1. 总述

W25X16、W25X32 和 W25X64 系列 FLASH 存储器可以为用户提供存储解决方案，具有“PCB 板占用空间少”、“引脚数量少”、“功耗低”等特点。与普通串行 FLASH 相比，使用更灵活，性能更出色。它非常适合做代码下载应用，例如存储声音，文本和数据。工作电压在 2.7V~3.6V 之间，正常工作状态下电流消耗 0.5 毫安，掉电状态下电流消耗 1 微安。所有的封装都是“节省空间”型的。

W25X16、W25X32 和 W25X64 分别有 8192、16384 和 32768 可编程页，每页 256 字节。用“页编程指令”每次就可以编程 256 个字节。用“扇区（sector）擦除指令”每次可以擦除 16 页，用“块(block)擦除指令”每次可以擦除 256 页，用“整片擦除指令”即可以擦除整个芯片。W25X16、W25X32 和 W25X64 分别有 512、1024 和 2048 个可擦除“扇区”或 32、64 和 128 个可擦除“块”。

W25X16、W25X32 和 W25X64 支持标准的 SPI 接口，传输速率最大 75MHz。四线制：

①：串行时钟引脚 CLK；

②：芯片选择引脚 CS；

③：串行数据输出引脚 DO；

④：串行数据输入输出引脚 DIO。

（注意：第④引脚“串行数据输入输出引脚 DIO”的解释：在普通情况下，这根引脚是“串行输入引脚（DI），当使用了“快读双输出指令（Fast Read Dual Output instruction）”时，这根引脚就变成了 DO 引脚，这种情况下，芯片就有了两个 DO 引脚了，所以叫做双输出，这时，如果与芯片通信的速率相当于翻了一倍，所以传输速度更快。）

另外，芯片还具有保持引脚（HOLD）、写保护引脚（WP）、可编程写保护位（位于状态寄存器 bit1）、顶部和底部块的控制等特征，使得控制芯片更具灵活性。

芯片支持 JEDEC 工业标准。

2. 特征

- 串行 FLASH 系列存储器系列

W25X16：16M 比特（bit）/2M 字节（byte）
W25X32：32M 比特（bit）/4M 字节（byte）
W25X64：64M 比特（bit）/8M 字节（byte）
每页 256 字节
统一的 4K 字节扇区（Sectors）和 64K 字节块区(Blocks)

- 单输出和双输出的 SPI 接口

时钟引脚（Clock），芯片选择引脚（CS），数据输入输出引脚（DIO），数据输出引脚（DO）
HOLD 引脚功能也可以灵活的控制 SPI

- 数据传输速率最大 150M 比特每秒

时钟运行频率 75M 赫兹
快读双输出指令
读指令地址自动增加

- 灵活的 4KB 扇区结构

扇区删除（4K 字节）
块区扇区（64K 字节）
页编程（256 字节）< 2 毫秒
最大 10 万次擦写周期
20 年存储

- 低能耗，宽温度范围

单电源供电：（2.7V~3.6V）
正常工作状态下：0.5 毫安，掉电状态下：1 微安
工作温度范围：-40° C ~ +85° C

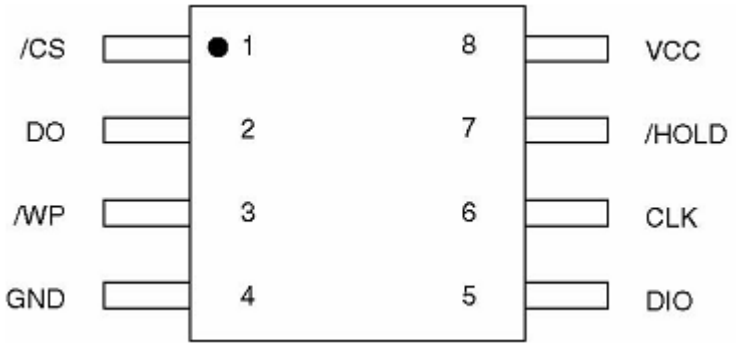
- 软件写保护和硬件写保护

部分或全部写保护
WP 引脚使能和关闭写保护
顶部和底部块保护

- 小空间封装

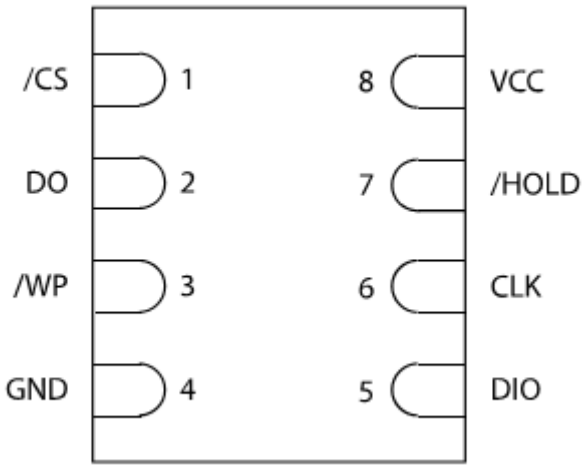
8 引脚(pin)SOIC 208mil 封装（W25X16,X32）
8 引脚(pin)PDIP 300mil 封装（W25X16,X32,X64）
16 引脚(pin)SOIC 300mil 封装（W25X16,X32,X64）
8 引脚(pad) WSON 6×5mm 封装（W25X16）
8 引脚(pad) WSON 8×6mm 封装（W25X32,X64）

3. SOIC 208mil 封装引脚配置



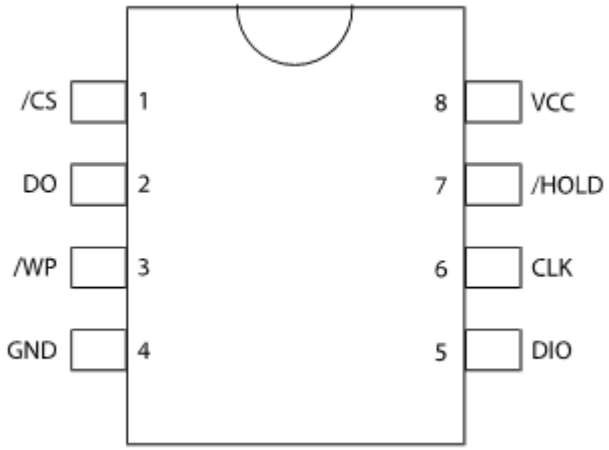
W25X16,W25X32

4. WSON 6×5MM 封装引脚配置



W25X16

5. PDIP 300MIL 封装引脚配置

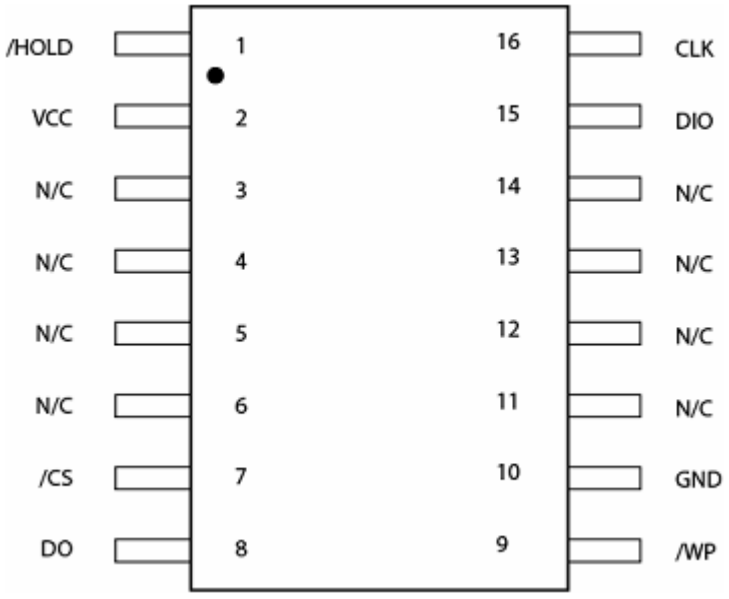


W25X16,W25X32,W25X64

6.引脚描述：SOIC 208mil 封装、PDIP 300mil 封装、WSON 6×5MM

引脚号（PAD.NO）	引脚名称（PAD.NAME）	输入输出类型（I/O）	功能（FUNCTION）
1	/CS	I	芯片选择
2	DO	O	数据输出
3	/WP	I	写保护
4	GND		地
5	DIO	I/O	数据输入/输出
6	CLK	I	串行时钟
7	/HOLD	I	保持
8	VCC		电源

7.SOIC 300mil 封装引脚配置



W25X16,W25X32,W25X64

8.SOIC 300mil 封装引脚描述

引脚号（PAD.NO）	引脚名称（PAD.NAME）	输入输出类型（I/O）	功能（FUNCTION）
1	/HOLD	I	保持
2	VCC		电源

3	N/C		空脚
4	N/C		空脚
5	N/C		空脚
6	N/C		空脚
7	/CS	I	芯片选择
8	DO	O	数据输出
9	/WP	I	写保护
10	GND		地
11	N/C		空脚
12	N/C		空脚
13	N/C		空脚
14	N/C		空脚
15	DIO	I/O	数据输出
16	CLK	I	串行时钟

