uORB 消息读取与写入实验

1. 总体说明.....1
   1.1. 文件目录.....1
   1.2. 预备知识.....1
   1.3. 关键知识点.....1
2. 关键子模块介绍.....2
   2.1. uORB Read Async—获取与 uORB Topic 相关的数据.....2
   2.2. uORB Read Function-CallTrigger—uORB 消息读取回调函数触发模块.....2
   2.3. uORB Write—uORB 消息数据发布接口模块.....4
   2.4. uORB Write Advanced—uORB 消息数据发布接口高级模块.....6
   2.5. uORB Write Advanced_dai—uORB 消息数据发布接口进阶模块.....7
   2.6. 其他模块.....7
3. 实验原理.....9
   3.1. uORB 消息读取实验原理.....9
   3.2. uORB 消息读取触发实验原理.....13
   3.3. uORB 发送接口使用实验原理.....15
   3.4. uORB ID 与文件名不一致处理实验原理.....17
   3.5. 自定义 uORB 消息实验.....19
4. 相关资料.....23

附加资源.....23

1. 总体说明

1.1. 文件目录

例程目录：[安装目录]\5.RflySimFlyCtrl\0.ApiExps\6.uORB-Read-Write

文件夹/文件名称	说明
Readme.pdf	uORB 消息读取与写入实验步骤
Exp1_uORB_read.slx	uORB 消息读取程序
Exp2_uORB_read_func.slx	uORB 消息读取触发程序
Exp3_uORB_write.slx	uORB 发送接口使用程序
Exp4_uORB_DifID.slx	uORBID 与文件名不一致处理程序
PX4ExtMsgSender.slx	飞控 uORB 消息监听程序

例程目录：[安装目录]\5.RflySimFlyCtrl\0.ApiExps\7.uORB-Create

文件夹/文件名称		说明
msg	PX4uORBMsgGen.m	uORB 消息类型生成程序。
	rfly_test.msg	uORB 消息数据结构体文件。
init_control.m		初始化文件。
px4demo_uORB_create.slx		自定义 uORB 消息定义模型文件。

1.2. 预备知识

uORB(MicroObjectRequestBroker,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。此外还需要了解 PX4 软件系统，PX4 官网连接：https://docs.px4.io/main/zh/index.html。

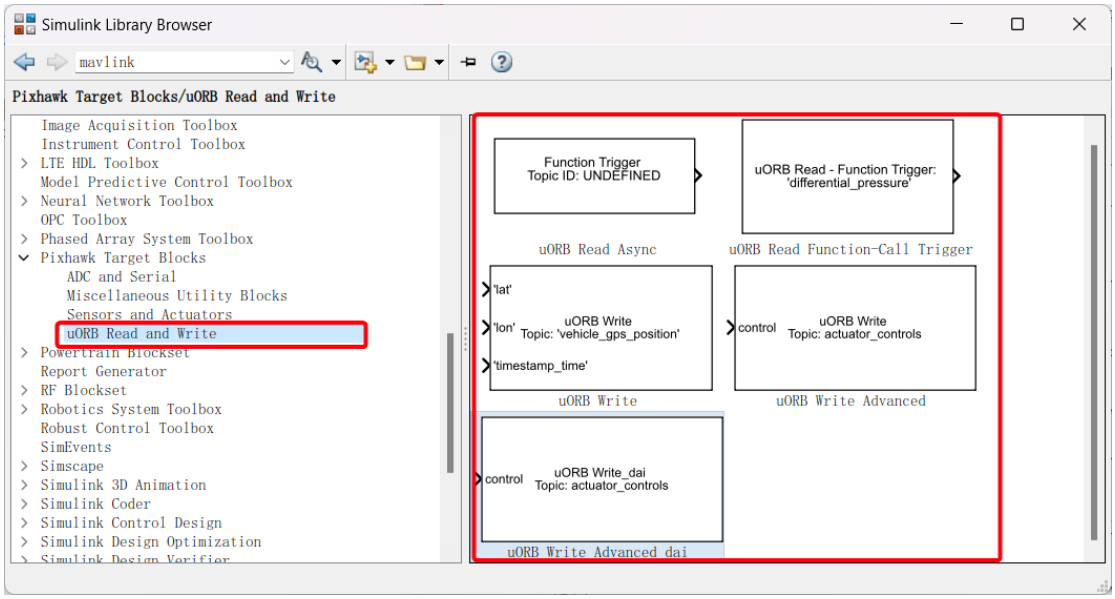
PX4 软件系统可以分为若干个小模块，每个模块独立运行（多线程并行），各个模块通过 uORB 消息模块的订阅与发布功能实现数据的传输与交互。更多信息可以参考 API.pdf。

1.3. 关键知识点

- Simulink 库浏览器中 uORB Read and Write 每个子模块的功能和运行机制。
- 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。
- 结合基于模型设计的思想，通过 uORB 订阅功能可以从消息池中读取感兴趣的话题，应该如何构建 Simulink 模型。
- Simulink 生成的代码部署到 PX4 自驾仪软件之后，如何在 PX4 自驾仪上运行。
- RflySim 平台软硬件在环仿真。

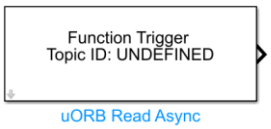
2. 关键子模块介绍

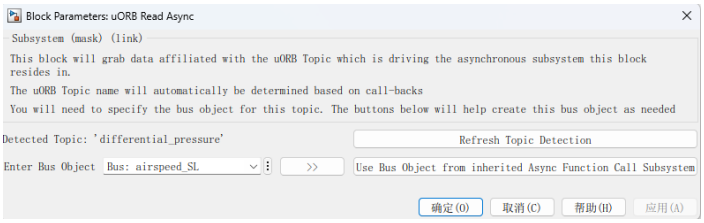
如下图所示，在 Simulink 库浏览器中可以看到 uORB Read and Write 每个子模块。



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

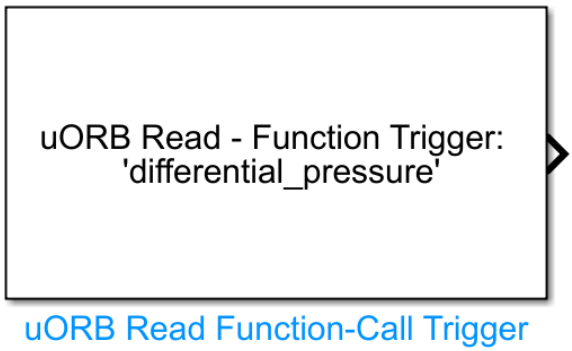
如下图所示，该块将获取与 uORB 主题相关的数据，该主题驱动该块所在的异步子系统。主题名称将根据回调自动确定，并且需要为该主题指定总线对象。更多信息可点击对话框的“帮助”按钮查看。



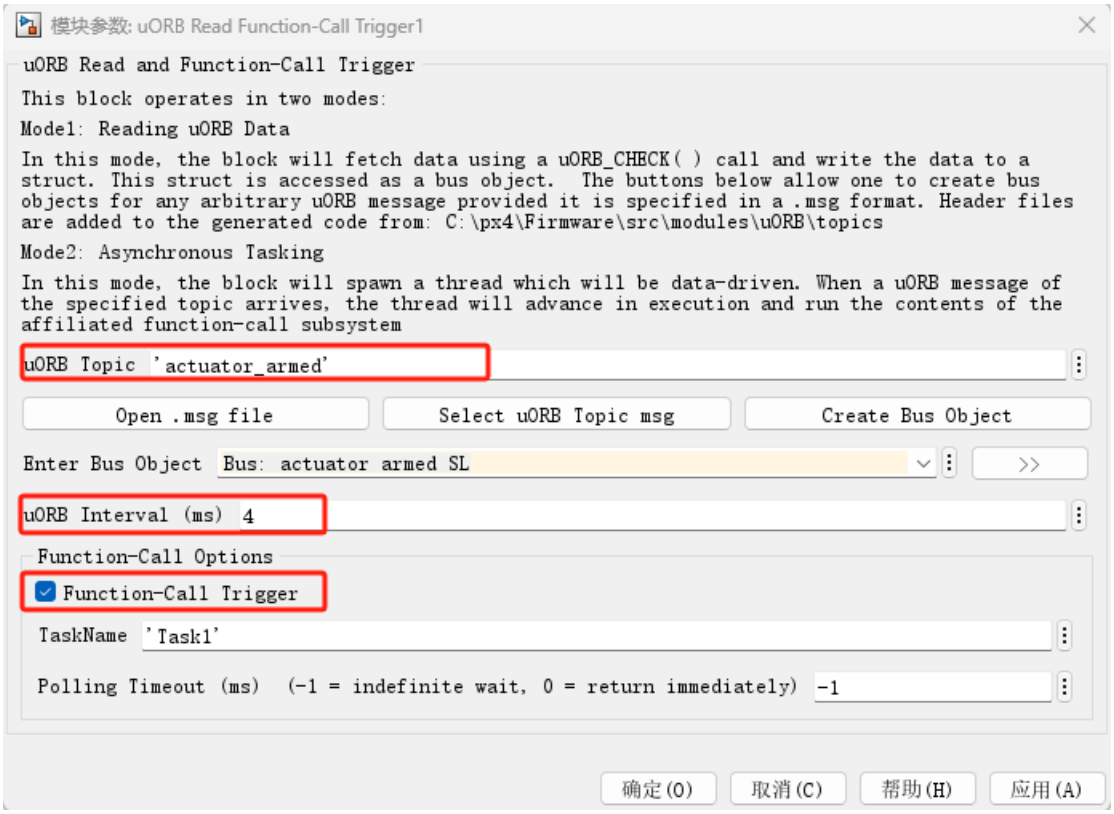


2.2. uORB Read Function-CallTrigger—uORB 消息读取回调函数触发模块

如下图所示，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。



如下图所示，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

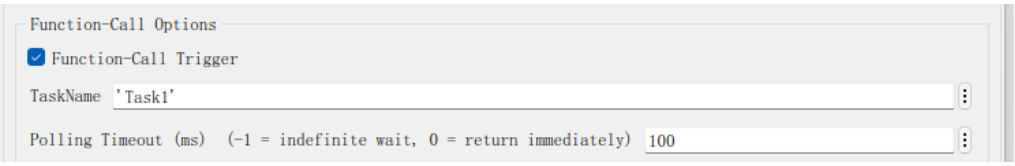
  Simulink 的总线对象用来接收 uORB 的消息，点击按钮“Create Bus Object”，Simulink 将从.msg 文件夹中找到对应的消息文件，并将其映射到 MATLAB 工作空间生成总线对象。

