0 前言

本篇指南仅用作 W25Qxx 系列初级学习使用，是本人在学习过程中的一个自我记录，不用做商业用途。代码多参考自正点原子例程，无本人版权。在本篇文档中如有部分不详尽的地方，欢迎评论留言，或联系本人邮箱 syujen_1997@163.com。

1 介绍

W25Q64 存储容量共 64M-bit/ 8M-byte，32768 页(pages)、每页 256-bytes。最大一次可编程 256-bytes。W25Q128JV 存储容量共 128M-bit/ 16M-byte

一次擦除大小可以为 16 页（4KB）、128 页（32KB）、256 页（64KB）或者全擦除。

W25Q64JV 分别有 2048 个可擦除扇区(sectors)和 128 个可擦除块(blocks)。关系 1 Block = 16 sectors；1 sector = 4KB，所以算起来能达到 8M-byte。

编程即写数据，由于 Flash 的特性，只能从 1 编程 0，所以写数据之前 Flash 里面的数据不是 0xFF 就必须先擦除，然后才能写数据。擦除即将 Flash 里面的数据恢复为 0xFF 的过程。

上电后设备自动处于写禁用状态（Write Enable Latch, WEL 为 0，WEL 是只读位）。在 Page Program, Sector Erase, Block Erase, Chip Erase or Write Status Register instruction（页编程、区擦除、块擦除、芯片擦除或者写状态寄存器指令）之前必须先进行写使能指令。在编程、擦除、写状态寄存器指令完成后，WEL 自动变成 0。

BUSY 位是状态寄存器 0 的第 0 位，并且是只读位。当执行页编程、区擦除、块擦除、芯片擦除、写状态寄存器、擦除/编程安全寄存器指令时，其值为 1，并且在此期间不再接收新的指令，但是可以接收读状态寄存器指令和擦除编程挂起指令。

2 寻址范围

5. BLOCK DIAGRAM

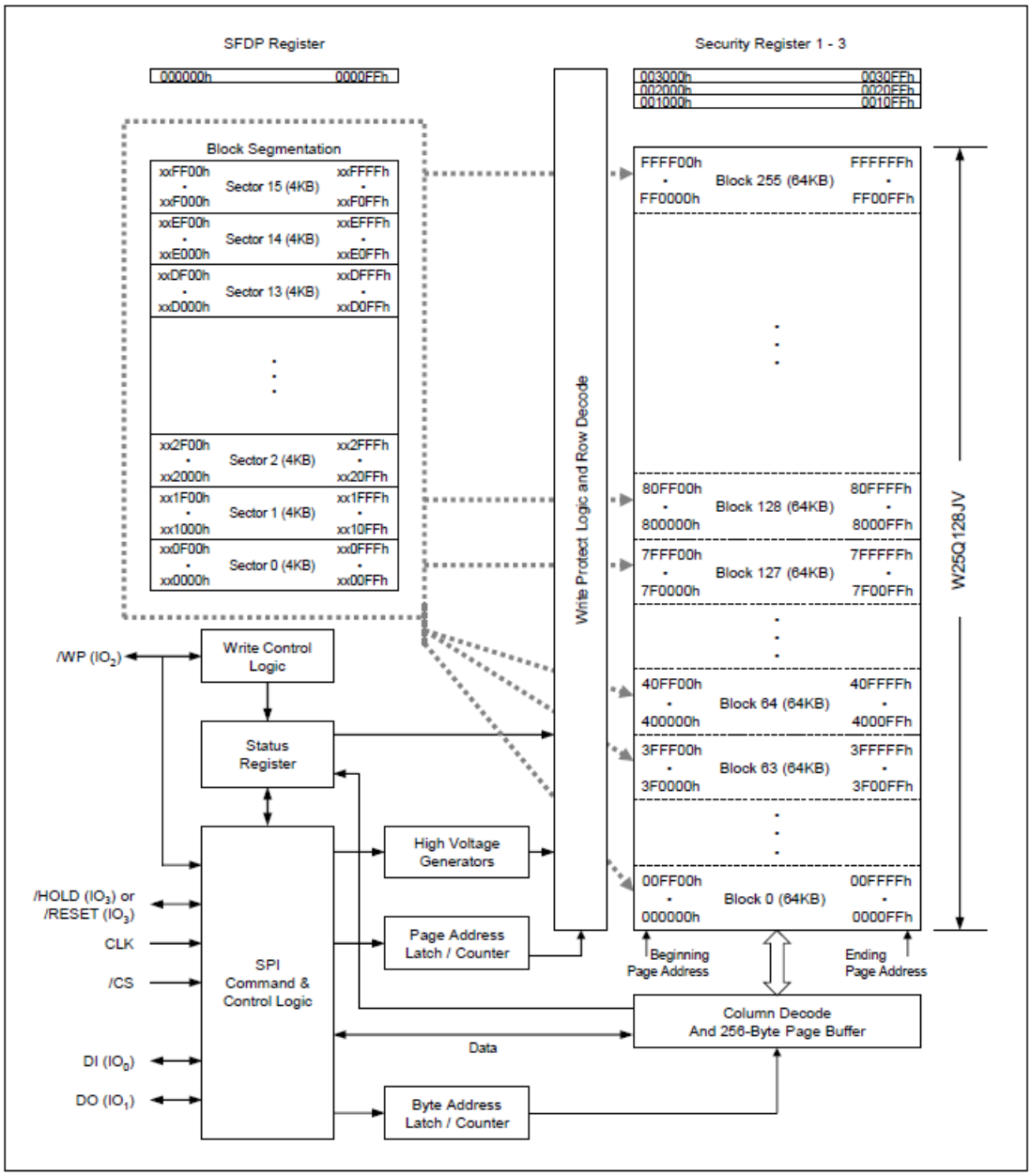


Figure 2. W25Q128JV Serial Flash Memory Block Diagram

这里仅给出 W25Q128JV 系列存储器框图，其余系列需查看其对应手册。

再次重申，1 Block = 16 sectors；1 sector = 4KB。从图中可以看出，Block 0 的 Sector 0 寻址范围是 0x000000 到 0x000FFF，会发现是 0x000FFF=2^12-1 = 4095，正好是 4KB（4 * 1024 bytes，地址加 1 的单位是字节，不是 bit）

