ACFly EDU 代码接口手册

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！！

本代码接口中所有位置、速度、加速度数据单位均为 cm 厘米

本代码接口中所有角度、角速度、角加速度数据单位均为 rad 弧度

本代码接口中所有磁场数据单位均为 Gauss 高斯

本飞控模块化编程，不同模块通过接口进行访问操作，此文档必看!!

此文档包括：传感器接口、接收机接口

一．传感器接口

位于 Drivers 目录下

传感器接口分为：传感器读取接口 和 传感器注册及更新接口。

函数声明分别位于：Sensors.h 和 Sensors_Backend.h 里

函数定义位于：Sensors.c 里

1.1 IMU 传感器定义（Sensors.h）

IMU 传感器包括加速度计、陀螺仪、磁力计，每种各支持 3 个。定义如下：

```
/*IMU传感器定义*/
  typedef struct
  {
    bool present; //传感器是否存在
    TIME last_update_time;  //上次更新时间
    float sample_time;  //采样时间

    float sensitivity;  //灵敏度（原始数据->实际单位 陀螺：rad/s 加速度：cm/s^2 磁场：gauss）

    vector3_int data_raw; //原始数据
    vector3_float data; //实际单位数据
  }IMU_Sensor;
/*IMU传感器定义*/
```

1.2 位置传感器定义（Sensors.h）

位置传感器包括气压、光流、超声波等，本飞控最多支持同时存在 8 个定位传感器。定义如下：

```
53  typedef struct
54  {
55    bool publishing;  //是否正在更新
56
57    bool present; //传感器是否存在
58    bool available; //传感器是否可用
59    TIME last_update_time;  //上次更新时间
60    TIME inavailable_start_time;  //传感器不可用开始时间
61    float delay;  //传感器延时
62    float sample_time;  //采样时间
63
64    bool safe;  //传感器是否安全（数据缓慢变化不会发生跳变 ！！注意！！如不确定不要设置为safe）
65    Position_Sensor_Type sensor_type; //传感器类型（见枚举注释）
66    Position_Sensor_DataType sensor_DataType; //传感器数据类型（见枚举注释）
67    Position_Sensor_frame velocity_data_frame;  //速度数据坐标系（见枚举注释）
68
69    vector3_double position_Global; //经纬度
70    vector3_float position; //位置(cm)
71    vector3_float velocity; //速度(cm/s)
72
73    //经纬度转平面坐标变量
74    Map_Projection mp;
75  }Position_Sensor;
```

其中：

sensor_type 定义了传感器是经纬度定位、相对定位，还是测距定位 传感器。

sensor_DataType 定义了传感器的数据类型：例如 z 轴位置数据，xy 速度数据等。

velocity_data_frame 针对速度传感器，定义了速度传感器所测速度所在的坐标系。

1.3 IMU 传感器读取接口（Sensors.h）

IMU 传感器读取接口会返回 const 的 IMU 传感器结构体：

```
84
85    /*IMU传感器读取函数*/
86      const IMU_Sensor* GetAccelerometer( unsigned char index );
87      const IMU_Sensor* GetGyroscope( unsigned char index );
88      const IMU_Sensor* GetMagnetometer( unsigned char index );
89    /*IMU传感器注册函数*/
```

1.4 IMU 传感器注册、更新接口（Sensors_Backend.h）

在 IMU 传感器更新前，首先调用注册函数进行注册，设置传感器的灵敏度。

注册完成后把 IMU 传感器编号及原始数据送入 update 接口即可完成更新。

```
11  /*IMU*/
12
13    /*IMU传感器注册函数*/
14      bool IMUAccelerometerRegister( unsigned char index , float sensitivity );
15      bool IMUGyroscopeRegister( unsigned char index , float sensitivity );
16      bool IMUMagnetometerRegister( unsigned char index , float sensitivity );
17    /*IMU传感器注册函数*/
18
19    /*IMU传感器更新函数*/
20      bool IMUAccelerometerUpdate( unsigned char index , vector3_int data );
21      bool IMUGyroscopeUpdate( unsigned char index , vector3_int data );
22      bool IMUMagnetometerUpdate( unsigned char index , vector3_int data );
23    /*IMU传感器更新函数*/
```

1.5 位置传感器读取接口（Sensors.h）

位置传感器读取接口会返回 const 的位置传感器结构体：

```
/*位置传感器*/

  /*位置传感器读取函数*/
    const Position_Sensor* GetPositionSensor( unsigned char index );
  /*位置传感器读取函数*/
```

1.6 位置传感器注册、更新接口（Sensors_Backend.h）

在位置传感器更新前，首先调用注册函数进行注册，设置传感器的类型等参数：

```
/*位置传感器注册函数*/
  //safe: 传感器是否安全（数据缓慢变化不会发生跳变 ！！注意！！如不确定不要设置为safe）
  bool PositionSensorRegister(
    unsigned char index ,\
    Position_Sensor_Type sensor_type ,\
    Position_Sensor_DataType sensor_data_type ,\
    Position_Sensor_frame sensor_vel_frame ,\
    float delay ,\
    bool safe \
  );
  //注销传感器
  bool PositionSensorUnRegister( unsigned char index );
/*位置传感器注册函数*/
```

如果位置传感器很久没有更新，MS_Main 解算任务中会自动把此传感器取消注册。

注册完成后把位置传感器编号及与传感器 sensor_data_type 对应的数据送入 update 接口即可完成更新：

```

  /*位置传感器更新函数*/
    //delay参数小于0则不会改变delay
    bool PositionSensorUpdatePositionGlobal( unsigned char index , vector3_double p
    bool PositionSensorUpdatePosition( unsigned char index , vector3_float position
    bool PositionSensorUpdatePositionGlobalVel( unsigned char index , vector3_doubl
    bool PositionSensorUpdatePositionVel( unsigned char index , vector3_float posit
    bool PositionSensorUpdateVel( unsigned char index , vector3_float vel , bool av
  /*IMU传感器更新函数*/

```

二．接收机接口

位于 Drivers 目录下

接收机接口分为：接收机读取接口 和 接收机更新接口。

函数声明分别位于：Receiver.h 和 Receiver _Backend.h 里

函数定义位于：Receiver.c 里

2.1 接收机定义（Receiver.h）

接收机可包含 SBUS、PPM 等协议的接收机（至少具有 6 个通道），定义如下：

```
6  //接收机定义
7  typedef struct
8  {
9    bool present; //是否存在
10   bool connected; //是否已连接
11   bool available; //是否可用
12   TIME last_update_time;  //上次更新时间
13   float update_time;  //更新时间间隔
14
15   float raw_data[16]; //原始数据
16   float data[8];  //校准后的数据
17 }Receiver;
```

2.2 接收机读取接口（Receiver.h）

接收机读取接口：

get_Receiver 会返回指定接收机的 const 结构体

get_current_Receiver 会返回当前接收机的 const 结构体（自动选择序号最低的可用接收机，无可用接收机是返回随机接收机）

get_current_Reciever_Type 返回当前接收机的类型（SBUS 接收机、PPM 接收机等）

```
26 //获取指定的接收机
27 const Receiver* get_Receiver( RC_Type rc );
28 //获取当前使用的接收机
29 const Receiver* get_current_Receiver();
30 //获取当前使用的接收机
31 RC_Type get_current_Receiver_Type();
```

2.3 接收机更新接口（Receiver_Backend.h）

把接收机类型、原始数据、是否已连接等信息发送给接口即可完成接收机数据更新。

```
8  //更新接收机数据
9  void Receiver_Update( RC_Type _rc , bool connected
```