目录

[一、 双向电压电平转换器TXS0108QFN20 2](#_Toc196309696)

[二、 stm32F4的引脚复用功能： 2](#_Toc196309697)

[三、 stm32F4的SPI配置： 2](#_Toc196309698)

[四、 关于MPU6000 3](#_Toc196309699)

[五、 关于const关键字： 3](#_Toc196309700)

[六、 MS5611 4](#_Toc196309701)

[七、 遥控器与SBUS协议 5](#_Toc196309702)

[八、 PWM 6](#_Toc196309703)

[九、 GPS/M8N/UBX/RTCM协议 7](#_Toc196309704)

[1. 数据手册 7](#_Toc196309705)

[2. UBX协议 7](#_Toc196309706)

[3. RTCM协议： 22](#_Toc196309707)

[十、 环形缓冲区： 23](#_Toc196309708)

[编程相关 24](#_Toc196309709)

[十一、 DMA（F4） 24](#_Toc196309710)

[十二、 SysTick 25](#_Toc196309711)

[十三、 USART 25](#_Toc196309712)

[十四、 编程技巧 25](#_Toc196309713)

* 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

# GPS/M8N/UBX/RTCM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）
* 在Across代码中，将环形缓冲区与GPS缓冲区进行了关联，在接收完成（转移完成）中断中将接受的单个字节数据压入了环形缓冲区。如果我在代码中使能了双缓冲区和空闲中断，那么应该在中断中怎么处理程序？我应该在中断中将数据慢慢压入？还是不用压，而是仅仅增加指针？
* NAV-PVT消息是u-blox及以上版本模块引入的新一代导航消息，提供更丰富的定位信息（如时间、位置、速度等）；NAV-PVT消息有些模块不支持，故在代码中通过\_use\_nav\_pvt标记设备是否支持；旧版模块不支持该消息，仅能通过旧消息组合（如NAV-SOL+NAV-POSLLH+NAV-TIMEUTC）获取消息
* configure函数编写思路：
* RTCM协议：一种用于差分GNSS的通信协议，用于传输GNSS参考站的差分修正数据，帮助GNSS接收机提升定位精度

## UBX协议

* UBX协议数据类型命名规范：

1. U：无符号整数
2. I： 有符号整数
3. X： 位字段（Bitfield,通常为无符号数，特定位有特殊含义）
4. R： 浮点数（Real number）
5. CH：字符
6. 数字后缀表示字节数

在所有多字节字段中（如U2、I4）在UBX协议中采用小端序，即低字节在前，高字节在后

* UBX协议的校验和：校验和需要计算从Class开始，至Payload结束，包括Class、ID、Length、Payload。计算步骤为：设置CK\_A、CK\_B为0，遍历数据字节，更新CK\_A=CK\_A+当前字节，更新CK\_B=CK\_B+CK\_A;计算完成后，为两个校验和字节
* **UBX-NAV-PVT消息**:

`UBX-NAV-PVT`（Navigation Position Velocity Time Solution）是u-blox GNSS模块的UBX协议中一个核心消息，用于提供完整的导航解，包括位置（经纬度、高度）、速度、时间、定位状态等信息。它广泛应用于标准定位、差分GNSS（DGNSS）、实时动态定位（RTK）等场景。以下是对`UBX-NAV-PVT`消息的详细解释，涵盖消息结构、字段定义、功能、应用场景，以及与u-blox 7和u-blox 8/M8模块的关联（结合您的问题背景）。

