Chipsmall
深圳市津利帝科技有限公司
Geelead Technology Limited
Chipsmall
采购部

# AT25DQ321

## 32-Mbit, 2.7V Minimum SPI Serial Flash Memory with Dual-I/O and Quad-I/O Support

**DATASHEET**

**(Not Recommended for New Designs)**

## Features

- Single 2.7V - 3.6V supply
- Serial Peripheral Interface (SPI) compatible
  - Supports SPI Modes 0 and 3
  - Supports RapidS™ operation
  - Supports Dual- and Quad-Input Program
  - Supports Dual- and Quad-Output Read
- Very high operating frequencies
  - 100MHz for RapidS
  - 85MHz for SPI
  - Clock-to-output ($t_V$) of 5ns maximum
- Flexible, optimized erase architecture for code + data storage applications
  - Uniform 4KB, 32KB, and 64KB Block Erase
  - Full Chip Erase
- Individual sector protection with Global Protect/Unprotect feature
  - 64 sectors of 64KB each
- Hardware controlled locking of protected sectors via $\overline{\text{WP}}$ pin
- Sector Lockdown
  - Make any combination of 64KB sectors permanently read-only
- 128-byte Programmable OTP Security Register
- Flexible programming
  - Byte/Page Program (1 to 256 bytes)
- Fast Program and Erase times
  - 1.5ms typical Page Program (256 bytes) time
  - 50ms typical 4KB Block Erase time
  - 250ms typical 32KB Block Erase time
  - 400ms typical 64KB Block Erase time
- Program and Erase Suspend/Resume
- Automatic checking and reporting of erase/program failures
- Software controlled reset
- JEDEC Standard Manufacturer and Device ID Read Methodology
- Low power dissipation
  - 7mA Active Read current (typical at 20MHz)
  - 5µA Deep Power-Down current (typical)
- Endurance: 100,000 program/erase cycles
- Data retention: 20 years
- Complies with full industrial temperature range
- Industry standard green (Pb/Halide-free/RoHS compliant) package options
  - 8-lead SOIC (208-mil wide)
  - 8-pad Ultra-thin DFN (5 x 6 x 0.6mm)
  - 16-lead SOIC (300-mil wide)

# 1. Description

The AT25DQ321 is a serial interface Flash memory device designed for use in a wide variety of high-volume consumer based applications in which program code is shadowed from Flash memory into embedded or external RAM for execution. The flexible erase architecture of the AT25DQ321, with its erase granularity as small as 4KB, makes it ideal for data storage as well, eliminating the need for additional data storage EEPROM devices.

The physical sectoring and the erase block sizes of the AT25DQ321 have been optimized to meet the needs of today's code and data storage applications. By optimizing the size of the physical sectors and erase blocks, the memory space can be used much more efficiently. Because certain code modules and data storage segments must reside by themselves in their own protected sectors, the wasted and unused memory space that occurs with large sectored and large block erase Flash memory devices can be greatly reduced. This increased memory space efficiency allows additional code routines and data storage segments to be added while still maintaining the same overall device density.

The AT25DQ321 also offers a sophisticated method for protecting individual sectors against erroneous or malicious program and erase operations. By providing the ability to individually protect and unprotect sectors, a system can unprotect a specific sector to modify its contents while keeping the remaining sectors of the memory array securely protected. This is useful in applications where program code is patched or updated on a subroutine or module basis, or in applications where data storage segments need to be modified without running the risk of errant modifications to the program code segments. In addition to individual sector protection capabilities, the AT25DQ321 incorporates Global Protect and Global Unprotect features that allow the entire memory array to be either protected or unprotected all at once. This reduces overhead during the manufacturing process since sectors do not have to be unprotected one-by-one prior to initial programming.

To take code and data protection to the next level, the AT25DQ321 incorporates a sector lockdown mechanism that allows any combination of individual 64KB sectors to be locked down and become permanently read-only. This addresses the need of certain secure applications that require portions of the Flash memory array to be permanently protected against malicious attempts at altering program code, data modules, security information or encryption/decryption algorithms, keys, and routines. The device also contains a specialized OTP (One-Time Programmable) Security Register that can be used for purposes such as unique device serialization, system-level Electronic Serial Number (ESN) storage, locked key storage, etc.

Specifically designed for use in 3V systems, the AT25DQ321 supports read, program, and erase operations with a supply voltage range of 2.7V to 3.6V. No separate voltage is required for programming and erasing.

## 2. Pin Descriptions and Pinouts

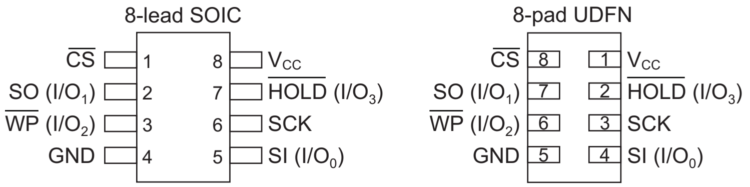
**Table 2-1. Pin Descriptions**

| Symbol | Name and Function | Asserted State | Type |
|---|---|---|---|
| $\overline{CS}$ | **Chip Select:** Asserting the $\overline{CS}$ pin selects the device. When the $\overline{CS}$ pin is deasserted, the device will be deselected and normally be placed in standby mode (not Deep Power-Down mode) and the SO pin will be in a high-impedance state. When the device is deselected, data will not be accepted on the SI pin.<br>A high-to-low transition on the $\overline{CS}$ pin is required to start an operation and a low-to-high transition is required to end an operation. When ending an internally self-timed operation such as a program or erase cycle, the device will not enter the standby mode until the completion of the operation. | Low | Input |
| SCK | **Serial Clock:** This pin is used to provide a clock to the device and is used to control the flow of data to and from the device. Command, address and input data present on the SI pin or I/O pins is always latched in on the rising edge of SCK, while output data on the SO pin or I/O pins is always clocked out on the falling edge of SCK. | - | Input |
| SI (I/O$_0$) | **Serial Input (I/O$_0$):** The SI pin is used to shift data into the device. The SI pin is used for all data input including command and address sequences. Data on the SI pin is always latched in on the rising edge of SCK.<br>With the Dual-Input and Quad-Input Byte/Page Program commands, the SI pin is used as an input pin (I/O$_0$) in conjunction with other pins to allow two bits (on I/O$_{1-0}$) or four bits (on I/O$_{3-0}$) of data to be clocked in on every rising edge of SCK. With the Dual-Output and Quad-Output Read Array commands, the SI pin becomes an output pin (I/O$_0$) and, along with other pins, allows two bits (on I/O$_{1-0}$) or four bits (on I/O$_{3-0}$) of data to be clocked out on every falling edge of SCK. To maintain consistency with SPI nomenclature, the SI (I/O$_0$) pin will be referenced as SI throughout the document with exception to sections dealing with the Dual-Input and Quad-Dual-Output Byte/Page Program commands as well as the Dual-Output and Quad-Output Read Array commands in which it will be referenced as I/O$_0$.<br>Data present on the SI pin will be ignored whenever the device is deselected ($\overline{CS}$ is deasserted). | - | Input/Output |
| SO (I/O$_1$) | **Serial Output (I/O$_1$):** The SO pin is used to shift data out from the device. Data on the SO pin is always clocked out on the falling edge of SCK.<br>With the Dual-Input and Quad-Input Byte/Page Program commands, the SO pin becomes an input pin (I/O$_1$) and, along with other pins, allows two bits (on I/O$_{1-0}$) or four bits (on I/O$_{3-0}$) of data to be clocked in on every rising edge of SCK. With the Dual-Output and Quad-Output Read Array commands, the SO pin is used as an output pin (I/O$_1$) in conjunction with other pins to allow two bits (on I/O$_{1-0}$) or four bits (on I/O$_{3-0}$) of data to be clocked out on every falling edge of SCK. To maintain consistency with SPI nomenclature, the SO (I/O$_1$) pin will be referenced as SO throughout the document with exception to sections dealing with the Dual-Input and Quad-Input Byte/Page Program commands as well as the Dual-Output and Quad-Output Read Array commands in which it will be referenced as I/O$_1$.<br>The SO pin will be in a high-impedance state whenever the device is deselected ($\overline{CS}$ is deasserted). | - | Input/Output |

Table 2-1. Pin Descriptions (Continued)

