Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!


## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

# Micron M25PE40 Serial Flash Memory

## Serial Flash Memory with Byte Alterability, 75 MHz SPI bus, Standard Pinout

## Features

- 4Mb of page-erasable Flash memory
- 2.7V to 3.6V single supply voltage
- SPI bus-compatible serial interface
- 75 MHz clock rate (maximum)
- Page size: 256 bytes
  - Page write in 11ms (TYP)
  - Page program in 0.8ms (TYP)
  - Page erase in 10ms (TYP)
- Subsector erase: 4KB
  - Sector erase: 64KB
  - Bulk erase: 4Mb
- Deep power-down mode: 1µA (TYP)
- Electronic signature
  - JEDEC standard 2-byte signature (8013h)
- Software write-protection on a 64KB sector basis
- Hardware write protection of the memory area selected using the BP0, BP1, and BP2 bits
- More than 100,000 write cycles
- More than 20 years of data retention
- Packages (RoHS compliant)
  - VFQFPN8 (MP) 6mm x 5mm (MLP8)
  - SO8W (MW) 208 mils
  - SO8N (MN) 150 mils
- Automotive grade parts available

# Contents

# List of Figures

# List of Tables

# Functional Description

The M25PE40 is a 4Mb (512Kb x 8 bit) serial-paged Flash memory device accessed by a high-speed SPI-compatible bus.

The memory can be written or programmed 1 to 256 bytes at a time using the PAGE WRITE or PAGE PROGRAM command. The PAGE WRITE command consists of an integrated PAGE ERASE cycle followed by a PAGE PROGRAM cycle.

The memory is organized as 8 sectors, divided into 16 subsectors each (128 subsectors total). Each sector contains 256 pages and each subsector contains 16 pages. Each page is 256 bytes wide. The entire memory can be viewed as consisting of 2048 pages, or 524,288 bytes.

The memory can be erased one page at a time using the PAGE ERASE command, one sector at a time using the SECTOR ERASE command, one subsector at a time using the SUBSECTOR ERASE command, or as a whoe using the BULK ERASE command.

The memory can be write-protected by either hardware or software using a mix of volatile and non-volatile protection features, depending on application needs. The protection granularity is 64Kb (sector granularity). The entire memory array is partitioned into 4KB subsectors.

**Figure 1: Logic Diagram**

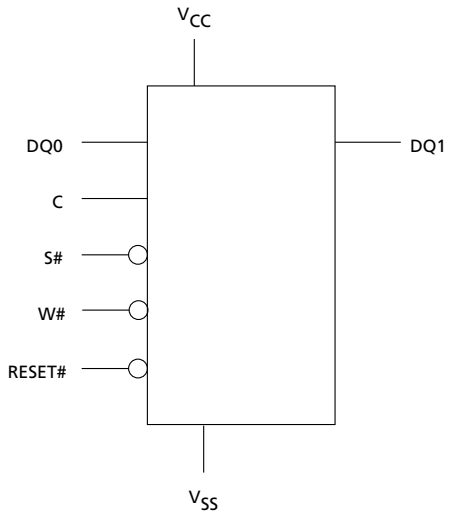
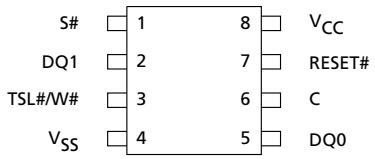


**Table 1: Signal Names**

| Signal Name | Function | Direction |
|---|---|---|
| C | Serial clock | Input |
| DQ0 | Serial data | Input |
| DQ1 | Serial data | Output |
| S# | Chip select | Input |
| W# | Write Protect | Input |
| RESET# | Reset | Input |
| $V_{CC}$ | Supply voltage | – |
| $V_{SS}$ | Ground | – |

**Figure 2: Pin Connections: VFQFPN and SO**




There is an exposed central pad on the underside of the VFQFPN package that is pulled internally to $V_{SS}$ and must not be connected to any other voltage or signal line on the PCB. The Package Mechanical section provides information on package dimensions and how to identify pin 1.

# Signal Descriptions

**Table 2: Signal Descriptions**

| Signal | Type | Description |
|---|---|---|
| DQ1 | Output | **Serial data:** The DQ1 output signal is used to transfer data serially out of the device. Data is shifted out on the falling edge of the serial clock (C). |
| DQ0 | Input | **Serial data:** The DQ0 input signal is used to transfer data serially into the device. It receives commands, addresses, and the data to be programmed. Values are latched on the rising edge of the serial clock (C). |
| C | Input | **Clock:** The C input signal provides the timing of the serial interface. Commands, addresses, or data present at serial data input (DQ0) is latched on the rising edge of the serial clock (C). Data on DQ1 changes after the falling edge of C. |
| S# | Input | **Chip select:** When the S# input signal is HIGH, the device is deselected and DQ1 is at HIGH impedance. Unless an internal READ, PROGRAM, ERASE, or WRITE cycle is in progress, the device will be in the standby power mode (not the DEEP POWER-DOWN mode). Driving S# LOW enables the device, placing it in the active power mode. After power-up, a falling edge on S# is required prior to the start of any command. |
| RESET# | Input | **Reset:** The RESET# input provides a hardware reset for the memory.<br>When RESET# is driven HIGH, the memory is in the normal operating mode. When RESET# is driven LOW, the memory will enter the Reset mode. In this mode, the output is at HIGH impedance.<br>Driving RESET# LOW while an internal operation is in progress affects the WRITE, PROGRAM, or ERASE cycle, and data may be lost. |
| W# | Input | **Write protect:** The W# input signal is used to freeze the size of the area of memory that is protected against WRITE, PROGRAM, and ERASE commands as specified by the values in the block protect bits in the status register. |
| $V_{CC}$ | Input | **Supply voltage** |
| $V_{SS}$ | Input | **Ground:** Reference for the VCC supply voltage. |

