3.2. PX4-1.13 版本更新说明

- 明确 Joystick 源选择：v1.13 版本添加一个名为 manual_control 的新模块，用于处理 commander 之外的 RC 和 MAVLink manual_control 输入。配置参数为 COM_RC_IN_MODE。
- v1.13 新增基于低电量的剩余飞行时间显示，根据经过大量过滤的 "平均 "电流消耗和配置的电池容量计算剩余飞行时间。
- 基于低电量 RTL 的剩余飞行时间，使用电池估算中的剩余电池时间，并将其与 RTL 所需的精确估算时间进行比较。在此之前的可配置余量触发 RTL，以便在电池耗尽之前有足够的时间返回。这将导致不同的返回时间，具体取决于平均电流消耗量和载具距离再次着陆的地点的距离。
- 改进默认电池估计参数。
- 使用距离传感器自主平缓着陆，在有距离传感器测量数据的情况下，飞行器会在着陆高度为 3 m 时减速至爬行速度，以实现更平稳的着陆。

轨道飞行模式的升级改进，具体可见：https://github.com/PX4/PX4-Autopilot/pull/18988、https://github.com/PX4/PX4-Autopilot/pull/19362、https://github.com/PX4/PX4-Autopilot/pull/19367 。

关于 PX4 v1.13 版本的更多更新说明请见：https://docs.px4.io/v1.14/en/releases/1.13.html#release-1-13

9.3.3. PX4-1.12 版本更新说明

- 基于剩余飞行范围 RTL 触发器，在考虑到飞行器速度、风速和目的地距离/方向情况下，估算返回 Home 点的时间。
- 预防地理围栏的侵犯，如果预测的当前轨迹将导致侵犯，则触发侵犯，允许重新规划飞行器到安全保持位置。
- 机架脚本:更改了设置默认值的语法。
- 安全(开关)默认关闭(电机解除武装，但舵机/襟翼可以移动)。
- 安全开关是锁定，该模式一旦被禁用，将保持禁用。
- 着陆检测器:如果存在距离传感器，将地面距离纳入着陆检测。
- 增加了对 IRC Ghost 的支持，包括遥测。

关于 PX4 v1.12 版本的更多更新说明请见：https://docs.px4.io/v1.14/en/releases/1.12.html

9.4. PX4 常用 uORB 消息简介

序号	名称	简介	文件地址
1	actuator_controls.msg	驱动器控制信号	*\PX4PSP\Firmware\msg\actuator_controls.msg
2	actuator_outputs.msg	驱动器输出信号	*\PX4PSP\Firmware\msg\actuator_outputs.msg
3	input_rc.msg	遥控器输入消息	*\PX4PSP\Firmware\msg\input_rc.msg
4	led_control.msg	控制单个或多个外部 LED。	*\PX4PSP\Firmware\msg\led_control.msg
5	vehicle_status.msg	载具状态消息	*\PX4PSP\Firmware\msg\vehicle_status.msg
6	vehicle_imu.msg	载具 IMU 输出的消息	*\PX4PSP\Firmware\msg\vehicle_imu.msg

更多 uORB 消息说明请见：https://docs.px4.io/main/en/msg_docs/

9.5. PX4 常用模块简介

序号	名称	简介	文件地址
1	mc_rate_control	多旋翼速度控制模块	*\PX4PSP\Firmware\src\modules\mc_rate_control
2	mc_pos_control	多旋翼位置控制模块	*\PX4PSP\Firmware\src\modules\mc_pos_control

3	mc_att_control	多旋翼姿态控制模块	*\PX4PSP\Firmware\src\modules\mc_att_control
4	navigator	导航控制模块	*\PX4PSP\Firmware\src\modules\navigator

更多 uORB 消息说明请见：https://docs.px4.io/main/en/

9.6. PX4 常用参数简介

序号	名称	简介
1	MPC_THR_MIN	自动推力控制中的最小推力
2	MPC_THR_HOVER	悬停推力
3	MPC_USE_HTE	悬停推力源选择器
4	MC_ROLLRATE_P	横滚角速度的比例增益
5	MC_ROLLRATE_I	横滚角速度的积分增益
6	MC_ROLLRATE_D	横滚角速度的微分增益
7	MC_ROLLRATE_FF	横滚加速度的前馈
8	MC_ROLLRATE_K	横滚角速度控制器增益，控制器的全局增益