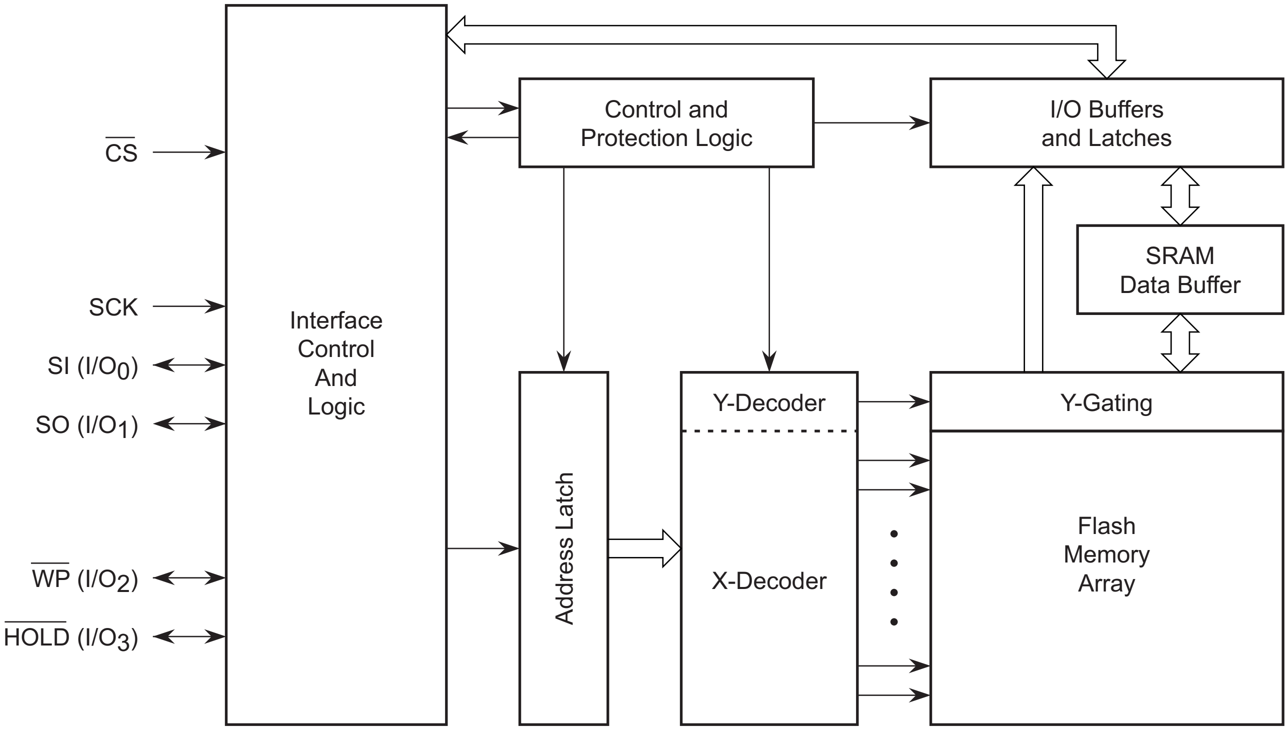
| Symbol | Name and Function | Asserted State | Type |
|---|---|---|---|
| $\overline{\text{WP}}$ ($I/O_2$) | **Write Protect ($I/O_2$):** The WP# pin controls the hardware locking feature of the device. See “Protection Commands and Features” on page 24 for more details on protection features and the $\overline{\text{WP}}$ pin.<br>With the Quad-Input Byte/Page Program command, the $\overline{\text{WP}}$ pin becomes an input pin ($I/O_2$) and, along with other pins, allows four bits (on $I/O_{3-0}$) of data to be clocked in on every rising edge of SCK. With the Quad-Output Read Array command, the $\overline{\text{WP}}$ pin becomes an output pin ($I/O_2$) and, when used with other pins, allows four bits (on $I/O_{3-0}$) of data to be clocked out on every falling edge of SCK. The QE bit in the Configuration Register must be set in order for the $\overline{\text{WP}}$ pin to be used as an I/O data pin.<br>The $\overline{\text{WP}}$ pin must be driven at all times or pulled-high using an external pull-up resistor. | Low | Input/Output |
| $\overline{\text{HOLD}}$ ($I/O_3$) | **Hold ($I/O_3$):** The $\overline{\text{HOLD}}$ pin is used to temporarily pause serial communication without deselecting or resetting the device. While the $\overline{\text{HOLD}}$ pin is asserted, transitions on the SCK pin and data on the SI pin will be ignored and the SO pin will be in a high-impedance state.<br>The $\overline{\text{CS}}$ pin must be asserted and the SCK pin must be in the low state in order for a Hold condition to start. A Hold condition pauses serial communication only and does not have an affect on internally self-timed operations such as a program or erase cycle. See “Hold” on page 53 for additional details on the Hold operation.<br>With the Quad-Input Byte/Page Program command, the $\overline{\text{HOLD}}$ pin becomes an input pin ($I/O_3$) and, along with other pins, allows four bits (on $I/O_{3-0}$) of data to be clocked in on every rising edge of SCK. With the Quad-Output Read Array command, the $\overline{\text{HOLD}}$ pin becomes an output pin ($I/O_3$) and, when used with other pins, allows four bits (on $I/O_{3-0}$) of data to be clocked out on every falling edge of SCK. The QE bit in the Configuration Register must be set in order for the $\overline{\text{HOLD}}$ pin to be used as an I/O data pin.<br>The $\overline{\text{HOLD}}$ pin must be driven at all times or pulled-high using an external pull-up resistor. | Low | Input/Output |
| $V_{CC}$ | **Device Power Supply:** The $V_{CC}$ pin is used to supply the source voltage to the device.<br>Operations at invalid $V_{CC}$ voltages may produce spurious results and should not be attempted. | - | Power |
| GND | **Ground:** The ground reference for the power supply. GND should be connected to the system ground. | - | Power |

Figure 2-1. Pin Configurations



## 3. Block Diagram

**Figure 3-1. Block Diagram**



Note: I/O$_{3-0}$ pin naming convention is used for Dual-I/O and Quad-I/O commands.

# 4. Memory Array

To provide the greatest flexibility, the memory array of the AT25DQ321 can be erased in four levels of granularity including a Full Chip Erase. In addition, the array has been divided into physical sectors of uniform size, of which each sector can be individually protected from program and erase operations. The size of the physical sectors is optimized for both code and data storage applications, allowing both code and data segments to reside in their own isolated regions. The Memory Architecture Diagram illustrates the breakdown of each erase level as well as the breakdown of each physical sector.

### Figure 4-1. Memory Architecture Diagram

**Block Erase Detail**

| Internal Sectoring for Sector Protection Function | 64KB Block Erase (D8h Command) | 32KB Block Erase (52h Command) | 4KB Block Erase (20h Command) | Block Address Range |
|---|---|---|---|---|
| 64KB (Sector 63) | 64KB | 32KB | 4KB | 3FFFFFh – 3FF000h |
| | | | 4KB | 3FEFFFh – 3FE000h |
| | | | 4KB | 3FDFFFh – 3FD000h |
| | | | 4KB | 3FCFFFh – 3FC000h |
| | | | 4KB | 3FBFFFh – 3FB000h |
| | | | 4KB | 3FAFFFh – 3FA000h |
| | | | 4KB | 3F9FFFh – 3F9000h |
| | | | 4KB | 3F8FFFh – 3F8000h |
| | | 32KB | 4KB | 3F7FFFh – 3F7000h |
| | | | 4KB | 3F6FFFh – 3F6000h |
| | | | 4KB | 3F5FFFh – 3F5000h |
| | | | 4KB | 3F4FFFh – 3F4000h |
| | | | 4KB | 3F3FFFh – 3F3000h |
| | | | 4KB | 3F2FFFh – 3F2000h |
| | | | 4KB | 3F1FFFh – 3F1000h |
| | | | 4KB | 3F0FFFh – 3F0000h |
| 64KB (Sector 62) | 64KB | 32KB | 4KB | 3EFFFFh – 3EF000h |
| | | | 4KB | 3EEFFFh – 3EE000h |
| | | | 4KB | 3EDFFFh – 3ED000h |
| | | | 4KB | 3ECFFFh – 3EC000h |
| | | | 4KB | 3EBFFFh – 3EB000h |
| | | | 4KB | 3EAFFFh – 3EA000h |
| | | | 4KB | 3E9FFFh – 3E9000h |
| | | | 4KB | 3E8FFFh – 3E8000h |
| | | 32KB | 4KB | 3E7FFFh – 3E7000h |
| | | | 4KB | 3E6FFFh – 3E6000h |
| | | | 4KB | 3E5FFFh – 3E5000h |
| | | | 4KB | 3E4FFFh – 3E4000h |
| | | | 4KB | 3E3FFFh – 3E3000h |
| | | | 4KB | 3E2FFFh – 3E2000h |
| | | | 4KB | 3E1FFFh – 3E1000h |
| | | | 4KB | 3E0FFFh – 3E0000h |
| ⋮ | ⋮ | ⋮ | ⋮ | |
| 64KB (Sector 0) | 64KB | 32KB | 4KB | 00FFFFh – 00F000h |
| | | | 4KB | 00EFFFh – 00E000h |
| | | | 4KB | 00DFFFh – 00D000h |
| | | | 4KB | 00CFFFh – 00C000h |
| | | | 4KB | 00BFFFh – 00B000h |
| | | | 4KB | 00AFFFh – 00A000h |
| | | | 4KB | 009FFFh – 009000h |
| | | | 4KB | 008FFFh – 008000h |
| | | 32KB | 4KB | 007FFFh – 007000h |
| | | | 4KB | 006FFFh – 006000h |
| | | | 4KB | 005FFFh – 005000h |
| | | | 4KB | 004FFFh – 004000h |
| | | | 4KB | 003FFFh – 003000h |
| | | | 4KB | 002FFFh – 002000h |
| | | | 4KB | 001FFFh – 001000h |
| | | | 4KB | 000FFFh – 000000h |

**Page Program Detail**

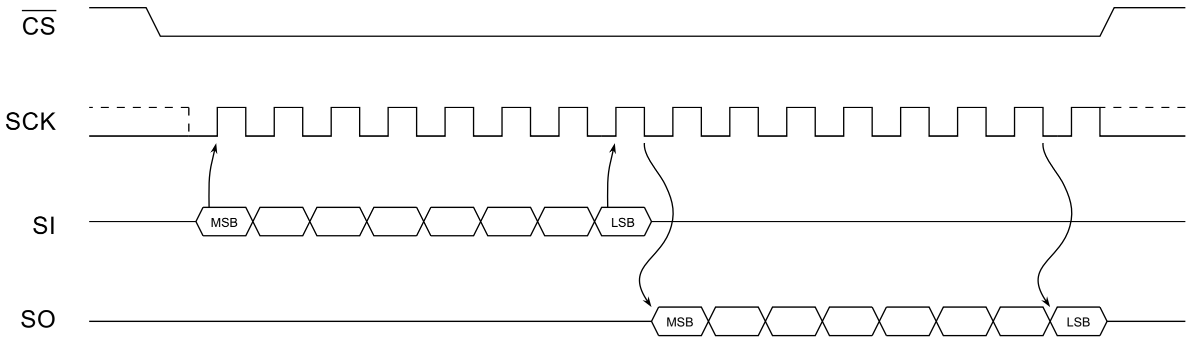
| 1-256 Byte Page Program (02h Command) | Page Address Range |
|---|---|
| 256 bytes | 3FFFFFh – 3FFF00h |
| 256 bytes | 3FFEFFh – 3FFE00h |
| 256 bytes | 3FFDFFh – 3FFD00h |
| 256 bytes | 3FFCFFh – 3FFC00h |
| 256 bytes | 3FFBFFh – 3FFB00h |
| 256 bytes | 3FFAFFh – 3FFA00h |
| 256 bytes | 3FF9FFh – 3FF900h |
| 256 bytes | 3FF8FFh – 3FF800h |
| 256 bytes | 3FF7FFh – 3FF700h |
| 256 bytes | 3FF6FFh – 3FF600h |
| 256 bytes | 3FF5FFh – 3FF500h |
| 256 bytes | 3FF4FFh – 3FF400h |
| 256 bytes | 3FF3FFh – 3FF300h |
| 256 bytes | 3FF2FFh – 3FF200h |
| 256 bytes | 3FF1FFh – 3FF100h |
| 256 bytes | 3FF0FFh – 3FF000h |
| 256 bytes | 3FEFFFh – 3FEF00h |
| 256 bytes | 3FEEFFh – 3FEE00h |
| 256 bytes | 3FEDFFh – 3FED00h |
| 256 bytes | 3FECFFh – 3FEC00h |
| 256 bytes | 3FEBFFh – 3FEB00h |
| 256 bytes | 3FEAFFh – 3FEA00h |
| 256 bytes | 3FE9FFh – 3FE900h |
| 256 bytes | 3FE8FFh – 3FE800h |
| ⋮ | |
| 256 bytes | 0017FFh – 001700h |
| 256 bytes | 0016FFh – 001600h |
| 256 bytes | 0015FFh – 001500h |
| 256 bytes | 0014FFh – 001400h |
| 256 bytes | 0013FFh – 001300h |
| 256 bytes | 0012FFh – 001200h |
| 256 bytes | 0011FFh – 001100h |
| 256 bytes | 0010FFh – 001000h |
| 256 bytes | 000FFFh – 000F00h |
| 256 bytes | 000EFFh – 000E00h |
| 256 bytes | 000DFFh – 000D00h |
| 256 bytes | 000CFFh – 000C00h |
| 256 bytes | 000BFFh – 000B00h |
| 256 bytes | 000AFFh – 000A00h |
| 256 bytes | 0009FFh – 000900h |
| 256 bytes | 0008FFh – 000800h |
| 256 bytes | 0007FFh – 000700h |
| 256 bytes | 0006FFh – 000600h |
| 256 bytes | 0005FFh – 000500h |
| 256 bytes | 0004FFh – 000400h |
| 256 bytes | 0003FFh – 000300h |
| 256 bytes | 0002FFh – 000200h |
| 256 bytes | 0001FFh – 000100h |
| 256 bytes | 0000FFh – 000000h |

# 5. Device Operation

The AT25DQ321 is controlled by a set of instructions that are sent from a host controller, commonly referred to as the SPI Master. The SPI Master communicates with the AT25DQ321 via the SPI bus which is comprised of four signal lines: Chip Select ($\overline{\text{CS}}$), Serial Clock (SCK), Serial Input (SI), and Serial Output (SO).

