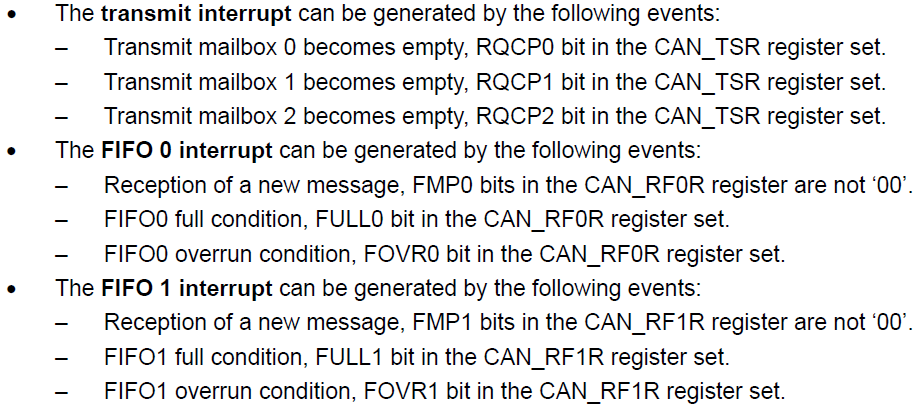
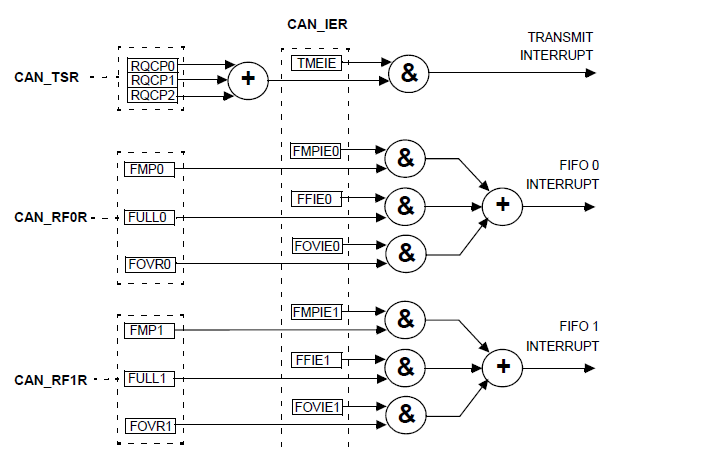
**CAN Interrupts**



**CAN Initialization mode**

**The software initialization can be done while the hardware is in Initialization mode.**

To enter this mode :

the software sets the INRQ bit in the CAN\_MCR register and waits until the hardware has confirmed the request by setting the INAK bit in the CAN\_MSR register.

To leave Initialization mode:

the software clears the INRQ bit. bxCAN has left Initialization mode once the INAK bit has been cleared by hardware.

While in Initialization Mode, all message transfers to and from the CAN bus are stopped and the status of the CAN bus output CANTX is recessive (high).

Entering Initialization Mode does not change any of the configuration registers.

To initialize the CAN Controller, software has to set up the Bit Timing (CAN\_BTR) and CAN

options (CAN\_MCR) registers.

To initialize the registers associated with the CAN filter banks (mode, scale, FIFO

assignment, activation and filter values), software has to set the FINIT bit (CAN\_FMR). Filter

initialization also can be done outside the initialization mode.

*Note: When FINIT=1, CAN reception is deactivated.*

*The filter values also can be modified by deactivating the associated filter activation bits (in*

*the CAN\_FA1R register).*

*If a filter bank is not used, it is recommended to leave it non active (leave the corresponding*

