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

Features

- Single 2.7V - 3.6V Supply
- Serial Peripheral Interface (SPI) Compatible
  - Supports SPI Modes 0 and 3
- 70 MHz Maximum Clock Frequency
- Flexible, Uniform Erase Architecture
  - 4-Kbyte Blocks
  - 32-Kbyte Blocks
  - 64-Kbyte Blocks
  - Full Chip Erase
- Individual Sector Protection with Global Protect/Unprotect Feature
  - Thirty-two 64-Kbyte Physical Sectors
- Hardware Controlled Locking of Protected Sectors
- Flexible Programming Options
  - Byte/Page Program (1 to 256 Bytes)
  - Sequential Program Mode Capability
- Automatic Checking and Reporting of Erase/Program Failures
- JEDEC Standard Manufacturer and Device ID Read Methodology
- Low Power Dissipation
  - 5 mA Active Read Current (Typical)
  - 10 µA 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 (150-mil and 208-mil wide)
  - 8-pad MLF (6 x 5 x 1.00 mm)

16-megabit 2.7-volt Only Serial Firmware DataFlash® Memory

AT26DF161A

For New Designs Use AT25DF161

3640D–DFLASH–8/09

1. Description

The AT26DF161A 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 AT26DF161A, with its erase granularity as small as 4 Kbytes, 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 AT26DF161A 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 AT26DF161A 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 AT26DF161A 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.

Specifically designed for use in 3-volt systems, the AT26DF161A 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 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	**SERIAL INPUT:** 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 on the rising edge of SCK.		Input
SO	**SERIAL OUTPUT:** 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.		Output
$\overline{WP}$	**WRITE PROTECT:** The $\overline{WP}$ pin controls the hardware locking feature of the device. Please refer to section "Protection Commands and Features" on page 15 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
$\overline{HOLD}$	**HOLD:** The $\overline{HOLD}$ pin is used to temporarily pause serial communication without deselecting or resetting the device. While the $\overline{HOLD}$ 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 $\overline{CS}$ 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. Please refer to section "Hold" on page 30 for additional details on the Hold operation. The $\overline{HOLD}$ pin is internally pulled-high and may be left floating if the Hold function will not be used. However, it is recommended that the $\overline{HOLD}$ 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. 8-SOIC Top View

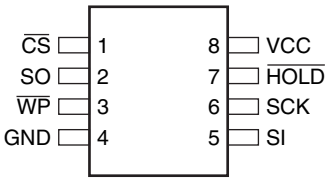
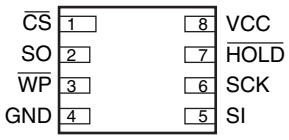
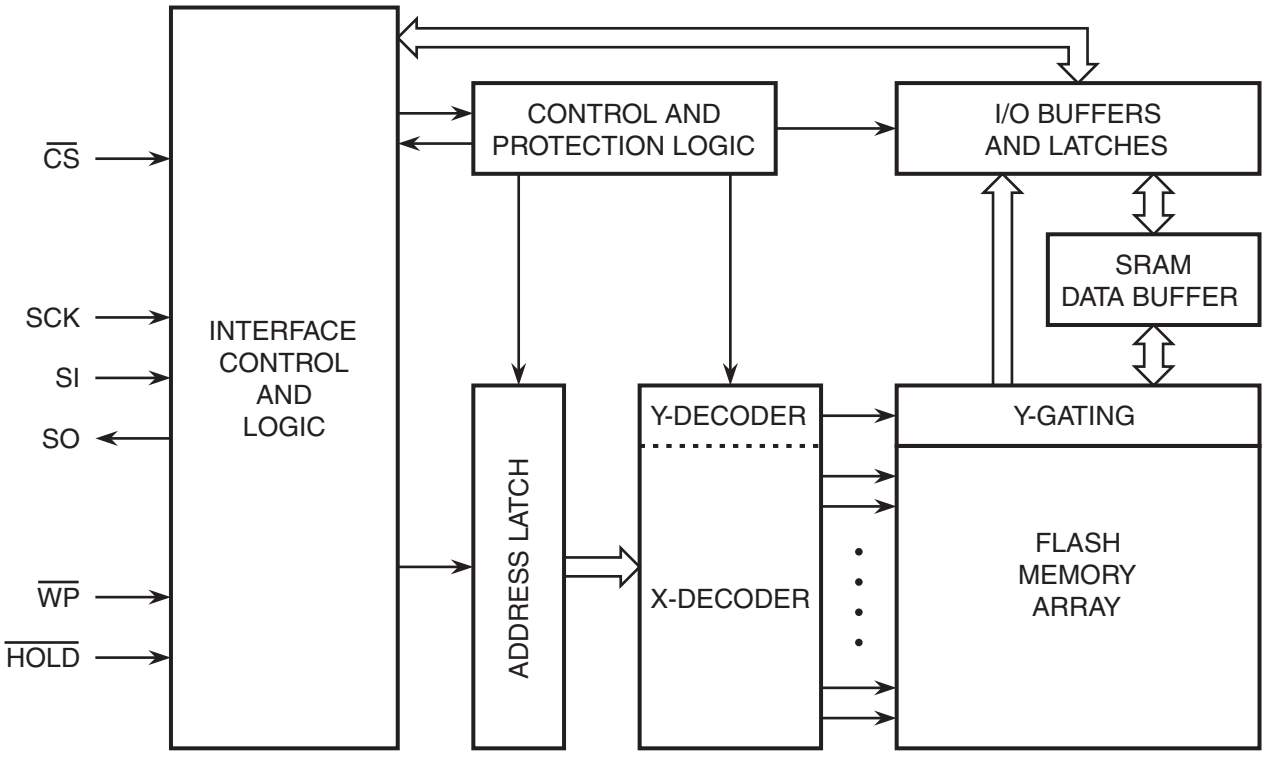


Figure 2-2. 8-MLF Top View



3. Block Diagram



4. Memory Array

To provide the greatest flexibility, the memory array of the AT26DF161A can be erased in four levels of granularity including a full chip erase. In addition, the array has been divided into physical sectors of various sizes, of which each sector can be individually protected from program and erase operations. The sizes of the physical sectors are optimized for both code and data storage applications, allowing both code and data segments to reside in their own isolated regions. Figure 4-1 on page 5 illustrates the breakdown of each erase level as well as the breakdown of each physical sector.

