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

adesto® TECHNOLOGIES

AT25DF641A

64-Mbit, 2.7V Minimum SPI Serial Flash Memory with Dual–I/O Support

DATASHEET

Features

- Single 2.7V - 3.6V supply
- Serial Peripheral Interface (SPI) compatible
  - Supports SPI Modes 0 and 3
  - Supports RapidS™ operation
  - Supports Dual-Input Program and Dual-Output Read
- Very high operating frequencies
  - 100MHz for RapidS
  - 85MHz for SPI
  - Clock-to-output time (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
  - 128 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 One-Time Programmable (OTP) Security Register
  - 64 bytes factory preprogrammed
  - 64 bytes user programmable
- Flexible programming
  - Byte/Page Program (1 to 256 bytes)
- Fast program and erase times
  - 2.5ms typical Page Program (256 bytes) time
  - 75ms typical 4KB Block Erase time
  - 300ms typical 32KB Block Erase time
  - 600ms 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
  - 25mA 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 (0.208” wide)
  - 8-pad Ultra-thin DFN (5 x 6 x 0.6mm)

1. Description

The AT25DF641A is a 2.7V minimum, serial-interface Flash memory ideally suited for 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 AT25DF641A, 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 AT25DF641A 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 AT25DF641A 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 AT25DF641A 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 AT25DF641A 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, or other purposes.

Specifically designed for use in 3V systems, the AT25DF641A 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

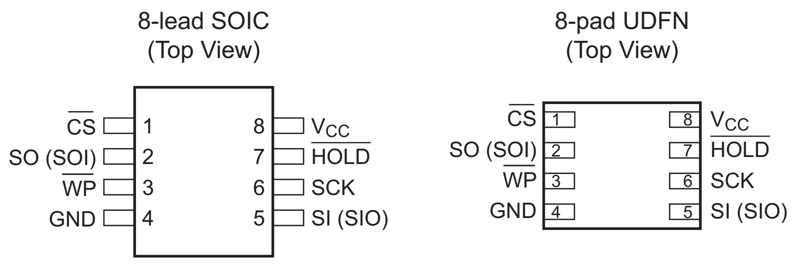
Table 2-1. Pin Descriptions

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. 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 is always latched in on the rising edge of SCK, while output data on the SO pin is always clocked out on the falling edge of SCK.	—	Input
SI (SIO)	**Serial Input (Serial Input/Output):** 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. With the Dual-Output Read Array command, the SI pin becomes an output pin (SIO) to allow two bits of data (on the SO and SIO pins) to be clocked out on every falling edge of SCK. To maintain consistency with SPI nomenclature, the SIO pin will be referenced as SI throughout the document with exception to sections dealing with the Dual-Output Read Array command in which it will be referenced as SIO. Data present on the SI pin will be ignored whenever the device is deselected ($\overline{CS}$ is deasserted).	—	Input/Output
SO (SOI)	**Serial Output (Serial Output/Input):** 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. With the Dual-Input Byte/Page Program command, the SO pin becomes an input pin (SOI) to allow two bits of data (on the SOI and SI pins) to be clocked in on every rising edge of SCK. To maintain consistency with SPI nomenclature, the SOI pin will be referenced as SO throughout the document with exception to sections dealing with the Dual-Input Byte/Page Program command in which it will be referenced as SOI. The SO pin will be in a high-impedance state whenever the device is deselected ($\overline{CS}$ is deasserted).	—	Output/Input
$\overline{WP}$	**Write Protect:** The $\overline{WP}$ pin controls the hardware locking feature of the device. Please refer to "Protection Commands and Features" on page 21 for more details on protection features and the $\overline{WP}$ pin. The $\overline{WP}$ pin is internally pulled-high and may be left floating if hardware controlled protection will not be used. However, it is recommended that the $\overline{WP}$ pin also be externally connected to V_{CC} whenever possible.	Low	Input

Table 2-1. Pin Descriptions (Continued)

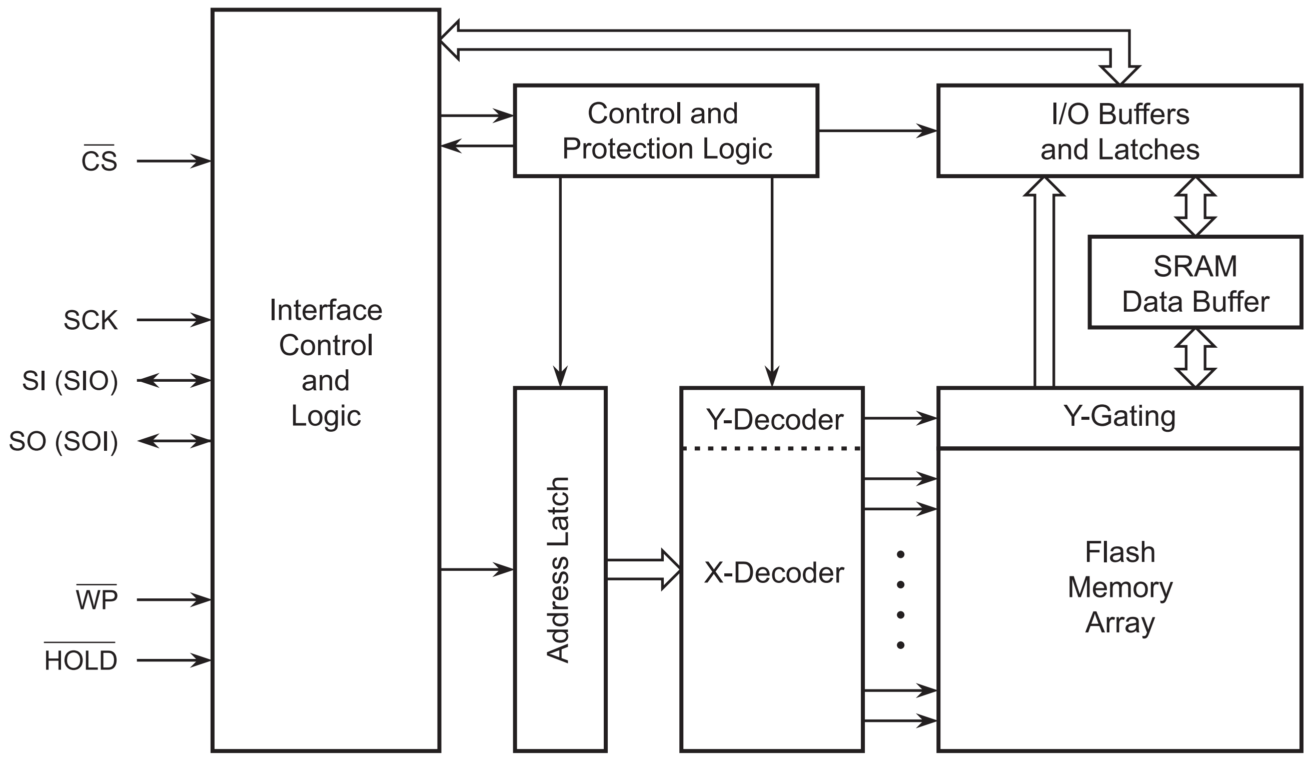
Symbol	Name and Function	Asserted State	Type
$\overline{\text{HOLD}}$	**Hold:** The $\overline{\text{HOLD}}$ pin is used to temporarily pause serial communication without deselecting or resetting the device. While the $\overline{\text{HOLD}}$ pin is asserted, transitions on the SCK pin and data on the SI pin will be ignored, and the SO pin will be in a high-impedance state. 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 effect on internally self-timed operations such as a program or erase cycle. See "Hold" on page 46 for additional details on the Hold operation. The $\overline{\text{HOLD}}$ pin is internally pulled-high and may be left floating if the Hold function will not be used. However, it is recommended that the $\overline{\text{HOLD}}$ pin also be externally connected to V_{CC} whenever possible.	Low	Input
V_{CC}	**Device Power Supply:** The V_{CC} pin is used to supply the source voltage to the device. 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. Pinouts



