1. 基于stm32的自行车码表的设计

主控采用了stm32f103c8t6

气压、温度、传感器：bmp180（I2C通信）

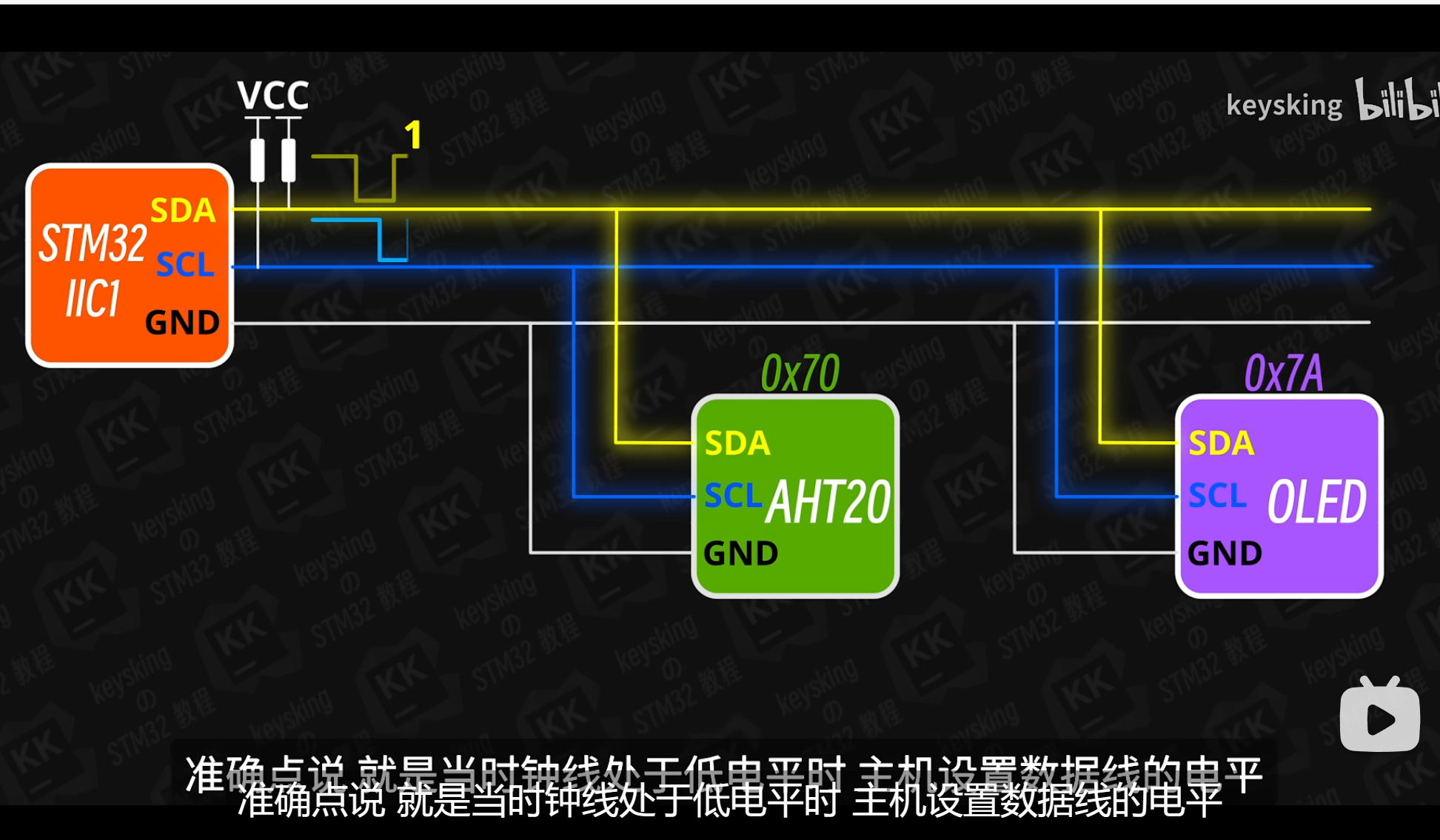
测速模块：霍尔传感器

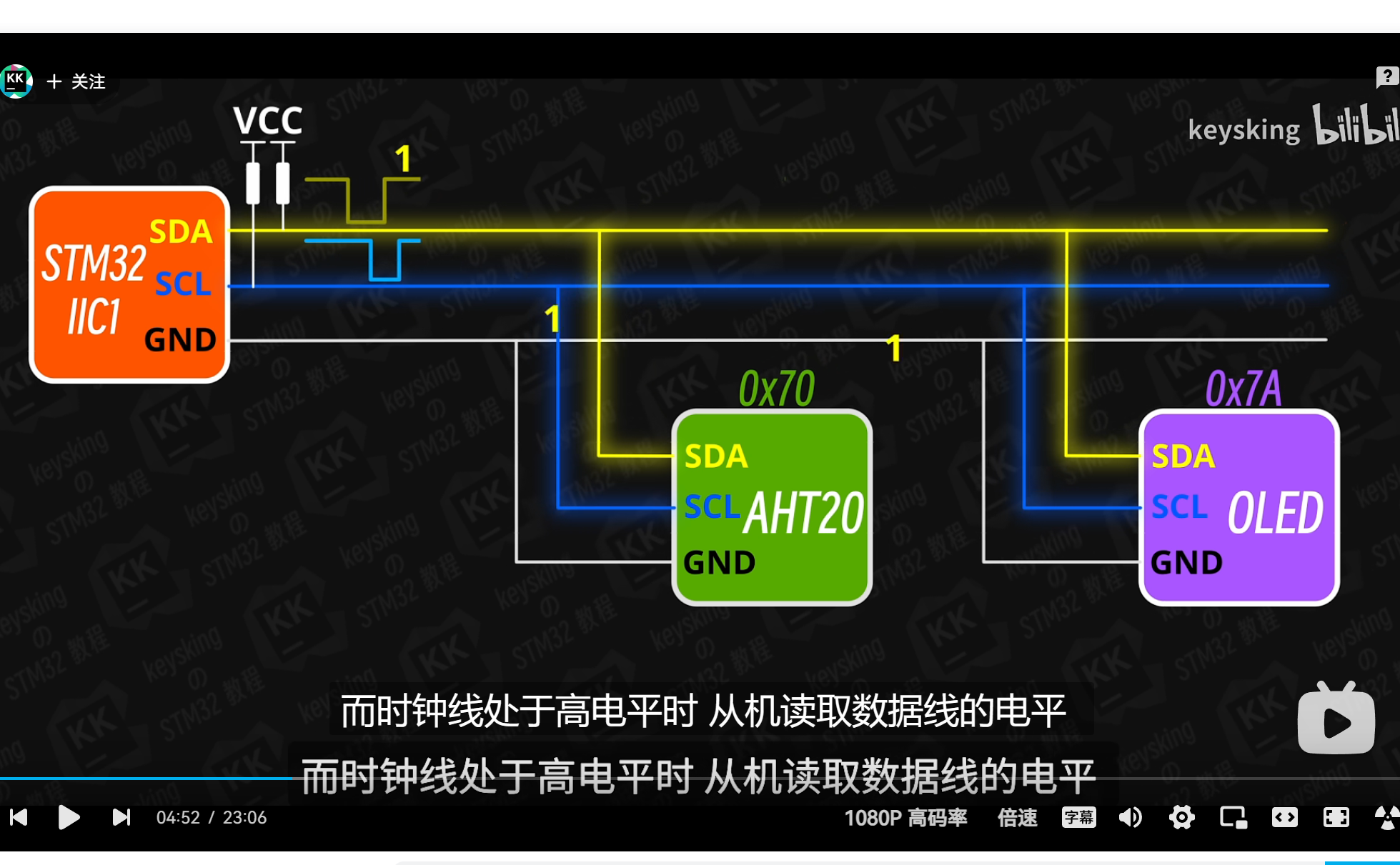
心率检测模块：pluse sensor传感器

用的是I2C通信，简述下I2C通信的过程？

起始信号：

SCL为高电平，SDA由高变低，为起始信号





发送地址：从机一般都是七位地址，但是发送的是8位地址，第8位 0：表示怎么样

1：表示怎么样，然后发送数据，发送数据时SCL低电平时数据改变，高电平时 读取数据

等待应答，所谓应答就是SCL高电平时，从机将SDA的电平拉低

然后循环往复 ，

直到遇到停止信号？

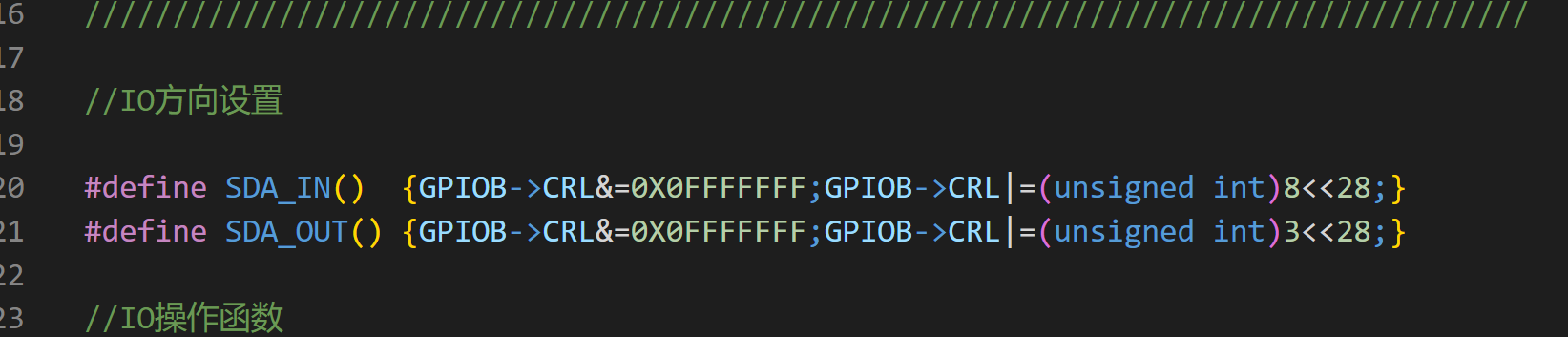
停止信号就是SCL为高电平，然后

I2C通信时的IO口是怎么设置的？

注意IO口要设置成开漏输出+上拉电阻。开漏输出是为了实现线与的功能，避免总线短路。接上拉电阻（直接接电源），是因为要确保不传输数据时，SCL和SDA都是高电平的状态。

