目录

第12章 Cortex-M4-SPI 总线

12.1 SPI 总线概述

uart 通信特征：异步串行全双工----→全双工-----→就是说可以同时发送一个进行接收。

IIC 通信特征：同步串行半双工---→只能同一个时刻进行单方面的接收或发送。

必须要掌握的三种通信协议：uart、iic、spi SPI--→同步串行全双工通信。

12.1.1 SPI 总线介绍

SPI(Serial Peripheral interface)：是由 Motorola 公司开发的串行外围设备接口，是一种高速的，全双工，同步的通信总线。主要应用在 EEPROM，FLASH，实时时钟，AD 转换器，还有数字信号处理器和数字信号解码器等器件。

SPI 通信特征：同步串行全双工 25Mhz

FLASH：具备掉电不丢失数据 读写速度快 特征：只能写 0(将 1 变为 0)不能写 1(不能将 0 变为 1)

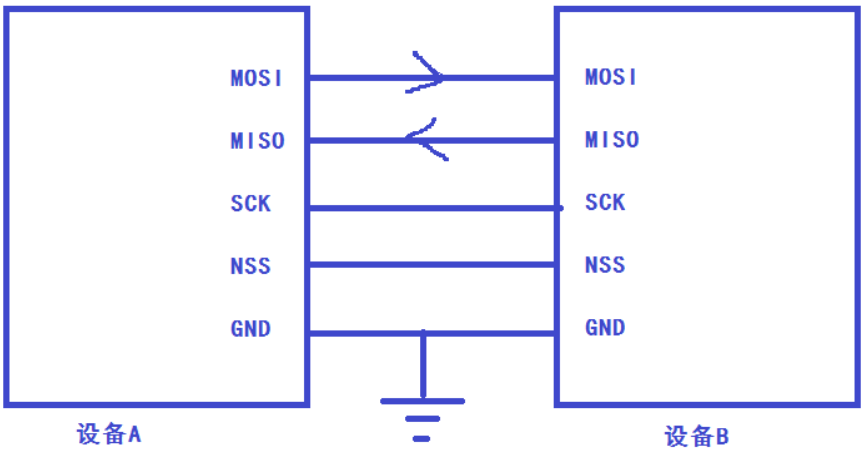
写之前必须要擦除要写的区域—>操作的话相当于写数据，必须进行擦除操作。 擦除之后的区域每个位都会变成 1

1001 1011

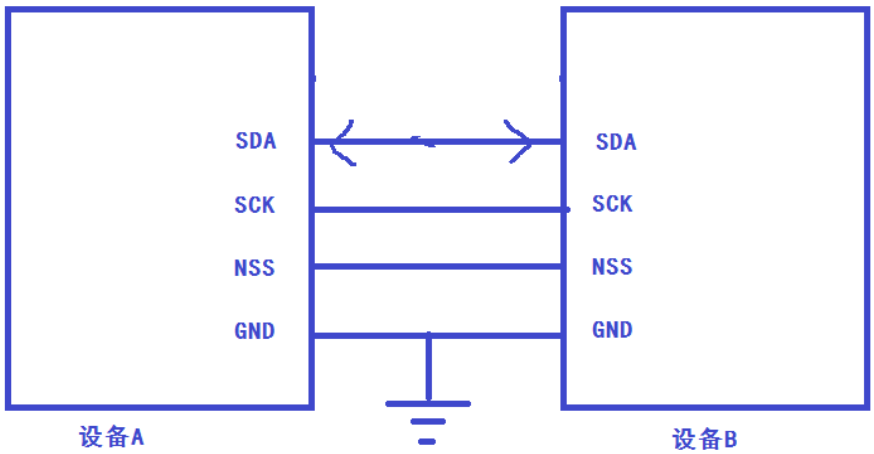
1001 1011

12.1.2 SPI 总线接口与物理拓扑结构

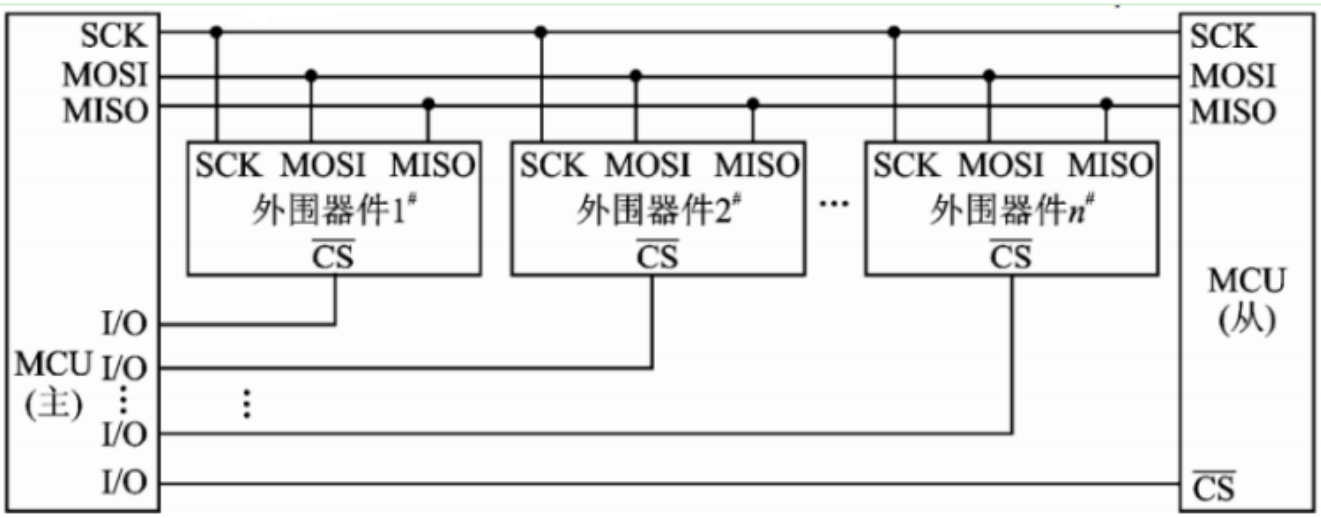
（1）五线制接口（4 线 spi）全双工



（2）四线制接口（3 线 spi）半双工



（3）4 线 SPI 拓扑结构图



在 SPI 总线上，主机只有一个，从机有多个(一主多从)

通信时，主机只能选择其中的一个从机进行通信

从机不可以主动发送数据给主机

时钟线都是由主机控制（时钟线是拉高还是拉低由主机控制）

MOSI：串行数据线，主机输出，从机输入-→

MISO：串行数据线，从机输出，主机输入

SCK：串行时钟线，时钟线控制数据的传输，由主机控制　从机与从机之间不能进行通信→SCK 有主机控制

NSS/CS：从设备片选，主机选择从机进行通信-→主从模式的选择

12.1.3 SPI 总线通信原理

1、主机拉低片选(激活从机)

2、主机控制时钟线（SCK）产生上升沿(下降沿)发送方发送数据-→具体是上升沿还是下降沿需要通过寄存器配置

3、主机控制时钟线（SCK）产生下降沿(上升沿)接收方采集数据→通过相应的寄存器进行配置。

….

