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 SSPI 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 IIC、SPI、UART 中的一种功能；对于简易 SSPI 模块，每一个 SSPI 占用一个通道。

2. 通用串行单元通道分配

以 BAT32G137 48PIN 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 SSPI00 功能，则 UART0 TX 和 IIC00 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 SSPI 应用库简介

中微 BAT32G137 系类软件 SSPI 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 SSPI 多种收发方式：轮询、DMA、中断收发。

使用方式：

需要将应用层 spi_demo.c spi_demo.h 驱动层 sspi.c sspi.h、gpio.c gpio.h、sci_common.c sci_common.h、isr.c isr.h 加入到工程中去；若搭配使用 DMA，则需要将相应 DMA 驱动文件以及 dma_demo 程序加入;

3.1.应用例程使用

包括 SSPI 初始化，读写函数以及 SSPI 相关接口

3.1.1. SSPI 初始化

```
1.int8_t Spi11_Init(void)
2.{
3. int8_t res;
4. GPIO_InitTypeDef GPIO_InitStruct={0};
5. SPI_InitTypeDef  SPI_InitStructure;
6.
7. GPIO_PinAFConfig(GPIO_PORT6,GPIO_Pin_2,GPIO_P62,GROUP_AF_ODEFAULT);
//SS10 can be disired to any pins
8. GPIO_PinAFConfig(GPIO_PORT1,GPIO_Pin_0,GPIO_P10,GROUP_AF_ODEFAULT);/
/ SCLK11_OUTPUT default fuction with P10
9. GPIO_PinAFConfig(GPIO_PORT1,GPIO_Pin_2,GPIO_P12,GROUP_AF_ODEFAULT);/
/ MOSI(SDO11) default fuction with P12
10.  GPIO_PinAFConfig(GPIO_PORT1,GPIO_Pin_1,GPIO_P11,GROUP_AF_ODEFAULT)
;// MSIO(SDI11) default fuction with P11
11.
12.  /*SS GPIO CONFIG*/
13.  GPIO_InitStruct.GPIO_Pin  = GPIO_Pin_2;
14.  GPIO_InitStruct.GPIO_Mode  = GPIO_Mode_OUT;
15.  GPIO_InitStruct.GPIO_Ctrl  = GPIO_Control_DIG;
16.  GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
17.  GPIO_InitStruct.GPIO_Level = GPIO_Level_HIGH;
18.  GPIO_Init(GPIO_PORT6,&GPIO_InitStruct);
19.
20.  /*SCLK GPIO CONFIG*/
21.  GPIO_InitStruct.GPIO_Pin  = GPIO_Pin_0;
22.  GPIO_InitStruct.GPIO_Mode  = GPIO_Mode_OUT;
23.  GPIO_InitStruct.GPIO_Ctrl  = GPIO_Control_DIG;
24.  GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
25.  GPIO_InitStruct.GPIO_Level = GPIO_Level_HIGH;
26.  GPIO_Init(GPIO_PORT1,&GPIO_InitStruct);
27.
28.  /*SDO(MOSI) GPIO CONFIG*/
29.  GPIO_InitStruct.GPIO_Pin  = GPIO_Pin_2;
30.  GPIO_InitStruct.GPIO_Mode  = GPIO_Mode_OUT;
31.  GPIO_InitStruct.GPIO_Ctrl  = GPIO_Control_DIG;
32.  GPIO_InitStruct.GPIO_OType   = GPIO_OType_PP;
33.  GPIO_InitStruct.GPIO_Level   = GPIO_Level_HIGH;
34.  GPIO_Init(GPIO_PORT1,&GPIO_InitStruct);
```

```
35.
36.   /*SDI(MIS0) GPIO CONFIG*/
37.   GPIO_InitStruct.GPIO_Pin = GPIO_Pin_1;
38.   GPIO_InitStruct.GPIO_Mode = GPIO_Mode_IN;
39.   GPIO_InitStruct.GPIO_Ctrl = GPIO_Control_DIG;
40.   GPIO_Init(GPIO_PORT1,&GPIO_InitStruct);
41.
42.   SPI_InitStructure.SPI_Mode = SPI_Mode_Master;        //设置SPI 工作
模式:设置为主SPI 从模式SLAVE
43.   SPI_InitStructure.SPI_DataSize = SPI_Data_Bits_8;    //设置SPI 的数
据大小:SPI 发送接收8 位帧结构
44.   SPI_InitStructure.SPI_Phase_Mode= SPI_Phase_Mode0;    //选择使用spi
的工作模式
45.   SPI_InitStructure.SPI_ClockSpeed= 2000000;            //选择spi 运行
时钟
46.   SPI_InitStructure.SPI_Bitorder = SPI_Bit_Msb;              //选择bit
高先传
47.
48.   res = SSPI_Init(SSPI11,&SPI_InitStructure);
49.   if(res)
50.   {
51.    return res;
52.   }
53.   ISR_Register(SPI11_IRQn, spi11_interrupt); //中断服务路径注册
54.
55.   NSS.Active = NSS_clr;                                  //NSS 片选初始化
56.   NSS.Inactive = NSS_set;
57.   NSS.Inactive();                                       //NSS 初始化空闲状
态 MODE0 情况下低电平为空闲，在clk 第一个边沿采样
58.   return SCI_SUCCESS;
59. }
```