更多 uORB 消息说明请见：https://docs.px4.io/main/en/

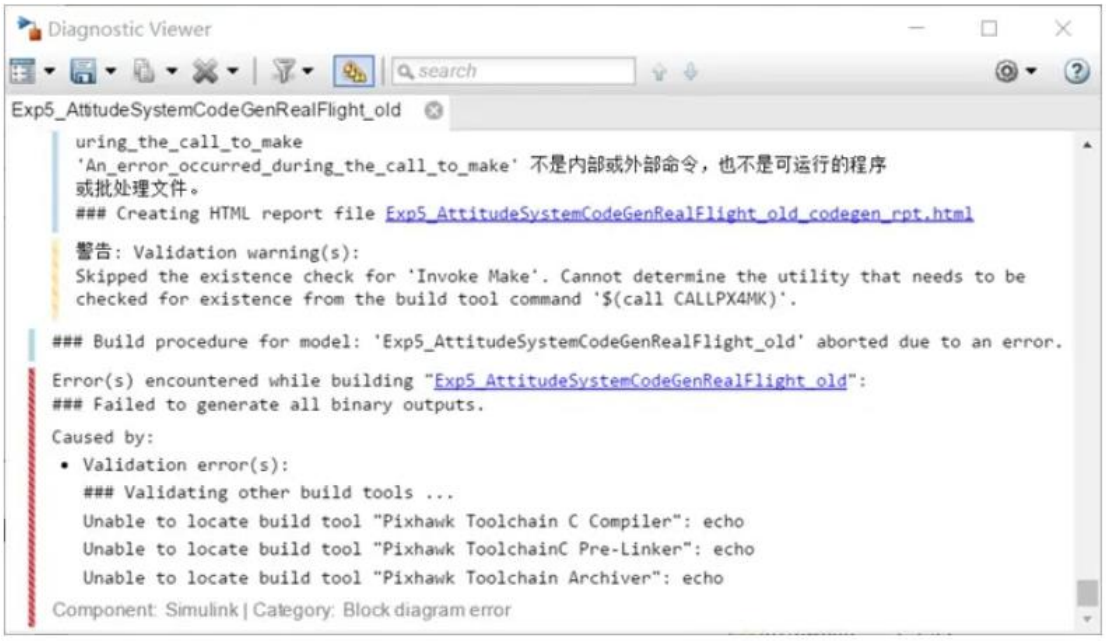
10.参考资料

[1]. 全权,杜光勋,赵峙尧,戴训华,任锦瑞,邓恒译.多旋翼飞行器设计与控制[M],电子工业出版社,2018.

[2]. 全权,戴训华,王帅.多旋翼飞行器设计与控制实践[M],电子工业出版社,2020.

11.常见问题及解答

11.1.MATLAB/Simulink 在进行代码自动生成时，有时会出现如下报错。

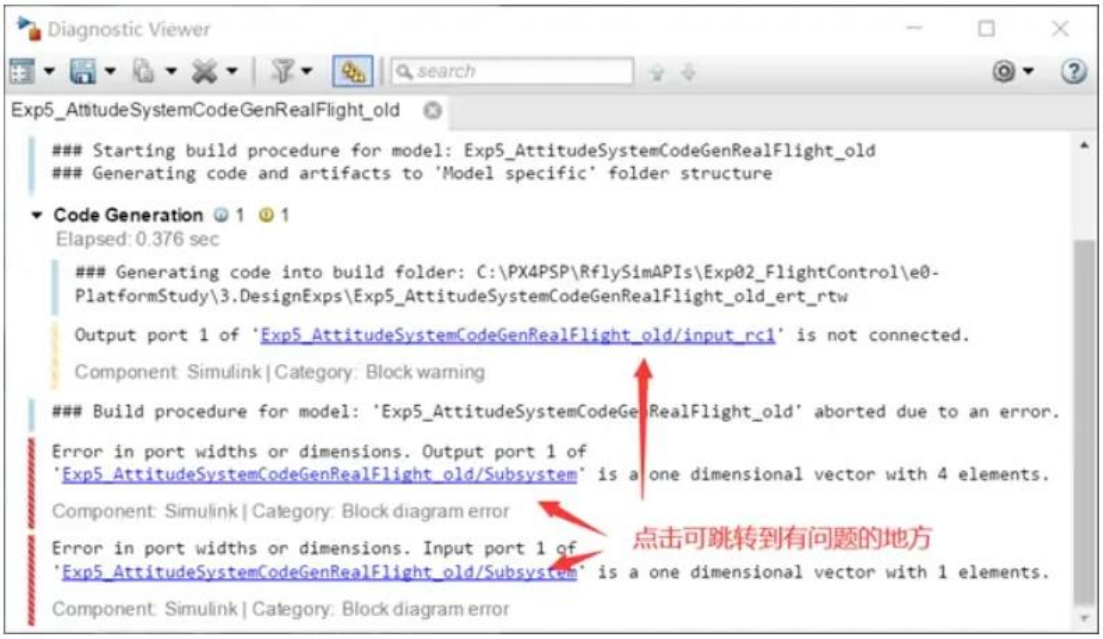



```
Causedby:
Validationerror(s):
###Validatingotherbuildtools...
Unabletolocatebuildtool"PixhawkToolchainCCompiler":echo
Unabletolocatebuildtool"PixhawkToolchainCPre-Linker":echo
Unabletolocatebuildtool"PixhawkToolchainArchiver":echo
```

