ACFly EDU 代码框架手册

注意：

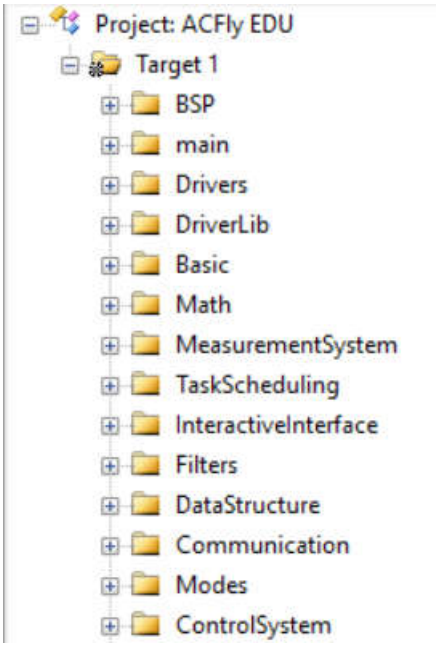
（1）请不要电机上电带桨进行程序烧录以免发生意外

（2）请尽量不要上电后用表笔直接戳板子进行测量，这个单片机很容易因此烧坏（这边已经因此烧了几个）

（3）此飞控源码只用作学习交流，不能用于商业用途！违者必究！！

一．代码总体布局

本飞控代码已经分组在 14 个大类里，分别是:。



建议看的部分：

1.3 驱动中的 Sensors 接口及 Receiver 接口

1.5 Basic 中的时间实现

1.7 解算系统中的解算系统接口

1.8 任务调度器接口

1.13 模式中的飞行模式

1.14 控制系统中的控制系统接口

1.1 BSP（板级支持包）

板级支持文件，最底层的库，**不用看基本不用修改（可以修改 startup.s 里面的堆栈设置）。**

1.2 Main（主函数文件）

主函数包括:

(1) 初始化所有需要用到的外设;

(2) 进入 STS 任务调度器（while 循环执行任务）

(3) 错误中断拉低所有输出

1.3 Driver（驱动）

驱动包含:

(1) 外设的初始化配置（drv_开头文件）

(2) 传感器接口，包括（**建议细看，二次开发必备**）:

(i) Sensors.c：传感器接口实现函数

(ii) Sensors.h：传感器读取接口函数声明，**建议细看**

(iii) Sensors_Backend.h：传感器注册、更新接口函数声明，**建议细看**

(3) 接收机接口，包括:

(i) Receiver.c：接收机接口实现函数

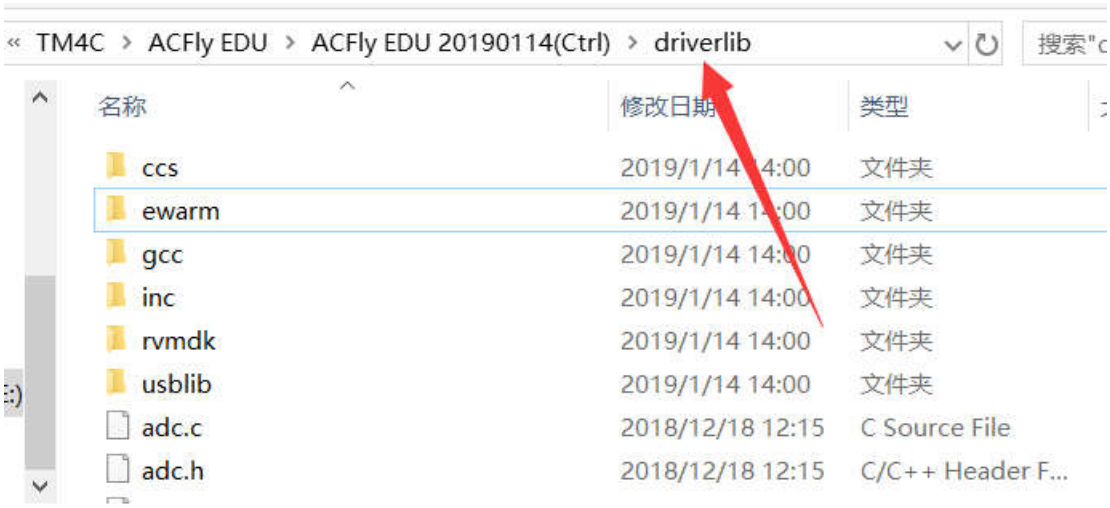
(ii) Receiver.h：接收机读取接口函数声明，**建议细看**

其中，drv_Sensors.c 里面执行 IMU 传感器读取操作并写入 Sensors 接口，然后通过挂起一个不用的中断的方式进入解算及控制中断。

```
598         }
599         //挂起解算控制任务中断
600         NVIC_SetPendingIRQ( I2C3_IRQn );
```