- 配置 GPIO 复用函数,对于 SSPI11 默认引脚位：P10(SCLK) P11(MISO) P12(MOSI)，片选可以选择任一引脚，在此我们使用 P62；
- 配置 NSS, CLK ,MOSI,MISO 引脚配置
- 配置主从模式
- 配置 SPI 传输数据大小
- 配置 SPI 工作模式：（SPI_MODE_0）
- 设置 SPI 运行速度 2MHz
- 配置高先传

- 中断服务函数注册
- 若使用片选信号时，将 NSS 电平拉高拉低注册到 NSS.Active NSS.Inactive 函数中便于操作

3.1.2. SSPI 轮询收/发函数

对于 SSPI 接口读写函数,SSPI 接口提供了 3 种读写方式实现 SSPI 数据传输；第一种为轮询方式：

发送函数：SSPI_TransmitData

```
1./**
2.  * @brief  Transmits one Byte Data through the SSPI.
3.  * @param  Data: Data to be transmitted.
4.  * @retval tmp: Receive the data by transmit.
5.  */
6.uint8_t SSPI_TransmitData(SCIAFSelect_TypeDef SSPIx, uint8_t Data)
7.
```

轮询发送，等待单字节数据发送成功之后才会继续发送下一个数据

接收函数 SSPI_ReceiveData (SCIAFSelect_TypeDef SSPIx)

等待 SSPI 接收缓冲区有数据时，将数据读取出来

```
1.  /**
2.    * @brief  Transmits one Byte Data through the SPI peripheral.
3.    * @param  Data: Data to be transmitted.
4.    * @retval tmp: Receive the data by transmit.
5.    */
6.  uint8_t SSPI_ReceiveData(SCIAFSelect_TypeDef SSPIx)
```

3.1.3. SSPI 中断收/发函数

对于 BAT32G137 系列的 SSPI，中断收发函数，在中断服务函数中发送或者获取数据；

中断发送函数 Spi11_Int_Write (uint8_t *tx_buf,uint16_t tx_num)

```
1./*************************************************************
2.* Function Name: Spi_Int_Write
3.* @brief  spi write data by interrupt,it can be used in the condition
 of master write or slave write
```

```
4.* @param  None
5.* @return None
6.*******/
7.void Spi11_Int_Write(uint8_t *tx_buf,uint16_t tx_num)
8.{
9.   SSPI_Set_TransmitMode(SSPI11, SSPI_TransmitMode_Send);
     SSPI_Start(SSPI11);
10.    INTC_EnableIRQ(SPI11_IRQn);
11.
12.    g_spi_tx_end = 0;
13.    g_spi_rx_end = 0;
14.    pData.data = tx_buf;    //send buffer pointer
15.    pData.len  = tx_num;  //set send data count
16.    pData.flag = INT_TX;   //set data direction
17.    INTC_SetPendingIRQ(SPI11_IRQn);
18.   }
```

- 使用 SPI 中断发送时候需要将模式设置为只发送模式
- 打开 SSPI 所在通道调用 SSPI_Start(SSPI11)
- 将数据送到中断服务函数中发送出去