The SPI protocol defines a total of four modes of operation (mode 0, 1, 2, or 3) with each mode differing in respect to the SCK polarity and phase and how the polarity and phase control the flow of data on the SPI bus. The AT25DQ321 supports the two most common modes, SPI Modes 0 and 3. The only difference between SPI Modes 0 and 3 is the polarity of the SCK signal when in the inactive state (when the SPI Master is in standby mode and not transferring any data). With SPI Modes 0 and 3, data is always latched in on the rising edge of SCK and always output on the falling edge of SCK.

**Figure 5-1. SPI Mode 0 and 3**



## 5.1 Dual-I/O and Quad-I/O Operation

The AT25DQ321 features a Dual-Input Program mode and a Dual-Output Read mode that allows two bits of data to be clocked into or out of the device every clock cycle to improve throughputs. To accomplish this, both the SI and SO pins are utilized as inputs/outputs for the transfer of data bytes. With the Dual-Input Byte/Page Program command, the the SO pin becomes an input along with the SI pin. Alternatively, with the Dual-Output Read Array command, the SI pin becomes an output along with the SO pin. For both Dual-I/O commands, the SO pin will be referred to as I/O$_{1}$ and the SI pin will be referred to as I/O$_{0}$.

The device also supports a Quad-Input Program mode and a Quad-Output Read mode in which the $\overline{\text{WP}}$ and $\overline{\text{HOLD}}$ pins become data pins for even higher throughputs. For the Quad-Input Byte/Page Program command and for the Quad-Output Read Array command, the $\overline{\text{HOLD}}$, $\overline{\text{WP}}$, SO, and SI pins are referred to as I/O$_{3\text{-}0}$ where $\overline{\text{HOLD}}$ becomes I/O$_{3}$, $\overline{\text{WP}}$ becomes I/O$_{2}$, SO becomes I/O$_{1}$, and SI becomes I/O$_{0}$. The QE bit in the Configuration Register must be set in order for both Quad-I/O commands to be enabled and for the $\overline{\text{HOLD}}$ and $\overline{\text{WP}}$ pins to be converted to I/O data pins.

# 6. Commands and Addressing

A valid instruction or operation must always be started by first asserting the $\overline{\text{CS}}$ pin. After the $\overline{\text{CS}}$ pin has been asserted, the host controller must then clock out a valid 8-bit opcode on the SPI bus. Following the opcode, instruction dependent information such as address and data bytes would then be clocked out by the host controller. All opcode, address, and data bytes are transferred with the Most-Significant Bit (MSB) first. An operation is ended by deasserting the $\overline{\text{CS}}$ pin.

Opcodes not supported by the AT25DQ321 will be ignored by the device and no operation will be started. The device will continue to ignore any data presented on the SI pin until the start of the next operation ($\overline{\text{CS}}$ pin being deasserted and then reasserted). In addition, if the $\overline{\text{CS}}$ pin is deasserted before complete opcode and address information is sent to the device, then no operation will be performed and the device will simply return to the idle state and wait for the next operation.

Addressing of the device requires a total of three bytes of information to be sent, representing address bits A23-A0. Since the upper address limit of the AT25DQ321 memory array is 3FFFFFh, address bits A23-A22 are always ignored by the device.

**Table 6-1. Command Listing**

| Command | Opcode | | Clock Frequency | Address Bytes | Dummy Bytes | Data Bytes |
|---|---|---|---|---|---|---|
| **Read Commands** | | | | | | |
| Read Array | 1Bh | 0001 1011 | Up to 100MHz | 3 | 2 | 1+ |
| | 0Bh | 0000 1011 | Up to 85MHz | 3 | 1 | 1+ |
| | 03h | 0000 0011 | Up to 50MHz | 3 | 0 | 1+ |
| Dual-Output Read Array | 3Bh | 0011 1011 | Up to 85MHz | 3 | 1 | 1+ |
| Quad-Output Read Array | 6Bh | 0110 1011 | Up to 66MHz | 3 | 1 | 1+ |
| **Program and Erase Commands** | | | | | | |
| Block Erase (4KB) | 20h | 0010 0000 | Up to 100MHz | 3 | 0 | 0 |
| Block Erase (32KB) | 52h | 0101 0010 | Up to 100MHz | 3 | 0 | 0 |
| Block Erase (64KB) | D8h | 1101 1000 | Up to 100MHz | 3 | 0 | 0 |
| Chip Erase | 60h | 0110 0000 | Up to 100MHz | 0 | 0 | 0 |
| | C7h | 1100 0111 | Up to 100MHz | 0 | 0 | 0 |
| Byte/Page Program (1 to 256 bytes) | 02h | 0000 0010 | Up to 100MHz | 3 | 0 | 1+ |
| Dual-Input Byte/Page Program (1 to 256 bytes) | A2h | 1010 0010 | Up to 100MHz | 3 | 0 | 1+ |
| Quad-Input Byte/Page Program (1 to 256 bytes) | 32h | 0011 0010 | Up to 100MHz | 3 | 0 | 1+ |
| Program/Erase Suspend | B0h | 1011 0000 | Up to 100MHz | 0 | 0 | 0 |
| Program/Erase Resume | D0h | 1101 0000 | Up to 100MHz | 0 | 0 | 0 |
| **Protection Commands** | | | | | | |
| Write Enable | 06h | 0000 0110 | Up to 100MHz | 0 | 0 | 0 |
| Write Disable | 04h | 0000 0100 | Up to 100MHz | 0 | 0 | 0 |
| Protect Sector | 36h | 0011 0110 | Up to 100MHz | 3 | 0 | 0 |
| Unprotect Sector | 39h | 0011 1001 | Up to 100MHz | 3 | 0 | 0 |
| Global Protect/Unprotect | Use Write Status Register Byte 1 Command | | | | | |
| Read Sector Protection Registers | 3Ch | 0011 1100 | Up to 100MHz | 3 | 0 | 1+ |
| **Security Commands** | | | | | | |
| Sector Lockdown | 33h | 0011 0011 | Up to 100MHz | 3 | 0 | 1 |
| Freeze Sector Lockdown State | 34h | 0011 0100 | Up to 100MHz | 3 | 0 | 1 |
| Read Sector Lockdown Registers | 35h | 0011 0101 | Up to 100MHz | 3 | 0 | 1+ |
| Program OTP Security Register | 9Bh | 1001 1011 | Up to 100MHz | 3 | 0 | 1+ |
| Read OTP Security Register | 77h | 0111 0111 | Up to 100MHz | 3 | 2 | 1+ |
| **Status and Configuration Register Commands** | | | | | | |
| Read Status Register | 05h | 0000 0101 | Up to 100MHz | 0 | 0 | 1+ |
| Write Status Register Byte 1 | 01h | 0000 0001 | Up to 100MHz | 0 | 0 | 1 |
| Write Status Register Byte 2 | 31h | 0011 0001 | Up to 100MHz | 0 | 0 | 1 |
| Read Configuration Register | 3Fh | 0011 1111 | Up to 100MHz | 0 | 0 | 1+ |
| Write Configuration Register | 3Eh | 0011 1110 | Up to 100MHz | 0 | 0 | 1 |
| **Miscellaneous Commands** | | | | | | |
| Reset | F0h | 1111 0000 | Up to 100MHz | 0 | 0 | 1 |
| Read Manufacturer and Device ID | 9Fh | 1001 1111 | Up to 85MHz | 0 | 0 | 1 to 4 |
| Deep Power-Down | B9h | 1011 1001 | Up to 100MHz | 0 | 0 | 0 |
| Resume from Deep Power-Down | ABh | 1010 1011 | Up to 100MHz | 0 | 0 | 0 |

# 7. Read Commands

## 7.1 Read Array

The Read Array command can be used to sequentially read a continuous stream of data from the device by simply providing the clock signal once the initial starting address has been specified. The device incorporates an internal address counter that automatically increments on every clock cycle.

Three opcodes (1Bh, 0Bh, and 03h) can be used for the Read Array command. The use of each opcode depends on the maximum clock frequency that will be used to read data from the device. The 0Bh opcode can be used at any clock frequency up to the maximum specified by $f_{CLK}$, and the 03h opcode can be used for lower frequency read operations up to the maximum specified by $f_{RDLF}$. The 1Bh opcode allows the highest read performance possible and can be used at any clock frequency up to the maximum specified by $f_{MAX}$; however, use of the 1Bh opcode at clock frequencies above $f_{CLK}$ should be reserved to systems employing the RapidS protocol.

To perform the Read Array operation, the $\overline{CS}$ pin must first be asserted and the appropriate opcode (1Bh, 0Bh, or 03h) must be clocked into the device. After the opcode has been clocked in, the three address bytes must be clocked in to specify the starting address location of the first byte to read within the memory array. Following the three address bytes, additional dummy bytes may need to be clocked into the device depending on which opcode is used for the Read Array operation. If the 1Bh opcode is used, then two dummy bytes must be clocked into the device after the three address bytes. If the 0Bh opcode is used, then a single dummy byte must be clocked in after the address bytes.

After the three address bytes (and the dummy bytes or byte if using opcodes 1Bh or 0Bh) have been clocked in, additional clock cycles will result in data being output on the SO pin. The data is always output with the MSB of a byte first. When the last byte (3FFFFFh) of the memory array has been read, the device will continue reading back at the beginning of the array (000000h). No delays will be incurred when wrapping around from the end of the array to the beginning of the array.