主机拉高片选(通信结束)

12.1.4 SPI 总线数据格式

uart 数据帧格式：起始位（1 位）+数据位（8 位）+校验位+停止位

iic 数据帧格式：起始条件+数据位+应答位+停止条件

spi 数据帧格式：有四种模式 MODE 0 – 3

数据帧格式的决定因素：时钟线什么边沿采集数据，时钟线空闲电平是什么电平

CPHA：时钟相位，决定时钟线什么边沿采集数据

1：后沿采集数据---相当于第二个边缘

0：前沿采集数据

CPOL：时钟极性，决定时钟线的空闲电平---时钟线空闲时电平状态

1：空闲电平是高电平

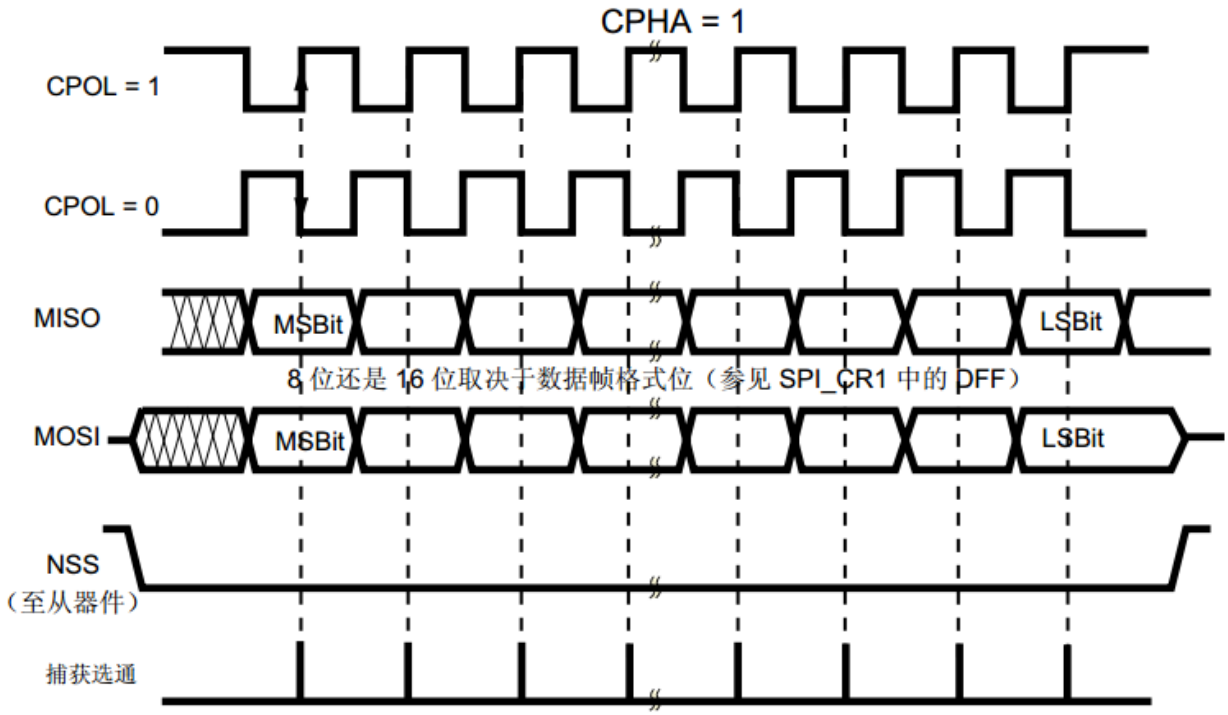
0：空闲电平是低电平

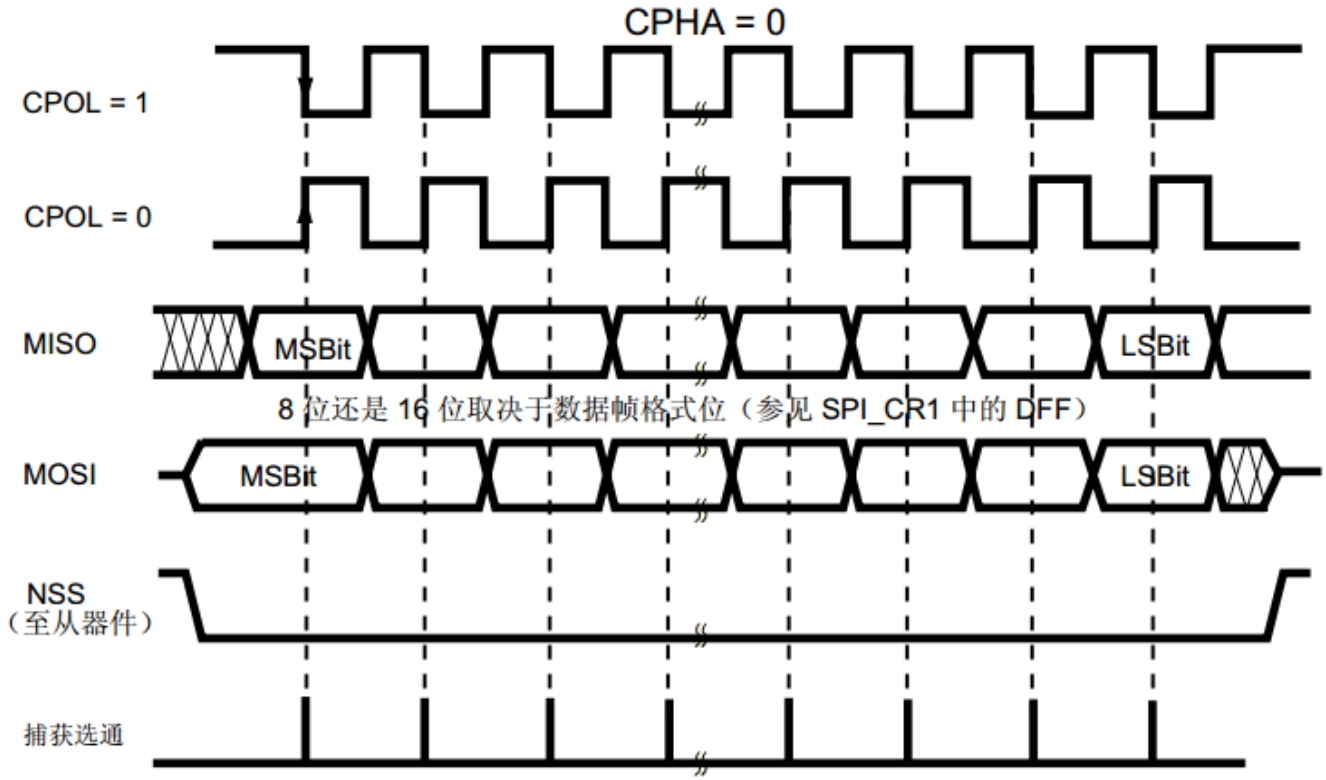
如果 CPHA = 0 CPOL = 0； → MODE 0

前沿：上升沿采集数据 下降沿准备数据

如果 CPHA = 1, CPOL = 1; → MODE 3

后沿：上升沿采集数据 下降沿准备数据





采集数据：肯定是数据稳定的情况下的进行进行的

交叉线：数据可变

平行线：数据稳定

CPHA：时钟相位，决定什么边沿采集数据

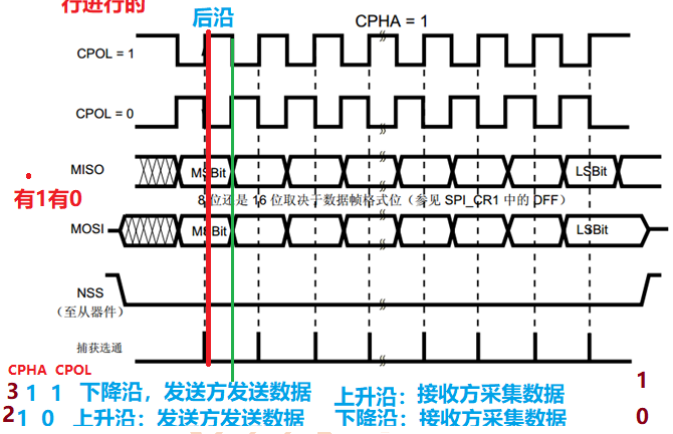
1：后沿采集数据

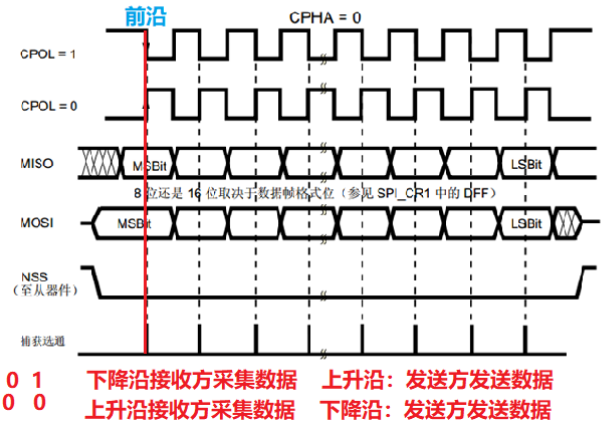
0：前沿采集数据

CPOL：时钟极性，决定时钟线的空闲电平

1：空闲电平是高电平

0：空闲电平是低电平



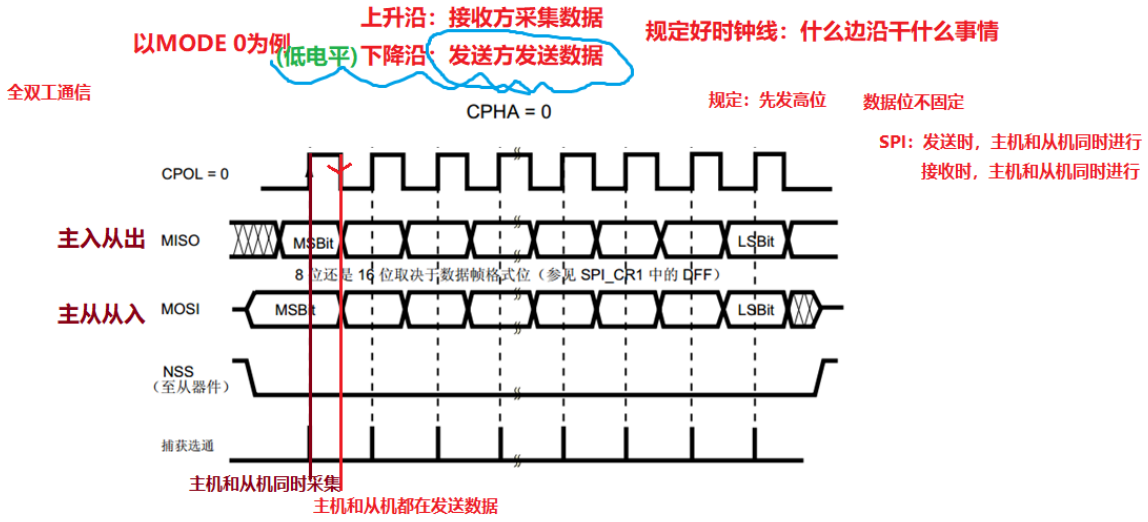