Figure 4-1. Memory Architecture Diagram

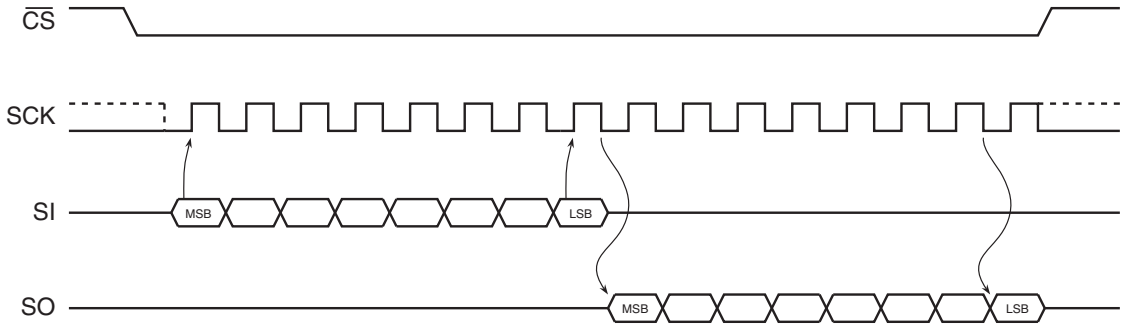
	Block Erase Detail				Page Program Detail	
Internal Sectoring for Sector Protection Function	64KB Block Erase (D8h Command)	32KB Block Erase (52h Command)	4KB Block Erase (20h Command)	Block Address Range	1-256 Byte Page Program (02h Command)	Page Address Range
64KB (Sector 31)	64KB	32KB	4KB	1FFFFFh – 1FF000h	256 Bytes	1FFFFFh – 1FFF00h
			4KB	1FEFFFh – 1FE000h	256 Bytes	1FFEFFh – 1FFE00h
			4KB	1FDFFFh – 1FD000h	256 Bytes	1FFDFFh – 1FFD00h
			4KB	1FCFFFh – 1FC000h	256 Bytes	0FFCFFh – 0FFC00h
			4KB	1FBFFFh – 1FB000h	256 Bytes	1FFBFFh – 1FFB00h
			4KB	1FAFFFh – 1FA000h	256 Bytes	1FFAFFh – 1FFA00h
			4KB	1F9FFFh – 1F9000h	256 Bytes	1FF9FFh – 1FF900h
			4KB	1F8FFFh – 1F8000h	256 Bytes	1FF8FFh – 1FF800h
		32KB	4KB	1F7FFFh – 1F7000h	256 Bytes	1FF7FFh – 1FF700h
			4KB	1F6FFFh – 1F6000h	256 Bytes	1FF6FFh – 1FF600h
			4KB	1F5FFFh – 1F5000h	256 Bytes	1FF5FFh – 1FF500h
			4KB	1F4FFFh – 1F4000h	256 Bytes	1FF4FFh – 1FF400h
			4KB	1F3FFFh – 1F3000h	256 Bytes	1FF3FFh – 1FF300h
			4KB	1F2FFFh – 1F2000h	256 Bytes	1FF2FFh – 1FF200h
			4KB	1F1FFFh – 1F1000h	256 Bytes	1FF1FFh – 1FF100h
			4KB	1F0FFFh – 1F0000h	256 Bytes	1FF0FFh – 1FF000h
64KB (Sector 30)	64KB	32KB	4KB	1EFFFFh – 1EF000h	256 Bytes	1FEFFFh – 1FEF00h
			4KB	1EEFFFh – 1EE000h	256 Bytes	1FEEFFh – 1FEE00h
			4KB	1EDFFFh – 1ED000h	256 Bytes	1FEDFFh – 1FED00h
			4KB	1ECFFFh – 1EC000h	256 Bytes	1FECFFh – 1FEC00h
			4KB	1EBFFFh – 1EB000h	256 Bytes	1FEBFFh – 1FEB00h
			4KB	1EAFFFh – 1EA000h	256 Bytes	1FEAFFh – 1FEA00h
			4KB	1E9FFFh – 1E9000h	256 Bytes	1FE9FFh – 1FE900h
			4KB	1E8FFFh – 1E8000h	256 Bytes	1FE8FFh – 1FE800h
		32KB	4KB	1E7FFFh – 1E7000h	⋮	
			4KB	1E6FFFh – 1E6000h		
			4KB	1E5FFFh – 1E5000h		
			4KB	1E4FFFh – 1E4000h	256 Bytes	0017FFh – 001700h
			4KB	1E3FFFh – 1E3000h	256 Bytes	0016FFh – 001600h
			4KB	1E2FFFh – 1E2000h	256 Bytes	0015FFh – 001500h
			4KB	1E1FFFh – 1E1000h	256 Bytes	0014FFh – 001400h
			4KB	1E0FFFh – 1E0000h	256 Bytes	0013FFh – 001300h
⋮	⋮	⋮	⋮		256 Bytes	0012FFh – 001200h
					256 Bytes	0011FFh – 001100h
					256 Bytes	0010FFh – 001000h
64KB (Sector 0)	64KB	32KB	4KB	00FFFFh – 00F000h	256 Bytes	000FFFh – 000F00h
			4KB	00EFFFh – 00E000h	256 Bytes	000EFFh – 000E00h
			4KB	00DFFFh – 00D000h	256 Bytes	000DFFh – 000D00h
			4KB	00CFFFh – 00C000h	256 Bytes	000CFFh – 000C00h
			4KB	00BFFFh – 00B000h	256 Bytes	000BFFh – 000B00h
			4KB	00AFFFh – 00A000h	256 Bytes	000AFFh – 000A00h
			4KB	009FFFh – 009000h	256 Bytes	0009FFh – 000900h
			4KB	008FFFh – 008000h	256 Bytes	0008FFh – 000800h
		32KB	4KB	007FFFh – 007000h	256 Bytes	0007FFh – 000700h
			4KB	006FFFh – 006000h	256 Bytes	0006FFh – 000600h
			4KB	005FFFh – 005000h	256 Bytes	0005FFh – 000500h
			4KB	004FFFh – 004000h	256 Bytes	0004FFh – 000400h
			4KB	003FFFh – 003000h	256 Bytes	0003FFh – 000300h
			4KB	002FFFh – 002000h	256 Bytes	0002FFh – 000200h
			4KB	001FFFh – 001000h	256 Bytes	0001FFh – 000100h
			4KB	000FFFh – 000000h	256 Bytes	0000FFh – 000000h

5. Device Operation

The AT26DF161A 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 AT26DF161A 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 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 AT26DF161A 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 SPI Master 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 SPI Master. 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 AT26DF161A 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 AT26DF161A memory array is 1FFFFFh, address bits A23 - A21 are always ignored by the device.