---

### 1. \*\*消息概述\*\*

- \*\*消息类（Class）\*\*：`0x01`（NAV，导航类）。

- \*\*消息ID\*\*：`0x07`（PVT）。

- \*\*作用\*\*：提供一次导航解的综合信息，包括：

- 位置：经纬度、海拔高度、海平面高度。

- 速度：三维速度（北、东、向下）、地面速度、运动方向。

- 时间：GPS时间、UTC时间、时间精度。

- 定位状态：定位类型（无解、2D、3D、DGNSS、RTK）、卫星数量、精度估计。

- 其他：DOP（几何精度因子）、标志位（如差分状态、RTK状态）。

- \*\*适用模块\*\*：u-blox 5、6、7、8/M8、9等系列，支持标准定位、DGNSS和RTK（u-blox 8/M8及以上支持RTK）。

- \*\*消息长度\*\*：固定92字节（消息体），不随固件版本变化。

\*\*与您的问题背景\*\*：

- 您询问了UBX-7和UBX-8的`UBX-NAV-PVT`差异，重点在于RTK支持、多星座和精度。

- 您提供的`UBX-CFG-PRT`配置代码表明应用场景可能涉及RTK基站或漫游站，`UBX-NAV-PVT`是输出定位结果的关键消息。

---

### 2. \*\*消息结构\*\*

`UBX-NAV-PVT`遵循UBX协议的通用格式：

- \*\*头部\*\*：

- 同步字符：`0xB5 0x62`（2字节）。

- 类：`0x01`（NAV，1字节）。

- ID：`0x07`（PVT，1字节）。

- 长度：`0x5C 0x00`（92字节，2字节，小端序）。

- \*\*消息体\*\*：92字节，包含导航数据。

- \*\*校验和\*\*：2字节（CK\_A, CK\_B），基于类、ID、长度和消息体的CRC校验。

\*\*总长度\*\*：100字节（头部4字节 + 长度2字节 + 消息体92字节 + 校验和2字节）。

\*\*消息格式\*\*：

```

+--------+--------+--------+--------+--------+--------+--------+----------+--------+

| 0xB5 | 0x62 | 0x01 | 0x07 | 0x5C | 0x00 | 消息体 | CK\_A | CK\_B |

| 同步1 | 同步2 | 类 | ID | 长度低 | 长度高 | 92字节 | 校验1 | 校验2 |

+--------+--------+--------+--------+--------+--------+--------+----------+--------+

```

---

### 3. \*\*消息体字段详解\*\*

`UBX-NAV-PVT`消息体的92字节包含多个字段，涵盖位置、速度、时间和状态信息。以下是每个字段的详细定义、类型、单位、用途，以及在u-blox 7和u-blox 8/M8中的差异：

| **偏移（字节）** | **字段名** | **类型** | **单位** | **描述** | **UBX-7特性** | **UBX-8特性** |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | iTOW | U4 | ms | GPS周内时间（Time of Week），自周日00:00起算的毫秒数 | 支持GPS/GLONASS时间系统 | 扩展支持Galileo/BeiDou，RTK模式更精确 |
| 4 | year | U2 | 年 | UTC年（如2023） | 标准UTC时间 | 同UBX-7，RTK模式时间更可靠 |
| 6 | month | U1 | 月 | UTC月（1-12） | 同UBX-8 | 同UBX-7 |
| 7 | day | U1 | 日 | UTC日（1-31） | 同UBX-8 | 同UBX-7 |
| 8 | hour | U1 | 小时 | UTC小时（0-23） | 同UBX-8 | 同UBX-7 |
| 9 | min | U1 | 分钟 | UTC分钟（0-59） | 同UBX-8 | 同UBX-7 |
| 10 | sec | U1 | 秒 | UTC秒（0-60，含闰秒） | 同UBX-8 | 同UBX-7 |
| 11 | valid | X1 | - | 有效性标志：<br>- 位0：validDate（日期有效）<br>- 位1：validTime（时间有效）<br>- 位2：fullyResolved（时间完全解析）<br>- 位3：validMag（磁偏角有效） | 基本支持（日期、时间有效性） | 扩展支持（如RTK时间验证更严格） |
| 12 | tAcc | U4 | ns | 时间精度估计（纳秒） | ~100 ns（典型） | <100 ns（RTK模式下更高精度） |
| 16 | nano | I4 | ns | 秒的小数部分（-1e9到1e9），用于亚秒级精度 | 支持亚秒精度 | 更高精度（多星座和RTK支持） |
| 20 | fixType | U1 | - | 定位类型：<br>- 0：无解<br>- 1：2D解<br>- 2：3D解<br>- 3：DGNSS<br>- 4：RTK固定<br>- 5：RTK浮动<br>- 6：推算（Dead Reckoning） | 0-3（无RTK） | 0-5（支持RTK固定/浮动） |
| 21 | flags | X1 | - | 定位标志：<br>- 位0：gnssFixOK（定位有效）<br>- 位1：diffSoln（差分解）<br>- 位5：psmState（省电模式状态）<br>- 位6-7：carrSoln（载波相位状态）：<br>  - 0：无<br>  - 1：浮动（DGNSS）<br>  - 2：RTK浮动<br>  - 3：RTK固定 | 基本标志（carrSoln=0/1） | 扩展标志（carrSoln=2/3支持RTK） |
| 22 | flags2 | X1 | - | 附加标志：<br>- 位0-1：保留<br>- 位5-7：confirmedAvai（时间确认状态） | 有限支持（基本标志） | 扩展支持（RTK确认、省电模式） |
| 23 | numSV | U1 | - | 使用的卫星数（所有星座） | ~20颗（GPS、GLONASS、SBAS） | ~30颗（新增Galileo、BeiDou） |
| 24 | lon | I4 | deg \* 1e-7 | 经度（分辨率0.0000001度） | 精度~2.5 m（标准），~0.5 m（DGNSS） | 精度~2.0 m（标准），~1-2 cm（RTK） |
| 28 | lat | I4 | deg \* 1e-7 | 纬度（分辨率0.0000001度） | 同上 | 同上 |
| 32 | height | I4 | mm | 海拔高度（相对于WGS84椭球） | 精度~4 m | 精度~3 m（标准），更高（RTK） |
| 36 | hMSL | I4 | mm | 海平面高度（相对于大地水准面） | 精度~4 m | 精度~3 m（标准），更高（RTK） |
| 40 | hAcc | U4 | mm | 水平精度估计（半径，68%置信度） | ~2500 mm（标准），~500 mm（DGNSS） | ~2000 mm（标准），~10-20 mm（RTK） |
| 44 | vAcc | U4 | mm | 垂直精度估计（68%置信度） | ~4000 mm（标准），~1000 mm（DGNSS） | ~3000 mm（标准），~20-40 mm（RTK） |
| 48 | velN | I4 | mm/s | 北向速度 | 标准精度 | RTK模式下更稳定 |
| 52 | velE | I4 | mm/s | 东向速度 | 同上 | 同上 |
| 56 | velD | I4 | mm/s | 向下速度 | 同上 | 同上 |
| 60 | gSpeed | I4 | mm/s | 地面速度（2D速度） | 同UBX-8 | 同UBX-7 |
| 64 | headMot | I4 | deg \* 1e-5 | 运动方向（相对于真北） | 同UBX-8 | 同UBX-7 |
| 68 | sAcc | U4 | mm/s | 速度精度估计 | 标准精度 | RTK模式下更精确 |
| 72 | headAcc | U4 | deg \* 1e-5 | 方向精度估计 | 同UBX-8 | 同UBX-7，RTK模式下更精确 |
| 76 | pDOP | U2 | 0.01 | 位置DOP（几何精度因子，典型值1.0-5.0） | 同UBX-8 | 同UBX-7 |
| 78 | flags3 | X1 | - | 附加标志（如多频状态） | 基本未使用 | 支持多频标志（M8双频模块） |
| 79 | reserved1 | U1[5] | - | 保留字段（未来扩展） | 同UBX-8 | 同UBX-7 |
| 84 | headVeh | I4 | deg \* 1e-5 | 车辆方向（需外部传感器支持） | 部分支持（特定模块） | 更广泛支持（M8复杂应用） |
| 88 | magDec | I2 | deg \* 1e-2 | 磁偏角（需外部磁力计） | 可选支持 | 同UBX-7 |
| 90 | magAcc | U2 | deg \* 1e-2 | 磁偏角精度 | 可选支持 | 同UBX-7 |

