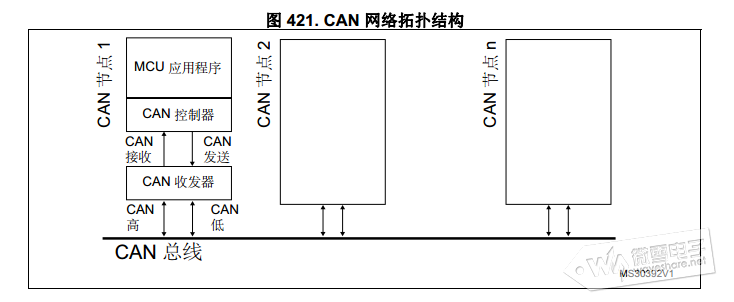
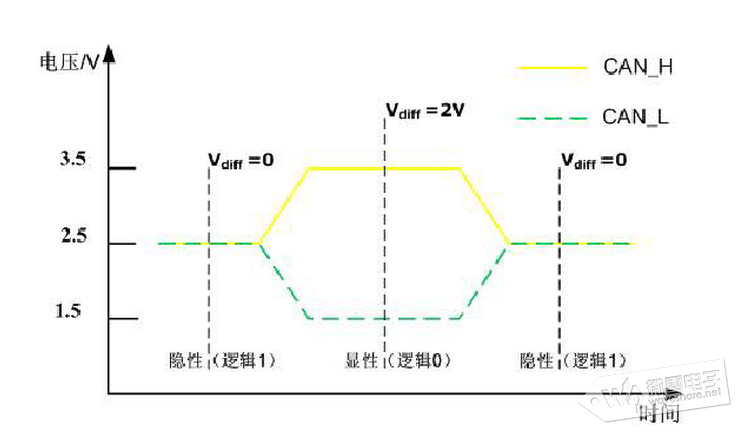
CAN总线通信配置

1. CAN总线说明
   1. CAN是控制器局域网络(Controller Area Network, CAN)的简称
   2. 物理拓扑：



在can总线的物理的两端需要加入120Ω电阻；

* 1. CAN控制器通过组成总线的2根线（CAN-H和CAN-L）的电位差来确定总线的电平，信号是以两线之间的“差分”电压形式出现，总线电平分为显性电平和隐性电平。 CAN总线采用两种互补的逻辑数值"显性"和"隐性"。"显性"数值表示逻辑"0"，而"隐性"表示逻辑"1"。当总线上同时出现“显性”位和“隐性”位时，最终呈现在总线上的是“显性”位。



* 1. CAN总线报文

报文传输由以下4个不同的帧类型所表示和控制：

数据帧：数据帧携带数据从发送器至接收器。

远程帧：总线单元发出远程帧，请求发送具有同一识别符的数据帧。

错误帧：任何单元检测到总线错误就发出错误帧。

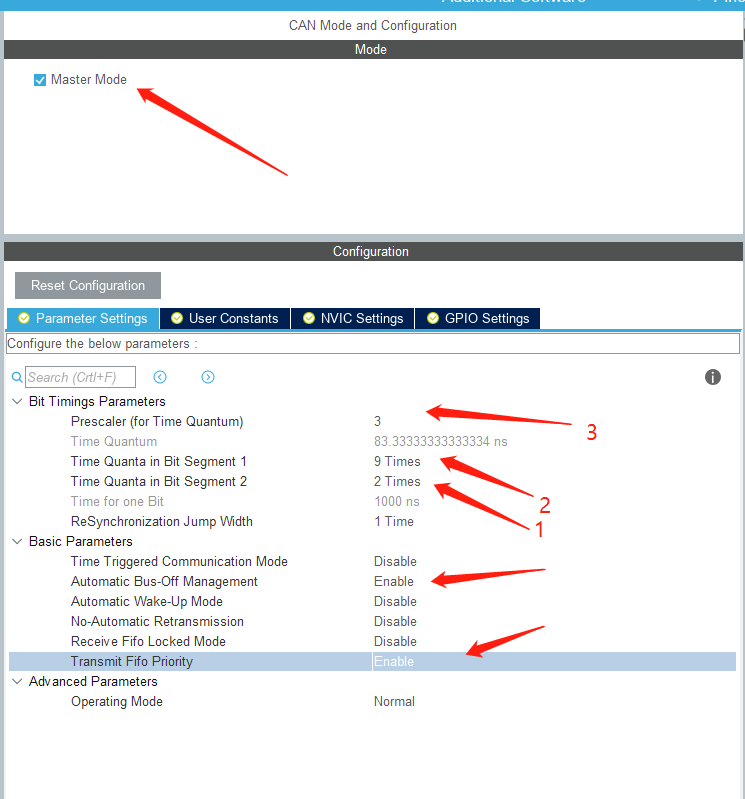
帧间空 : 数据帧（或远程帧）通过帧间空间与前述的各帧分开。

过载帧：过载帧用以在先行的和后续的数据帧（或远程帧）之间提供一附加的延时。

传输数据的数据帧可携带8个8bit的数据。

* 1. CAN总线过滤器
     1. 在CAN通信中接收到信息时，接收器节点会根据标识符的值来判断信息是否需要，若不需要则丢弃信息。
     2. Can过滤信息有硬件过滤和软件过滤两个过滤方式，
        1. 软件过滤就是在接收到can总线消息时代码中进行判断；
        2. 硬件过滤器需要对can进行配置，有两种方式，idlist模式和idmask模式，idlist是只接收标识符在list中的消息，idmask是指掩码模式，标识符只关注几个位，一般使用掩码模式进行过滤；

