ispsubb
数字签名人 ispsubb
DN cn=ispsubb,c=<无>,email=lichaovslijia@163.com
原因：我是该文档的作者
日期 2010.08.09 14:12:42 +08'00'

1,描述

串行 flash 存储器 W25Q128BV 为那些对空间大小，引脚数，功耗有限制的系统提供了一个存储解决方案。25Q 系列的灵活性和性能比一般的串行 flash 设备要高。利用串行 FLASH 可以实现代码映射到 RAM，直接通过 DUAL/QUAD SPI 方式来执行代码，存储声音，文本，数据。W25Q128BV 供电范围为 2.7——3.6V，在激活状态下电流功耗低到 4MA，睡眠状态下则降低到 1UA。所有的 25Q 系列都提供节省空间的封装。

W25Q128BV 由 65536 可编程的页组成的，每页有 256 个字节。一次最多可以写 256 个字节。可以一次擦除 16 页（4KB sector erase），128 页（32KB block erase）,256 页(64KB block erase),或者擦除一整片。W25Q128BV 有 4096 个可擦除的扇区，256 可擦除的块。4KB 的扇区对于数据和参数存储有更高的灵活性。

W25Q128BV 支持标准 SPI 接口，以及更高性能的 DUAL/QUAD SPI ，对应的管脚为时钟，片选，（I/O0）DI，（I/O1）DO，I/O2（/WP），I/O3（/HOLD）。SPI 时钟可以达到 104MHz，在 DUAL 使用快速读时就相当于 208MHz，在 QUAD 使用快速读时相当于 320MHz。这个传输速率比一般的异步 8 位，16 位并行 FLASH 存储器要快。连续读模式访问存储器的效率很高，只要 8 个时钟的指令开销就可以读 24 位地址的数据，这样就可以实现 XIP。

HOLD，WP 管脚，可编程的写保护，可分为顶部，底部，整个存储器。这些提供了更灵活的控制。一般地，W25Q128BV 支持 JEDEC 标准，一个 64 位的独立串行数字，包含制造商和芯片 ID。

6．管脚描述

6.1，封装类型

W25Q128BV 有 8x6 mm 的 WSON（封装代码为 E），300mil 的 SOIC(封装代码为 F)，封装的细节在数据手册的最后。

6.2，片选信号（/CS）

SPI 的片选信号可以使能和禁止芯片的操作。当/CS 管脚为高电平时，芯片处在不选择的状态，此时串行数据输出管脚都处在高阻态。当芯片处在不选中的状态，芯片的功耗将处在待机状态的功耗，除非内部的擦写，编程，写寄存器周期还在进行。当/CS 低电平时，芯片被选择，功耗处在激活状态下的功耗，这时可以进行相应的读写与擦除。上电后，在指令接受之前，/CS 必须从高到低。上电后，/CS 管脚必须跟踪 VCC，所以在/CS 管脚上加一个上拉电阻比较好。

6.3，串行数据输入，输出，IO 口

W25Q128BV 支持标准的 SPI，DUAL SPI，QUAD SPI 操作。标准的 SPI 指令使用单向的 DI 管脚在时钟的上升沿进行传输串行写指令，地址，数据到芯片上。也使用单向的 DO 管脚在时钟的下降沿从芯片上读数据，读寄存器的值。

DUAL 和 QUAD SPI 指令使用双向的 IO 管脚，在时钟的上升沿写指令，地址，数据到芯片上，在时钟的下降沿从芯片上读取数据，寄存器值。QUAD SPI 指令的使用必须把状态寄存器 2 中的非易失性的 QUAD 使能位置位。QE = 1 ，/WP，/HOLD 管脚依次变为 IO2，IO3。

6.4，写保护（/WP）

结合 SRBP 和 SRP 这两个寄存器，小至 4KB，大至整个芯片能被硬件保护。/WP 是低电平有效。当寄存器 SR2 中的 QE 位，被置 1 时，这时的/WP 功能将无效，此时/WP 已经被用为 IO2。

6.5，HOLD（/HOLD）

当芯片被选择时，/HOLD 管脚可以使芯片暂停中止。当芯片/HOLD 为低电平，/CS 也为低电平时，DO 管脚将为高阻态，在 DI 和时钟管脚上的信号将被忽略。当/HOLD 为高电平时，芯片恢复正常操作。当 SPI 总线上挂有多个设备时，/HOLD 脚就有用了。/HOLD 管脚低电平有效。当 QE 置位，/HOLD 管脚作为 IO3，/HOLD 功能此时无效。

6.6，串行时钟

串行时钟为串行输入和输出操作提供时序。

7,存储示意图

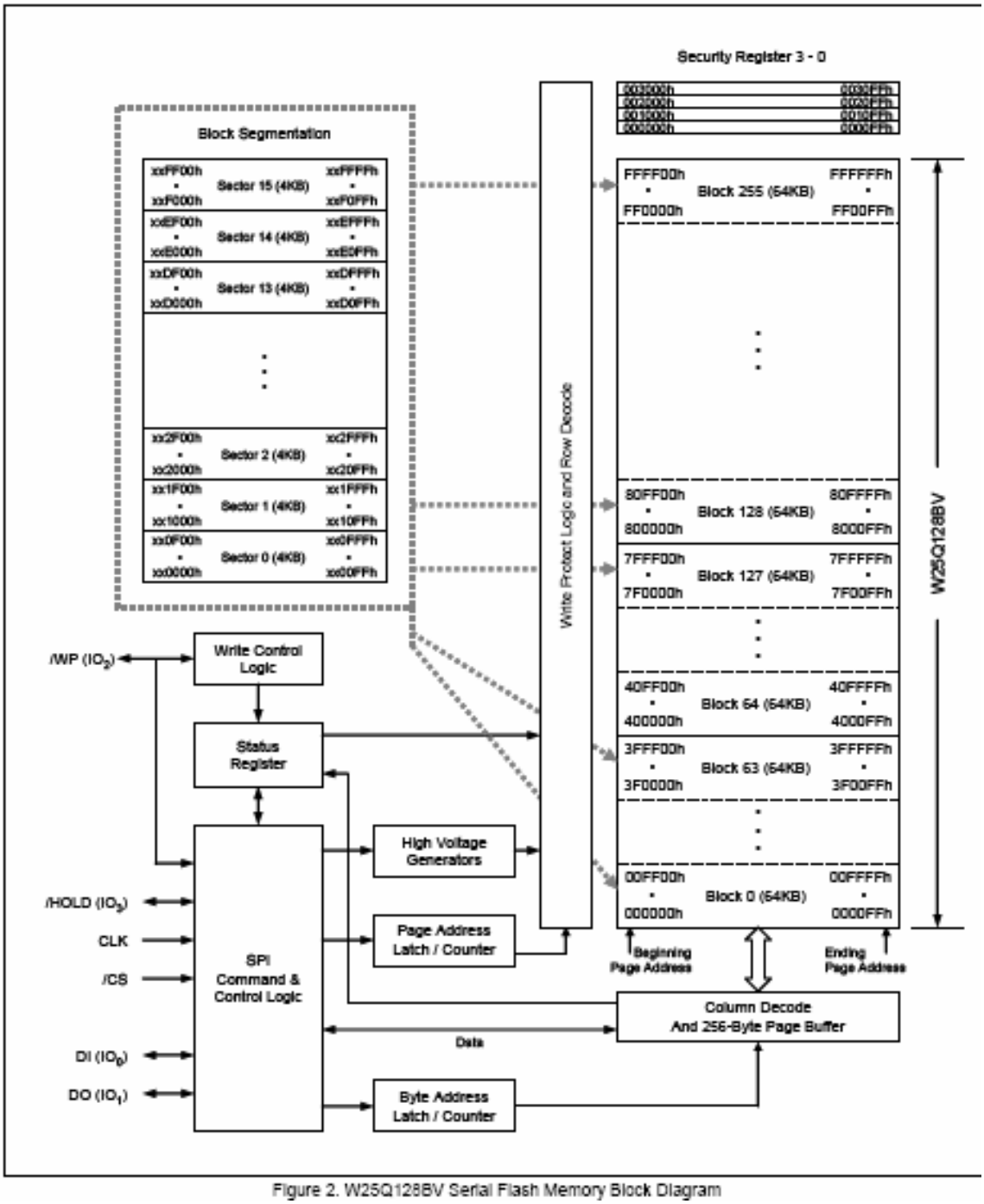



Figure 2. W25Q128BV Serial Flash Memory Block Diagram

8.功能描述

8.1，SPI 操作

8.1.1，标准 SPI 指令

访问 W25Q128BV 可以通过与 SPI 兼容的四线制总线：CLK, /CS , DI, DO。标准 SPI 使用 DI 管脚，在时钟的上升沿，来进行传输串行写指令，地址，数据到芯片。在时钟的下降沿使用 DO 管脚来进行读数据和读状态。

SPI 总线操作模式 0 和模式 3 都是被支持的。模式 0 与模式 3 的最大区别在于，当 SPI 总线的主设备处在待机状态，并且没有数据传输到串行 FLASH 时的 CLK 的信号状态。对模式 0 来说，当/CS 在上升沿和下降沿的时候，CLK 总是在处在低电平。在模式 3 的时候，当/CS 处在上升沿和下降沿的时候 ，CLK 总是处在高电平。

8.1.2 DUAL SPI instructions

8.1.3 QUAD SPI instructions

8.1.4 HOLD 功能

对于标准的 SPI 和 DUAL SPI 操作时，当 W25Q128BV 被激活时（/CS 为低电平），可以使用/HOLD 信号使 W25Q128BV 的操作暂停。当 SPI 的数据与时钟信号与别的设备共用的时候，/HOLD 这个功能就可以用到。例如，当页缓冲区只有部分写进 FLASH，这时一个高优先级的中断要使用 SPI 总线。在这种情况下，/HOLD 功能就可以保持当前指令的状态，等到 SPI 总线再次有效的时候 ，就可以把缓冲区的数据写入 FLASH。/HOLD 功能只在标准的 SPI 和 DUAL SPI 操作下有效，在 QUAD 下无效。

开始/HOLD 功能的条件，芯片必须是处在激活状态，也就是/CS 为低电平。锁定功能在/HOLD 信号下降沿如果 CLK 为低电平的时候被激活。如果 CLK 不在低电平，锁定功能将在下一时钟下降沿被激活。在/HOLD 信号的上升沿时，如果 CLK 已经为低电平，锁定功能将被终止。如果 CLK 不是低电平，那么锁定功能将在下一个时钟下降沿被终止。在锁定期间，DO 处在高阻态，DI 和 CLK 信号被忽略。在整个锁定期间，片选信号应该一直保持低电平，避免重启片内逻辑状态。

8.2，写保护

对于把 W25Q128BV 当非易失性的存储器的应用来说，必须考虑噪声污染和其他的可能破坏数据完整性的系统因素。 为了解决这一问题，W25Q128BV 提供了几种保护数据不意外写进 FLASH 的方法。

8.2.1 写保护的特性

* 芯片复位当 VCC 低于门槛电压
* 唤醒后禁止写的延迟时间
* 写禁止指令，擦写之后自动禁止写
* 软件写保护和硬件写保护
* 使用 Power-down 指令
* Lock Down 写保护
* 一次编程写保护（永久的，不可逆）

无论是唤醒状态还是睡眠状态，当电压 VCC 低于门限电压 Vwi 时，W25Q128BV 都会复。位。当复位重启这个过程中，所有的操作都是被禁止的，指令也是不被识别的。在唤醒

过程中,当电压 VCC 升到高于 Vwi 时,编程和擦除的相关指令仍然要等待一段时间 tpuw，这些指令包括写使能，页写，擦除扇区，擦除块，擦除整个芯片和写状态寄存器。上电过程中，/CS 必须踪电压 VCC，直到电压达到 VCC_min 后再延迟 tvsl 时间。可以在/CS 管脚上加一个上拉电阻来实现。

上电后，芯片处在禁止写的状态，WEL 位为 0。在进行页写，扇区擦除，段擦除，芯片擦除，写状态寄存器之前，必须先发送写使能。完成这些指令后，WEL 位自动清除。

通过写状态寄存器和设置 SRP0，SRP1，BP 相应的位来实现软件控制写保护是很方面的。这种方法可以设置小到 4KB，大到整个芯片为只读，也就是写保护。结合/WP 管脚，可以通过硬件控制来使能或者禁止状态寄存器的改变。详细的说明请看状态寄存器那一段。在睡眠模式下，其实提供了一种额外的写保护，因为此时只识别唤醒指令。

