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

AT25SF321

32-Mbit, 2.5V Minimum
SPI Serial Flash Memory with Dual-I/O and Quad-IO Support

Features

- Single 2.5V - 3.6V Supply
- Serial Peripheral Interface (SPI) Compatible
  - Supports SPI Modes 0 and 3
  - Supports Dual and Quad Output Read
- 104MHz Maximum Operating Frequency
  - Clock-to-Output (t_V) of 6 ns
- Flexible, Optimized Erase Architecture for Code + Data Storage Applications
  - Uniform 4-Kbyte Block Erase
  - Uniform 32-Kbyte Block Erase
  - Uniform 64-Kbyte Block Erase
- Full Chip Erase
- Hardware Controlled Locking of Protected Blocks via $\overline{\text{WP}}$ Pin
- 3 Protected Programmable Security Register Pages
- Flexible Programming
  - Byte/Page Program (1 to 256 Bytes)
- Fast Program and Erase Times
  - 0.7ms Typical Page Program (256 Bytes) Time
  - 70ms Typical 4-Kbyte Block Erase Time
  - 300ms Typical 32-Kbyte Block Erase Time
  - 600ms Typical 64-Kbyte Block Erase Time
- JEDEC Standard Manufacturer and Device ID Read Methodology
- Low Power Dissipation
  - 2µA Deep Power-Down Current (Typical)
  - 10µA Standby current (Typical)
  - 4mA Active Read 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)
  - 8-pad Ultra Thin DFN (5 x 6 x 0.6 mm)
  - Die in Wafer Form

Description

The Adesto® AT25SF321 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 AT25SF321 is ideal for data storage as well, eliminating the need for additional data storage devices.

The erase block sizes of the AT25SF321 have been optimized to meet the needs of today's code and data storage applications. By optimizing the size of the 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 erase regions, the wasted and unused memory space that occurs with 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 device also contains three pages of Security Register that can be used for purposes such as unique device serialization, system-level Electronic Serial Number (ESN) storage, locked key storage, etc. These Security Register pages can be individually locked.

1. Pin Descriptions and Pinouts

Table 1-1. 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 (I/O_0)	**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 in on the rising edge of SCK. With the Dual-Output and Quad-Output Read commands, the SI Pin becomes an output pin (I/O_0) in conjunction with other pins to allow two or four bits of data on (I/O_{3-0}) to be clocked in on every falling edge of SCK To maintain consistency with the SPI nomenclature, the SI (I/O_0) pin will be referenced as the SI pin unless specifically addressing the Dual-I/O and Quad-I/O modes in which case it will be referenced as I/O_0 Data present on the SI pin will be ignored whenever the device is deselected ($\overline{CS}$ is deasserted).	-	Input/Output

Table 1-1. Pin Descriptions (Continued)

Symbol	Name and Function	Asserted State	Type
SO (I/O_1)	**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. With the Dual-Output Read commands, the SO Pin remains an output pin (I/O_0) in conjunction with other pins to allow two bits of data on (I/O_{1-0}) to be clocked in on every falling edge of SCK To maintain consistency with the SPI nomenclature, the SO (I/O_1) pin will be referenced as the SO pin unless specifically addressing the Dual-I/O modes in which case it will be referenced as I/O_1 The SO pin will be in a high-impedance state whenever the device is deselected ($\overline{CS}$ is deasserted).	-	Input/Output
$\overline{WP}$ (I/O_2)	**WRITE PROTECT:** The $\overline{WP}$ pin controls the hardware locking feature of the device. With the Quad-Input Byte/Page Program command, the $\overline{WP}$ pin becomes an input pin (I/O_2) and, along with other pins, allows four bits (on I/O_{3-0}) of data to be clocked in on every rising edge of SCK. With the Quad-Output Read commands, the $\overline{WP}$ Pin becomes an output pin (I/O_2) in conjunction with other pins to allow four bits of data on ($I/O3_{3-0}$) to be clocked in on every falling edge of SCK. To maintain consistency with the SPI nomenclature, the $\overline{WP}$ (I/O_2) pin will be referenced as the $\overline{WP}$ pin unless specifically addressing the Quad-I/O modes in which case it will be referenced as I/O_2 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.	-	Input/Output
$\overline{HOLD}$ (I/O_3)	**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 "Hold Function" on page 34 for additional details on the Hold operation. With the Quad-Input Byte/Page Program command, the $\overline{HOLD}$ pin becomes an input pin (I/O_3) and, along with other pins, allows four bits (on I/O_{3-0}) of data to be clocked in on every rising edge of SCK. With the Quad-Output Read commands, the $\overline{HOLD}$ Pin becomes an output pin (I/O_3) in conjunction with other pins to allow four bits of data on ($I/O3_{3-0}$) to be clocked in on every falling edge of SCK. To maintain consistency with the SPI nomenclature, the $\overline{HOLD}$ (I/O_3) pin will be referenced as the $\overline{HOLD}$ pin unless specifically addressing the Quad-I/O modes in which case it will be referenced as I/O_3 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.	-	Input/Output
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 1-1. 8-SOIC (Top View)

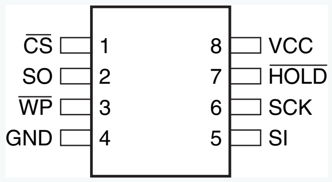
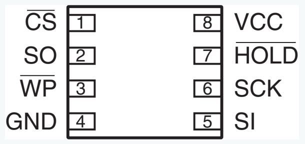
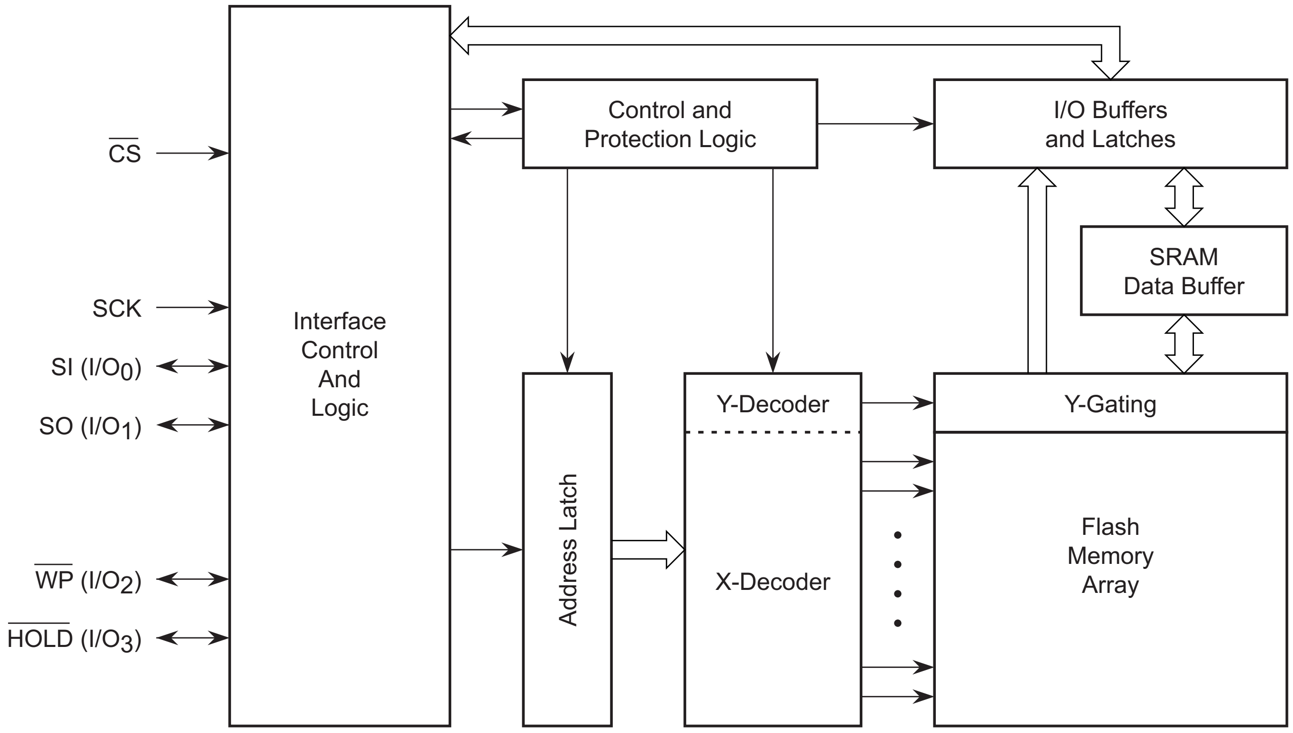