	Leading Edge	Trailing Edge	SPI Mode
CPOL = 0, CPHA = 0	Sample (Rising)	Setup (Falling)	0
CPOL = 0, CPHA = 1	Setup (Rising)	Sample (Falling)	1
CPOL = 1, CPHA = 0	Sample (Falling)	Setup (Rising)	2
CPOL = 1, CPHA = 1	Setup (Falling)	Sample (Rising)	3

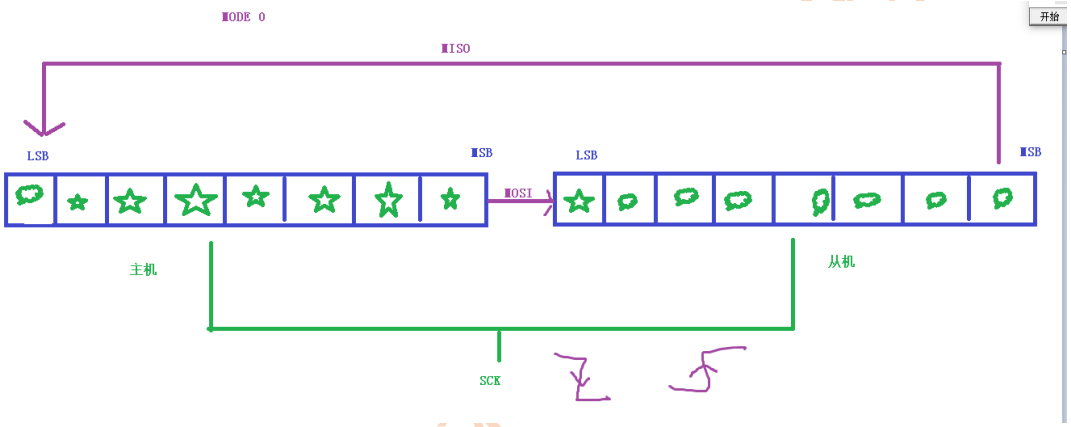
一般情况下支持 MODE 0 设备也支持 MODE3

一般情况下支持 MODE 1 设备也支持 MODE2

以 MODE 0 为例



SPI 通信的过程就是数据交换的过程



伪代码：(只写代码顺序)
以 MODE 0 伪代码
上升沿：接收方采集数据
（低电平）下降沿：发送方发送数据
主机发送一个字节(8bit)数据给从机，并且读取从机的一个字节数据

```
//SCK = 1;//拉高时钟线
for(循环 8 次)
{
   SCK = 0; //产生下降沿   主机/从机发送数据
   MOSI = 0/1;//主机发送数据
   SCK = 1; //产生上升沿   主机/从机都在采集数据
   读取 MISO//主机读取 W25Q64 发送过来的数据
}
```

正常情况下要加时间 W25Q64 来说 80Mhz
80Mhz T = 1/80 us = 12.5ns 得出来的结果 ns
1/168 us = 约等于 6ns 执行一条代码的时间
后面写 SPI 的时序 都不需要添加时间
IIC 对时间要求严格一点