## SPI Modes

These devices can be driven by a microcontroller with its serial peripheral interface (SPI) running in either of the following two SPI modes:

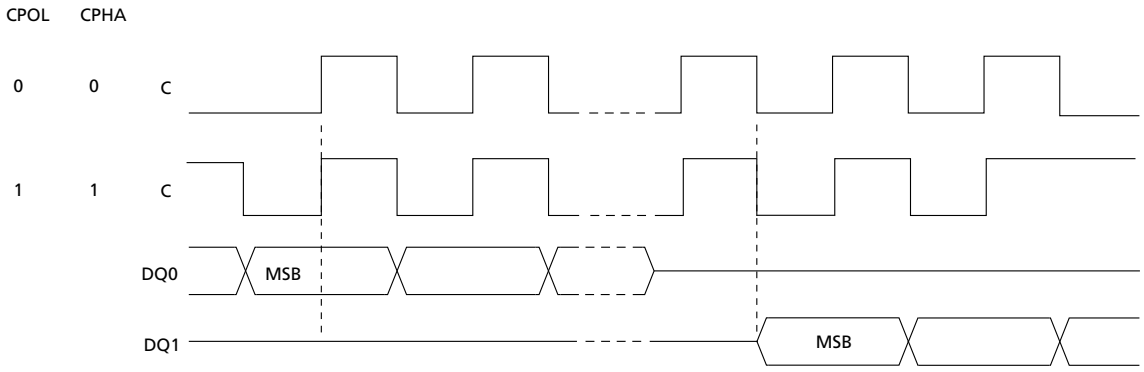
- CPOL=0, CPHA=0
- CPOL=1, CPHA=1

For these two modes, input data is latched in on the rising edge of serial clock (C), and output data is available from the falling edge of C.

The difference between the two modes is the clock polarity when the bus master is in STANDBY mode and not transferring data:

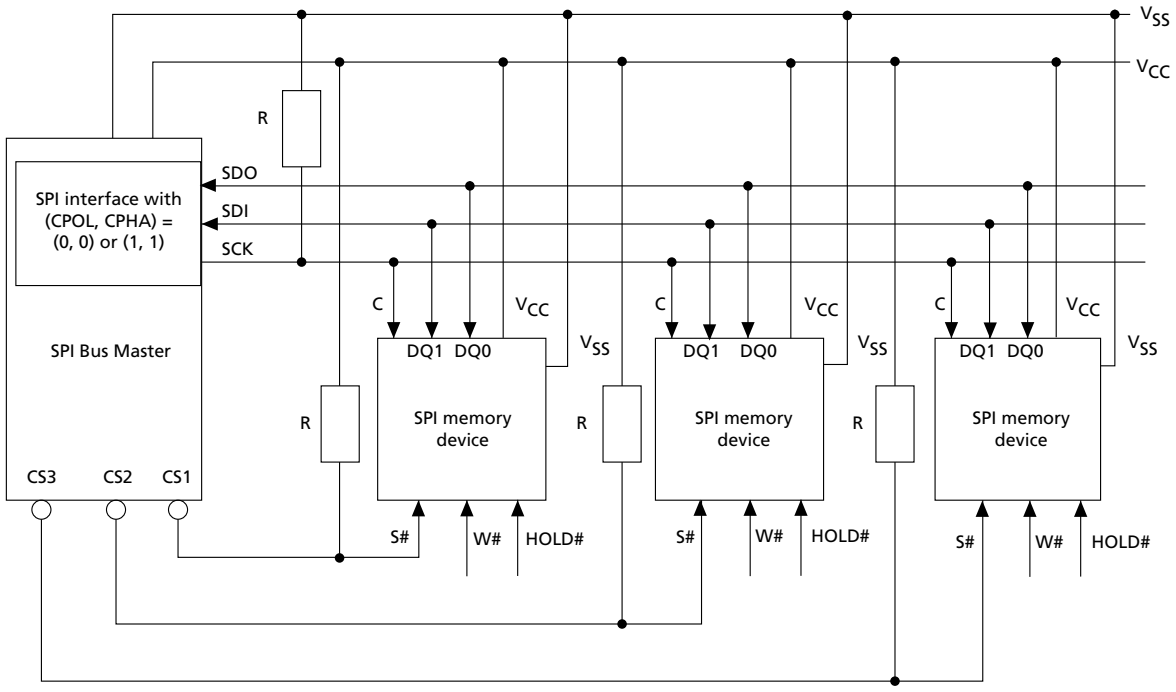
- C remains at 0 for (CPOL=0, CPHA=0)
- C remains at 1 for (CPOL=1, CPHA=1)

**Figure 3: SPI Modes Supported**




Because only one device is selected at a time, only one device drives the serial data output (DQ1) line at a time, while the other devices are HIGH-Z. An example of three devices connected to an MCU on an SPI bus is shown here.

**Figure 4: Bus Master and Memory Devices on the SPI Bus**




Notes:
1. WRITE PROTECT (W#) and HOLD# should be driven HIGH or LOW as appropriate.
2. Resistors (R) ensure that the memory device is not selected if the bus master leaves the S# line HIGH-Z.
3. The bus master may enter a state where all I/O are HIGH-Z at the same time; for example, when the bus master is reset. Therefore, the C must be connected to an external pull-down resistor so that when all I/O are HIGH-Z, S# is pulled HIGH while C is pulled LOW. This ensures that S# and C do not go HIGH at the same time and that the $^{t}SHCH$ requirement is met.
4. The typical value of R is 100 kΩ, assuming that the time constant R × $C_p$ ($C_p$ = parasitic capacitance of the bus line) is shorter than the time during which the bus master leaves the SPI bus HIGH-Z.
5. Example: Given that $C_p$ = 50 pF (R × $C_p$ = 5µs), the application must ensure that the bus master never leaves the SPI bus HIGH-Z for a time period shorter than 5µs.

