青风手把手教你学 stm32f030 系列教程

------------- 库函数操作版本

出品论坛： www.qfv8.com 青风电子社区

作者： 青风

出品论坛: www.qfv8.com

淘宝店：http://qfv5.taobao.com

QQ 技术群：241364123

硬件平台：QF-STM32F030 开发板

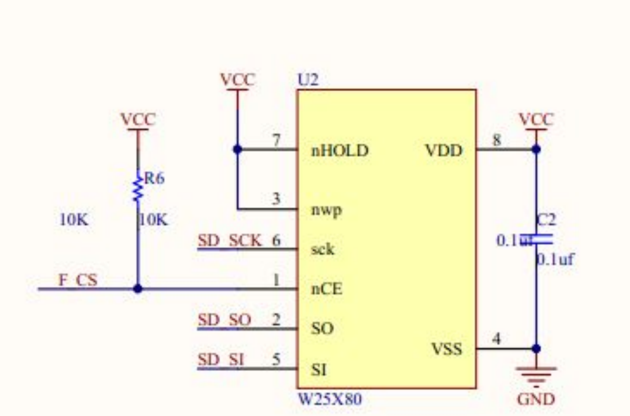
2.11 SPI 读写串行 FLASH

2.11.1 原理分析：

SPI(Serial Peripheral Interface--串行外设接口)总线系统是一种同步串行外设接口，它可以使 MCU 与各种外围设备以串行方式进行通信以交换信息。SPI 有三个寄存器分别为：控制寄存器 SPCR，状态寄存器 SPSR，数据寄存器 SPDR。外围设备包括 FLASHRAM、网络控制器、LCD 显示驱动器、A/D 转换器和 MCU 等。SPI 总线系统可直接与各个厂家生产的多种标准外围器件直接接口，该接口一般使用 4 条线：串行时钟线（SCLK）、主机输入/从机输出数据线 MISO、主机输出/从机输入数据线 MOSI 和低电平有效的从机选择线 SS(有的 SPI 接口芯片带有中断信号线 INT、有的 SPI 接口芯片没有主机输出/从机输入数据线 MOSI)。 本实验通过 SPI 读写串行 FLASH ,串行 FLASH 采样 W25X16。

2.11.2 硬件准备：

硬件配置入下图所示，在 TFT 转接板和 SD 卡共用一个 SPI 接口：



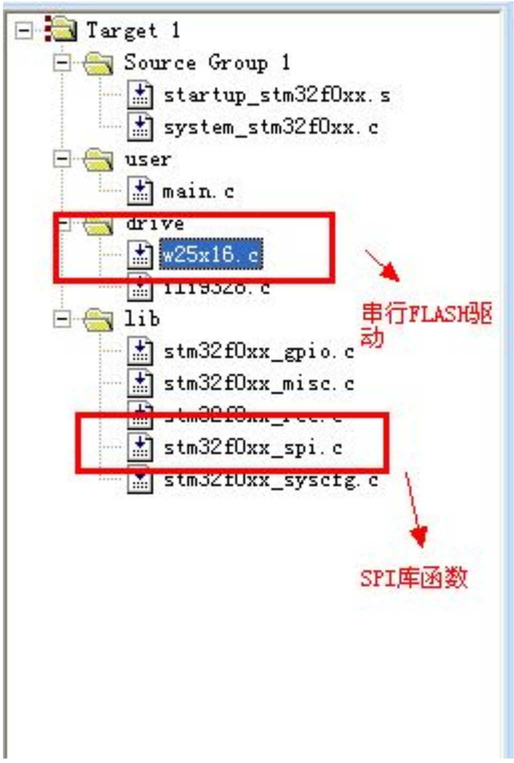
| PF5-CS : W25X16-CS |
| PB13-SPI2-SCK : W25X16-CLK |
| PB14-SPI2-MISO : W25X16-DO |
| PB15-SPI2-MOSI : W25X16-DIO |

CS:FLASH 片选信号引脚。
SCK:FLASH 时钟信号引脚。
MISO:FLASH 主入从出引脚。
MOSI：FLASH 主出从进引脚。

硬件按照如上方式连接后，下面来配置驱动程序。

2.11.3 软件配置：

采用库函数编写驱动，工程目录如下图所示，用户需要编写 FALSH 驱动函数 w25x16.c 驱动函数和主函数 main.c.