- 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。

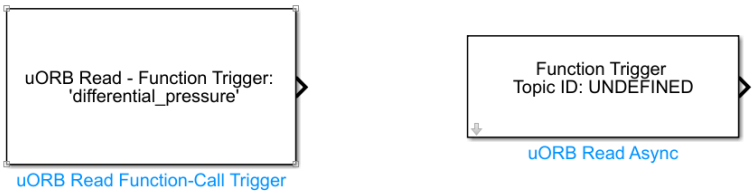
第二个功能的使用步骤为：

- 选择函数调用触发器



- 设置查询超时和任务名

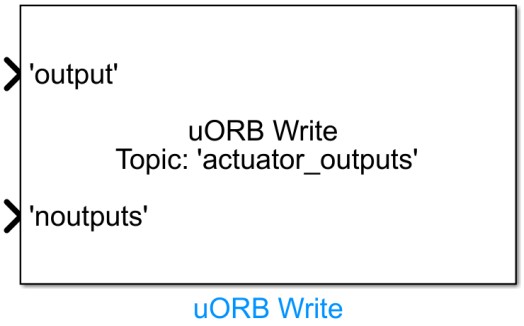
当选择了异步功能，采样时间设置框就消失了，这个时候需要设置查询超时参数和任务名。异步功能会衍生出一个新的线程，负责运行和函数触发信号相关的代码，它会通过查询来等待话题上新的数据的到来。此时需要另一个模块来读取话题上的数据，即“读 uORB 函数触发器数据模块”，如下图所示。

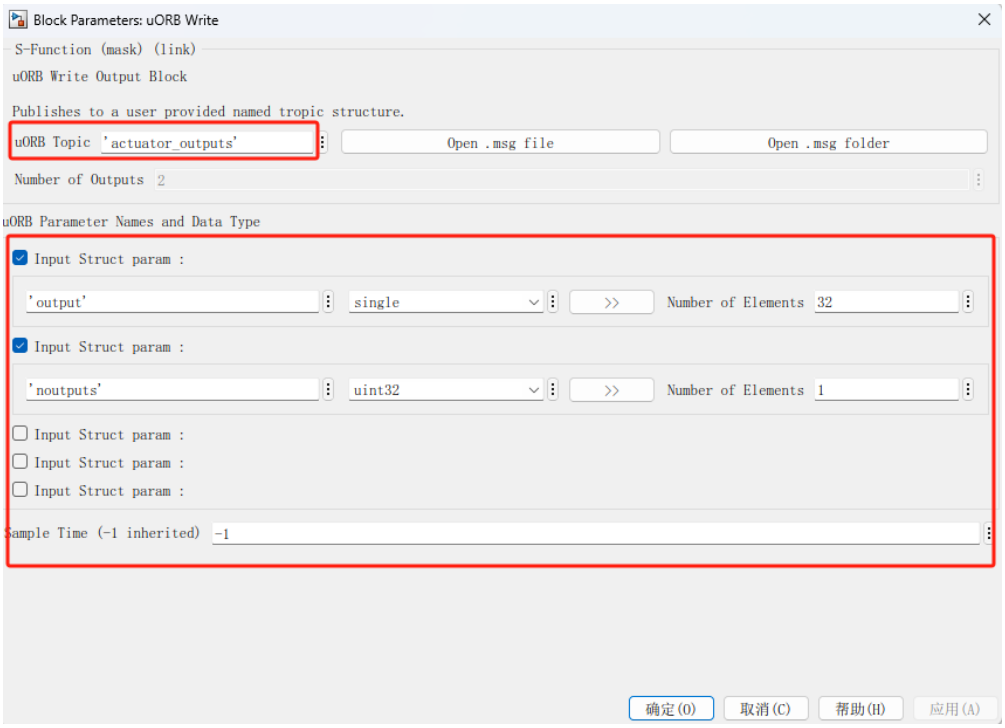


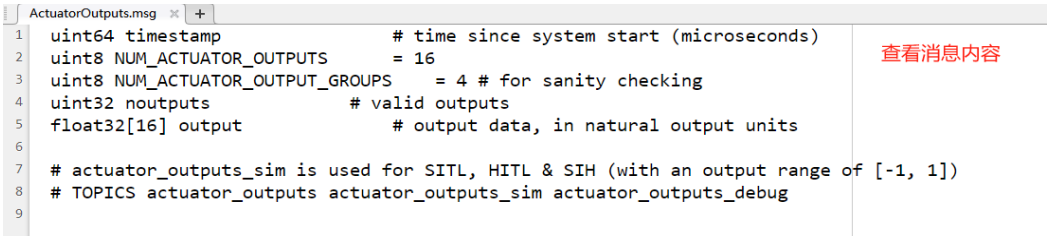
2.3. uORB Write—uORB 消息数据发布接口模块

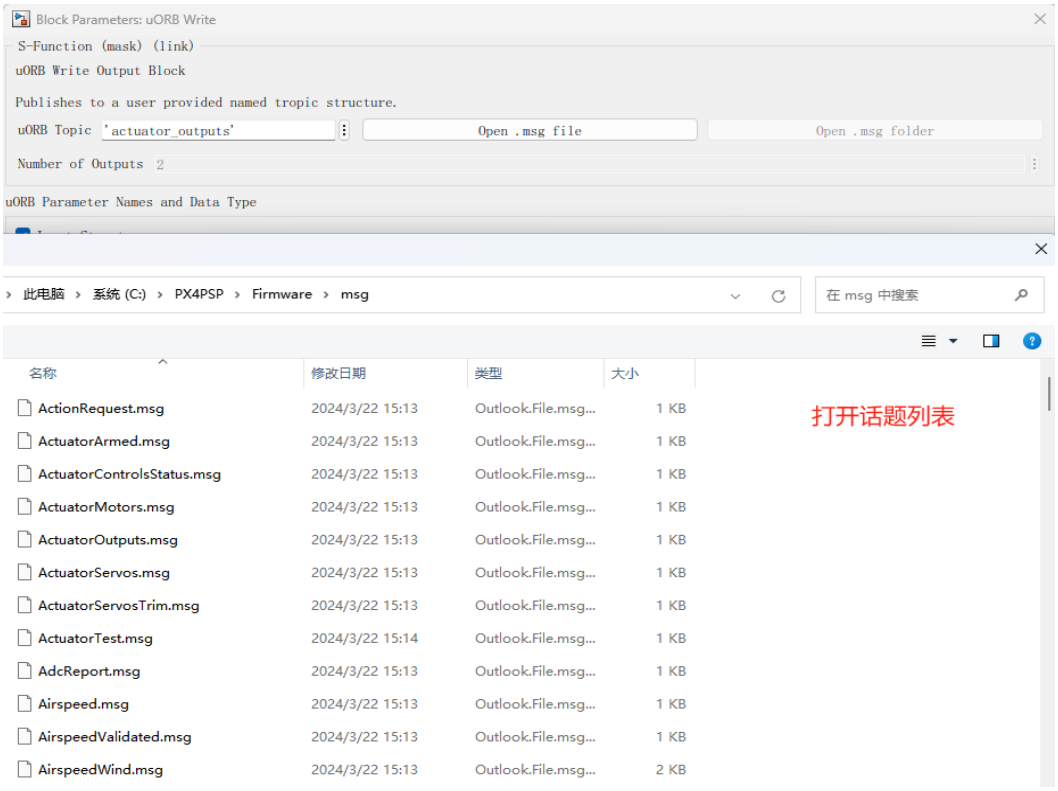
该接口允许用户向 uORB 话题发布指定的值或结构体，通过这个模块可以向某个 uORB 话题发布对应的消息，话题必须经过正确的定义，一些已定义的话题放在目录 C:\PX4PSP\Firmware\msg 下，在生成的代码中会自动包含话题的定义文件。

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。



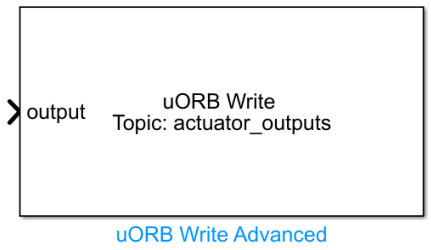

Block Parameters: uORB Write
S-Function (mask) (link)
uORB Write Output Block
Publishes to a user provided named tropic structure.
uORB Topic 'actuator_outputs'
Open .msg file
Open .msg folder
Number of Outputs 2
uORB Parameter Names and Data Type
Input Struct param :
'output'
single
>>
Number of Elements 32
Input Struct param :
'noutputs'
uint32
>>
Number of Elements 1
Input Struct param :
Input Struct param :
Input Struct param :
Sample Time (-1 inherited) -1
确定(O)
取消(C)
帮助(H)
应用(A)


ActuatorOutputs.msg
uint64 timestamp # time since system start (microseconds)
uint8 NUM_ACTUATOR_OUTPUTS = 16
uint8 NUM_ACTUATOR_OUTPUT_GROUPS = 4 # for sanity checking
uint32 noutputs # valid outputs
float32[16] output # output data, in natural output units
actuator_outputs_sim is used for SITL, HITL & SIH (with an output range of [-1, 1])
TOPICS actuator_outputs actuator_outputs_sim actuator_outputs_debug
查看消息内容