# Operating Features

## Sharing the overhead of modifying data

To write or program one or more data bytes, two commands are required: WRITE ENABLE (WREN), which is one byte, and a PAGE WRITE (PW) or PAGE PROGRAM (PP) sequence, which consists of four bytes plus data. This is followed by the internal cycle of duration $t_{PW}$ or $t_{PP}$.

To share this overhead, the PW or PP command allows up to 256 bytes to be programmed (changing bits from 1 to 0) or written (changing bits to 0 or 1) at a time, provided that they lie in consecutive addresses on the same page of memory.

## An easy way to modify data

The Page Write (PW) instruction provides a convenient way of modifying data (up to 256 contiguous bytes at a time), and simply requires the start address, and the new data in the instruction sequence.

The Page Write (PW) instruction is entered by driving Chip Select (S#) LOW, and then transmitting the instruction byte, three address bytes (A23-A0) and at least one data byte, and then driving S# HIGH. While S# is being held LOW, the data bytes are written to the data buffer, starting at the address given in the third address byte (A7-A0). When Chip S# is driven HIGH, the Write cycle starts. The remaining unchanged bytes of the data buffer are automatically loaded with the values of the corresponding bytes of the addressed memory page. The addressed memory page is then automatically put into an Erase cycle. Finally, the addressed memory page is programmed with the contents of the data buffer.

All of this buffer management is handled internally, and is transparent to the user. The user is given the facility of being able to alter the contents of the memory on a byte-by-byte basis. For optimized timings, it is recommended to use the PAGE WRITE (PW) instruction to write all consecutive targeted bytes in a single sequence versus using several PAGE WRITE (PW) sequences with each containing only a few bytes.

## A fast way to modify data

The PAGE PROGRAM (PP) command providesa fast way of modifying data (up to 256 contiguous bytes at a time), provided that it only involves resetting bits to 0 that had previously been set to 1.

This might be:

- When the designer is programming the device for the first time.
- When the designer knows that the page has already been erased by an earlier PAGE ERASE (PE), SUBSECTOR ERASE (SSE), SECTOR ERASE (SE), or BULK ERASE (BE) command. This is useful, for example, when storing a fast stream of data, having first performed the erase cycle when time was available.
- When the designer knows that the only changes involve resetting bits to 0 that are still set to 1. When this method is possible, it has the additional advantage of minimizing the number of unnecessary erase operations, and the extra stress incurred by each page.

For optimized timings, it is recommended to use the PAGE PROGRAM (PP) instruction to program all consecutive targeted bytes in a single sequence versus using several PAGE PROGRAM (PP) sequences with each containing only a few bytes.

## Polling during a Write, Program, or Erase Cycle

An improvement in the time to complete the following commands can be achieved by not waiting for the worst case delay ($t_{PW}$, $t_{PP}$, $t_{PE}$, $t_{BE}$, $t_{W}$or $t_{SE}$).

The write in progress (WIP) bit is provided in the status register so that the application program can monitor this bit in the status register, polling it to establish when the previous WRITE cycle, PROGRAM cycle, or ERASE cycle is complete.

## Reset

An internal power-on reset circuit helps protect against inadvertent data writes. Additional protection is provided by driving RESET# LOW during the power-on process, and only driving it HIGH when $V_{CC}$ has reached the correct voltage level, $V_{CC}$(min).

## Active Power, Standby Power, and Deep Power-Down

When chip select (S#) is LOW, the device is selected and in the ACTIVE POWER mode.

When S# is HIGH, the device is deselected, but could remain in the ACTIVE POWER mode until all internal cycles have completed (PROGRAM, ERASE, WRITE). The device then goes in to the STANDBY POWER mode. The device consumption drops to $I_{CC1}$.

The DEEP POWER-DOWN mode is entered when the DEEP POWER-DOWN command is executed. The device consumption drops further to $I_{CC2}$. The device remains in this mode until the RELEASE FROM DEEP POWER-DOWN command is executed. While in the DEEP POWER-DOWN mode, the device ignores all WRITE, PROGRAM, and ERASE commands. This provides an extra software protection mechanism when the device is not in active use, by protecting the device from inadvertent WRITE, PROGRAM, or ERASE operations.

## Status Register

The status register contains a number of status bits that can be read by the READ STATUS REGISTER (RDSR) command.

## Protection Modes

Non-volatile memory is used in environments that can include excessive noise. The following capabilities help protect data in these noisy environments.

Power on reset and an internal timer ($t_{PUW}$) can provide protection against inadvertent changes while the power supply is outside the operating specification.

WRITE, PROGRAM, and ERASE commands are checked before they are accepted for execution to ensure they consist of a number of clock pulses that is a multiple of eight.

All commands that modify data must be preceded by a WRITE ENABLE command to set the write enable latch (WEL) bit. This bit is returned to its reset state by the following events.

- Power-up
- Reset (RESET#) driven LOW

- WRITE DISABLE (WRDI) command completion
- PAGE WRITE (PW) command completion
- PAGE PROGRAM (PP) command completion
- WRITE TO LOCK REGISTER (WRLR) command completion
- PAGE ERASE (PE) command completion
- SUBSECTOR ERASE (SSE) command completion
- SECTOR EASE (SE) command completion
- BULK ERASE (BE) command completion

The Reset (RESET#) signal can be driven LOW to freeze and reset the internal logic.

In addition to the low power consumption feature, DEEP POWER-DOWN mode offers extra software protection from inadvertant WRITE, PROGRAM, and ERASE commands while the device is not in active use.