1.4 DriverLib（TI 的库）

有需要自己写驱动的，可以在下面目录下查看库函数的实现及说明:



1.5 Basic（基本）

Basic.c 里面初始化 systic 定时器用于计时，实现了 TIME 结构体用于时间计算，程序其他部分程序所有时间相关操作都是基于 TIME，**建议细看。**

Configurations.c 里有 EEPROM 的读取保存操作，用于保存记录参数等，可以不看。

1.6 Math（数学库）

包含四元数、三维向量运算，以及一些简单的数学运算，重力等常量的定义。

1.7 MeasurementSystem（解算系统）

姿态解算及位置解算。

建议细看解算系统接口 MeasurementSystem.h，包含解算结果的获取函数声明及使用说明。

1.8 TaskScheduling（任务调度器）

简易任务调度器，就是在主循环刷任务。

建议细看 STS.h，包含任务调度器的函数声明及使用说明。

1.9 InteractiveInterface（用户交互接口）

目前仅包含 LED 相关操作

1.10 Filters（滤波器）

包含巴特沃斯低通滤波器、TD4 非线性滤波器、位置估计卡尔曼滤波器的实现。

1.11 DataStructure（数据结构）

包含环形缓冲区的实现。

1.12 Communic（通讯）

包含 Mavlink 库、调试通讯文件 Debug.c、通用端口交互文件 Commulink.c（驱动程序可通过 Commulink.h 里的函数注册端口成为通用端口用于 mavlink 等标准通讯）

1.13 Modes（模式）

建议细看飞行模式！二次开发必备

0-9 号为非飞行非校准的其他模式

10-19 号为校准模式

30-39 号为飞行模式

M00 为初始化模式，等待解算系统初始化完成。然后进入 M01 地面模式。

M01 下可通过遥控或上位机命令进入其他校准及飞行模式。

1.14 ControlSystem（控制系统）

建议细看 ControlSystem.h！二次开发必备

ControlSystem.h 包含控制系统的 API 接口。

Ctrl_Attitude 和 Ctrl_Position 分别为姿态和位置控制器。

二．代码执行流程

2.1 初始化及任务调度

```
41  int main()
42  {
43    init_Basic();
44    init_drv_EEPROM();
45    init_MS();
46    init_ControlSystem();
47    init_Drivers();
48
49    init_Configurations();
50
51    init_Modes();
52    init_CommuLink();
53
54    init_Debug();
55    //while(1);
56    STS_Run();
57  }
```

Main 函数中首先调用 init 开头的函数进行初始化。

初始化函数中，会调用 STS 任务调度接口将需要在主循环里执行的函数加入到任务调度列表。

STS_Run 函数判断任务调度列表中的任务是否需要被执行，是就执行。

2.2 解算及控制任务

只有解算及控制是在中断中执行的。

在 drv_Sensors.c 文件中，每次获取到传感器数据后，会通过挂起一个不用的中断的方式进入解算及控制中断。

```
434   //开启一个不用的外设中断
435   //用于在中断中运行解算及控制任务
436   IntPrioritySet(INT_I2C3 , INT_PRIO_7);
437   I2CIntRegister( I2C3_BASE , MainMCHandler );
438   IntEnable(INT_I2C3);
```

```
598           }
599           //挂起解算控制任务中断
600           NVIC_SetPendingIRQ( I2C3_IRQn );
```

```
785   //主解算控制任务中断
786   static void MainMCHandler()
787   {
788     NVIC_ClearPendingIRQ( I2C3_IRQn );
789
790     MS_main();
791     static uint16_t Ctrl_Counter = 0;
792     if( ++Ctrl_Counter >= 5 )
793     {
794       Ctrl_Counter = 0;
795       //200hz运行控制
796       ctrl_main();
797     }
798   }
```

该中断的优先级是 INT_PRIO_7，**位置传感器更新中断优先级不要高于此优先级（也就是只能 INT_PRIO_7,，否则会导致线程问题）。**

2.3 模式的执行

模式是在任务调度器主循环中执行的。

Modes.c 里将模式任务加入到任务列表：

```
62    Mode_Task_ID = STS_Add_Task( STS_Task_Trigger_Mode_RoughTime ,
63    Modes[ current_mode ].mode_enter();
```

模式任务触发时，进入模式的主函数：

```
23  static void Modes_Server( unsigned int Task_ID )
24  {
25    Modes[ current_mode ].mode_main_func();
26  }
```