BAT32G137（库函数版本）

Rev 1.0

修订历史

版本	日期	修订人	修订内容
Rev1.1	22.6.22	缪勤文 张刚	

目录

1. 前言

简易 IIC 功能可在 BAT32G137 系列手册的“通用串行单元(SCI)”查看。SCI 使用了三合一功能，SCI 单元下的通道同一时刻只能复用为 I2C（IIC）、SPI、UART 中的一种功能；对于简易 IIC，每一个简易 IIC（下面称作 IIC）占用一个通道。

2. 通用串行通信单元通道分配

以 BAT32G137 64PIN 为例:

单元	通道	用作SSPI	用作UART	用作简易I²C
0	0	SSPI00 （支持从属选择输入功能）	UART0 TX	IIC00
	1	SSPI01	RX	IIC01
	2	SSPI10	UART1	IIC10
	3	SSPI11		IIC11
1	0	SSPI20	UART2	IIC20
	1	SSPI21		IIC21

对于单元 0 ，SCI 模块分配了 4 个通道，当芯片使用 IIC00 功能，则 UART0 TX 和 SSPI00 无法使用，通道 1 通道 2 通道 3 仍然可以使用;单元 1 也是类似。

3. 中微 BAT32G137 系列简易 IIC 应用库简介

中微 BAT32G137 系类软件简易 IIC 应用库是一个便于移植的标准库代码风格，用户只需要对软件接口相关参数进行简单配置、以及封装接口函数调用即可实现所需功能，节约时间，提高开发效率。应用库提供了基于 IIC 中断读写方式，轮询方式读写。

使用方式：

需要将应用层 iic_demo.c iic_demo.h 驱动层 i2c.c i2c.h、gpio.c gpio.h、sci_common.c sci_common.h、isr.c isr.h 加入到工程中去;

3.1.应用例程使用

包括 IIC 初始化，读写函数以及 IIC 相关接口;简易 IIC 无法用作从机，因此只能做主机。

3.1.1. IIC 初始化

```
/*********************************************
* Function Name: Iic20_Init
* @brief  iic20 init
* @param
* @return error or success
*********************************************/
 int8_t Iic20_Init(void)
{
 int8_t res ;
 GPIO_InitTypeDef GPIO_InitStruct={0};
 I2C_InitTypeDef  IIC_InitStructure;

 GPIO_PinAFConfig(GPIO_PORT1,GPIO_Pin_5,GPIO_P15,GROUP_AF_SCKO20);//SCL20 to any desired pins with PxxCFG register
 GPIO_PinAFConfig(GPIO_PORT1,GPIO_Pin_4,GPIO_P14,GROUP_AF_ODEFAULT);//P14 default function SDA20

 /*SCL20 GPIO CONFIG*/
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_5;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
 GPIO_InitStruct.GPIO_Level  = GPIO_Level_HIGH;
 GPIO_InitStruct.GPIO_OType  = GPIO_OType_OD;
 GPIO_InitStruct.GPIO_Ctrl = GPIO_Control_DIG;
 GPIO_Init(GPIO_PORT1,&GPIO_InitStruct);

 /*SDA20 GPIO CONFIG*/
 GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4;
 GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
 GPIO_InitStruct.GPIO_Level  = GPIO_Level_HIGH;
 GPIO_InitStruct.GPIO_OType  = GPIO_OType_OD;
 GPIO_InitStruct.GPIO_Ctrl = GPIO_Control_DIG;
 GPIO_Init(GPIO_PORT1,&GPIO_InitStruct);
 IIC_InitStructure.I2C_ClockSpeed = 80000;  //配置 iic 速度

 res = I2C_Init(I2C20, &IIC_InitStructure);
 if(res)
```

```
35.  {
36.   SCI_ERROR_LOG(res);
37.   return res;
38.  }
39.
40.  ISR_Register(IIC20_IRQn,iic20_interrupt);     //中断服务路径注册
41.
42.  return SCI_SUCCESS;
43. }
```

- GPIO 复用，调用 GPIO_PinAFConfig 函数，其可以将普通 GPIO 引脚复用为数字功能，对于 IIC20 的 SCLK 引脚，由数字功能可知可以映射到任意管脚；IIC20 的 SDA 引脚由芯片引脚图可知，其是 P14 默认功能；
- 配置 IIC 运行速率
- 中断服务函数注册，中断收发，将 iic20_interrupt 注册到中断号为 IIC20_IRQn 的中断函数函数中去，若不使用中断方式进行收发则不用注册中断服务函数