3 SPI 通讯

6. FUNCTIONAL DESCRIPTIONS

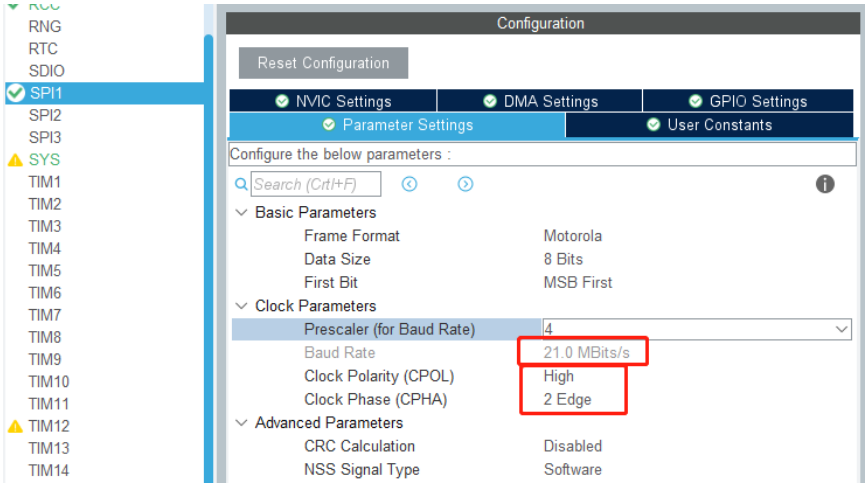
6.1 Standard SPI Instructions

The W25Q128JV is accessed through an SPI compatible bus consisting of four signals: Serial Clock (CLK), Chip Select (/CS), Serial Data Input (DI) and Serial Data Output (DO). Standard SPI instructions use the DI input pin to serially write instructions, addresses or data to the device on the rising edge of CLK. The DO output pin is used to read data or status from the device on the falling edge of CLK.

SPI bus operation Mode 0 (0,0) and 3 (1,1) are supported. The primary difference between Mode 0 and Mode 3 concerns the normal state of the CLK signal when the SPI bus master is in standby and data is not being transferred to the Serial Flash. For Mode 0, the CLK signal is normally low on the falling and rising edges of /CS. For Mode 3, the CLK signal is normally high on the falling and rising edges of /CS.

标准 SPI 通讯下支持 SPI 总线操作模式为 0(0,0)和 3(1,1)。通常采用模式 3（即 CPOL=1, CPHA=1，SCK 引脚在空闲状态下处于高电平，SCK 引脚上的第二个边沿对 MSBit 采样）。Dual SPI、Quad SPI 在此不深入讨论。

需要注意的是，W25Qxx 系列对 SPI 的通讯时钟是有限制的，最大为 133M，配置 stm32 时需注意。



4 判断 W25Qxx 的内存大小

8.1.2 Instruction Set Table 1 (Standard SPI Instructions)(1)

Data Input Output	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Number of Clock(1-1-1)	8	8	8	8	8	8	8
Write Enable	06h						
Volatile SR Write Enable	50h						
Write Disable	04h						
Release Power-down / ID	ABh	Dummy	Dummy	Dummy	(ID7-ID0)(2)		
Manufacturer/Device ID	90h	Dummy	Dummy	00h	(MF7-MF0)	(ID7-ID0)	
JEDEC ID	9Fh	(MF7-MF0)	(ID15-ID8)	(ID7-ID0)			
Read Unique ID	4Bh	Dummy	Dummy	Dummy	Dummy	(UID63-0)	

8.1 Device ID and Instruction Set Tables

8.1.1 Manufacturer and Device Identification

MANUFACTURER ID	**(MF7 - MF0)**	
Winbond Serial Flash	EFh	
Device ID	**(ID7 - ID0)**	**(ID15 - ID0)**
Instruction	**ABh, 90h, 92h, 94h**	**9Fh**
W25Q128JV-IN/IQ/JQ	17h	4018h
W25Q128JV-IM*/JM*	17h	7018h

Note: For DTR, QPI supporting, please refer to W25Q128JV-M DTR datasheet.

方法 1 根据 Manufacturer/Device ID 判断

8.2.23 Read Manufacturer / Device ID (90h)

The Read Manufacturer/Device ID instruction is an alternative to the Release from Power-down / Device ID instruction that provides both the JEDEC assigned manufacturer ID and the specific device ID.

The Read Manufacturer/Device ID instruction is very similar to the Release from Power-down / Device ID instruction. The instruction is initiated by driving the /CS pin low and shifting the instruction code "90h" followed by a 24-bit address (A23-A0) of 000000h. After which, the Manufacturer ID for Winbond (EFh) and the Device ID are shifted out on the falling edge of CLK with most significant bit (MSB) first as shown in Figure 39. The Device ID values for the W25Q128JV are listed in Manufacturer and Device Identification table. The instruction is completed by driving /CS high.

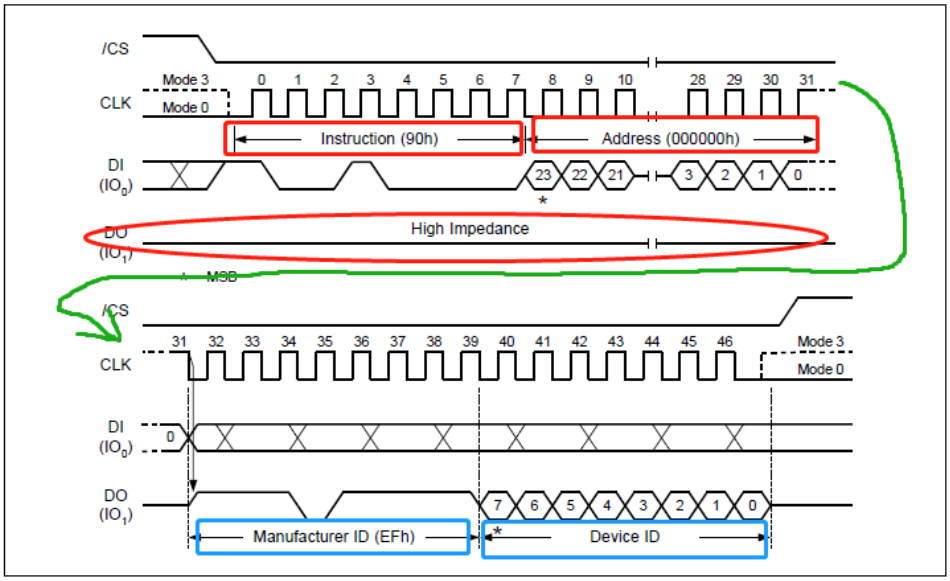


Figure 39. Read Manufacturer / Device ID Instruction

翻译：该指令通过将 /CS 引脚驱动为低电平并移位指令代码"90h"后跟 24 位地址 (A23-A0) 000000h 来启动。之后，Winbond 的制造商 ID (EFh) 和设备 ID 在 CLK 的下降沿移出，最高有效位 (MSB) 在前，如图 39 所示。