对于中断发送函数，其功能只是将数据送到中断服务函数中发送出去，不能同时读取 MISO 数据；

中断接收函数 Spi11_Int_Read(uint8_t *rx_buf,uint16_t rx_num)

```
1. /*************************************************************
2. * Function Name: Spi_Int_Read
3. * @brief  spi read data by interrupt,it can be used in the condition
    of master write or slave write
4. * @param  None
5. * @return None
6. *************************************************************/
7. void Spi11_Int_Read(uint8_t *rx_buf,uint16_t rx_num)
8. {
9.
10. SSPI_Set_TransmitMode(SSPI11, SSPI_TransmitMode_Recv);
11. SSPI_Start(SSPI11);
12. INTC_EnableIRQ(SPI11_IRQn);
13.
14. pData.data = rx_buf;   //read buffer pointer
15. pData.len  = rx_num;   //set receive data count
16. pData.flag = INT_RX;   //set data direction
17.
```

```
18. SSPI_SendByte(SSPI11,0xFF);
19.}
```

对于中断接收函数来说，首先需要将模式设置为只读模式，在中断服务函数中接收数据。

注意：往往在实际应用过程中，SSPI 读取数据时：一般先写数据，在写哑字时同时读取数据；这时候需要结合轮询发送函数或者中断发送函数，等待发送字节完成再使用中断接收函数；

3.1.4. SSPI DMA 收/发函数

对于 BAT32G137 系列的 SSPI，DMA 收发函数，在通过 DMA 将要发送或者获取数据通过 DMA 搬移到指定 buf 中去。

DMA 发送函数 Spihs_Dma_Write(uint8_t *tx_buf,uint16_t tx_num)

```
1. /*****************************************************************
2. * Function Name: Spi11_Dma_Write
3. * @brief  spi write data by dma
4. * @param  None
5. * @return None
6. *****************************************************************/
7. void Spi11_Dma_Write(uint8_t *tx_buf,uint16_t tx_num)
8. {
9.
10. SSPI_Set_TransmitMode(SSPI11, SSPI_TransmitMode_Send);  //使能写标志位
11. SSPI_Start(SSPI11);
12.
13. INTC_EnableIRQ(SPI11_IRQn);
14.
15. DMA_Sspi_Write(DMA_VECTOR_SPI11,CTRL_DATA_SPI11,DMA_Mode_Normal,tx_buf,(void *)&SSPI11_SDR,tx_num); //config dma transmission
16. DMA_Trigger(DMA_VECTOR_SPI11);
17. pData.flag = INT_DMA;   //set data direction
18.
19.}
```

对于 DMA 发送函数，其功能将数据通过 DMA 搬移到 SSPI11_SDR 发送缓冲器中，但是不能同时读取接收缓冲区数据。

DMA 接收函数 Spi11_Dma_Read (uint8_t *rx_buf,uint16_t rx_num)

```
/*****************************************************************
* Function Name: Spi11_Dma_Read
* @brief  spi read data by dma
* @param  None
* @return None
*****************************************************************
**/
void Spi11_Dma_Read(uint8_t *rx_buf,uint16_t rx_num)
{
 static uint8_t dummy_sio11 = 0xFFU;

SSPI_Set_TransmitMode(SSPI11, SSPI_TransmitMode_Recv);

SSPI_Start(SSPI11);
DMA_Sspi_Read(DMA_VECTOR_SPI11,CTRL_DATA_SPI11,DMA_Mode_Normal,(void *)&SSPI11_SDR,rx_buf,rx_num-1); //config dma transmission
pData.flag = INT_RX;   //set data direction
pData.len  = 1;   //set data direction
pData.data = rx_buf+ rx_num - 1;
SSPI_SendByte(SSPI11,dummy_sio11);//interrupt triger the dma

}
```