3.1.2. IIC 轮询收/发函数

对于简易 IIC 接口收发函数，对于轮询方式来说，其无法进行 ACK 检测，默认其能收到从机响应：

发送函数：I2C_WriteData

```
1.  /**
2.    * @brief  Send a data buffer through the I2Cx peripheral to write.
3.    * @param  I2Cx: where x can be 1, 2 or 3 to select the I2C periphe
    ral.
4.    * @param  Address: Device address in the I2C bus.
5.    * @param  Reg: Register address in the Device.
6.    * @param  Data: Data buffer address.
7.    * @param  Len: Data buffer length need to transmit.
8.    * @retval None
9.    */
10. void I2C_WriteData(SCIAFSelect_TypeDef I2Cx, uint8_t Address, uint8_
    t Reg, uint8_t *Data, uint16_t Len)
```

轮询发送，选择使用的 IIC，IIC 从机地址，从机寄存器地址，数据

接收函数 I2C_ReadData

```
/**
  * @brief  Read a data buffer through the I2Cx peripheral to device register address.
  * @param  I2Cx: where x can be 1, 2 or 3 to select the I2C peripheral.
  * @param  Address: Device address in the I2C bus.
  * @param  Reg: Register address in the Device.
  * @param  Data: Data buffer address.
  * @param  Len: Data buffer length need to transmit read.
  * @retval None
  */
void I2C_ReadData(SCIAFSelect_TypeDef I2Cx, uint8_t Address, uint8_t Reg, uint8_t *Data, uint16_t Len)
{
    IRQn_Type irq;

    /* Check the parameters */
    assert_param(IS_I2C_ALL_PERIPH(I2Cx));

    /* Get the IQR number depend on the I2Cx in SCI unit */
    irq = I2C_IRQTable[((I2Cx >> 13) * 4) + (I2Cx & 0x0F)];

    /* When the I2C bus start,we should set the bus register mode in Send */
  I2C_Set_TransmitMode(I2Cx, I2C_TransmitMode_Send);

    /* Generate a START signal to transmission */
  I2C_GenerateSTART(I2Cx);

    /* Device address send */
  I2C_SendByte(I2Cx, Address & 0xFE);

    /* Wait device address send succes */
  while(!INTC_GetPendingIRQ(irq));
  INTC_ClearPendingIRQ(irq);

    /* Register address send */
  I2C_SendByte(I2Cx, Reg);

    /* Wait register address send succes */
  while(!INTC_GetPendingIRQ(irq));
    INTC_ClearPendingIRQ(irq);

```

```
40.  /* I2C bus restart and mode configurated by receive mode */
41.     I2C_GenerateSTOP(I2Cx);
42.     I2C_Set_TransmitMode(I2Cx, I2C_TransmitMode_Recv);
43.     I2C_GenerateSTART(I2Cx);
44.
45.  /* Send device address retry and sign the WR bit to read */
46.  I2C_SendByte(I2Cx, Address | 0x01);
47.  while(!INTC_GetPendingIRQ(irq));
48.  INTC_ClearPendingIRQ(irq);
49.
50.     /* Start to receive data to the dest memory buffer */
51.  do {
52.   /* When the last byte to receive, we should not output ACK */
53.   if(Len == 1U)
54.    I2C_Output_Cmd(I2Cx, DISABLE);
55.
56.   /* Write the virtual data to SIO in order to start receive data */
57.   I2C_SendByte(I2Cx, 0xFFU);
58.
59.   /* Wait the virtual data send success */
60.   while(!INTC_GetPendingIRQ(IIC20_IRQn));
61.   INTC_ClearPendingIRQ(IIC20_IRQn);
62.
63.   /* Read the receive data from SIO register to target memory buffer
      */
64.   *Data++ = I2C_ReceiveByte(I2Cx);
65.
66.  }while(--Len);
67.
68.  /* Generate STOP signal to the read flow completed */
69.     I2C_GenerateSTOP(I2Cx);
70. }
```