9. 控制和状态寄存器

9.1.1，BUSY

BUSY 位是在 S0 中的一个只读位。当执行页写，QUAD 页写，扇区擦除，块擦除，片擦除，写状态寄存器，擦写安全寄存器时，BUSY 位将会置 1。在 BUSY 位置 1 期间，只识别读状态寄存器指令，擦写 SUSPEND 指令。当擦写指令，写状态寄存器的指令完成后，BUSY 位自动清零，表示芯片已经准备好接受其他指令。

9.1.2，写使能锁存（WEL）

WEL 位是在 S1 中的一个只读位，当执行完写使能的时候置 1。当芯片被写禁止的时候清 0. 当上电过程中或执行以下这些指令后：写禁止，页写，QUAD 页写，扇区擦除，块擦除，芯片擦除，写状态寄存器，擦写安全寄存器时，芯片将处在写禁止状态。

9.1.3，块保护位（BP2，BP1，BP0）

BP 位都是在状态寄存器中非易失性的，可读可写的位（对应为 S4，S3，S2）。写这些位可以进行相应的写保护控制，读这些位可以知道当前块保护的状态。读写这些位都是通过相应的状态寄存器指令。整个芯片或者一部分可以保护起来，当然也可以不保护。默认的就是不保护。

9.1.4，顶部/底部块保护（TB)、

非易失性的 TB 位，可以控制 BP 是从顶部开始还是底部开始。TB=0，从顶部开始，TB=1 从底部开始。默认的是从 TB=0 顶部开始的。TB 位是可以通过写状态寄存器指令来操作的。

9.1.5，扇区/块保护（SEC）

非易失性的 SEC 位决定是以 4KB 扇区还是以 64KB 的块来进行存储。SEC 默认值是 0，以 64KB 的块来进行保护。

9.1.6，完全保护（CMP）

CMP 是在状态寄存器中的一位非易失性的可读可写的位（S14）。与 SEC，TB，BP 位结合，使得存储器的保护措施更加灵活。一旦 CMP 位置 1 ，SEC，TB，BP 位将保留。当 CMP=0，当顶部 4KB 以外的部分没有被保护的时候，顶部 4KB 将被保护。当 CMP = 1，除顶部 4KB 以外的部分将被保护。默认的 CMP = 0。详细情况参看 SRMP 参考。

9.1.7，状态寄存器保护位（SRP1，SRP0）

SRP1，SRP0 是状态寄存器中非易失性的可读写的两位（S8，S7）。SRP 位决定着写保护的方法，软保护，硬件保护，电源锁定，一次编程保护。

| SRP1 | SRP0 | /WP | Status Register | Description |
|---|---|---|---|---|
| 0 | 0 | X | Software Protection | /WP pin has no control. The Status register can be written to after a Write Enable instruction, WEL=1. [Factory Default] |
| 0 | 1 | 0 | Hardware Protected | When /WP pin is low the Status Register locked and can not be written to. |
| 0 | 1 | 1 | Hardware Unprotected | When /WP pin is high the Status register is unlocked and can be written to after a Write Enable instruction, WEL=1. |
| 1 | 0 | X | Power Supply Lock-Down | Status Register is protected and can not be written to again until the next power-down, power-up cycle.(1) |
| 1 | 1 | X | One Time Program(2) | Status Register is permanently protected and can not be written to. |

注意：1，当 SPR1，SRP0 = (1,0)，一个 POWER-DOWN，POWER-UP 周期后，SRP1，SRP0 就变为了（0，0）状态。

2，这是一条特别的命令。详细情况联系华邦。

9.1.8，擦写暂停寄状态位（SUS）

SUS 是在状态寄存器中的只读位（S15）。当执行完擦写暂停指令位后，SUS 置位。当执行指令 7AH 以及 POWER-DOWN，POWER-UP 周期后，SUS 被清零。

9.1.9，安全寄存器锁定位（LB3，LB2，LB1，LB0）

LB 位是在状态寄存器中非易失性的一次编程位（S13，S12，S11，S10）。LB 控制写保护，描述安全寄存器的状态。默认的 LB 位都为 0,即安全寄存器没有被锁定。LB 位可以通写状态寄存器指令来置 1 。LB 位都是一次编程，所以一旦写 1 ，相应的 256 字节的安全寄存器将会变得永久只读。

9.1.10，QUAD 使能（QE）

QE 是在状态寄存器中的非易失性的可读可写的一位（S9）。QE 置位，表示进行的是 QUAD SPI 操作。默认的 QE 是 0，/WP，/HOLD 都是使能的。当 QE 是 1 时，IO2，IO3 管脚被被使能。

注意：当芯片处在标准 SPI 或者 DUAL SPI ，并且/WP，/HOLD 管脚夫是直接接在高电电平或者地上时，QE 位不要设置为 1。

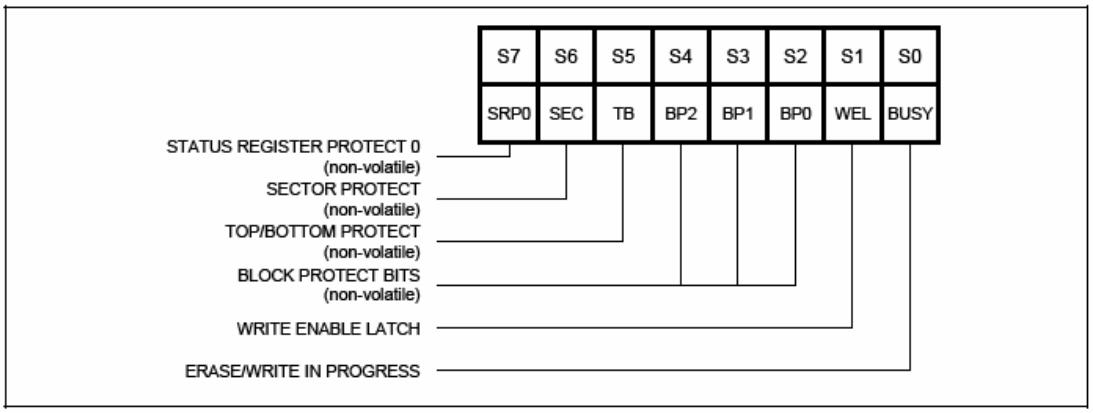



Figure 3a. Status Register-1

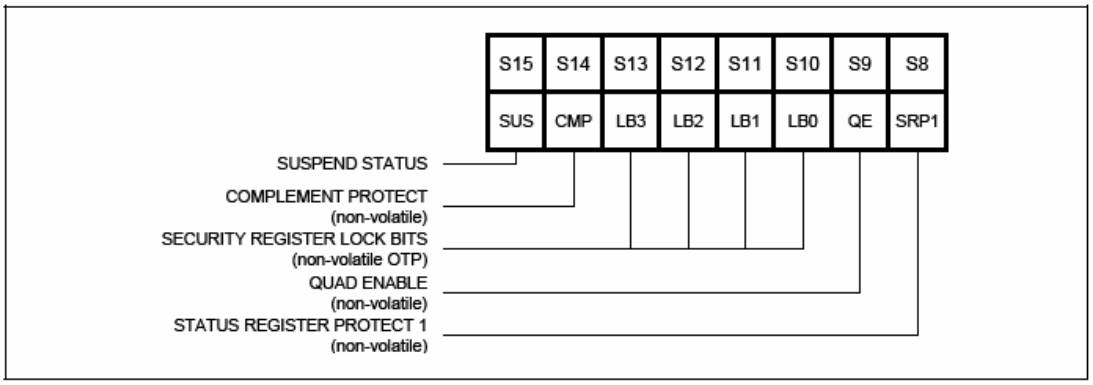



Figure 3b. Status Register-2

9.1.11，状态寄存器存储保护（CMP=0）

| STATUS REGISTER[1] | | | | | W25Q128BV (128M-BIT) MEMORY PROTECTION | | | |
|---|---|---|---|---|---|---|---|---|
| SEC | TB | BP2 | BP1 | BP0 | PROTECTED BLOCK(S) | PROTECTED ADDRESSES | PROTECTED DENSITY | PROTECTED PORTION[2] |
| X | X | 0 | 0 | 0 | NONE | NONE | NONE | NONE |
| 0 | 0 | 0 | 0 | 1 | 252 ~ 255 | FC0000h – FFFFFFh | 256KB | Upper 1/64 |
| 0 | 0 | 0 | 1 | 0 | 248 ~ 255 | F80000h – FFFFFFh | 512KB | Upper 1/32 |
| 0 | 0 | 0 | 1 | 1 | 240 ~ 255 | F00000h – FFFFFFh | 1MB | Upper 1/16 |
| 0 | 0 | 1 | 0 | 0 | 224 ~ 255 | E00000h – FFFFFFh | 2MB | Upper 1/8 |
| 0 | 0 | 1 | 0 | 1 | 192 ~ 255 | C00000h – FFFFFFh | 4MB | Upper 1/4 |
| 0 | 0 | 1 | 1 | 0 | 128 ~ 255 | 800000h – FFFFFFh | 8MB | Upper 1/2 |
| 0 | 1 | 0 | 0 | 1 | 0 ~ 3 | 000000h – 03FFFFh | 256KB | Lower 1/64 |
| 0 | 1 | 0 | 1 | 0 | 0 ~ 7 | 000000h – 07FFFFh | 512KB | Lower 1/32 |
| 0 | 1 | 0 | 1 | 1 | 0 ~ 15 | 000000h – 0FFFFFh | 1MB | Lower 1/16 |
| 0 | 1 | 1 | 0 | 0 | 0 ~ 31 | 000000h – 1FFFFFh | 2MB | Lower 1/8 |
| 0 | 1 | 1 | 0 | 1 | 0 ~ 63 | 000000h – 3FFFFFh | 4MB | Lower 1/4 |
| 0 | 1 | 1 | 1 | 0 | 0 ~ 127 | 000000h – 7FFFFFh | 8MB | Lower 1/2 |
| X | X | 1 | 1 | 1 | 0 ~ 255 | 000000h – FFFFFFh | 16MB | ALL |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 ~ 3 | 000000h – 03FFFFh | 256KB | Lower 1/64 |
| 0 | 1 | 0 | 1 | 0 | 0 ~ 7 | 000000h – 07FFFFh | 512KB | Lower 1/32 |
| 0 | 1 | 0 | 1 | 1 | 0 ~ 15 | 000000h – 0FFFFFh | 1MB | Lower 1/16 |
| 0 | 1 | 1 | 0 | 0 | 0 ~ 31 | 000000h – 1FFFFFh | 2MB | Lower 1/8 |
| 0 | 1 | 1 | 0 | 1 | 0 ~ 63 | 000000h – 3FFFFFh | 4MB | Lower 1/4 |
| 0 | 1 | 1 | 1 | 0 | 0 ~ 127 | 000000h – 7FFFFFh | 8MB | Lower 1/2 |
| X | X | 1 | 1 | 1 | 0 ~ 255 | 000000h – FFFFFFh | 16MB | ALL |
| 1 | 0 | 0 | 0 | 1 | 255 | FFF000h – FFFFFFh | 4KB | U - 1/4096 |
| 1 | 0 | 0 | 1 | 0 | 255 | FFE000h – FFFFFFh | 8KB | U - 1/2048 |
| 1 | 0 | 0 | 1 | 1 | 255 | FFC000h – FFFFFFh | 16KB | U - 1/1024 |
| 1 | 0 | 1 | 0 | X | 255 | FF8000h – FFFFFFh | 32KB | U - 1/512 |
| 1 | 1 | 0 | 0 | 1 | 0 | 000000h – 000FFFh | 4KB | L - 1/4096 |
| 1 | 1 | 0 | 1 | 0 | 0 | 000000h – 001FFFh | 8KB | L - 1/2048 |
| 1 | 1 | 0 | 1 | 1 | 0 | 000000h – 003FFFh | 16KB | L - 1/1024 |
| 1 | 1 | 1 | 0 | X | 0 | 000000h – 007FFFh | 32KB | L - 1/512 |