Figure 1-2. 8-UDFN (Top View)



2. Block Diagram

Figure 2-1. Block Diagram

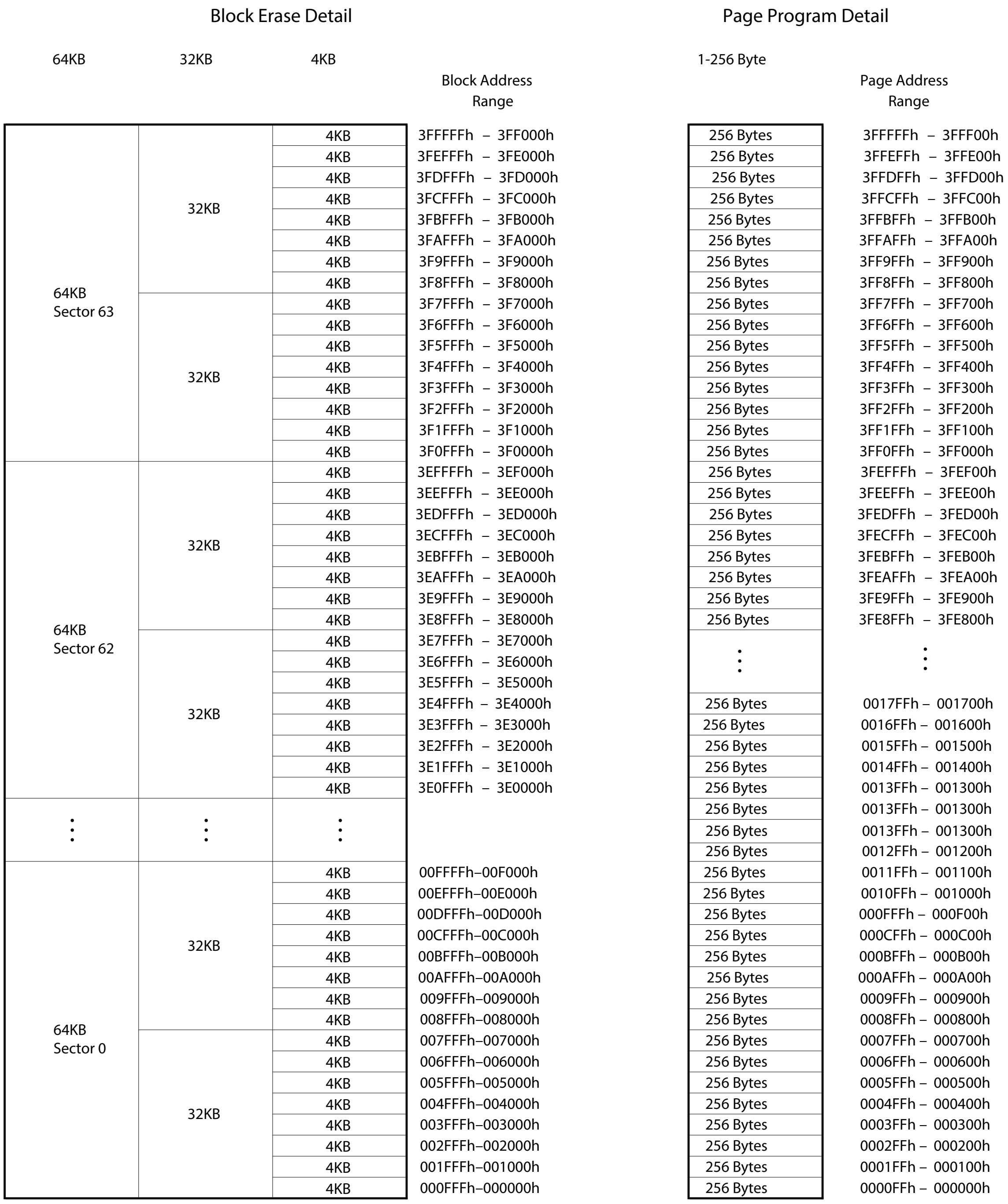


Note: I/O_{3-0} pin naming convention is used for Dual-I/O and Quad-I/O commands.

3. Memory Array

To provide the greatest flexibility, the memory array of the AT25SF321 can be erased in four levels of granularity including a full chip erase. The size of the erase blocks is optimized for both code and data storage applications, allowing both code and data segments to reside in their own erase regions. The Memory Architecture Diagram illustrates the breakdown of each erase level.

Figure 3-1. Memory Architecture Diagram



Block Erase Detail

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

Page Program Detail

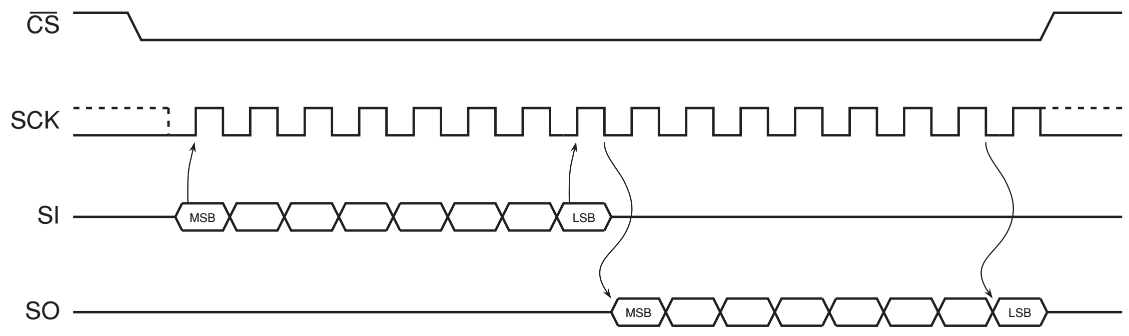
1-256 Byte	Page Address Range
256 Bytes	3FFFFFh – 3FFF00h
256 Bytes	3FFEFFh – 3FFE00h
256 Bytes	3FFDFFh – 3FFD00h
256 Bytes	3FFCFFh – 3FFC00h
256 Bytes	3FFBFFh – 3FFB00h
256 Bytes	3FFAFFh – 3FFA00h
256 Bytes	3FF9FFh – 3FF900h
256 Bytes	3FF8FFh – 3FF800h
256 Bytes	3FF7FFh – 3FF700h
256 Bytes	3FF6FFh –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

4. Device Operation

The AT25SF321 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 AT25SF321 via the SPI bus which is comprised of four signal lines: Chip Select ($\overline{\text{CS}}$), Serial Clock (SCK), Serial Input (SI), and Serial Output (SO).

The SPI protocol defines a total of four modes of operation (mode 0, 1, 2, or 3) with each mode differing in respect to the SCK polarity and phase and how the polarity and phase control the flow of data on the SPI bus. The AT25SF321 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 4-1. SPI Mode 0 and 3



4.1 Dual Output Read

The AT25SF321 features a Dual-Output Read mode that allow two bits of data to be clocked out of the device every clock cycle to improve throughput. To accomplish this, both the SI and SO pins are utilized as outputs for the transfer of data bytes. With the Dual-Output Read Array command, the SI pin becomes an output along with the SO pin.

4.2 Quad Output Read

The AT25SF321 features a Quad-Output Read mode that allow four bits of data to be clocked out of the device every clock cycle to improve throughput. To accomplish this, the SI, SO, $\overline{\text{WP}}$, $\overline{\text{HOLD}}$ pins are utilized as outputs for the transfer of data bytes. With the Quad-Output Read Array command, the SI, $\overline{\text{WP}}$, $\overline{\text{HOLD}}$ pins become outputs along with the SO pin.

5. Commands and Addressing

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

Opcodes not supported by the AT25SF321 will be ignored by the device and no operation will be started. The device will continue to ignore any data presented on the SI pin until the start of the next operation ($\overline{\text{CS}}$ pin being deasserted and then reasserted). In addition, if the $\overline{\text{CS}}$ pin is deasserted before complete opcode and address information is sent to the device, then no operation will be performed and the device will simply return to the idle state and wait for the next operation.