值得注意的一件事，当我用proteus仿真时，必须接上上拉电阻，I2C的波形才正常，能够读取到BMP180的数据。但是实物中，在最小系统板上没有接上上拉电阻，也可以正常输出波形。

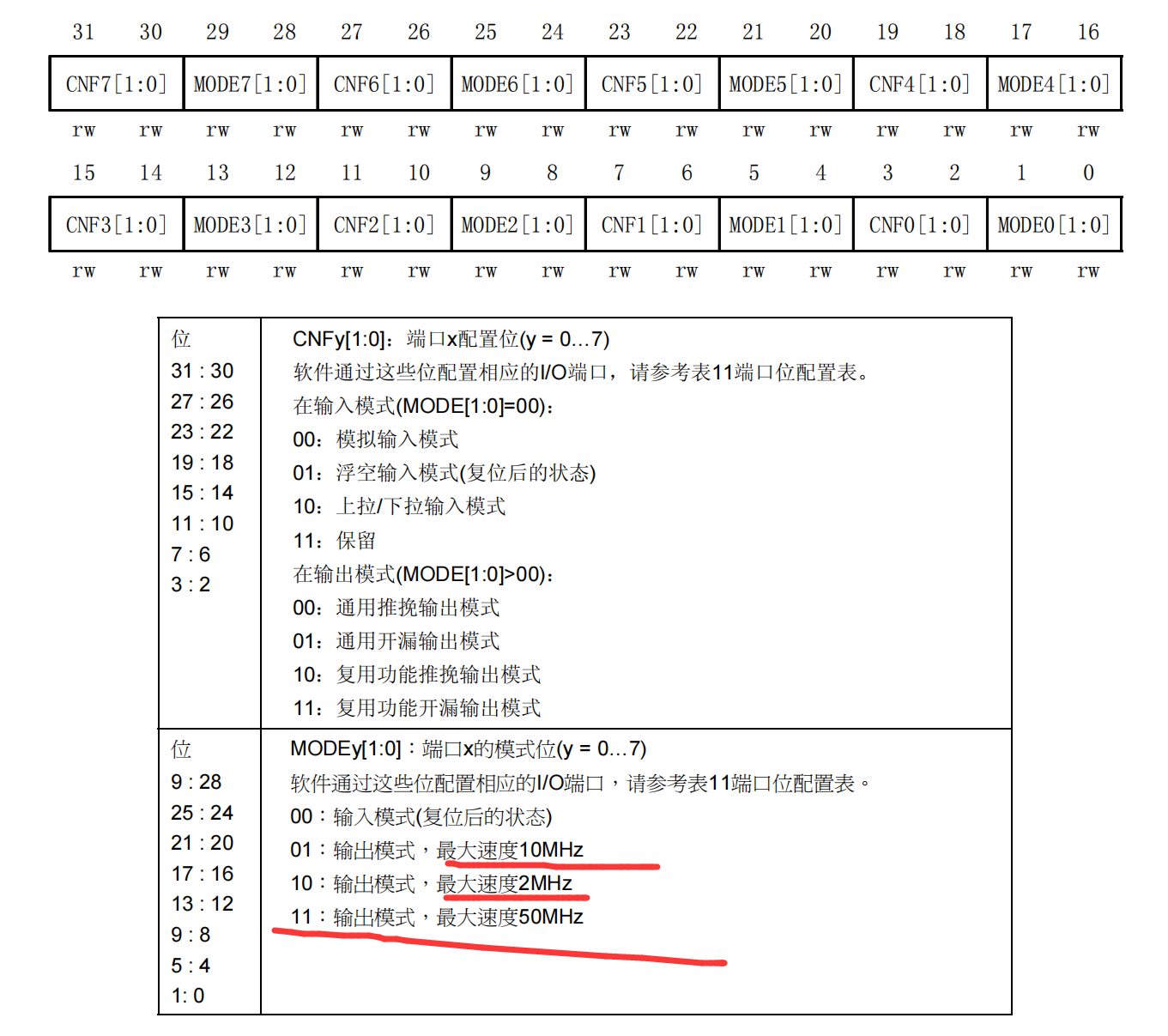
小细节：



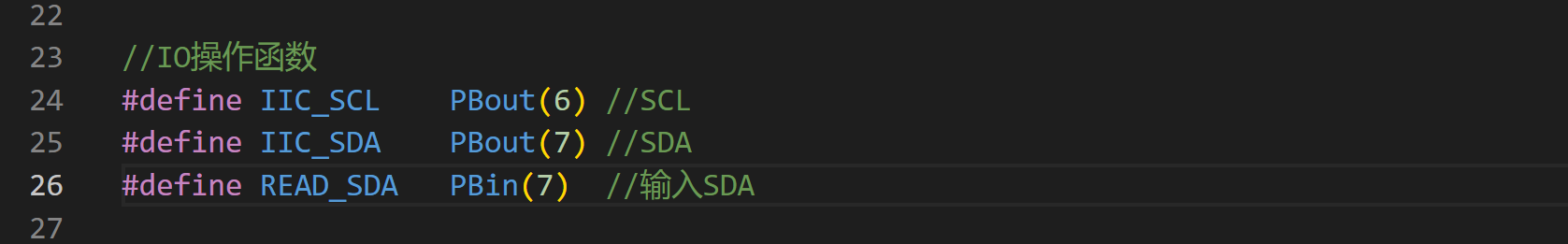
1.查看寄存器手册，首先将GPIOB->CRL&=0x0fffffff 将B7设置为输入模式