## Specific Hardware and Software Protection

The device features a hardware protected mode (HPM) and two software protected modes (SPM1 and SPM2). SPM1 and SPM2 can be combined to protect the memory array as required.

**Hardware Protected Mode (HPM)**

The Hardware Protected Mode (HPM) is used to write-protect the non-volatile bits of the Status Register (that is, the Block Protect Bits and the Status Register bit). HPM is entered by driving the Write Protect (W#) signal LOW with the SRWD bit set to HIGH. This additional protection allows the Status Register to be hardware-protected.

**SPM1 and SPM2**

The first Software Protected Mode (SPM1) is managed by specific Lock Registers assigned to each 64KB sector.

The Lock Registers can be read and written using the Read Lock Register (RDLR) and Write to Lock Register (WRLR) commands.

In each Lock Register, two bits control the protection of each sector: the Write Lock bit and the Lock Down bit.

- Write lock bit: This bit determines whether the contents of the sector can be modified using the WRITE, PROGRAM, and ERASE commands. When the bit is set to ‘1’, the sector is write protected, and any operations that attempt to change the data in the sector will fail. When the bit is reset to ‘0’, the sector is not write protected by the lock register, and may be modified.
- Lock down bit: This bit provides a mechanism for protecting software data from simple hacking and malicious attack. When the bit is set to '1’, further modification to the write lock bit and lock down bit cannot be performed. A power-up, is required before changes to these bits can be made. When the bit is reset to ‘0’, the write lock bit and lock down bit can be changed.

The Write Lock bit and the Lock Down bit are volatile and their value is reset to 0 after a power-down or reset.

The software protection truth table shows the lock down bit and write lock bit settings and the sector protection status.

**Table 3: Software Protection Truth Table, 64KB granularity (sectors 0-7)**

| Sector Lock Register: Lock Down Bit | Sector Lock Register: Write Lock Bit | Protection Status |
|---|---|---|
| 0 | 0 | Sector unprotected from PROGRAM / ERASE / WRITE operations; protection status reversible |
| 0 | 1 | Sector protected from PROGRAM / ERASE / WRITE operations; protection status reversible |
| 1 | 0 | Sector unprotected from PROGRAM / ERASE / WRITE operations; protection status cannot be changed except by a reset or power-up. |
| 1 | 1 | Sector protected from PROGRAM / ERASE / WRITE operations; protection status cannot be changed except by a reset or power-up. |

The second Software Protected Mode (SPM2) uses the block protect (BP2, BP1, BP0) bits to allow part of the memory to be configured as read-only.

**Table 4: Protected Area Sizes**

| Status Register Content | | | Memory Content | | |
|---|---|---|---|---|---|
| BP Bit 2 | BP Bit 1 | BP Bit 0 | Protected Area | Unprotected Area | Notes |
| 0 | 0 | 0 | none | All sectors (sectors 0 to 7) | 1 |
| 0 | 0 | 1 | Upper eighth (sector 7) | Lower 7/8ths (sectors 0 to 6) | |
| 0 | 1 | 0 | Upper quarter (sectors 6 and 7) | Lower three-quarters (sectors 0 to 5) | |
| 0 | 1 | 1 | Upper half (sectors 4 to 7) | Lower half (sectors 0 to 3) | |
| 1 | 0 | 0 | All sectors (sectors 0 to 7) | none | |
| 1 | 0 | 1 | All sectors (sectors 0 to 7) | none | |
| 1 | 1 | 0 | All sectors (sectors 0 to 7) | none | |
| 1 | 1 | 1 | All sectors (sectors 0 to 7) | none | |

Note: 1. The device is ready to accept a BULK ERASE command only if all block protect bits (BP2, BP1, BP0) are 0.

# Memory Organization

The M25PE40 memory is organized as follows:

- 2048 pages (256 bytes each)
- 524,288 bytes (8 bits each)
- 128 subsectors (32Kb, 4096 bytes each)
- 8 sectors (512Kb, 65,536 bytes each)

Each page can be individually:

- programmed (bits are programmed from 1 to 0)
- erased (bits are erased from 0 to 1)
- written (bits are changed to either 0 or 1)

The device is page- or sector-erasable (bits are erased from 0 to 1).