对 W25Q64 的功能要结合芯片手册。

12.2 STM32 的 SPI 控制器(配置相关的寄存器)

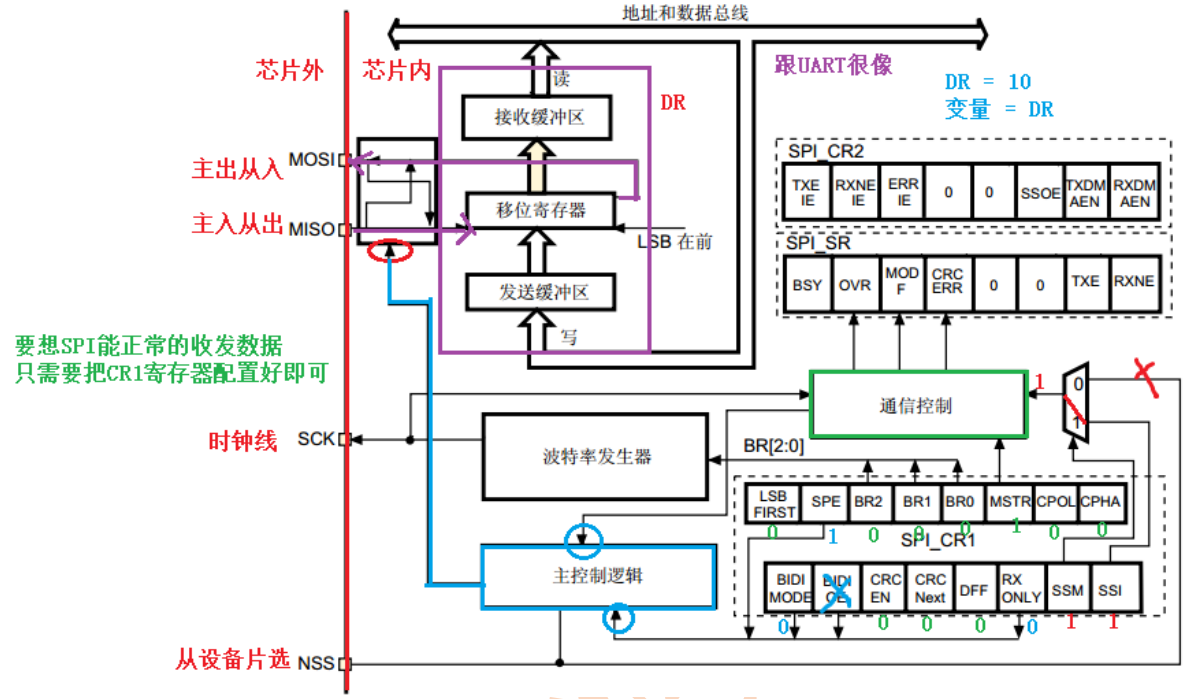
12.2.1 STM32 的 SPI 控制器特征

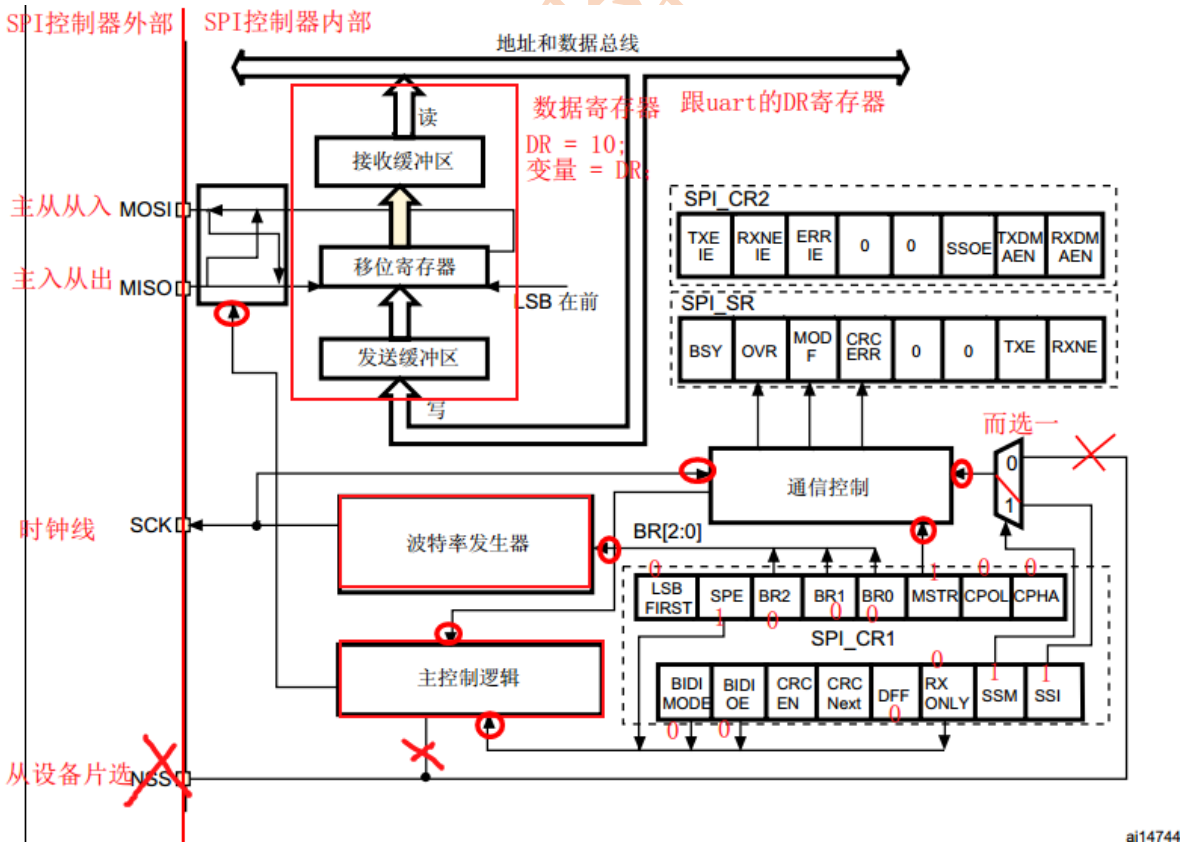
串行外设接口 (SPI) 可与外部器件进行半双工/全双工的同步串行通信。该接口可配置为主模式，在这种情况下，它可为外部从器件提供通信时钟 (SCK)。该接口还能够在多主模式配置下工作。

- 基于三条线的全双工同步传输(MOSI MISO SCK)
- 基于双线的单向同步传输，其中一条可作为双向数据线
- 8 位或 16 位传输帧格式选择
- 主模式或从模式操作
- 多主模式功能
- 8 个主模式波特率预分频器（最大值为 fPCLK/2）
- 从模式频率（最大值为 fPCLK/2）
- 对于主模式和从模式都可实现更快的通信
- 对于主模式和从模式都可通过硬件或软件进行 NSS(从设备片选)管理：动态切换主/从操作
- 可编程的时钟极性和相位(决定 SPI 的通信方式 MODE0 – MODE3)
- 可编程的数据顺序，最先移位 MSB 或 LSB→通过相应的寄存器进行配置
- 可触发中断的专用发送和接收标志
- SPI 总线忙状态标志
- SPI TI 模式
- 用于确保可靠通信的硬件 CRC 功能：
  - 在发送模式下可将 CRC 值作为最后一个字节发送
  - 根据收到的最后一个字节自动进行 CRC 错误校验
- 可触发中断的主模式故障、上溢和 CRC 错误标志
- 具有 DMA 功能的 1 字节发送和接收缓冲器：发送和接收请求

12.2.2 STM32 的 SPI 控制器框架分析

数据寄存器分为 2 个缓冲区，一个用于写入（发送缓冲区），一个用于读取（接收缓冲区）。对数据寄存器执行写操作时，数据将写入发送缓冲区，从数据寄存器执行读取时，将返回接收缓冲区中的值。





