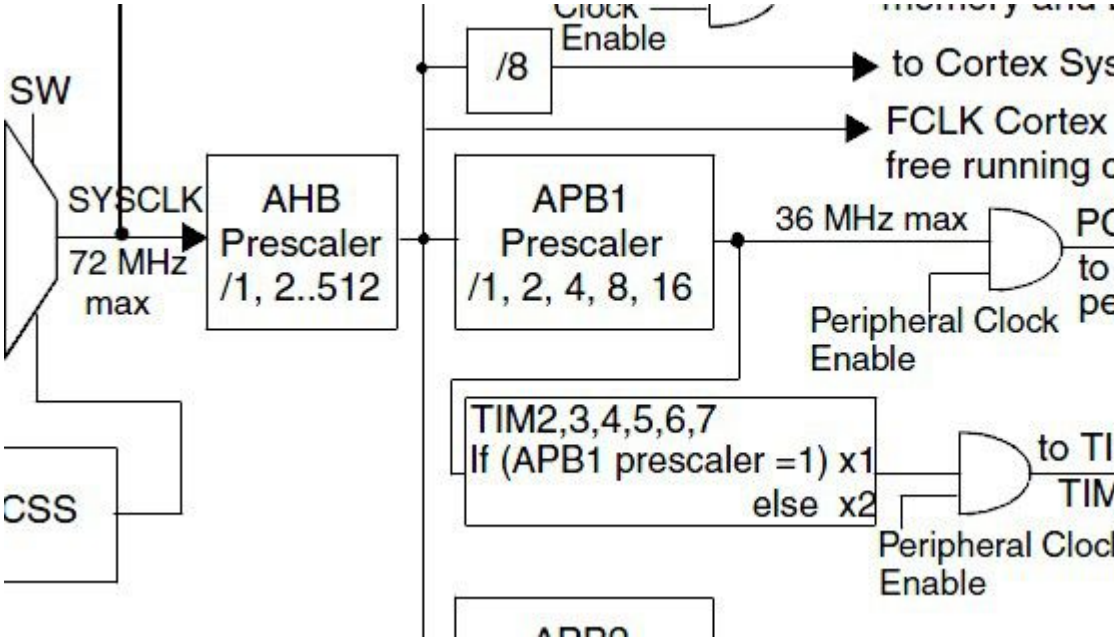
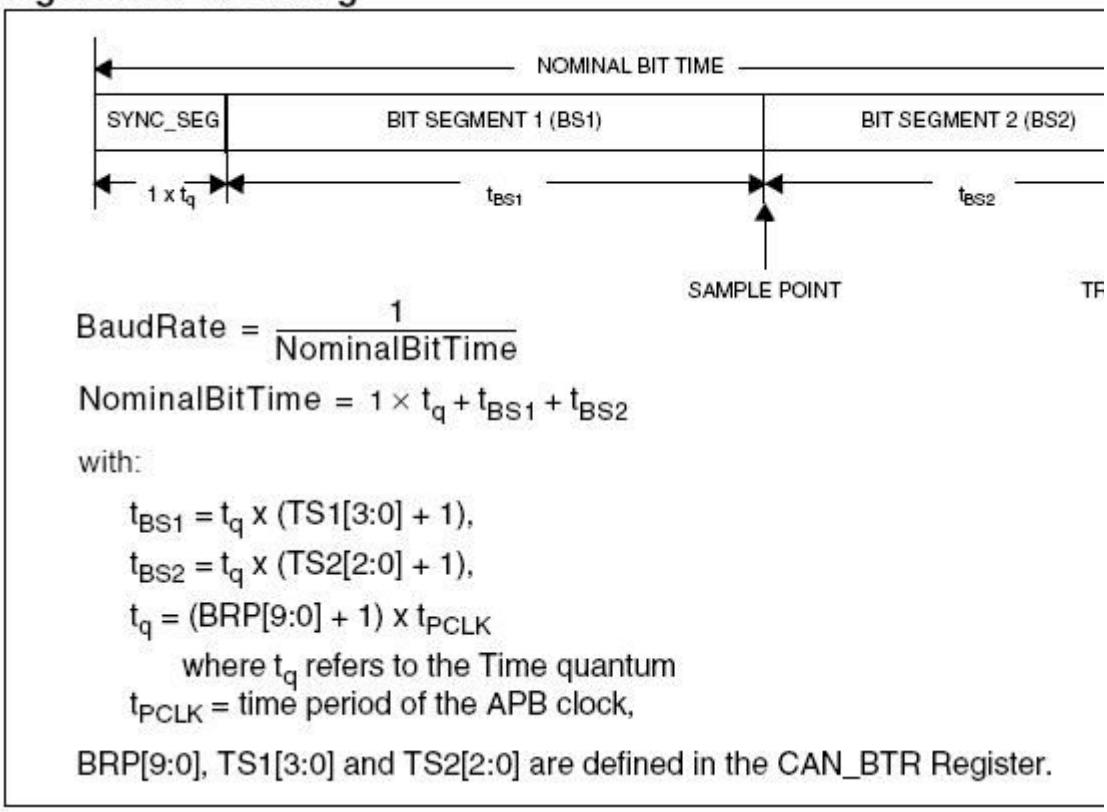
STM32 的 CAN 波特率计算