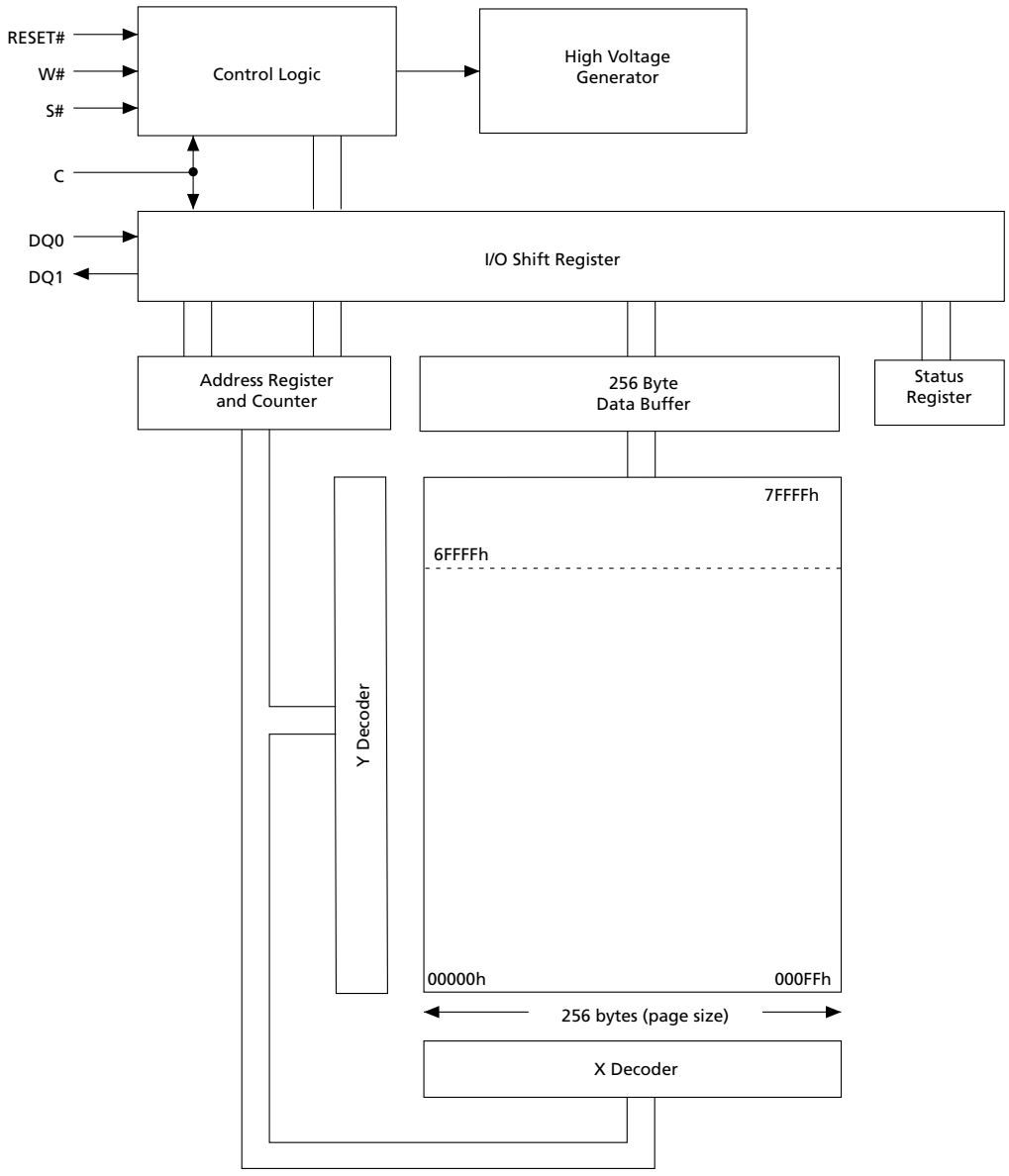
**Table 5: Memory Organization**

| Sector | Subsector | Address Range | |
|---|---|---|---|
| | | Start | End |
| 7 | 127 | 7F000h | 7FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 112 | 70000h | 70FFFh |
| 6 | 111 | 6F00h | 6FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 96 | 60000h | 60FFFh |
| 5 | 95 | 5F000h | 5FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 80 | 50000h | 50FFFh |
| 4 | 79 | 4F000h | 4FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 64 | 40000h | 40FFFh |
| 3 | 63 | 3F000h | 3FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 48 | 30000h | 30FFFh |
| 2 | 47 | 2F000h | 2FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 32 | 20000h | 20FFFh |
| 1 | 31 | 1F000h | 1FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 16 | 10000h | 10FFFh |

**Table 5: Memory Organization (Continued)**

| Sector | Subsector | Address Range | |
|---|---|---|---|
| | | Start | End |
| 0 | 15 | 0F000h | 0FFFFh |
| | ⋮ | ⋮ | ⋮ |
| | 4 | 04000h | 04FFFh |
| | 3 | 03000h | 03FFFh |
| | 2 | 02000h | 02FFFh |
| | 1 | 01000h | 01FFFh |
| | 0 | 00000h | 00FFFh |

**Figure 5: Block Diagram**




Note: 1. Entire memory array can be made read-only on a 64KB basis through the lock registers.

# Command Set Overview

All commands, addresses, and data are shifted in and out of the device, most significant bit first.

Serial data input (DQ1) is sampled on the first rising edge of serial clock (C) after chip select (S#) is driven LOW. Then, the one-byte command code must be shifted in to the device, most significant bit first, on DQ1, each bit being latched on the rising edges of C.

Every command sequence starts with a one-byte command code. Depending on the command, this command code might be followed by address or data bytes, by address and data bytes, or by neither address or data bytes. For the following commands, the shifted-in command sequence is followed by a data-out sequence. S# can be driven HIGH after any bit of the data-out sequence is being shifted out.

- READ DATA BYTES (READ)
- READ DATA BYTES at HIGHER SPEED
- READ STATUS REGISTER
- READ TO LOCK REGISTER

For the following commands, S# must be driven HIGH exactly at a byte boundary. That is, after an exact multiple of eight clock pulses following S# being driven LOW, S# must be driven HIGH. Otherwise, the command is rejected and not executed.

- PAGE WRITE
- PAGE PROGRAM
- PAGE ERASE
- SUBSECTOR ERASE
- SECTOR ERASE
- BULK ERASE
- WRITE ENABLE
- WRITE DISABLE
- WRITE STATUS REGISTER
- WRITE TO LOCK REGISTER
- DEEP POWER-DOWN
- RELEASE FROM DEEP POWER-DOWN

All attempts to access the memory array are ignored during a WRITE cycle, a PROGRAM cycle, or an ERASE cycle. In addition, the internal cycle for each of these commands continues unaffected.