下面我们首先来讨论 **w25x16.c** 的驱动编写。首先对 **FLASH** 进行初始化，包括初始化几个方面：

时钟设置， **IO** 端口复用，**SPI** 参数设置，

```
void SPI_FLASH_Init(void)
{

    GPIO_InitTypeDef   GPIO_InitStruct;
    SPI_InitTypeDef     SPI_InitStruct;

    RCC_AHBPeriphClockCmd(RCC_AHBPeriph_GPIOF| RCC_AHBPeriph_GPIOB,
  ENABLE);//配置 gpio 时钟

    RCC_APB1PeriphClockCmd(FLASH_SPI2, ENABLE); //配置 spi 时钟

    GPIO_InitStruct.GPIO_Pin = FLASH_SCK_PIN;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AF;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_Level_3;
    GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
    GPIO_InitStruct.GPIO_PuPd   = GPIO_PuPd_UP;
    GPIO_Init(FLASH_SCK_PORT, &GPIO_InitStruct);//时钟 gpio 端口模式

    /*!< Configure SPI pins: MISO */
    GPIO_InitStruct.GPIO_Pin = FLASH_MISO_PIN;
    GPIO_Init(FLASH_MISO_PORT, &GPIO_InitStruct);

    /*!< Configure SPI pins: MOSI */
    GPIO_InitStruct.GPIO_Pin =FLASH_MOSI_PIN;
    GPIO_Init(FLASH_MOSI_PORT, &GPIO_InitStruct);

    /* Connect PXx to SPI_SCK */
    GPIO_PinAFConfig(FLASH_SCK_PORT, FLASH_SCK_SOURCE, FLASH_SCK_AF);

    /* Connect PXx to SPI_MISO */
    GPIO_PinAFConfig(FLASH_MISO_PORT, FLASH_MISO_SOURCE, FLASH_MISO_AF);

    /* Connect PXx to SPI_MOSI */
    GPIO_PinAFConfig(FLASH_MOSI_PORT, FLASH_MOSI_SOURCE, FLASH_MOSI_AF);
 //设置 gpio 端口的复用

    GPIO_InitStruct.GPIO_Pin =FLASH_CS_PIN;
    GPIO_InitStruct.GPIO_Mode = GPIO_Mode_OUT;
    GPIO_InitStruct.GPIO_OType = GPIO_OType_PP;
   GPIO_InitStruct.GPIO_PuPd = GPIO_PuPd_UP;
    GPIO_InitStruct.GPIO_Speed = GPIO_Speed_Level_3;
    GPIO_Init(FLASH_CS_PORT, &GPIO_InitStruct);

    SPI_FLASH_CS_HIGH();

    SPI_InitStruct.SPI_Direction = SPI_Direction_2Lines_FullDuplex;//配置 spi 方向
    SPI_InitStruct.SPI_Mode = SPI_Mode_Master;//配置 spi 模式
    SPI_InitStruct.SPI_DataSize = SPI_DataSize_8b;//配置数据格式
    SPI_InitStruct.SPI_CPOL = SPI_CPOL_High;//配置时钟高电平稳态
    SPI_InitStruct.SPI_CPHA = SPI_CPHA_2Edge;//配置时钟 bit 位捕获方式
    SPI_InitStruct.SPI_NSS = SPI_NSS_Soft;//设置 nss 管脚软件管理
    SPI_InitStruct.SPI_BaudRatePrescaler = SPI_BaudRatePrescaler_2;//设置 spi 波特率分频值
    SPI_InitStruct.SPI_FirstBit = SPI_FirstBit_MSB;//指定数据传输从 msb 位开始
    SPI_InitStruct.SPI_CRCPolynomial = 7;//指定用于 CRC 计算的值
    SPI_Init(SPI2, &SPI_InitStruct);//调入结构体
    SPI_RxFIFOThresholdConfig(SPI2, SPI_RxFIFOThreshold_QF);//设置接收缓冲
    SPI_Cmd(SPI2, ENABLE); /*!< SD_SPI enable */
  }
```