**Note:**
1. x = don't care
2. L = Lower; U = Upper

9.1.12，状态寄存器存储保护（CMP=1）

| STATUS REGISTER[1] | | | | | W25Q128BV (128M-BIT) MEMORY PROTECTION | | | |
|---|---|---|---|---|---|---|---|---|
| SEC | TB | BP2 | BP1 | BP0 | PROTECTED BLOCK(S) | PROTECTED ADDRESSES | PROTECTED DENSITY | PROTECTED PORTION[2] |
| X | X | 0 | 0 | 0 | 0 ~ 255 | 000000h - FFFFFFh | 16MB | ALL |
| 0 | 0 | 0 | 0 | 1 | 0 ~ 251 | 000000h - FBFFFFh | 16,128KB | Lower 63/64 |
| 0 | 0 | 0 | 1 | 0 | 0 ~ 247 | 000000h – F7FFFFh | 15,872KB | Lower 31/32 |
| 0 | 0 | 0 | 1 | 1 | 0 ~ 239 | 000000h - EFFFFFh | 15MB | Lower 15/16 |
| 0 | 0 | 1 | 0 | 0 | 0 ~ 223 | 000000h - DFFFFFh | 14MB | Lower 7/8 |
| 0 | 0 | 1 | 0 | 1 | 0 ~ 191 | 000000h - BFFFFFh | 12MB | Lower 3/4 |
| 0 | 0 | 1 | 1 | 0 | 0 ~ 127 | 000000h - 7FFFFFh | 8MB | Lower 1/2 |
| 0 | 1 | 0 | 0 | 1 | 4 ~ 255 | 040000h - FFFFFFh | 16,128KB | Upper 63/64 |
| 0 | 1 | 0 | 1 | 0 | 8 ~ 255 | 080000h - FFFFFFh | 15,872KB | Upper 31/32 |
| 0 | 1 | 0 | 1 | 1 | 16 ~ 255 | 100000h - FFFFFFh | 15MB | Upper 15/16 |
| 0 | 1 | 1 | 0 | 0 | 32 ~ 255 | 200000h - FFFFFFh | 14MB | Upper 7/8 |
| 0 | 1 | 1 | 0 | 1 | 64 ~ 255 | 400000h - FFFFFFh | 12MB | Upper 3/4 |
| 0 | 1 | 1 | 1 | 0 | 128 ~ 255 | 800000h - FFFFFFh | 8MB | Upper 1/2 |
| X | X | 1 | 1 | 1 | NONE | NONE | NONE | NONE |
| 1 | 0 | 0 | 0 | 1 | 0 ~ 254 | 000000h – FFEFFFh | 16,380KB | L - 4095/4096 |
| 1 | 0 | 0 | 1 | 0 | 0 ~ 254 | 000000h – FFDFFFh | 16,376KB | L - 2047/2048 |
| 1 | 0 | 0 | 1 | 1 | 0 ~ 254 | 000000h – FFBFFFh | 16,368KB | L - 1023/1024 |
| 1 | 0 | 1 | 0 | X | 0 ~ 254 | 000000h – FF7FFFh | 16,352KB | L - 511/512 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 1 ~ 255 | 001000h – FFFFFFh | 16,380KB | U - 4095/4096 |
| 1 | 1 | 0 | 1 | 0 | 1 ~ 255 | 002000h – FFFFFFh | 16,376KB | U - 2047/2048 |
| 1 | 1 | 0 | 1 | 1 | 1 ~ 255 | 004000h – FFFFFFh | 16,368KB | U -1023/1024 |
| 1 | 1 | 1 | 0 | X | 1 ~ 255 | 008000h – FFFFFFh | 16,352KB | U - 511/512 |

**Note:**
1. x = don't care
2. L = Lower; U = Upper

9.2，指令

W25Q128BV 共有 34 个基本指令。指令都是以/CS 下降沿开始的。第一个传输的字节是指令码。在 DI 上传输的数据是在时钟的上升沿被锁存的。MSB 首先被传输。

指令的长度是不定的，有单字节的，也有多字节的。指令代码后可能还跟有地址，数据，无关的字节，有可能还是几个的组合。指令是以/CS 上升沿结束的。每个指令的时序图都包含在图片 4 中。所有的读指令可以在任何时钟周期结束。但是，所有的擦写指令必须在/CS 拉高之后还有一个 8 位的时钟间隔，否则前面的擦写指令将被忽略。这样可以保护芯片意外被写。当芯片正在进行擦写操作，写寄存器时，直到完成，除了读状态寄存器以外的所有指令都将被忽略。

9.2.1，制造商和芯片 ID

| **MANUFACTURER ID** | **(MF7-MF0)** | |
|---|---|---|
| Winbond Serial Flash | EFh | |
| | | |
| **Device ID** | **(ID7-ID0)** | **(ID15-ID0)** |
| **Instruction** | **ABh, 90h, 92h, 94h** | **9Fh** |
| W25Q128BV | 17h | 4018h |

9.2.2，指令表 1（擦写表 1）

| INSTRUCTION NAME | BYTE 1 (CODE) | BYTE 2 | BYTE 3 | BYTE 4 | BYTE 5 | BYTE 6 |
|---|---|---|---|---|---|---|
| Write Enable | 06h | | | | | |
| Write Enable for Volatile Status Register | 50h | | | | | |
| Write Disable | 04h | | | | | |
| Read Status Register-1 | 05h | (S7–S0) (2) | | | | |
| Read Status Register-2 | 35h | (S15–S8) (2) | | | | |
| Write Status Register | 01h | S7–S0 | S15-S8 | | | |
| Page Program | 02h | A23–A16 | A15–A8 | A7–A0 | D7–D0 | |
| Quad Page Program | 32h | A23–A16 | A15–A8 | A7–A0 | D7–D0, ... (3) | |
| Sector Erase (4KB) | 20h | A23–A16 | A15–A8 | A7–A0 | | |
| Block Erase (32KB) | 52h | A23–A16 | A15–A8 | A7–A0 | | |
| Block Erase (64KB) | D8h | A23–A16 | A15–A8 | A7–A0 | | |
| Chip Erase | C7h/60h | | | | | |
| Erase / Program Suspend | 75h | | | | | |
| Erase / Program Resume | 7Ah | | | | | |
| Power-down | B9h | | | | | |
| Continuous Read Mode Reset (4) | FFh | FFh | | | | |

9.2.3，指令表 2（读指令）

| INSTRUCTION NAME | BYTE 1 (CODE) | BYTE 2 | BYTE 3 | BYTE 4 | BYTE 5 | BYTE 6 |
|---|---|---|---|---|---|---|
| Read Data | 03h | A23-A16 | A15-A8 | A7-A0 | (D7-D0) | |
| 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 Quad Output | 6Bh | A23-A16 | A15-A8 | A7-A0 | dummy | (D7-D0, ...)(3) |
| Fast Read Dual I/O | BBh | A23-A8(2) | A7-A0, M7-M0(2) | (D7-D0, ...)(1) | | |
| Fast Read Quad I/O | EBh | A23-A0, M7-M0(4) | (x,x,x,x, D7-D0, ...)(5) | (D7-D0, ...)(3) | | |
| Word Read Quad I/O(7) | E7h | A23-A0, M7-M0(4) | (x,x, D7-D0, ...)(6) | (D7-D0, ...)(3) | | |
| Octal Word Read Quad I/O(8) | E3h | A23-A0, M7-M0(4) | (D7-D0, ...)(3) | | | |
| Set Burst with Wrap | 77h | xxxxxx, W6-W4(4) | | | | |

9.2.4，指令表 4（ID，安全指令）

| INSTRUCTION NAME | BYTE 1 (CODE) | BYTE 2 | BYTE 3 | BYTE 4 | BYTE 5 | BYTE 6 |
|---|---|---|---|---|---|---|
| Release Power down/ Device ID | ABh | dummy | dummy | dummy | (ID7-ID0)(1) | |
| Manufacturer/ Device ID(2) | 90h | dummy | dummy | 00h | (MF7-MF0) | (ID7-ID0) |
| Manufacturer/Device ID by Dual I/O | 92h | A23-A8 | A7-A0, M[7:0] | (MF[7:0], ID[7:0]) | | |
| Manufacture/Device ID by Quad I/O | 94h | A23-A0, M[7:0] | xxxx, (MF[7:0], ID[7:0]) | (MF[7:0], ID[7:0], ...) | | |
| JEDEC ID | 9Fh | (MF7-MF0) Manufacturer | (ID15-ID8) Memory Type | (ID7-ID0) Capacity | | |
| Read Unique ID | 4Bh | dummy | dummy | dummy | dummy | (ID63-ID0) |
| Erase Security Registers(3) | 44h | A23–A16 | A15–A8 | A7–A0 | | |
| Program Security Registers(3) | 42h | A23–A16 | A15–A8 | A7–A0 | D7-D0 | D7-D0 |
| Read Security Registers(3) | 48h | A23–A16 | A15–A8 | A7–A0 | dummy | (D7-0) |

9.2.5，写使能（06h）

写使能指可以设置状态寄存器中的 WEL 位置 1 。在页写，QUAD 页写，扇区擦除，块擦除，片擦除，写状态寄存器，擦写安全寄存器指令之前，必须先将 WEL 位置 1 。写使能指令是以/CS 拉低开始的，将 06H 通过 DI 在时钟的上升沿锁存，然后/CS 拉高来结束指令。

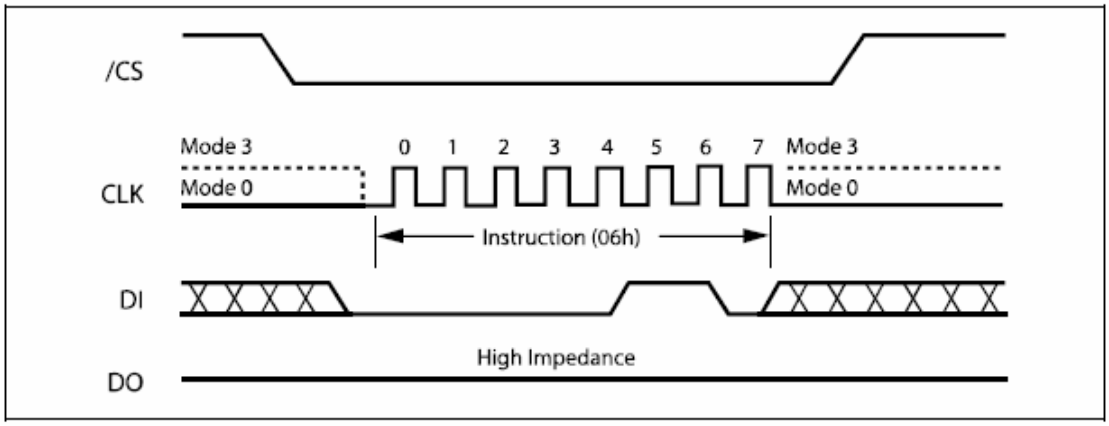



Figure 4. Write Enable Instruction Sequence Diagram

9.2.6，易失性的寄存器写使能（50H）

非易失性的状态寄存器位在 9.1 节已经描述过，也可以像易失性的位一样进行写操作。这样对芯片配置会更灵活，对存储保护也不用等待慢速的非易失性位的写入，也就不会影响到状态寄存器中的非易失性位。为了写状态寄存器中的那些易失性位的值，必须在在写状态寄存器指令（01H）前，写入易失性的状态寄存器指令（50H）。写易失性的状态寄存器指令不会使改变 WEL 位。这条指令的惟一的作用是使得写入写状态寄存器指令可以改变状态寄存器中的易失性位。

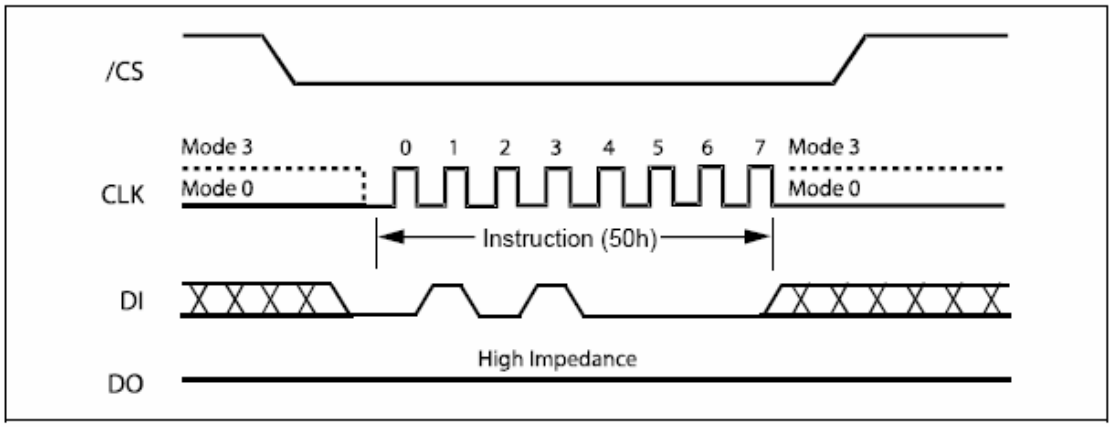



Figure 5. Write Enable for Volatile Status Register Instruction Sequence Diagram

9.2.7，写禁止（04H）

写禁止指令使得状态寄存器中的 WEL 位为 0。写禁止指令是以/CS 低电平开始的，然后在 DI 传输指令代码 04H 到芯片，最后是以/CS 拉高结束。当上电后，以及完成写状态寄存器，擦写安全寄存器，页写，QUAD 页写，扇区擦写，块擦写，芯片擦写指令后，WEL 位是自动清零的。