**Table 6: Command Set Codes**

| Command Name | One-Byte Command Code | | Bytes | | |
|---|---|---|---|---|---|
| | | | Address | Dummy | Data |
| WRITE ENABLE | 0000 0110 | 06h | 0 | 0 | 0 |
| WRITE DISABLE | 0000 0100 | 04h | 0 | 0 | 0 |
| READ IDENTIFICATION | 1001 1111 | 9Fh | 0 | 0 | 1 to 3 |
| READ STATUS REGISTER | 0000 0101 | 05h | 0 | 0 | 1 to ∞ |
| WRITE STATUS REGISTER | 0000 0001 | 01h | 0 | 0 | 1 |
| WRITE to LOCK REGISTER | 1110 0101 | E5h | 3 | 0 | 1 |
| READ LOCK REGISTER | 1110 1000 | E8h | 3 | 0 | 1 |
| READ DATA BYTES | 0000 0011 | 03h | 3 | 0 | 1 to ∞ |
| READ DATA BYTES at HIGHER SPEED | 0000 1011 | 0Bh | 3 | 1 | 1 to ∞ |
| PAGE WRITE | 0000 1010 | 0Ah | 3 | 0 | 1 to 256 |
| PAGE PROGRAM | 0000 0010 | 02h | 3 | 0 | 1 to 256 |
| PAGE ERASE | 1101 1011 | DBh | 3 | 0 | 0 |
| SUBSECTOR ERASE | 0010 0000 | 20h | 3 | 0 | 0 |
| SECTOR ERASE | 1101 1000 | D8h | 3 | 0 | 0 |
| BULK ERASE | 1100 0111 | C7h | 0 | 0 | 0 |
| DEEP POWER-DOWN | 1011 1001 | B9h | 0 | 0 | 0 |
| RELEASE from DEEP POWER-DOWN | 1010 1011 | ABh | 0 | 0 | 0 |

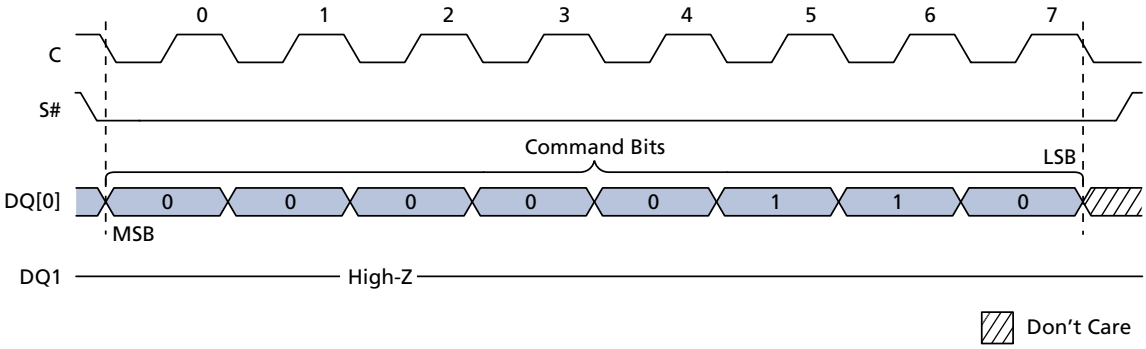
# WRITE ENABLE

The WRITE ENABLE command sets the write enable latch (WEL) bit.

The WEL bit must be set before execution of every PAGE WRITE, PAGE PROGRAM, PAGE ERASE, and SECTOR ERASE command.

The WRITE ENABLE command is entered by driving chip select (S#) LOW, sending the command code, and then driving S# HIGH.

**Figure 6: WRITE ENABLE Command Sequence**



# WRITE DISABLE

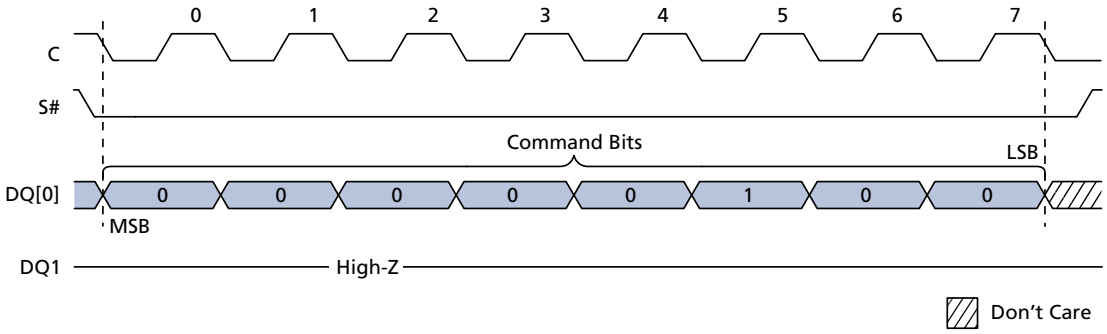
The WRITE DISABLE command resets the write enable latch (WEL) bit.

The WRITE DISABLE command is entered by driving chip select (S#) LOW, sending the command code, and then driving S# HIGH.

The WEL bit is reset under the following conditions:

- Power-up
- Completion of WRITE DISABLE operation
- Completion of PAGE WRITE operation
- Completion of PAGE PROGRAM operation
- Completion of WRITE STATUS REGISTER operation
- Completion of WRITE TO LOCK REGISTER operation
- Completion of PAGE ERASE operation
- Completion of SUBSECTOR ERASE operation
- Completion of SECTOR ERASE operation
- Completion of BULK ERASE operation

**Figure 7: WRITE DISABLE Command Sequence**



# READ IDENTIFICATION

The READ IDENTIFICATION command reads the following device identification data:

- Manufacturer identification (1 byte): This is assigned by JEDEC.
- Device identification (2 bytes): This is assigned by device manufacturer; the first byte indicates memory type and the second byte indicates device memory capacity.
- A Unique ID code (UID) (17 bytes,16 available upon customer request): The first byte contains length of data to follow; the remaining 16 bytes contain optional Customized Factory Data (CFD) content.