在 stm32f0xx_spi.h 文件中设置了 spi 参数结构体，如下代码所示，初始化的时候直接进行调用：

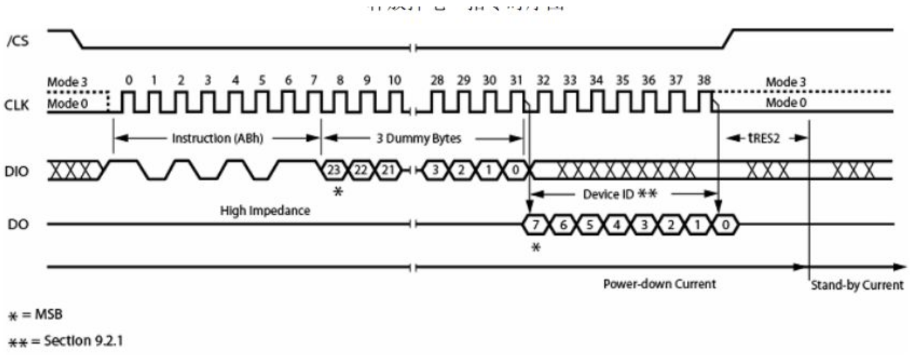
```
typedef struct
{
    uint16_t SPI_Direction;
    uint16_t SPI_Mode;
    uint16_t SPI_DataSize;
    uint16_t SPI_CPOL;
    uint16_t SPI_CPHA;
    uint16_t SPI_NSS;
    uint16_t SPI_BaudRatePrescaler;
    uint16_t SPI_FirstBit;
    uint16_t SPI_CRCPolynomial;
}SPI_InitTypeDef;//spi 参数结构体
```

初始化后，开始编写 读和写 W25X16 的代码，时序关系我们需要参考 w25x16 的数据手册：

首先通过 SPI 接口发送字节，同时接收：

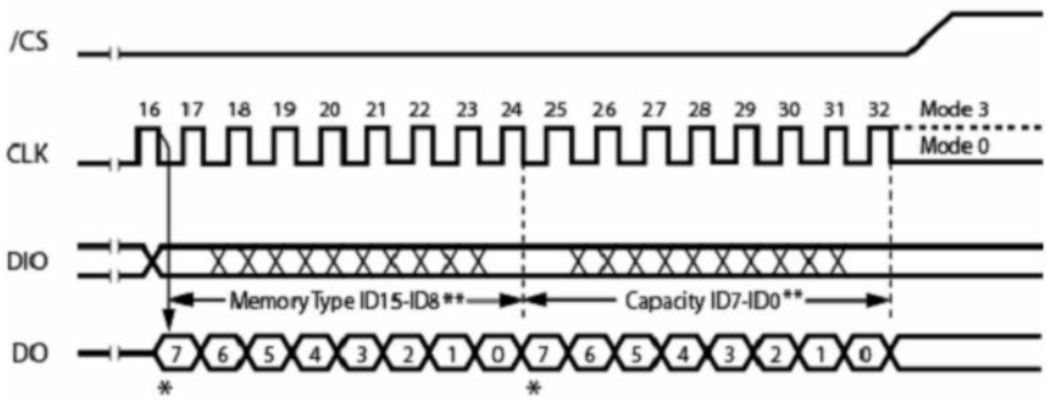
```
uint8_t SPI_FLASH_SendByte(uint8_t byte)
{
    while (SPI_I2S_GetFlagStatus(SPI2, SPI_I2S_FLAG_TXE) == RESET);//判断是否发送完成
    SPI_SendData8(SPI2, byte);//SPI 发送字节

    /* Wait to receive a byte */
    while (SPI_I2S_GetFlagStatus(SPI2, SPI_I2S_FLAG_RXNE) == RESET);//是否已经读取
    /* Return the byte read from the SPI bus */
    return SPI_ReceiveData8(SPI2);//SPI 接收
}
```