Addressing of the device requires a total of three bytes of information to be sent, representing address bits A23-A0. Since the upper address limit of the AT25SF321 memory array is 3FFFFFh, address bits A23-A22 are always ignored by the device.

Table 5-1. **Command Listing**

Command	Opcode		Clock Frequency	Address Bytes	Dummy Bytes	Data Bytes	Section Link
Read Commands							
Read Array	0Bh	0000 1011	Up to 85 MHz	3	1	1+	6.1
	03h	0000 0011	Up to 50 MHz	3	0	1+	
Dual Output Read	3Bh	0011 1011	Up to 85 MHz	3	1	1+	6.2
Dual I/O Read	BBh	1011 1011	Up to 85 MHz	3	0	1+	6.3
Quad Output Read	6Bh	0110 1011	Up to 85 MHz	3	1	1+	6.4
Quad I/O Read	EBh	1110 1011	Up to 85 MHz	3	1	1+	6.5
Continuous Read Mode Reset - Dual	FFFFh	1111 1111 1111 1111	Up to 104 MHz	0	0	0	6.6
Continuous Read Mode Reset - Quad	FFh	1111 1111	Up to 104 MHz	0	0	0	6.6
Program and Erase Commands							
Block Erase (4 Kbytes)	20h	0010 0000	Up to 104 MHz	3	0	0	7.2
Block Erase (32 Kbytes)	52h	0101 0010	Up to 104 MHz	3	0	0	
Block Erase (64 Kbytes)	D8h	1101 1000	Up to 104MHz	3	0	0	
Chip Erase	60h	0110 0000	Up to 104 MHz	0	0	0	7.3
	C7h	1100 0111	Up to 104 MHz	0	0	0	
Byte/Page Program (1 to 256 Bytes)	02h	0000 0010	Up to 104 MHz	3	0	1+	7.1
Program/Erase Suspend	75h	0111 0101	Up to 104 MHz	0	0	0	7.4
Program/Erase Resume	7Ah	0111 1010	Up to 104 MHz	0	0	0	7.5
Protection Commands							
Write Enable	06h	0000 0110	Up to 104 MHz	0	0	0	8.1
Write Disable	04h	0000 0100	Up to 104 MHz	0	0	0	8.2
Security Commands							
Erase Security Register Page	44h	0100 0100	Up to 104 MHz	3	0	0	9.1
Program Security Register Page	42h	0100 0010	Up to 104 MHz	3	0	1+	9.2
Read Security Register Page	48h	0100 1000	Up to 85MHz	3	1	1+	9.3
Status Register Commands							
Read Status Register Byte 1	05h	0000 0101	Up to 104 MHz	0	0	1	10.1
Read Status Register Byte 2	35h	0011 0101	Up to 104 MHz	0	0	1	
Write Status Register	01h	0000 0001	Up to 104 MHz	0	0	1 or 2	10.2
Write Enable for Volatile Status Register	50h	0101 0000	Up to 104MHz	0	0	0	10.3

Table 5-1. Command Listing

Command	Opcode		Clock Frequency	Address Bytes	Dummy Bytes	Data Bytes	Section Link
Miscellaneous Commands							
Read Manufacturer and Device ID	9Fh	1001 1111	Up to 104MHz	0	0	3	11.1
Read ID	90h	1001 0000	Up to 104 MHz	0	3	2	11.2
Deep Power-Down	B9h	1011 1001	Up to 104 MHz	0	0	0	11.3
Resume from Deep Power-Down	ABh	1010 1011	Up to 104 MHz	0	0	0	11.4
Resume from Deep Power-Down and Read ID	ABh	1010 1011	Up to 104 MHz	0	3	1	11.4

6. Read Commands

6.1 Read Array (0Bh and 03h)

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 is specified. The device incorporates an internal address counter that automatically increments 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 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}.

To perform the Read Array operation, the $\overline{\text{CS}}$ 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. Following the three address bytes, an additional dummy byte needs to be clocked into the device if the 0Bh opcode is used for the Read Array operation.

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

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

Figure 6-1. Read Array - 03h Opcode

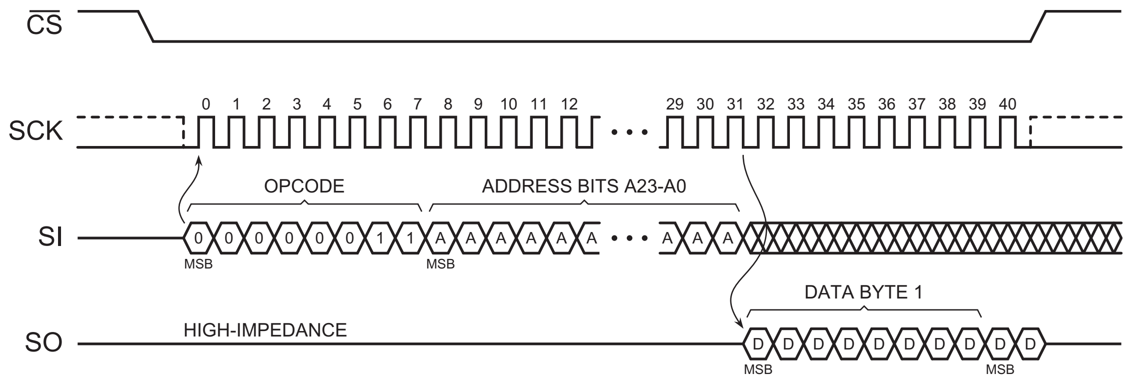
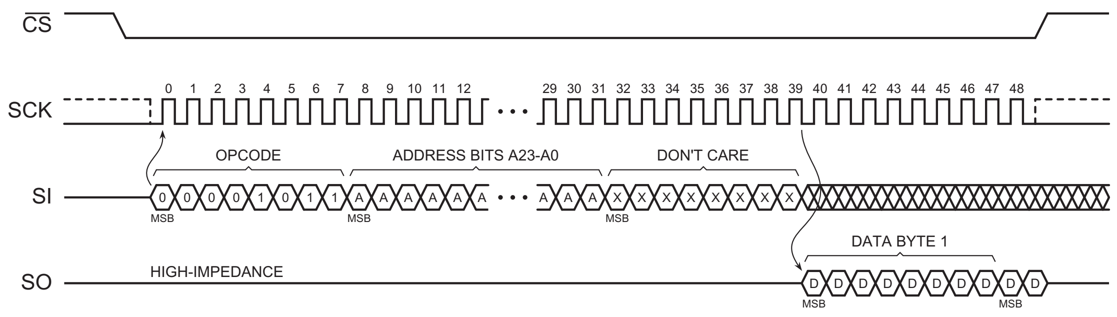


Figure 6-2. Read Array - 0Bh Opcode



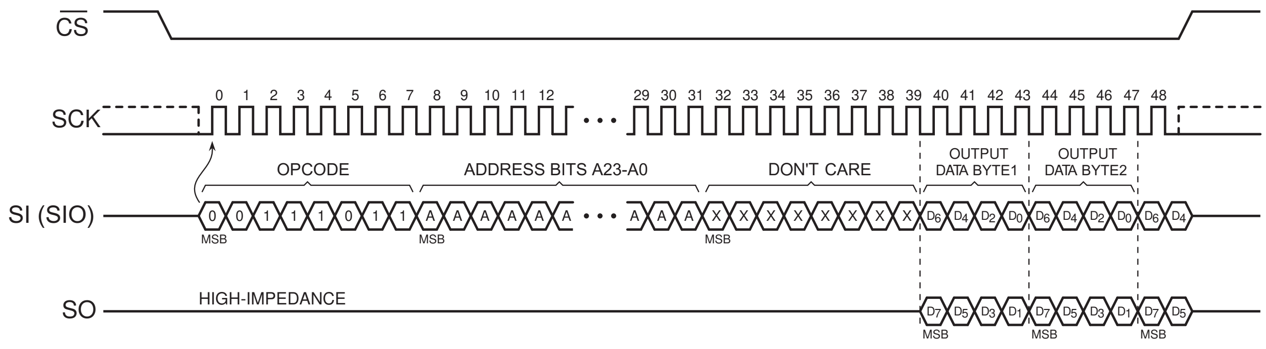
6.2 Dual-Output Read Array (3Bh)