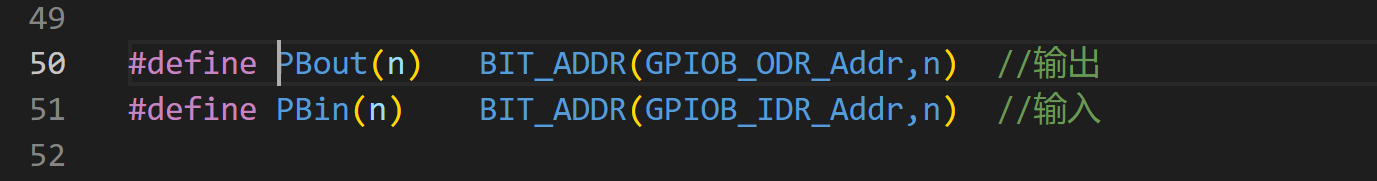
GPIOB->CRL |= 8<<28 上拉下拉输入模式

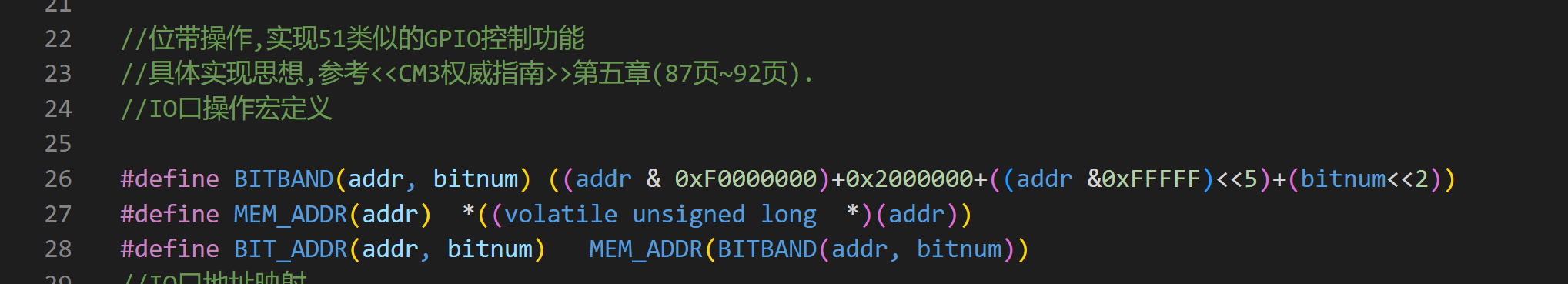
同理，查看手册可知，将B7 设置为复用功能开漏输出模式，CRL位不发生改变



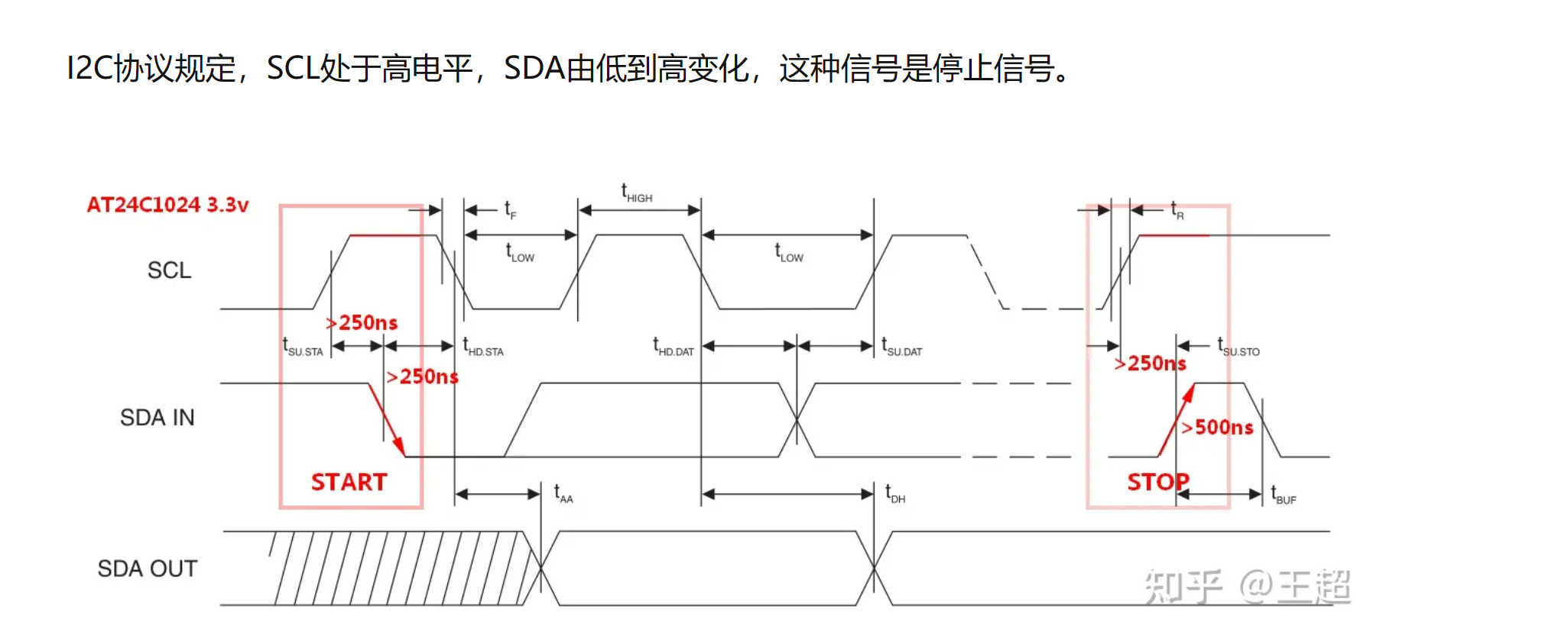
1. 通过位带操作，将对某一位的操作，转化为了对32位地址的操作？不太会



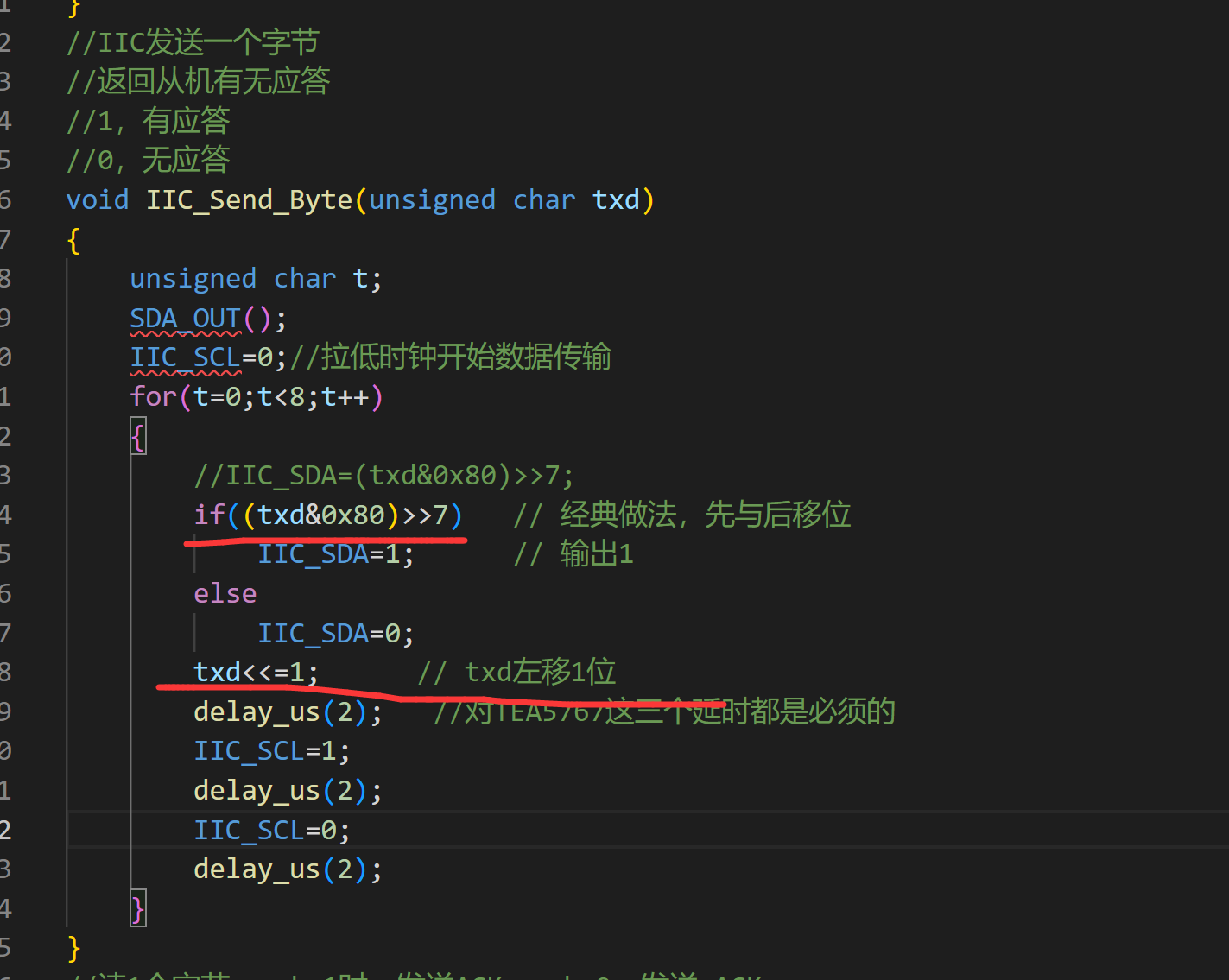




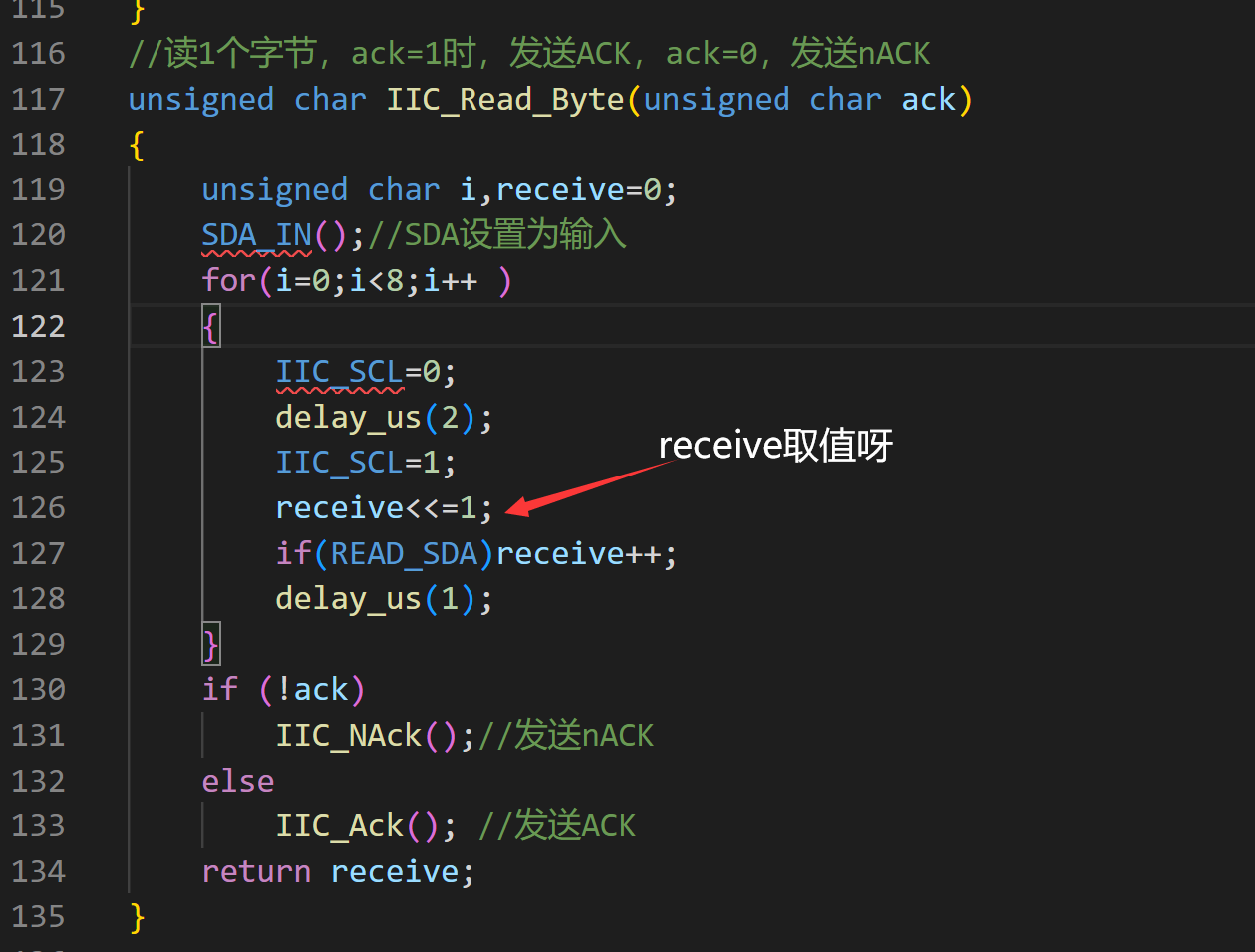
1. 组成I2C的操作



先发送高位，发送高位的方法：与法取值，右移七位，后左移位（下一个数据移到最高位）