也就是说，采取方法 1 的情况下，向 W25Q128 发送

字节序列	1	2	3	4	5	6
STM32 发送字节	0x90	0x00	0x00	0x00	0xFF	0xFF
W25Q128 返回字节					0xEF	0x17

```
//读取芯片ID
//返回值如下:
//0XEF13,表示芯片型号为W25Q80
//0XEF14,表示芯片型号为W25Q16
//0XEF15,表示芯片型号为W25Q32
//0XEF16,表示芯片型号为W25Q64
//0XEF17,表示芯片型号为W25Q128
//0XEF18,表示芯片型号为W25Q256
u16 W25QXX_ReadID(void)
{
  u16 Temp = 0;
  W25QXX_CS=0;
  SPI1_ReadWriteByte(0x90);//发送读取ID命令
```

方法 2 根据 JEDEC ID

8.2.27 Read JEDEC ID (9Fh)

For compatibility reasons, the W25Q128JV provides several instructions to electronically determine the identity of the device. The Read JEDEC ID instruction is compatible with the JEDEC standard for SPI compatible serial memories that was adopted in 2003. The instruction is initiated by driving the /CS pin low and shifting the instruction code "9Fh". The JEDEC assigned Manufacturer ID byte for Winbond (EFh) and two Device ID bytes, Memory Type (ID15-ID8) and Capacity (ID7-ID0) are then shifted out on the falling edge of CLK with most significant bit (MSB) first as shown in Figure 43a. For memory type and capacity values refer to Manufacturer and Device Identification table.

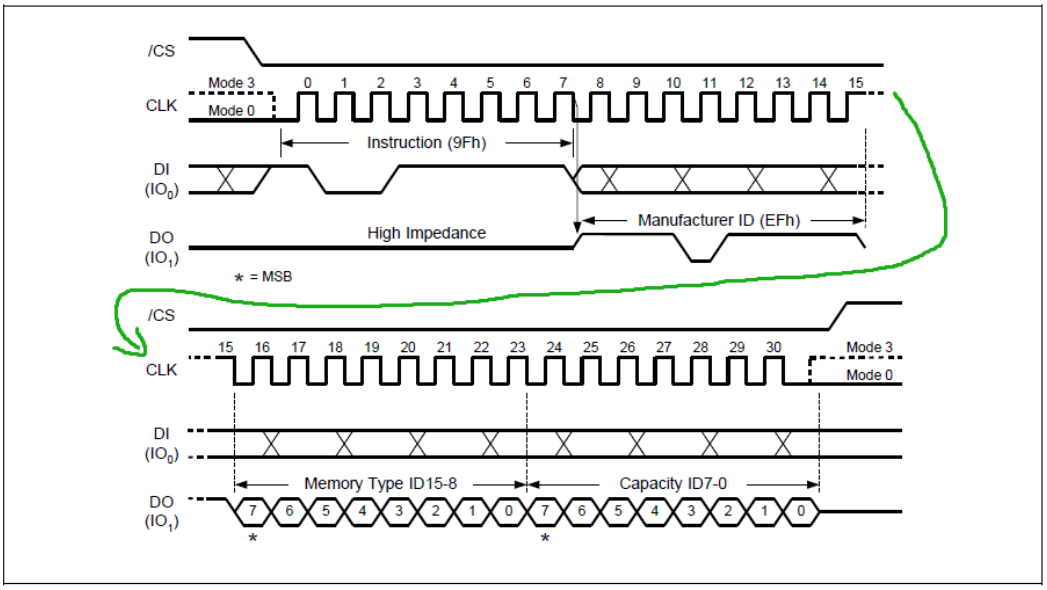


Figure 43a. Read JEDEC ID Instruction

该指令通过将 /CS 引脚拉低并移位指令代码"9Fh"来启动。

字节序列	1	2	3	4
STM32 发送字节	0x9F	0x00	0x00	0x00
W25Q128 返回字节		0xEF	0x40	0x18

```
//#define  sFLASH_ID        0xEF3015   //W25X16
//#define  sFLASH_ID        0xEF4015   //W25Q16
//#define  sFLASH_ID        0XEF4018   //W25Q128
#define  sFLASH_ID         0XEF4017   //W25Q64
```

5 操作 W25Qxx 的各种操作

1 读操作

8.2.6 Read Data (03h)

The Read Data instruction allows one or more data bytes to be sequentially read from the memory. The instruction is initiated by driving the /CS pin low and then shifting the instruction code "03h" followed by a 24-bit address (A23-A0) into the DI pin. The code and address bits are latched on the rising edge of the CLK pin. After the address is received, the data byte of the addressed memory location will be shifted out on the DO pin at the falling edge of CLK with most significant bit (MSB) first. The address is automatically incremented to the next higher address after each byte of data is shifted out allowing for a continuous stream of data. This means that the entire memory can be accessed with a single instruction as long as the clock continues. The instruction is completed by driving /CS high.

The Read Data instruction sequence is shown in Figure 14. If a Read Data instruction is issued while an Erase, Program or Write cycle is in process (BUSY=1) the instruction is ignored and will not have any effects on the current cycle. The Read Data instruction allows clock rates from D.C. to a maximum of f_R (see AC Electrical Characteristics).

The Read Data (03h) instruction is only supported in Standard SPI mode.

