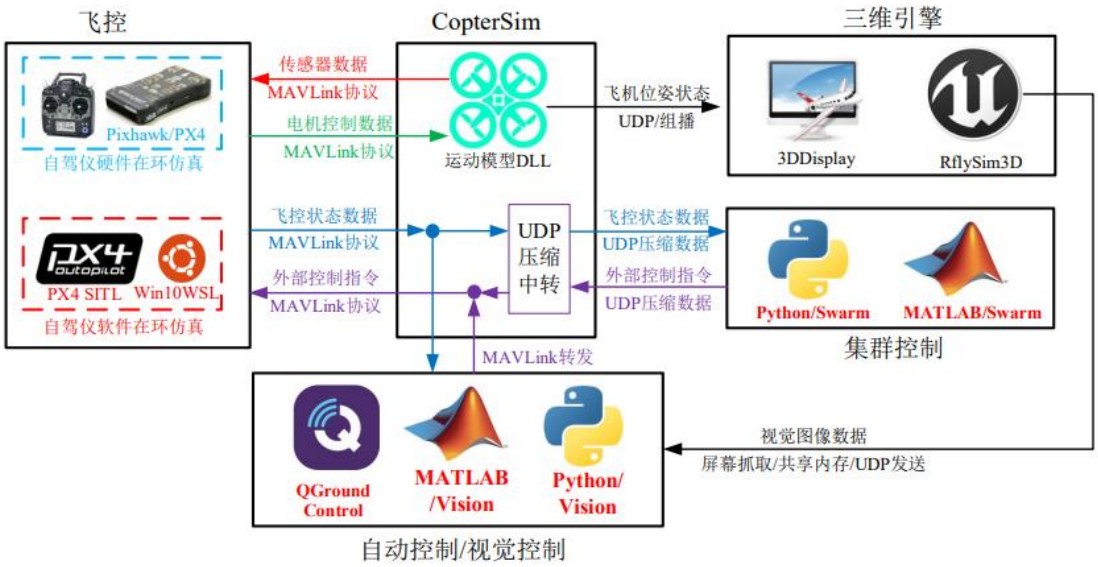
# API 说明文件检索大纲

# 1. RflySim 平台集群架构简介与使用

## 1.1 集群通信架构介绍




1.RflySim 通信构架如上图所示，其中集群控制模块主要使用了 CopterSim 进行压缩中转机制。

2.真机使用的 MAVLink 通信的数据量过于庞大，不适合在大规模集群时在通信网络中直接传输，因此需要对 MAVLink 数据进行压缩中转。

3.Python 或 Simulink 集群控制器接收飞控状态数据并发送 Offboard 指令（速度、位置、加速度等顶层控制）来控制飞机完成指定飞行任务。

4.多个 CopterSim+飞控组合会同时向局域网内的三维引擎、集群控制、视觉控制模块发送数据，从而实现集群通信仿真。

## 1.2 集群控制通信优化

随着飞机数量的增加，网络通信负载越来越大，为了在有限带宽下实现更多数量的无人机集群仿真，需要对通信进行优化。

目前平台的数据协议主要有两种： 1 ）MAVLink 数据（包含大量无关数据，且数据包非常大，不适合集群）和 UDP 压缩结构体（仅包含需要信息，且数据包小，适合集群）。

目前平台的通信优化方式也有两种： 1 ）Full 模式，包含尽量多的数据、以尽量高频率发送来保证完 整性（适合飞机 ≤ 8）；Simple 模式，仅包含必需 数据、降低发送频率来保证大规模集群下的通信 实时性和流畅性（适合飞机数量＞ 4）。

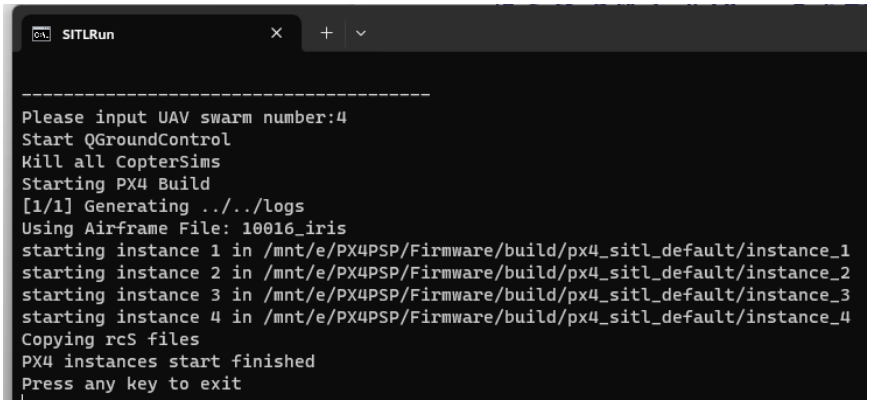
两者组合起来有：UDP_Full 、UDP_Simple 、 MAVLink_Full 、MAVLink_Simple 四种模式；此外，MAVLink 通信针对多电脑视觉硬件在环仿真优化，还增加一种 MAVLink_No

Send 协议。

## 1.3 集群软件在环仿真

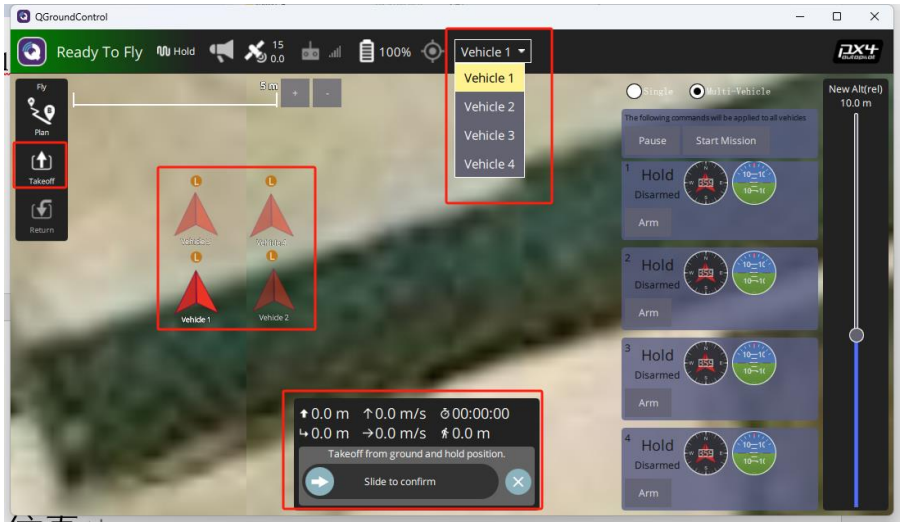
双击运行“RflySimAPIs\SITLRun.bat”（或 Rflytools 文件夹内同名快捷方式），在弹出命令提示符中输入“4”，点击回车键，就能快速开启 4 个飞机的集群仿真系统。

自动打开的软件包括 4 个 CopterSim（每个飞机对应一个）、1 个 RflySim3D（会同 时显示所有飞机）、1 个 QGC 地面站（同时显示并控制所有飞机）和 1 个命令提示 符窗口（调用 Win10WSL 编译器开启 4 个 PX4 SITL 控制器，每个控制器的 log 和参数可以访问 PX4PSP\Firmware\build\px4_sitl_default\instance_*）

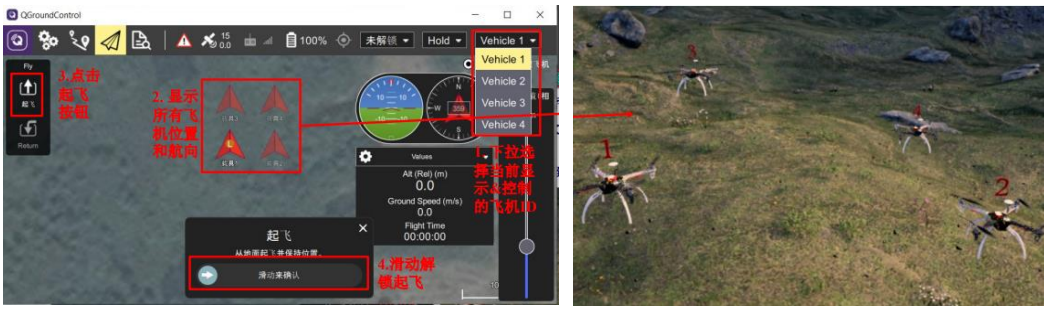




如下图所示，在 QGroundControl 可以切换不同的飞机。依次切换各个飞机（Vehicle 1~4），并依次点击“起飞”按钮，然后点 “滑动来确认”，即可控制 4 个飞机依次起飞。



起飞后再在 RflySim3D 中按下“S”键可显示各飞机 ID，按“T”键显示轨迹，按“D”键显示飞机信息，按“B”键切换飞机，按“V”键切换视角，效果如下。




## 1.4 集群硬件在环仿真

如下图所示，将所有 Pixhawk 飞控插入电脑（请提前配置还原好 Pixhawk 固件， 并设置进入 HITL 模式）

双击运行“RflySimAPIs\HITLRun.bat”(或桌面同名快捷方式)，在右图所示 弹出窗口中根据提示输入飞控串口号（英文逗号隔开），点击回车键，就能快速 开启多个飞机的集群仿真系统。