最先收到的是高位，因此要左移一位才可以，判断读取到的数据，如果为1就加1 ，如果为零就不动 ，时钟是在第一位加1

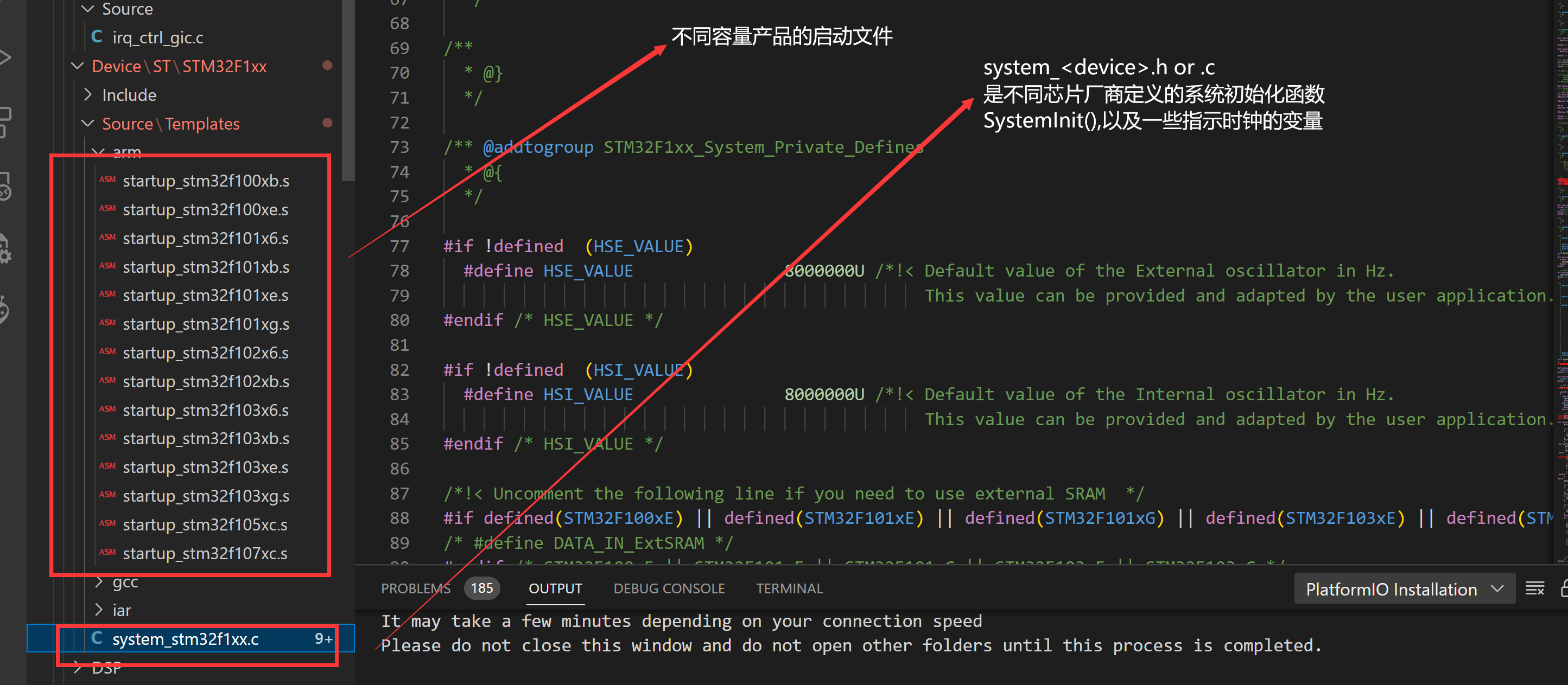


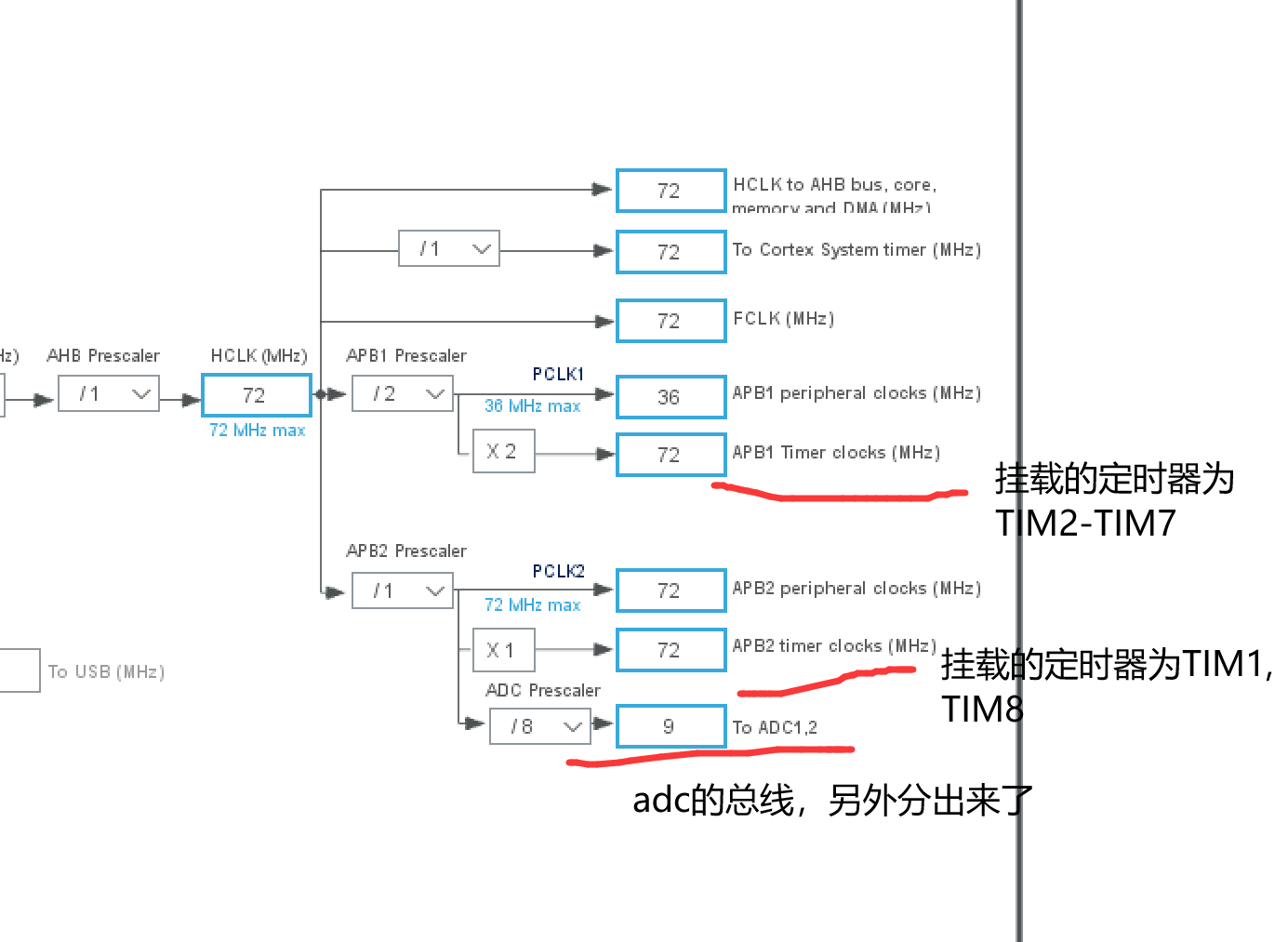
参考裸机笔记->ARM处理器的工作模式 7种工作模式 37个寄存器 因此30个为通用型。

pc固定 cpsr固定 还有 5个spsr固定

stm32的启动过程：

1. 确定启动设备
2. 构建堆和栈的大小进行定义，并在代码区（如果是从flash中启动，地址就是0x8000 0000）的起始地址建立中断向量表，其中第一项表示栈顶地址，第二表项为中断服务入口地址。然后在复位中断服务函数中跳转到c/c++ 标准实时库的\_\_main函数。