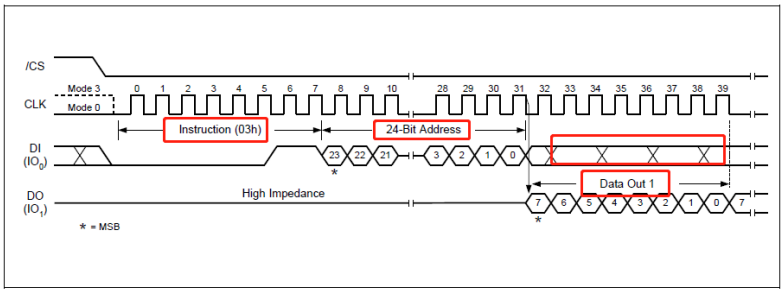


Figure 14. Read Data Instruction

读取数据指令允许从存储器中顺序读取一个或多个数据字节。通过将 /CS 引脚驱动为低电平来启动指令，然后将指令代码"03h"和一个 24 位地址 (A23-A0) 移入 DI 引脚。代码和地址位在 CLK 引脚的上升沿锁存。接收到地址后，寻址存储器位置的数据字节将在 CLK 的下降沿移出 DO 引脚，最高有效位(MSB)在前。在移出每个数据字节后，地址会自动递增到下一个更高的地址，从而实现连续的数据流。这意味着只要时钟继续，就可以用一条指令访问整个内存。该指令通过驱动 /CS 为高电平来完成。

读数据指令序列如图 14 所示。如果在擦除、编程或写周期正在进行（BUSY=1）时发出读数据指令，该指令将被忽略并且不会有任何对当前周期的影响。

读取数据 (03h) 指令仅在标准 SPI 模式下受支持。

需注意的是，W25Q256 系列应发送地址为 4 字节（即 32-bit）

```
void W25QXX_Read(uint8_t* pBuffer, uint32_t ReadAddr, uint16_t NumByteToRead)
{
  uint16_t i;
  W25QXX_CS_Enable;                              //使能器件
   SPI1_ReadWriteByte(W25X_ReadData);          //发送读取命令
    if(W25QXX_TYPE==W25Q256)                    //如果是W25Q256的话地址为4字节的，要发送最高8位
    {
        SPI1_ReadWriteByte((u8)((ReadAddr)>>24));
    }
    SPI1_ReadWriteByte((uint8_t)((ReadAddr)>>16));   //发送24bit地址
    SPI1_ReadWriteByte((uint8_t)((ReadAddr)>>8));
    SPI1_ReadWriteByte((uint8_t)ReadAddr);
    for(i=0; i<NumByteToRead; i++)
  {
        pBuffer[i]=SPI1_ReadWriteByte(0XFF);    //循环读数
    }
  W25QXX_CS_Disable;
}
```

2 W25QXX 写使能

8.2.1 Write Enable (06h)

The Write Enable instruction (Figure 5) sets the Write Enable Latch (WEL) bit in the Status Register to a 1. The WEL bit must be set prior to every Page Program, Quad Page Program, Sector Erase, Block Erase, Chip Erase, Write Status Register and Erase/Program Security Registers instruction. The Write Enable instruction is entered by driving /CS low, shifting the instruction code "06h" into the Data Input (DI) pin on the rising edge of CLK, and then driving /CS high.

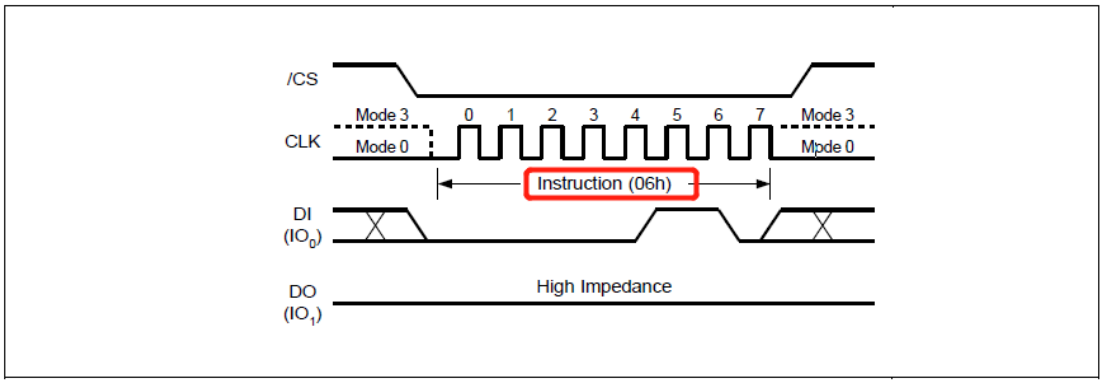


Figure 5. Write Enable Instruction for SPI Mode

写启用指令（图 5）将状态寄存器中的写启用闩锁（WEL）位设置为 1。WEL 位必须在每个页面程序、四页程序、扇区擦除、块擦除、芯片擦除、写入状态寄存器和擦除/程序安全寄存器指令之前设置。写入启用指令是通过驱动/CS 低位，将指令代码“06h”移动到 CLK 上升沿上的数据输入（DI）引脚，然后驱动/CS 高位来输入的。

```
//W25QXX写使能
//将WEL置位
void W25QXX_Write_Enable(void)
{
  W25QXX_CS_Enable;                           //使能器件
  SPI1_ReadWriteByte(W25X_WriteEnable);  //发送写使能
  W25QXX_CS_Disable;                          //取消片选
}
```

3 W25QXX 写禁止

8.2.3 Write Disable (04h)

The Write Disable instruction (Figure 7) resets the Write Enable Latch (WEL) bit in the Status Register to a 0. The Write Disable instruction is entered by driving /CS low, shifting the instruction code "04h" into the DI pin and then driving /CS high. Note that the WEL bit is automatically reset after Power-up and upon completion of the Write Status Register, Erase/Program Security Registers, Page Program, Quad Page Program, Sector Erase, Block Erase, Chip Erase and Reset instructions.

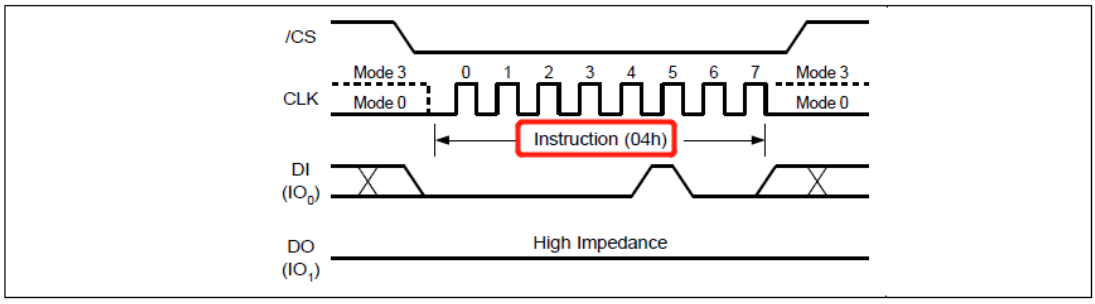


Figure 7. Write Disable Instruction for SPI Mode

```
//W25QXX写禁止
//将WEL清零
void W25QXX_Write_Disable(void)
{
  W25QXX_CS_Enable;                                  //使能器件
  SPI1_ReadWriteByte(W25X_WriteDisable);  //发送写禁止指令
  W25QXX_CS_Disable;                                 //取消片选
}
```

4 读状态寄存器

8.2.4 Read Status Register-1 (05h), Status Register-2 (35h) & Status Register-3 (15h)

The Read Status Register instructions allow the 8-bit Status Registers to be read. The instruction is entered by driving /CS low and shifting the instruction code "05h" for Status Register-1, "35h" for Status Register-2 or "15h" for Status Register-3 into the DI pin on the rising edge of CLK. The status register bits are then shifted out on the DO pin at the falling edge of CLK with most significant bit (MSB) first as shown in Figure 8. Refer to section 7.1 for Status Register descriptions.