3. Block Diagram

Figure 3-1. Block Diagram

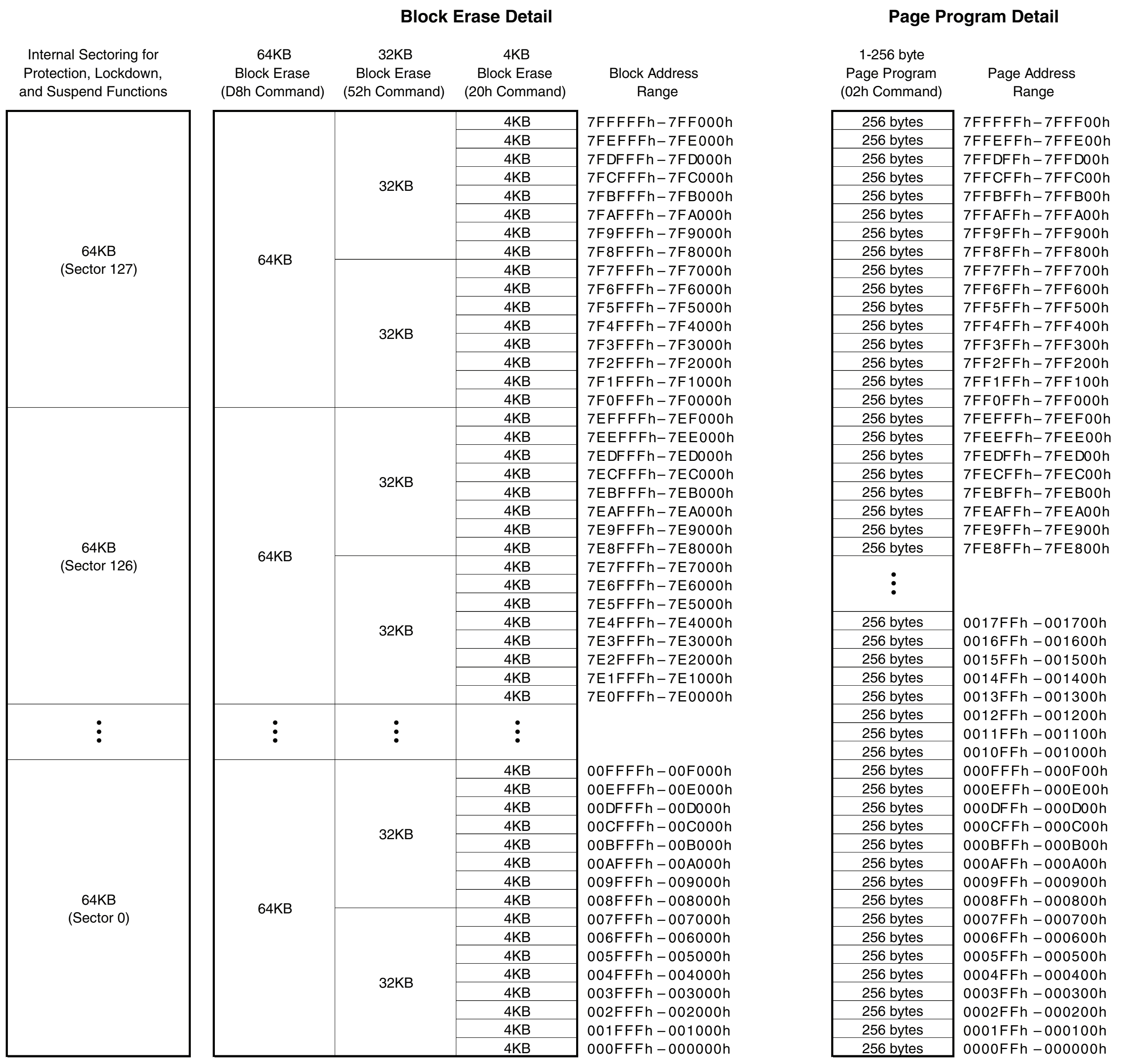


Note: SIO and SOI pin naming convention is used for Dual-I/O commands.