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 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 SI 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 seven of the first data byte is output on the SO pin, while bit six of the same data byte is output on the SI pin. During the next clock cycle, bits five and four of the first data byte are output on the SO and SI pins, respectively. The sequence continues with each byte of data being output after every four clock cycles. When the last byte (3FFFFFh) of the memory array has been read, the device will continue reading 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 SI 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 6-3. Dual-Output Read Array



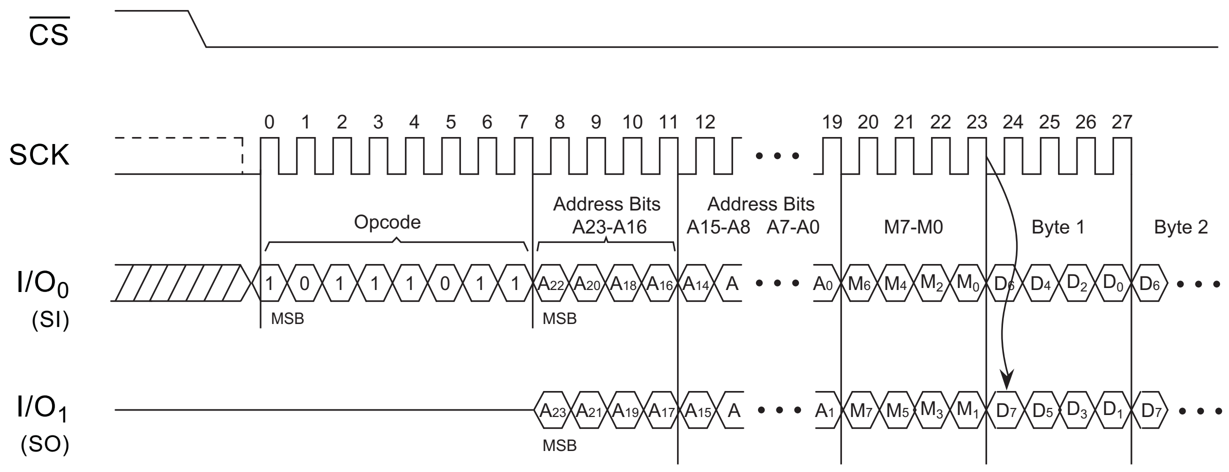
6.3 Dual-I/O Read Array (BBh)

The Dual-I/O Read Array command is similar to the Dual-Output Read Array command and can be used to sequentially read a continuous stream of data from the device by simply providing the clock signal once the initial starting address with two bits of address on each clock and two bits of data on every clock cycle.

The Dual-I/O Read Array command can be used at any clock frequency, up to the maximum specified by f_{RDDO}. To perform the Dual-I/O Read Array operation, the $\overline{CS}$ pin must first be asserted and then the opcode BBh 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 mode byte must also be clocked into the device.

After the three address bytes and the mode byte have been clocked in, additional clock cycles will result in data being output on both the SO and SI 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 seven of the first data byte is output on the SO pin, while bit six of the same data byte is output on the SI pin. During the next clock cycle, bits five and four of the first data byte are output on the SO and SI pins, respectively. The sequence continues with each byte of data being output after every four clock cycles. When the last byte (3FFFFFh) of the memory array has been read, the device will continue reading 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 SI 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.

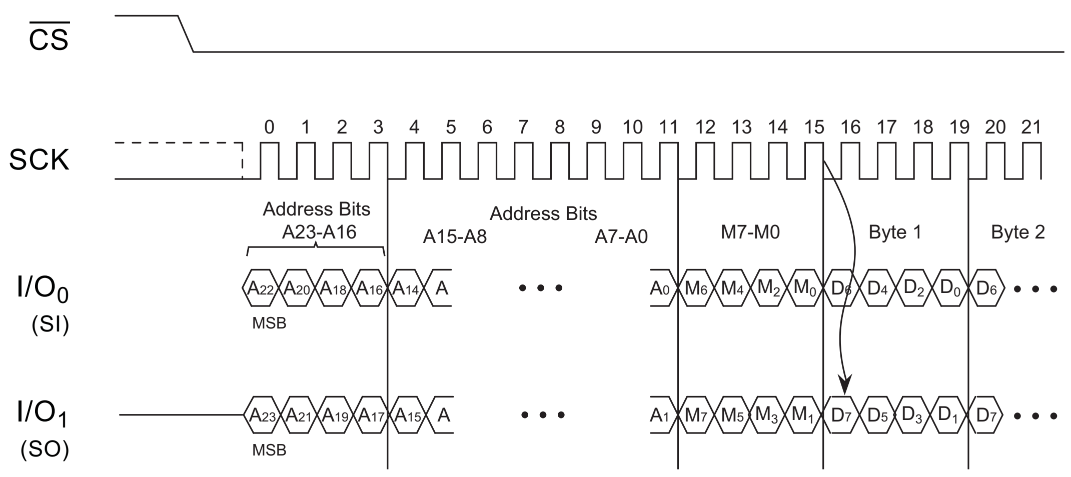
Figure 6-4. Dual I/O Read Array (Initial command or previous M5, M4≠1,0)



6.3.1 Dual-I/O Read Array (BBh) with Continuous Read Mode

The Fast Read Dual I/O command can further reduce instruction overhead through setting the Continuous Read Mode bits (M7-0) after the input Address bits (A23-0), as shown in Figure 6-5. The upper nibble of the (M7-4) controls the length of the next Fast Read Dual I/O command through the inclusion or exclusion of the first byte instruction code. The lower nibble bits of the (M3-0) are don't care ("x"). However, the IO pins should be high-impedance prior to the falling edge of the first data out clock. If the "Continuous Read Mode" bits M5-4 = (1,0), then the next Fast Read Dual I/O command (after $\overline{CS}$ is raised and then lowered) does not require the BBH instruction code, as shown in Figure 15. This reduces the command sequence by eight clocks and allows the Read address to be immediately entered after $\overline{CS}$ is asserted low. If the "Continuous Read Mode" bits M5-4 do not equal to (1,0), the next command (after $\overline{CS}$ is raised and then lowered) requires the first byte instruction code, thus returning to normal operation. A Continuous Read Mode Reset command can also be used to reset (M7-0) before issuing normal commands.

Figure 6-5. Dual-I/O Read Array (Previous command set M5, M4 = 1,0)



6.4 Quad-Output Read Array (6Bh)

The Quad-Output Read Array command is similar to the Dual-Output Read Array command. The Quad-Output Read Array command allows four bits of data to be clocked out of the device on every clock cycle, rather than just one or two.

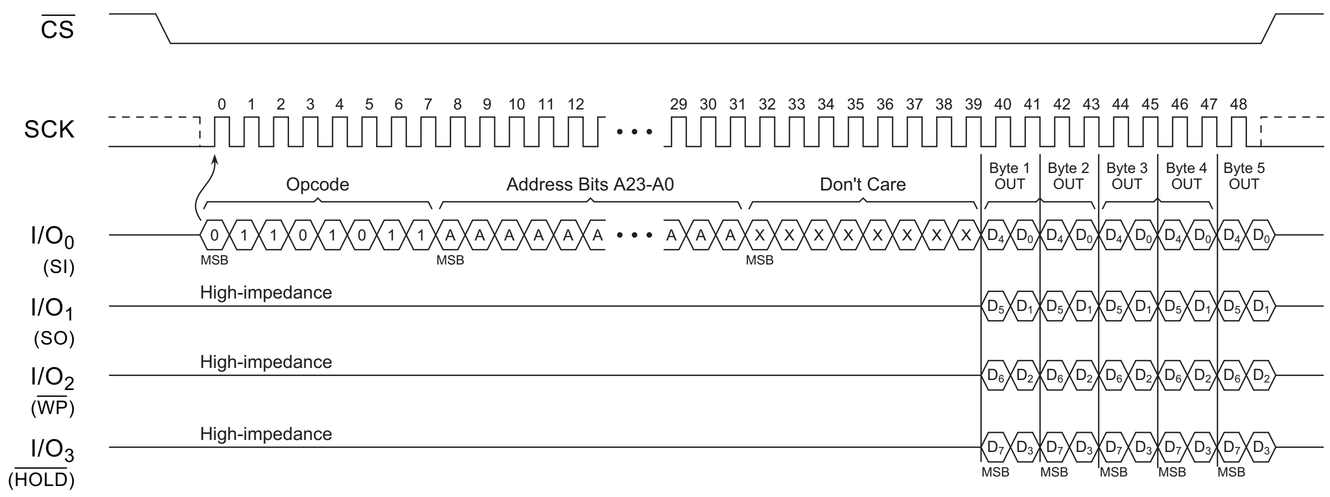
The Quad Enable bit (QE) of the Status Register must be set to enable for the Quad-Output Read Array instruction.