读取从机设备相对应的寄存器，到指定 rxbuf 中

3.1.3. IIC 中断收/发函数

对于 BAT32G137 系列的 IIC，中断收发函数，在中断服务函数中进行数据处理；

中断发送函数 Iic20_Write

```
1.  /*****************************************************
2.  * Function Name: Iic_Write
```

```
3.  * @brief  This function starts transferring data for IIC in master m
    ode.
4.  * @param  adr - set address for select slave
5.  * @param  tx_buf - transfer buffer pointer
6.  * @param  tx_num - buffer size
7.  * @return None
8.  ***********************************************/
9.  void Iic20_Write(uint8_t address, uint8_t *tx_buf, uint16_t tx_num)
10. {
11.  pData.flag = INT_IDLE;
12.  address &= 0xFE; //iic write flag
13.
14.  I2C_Set_TransmitMode(I2C20, I2C_TransmitMode_Send);
15.  I2C_GenerateSTART(I2C20);
16.
17.  INTC_ClearPendingIRQ(IIC20_IRQn); // clear INTIIC01 interrupt flag
18.  NVIC_ClearPendingIRQ(IIC20_IRQn); // clear INTIIC01 interrupt flag

19.  INTC_EnableIRQ(IIC20_IRQn);  // enable INTIIC01 interrupt flag
20.  NVIC_SetPriority(IIC20_IRQn, 3);  /* low priority */
21.
22.  I2C_SendByte(I2C20, address);
23.
24.  g_iic_tx_end = 0;
25.  pData.flag = INT_TX;
26.  pData.data = tx_buf;    //send buffer pointer
27.  pData.len  = tx_num;    //set send data count
28. }
```

对于中断发送函数，其功能在产生起始信号后，将数据给到中断服务函数，通过发送中断服务发送出去，在中断中处理检查应答信号。

中断接收函数 Iic20_Read

```
1.  /*******************************************************
2.  * Function Name: Iic_Read
3.  * @brief  This function starts receiving data for IIC in master mode
    .
4.  * @param  adr - set address for select slave
5.  * @param  rx_buf - receive buffer pointer
6.  * @param  rx_num - buffer size
7.  * @return None
8.  *******************************************************/
9.  void Iic20_Read(uint8_t address, uint8_t *rx_buf, uint16_t rx_num)
10. {
```

```
11.  pData.flag = INT_IDLE;
12.  address |= 0x01; //iic read flag
13.
14.  I2C_Set_TransmitMode(I2C20, I2C_TransmitMode_Send);
15.  I2C_GenerateSTART(I2C20);
16.
17.  INTC_ClearPendingIRQ(IIC20_IRQn); // clear INTIIC01 interrupt flag
18.  NVIC_ClearPendingIRQ(IIC20_IRQn); // clear INTIIC01 interrupt flag

19.  INTC_EnableIRQ(IIC20_IRQn);  // enable INTIIC01 interrupt flag
20.
21.  I2C_SendByte(I2C20, address);
22.
23.  g_iic_rx_end = 0;
24.  g_iic_rx_count = 0;
25.  pData.flag = INT_RX;
26.  pData.data = rx_buf;    //send buffer pointer
27.  pData.len  = rx_num;    //set send data count
28. }
```

对于 IIC 中断接收函数：在产生起始信号后，发送完设备地址之后，将接收 buf 传送到中断服务函数，在中断服务函数中接收数据。

3.1.5. IIC 开始/关闭运行

```
1.  /**
2.    * @brief  Set I2Cx bus output channel start or stop
3.    * @param  I2Cx: where x can be 0, 1, 2, 3, 4 or 5 to select the I2
    C peripheral.
4.    * @param  NewState: I2Cx bus signal output status.
5.    *          This parameter can be one of the following values:
6.    *            @arg ENABLE: I2Cx bus channle start enable.
7.    *            @arg DISABLE: I2Cx bus channle stop
8.    * @retval None.
9.    */
10. void I2C_ChannelStart_Cmd(SCIAFSelect_TypeDef I2Cx, FunctionalState
    NewState)
```