4. Memory Array

To provide the greatest flexibility, the AT25DF641A memory array 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, which 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



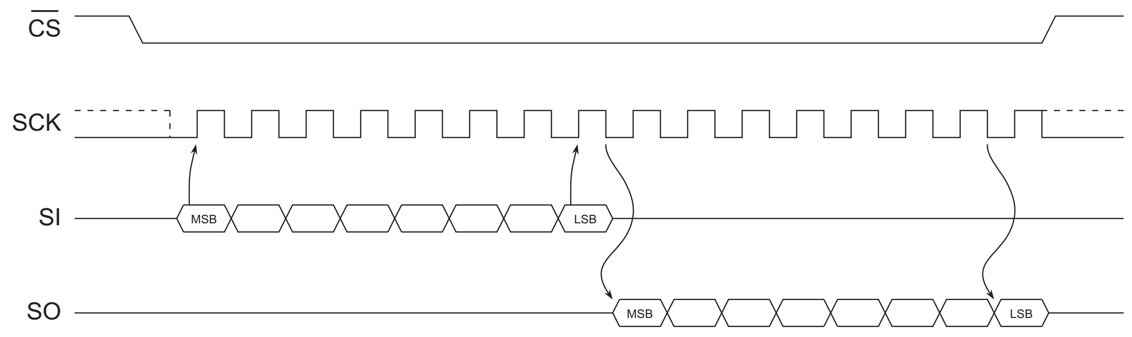
5. Device Operation

The AT25DF641A 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 AT25DF641A via the SPI bus which is comprised of four signal lines: Chip Select ($\overline{CS}$), Serial Clock (SCK), Serial Input (SI), and Serial Output (SO).

The AT25DF641A features a Dual-Input Program mode in which the SO pin becomes an input. Similarly, the device also features a Dual-Output Read mode in which the SI pin becomes an output. In the Dual-Input Byte/Page Program command description, the SO pin will be referred to as the SOI (Serial Output/Input) pin, and in the Dual-Output Read Array command, the SI pin will be referenced as the SIO (Serial Input/Output) pin.

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 AT25DF641A 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



6. Commands and Addressing

A valid instruction or operation must always be started by first asserting the $\overline{CS}$ pin. After the $\overline{CS}$ 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 $\overline{CS}$ pin.

Opcodes not supported by the AT25DF641A 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 ($\overline{CS}$ pin being deasserted and then reasserted). In addition, if the $\overline{CS}$ 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 AT25DF641A memory array is 7FFFFFh, address bit A23 is 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 40MHz	3	0	1+
Dual-Output Read Array	3Bh	0011 1011	Up to 65MHz	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+
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 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
Miscellaneous Commands						
Reset	F0h	1111 0000	Up to 100MHz	0	0	1
Read Manufacturer and Device ID	9Fh	1001 1111	Up to 85 MHz	0	0	1 to 4
Deep Power-Down	B9h	1011 1001	Up to 100 MHz	0	0	0
Resume from Deep Power-Down	ABh	1010 1011	Up to 100 MHz	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 then 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 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 (7FFFFFh) 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

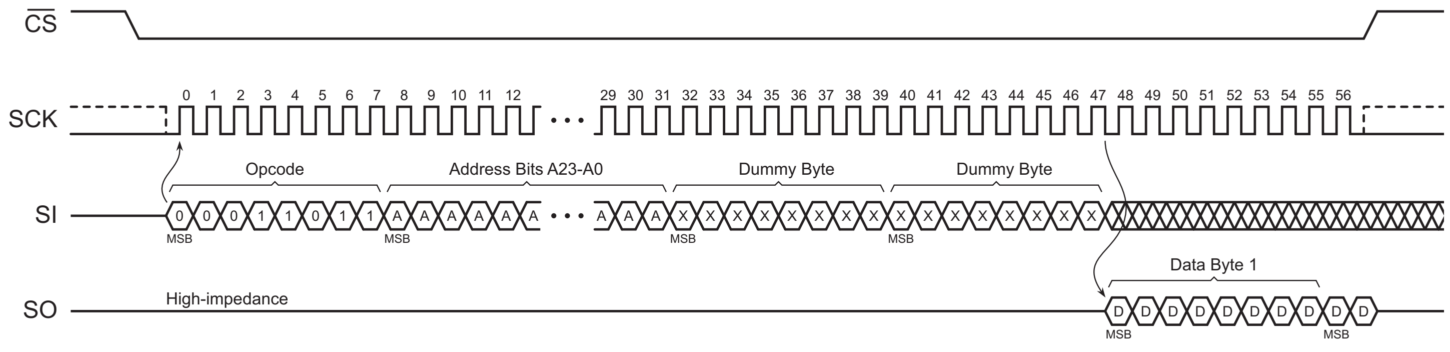


Figure 7-2. Read Array – 0Bh Opcode

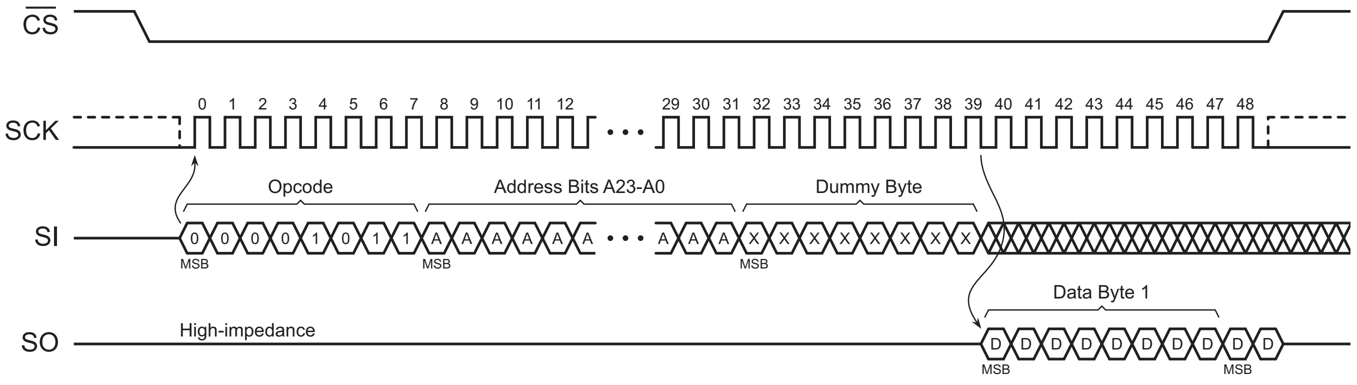
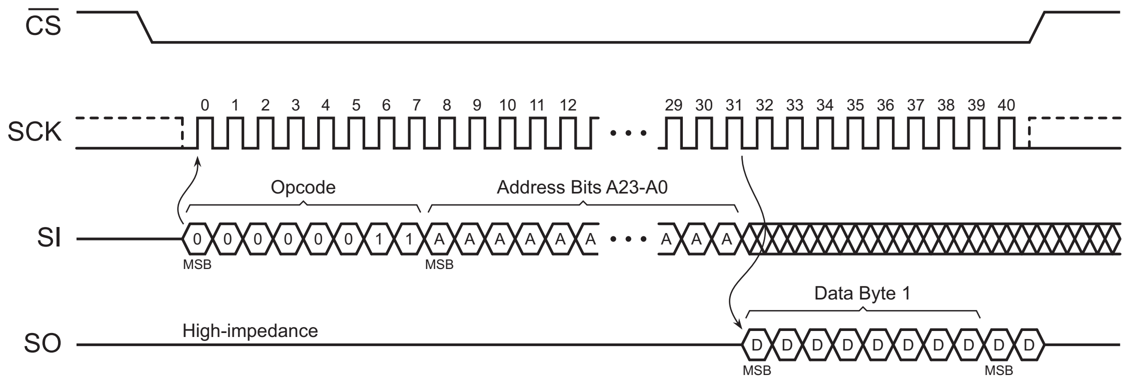