Block Parameters: uORB Write
S-Function (mask) (link)
uORB Write Output Block
Publishes to a user provided named tropic structure.
uORB Topic 'actuator_outputs'
Open .msg file
Open .msg folder
Number of Outputs 2
uORB Parameter Names and Data Type
此电脑 › 系统 (C:) › PX4PSP › Firmware › msg
在 msg 中搜索
名称
修改日期
类型
大小
ActionRequest.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
ActuatorArmed.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
ActuatorControlsStatus.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
ActuatorMotors.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
ActuatorOutputs.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
ActuatorServos.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
ActuatorServosTrim.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
ActuatorTest.msg 2024/3/22 15:14 Outlook.File.msg... 1 KB
AdcReport.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
Airspeed.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
AirspeedValidated.msg 2024/3/22 15:13 Outlook.File.msg... 1 KB
AirspeedWind.msg 2024/3/22 15:13 Outlook.File.msg... 2 KB
打开话题列表

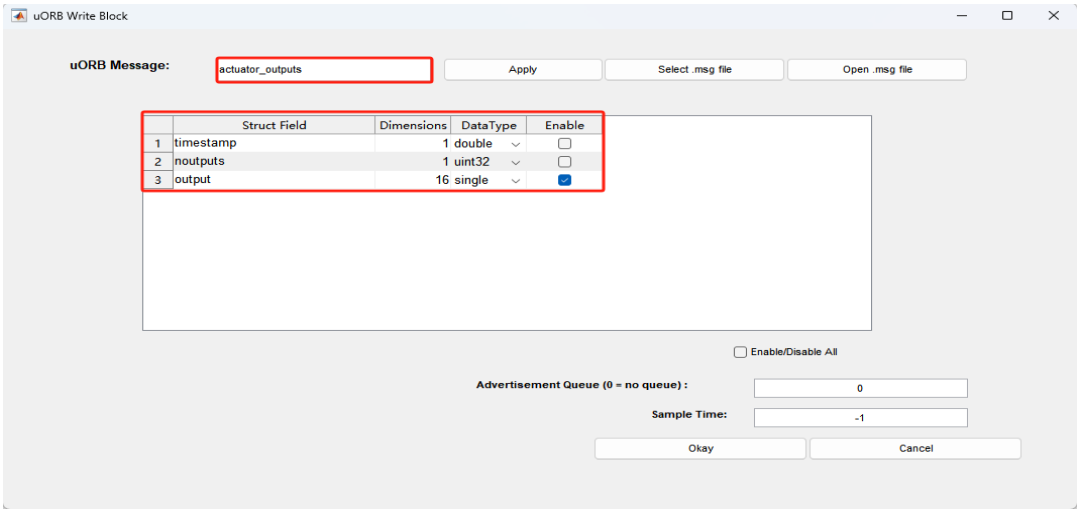
2.4. uORB Write Advanced—uORB 消息数据发布接口高级模块

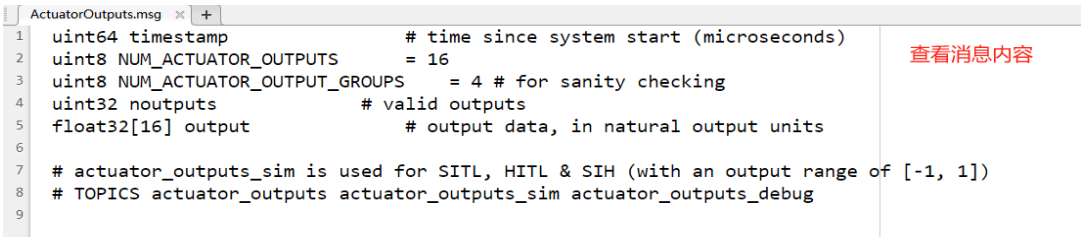
该接口可允许用户对其发布的数据进行更灵活的控制。在 Simulink 中使用 uORB Write Advanced 接口，可以实现更加复杂和精确的消息发布方式。可选择要写入的消息文件和一个消息 ID。此外，还可以设置消息的优先级、队列大小等高级选项。

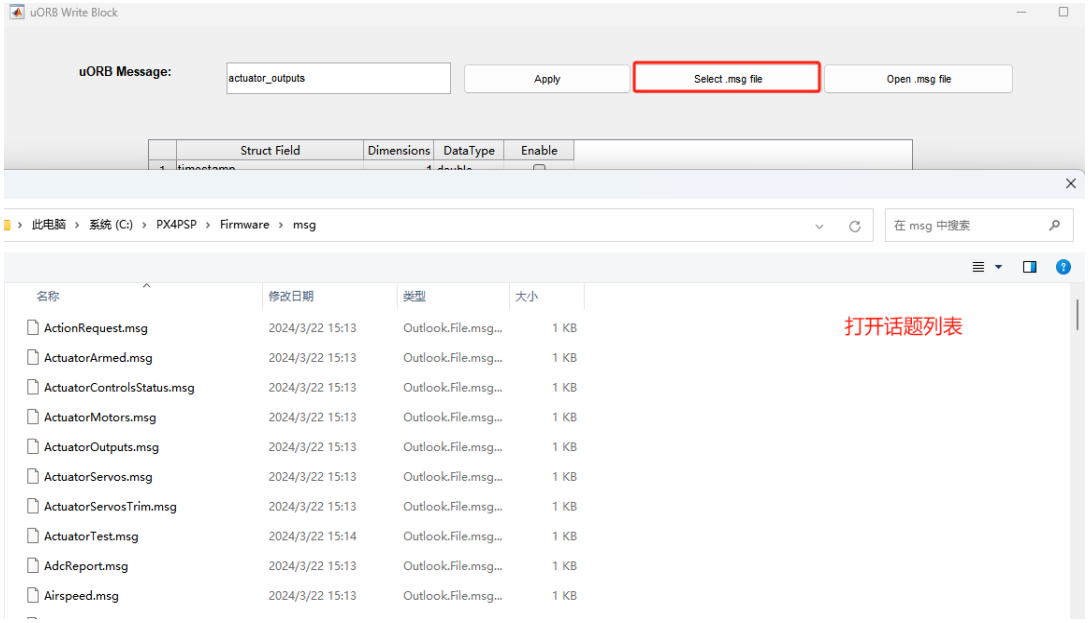


uORB Write Advanced

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。

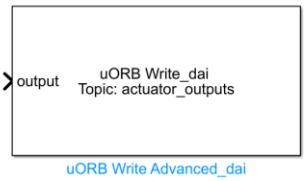


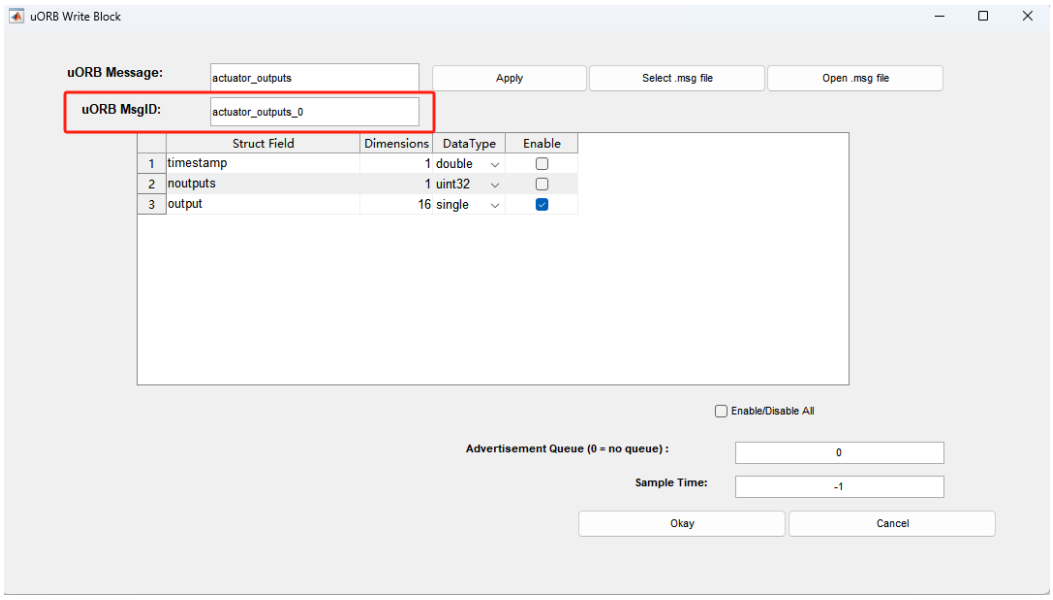




2.5. uORB Write Advanced_dai—uORB 消息数据发布接口进阶模块

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





2.6. 其他模块

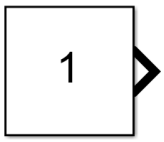
以下模块均为 Simulink 官方模块。

1. Data Type Conversion 模块

Data Type Conversion

该模块将输入转换为输出的数据类型和定标。转换有两个可能的目标。一个目标是使输入和输出的真实值相等。另一个目标是使输入和输出的存储的整数值相等。溢出和量化误差会妨碍目标的完全实现。

2. Constant 模块



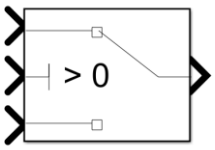
Constant

Constant 模块生成实数或复数常量值信号。使用此模块提供常量信号输入。此模块是生成标量、向量还是混合输出要取决于：

- 常量值参数的维度
- 将向量参数解释为一维向量参数的设置

此模块的输出与常量值参数具有相同的维度和元素。如果为此参数指定向量，即希望模块将其解析为向量，请选中将向量参数解释为一维向量复选框。否则，如果为常量值参数指定了向量，模块会将该向量视为一个矩阵。

3. Switch 模块



Switch

Switch 模块根据第二个输入的值，传递第一个输入或第三个输入。第一个和第三个输入称为数据输入。第二个输入称为控制输入。可以使用首个输入传递条件和阈值参数指定模块传递第一个输入的条件。