The Read Status Register instruction may be used at any time, even while a Program, Erase or Write Status Register cycle is in progress. This allows the BUSY status bit to be checked to determine when the cycle is complete and if the device can accept another instruction. The Status Register can be read continuously, as shown in Figure 8. The instruction is completed by driving /CS high.

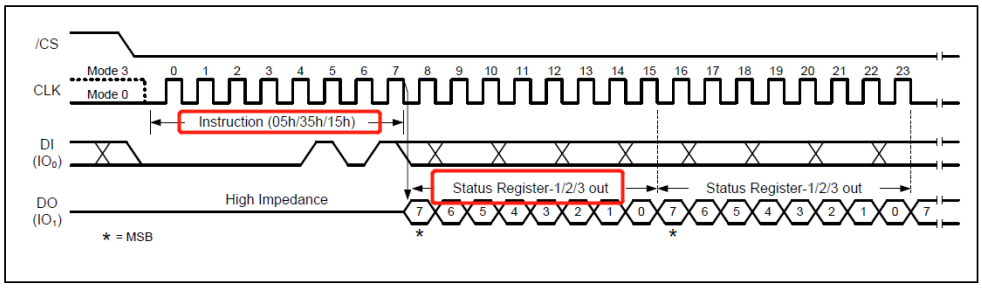


Figure 8a. Read Status Register Instruction

读取状态寄存器指令允许读取 8 位状态寄存器。通过将/CS 驱动为低电平并将状态寄存器 1 的指令代码“05h”、状态寄存器 2 的指令代码“35h”或状态寄

存器 3 的指令代码“15h”在 CLK 的上升沿移入 DI 引脚来输入指令。然后状态寄存器位在 CLK 的下降沿移出 DO 引脚，最高有效位 (MSB) 在前，如图 8 所示。有关状态寄存器的说明，请参阅第 7.1 节。

读取状态寄存器指令可以在任何时候使用，即使是在编程、擦除或写入状态寄存器周期正在进行时。这允许检查 BUSY 状态位以确定周期何时完成以及设备是否可以接受另一条指令。可以连续读取状态寄存器，如图 8 所示。该指令通过驱动/CS 为高电平来完成。

状态寄存器 1

字节序列	1	2	3	4	5	6	7	8
名称	SRP	SEC	TB	BP2	BP1	BP0	WEL	BUSY
功能								

BUSY — 当设备执行页面程序、四页程序、扇区擦除、块擦除、芯片擦除、写入状态寄存器或擦除/程序安全寄存器指令时，该位被设置为 1 状态。当程序、擦除或写入状态/安全寄存器指令完成时，忙碌位将被清除为 0 状态，表明设备已准备好接受进一步指令。

```
//读取W25QXX的状态寄存器，W25QXX一共有3个状态寄存器
uint8_t W25QXX_ReadSR(uint8_t regno)
{
  uint8_t byte=0,command=0;
    switch(regno)
    {
        case 1:
            command=W25X_ReadStatusReg1;    //读状态寄存器1指令
            break;
        case 2:
            command=W25X_ReadStatusReg2;    //读状态寄存器2指令
            break;
        case 3:
            command=W25X_ReadStatusReg3;    //读状态寄存器3指令
            break;
        default:
            command=W25X_ReadStatusReg1;
            break;
    }
  W25QXX_CS_Enable;                            //使能器件
  SPI1_ReadWriteByte(command);               //发送读取状态寄存器命令
  byte=SPI1_ReadWriteByte(0Xff);             //读取一个字节
  W25QXX_CS_Disable;                           //取消片选
  return byte;
}

//等待空闲
void W25QXX_Wait_Busy(void)
{
  while((W25QXX_ReadSR(1)&0x01)==0x01);   // 等待BUSY位清空
}
```

需注意的是 W25Q256 的状态寄存器 3 最低位与其它 W25Qxx 系列不同。

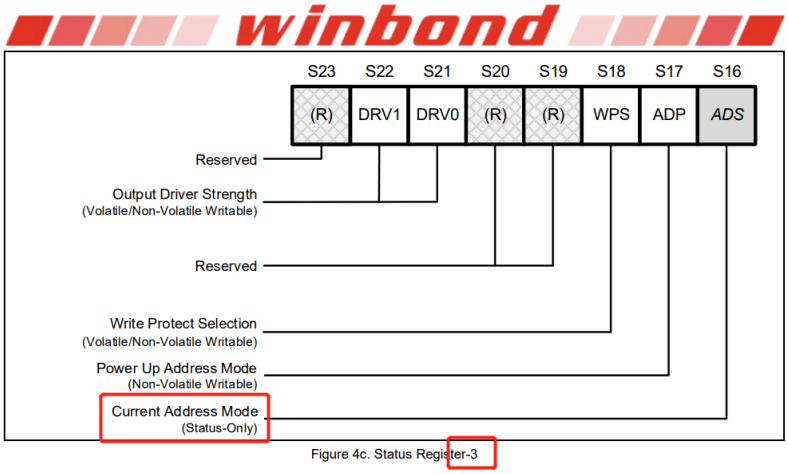


Figure 4c. Status Register-3

ADS — 当前地址模式位是状态寄存器-3 中的只读位，指示设备当前运行的地址模式。当 ADS=0 时，设备处于 3 字节地址模式，当 ADS=1 时，设备处于 4 字节地址模式。

5 扇区擦除

8.2.15 Sector Erase (20h)

The Sector Erase instruction sets all memory within a specified sector (4K-bytes) to the erased state of all 1s (FFh). A Write Enable instruction must be executed before the device will accept the Sector Erase Instruction (Status Register bit WEL must equal 1). The instruction is initiated by driving the /CS pin low and shifting the instruction code "20h" followed a 24-bit sector address (A23-A0). The Sector Erase instruction sequence is shown in Figure 31a.