Deasserting the $\overline{CS}$ pin will terminate the read operation and put the SO pin into a high-impedance state. The $\overline{CS}$ pin can be deasserted at any time and does not require that a full byte of data be read.

**Figure 7-1. Read Array – 1Bh Opcode**

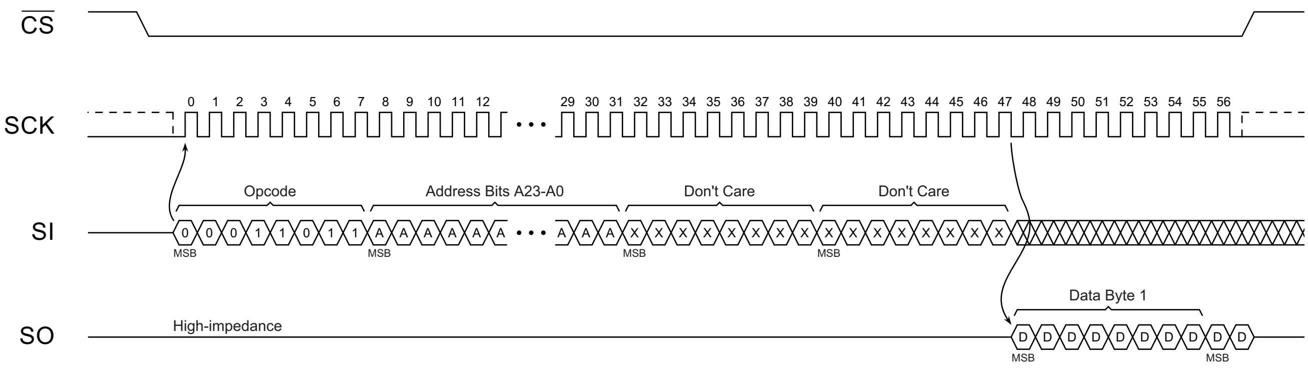


**Figure 7-2. Read Array – 0Bh Opcode**

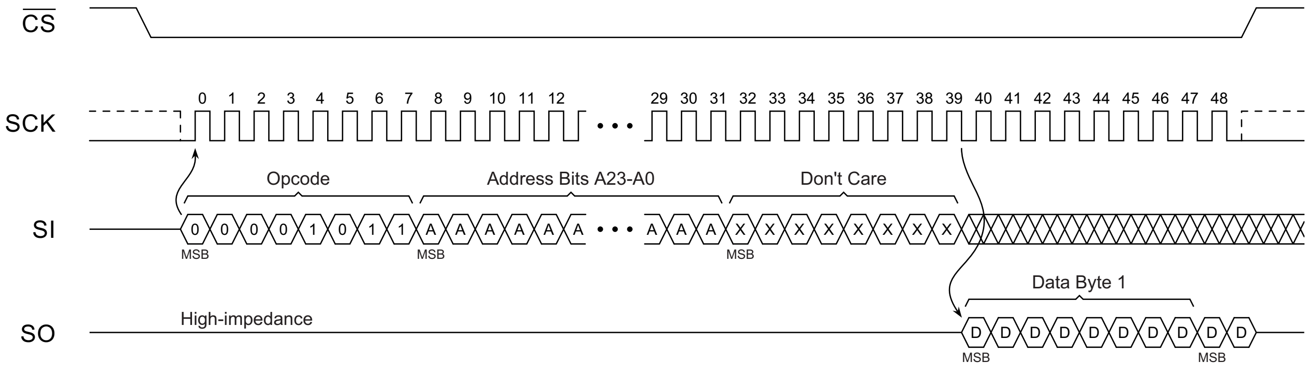
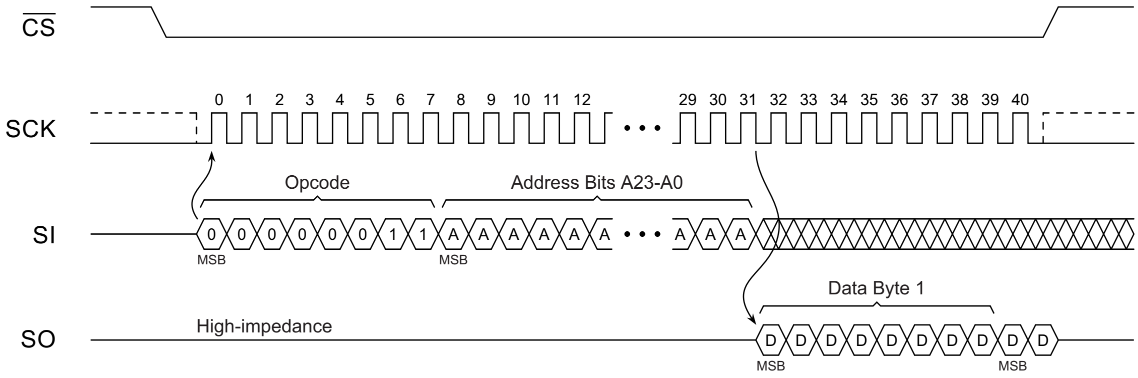


**Figure 7-3. Read Array – 03h Opcode**



## 7.2 Dual-Output Read Array

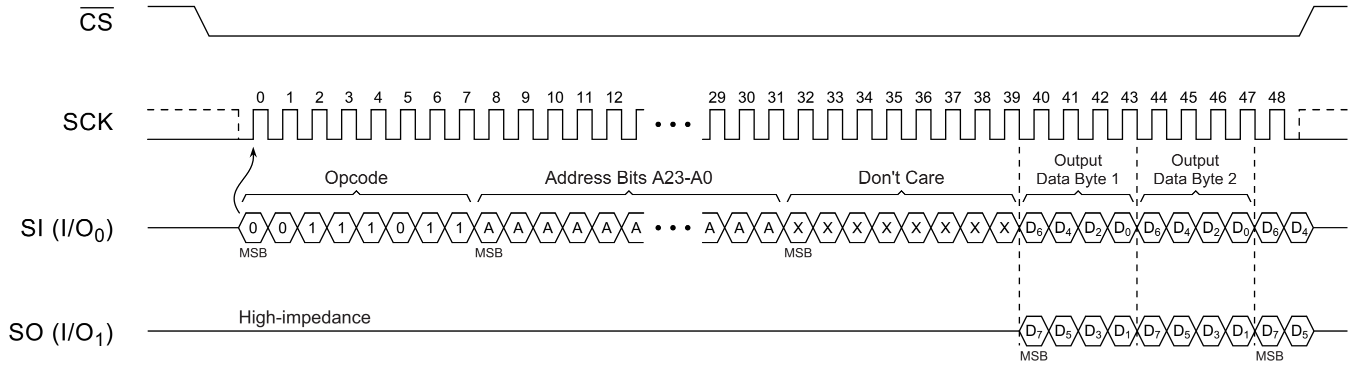
The Dual-Output Read Array command is similar to the standard Read Array command and can be used to sequentially read a continuous stream of data from the device by simply providing the clock signal once the initial starting address has been specified. Unlike the standard Read Array command however, the Dual-Output Read Array command allows two bits of data to be clocked out of the device on every clock cycle rather than just one.

The Dual-Output Read Array command can be used at any clock frequency up to the maximum specified by $f_{RDDO}$. To perform the Dual-Output Read Array operation, the $\overline{CS}$ pin must first be asserted and the opcode of 3Bh must be clocked into the device. After the opcode has been clocked in, the three address bytes must be clocked in to specify the starting address location of the first byte to read within the memory array. Following the three address bytes, a single dummy byte must also be clocked into the device.

After the three address bytes and the dummy byte have been clocked in, additional clock cycles will result in data being output on both the $I/O_1$ and $I/O_0$ pins. The data is always output with the MSB of a byte first, and the MSB is always output on the $I/O_1$ pin. During the first clock cycle, bit 7 of the first data byte will be output on the $I/O_1$ pin while bit 6 of the same data byte will be output on the $I/O_0$ pin. During the next clock cycle, bits 5 and 4 of the first data byte will be output on the $I/O_1$ and $I/O_0$ pins, respectively. The sequence continues with each byte of data being output after every four clock cycles. When the last byte (3FFFFFh) of the memory array has been read, the device will continue reading back at the beginning of the array (000000h). No delays will be incurred when wrapping around from the end of the array to the beginning of the array.

Deasserting the $\overline{CS}$ pin will terminate the read operation and put the $I/O_{1-0}$ pins into a high-impedance state. The $\overline{CS}$ pin can be deasserted at any time and does not require that a full byte of data be read.

**Figure 7-4. Dual-Output Read Array**



## 7.3 Quad-Output Read Array

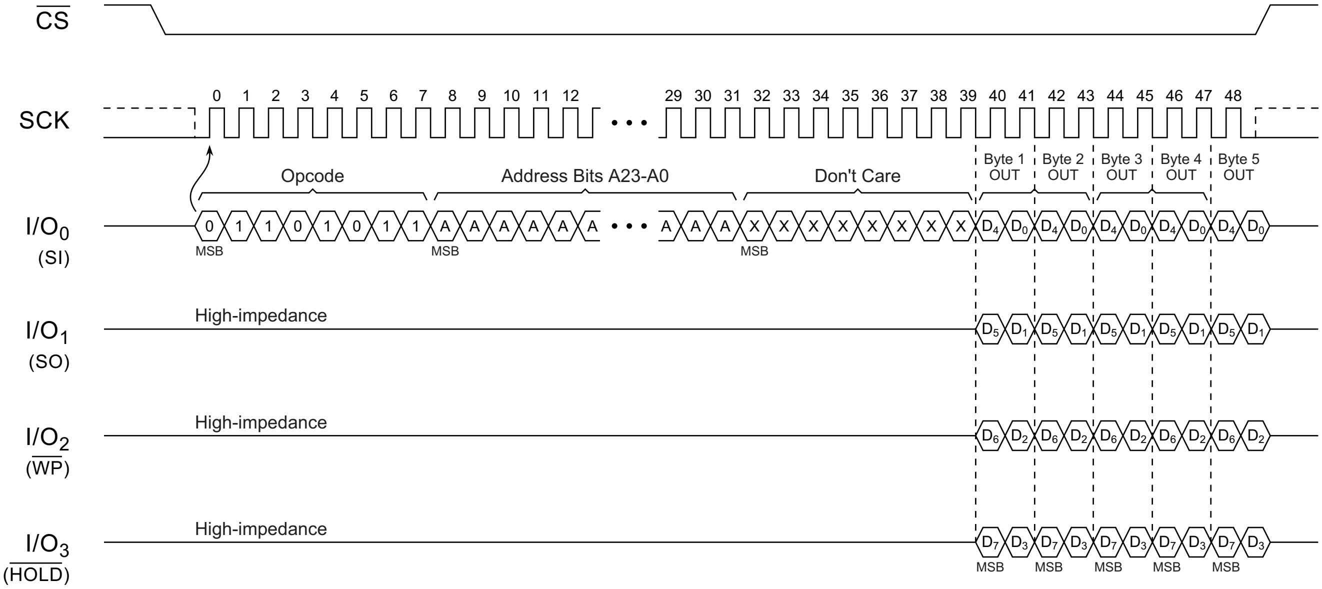
The Quad-Output Read Array command is similar to the Dual-Output Read Array command and can be used to sequentially read a continuous stream of data from the device by simply providing the clock signal once the initial starting address has been specified. Unlike the Dual-Output Read Array command however, the Quad-Output Read Array command allows four bits of data to be clocked out of the device on every clock cycle rather than two.