*FACT bit cleared).*

**void can\_enter\_init()**

**{**

**sbi(CAN1->MCR,0); //Enter into initialization mode by setting INRQ in MCR**

**cbi(CAN1->MCR,1); //Exit from Sleep mode by clearing SLEEP bit as its auto SET at Reset**

**sbi(CAN1->MCR,4); //No Retransmit, Msg will be Transmitted Once**

**while(!(cb(CAN1->MSR,0))); //normally it is 0 when it set-->initialization mode**

**while(cb(CAN1->MSR,1)); //in Sleep mode SLAK=1 when SLAK=0-->exit from sleep mode**

**}**

**void can\_exit\_init()**

**{**

**cbi(CAN1->MCR,0); //Enter into normal mode by clearing INRQ bit in MCR**

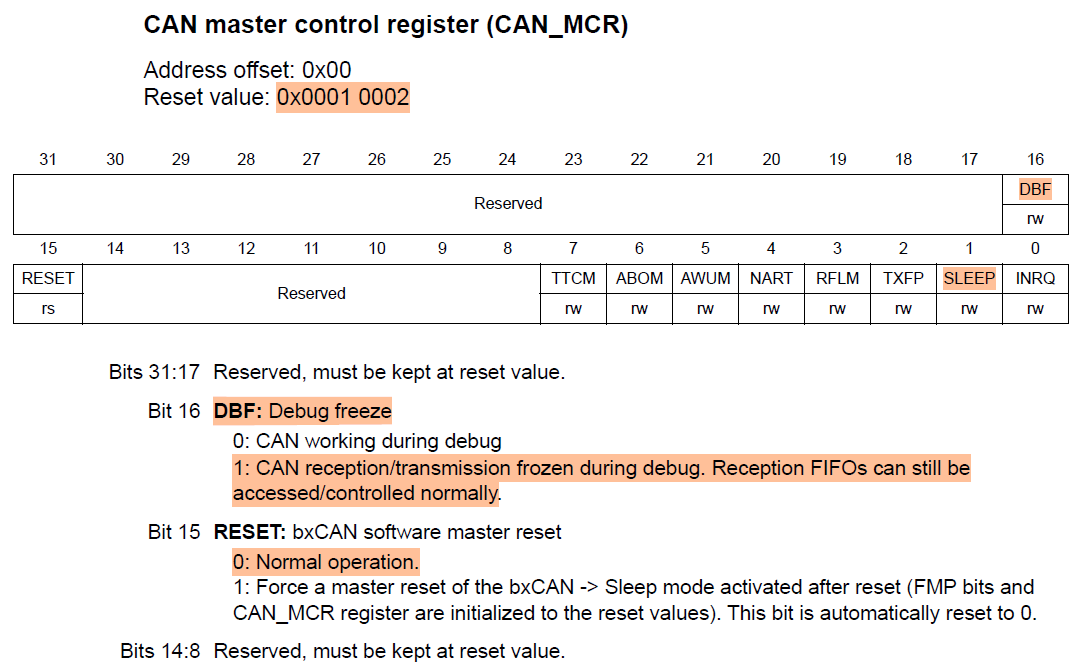
**cbi(CAN1->MCR,1); //Exit from Sleep mode by clearing SLEEP bit**