The /CS pin must be driven high after the eighth bit of the last byte has been latched. If this is not done the Sector Erase instruction will not be executed. After /CS is driven high, the self-timed Sector Erase instruction will commence for a time duration of t_{SE} (See AC Characteristics). While the Sector Erase cycle is in progress, the Read Status Register instruction may still be accessed for checking the status of the BUSY bit. The BUSY bit is a 1 during the Sector Erase cycle and becomes a 0 when the cycle is finished and the device is ready to accept other instructions again. After the Sector Erase cycle has finished the Write Enable Latch (WEL) bit in the Status Register is cleared to 0. The Sector Erase instruction will not be executed if the addressed page is protected by the Block Protect (CMP, SEC, TB, BP2, BP1, and BP0) bits or the Individual Block/Sector Locks.

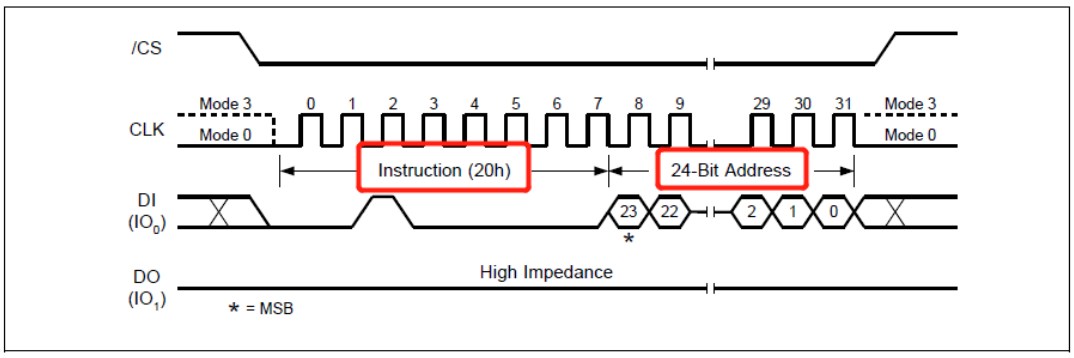


Figure 31a. Sector Erase Instruction

扇区擦除指令将指定扇区（4K 字节）内的所有存储器设置为全 1 (FFh) 的擦除状态。在设备接受扇区擦除指令之前必须执行写使能指令（状态寄存器位

WEL 必须等于 1）。该指令通过将 /CS 引脚驱动为低电平并将指令代码“20h”移动到 24 位扇区地址 (A23-A0) 来启动。扇区擦除指令序列如图 31a 所示。

在最后一个字节的第 8 位被锁存后，/CS 引脚必须被驱动为高电平。如果不这样做，则不会执行扇区擦除指令。 /CS 被驱动为高电平后，自定时扇区擦除指令将开始持续 tSE 时间（参见 AC 特性）。当扇区擦除周期正在进行时，仍可以访问读取状态寄存器指令以检查 BUSY 位的状态。 BUSY 位在扇区擦除周期内为 1，当周期结束且器件准备好再次接受其他指令时变为 0。扇区擦除周期完成后，状态寄存器中的写使能锁存器 (WEL) 位清零。如果寻址页受到块保护（CMP、SEC、TB、BP2）的保护，则不会执行扇区擦除指令、BP1 和 BP0) 位或单个块/扇区锁定。

```
//擦除一个扇区
//Dst Addr:扇区地址 根据实际容量设置
//擦除一个扇区的最少时间:150ms
void W25QXX Erase Sector(uint32 t Dst Addr)
{
  //监视falsh擦除情况,测试用
  //printf("fe:%x\r\n",Dst Addr);
  Dst Addr*=4096;
    W25QXX Write Enable();                  //SET WEL
    W25QXX Wait Busy();
    W25QXX CS Enable();                          //使能器件
    SPI1 ReadWriteByte(W25X SectorErase);   //发送扇区擦除指令
    if(W25QXX TYPE==W25Q256)                //如果是W25Q256的话地址为4字节的，要发送最高8位
    {
        SPI1 ReadWriteByte((uint8 t)((Dst Addr)>>24));
    }
    SPI1 ReadWriteByte((uint8 t)((Dst Addr)>>16));  //发送24bit地址
    SPI1 ReadWriteByte((uint8 t)((Dst Addr)>>8));
    SPI1 ReadWriteByte((uint8 t)Dst Addr);
    W25QXX CS Disable();                         //取消片选
    W25QXX Wait Busy();               //等待擦除完成
}
```

6 写操作

写操作较为复杂，主要是因为写之前要擦除、最大一次性只能写 256 Byte（即一个完整的 page，硬件不会自动换 page，所以要编程）所以要考虑换页、写操作给的指令中的起始地址再某一页的哪个位置等等

8.2.13 Page Program (02h)

The Page Program instruction allows from one byte to 256 bytes (a page) of data to be programmed at previously erased (FFh) memory locations. A Write Enable instruction must be executed before the device will accept the Page Program Instruction (Status Register bit WEL= 1). The instruction is initiated by driving the /CS pin low then shifting the instruction code "02h" followed by a 24-bit address (A23-A0) and at least one data byte, into the DI pin. The /CS pin must be held low for the entire length of the instruction while data is being sent to the device. The Page Program instruction sequence is shown in Figure 29.

If an entire 256 byte page is to be programmed, the last address byte (the 8 least significant address bits) should be set to 0. If the last address byte is not zero, and the number of clocks exceeds the remaining page length, the addressing will wrap to the beginning of the page. In some cases, less than 256 bytes (a partial page) can be programmed without having any effect on other bytes within the same page. One condition to perform a partial page program is that the number of clocks cannot exceed the remaining page length. If more than 256 bytes are sent to the device the addressing will wrap to the beginning of the page and overwrite previously sent data.

As with the write and erase instructions, the /CS pin must be driven high after the eighth bit of the last byte has been latched. If this is not done the Page Program instruction will not be executed. After /CS is driven high, the self-timed Page Program instruction will commence for a time duration of tpp (See AC Characteristics). While the Page Program cycle is in progress, the Read Status Register instruction may still be accessed for checking the status of the BUSY bit. The BUSY bit is a 1 during the Page Program cycle and becomes a 0 when the cycle is finished and the device is ready to accept other instructions again. After the Page Program cycle has finished the Write Enable Latch (WEL) bit in the Status Register is cleared to 0. The Page Program instruction will not be executed if the addressed page is protected by the Block Protect (CMP, SEC, TB, BP2, BP1, and BP0) bits or the Individual Block/Sector Locks.