Table 6-1. Command Listing

Command	Opcode		Address Bytes	Dummy Bytes	Data Bytes
Read Commands					
Read Array	0Bh	0000 1011	3	1	1+
Read Array (Low Frequency)	03h	0000 0011	3	0	1+
Program and Erase Commands					
Block Erase (4 Kbytes)	20h	0010 0000	3	0	0
Block Erase (32 Kbytes)	52h	0101 0010	3	0	0
Block Erase (64 Kbytes)	D8h	1101 1000	3	0	0
Chip Erase	60h	0110 0000	0	0	0
	C7h	1100 0111	0	0	0
Byte/Page Program (1 to 256 Bytes)	02h	0000 0010	3	0	1+
Sequential Program Mode	ADh	1010 1101	3, 0[1]	0	1
	AFh	1010 1111	3, 0[1]	0	1
Protection Commands					
Write Enable	06h	0000 0110	0	0	0
Write Disable	04h	0000 0100	0	0	0
Protect Sector	36h	0011 0110	3	0	0
Unprotect Sector	39h	0011 1001	3	0	0
Global Protect/Unprotect	Use Write Status Register command				
Read Sector Protection Registers	3Ch	0011 1100	3	0	1+
Status Register Commands					
Read Status Register	05h	0000 0101	0	0	1+
Write Status Register	01h	0000 0001	0	0	1
Miscellaneous Commands					
Read Manufacturer and Device ID	9Fh	1001 1111	0	0	1 to 4
Deep Power-down	B9h	1011 1001	0	0	0
Resume from Deep Power-down	ABh	1010 1011	0	0	0

Note: 1. Three address bytes are only required for the first operation to designate the address to start programming at. Afterwards, the internal address counter automatically increments, so subsequent Sequential Program Mode operations only require clocking in of the opcode and the data byte until the Sequential Program Mode has been exited.

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 SCK signal once the initial starting address has been specified. The device incorporates an internal address counter that automatically increments on every clock cycle.

Two opcodes, 0Bh and 03h, can be used for the Read Array command. The use of each opcode depends on the maximum SCK frequency that will be used to read data from the device. The 0Bh opcode can be used at any SCK frequency up to the maximum specified by f_{SCK}. The 03h opcode can be used for lower frequency read operations up to the maximum specified by f_{RDLF}.

To perform the Read Array operation, the $\overline{CS}$ pin must first be asserted and the appropriate opcode (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. If the 0Bh opcode is used, then one don't care byte must also be clocked in after the three address bytes.

After the three address bytes (and the one don't care byte if using opcode 0Bh) have been clocked in, additional clock cycles will result in serial data being output on the SO pin. The data is always output with the MSB of a byte first. When the last byte (1FFFFFh) 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 – 0Bh Opcode

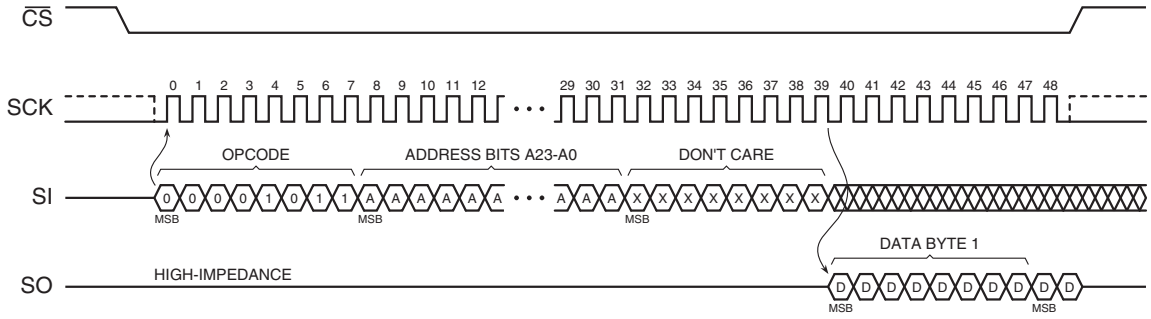
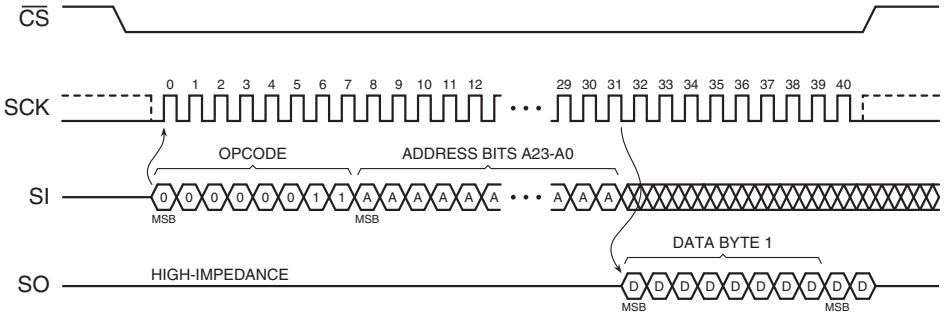


Figure 7-2. Read Array – 03h Opcode



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 15 command description) 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’s), then special circumstances regarding which memory locations will 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. For 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 be unaffected and will not change. 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 altered. The programming of the data bytes is internally self-timed and should take place in a time of t_{PP}.

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 section “Protect Sector” on page 16), 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, or because the memory location to be programmed is protected.

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.

The Byte/Page Program mode is the default programming mode after the device powers-up or resumes from a device reset.