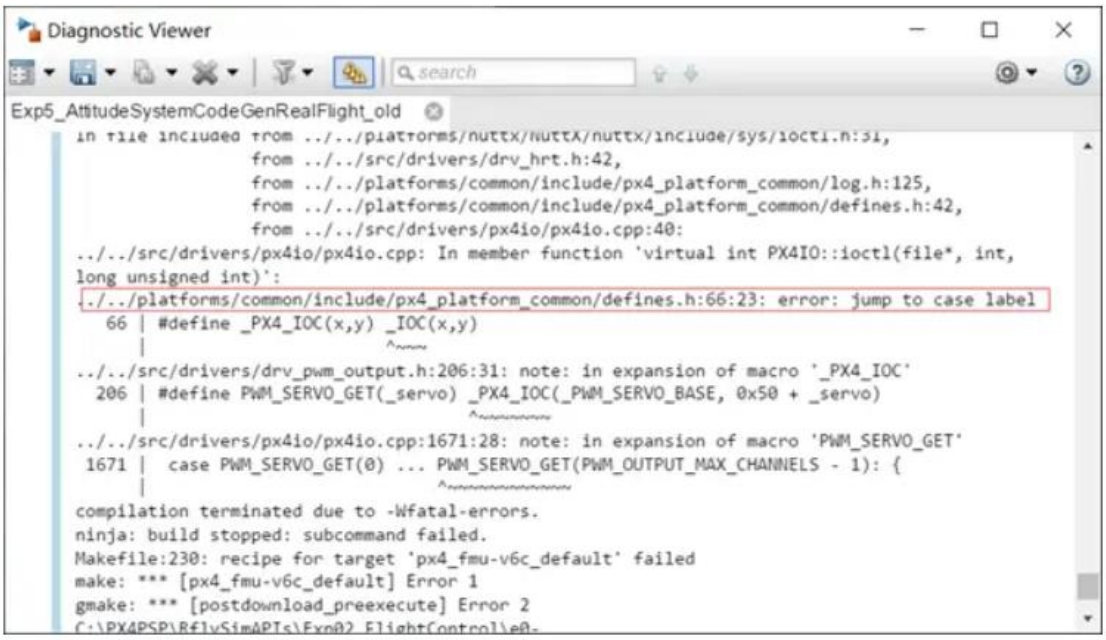
若出现编译错误，可能的编译问题可以分为：MATLAB 模型问题、PX4 固件问题、MATLAB 模型与 PX4 固件链接问题。