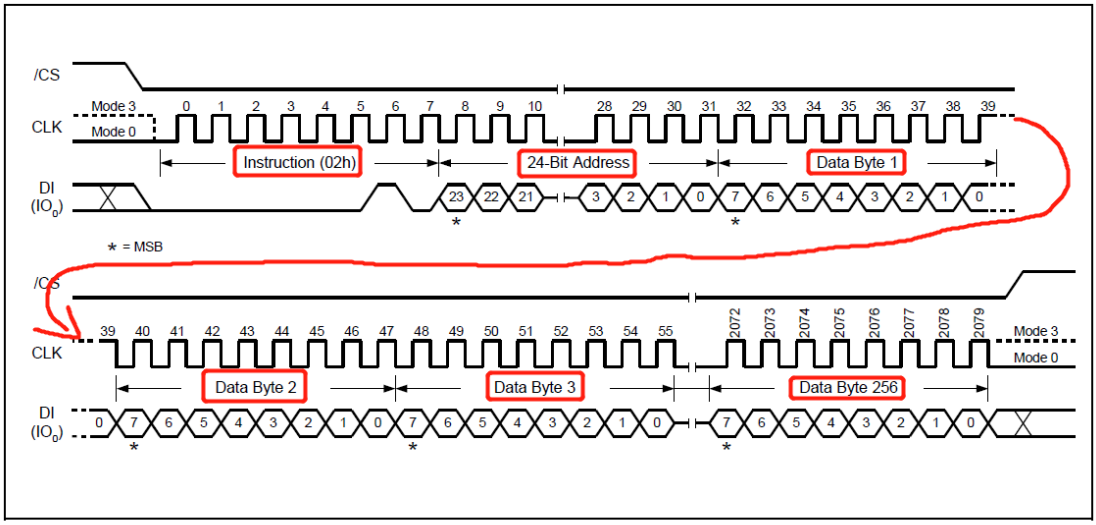


Figure 29a. Page Program Instruction

页编程指令允许在先前已擦除(FFh)的存储器位置对 1 个字节到 256 个字节（一页）的数据进行编程。在设备接受页编程指令（状态寄存器位 WEL=1）之前，必须执行写使能指令。该指令通过将/CS 引脚驱动为低电平，然后将指令代码“02h”、后跟 24 位地址(A23-A0)和至少一个数据字节移入 DI 引脚来启动。在向器件发送数据时，/CS 引脚必须在整个指令长度内保持低电平。

如果要对整个 256 字节页面进行编程，则最后一个地址字节（8 个最低有效地址位）应设置为 0。如果最后一个地址字节不为零，并且时钟数超过剩余页面长度，则寻址将换行到页面的开头。在某些情况下，可以对少于 256 个字节（部分页面）进行编程，而不会对同一页面中的其他字节产生任何影响。执行部分页面编程的一个条件是时钟数不能超过剩余页面长度。如果发送到设备的字节数超过 256 个，则寻址将换行到页面的开头并覆盖先前发送的数据。

与写入和擦除指令一样，在最后一个字节的第 8 位被锁存后，/CS 引脚必须被驱动为高电平。如果不这样做，页面编程指令将不会被执行。在/CS 被驱动为高电平后，自定时页面编程指令将开始持续 tpp 时间（参见 AC 特性）。在页面编程周期正在进行时，仍可以访问读取状态寄存器指令以检查 BUSY 位的状态。BUSY 位在页编程周期内为 1，当周期结束且器件准备好再次接受其他指令时变为 0。页面编程周期完成后，状态寄存器中的写使能锁存器 (WEL) 位被清零。如果寻址页面受到块保护（CMP、SEC、TB、BP2）的保护，则不会执行页面编程指令、BP1 和 BP0)位或单个块/扇区锁定。

```
//SPI在一页(0~65535)内写入少于256个字节的数据
//在指定地址开始写入最大256字节的数据
//pBuffer:数据存储区
//WriteAddr:开始写入的地址(24bit)
//NumByteToWrite:要写入的字节数(最大256),该数不应该超过该页的剩余字节数!!!
void W25QXX Write Page(uint8 t* pBuffer,uint32 t WriteAddr,uint16 t NumByteToWrite)
{
  uint16 t i;
    W25QXX Write Enable();                  //SET WEL
  W25QXX CS Enable;                           //使能器件
    SPI1 ReadWriteByte (W25X PageProgram );   //发送写页命令
    if(W25QXX TYPE==W25Q256)                //如果是W25Q256的话地址为4字节的，要发送最高8位
    {
        SPI1 ReadWriteByte ((uint8 t)((WriteAddr)>>24));
    }
    SPI1 ReadWriteByte ((uint8 t)((WriteAddr)>>16)); //发送24bit地址
    SPI1 ReadWriteByte ((uint8 t)((WriteAddr)>>8));
    SPI1 ReadWriteByte ((uint8 t)WriteAddr);
    for(i=0;i<NumByteToWrite;i++)SPI1 ReadWriteByte (pBuffer[i]);//循环写数
  W25QXX CS Disable;                              //取消片选
  W25QXX Wait Busy();             //等待写入结束
}

//无检验写SPI FLASH
//必须确保所写的地址范围内的数据全部为0XFF,否则在非0XFF处写入的数据将失败!
//具有自动换页功能
//在指定地址开始写入指定长度的数据,但是要确保地址不越界!
//pBuffer:数据存储区
//WriteAddr:开始写入的地址(24bit)
//NumByteToWrite:要写入的字节数(最大65535)
//CHECK OK
void W25QXX_Write_NoCheck(uint8_t* pBuffer,uint32_t WriteAddr,uint16_t NumByteToWrite)
{
  uint16_t pageremain;
  pageremain=256-WriteAddr%256; //单页剩余的字节数
  if(NumByteToWrite<=pageremain)pageremain=NumByteToWrite;//不大于256个字节
  while(1)
  {
    W25QXX_Write_Page(pBuffer,WriteAddr,pageremain);
    if(NumByteToWrite==pageremain)break;//写入结束了
    else //NumByteToWrite>pageremain
    {
      pBuffer+=pageremain;
      WriteAddr+=pageremain;

      NumByteToWrite-=pageremain;       //减去已经写入了的字节数
      if(NumByteToWrite>256)pageremain=256; //一次可以写入256个字节
      else pageremain=NumByteToWrite;     //不够256个字节了
    }
  };
}
```

```
//写SPI FLASH
//在指定地址开始写入指定长度的数据
//该函数带擦除操作!
//pBuffer:数据存储区
//WriteAddr:开始写入的地址(24bit)
//NumByteToWrite:要写入的字节数(最大65535)
uint8_t W25QXX_BUFFER[4096];
void W25QXX_Write(uint8_t* pBuffer,uint32_t WriteAddr,uint16_t NumByteToWrite)
{
  uint32_t secpos;
  uint16_t secoff;
  uint16_t secremain;
  uint16_t i;
  uint8_t * W25QXX_BUF;
    W25QXX_BUF=W25QXX_BUFFER;
  secpos=WriteAddr/4096;//扇区地址
  secoff=WriteAddr%4096;//在扇区内的偏移
  secremain=4096-secoff;//扇区剩余空间大小
  //printf("ad:%X,nb:%X\r\n",WriteAddr,NumByteToWrite);//测试用
  if(NumByteToWrite<=secremain)
    secremain=NumByteToWrite;//不大于4096个字节
  while(1)
  {
    W25QXX_Read(W25QXX_BUF,secpos*4096,4096);//读出整个扇区的内容
    for(i=0;i<secremain;i++)//校验数据
    {
      if(W25QXX_BUF[secoff+i]!=0XFF)break;//需要擦除
    }
    if(i<secremain)//需要擦除
    {
      W25QXX_Erase_Sector(secpos);//擦除这个扇区
      for(i=0;i<secremain;i++)     //复制
      {
        W25QXX_BUF[i+secoff]=pBuffer[i];
      }
      W25QXX_Write_NoCheck(W25QXX_BUF,secpos*4096,4096);//写入整个扇区

    }
    else W25QXX_Write_NoCheck(pBuffer,WriteAddr,secremain);//写已经擦除了的,直接写入扇区剩余区间.

    if(NumByteToWrite==secremain)break;//写入结束了
    else//写入未结束
    {
      secpos++;//扇区地址增1
      secoff=0;//偏移位置为0

        pBuffer+=secremain;  //指针偏移
      WriteAddr+=secremain;//写地址偏移
        NumByteToWrite-=secremain;          //字节数递减
      if(NumByteToWrite>4096)secremain=4096; //下一个扇区还是写不完
      else secremain=NumByteToWrite;       //下一个扇区可以写完了
    }
  };
```