STM32 里的 CAN 支持 2.0A,2.0B, 带有 FIFO,中断等, 这里主要提一下内部的时钟应用.
bxCAN 挂接在 APB1 总线上,采用总线时钟,所以我们需要知道 APB1 的总线时钟是多少. 我们先看看下图,看看 APB1 总线时钟:



APB1 时钟取自 AHB 的分频, 而 AHB 又取自系统时钟的分频, 系统时钟可选 HSI,HSE, PLLCLK, 这个在例程的 RC 设置里都有的,
然后再看看有了 APB1 的时钟后,如何算 CAN 的总线速率, 先看下图:

Figure 203. Bit timing



有了上边的这个图,基本就清楚了.

总线时钟 MHz (3+TS1+TS2)*(BRP+1)

==========

下面是我的计算：

CAN_InitStructure.CAN_SJW = CAN_SJW_1tq;

CAN_InitStructure.CAN_BS1 = CAN_BS1_3tq;

注意//#define CAN_BS1_3tq ((uint8_t)0x02) /*!< 3 time quantum */

CAN_InitStructure.CAN_BS2 = CAN_BS2_5tq;

CAN_InitStructure.CAN_Prescaler = 4;//2

nominal bit time（3+5+1）tq=9tq

关于分频系数 查看 system_stm32f10x.c 下面的

static void SetSysClockTo72(void) 函数

/* HCLK = SYSCLK */

/* PCLK2 = HCLK */

/* PCLK1 = HCLK/2 */

所以 can 时钟 72MHZ/2/4=9 Mhz

tq=1/36Mhz

波特率为 1/nominal bit time= 9/9=1MHZ

==========

==========

void CAN_Configuration(void)

{

```
CAN_InitTypeDef        CAN_InitStructure;
CAN_FilterInitTypeDef    CAN_FilterInitStructure;

/* CAN register init */
CAN_DeInit();
CAN_StructInit(&CAN_InitStructure);

/* CAN cell init */
CAN_InitStructure.CAN_TTCM=DISABLE;
CAN_InitStructure.CAN_ABOM=DISABLE;
CAN_InitStructure.CAN_AWUM=DISABLE;
CAN_InitStructure.CAN_NART=DISABLE;
CAN_InitStructure.CAN_RFLM=DISABLE;
CAN_InitStructure.CAN_TXFP=DISABLE;
CAN_InitStructure.CAN_Mode=CAN_Mode_Normal;
CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
CAN_InitStructure.CAN_BS1=CAN_BS1_9tq;
CAN_InitStructure.CAN_BS2=CAN_BS2_8tq;
CAN_InitStructure.CAN_Prescaler=200;
CAN_Init(&CAN_InitStructure);

/* CAN filter init */
CAN_FilterInitStructure.CAN_FilterNumber=0;
CAN_FilterInitStructure.CAN_FilterMode=CAN_FilterMode_IdMask;
CAN_FilterInitStructure.CAN_FilterScale=CAN_FilterScale_16bit;
CAN_FilterInitStructure.CAN_FilterIdHigh=0x0000;
CAN_FilterInitStructure.CAN_FilterIdLow=0x0000;
CAN_FilterInitStructure.CAN_FilterMaskIdHigh=0x0000;
CAN_FilterInitStructure.CAN_FilterMaskIdLow=0x0000;
CAN_FilterInitStructure.CAN_FilterFIFOAssignment=0;
CAN_FilterInitStructure.CAN_FilterActivation=ENABLE;
CAN_FilterInit(&CAN_FilterInitStructure);
}
```

注意//#define CAN_BS1_3tq ((uint8_t)0x02) /*!< 3 time quantum */

拨特率 10K，公式：72MHZ/2/200/（1+9+8）=0.01，即 10K，和 SJA1000 测试通过

==========

120 欧姆电阻要加上!!!

哦　确实是

```
CAN->BTR = (u32)((u32)CAN_InitStruct->CAN_Mode << 30) |
((u32)CAN_InitStruct->CAN_SJW << 24) |
        ((u32)CAN_InitStruct->CAN_BS1 << 16) | ((u32)CAN_InitStruct->CAN_BS2 << 20) |
        ((u32)CAN_InitStruct->CAN_Prescaler - 1);
```

总结一下

Fpclk=36M 时 can 波特率为 250k 的配置为

```
/* CAN cell init */
  CAN_InitStructure.CAN_TTCM=DISABLE;
  CAN_InitStructure.CAN_ABOM=DISABLE;
  CAN_InitStructure.CAN_AWUM=DISABLE;
  CAN_InitStructure.CAN_NART=DISABLE;
  CAN_InitStructure.CAN_RFLM=DISABLE;
  CAN_InitStructure.CAN_TXFP=DISABLE;
  CAN_InitStructure.CAN_Mode=CAN_Mode_LoopBack;
  CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
  CAN_InitStructure.CAN_BS1=CAN_BS1_8tq;
  CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
  CAN_InitStructure.CAN_Prescaler=9;
  CAN_Init(&CAN_InitStructure); 250k
==================================
```