12.2.3 STM32 相关寄存器

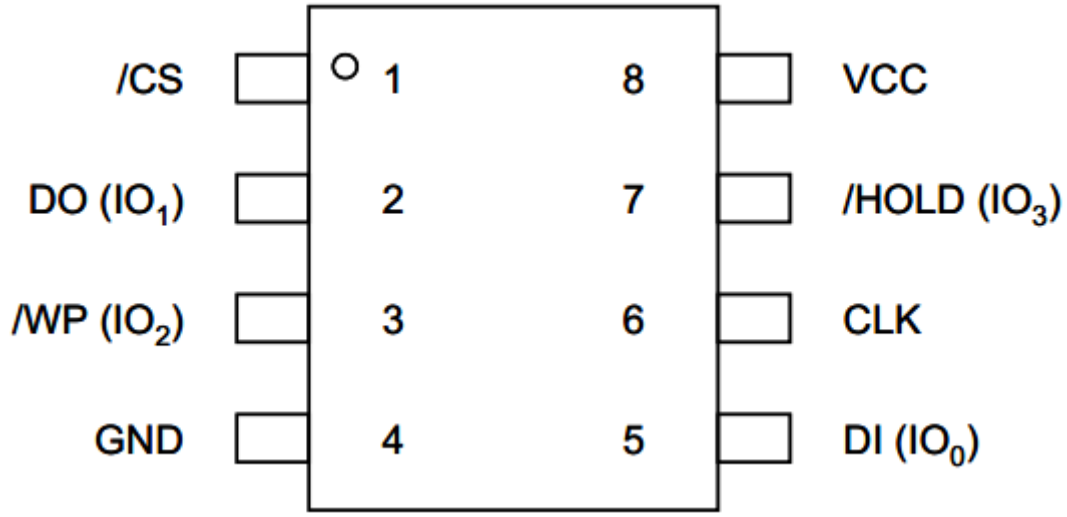
12.3 W25Q64 芯片

12.3.1 芯片介绍

大小：8Mbyte → 8388608byte → 8192kb
一页：256byte
标准通信速度：80Mhz
Sector Erase：扇区擦除 4K(最小的擦除单位) 给它擦除的首地址(只能是 4096 的倍数 0 4096 8192)
1Kbyte = 1024 字节 4K = 4096byte 10 (0～4095) 5000 （4096－8191）
Block Erase：块擦除：32K – 64K
最多擦除：10W 次
数据保存：20 年

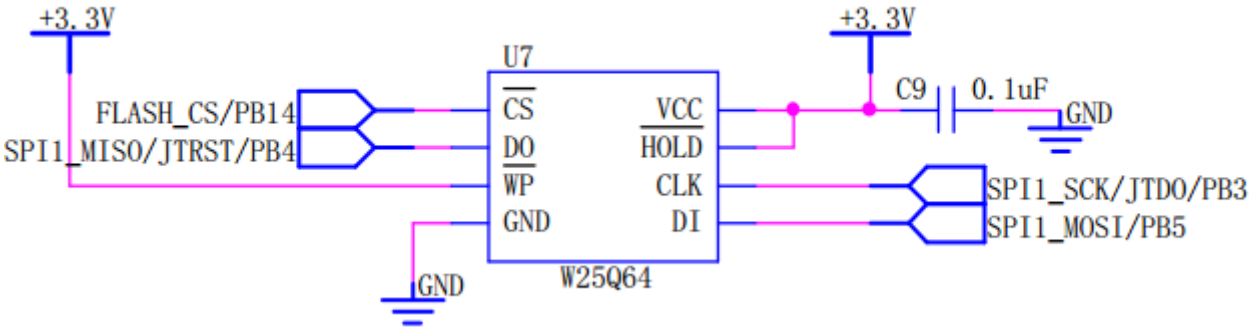
MCU 要想跟外部芯片进行数据交流
1. 相关管脚的连线
2. 满足一定的通信协议

12.3.2 芯片管脚说明



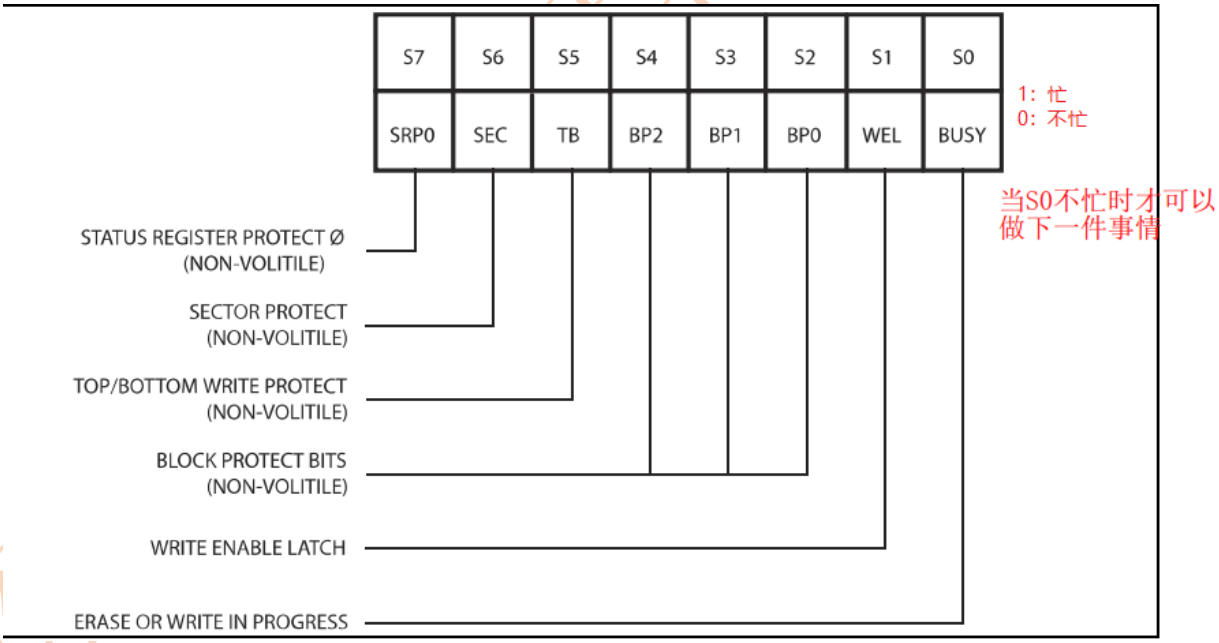
PIN NO.	PIN NAME	I/O	FUNCTION
1	/CS	I	Chip Select Input 片选
2	DO (IO1)	I/O	Data Output (Data Input Output 1)*1 MISO
3	/WP (IO2)	I/O	Write Protect Input (Data Input Output 2)*2
4	GND		Ground
5	DI (IO0)	I/O	Data Input (Data Input Output 0)*1 MOSI
6	CLK	I	Serial Clock Input SCK
7	/HOLD (IO3)	I/O	Hold Input (Data Input Output 3)*2
8	VCC		Power Supply