对于 DMA 接收函数来说，首先需要将模式设置为读模式，在通过 DMA 将 SSPI11_SDR 接收缓冲区的数据读取到 rxbuf 中。

注意：同样的，往往在实际应用过程中，SPI 读取数据时：一般先写数据，在写数据时同时读取数据。

DMA 收发函数 Spi11_Dma_TxRx

```
void Spi11_Dma_TxRx(uint8_t *tx_buf, uint16_t tx_num, uint8_t *rx_buf, uint16_t rx_num)
{
 SSPI_Set_TransmitMode(SSPI11, SSPI_TransmitMode_TxRx);

 SSPI_Start(SSPI11);
 g_spi_tx_end = 0;
 g_spi_rx_end = 0;

 DMA_Sspi_WriteRead(DMA_VECTOR_SPI11, CTRL_DATA_SPI11, DMA_Mode_Normal,tx_buf+1, rx_buf, tx_num-1); //config dma transmission
pData.flag = INT_DMA;   //set data direction
pData.data = &rx_buf[rx_num-1];
SSPI_SendByte(SSPI11, tx_buf[0]); //interrupt triger the dma
```

```
13.}
```

对于 DMA 收发方式，主要是针对 SPI 读取数据，调用 DMA_Sspi_WriteRead 函数，在 DMA 写数据时，同时用 DMA 方式从接收缓冲区中读取数据到 rxbuf 中；注意在使用 DMA 收发时需要将模式改为 TxRx；这种方式更贴切于实际使用。

3.1.5. SSPI 通道开始/关闭运行

```
1. void SSPI_Start(SCIAFSelect_TypeDef SSPIx)
```

打开 SSPI 所在运行通道允许输出

```
2. void SSPI_Stop(SCIAFSelect_TypeDef SSPIx)
```

停止 SPI 运行，运行通道关闭

3.1.6. 关闭 SSPI 外设

```
1. void SSPI_DeInit(SCIAFSelect_TypeDef SSPIx)
```

关闭 SSPI 外设，包括运行时钟以及所占用通道

3.1.7. 配置 SSPI 工作模式

```
1. /**
2.   * @brief  Set SSPIx communication mode in send or receive.
3.   * @param  SSPIx: where x can be 0, 1, 2, 3, 4 or 5 to select the S
   SPI peripheral.
4.   * @param  Mode: SSPIx Transmit mode configurate.
5.   *          This parameter can be one of the following values:
6.   *            @arg SSPI_TransmitMode_Send: Send mode setting for SS
   PIx bus.
7.   *            @arg SSPI_TransmitMode_Recv: Receive mode setting for
    SSPIx bus.
8.   *            @arg SSPI_TransmitMode_TxRx: Receive and send mode se
   tting for SSPIx bus.
9.   * @retval None.
10.  */
11.void SSPI_Set_TransmitMode(SCIAFSelect_TypeDef SSPIx, uint8_t Mode)
```

对于 SPI 有三种工作模式:收、发、收发一体模式；对于工作在收发模式下的 SPI、可以发送数据，也可以发送数据同时接收数据；对于只收模式下的 SPI 只能读取 SPI 接收缓冲区的数据，发送模式下只能发送数据。

3.2. SSPI 驱动 w25qxx 设备

在 SPI 接口应用例程 spi00MasterSendReceive 中，添加了 drv_wq25qxx.c 的驱动文件；例程演示读取设备 device-id JEDEC ID 以及 uniq_id；给出了轮询、中断、DMA 方式的读写方式。

```
//#define SPI_WITH_DMA

uint16_t drv_w25qxx_read_device_id(void)
{
    uint16_t tmp = 0;
    static uint8_t rxbuf[6];
    static uint8_t txbuf[7] = {0x90, 0x00, 0x00, 0x00, 0xff, 0xff};
#ifdef  SPI_POLLING
    SSPI_Start(SSPI11);

    SSPI_TransmitData(SSPI11,0x90);
    SSPI_TransmitData(SSPI11,0x00);
    SSPI_TransmitData(SSPI11,0x00);
    SSPI_TransmitData(SSPI11,0x00);
    tmp |= SSPI_TransmitData(SSPI11,0xFF) << 8;
    tmp |= SSPI_TransmitData(SSPI11,0xFF);

    SSPI_Stop(SSPI11);

    return tmp;
#else
#ifdef SPI_WITH_DMA
    Spi11_Dma_TxRx(txbuf, 6, rxbuf, 6);
    delayMS(4);
    while (g_spi_tx_end == 0);

    tmp |= (rxbuf[4] << 8);
    tmp |= rxbuf[5];
    return tmp;
#else
    SSPI_Start(SSPI11);

    SSPI_Set_TransmitMode(SSPI11, SSPI_TransmitMode_TxRx);

    SSPI_TransmitData(SSPI11,0x90);
    SSPI_TransmitData(SSPI11,0x00);
    SSPI_TransmitData(SSPI11,0x00);
    SSPI_TransmitData(SSPI11,0x00);

    Spi11_Int_Read(rxbuf, 2);
    delayMS(1);
    while (g_spi_rx_end == 0);

    tmp |= (rxbuf[0] << 8);
    tmp |= rxbuf[1];

    return tmp;
#endif
#endif
```