写禁止指令（04H）也被用来取消易失性的写使能指令（50H）。

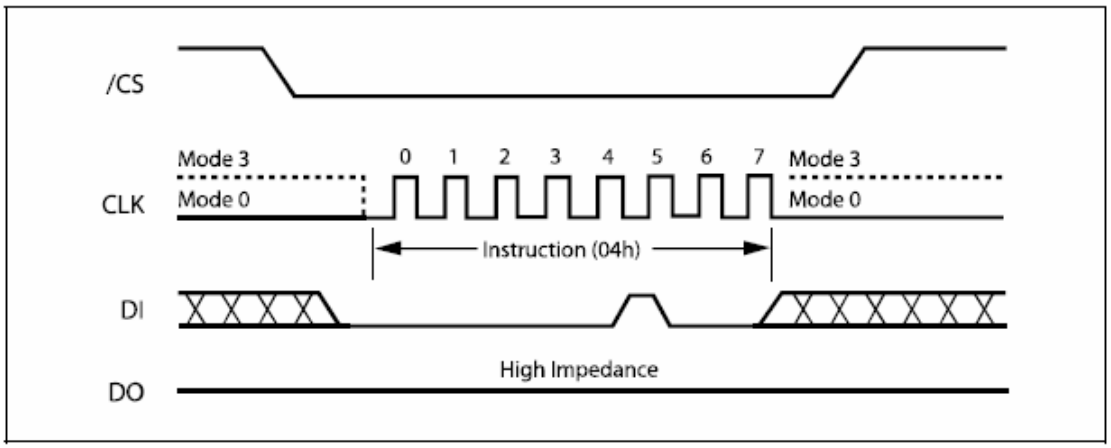



Figure 6. Write Disable Instruction Sequence Diagram

9.2.8，读状态寄存器 1（05H）和读状态寄存器 2（35H）指令

读状态寄存器指令允许读 8 位状态寄存器位。这条指令是以/CS 拉低开始，然后通过 DI 在时钟的上升沿传输指令代码 05H(读寄存器 1 指令)或者是 35H(读寄存器 2 指令)，然后状态寄存器的相应位通过 DO 在时钟的下降沿从高位到低位依次传出。最后以/CS 拉高结束。读状态寄存指令可以任何时间使用，在擦写，写状态寄存器指令周期中依然可以。这样就可以随时检查 BUSY 位，检查相应的指令周期有没有结束，芯片是不是可以接受新的指令。状态寄存器可以连续的读出来，如图 7。

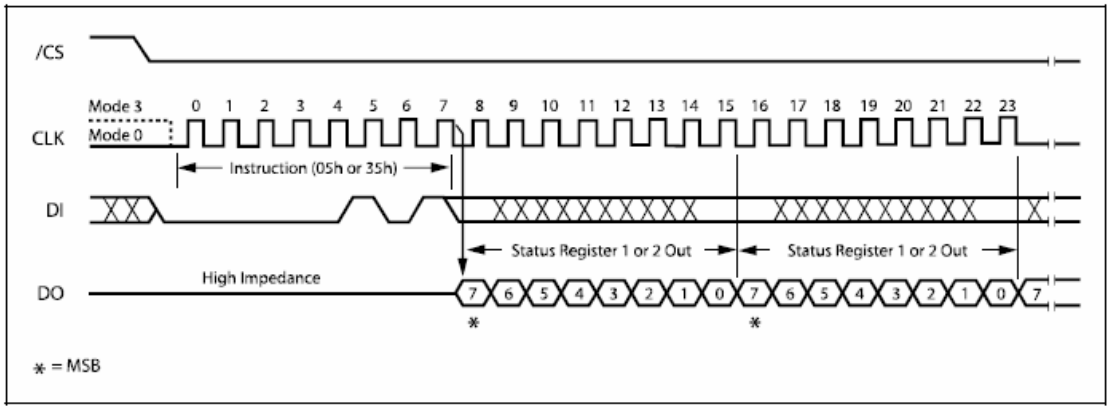



Figure 7. Read Status Register Instruction Sequence Diagram

9.2.9，写状态寄存器（01H）

写状态寄存器可对状态寄存器进行写操作。但是仅仅是那非易失性的位，SRP0，SEC，TB，BP，CMP，LB，QE，SRP1。其他的位将不会受到这条指令的影响。LB 是非易失性的一次编程（OTP）位，一旦写 1，将不能清零。

对于写非易失性的状态寄存器位，写状态寄存器指令（01H）之前，必须先写入写使能(06H)指令（要保证 WEL 位为 1）。一旦写使能，写状态寄存器指令（01H）将会以/CS 拉低开始，然后通过 DI 线上传输指令代码（01H），然后传输相应的状态寄存器位。最后以/CS 拉高结束。

对于写易失性的状态寄存位，在写状态寄存器之前必须先写入写易失性状态寄存器位指令（50H）。SRP1，LB 位是一次编程，所以不可能从 1 写为 0。易失性的位在掉电后其值就被丢掉了，非易失性的位则一直保持。

为了完成写状态寄存器指令，/CS 管脚必须在数据传输完成后拉高。如果/CS 没有拉高，那么写状态寄存指令将不会被最终执行。如果/CS 在第 8 个时钟拉高（25 系列的），CMP，QE，SRP1 将被清零。

在对非易失性的状态寄存器位写操作（06 指令与 01H 指令）期间，当/CS 拉高后，其内建时间是 tw。当写状态寄存指令在执行期间，读状态寄存器指令仍然可以继续，去检查 BUYS 位。在写状态寄存器指令周期中 BUSY 位为 1，当该指令完成后，BUSY 位被清零，WEL 位也被清零，表明芯片准备接受的新指令。

在对易失性寄存器位（50H 和 01H）操作时，/CS 拉高后，在 tSHSL2 时间内，状态寄存器的相应位将被新的值刷新。在状态寄存器刷新期间，BUSY 位仍然是 0。

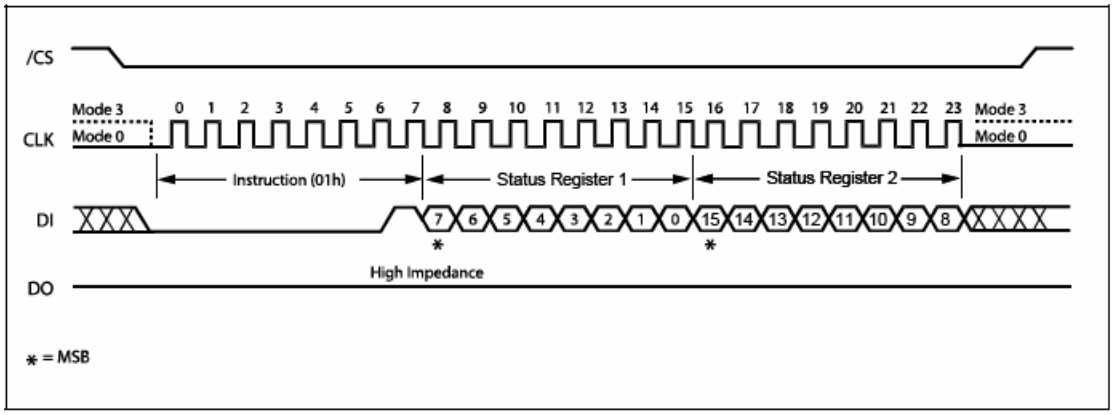



Figure 8. Write Status Register Instruction Sequence Diagram

9.2.10，读数据（03H）

读数据指令允许从存储器读一个字节和连续多个字节。该指令是以/CS 拉低开始，然后通 DI 在时钟的上升沿来传输指令代码（03H）和 24 位地址。当芯片接受完地址位后，相应地址处的值将会，在时钟的下降沿，以高位在前低位在后的方式，在 DO 上传输。如果连续的读多个字节的话，地址是自动加 1 的。这意味着可以一次读出整个芯片。该指令也是以/CS 拉高来结束的。

如果当 BUSY=1 时执行该指令，该指令将被忽略，并且对正在执行的其他指令不会有任何影响。读数据指令的时钟可以从 D.C 到最大的 fR.

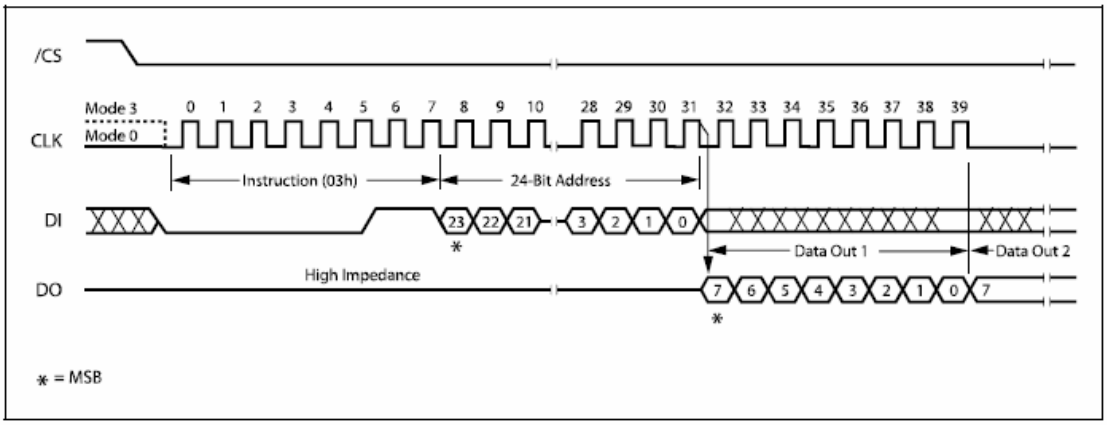



Figure 9. Read Data Instruction Sequence Diagram

9.2.11,快速读指令（0BH）

快速读指令与读数据指令比较相似，差别在于快速读数据指令可以以最高的频率读。只是在传输完 24 位地址后，另加 8 位无关的数据。在传输 8 位无关的数据时间内，芯片用来建立初始地址，这时 DO 上的数据也是无关的。

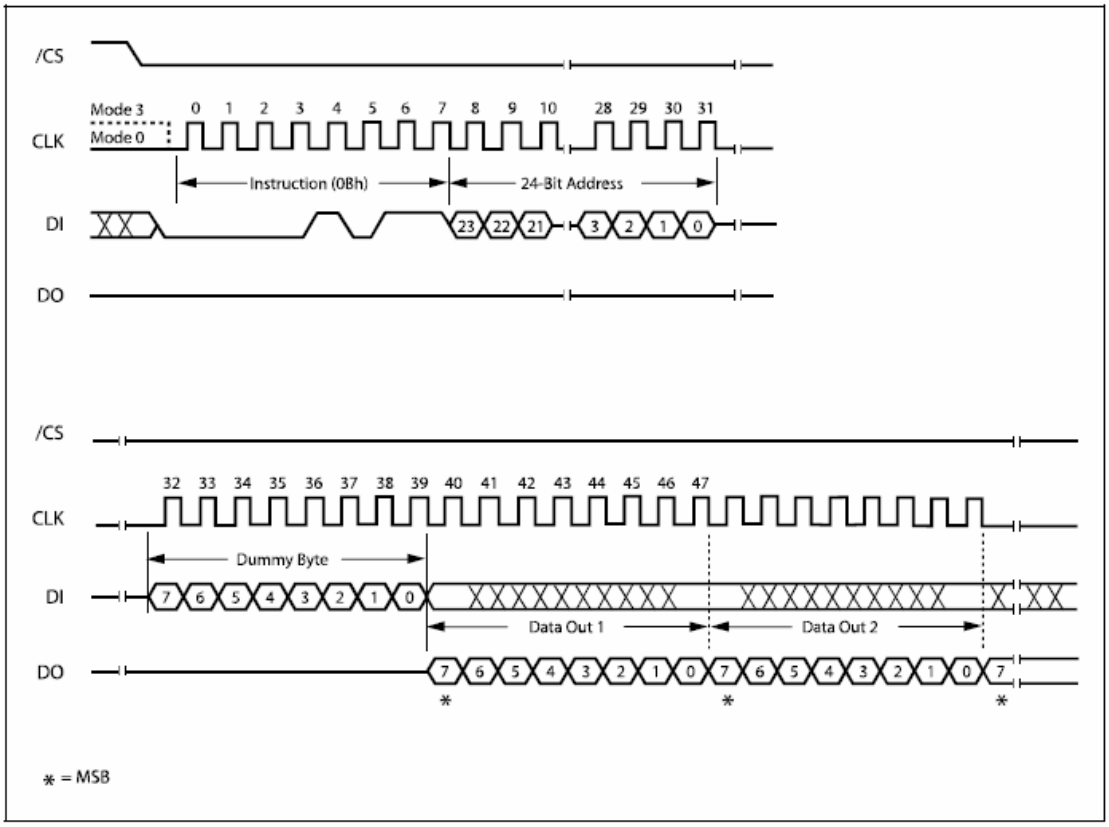



Figure 10. Fast Read Instruction Sequence Diagram

9.2.12，DUAL 输出快速读（3BH） 略
9.2.13，QUAD 快速读（6BH） 略
9.2.14，DUAL IO 快速读（BBH） 略
9.2.15，QUAD IO 快速读（EBH） 略
9.2.16，字读 QUAD（E7H） 略
9.2.17，8 个字一起读（E3H）略
9.2.18————9.2.20 略