问题解答：

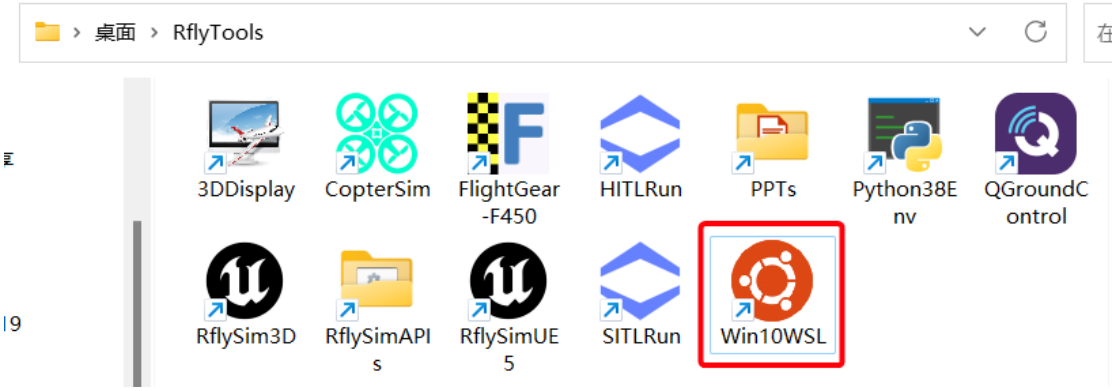
处理 MATLAB 模型问题。MATLAB 自动代码生成会在编译的初始阶段检查模型，所以这类问题往往在点击编译按钮几秒钟就会显示出来。最为常见的 MATLAB 问题是各个接口的数据不匹配，点击提示错误的模块可跳转到有问题的地方。



处理 PX4 固件问题。若 PX4 源码有编译问题，那么一般会在 MATLAB 的编译日志提示中显示出来，下图显示了问题代码出现的位置，根据提示去修改。平台中 PX4 的固件位于 PX4PSP\Firmware。



处理 MATLAB 模型与 PX4 固件链接问题。这类问题往往是 PX4 固件由于版本的升级而导致一些接口发生变化，而 MATLAB 自动代码生成的接口可能不匹配，所以在最终的链接阶段会发生错误。这类问题在 MATLAB 中看不到具体的错误，需要打开 Win10WSL(选用其它编译工具的参考其它工具)，重新执行下编译命令如：`makepx4_fmu-v6c_default`(其它版本换成其对应的命令)方可看到具体的问题。



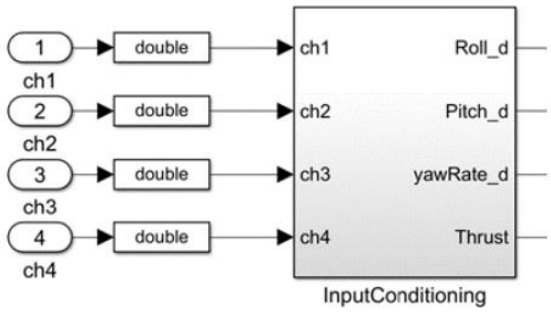
注意：在定位飞控编译问题时，应该保证平台是正确安装的，代码版本和编译命令能够相匹配。

11.2.在自动代码生成控制器中，用延迟模块直接生成控制指令，导致飞机乱飞。

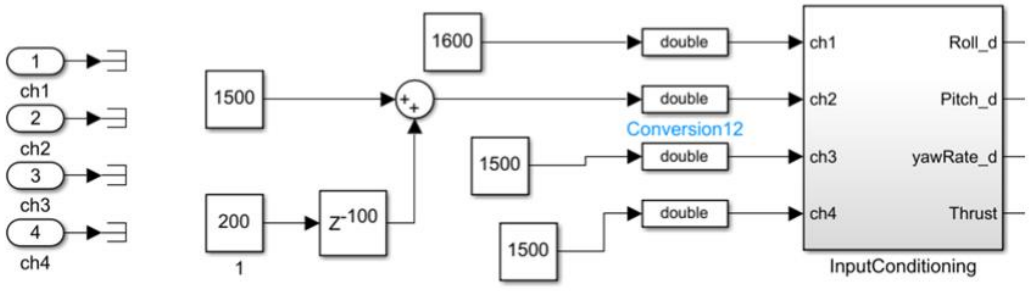
问题描述：

在对例程：[1.BasicExps\e5-AttitudeCtrl\Readme.pdf](1.BasicExps\e5-AttitudeCtrl\Readme.pdf) 将遥控器输入 CH1-CH4 与 inputConditioning 模块断开，更改滚转（CH1）、俯仰（CH2）、油门（CH3）、偏航（CH4）的输入信号为我们自己想要的定值输入，只对 AttitudeControl_HIL.slx 做了这一个地方的修改，然后跟原 AttitudeControl_HIL.slx 例程一样的操作步骤对其进行硬件在环仿真，启动硬件在环后，飞机自己在地上乱飞，过一会后会出现报错，这是什么原因？该怎么解决？