上面的发送命令接收数据是基本操作步骤，下面写读取器件 ID 代码，参考 w25x16 代码参考手册中的时序图：



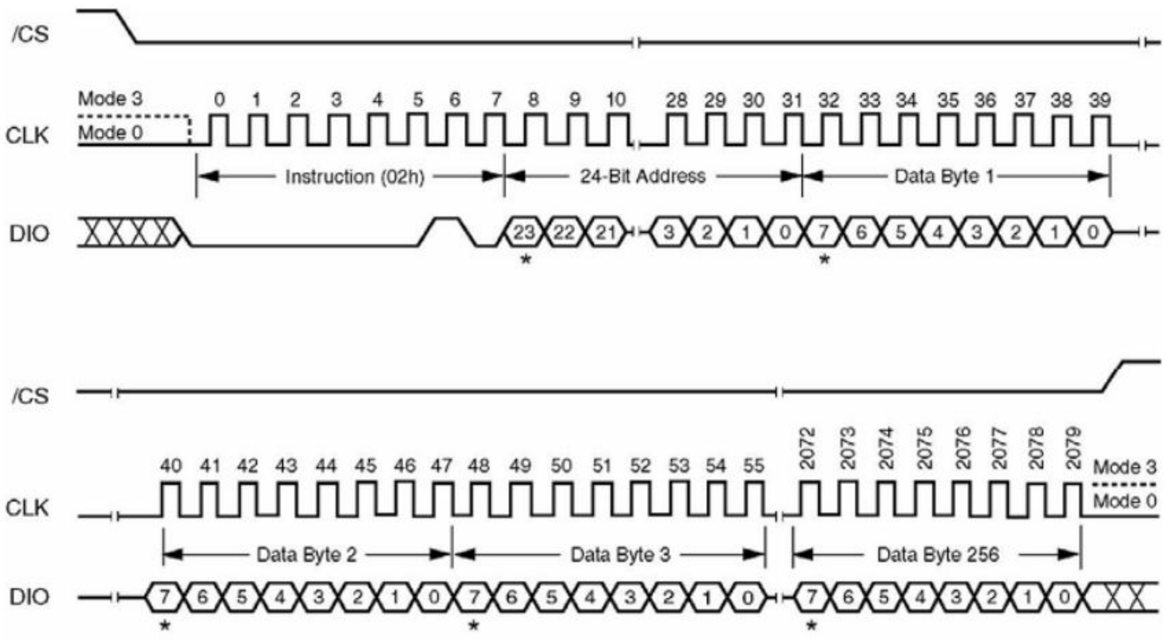
```
uint32_t SPI_FLASH_ReadDeviceID(void)
{
    uint32_t Temp = 0;

    /* Select the FLASH: Chip Select low */
    SPI_FLASH_CS_LOW();

    /* Send "RDID " instruction */
    SPI_FLASH_SendByte(W25X_DeviceID);
    SPI_FLASH_SendByte(Dummy_Byte);
    SPI_FLASH_SendByte(Dummy_Byte);
    SPI_FLASH_SendByte(Dummy_Byte);

    /* Read a byte from the FLASH */
    Temp = SPI_FLASH_SendByte(Dummy_Byte);

    /* Deselect the FLASH: Chip Select high */
    SPI_FLASH_CS_HIGH();

    return Temp;
}
```

读取制造 ID 参考时序图：



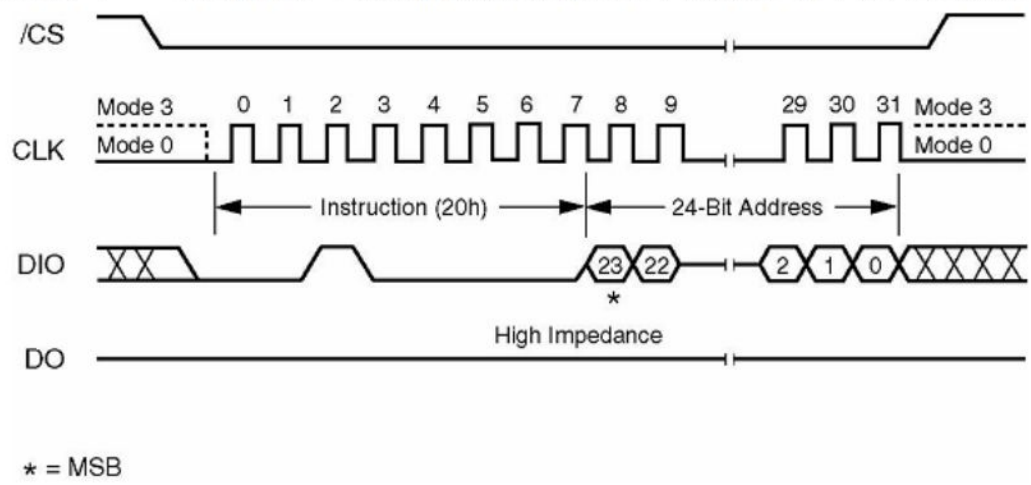
```
uint32_t SPI_FLASH_ReadID(void)
{
    uint32_t Temp = 0, Temp0 = 0, Temp1 = 0, Temp2 = 0;

    /* Select the FLASH: Chip Select low */
    SPI_FLASH_CS_LOW();

    /* Send "RDID " instruction */
    SPI_FLASH_SendByte(W25X_JedecDeviceID);

   /* Read a byte from the FLASH */
   Temp0 = SPI_FLASH_SendByte(Dummy_Byte);

   /* Read a byte from the FLASH */
   Temp1 = SPI_FLASH_SendByte(Dummy_Byte);

   /* Read a byte from the FLASH */
   Temp2 = SPI_FLASH_SendByte(Dummy_Byte);

    /* Deselect the FLASH: Chip Select high */
    SPI_FLASH_CS_HIGH();

    Temp = (Temp0 << 16) | (Temp1 << 8) | Temp2;

    return Temp;
}
```