*1 IO0 and IO1 are used for Standard and Dual SPI instructions



12.3.3 芯片工作原理

利用 SPI 发送指令来操作 W25Q64 的



11.2.2 Instruction Set Table 1 (1)

INSTRUCTION NAME	BYTE 1 (CODE)	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6
Write Enable 写使能	06h	每次写FLASH之前都需要进行写使能				
Write Disable 禁止写使能	04h					
读状态寄存器1 Read Status Register-1	05h	(S7–S0) (2)	来判断W25Q64当前忙不忙			
读状态寄存器2 Read Status Register-2	35h	(S15-S8) (2)				
写状态寄存器 Write Status Register	01h	(S7–S0)	(S15-S8)		待写入的数据	
页写 Page Program	02h	A23–A16	A15–A8	A7–A0	(D7–D0)	
Quad Page Program	32h	A23–A16	A15–A8	A7–A0	(D7–D0, …)(3)	
块擦除 Block Erase (64KB)	D8h	A23–A16	A15–A8	A7–A0		
半块擦除 Block Erase (32KB)	52h	A23–A16	A15–A8	A7–A0		
扇区擦除 Sector Erase (4KB)	20h	A23–A16	A15–A8	A7–A0	要擦除区域的首地址，只能是4096倍数	
整片擦除 Chip Erase	C7h/60h					
Erase Suspend	75h					
Erase Resume	7Ah					

内部地址

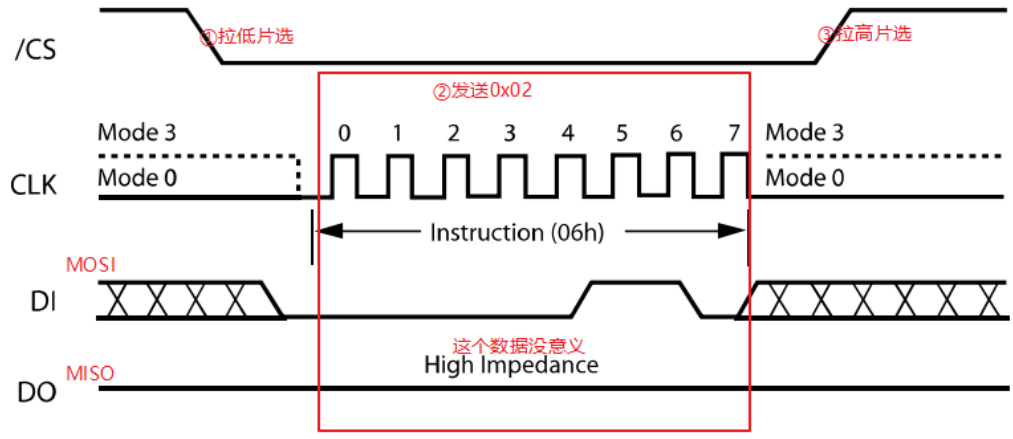
Manufacturer/ Device ID(6) 读ID	90h	dummy	dummy	00h	(MF7-MF0)	(ID7-ID0)	EF16

读 ID：可以验证芯片的真伪，检验 SPI 通信正常不正常

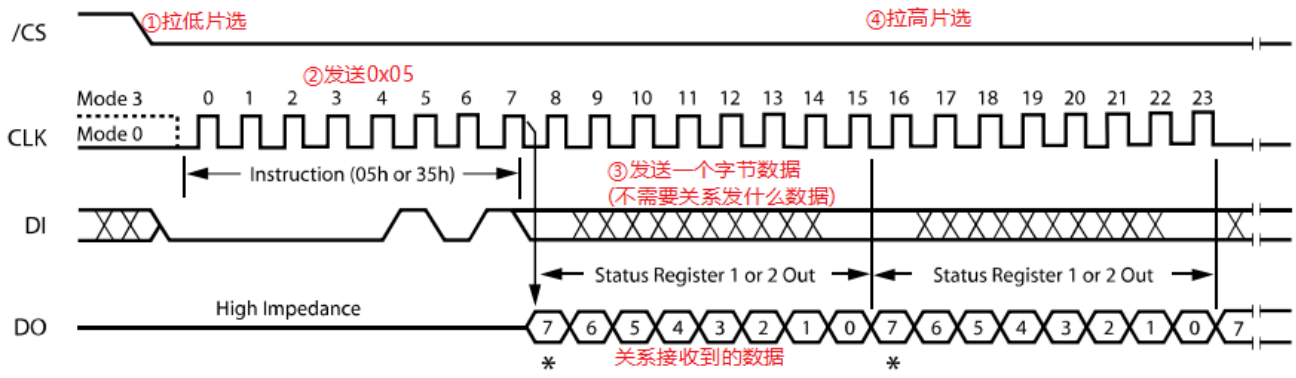
INSTRUCTION NAME	BYTE 1 (CODE)	BYTE 2	BYTE 3	BYTE 4	BYTE 5	BYTE 6
Read Data 读操作	03h	A23-A16	A15-A8	A7-A0	(D7-D0) 发什么无所谓	
Fast Read	0Bh	A23-A16	A15-A8	A7-A0	dummy	(D7-D0)
Fast Read Dual Output	3Bh	A23-A16	A15-A8	A7-A0	dummy	(D7-D0, …)(1)
Fast Read Dual I/O	BBh	A23-A8(2)	A7-A0, M7-M0(2)	(D7-D0, …)(1)		
Fast Read Quad Output	6Bh	A23-A16	A15-A8	A7-A0	dummy	(D7-D0, …)(3)
Fast Read Quad I/O	EBh	A23-A0, M7-M0(4)	(x,x,x,x, D7-D0, …)(5)	(D7-D0, …)(3)		
Octal Word Read Quad I/O(6)	E3h	A23-A0, M7-M0(4)	(D7-D0, …)(3)			