| 偏移（字节） | 字段名 | 类型 | 单位 | 描述 | UBX-7特性 | UBX-8特性 |

|--------------|-----------------|------|---------------|----------------------------------------------------------------------|--------------------------------------------|--------------------------------------------|

| 0 | `iTOW` | U4 | ms | GPS周内时间（Time of Week），自周日00:00起算的毫秒数 | 支持GPS/GLONASS时间系统 | 扩展支持Galileo/BeiDou，RTK模式更精确 |

| 4 | `year` | U2 | 年 | UTC年（如2023） | 标准UTC时间 | 同UBX-7，RTK模式时间更可靠 |

| 6 | `month` | U1 | 月 | UTC月（1-12） | 同UBX-8 | 同UBX-7 |

| 7 | `day` | U1 | 日 | UTC日（1-31） | 同UBX-8 | 同UBX-7 |

| 8 | `hour` | U1 | 小时 | UTC小时（0-23） | 同UBX-8 | 同UBX-7 |

| 9 | `min` | U1 | 分钟 | UTC分钟（0-59） | 同UBX-8 | 同UBX-7 |

| 10 | `sec` | U1 | 秒 | UTC秒（0-60，含闰秒） | 同UBX-8 | 同UBX-7 |

| 11 | `valid` | X1 | - | 有效性标志：<br>- 位0：`validDate`（日期有效）<br>- 位1：`validTime`（时间有效）<br>- 位2：`fullyResolved`（时间完全解析）<br>- 位3：`validMag`（磁偏角有效） | 基本支持（日期、时间有效性） | 扩展支持（如RTK时间验证更严格） |

| 12 | `tAcc` | U4 | ns | 时间精度估计（纳秒） | ~100 ns（典型） | <100 ns（RTK模式下更高精度） |

| 16 | `nano` | I4 | ns | 秒的小数部分（-1e9到1e9），用于亚秒级精度 | 支持亚秒精度 | 更高精度（多星座和RTK支持） |

| 20 | `fixType` | U1 | - | 定位类型：<br>- 0：无解<br>- 1：2D解<br>- 2：3D解<br>- 3：DGNSS<br>- 4：RTK固定<br>- 5：RTK浮动<br>- 6：推算（Dead Reckoning） | 0-3（无RTK） | 0-5（支持RTK固定/浮动） |