**Table 7: READ IDENTIFICATION Data Out Sequence**

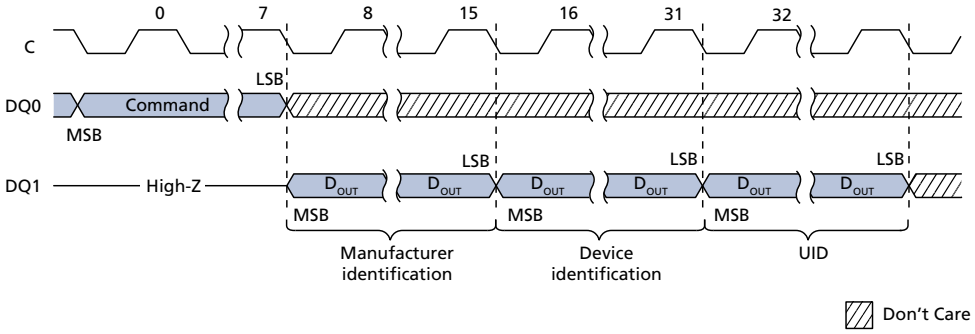
| Manufacturer Identification | Device Identification | |
|---|---|---|
| | Memory Type | Memory Capacity |
| 20h | 80h | 13h |

A READ IDENTIFICATION command is not decoded while an ERASE or PROGRAM cycle is in progress and has no effect on a cycle in progress.

The device is first selected by driving chip select (S#) LOW. Then, the 8-bit command code is shifted in and content is shifted out on serial data output (DQ1) as follows: the 24-bit device identification that is stored in the memory, the 8-bit CFD length, followed by 16 bytes of CFD content. Each bit is shifted out during the falling edge of serial clock (C).

The READ IDENTIFICATION command is terminated by driving S# HIGH at any time during data output. When S# is driven HIGH, the device is put in the STANDBY POWER mode and waits to be selected so that it can receive, decode, and execute commands.

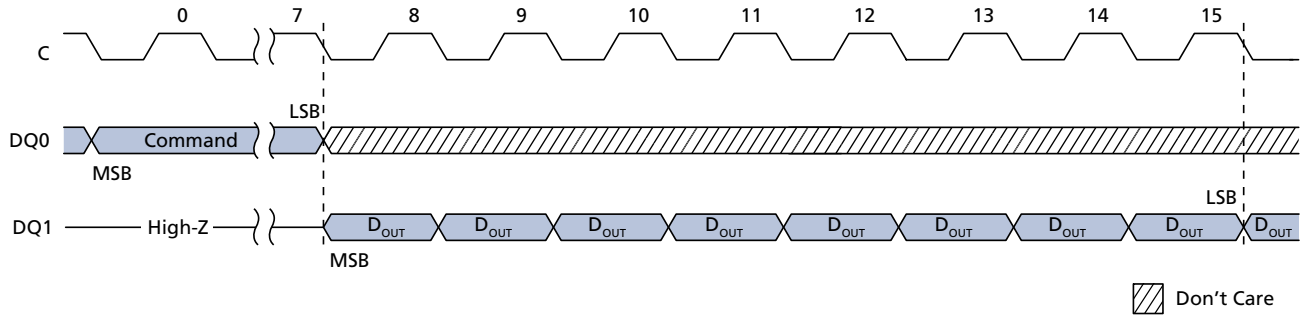
**Figure 8: READ IDENTIFICATION Command Sequence**




# READ STATUS REGISTER

The READ STATUS REGISTER command allows the status register to be read. The status register may be read at any time, even while a PROGRAM, ERASE, or WRITE STATUS REGISTER cycle is in progress. When one of these cycles is in progress, it is recommended to check the write in progress (WIP) bit before sending a new command to the device. It is also possible to read the status register continuously.

**Figure 9: READ STATUS REGISTER Command Sequence**




## WIP Bit

The write in progress (WIP) bit indicates whether the memory is busy with a WRITE cycle, a PROGRAM cycle, or an ERASE cycle. When the WIP bit is set to 1, a cycle is in progress; when the WIP bit is set to 0, a cycle is not in progress.

## WEL Bit

The write enable latch (WEL) bit indicates the status of the internal write enable latch. When the WEL bit is set to 1, the internal write enable latch is set; when the WEL bit is

set to 0, the internal write enable latch is reset and no WRITE , PROGRAM, or ERASE command is accepted.

## Block Protect Bits

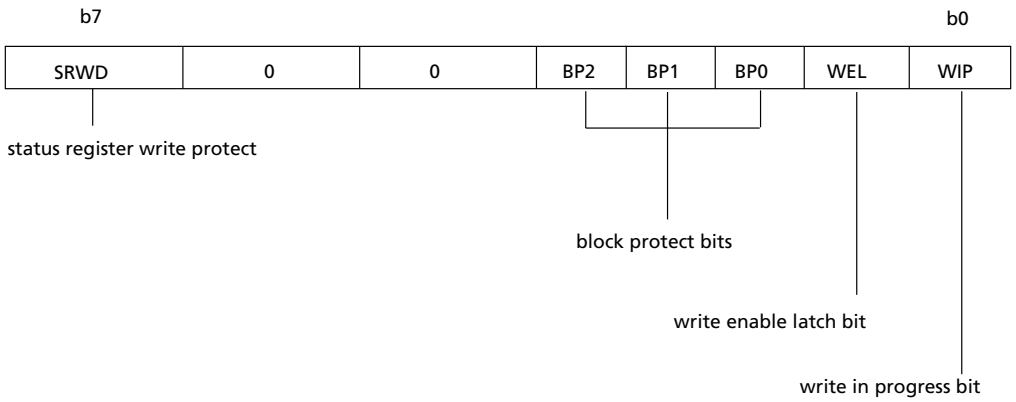
The block protect (BP2, BP1, BP0) bits are non-volatile. They define the size of the area to be software protected against PROGRAM and ERASE commands. The block protect bits are written with the WRITE STATUS REGISTER command.