8.1 封装类型

看上面的图就知道了。

8.2 芯片选择引脚（/CS）

引脚 CS 使能和禁能芯片。当 CS 为高电平时，芯片被禁能，DO 引脚高阻抗，在此时，如果器件内部没有擦除、编程或处于状态寄存器周期进程，器件功耗将处于待机水平。当 CS 为低电平时，使能芯片，此时功耗增加到激活水平，此时，就可以进行芯片的读写操作了。上电之后，执行一条新指令之前必须使 CS 引脚先有一个下降沿。CS 引脚可以根据需要加上拉电阻。

8.3 数据输出引脚（DO）

下降沿输出数据。

8.4 写保护引脚（/WP）

写保护引脚可以被用来保护状态寄存器不被意外改写。

8.5 保持引脚（/HOLD）

当 CS 为低电平，且 HOLD 为低电平时，DO 引脚将处于高阻抗状态，而且也会忽略 DIO 和 CLK 引脚上的信号。把 HOLD 引脚拉高，器件恢复正常工作。当芯片与多个其它芯片共享微控制器上的同一个 SPI 接口时，此引脚就会显得很有作用。

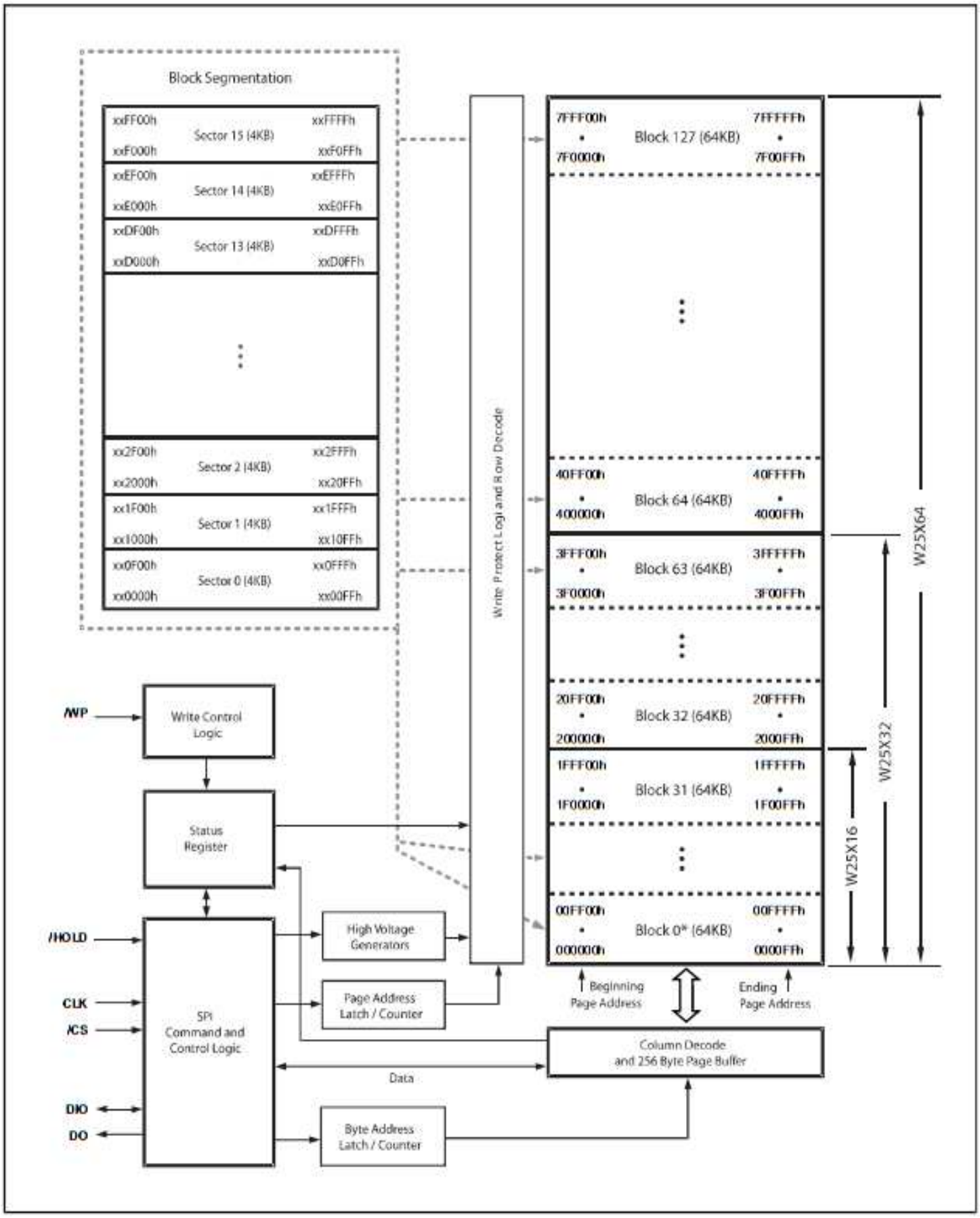
8.6 串行时钟引脚（CLK）

SPI 时钟引脚，为输入输出提供时序。

8.7 串行数据输入/输出引脚（DIO）

数据，地址和命令从 DIO 引脚送到芯片内部，在 CLK 引脚的上升沿捕获。当使用“快读双输出指令（Fast Read Dual Output）”时，此引脚作为输出引脚使用。

9. 模块图


Block Segmentation
xxFF00h
xxFFFFh
Sector 15 (4KB)
xxF000h
xxF0FFh
xxEF00h
xxEFFFh
Sector 14 (4KB)
xxE000h
xxE0FFh
xxDF00h
xxDFFFh
Sector 13 (4KB)
xxD000h
xxD0FFh
xx2F00h
xx2FFFh
Sector 2 (4KB)
xx2000h
xx20FFh
xx1F00h
xx1FFFh
Sector 1 (4KB)
xx1000h
xx10FFh
xx0F00h
xx0FFFh
Sector 0 (4KB)
xx0000h
xx00FFh
Write Protect Logic and Row Decode
7FFF00h
7FFFFFh
Block 127 (64KB)
7F0000h
7F00FFh
40FF00h
40FFFFh
Block 64 (64KB)
400000h
4000FFh
3FFF00h
3FFFFFh
Block 63 (64KB)
3F0000h
3F00FFh
20FF00h
20FFFFh
Block 32 (64KB)
200000h
2000FFh
1FFF00h
1FFFFFh
Block 31 (64KB)
1F0000h
1F00FFh
00FF00h
00FFFFh
Block 0* (64KB)
000000h
0000FFh
W25X64
W25X32
W25X16
Beginning Page Address
Ending Page Address
/WP
Write Control Logic
Status Register
/HOLD
CLK
/CS
SPI Command and Control Logic
DIO
DO
High Voltage Generators
Page Address Latch / Counter
Byte Address Latch / Counter
Column Decode and 256 Byte Page Buffer
Data

10. 功能描述

10.1 SPI 运行方式

10.1.1 SPI 模式

W25X16\X32\X64 通过四线制 SPI 总线方式访问：CLK\CS\DIO\DO。两种 SPI 通信方式都支持：模式 0（0，0）和模式 3（1，1）。模式 0 和模式 3 的主要区别是当 SPI 主机的 SPI 口处于空闲或者是没有数据传输的时候 CLK 的电平是高电平还是低电平。对于模式 0，CLK 处于低电平；对于模式 3，CLK 处于高电平。不过，在两种模式下芯片都是在 CLK 的上升沿采集输入数据，下降沿输出数据。

10.1.2 双输出 SPI 方式

W25X16\X32X64 支持 SPI 双输出方式，需要使用“快读双输出指令（Fast Read Dual Output：0x3B）”，这时，传输速率相当于两倍于标准的 SPI 传输速率。这个命令非常适合在需要一上电就快速下载代码到内存中的情况下（code-shadowing）或者是需要缓存代码段到内存中运行的情况下（cache code-segments to RAM for execution）。在使用“快读双输出指令”后，DIO 引脚变为输出引脚。

10.1.3 保持功能（/HOLD）

芯片处于使能状态下（CS=0）时，把 HOLD 引脚拉低可以使芯片“暂停”工作。适用于当芯片和其它器件共享 SPI 主机上的 SPI 口的时候。例如，当 SPI 主机接收到一个更高优先级的中断抢占了 SPI 主机的 SPI 口，而这时芯片的“页缓存区（page buffer）”还有一部分没有写完，这时，保持功能就可以使得芯片当中的“页缓存区”保存好数据，等到那个 SPI 从机释放 SPI 口时，将继续完成刚才没有写完的工作。

使用保持功能，CS 引脚必须为低电平。在 HOLD 的下降沿以后，如果时钟引脚 CLK 处于低电平，保持功能开始；如果 CLK 引脚为高电平，保持功能在 CLK 的下一个下降沿后开始。在 HOLD 的上升沿以后，如果 CLK 引脚为低电平，保持功能结束；如果 CLK 为高电平，在 CLK 引脚的下一个下降沿，保持功能结束。

在保持功能起作用期间，DO 引脚处于高阻抗状态，DIO 引脚和 DO 引脚上的信号忽略。而且，在此期间，CS 引脚也必须保持低电平。如果在此期间 CS 拉高，芯片内部逻辑将会被重置。