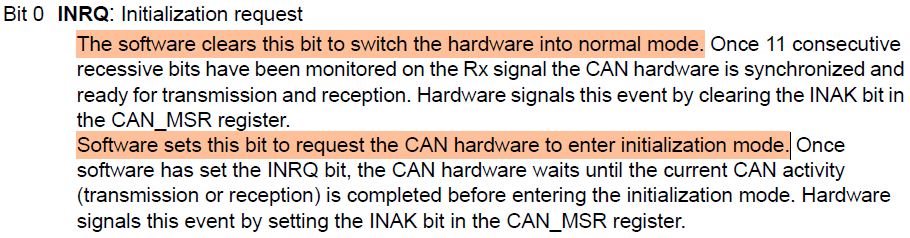
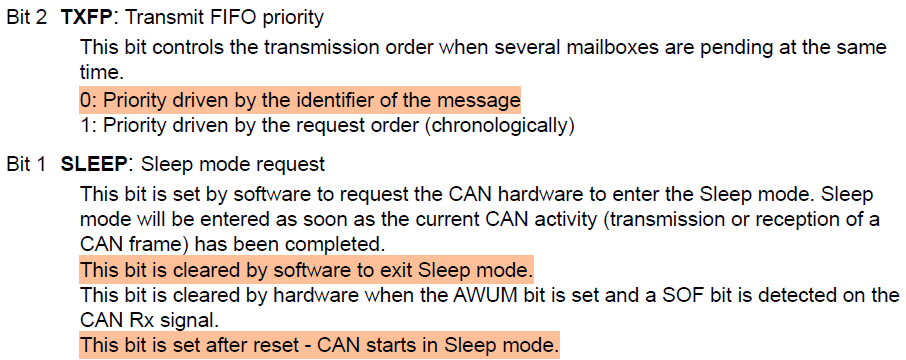
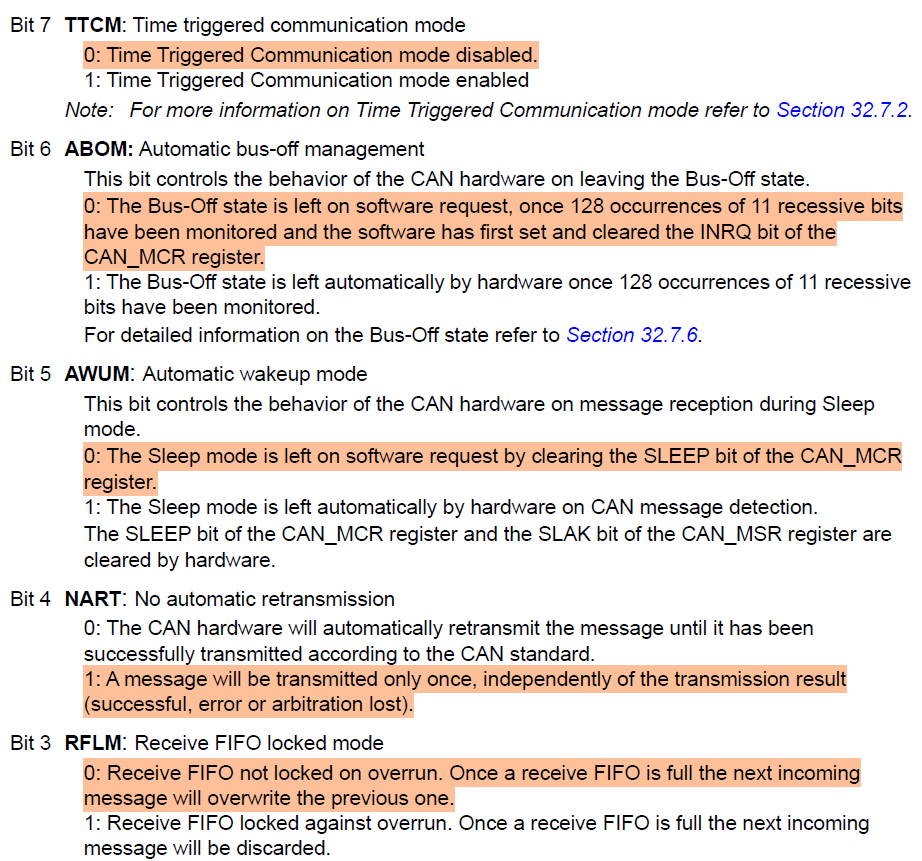
**//After Reseting INRQ bit,CAN Controller waits for 11 consicutive resecive bits**

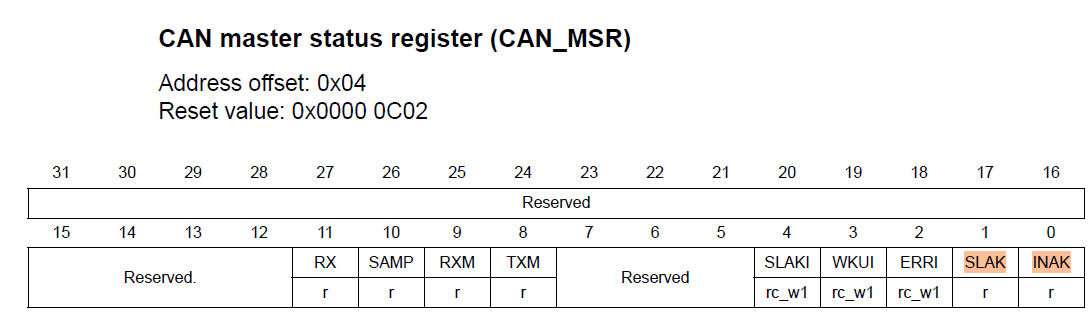
**//After it is synchronized and ready for normal mode**