12.3.4 芯片操作时序

1. 写使能

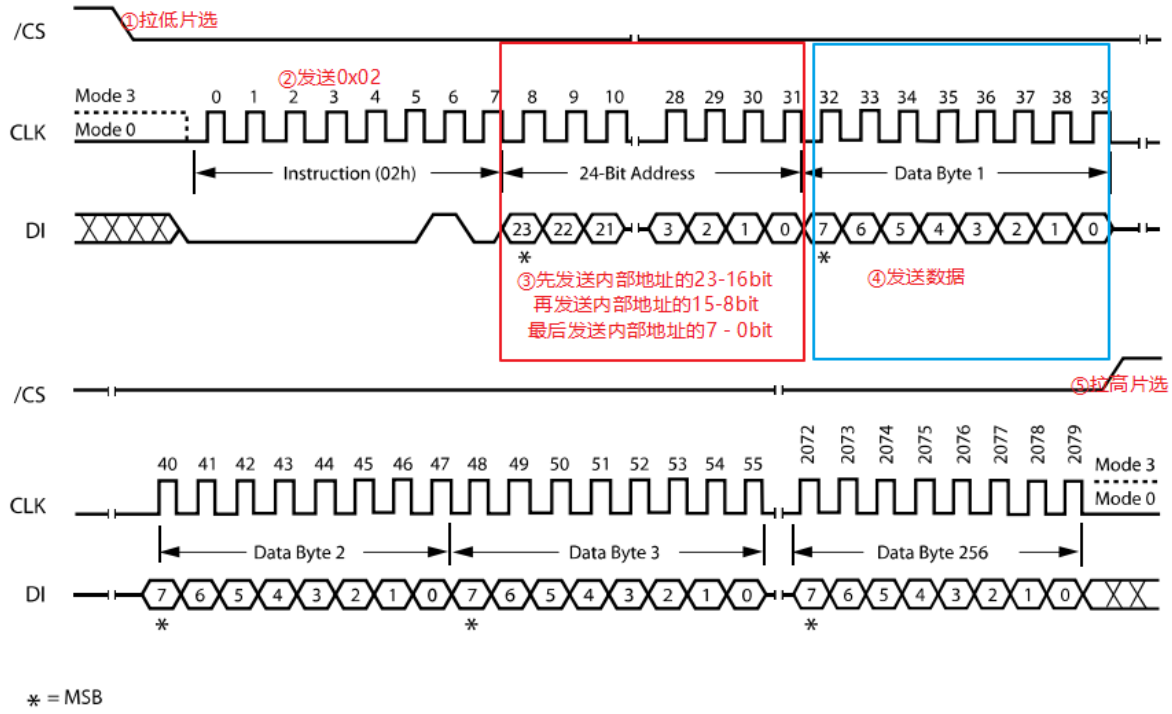


2. 读状态寄存器 1

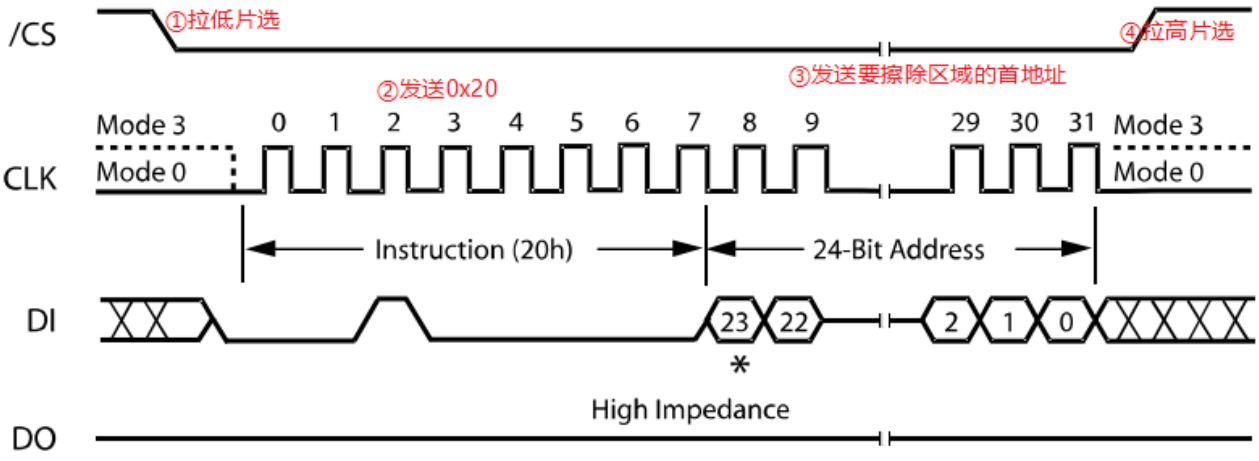


3. 页写 256byte 为一页

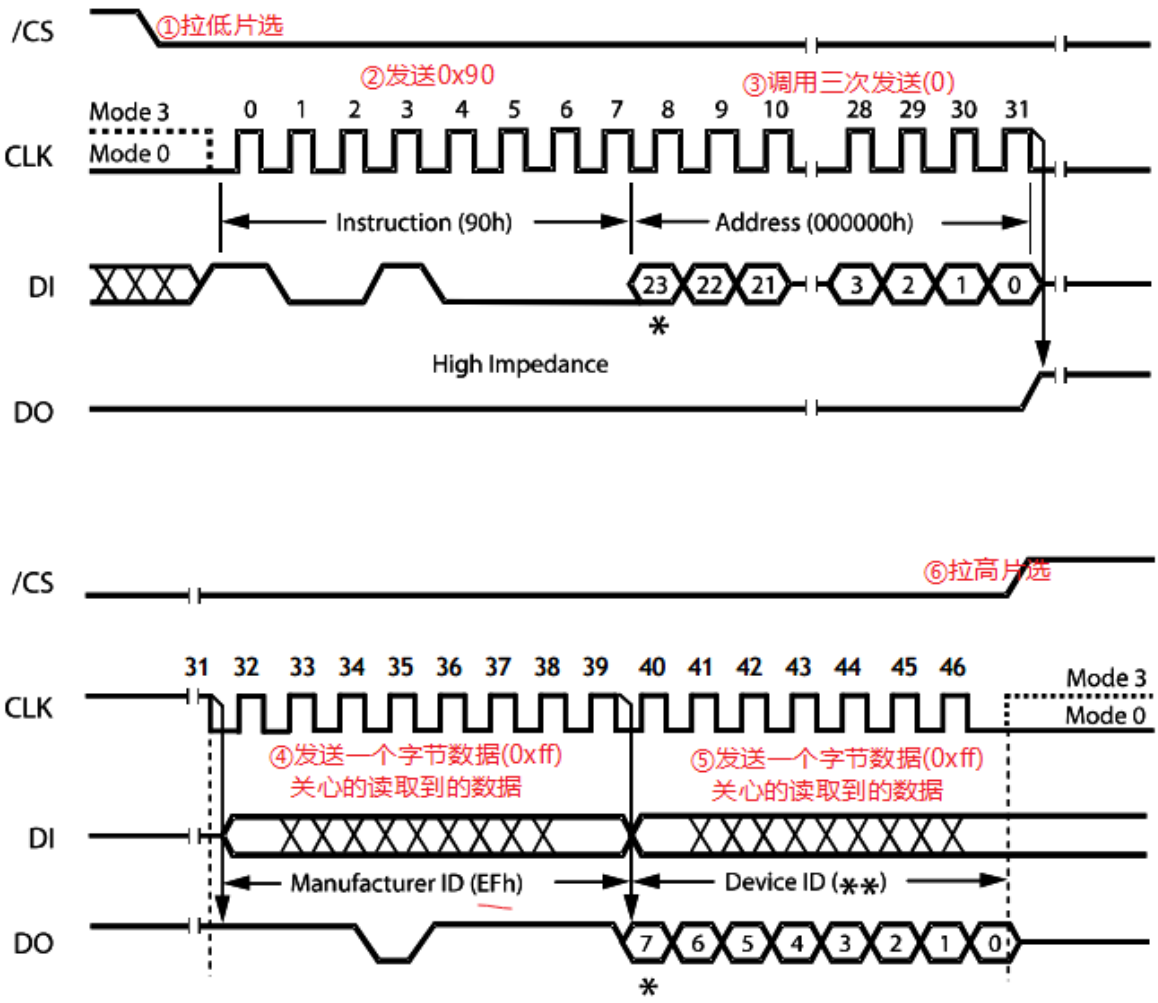
   页写最多写 256 字节的数据(不会自动换页)