10.2 写保护功能（/WR）

这个功能主要应用在芯片处于存在干扰噪声（这个噪声是指电磁干扰）等恶劣环境下工作的时候。

10.2.1 写保护特征

- 当 VCC 低于阀值电压器件重置
- 上电之后禁能延迟写入
- 写使能、写禁能指令
- 编程或擦除之后自动禁止写入
- 用“状态寄存器”实现软件写保护
- 用“状态寄存器”和“/WP”引脚实现写保护
- 用“掉电指令”实现写保护

在“上电状态”或“掉电状态”下，如果 VCC 引脚的电平低于阀值电压(VWI)，芯片将处于重置状态。此时，所有对芯片的操作都被禁止，指令也无法识别。当上电之后，如果 VCC 电平超过了阀值电压(VWI)，所有的和编程或者是擦除相关的指令都会被禁止一段时间（tpuw）。这些指令包括：写使能（Write Enable）、页编程（Page Program）、扇区擦除（Sector Erase）、块区擦除（Block Erase）、芯片擦除（Chip Erase）和写状态寄存器指令（Write Status Register）。注意，在上电过程中，CS 引脚电平必须和 VCC 引脚电平保持一致，直到 VCC

引脚电平达到最小值（VCCmin）而且（TVSL）时间延迟达到。CS 可以接一个上拉电阻完成这个工作。

芯片上电之后自动进入写保护状态，因为此时，状态寄存器当中的写保护状态位（WEL）为 0。在执行“页编程指令”“扇区擦除指令”“芯片擦除指令”“写状态寄存器指令”之前，必须先执行“写使能”指令。在执行完编程、擦除、写指令之后，状态寄存器中的写保护位 WEL 自动变为 0，使写禁能。

软件控制写保护功能可以“写状态寄存器指令”置位“状态寄存器保护位（SPR）”“块区保护位（TB,BP2,BP1,BP0）”，这些状态寄存器位允许芯片部分或者是整个芯片都为只读。结合/WP 引脚，就可以实现硬件控制使能或禁能写保护。

另外，利用“掉电指令”也可以实现一定意义上的写保护，此时，除了“释放掉电指令”，其它的所有指令无效。

11. 控制和状态寄存器

S7	S6	S5	S4	S3	S2	S1	S0
SRP	(Reservd)	TB	BP2	BP1	BP0	WEL	BUSY

通过“读状态寄存器指令”读出的状态数据可以知道芯片存储器阵列是否可写或不可写，或是否处于写保护状态。通过“写状态寄存器指令”可以配置芯片写保护特征。

11.1 状态寄存器

11.1.1 忙位（BUSY）

BUSY 位是个只读位，位于状态寄存器中的 S0。当器件在执行“页编程”“扇区擦除”“块区擦除”“芯片擦除”“写状态寄存器”指令时，该位自动置 1。这时，除了“读状态寄存器”指令，其它指令都忽略。当编程、擦除和写状态寄存器指令执行完毕之后，该位自动变为 0，表示芯片可以接收其它指令了。

11.1.2 写保护位（WEL）

WEL 位是个只读位，位于状态寄存器中的 S1。执行完“写使能”指令后，该位置 1。当芯片处于“写保护状态”下，该位为 0。在下面两种情况下，会进入“写保护状态”：

- 掉电后
- 执行以下指令后：写禁能、页编程、扇区擦除、块区擦除、芯片擦除和写状态寄存器

11.1.3 块区保护位（BP2, BP1, BP0）

BP2\BP1\BP0 位是可读可写位，分别位于状态寄存器的 S4\S3\S2。可以用“写状态寄存器”命令置位这些块区保护位。在默认状态下，这些位都为 0，即块区处于未保护状态下。可以设置块区没有保护、部分保护或者是全部处于保护状态下。当 SPR 位为 1 或/WP 引脚为低的时候，这些位不可以被更改。

11.1.4 底部和顶部块区保护位（TB）

TB 位是可读可写位，位于状态寄存器的 S5。该位默认为 0，表明顶部和底部块区处于未被保护状态下。可以用“写状态寄存器”命令置位该位。当 SPR 位为 1 或/WP 引脚为低的时候，这些位不可以被更改。

11.1.5 保留位

状态寄存器的 S6 为保留位，读出状态寄存器值时，该位为 0。建议读状态寄存器值用于测试时将该位屏蔽。

11.1.6 状态寄存器保护位（SRP）

SRP 位是可读可写位，位于状态寄存器的 S7。该位结合/WP 引脚可以实现禁能写状态寄存器功能。该位默认值为 0。当 SRP 位=0 时，/WP 引脚不能控制状态寄存器的“写禁能”。当 SRP 位=1，/WP 引脚=0 时，“写状态寄存器”命令失效。当 SRP 位=1，/WP 引脚=1 时，

可以执行“写状态寄存器”命令。

11.1.7 状态寄存器存储保护模块

STATUS REGISTER(1)				W25X64 (64M-BIT) MEMORY PROTECTION			
TB	BP2	BP1	BP0	BLOCK(S)	ADDRESSES	DENSITY (KB)	PORTION
x	0	0	0	NONE	NONE	NONE	NONE
0	0	0	1	126 and 127	7E0000h - 7FFFFFh	1M -bit	Upper 1/64
0	0	1	0	124 and 127	7C0000h - 7FFFFFh	2M-bit	Upper 1/32
0	0	1	1	120 thru 127	780000h - 7FFFFFh	4M-bit	Upper 1/16
0	1	0	0	112 thru 127	700000h - 7FFFFFh	8M-bit	Upper 1/8
0	1	0	1	96 thru 127	600000h - 7FFFFFh	16M-bit	Upper 1/4
0	1	1	0	64 thru 127	400000h - 7FFFFFh	32M-bit	Upper 1/2
1	0	0	1	0 and 1	000000h - 01FFFFh	1M-bit	Lower 1/64
1	0	1	0	0 thru 3	000000h - 03FFFFh	2M-bit	Lower 1/32
1	0	1	1	0 thru 7	000000h - 07FFFFh	4M-bit	Lower 1/16
1	1	0	0	0 thru 15	000000h – 0FFFFFh	8M-bit	Lower 1/8
1	1	0	1	0 thru 31	000000h – 1FFFFFh	16M-bit	Lower 1/4
1	1	1	0	0 thru 63	000000h – 3FFFFFh	16M-bit	Lower 1/2
x	1	1	1	0 thru 127	000000h - 7FFFFFh	64M-bit	ALL

STATUS REGISTER(1)				W25X32 (32M-BIT) MEMORY PROTECTION			
TB	BP2	BP1	BP0	BLOCK(S)	ADDRESSES	DENSITY (KB)	PORTION
x	0	0	0	NONE	NONE	NONE	NONE
0	0	0	1	63	3F0000h - 3FFFFFh	512K-bit	Upper 1/64
0	0	1	0	62 and 63	3E0000h - 3FFFFFh	1M-bit	Upper 1/32
0	0	1	1	60 thru 63	3C0000h - 3FFFFFh	2M-bit	Upper 1/16
0	1	0	0	56 thru 63	380000h - 3FFFFFh	4M-bit	Upper 1/8
0	1	0	1	48 thru 63	300000h - 3FFFFFh	8M-bit	Upper 1/4
0	1	1	0	32 thru 63	200000h - 3FFFFFh	16M-bit	Upper 1/2
1	0	0	1	0	000000h - 00FFFFh	512K-bit	Lower 1/64
1	0	1	0	0 and 1	000000h - 01FFFFh	1M-bit	Lower 1/32
1	0	1	1	0 thru 3	000000h - 03FFFFh	2M-bit	Lower 1/16
1	1	0	0	0 thru 7	000000h - 07FFFFh	4M-bit	Lower 1/8
1	1	0	1	0 thru 15	000000h – 0FFFFFh	8M-bit	Lower 1/4
1	1	1	0	0 thru 31	000000h – 1FFFFFh	16M-bit	Lower 1/2
x	1	1	1	0 thru 63	000000h - 3FFFFFh	32M-bit	ALL

STATUS REGISTER(1)				W25X16 (16M-BIT) MEMORY PROTECTION			
TB	BP2	BP1	BP0	BLOCK(S)	ADDRESSES	DENSITY (KB)	PORTION
x	0	0	0	NONE	NONE	NONE	NONE
0	0	0	1	31	1F0000h - 1FFFFFh	512K-bit	Upper 1/32
0	0	1	0	30 and 31	1E0000h - 1FFFFFh	1M-bit	Upper 1/16
0	0	1	1	28 thru 31	1C0000h - 1FFFFFh	2M-bit	Upper 1/8
0	1	0	0	24 thru 31	180000h - 1FFFFFh	4M-bit	Upper 1/4
0	1	0	1	16 thru 31	100000h - 1FFFFFh	8M-bit	Upper 1/2
1	0	0	1	0	000000h - 00FFFFh	512K-bit	Lower 1/32
1	0	1	0	0 and 1	000000h - 01FFFFh	1M-bit	Lower 1/16
1	0	1	1	0 thru 3	000000h - 03FFFFh	2M-bit	Lower 1/8
1	1	0	0	0 thru 7	000000h - 07FFFFh	4M-bit	Lower 1/4
1	1	0	1	0 thru 15	000000h - 0FFFFFh	8M-bit	Lower 1/2
x	1	1	x	0 thru 31	000000h - 1FFFFFh	16M-bit	ALL