Figure 7-3. Read Array – 03h Opcode



7.2 Dual-Output Read Array

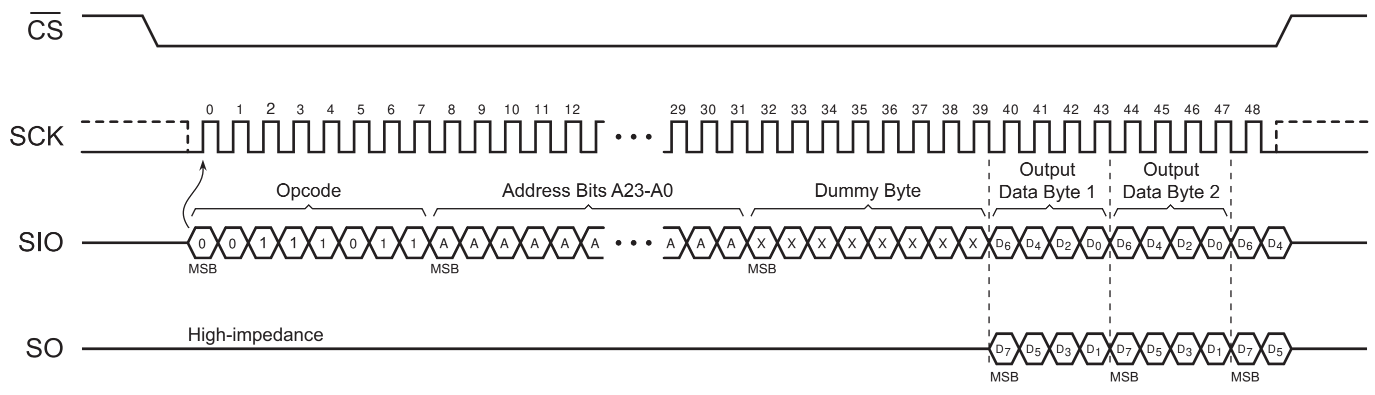
The Dual-Output Read Array command is similar to the standard Read Array command and can be used to sequentially read a continuous stream of data from the device by simply providing the clock signal once the initial starting address has been specified. Unlike the standard Read Array command, however, 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 then 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 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 SO and SIO pins. The data is always output with the MSB of a byte first, and the MSB is always output on the SO pin. During the first clock cycle, bit 7 of the first data byte is output on the SO pin, while bit 6 of the same data byte is output on the SIO pin. During the next clock cycle, bits 5 and 4 of the first data byte are output on the SO and SIO pins, respectively. The sequence continues with each byte of data being output after every four clock cycles. When the last byte (7FFFFFh) of the memory array has been read, the device will continue reading from 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 and SIO 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



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 21) to set the Write Enable Latch (WEL) bit of the Status Register to a Logical 1 state.

Note: The internal programming operation of the AT25DF641A occurs on a nibble-wide basis, and all four bits of the nibble must be in an erased state (Logic 1 state) prior to the programming of any of the four bits. If any one of the four bits is in the programmed state (Logic 0 state) and an attempt is made to program one of the other four bits in the nibble from a Logic 1 to a Logic 0, then the contents of the nibble cannot be guaranteed and may contain erroneous data.

Example 1: A 4KB Block Erase is performed to erase the first 4KB block of memory (all bytes set to FFh). The application then programs the first byte location with a value of 7Fh (0111 1111). Without erasing the 4KB block again, the application then attempts to program the same byte location with BFh (1011 1111) expecting that the resulting byte value stored in memory will be 3Fh (0011 1111). However, because of the way the AT25DF641A programs bytes internally and because all four bits of the most-significant nibble were not in the erased state prior to the programming of the BFh value, the most-significant nibble will not contain the value of 0011b but will instead contain a different value.

Example 2: A 4KB Block Erase is performed to erase the first 4KB block of memory (all bytes set to FFh). The application then programs the first byte location with a value of 7Fh (0111 1111). Without erasing the 4KB block again, the application then attempts to program the same byte location with FCh (1111 1100), expecting that the resulting byte value stored in memory will be 7Ch (0111 1100). The resulting byte value stored in the memory will indeed be 7Ch because no additional bits in the most-significant nibble were being programmed from a Logic 1 to a Logic 0, and all four bits in the least-significant nibble were in the erased state prior to the programming of the 7Ch byte value.

To perform a Byte/Page Program command, a 02h opcode must be clocked into the device followed by the three address bytes denoting the first 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 are 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 to the beginning of the same page. 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.

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).