修改前：

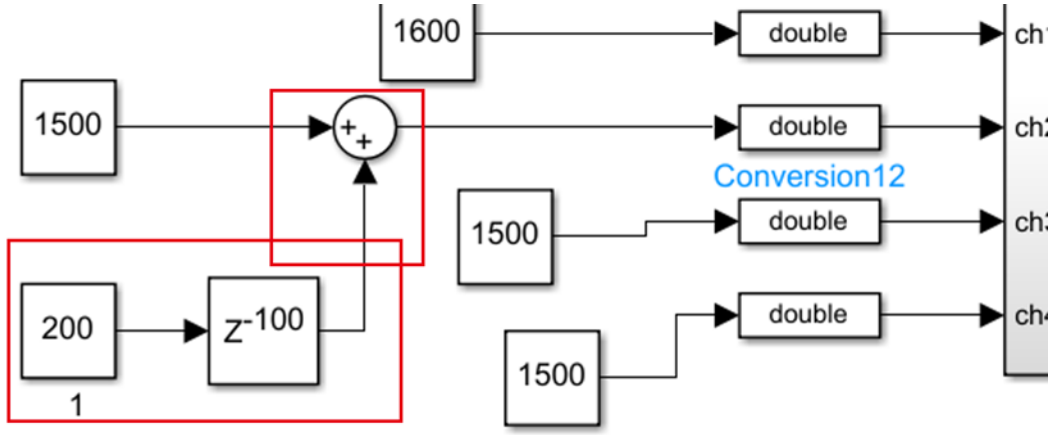


修改后：



仿真现象：更改后，进行硬件在环仿真时，无人机在地上乱飞。

这种修改方式不可行，因为这个 200 个 step 的延迟，总共才 200*0.001=0.2 秒。因为 px4_simulink_app 程序是上电就开始运行的，因此相当于上电之后，过一下就输入了控制指令，变成满油门状态了。这种情况下，飞控还没初始化完毕，滤波器触发发散状态，没有得到准确的位姿信息。



完美方案：直接读取 EKF 状态的 uORB 消息，判断滤波器初始化完毕后，再延迟一段时间给控制指令。简单方案：把延迟时间再拉大一点，或者直接用 MATLAB 函数写一个触发机制，等仿真时间达到 60s 以后，再给控制指令。

11.3.SIL 或 HIL 仿真时，RflySim3D 出现：Fatalerror:[File:D://Build/++UE4....]...报错

具体报错界面如下所示：

问题解答：

上述 RflySim3D 的报错问题，可能是由于电脑的显卡驱动所造成的兼容性问题，建议升级显卡驱动到最新版，看下能否解决；如果不能解决，请联系 RflySim 平台相关售后人员。

11.4.如何做无人机姿态自主控制？

问题描述：

详细描述为：我们需要让无人机飞到一定高度定住，然后跟据我们自己给定的姿态角期望值输入，让无人机自主去动作，使三个姿态角动作到给定的期望值。

问题解答：

遥控器的通道可以用于传递触发信息，例如，CH5 拨到满位置，程序就开始自动执行，在 Simulink 里面写个状态机，让飞机先起飞，等达到高度后，再且姿态控制。具体的控制器设计方法，属于 Simulink 编程范畴，可查阅相关文献资料进行了解。如果切换定高后，想从外部输入期望姿态角之类信息，请使用 0.ApiExps\9.PX4CtrlExternalTune\Readme.pdf 这个目录里面的外部控制接口，在飞控内部订阅 rfly_ctrls 的消息（外部程序传入的），作为期望姿态角即可。