要立即将已知的输出数据类型反向传播到第一个和第三个输入端口，将输出数据类型参数设置为继承:通过内部规则继承并选中要求所有数据端口输入具有相同的数据类型复选框。

4. Bus Selector 模块

Bus
Selector

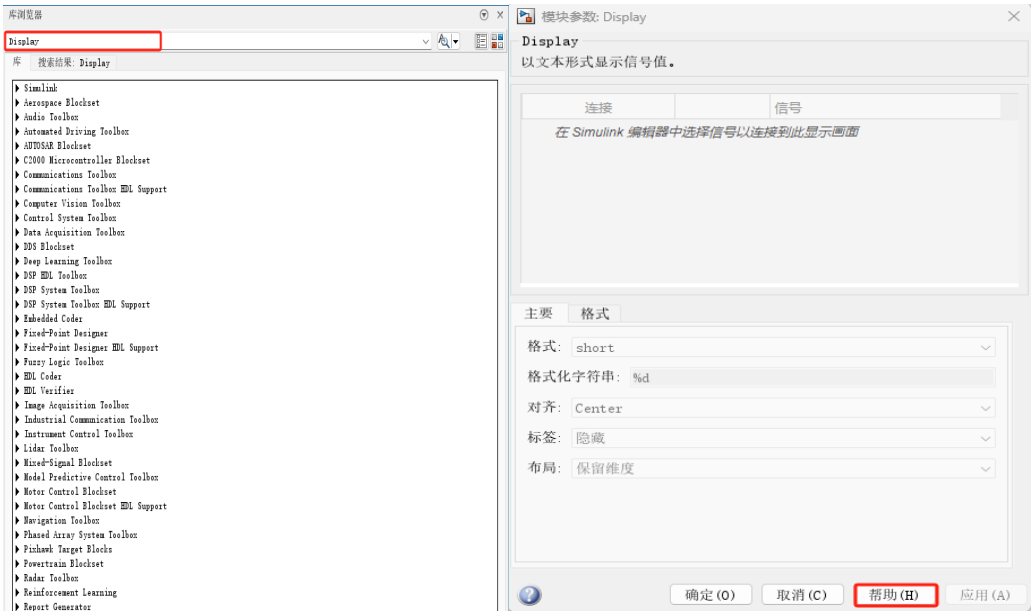
Bus Selector 模块从输入总线层次结构中按名称提取您选择的元素。该模块可以分别输出选定的各个元素或在一个新的虚拟总线中输出所有选定元素。当模块分别输出选定元素

时，每个选定元素对应于一个输出端口。当模块输出新的虚拟总线时，对于包含所有选定元素的虚拟总线，该模块具有一个与之对应的输出端口。

5. Clock 模块

Clock 模块在每个仿真时间步输出当前仿真时间。

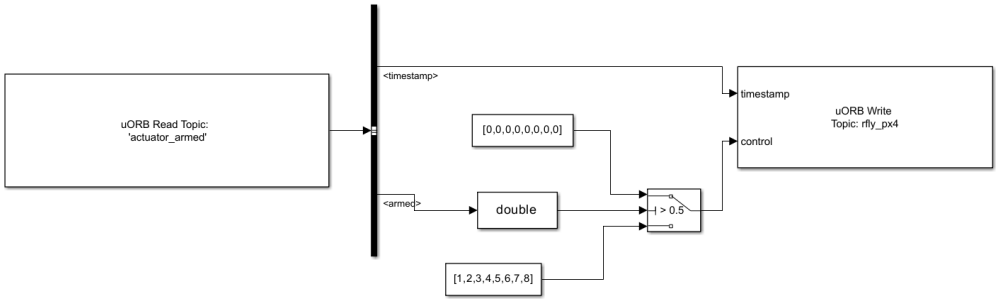
如下图所示，更多 Simulink 官方模块的详细用法和功能，请到 Simulink 界面的库浏览器中搜索，找到对应的模块，双击该模块，点“帮助”即可查看该模块的详细用法和功能。



3. 实验原理

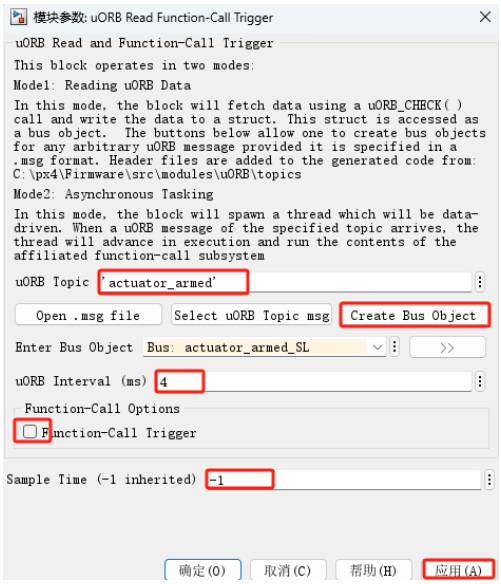
3.1. uORB 消息读取实验原理

例程见：[安装目录]\RflySimAPIs\5.RflySimFlyCtrl\0.ApiExps\6.uORB-Read-Write\Exp1_uORB_read.slx



如上图所示，该模型用 uORB Read Function-Call Trigger 订阅'actuator_armed'消息，来接收是否解锁，如果未解锁，就输出 12345678，反之则输出 00000000，然后，数据通过 rfly_px4 消息发布出去，被其他程序接收。

uORB Read Function-Call Trigger 模块的关键配置如下：



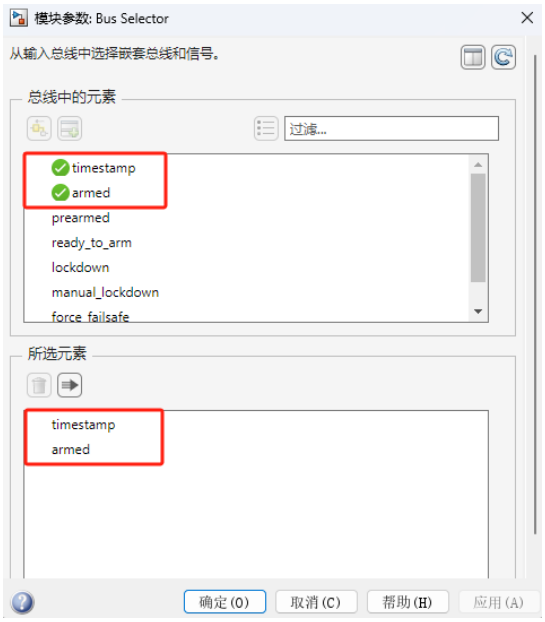
订阅消息要选中 actuator_armed'

1) 点击“Create bus Object”就能创建数据总线

2) uORB interval 时间间隔，这里设置为 4ms，也就是 250Hz，表示每隔多长时间检查消息更新。实际使用时根据情况设置，通常对于低延迟高频率的消息，可以设置成 1ms，也就是 1000Hz；这样做会多消耗一些计算量，但是能降低延迟。

3) Function-Call Options 为触发函数的功能，本例子不需要使用。默认情况也不需要使用。

4) Sample Time 仿真步长，可以设置为-1（使用默认步长，也就是 Simulink 设置页面的值，或者上一级数据源的步长），这里也可以直接设置为 0.01。注意，所有模块的步长与设置页面的基础步长之间，应该成倍数关系。



如上图所示，设置好订阅模块后，就能从 Bus Selector 中，选择自己感兴趣的消息，进行订阅了。

本例中，选择了 timestamp 和 armed 标志位，其中 armed 表示是否解锁。

具体有哪些 uORB 消息，以及每个消息的定义，可以参考网站

https://docs.px4.io/main/en/middleware/uorb.html

或者在 PX4PSP\Firmware\msg 文件夹中自行查看注释。

如下图所示，Switch 模块的含义是，如果未解锁，输出 12345678，反之则输出 00000000。

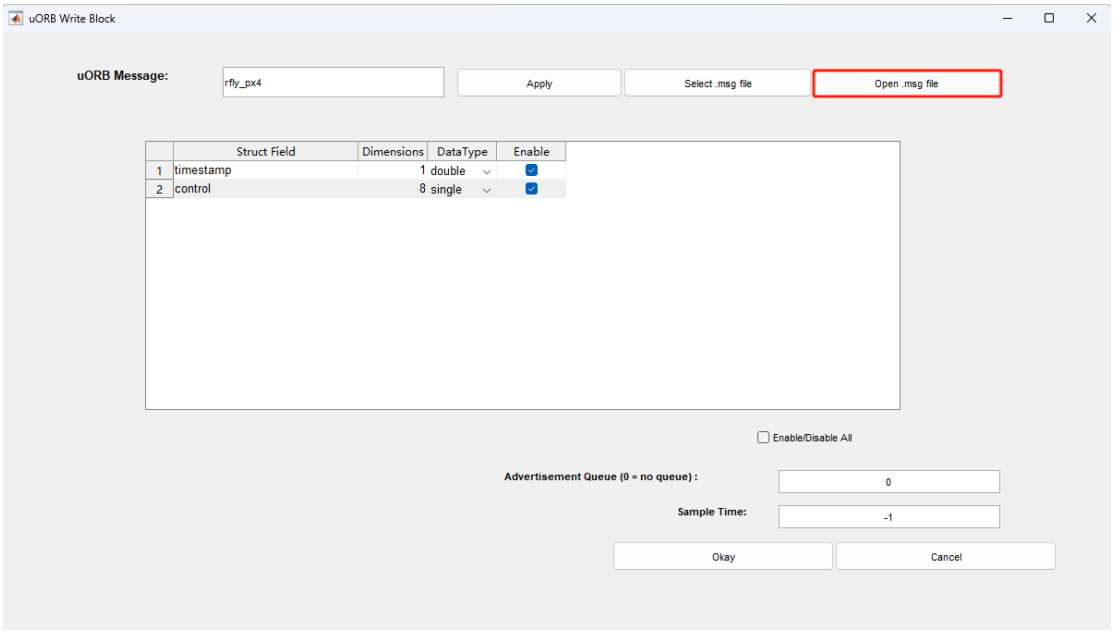
如下图所示，在 uORB Write Advanced1 模块中订阅了 rfy_px4 消息，用来向其他程序发布数据。点击 Open .msg file 可查阅该 uORB 消息，该消息包含两个结构体，具体定义如下：