Figure 8-1. Byte Program

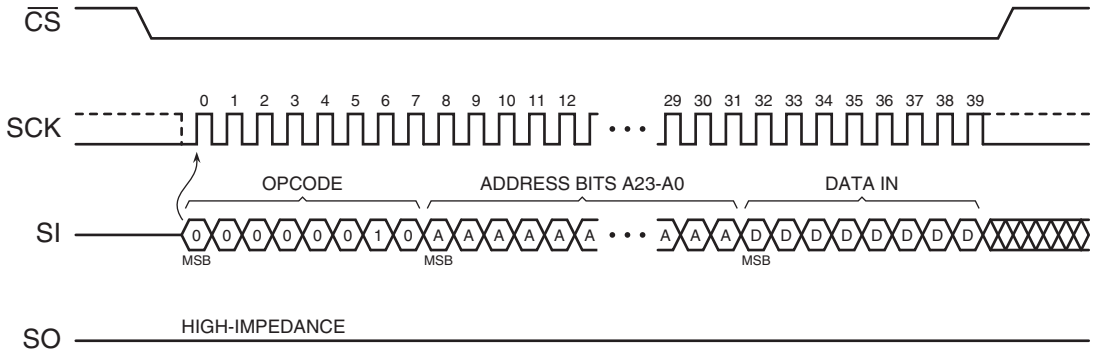
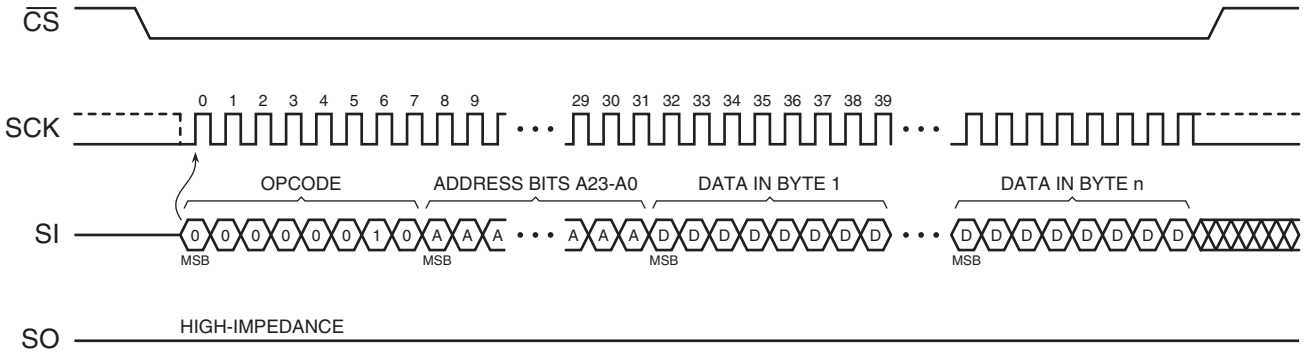


Figure 8-2. Page Program



8.2 Sequential Program Mode

The Sequential Program Mode improves throughput over the Byte/Page Program command when the Byte/Page Program command is used to program single bytes only into consecutive address locations. For example, some systems may be designed to program only a single byte of information at a time and cannot utilize a buffered Page Program operation due to design restrictions. In such a case, the system would normally have to perform multiple Byte Program operations in order to program data into sequential memory locations. This approach can add considerable system overhead and SPI bus traffic.

The Sequential Programming Mode helps reduce system overhead and bus traffic by incorporating an internal address counter that keeps track of the byte location to program, thereby eliminating the need to supply an address sequence to the device for every byte to program. When using the Sequential Program mode, all address locations to be programmed must be in the erased state. Before the Sequential Program mode can first be entered, 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 start the Sequential Program Mode, the $\overline{CS}$ pin must first be asserted, and either an opcode of ADh or AFh must be clocked into the device. For the first program cycle, three address bytes must be clocked in after the opcode to designate the first byte location to program. After the address bytes have been clocked in, the byte of data to be programmed can be sent to the

device. Deasserting the $\overline{CS}$ pin will start the internally self-timed program operation, and the byte of data will be programmed into the memory location specified by A23 - A0.

After the first byte has been successfully programmed, a second byte can be programmed by simply reasserting the $\overline{CS}$ pin, clocking in the ADh or AFh opcode, and then clocking in the next byte of data. When the $\overline{CS}$ pin is deasserted, the second byte of data will be programmed into the next sequential memory location. The process would be repeated for any additional bytes. There is no need to reissue the Write Enable command once the Sequential Program Mode has been entered.

When the last desired byte has been programmed into the memory array, the Sequential Program Mode operation can be terminated by reasserting the $\overline{CS}$ pin and sending the Write Disable command to the device to reset the WEL bit in the Status Register back to the logical “0” state.

If more than one byte of data is ever clocked in during each program cycle, then only the last byte of data sent on the SI pin will be stored in the internal latches. The programming of each byte is internally self-timed and should take place in a time of t_{BP}. For each program cycle, a 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, the byte of data will not be programmed into the memory array, and the WEL bit in the Status Register will be reset back to the logical “0” state.

If the address initially specified by A23 - A0 points to a memory location within a sector that is in the protected state, then the Sequential Program Mode 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 also be reset back to the logical “0” state.

There is no address wrapping when using the Sequential Program Mode. Therefore, when the last byte (1FFFFFh) of the memory array has been programmed, the device will automatically exit the Sequential Program mode and reset the WEL bit in the Status Register back to the logical “0” state. In addition, the Sequential Program mode will not automatically skip over protected sectors; therefore, once the highest unprotected memory location in a programming sequence has been programmed, the device will automatically exit the Sequential Program mode and reset the WEL bit in the Status Register. For example, if Sector 1 was protected and Sector 0 was currently being programmed, once the last byte of Sector 0 was programmed, the Sequential Program mode would automatically end. To continue programming with Sector 2, the Sequential Program mode would have to be restarted by supplying the ADh or AFh opcode, the three address bytes, and the first byte of Sector 2 to program.

While the device is programming a byte, 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 at the end of each program cycle rather than waiting the t_{BP} time to determine if the byte has finished programming before starting the next Sequential Program mode cycle.

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. Sequential Program Mode – Status Register Polling