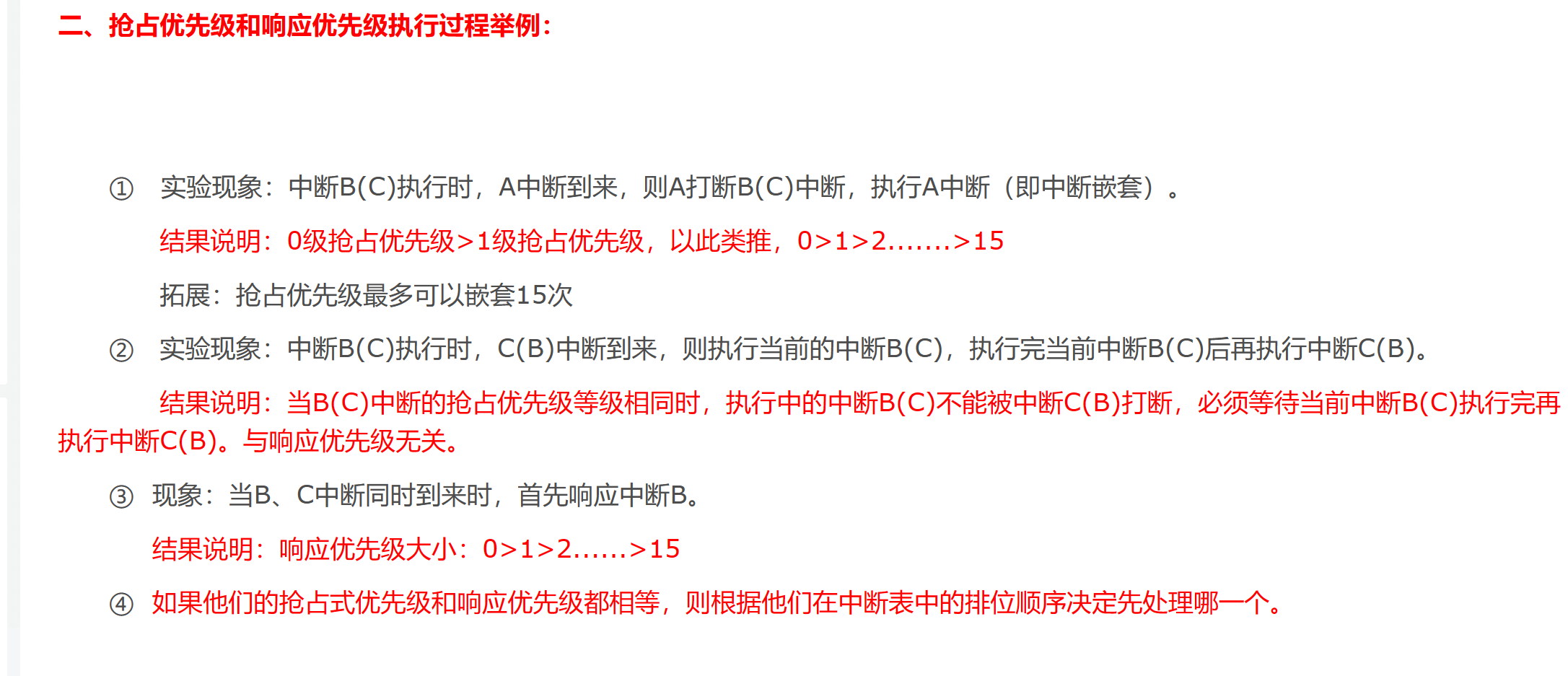




adc是要初始化时钟的，初始化的时钟和adc的转换速率有关 9MHZ

中断优先级的配置

本次中断优先级 配置为2个占先优先级 2个副优先级

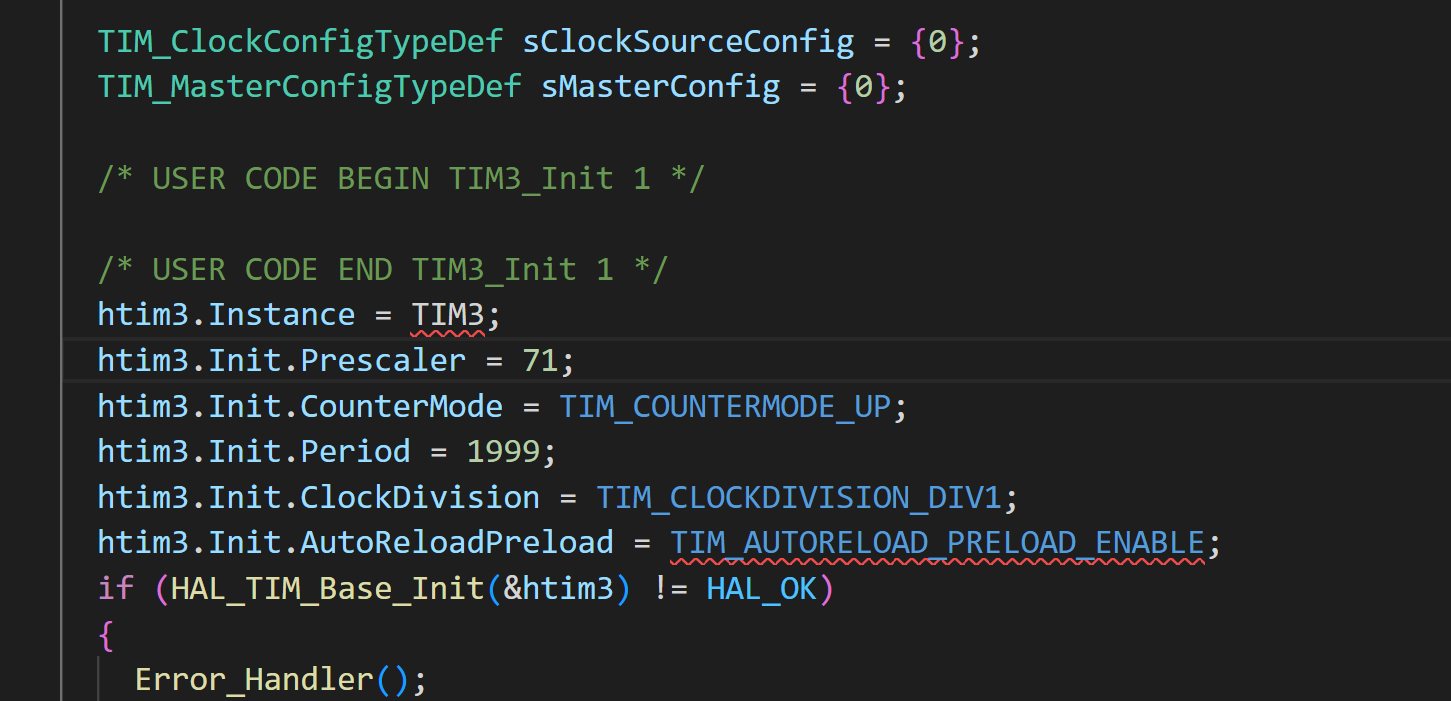


外部中断的优先级为1，0

定时器的设置：

采用多个定时器

定时器3 定时2ms



作用每2个ms读取一下adc的值

定时器 4 定时 1s

各个寄存器的参数如下

定时器的定时时间的确定：

定时时间由Prescaler 和period共同确定

定时公式为：

T = (TIM\_Period+1)✖（TIM\_Prescaler+1）/ TIMxCLK

算出来的定时时间就是1s

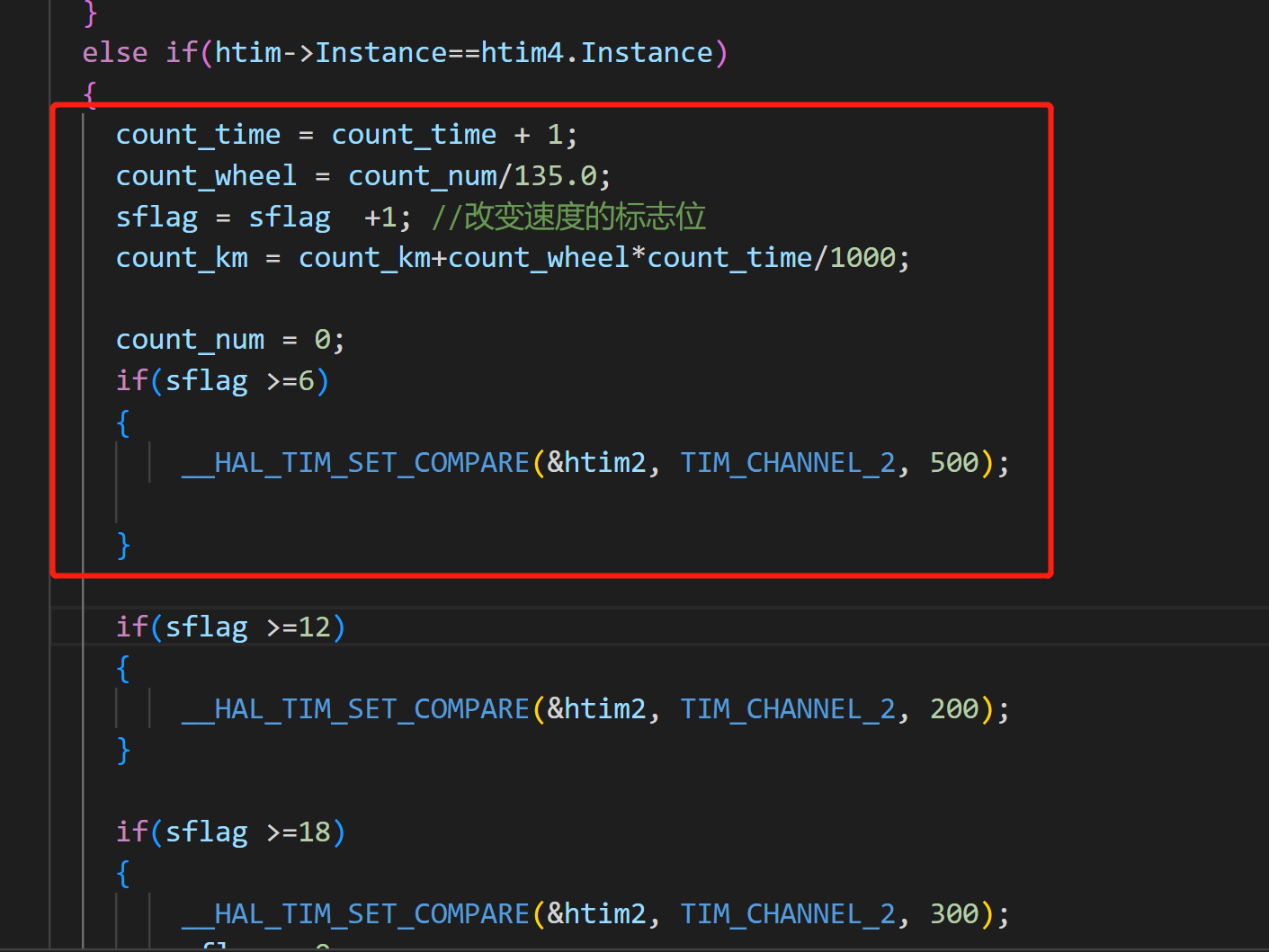
定时器4的作用就是

功能1：每隔一秒测一下速度

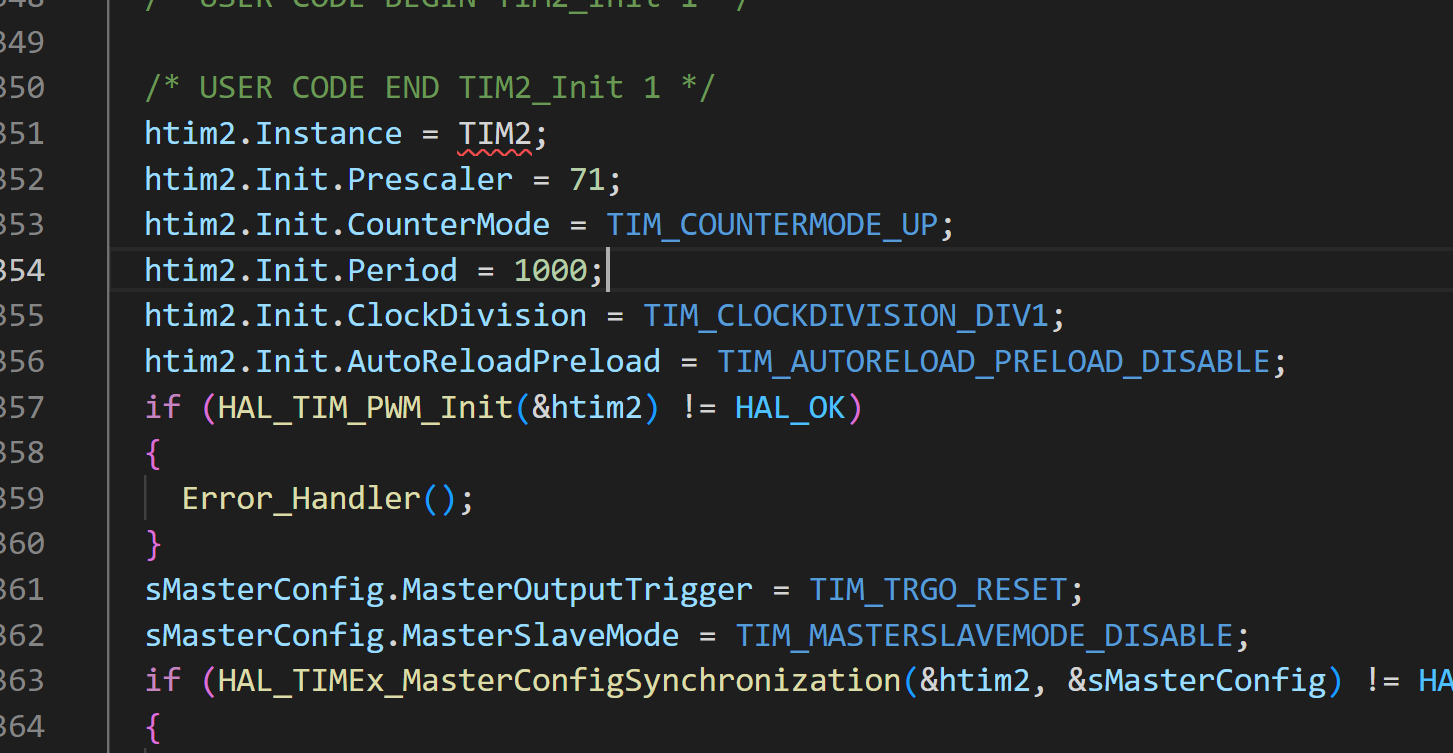