为了完成写操作，共定义了 3 个函数，“页面写操作（限定小于 256 byte）”、“带换页的写操作（没有进行擦除操作）”，以及给用户直接调用的“写操作（带擦除）”。代码较为复杂，这里大致介绍一下思路：

写操作 W25QXX_Write（）

1 计算用户想写入起始地址在第几扇区，及在扇区内的偏移地址（因为擦除操作最小只能擦除一个扇区，即4KB，而不是1页(256 byte)）

读出要写入到的起始扇区内的全部内容，放到缓冲区buffer里，用于判断是否需要擦除（由于W25Qxx系列可擦除次数有限，所以判断是否需要擦除再擦除）

如果当前扇区需要擦除，则调用W25QXX_Erase_Sector()函数

W25QXX_Erase_Sector()函数

W25QXX_Erase_Sector里需要注意的问题是有一行代码"Dst_Addr*=4096;"这是因为调用该函数的，只有W25QXX_Write()，传的参数是"第几个扇区"，而不是"第几个扇区的起始地址"

W25QXX_Erase_Sector()函数剩下的内容就很简单了，就是扇区擦除那部分的内容

擦除完之后，把用户想要写入的内容写入到FLASH里（同时保护了偏移地址前的内容）

该操作的原理是：W25QXX_BUF缓存区保存了该起始扇区内原有的所有内容，而只把用户想写入的内容，覆盖偏移地址secoff即以后的内容（画了张图帮助理解）（虽然保护的这部分数据可能没用吧）

如果不需要擦除，就直接写入当前扇区，调用函数 W25QXX_Write_NoCheck

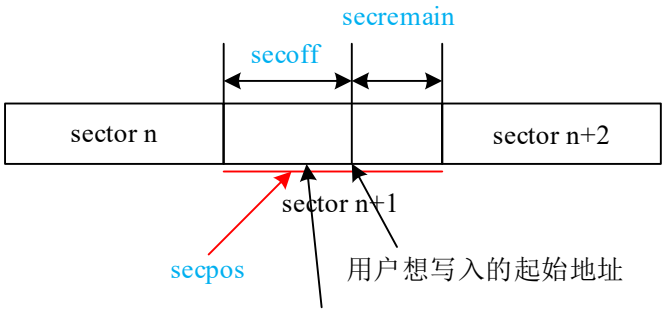
W25QXX_Write_NoCheck()函数

W25QXX_Write_NoCheck()的逻辑也很简单，重复256 byte写入过程

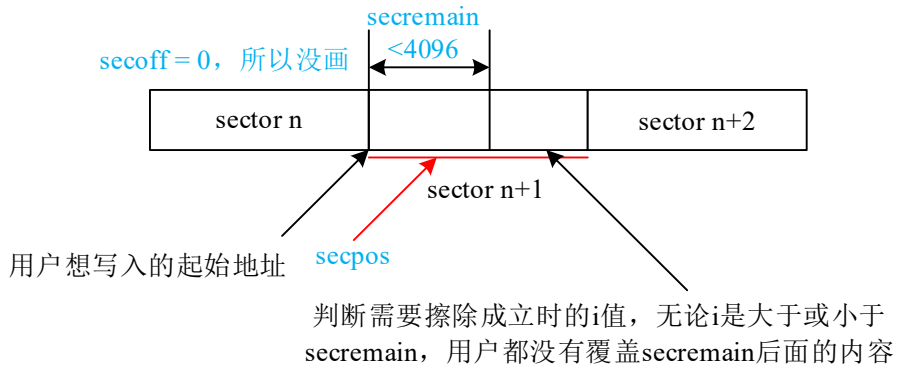
2 如果只需要写1个扇区，那么break语句会在步骤1完成后，跳出while(1)，使得 W25QXX_Write()函数结束

3 如果需要写多个扇区，则 指向下一扇区，要写入的地址也移动，要写入的字节数减掉已写入的数，判断写入写一个扇区还是要写入多个扇区，最后重复步骤1

需要注意的是，在步骤1中，保护扇区内原有的内容的代码，是可以保护最后一个扇区中，不需要被覆盖的内容的



判断需要擦除成立时的i值，如果i<secoff，即意味着 i到secoff这部分有之前写入的数据



不得不说，原子哥写程序的逻辑是有点儿绕 2333