11.2 命令

W25X16\X32\X64 包括 15 个基本的命令，这 15 个基本的指令可以通过 SPI 总线完全控制芯片。指令在/CS 引脚的下降沿开始传送，DIO 引脚上数据的第一个字节就是指令代码。

在时钟引脚的上升沿采集 DIO 数据，高位在前。

指令的长度从一个字节到多个字节，有时还会跟随地址字节、数据字节、伪字节（dummy bytes），有时候还会是它们的组合。在/CS 引脚的上升沿完成指令传输。所有的读指令都可以在任意的时钟位完成。而所有的写、编程和擦除指令在一个字节的边界后才能完成，否则，指令将不起作用。这个特征可以保护芯片被意外写入。当芯片正在被编程、擦除或写状态寄存器的时候，除了“读状态寄存器”指令，其它所有的指令都会被忽略知道擦写周期结束。

11.2.1 器件标识

MANUFACTURER ID	**(M7-M0)**	
Winbond Serial Flash	EFH	
Device ID	**(ID7-ID0)**	**(ID15-ID0)**
Instruction	**ABh, 90h**	**9Fh**
W25X16	14h	3015h
W25X32	15h	3016h
W25X64	16h	3017h

11.2.2 命令表

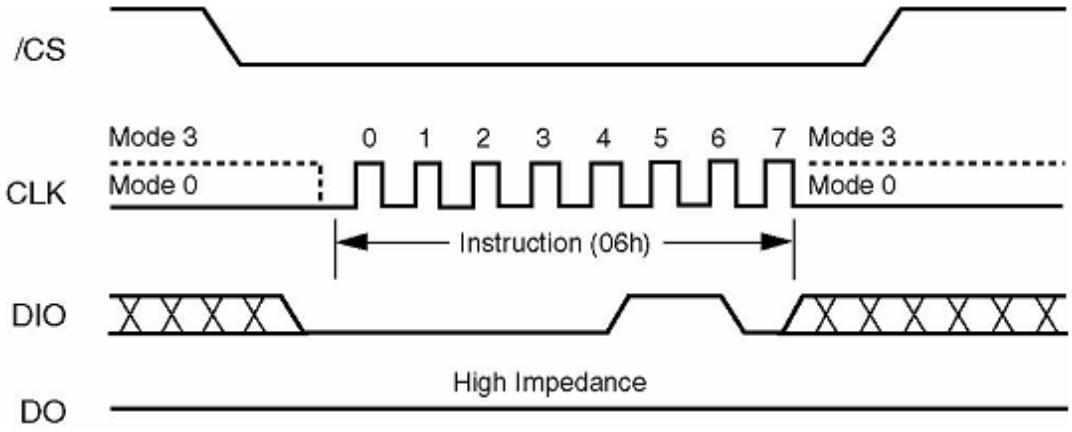
指令名称	字节 1	字节 2	字节 3	字节 4	字节 5	字节 6	下一个字节
写使能	06h						
写禁能	04h						
读状态寄存器	05h	(S7~S0)					
写状态寄存器	01h	S7~S0					
读数据	03h	A23~A16	A15~A8	A7~A0	(D7~D0)	下个字节	继续
快读	0Bh	A23~A16	A15~A8	A7~A0	伪字节	D7~D0	下个字节
快读双输出	3Bh	A23~A16	A15~A8	A7~A0	伪字节	I/O=(D6,D4,D2,D0) O=(D7,D5,D3,D1)	每四个时钟一个字节
页编程	02h	A23~A16	A15~A8	A7~A0	(D7~D0)	下个字节	直到 256 个字节
块擦除（64K）	D8h	A23~A16	A15~A8	A7~A0			
扇区擦除(4K)	20h	A23~A16	A15~A8	A7~A0			
芯片擦除	C7h						
掉电	B9h						
释放掉电/器件 ID	ABh	伪字节	伪字节	伪字节	(ID7~ID0)		
制造/器件 ID	90h	伪字节	伪字节	00h	(M7~M0)	(ID7~ID0)	
JEDEC ID	9Fh	(M7~M0)	(ID15~ID8)	(ID7~ID0)			

1.数据传输高位在前，带括号的数据表示数据从 DO 引脚读出。

11.2.3 写使能（06h）（Write Enable）

“写使能”指令将会使“状态寄存器”WEL 位置位。在执行每个“页编程”“扇区擦除”“块区擦除”“芯片擦除”和“写状态寄存器”命令之前，都要先置位 WEL。/CS 引脚先拉

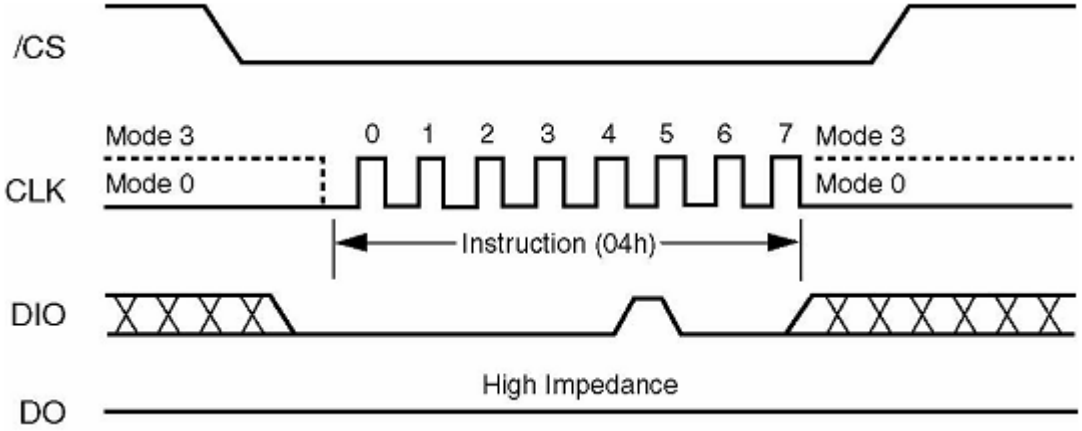
低之后，“写使能”指令代码 06h 从 DI 引脚输入，在 CLK 上升沿采集，然后再拉高/CS 引脚。



“写使能”指令序列图

11.2.4 “写禁能”指令（04h）（Write Disable）

“写禁能”指令将会使 WEL 位变为 0。/CS 引脚拉低之后，把 04h 从 DIO 引脚送到芯片之后，拉高/CS，就完成了这个指令。在执行完“写状态寄存器”“页编程”“扇区擦除”“块区擦除”“芯片擦除”指令之后，WEL 位就会自动变为 0。

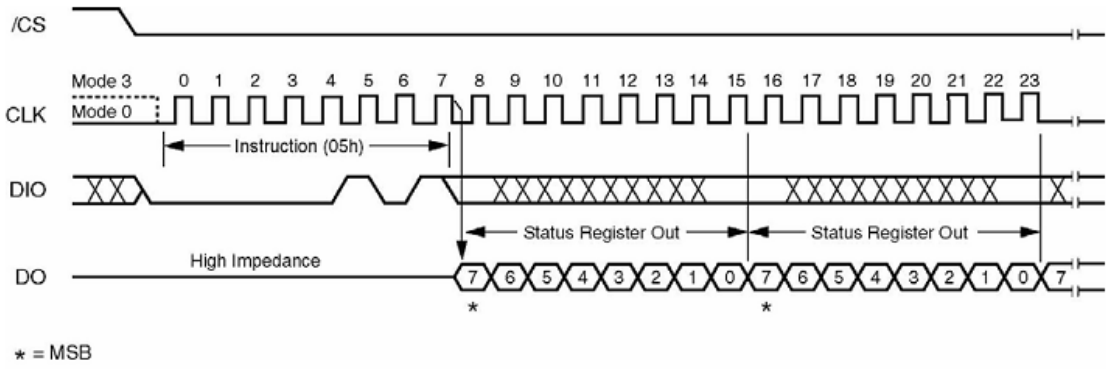


“写禁能”指令序列图

11.2.5 “读状态寄存器”指令（05h）（Read Status Register）

当/CS 拉低之后，开始把 05h 从 DIO 引脚送到芯片，在 CLK 的上升沿数据被芯片采集，当芯片认出采集到的数据时 05h 时，芯片就会把“状态寄存器”的值从 DO 引脚输出，数据在 CLK 的下降沿输出，高位在前。

“读状态寄存器”指令在任何时候都可以用，甚至在编程、擦除和写状态寄存器的过程中也可以用，这样，就可以从状态寄存器的 BUSY 位判断编程、擦除和写状态寄存器周期有没有结束，从而让我们知道芯片是否可以接收下一条指令了。如果/CS 不被拉高，状态寄存器的值将一直从 DO 引脚输出。/CS 拉高之后，读指令结束。