9.2.21，页编程指令（02H）

页编程指令允许 1 到 256 字节写入存储器的某一页，这一页必须是被擦除过的（也就是只能写 0，不能写 1，擦除时是全写为 1）。在页编程指令之前，必须先写入写使能指令。页编程指令是以/CS 拉低开始，然后在 DI 上传输指令代码 02H，再接着传输 24 位的地址，接着是至少一个字节的数据。/CS 管脚必须一直保持低。页编程指令的时序图如图 19。

如果一次写一整页数据（256 字节），最后的地址字节应该全为 0。如果最后 8 字节地址不为 0，但是要写入的数据长度超过页剩下的长度，那么芯片会回到当前页的开始地址写。写入少于 256 字节的的数据，对页内的其他数据没有任何影响。对于这种情况的惟一要求是，时钟数不能超过剩下页的长度。如果一次写入多于是 256 字节的数据，那么在页内会回头写，先前写的数据可能已经被覆盖。

作为擦写指令，当最后字节的第 8 位进入芯片后，/CS 必须拉高。如果/CS 没有拉高，

那么页写指令将不被执行。/CS 拉高后，页编程指令的内建时间为 tpp。在页写指令执行期间，读状态寄存器指令仍然可以识别，以此来进行检查 BUSY 位。当页写指令执行期间，BUSY 位为了 1 。当执行完后，BUSY 为 0，表明可以接受新的指令了。页写指令完成后 WEL 位自动清零。如果该指令要操作的页已经被保护起来，那么该指令也将不执行。

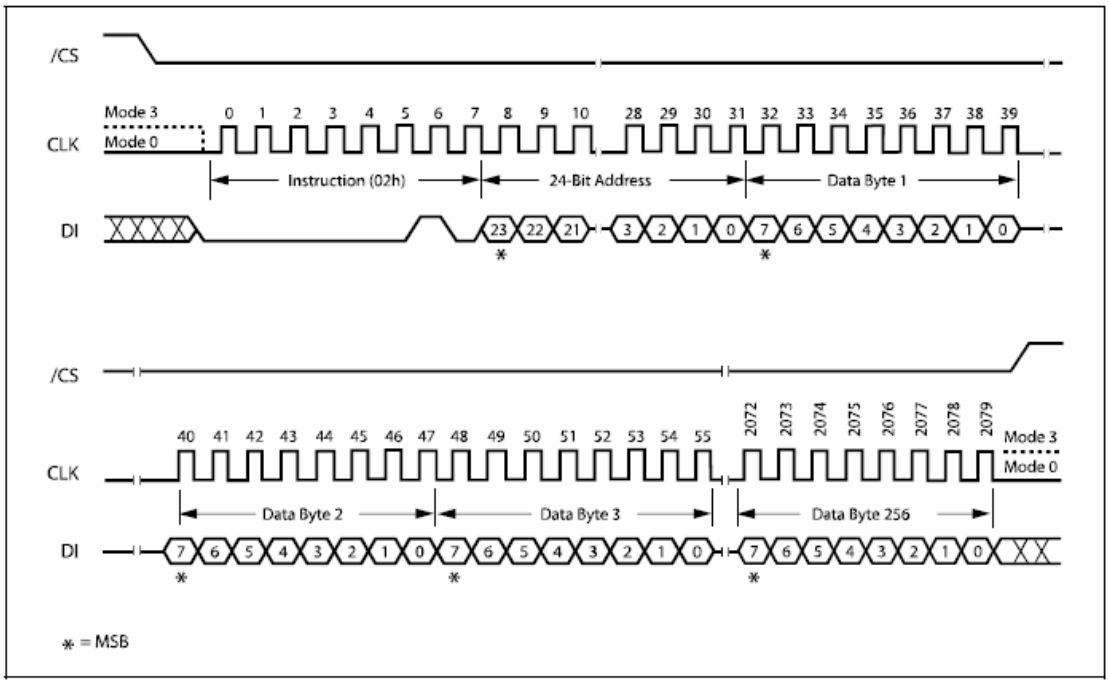



Figure 19. Page Program Instruction Sequence Diagram

9.2.22，QUAD INPUT PAGE PROGRAM(32h) 略

9.2.23，扇区擦除（20H）

扇区擦除可以擦除 4K-byte 存储空间（全为 0XFF）。进行扇区擦写指令之前，必须进行写使能指令。该指令是以/CS 拉低开始的，然后在 DI 上传输指令代码 20H 和 24 位地址。时序图如图 21。

当最后字节的第 8 位进入芯片后，/CS 必须拉高。如果/CS 没有拉高，那么扇区擦写指令将不被执行。/CS 拉高后，扇区擦写指令的内建时间为 tSE。在扇区擦写指令执行期间，读状态寄存器指令仍然可以识别，以此来进行检查 BUSY 位。当扇区擦写指令执行期间，BUSY 位为了 1 。当执行完后，BUSY 为 0，表明可以接受新的指令了。扇区擦写指令完成后 WEL 位自动清零。如果该指令要操作的任何一页已经被保护起来，那么该指令也将不执行。

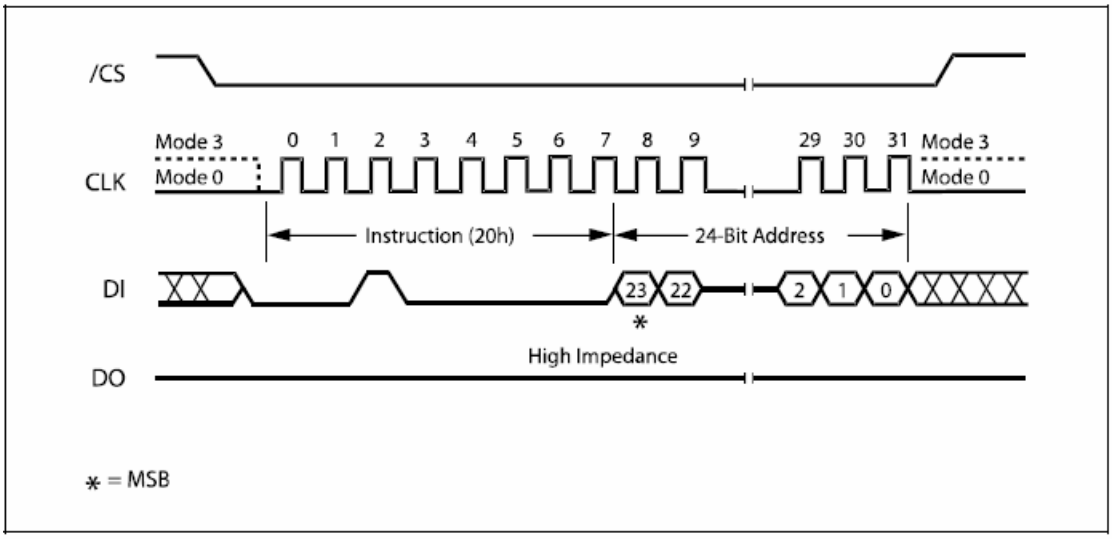



Figure 21. Sector Erase Instruction Sequence Diagram

9.2.24，32KB 块擦除指令（52H）

块擦除可以擦除 32K-byte 存储空间（全为 0XFF）。进行块擦写指令之前，必须进行写使能指令。该指令是以/CS 拉低开始的，然后在 DI 上传输指令代码 02H 和 24 位地址。时序图如图 22。

当最后字节的第 8 位进入芯片后，/CS 必须拉高。如果/CS 没有拉高，那么块擦写指令将不被执行。/CS 拉高后，块擦写指令的内建时间为 tBE1。在块擦写指令执行期间，读状态寄存器指令仍然可以识别，以此来进行检查 BUSY 位。当块擦写指令执行期间，BUSY 位为了 1。当执行完后，BUSY 为 0，表明可以接受新的指令了。块擦写指令完成后 WEL 位自动清零。如果该指令要操作的任何一页已经被保护起来，那么该指令也将不执行。

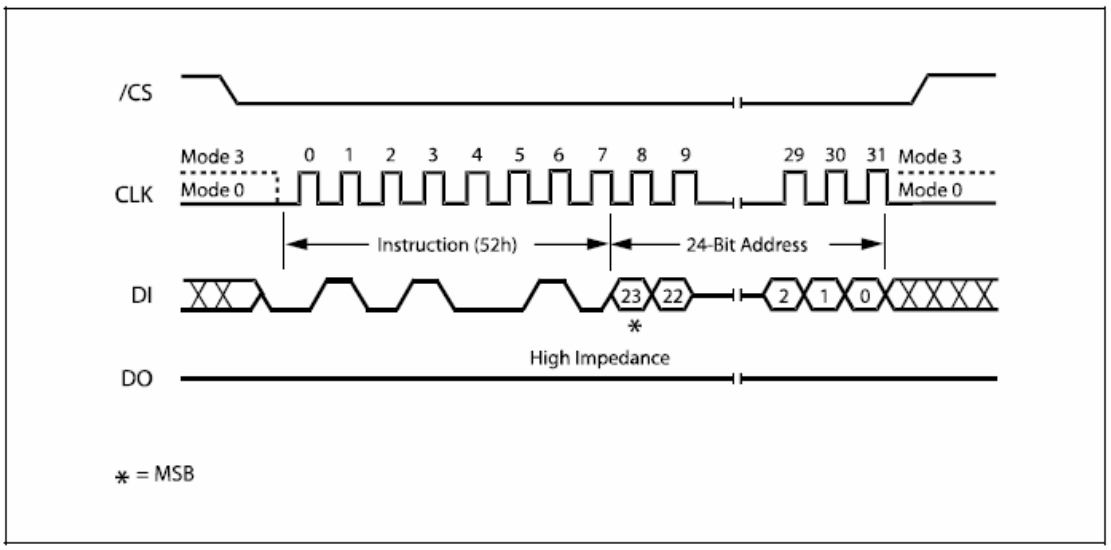



Figure 22. 32KB Block Erase Instruction Sequence Diagram

9.2.25，64KB 块擦除指令（D8H）

块擦除可以擦除 64K-byte 存储空间（全为 0XFF）。进行块擦写指令之前，必须进行写使能指令。该指令是以/CS 拉低开始的，然后在 DI 上传输指令代码 D8H 和 24 位地址。

时序图如图 23。

当最后字节的第 8 位进入芯片后，/CS 必须拉高。如果/CS 没有拉高，那么块擦写指令将不被执行。/CS 拉高后，块擦写指令的内建时间为 tBE。在块擦写指令执行期间，读状态寄存器指令仍然可以识别，以此来进行检查 BUSY 位。当块擦写指令执行期间，BUSY 位为了 1。当执行完后，BUSY 为 0，表明可以接受新的指令了。块擦写指令完成后 WEL 位自动清零。如果该指令要操作的任何一页已经被保护起来，那么该指令也将不执行。