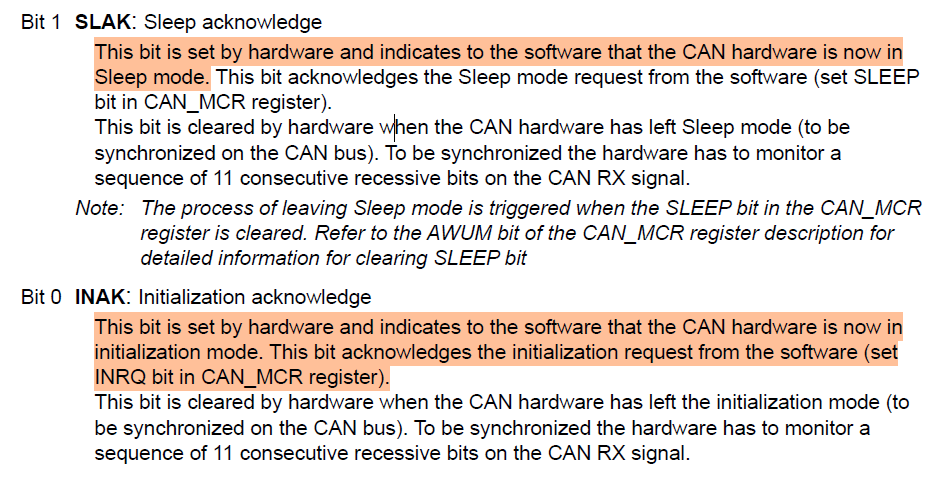
**while((cb(CAN1->MSR,0)) && (cb(CAN1->MSR,1)));**