1. Can总线配置
   1. Can总线波特率



需要按照图中的顺序设置波特率。由于STM32cubeMX会自动纠正错误的参数，若设置的参数暂时算出出的波特率超出范围，会直接重置，所以需要有一定的顺序才能设置成想要的参数。

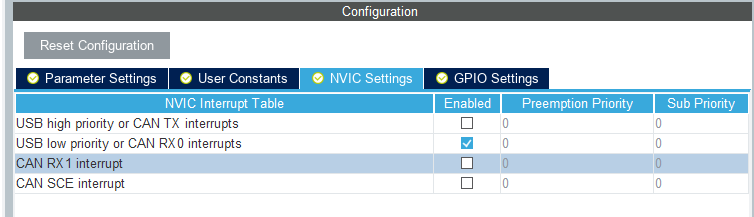
Can挂载在APB1上，时钟为36M，现在想要设置为can的最大波特率1M，上面设置分频为Prescaler=3, BS1=9, BS2=2, SJW=1，计算公式如下：

经过分频后的频率为36MHz / 3 =12MHz

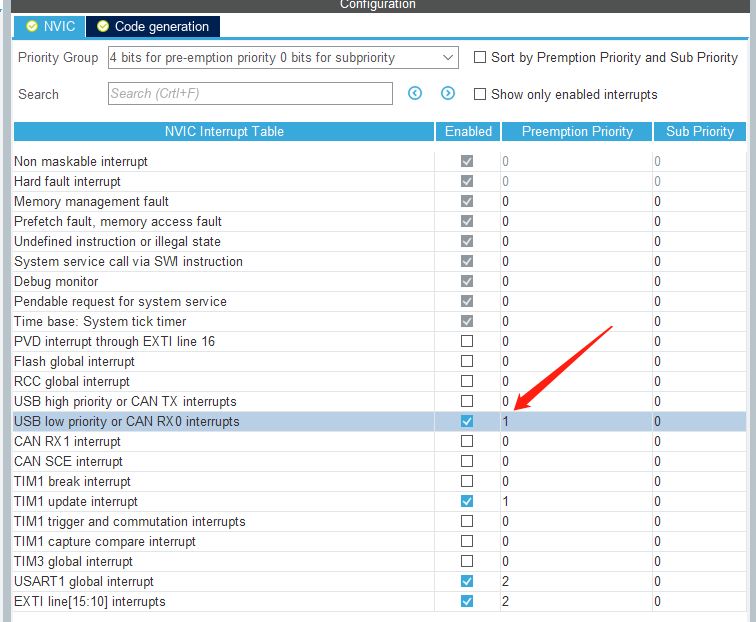
CAN波特率为 36MHz / 3 / (SJW+BS1 +BS2 ) = 36MHz /3 /12 = 1MHz

下面两个参数是使能了自动离线管理和发送队列的优先级。

* 1. 开启中断



* 1. NVIC配置中断优先级



1. 生成代码完成配置
   1. 生成代码后，在can.c中添加如下函数，在main函数中can初始化结束后调用

/\*\*

\* @brief Configures the CAN.

\* @param None

\* @retval None

\*/

void CAN\_Config(void)

{

CAN\_FilterTypeDef sFilterConfig;

/\* Configure the CAN Filter \*/

sFilterConfig.FilterBank = 0;

sFilterConfig.FilterMode = CAN\_FILTERMODE\_IDMASK;

sFilterConfig.FilterScale = CAN\_FILTERSCALE\_32BIT;

sFilterConfig.FilterIdHigh = 0x0000;

sFilterConfig.FilterIdLow = 0x0000;

sFilterConfig.FilterMaskIdHigh = 0x0000;

sFilterConfig.FilterMaskIdLow = 0x0000;

sFilterConfig.FilterFIFOAssignment = CAN\_RX\_FIFO0;

sFilterConfig.FilterActivation = ENABLE;

sFilterConfig.SlaveStartFilterBank = 14;

if (HAL\_CAN\_ConfigFilter(&CanHandle, &sFilterConfig) != HAL\_OK)

{

/\* Filter configuration Error \*/

Error\_Handler();

}

/\* Start the CAN peripheral \*/

if (HAL\_CAN\_Start(&CanHandle) != HAL\_OK)

{

/\* Start Error \*/

Error\_Handler();

}

/\* Activate CAN RX notification \*/

HAL\_CAN\_ActivateNotification(&CanHandle, CAN\_IT\_RX\_FIFO0\_MSG\_PENDING);

HAL\_CAN\_ActivateNotification(&CanHandle, CAN\_IT\_RX\_FIFO0\_FULL);

}

这个函数就是在对can总线的过滤器进行配置，使用了掩码模式进行过滤,并开启了两个回调函数，在can.c中重新定义两个回调函数：

void HAL\_CAN\_RxFifo0MsgPendingCallback(CAN\_HandleTypeDef \*hcan)

{

//读取消息

HAL\_CAN\_GetRxMessage(CanHandle, CAN\_RX\_FIFO0, &RxHeader, RxData)；

……

}

void HAL\_CAN\_RxFifo0FullCallback(CAN\_HandleTypeDef \*hcan)

{

//队列满时的处理

}

作业：

1. 完成can总线的配置，和另一个同学组队，一个作为发送端，一个作为接收端，将can\_H和can\_L对应连接，相隔10ms发送一个消息，共发10次，在接收端将接受到的消息通过串口打印出来，检查数据的正确性和完整性。

发送和接收相关函数和结构体请自行查看官方给的例程，例程位置如下：