When the $\overline{\text{CS}}$ pin is deasserted, the device will take the data stored in the internal buffer 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 23) or locked down (see "Sector Lockdown" on page 29), 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 uneven 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

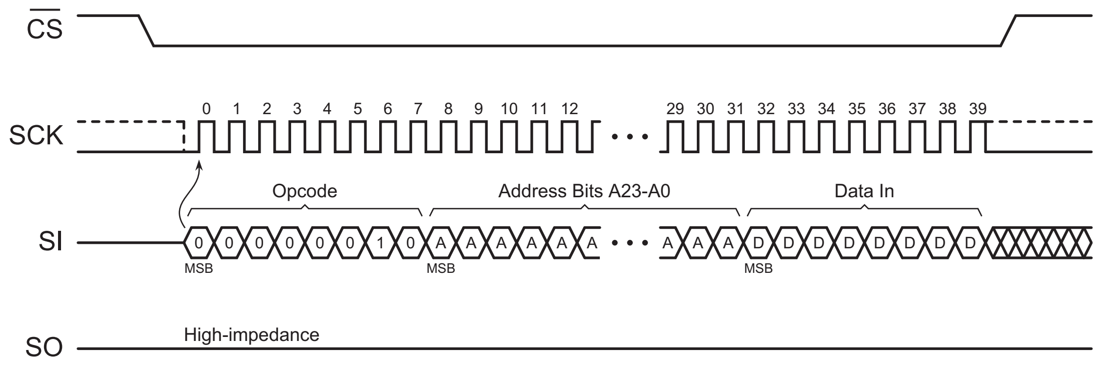
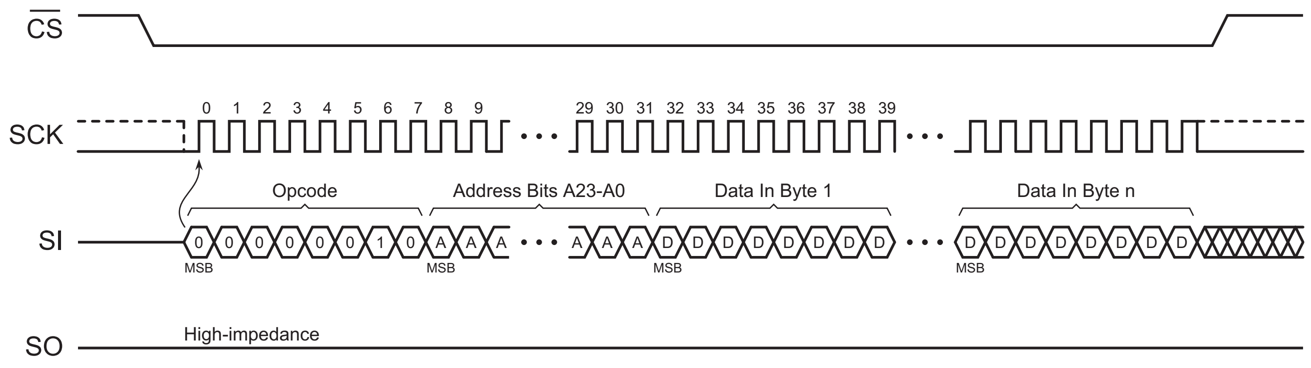


Figure 8-2. Page Program



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, however, 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.

Note: The internal programming operation of the AT25DF641A occurs on a nibble-wide basis, and all four bits of the nibble must be in an erased state (Logic 1 state) prior to the programming of any of the four bits. If any one of the four bits is in the programmed state (Logic 0 state) and an attempt is made to program one of the other four bits in the nibble from a Logic 1 to a Logic 0, then the contents of the nibble cannot be guaranteed and may contain erroneous data.

Example 1: A 4KB Block Erase is performed to erase the first 4KB block of memory (all bytes set to FFh). The application then programs the first byte location with a value of 7Fh (0111 1111). Without erasing the 4KB block again, the application then attempts to program the same byte location with BFh (1011 1111) expecting that the resulting byte value stored in memory will be 3Fh (0011 1111). However, because of the way the AT25DF641A programs bytes internally and because all four bits of the most-significant nibble were not in the erased state prior to the programming of the BFh value, the most-significant nibble will not contain the value of 0011b but will instead contain a different value.

Example 2: A 4KB Block Erase is performed to erase the first 4KB block of memory (all bytes set to FFh). The application then programs the first byte location with a value of 7Fh (0111 1111). Without erasing the 4KB block again, the application then attempts to program the same byte location with FCh (1111 1100), expecting that the resulting byte value stored in memory will be 7Ch (0111 1100). The resulting byte value stored in the memory will indeed be 7Ch because no additional bits in the most-significant nibble were being programmed from a Logic 1 to a Logic 0, and all four bits in the least-significant nibble were in the erased state prior to the programming of the 7Ch byte value.

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 21) 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 A2h opcode must be clocked into the device followed by the three address bytes denoting the first 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 SOI and SI pins.

The data is always input with the MSB of a byte first, and the MSB is always input on the SOI pin. During the first clock cycle, bit 7 of the first data byte is input on the SOI pin while bit 6 of the same data byte is input on the SI pin. During the next clock cycle, bits 5 and 4 of the first data byte are input on the SOI and SI pins, respectively. The sequence continues 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 are 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. 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.

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).

When the $\overline{\text{CS}}$ pin is deasserted, the device will program the data stored in the internal buffer 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 are 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 23) or locked down (see "Sector Lockdown" on page 29), 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 uneven 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