下面两个图对应了连接 5 个飞控进行硬件在环仿真的情形，运行结果和 SITL 相同。


## 1.5 集群 bat 一键脚本修改

集群的 bat 一脚脚本存放目录为：PX4PSP\RflySimAPIs

Bat 文件内的参数说明：

START_INDEX：起始飞机序号，如果这里设 为 1，同时 SITL 输入飞机数量 4（或 HITL 输入 4 个串口号），那么创建飞机序号为 1 到 4；如果 START_INDEX 设为 5，那么就是 5 到 8 号飞机。

TOTOAL_COPTER：非必需参数，多电脑分布式组网仿真才时需要用到，描述局域网内所有飞机数量(限高级完整版)；用于根据给定 初始位置和间隔，以矩形方式自动排列飞机。

UE4_MAP ：RflySim3D 地图名称，请去目录 PX4PSP \CopterSim \external \map 选择地图名字。

ORIGIN_***：飞机在地图上的初始位置和偏航

VEHICLE_INTERVAL：多机时排列间隔（米）

IS_BROADCAST：是否启用多台电脑联机仿真， 或指定联机电脑 IP 地址列表（限高级完整版）

UDPSIMMODE：通信数据收发模式

修改示例：

需求 1：SITLRun 脚本自动设置飞机数量（例如 4），不用手动输入。删除从“:Top”到“:StartSim”代码，并添 加行“SET VehicleNum=4”即可。请参考 RflySimAPIs\SimulinkSwarmAPI\RflyUdpFullFour.bat

需求 2：两台电脑仿真 20 个飞机（限完整版）。需要两个 bat 脚本，IS_BROADCAST 设为 1，启用联机功能；第一个 bat 脚本 START_INDEX=1 和 TOTOAL_COPTER=20，运行 SITL 脚本时输入 10，第二个 bat 脚本 START_INDEX=11 和 TOTOAL_COPTER=20，输入 10。再选择合适的地图、起始位置和飞机间隔即可。

需求 3：改变地图名称。直接修改 UE4_MAP 为需要地图即可。

需求 4：手动指定每个飞机位置。请参考 RflySimAPIs 目录下的 HITLRunPos.bat 和 SITLRunPos.bat 快捷方式，支 持手动指定每个飞机的 x 坐标序列、y 坐标序列和偏航角序列，飞机数量根据输入序列自动确定。

## 1.6 RflySim3D 集群快捷方式使用

F1：弹出帮助菜单提示；

ESC：清除所有飞机

S：显示/隐藏飞机 ID；

B：在不同飞机间切换视角焦点；

B+数字*：切换到第*号飞机

T：开启或关闭飞机轨迹记录功能

T+数字*：开启/更改轨迹粗细为*号

P：开启物理碰撞引擎（不同飞机会碰 撞坠机，仅限高级完整版）

CTRL+鼠标滚轮：缩放所有飞机尺寸 (多机时便于观察)；

CTRL + C：切换全部飞机三维样式

# 2. 集群控制模型

## 2.1 高精度模型+PX4 控制器 组成的软/硬件在环仿真模型

### 2.1.1 PX4MavCtrler: __init__()

参数：self, ID=1, ip='127.0.0.1',Com='udp',port=0

ID：仿真 ID

Ip：数据向外发送的 IP 地址

Com：与 Pixhawk 的连接模式

Port：端口号

函数用以初始化飞机，飞机的通信模式。

### 2.1.2 InitMavLoop()

参数：UDPMode=2

UDPMode：0 和 1 对应 UDP_Full 和 UDP_Simple Mode，2 和 3 对应 MAVLink_Full and MAVLink_Simple mode，4 对应 MAVLink_NoSend

isCom：串口通信的标志

isRealFly：网络直连模式的标志

isRedis：Redis 模式的标志

函数用以建立通信。默认模式是 MAVLink_Full

### 2.1.3 initOffboard()

这个函数不需要输入参数启动。

函数发送 Offboard 指令，使飞机进入 Offboard 模式，并且开始以 30Hz 的频率发送 Offboard 信息。

函数通过调用 sendMavOffboardAPI()函数进行信息发送。

### 2.1.4 sendMavOffboardAPI()

参数：type_mask=0,coordinate_frame=0,pos=[0,0,0],vel=[0,0,0],acc=[0,0,0],yaw=0,yawrate

=0

type_mask：控制模式，例如本地位置控制，全球位置控制等

coordinate_frame：坐标系类型

pos：位置信息

vel：速度信息

acc：加速度信息

yaw：偏航角信息

yawrate：偏航角速率信息

函数根据 offMode 变量判定 offboard 模式，调用相应的函数发送信息。

### 2.1.5 SendPosNED()

参数：x=0,y=0,z=0,yaw=0

x：目标位置信息

y：目标位置信息

z：目标位置信息

yaw：偏航角

函数将把 offMode 变量赋值为 0，代表进入本地北东地坐标系位置控制。

### 2.1.6 endOffboard()

调用不需要参数，直接调用。

函数将会发送 PX4 退出 offboard 模式的指令，停止信息发送循环。

### 2.1.7 stopRun()

调用不需要参数，直接调用即可。

函数将会停止 mavlink 的信息监听循环。

### 2.1.8 initPointMassModel()

参数：intAlt=0,intState=[0,0,0]

intAlt：飞机初始高度

intState：飞机初始位置，初始偏航角

函数用以开启质点模型仿真。

### 2.1.9 PointMassModelLoop()

函数是质点模型的更新循环。将会由 initPointMassModel()函数调用。不需用户自启。

### 2.1.10 sendUE4PosNew()

参数：copterID=1,vehicleType=3,PosE=[0,0,0],AngEuler=[0,0,0],VelE=[0,0,0],PWMs=[0]*8,runnedTime=-1,windowID=-1

copterID：模拟飞机 ID

vehicleType：飞机样式

PosE：北东地坐标系位置

AngEuler：飞机姿态角

VelE：飞机速度

PWMs：转速

runnedTime：当前时间

windowID：窗口号

函数的作用是将传入的数据打包发送给 RflySim3D，用以创建一个新的 3D 模型或者更新模型的状态信息。

### 2.1.11 InitTrueDataLoop()

函数开启后将会初始化 UDP 真实数据监听循环。通过 30100 系列端口从 CopterSim 监听数据。

### 2.1.12 EndTrueDataLoop()

函数会关闭 UDP 真实数据监听循环。

### 2.1.13 endMavLoop()

函数和 stopRun()函数具有相同功能。将会关闭 20100 系列端口的数据监听。

### 2.1.14 SendMavCmdLong()

参数：command, param1=0, param2=0, param3=0, param4=0, param5=0, param6=0, pa

ram7=0

关于 command 和 param1~7 的定义，请参考：

https://mavlink.io/en/messages/common.html#COMMAND_LONG

https://mavlink.io/en/messages/common.html#MAV_CMD

函数的作用是，向 PX4 发送消息

### 2.1.15 sendMavOffboardCmd()

参数：type_mask,coordinate_frame, x, y, z, vx, vy, vz, afx, afy, afz, yaw, yaw_rate

type_mask：控制模式

coordinate_frame：坐标系信息

x：位置信息

y：位置信息

z：位置信息

vx：速度信息

vy：速度信息

vz：速度信息

afx：加速度信息

afy：加速度信息

afz：加速度信息

yaw：偏航角

yaw_rate：偏航角速率

函数用以向 PX4 发送 offboard 指令

### 2.1.16 sendUDPSimpData()

参数：ctrlMode,ctrls

ctrlMode：飞行模式

ctrls：控制参数。例如：如果飞行模式为起飞，则控制参数应该为[x, y, z, yaw]