When one or more of the block protect (BP2, BP1, BP0) bits is set to 1, the relevant memory area, as defined in the Protected Area Sizes table, becomes protected against PAGE PROGRAM, SECTOR ERASE, and SUBSECTOR ERASE commands. The block protect BP2, BP1, BP0) bits can be written provided that the HARDWARE PROTECTED mode has not been set. The BULK ERASE command is executed only if all block protect (BP2, BP1, BP0) bits are 0 and the Lock Register protection bits are not all set to 1.

## SRWD Bit

The status register write disable (SRWD) bit is operated in conjunction with the write protect (W#) signal. When the SRWD bit is set to 1 and W# is driven LOW, the device is put in the hardware protected mode. In the hardware protected mode, the non-volatile bits of the status register (SRWD, BP2, BP1, BP0) become read-only bits and the WRITE STATUS REGISTER command is no longer accepted for execution.

**Figure 10: Status Register Format**



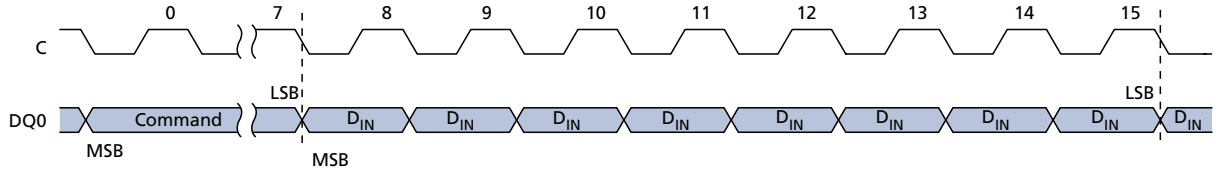

**Note:** WEL and WIP are volatile read-only bits (WEL is set and reset by specific instructions; WIP is automatically set and rest by the internal logic of the device). SRWD = status register write protect bit; BP0, BP1, BP2 = block protect bits.

# WRITE STATUS REGISTER

The WRITE STATUS REGISTER command allows new values to be written to the status register. Before the WRITE STATUS REGISTER command can be accepted, a WRITE ENABLE command must have been executed previously. After the WRITE ENABLE command has been decoded and executed, the device sets the write enable latch (WEL) bit.

The WRITE STATUS REGISTER command is entered by driving chip select (S#) LOW, followed by the command code and the data byte on serial data input (DQ0). The WRITE STATUS REGISTER command has no effect on b6, b5, b4, b1 and b0 of the status register. The status register b6, b5, and b4 are always read as 0. S# must be driven HIGH after the eighth bit of the data byte has been latched in. If not, the WRITE STATUS REGISTER command is not executed.

**Figure 11: WRITE STATUS REGISTER Command Sequence**




As soon as S# is driven HIGH, the self-timed WRITE STATUS REGISTER cycle is initiated; its duration is $t_W$. While the WRITE STATUS REGISTER cycle is in progress, the status register may still be read to check the value of the write in progress (WIP) bit. The WIP bit is 1 during the self-timed WRITE STATUS REGISTER cycle, and is 0 when the cycle is completed. Also, when the cycle is completed, the WEL bit is reset.

The WRITE STATUS REGISTER command allows the user to change the values of the block protect bits. Setting these bit values defines the size of the area that is to be treated as read-only, as defined in the Protected Area Sizes table.

The WRITE STATUS REGISTER command also allows the user to set and reset the status register write disable (SRWD) bit in accordance with the write protect (W#) signal. The SRWD bit and the W# signal allow the device to be put in the HARDWARE PROTECTED (HPM) mode. The WRITE STATUS REGISTER command is not executed once the HPM is entered. The options for enabling the status register protection modes are summarized here.

**Table 8: Status Register Protection Modes**

| W# Signal | SRWD Bit | Protection Mode (PM) | Status Register Write Protection | Memory Content | |
|---|---|---|---|---|---|
| | | | | Protected Area[1] | Unprotected Area[1] |
| 1 | 0 | SECOND SOFTWARE PROTECTED mode (SPM2) | Software protection | Commands not accepted | Commands accepted |
| 0 | 0 | | | | |
| 1 | 1 | | | | |
| 0 | 1 | HARDWARE PROTECTED mode (HPM) | Hardware protection | Commands not accepted | Commands accepted |

Note: 1. As defined by the values in the Block Protect bits of the status register.

When the SRWD bit of the status register is 0 (its initial delivery state), it is possible to write to the status register provided that the WEL bit has been set previously by a WRITE ENABLE command, regardless of whether the W# signal is driven HIGH or LOW. When the status register SRWD bit is set to 1, two cases need to be considered depending on the state of the W# signal:

- If the W# signal is driven HIGH, it is possible to write to the status register provided that the WEL bit has been set previously by a WRITE ENABLE command.
- If the W# signal is driven LOW, it is not possible to write to the status register even if the WEL bit has been set previously by a WRITE ENABLE command. Therefore, attempts to write to the status register are rejected, and are not accepted for execution. The result is that all the data bytes in the memory area that have been put in SPM2 by the status register block protect bits (BP1, BP0) are also hardware protected against data modification.

Regardless of the order of the two events, the HPM can be entered in either of the following ways:

- Setting the status register SRWD bit after driving the W# signal LOW
- Driving the W# signal LOW after setting the status register SRWD bit.

The only way to exit the HPM is to pull the W# signal HIGH. If the W# signal is permanently tied HIGH, the HPM can never be activated. In this case, only the SPM2 is available, using the status register block protect bits.