功能2：然后还有个作用就是计时的作用

这个计时作用的意思就是 每六秒 改变一下占空比 分别为 500，200和100

PWM波的产生由定时器2负责。



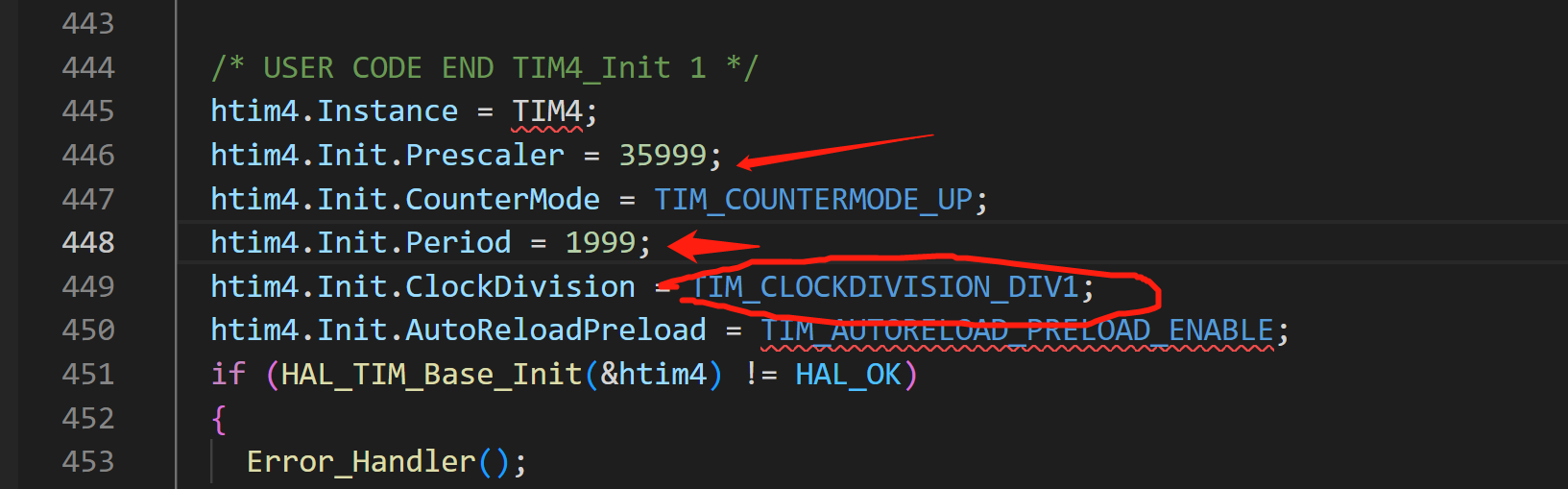
定时器2 来产生pwm波



Tim2工作在pwm1模式下的工作原理是，当定时器启动计数后，若当前计数值小于某通道（假设为x通道）比较值，则对应x通道的输出引脚保持高电平；若当前计数值增至大于x通道比较值，则引脚反转为低电平；计数值继续增大至重装值，引脚恢复高电平，计数值重装再次计数，上述过程重复执行。如果将输出比较值设为Ncom,重装值设为Nprer,则PWM的信号频率为



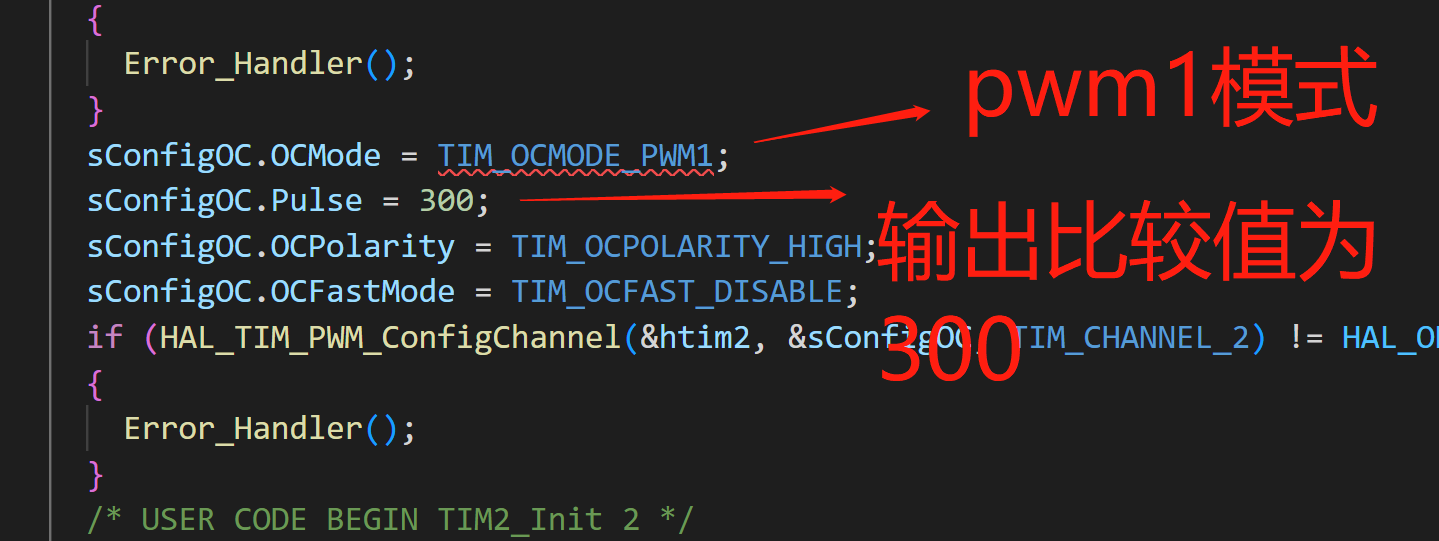




因此频率由重装值来确定，大概为

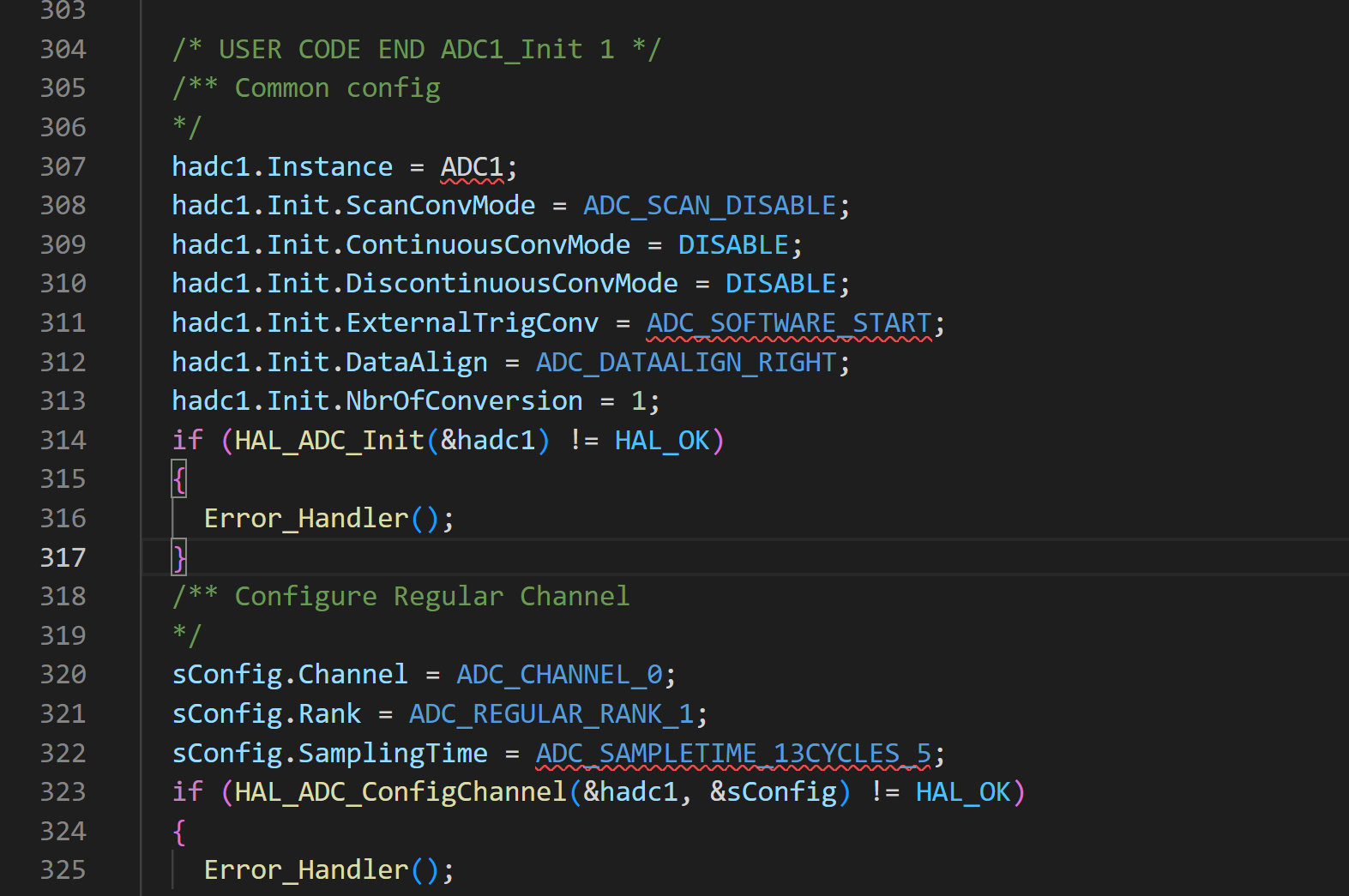
频率为：72 000000 /72 /1000 = 1000000hz =1khz