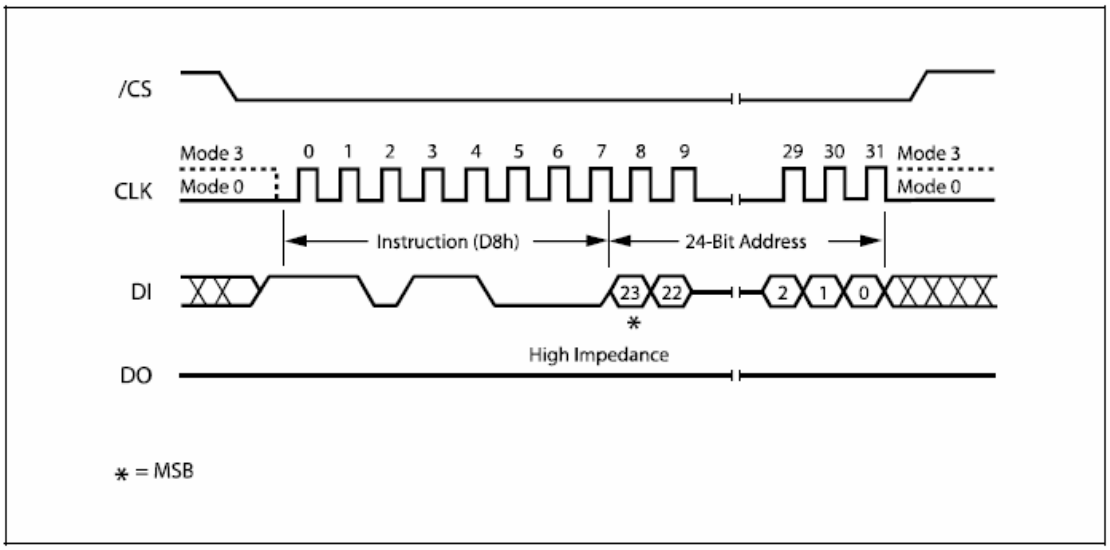



Figure 23. 64KB Block Erase Instruction Sequence Diagram

9.2.26，芯片擦除（C7/60H）

芯片擦除可以擦除 64K-byte 存储空间（全为 0XFF）。进行芯片擦写指令之前，必须进行写使能指令。该指令是以/CS 拉低开始的，然后在 DI 上传输指令代码 C7H 或者 60H。时序图如图 24。

当最后字节的第 8 位进入芯片后，/CS 必须拉高。如果/CS 没有拉高，那么芯片擦写指令将不被执行。/CS 拉高后，芯片擦写指令的内建时间为 tBE。在芯片擦写指令执行期间，读状态寄存器指令仍然可以识别，以此来进行检查 BUSY 位。当芯片擦写指令执行期间，BUSY 位为了 1。当执行完后，BUSY 为 0，表明可以接受新的指令了。芯片擦写指令完成后 WEL 位自动清零。如果该指令要操作的任何一页已经被保护起来，那么该指令也将不执行。

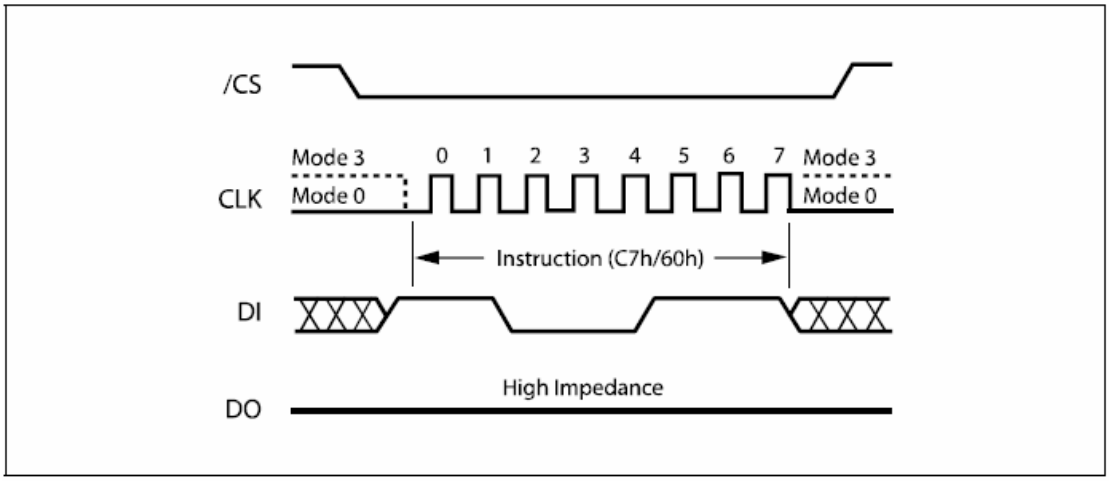



Figure 24. Chip Erase Instruction Sequence Diagram

9.2.27-------9.2.28　略

9.2.29，Power-Down 指令（B9H）

尽管在正常工作时，待机电流已经很小，但是利用 Power-Down 指令可以使得待机电流更小。对于利用电池供电的系统应用来说，Power-Down 指令会变得非常有用。该指令是以/CS 拉低开始，然后在 DI 上传输指令代码 B9H。

当最后字节的第 8 位进入芯片后,/CS 必须拉高。如果/CS 没有拉高,那么 Power-Down 指令将不被执行。/CS 拉高后，芯片进入到 Power-Down 状态的时间为 tDP。当芯片处在 Power-Down 状态时，只识别 Release from Power-Down/芯片 ID 指令，这两个指令可以使芯片恢复到正常状态。其他的指令都不识别，读寄存器指令也不识别。这是 Power-Down 状态下最大的写保护，只识别一条指令。芯片在正常工作状态下都是处于 Power-Up 状态，待机电流为 ICC1。

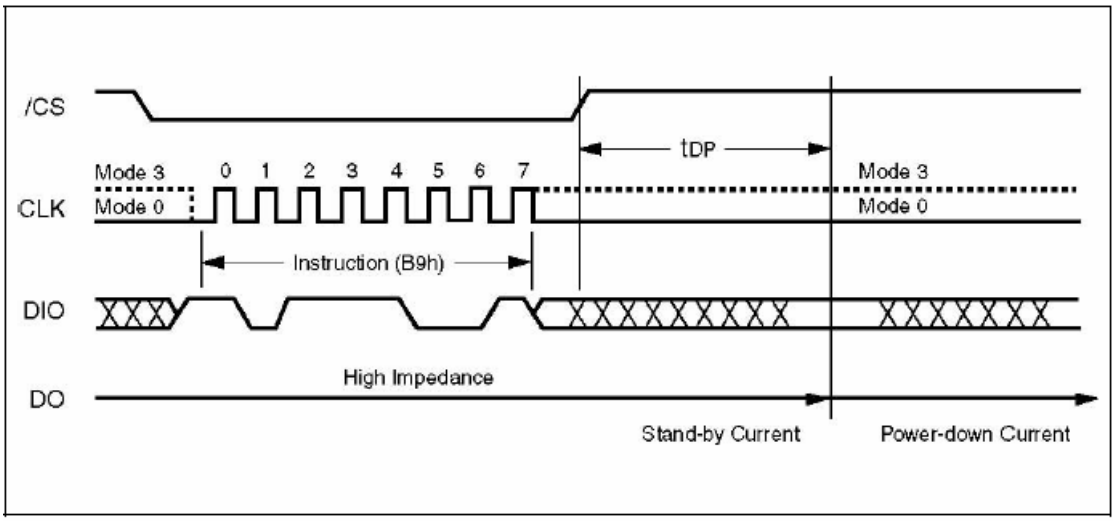



Figure 27. Deep Power-down Instruction Sequence Diagram

9.2.30，Release Power-Down / Device ID (ABH)

该指令是一个多功能的指令。可以用来把芯片从 Power-Down 状态下唤醒，也可以用来读取芯片的 ID 号。

为了把芯片从 Power-Down 状态下唤醒，该指令以/CS 拉低开始，然后在 DI 上传输指令代码 ABH，然后拉高/CS 。从 Power-Down 状态下唤醒到芯片恢复正常操作，其他的指令都识别需要的时间为 tRES1。在这段时间内/CS 必须还一直保持高电平。

当不在 Power-Down 状时，就用该指令来读取芯片 ID。以/CS 拉低开始，然后在 DI 传输指令代码 ABH，接着传 3 字节的无关位。芯片 ID 将会在时钟的下降沿以高位在前的方式在 DO 上输出。W25Q128BV 的芯片 ID 是由制造商和芯片 ID 组成。芯片 ID 可以连续读出来。该指令以/CS 拉高结束。

当该指令用来唤醒芯片和获取芯片 ID 时，跟前面描述的基本都是一样的。但是作为唤醒芯片时，/CS 拉高后其内建时间为 tRES2 ，内建完成后，芯片恢复到正常状态，其他的指令也将被识别。当 BUSY=1 时执行这条指令，将被嗯略，并且对当前正在执行的指令没有任何影响。

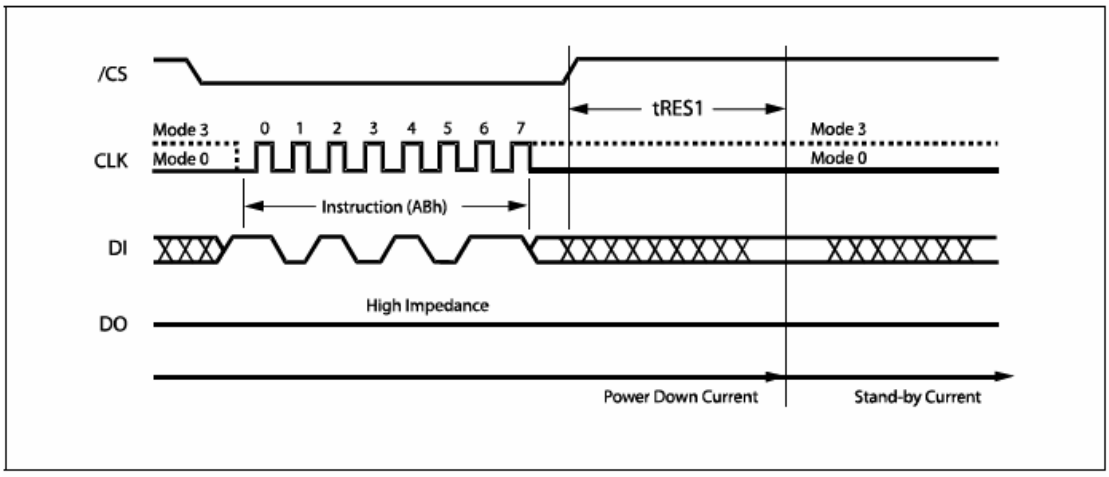



Figure 28a. Release Power-down Instruction Sequence

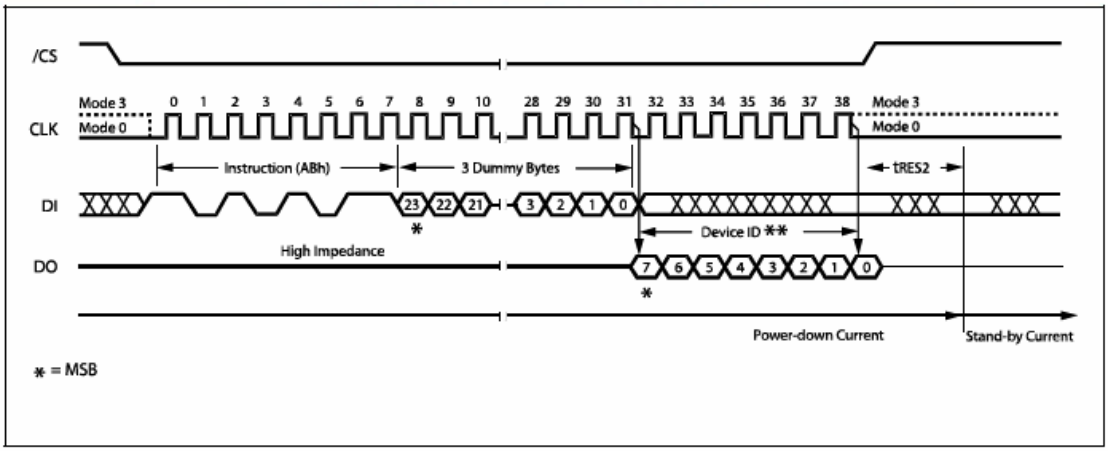



Figure 28b. Release Power-down / Device ID Instruction Sequence Diagram

9.2.31，读制造商和芯片 ID（90h）

该指令可以替代 Release from Power-Down/Device ID 指令，该指令读出的 JEDEC 签名的制造商 ID 和特殊的芯片 ID。

该指令与 Release from Power-Down/Device ID 指令相似。该指令以/CS 拉低开始，然后通过 DI 传输指令代码 90H 和 24 位的地址（全为 000000H）。这之后，WINBOND 的 ID（EFH）和芯片 ID 将在时钟的下降沿以高位在前的方式传出。关于 W25Q128BV 的芯片和制造商 ID，在图 29 中列出。如果 24 位地址传输的是 000001H，那么芯片 ID 将首先被传出，然后紧接着的是制造商 ID。这两个是连续读出来的。该指令以/CS 拉高结束。

