* 芯片锁定与解锁
* 项目初始，应设置时钟，包括晶振大小、各个总线的时钟，时钟频率设置过大会导致芯片不工作甚至锁定
* 关于正点原子systick的驱动程序在不同芯片中并不能直接使用，还需要考虑芯片工作频率，自动重装载寄存器最大值，systick时钟源选择（/8时若工作频率不能整除会导致计时错误）
* 推挽模式与复用推挽模式区别：见GPT

# 双向电压电平转换器TXS0108QFN20

在pxhawk2.4.8中，在两串口之间使用了TXS0108QFN20，用于处理两个电压域之间的逻辑电平转换，其在A侧与B侧支持1.8V至5.5V之间的电压，从而保证了两侧USART设备的兼容性

# stm32F4的引脚复用功能：

在stm32f1系列中，配置引脚复用功能不需使用特殊的标准库函数，如配置USART时，TX引脚需要配置为复用推挽功能，此时初始化结构体能直接关联外设；但在F4系列单片机zhong，不仅初始化时需要配置引脚复用功能，还需调用函数GPIO\_PinAFConfig配置AFR寄存器

# stm32F4的SPI配置：

* 关于对SPIGPIO的配置：在f1参考手册中，对MISO的推荐配置是浮空输入模式。但实际应配置为复用模式，由外设控制输入模式，另外最好配置为无上下拉，应该对应浮空
* SPI工作于全双工模式，发送一个字节后必然会接收一个字节，在写发送接收函数时需注意，且接收了直接并非要读取的直接（如果是读取操作的话），需要再次发送一个无效数据才会返回要读取的字节。
* 在读取MPU6000数据时遇到无法读取角速度的情况，一开始是以为MPU6000初始化流程与MPU6050不一样，后来发现是因为在写读取mpu6000寄存器函数时，最后拉高片选引脚未做延时处理，导致在连续读寄存器数据时片选引脚未被正常拉高使SPI时序错误。
* 注意，对于大多数设备SPI通信时序，每次执行新的操作（发送新的命令或者读取新的寄存器），都需要重新拉低拉高CS引脚，有些设备支持批量读写

# 关于MPU6000

* 对于正点原子对MPU6500数据的处理，应该是先获取数据，在进行滤波并存入队列中，最后从队列中取数据。在正点原子，滤波器采样频率与MPU6500设定频率相同
* 关于对MPU6000读取数据，应该使用SPI6个数据连续读取（SPI时序发送一个地址后，地址连续递增读取数据）
* 在写MPU6000二阶低通滤波函数时，被困扰了好久，原因在直接II型二阶低通滤波器参数结构体中包含的两个延迟变量在处理每组测量数据时被混用，之后定义了6个参数结构体变量，分通道滤波解决了该问题。
* 关于对MPU6000的校准方法：包括六面校准法、重力参考法、简化的重力参考法（正点原子里面的方法，或者称为比例因子校准方法更好些）。六面校准法实现难度较大，在一个博客中说需要借助高精度转台测量数据。重力参考法的好处在于不必要求MPU6050是水平放置的，测量要求低，但需要通过LM算法求解，需要借助matlab，不知道怎么通过c语言实现
* 正点原子校准思路（比例因子校准）：

1. 处理陀螺仪偏置：建立环形缓冲区存储采样点，在未补偿偏置的情况下，开始处理传感器偏置（还需要保证环形缓冲区充满采样点）---计算偏置方差与平均值（方差小于阈值判断飞机处于静止状态）；同样，在静止状态下，计算加速度缩放因子---根据飞机期望三轴加速度平方和应为1g，通过计算采样点平均值获得缩放因子。