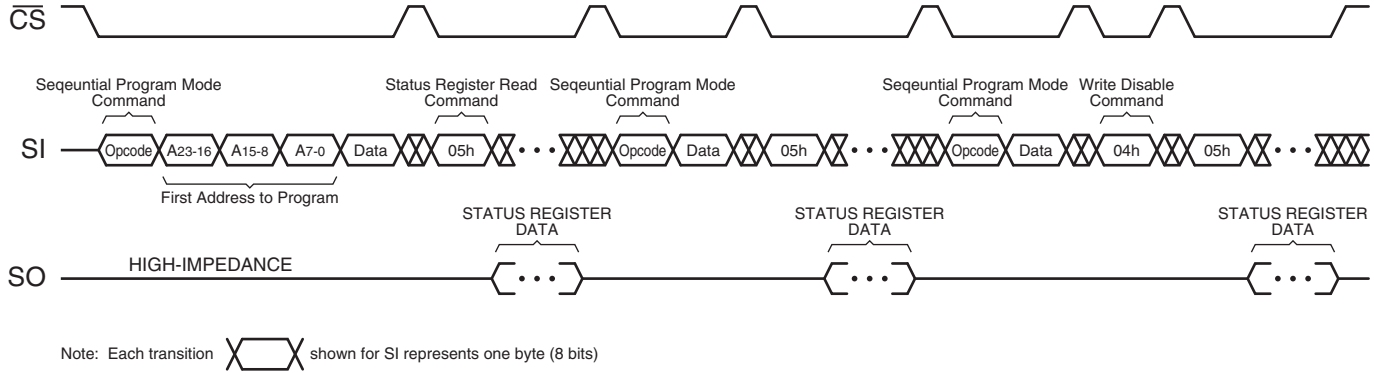
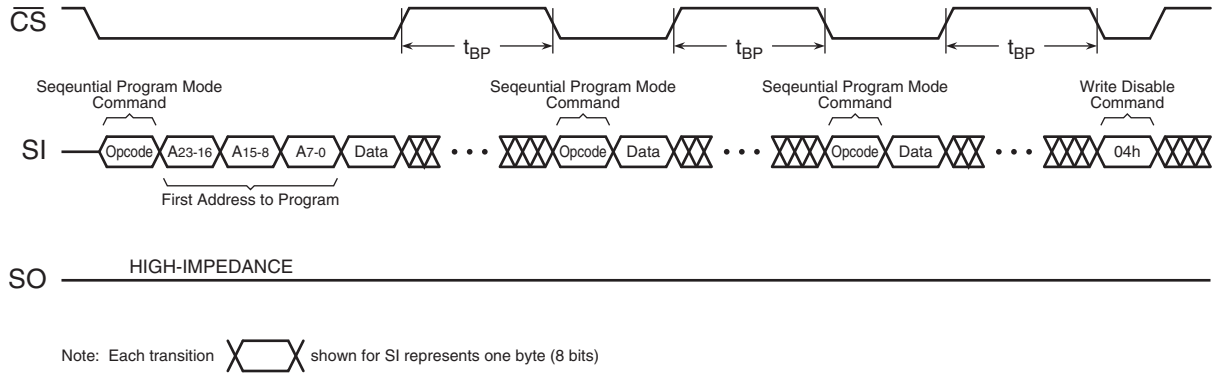


Figure 8-4. Sequential Program Mode – Waiting Maximum Byte Program Time



8.3 Block Erase

A block of 4, 32, or 64 Kbytes 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 4-Kbyte erase, an opcode of 52h is used for a 32-Kbyte erase, and an opcode of D8h is used for a 64-Kbyte 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 $\overline{CS}$ 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 4-, 32-, or 64-Kbyte block to be erased must be clocked in. Any additional data clocked into the device will be ignored. When the $\overline{CS}$ 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 4-Kbyte erase, address bits A11 - A0 will be ignored by the device and their values can be either a logical “1” or “0”. For a 32-Kbyte erase, address bits A14 - A0 will be ignored, and for a 64-Kbyte 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 $\overline{CS}$ 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 and no erase operation will be performed.

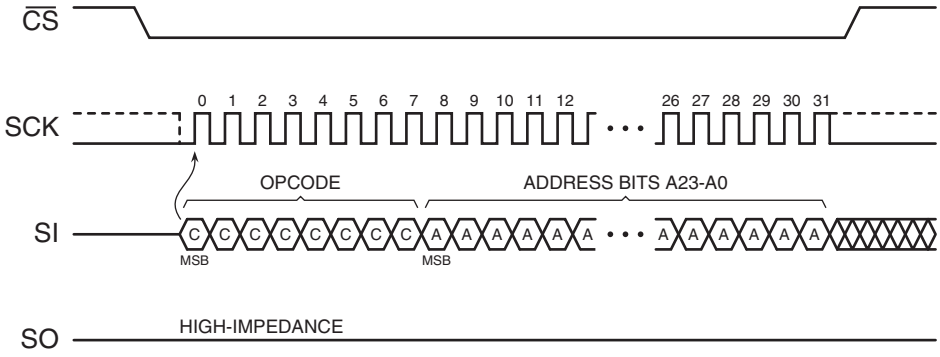
If the address specified by A23 - A0 points to a memory location within a sector that is in the protected state, then the Block Erase command will not be executed, and the device will return to the idle state once the $\overline{CS}$ pin has been deasserted. In addition, with the larger Block Erase sizes of 32K and 64 Kbytes, more than one physical sector may be erased (e.g. sectors 18 through 15) at one time. Therefore, in order to erase a larger block that may span more than one sector, all of the sectors in the span must be in the unprotected state. If one of the physical sectors within the span is in the protected state, then the device will ignore the Block Erase command and will return to the idle state once the $\overline{CS}$ pin is 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 or because a memory location within the region to be erased is protected.

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.

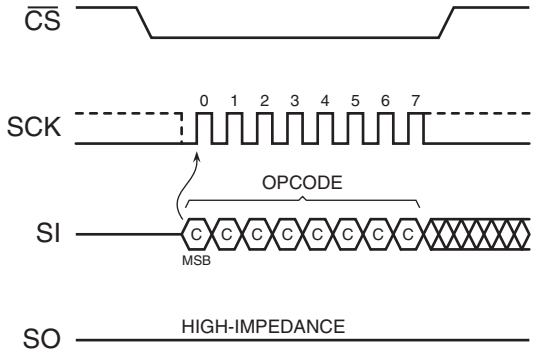
Two opcodes, 60h and C7h, can be used for the Chip Erase command. There is no difference in device functionality when utilizing the two opcodes, so they can be used interchangeably. To perform a Chip Erase, one of the two opcodes (60h or C7h) must be clocked into the device. Since the entire memory array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



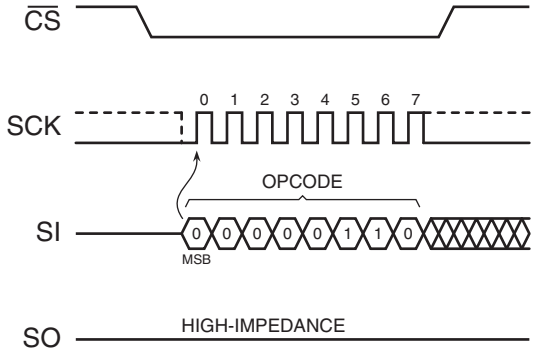
9. Protection Commands and Features

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 program, erase, Protect Sector, Unprotect Sector, 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 $\overline{CS}$ 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 $\overline{CS}$ 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 $\overline{CS}$ 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 and the state of the WEL bit will not change.