The Quad-Output Read Array command can be used at any clock frequency up to the maximum specified by $f_{RDQO}$. To perform the Quad-Output Read Array operation, the $\overline{CS}$ pin must first be asserted and the opcode of 6Bh must be clocked into the device. After the opcode has been clocked in, the three address bytes must be clocked in to specify the starting address location of the first byte to read within the memory array. Following the three address bytes, a single dummy byte must also be clocked into the device.

After the three address bytes and the dummy byte have been clocked in, additional clock cycles will result in data being output on the $I/O_{3-0}$ pins. The data is always output with the MSB of a byte first and the MSB is always output on the $I/O_3$ pin. During the first clock cycle, bit 7 of the first data byte will be output on the $I/O_3$ pin while bits 6, 5, and 4 of the same data byte will be output on the $I/O_2$, $I/O_1$, and $I/O_0$ pins, respectively. During the next clock cycle, bits 3, 2, 1, and 0 of the first data byte will be output on the $I/O_3$, $I/O_2$, $I/O_1$ and $I/O_0$ pins, respectively. The sequence continues with each byte of data being output after every two clock cycles. When the last byte (3FFFFFh) of the memory array has been read, the device will continue reading back at the beginning of the array (000000h). No delays will be incurred when wrapping around from the end of the array to the beginning of the array.

Deasserting the $\overline{CS}$ pin will terminate the read operation and put the $I/O_{3-0}$ pins into a high-impedance state. The $\overline{CS}$ pin can be deasserted at any time and does not require that a full byte of data be read.

**Figure 7-5. Quad-Output Read Array**



# 8. Program and Erase Commands

## 8.1 Byte/Page Program

The Byte/Page Program command allows anywhere from a single byte of data to 256 bytes of data to be programmed into previously erased memory locations. An erased memory location is one that has all eight bits set to the Logical 1 state (a byte value of FFh). Before a Byte/Page Program command can be started, the Write Enable command must have been previously issued to the device (see "Write Enable" on page 24) to set the Write Enable Latch (WEL) bit of the Status Register to a Logical 1 state.

To perform a Byte/Page Program command, an opcode of 02h must be clocked into the device followed by the three address bytes denoting the first byte location of the memory array to begin programming at. After the address bytes have been clocked in, data can then be clocked into the device and will be stored in an internal buffer.

If the starting memory address denoted by A23-A0 does not fall on an even 256-byte page boundary (A7-A0 are not all 0), then special circumstances regarding which memory locations to be programmed will apply. In this situation, any data that is sent to the device that goes beyond the end of the page will wrap around back to the beginning of the same page.

**Example:** If the starting address denoted by A23-A0 is 0000FEh and three bytes of data are sent to the device, then the first two bytes of data will be programmed at addresses 0000FEh and 0000FFh while the last byte of data will be programmed at address 000000h. The remaining bytes in the page (addresses 000001h through 0000FDh) will not be programmed and will remain in the erased state (FFh). In addition, if more than 256 bytes of data are sent to the device, then only the last 256 bytes sent will be latched into the internal buffer.

When the $\overline{\text{CS}}$ pin is deasserted, the device will take the data stored in the internal buffer and program it into the appropriate memory array locations based on the starting address specified by A23-A0 and the number of data bytes sent to the device. If less than 256 bytes of data were sent to the device, then the remaining bytes within the page will not be programmed and will remain in the erased state (FFh). The programming of the data bytes is internally self-timed and should take place in a time of $t_{PP}$ or $t_{BP}$ if only programming a single byte.

The three address bytes and at least one complete byte of data must be clocked into the device before the $\overline{\text{CS}}$ pin is deasserted, and the $\overline{\text{CS}}$ pin must be deasserted on even byte boundaries (multiples of eight bits); otherwise, the device will abort the operation and no data will be programmed into the memory array. In addition, if the address specified by A23-A0 points to a memory location within a sector that is in the protected state (see "Protect Sector" on page 26) or locked down (see "Sector Lockdown" on page 32), then the Byte/Page Program command will not be executed and the device will return to the idle state once the $\overline{\text{CS}}$ pin has been deasserted. The WEL bit in the Status Register will be reset back to the Logical 0 state if the program cycle aborts due to an incomplete address being sent, an incomplete byte of data being sent, the $\overline{\text{CS}}$ pin being deasserted on non-byte boundaries, or because the memory location to be programmed is protected or locked down.

While the device is programming, the Status Register can be read and will indicate that the device is busy. For faster throughput, it is recommended that the Status Register be polled rather than waiting the $t_{BP}$ or $t_{PP}$ time to determine if the data bytes have finished programming.  At some point before the program cycle completes, the WEL bit in the Status Register will be reset back to the Logical 0 state.

The device also incorporates an intelligent programming algorithm that can detect when a byte location fails to program properly. If a programming error arises, it will be indicated by the EPE bit in the Status Register.

**Figure 8-1. Byte Program**

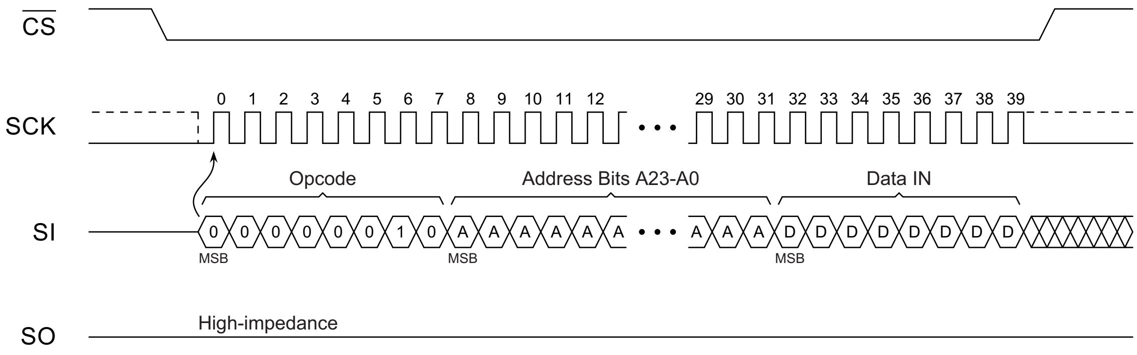
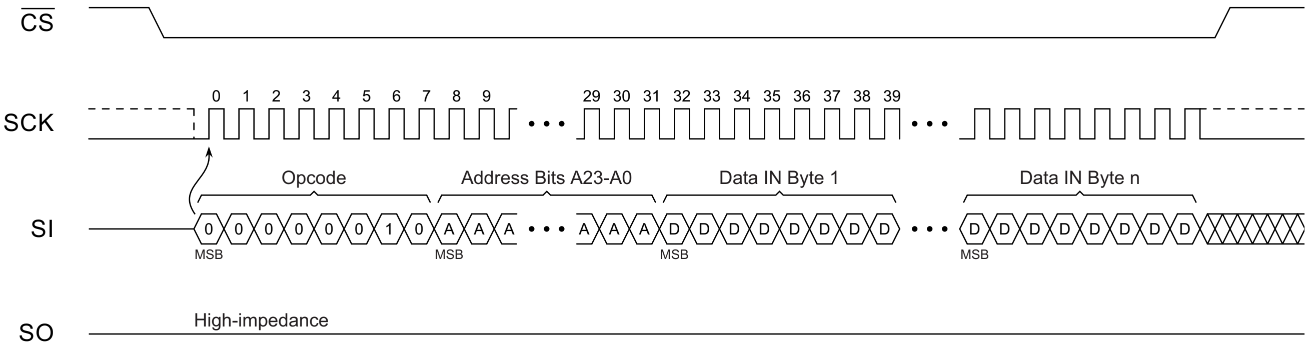


**Figure 8-2. Page Program**



## 8.2 Dual-Input Byte/Page Program

The Dual-Input Byte/Page Program command is similar to the standard Byte/Page Program command and can be used to program anywhere from a single byte of data up to 256 bytes of data into previously erased memory locations. Unlike the standard Byte/Page Program command, the Dual-Input Byte/Page Program command allows two bits of data to be clocked into the device on every clock cycle rather than just one.

Before the Dual-Input Byte/Page Program command can be started, the Write Enable command must have been previously issued to the device (see "Write Enable" on page 24) to set the Write Enable Latch (WEL) bit of the Status Register to a Logical 1 state. To perform a Dual-Input Byte/Page Program command, an opcode of A2h must be clocked into the device followed by the three address bytes denoting the first byte location of the memory array to begin programming at. After the address bytes have been clocked in, data can then be clocked into the device two bits at a time on both the $I/O_1$ and $I/O_0$ pins.

The data is always input with the MSB of a byte first, and the MSB is always input on the $I/O_1$ pin. During the first clock cycle, bit 7 of the first data byte would be input on the $I/O_1$ pin while bit 6 of the same data byte would be input on the $I/O_0$ pin. During the next clock cycle, bits 5 and 4 of the first data byte would be input on the $I/O_1$ and $I/O_0$ pins respectively. The sequence would continue with each byte of data being input after every four clock cycles. Like the standard Byte/Page Program command, all data clocked into the device is stored in an internal buffer.

If the starting memory address denoted by A23-A0 does not fall on an even 256-byte page boundary (A7-A0 are not all 0), then special circumstances regarding which memory locations to be programmed will apply. In this situation, any data that is sent to the device that goes beyond the end of the page will wrap around back to the beginning of the same page.

**Example:** If the starting address denoted by A23-A0 is 0000FEh and three bytes of data are sent to the device, then the first two bytes of data will be programmed at addresses 0000FEh and 0000FFh while the last byte of data will be programmed at address 000000h. The remaining bytes in the page (addresses 000001h through 0000FDh) will not be programmed and will remain in the erased state (FFh). In addition, if more than 256 bytes of data are sent to the device, then only the last 256 bytes sent will be latched into the internal buffer.