W25X16 页写参考时序图：



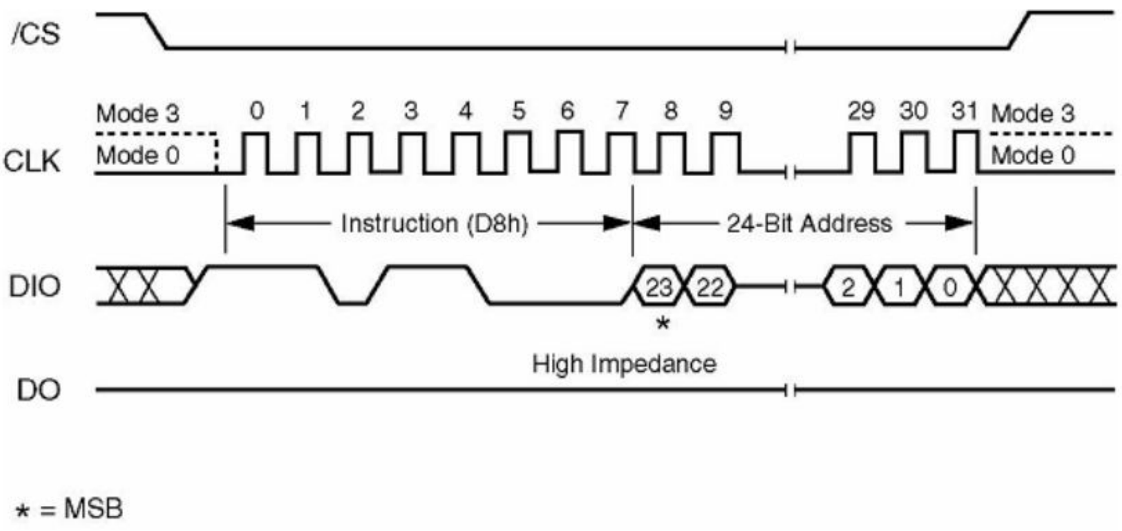
```
void SPI_FLASH_PageWrite(uint8_t* pBuffer, uint32_t WriteAddr, uint16_t NumByteToWrite)
{
    /* Enable the write access to the FLASH */
    SPI_FLASH_WriteEnable();

    /* Select the FLASH: Chip Select low */
    SPI_FLASH_CS_LOW();
    /* Send "Write to Memory " instruction */
    SPI_FLASH_SendByte(W25X_PageProgram);
    /* Send WriteAddr high nibble address byte to write to */
    SPI_FLASH_SendByte((WriteAddr & 0xFF0000) >> 16);
    /* Send WriteAddr medium nibble address byte to write to */
    SPI_FLASH_SendByte((WriteAddr & 0xFF00) >> 8);
    /* Send WriteAddr low nibble address byte to write to */
    SPI_FLASH_SendByte(WriteAddr & 0xFF);

    if(NumByteToWrite > SPI_FLASH_PerWritePageSize)
    {
        NumByteToWrite = SPI_FLASH_PerWritePageSize;
        //printf("\n\r Err: SPI_FLASH_PageWrite too large!");
    }

    /* while there is data to be written on the FLASH */
    while (NumByteToWrite--)
    {
        /* Send the current byte */
        SPI_FLASH_SendByte(*pBuffer);
        /* Point on the next byte to be written */
        pBuffer++;
    }

    /* Deselect the FLASH: Chip Select high */
    SPI_FLASH_CS_HIGH();
    /* Wait the end of Flash writing */
    SPI_FLASH_WaitForWriteEnd();
}
```