Figure 9-1. Write Enable

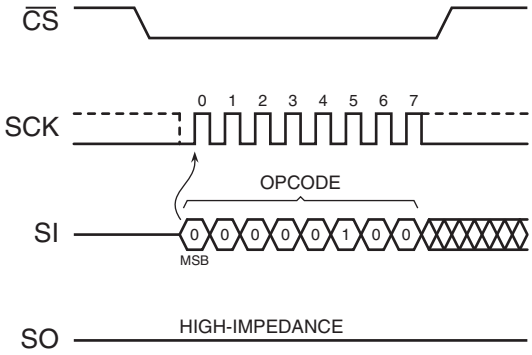


9.2 Write Disable

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 program, erase, Protect Sector, Unprotect Sector, and Write Status Register commands will not be executed. The Write Disable command is also used to exit the Sequential Program Mode. 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 $\overline{CS}$ 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 $\overline{CS}$ 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 $\overline{CS}$ 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 and the state of the WEL bit will not change.

Figure 9-2. Write Disable



9.3 Protect Sector

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. Upon device power-up or after a device reset, 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

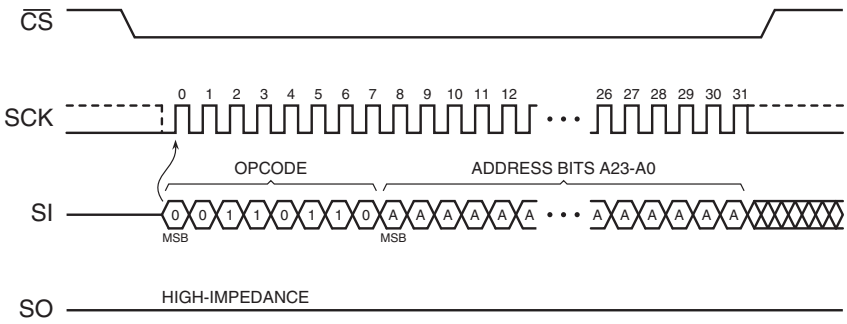
Value	Sector Protection Status
0	Sector is unprotected and can be programmed and erased.
1	Sector is protected and cannot be programmed or erased. This is the default state.

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 the opcode of 36h must be clocked into the device followed by three address bytes designating any address within the sector to be locked. 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 complete three 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, 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

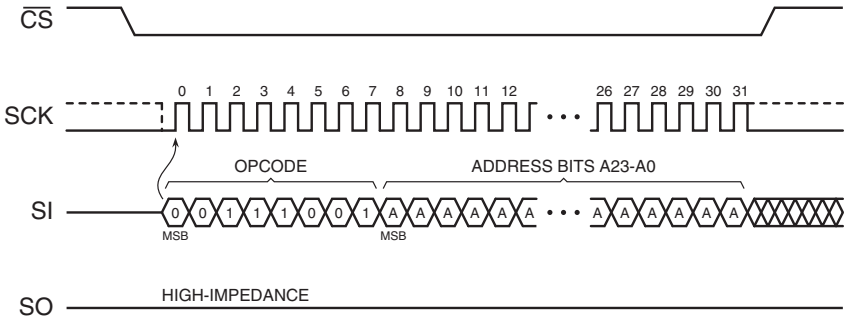
Issuing the Unprotect Sector command to a particular sector address will reset the corresponding Sector Protection Register to the logical “0” state (see Table 9-1 for Sector Protection Register values). Every physical sector of the device has a corresponding single-bit Sector Protection Register that 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 $\overline{CS}$ pin must first be asserted and 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 unlocked must be clocked in. Any additional data clocked into the device after the address bytes will be ignored. When the $\overline{CS}$ 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 complete three 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, 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 (please refer to the Status Register description 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 $\overline{CS}$ 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. For 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 command (see “Write Status Register” section on page 26 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 $\overline{WP}$ pin and SPRL conditions must be met, and the system must write a logical “1” to bits 5, 4, 3, and 2 of the Status Register. Conversely, to perform a Global Unprotect, the same $\overline{WP}$ and SPRL conditions must be met but the system must write a logical “0” to bits 5, 4, 3, and 2 of the Status Register. Table 9-2 details the conditions necessary for a Global Protect or Global Unprotect to be performed.

Table 9-2. Valid SPRL and Global Protect/Unprotect Conditions

$\overline{WP}$ State	Current SPRL Value	New Write Status Register Data Bit 7 6 5 4 3 2 1 0	Protection Operation	New SPRL Value
0	0	0 x 0 0 0 0 x x	Global Unprotect – all Sector Protection Registers reset to 0	0
		0 x 0 0 0 1 x x	No change to current protection.	0
			No change to current protection.	0
		0 x 1 1 1 0 x x	No change to current protection.	0
		0 x 1 1 1 1 x x	Global Protect – all Sector Protection Registers set to 1	0
		1 x 0 0 0 0 x x	Global Unprotect – all Sector Protection Registers reset to 0	1
		1 x 0 0 0 1 x x	No change to current protection.	1
			No change to current protection.	1
		1 x 1 1 1 0 x x	No change to current protection.	1
		1 x 1 1 1 1 x x	Global Protect – all Sector Protection Registers set to 1	1
0	1	x x x x x x x x	No change to the current protection level. All sectors currently protected will remain protected and all sectors currently unprotected will remain unprotected. The Sector Protection Registers are hard-locked and cannot be changed when the $\overline{WP}$ pin is LOW and the current state of SPRL is 1. Therefore, a Global Protect/Unprotect will not occur. In addition, the SPRL bit cannot be changed (the $\overline{WP}$ pin must be HIGH in order to change SPRL back to a 0).	
1	0	0 x 0 0 0 0 x x	Global Unprotect – all Sector Protection Registers reset to 0	0
		0 x 0 0 0 1 x x	No change to current protection.	0
			No change to current protection.	0
		0 x 1 1 1 0 x x	No change to current protection.	0
		0 x 1 1 1 1 x x	Global Protect – all Sector Protection Registers set to 1	0
		1 x 0 0 0 0 x x	Global Unprotect – all Sector Protection Registers reset to 0	1
		1 x 0 0 0 1 x x	No change to current protection.	1
			No change to current protection.	1
		1 x 1 1 1 0 x x	No change to current protection.	1
		1 x 1 1 1 1 x x	Global Protect – all Sector Protection Registers set to 1	1
1	1	0 x 0 0 0 0 x x	No change to the current protection level. All sectors currently protected will remain protected, and all sectors currently unprotected will remain unprotected. The Sector Protection Registers are soft-locked and cannot be changed when the current state of SPRL is 1. Therefore, a Global Protect/Unprotect will not occur. However, the SPRL bit can be changed back to a 0 from a 1 since the $\overline{WP}$ pin is HIGH. To perform a Global Protect/Unprotect, the Write Status Register command must be issued again after the SPRL bit has been changed from a 1 to a 0.	0
		0 x 0 0 0 1 x x		0
				0
		0 x 1 1 1 0 x x		0
		0 x 1 1 1 1 x x		0
		1 x 0 0 0 0 x x		1
		1 x 0 0 0 1 x x		1
				1
		1 x 1 1 1 0 x x		1
		1 x 1 1 1 1 x x		1