轮询方式

DMA收发

轮询发送中断接收

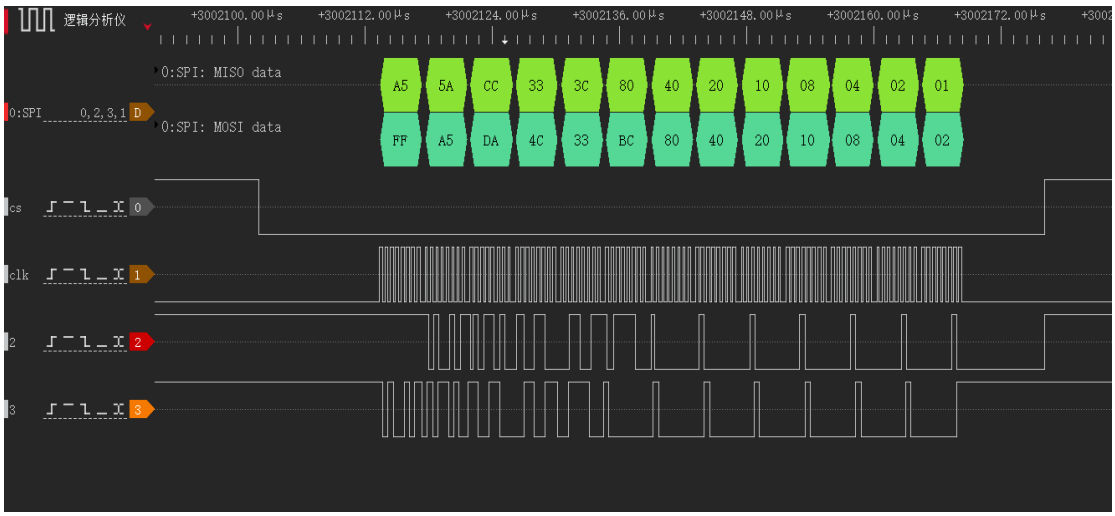
以读取 device_id 为例子；分别给出了上述介绍的几种方式；用户可以通过宏定义来选择不同方式来验证；

3.3. SSPI 读取 BAT32G137 系列 SSPI 从机

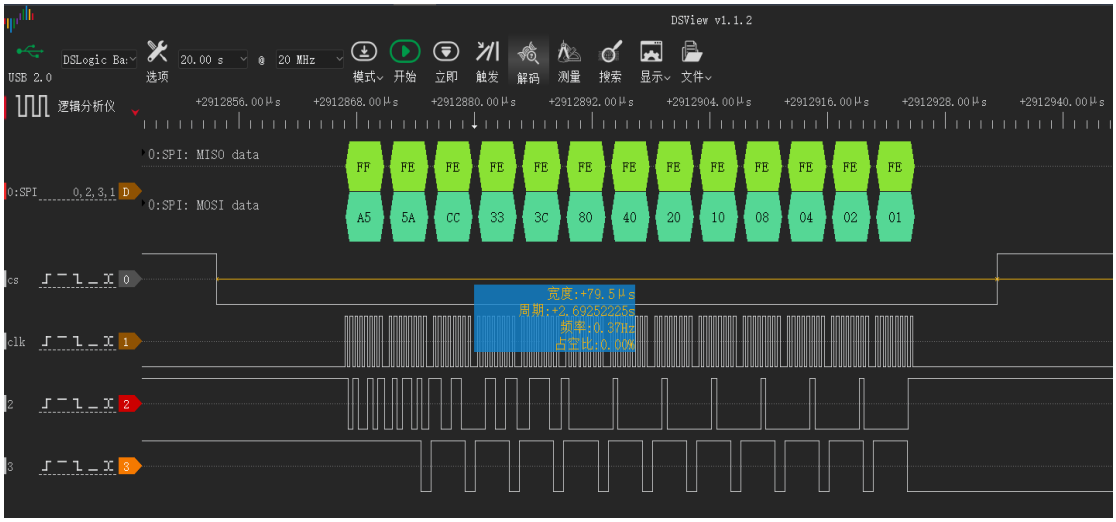
在主函数中通过宏去选择使用演示的例程，提供了读取 BAT32G137 系列的 SPI 从机的例程；有 DMA 和中断收发例程