When the $\overline{CS}$ pin is deasserted, the device will take the data stored in the internal buffer and program it into the appropriate memory array locations based on the starting address specified by A23-A0 and the number of data bytes sent to the device. If less than 256 bytes of data were sent to the device, then the remaining bytes within the page will not be programmed and will remain in the erased state (FFh). The programming of the data bytes is internally self-timed and should take place in a time of $t_{PP}$ or $t_{BP}$ if only programming a single byte.

The three address bytes and at least one complete byte of data must be clocked into the device before the $\overline{CS}$ pin is deasserted and the $\overline{CS}$ pin must be deasserted on even byte boundaries (multiples of eight bits); otherwise, the device will abort the operation and no data will be programmed into the memory array. In addition, if the address specified by A23-A0 points to a memory location within a sector that is in the protected state (see "Protect Sector" on page 26) or locked down (see "Sector Lockdown" on page 32), then the Byte/Page Program command will not be executed and the device will return to the idle state once the $\overline{CS}$ pin has been deasserted. The WEL bit in the Status Register will be reset back to the Logical 0 state if the program cycle aborts due to an incomplete address being sent, an incomplete byte of data being sent, the $\overline{CS}$ pin being deasserted on non-byte boundaries or because the memory location to be programmed is protected or locked down.

While the device is programming, the Status Register can be read and will indicate that the device is busy. For faster throughput, it is recommended that the Status Register be polled rather than waiting the $t_{BP}$ or $t_{PP}$ time to determine if the data bytes have finished programming. At some point before the program cycle completes, the WEL bit in the Status Register will be reset back to the Logical 0 state.

The device also incorporates an intelligent programming algorithm that can detect when a byte location fails to program properly. If a programming error arises, it will be indicated by the EPE bit in the Status Register.

**Figure 8-3. Dual-Input Byte Program**

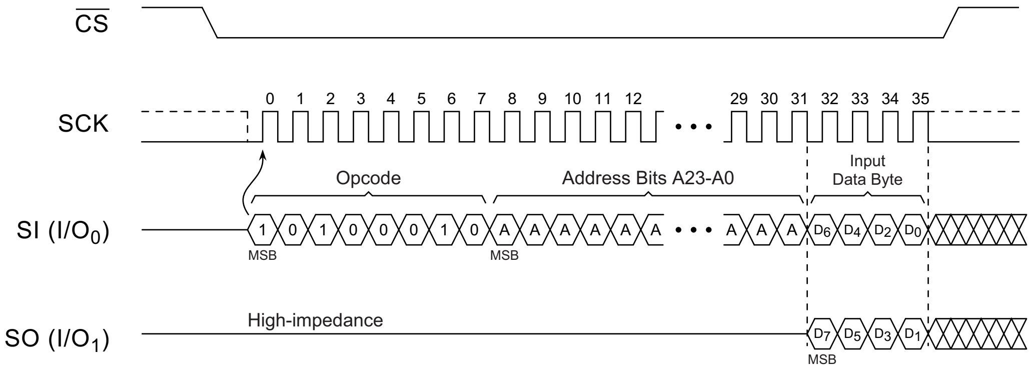
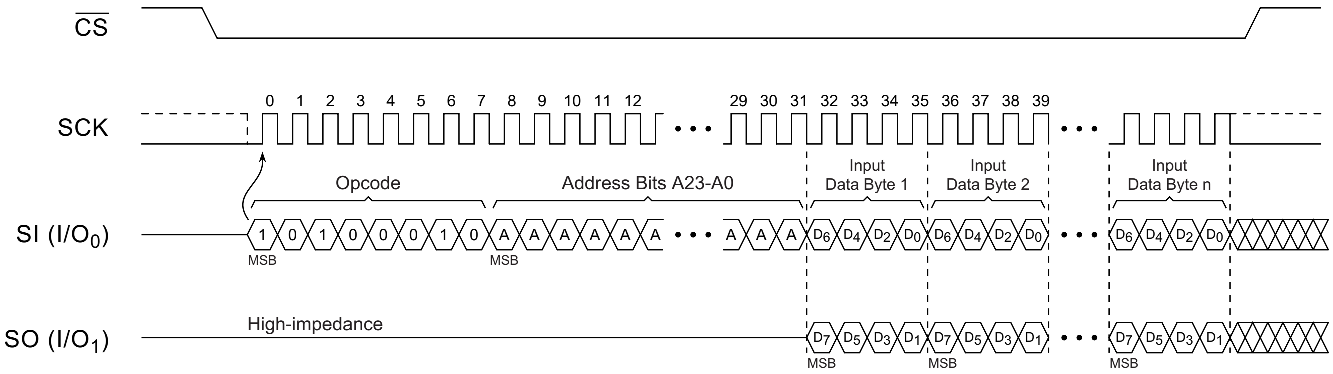


**Figure 8-4. Dual-Input Page Program**



## 8.3 Quad-Input Byte/Page Program

The Quad-Input Byte/Page Program command is similar to the Dual-Input Byte/Page Program command and can be used to program anywhere from a single byte of data up to 256 bytes of data into previously erased memory locations. Unlike the Dual-Input Byte/Page Program command, the Quad-Input Byte/Page Program command allows four bits of data to be clocked into the device on every clock cycle rather than two.

Before the Quad-Input Byte/Page Program command can be started, the Write Enable command must have been previously issued to the device (See “Write Enable” on page 24) to set the Write Enable Latch (WEL) bit of the Status Register to a Logical 1 state. To perform a Quad-Input Byte/Page Program command, an opcode of 32h must be clocked into the device followed by the three address bytes denoting the first byte location of the memory array to begin programming at. After the address bytes have been clocked in, data can then be clocked into the device four bits at a time on the $I/O_{3\text{-}0}$ pins.

The data is always input with the MSB of a byte first, and the MSB is always input on the $I/O_3$ pin. During the first clock cycle, bit 7 of the first data byte would be input on the $I/O_3$ pin while bits 6, 5, and 4 of the same data byte would be input on the $I/O_2$, $I/O_1$, and $I/O_0$ pins, respectively. During the next clock cycle, bits 3, 2, 1, and 0 of the first data byte would be input on the $I/O_3$, $I/O_2$, $I/O_1$, and $I/O_0$ pins, respectively. The sequence would continue with each byte of data being input after every two clock cycles. Like the standard Byte/Page Program and Dual-Input Byte/Page Program commands, all data clocked into the device is stored in an internal buffer.

If the starting memory address denoted by A23-A0 does not fall on a 256-byte page boundary (A7-A0 are not all 0), then special circumstances regarding which memory locations to be programmed will apply. In this situation, any data that is sent to the device that goes beyond the end of the page will wrap around back to the beginning of the same page.

**Example:** If the starting address denoted by A23-A0 is 0000FEh and three bytes of data are sent to the device, then the first two bytes of data will be programmed at addresses 0000FEh and 0000FFh while the last byte of data will be programmed at address 000000h. The remaining bytes in the page (addresses 000001h through 0000FDh) will not be programmed and will remain in the erased state (FFh). In addition, if more than 256 bytes of data are sent to the device, then only the last 256 bytes sent will be latched into the internal buffer.

When the $\overline{\text{CS}}$ pin is deasserted, the device will take the data stored in the internal buffer and program it into the appropriate memory array locations based on the starting address specified by A23-A0 and the number of data bytes sent to the device. If less than 256 bytes of data were sent to the device, then the remaining bytes within the page will not be programmed and will remain in the erased state (FFh). The programming of the data bytes is internally self-timed and should take place in a time of $t_{PP}$ or $t_{BP}$ if only programming a single byte.

The three address bytes and at least one complete byte of data must be clocked into the device before the $\overline{\text{CS}}$ pin is deasserted and the $\overline{\text{CS}}$ pin must be deasserted on byte boundaries (multiples of eight bits); otherwise, the device will abort the operation and no data will be programmed into the memory array. In addition, if the address specified by A23-A0 points to a memory location within a sector that is in the protected state (See “Protect Sector” on page 26) or locked down (See “Sector Lockdown” on page 32), then the Quad-Input Byte/Page Program command will not be executed and the device will return to the idle state once the $\overline{\text{CS}}$ pin has been deasserted. The WEL bit in the Status Register will be reset back to the Logical 0 state if the program cycle aborts due to an incomplete address being sent, an incomplete byte of data being sent, the $\overline{\text{CS}}$ pin being deasserted on non-byte boundaries or because the memory location to be programmed is protected or locked down.

While the device is programming, the Status Register can be read and will indicate that the device is busy. For faster throughput, it is recommended that the Status Register be polled rather than waiting the $t_{BP}$ or $t_{PP}$ time to determine if the data bytes have finished programming. At some point before the program cycle completes, the WEL bit in the Status Register will be reset back to the Logical 0 state.

The device also incorporates an intelligent programming algorithm that can detect when a byte location fails to program properly. If a programming error arises, it will be indicated by the EPE bit in the Status Register.