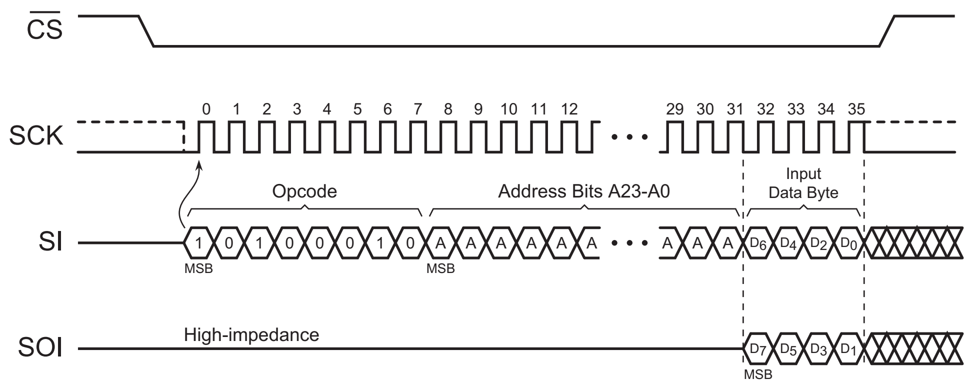
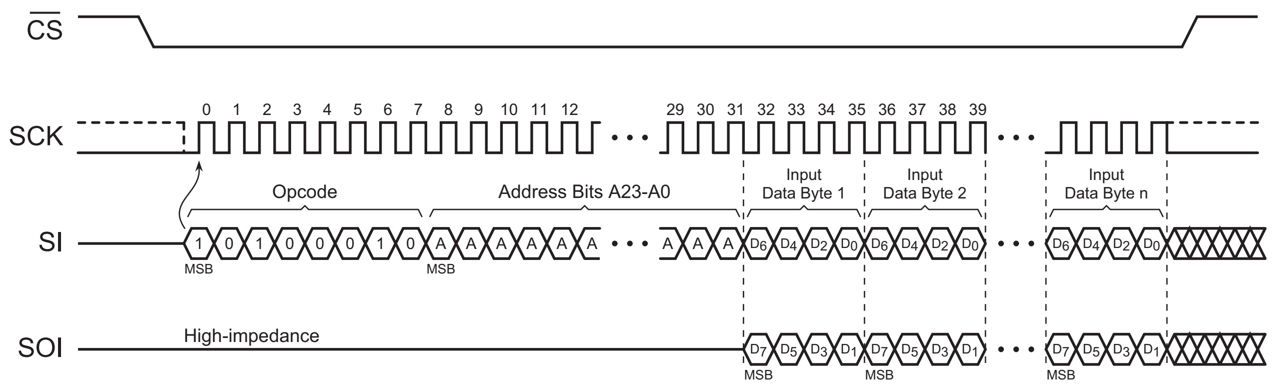


Figure 8-4. Dual-Input Page Program



8.3 Block Erase

A block of 4KB, 32KB, or 64KB 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. A 20h opcode is used for a 4KB erase, a 52h opcode is used for a 32KB erase, and a D8h opcode 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. 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 an even byte boundary (multiples of eight bits); otherwise, the device will abort the operation and no erase operation will be performed.

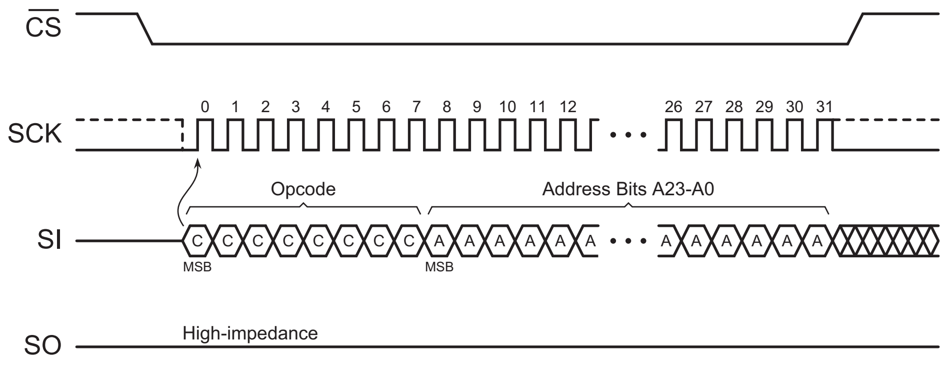
If the address specified by A23-A0 points to a memory location within a sector that is in the protected or locked down state, then the Block Erase command will not be executed, and the device will return to the idle state once the $\overline{\text{CS}}$ pin has been deasserted.

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 uneven 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 8-5. Block Erase



8.4 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.

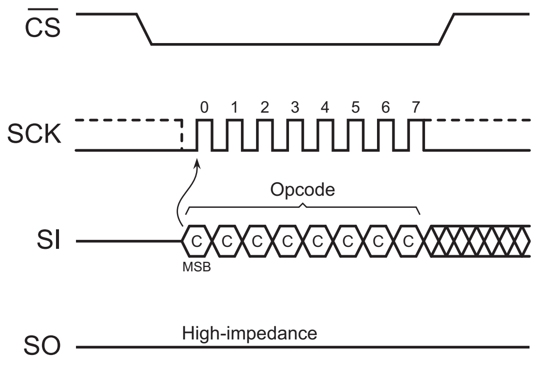
Two opcodes, 60h and C7h, can be used for the Chip Erase command. There is no difference in device functionality when utilizing the two opcodes, so they can be used interchangeably. To perform a Chip Erase, one of the two opcodes (60h or C7h) must be clocked into the device. Since the entire memory array 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 an even 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 uneven 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 8-6. Chip Erase



8.5 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.

By suspending an erase operation to a particular sector, the system can perform 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 while a program or erase operation is suspended.

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 then the opcode 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 an even byte boundary (multiples of eight bits); otherwise, no suspend operation will be performed.

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 on a sector that has been erase suspended. If a program operation is attempted on 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, which will be indicated by the ES and PS bits in the Status Register being set to the Logical 1 state.

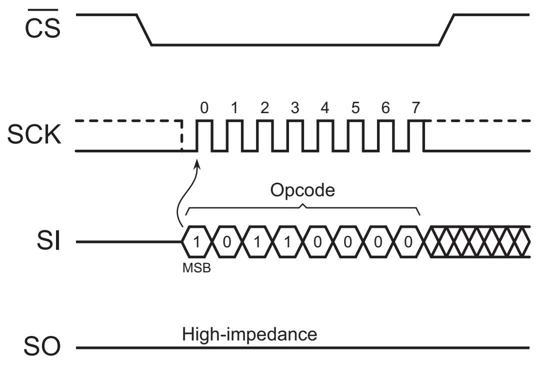
If a Reset operation (see "Reset" on page 41) is performed while a sector is erase suspended, the suspend operation will abort and the contents of the block in the suspended sector will be left in an undefined state. However, if a Reset is performed while a sector is program suspended, the suspend operation 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 while a program or erase operation is suspended, such as a Protect Sector command, 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 Register Commands		
Read Status Register	Allowed	Allowed
Write Status Register (All Opcodes)	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 8-7. **Program/Erase Suspend**