11.5.姿态控制硬件在环的结果数据怎么得到？目前我了解的下载飞行日志的方式可以得到我想要的吗？

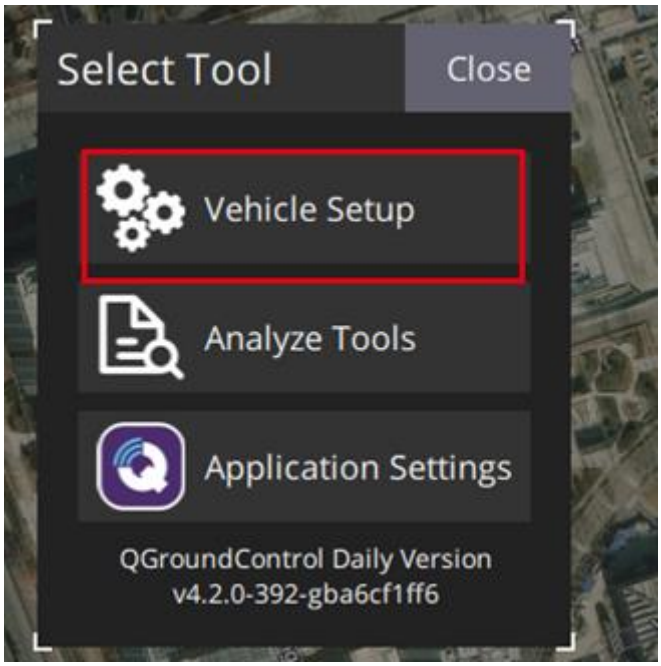
问题解答：

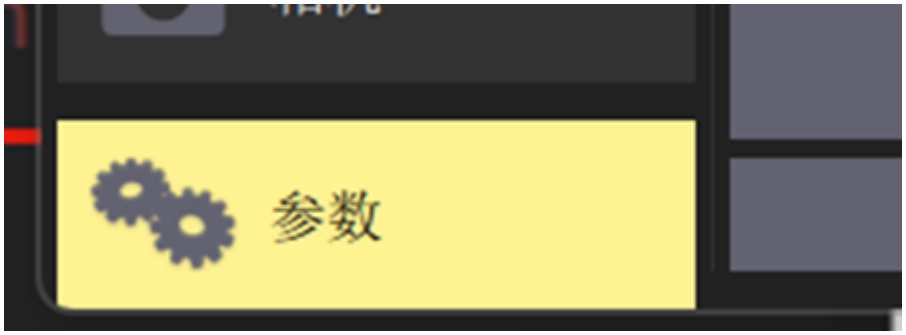
请使用 0.ApiExps\9.PX4CtrlExternalTune\Readme.pdf 例程的接口，可以在 Simulink 里面实时获取模型的真实姿态（仿真值），和飞控内部传出的数据（填入飞控姿态），飞行日志也可以得到类似数据。