* 关于数组越界意外访问其他变量、有符号数到无符号数的转换、短整型变量赋给长整型变量

# 关于const关键字：

* 当不希望一个变量定义后发生变化时，可以用该关键字修饰。改变const修饰变量的值的方法是再次定义该变量。

# MS5611

1. **MS5611编程流程：**

* **初始化阶段**

**1 SPI、cs引脚**

**2 编写复位函数**

**3 读PROM，获取校准系数：**对PROM的读取命令应在复位后由执行一次，以读取校准PROM中的内容并计算校准系数。PROM共有8个地址，地址0存储出厂数据和配置信息，地址1至6存储校准系数，地址7包含序列代码和循环冗余校验（CRC），每个地址占16位，对应输出的16位结果按最高有效位（MSB）优先的时钟顺序传输。**（以上两步上电后要执行一次，可放入ms5611\_init函数中。**

* **ADC转换和数据读取**

**4 使能D1、D2转换与读取：**转换命令用于启动未补偿压力（D1）或未补偿温度（D2）的转换过程。在此期间可禁用片选信号，以便与其他设备进行通信。转换完成后，使用ADC读取命令（需要读取32位无符号数数据）可将结果按最高有效位（MSB）优先的时钟顺序输出。若在发送ADC读取命令前未执行转换，或重复发送ADC读取命令，输出结果将为0。若在转换过程中发送ADC读取命令，输出结果将为0，且转换不会中断，但最终结果将出现错误。在已启动的转换过程中发送转换指令序列同样会导致结果错误。

**5 压力与温度计算：**

1. MS5611的odr：过采样率；传感器内部的一种噪声抑制技术,通过多次采样，对采样结果进行平均提高数据分辨率与精度，越大，数据精度越高，但转换时间越长。根据规格书，OSR=4096时，压力分辨率可达**0.012毫巴**​（约10厘米垂直高度差）。OSR=256时，分辨率降至**0.065毫巴**​（约50厘米垂直高度差）
2. ms5611SPI时序要求每次开始新的命令时必须重新拉低拉高CS
3. 在正点原子飞行器例程中，编写了限幅平均滤波方法函数，该方法可以处理脉冲噪声以及高频噪声，但并未使用该函数处理bmp280（另一种气压计），deepseek中说，最好对数据进行滤波处理，以后可以试一下
4. 下一步要把气压值转海拔函数写了，包括绝对高度、相对高度，好像还要写另一个气压计驱动，之后写磁力计驱动
5. 查海平面基准气压，公式中的P0为参考点的气压，如果是绝对高度则是当地海平面标准气压，如果是相对高度这是参考位置的气压
6. 关于正点原子对相对高度的计算（这里仅分析起飞初始），是先将第一次计算的初始高度作为基准高度，下一次将测量的绝对高度减去基准高度得到绝对高度。我可以使用两种方法分别计算相对高度，一种是将海平面标准大气压换为地面测量时测得的气压。另一种是先测一个地方的气压，将其作用基准高度，减去其他地方高度测得相对高度
7. 关于在进行温度补偿时定义的变量都是整数的问题，在补偿温度公式中，除的都是2的n次方，本质上是移位操作，所以定义为整数精度更高

# 遥控器与SBUS协议

1. 关于ACROSS的程序思路与知识点：

* 编程思路：配置USART、DMA，使能USART空闲中断检测帧结束，使用DMA转移数据到缓冲区，定义一个双双缓冲区，一个用于接收，一个用于解码；在空闲中断中切换缓冲区
* 解码矩阵作用：

1.首先，理解SBUS可以传输16个比例通道和2个数字通道，其帧格式为帧头(起始位)+8位数据位\*22(即16\*11，16为通道数，11为通道数据)+标志位+结束位，共25位，而解码矩阵的SBUS\_INPUT\_CHANNELS=16便代表的通道。

2.以二维解码矩阵第一行数据为例：

首先，定义了下面的结构体，也即解码矩阵每个{}中数字的含义：  
struct sbus\_bit\_pick

{

uint8\_t byte;// 数据在帧中的字节偏移（从第1字节开始，即frame[0]对应byte=0，frame定义在下面解码函数解释处）

uint8\_t rshift; //右移位数：将目标位段移动到低位

uint8\_t mask;// 位掩码：过滤无关位，保留目标位段

uint8\_t lshift;// 左移位数：将提取的位段移动到正确位置，以便与其他部分组合

};