4. 扇区擦除

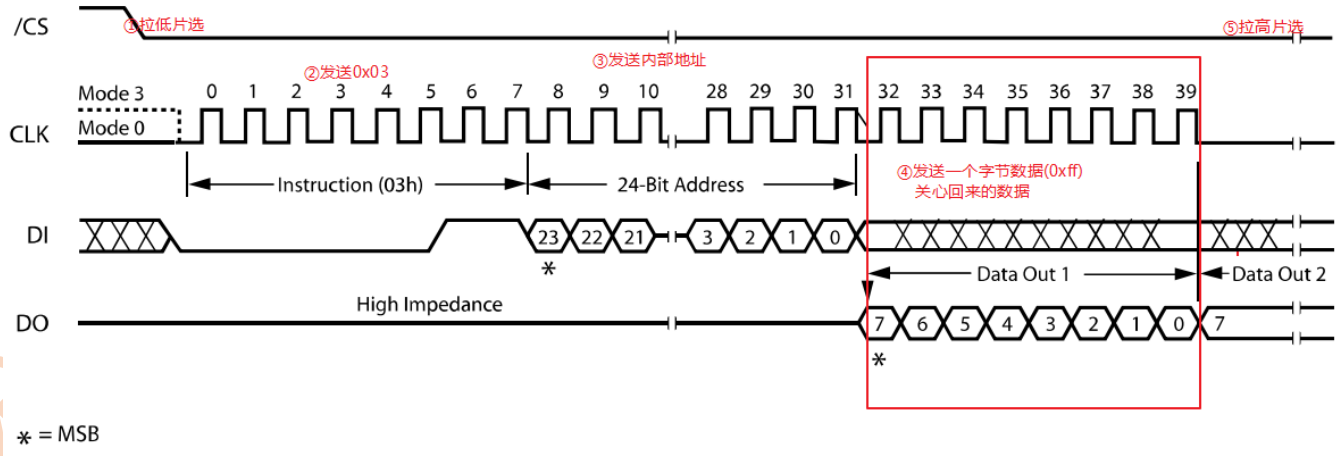


5. 读 ID



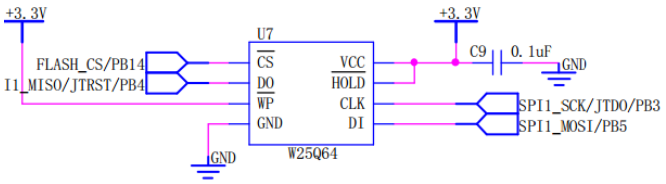
读到 0xEF16 表示 SPI 驱动成功 芯片正品

6. 读数据



* = MSB

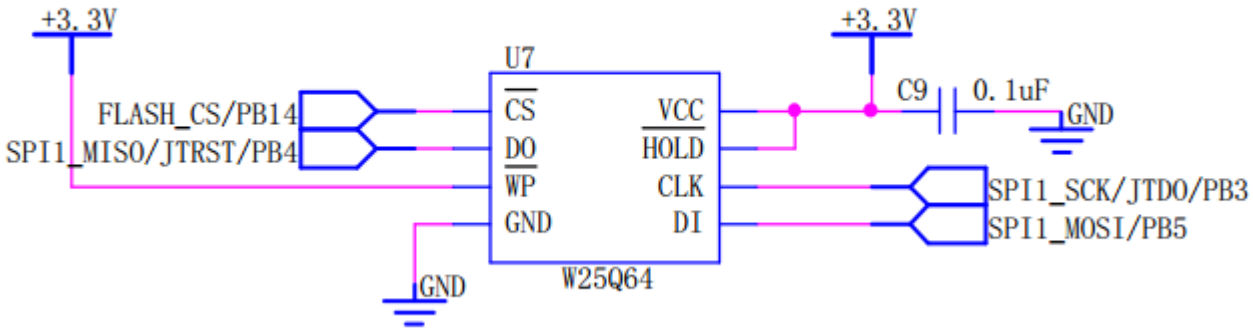
12.4 IO 模拟 SPI 实验

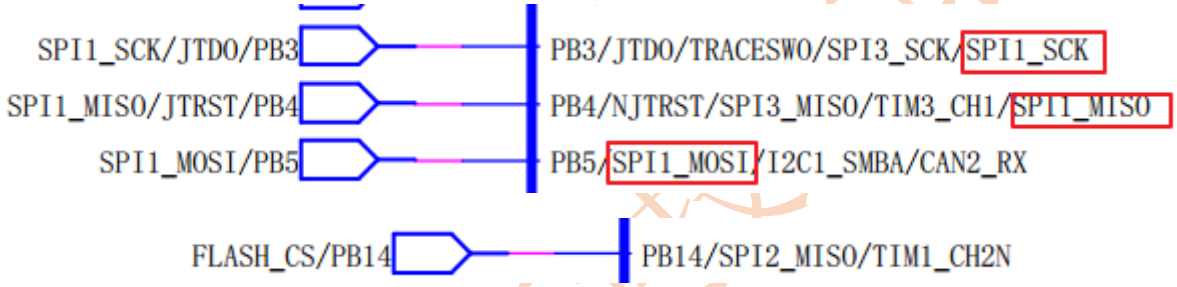


PB14 – 推挽输出 PB3 、PB5：推挽输出 PB4：浮空输入

12.5 STM32 的 SPI 控制器 SPI 实验

12.5.1 硬件设计





PB3、PB4、PB5 配置复用为 SPI1

Table 9. Alternate function mapping (continued)

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8	AF9	AF10
		SYS	TIM1/2	TIM3/4/5	TIM8/9/10/11	I2C1/2/3	SPI1/SPI2/I2S2/I2S2ext	SPI3/I2Sext/I2S3	USART1/2/3/I2S3ext	UART4/5/USART6	CAN1/2 TIM12/13/14	OTG_FS/OTG_HS
Port B	PB0	-	TIM1_CH2N	TIM3_CH3	TIM8_CH2N	-	-	-	-	-	-	OTG_HS_ULPI_D1
	PB1	-	TIM1_CH3N	TIM3_CH4	TIM8_CH3N		-	-	-	-	-	OTG_HS_ULPI_D2
	PB2	-	-	-	-	-	-	-	-	-	-	-
	PB3	JTDO/TRACESWO	TIM2_CH2	-	-	-	SPI1_SCK	SPI3_SCK I2S3_CK	-	-	-	-
	PB4	NJTRST	-	TIM3_CH1		-	SPI1_MISO	SPI3_MISO	I2S3ext_SD	-	-	-
	PB5	-	-	TIM3_CH2		I2C1_SMBA	SPI1_MOSI	SPI3_MOSI I2S3_SD		-	CAN2_RX	OTG_HS_ULPI_D7
	PB6	-	-	TIM4_CH1		I2C1_SCL	-	-	USART1_TX	-	CAN2_TX	-
	PB7	-	-	TIM4_CH2		I2C1_SDA	-	-	USART1_RX	-	-	-
	PB8	-	-	TIM4_CH3	TIM10_CH1	I2C1_SCL	-	-	-	-	CAN1_RX	-
	PB9	-	-	TIM4_CH4	TIM11_CH1	I2C1_SDA	SPI2_NSS I2S2_WS	-	-	-	CAN1_TX	-

PB14 配置为推挽输出

12.5.2 软件设计

初始化 SPI1 的函数

1. GPIO 口初始化

   PB3、PB4、PB5 配置复用为 SPI1

PB14 配置为推挽输出

2. 打开 SPI1 时钟
3. 配置 CR1 寄存器
4. 使能 SPI1

发送一个字节数据得到一个字节数据函数

1. 判断发送缓冲区为空
2. 把数据写到 DR
3. 判断接收缓冲区非空
4. 读 DR 寄存器

今天作业：1、编写 SPI 通信实验。

2、每条代码，对应芯片手册的寄存器各个位（具体功能）。