函数将简易 UDP 消息发送到通信端口。

### 2.1.17 SendVelNED()

参数：vx=0,vy=0,vz=0,yawrate=0

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

yawrate：偏航角速率指令

函数用以将飞行模式切换为地球速度控制模式，切换 offboard 模式，发送消息的判别位，设定坐标系。

### 2.1.18 SendVelNEDNoYaw()

参数：vx,vy,vz

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

函数与 SendVelNED()具有相似功能。不同的是，函数不会设置偏航角速率指令，只发送速度控制指令。

### 2.1.19 SendVelFRD()

参数：vx=0,vy=0,vz=0,yawrate=0

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

yawrate：偏航角速率指令

函数将切换飞行模式为机体速度控制模式，offboard 模式为速度控制模式。切换工作坐标系为集体坐标系。坐标系正方向为前，右，下。

### 2.1.20 SendAttPX4()

参数：att=[0,0,0,0],thrust=0.5,CtrlFlag=0,AltFlg=0

att：根据 CtrlFlag 确定

CtrlFlag 0 ：att 是三维向量，包含横滚角，俯仰角，偏航角，单位是角度

CtrlFlag 1 ：att 是三维向量，包含横滚角，俯仰角，偏航角，单位是弧度

CtrlFlag 2 ：att 是四维向量，包含四元数

CtrlFlag 3 ：att 是三维向量，包含横滚角速率，俯仰角速率，偏航角速率，单位是 rad/s

CtrlFlag 4 ：att 是三维向量，包含横滚角速率，俯仰角速率，偏航角速率，单位是 degree/s

thrust：根据 AltFlg 确定

AltFlg 0 ：总推力，归在 0-1 范围内，（-1~1 适用于有反向推力的飞机）

AltFlg>0：期望高度

CtrlFlag：用于确定输入 att 定义的标志

AltFlg：用于确定输入 thrust 定义的标志

函数将切换飞机的飞行模式为姿态控制模式，切换 offboard 模式。将 FRD（前，右，下）坐标系下飞机目标姿态信息发送到 PX4

### 2.1.21 SendAccPX4()

参数：afx=0,afy=0,afz=0,yawValue=0,yawType=0,frameType=0

afx：加速度信息

afy：加速度信息

afz：加速度信息

yawValue：偏航信息

yawType：1，偏航角控制。2，偏航角速率控制

frameType：0，地北东地坐标系。1，机体 FRD 坐标系

函数将切换飞机的飞行模式为加速度控制模式，切换 offboard 模式。根据 frametype 切换坐标系。发送加速度控制信息到 PX4

### 2.1.22 SendVelNoYaw()

参数：vx,vy,vz

vx：速度控制指令

vy：速度控制指令

vz：速度控制指令

函数与 SendVelNEDNoYaw()功能相同，不同的是函数将当前坐标系设定为，机体 FRD 坐标系。

### 2.1.23 SendVelYawAlt()

参数：vel=10,yaw=6.28,alt=-100

vel：速度指令

yaw：偏航角指令

alt：高度指令

函数将设置飞行模式为速度高度偏航模式。切换 offboard 模式。切换坐标系为北东地坐标系。发送飞机的偏航信息，速度信息和高度信息给 PX4

### 2.1.24 SendPosGlobal()

参数：lat=0,lon=0,alt=0,yawValue=0,yawType=0

lat：纬度信息

lon：经度信息

alt：高度

yawvalue：偏航

yawType：0，无偏航。1，yawvalue 为偏航角。2，yawvalue 为偏航角速率

函数将切换飞行模式为地球位置控制模式，切换 offboard 模式，坐标系为北东地地球坐标系。发送飞机的位置控制信息和偏航信息给 PX4

### 2.1.25 SendPosNEDNoYaw()

参数：x=0,y=0,z=0,

x：目标位置信息

y：目标位置信息

z：目标位置信息

函数与 2.1.5 SendPosNED()功能类似。使用位置控制方式飞行。不同的是函数不会发送飞机的偏航信息。

### 2.1.26 SendPosFRD()

参数：x=0,y=0,z=0,yaw=0

x：目标位置信息

y：目标位置信息

z：目标位置信息

yaw：偏航信息

函数设定飞行模式为机体位置控制模式。设定 offboard 模式为本地北东地坐标系位置控制模式。设定工作坐标系为机体 FRD 坐标系，函数将位置控制指令和偏航角控制指令发送给 PX4.

### 2.1.27 SendPosFRDNoYaw()

参数：x=0,y=0,z=0

x：目标位置信息

y：目标位置信息

z：目标位置信息

函数的功能与 2.1.26 SendPosFRD()功能类似，不同的是函数不会发送飞机偏航角控制指令。

### 2.1.28 SendPosNEDExt()

参数：x=0,y=0,z=0,mode=3,isNED=True

x：目标位置信息

y：目标位置信息

z：目标位置信息

mode：0，滑翔模式。1，起飞模式。2，降落模式。3，旋翼机悬停模式，固定翼盘旋模式。4，固定翼飞机，无油门，无横滚/俯仰

isNED：True，本地北东地坐标系。False，机体坐标系北东地

函数用以发布飞机的位置控制。根据输入重新设定飞机的工作模式后发布位置控制指令。

### 2.1.29 enFixedWRWTO()

函数用以向在跑道的飞机发送允许起飞指令。函数不需参数调用。

### 2.1.30 SendCruiseSpeed()

参数：Speed=0

Speed：飞机的巡航速度。

函数用以改变飞机的巡航速度。

### 2.1.31 SendCruiseRadius()

参数：rad=0

rad：飞机的巡航半径。

函数用以修改飞机的巡航半径。

### 2.1.32 sendMavTakeOffLocal()

参数：xM=0,yM=0,zM=0,YawRad=0,PitchRad=0,AscendRate=2

xM：位置信息

yM：位置信息

zM：位置信息

YawRad：偏航角速率

PitchRad：俯仰角速率

AscendRate：升率

函数用以发送期望本地位置指令给飞机，使飞机起飞到期望位置。

## 2.2 高精度模型+Simulink 控制器 组成的高精度综合模型

### 2.2.1 outHILStateData

uint32_t time_boot_ms; 消息时间戳

uint32_t copterID; 飞机 ID

int32_t GpsPos[3]; 模拟 GPS 位置，经纬度*1e7，高度*1e3，高度正方向是上

int32_t GpsVel[3]; 模拟 GPS 速度，北东地坐标系

int32_t gpsHome[3]; GPS 初始位置，单位与 GpsPos 相同

int32_t relative_alt; 高度，正向是上

int32_t hdg; 航向角，北东地坐标系。

int32_t satellites_visible; GPS 原始数据，卫星总和

int32_t fix_type; GPS 原始数据，固定翼类型

int32_t resrveInit; 留作未来使用

float AngEular[3]; 模拟姿态角，单位：弧度

float localPos[3]; 模拟位置，北东地坐标系。单位：米

float localVel[3]; 模拟速度，北东地坐标系，单位：米 / 秒

float pos_horiz_accuracy; GPS 水平精度，单位: 米

float pos_vert_accuracy; GPS 垂直精度，单位：米

float resrveFloat; 留作未来使用

### 2.2.2 outHILStateShort

int checksum;　　消息检查位
int32_t gpsHome[3];　　GPS 初始位置，经纬度*1e7，高度*1e3，高度正方向是上
float AngEular[3];　　模拟姿态角，单位：弧度
float localPos[3];　　模拟位置，北东地坐标系。单位：米
float localVel[3];　　模拟速度，北东地坐标系，单位：米 / 秒

### 2.2.3 inHILCMDData

uint32_t time_boot_ms;　　消息时间戳
uint32_t copterID;　　飞机 ID
uint32_t modes;　　飞行模式
uint32_t flags;　　飞行阶段标志
float ctrls[16];　　控制指令信息

### 2.2.4 inOffboardShortData

int checksum;　　消息检查位
int ctrlMode;　　控制模式标志位
float controls[4];　　控制指令信息

### 2.2.5 CreateStructure 函数

参数：SimStruct *S, int indx

函数的作用是在输入的结构体中存储一个 SInfo 类型的动态分配内存的指针。这个指针可以通过索引 indx 来访问或修改。

### 2.2.6 mdlStart 函数

参数：SimStruct *S

函数的主要作用是为每个飞机创建一个 UDP 服务器并启动它。函数将调用 StartUDPServer()函数来创建并开启 UDP 服务器。通过 CreateStructure()函数为每个飞机创建一个新的结构体。对于非 Linux 系统，这段代码会激活 Winsock DLL。Winsock DLL 是 Windows 系统上提供网络通信功能的库。

### 2.2.7 mdlOutputs 函数

参数：SimStruct *S, int_T tid
函数将获取 UDP 模式，根据不同的模式，更新飞机信息。

### 2.2.8 mdlUpdate 函数

参数：SimStruct *S, int_T tid
函数用以更新飞机的状态信息。

### 2.2.9 mdlTerminate()

函数用以终止模拟。

### 2.2.10 StartUDPServer()

参数：SimStruct *S, SInfo *info, int index
函数用以初始化并启动一个 UPD 服务器。

### 2.2.11 mdlInitializeSizes()

函数用以初始化 MATLAB 的 Simulink 自定义模块，主要作用是设置模型参数，包括输入输出端口的数量、数据类型和宽度等。

### 2.2.12 netServerInit()

函数的主要功能是在指定的端口上创建一个 UDP 服务器并返回套接字。

### 2.2.13 mdlCheckParameters()

函数用以检查传入参数是否满足要求。参数值，分别是端口号、车辆数量和 UDP 模式，并将其转换为整数格式。

## 2.3 基于质点模型的固定翼模型

在这部分最后增加了一个示例，用以更清晰的说明如何使用固定翼质点模型的接口来实现集群控制。

### 2.3.1 Vehicle: __init__()

函数原型：__init__(self,CopterID = 1,Vehicletype = 3,mapName='Grasslands', updatefreq = 100, isBroadCast=False)

参数：(self,CopterID = 1,Vehicletype = 3,mapName='Grasslands', updatefreq = 100, isBroadCast=False)