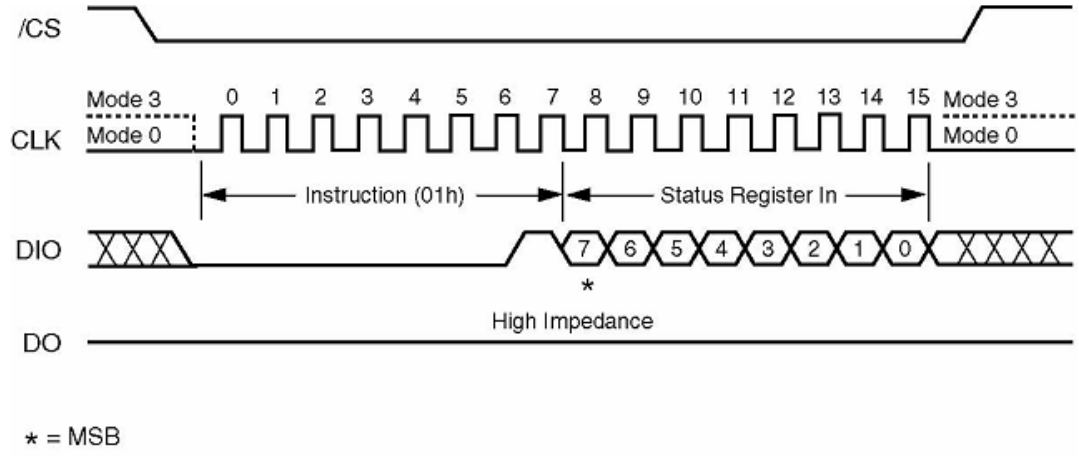
“读状态寄存器”指令时序图

11.2.6 “写状态寄存器”指令（01h）（Write Status Register）

在执行“写状态寄存器”指令之前，需要先执行“写使能”指令。先拉低/CS 引脚，然后把 01h 从 DIO 引脚送到芯片，然后再把你想要设置的状态寄存器值通过 DIO 引脚送到芯片，拉高/CS 引脚，指令结束，如果此时没有把/CS 引脚拉高，或者是拉的晚了，值将不会被写入，指令无效。

只有“状态寄存器”当中的“SRP、TB、BP2、BP1、BP0 位”可以被写入，其它“只读位”值不会变。在该指令执行的过程中，状态寄存器中的 BUSY 位为 1，这时候可以用“读状态寄存器”指令读出状态寄存器的值判断，当指令执行完毕，BUSY 位将自动变为 0，WEL 位也自动变为 0。

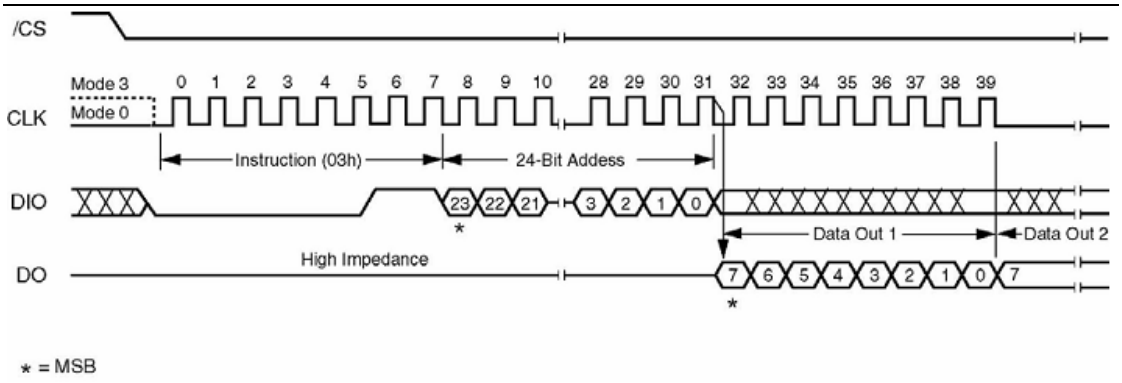
通过对“TB”“BP2”“BP1”“BP0”位写 1，就可以实现将芯片的部分或全部存储区域设置为只读。通过对“SRP 位”写 1，再把/WP 引脚拉低，就可以实现禁止写入状态寄存器的功能。



“写状态寄存器”指令时序图

11.2.7 “读数据”指令（03h）（Read Data）

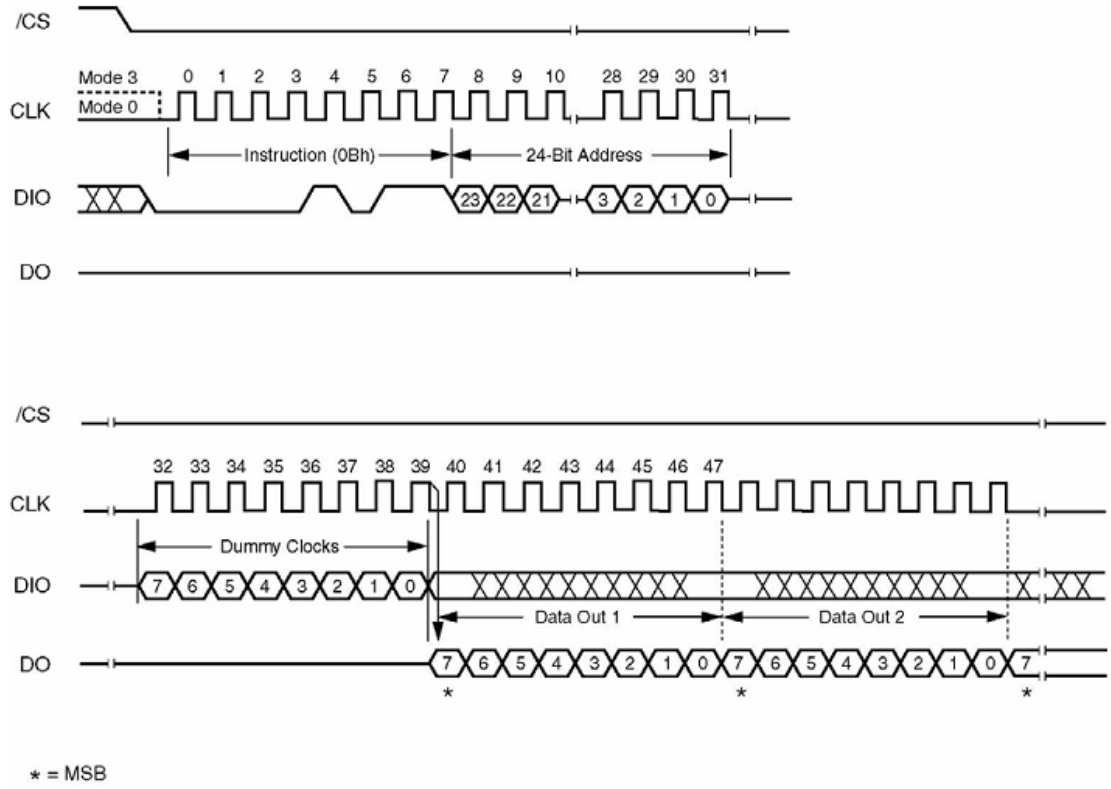
“读数据”指令允许读出一个字节或一个以上的字节被读出。先把/CS 引脚拉低，然后把 03h 通过 DIO 引脚送到芯片，之后再送入 24 位的地址，这些数据在 CLK 的上升沿被芯片采集。芯片接收完 24 位地址之后，就会把相应地址的数据在 CLK 引脚的下降沿从 DO 引脚送出去，高位在前。当读完这个地址的数据之后，地址自动增加，然后通过 DO 引脚把下一个地址的数据送出去，形成一个数据流。也就是说，只要时钟在工作，通过一条读指令，就可以把整个芯片存储区的数据读出来。把/CS 引脚拉高，“读数据”指令结束。当芯片在执行编程、擦除和读状态寄存器指令的周期内，“读数据”指令不起作用。



“读数据”指令时序图

11.2.8 “快读”指令（0Bh）（Fast Read）

“快读”指令和“读数据”指令很相似，不过，“快读”指令可以运行在更高的传输速率下。先把/CS 引脚拉低，然后把 0Bh 通过引脚 DIO 送到芯片，然后把 24 位地址通过 DIO 引脚送到芯片，接着等待 8 个时钟，之后数据将会从 DO 引脚送出去。