8.6 Program/Erase Resume

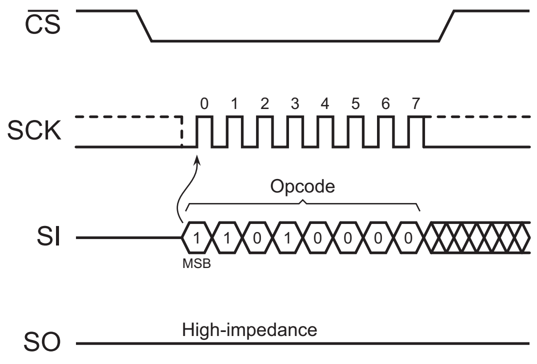
The Program/Erase Resume command allows a suspended program or erase operation to be resumed and continue programming a Flash page or erasing a Flash memory block where it left off. As with the Program/Erase Suspend command, the Write Enable command does not need to be issued prior to the Program/Erase 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 $\overline{\text{CS}}$ pin must first be asserted and then the opcode 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 $\overline{\text{CS}}$ 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 $\overline{\text{CS}}$ pin is deasserted, and the $\overline{\text{CS}}$ pin must be deasserted on an even 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 8-8. Program/Erase Resume



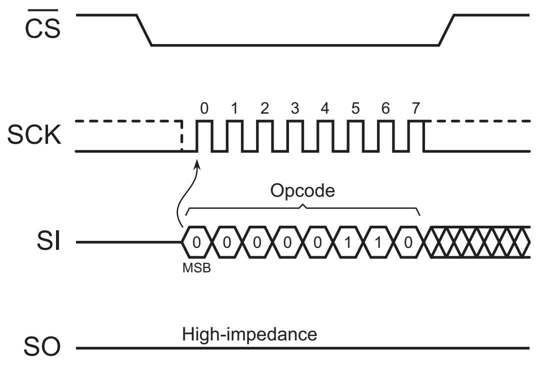
9. Protection Commands and Features

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, or Write Status 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

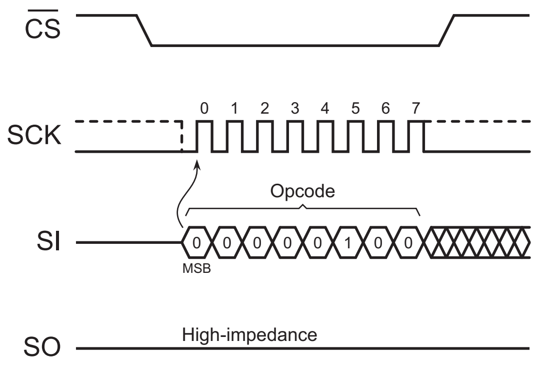


9.2 Write Disable

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, and Write Status Register commands will not be executed. Other conditions can also cause the WEL bit to be reset; for more details, see Section 11.1.5, "WEL Bit" on page 37 in "Status Register Commands" .

To issue the Write Disable command, the $\overline{\text{CS}}$ pin must first be asserted and then the opcode 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 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 9-2. Write Disable



9.3 Protect Sector

Every physical 64KB sector of the device has a corresponding single-bit Sector Protection Register that is used to control the software protection of a sector. Upon device power-up, each Sector Protection Register will default to the Logical 1 state indicating that all sectors are protected and cannot be programmed or erased.

Issuing the Protect Sector command to a particular sector address will set the corresponding Sector Protection Register to the Logical 1 state. The following table outlines the two states of the Sector Protection Registers.

Table 9-1. Sector Protection Register Values

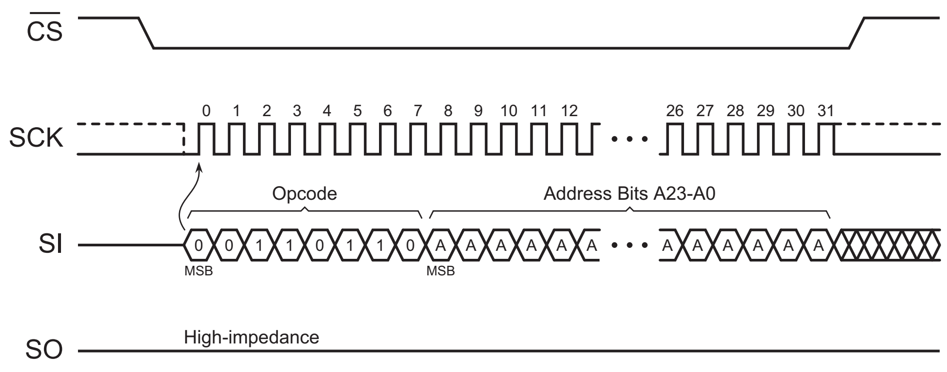
Value	Sector Protection Status
0	Sector is unprotected and can be programmed and erased.
1	Sector is protected and cannot be programmed or erased. This is the default state.

Before the Protect Sector command can be issued, the Write Enable command must have been previously issued to set the WEL bit in the Status Register to a Logical 1. To issue the Protect Sector command, the $\overline{CS}$ pin must first be asserted, and then the opcode 36h must be clocked into the device followed by three address bytes designating any address within the sector to be protected. Any additional data clocked into the device will be ignored. When the $\overline{CS}$ pin is deasserted, the Sector Protection Register corresponding to the physical sector addressed by A23-A0 will be set to the Logical 1 state, and the sector itself will then be protected from program and erase operations. In addition, the WEL bit in the Status Register will be reset back to the Logical 0 state.

The three complete address bytes must be clocked into the device before the $\overline{CS}$ pin is deasserted, and the $\overline{CS}$ pin must be deasserted on an even byte boundary (multiples of eight bits); otherwise, the device will abort the operation. When the device aborts the Protect Sector operation, the state of the Sector Protection Register will be unchanged, and the WEL bit in the Status Register will be reset to a Logical 0.