| 21 | `flags` | X1 | - | 定位标志：<br>- 位0：`gnssFixOK`（定位有效）<br>- 位1：`diffSoln`（差分解）<br>- 位5：`psmState`（省电模式状态）<br>- 位6-7：`carrSoln`（载波相位状态）：<br>&nbsp;&nbsp;- 0：无<br>&nbsp;&nbsp;- 1：浮动（DGNSS）<br>&nbsp;&nbsp;- 2：RTK浮动<br>&nbsp;&nbsp;- 3：RTK固定 | 基本标志（`carrSoln`=0/1） | 扩展标志（`carrSoln`=2/3支持RTK） |

| 22 | `flags2` | X1 | - | 附加标志：<br>- 位0-1：保留<br>- 位5-7：`confirmedAvai`（时间确认状态） | 有限支持（基本标志） | 扩展支持（RTK确认、省电模式） |

| 23 | `numSV` | U1 | - | 使用的卫星数（所有星座） | ~20颗（GPS、GLONASS、SBAS） | ~30颗（新增Galileo、BeiDou） |

| 24 | `lon` | I4 | deg \* 1e-7 | 经度（分辨率0.0000001度） | 精度~2.5 m（标准），~0.5 m（DGNSS） | 精度~2.0 m（标准），~1-2 cm（RTK） |

| 28 | `lat` | I4 | deg \* 1e-7 | 纬度（分辨率0.0000001度） | 同上 | 同上 |

| 32 | `height` | I4 | mm | 海拔高度（相对于WGS84椭球） | 精度~4 m | 精度~3 m（标准），更高（RTK） |

| 36 | `hMSL` | I4 | mm | 海平面高度（相对于大地水准面） | 精度~4 m | 精度~3 m（标准），更高（RTK） |

| 40 | `hAcc` | U4 | mm | 水平精度估计（半径，68%置信度） | ~2500 mm（标准），~500 mm（DGNSS） | ~2000 mm（标准），~10-20 mm（RTK） |

| 44 | `vAcc` | U4 | mm | 垂直精度估计（68%置信度） | ~4000 mm（标准），~1000 mm（DGNSS） | ~3000 mm（标准），~20-40 mm（RTK） |

| 48 | `velN` | I4 | mm/s | 北向速度 | 标准精度 | RTK模式下更稳定 |

| 52 | `velE` | I4 | mm/s | 东向速度 | 同上 | 同上 |

| 56 | `velD` | I4 | mm/s | 向下速度 | 同上 | 同上 |

| 60 | `gSpeed` | I4 | mm/s | 地面速度（2D速度） | 同UBX-8 | 同UBX-7 |

| 64 | `headMot` | I4 | deg \* 1e-5 | 运动方向（相对于真北） | 同UBX-8 | 同UBX-7 |

| 68 | `sAcc` | U4 | mm/s | 速度精度估计 | 标准精度 | RTK模式下更精确 |

| 72 | `headAcc` | U4 | deg \* 1e-5 | 方向精度估计 | 同UBX-8 | 同UBX-7，RTK模式下更精确 |

| 76 | `pDOP` | U2 | 0.01 | 位置DOP（几何精度因子，典型值1.0-5.0） | 同UBX-8 | 同UBX-7 |

| 78 | `flags3` | X1 | - | 附加标志（如多频状态） | 基本未使用 | 支持多频标志（M8双频模块） |

| 79 | `reserved1` | U1[5]| - | 保留字段（未来扩展） | 同UBX-8 | 同UBX-7 |

| 84 | `headVeh` | I4 | deg \* 1e-5 | 车辆方向（需外部传感器支持） | 部分支持（特定模块） | 更广泛支持（M8复杂应用） |