这是一个无符号 64 位整数，用于表示自系统启动以来经过的时间，单位是微秒（microseconds）。

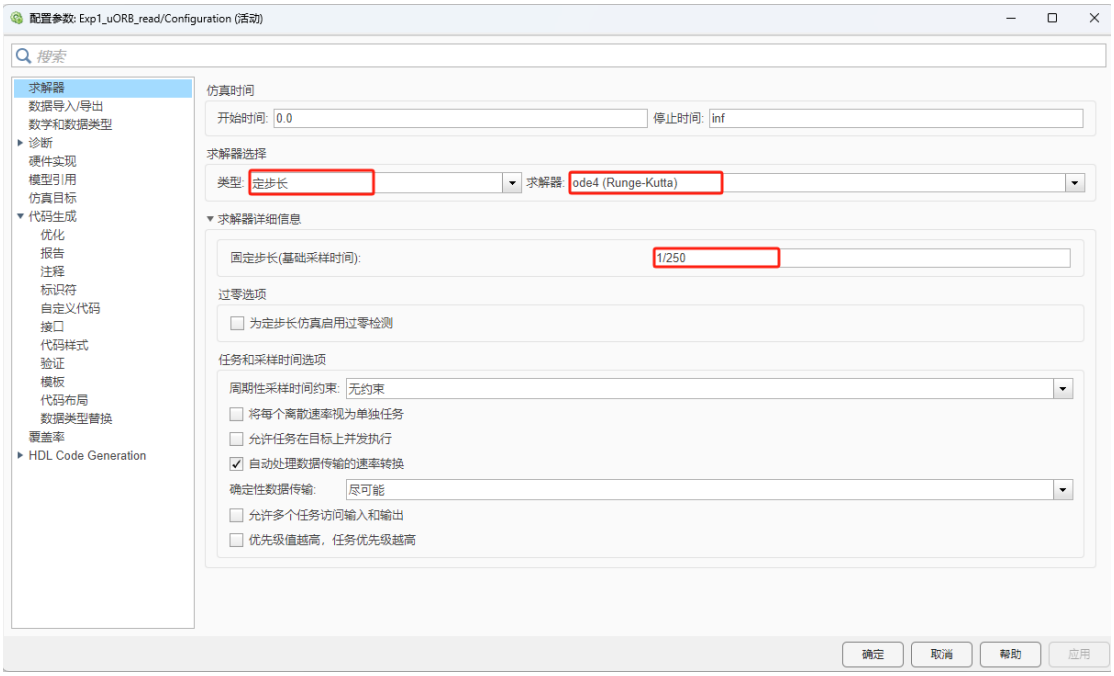
```
uint64 timestamp            # time since system start (microseconds)
```

这是一个包含 8 个元素的浮点数数组，用于表示 8 维控制信号。

```
float32[8] control          # 8D control signals
```



如下图所示，在进行自动代码生成前，需要配置仿真环境。

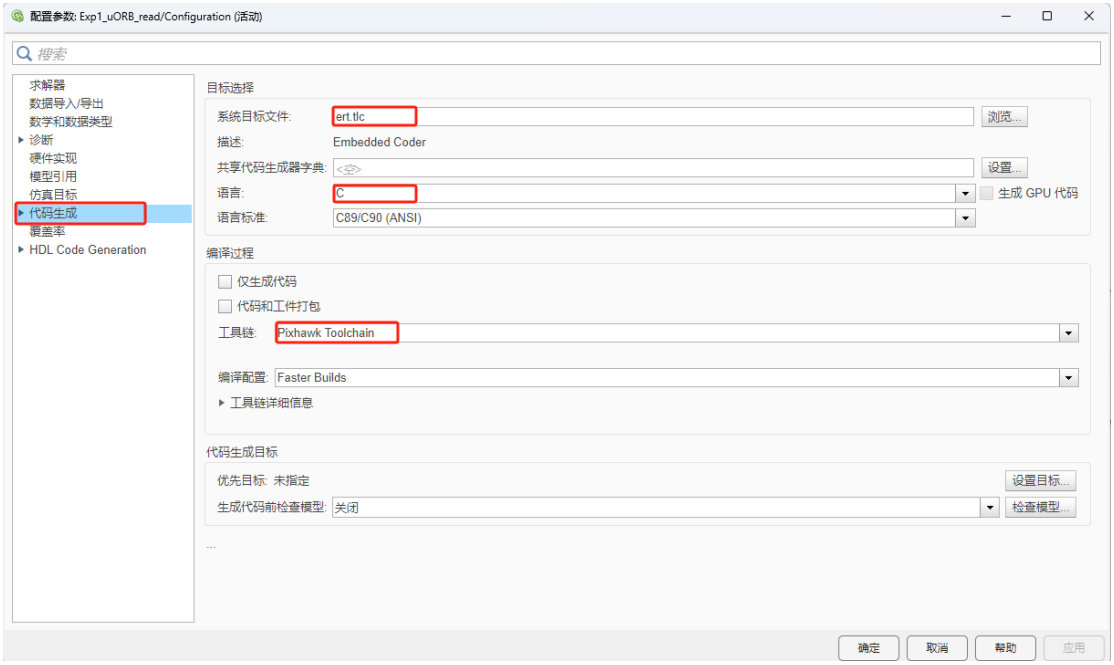


1)必须设置为固定步长。

2)求解器选龙格库塔或其他，阶数越高计算量越大。

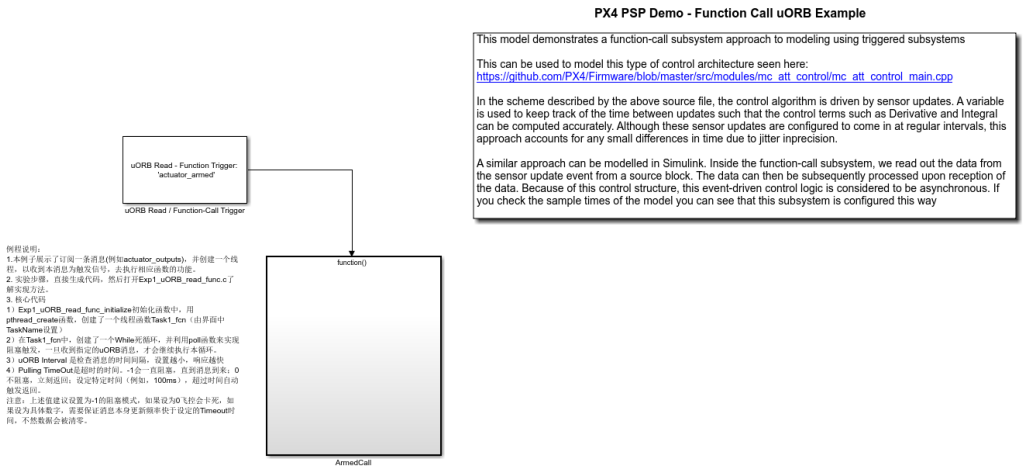
3)仿真步长根据实际情况选择，通常为 1/250 或 1/400，也就是 250Hz 或 400Hz。

4)其他的照页面设置，最重要的确认 Toolchain 为 Pixhawk Toolchain。



3.2. uORB 消息读取触发实验原理

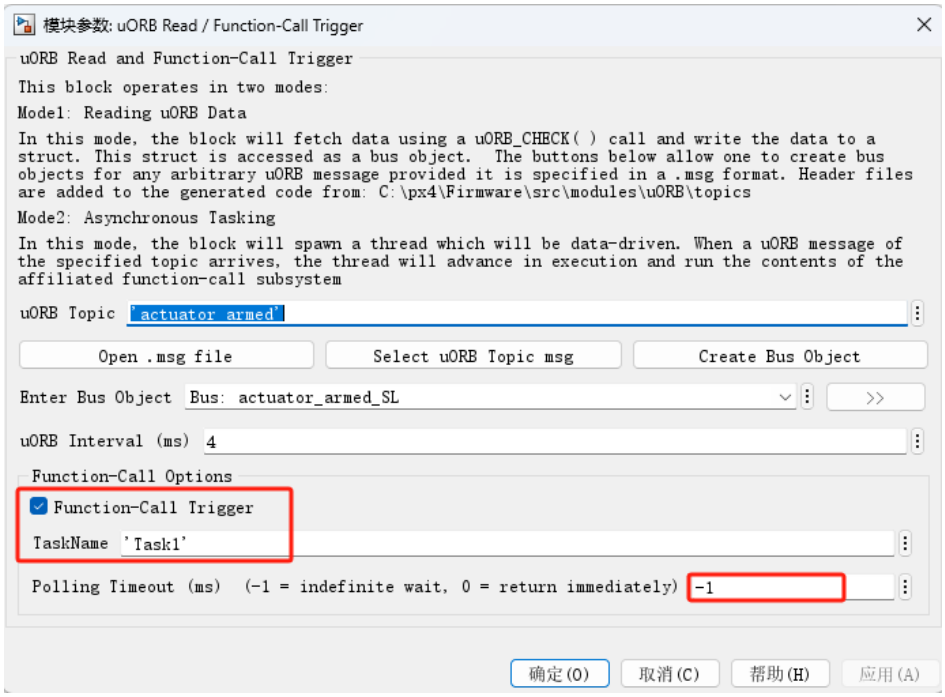
例程见：[安装目录]\RflySimAPIs\5.RflySimFlyCtrl\0.ApiExps\6.uORB-Read-Write\Exp2_uORB_read_func.slx



如上图所示，该模型用 uORB Read Function-Call Trigger 订阅'actuator_armed'消息，来接收是否解锁，如果未解锁，就输出 12345678，反之输出 00000000，数据通过 rfly_px4 消息发布出去，被其他程序接收。和“Exp1_uORB_read.slx”实验不同，本例程的数据处理采用了触发方式，即只有 actuator_armed 更新时，才会去处理数据，并发布 rfly_px4 消息。

例程实现原理：

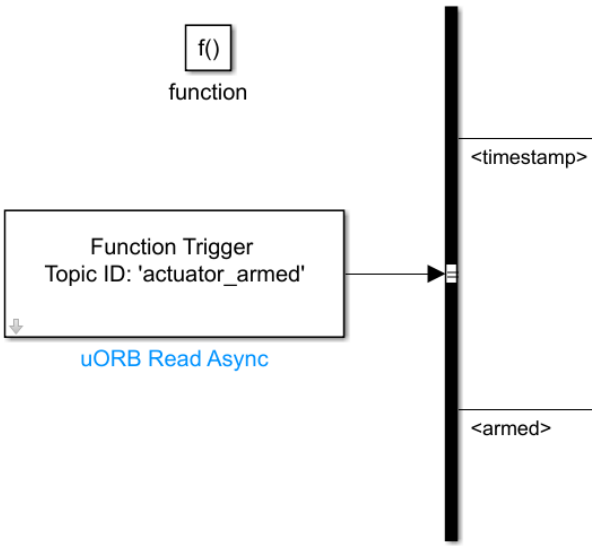
uORB Read Function-Call Trigger 的配置页面，要勾选 Function-Call Triggle