4. 示例演示

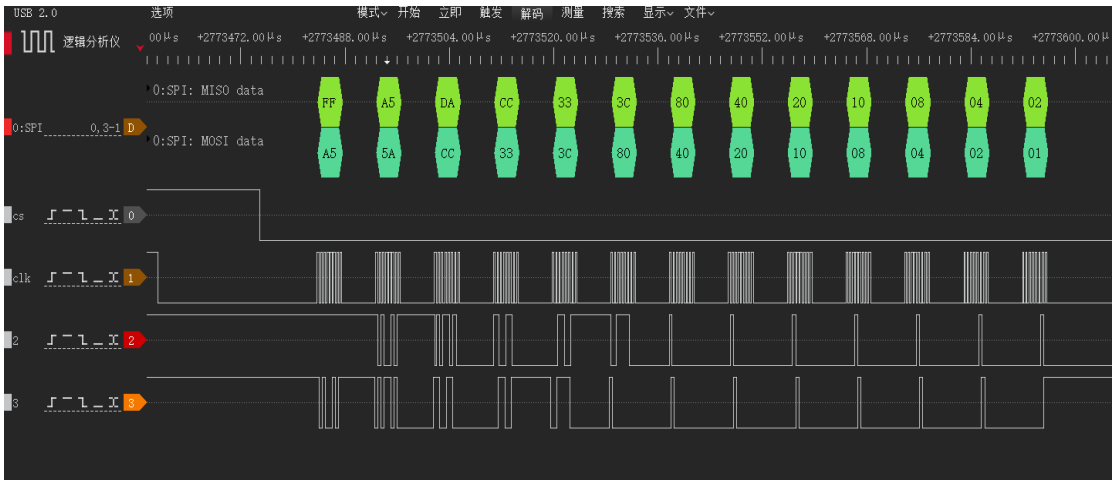
在例程使用 spiSlaveSendReceive 作为 SPI 从机，spiMasterSendReceive 作为 SPI 主机，进行数据互发，交互时序图如下：