**Figure 8-5. Quad-Input Byte Program**

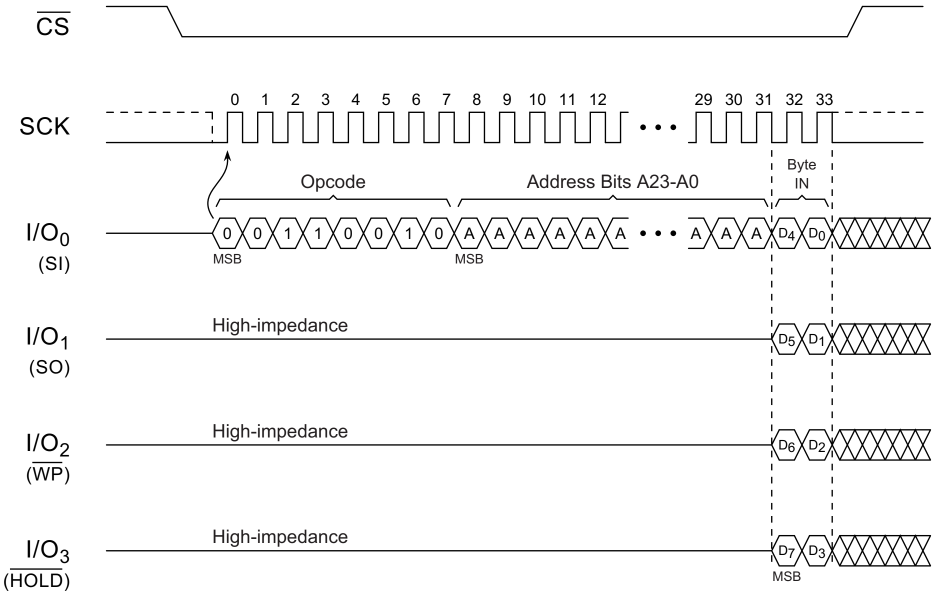
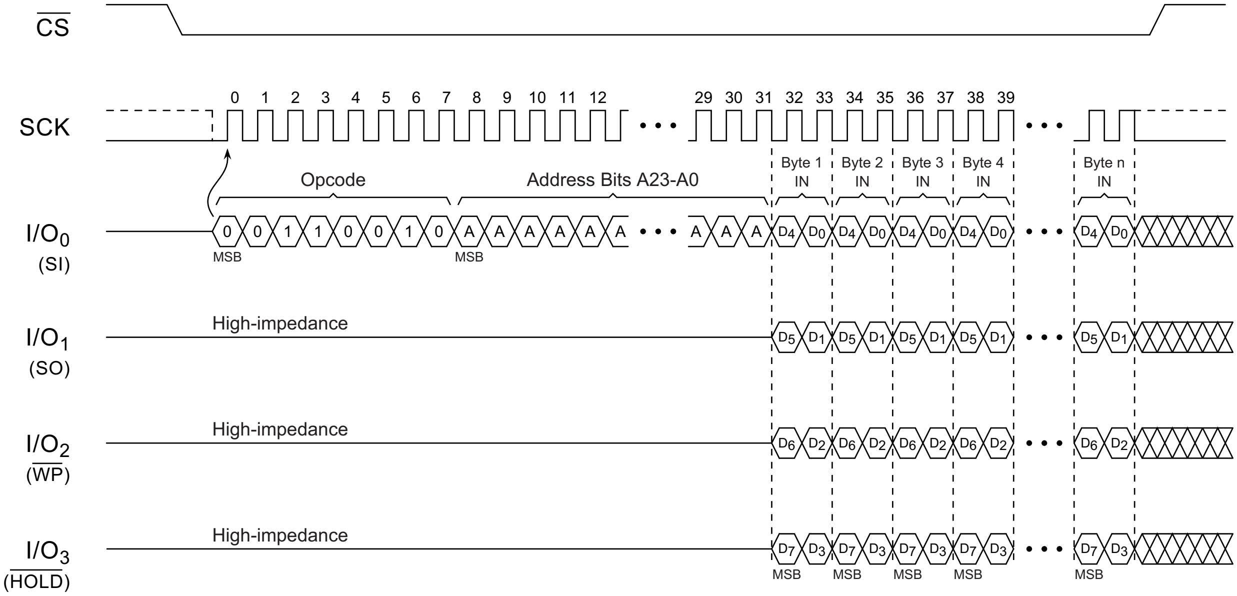


**Figure 8-6. Quad-Input Page Program**



## 8.4 Block Erase

A block of 4, 32, or 64 KB can be erased (all bits set to the Logical 1 state) in a single operation by using one of three different opcodes for the Block Erase command. An opcode of 20h is used for a 4KB erase, an opcode of 52h is used for a 32KB erase and an opcode of D8h is used for a 64KB erase. Before a Block Erase command can be started, the Write Enable command must have been previously issued to the device to set the WEL bit of the Status Register to a Logical 1 state.

To perform a Block Erase, the $\overline{\text{CS}}$ pin must first be asserted and the appropriate opcode (20h, 52h, or D8h) must be clocked into the device. After the opcode has been clocked in, the three address bytes specifying an address within the 4KB, 32KB, or 64KB block to be erased must be clocked in. Any additional data clocked into the device will be ignored. When the $\overline{\text{CS}}$ pin is deasserted, the device will erase the appropriate block. The erasing of the block is internally self-timed and should take place in a time of $t_{BLKE}$.

Since the Block Erase command erases a region of bytes, the lower order address bits do not need to be decoded by the device. Therefore, for a 4KB erase, address bits A11-A0 will be ignored by the device and their values can be either a Logical 1 or 0. For a 32KB erase, address bits A14-A0 will be ignored and for a 64KB erase, address bits A15-A0 will be ignored by the device. Despite the lower order address bits not being decoded by the device, the complete three address bytes must still be clocked into the device before the $\overline{\text{CS}}$ pin is deasserted and the $\overline{\text{CS}}$ pin must be deasserted on a byte boundary (multiples of eight bits); otherwise, the device will abort the operation and no erase operation will be performed.

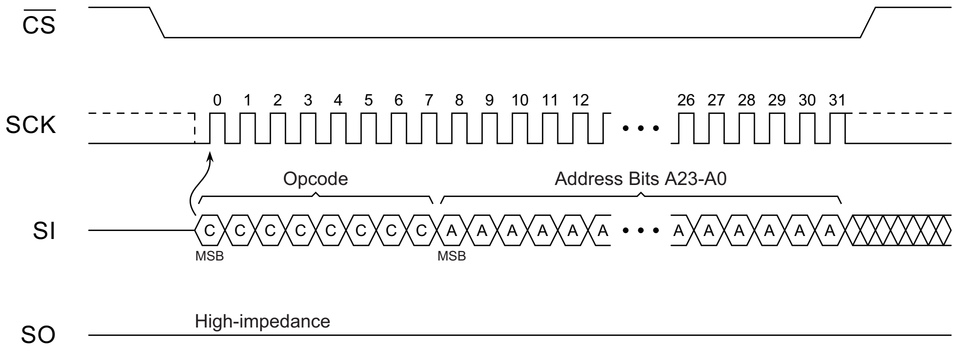
If the address specified by A23-A0 points to a memory location within a sector that is in the protected or locked down state, then the Block Erase command will not be executed and the device will return to the idle state once the $\overline{\text{CS}}$ pin has been deasserted.

The WEL bit in the Status Register will be reset back to the Logical 0 state if the erase cycle aborts due to an incomplete address being sent, the $\overline{\text{CS}}$ pin being deasserted on non-byte boundaries or because a memory location within the region to be erased is protected or locked down.

While the device is executing a successful erase cycle, the Status Register can be read and will indicate that the device is busy. For faster throughput, it is recommended that the Status Register be polled rather than waiting the $t_{BLKE}$ time to determine if the device has finished erasing. At some point before the erase cycle completes, the WEL bit in the Status Register will be reset back to the Logical 0 state.

The device also incorporates an intelligent erase algorithm that can detect when a byte location fails to erase properly. If an erase error occurs, it will be indicated by the EPE bit in the Status Register.

**Figure 8-7. Block Erase**



## 8.5 Chip Erase

The entire memory array can be erased in a single operation by using the Chip Erase command. Before a Chip Erase command can be started, the Write Enable command must have been previously issued to the device to set the WEL bit of the Status Register to a Logical 1 state.

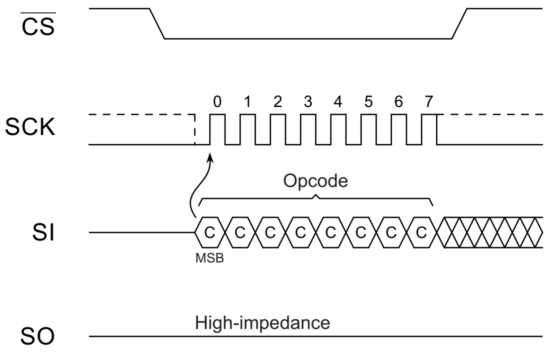
Two opcodes, 60h and C7h, can be used for the Chip Erase command. There is no difference in device functionality when utilizing the two opcodes, so they can be used interchangeably. To perform a Chip Erase, one of the two opcodes (60h or C7h) must be clocked into the device. Since the entire memory array is to be erased, no address bytes need to be clocked into the device and any data clocked in after the opcode will be ignored. When the $\overline{CS}$ pin is deasserted, the device will erase the entire memory array. The erasing of the device is internally self-timed and should take place in a time of $t_{CHPE}$.

The complete opcode must be clocked into the device before the $\overline{CS}$ pin is deasserted and the $\overline{CS}$ pin must be deasserted on a byte boundary (multiples of eight bits); otherwise, no erase will be performed. In addition, if any sector of the memory array is in the protected or locked down state, then the Chip Erase command will not be executed and the device will return to the idle state once the $\overline{CS}$ pin has been deasserted. The WEL bit in the Status Register will be reset back to the Logical 0 state if the $\overline{CS}$ pin is deasserted on non-byte boundaries or if a sector is in the protected or locked down state.

While the device is executing a successful erase cycle, the Status Register can be read and will indicate that the device is busy. For faster throughput, it is recommended that the Status Register be polled rather than waiting the $t_{CHPE}$ time to determine if the device has finished erasing. At some point before the erase cycle completes, the WEL bit in the Status Register will be reset back to the Logical 0 state.

The device also incorporates an intelligent erase algorithm that can detect when a byte location fails to erase properly. If an erase error occurs, it will be indicated by the EPE bit in the Status Register.

**Figure 8-8. Chip Erase**



## 8.6 Program/Erase Suspend

In some code plus data storage applications, it is often necessary to process certain high-level system interrupts that require relatively immediate reading of code or data from the Flash memory. In such an instance, it may not be possible for the system to wait the microseconds or milliseconds required for the Flash memory to complete a program or erase cycle. The Program/Erase Suspend command allows a program or erase operation in progress to a particular 64KB sector of the Flash memory array to be suspended so that other device operations can be performed.

**Example:** By suspending an erase operation to a particular sector, the system can perform functions such as a program or read operation within another 64KB sector in the device. Other device operations, such as a Read Status Register, can also be performed while a program or erase operation is suspended. Table 8-1 outlines the operations that are allowed and not allowed during a program or erase suspend.

Since the need to suspend a program or erase operation is immediate, the Write Enable command does not need to be issued prior to the Program/Erase Suspend command being issued. Therefore, the Program/Erase Suspend command operates independently of the state of the WEL bit in the Status Register.

To perform a Program/Erase Suspend, the $\overline{\text{CS}}$ pin must first be asserted and the opcode of B0h must be clocked into the device. No address bytes need to be clocked into the device and any data clocked in after the opcode will be ignored. When the $\overline{\text{CS}}$ pin is deasserted, the program or erase operation currently in progress will be suspended within a time of $t_{SUSP}$. The Program Suspend (PS) bit or the Erase Suspend (ES) bit in the Status Register will then be set to the Logical 1 state to indicate that the program or erase operation has been suspended. In addition, the RDY/BSY bit in the Status Register will indicate that the device is ready for another operation. The complete opcode must be clocked into the device before the $\overline{\text{CS}}$ pin is deasserted and the $\overline{\text{CS}}$ pin must be deasserted on non-byte boundary (multiples of eight bits); otherwise, no suspend operation will be performed.