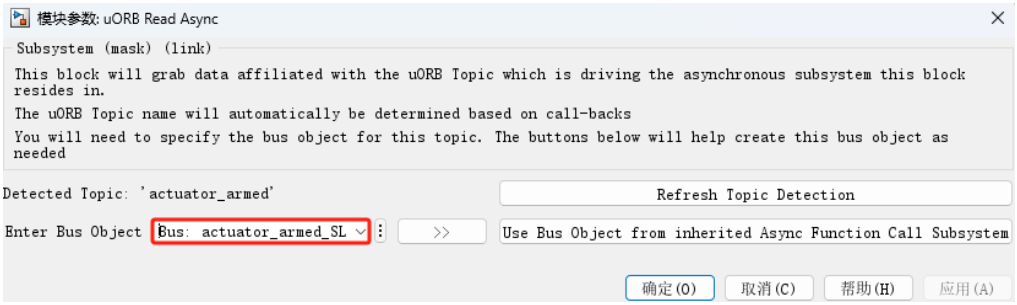


其中 Polling Timeout 可以设置为-1，或指定值。其中，-1 表示会一直阻塞（不会消耗计算性能），直到收到消息，才触发 Task1；设置为 0，则表示不超时立即返回，即不阻塞；设置为对应毫秒，如果超过指定时间没有收到消息，也会强行触发 Task1。注：不能设定为 0，会导致飞控以无限快速度执行 Task1，导致飞控卡死。

配合 function 和 uORB Read Async 模块，来触发并处理 actuator_armed 数据。

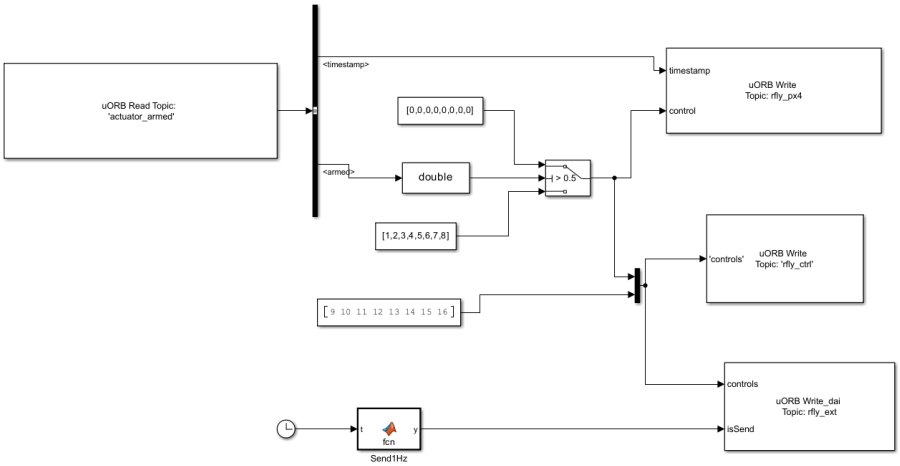
uORB Read Async 里面的总线数据要和外层触发模块保持一致，这里是 actuator_armed





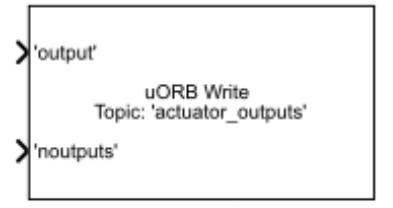
3.3. uORB 发送接口使用实验原理

例程见：[安装目录]\RflySimAPIs\5.RflySimFlyCtrl\0.ApiExps\6.uORB-Read-Write\Exp3_uORB_write.slx

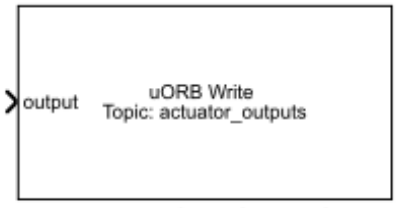


程序内容：监听 actuator_armed 消息，并通过 uORB Write、uORB Write Advanced、uORB Write Advanced_dai 三个发送接口，依次发送 rfly_ctrl、rfly_px4、rfly_ext 三条消息。

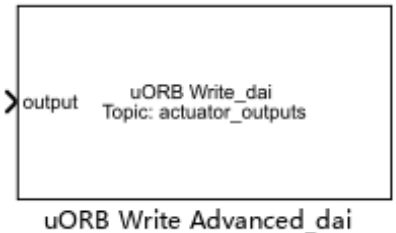
在 PSP 工具库中，三个 uORB 发送函数如下



uORB Write



uORB Write Advanced

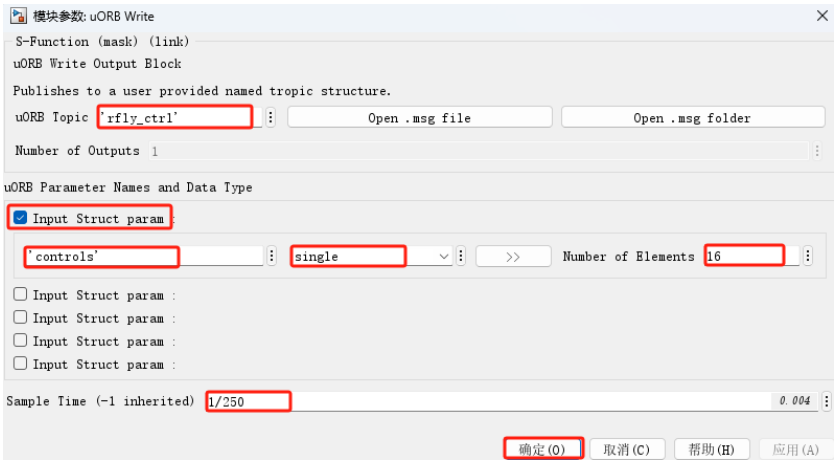


uORB Write Advanced_dai

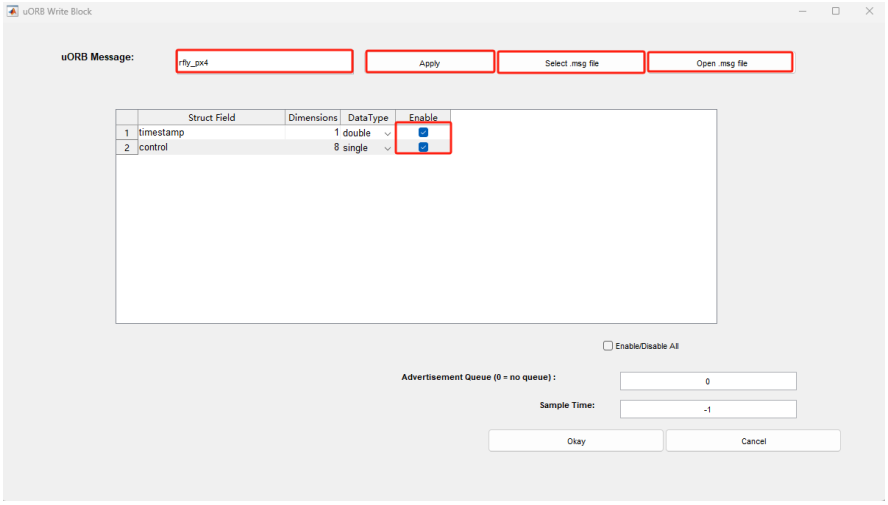
配置方法：

uORB Write 模块需要手动输入消息名，变量名，数据类型，数据维度等，使用较为麻

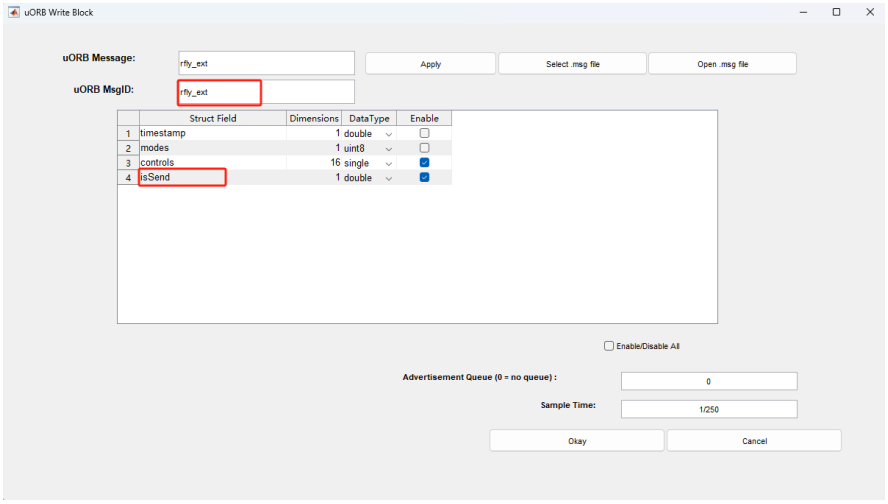
烦



uORB Write Advanced 模块能自动索引 uORB 消息文件，并可以通过勾选使能想要发送的消息接口，使用较为方便。



uORB Write Advanced_dai 模块支持 uORB 消息 ID 和文件名不同，且支持通过 isSend 接口，控制是否发送消息，适合命令类消息（只需发送一次，而不是持续发送），或者控制发送频率等。