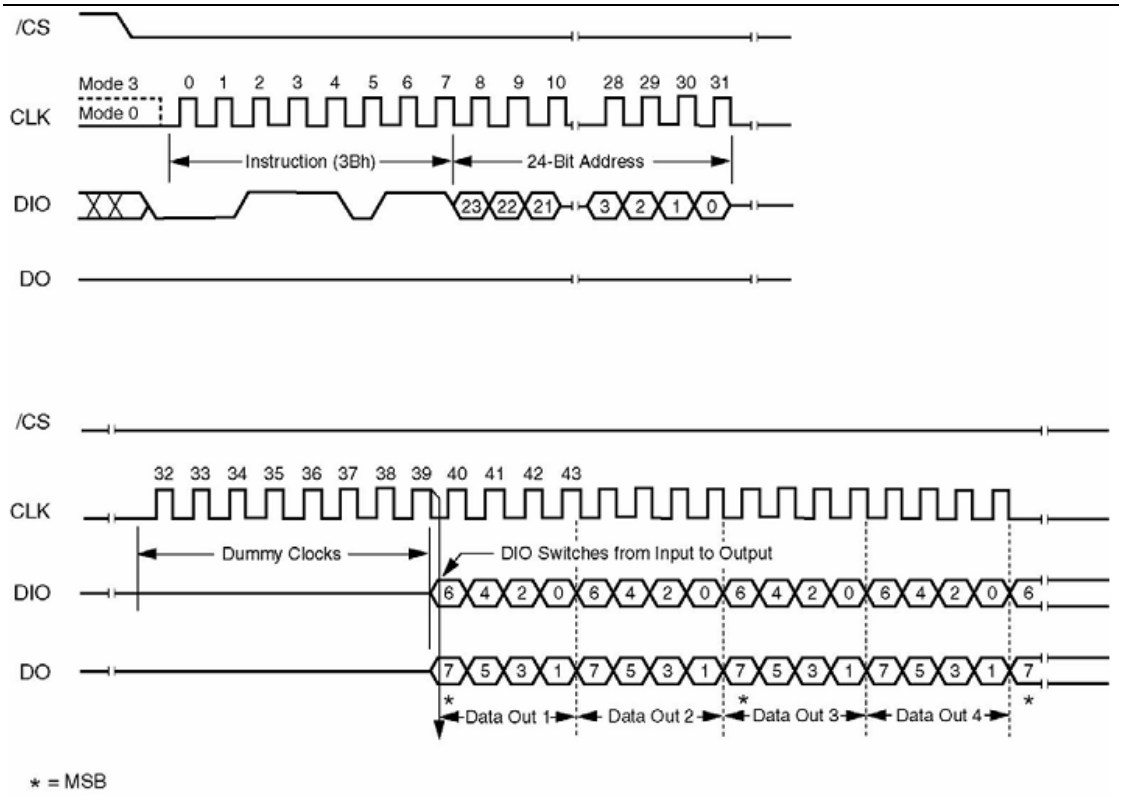


“快读”指令时序图

11.2.9 “快读双输出”指令（3Bh）（Fast Read Dual Output）

“快读双输出”指令和“快读”指令很相似，不过，“快读双输出”指令是从两个引脚上输出数据：DI 和 DIO。这样，传输速率就相当于两倍于标准的 SPI 传输速率了。这个指令特别适合于需要在一上电就把代码从芯片下载到内存中的情况或者缓存代码段到内存中运行的情况。

“快读双输出”指令和“快读”指令的时序差不多。先把/CS 引脚拉低，然后把 3Bh 通过引脚 DIO 送到芯片，然后把 24 位地址通过 DIO 引脚送到芯片，接着等待 8 个时钟，之后数据将会分别从 DO 引脚和 DIO 引脚送出去，DIO 送偶数位，DO 送几数位。



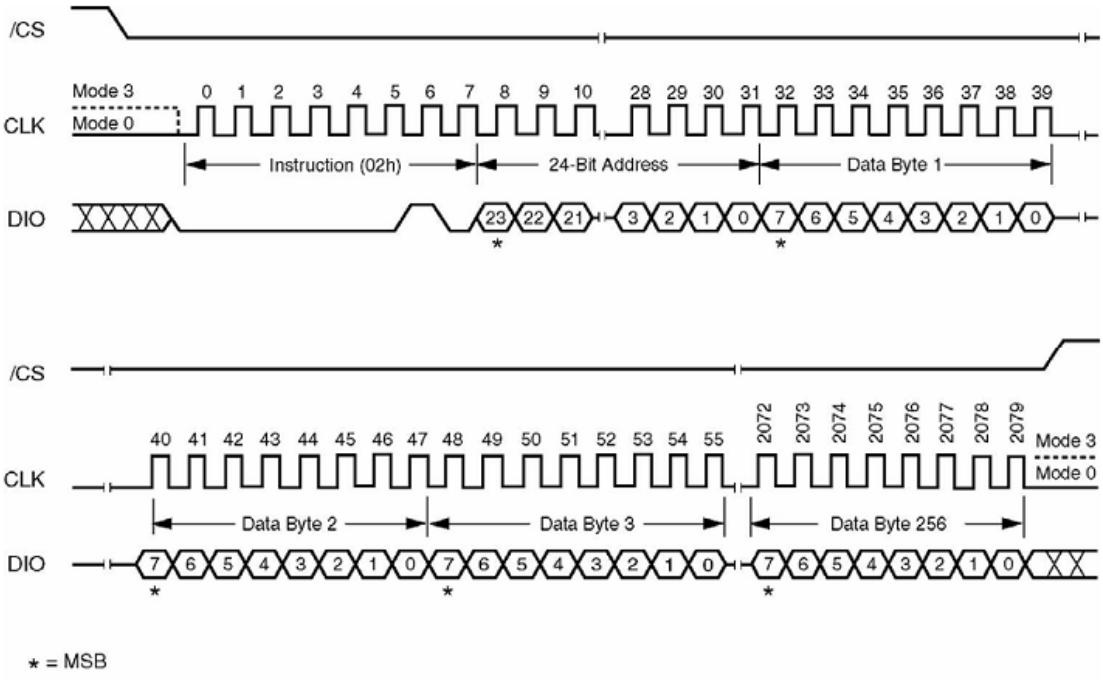
“快读双输出”指令时序图

11.2.10 “页编程”指令（02h）（Page Program）

执行“页编程”指令之前，需要先执行“写使能”指令，而且要求待写入的区域位都为1，也就是需要先把待写入的区域擦除。先把/CS 引脚拉低，然后把代码 02h 通过 DIO 引脚送到芯片，然后再把 24 位地址送到芯片，然后接着送要写的字节到芯片。在写完数据之后，把/CS 引脚拉高。

写完一页（256 个字节）之后，必须把地址改为 0，不然的话，如果时钟还在继续，地址将自动变为页的开始地址。在某些时候，需要写入的字节不足 256 个字节的话，其它写入的字节都是无意义的。如果写入的字节大于了 256 个字节，多余的字节将会加上无用的字节覆盖刚刚写入的的 256 个字节。所以需要保证写入的字节小于等于 256 个字节。

在指令执行过程中，用“读状态寄存器”可以发现 BUSY 位为 1，当指令执行完毕，BUSY 位自动变为 0。如果需要写入的地址处于“写保护”状态，“页编程”指令无效。

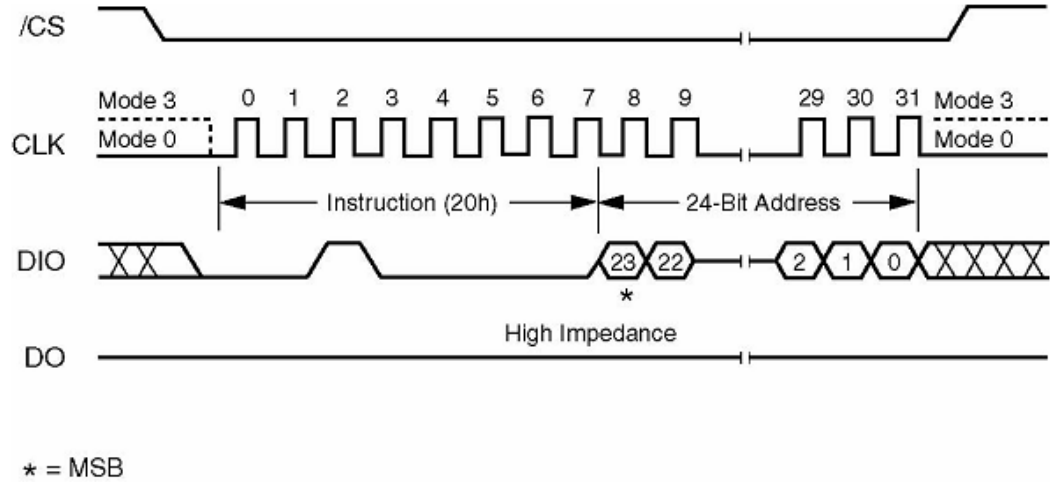


“页编程”指令时序图

11.2.11 “扇区擦除”指令（20h）（Sector Erase）

“扇区擦除”指令将一个扇区（4K 字节）擦除，擦除后扇区位都为 1，扇区字节都为 FFh。在执行“扇区擦除”指令之前，需要先执行“写使能”指令，保证 WEL 位为 1。

先拉低/CS 引脚，然后把指令代码 20h 通过 DIO 引脚送到芯片，然后接着把 24 位扇区地址送到芯片，然后拉高/CS 引脚。如果没有及时把/CS 引脚拉高，指令将不会起作用。在指令执行期间，BUSY 位为 1，可以通过“读状态寄存器”指令观察。当指令执行完毕，BUSY 位变为 0，WEL 位也会变为 0。如果需要擦除的地址处于只读状态，指令将不会起作用。