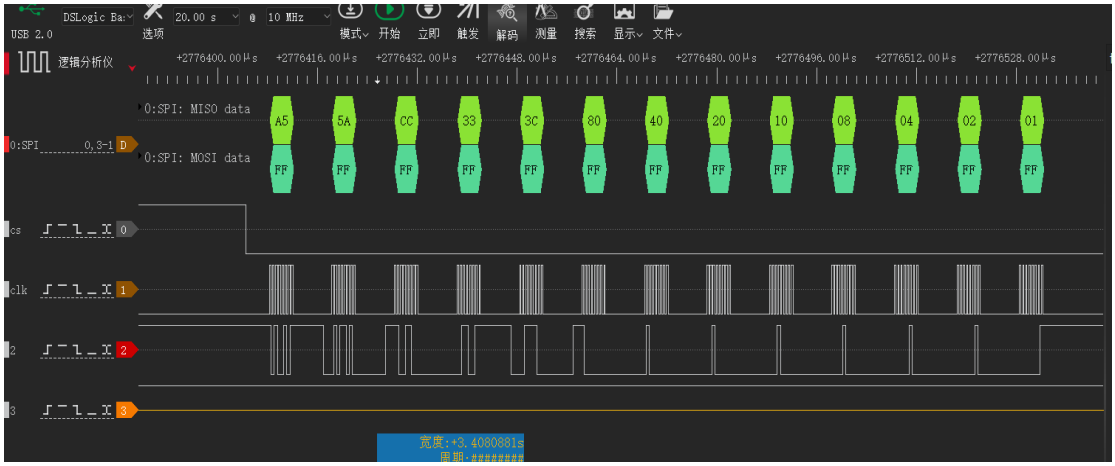
图一 SPI DMA 发送时序图



图二 SPI DMA 接收时序图

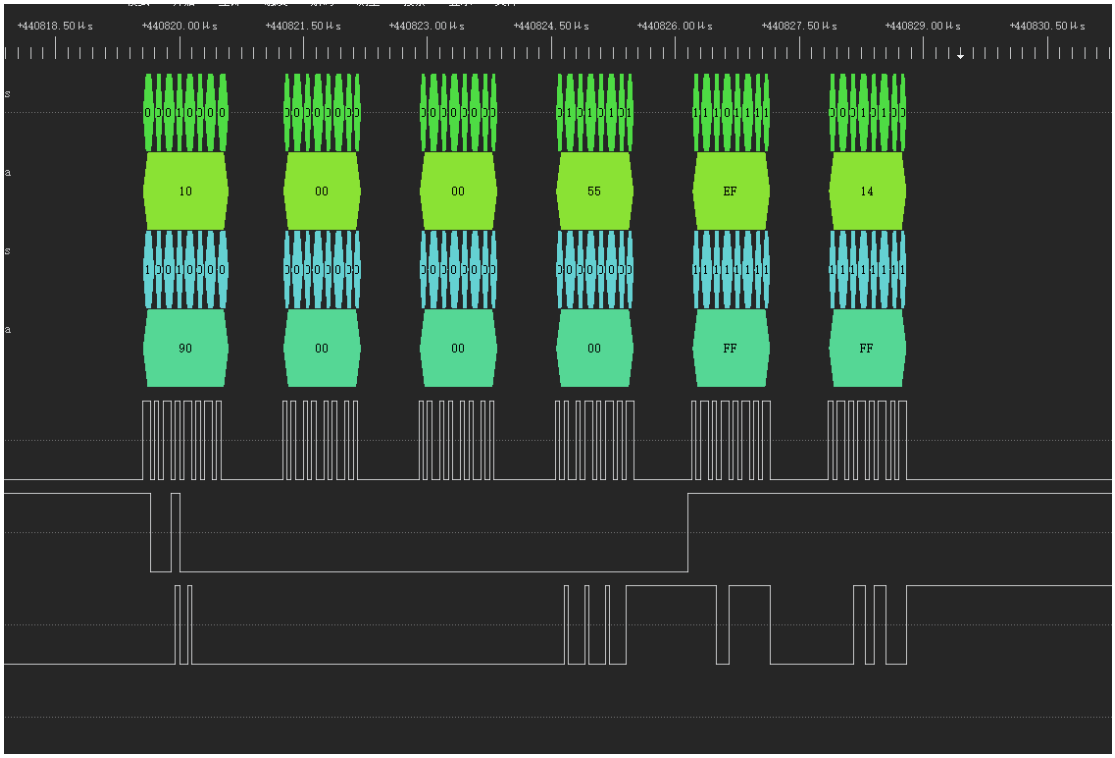


图三 中断发送时序