例：

{ { 0, 0, 0xff, 0}, { 1, 0, 0x07, 8}, { 0, 0, 0x00, 0} }/\* 0 \*/,

第1部分：byte=0（frame[0]），rshift=0：不右移。mask=0xff：保留全部8位。lshift=0：不移位。

结果：frame[1]的完整8位 → value = 0bXXXXXXXX.

第2部分：byte=1（frame[2]）rshift=0：不右移。mask=0x07（二进制00000111）：保留低3位。lshift=8：左移8位。

结果：(frame[2] & 0x07) << 8 → 0b00000YYY << 8 = 0bYYY00000000.

合并：value = 0bXXXXXXXX | 0bYYY00000000 = 0bYYYXXXXXXXX（11位）.

* 解码函数：SBUS\_Decode():

1.首先：switch (frame[24])程序段，frame为SBUS接收到的一整个数据帧，switch段程序通过检查帧尾数据，判断协议版本或扩展数据类型，为后续处理提供分支依据（在ACROSS程序中并没有实现处理不同协议版本的程序，只是列出了分支）。case 0x00: SBUS1标准帧，帧尾为00.

2.max\_values：需要处理的通道，不能大于SBUS协议的最大通道数（16）

3.遍历解码矩阵，解码数据，从SBUS转化为PWM输出公式：  


4. if (chancount > 15)代码段：数字开关通道解码，当请求的通道数大于15时，解析SBUS帧中的两个数字开关通道（通道17和18），并将其转换为开关值（标准PWM协议PWM范围为：1000us~2000us）。

5. 处理故障保护和丢帧：很奇怪的是，在进行仿真时，程序运行一段时间就会停在丢帧程序段，不知道是程序的原因还是接收机的原因

# PWM

* 对于定时器的配置相关:在PIX4中，输出400Hz的PWM信号。在我们的飞控程序中，控制了PWM的范围为1000us-2000us（PWM周期内高电平时间），尽管并不能达到满占空比，但对电机的控制是通过电调识别占空比的范围，1000us对于就是0输出，2000us对于就是满转速
* 关于TIM时钟，在stm32f103c8中，如果APB1的预分频系数为1，则TIM时钟=APB1时钟，如果APB1的预分频系数>1，则TIM时钟=2\*APB1时钟,注意，并不总= APB1时钟
* 对于新西达电调的校准：在电调不通电的情况下，将遥控器拉到最高（2000us），通电电调，将遥控器拉低。还要注意对PWM的配置，需要配置一个周期高电平在前面，低电平在后面，关于电调的校准和设定（电调可以设定模式），具体可参考：https://blog.csdn.net/hehu8/article/details/51338644

# M8N/UBX协议

## 数据手册

* m8n只有在配置为使用GPS运行时，才可使用星基增强系统（SBAS）准天线卫星系统（QZSS）增强系统才能启用
* SPI 的最大传输速率为 125 kB/s，最大 SPI 时钟频率为 5.5 MHz
* 在UBX协议中，向接收机发送一些需要确定的命令（如配置类命令UBX \_CFG\_），接收机会返回**UBX 协议中命令确认（ACK/NACK）**

编程技巧

* 当两个数二进制位不包含重合的1时，则二者的按位或与数学+等价，例：(a<<16)|((uint16\_t)b),等价于a+b
* 状态标志位最好定义为volatile
* 状态机：状态机适合处理来自串口或DMA的流式数据（如GPS模块的连续输出），无需等待完整数据包即可逐步解析，不需要缓存完整数据包，适合资源受限的嵌入式系统。在解析过程中发现错误（如校验失败、无效长度）时，可立即重置状态机，避免错误扩散