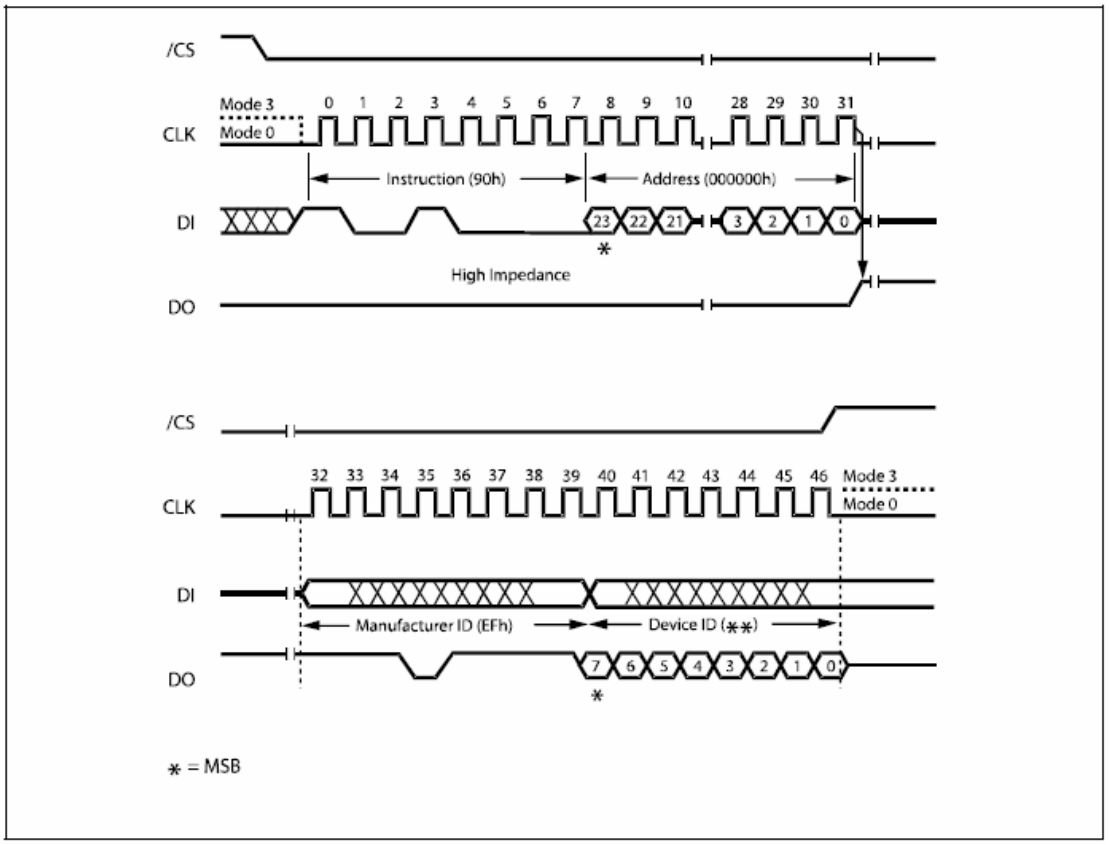



Figure 29. Read Manufacturer / Device ID Diagram

9.2.32，Read Manufacturer /Device ID Dual I/O(92h) (略)

9.2.33，Read Manufacturer /Device ID QUAD I/O(94h) (略)

9.2.34，读独立的 ID 序列（4BH）

该指令可以读取一个芯片已经固化的只读的 64 位 ID。每一片 W25Q128BV 的 ID 都是独立的。该 ID 序列与用户软件可以一起实现防止对系统的拷贝和克隆。该指令以/CS 拉低开始，然后传输指令代码（4BH）和 4 字节的无关位。这之后，64 位的独立 ID 在时在时钟的下降沿通过 DO 传出。

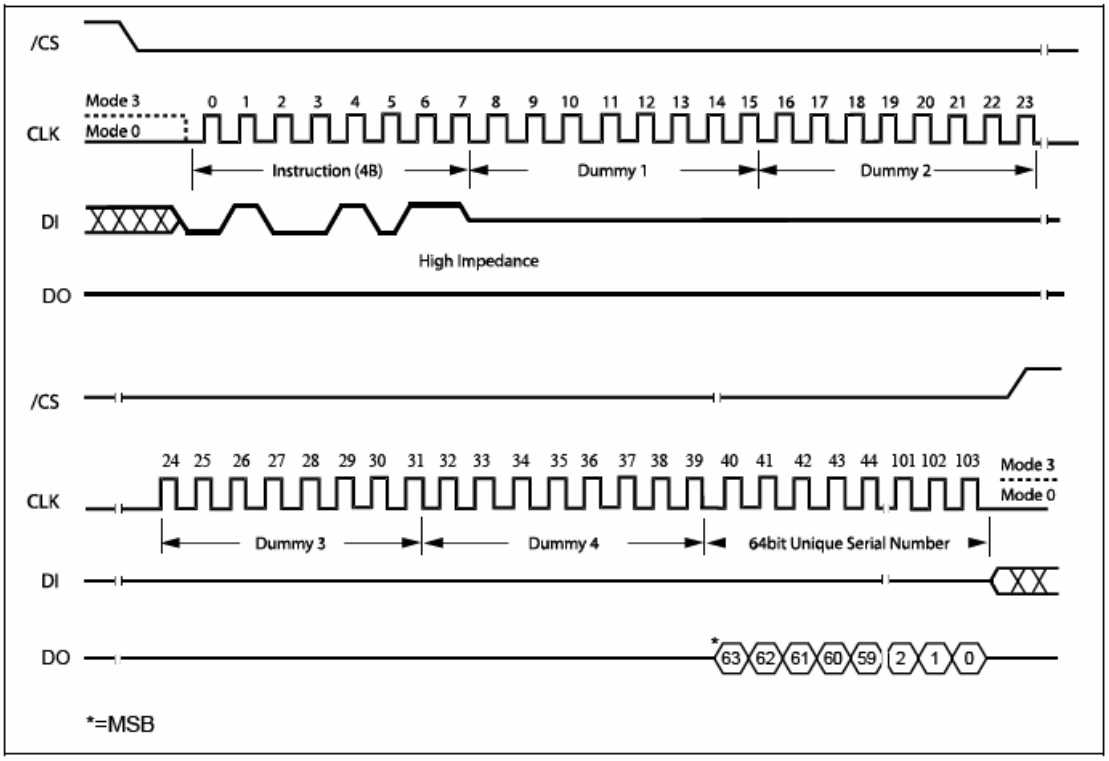



Figure 32. Read Unique ID Number Instruction Sequence

9.2.35，读 JEDEC ID（9FH）

考虑到兼容性的原因，W25Q128BV 提供了一系列的确定芯片 ID 的指令。该指令与 2003 通过的 SPI 串行存储设备的 JEDEC 标准是兼容的。该指令是以/CS 拉低开始，然后在传输指令代码 9FH，一个字节的 WINBOND 的 JEDEC 制造商签名和两字节的芯片 ID 字节（存储类型，容量）将在时钟的下降沿以商位在前的方式在 DO 上传输出来。

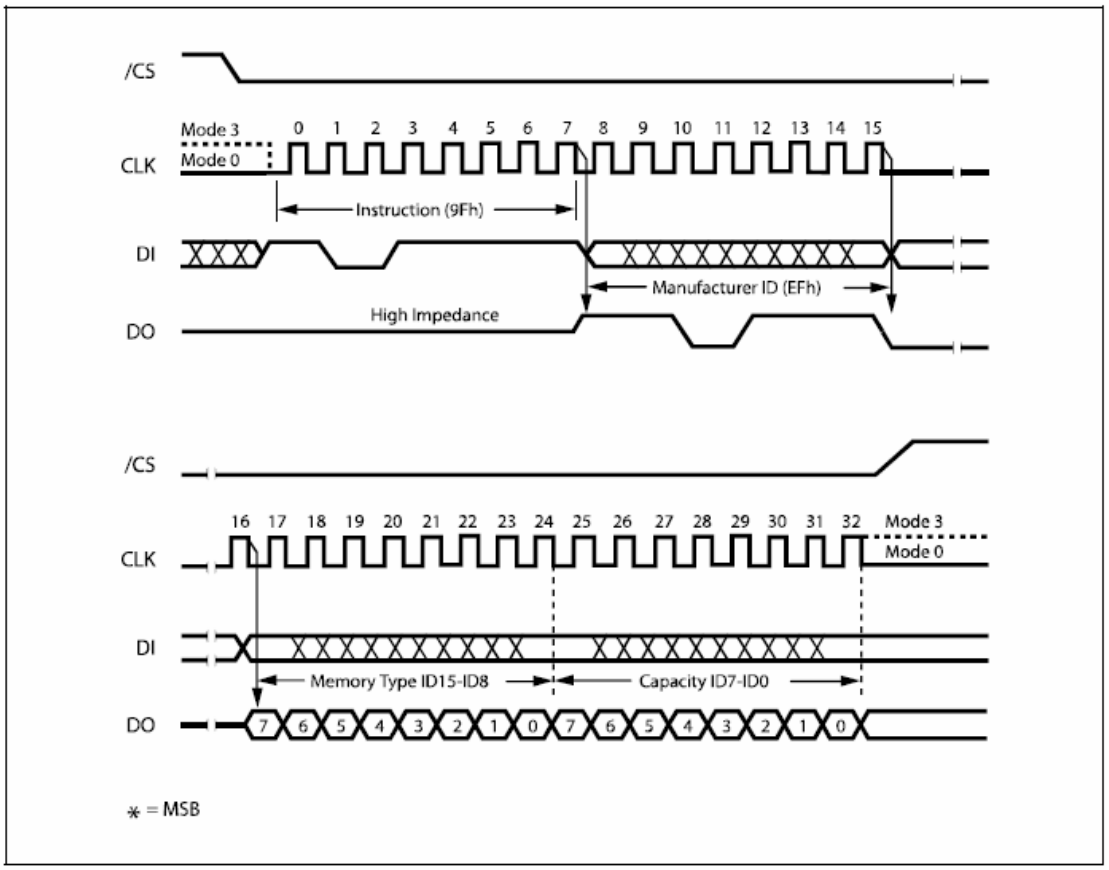



Figure 33. Read JEDEC ID Instruction Sequence

9.2.36，擦除安全寄存器（44H）

W25Q128BV 提供了 4 个 256 字节的安全寄存器，每个都可以单独的进行擦写。这些寄存器可以用来存储的一些安全和重要的信息。

擦除安全寄存器指令与扇区擦写指令是相似的。擦除安全寄存器指令执行前必须先执行写使能指令（WEL 位为 1）。该指令以/CS 拉低开始，然后传输指令代码（44H）和 24 位地址，将会擦除其中的一个安全寄存器。

| ADDRESS | A23-16 | A15-12 | A11-8 | A7-0 |
|---|---|---|---|---|
| Security Register #0* | 00h | 0 0 0 0 | 0 0 0 0 | Don't Care |
| Security Register #1 | 00h | 0 0 0 1 | 0 0 0 0 | Don't Care |
| Security Register #2 | 00h | 0 0 1 0 | 0 0 0 0 | Don't Care |
| Security Register #3 | 00h | 0 0 1 1 | 0 0 0 0 | Don't Care |

注意：安全寄存器 0 被华邦公保留了，所以在使用安全寄存器 0 之前，先使用其他的。

擦除安全寄存器指令时序如图 34 所示。当最后要传输的一个字节的第 8 位锁定后，/CS 必须拉高。如果没有这样，该指令将不会执行。当/CS 拉高后，该指令的内建时间为 tSE. 当该指令在执行期间，读取状态寄存器指令还是可以执行的，这样可以去检查 BUSY 位。当该指令执行时，BUSY 位为 1 ，执行完后为 0，为 0 后就芯片可以接受其他的指令，此时 WEL 位也被自动清零。在状态寄存器 2 中的安全寄存器锁定位可以一次编程来保护相应的安全寄存器。一旦锁定位为 1，相应的安全寄存将永久的锁定，擦除安全寄存器指令