CopterID： 飞机 ID

Vehicletype： 飞机样式

mapName： 地图名称

updatefreq： 更新频率

isBroadCast： 是否广播，也就是飞机运动数据，是否发送到其他电脑

RecIPPort： Mavlink 接收接口 20100 系列，20100+(CopterID-1)*2

SendIpPort： Mavlink 发送接口 20101 系列，20100+(CopterID-1)*2+1

isInPointMode： 启用质点模型的标志位

CurFlag： 当前控制模式。0：Offboard 控制模式。2：位置模型。10：起飞模式滑跑阶段。11：起飞模式爬升阶段。12：平飞模式。121：盘旋模式。13：速度高度偏航模型。

CircleDir：飞机盘旋时盘旋方向。1： 顺时针盘旋。 2： 逆时针盘旋。

EnList： 控制模式的标志位。EnList[0] 位置控制。EnList[1] 速度控制。EnList[2] 加速度控制。EnList[3] 力控制。EnList[4] 偏航角控制。EnList[5] 偏航角速率控制。

### 2.3.2 initSimpleModel()

这个函数用于建立接收连接和发送连接，初始化对象.

intState: intState[0] 飞机初始化位置 intState[1] 飞机初始化位置 intState[2] 初始偏航角 表明飞机初始位置相对场景原点偏移

targetIP： 数据发送目标 IP 地址

GPSOrigin： 地图原点位置

这个函数是开启质点模型仿真的起点，将用以创建一个新的质点模型对象。这是函数被调用时的默认参数，可以根据自己的需求，修改相应的参数。

def initSimpleModel(self,intState=[0,0,0],targetIP = '127.0.0.1',GPSOrigin=[40.1540302,116.2593683,50]):

这个函数中会调用 UE 服务中的 getTerrainAltData 接口，用以获取飞机初始位置的高度。这是调用的代码行：self.intAlt=self.map.getTerrainAltData(self.intStateX,self.intStateY)

判断 self.updatefreq（数据更新频率），决定是否调用 SimpleModelLoop()，用以开启飞机的自身数据更新。

调用这个 RecMavLoop() 函数，从绑定的 20100 系列接口监听数据。

### 2.3.3 SimpleModelLoop()

这个函数将由 initSimpleModel()函数调用。

这个函数是质点模型的根函数。主要功能是当 self.isInPointMode 为 true 时，通过循环调用 ModelStep()函数不断更新模型信息，并将模型信息传输给外部接口。

其中会通过使用 self.updatefreq 参数来控制调用 ModelStep()函数的频率来实现控制数据更新的频率。 self.isInPointMode 为 true 时，表明当前模型是质点模型。

### 2.3.4 ModelStep()

模型数据开始更新的函数。包括开启将数据发送给外部接口的循环函数。这个函数已经在 SimpleModelLoop()函数中被调用。

这个函数首先通过判断当前时间和上一次被调用的时间的时间差，保证数据发送在正确的时间线上。首先调用 ProssInput()函数，处理用户发送的指令，更新飞机的状态信息。

在 Step()函数中，根据控制模型的不同，对 ProssInput()函数得到的数据进行进一步处理，更新飞机的姿态信息等。

调用 SendUavState()函数，将飞机的状态信息数据发送给 20101 系列端口，模拟 PX4 的内部状态。

调用 SendOutput()函数，将数据发送给 RflySim3D，模拟飞机的三维真值数据。

### 2.3.5 ProssInput()

这个函数将由 ModelStep()函数调用。用以处理用户的指令，生成飞机的期望控制信息。

根据模式发布控制指令。如果是 Offboard 控制模式，还要区分速度控制，位置控制等方式。根据 self.CurFlag 区分控制模式。这个参数定义已经在前文 2.3.1 中说明。

self.velOff: 期望速度控制。

self.yawRateOff： 期望偏航角速率。

这个函数中调用的 fixedPitchFromVel()函数的功能是根据速度指令生成飞机的俯仰角。俯仰角的单位是弧度。

### 2.3.6 Step()

这个函数由 ModelStep()函数中调用。

这个函数里面，用龙格库塔法之类，向前推进一步. 对于不同的模式，计算飞机的期望状态信息。

uavVelNED 北东地坐标系下无人机速度

uavPosNED 北东地坐标系下无人机相对于场景原点的偏移位置。

这个函数中将根据 self.CurFlag 参数确定当前的模型，这个参数定义已经在前文 2.3.1 中说明。

这个函数将再次调用 getTerrainAltData 接口来确定飞机真实位置的地图高度。

### 2.3.7 SendUavState()

这个函数将由 ModelStep()函数调用。

将 Step()函数更新的飞机状态信息发送到绑定的 20100 系列接口。

函数的具体操作是设定发送数据的校验位。将校验位，经纬高的场景原定位置，飞机的姿态角信息，飞机的偏移位置和飞机的速度打包。将数据发送到 20100 系列接口。

接口的设定方式，在 2.3.1 中做了说明。

### 2.3.8 SendOutput()

这个函数将由 ModelStep()函数调用。

这个函数将调用 sendUE4PosNew() 函数，将更新的数据发送到 RflySim3D。

打包发送的数据包括：飞机的 ID，飞机的样式，北东地坐标系下的飞机真实位置，飞机的姿态角信息，北东地坐标系下无人机的速度信息，飞机电机的转速。

### 2.3.9 fixedPitchFromVel()

这个函数将由 ProssInput() 函数调用。

函数的输入为飞机的飞行速度。计算返回飞机期望俯仰角。

### 2.3.10 RecMavLoop()

这个函数将由 initSimpleModel()函数调用。

函数将调用的函数有 SendPosNED()函数，SendMavArm()函数，SendCruiseSpeed()函数，SendCruiseRadius()函数，sendMavTakeOff()函数，sendMavTakeOffGPS()函数，SendVelYaw Alt()函数。

调用的函数的主要功能分别是，发送位置控制指令，发送飞机解锁指令，发送飞机巡航速度指令，发送飞机盘旋半径指令，发送起飞指令，发送经纬度坐标系下的起飞指令，发送飞机飞行控制指令。

函数的主要功能为开启数据监听循环。根据控制模式，调用不同的数据发送函数发送数据。会根据 ctrlMode 选择不同的调用函数，其中 ctrlMode 由 ListenDate[1] 确定。

ListenDate 为接收到的信息。ListenDate[0] 数据校验位。ListenDate[1] 控制模式标志位。 ListenDate[2:6] 控制信息位。

控制模式位：

0：空命令

2：位置控制，将调用 SendPosNED()函数。

9：解锁，将调用 SendMavArm()函数。

10：盘旋，将调用 SendCruiseSpeed()函数，SendCruiseRadius()函数。

11：起飞，调用 sendMavTakeOff()函数。

12：全球坐标系起飞，调用 sendMavTakeOffGPS()函数。

13：速度高度偏航控制，调用 SendVelYawAlt()函数。

### 2.3.11 SendPosNED()

这个函数由 RecMavLoop() 函数调用。

发送位置控制指令。修改 EnList 调整控制模式，发送期望位置信息和期望偏航。

当 ctrlMode == 2 时， ListenDate[2:6] 分别代表 x,y,z,yaw 控制信息。

### 2.3.12 SendMavArm()

这个函数将在 RecMavLoop() 函数中调用。

发送飞机解锁指令。

self.isArmed 飞机解锁的标志位。

### 2.3.13 SendCruiseSpeed()

这个函数将在 RecMavLoop() 函数中调用。

当 ctrlMode == 10 时， ListenDate[2:4] 表示无人机的巡航速度和盘旋半径。

设置固定翼无人机的巡航速度。

### 2.3.14 SendCruiseRadius()

这个函数将在 RecMavLoop() 函数中调用。

设置固定翼的盘旋半径。

### 2.3.15 sendMavTakeOff()

这个函数将在 RecMavLoop() 函数中调用。

进行起飞模式，并设定好后续起飞需要的参数。

### 2.3.16 sendMavTakeOffGPS()

这个函数将在 RecMavLoop() 函数中调用。

具有和 2.3.15 相同的功能，区别在于这个函数使用经纬度坐标信息作为输入。

### 2.3.17 SendVelYawAlt()

这个函数将在 RecMavLoop() 函数中调用。

当 ctrlMode == 13 时， ListenDate[2:5] 表示无人机的速度，高度，偏航信息。

发送速度，高度和偏航控制的接口。

### 2.3.18 sendUE4PosNew()

这个函数将由 SendOutput() 函数调用。

这个函数将调用 sendBuf()函数。

这个函数的主要作用是把收到的数据通过调用 sendBuf()函数发送给 RflySim3D。

这个函数将把需要发送的数据打包，打包的数据包括：数据校验位，仿真飞机的 ID，飞机的样式，飞机电机的转速，飞机的速度，飞机的姿态信息，飞机在场景中的偏移位置，程序的运行时长。说明顺序与程序中打包顺序相同。