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

After the three address bytes and the dummy byte have been clocked in, additional clock cycles will result in data being output on the I/O_{3-0} pins. The data is always output with the MSB of a byte first and the MSB is always output on the I/O_3 pin. During the first clock cycle, bit 7 of the first data byte will be output on the I/O_3 pin while bits 6, 5, and 4 of the same data byte will be output on the I/O_2, I/O_1, and I/O_0 pins, respectively. During the next clock cycle, bits 3, 2, 1, and 0 of the first data byte will be output on the I/O_3, I/O_2, I/O_1 and I/O_0 pins, respectively.

The sequence continues with each byte of data being output after every two clock cycles. When the last byte (3FFFFFh) of the memory array has been read, the device will continue reading 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 $\overline{WP}$, $\overline{HOLD}$, SO, SI 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 6-6. Quad-Output Read Array



6.5 Quad-I/O Read Array(EBh)

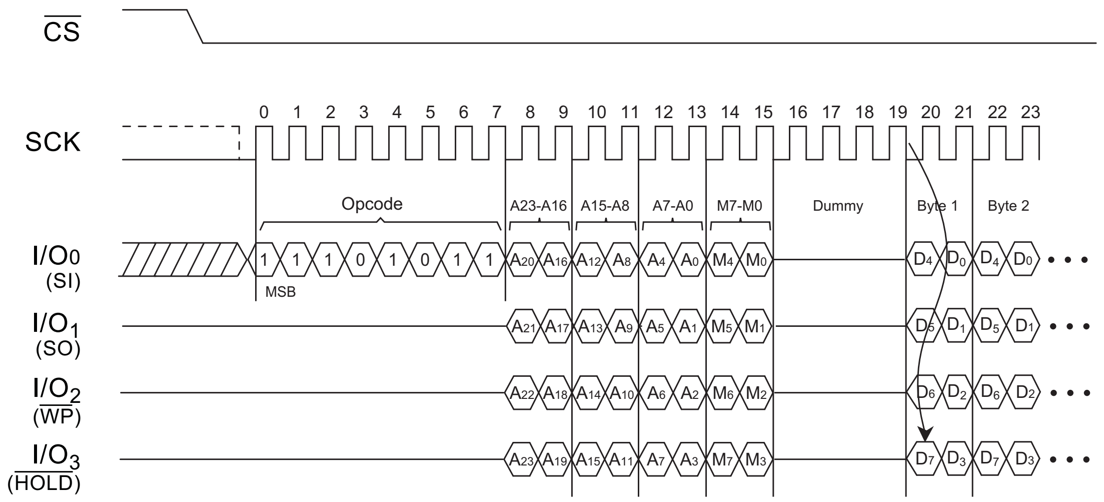
The Quad-I/O Read Array command is similar to the Quad-Output Read Array command. The Quad-I/O Read Array command allows four bits of address to be clocked into the device on every clock cycle, rather than just one.

The Quad-I/O Read Array command can be used at any clock frequency, up to the maximum specified by f_{RDQO}. To perform the Quad-I/O Read Array operation, the $\overline{CS}$ pin must first be asserted and then the opcode EBh 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 mode byte must also be clocked into the device.

After the three address bytes, the mode byte and two dummy bytes have been clocked in, additional clock cycles will result in data being output on the $I/O_{3\text{-}0}$ pins. The data is always output with the MSB of a byte first and the MSB is always output on the I/O_3 pin. During the first clock cycle, bit 7 of the first data byte will be output on the I/O_3 pin while bits 6, 5, and 4 of the same data byte will be output on the I/O_2, I/O_1 and I/O_0 pins, respectively. During the next clock cycle, bits 3, 2, 1, and 0 of the first data byte will be output on the I/O_3, I/O_2, I/O_1 and I/O_0 pins, respectively. The sequence continues with each byte of data being output after every two clock cycles.

When the last byte (3FFFFFh) of the memory array has been read, the device will continue reading 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 I/O_3, I/O_2, I/O_1 and I/O_0 pins into a high-impedance state. The $\overline{CS}$ pin can be deasserted at any time and does not require that a full byte of data be read.The Quad Enable bit (QE) of the Status Register must be set to enable for the Quad-I/O Read Array instruction.

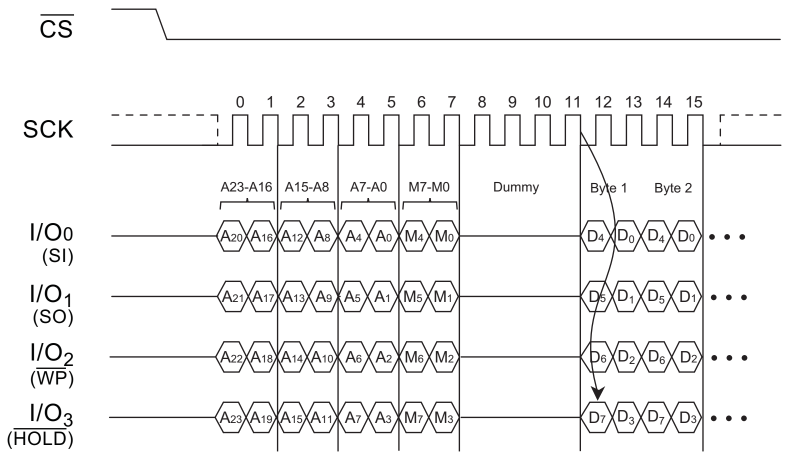
Figure 6-7. Quad-I/O Read Array (Initial command or previous M5, M4 ≠ 1,0)



6.5.1 Quad-I/O Read Array (EBh) with Continuous Read Mode

The Fast Read Quad I/O command can further reduce instruction overhead through setting the Continuous Read Mode bits (M7-0) after the input Address bits (A23-0), as shown in Figure 6-6. The upper nibble (M7-4) of the Continuous Read Mode bits controls the length of the next Fast Read Quad I/O command through the inclusion or exclusion of the first byte instruction code. The lower nibble bits (M3-0) of the Continuous Read Mode bits are don't care. However, the IO pins should be high-impedance prior to the falling edge of the first data out clock. If the Continuous Read Mode bits M5-4 = (1,0), then the next Quad-I/O Read Array command (after $\overline{CS}$ is raised and then lowered) does not require the EBh instruction code, as shown in Fig 6-8. This reduces the command sequence by eight clocks and allows the Read address to be immediately entered after $\overline{CS}$ is asserted low. If the "Continuous Read Mode" bits M5-4 do not equal to (1,0), the next command (after $\overline{CS}$ is raised and then lowered) requires the first byte instruction code, thus returning to normal operation. A Continuous Read Mode Reset command can also be used to reset (M7-0) before issuing normal commands.

Figure 6-8. Quad I/O Read Array with Continuous Read Mode (Previous Command Set M5-4 =1,0)



6.6 Continuous Read Mode Reset (FFh or FFFFh)

The Continuous Read Mode bits are used in conjunction with the Dual I/O Read Array and the Quad I/O Read Array commands to provide the highest random Flash memory access rate with minimum SPI instruction overhead, thus allowing more efficient XIP (execute in place) with this device family.

The "Continuous Read Mode" bits M7-0 are set by the Dual/Quad I/O Read Array commands. M5-4 are used to control whether the 8-bit SPI instruction code (BBh or EBh) is needed or not for the next instruction. When M5-4 = (1,0), the next instruction will be treated the same as the current Dual/Quad I/O Read Array command without needing the 8-bit instruction code. When M5-4 do not equal (1,0), the device returns to normal SPI instruction mode, in which all instructions can be accepted. M7-6 and M3-0 are reserved bits for future use; either 0 or 1 values can be used.

See Figure 6-9, the Continuous Read Mode Reset instruction (FFh or FFFFh) can be used to set M4 = 1, thus the device will release the Continuous Read Mode and return to normal SPI operation.

To reset Continuous Read Mode during Quad I/O operation, only eight clocks are needed to shift in instruction FFh. To reset Continuous Read Mode during Dual I/O operation, sixteen clocks are needed to shift in instruction FFFFh.