Send1Hz 模块，通过时间整除运算，得到 1 秒钟触发一次的信号，发送给 isSend,使得 r

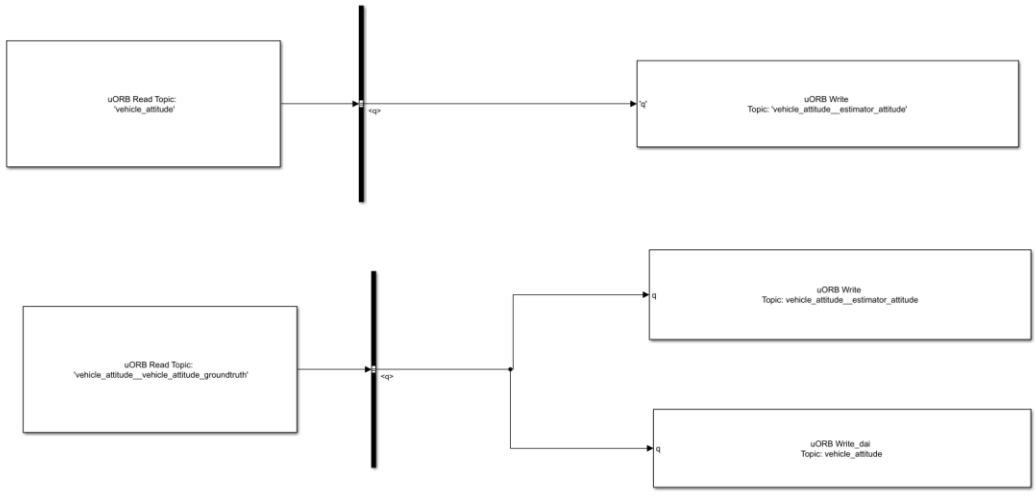
fly_ext 消息的频率被限制在 1Hz

Exp3_uORB_write ▸ Send1Hz

```
function y = fcn(t)

% y=1 for 1Hz
tsec = ceil(t*1000);
r = rem(tsec,1000);

y = 0;
if r<0.5
    y = 1;
end
```

3.4. uORB ID 与文件名不一致处理实验原理

例程见：[安装目录]\RflySimAPIs\5.RflySimFlyCtrl\0.ApiExps\6.uORB-Read-Write\Exp4_uORB_DifID.slx

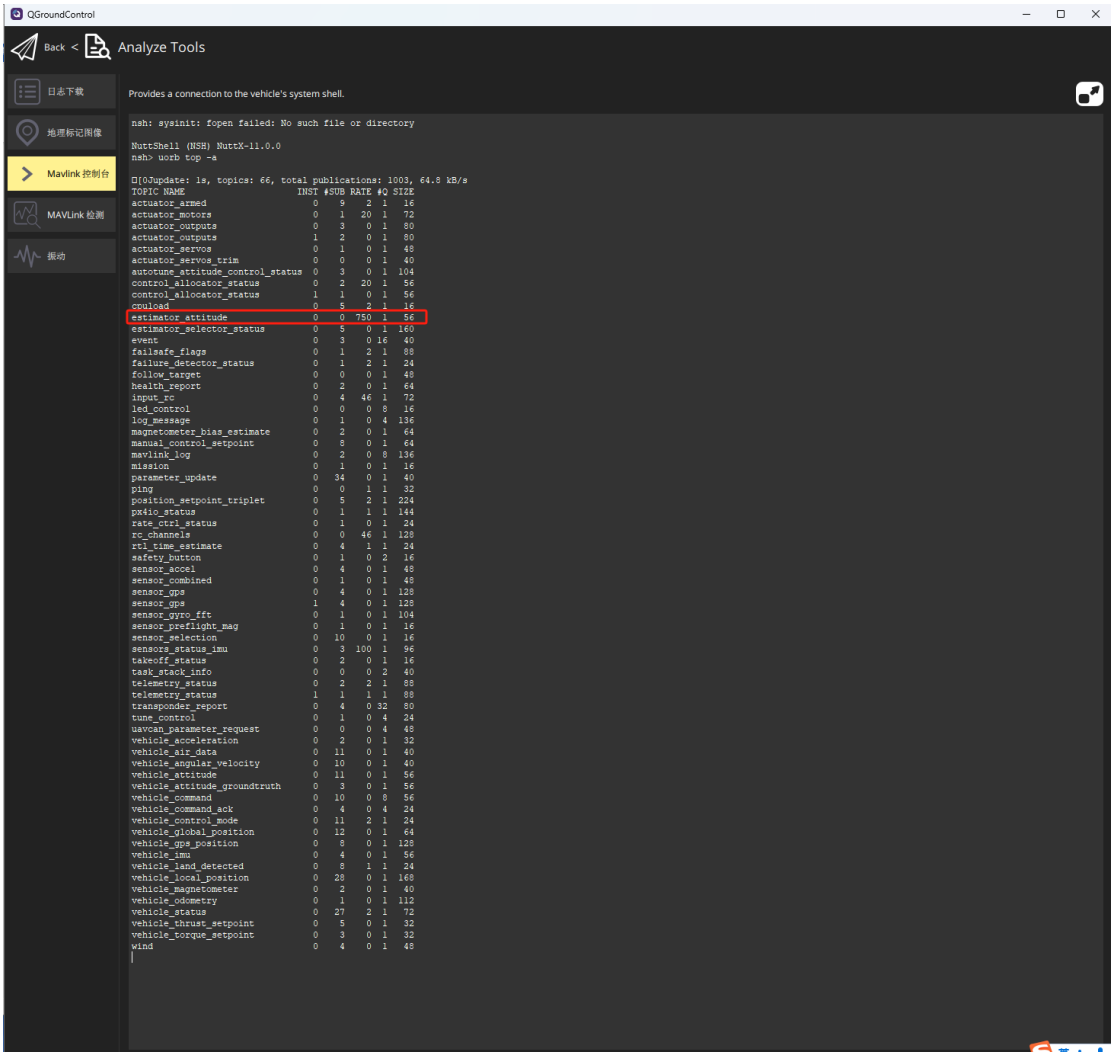


如上图所示，该模型用 uORB Read Function-Call Trigger1 订阅'vehicle_attitude_vehicle_attitude_groundtruth'消息，同时用 uORB Read Function-Call Trigger2 订阅'vehicle_attitude',然后，数据通过三条'vehicle_attitude_estimator_attitude'消息发布出去，被其他程序接收。

在进行自动代码生产和固件烧录后，打开 QGroundControl 软件，点击左上角 Logo 在

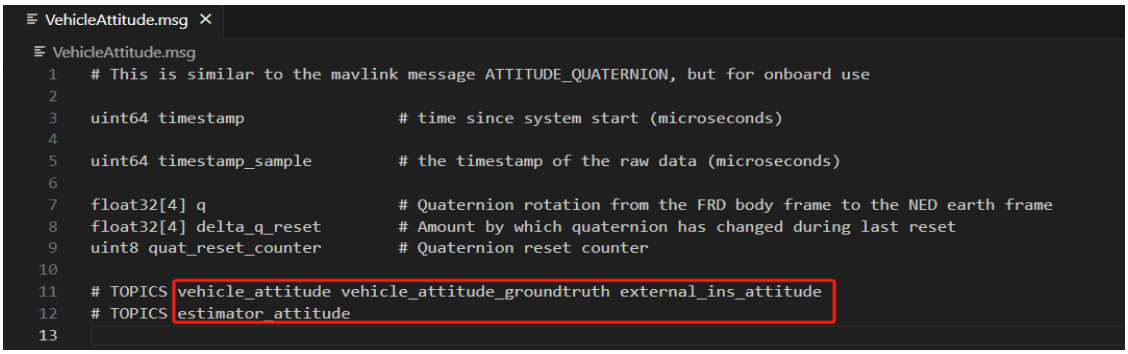
弹出的对话框中，选中 Analyze Tools，在 Mavlink 控制台中输入：

```
uorb top -a
```



通过观察，可以发现飞控接受了三条'vehicle_attitude_estimator_attitude'消息。

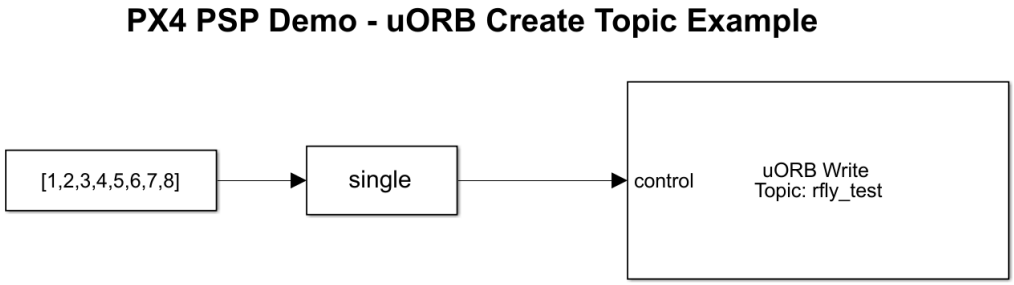
另外，打开 PX4PSP\Firmware\msg\vehicle_attitude.msg 文件（注意：1.14 版本开始是 VehicleAttitude.msg 命名方式）



从文件中可以看出，文件名对应的 uORB ID 默认为 vehicle_attitude，除此之外基于 vehicle_attitude.msg 消息格式，还支持其他的 uORB ID 的消息，包括 vehicle_attitude_groundtruth、external_ins_attitude、estimator_attitude。

3.5. 自定义 uORB 消息实验

例程目录：[安装目录]\RflySimAPIs\5.RflySimFlyCtrl\0.ApiExps\7.uORB-Create\



打开 MATLAB 软件，在 MATLAB 中运行 init_control.m 文件，即可将自定义的 uORB 消息加载到飞控固件中。

init_control.m 脚本解析：

添加路径 '.\msg'，这个路径应该包含了消息定义的文件。

```
addpath('.\msg')
```

用来生成消息结构体,是一个自定义的脚本，用于从消息定义文件中生成相应的代码。

```
PX4uORBMsgGen
```

msg 文件夹中的 PX4uORBMsgGen.m 脚本的作用是自动化地处理消息文件，并将其集成到 PX4 固件中，以便固件能够正确地处理这些消息。

以下是对 PX4uORBMsgGen.m 脚本的部分解析：

检查固件版本，如果这个变量被设置为 `false`，那么意味着固件版本不低于 1.8。

```
isFirmwareLessThan1_8=false;
```