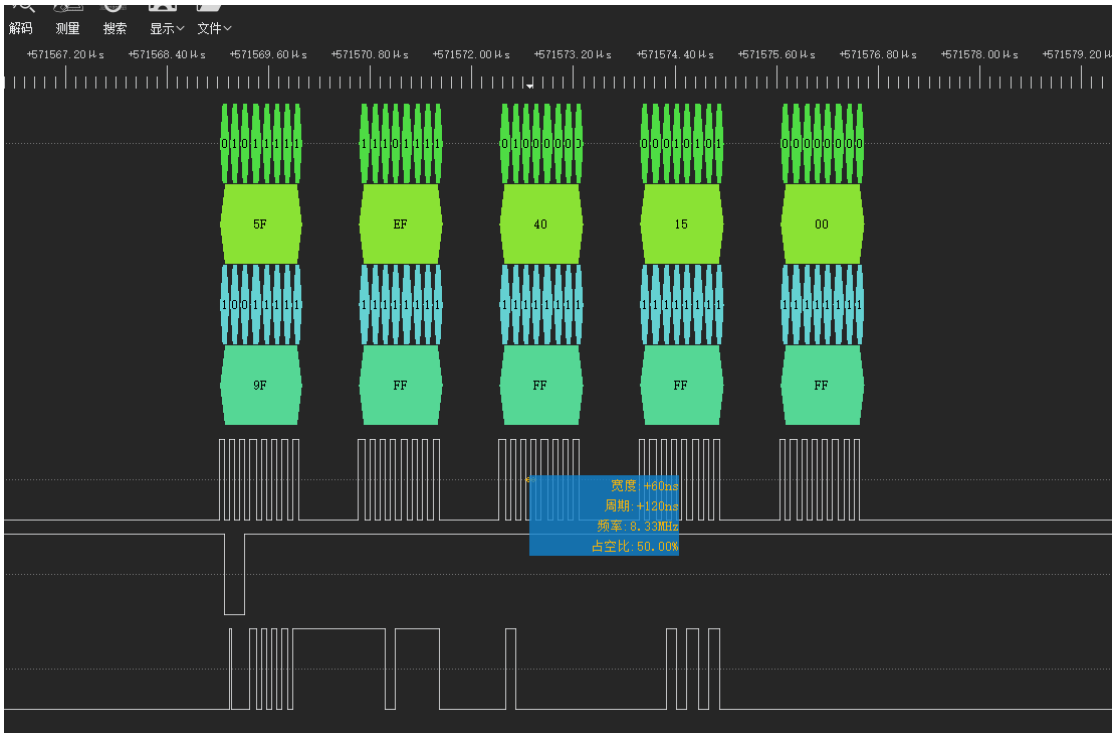


图四 中断接收时序

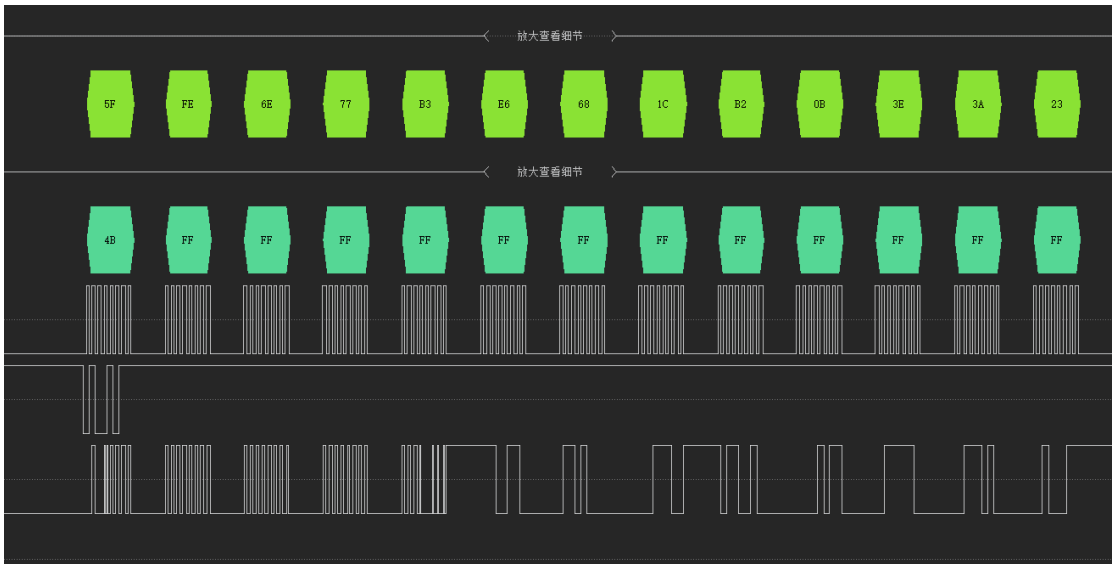
读取 W25QXX 设备的时序图仿真：



图五 读取设备 id



图六 读取 jecid



图七 读取 unique id