11.6.QGC 的 AnalyzeTools-飞行日志，下载时刷新完之后，找不到我做硬件在环对应时间的日志

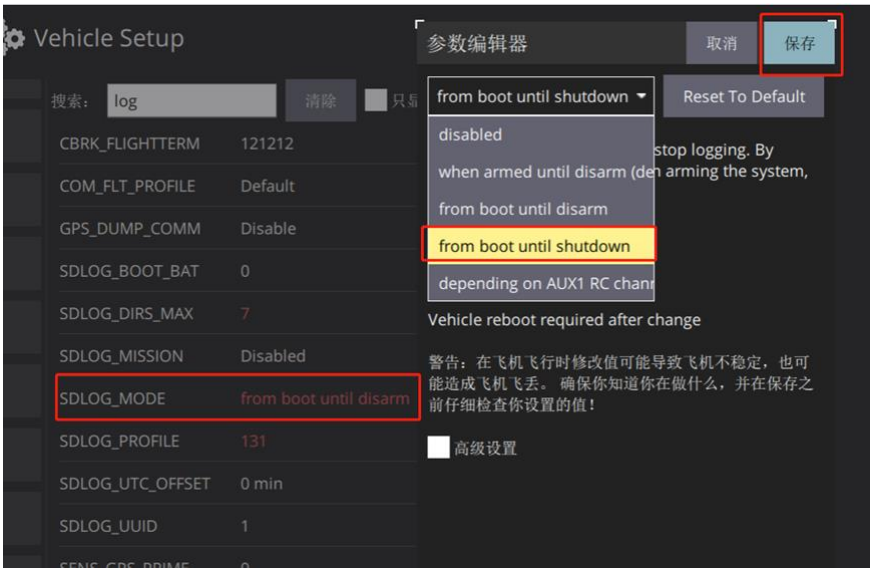
问题解答：

打开 QGC，进入载具设置



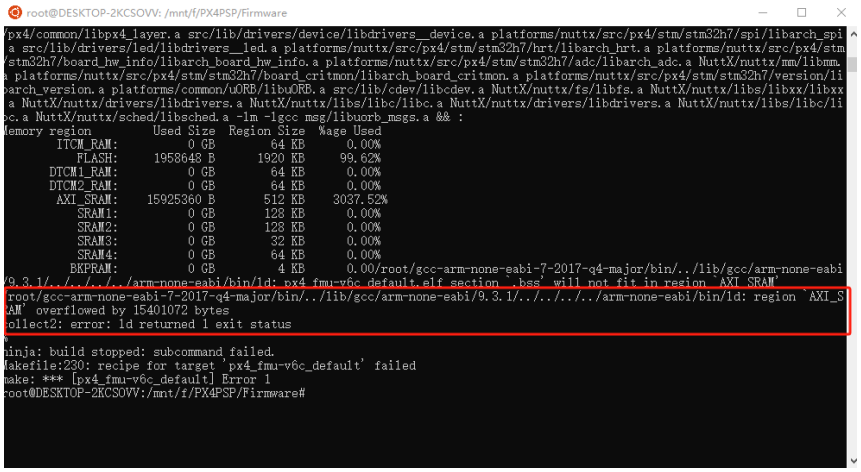


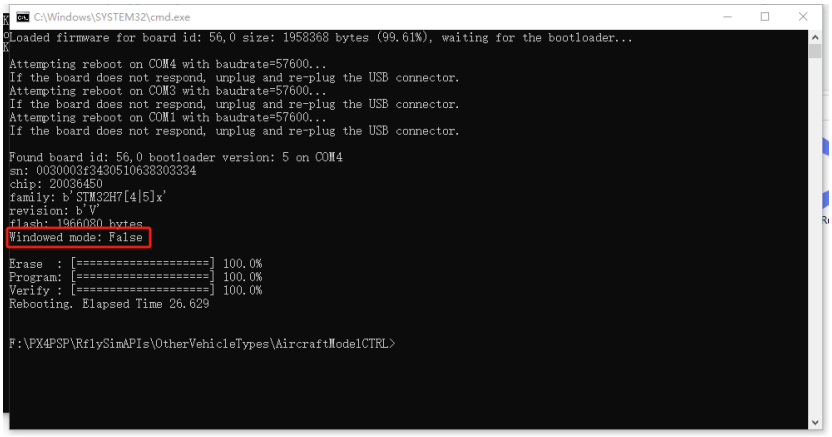
到最下面的参数标签，搜索“log”，找到了 SDLOG_MODE 的参数，将其改成下图，从开机到关机，一直记录 log，这样就能看到 log 日志了。



11.7.Win10WSL 编译固件时，显示：region`AXI_SRAM'overflowed by15401072bytes

问题描述：





问题解答：

该问题为所编译的固件较大，超出了飞控的内容，造成固件溢出报错。可进入 PX4 的

Cmake 文件中注释掉部分未在使用的模块，Pixhawk 系列飞控具体地址为：*PX4PSP\Firmware\boards\px4。如：所进行的实验为四旋翼相关底层控制算法开发实验，所使用的飞控为 Pixhawk6C，可对 C:\PX4PSP\Firmware\boards\px4\fmu-v6c\default.cmake，该文件中的未使用的模块进行注释(如下)，然后再编译。

```
……
    MODULES
        airspeed_selector
        px4_simulink_app
        attitude_estimator_q
        camera_feedback
        commander
        dataman
        ekf2
        esc_battery
        events
        flight_mode_manager
        #fw_att_control 注释掉固定翼姿态控制模块
        fw_pos_control_l1
        gyro_calibration
        gyro_fft
        land_detector
        landing_target_estimator
        load_mon
        local_position_estimator
        logger
        mavlink
        mc_att_control
        mc_hover_thrust_estimator
        mc_pos_control
        mc_rate_control
        #micrortps_bridge
        navigator
        rc_update
        rover_pos_control
        sensors
        sih
        temperature_compensation
        #uuv_att_control 注释掉 UUV 姿态控制模块
        #uuv_pos_control 注释掉 UUV 位置控制模块
        vmount
        #vtol_att_control 注释掉 VTOL 姿态控制模块
……
```