| 88 | `magDec` | I2 | deg \* 1e-2 | 磁偏角（需外部磁力计） | 可选支持 | 同UBX-7 |

| 90 | `magAcc` | U2 | deg \* 1e-2 | 磁偏角精度 | 可选支持 | 同UBX-7 |

---

### 4. \*\*字段功能详解\*\*

以下是对关键字段的深入解释，突出其用途和在不同场景中的意义：

#### \*\*(1) 时间字段（`iTOW`, `year`, `month`, `day`, `hour`, `min`, `sec`, `nano`, `tAcc`, `valid`）\*\*

- \*\*用途\*\*：

- `iTOW`：GNSS时间（毫秒级），用于同步导航解。

- `year`至`sec`：UTC时间，便于用户读取。

- `nano`：亚秒级精度（纳秒），支持高精度时间应用。

- `tAcc`：时间精度估计，反映时间解的质量。

- `valid`：指示时间和日期是否可靠（例如，`validDate=1`表示日期有效）。

- \*\*UBX-7 vs UBX-8\*\*：

- UBX-7：基于GPS/GLONASS时间，`tAcc`~100 ns。

- UBX-8：支持Galileo/BeiDou，RTK模式下`tAcc`<100 ns，时间更鲁棒。

#### \*\*(2) 定位类型（`fixType`）\*\*

- \*\*值\*\*：

- 0：无解（无有效定位）。

- 1：2D解（仅经纬度）。

- 2：3D解（经纬高）。

- 3：DGNSS（差分定位，亚米级）。

- 4：RTK固定（厘米级，整周模糊度解算）。

- 5：RTK浮动（亚厘米级，未完全解算模糊度）。

- 6：推算（Dead Reckoning，依赖惯性导航）。

- \*\*UBX-7\*\*：仅支持0-3，无RTK。

- \*\*UBX-8\*\*：支持0-5，RTK固定/浮动是高精度应用的核心。

- \*\*用途\*\*：判断定位质量，RTK固定（4）是最高精度。

#### \*\*(3) 定位标志（`flags`, `flags2`, `flags3`）\*\*

- \*\*flags\*\*：

- `gnssFixOK`：定位解有效。

- `diffSoln`：使用差分数据（DGNSS或RTK）。

- `carrSoln`：载波相位状态（0=无，1=DGNSS浮动，2=RTK浮动，3=RTK固定）。

- `psmState`：省电模式状态（UBX-8更详细）。

- \*\*flags2\*\*：

- 时间确认状态（UBX-8支持RTK时间验证）。

- \*\*flags3\*\*：

- UBX-8支持多频标志（如L1+L2），UBX-7基本未使用。

- \*\*用途\*\*：提供定位状态的详细信息，RTK应用需检查`carrSoln`。

#### \*\*(4) 位置字段（`lon`, `lat`, `height`, `hMSL`, `hAcc`, `vAcc`）\*\*

- \*\*用途\*\*：

- `lon`, `lat`：经纬度，分辨率0.0000001度。

- `height`：WGS84椭球高度，毫米级分辨率。

- `hMSL`：海平面高度，反映实际地形。

- `hAcc`, `vAcc`：水平和垂直精度估计（68%置信度）。

- \*\*UBX-7\*\*：

- 精度：~2.5 m（水平），~4 m（垂直）。

- DGNSS：~0.5 m（水平），~1 m（垂直）。

- \*\*UBX-8\*\*：

- 标准：~2.0 m（水平），~3 m（垂直）。

- RTK：~10-20 mm（水平），~20-40 mm（垂直）。

- \*\*用途\*\*：RTK模式下提供厘米级坐标，适合测绘、自动驾驶。

#### \*\*(5) 速度字段（`velN`, `velE`, `velD`, `gSpeed`, `headMot`, `sAcc`, `headAcc`）\*\*

- \*\*用途\*\*：

- `velN`, `velE`, `velD`：三维速度（毫米/秒）。

- `gSpeed`：地面速度（2D）。

- `headMot`：运动方向（相对于真北）。

- `sAcc`, `headAcc`：速度和方向精度。

- \*\*UBX-7 vs UBX-8\*\*：

- UBX-8在RTK模式下速度更稳定，`sAcc`更精确。

- 功能一致，但UBX-8受益于多星座。

#### \*\*(6) 其他字段\*\*

- \*\*`numSV`\*\*：使用的卫星数，UBX-8支持更多卫星（~30 vs ~20）。

- \*\*`pDOP`\*\*：几何精度因子，反映卫星分布质量。

- \*\*`headVeh`\*\*：车辆方向（需外部传感器，UBX-8支持更复杂场景）。

- \*\*`magDec`, `magAcc`\*\*：磁偏角（可选，需磁力计）。

- \*\*`reserved1`\*\*：保留，未来扩展。

---

### 5. \*\*功能与应用\*\*

`UBX-NAV-PVT`消息是u-blox模块输出的核心导航数据，适用于以下场景：

- \*\*标准定位\*\*：

- 应用：车载导航、无人机、资产追踪。

- 精度：米级（UBX-7：~2.5 m，UBX-8：~2.0 m）。

- 关键字段：`lon`, `lat`, `height`, `fixType=2/3`.

- \*\*DGNSS\*\*：

- 应用：海洋导航、精准定位。

- 精度：亚米级（~0.5 m）。

- 关键字段：`fixType=3`, `diffSoln`, `hAcc`, `vAcc`.