输出比较值为300



ADC转换器：

STM32F10X 系列微控制器 内置3个12位A/D 转换模块（ADC）,最快转换时间为1us,并且具有自校验功能，能够在环境条件变化时提高转换精度。



这段代码涉及了一些与 ADC（模数转换器）相关的寄存器和配置。让我逐行解释每个寄存器的含义：

1. hadc1.Instance = ADC1;：这行代码将 hadc1 结构体的 Instance 成员设置为 ADC1，表示使用 ADC1 模块。
2. hadc1.Init.ScanConvMode = ADC\_SCAN\_DISABLE;：这行代码禁用了扫描模式，即只对单个通道进行转换。
3. hadc1.Init.ContinuousConvMode = DISABLE;：这行代码禁用了连续转换模式，即只进行单次转换。
4. hadc1.Init.DiscontinuousConvMode = DISABLE;：这行代码禁用了不连续转换模式，即不跳过转换序列中的任何通道。
5. hadc1.Init.ExternalTrigConv = ADC\_SOFTWARE\_START;：这行代码将外部触发转换设置为软件触发模式，即通过软件启动转换。
6. hadc1.Init.DataAlign = ADC\_DATAALIGN\_RIGHT;：这行代码将数据对齐设置为右对齐模式，即转换结果的高位对齐到 ADC 寄存器的高位。
7. hadc1.Init.NbrOfConversion = 1;：这行代码设置转换序列中的转换次数为 1，即只进行一次转换。
8. if (HAL\_ADC\_Init(&hadc1) != HAL\_OK)：这行代码使用 HAL 库函数初始化 ADC1 模块。如果初始化失败，则调用 Error\_Handler() 函数进行错误处理。
9. sConfig.Channel = ADC\_CHANNEL\_0;：这行代码配置 ADC 转换的通道为通道 0。
10. sConfig.Rank = ADC\_REGULAR\_RANK\_1;：这行代码配置转换的通道在转换序列中的排名为 1。
11. sConfig.SamplingTime = ADC\_SAMPLETIME\_13CYCLES\_5;：这行代码配置采样时间为 13.5 个时钟周期。

这些寄存器和配置用于初始化和配置 ADC1 模块及其转换参数。通过设置这些寄存器的值，可以控制 ADC 的工作模式、转换通道、数据对齐方式、触发方式等。根据您提供的代码，这段代码的作用是初始化 ADC1 模块，并配置转换通道 0 的参数。

经过上面的总结可以看出：

总转换时间的公式为：

总转换时间Tconv = 采样时间+12.5cycles

我们这里的采样时间很明显为13.5个时钟周期，即总体的转换时间为2.8888us

需要说明的是采样时间越长，转换结果越稳定。

Stm32f103的adc分为注入通道组和规则通道组。

注入通道组有4个数据寄存器，最多允许4个通道转换，可随时读取相应寄存器的值，没有DMA请求。

例如，n=1,被转换的通道数为1，2，3。第一次触发，通道1被转换；第二次触发，通道2被转换；第三次触发，通道3被转换；并且产生EOC和JEOC事件；第4次触发，通道1被转换。

规则通道组的转换好比程序的正常执行。

DMA控制器，经理只告诉保管员去哪个区域（源地址）要哪种类型的物品（数据类型）、数量多少（数据长度），送到哪里去（目标地址）等信息，其他事情就不用管了；然后保管员完成任务回来，打断一下正在做其他事情的的经理（中断）并告诉他完成情况，或者不打断经理的工作只是把完成任务牌（标志位）挂到经理面前即可，这个仓库保管员正是DMA控制器。

在监控系统中，往往需要对ADC采集的一批数据进行滤波处理（如中值滤波）。ADC先高速采集，通过DMA把数据填充到RAM中，填充到一定数量后，再传给微控制器使用，这样处理比较好。

DMA允许外设直接访问内存，从而形成对总线的独占，这是DMA技术的缺点。如果DMA传输数据量大，会造成中断延时过长，在一些实时性（硬实时）的嵌入式系统中这个不允许的。

adc的工作模式：

1. 单次转换模式2.连续转换模式 3.扫描模式 4.间断模式

我们用的单次转换模式 采用的是软件触发的方式

显示屏 用的SPI通信

1.1.44寸lcd用的SPI通信，但是SPI通信我并不是很熟，只知道有四种模式

系统的整体功能为：

系统主要有stm32f103c8t6芯片，1.44寸lcd显示器，心率传感器pluse sensor, bmp180(气压，海拔，温度)

1. Lcd屏幕和stm32采用的通信方式为SPI通信

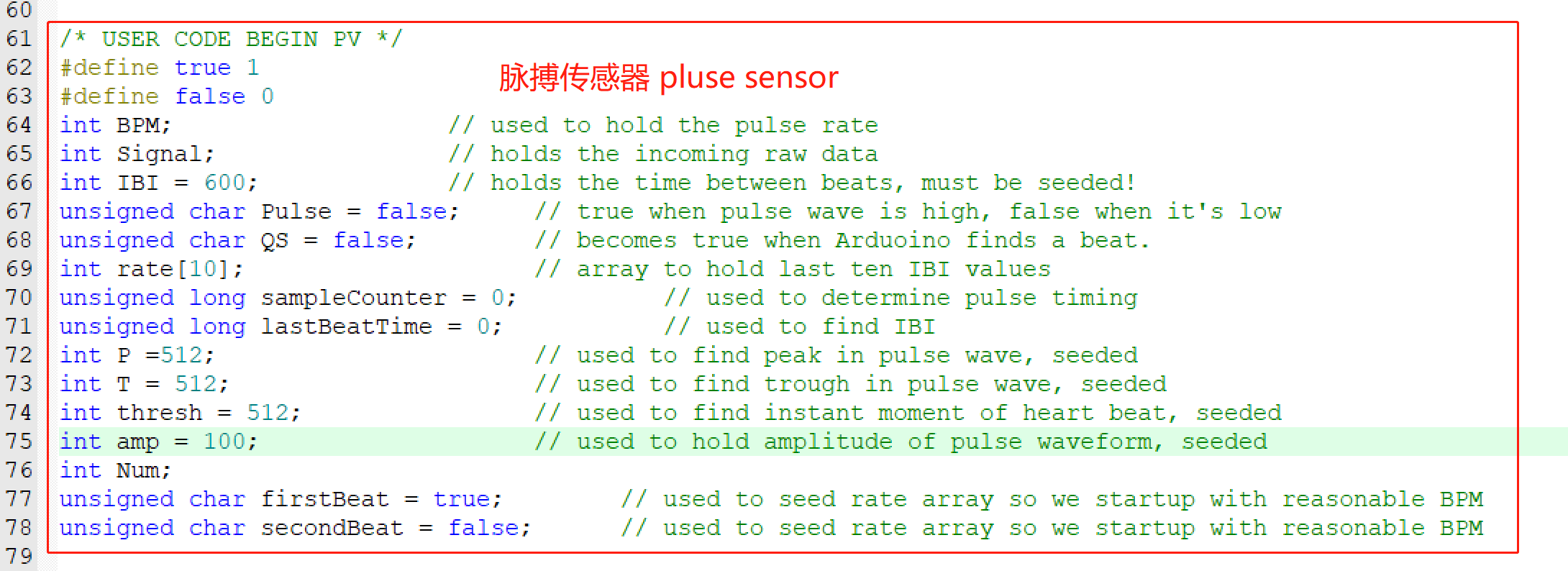
2. pluse sensor是通过adc采集的，每隔2ms采集一次，每次adc的转换时间为2.888us,经过数据处理后显示出心率。

具体的数据处理过程：

[pulsesensor传感器以及STM32相关学习笔记\_艾卡西亚没有雨的博客-CSDN博客](https://blog.csdn.net/WANGXINGYUUUUU/article/details/107726531)