### 2.3.19 sendBuf()

这个函数将由 sendUE4PosNew()函数和 sendUE4Cmd()函数调用。

函数首先区分是否确认 WindowID，如果确认 WindowID，则向指定端口发送数据。如果没有确定，则由程序循环确定。

函数的功能是根据 self.isBroadCast 参数判断是否为广播模式，如果不是广播模式，则向本机发送数据，如果是广播模式，则向网络中的所有电脑发送数据。

发送端口为 20010 系列端口。

### 2.3.20 EndSimpleModel()

这是结束质点模型仿真的函数。函数将 self.isInPointMode 和 self.t4Flag 两个参数设置为 False，从而结束模型自身数据的更新循环和监听数据的循环。

调用这个函数时，不需要输入参数。

### 2.3.21 模型使用示例

```
mav = VehicleApi.Vehicle(1,100,'OldFactory')
```

首先使用 VehicleApi 的 Vehicle 类 生成对象。这里说明飞机 ID，数据更新频率，地图名称。

__init__(self,CopterID = 1,Vehicletype = 3,mapName='Grasslands', updatefreq = 100, isBroadCast=False) 这是函数可以设置的参数列表，依次是飞机 ID，飞机样式，地图名称，数据更新频率，是否为广播模式。可以看到，默认飞机 ID 是 1，飞机样式是 3，地图是草地，更新频率是 100，不是广播模式。

```
mav.initSimpleModel([-250,-119,0])
```

调用 initSimpleModel()函数初始化仿真对象。这里设置飞机相对场景原点的偏移位置是[-250,-119,0]。

initSimpleModel(self,intState=[0,0,0],targetIP = '127.0.0.1',GPSOrigin=[40.1540302,116.2593683,50]) 可设置的参数有，飞机相对场景原点的偏移位置，数据发送的 IP 地址，场景原点坐标。默认飞机相对场景原点没有偏移，数据发送的目标 IP 地址是'127.0.0.1'，场景原点设置为[40.1540302,116.2593683,50]，这是经纬度坐标信息。

```
mav.sendMavTakeOff(500, 0, -100, 0)
```

这里调用飞机起飞函数。这里设置飞机盘旋的坐标中心在(500,0,-100)。单位是 m，坐标中心的坐标系是北东地坐标系。

sendMavTakeOff(self,xM=0,yM=0,zM=0,YawRad=0,PitchRad=20/180.0*math.pi) 调用时可以设置的参数有 飞机的盘旋中心坐标，滑跑时的偏航方向，爬升阶段时的斜率。

```
mav.SendMavArm(True)
```

这里调用飞机的解锁函数。这里只有一个参数。

```
mav.EndSimpleModel()
```

这是结束仿真的函数。

### 2.3.22 模型图解

## 2.4 基于质点模型的旋翼模型

### 2.4.1 PX4MavCtrler:__init__()

函数原型：def __init__(self, port=20100, ip='127.0.0.1'):

参数：(self, port=20100, ip='127.0.0.1')

这个函数将会创建数据输出接口以及数据监听接口。并初始化无人机数据。

Port: 端口号

Ip：通信地址

isInPointMode：启用质点模型的标志位

isCom：硬件在环仿真标志位

type_mask：控制方式标志位

coordinate_frame：坐标系选用

EnList：控制模式的标志位。EnList[0] 位置控制。EnList[1] 速度控制。EnList[2] 加速度控制。EnList[3] 力控制。EnList[4] 偏航角控制。EnList[5] 偏航角速率控制。

uavAngEular：PX4 姿态角

trueAngEular：CopterSim DLL 模型的真实模拟姿态角

uavAngRate：PX4 姿态角速率

trueAngRate： CopterSim DLL 模型的真实模拟角速率

uavPosNED：PX4 相关于起飞位置的当前位置（北东地坐标系）

truePosNED： 相关于 UE4 地图原点 CopterSim DLL 模型的真实模拟位置

uavVelNED：PX4 北东地坐标下的速度

trueVelNED： CopterSim DLL 模型的真实模拟速度

isVehicleCrash： 发生碰撞的标志位

isVehicleCrashID： 与本机发生碰撞的飞机 ID

uavPosGPS： PX4 的 GPS 位置在北东地坐标系下，经度，维度，高度，时间，相关高

度，速度，航向。

truePosGPS： CopterSim DLL 模型的真实模拟 GPS 位置

uavPosGPSHome： PX4 北东地坐标系下起飞位置（GPS 原点）

uavGlobalPos： 由 PX4 获取的位置信息转换为 UE4 中的位置

trueAngQuatern： CopterSim DLL 模型的真实模拟四元数

trueMotorRPMS： CopterSim DLL 模型的真实模拟电机速率

trueAccB： CopterSim DLL 模型的真实模拟加速度

### 2.4.2 initPointMassModel()

这个函数用以启用质点模型

参数：(self,intAlt=0,intState=[0,0,0])

intAlt：CopterSim 的初始高度相对于当前地图的地面高度

intState： [0]：初始位置 [1]：初始位置 [2]：偏航角

### 2.4.3 PointMassModelLoop()

这个函数将由 initPointMassModel() 函数调用。

这个函数将会初始化飞机状态。并开启模型循环。将会 EnList 标志位确定控制方式。更新飞机的状态信息，包括位置，速度，角速度等。

这个函数将调用 sendUE4PosNew()将更新数据发送到 UE4 中。

### 2.4.4 sendUE4PosNew()

这个函数由 PointMassModelLoop()调用。

这个函数用以将飞机的信息打包发送给指定地址和端口。使用 20010 系列端口进行数据发送。打包的数据包括：数据校验位，仿真飞机的 ID，飞机的样式，飞机电机的转速，飞机的速度，飞机的姿态信息，飞机在场景中的偏移位置，程序的运行时长。说明顺序与程序中打包顺序相同。

### 2.4.5 SendPosNED()

发送位置控制指令。修改 EnList 调整控制模式，发送期望位置信息和期望偏航。

当 ctrlMode == 2 时， ListenDate[2:6] 分别代表 x,y,z,yaw 控制信息。

### 2.4.6 EndPointMassModel()

这是结束质点模型仿真的函数。函数将 self.isInPointMode 参数设置为 False，从而结束模型自身数据的更新循环和监听数据的循环。PointMassModelLoop()函数会根据

isInPointMode 判断程序运行。因此将 isInPointMode 设为 False 后，会自动停止。

调用这个函数时，不需要输入参数。

## 2.5 固定翼制导模型

### 2.5.1 Vehicle: __init__()

参数：self,CopterID = 1,Vehicletype = 3,mapName='Grasslands', updatefreq = 100, isBroadCast=False

CopterID：CopterSim 中仿真 ID

Vehicletype：飞机在 RflySim3D 中的显示样式

mapName：RflySim3D 中的场景

updatafreq：往 RflySim3D 发送消息的频率，同时也是模型的默认更新频率

isBroadCast：表示飞机的三维数据是否广播到局域网其他电脑

这个函数的功能时初始化飞机以及场景。同时确定模型运行的信息交互方式。

### 2.5.2 initSimpleModel()

参数：self,intState=[0,0,0],targetIP = '127.0.0.1',GPSOrigin=[40.1540302,116.2593683,50]

intState：飞机的初始化位置，初始偏航角

targetIP：飞机数据的发送 IP 地址

GPSOrigin：飞机初始化位置的 GPS 位置

这个函数的功能是初始飞机模型。确定飞机的初始化状态和信息交互方式。

### 2.5.3 SendMavArm()

参数：isArm

isArm：解锁标志位

### 2.5.4 sendMavTakeOff()

参数：xM=0,yM=0,zM=0,YawRad=0,PitchRad=20/180.0*math.pi

xM: 下一个航路点的位置

yM: 下一个航路点的位置

zM: 下一个航路点的位置

YawRad: 滑跑的偏航方向

PitchRad: 爬升的斜率

### 2.5.5 SendPosNED()

参数：x=0,y=0,z=0,yaw=0

x: 目标位置

y: 目标位置

z: 目标位置

yaw: 偏航角

### 2.5.6 SendCruiseRadius()

参数：rad=20

rad：固定翼的盘旋半径

### 2.5.7 SendCruiseSpeed()

参数：Speed=10

speed：固定翼的巡航速度

### 2.5.8 sendMavTakeOffGPS()

参数：lat,lon,alt,yawDeg=0,pitchDeg=15

lat：维度信息

lon：经度信息

alt：高度信息

yawDeg：偏航方向设置，这里单位是°

pitchDeg：爬升角度设置，这里单位是°

### 2.5.9 SendVelYawAlt()

参数：vel=10,alt=-100,yaw=6.28

vel：速度控制

alt：高度控制

yaw：偏航角度

### 2.5.10 EndSimpleModel()

参数：不需要参数

调用函数用以关闭模型仿真。

### 2.5.11 UEMapServe：LoadPngData()

参数：name

name：要读取的文件名

### 2.5.12 getTerrainAltData()

参数：xin,yin

xin：要获取高度信息的位置的 x 坐标

yin：要获取高度信息的位置的 y 坐标

### 2.5.13 EarthModel：lla2ecef()

参数：lat, lon, h

lat：纬度

lon：经度

h：高度

由经纬度坐标系转换为地心坐标系

### 2.5.14 ecef2enu()

参数：x, y, z, lat0, lon0, h0

x：位置信息

y：位置信息

z：位置信息

lat0：纬度信息

lon0：经度信息

h0：高度信息

由地心坐标系转换为东北天坐标系

#### 2.5.15 enu2ecef()

参数：xEast, yNorth, zUp, lat0, lon0, h0

xEast：

# 3. 集群通信接口和协议

## 3.1 CopterSim 通信模式

### 3.1.1 UDP_Simple 模式

**UDP_Simple 模式**适用于集群开发，为完成基本的位置、速度控制提供了最简实现。在 UDP_Simple 模式下，接收的控制指令通过 inOffboardShortData 描述。包含校验位 checksum、坐标模式选择 ctrlMode 和 4 个 float 控制量 controls[4]。ctrlMode 可以有多种协议（要改 RflyUdpFast.cpp 和 CopterSim），ctrlMode 这里如果设置成<0（小于 0），表示是空命令，模块不会向外发布消息。ctrlMode 支持的具体协议如下表所示。