- \*\*RTK\*\*（仅UBX-8）：

- 应用：测绘、精准农业、自动驾驶。

- 精度：厘米级（1-2 cm）。

- 关键字段：`fixType=4/5`, `carrSoln`, `hAcc`（~10 mm）.

- \*\*时间同步\*\*：

- 应用：电信、科研。

- 关键字段：`iTOW`, `nano`, `tAcc`, `valid`.

- \*\*调试与监控\*\*：

- 使用u-center查看`numSV`, `pDOP`, `flags`，评估定位性能。

\*\*与您的代码关联\*\*：

- 您的`UBX-CFG-PRT`配置：

```c

#define UBX\_TX\_CFG\_PRT\_OUTPROTOMASK\_GPS (0x01) /\*\*< UBX out \*/

#define UBX\_TX\_CFG\_PRT\_OUTPROTOMASK\_RTCM ((1<<5) | 0x01) /\*\*< RTCM3 out and UBX out \*/

```

- \*\*漫游站模式\*\*（`OUTPROTOMASK\_GPS`）：

- 输出`UBX-NAV-PVT`，提供定位结果。

- UBX-8：支持RTK（`fixType=4/5`），厘米级精度。

- UBX-7：仅支持DGNSS（`fixType=3`），亚米级。

- \*\*基站模式\*\*（`OUTPROTOMASK\_RTCM`）：

- 输出RTCM3和`UBX-NAV-PVT`，后者用于监控基站状态。

- UBX-8：支持RTK基站，`NAV-PVT`反映基站定位质量。

- UBX-7：不支持RTCM3输出，`NAV-PVT`作用有限。

---

### 6. \*\*UBX-7 vs UBX-8的差异\*\*

结合您的问题，以下是`UBX-NAV-PVT`在UBX-7和UBX-8中的具体差异：

- \*\*定位类型\*\*：

- UBX-7：`fixType=0-3`，无RTK支持。

- UBX-8：`fixType=0-5`，支持RTK固定/浮动（M8P、M8T）。

- \*\*精度\*\*：

- UBX-7：水平~2.5 m，垂直~4 m，DGNSS~0.5 m。

- UBX-8：水平~2.0 m，垂直~3 m，RTK~1-2 cm。

- \*\*GNSS支持\*\*：

- UBX-7：GPS、GLONASS、SBAS，~20颗卫星。

- UBX-8：新增Galileo、BeiDou，~30颗卫星。

- \*\*标志字段\*\*：

- UBX-7：`flags`支持DGNSS（`carrSoln=0/1`），`flags2`简单。

- UBX-8：`flags`支持RTK（`carrSoln=2/3`），`flags2`/`flags3`扩展。

- \*\*时间精度\*\*：

- UBX-7：`tAcc`~100 ns。

- UBX-8：`tAcc`<100 ns（RTK更精确）。

- \*\*应用\*\*：

- UBX-7：标准导航、DGNSS。

- UBX-8：RTK、复杂环境导航、后处理。

\*\*代码影响\*\*：

- 您的RTK配置（`INPROTOMASK\_GPS`, `OUTPROTOMASK\_RTCM`）需要UBX-8模块（如NEO-M8P），因为：

- UBX-7的`NAV-PVT`不支持RTK（`fixType`无4/5）。

- UBX-7不支持RTCM3输出（`OUTPROTOMASK\_RTCM`无效）。

---

### 7. \*\*解析与实现示例\*\*

以下是一个C语言示例，用于解析`UBX-NAV-PVT`消息，提取关键字段：

```c

#include <stdio.h>

#include <stdint.h>

#pragma pack(push, 1)

typedef struct {

uint32\_t iTOW; // GPS时间（ms）

uint16\_t year; // UTC年

uint8\_t month; // UTC月

uint8\_t day; // UTC日

uint8\_t hour; // UTC小时

uint8\_t min; // UTC分钟

uint8\_t sec; // UTC秒

uint8\_t valid; // 有效性标志

uint32\_t tAcc; // 时间精度（ns）

int32\_t nano; // 秒小数部分（ns）

uint8\_t fixType; // 定位类型

uint8\_t flags; // 定位标志

uint8\_t flags2; // 附加标志

uint8\_t numSV; // 卫星数

int32\_t lon; // 经度（deg \* 1e-7）

int32\_t lat; // 纬度（deg \* 1e-7）

int32\_t height; // 海拔高度（mm）

int32\_t hMSL; // 海平面高度（mm）

uint32\_t hAcc; // 水平精度（mm）

uint32\_t vAcc; // 垂直精度（mm）

int32\_t velN; // 北向速度（mm/s）

int32\_t velE; // 东向速度（mm/s）

int32\_t velD; // 向下速度（mm/s）