As a safeguard against accidental or erroneous protecting or unprotecting of sectors, the Sector Protection Registers can themselves be locked from updates by using the SPRL (Sector Protection Registers Locked) bit of the Status Register (please refer to the Status Register description for more details). If the Sector Protection Registers are locked, then any attempts to issue the Protect Sector command will be ignored, and the device will reset the WEL bit in the Status Register back to a Logical 0 and return to the idle state once the $\overline{CS}$ pin has been deasserted.

Figure 9-3. Protect Sector



9.4 Unprotect Sector

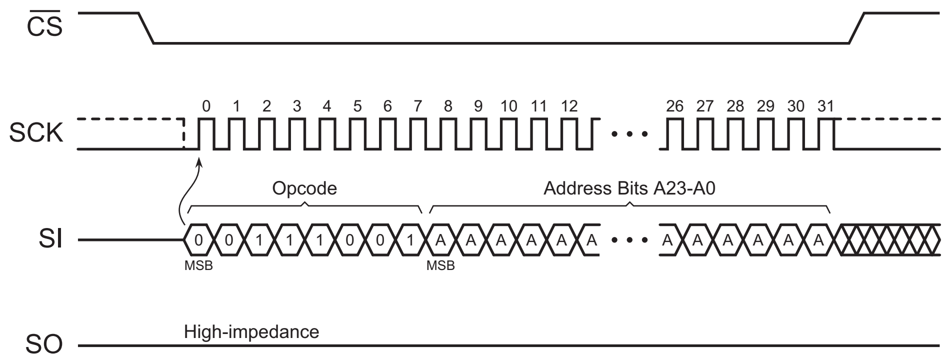
Issuing the Unprotect Sector command to a particular sector address will reset the corresponding Sector Protection Register to the Logical 0 state (see Table 9-4 for Sector Protection Register values). Every physical sector of the device has a corresponding single-bit Sector Protection Register that is used to control the software protection of a sector.

Before the Unprotect Sector command can be issued, the Write Enable command must have been previously issued to set the WEL bit in the Status Register to a Logical 1. To issue the Unprotect Sector command, the $\overline{\text{CS}}$ pin must first be asserted and then the opcode of 39h must be clocked into the device. After the opcode has been clocked in, the three address bytes designating any address within the sector to be unprotected must be clocked in. Any additional data clocked into the device after the address bytes will be ignored. When the $\overline{\text{CS}}$ pin is deasserted, the Sector Protection Register corresponding to the sector addressed by A23-A0 will be reset to the Logical 0 state, and the sector itself will be unprotected. In addition, the WEL bit in the Status Register will be reset back to the Logical 0 state.

The three complete address bytes 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, the state of the Sector Protection Register will be unchanged, and the WEL bit in the Status Register will be reset to a Logical 0.

As a safeguard against accidental or erroneous locking or unlocking of sectors, the Sector Protection Registers can themselves be locked from updates by using the SPRL (Sector Protection Registers Locked) bit of the Status Register (see Section 11., "Status Register Commands" on page 35 for more details). If the Sector Protection Registers are locked, then any attempts to issue the Unprotect Sector command will be ignored, and the device will reset the WEL bit in the Status Register back to a Logical 0 and return to the idle state once the $\overline{\text{CS}}$ pin has been deasserted.

Figure 9-4. Unprotect Sector



9.5 Global Protect/Unprotect

The Global Protect and Global Unprotect features can work in conjunction with the Protect Sector and Unprotect Sector functions.

Example: A system can globally protect the entire memory array and then use the Unprotect Sector command to individually unprotect certain sectors and individually reprotect them later by using the Protect Sector command. Likewise, a system can globally unprotect the entire memory array and then individually protect certain sectors as needed.

Performing a Global Protect or Global Unprotect is accomplished by writing a certain combination of data to the Status Register using the Write Status Register Byte 1 command (see "Write Status Register Byte 1" on page 39 for command execution details). The Write Status Register command is also used to modify the SPRL (Sector Protection Registers Locked) bit to control hardware and software locking.

To perform a Global Protect, the appropriate $\overline{\text{WP}}$ pin and SPRL conditions must be met, and the system must write a Logical 1 to bits 5, 4, 3, and 2 of the first byte of the Status Register. Conversely, to perform a Global Unprotect, the same $\overline{\text{WP}}$ and SPRL conditions must be met but the system must write a Logical 0 to bits 5, 4, 3, and 2 of the first byte of the Status Register. Table 9-2 details the conditions necessary for a Global Protect or Global Unprotect to be performed.

Sectors that have been erase or program suspended must remain in the unprotected state. If a Global Protect operation is attempted while a sector is erase or program suspended, the protection operation will abort, the protection states of all sectors in the Flash memory array will not change, and WEL bit in the Status Register will be reset back to a Logical 0.

Essentially, if the SPRL bit of the Status Register is in the Logical 0 state (Sector Protection Registers are not locked), then writing a 00h to the first byte of the Status Register will perform a Global Unprotect without changing the state of the SPRL bit. Similarly, writing a 7Fh to the first byte of the Status Register will perform a Global Protect and keep the SPRL bit in the Logical 0 state. The SPRL bit can, of course, be changed to a Logical 1 by writing an FFh if software-locking or hardware-locking is desired along with the Global Protect.

If the desire is to only change the SPRL bit without performing a Global Protect or Global Unprotect, then the system can simply write a 0Fh to the first byte of the Status Register to change the SPRL bit from a Logical 1 to a Logical 0 provided the $\overline{\text{WP}}$ pin is deasserted. Likewise, the system can write an F0h to change the SPRL bit from a Logical 0 to a Logical 1 without affecting the current sector protection status (no changes will be made to the Sector Protection Registers).

When writing to the first byte of the Status Register, bits 5, 4, 3, and 2 will not actually be modified, but will be decoded by the device for the purposes of the Global Protect and Global Unprotect functions. Only bit 7, the SPRL bit, will actually be modified. Therefore, when reading the first byte of the Status Register, bits 5, 4, 3, and 2 will not reflect the values written to them but will instead indicate the status of the $\overline{\text{WP}}$ pin and the sector protection status. Please see "Read Status Register" on page 35 and Table 11-1 on page 35 for details on the Status Register format and what values can be read for bits 5, 4, 3, and 2.