Figure 6-9. Continuous Read Mode Reset (Quad)

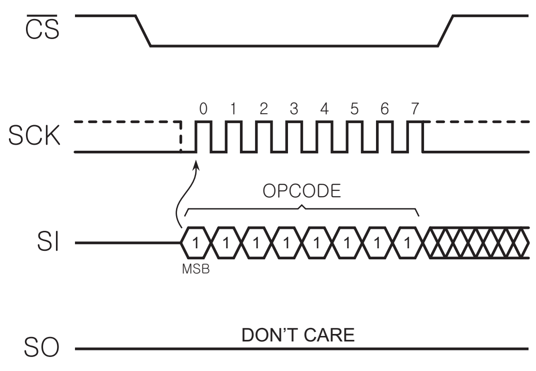
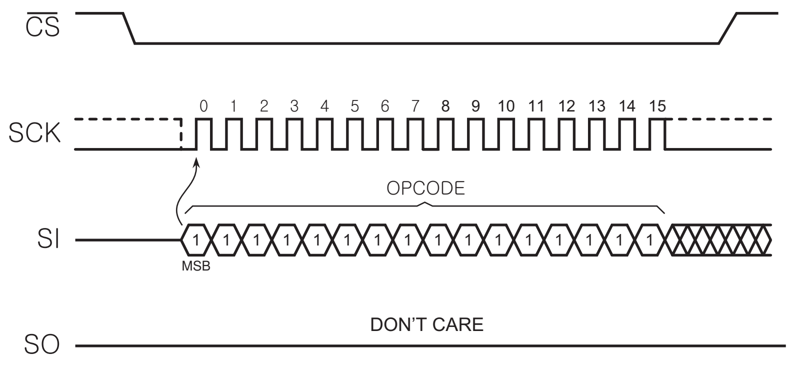


Figure 6-10. Continuous Read Mode Reset (Dual)



7. Program and Erase Commands

7.1 Byte/Page Program (02h)

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 (06h)" on page 20) to set the Write Enable Latch (WEL) bit of the Status Register to a logical "1" state.

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

If the starting memory address denoted by A23-A0 does not fall on an even 256-byte page boundary (A7-A0 are not all 0), then special circumstances regarding which memory locations to be programmed will apply. In this situation, any data that is sent to the device that goes beyond the end of the page will wrap around back to the beginning of the same page. 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 not be programmed and will remain in the erased state (FFh). In addition, if more than 256 bytes of data are sent to the device, then only the last 256 bytes sent will be latched into the internal buffer.

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

The three address bytes and at least one complete byte of data must be clocked into the device before the $\overline{CS}$ pin is deasserted, and the $\overline{CS}$ pin must be deasserted on even byte boundaries (multiples of eight bits); otherwise, the device will abort the operation and no data will be programmed into the memory array. In addition, if the memory is in the protected state (see "Non-Volatile Protection" on page 21), 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.

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.

Figure 7-1. Byte Program

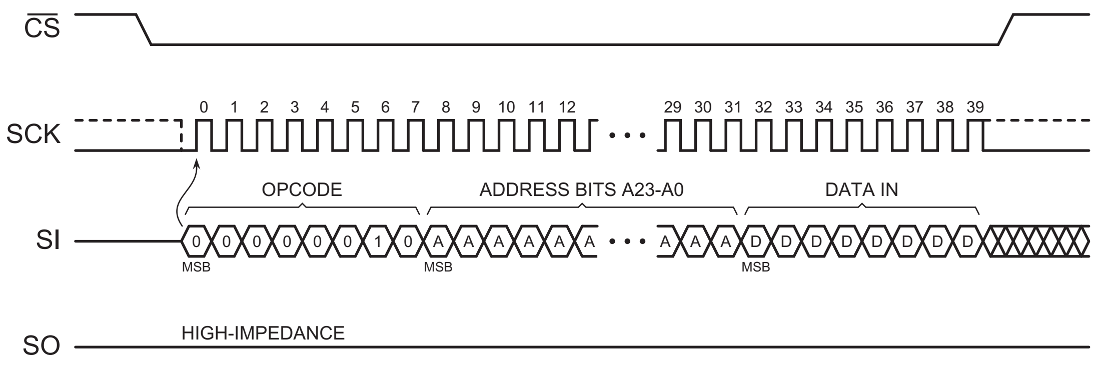
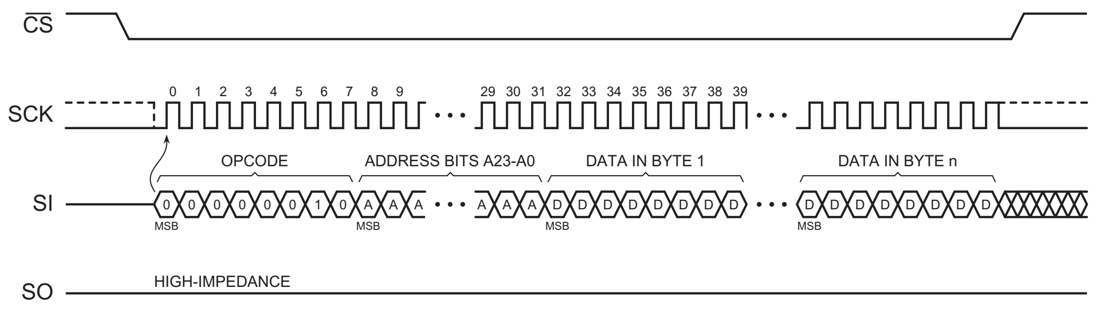


Figure 7-2. Page Program



7.2 Block Erase (20h, 52h, or D8h)

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, or 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 $\overline{\text{CS}}$ 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 4- or 32- or 64-Kbyte block to be erased must be clocked in. Any additional data clocked into the device will be ignored. When the $\overline{\text{CS}}$ 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}.

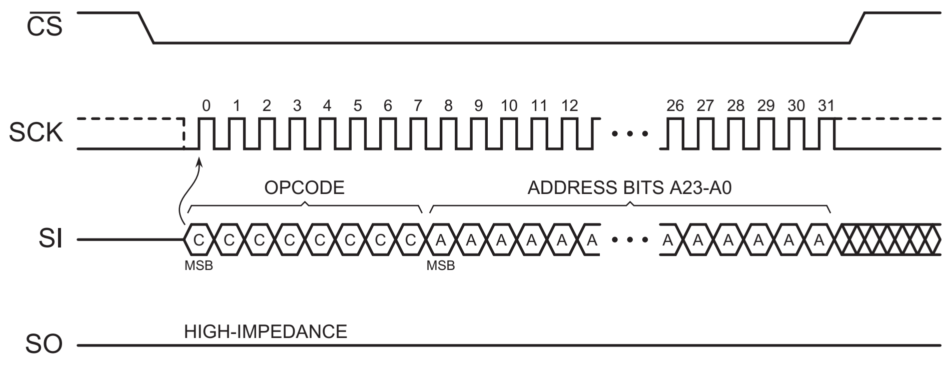
Since the Block Erase command erases a region of bytes, the lower order address bits do not need to be decoded by the device. Therefore, for a 4-Kbyte erase, address bits A11-A0 will be ignored by the device and their values can be either a logical "1" or "0". For a 32-Kbyte erase, address bits A14-A0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

If the memory 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 $\overline{\text{CS}}$ 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.

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.

Figure 7-3. Block Erase



7.3 Chip Erase (60h or C7h)

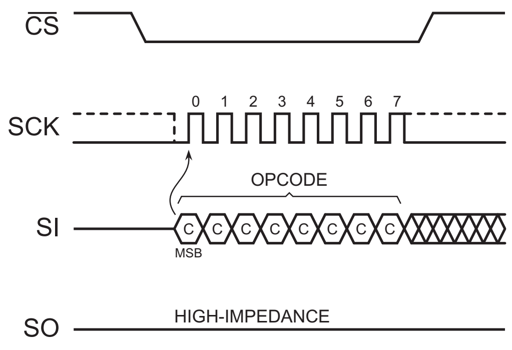
The entire memory array can be erased in a single operation by using the Chip Erase command. Before a Chip Erase command can be started, the Write Enable command must have been previously issued to the device to set the WEL bit of the Status Register to a logical "1" state.

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 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 byte boundary (multiples of eight bits); otherwise, no erase will be performed. In addition, if 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 the $\overline{CS}$ pin is deasserted on uneven byte boundaries or if the memory 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.