1.缓存一个波形周期内的多次采样值，求出最大最小值，计算出振幅中间值作为信号判定阈值。

2.通过把当前采样值和上一采样值与阈值作比较，寻找到「信号上升到振幅中间位置」的特征点，记录当前时间。



sprintf(str1,"%d",count) // 将数字转化为字符串

3.寻找下一个特征点并记录时间，算出两个点的时间差值，即相邻两次脉搏的时间间隔 IBI。

由 IBI 计算心率值 BPM。

程序功能图：

1. 发生中断 2ms

获取adc数值并且除以4

采样计时

获取距离上次心跳过去了多久

如果adc的值小于阈值并且 距离上次心跳至少过去了IBI的 3/5

如果adc的值小于最小值（波谷）那么adc的值等于波谷

如果adc的值大于阈值 且大于波峰

那么adc的值就等于波峰

如果 adc的值小于阈值 而且 pluse == true

计算振幅

计算阈值

波峰等于阈值

波谷等于阈值

pluse =false

如果距离上次心跳过去了250ms（相当于每分钟240次）

如果adc值大于阈值 ，pluse=false, 而且时间也过去了IBI的3/5

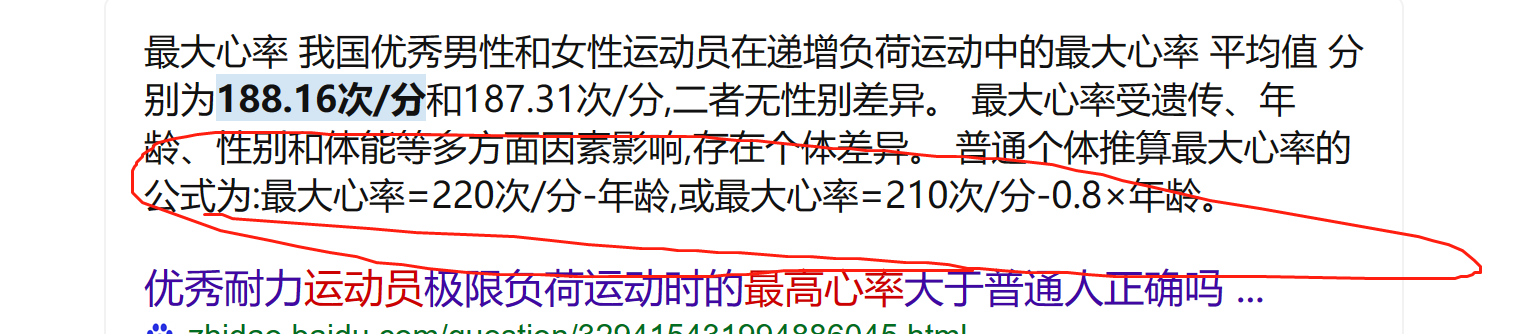
Pluse==true

IBI = 采样时间-上次心跳的时间

上次心跳的时间 = 采样时间

如果是第二次心跳

心率范围：



————————————————

版权声明：本文为CSDN博主「艾卡西亚没有雨」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/WANGXINGYUUUUU/article/details/107726531

3. bmp180是通过I2C和主机进行通信的（0xee,或者0xef）

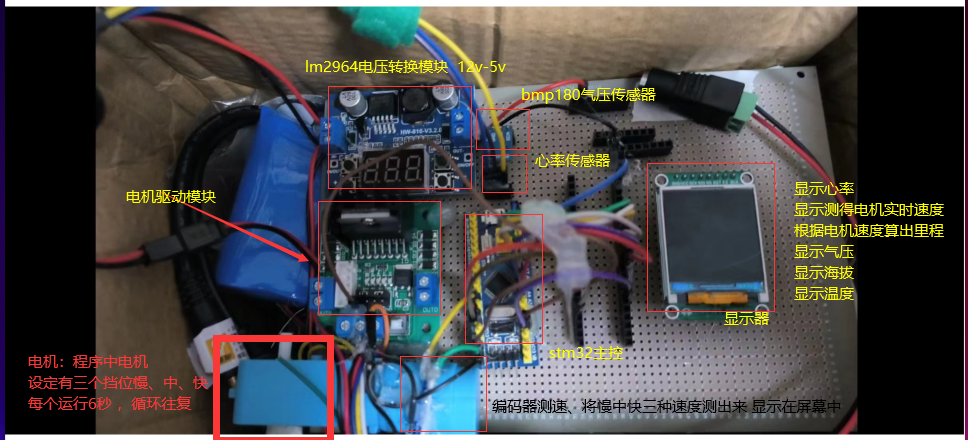
气压、温湿度的采集是放在主程序里，没事了就测测

还有lcd的显示都是放在主程序里的

1. 定时器4的定时时间为1s, 在定时器4中断中主要负责计算速度、里程数和改变占空比
2. 定时器2是用来产生pwm波的，pwm的频率为1khz，输出引脚是PA1

起始电平为高电平，当定时器计数小于设定值时 输出高电平，大于输出低电平

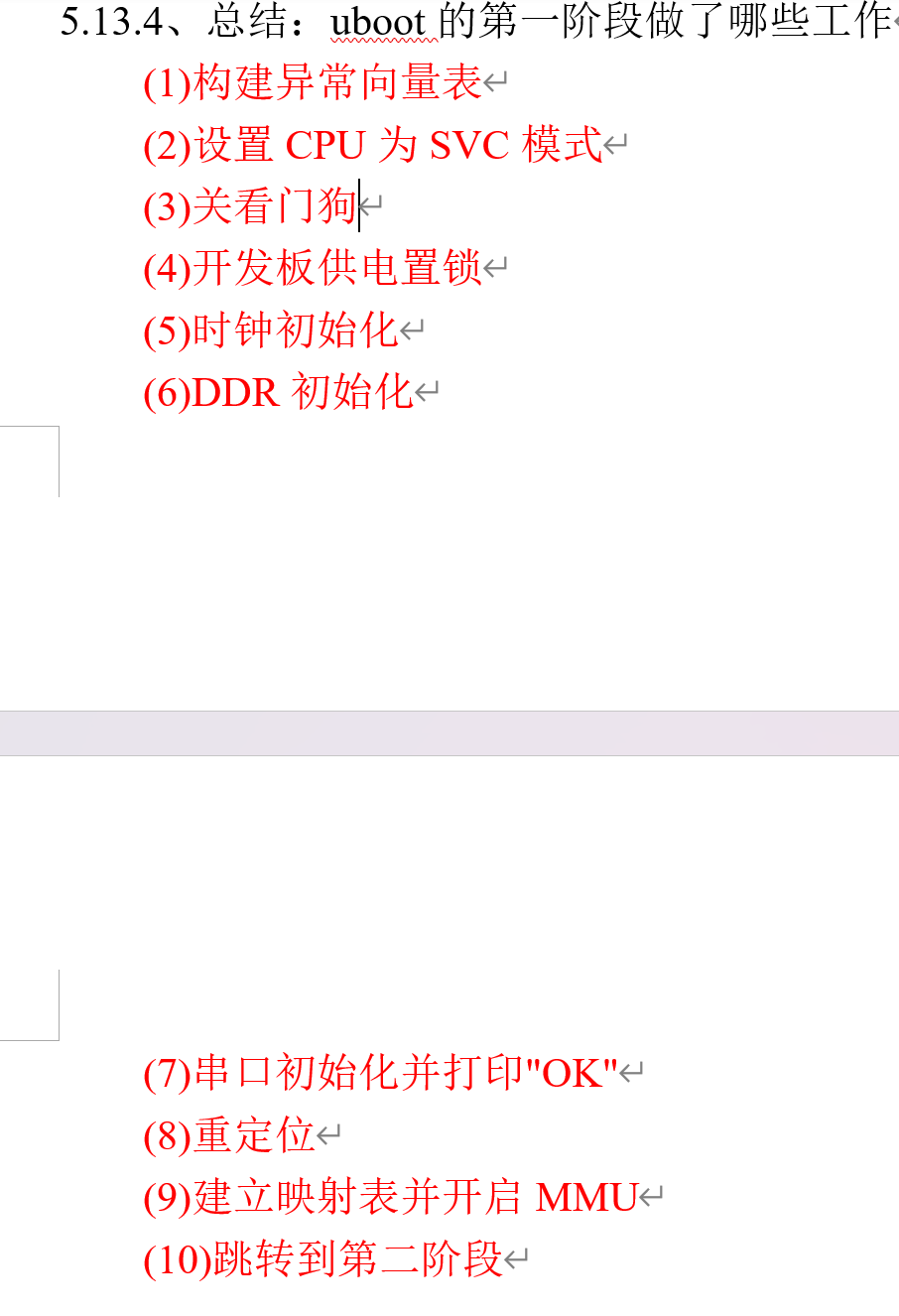
1. 还有用一个**PB4**产生外部中断，用来计算脉冲数

，

[全网最详细解释Keil-MDK中Code、RO-data、RW-data、ZI-data的含义\_keil code ro rw zi\_wmdszmr的博客-CSDN博客](https://blog.csdn.net/praguejing/article/details/113470155)

端口配置：

|  |  |
| --- | --- |
| I2C |  |
| PB6,PB7 |  |
| // SCL ½ÓPA5£¨SCL£©  // SDA ½ÓPA7£¨SDA£©  // RES ½ÓPB0  // DC ½ÓPB1  // CS ½ÓPA4  // BL ½ÓPB10 |  |
|  |  |



整体的系统框图：

Uboot移植过程中遇到过什么困难:

1. Uboot 第二阶段，我按照三星官方的来，然后停在nand： 。。。 ，我记得分析源码的时候 在初始化板级设备的时候，如果初始化不成功会调用hang()函数，那么我就把这个nand 在source insight上搜，果然调用了一个nand\_init（）函数，然后注释掉，注释掉了之后，就能顺利进入命令行了，但是第二天我去 不能打印信息，然后我找同学换了根线，可以打印了，但是无法进入命令行，我就搜 为什么不能进入命令行，禁掉流控之类的，还是没有办法，当时也没往硬件方面想，来来回回整了好久，我就想是不是硬件的问题，然后我就把uboot中的串口换成串口0，用串口0 ，果然能打印出信息，但是由于内核中用的是串口2 ，我还是无法 启动内核，那我就想着要解决啊 为什么，我还是把它改成串口0 了，抱着试试看的态度
2. 启动了uboot 成功打印信息，进入命令行，无法启动内核
3. 无法打印出信息，无法进入命令行，买了新线不行，借同学的线尽然可以打印出信息，但是依然无法进入命令行，怀疑是硬件问题，我就在那里瞎掰，突然可以了。然后我又把调试信息的串口改成另一个，果然是可以的，然后就行了。
4. 但是依然无法启动内核，一一对照，发现 宏定义中的 寄存器置没有错呀，但是就是启动不了，很郁闷，明明就按照他的来的怎么就不行，然后就对照，怀疑是内存虚拟地址映射表没映射好，于是和正常的uboot就一行一行的对，但是还是无法启动内核，那怎么办，看后面的课程才知道可能有两个原因，内核正常启动了，但是串口搞的不对，那怎么办 我得看串口信息啊，于是买了个串口端子把它换上后，就可以了。内核就顺利启动了。