int32\_t gSpeed; // 地面速度（mm/s）

int32\_t headMot; // 运动方向（deg \* 1e-5）

uint32\_t sAcc; // 速度精度（mm/s）

uint32\_t headAcc; // 方向精度（deg \* 1e-5）

uint16\_t pDOP; // 位置DOP（0.01）

uint8\_t flags3; // 附加标志

uint8\_t reserved1[5]; // 保留

int32\_t headVeh; // 车辆方向（deg \* 1e-5）

int16\_t magDec; // 磁偏角（deg \* 1e-2）

uint16\_t magAcc; // 磁偏角精度（deg \* 1e-2）

} ubx\_nav\_pvt\_t;

#pragma pack(pop)

void parse\_ubx\_nav\_pvt(uint8\_t \*buf, ubx\_nav\_pvt\_t \*pvt) {

memcpy(pvt, buf, sizeof(ubx\_nav\_pvt\_t));

printf("iTOW: %u ms\n", pvt->iTOW);

printf("UTC: %04u-%02u-%02u %02u:%02u:%02u\n", pvt->year, pvt->month, pvt->day, pvt->hour, pvt->min, pvt->sec);

printf("Fix Type: %u (%s)\n", pvt->fixType, pvt->fixType == 4 ? "RTK Fixed" : pvt->fixType == 5 ? "RTK Float" : "Other");

printf("Lat: %.7f deg, Lon: %.7f deg\n", pvt->lat / 1e7, pvt->lon / 1e7);

printf("Height: %d mm, hAcc: %u mm\n", pvt->height, pvt->hAcc);

printf("Num SV: %u, pDOP: %.2f\n", pvt->numSV, pvt->pDOP / 100.0);

}

```

\*\*说明\*\*：

- 假设`buf`指向`UBX-NAV-PVT`消息体（92字节）。

- 提取关键字段（如时间、位置、精度、状态）。

- 检查`fixType`以判断RTK状态（UBX-8特有）。

---

### 8. \*\*注意事项\*\*

- \*\*波特率\*\*：`UBX-NAV-PVT`数据量较大（100字节/消息），高更新率（如10 Hz）需高波特率（如115200）。

- \*\*消息频率\*\*：通过`UBX-CFG-RATE`设置输出频率（默认1 Hz）。

- \*\*RTK支持\*\*：仅UBX-8模块（如M8P）支持`fixType=4/5`，UBX-7限于`0-3`。

- \*\*固件版本\*\*：UBX-8固件（2.0x/3.0x）可能优化精度和标志，查阅《u-blox 8/M8 Receiver Description》。

- \*\*调试\*\*：使用u-center查看`UBX-NAV-PVT`字段，验证RTK状态和精度。

---

### 9. \*\*总结\*\*

- \*\*作用\*\*：`UBX-NAV-PVT`提供位置、速度、时间和状态的综合导航解，是u-blox模块的核心输出消息。

- \*\*结构\*\*：92字节消息体，包含时间（`iTOW`, `year`等）、位置（`lon`, `lat`等）、速度（`velN`, `gSpeed`等）、状态（`fixType`, `flags`等）。

- \*\*关键字段\*\*：

- `fixType`：定位类型（0-5，UBX-8支持RTK）。

- `hAcc`, `vAcc`：精度估计（UBX-8 RTK达毫米级）。

- `numSV`：卫星数（UBX-8支持更多星座）。

- `flags`：RTK和差分状态（UBX-8更详细）。

- \*\*UBX-7 vs UBX-8\*\*：

- UBX-7：支持标准定位和DGNSS，精度米级/亚米级。

- UBX-8：支持RTK，精度厘米级，更多星座和标志。

- \*\*与您的代码\*\*：

- `OUTPROTOMASK\_GPS`：输出`UBX-NAV-PVT`，UBX-8支持RTK状态。

- `OUTPROTOMASK\_RTCM`：基站模式输出RTCM3，`NAV-PVT`监控状态，需UBX-8模块。

\*\*建议\*\*：

- 确认模块型号（NEO-7M、NEO-M8P等），UBX-7不支持RTK。

- 使用u-center验证`fixType=4/5`和`hAcc`（RTK模式）。

- 如果需要RTK，升级到u-blox 8/M8或9（如ZED-F9P）。

如果您有具体模块型号、固件版本或应用场景（如解析代码、RTK配置），我可以提供更详细的实现或优化建议！

## RTCM协议：

* RTCM协议是一种专门为差分全球导航卫星系统（DGNSS）和实时动态定位设计的通信协议，编码形式是二进制，大端序。
* 消息框架：每个RTCM3消息由以下部分组成：  
   1. 前导码：固定为0xD3，用于消息同步