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 Status Register will perform a Global Unprotect without changing the state of the SPRL bit. Similarly, writing a 7Fh to 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 Status Register to change the SPRL bit from a logical “1” to a logical “0” provided the $\overline{WP}$ 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 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 Status Register, bits 5, 4, 3, and 2 will not reflect the values written to them but will instead indicate the status of the $\overline{WP}$ pin and the sector protection status. Please refer to the “Read Status Register” section and Table 10-1 on page 23 for details on the Status Register format and what values can be read for bits 5, 4, 3, and 2.

9.6 Read Sector Protection Registers

The Sector Protection Registers can be read to determine the current software protection status of each sector. Reading the Sector Protection Registers, however, will not determine the status of the $\overline{WP}$ pin.

To read the Sector Protection Register for a particular sector, the $\overline{CS}$ pin must first be asserted and the opcode of 3Ch must be clocked in. Once the opcode has been clocked in, three address bytes designating any address within the sector must be clocked in. After the last address byte has been clocked in, the device will begin outputting data on the SO pin during every subsequent clock cycle. The data being output will be a repeating byte of either FFh or 00h to denote the value of the appropriate Sector Protection Register.

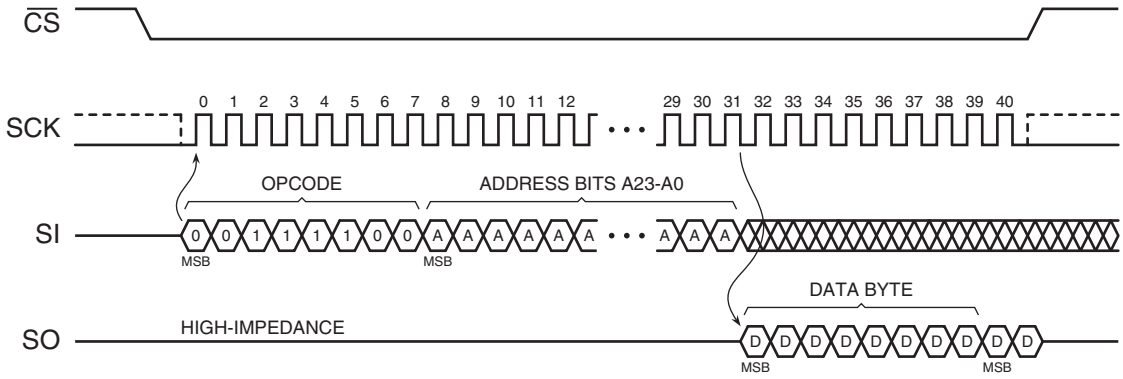
Table 9-3. Read Sector Protection Register – Output Data

Output Data	Sector Protection Register Value
00h	Sector Protection Register value is 0 (sector is unprotected).
FFh	Sector Protection Register value is 1 (sector is protected).

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.

In addition to reading the individual Sector Protection Registers, the Software Protection Status (SWP) bit in the Status Register can be read to determine if all, some, or none of the sectors are software protected (refer to the “Status Register Commands” on page 23 for more details).

Figure 9-5. Read Sector Protection Register



9.7 Protected States and the Write Protect ($\overline{WP}$) Pin

The $\overline{WP}$ pin is not linked to the memory array itself and has no direct effect on the protection status of the memory array. Instead, the $\overline{WP}$ pin, in conjunction with the SPRL (Sector Protection Registers Locked) bit in the Status Register, is used to control the hardware locking mechanism of the device. For hardware locking to be active, two conditions must be met – the $\overline{WP}$ pin must be asserted and the SPRL bit must be in the logical "1" state.

When hardware locking is active, the Sector Protection Registers are locked and the SPRL bit itself is also locked. Therefore, sectors that are protected will be locked in the protected state, and sectors that are unprotected will be locked in the unprotected state. These states cannot be changed as long as hardware locking is active, so the Protect Sector, Unprotect Sector, and Write Status Register commands will be ignored. In order to modify the protection status of a sector, the $\overline{WP}$ pin must first be deasserted, and the SPRL bit in the Status Register must be reset back to the logical "0" state using the Write Status Register command. When resetting the SPRL bit back to a logical "0", it is not possible to perform a Global Protect or Global Unprotect at the same time since the Sector Protection Registers remain soft-locked until after the Write Status Register command has been executed.

If the $\overline{WP}$ pin is permanently connected to GND, then once the SPRL bit is set to a logical "1", the only way to reset the bit back to the logical "0" state is to power-cycle or reset the device. This allows a system to power-up with all sectors software protected but not hardware locked. Therefore, sectors can be unprotected and protected as needed and then hardware locked at a later time by simply setting the SPRL bit in the Status Register.

When the $\overline{WP}$ pin is deasserted, or if the $\overline{WP}$ pin is permanently connected to VCC, the SPRL bit in the Status Register can still be set to a logical "1" to lock the Sector Protection Registers. This provides a software locking ability to prevent erroneous Protect Sector or Unprotect Sector commands from being processed. When changing the SPRL bit to a logical "1" from a logical "0", it is also possible to perform a Global Protect or Global Unprotect at the same time by writing the appropriate values into bits 5, 4, 3, and 2 of the Status Register.

Tables 9-4 and 9-5 detail the various protection and locking states of the device.

Table 9-4. Sector Protection Register States

$\overline{WP}$	Sector Protection Register n[1]	Sector n[1]
X (Don't Care)	0	Unprotected
	1	Protected

Note: 1. "n" represents a sector number

Table 9-5. Hardware and Software Locking

$\overline{WP}$	SPRL	Locking	SPRL Change Allowed	Sector Protection Registers
0	0		Can be modified from 0 to 1	Unlocked and modifiable using the Protect and Unprotect Sector commands. Global Protect and Unprotect can also be performed.
0	1	Hardware Locked	Locked	Locked in current state. Protect and Unprotect Sector commands will be ignored. Global Protect and Unprotect cannot be performed.
1	0		Can be modified from 0 to 1	Unlocked and modifiable using the Protect and Unprotect Sector commands. Global Protect and Unprotect can also be performed.
1	1	Software Locked	Can be modified from 1 to 0	Locked in current state. Protect and Unprotect Sector commands will be ignored. Global Protect and Unprotect cannot be performed.