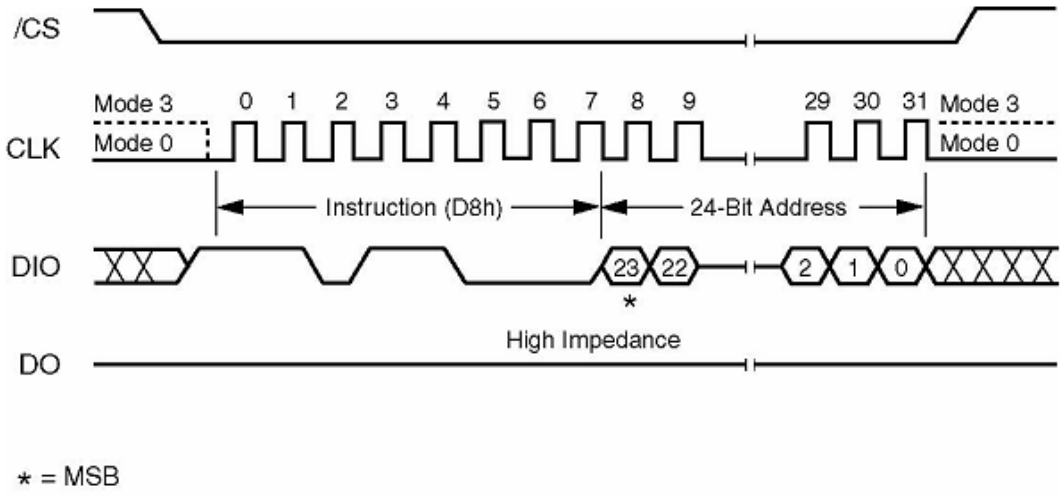
“扇区擦除”指令时序图

11.2.12 块区擦除指令（D8h）（Block Erase）

“块区擦除”指令将一个块区（64K）全部变为 1，即字节都变为 FFh。在“块区擦除”指令执行前需要先执行“写使能”指令。

先拉低/CS 引脚，然后把指令代码 D8h 通过 DIO 引脚送到芯片，然后把 24 位块区地址送到芯片，然后把/CS 引脚拉高。如果没有及时把/CS 引脚拉高，指令将不会起作用。在指令执行周期内，可以执行“读状态寄存器”指令，可以看到 BUSY 位为 1，当“块区擦除指

令”执行完毕，BUSY 位为 0，WEL 位也变为 0。如果需要擦除的地址处于只读状态，指令将不会起作用。

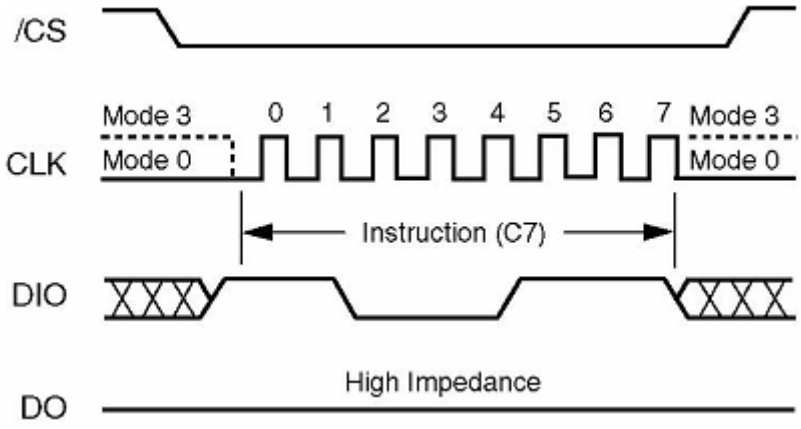


“块区擦除”指令时序图

11.2.13 “芯片擦除”指令（C7h）(Chip Erase)

“芯片擦除”指令将会使整个芯片的存储区位都变为 1，即字节都变位 FFh。在执行“芯片擦除”指令之前需要先执行“写使能”指令。

先把/CS 引脚拉低，然后再把指令代码 C7h 通过 DIO 引脚送到芯片，然后拉高/CS 引脚。如果没有及时拉高/CS 引脚，指令无效。在“芯片擦除”指令执行周期内，可以执行“读状态寄存器”指令访问 BUSY 位，这时 BUSY 位为 1，当“芯片擦除”指令执行完毕，BUSY 变为 0，WEL 位也变为 0。任何一个块区处于保护状态（BP2\BP1\BP0），指令都会失效。

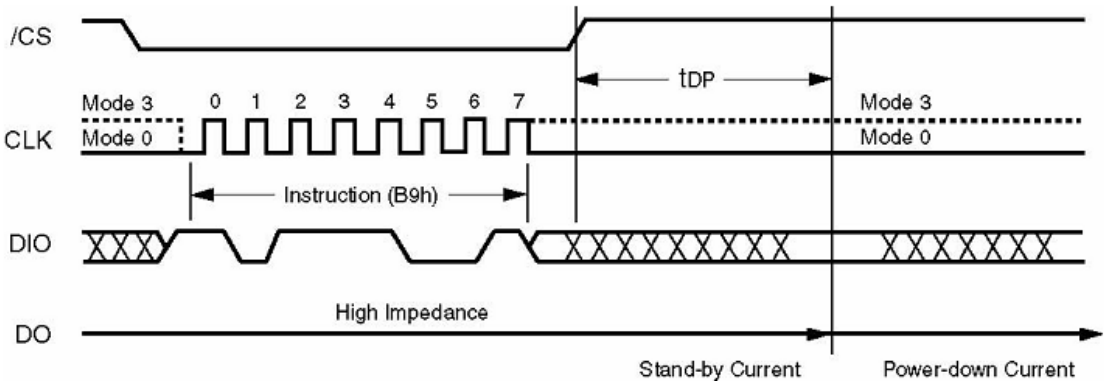


芯片擦除指令时序图

11.2.14 “掉电”指令（B9h）(Power-down)

尽管在待机状态下的电流消耗已经很低了，但“掉电”指令可以使得待机电流消耗更低。这个指令很适合在电池供电的场合。

先把/CS 引脚拉低，然后把指令代码 B9h 通过 DIO 引脚送到芯片。然后把/CS 引脚拉高，指令执行完毕。如果没有及时拉高，指令无效。执行完“掉电”指令之后，除了“释放掉电/器件 ID”指令，其它指令都无效。

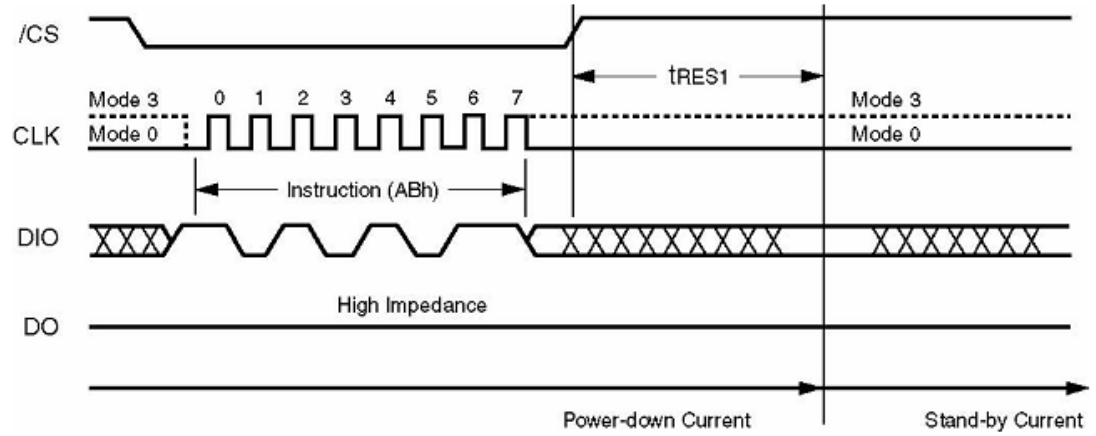


“掉电”指令时序图

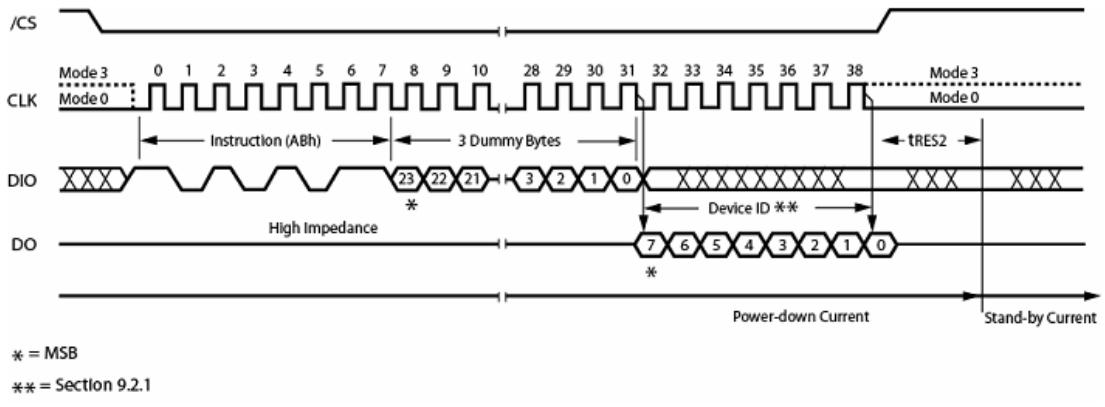
11.2.15 “释放掉电/器件 ID”指令（ABh）（Release Power-down/Device ID）