1. 保留位：6位
2. 消息长度：10位，表示消息体的字节数（不包含前导码、保留位、CRC）
3. 消息体：包含实际数据
4. CRC校验：24位循环冗余校验，用于验证消息完整性

* 消息类型：

(1) 观测数据消息

这些消息包含基站的GNSS观测数据（如伪距、载波相位），是RTK和DGNSS的核心。

1001-1004：GPS观测数据

1001：L1伪距观测（基本DGNSS）。

1002：L1伪距+载波相位（RTK）。

1003：L1/L2伪距观测。

1004：L1/L2伪距+载波相位（最常用）。

1009-1012：GLONASS观测数据

类似1001-1004，但针对GLONASS卫星。

1071-1077：GPS MSM（Multiple Signal Messages，多信号消息）

MSM1-MSM7，支持多频段（如L1、L2、L5），MSM4（伪距+载波相位）和MSM7（高精度）最常用。

1081-1087：GLONASS MSM

1091-1097：Galileo MSM

1101-1107：SBAS MSM

1111-1117：QZSS MSM

1121-1127：BeiDou MSM

用途：漫游站使用这些观测数据校正自己的GNSS测量，计算高精度位置。

……(还有很多，可问下ai)

# 环形缓冲区：

* 编程思路：在across的代码中，使用的环形缓冲区长度实际上是length-1，即pr（读指针）前的一个字节并不存放新数据。**向缓冲区压1byte的思路**：pw的下一字节！=rw则存放数据，=则置溢出标志为1
* **inline关键字**：在c语言中，inline关键字用于建议编译器将函数内联展开（应该是将调用函数处直接替换为函数的实现），从而减少函数调用的开销，代价是增大代码体积（以空间换时间）。适合用于短小且频繁调用的函数。

**使用实例：**test.h:

static inline void test(void)

{

……

}

其他不确定的使用方法：  
test.h:

inline void test(void)

{

……

}

## 编程相关

* 现在定义了以下几个状态：解码状态、接收消息状态
* 16 27 32

# DMA（F4）

* 关于DMA配置结构体的DMA\_MemoryBurst：用于配置内存和外设端的数据突发传输模式。突发传输（Burst Transfer）是一种高效的数据传输方式，通过单次总线访问传输多个数据单元，减少总线占用时间，提升传输效率。​**FIFO 是突发传输的硬件基础**，启用了突发传输必须启用fifo。

# SysTick

* 系统时钟也即每秒的时钟周期数，如72M的时钟频率，这每us的时钟周期是72，systick的计数器每增加1便是一个时钟周期
* 使用SysTick实现时基函数以及us、ms级延时：

1. 时基函数：使用SysTick实现的原因是S是内核的外设，编写的代码兼容性更好，不区分硬件平台。可使用SysTick\_Config配置1ms中断，该函数配置了S的重装载值、中断优先级(正点原子配置的中断优先级为15，最低)、清空计数值、时钟源、使能中断、使能Systick
2. ms级延时：基于时基函数实现即可
3. us级延时：基于systick时钟频率实现。

# USART

* usart发送接收超时机制设计原则：发送超时时间设计原则：usart发送单字节时间：10byte/baubrate（8字节+1校验位+1停止位）；设置时需要加入一定余量；接收超时时间设计原则：需要根据通信模块设置（可根据参考手册模块响应时间设置）

# 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

1. 枚举变量：

为什么定义枚举变量：1.语义清晰，代码可读性强。枚举类型直接给数据赋予有意义的名称，2.限制取值范围，枚举类型只能取定义中的值，防止赋予非法值

* sizeof()：sizeof不是函数，而是一个运算符，用于计算变量或类型所占内存的字节数
* 联合体union：在c语言中，联合体允许在同一块内存区域中存储不同类型是的变量，与结构体不同，联合体在同一时间只能存储一个时间的值。这种特性使得联合体在节省内存和处理不同变量类型时非常有用。联合体所有成员占用同一块内存，大小由最大成员决定
* void\* memset(void\* s, int c, size\_t n)函数：

***参数说明****：*

s：指向要填充的内存区域的指针。c：要设置的值（以int形式传递，但实际存储会转换成unsigned char）。n：要填充的字节数。

***返回值****：*

返回指向目标内存区域的指针（即s）

***工作原理***

memset 将内存区域 s 的前 n 个字节设置为值 c（按字节操作）。

c 的值会被截断为 unsigned char（0到255），因此只能设置单个字节的值。

***注：***

memset 比手动循环更快，因为它通常使用底层硬件指令优化。