**}**

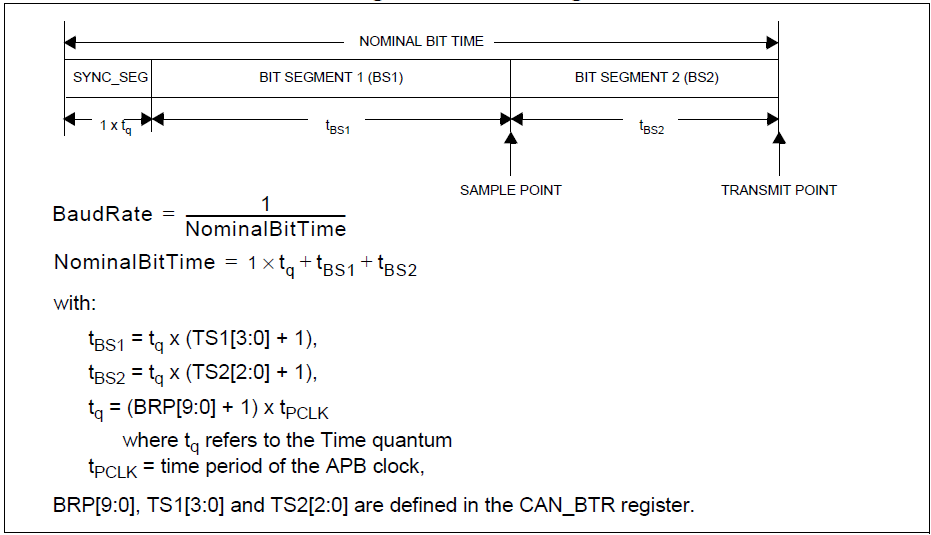








**CAN Bit Timing**

****

**void can\_testmode()**

**{**

**unsigned int brp;**

**can\_setup();**

**can\_enter\_init();**

**brp = (42000000 / 7) / 500000; /\* baudrate is set to 500k bit/s \*/**

**/\* set BTR register so that sample point is at about 71% bit time from bit start \*/**

**/\* TSEG1 = 4, TSEG2 = 2, SJW = 3 => 1 CAN bit = 7 TQ, sample at 71% \*/**

**CAN1->BTR &=~(((0x03) << 24)|((0x07) << 20)|((0x0F) << 16)|(0x3FF));//Clear Bits of BTR**

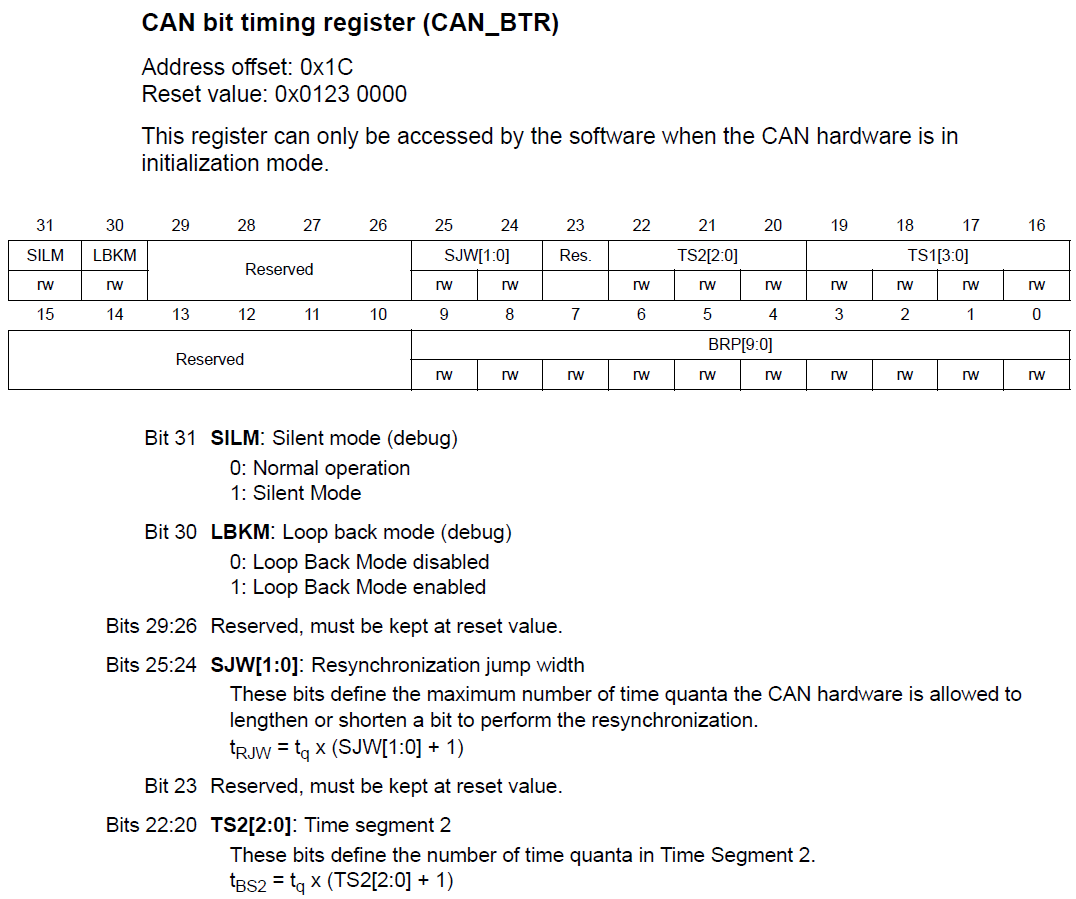
**CAN1->BTR |=((((3-1)&0x03)<<24)|(((2-1)&0x07)<< 20)|(((4-1)&0x0F)<<16)|((brp-1)& 0x3FF));**

**CAN1->BTR &= ~((1<<30)|(1<<31));**

**// CAN1->BTR |= ((1<<30)|(1<<31));// comment it to disable loopback & silent mode**

**can\_exit\_init();**

**}**



Here is my code for setting 1MBit...  I have to say there is usually a slight discrepancy when comparing how other people calculate this, but it has always worked ok for me.