W25x16 扇区擦除时序图：



```
void SPI_FLASH_SectorErase(uint32_t SectorAddr)
{
    /* Send write enable instruction */
    SPI_FLASH_WriteEnable();
    SPI_FLASH_WaitForWriteEnd();
    /* Sector Erase */
    /* Select the FLASH: Chip Select low */
    SPI_FLASH_CS_LOW();
    /* Send Sector Erase instruction */
    SPI_FLASH_SendByte(W25X_SectorErase);
    /* Send SectorAddr high nibble address byte */
    SPI_FLASH_SendByte((SectorAddr & 0xFF0000) >> 16);
    /* Send SectorAddr medium nibble address byte */
    SPI_FLASH_SendByte((SectorAddr & 0xFF00) >> 8);
    /* Send SectorAddr low nibble address byte */
    SPI_FLASH_SendByte(SectorAddr & 0xFF);
    /* Deselect the FLASH: Chip Select high */
    SPI_FLASH_CS_HIGH();
    /* Wait the end of Flash writing */
    SPI_FLASH_WaitForWriteEnd();
}
```

W25x16 块擦除参考时序图：



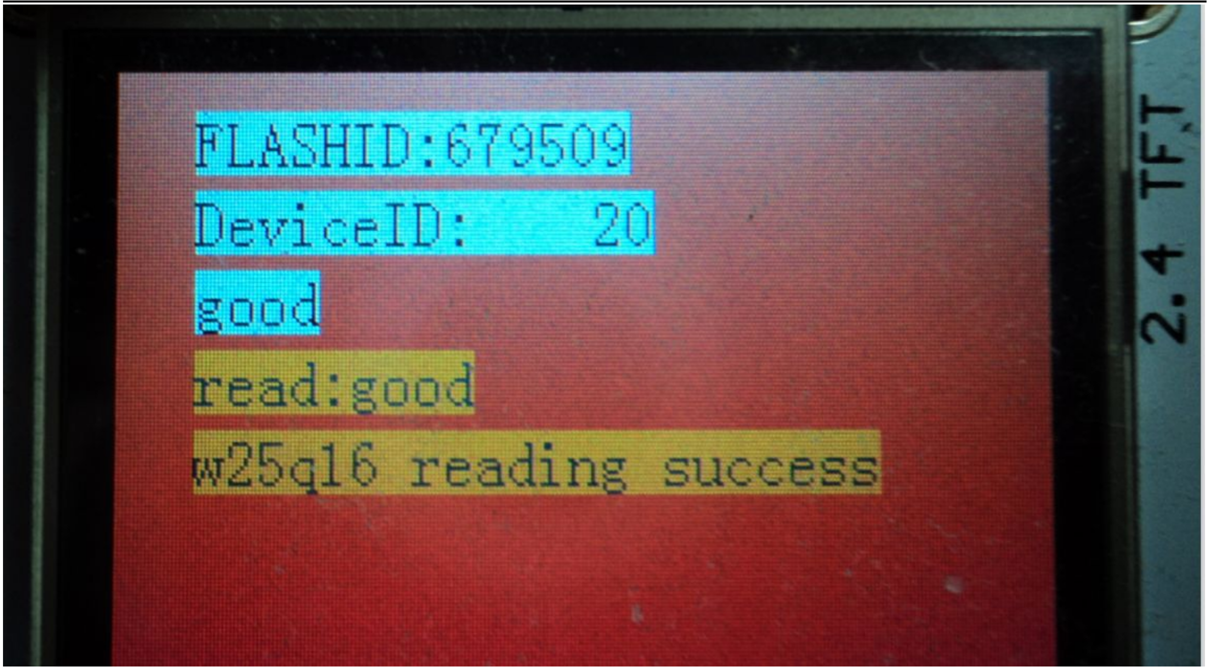
```
void SPI_FLASH_BulkErase(void)
{
    /* Send write enable instruction */
    SPI_FLASH_WriteEnable();

    /* Bulk Erase */
    /* Select the FLASH: Chip Select low */
    SPI_FLASH_CS_LOW();
    /* Send Bulk Erase instruction   */
    SPI_FLASH_SendByte(W25X_ChipErase);
    /* Deselect the FLASH: Chip Select high */
    SPI_FLASH_CS_HIGH();

    /* Wait the end of Flash writing */
    SPI_FLASH_WaitForWriteEnd();
}
```