If the Program/Erase Suspend command is issued while the device is not busy performing a program or erase operation, (CRDY/BSY is in the Logic 1 state), then the device is simply ignore the Program/Erase Suspend command and the PS and ES bits in the status Register will indicate that the device is not in a suspend state, (PS and ES will be a Logic 0).

Read operations are not allowed to a 64KB sector that has had its program or erase operation suspended. If a read is attempted to a suspended sector, then the device will output undefined data. Therefore, when performing a Read Array operation to an unsuspended sector and the device's internal address counter increments and crosses the sector boundary to a suspended sector, the device will then start outputting undefined data continuously until the address counter increments and crosses a sector boundary to an unsuspended sector.

A program operation is not allowed to a sector that has been erase suspended. If a program operation is attempted to an erase suspended sector, then the program operation will abort and the WEL bit in the Status Register will be reset back to the Logical 0 state. Likewise, an erase operation is not allowed to a sector that has been program suspended. If attempted, the erase operation will abort and the WEL bit in the Status Register will be reset to a Logical 0 state.

During an Erase Suspend, a program operation to a different 64KB sector can be started and subsequently suspended. This results in a simultaneous Erase Suspend/Program Suspend condition and will be indicated by the states of both the ES and PS bits in the Status Register being set to the Logical 1 state.

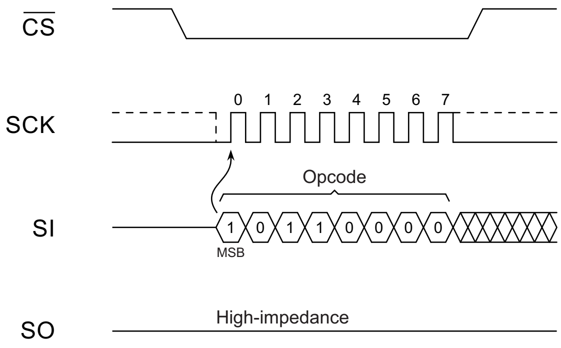
If a Reset operation (see "Reset" on page 48) is performed while a sector is erase suspended, the suspend operation will abort and the contents of the block in the suspended sector will be left in an undefined state. However, if a Reset is performed while a sector is program suspended, the suspend operation will abort but only the contents of the page that was being programmed and subsequently suspended will be undefined. The remaining pages in the 64KB sector will retain their previous contents.

If an attempt is made to perform an operation that is not allowed during a program or erase suspend, such as a Protect Sector operation, then the device will simply ignore the opcode and no operation will be performed. The state of the WEL bit in the Status Register, as well as the SPRL (Sector Protection Registers Locked) and SLE (Sector Lockdown Enabled) bits, will not be affected.

Table 8-1. Operations Allowed and Not Allowed During a Program or Erase Suspend

| Command | Operation During Program Suspend | Operation During Erase Suspend |
|---|---|---|
| **Read Commands** | | |
| Read Array (All Opcodes) | Allowed | Allowed |
| **Program and Erase Commands** | | |
| Block Erase | Not Allowed | Not Allowed |
| Chip Erase | Not Allowed | Not Allowed |
| Byte/Page Program (All Opcodes) | Not Allowed | Allowed |
| Program/Erase Suspend | Not Allowed | Allowed |
| Program/Erase Resume | Allowed | Allowed |
| **Protection Commands** | | |
| Write Enable | Not Allowed | Allowed |
| Write Disable | Not Allowed | Allowed |
| Protect Sector | Not Allowed | Not Allowed |
| Unprotect Sector | Not Allowed | Not Allowed |
| Global Protect/Unprotect | Not Allowed | Not Allowed |
| Read Sector Protection Registers | Allowed | Allowed |
| **Security Commands** | | |
| Sector Lockdown | Not Allowed | Not Allowed |
| Freeze Sector Lockdown State | Not Allowed | Not Allowed |
| Read Sector Lockdown Registers | Allowed | Allowed |
| Program OTP Security Register | Not Allowed | Not Allowed |
| Read OTP Security Register | Allowed | Allowed |
| **Status and Configuration Register Commands** | | |
| Read Status Register | Allowed | Allowed |
| Write Status Register (All Opcodes) | Not Allowed | Not Allowed |
| Read Configuration Register | Allowed | Allowed |
| Write Configuration Register | Not Allowed | Not Allowed |
| **Miscellaneous Commands** | | |
| Reset | Allowed | Allowed |
| Read Manufacturer and Device ID | Allowed | Allowed |
| Deep Power-Down | Not Allowed | Not Allowed |
| Resume from Deep Power-Down | Not Allowed | Not Allowed |

Figure 8-9. Program/Erase Suspend



## 8.7 Program/Erase Resume

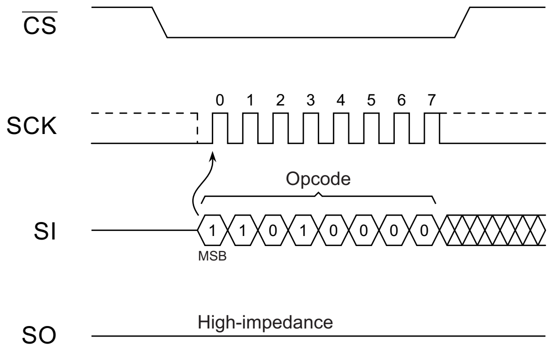
The Program/Erase Resume command allows a suspended program or erase operation to be resumed and continue programming a Flash page or erasing a Flash memory block where it left off. As with the Program/Erase Suspend command, the Write Enable command does not need to be issued prior to the Program/Erase Resume command being issued. Therefore, the Program/Erase Resume command operates independently of the state of the WEL bit in the Status Register.

To perform a Program/Erase Resume, the $\overline{CS}$ pin must first be asserted and the opcode of D0h must be clocked into the device. No address bytes need to be clocked into the device and any data clocked in after the opcode will be ignored. When the $\overline{CS}$ pin is deasserted, the program or erase operation currently suspended will be resumed within a time of $t_{RES}$. The PS bit or the ES bit in the Status Register will then be reset back to the Logical 0 state to indicate that the program or erase operation is no longer suspended. In addition, the RDY/BSY bit in the Status Register will indicate that the device is busy performing a program or erase operation. The complete opcode must be clocked into the device before the $\overline{CS}$ pin is deasserted and the $\overline{CS}$ pin must be deasserted on a byte boundary (multiples of eight bits); otherwise, no resume operation will be performed.

During a simultaneous Erase Suspend/Program Suspend condition, issuing the Program/Erase Resume command will result in the program operation resuming first. After the program operation has been completed, the Program/Erase Resume command must be issued again in order for the erase operation to be resumed.

While the device is busy resuming a program or erase operation, any attempts at issuing the Program/Erase Suspend command will be ignored. Therefore, if a resumed program or erase operation needs to be subsequently suspended again, the system must either wait the entire $t_{RES}$ time before issuing the Program/Erase Suspend command, or it must check the status of the RDY/BSY bit or the appropriate PS or ES bit in the Status Register to determine if the previously suspended program or erase operation has resumed.

**Figure 8-10. Program/Erase Resume**



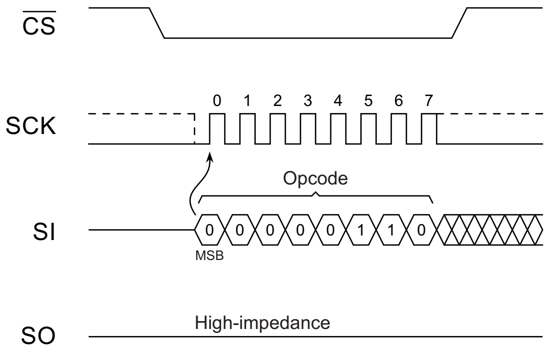
# 9. Protection Commands and Features

## 9.1 Write Enable

The Write Enable command is used to set the Write Enable Latch (WEL) bit in the Status Register to a Logical 1 state. The WEL bit must be set before a Byte/Page Program, Erase, Protect Sector, Unprotect Sector, Sector Lockdown, Freeze Sector Lockdown State, Program OTP Security Register, Write Status Register, or Write Configuration Register command can be executed. This makes the issuance of these commands a two step process, thereby reducing the chances of a command being accidentally or erroneously executed. If the WEL bit in the Status Register is not set prior to the issuance of one of these commands, then the command will not be executed.

To issue the Write Enable command, the $\overline{\text{CS}}$ pin must first be asserted and the opcode of 06h must be clocked into the device. No address bytes need to be clocked into the device and any data clocked in after the opcode will be ignored. When the $\overline{\text{CS}}$ pin is deasserted, the WEL bit in the Status Register will be set to a Logical 1. The complete opcode must be clocked into the device before the $\overline{\text{CS}}$ pin is deasserted and the $\overline{\text{CS}}$ pin must be deasserted on an even byte boundary (multiples of eight bits); otherwise, the device will abort the operation and the state of the WEL bit will not change.

**Figure 9-1. Write Enable**

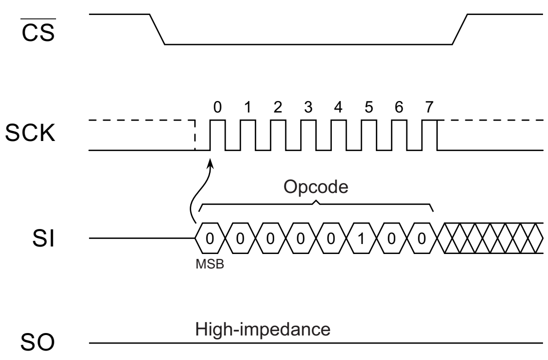


## 9.2 Write Disable

The Write Disable command is used to reset the Write Enable Latch (WEL) bit in the Status Register to the Logical 0 state. With the WEL bit reset, all Byte/Page Program, Erase, Protect Sector, Unprotect Sector, Sector Lockdown, Freeze Sector Lockdown State, Program OTP Security Register, Write Status Register, and Write Configuration Register commands will not be executed. Other conditions can also cause the WEL bit to be reset; for more details, refer to the WEL bit section of the Status Register description.

To issue the Write Disable command, the $\overline{\text{CS}}$ pin must first be asserted and the opcode of 04h must be clocked into the device. No address bytes need to be clocked into the device and any data clocked in after the opcode will be ignored. When the $\overline{\text{CS}}$ pin is deasserted, the WEL bit in the Status Register will be reset to a Logical 0. The complete opcode must be clocked into the device before the $\overline{\text{CS}}$ pin is deasserted and the $\overline{\text{CS}}$ pin must be deasserted on a byte boundary (multiples of eight bits); otherwise, the device will abort the operation and the state of the WEL bit will not change.

**Figure 9-2. Write Disable**



$\overline{\text{CS}}$
SCK
0
1
2
3
4
5
6
7
Opcode
SI
0
0
0
0
0
1
0
0
MSB
SO
High-impedance