这个指令有两个作用。一个是“释放掉电”，一个是读出“器件 ID”。

当只需要发挥“释放掉电”用途时，指令时序是：先把/CS 引脚拉低，然后把代码“ABh”通过 DIO 引脚送到芯片，然后拉高/CS 引脚。然后经过 tRES1 时间间隔，芯片恢复正常工作状态。在编程、擦除和写状态寄存器指令执行周期内，执行该指令无效。



“释放掉电”指令时序图



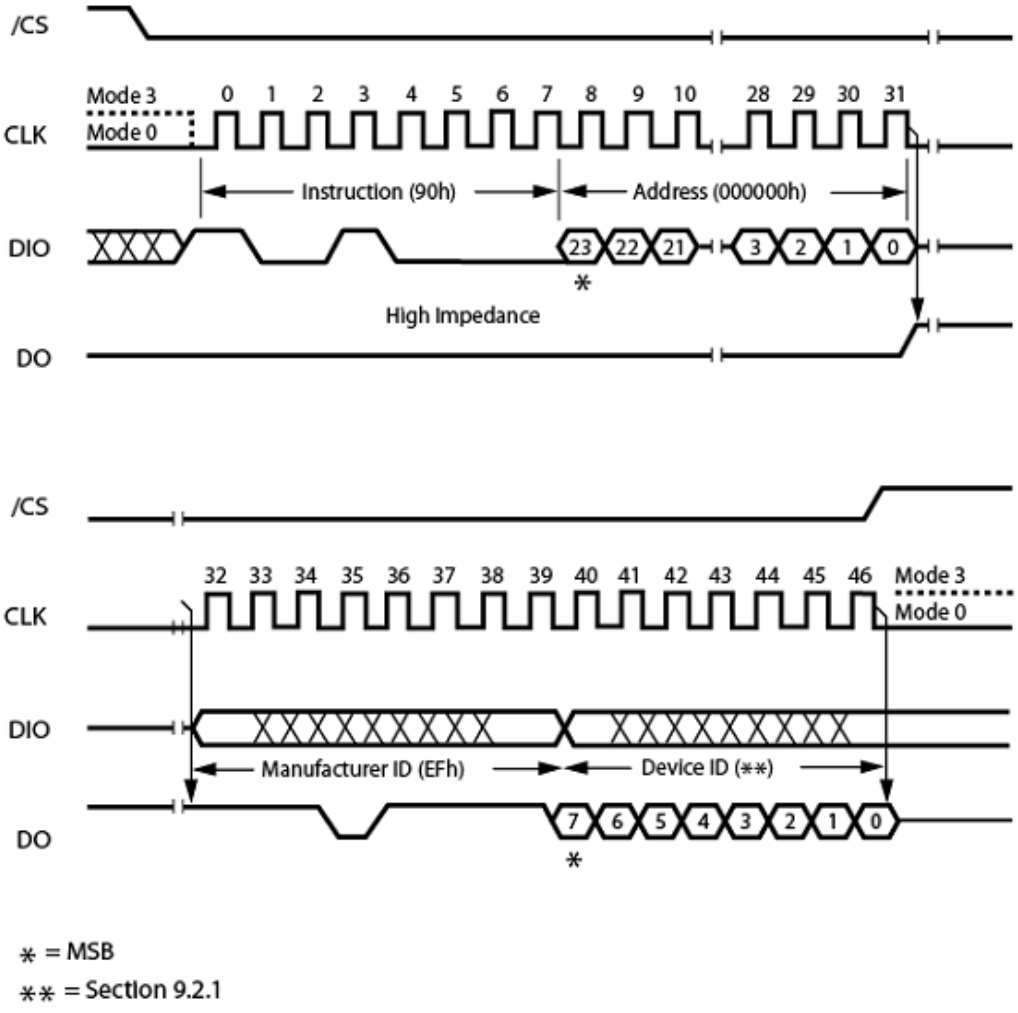
“释放掉电/器件 ID”指令时序图

11.2.16 “读制造/器件号”指令（90h）（Read Manufacture/Device ID）

“读制造/器件号”指令不同于“释放掉电/器件 ID 指令”，“读制造/器件号”指令读出的数据包含 JEDEC 标准制造号和特殊器件 ID 号。

先把/CS 引脚拉低，然后把指令 90h 通过引脚 DIO 送到芯片，然后接着把 24 位地址 000000h 送到芯片，然后芯片会先后把“生产 ID”和“器件 ID”通过 DO 引脚在 CLK 的上升

沿发送出去。如果把 24 位地址写为 000001h，ID 号的发送顺序会颠倒，即先发“器件 ID”后发“生产 ID”。ID 号都是 8 位数据。

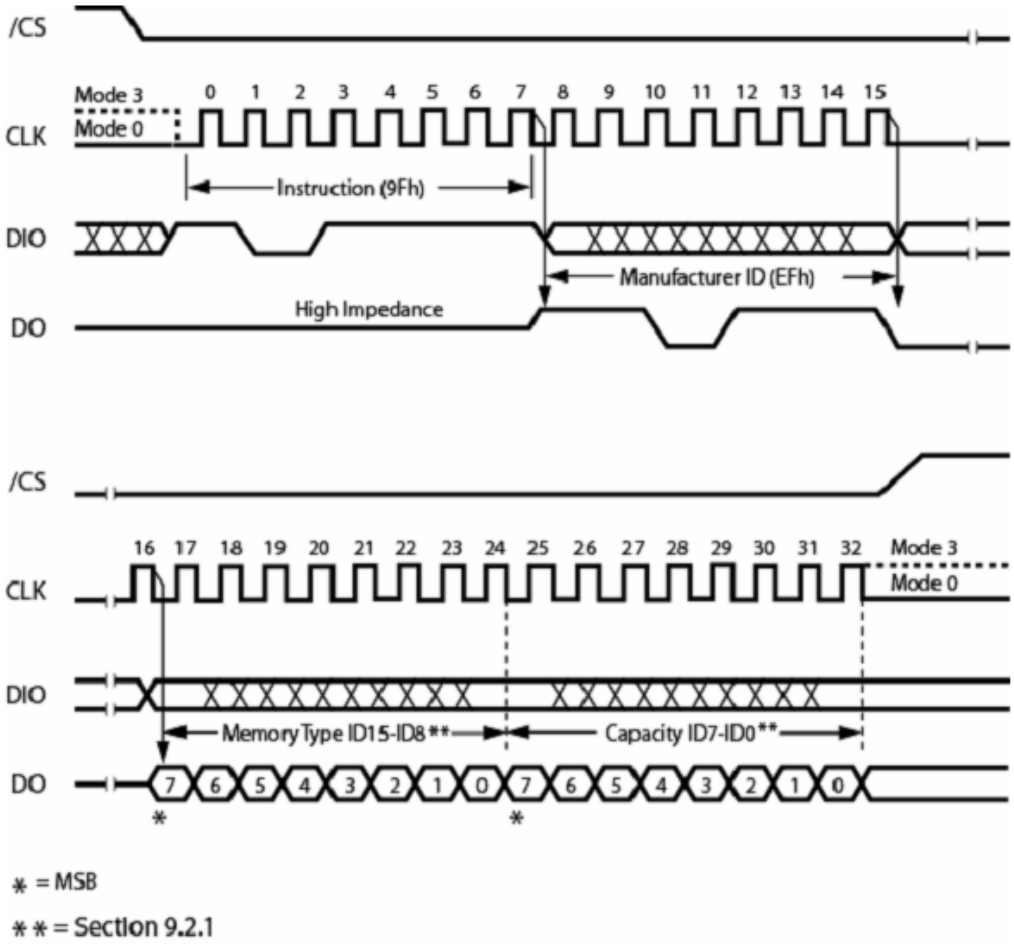


“读生产/器件 ID”指令时序图

11.2.17 JEDEC ID 指令（9Fh）

出于兼容性考虑，W25X16/X32/X64 提供一些指令供电子识别器件 ID 号。

先把/CS 引脚拉低，然后把指令码 9Fh 通过 DIO 引脚发送到芯片，然后“制造 ID”“存储器 ID”“兼容性 ID”将会依次从 DO 引脚在 CLK 的下降沿送出去。每个 ID 都是 8 位数据，高位在前。



“读 JEDEC ID”指令时序图

制造和器件 ID 号：（Manufacturer and Device Identification）

MANUFACTURER ID	(M7-M0)	
Winbond Serial Flash	EFH	
Device ID	**(ID7-ID0)**	**(ID15-ID0)**
Instruction	**ABh, 90h**	**9Fh**
W25X16	14h	3015h
W25X32	15h	3016h
W25X64	16h	3017h

注明：本中文手册是 ration 业余时间翻译，不做商业用途，仅作为和广大电子工程师交流用途，看权威手册请去官方网站下载英文数据手册。W25X16 已经经常被当做中文字库芯片使用，读取数据相当出色。同时也欢迎广大单片机爱好者去 ration 的 ARM Cortex-M0 论坛交流。ration.5d6d.com