10. Status Register Commands

10.1 Read Status Register

The Status Register can be read to determine the device's ready/busy status, as well as the status of many other functions such as Hardware Locking and Software Protection. The Status Register can be read at any time, including during an internally self-timed program or erase operation.

To read the Status Register, the $\overline{CS}$ pin must first be asserted and the opcode of 05h must be clocked into the device. After the last bit of the opcode has been clocked in, the device will begin outputting Status Register data on the SO pin during every subsequent clock cycle. After the last bit (bit 0) of the Status Register has been clocked out, the sequence will repeat itself starting again with bit 7 as long as the $\overline{CS}$ pin remains asserted and the SCK pin is being pulsed. The data in the Status Register is constantly being updated, so each repeating sequence will output new data.

Deasserting the $\overline{CS}$ pin will terminate the Read Status Register 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.

Table 10-1. Status Register Format

Bit[1]	Name		Type[2]	Description	
7	SPRL	Sector Protection Registers Locked	R/W	0 1	Sector Protection Registers are unlocked (default). Sector Protection Registers are locked.
6	SPM	Sequential Program Mode Status	R	0 1	Byte/Page Programming Mode (default). Sequential Programming Mode entered.
5	EPE	Erase/Program Error	R	0 1	Erase or program operation was successful. Erase or program error detected.
4	WPP	Write Protect ($\overline{WP}$) Pin Status	R	0 1	$\overline{WP}$ is asserted. $\overline{WP}$ is deasserted.
3:2	SWP	Software Protection Status	R	00 01 10 11	All sectors are software unprotected (all Sector Protection Registers are 0). Some sectors are software protected. Read individual Sector Protection Registers to determine which sectors are protected. *Reserved for future use.* All sectors are software protected (all Sector Protection Registers are 1 – default).
1	WEL	Write Enable Latch Status	R	0 1	Device is not write enabled (default). Device is write enabled.
0	RDY/BSY	Ready/Busy Status	R	0 1	Device is ready. Device is busy with an internal operation.

Notes: 1. Only bit 7 of the Status Register will be modified when using the Write Status Register command.

2. R/W = Readable and writable
   R = Readable only

10.1.1 SPRL Bit

The SPRL bit is used to control whether the Sector Protection Registers can be modified or not. When the SPRL bit is in the logical “1” state, all Sector Protection Registers are locked and cannot be modified with the Protect Sector and Unprotect Sector commands (the device will ignore these commands). In addition, the Global Protect and Global Unprotect features cannot be performed. Any sectors that are presently protected will remain protected, and any sectors that are presently unprotected will remain unprotected.

When the SPRL bit is in the logical “0” state, all Sector Protection Registers are unlocked and can be modified (the Protect Sector and Unprotect Sector commands, as well as the Global Protect and Global Unprotect features, will be processed as normal). The SPRL bit defaults to the logical “0” state after a power-up or a device reset.

The SPRL bit can be modified freely whenever the $\overline{WP}$ pin is deasserted. However, if the $\overline{WP}$ pin is asserted, then the SPRL bit may only be changed from a logical “0” (Sector Protection Registers are unlocked) to a logical “1” (Sector Protection Registers are locked). In order to reset the SPRL bit back to a logical “0” using the Write Status Register command, the $\overline{WP}$ pin will have to first be deasserted.

The SPRL bit is the only bit of the Status Register that can be user modified via the Write Status Register command.

10.1.2 SPM Bit

The SPM bit indicates whether the device is in the Byte/Page Program mode or the Sequential Program Mode. The default state after power-up or device reset is the Byte/Page Program mode.

10.1.3 EPE Bit

The EPE bit indicates whether the last erase or program operation completed successfully or not. If at least one byte during the erase or program operation did not erase or program properly, then the EPE bit will be set to the logical “1” state. The EPE bit will not be set if an erase or program operation aborts for any reason such as an attempt to erase or program a protected region or if the WEL bit is not set prior to an erase or program operation. The EPE bit will be updated after every erase and program operation.

10.1.4 WPP Bit

The WPP bit can be read to determine if the $\overline{WP}$ pin has been asserted or not.

10.1.5 SWP Bits

The SWP bits provide feedback on the software protection status for the device. There are three possible combinations of the SWP bits that indicate whether none, some, or all of the sectors have been protected using the Protect Sector command or the Global Protect feature. If the SWP bits indicate that some of the sectors have been protected, then the individual Sector Protection Registers can be read with the Read Sector Protection Registers command to determine which sectors are in fact protected.

10.1.6 WEL Bit

The WEL bit indicates the current status of the internal Write Enable Latch. When the WEL bit is in the logical “0” state, the device will not accept any program, erase, Protect Sector, Unprotect Sector, or Write Status Register commands. The WEL bit defaults to the logical “0” state after a device power-up or reset. In addition, the WEL bit will be reset to the logical “0” state automatically under the following conditions:

- Write Disable operation completes successfully
- Write Status Register operation completes successfully or aborts
- Protect Sector operation completes successfully or aborts
- Unprotect Sector operation completes successfully or aborts
- Byte/Page Program operation completes successfully or aborts
- Sequential Program Mode reaches highest unprotected memory location
- Sequential Program Mode reaches the end of the memory array
- Sequential Program Mode aborts
- Block Erase operation completes successfully or aborts
- Chip Erase operation completes successfully or aborts
- Hold condition aborts

If the WEL bit is in the logical “1” state, it will not be reset to a logical “0” if an operation aborts due to an incomplete or unrecognized opcode being clocked into the device before the $\overline{CS}$ pin is deasserted. In order for the WEL bit to be reset when an operation aborts prematurely, the entire opcode for a program, erase, Protect Sector, Unprotect Sector, or Write Status Register command must have been clocked into the device.

10.1.7 RDY/BSY Bit

The RDY/BSY bit is used to determine whether or not an internal operation, such as a program or erase, is in progress. To poll the RDY/BSY bit to detect the completion of a program or erase cycle, new Status Register data must be continually clocked out of the device until the state of the RDY/BSY bit changes from a logical “1” to a logical “0”.

Figure 10-1. Read Status Register