在 UDP_Simple 模式下，一旦收到了 inOffboardShortData 消息，那么 Copter 会自动给 PX4 发送解锁和进入 Offboard 模式的消息。这样就简化了用户控制飞机的流程。

```
struct inOffboardShortData{
    int checksum; // 校验位 1234567890
    int ctrlMode; // 模式选择
    float controls[4]; //四位控制量
}
```

UDP_Simple 支持的控制模式

| 模式标号 | 描述 |
| --- | --- |

| | |
|---|---|
| 0 | 导航坐标系下速度模式[vx,vy,vz, yaw_rate] |
| 1 | 机体坐标系下速度模式[vx,vy,vz, yaw_rate] |
| 2 | 导航坐标系下位置模式[x,y,z, yaw] |
| 3 | 机体坐标系下位置模式[x,y,z, yaw] |
| 4 | 姿态油门控制指令[滚转、俯仰、偏航(弧度)、油门(0~1)]，可自动解锁，可自动进入 OffBoard 模式 |
| 5 | 姿态油门增量控制指令[滚转、俯仰、偏航、油门增量]--可自动解锁，可自动进入 OffBoard 模式 |
| 6 | 加速度控制模式[ax,ay,az,yaw] |
| 7 | 加速度控制模式[ax,ay,az,yaw_rate] |
| 8 | 加速度控制模式[ax,ay,az,yaw_rate] |
| 9 | 解锁所示模式[解锁,-,- ,-] |
| 10 | 表示设置固定翼飞机的速度和盘旋半径，[speed, radius, -, -] |
| 11 | 表示 Mavlink 起飞命令，自动解锁，导航坐标系位置[x, y, z, -] |
| 12 | 表示 Mavlink 起飞命令，自动解锁，GPS 坐标系位置[纬度, 经度, 高度, -] |
| 13 | 速度高度航向命令，自动解锁并进入 offBoard 模式，GPS 坐标系位置[速度, 高度, 航向, -] |
| 14 | Global 坐标系下位置模式，GPS 坐标系位置[lat_int, lon_int, alt_float, yaw_float] |
| 30 | 用于 VTOL 模式切换，表示飞行模态切换指令，对应 MAV_CMD_DO_VTOL_TRANSITION 的 mavlink 命令，controls[0]表示 State 位。定义见链接（https://mavlink.io/en/messages/common.html#MAV_VTOL_STATE）。<br>controls[1]表示 Immediate 位（1: Force immediate 前置切换, 0: normal transition 普通切换.）。 |

上述的控制模式，用于给 PX4 发送控制指令。通常控制算法在计算出控制指令时，需要依据载具当前的位置、速度信息。UDP_Simple 模式提供了获取这些信息的数据结构。

如下所示是 UDP_Simple 模式接收 CopterSim 的简化数据。具体而言 checksum 需要确保为 1234567890，来验证数据正确性。gpsHome 飞机起飞点的经纬高数据。AngEular 飞机的欧拉角，俯仰滚转偏航，单位为度。localPos 飞机相对起飞点的相对坐标，北东地坐标系，单位米。localVel 飞机的速度，北东地，单位 m/s。gpsHome 是 int 型，先得到度（乘 1e-7）度（乘 1e-7）米（乘 1e-3）的格式，再用 Simulink 的模块，结合统一的 GPS 原点，可以得到本飞机相对统一 GPS 原点的偏移位置，再结合 localPos，可以得到飞机相对统一原点的实时

位置。

```
struct outHILStateShort{
    int checksum; //校验位 1234567890
    int32_t gpsHome[3];
    float AngEular[3];
    float localPos[3];
    float localVel[3];
}
```

由上的描述可以看出,UDP_Simple 模式下只提供了最简单的指令和获取最简单的信息。

### 3.1.2 UDP_Full 模式

UDP_Full 模式下，收到 Offboard 消息同样也会自动解锁并进入 Offboard 模式。所以 UDP_Full 模式下，用户也不需要手动发送指令来解锁和进入 Offboard 模式。UDP_Full 模式下收到的数据如 4.1.1 所示，用 outHILStateData 表示。包含 GPS 坐标系下的位置、速度，NED 坐标系下位置、速度等。在实际使用的时候，并不需要将 outHILStateData 里面的所有数据度解析出来，只需将感兴趣的数据提取即可。outHILStateData 是传感器数据经过 PX4 EKF2 处理后的结果。

对于 CopterSim 而言要传输 UDP 的数据，还需对数据包进行进一步封装。如下是 **netDataShortShort** 的数据结构，最大支持 112 个数据。由此可见 netDataShortShort，是完整支持了 outHILStateData 数据的传输。

```
typedef struct _netDataShortShort {
  TargetType tg;
  int            len;
  char           payload[112];
}netDataShortShort;
```

用户还可以获取真实的数据 **SOut2Simulator**，该类型数据既可以在 UDP_Simple 模式下读取也可以在 UDP_Full 模式下读取，因为这类数据使用的是额外的端口。CopterSim 相应的收端口为 30100，发端口为 30101 端口。SOut2Simulator 是来自模型的真实状态信息，没有添加噪声，这也是只有在仿真过程中才能获取的数据。

类似的，SOut2Simulator 数据通过 **netDataShort** 结构进行传输，也就是说 SOut2Simulator 所有数据的总长度是 192 个字节。

```
typedef struct _netDataShort {
        int tg;
        int            len;
```

```
    char        payload[192];
}netDataShort;
```

## 3.2 集群通信接口

### 3.2.1 20100 系列端口

对于非 Simulink_DLL 模式，使用 20100 系列端口进行通信。对于 CopterSim 而言，使用 20100+2n 端口收消息，使用 20101+2n 端口发消息，其中 n=0,1,2…。当增加飞机数量时，端口值逐步递增，如增加第 2 架飞机，其端口为 20102、20103。

### 3.2.2 30100 系列端口

对于 Simulink_DLL 模式，CopterSim 使用 30100 端口接收信息，使用 30101 端口发送信息。类似的，当有多架飞机时，端口也从 30100 开始递增。

### 3.2.3 20010 系列端口

用以 UE 通信.

SocketReceiverMap：接收本机 20010~20029 端口中的一个（这是因为一台电脑上多个 RflySim3D 会争夺端口，所以根据序号会依次分配这 20 个端口）

## 3.3 集群通信协议

### 3.3.1 基本数据结构

每个数据结构包含一个 reset()函数，用以重置数据。

#### 3.3.1.1 outHILStateData

```
uint32_t time_boot_ms;   消息时间戳
uint32_t copterID;       飞机 ID
int32_t GpsPos[3];       模拟 GPS 位置，经纬度*1e7，高度*1e3，高度正方向是上
int32_t GpsVel[3];       模拟 GPS 速度，北东地坐标系
```