ENABLE：IIC 通道开始使能

DISABLE：IIC 通道停止运行

3.1.6. IIC 获取运行状态

```
1.  FlagStatus I2C_GetErrStaus(SCIAFSelect_TypeDef I2Cx, uint16_t I2C_FL
    AG)
```

根据 I2C_FLAG 获取 UART 指定的接收状态

3.1.7. 关闭 IIC 外设

```
1.  void I2C_DeInit(SCIAFSelect_TypeDef UARTx)
```

将 I2C 外设关闭，包括关闭其运行时钟，释放 IIC 所占用通道

3.3. IIC 驱动 EEPROM AT24C02 例程示例

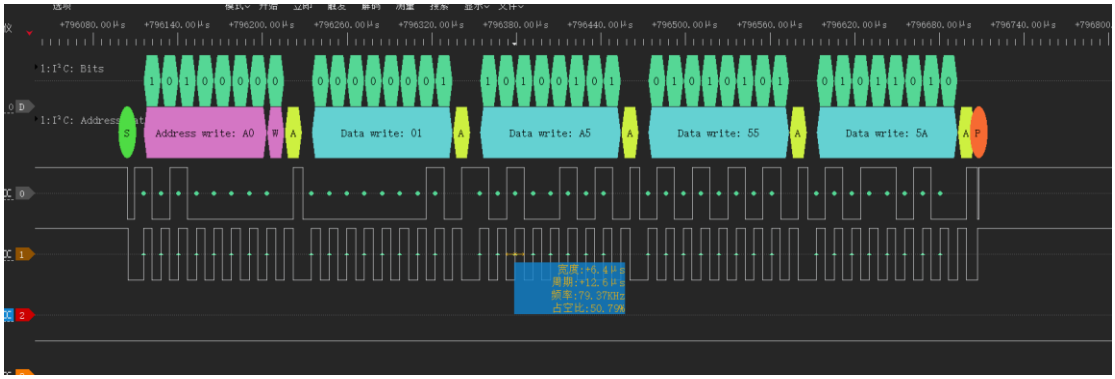
在 main 函数中演示驱动 AT24C02 驱动程序，向 AT24C02 设备写数据，同时将所写的数据读取出来；使用了轮询方式和中断读取 2 种方式；

```
45      uint8_t tx_buf[3]={0xA5,0x55,0x5A};
46      uint8_t tx_buf1[4]={0x01,0xA5,0x55,0x5A};
47      uint32_t err;
48      uint32_t i;
49  //-----------------------------------------------------------------------
50  // Systick setting
51  //-----------------------------------------------------------------------
52      SystemCoreClockUpdate();
53      msCnt = SystemCoreClock / 1000;
54      SysTick_Config(msCnt);
55      delay_init(SystemCoreClock); //延时初始化
56
57      Uart0_Init(19200);
58      printf("IIC\n");
59      Iic20_Init();
60  #ifdef USING_24C02
61      #if 0/*offer two ways to driver 24C02*/
62      I2C_WriteData(I2C20, SLVADDR, 0x01, tx_buf,3);
63      delayMS(200);
64      I2C_ReadData(I2C20,SLVADDR,0x01,rx_buf,3);
65      #else
66      Iic20_Write(SLVADDR,tx_buf1,1);
67      while(g_iic_tx_end == 0);
68      printf("IIC Send ok\n");
69      delayMS(200);
70
71      Iic20_Write(SLVADDR,tx_buf1,1);
72      while(g_iic_tx_end == 0);
73      Iic20_Read(SLVADDR,rx_buf, 3);
74      while(g_iic_rx_end == 0);
75      #endif
76  #else
77      Iic20_Write(SLVADDR,tx_buf,3);
```

通过定义宏选择使用方式。

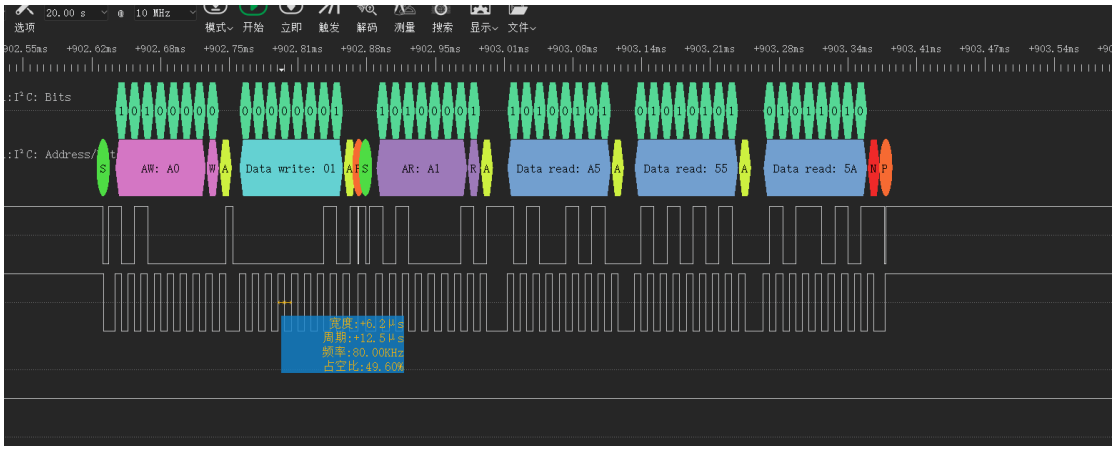
4. 示例演示

IIC 作为主机，读写 24C02



图一 写时序

注释：24C02 设备地址为 0XA0，向寄存器 0X01 地址写 3 位数据 0XA5,0X55,0X5A;



图二 读时序

注释：在读时候，先要写设备地址，然后从设备地址读取 0X01 寄存器地址读取 3 位数据