的：将 can 总线波特率设置为 250k

在官方的 can 例程上 给出了 100k 查询 和 500k 中断方式的例子 分别设置如下：

CAN_Polling：

```
  /* CAN cell init */
  CAN_InitStructure.CAN_TTCM=DISABLE;
  CAN_InitStructure.CAN_ABOM=DISABLE;
  CAN_InitStructure.CAN_AWUM=DISABLE;
  CAN_InitStructure.CAN_NART=DISABLE;
  CAN_InitStructure.CAN_RFLM=DISABLE;
  CAN_InitStructure.CAN_TXFP=DISABLE;
  CAN_InitStructure.CAN_Mode=CAN_Mode_LoopBack;
  CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
  CAN_InitStructure.CAN_BS1=CAN_BS1_8tq;
  CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
  CAN_InitStructure.CAN_Prescaler=5;
  CAN_Init(&CAN_InitStructure); 100k

  /* CAN cell init */    CAN_Interrupt
  CAN_InitStructure.CAN_TTCM=DISABLE;
  CAN_InitStructure.CAN_ABOM=DISABLE;
  CAN_InitStructure.CAN_AWUM=DISABLE;
  CAN_InitStructure.CAN_NART=DISABLE;
  CAN_InitStructure.CAN_RFLM=DISABLE;
  CAN_InitStructure.CAN_TXFP=DISABLE;
  CAN_InitStructure.CAN_Mode=CAN_Mode_LoopBack;
  CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;
```

```
  CAN_InitStructure.CAN_BS1=CAN_BS1_8tq;
  CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
  CAN_InitStructure.CAN_Prescaler=1;
  CAN_Init(&CAN_InitStructure); //500k
```

can 时钟是 RCC_APB1PeriphClock，你要注意 CAN 时钟频率

CAN 波特率 = RCC_APB1PeriphClock/CAN_SJW+CAN_BS1+CAN_BS2/CAN_Prescaler;

如果 CAN 时钟为 8M， CAN_SJW = 1，CAN_BS1 = 8，CAN_BS2 = 7，CAN_Prescaler = 2

那么波特率就是=8M/(1+8+7)/2=250K

==========

得到 500Kb/s 的波特率

```
CAN_InitStructure.CAN_SJW=CAN_SJW_1tq;

CAN_InitStructure.CAN_BS1=CAN_BS1_8tq;
CAN_InitStructure.CAN_BS2=CAN_BS2_7tq;
CAN_InitStructure.CAN_Prescaler=1;
```

每一位的 Tq 数目 = 1 (固定 SYNC_SEG) + 8 (BS1) + 7 (BS2) = 16

如果 CAN 时钟是 8 MHz : (8M / 1) / 16 = 500K

其中：

1 为分频系数

16 为每一位的 Tq 数目

为了设置为 100K, 把分频系数改为 5 即可, BS1 BS2 不变

每一位的 Tq 数目 = 1 (固定) + 8 (BS1) + 7 (BS2) = 16

如果 CAN 时钟是 8 MHz : (8M / 5) / 16 = 100K

如果想得到 1M 的波特率， CAN 时钟仍然是 8 MHz 的情况下， 分频系数不变

应该改变 BS1 BS2

```
CAN_InitStructure.CAN_BS1=CAN_BS1_5tq;
CAN_InitStructure.CAN_BS2=CAN_BS2_2tq;
```

每一位的 Tq 数目 = 1 (固定) + 5 (BS1) + 2 (BS2) = 8

如果 CAN 时钟是 8 MHz : (8M / 1) / 8 = 1000K

另外尽可能的把采样点设置为 CiA 推荐的值：

75%　　when 波特率 > 800K
80%　　when 波特率 > 500K
87.5%　when 波特率 <= 500K

所以对于 100K 的波特率（假定使用 8MHz 时钟）
可以修改该 BS1 BS2 为：

```
CAN_InitStructure.CAN_Prescaler=5;
CAN_InitStructure.CAN_BS1=CAN_BS1_13tq;
CAN_InitStructure.CAN_BS2=CAN_BS2_2tq;
```

(1+13) / (1+13+2) = 87.5%

所以对于 500K 的波特率（假定使用 8MHz 时钟）
可以修改该 BS1 BS2 为：

```
CAN_InitStructure.CAN_Prescaler=1;
CAN_InitStructure.CAN_BS1=CAN_BS1_13tq;
CAN_InitStructure.CAN_BS2=CAN_BS2_2tq;
```

(1+13) / (1+13+2) = 87.5%

所以对于 1000K 的波特率（假定使用 8MHz 时钟）
可以修改该 BS1 BS2 为：

```
CAN_InitStructure.CAN_Prescaler=1;
CAN_InitStructure.CAN_BS1=CAN_BS1_5tq;
CAN_InitStructure.CAN_BS2=CAN_BS2_2tq;
```

(1+5) / (1+5+2) = 75%

个人见解，仅供参考。
上边这个公式算出来的就是 CAN 的速率了.