检查是否存在两个.mat 文件：CmakeInfo.mat 和 FirmwareVersion.mat。如果这两个文件都存在，它们将被加载到 MATLAB 的工作区中。

```
if exist([PSPPath,'\CmakeInfo.mat'],'file')
    load([PSPPath,'\CmakeInfo.mat']);
    load([PSPPath,'\FirmwareVersion.mat']);
```

检查是否在加载的 Px4PSP_CmakeInfo 结构体变量和 FirmwareVersion 变量中找到了相应的字段。如果找到了，它们将会被提取出来，并且根据 FirmwareVersion 的值来判断固件版本是否小于 2.5。如果固件版本小于等于 1.8，则会将 isFirmwareLessThan1_8 设置为 true。

```
    if exist('Px4PSP_CmakeInfo','var')
        Px4_Base_Dir=Px4PSP_CmakeInfo.Px4_Base_Dir;
        disp('Found PX4PSP Path.')
    end
    if exist('FirmwareVersion','var')
```

```
        if str2num(FirmwareVersion)<2.5 % the Firmware verison <= 1.8
            isFirmwareLessThan1_8=true;
        end
```

输出一些提示信息，指示是否找到了相应的路径和固件版本信息。

```
        disp('Found PX4 Firmware version.')
    end
end
```

然后是一个循环，遍历指定路径下的所有消息文件。在循环中，首先读取消息文件内容，并根据固件版本进行一些处理。例如，如果固件版本较低，则可能会删除或修改一些与新固件版本不兼容的内容。

```
for ii=1:length(AFileList)
    nuttx_str = [AFileList(ii).folder,'\',AFileList(ii).name];
    fid = fopen(nuttx_str);
    strTmp=[];
    while ~feof(fid)
         str = string(fgetl(fid));
         if isFirmwareLessThan1_8 && contains(str,'uint64 timestamp')
             continue;
         end
         strTmp=[strTmp;str];
    end
    fclose(fid);

    fiName = AFileList(ii).name(1:end-4);
    if str2num(FirmwareVersion)>7.5
        if contains(fiName,'_')
            msg_list = strsplit(fiName,'_');
            UpStr='';
            for i=1:length(msg_list)
                crStr=msg_list{1,i};
                crStr=[upper(crStr(1)),crStr(2:end)];
                UpStr=[UpStr,crStr];
            end
            fiName=UpStr;
        end
    else
        if ~contains(fiName,'_')
            MsgFileName = regexprep(fiName,'(.)([A-Z][a-z]+)','$1_$2');
            MsgFileName = regexprep(MsgFileName,'([a-z0-9])([A-Z])','$1_$2');
            fiName=lower(MsgFileName);
        end
    end
```

将处理后的消息内容写入到 PX4 固件的消息目录中，并更新 CMakeLists.txt 文件以确保消息被正确包含在固件编译中。

```
nuttx_str = [Px4_Base_Dir,'\Firmware\msg\CMakeLists.txt'];
    fid = fopen(nuttx_str);
    strTmp=[];
    isMsgExist=false;
    idx=50;
    i=1;
    while ~feof(fid)
         str = string(fgetl(fid));
```

```
            if contains(str,fiName)
                isMsgExist=true;
                break;
            end
            if str2num(FirmwareVersion)>7.5
               if contains(str,'RcParameterMap.msg')
                   idx=i;
               end
            else
               if contains(str,'actuator_controls.msg')
                   idx=i;
               end
            end

            strTmp=[strTmp;str];
            i=i+1;
        end
        fclose(fid);

        if ~isMsgExist
            strTmp=[strTmp(1:idx);"   "+string(fiName);strTmp(idx+1:end)];
            fid = fopen(nuttx_str,'w+');
            for i=1:length(strTmp)
                fprintf(fid, '%s\n', strTmp(i,:));
            end
            fclose(fid);
        end
        addLoggerList=[addLoggerList;string(msgName)];
    end
```

根据固件版本修改日志文件（如 logger.cpp）以确保新的消息能够被正确记录。

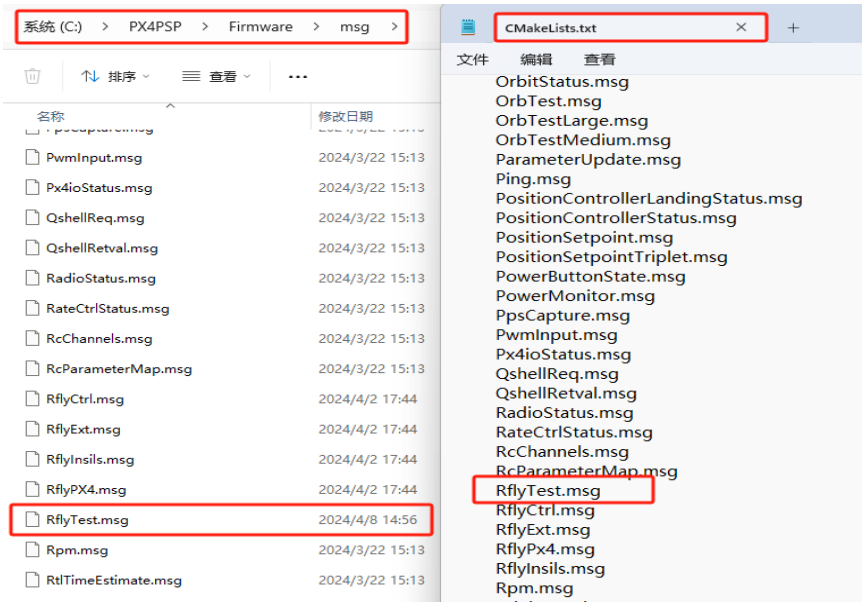
```
    LogName='logger.cpp';
    if str2double(FirmwareVersion)>4.5
        LogName='logged_topics.cpp';
    end
    nuttx_str = [Px4_Base_Dir,'\Firmware\src\modules\logger\',LogName];

    for i=1:length(addLoggerList)
        logMsgName=['add_topic("',char(addLoggerList(i)),'", 20);'];

        fid = fopen(nuttx_str);
        strTmp=[];
        j=0;
        add_default_topics_idx=0;
        msg_idx=0;
        while ~feof(fid)
            j=j+1;
            str = string(fgetl(fid));
            if add_default_topics_idx<1 && contains(str,'::add_default_topics()')
                add_default_topics_idx=j;
            end

            if msg_idx<1 && contains(str,logMsgName)
                msg_idx=j;
            end
```

```
            strTmp=[strTmp;str];
        end
        fclose(fid);


        if msg_idx<1 && add_default_topics_idx>1

            strTmp = [strTmp(1:add_default_topics_idx+1);"    "+logMsgName;strTmp(add_default_
topics_idx+2:end)];
            fid = fopen(nuttx_str,'w+');
            for k=1:length(strTmp)
                fprintf(fid, '%s\n', strTmp(k,:));
            end
            fclose(fid);
        end
    end
```

输出 PX4 Firmware uORB modified.表示修改完成。

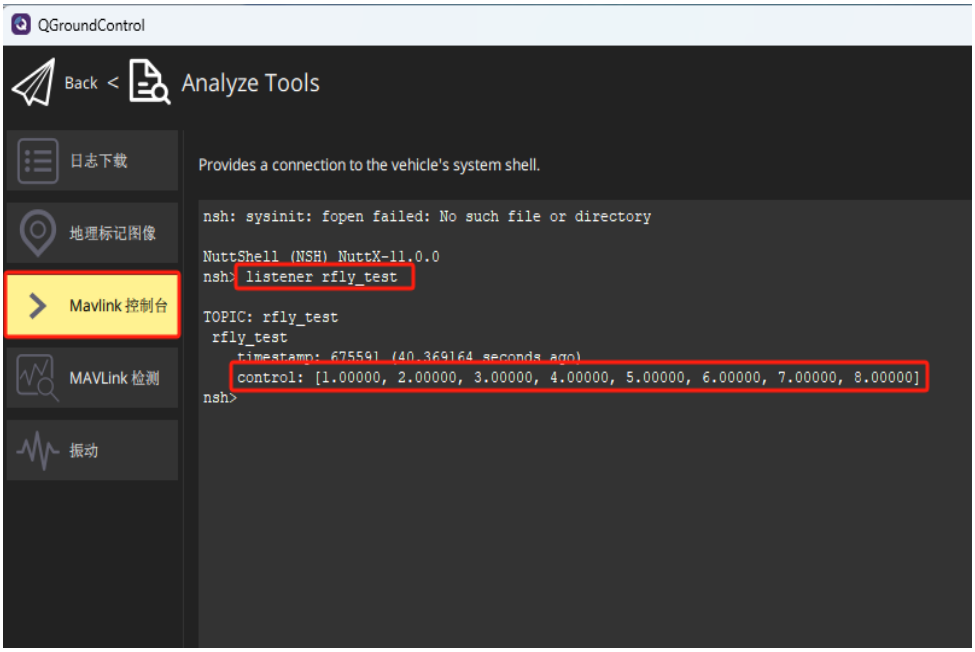
```
    disp('PX4 Firmware uORB modified.')
```

可在“\PX4PSP\Firmware\msg\rfly_test.msg”中查看到 rfly_test.msg 文件（注意：1.14 版本开始是 RflyTest.msg 命名方式）。同时在本文件夹的 CMakeList.txt 中也可看到新增的 rfly_test.msg 消息。如下图所示。



在进行自动代码生产和固件烧录后，打开 QGroundControl 软件，点击左上角 Logo 在弹出的对话框中，选中 Analyze Tools，在 Mavlink 控制台中输入：

```
    listener rfly_test
```



可以发现，和本例程我们自定义的 uORB 消息的输入是一致的。

4. 相关资料

[1]. 全权,杜光勋,赵峙尧,戴训华,任锦瑞,邓恒译.四旋翼飞行器设计与控制[M],电子工业出版社,2018.

[2]. PX4 官网：https://docs.px4.io/main/zh/index.html

附加资源

官方文档：RflySim 官方文档：https://rflysim.com/doc/zh/

社区交流：加入 RflySim 技术交流群：951534390