Figure 7-4. Chip Erase



7.4 Program/Erase Suspend (75h)

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 be suspended so that other device operations can be performed. For example, by suspending an erase operation to a particular block, the system can perform functions such as a program or read to a different block.

Chip Erase cannot be suspended. The Program/Erase Suspend command will be ignored if it is issued during a Chip Erase.A program operation can be performed while an erase operation is suspended, but the program operation cannot be suspended while an erase operation is currently suspended.

Other device operations, such as a Read Status Register, can also be performed while a program or erase operation is suspended. Table 7-4 outlines the operations that are allowed and not allowed during a program or erase suspend.

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

To perform a Program/Erase Suspend, the $\overline{CS}$ pin must first be asserted and the opcode of 75h must be clocked into the device. No address bytes need to be clocked into the device, and any data clocked in after the opcode will be ignored. When the $\overline{CS}$ pin is deasserted, the program or erase operation currently in progress will be suspended within a time of t_{SUSE}. The Suspend (SUS) bit in the Write Status Register will be set to the logical "1" state to indicate that the program or erase operation has been suspended. In addition, the $\overline{RDY}$/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 $\overline{CS}$ pin is deasserted, and the $\overline{CS}$ pin must be deasserted on a byte boundary (multiples of eight bits). Otherwise, no suspend operation will be performed.

If a read operation is attempted to a suspended area (page for programming or block for erasing), then the device will output undefined data. Therefore, when performing a Read Array operation to an unsuspended area and the device's internal address counter increments and crosses into the suspended area, the device will then start outputting undefined data until the internal address counter crosses to an unsuspended area.

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

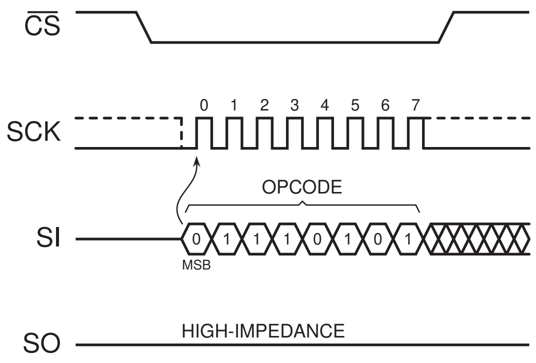
If an attempt is made to perform an operation that is not allowed during a program or erase suspend, such as a Write Status Register operation, then the device will simply ignore the opcode and no operation will be performed. The state of the WEL bit in the Status Register will not be affected.

Table 7-1. Operations Allowed and Not Allowed During a Program/Erase Suspend Command

Command	During Program Suspend	During Erase Suspend
Read Commands		
Read Array (03h, 0Bh, 3Bh, BBh, 6Bh, EBh)	Allowed [1]	Allowed[1]
Continuous Read Reset (FFh)	Allowed[1]	Allowed[1]
Program and Erase Commands		
Block Erase (20h, 52h, D8h)	Not Allowed	Not Allowed
Chip Erase (C7h, 60h)	Not Allowed	Not Allowed
Byte/Page Program (02h)	Not Allowed	Allowed[1]
Program/Erase Suspend (75h)	Not Allowed	Not Allowed
Program/Erase Resume (7Ah)	Allowed	Allowed
Protection Commands		
Write Enable (06h)	Allowed	Allowed
Write Disable (04h)	Allowed	Allowed
Security Commands		
Erase Security Register Page (44h)	Not Allowed	Not Allowed
Program Security Register Page (42h)	Not Allowed	Not Allowed
Read Security Register Page (48h)	Allowed	Allowed
Status Register Commands		
Read Status Register (05h, 35h)	Allowed	Allowed
Write Status Register (01h)	Not Allowed	Not Allowed
Volatile Write Enable Status Register (50h)	Not Allowed	Not Allowed
Miscellaneous Commands		
Read Manufacturer and Device ID (9Fh)	Allowed	Allowed
Read ID (90h)	Allowed	Allowed
Deep Power-Down (B9h)	Not Allowed	Not Allowed
Resume from Deep Power-Down (ABh)	Allowed [2]	Allowed[2]

1. Allowed for all 64K-byte blocks other than the one currently suspended.
2. Allowed for reading Device ID.

Figure 7-5. Program/Erase Suspend



7.5 Program/Erase Resume (7Ah)

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. The Program/Erase Resume instruction will be accepted by the device only if the SUS bit in the Write Status Register equals 1 and the $\overline{\text{RDY}}$/BSY bit equals 0. If the SUS bit equals 0 or the $\overline{\text{RDY}}$/BSY bit equals to 1, the Program/Erase Resume command will be ignored by the device. 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.

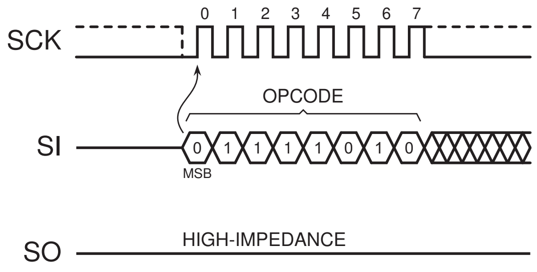
To perform Program/Erase Resume, the $\overline{\text{CS}}$ pin must first be asserted and opcode 7Ah must be clocked into the device.

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 resume within a time of t_{RES}. The SUS bit in the Status Register will then be reset back to the logical "0" state to indicate the program or erase operation is no longer suspended. In addition, the $\overline{\text{RDY}}$/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 a byte boundary (multiples of eight bits). Otherwise, no resume operation will perform.

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 $\overline{\text{RDY}}$/BSY bit or the SUS bit in the Status Register to determine if the previously suspended program or erase operation has resumed.

Figure 7-6. Program/Erase Resume.



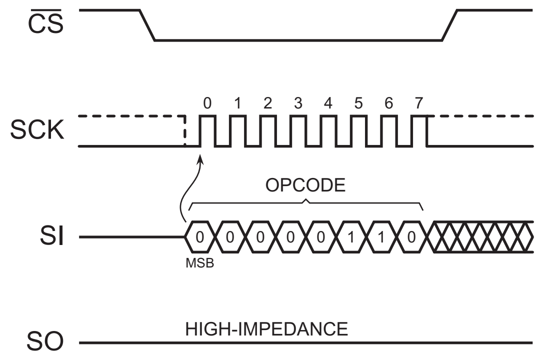
8. Protection Commands and Features

8.1 Write Enable (06h)

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, Program Security Register Pages, Erase Security Register Pages 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 byte boundary (multiples of eight bits); otherwise, the device will abort the operation and the WEL bit state will not change.

Figure 8-1. Write Enable

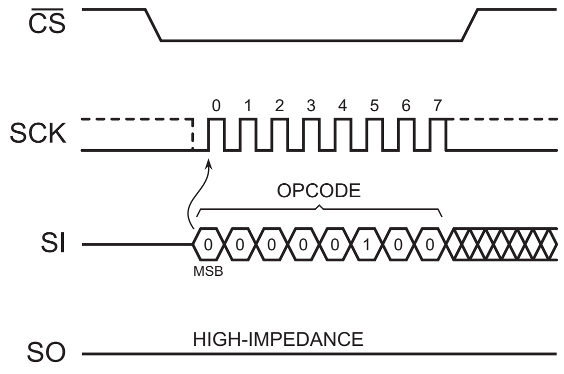


8.2 Write Disable (04h)

The Write Disable command is us0d 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, Program Security Register Page, and Write Status Register commands will not be executed. Other conditions can also cause the WEL bit to be reset; for more details, refer to the WEL bit section of the Status Register description.

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

Figure 8-2. Write Disable



8.3 Non-Volatile Protection

The device can be software protected against erroneous or malicious program or erase operations by utilizing the Non-Volatile Protection feature of the device. Non-Volatile Protection can be enabled or disabled by using the Write Status Register command to change the value of the Protection (CMP, SEC, TB, BP2, BP1, BP0) bits in the Status Register. The following table outlines the states of the Protection bits and the associated protection area

.