int32_t gpsHome[3]; GPS 初始位置，单位与 GpsPos 相同

int32_t relative_alt; 高度，正向是上

int32_t hdg; 航向角，北东地坐标系。

int32_t satellites_visible; GPS 原始数据，卫星总和

int32_t fix_type; GPS 原始数据，固定翼类型

int32_t resrveInit; 留作未来使用

float AngEular[3]; 模拟姿态角，单位：弧度

float localPos[3]; 模拟位置，北东地坐标系。单位：米

float localVel[3]; 模拟速度，北东地坐标系，单位：米 / 秒

float pos_horiz_accuracy; GPS 水平精度，单位: 米

float pos_vert_accuracy; GPS 垂直精度，单位：米

float resrveFloat; 留作未来使用

#### 3.3.1.2 outHILStateShort

int checksum; 消息检查位

int32_t gpsHome[3]; GPS 初始位置，经纬度*1e7，高度*1e3，高度正方向是上

float AngEular[3]; 模拟姿态角，单位：弧度

float localPos[3]; 模拟位置，北东地坐标系。单位：米

float localVel[3]; 模拟速度，北东地坐标系，单位：米 / 秒

#### 3.3.1.3 inHILCMDData

uint32_t time_boot_ms; 消息时间戳

uint32_t copterID; 飞机 ID

uint32_t modes; 飞行模式

uint32_t flags; 飞行阶段标志

float ctrls[16]; 控制指令信息

#### 3.3.1.4 inOffboardShortData

int checksum; 消息检查位

int ctrlMode; 控制模式标志位

float controls[4]; 控制指令信息

### 3.3.2 平台私有 UDP 协议

有三个模式，分为 FullData 模式，SimpleData 模式，UltraSimple 模式。

Fulldata 模式下，模块输入为 15 维的 double 型向量，模块输出为 28 维的 double 型向量。

SimpleData 模式下，输入为 5 维的 double 型向量，输出为 12 维 double 型的向量。

UltraSimple 模式下，输入为 5 维的 double 型向量，输出为 12 维 double 型的向量

具体定义见 4.1.4

### 3.3.3 Mavlink 协议

MAVLink 协议对载具类型、飞行模式、载具状态、系统组件、消息帧坐标格式、地面站指令格式、微型载具的数据流等进行了规定。下面就协议的几个重点内容进行介绍。

(1)载具类型：包括通用、固定翼、四旋翼、共轴双桨结构、普通有尾旋翼直升机、自由飞气球、火箭、地面车辆、潜艇等

(2)载具状态：该协议要求可以将载具状态实时传回地面站，包括：未初始化/未知状态、正在启动、正在校准、待命、开车、系统失常但可导航、完全失常、执行关机指令。

(3)系统组件：协议要求至少对下列组件进行支持：GPS、任务管理器、航线管理器、地图、照相机/摄像机、3 个姿态传感器、网络和数传中继、系统控制器、14 个舵机。GPS 和地图在 MAVLink 协议中占据重要地位。GPS 可以为飞行控制器提供重要的经纬度信息，并辅助提供高度信息。微型载具能够按照规划的轨迹运动完全依靠 GPS 的定位导航功能。因此地面站需要实时显示 GPS 的卫星数、定位精度、载具所在位置等信息。MAVLink 完全提供了这些功能。

(4)航线规划指令：对于普通无人机来说能够精确地按航线飞行是飞行器的核心任务。MAVLink 可以分为立即执行和任务脚本两种指令。指令参数规定为 7 个，分别代表不同的数据类型。这些规定参照民航飞机的 ARINC424 导航数据库标准，包括航点、持续盘旋、在航点盘旋 N 圈、在航点盘旋 N 秒、返回起飞点、设定点着陆等指令。这些丰富的指令保证了飞行器可以使飞机按照航线飞行，在航测中，要求飞行器能够严格地按照航线飞行，以覆盖所有待测绘的地区，以在后期对所拍摄的图片进行拼接。而立即执行的指令可以使地面操作人员灵活地操作无人机进行航拍作业，微型无人机航拍不仅价格低，还具有比有人飞机飞得低、比手持设备飞得快的特点，可以完成不同拍摄视角的视频，受到普遍关注。但是 MAVLink 受数据吞吐率的限制无法实现视频传输。将数据传输与图像传输相融合已成为完善 MAVLink 方向。

# 4. RflyUdpFast 接口

## 4.1 Simulink 组件

### 4.1.1 IP Address

需要控制飞机的 CopterSim 所在电脑的 IP 地址，通常情况下，取本机地址“127.0.0.1”即可。如果要在本电脑上控制另一台电脑上的 CopterSim，则可以在此处填入目标电脑的 IP 地址。

### 4.1.2 Start CopterID（起始飞机 ID）

UDP Port 是第一个飞机的初始端口号，默认起始端口是 20100。每个 CopterSim 的收发消息需要各占用一个端口

### 4.1.3 Vehicle Number（飞机数量）

Vehicle number 飞机数量表示需要连接的 CopterSim 数量，该模块的输入输出端口数量由该选项控制，如果输入 10，则模块会自动生成 10 对输入输出接口。

### 4.1.4 UDP Mode（通信模式）

和 CopterSim 界面上的“通信模式”下的 UDP_Full 和 UDP_Simple 相对应。Full 模式数据更全，但是数据量较大，适合少量飞机时的仿真与控制；Simple 模式数据更精简，适合大规模集群时的控制。

#### A. FullData 完整模式

模块输入为 15 维的 double 型向量，具体定义（实现 MAVLink 的 Offboard 消息）如下

第 1 维：time_boot_ms;  %当前时间戳（填 0 即可，目前没有使用）

第 2 维：copterID; %飞机 ID（填 1 即可，目前没有使用）

第 3 维：type_mask; %输入控制模式（同 Offboard 定义）

第 4 维：coordinate_frame; %坐标系模式（同 Offboard 定义）

第 5~15 维：ctrls[11]; %分别对应了 3 维的期望位置 pos,3 维的期望速度 vel，3 维的期望加速度 acc，1 维的期望偏航角 yaw，1 维的期望偏航角速度 yawRate。（同 Offboard 定义）

模块输出为 28 维的 double 型向量（全部转发自 Pixhawk 内部滤波值），具体定义如下

第 1~3 维：gpsHome[3]; %Home 点（上电之后不会变）的经纬高坐标，经纬度需要除以 1e7 才能得到度为单位的经纬度，高需要除以 1e3 才能得到 m 为单位的高（向上为正）

第 4~6 维：AngEular[3]; %Pixhawk 估计得到的姿态欧拉角，单位弧度

第 7~9 维：localPos[3]; %Pixhawk 估计得到的以 gpsHome 为原点的相对北东地位置向量，单位 m，z 轴向下为正

第 10~12 维：localVel[3]; %北东地的运动速度向量，单位 m/s

第 13~15 维：GpsPos[3]; %实时的 GPS 位置，单位和 gpsHome 相同，但是会实时变化

第 16~18 维：GpsVel[3]; %GPS 速度，需要除以 100 得到 m/s 为单位的速度

第 19 维：time_boot_ms；%上电时间

第 20 维： copterID; %飞机 ID

第 21 维： relative_alt; % GPS 相对高度，需要除以 1000 得到 m 为单位的高度，向上为正

第 22 维： hdg; % GPS 航向角，需要除以 1000 得到 0~360 度范围的角度

第 23 维： satellites_visible; %可见卫星数量

第 24 维： fix_type; %定位精度

第 25 维： resrveInit; % int 类型的保留位

第 26 维： pos_horiz_accuracy; %水平定位精度，单位 m

第 27 维： pos_vert_accuracy; %竖直定位精度，单位 m

第 28 维： resrveFloat; % float 型保留位，未被启用

第 29-31 维： GlobalPos [3]; % CopterSim 全局位置，单位 m。根据飞机的 GPS 坐标解算得到，以 Ue 地图的坐标中心为原点，使用多机控制时，应该使用本接口。

## B. SimpleData 精简模式

- 输入为 5 维的 double 型向量，具体定义（实现 MAVLink 的 Offboard 消息）如下

第 1 维：ctrlMode; %第一位为标志位，0: 表示地球速度控制模式 Earth Vel; 1: 机体速度控制模式 Body Vel; 2: 地球位置控制模式 Earth Pos; 3:机体位置控制模式 Body Pos

第 2~5 维：如果 ctrlMode =0，则这四维对应了地球坐标系（以解锁时的位置）下的 vx, vy,vz 速度+ yawRate 偏航速率信号；如果 ctrlMode =1，则这四维对应了机体坐标系（以解锁时机体姿态和位置）下的 vx,vy,vz 速度+ yawRate 偏航速率信号；如果 ctrlMode =2，则这四维对应了地球坐标系（以解锁时的位置为原点）下的 x,y,z 位置+ yaw 偏航信号；如果

ctrlMode =3，则这四维对应了机体坐标系（以解锁时机体姿态和位置）下的 x,y,z 位置+ yaw 偏航信号。

一般 ctrlMode=0 比较多，直接给定全局速度做轨迹控制。如果做位置和速度的切换，需要实时修改 ctrlMode 的值，并调整第 2~5 维信号的定义。

注：如果 ctrlMode<0 则不会发送控制指令。

- 输出为 12 维 double 型的向量，顺序定义如下

第 1~3 维： GlobalPos [3]; % CopterSim 全局位置，单位 m。根据飞机的 GPS 坐标解算得到，以 Ue 地图的坐标中心为原点，使用多机控制时，应该使用本接口。

第 4~6 维： AngEular[3]; % Pixhawk 估计得到的姿态欧拉角，单位弧度

第 7~9 维： localPos[3]; % Pixhawk 估计得到的，以 gpsHome 为原点的相对北东地位置向量，单位 m，z 轴向下为正。本接口以每个飞机的各自上电 GPS 坐标为原点，在集群仿真时，不应该直接使用本接口，因为所有飞机起飞时，localPos 都是 0，但是初始位置在地图上时有区别的。