将无效。

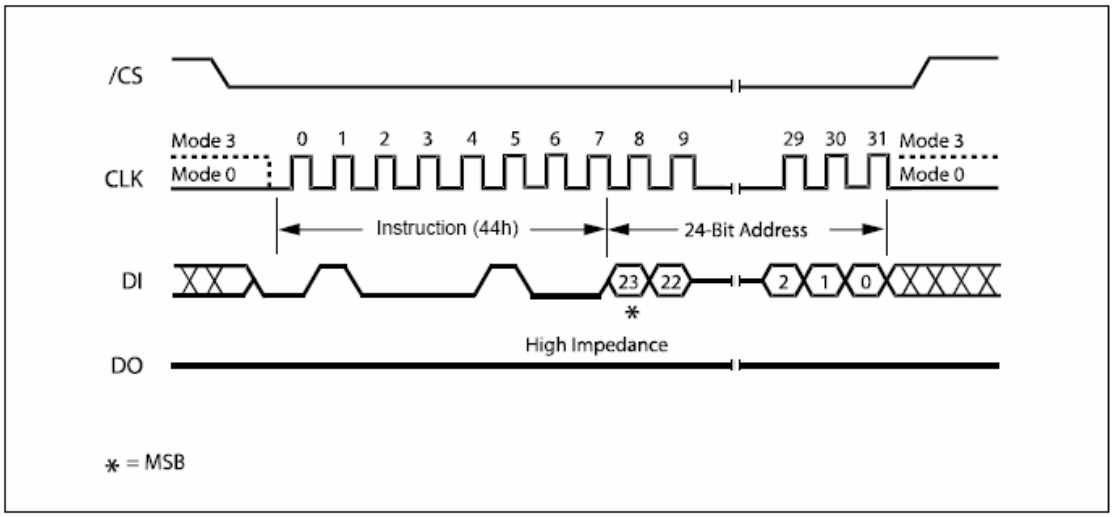



Figure 34. Erase Security Registers Instruction Sequence

9.2.37，写安全寄存器指令（42H）

写安全寄存器指令与页写指令相似。在相应的位是擦除状态时，可以一次写入 1 到 256 字节。擦除安全寄存器指令执行前必须先执行写使能指令（WEL 位为 1）。该指令以/CS 拉低开始，然后传输指令代码（42H），24 位地址和至少一个字节的数据。在整个数据传输期间/CS 必须一直保持低电平。

该指令的时序图如图 35。在状态寄存器 2 中的安全寄存器锁定位如果被编程，那么相应的安全寄存器将永远被锁定，写安全寄存器指令将无效。

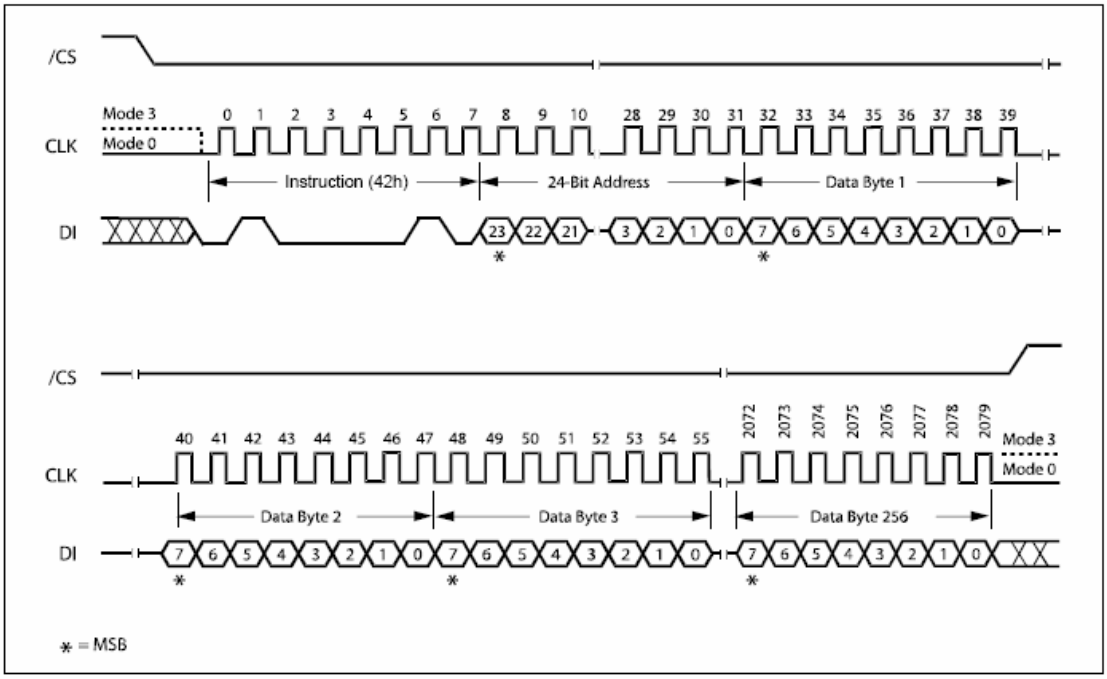



Figure 35. Program Security Registers Instruction Sequence

9.2.38，读安全寄存器指令（48H）

该指令与快速读指令相似，允许一到多个字节连续从其中一个寄存器读出。该指令以/CS 拉低开始，然后在 DI 上传输指令代码 48h，24 位地址和 1 个字节的无效数据。指令代码和地址是在时钟的上升沿锁存的，相应地址处的数据是时钟的下降沿以高位在前的方式在 DO 上传输的。当一个字节的数据传输后，地址自动加 1。当地址已经是 0XFF 时，自动加 1 后变为了 0。该指令以相应/CS 拉高结束。该指令的时序图如图 36。当 BUSY 位为 1 时执行该指令，将被忽略，并且对当前正在执行的指令没有任何影响。读安全寄存器指令允许时钟频率从 D.C 到最大的 FR。

| ADDRESS | A23-16 | A15-12 | A11-8 | A7-0 |
|---|---|---|---|---|
| Security Register #0* | 00h | 0 0 0 0 | 0 0 0 0 | Don't Care |
| Security Register #1 | 00h | 0 0 0 1 | 0 0 0 0 | Don't Care |
| Security Register #2 | 00h | 0 0 1 0 | 0 0 0 0 | Don't Care |
| Security Register #3 | 00h | 0 0 1 1 | 0 0 0 0 | Don't Care |

注意：安全寄存器 0 被华邦公保留了，所以在使用安全寄存器 0 之前，先使用其他的。

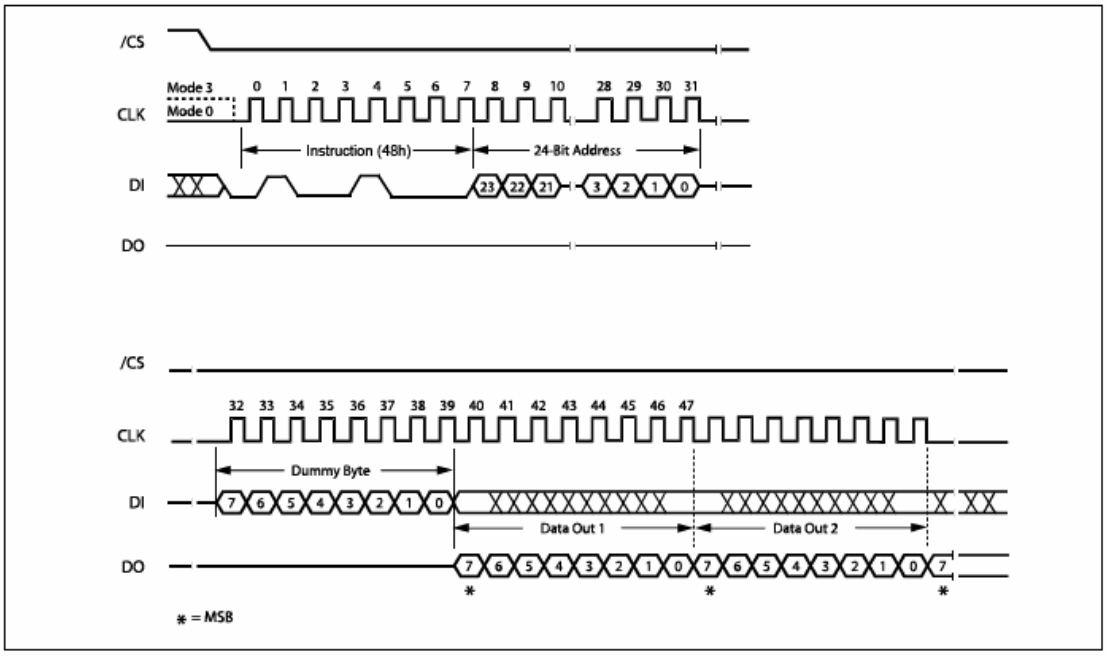



Figure 36. Read Security Registers Instruction Sequence

附：几种不同的封装！

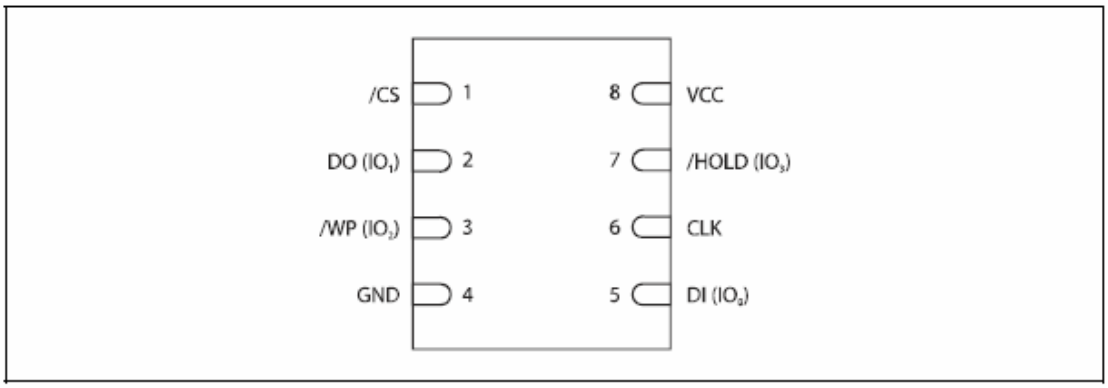



Figure 1b. W25Q128BV Pad Assignments, 8-pad WSON 8x6-mm (Package Code E)

| PIN NO. | PIN NAME | I/O | FUNCTION |
|---|---|---|---|
| 1 | /CS | I | Chip Select Input |
| 2 | DO (IO1) | I/O | Data Output (Data Input Output 1)*1 |
| 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 |
| 6 | CLK | I | Serial Clock Input |
| 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

*2 IO0 – IO3 are used for Quad SPI instructions

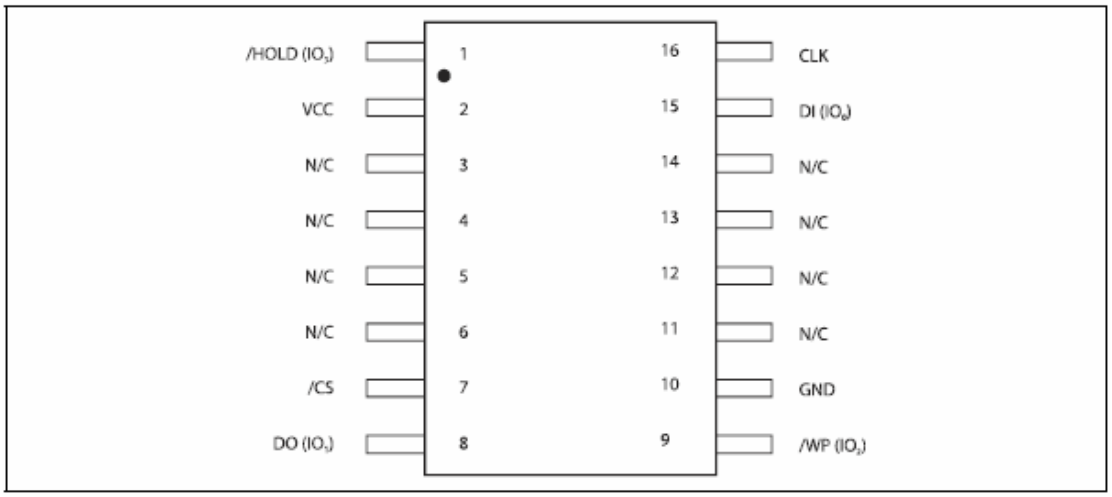



Figure 1c. W25Q128BV Pin Assignments, 16-pin SOIC 300-mil (Package Code F)

| PAD NO. | PAD NAME | I/O | FUNCTION |
|---|---|---|---|
| 1 | /HOLD (IO3) | I/O | Hold Input (Data Input Output 3)*2 |
| 2 | VCC | | Power Supply |
| 3 | N/C | | No Connect |
| 4 | N/C | | No Connect |
| 5 | N/C | | No Connect |
| 6 | N/C | | No Connect |
| 7 | /CS | I | Chip Select Input |
| 8 | DO (IO1) | I/O | Data Output (Data Input Output 1)*1 |
| 9 | /WP (IO2) | I/O | Write Protect Input (Data Input Output 2)*2 |
| 10 | GND | | Ground |
| 11 | N/C | | No Connect |
| 12 | N/C | | No Connect |
| 13 | N/C | | No Connect |
| 14 | N/C | | No Connect |
| 15 | DI (IO0) | I/O | Data Input (Data Input Output 0)*1 |
| 16 | CLK | I | Serial Clock Input |

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

*2 IO0 – IO3 are used for Quad SPI instructions