CAN总线报文波特率152380

问题：

CAN总线通讯偶发性死亡（CPU正常工作，CAN网络通讯死亡）

CAN配置代码：

void InitMcuCan2(INT32U Bps)

{

mcan\_frame\_filter\_config\_t CAN\_RxFilter;

mcan\_timing\_config\_t timingConfigA;

INT32U BRP;

INT16U SYSNC\_SEG;

INT16U PHASE\_SEG1;

INT16U PHASE\_SEG2;

INT32U CANPclk;

float f1, f2;

// 设置分频系数，CAN 运行时钟太小可能造成 初始化不成功

// 需要初始化好合适的主频。验证波特率的精度

CLOCK\_SetClkDiv(kCLOCK\_DivCan1Clk, 2U, true); //140/2=70M

CLOCK\_EnableClock(kCLOCK\_Mcan1);

/\* Set INIT bit. \*/

CAN1->CCCR |= CAN\_CCCR\_INIT\_MASK;

/\* Confirm the value has been accepted. \*/

while (!((CAN1->CCCR & CAN\_CCCR\_INIT\_MASK) >> CAN\_CCCR\_INIT\_SHIFT))

{

}

/\* Set CCE bit to have access to the protected configuration registers,

and clear some status registers. \*/

CAN1->CCCR |= CAN\_CCCR\_CCE\_MASK;

// CANPclk = CLOCK\_GetFreq(kCLOCK\_MCAN1);

CANPclk = CLOCK\_GetMCanClkFreq(1U);

if(Bps == 20000)

{

SYSNC\_SEG = 0;

PHASE\_SEG1 = 0x0F; //四位(BTR1 bit0~bit3)

PHASE\_SEG2 = 0x04; //三位(BTR1 bit4~bit6)

}

else if(Bps == 40000)

{

SYSNC\_SEG = 0;

PHASE\_SEG1 = 0x0F;

PHASE\_SEG2 = 0x04;

}

else if(Bps == 57600)

{

SYSNC\_SEG = 0;

PHASE\_SEG1 = 0x0A;

PHASE\_SEG2 = 0x03;

}

else if(Bps == 115200)

{

SYSNC\_SEG = 0; //同步跳转宽度是 + 1

PHASE\_SEG1 = 0x0A; //4位 + 1

PHASE\_SEG2 = 0x03; //3位 + 1

}

else if(Bps == 153600)

{

SYSNC\_SEG = 0;

PHASE\_SEG1 = 0x07;

PHASE\_SEG2 = 0x02;

}

else if(Bps == 230400)

{

SYSNC\_SEG = 0; //同步跳转宽度是 + 1

PHASE\_SEG1 = 0x0A; //4位 + 1

PHASE\_SEG2 = 0x03; //3位 + 1

}

else

{

SYSNC\_SEG = 0;

PHASE\_SEG1 = 4;

PHASE\_SEG2 = 3;

}

#if 1

f1 = Bps;

f2 = (SYSNC\_SEG + PHASE\_SEG1 + PHASE\_SEG2 + 3);

CANPclk /= f1;

CANPclk /= f2;

CANPclk += 0.5;

BRP = (INT16U)CANPclk;

#else

switch(Bps)

{

case 115200:

BRP = 0x4C;

break;

case 20000:

BRP = 0x13E;

break;

case 40000:

BRP = 0x9F;

break;

case 57600:

BRP = 0x98;

break;

case 153600:

BRP = 0x4C;

break;

case 230400:

BRP = 0x26;

break;

}

#endif

timingConfigA.preDivider = (BRP - 1);

timingConfigA.seg1 = PHASE\_SEG1;

timingConfigA.seg2 = PHASE\_SEG2;

timingConfigA.rJumpwidth = SYSNC\_SEG;

/\* Update actual timing characteristic. \*/

MCAN\_SetArbitrationTimingConfig(CAN1, &timingConfigA);

//设置CAN消息基础地址 发送和接收都是用的这个地址

MCAN\_SetMsgRAMBase(CAN1 , SLAVE\_CAN\_MESSAGE\_ADDER);

uint32\_t \*p=(uint32\_t \*)(SLAVE\_CAN\_MESSAGE\_ADDER);

// 清楚内存避免出现错误

memset(p, 0, (8U + CAN\_DATASIZE)\*sizeof(uint8\_t));

CAN\_RxFilter.address = 0x10;

CAN\_RxFilter.idFormat = kMCAN\_FrameIDStandard; //接收扩展帧数据 kMCAN\_FrameIDStandard 0

CAN\_RxFilter.listSize = 2; // FilterNumber

CAN\_RxFilter.nmFrame = kMCAN\_reject0; // 2 (0 1 2 3)

CAN\_RxFilter.remFrame = kMCAN\_rejectFrame; //1 , 0

MCAN\_SetFilterConfig(CAN1 , &CAN\_RxFilter);

CAN1->RXF0C |= (0x10 << 2); //设置FIFO message Ram 中的地址偏移0x20

CAN1->RXF0C |= (1 << 16); //选择FIFO 1个滤波器 ，

CAN1->RXF0C |= (0 << 31); // 0 FIFO 为块工作模式 1：为覆盖模式

CAN1->RXESC |= (0 << 0); //FIFO 为8位数据接收

// 发送初始化

CAN1->TXBC |= (0x50 << 2); //TxBuffer在message Ram 中的地址偏移0x30

CAN1->TXBC |= (0x01 << 16); //专用缓冲数量 1个

CAN1->TXESC |= (0 << 8); // 发送数据为 8位

// 使能中断

// MCAN\_EnableInterrupts(CAN1 , 0 , 0x01); //新消息中断 中断行1 FIFO接收中断

// MCAN\_EnableInterrupts(CAN1 , 0 , 0x04);//接收溢出中断

// MCAN\_EnableInterrupts(CAN1 , 0 , 0x08); //接收数据丢失中断

// MCAN\_EnableInterrupts(CAN1 , 0 , 0x100); //发送完成中断

// MCAN\_EnableInterrupts(CAN1 , 0 , (0x01 << 14)); //Tx event FIFO full interrupt enable

MCAN\_EnableInterrupts(CAN1 , 0 , 0x3FFFFFFF);//开启所有中断 测试用

EnableIRQ(CAN1\_IRQ0\_IRQn);

CAN1->IR = 0x3FFFFFFF; //清楚所有中断标志位

// 初始化can1

MCAN\_EnterNormalMode(CAN1);

}