/\* CAN Baudrate = 125Kbps\*/

//Calculation is CAN\_CLK/(BRP\*(BS1+BS2+1)) = 24000000/BRP\*(13+2+1) = 125000,

24000000/BRP\*16=125000

24000000/125000=BRP\*16

24000/125\*16=BRP

3000/125\*2=BRP

1500/125=BRP

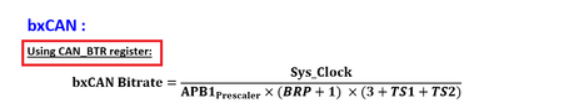
12=BRP

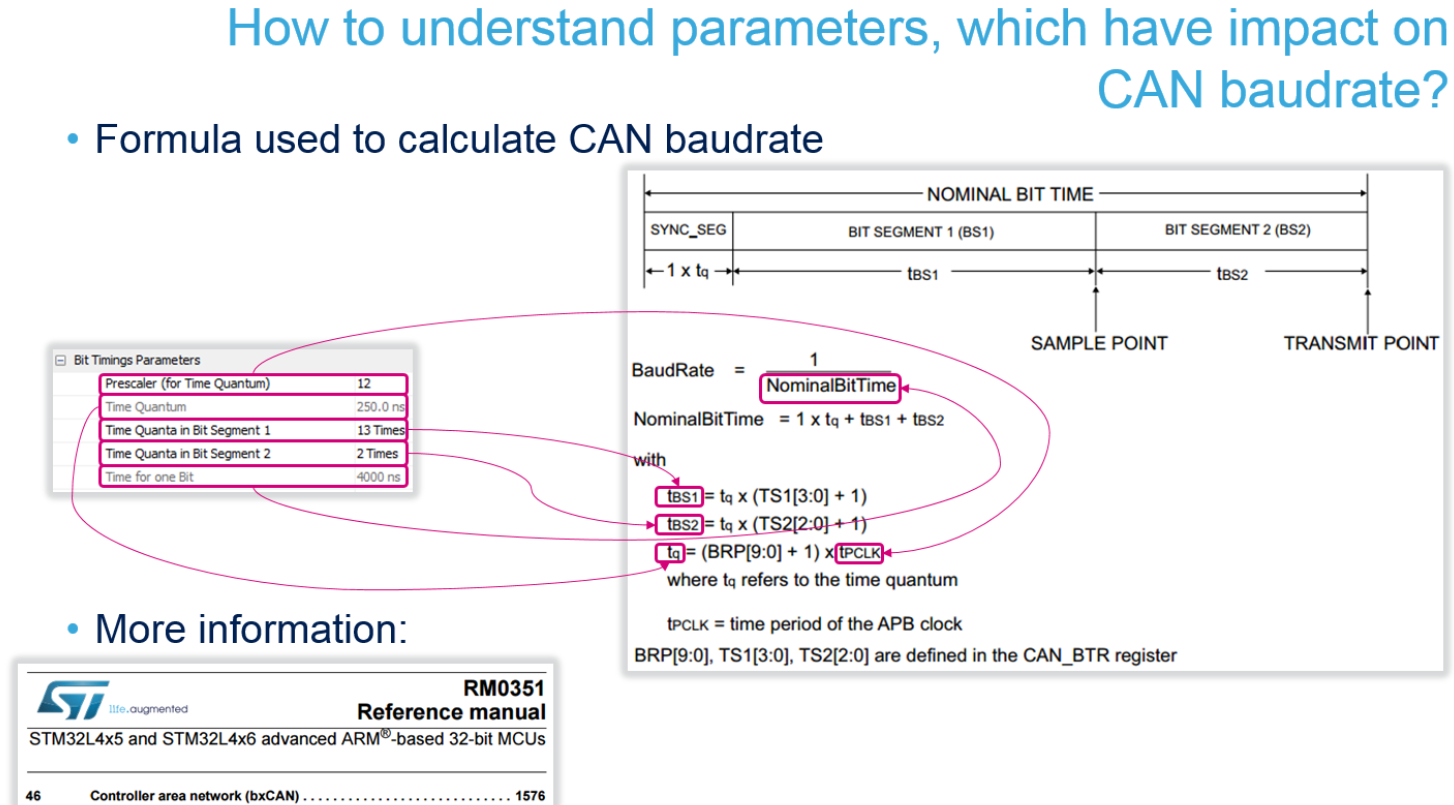
BRP is Prescaler (12)

CANCLK is your APB1 Bus speed (24MHz)

You may struggle with clock of 24MHz, because you end up with a fraction in most Baud rates.

Make BS1 bigger than BS2 to ensure the "bit" is sampled at about 75% (I have 8:1 so 80%).





**/\* TSEG1 = 13, TSEG2 = 2, SJW = 1 => 1 CAN bit = 16 TQ, sample at 71% \*/**

**//brp=12 by above calculation**

**CAN1->BTR &=~(((0x03) << 24)|((0x07) << 20)|((0x0F) << 16)|(0x3FF));//Clear Bits of BTR**

**CAN1->BTR |=((((1-1)&0x03)<<24)|(((2-1)&0x07)<< 20)|(((13-1)&0x0F)<<16)|((brp-1)& 0x3FF));**

**CAN1->BTR &= ~((1<<30)|(1<<31));**