主函数如下：

```
/******************** (C) COPYRIGHT 2011 青风电子********************************
 * 文件名    ：main.c
 * 描述      ：SPI 测试。
 *
 * 实验平台：QF-STM32F030 开发板
 * 库版本    ：ST3.0.0
 *
 * 作者      ：青风
 *
**********************************************************************************/
/***头文件调用****/
#include "stm32f0xx.h"
```

```
#include "w25x16.h"
#include "ili9328.h"

typedef enum { FAILED = 0, PASSED = !FAILED} TestStatus;
__IO uint32_t DeviceID = 0;
__IO uint32_t FlashID = 0;
__IO TestStatus TransferStatus1 = FAILED;

/*  获取缓冲区的长度  */
#define TxBufferSize1      (countof(TxBuffer1) - 1)
#define RxBufferSize1      (countof(TxBuffer1) - 1)
#define countof(a)         (sizeof(a) / sizeof(*(a)))
#define   BufferSize (countof(Tx_Buffer)-1)

#define   FLASH_WriteAddress        0x00000
#define   FLASH_ReadAddress          FLASH_WriteAddress
#define   FLASH_SectorToErase       FLASH_WriteAddress
#define   sFLASH_ID                 0xEF3015

uint8_t Tx_Buffer[] = "good";
uint8_t Rx_Buffer[];


//-----------------------------------------------------------
//     * @brief   Compares two buffers.
//     * @param   pBuffer1, pBuffer2: buffers to be compared.
//     * @param   BufferLength: buffer's length
//     * @retval PASSED: pBuffer1 identical to pBuffer2
//     *               FAILED: pBuffer1 differs from pBuffer2
// ------------------------------------------------------------
TestStatus Buffercmp(uint8_t* pBuffer1, uint8_t* pBuffer2, uint16_t BufferLength)
{
    while(BufferLength--)
    {
        if(*pBuffer1 != *pBuffer2)
        {
            return FAILED;
        }

        pBuffer1++;
        pBuffer2++;
    }

    return PASSED;
}
//////////////////////////////////////////////////
void Delay(__IO uint32_t nCount)
{
    for(; nCount != 0; nCount--);
}
/////////////////////////////////////////////////
int main(void)
{
    SystemInit();
        SPI_FLASH_Init();
        LCD_init();              //  液晶显示器初始化
        LCD_Clear(ORANGE);        //  全屏显示白色
        POINT_COLOR =BLACK; //  定义笔的颜色为黑色
        BACK_COLOR = WHITE   ;      //  定义笔的背景色为白色
        DeviceID = SPI_FLASH_ReadDeviceID();
        Delay( 200 );
    /* Get SPI Flash ID */
        FlashID = SPI_FLASH_ReadID();
        LCD_ShowString(20,10,"FLASHID:");
        LCD_ShowNum(84,10,FlashID,6);//读取 FLASHID
        LCD_ShowString(20,30,"DeviceID:");
        LCD_ShowNum(90,30,DeviceID,6);//读取 DEVICEID
        SPI_FLASH_SectorErase(FLASH_SectorToErase);
        SPI_FLASH_BufferWrite(Tx_Buffer,0x00000, 5);
        LCD_ShowString(20,50,    Tx_Buffer);//显示发送缓冲内的内容

        SPI_FLASH_BufferRead(Rx_Buffer,0x00000, 5);//读取写入的内容
        LCD_ShowString(20,70,"read:");
        LCD_ShowString(60,70,Rx_Buffer);//显示接收缓冲内容

        if(*Tx_Buffer==*Rx_Buffer)
        {
                LCD_ShowString(20,90,"w25q16 reading success");
                }
        else
        { LCD_ShowString(20,90,"w25q16 reading error");
                }//比较接收和发送的内容是否相同，相同则判断写入正确
}
```

实验现象：



```
FLASHID:679509
DeviceID:      20
good
read:good
w25q16 reading success
```

液晶 TFT 显示我们目前对 W25Q16 的操作情况