第 10~12 维： localVel[3]; % 北东地的运动速度向量，单位 m/s。

### 4.1.5 GPS Orin（GPS 坐标原点）

GPS 坐标原点，本值用于解算每个飞机的 GlobalPos，通过收到数据的当前 GPS 坐标，根据统一的 GPS 坐标原点，可以得到相对坐标原点的 xyz 坐标，这个坐标就是 GlobalPos，和 CopterSim 软件上的 PosE 是一致的，对应地图上的全局坐标。


### 4.1.6 Sample Time（采样时间）

Sample Time 采样时间，该时间应该与 Simulink 仿真时间对应。

### 4.1.7 RflySwarmAPI 模块优点

效率高：C/C++语言直接实现，比 m 语言高效

运算小：Simulink 自带 UDP 模块输入输出都是高维（几百维）的 unit8 向量，在飞机增多时，整个项目维度急剧扩张；而 S 函数方式直接输出 double 型的位置、速度等数据，维度低（几维），运算量小

延迟低：Simulink 自带的 UDP 模块为了防止数据丢失，会将收到数据全都保存在缓存中，依次调用，这样当外部程序发送频率大于 Simulink 的运行频率是，就会产生大的延迟；而编写 S 函数的方法则能避免本问题

更可靠：Simulink 自带的 UDP 模块每个仿真步长读取一个数据，如果外部程序发送频率小于 Simulink 运行频率，则 Simulink 没有读到数据，会输出 0，导致运算出错；而自己编写 S 函数则可以避免本问题。

扩展性强：S 函数可以轻易扩展为串口、Tcp、共享内存、MAVLink 等其他通信协议

## 4.2 RflyUdpFast.cpp 接口

### 4.2.1 CreateStructure 函数

参数：SimStruct *S, int indx

函数的作用是在输入的结构体中存储一个 SInfo 类型的动态分配内存的指针。这个指针可以通过索引 indx 来访问或修改。

### 4.2.2 mdlStart 函数

参数：SimStruct *S

函数的主要作用是为每个飞机创建一个 UDP 服务器并启动它。函数将调用 StartUDPServer()函数来创建并开启 UDP 服务器。通过 CreateStructure()函数为每个飞机创建一个新的结构体。对于非 Linux 系统，这段代码会激活 Winsock DLL。Winsock DLL 是 Windows 系统上提供网络通信功能的库。

### 4.2.3 mdlOutputs 函数

参数：SimStruct *S, int_T tid

函数将获取 UDP 模式，根据不同的模式，更新飞机信息。

### 4.2.5 mdlUpdate 函数

参数：SimStruct *S, int_T tid
函数用以更新飞机的状态信息。

### 4.2.6 mdlTerminate()

函数用以终止模拟。

### 4.2.7 StartUDPServer()

参数：SimStruct *S, SInfo *info, int index
函数用以初始化并启动一个 UPD 服务器。

### 4.2.8 mdlInitializeSizes()

函数用以初始化 MATLAB 的 Simulink 自定义模块，主要作用是设置模型参数，包括输入输出端口的数量、数据类型和宽度等。

### 4.2.9 netServerInit()

函数的主要功能是在指定的端口上创建一个 UDP 服务器并返回套接字。

### 4.2.10 mdlCheckParameters()

函数用以检查传入参数是否满足要求。参数值，分别是端口号、车辆数量和 UDP 模式，并将其转换为整数格式。

## 4.3 接口使用示例

### 4.3.1 单机控制示例

**A. UDP FullData 模式**

**B. UDP SimpleData 模式**

**C. UDP UltraSimpleData 模式**

### 4.3.2 多机控制示例

**A. UDP FullData 模式**

**B. UDP SimpleData 模式**

**C. UDP UltraSimpleData 模式**

### 4.3.3 外部机控制示例

**A. bat 脚本编写注意**

**B. Broadcast 模式**

**C. Use IP 模式**

### 4.3.4 带自动防撞的无编队集群实验

### 4.3.5 数据记录与分析

# 5. PX4MavCtrlV4 接口

## 5.1PX4MavCtrler

### 5.1.1UAV 参数说明

### 5.1.2 简化模型函数说明

**a. initPointMassModel 函数**

**b. PointMassModelLoop 函数**

**c. EndPointMassModel 函数**

### 5.1.3 InitMavLoop 函数

### 5.1.4 initOffboard 函数

### 5.1.5 SendPosNED 函数

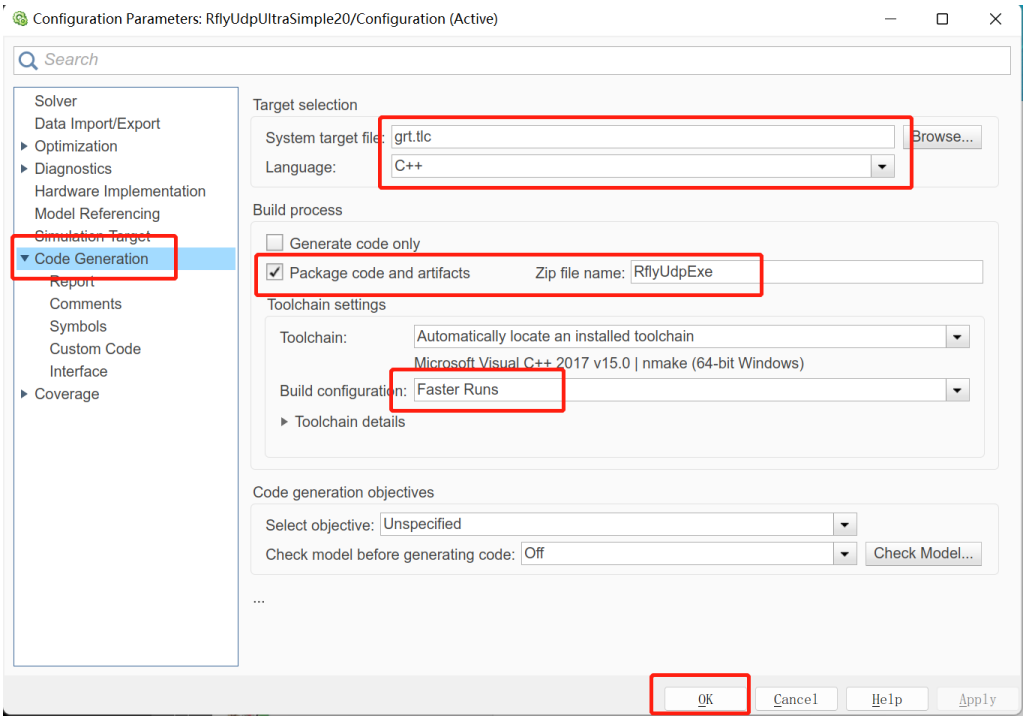
### 5.1.6 SendMavArm 函数

### 5.1.7 SendVelNED 函数

### 5.1.8 endOffboard 函数

### 5.1.9 sendRebootPix 函数

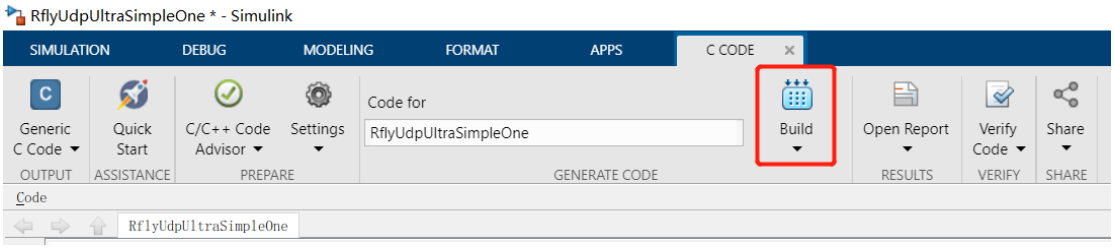
### 5.1.10 initUE4MsgRec 函数




## 6.3 操作步骤

选择要生成 e x e 的 slx 文件，保证 GenerateSwarmExe.p、RflyUdpFast.cpp 和 RflyUdpFast. mexw64 拷贝到同一目录。

打开 s l x 文件，生成工具栏后如下图点击 Build 按钮就可以生成代码。




编译完成后，在“诊断视图”（Diagnostic Viewer）中得到编译成功完成提示（请忽略黄色的警告）我们可以得到一个“RflyUdpExe.zip”压缩包文件，里面包含所有生成的代码，会用于我们后续生成 exe 文件。

注意：上一步编译命令时生成了一个 e x e 文件，双击它可以运行，但是它不符合实时控制要求（始终以最快速度运行，不和现实时钟同步），因此需要运行我们的脚本来生成实时控制的 exe 文件。

鼠标右键点击 GenerateSwarmExe.p 文件，并点击“运行。或者直接在 MATLAB 命令行输入“GenerateSwarmExe”，即可如下图所示得到 Simulink 文件名对应的.exe 文件

## 6.3 实验示例

实验案例见 RflySimSwarm\0.ApiExps\3.EXEFileGener