Table 8-1. Memory Array with CMP=0

Protection Bits					Memory Content	
SEC	**TB**	**BP2**	**BP1**	**BP0**	**Address Range**	**Portion**
X	X	0	0	0	None	None
0	0	0	0	1	3F0000h-3FFFFFh	Upper 1/64
0	0	0	1	0	3E0000h-3FFFFFh	Upper 1/32
0	0	0	1	1	3C0000h-3FFFFFh	Upper 1/16
0	0	1	0	0	380000h-3FFFFFh	Upper 1/8
0	0	1	0	1	300000h-3FFFFFh	Upper 1/4
0	0	1	1	0	200000h-3FFFFFh	Upper 1/2
0	1	0	0	1	000000h-00FFFFh	Lower 1/64
0	1	0	1	0	000000h-01FFFFh	Lower 1/32
0	1	0	1	1	000000h-03FFFFh	Lower 1/16
0	1	1	0	0	000000h-07FFFFh	Lower 1/8
0	1	1	0	1	000000h-0FFFFFh	Lower 1/4
0	1	1	1	0	000000h-1FFFFFh	Lower 1/2
X	X	1	1	1	000000h-3FFFFFh	ALL
1	0	0	0	1	3FF000h-3FFFFFh	Upper 1/512
1	0	0	1	0	3FE000h-3FFFFFh	Upper 1/256
1	0	0	1	1	3FC000h-3FFFFFh	Upper 1/128
1	0	1	0	X	3F8000h-3FFFFFh	Upper 1/64
1	0	1	1	0	3F8000h-3FFFFFh	Upper 1/32
1	1	0	0	1	000000h-000FFFh	Lower 1/512
1	1	0	1	0	000000h-001FFFh	Lower 1/256
1	1	0	1	1	000000h-003FFFh	Lower 1/128
1	1	1	0	X	000000h-007FFFh	Lower 1/64
1	1	1	1	0	000000h-007FFFh	Lower 1/32

Table 8-2. Memory Array Protection with CMP=1

Protection Bits					Memory Content	
SEC	TB	BP2	BP1	BP0	Address Range	Portion
X	X	0	0	0	000000h-3FFFFFh	All
0	0	0	0	1	000000h-3EFFFFh	Lower 63/64
0	0	0	1	0	000000h-3DFFFFh	Lower 31/32
0	0	0	1	1	000000h-3BFFFFh	Lower 15/16
0	0	1	0	0	000000h-37FFFFh	Lower 7/8
0	0	1	0	1	000000h-2FFFFFh	Lower 3/4
0	0	1	1	0	000000h-1FFFFFh	Lower 1/2
0	1	0	0	1	010000h-3FFFFFh	Upper 63/64
0	1	0	1	0	020000h-3FFFFFh	Upper 31/32
0	1	0	1	1	040000h-3FFFFFh	Upper 15/16
0	1	1	0	0	080000h-3FFFFFh	Upper 7/8
0	1	1	0	1	100000h-3FFFFFh	Upper 3/4
0	1	1	1	0	200000h-3FFFFFh	Upper 1/2
X	X	1	1	1	NONE	NONE
1	0	0	0	1	000000h-3FEFFFh	Lower 1023/1024
1	0	0	1	0	000000h-3FDFFFh	Lower 511/512
1	0	0	1	1	000000h-3FBFFFh	Lower 255/256
1	0	1	0	X	000000h-3F7FFFh	Lower 127/128
1	0	1	1	0	000000h-3F7FFFh	Lower 127/128
1	1	0	0	1	001000h-3FFFFFh	Upper 1023/1024
1	1	0	1	0	002000h-3FFFFFh	Upper 511/512
1	1	0	1	1	004000h-3FFFFFh	Upper 255/256
1	1	1	0	X	008000h-3FFFFFh	Upper 127/128
1	1	1	1	0	008000h-3FFFFFh	Upper 127/128

As a safeguard against accidental or erroneous protecting or unprotecting of the memory array, the Protection can be locked from updates by using the $\overline{\text{WP}}$ pin (see "Protected States and the Write Protect Pin" on page 22 for more details).

8.4 Protected States and the Write Protect Pin

The $\overline{\text{WP}}$ pin is not linked to the memory array itself and has no direct effect on the protection status of the memory array. Instead, the $\overline{\text{WP}}$ pin, is used to control the hardware locking mechanism of the device.

If the $\overline{\text{WP}}$ pin is permanently connected to GND, then the protection bits cannot be changed.

9. Security Register Commands

The device contains three extra pages called Security Registers that can be used for purposes such as unique device serialization, system-level Electronic Serial Number (ESN) storage, locked key storage, etc. The Security Registers are independent of the main Flash memory.

Each page of the Security Register can be erased and programmed independently. Each page can also be independently locked to prevent further changes.

9.1 Erase Security Registers (44h)

Before an erase Security Register Page 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 an Erase Security Register Page command, the $\overline{CS}$ pin must first be asserted and the opcode 44h must be clocked into the device. After the opcode has been clocked in, the three address bytes specifying the Security Register Page to be erased must be clocked in. 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_{BE}.

Since the Erase Security Register Page command erases a region of bytes, the lower order address bits do not need to be decoded by the device. Therefore address bits A7-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 right after the last address bit (A0); otherwise, the device will abort the operation and no erase operation will be performed.

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_{BE} time to determine if the device has finished erasing. At some point before the erase cycle completes, the $\overline{RDY}$/BSY bit in the Status Register will be reset back to the logical "0" state.

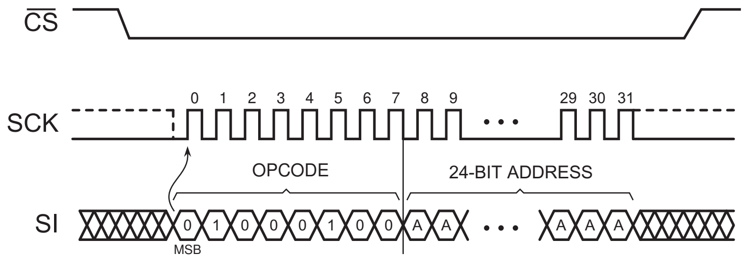
The WEL bit in the Status Register will be reset back to the logical "0" state if the erase cycle aborts due to an incomplete address being sent, the $\overline{CS}$ pin being deasserted on uneven byte boundaries, or because a memory location within the region to be erased is protected.

The Security Registers Lock Bits (LB3-LB1) in the Status Register can be used to OTP protect the security registers. Once a Lock Bit is set to 1, the corresponding Security Register will be permanently locked. The Erase Security Register Page instruction will be ignored for Security Registers which have their Lock Bit set.

Table 9-1. Security Register Addresses for Erase Security Register Page Command

Address	A23-A16	A15-A8	A7-A0
Security Register 1	00H	01H	Don't Care
Security Register 2	00H	02H	Don't Care
Security Register 3	00H	03H	Don't Care

Figure 9-1. Erase Security Register Page



9.2 Program Security Registers (42h)

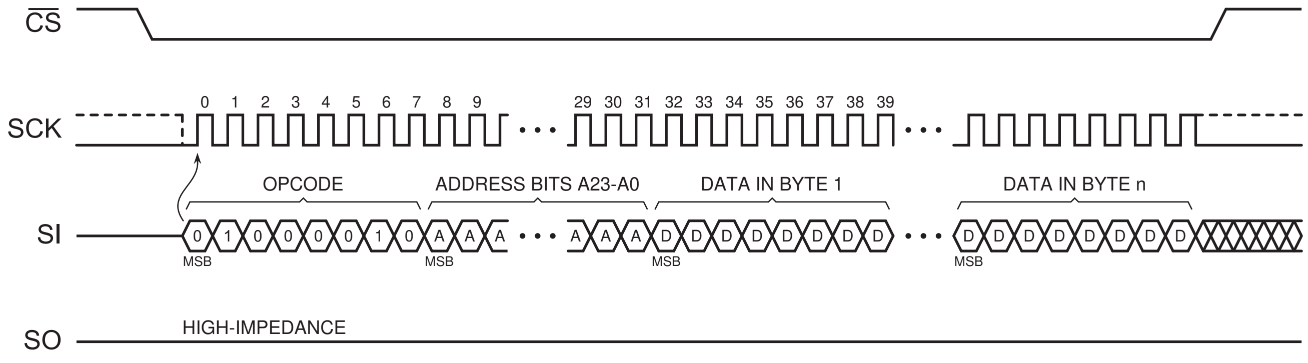
The Program Security Registers command utilizes the internal 256-byte buffer for processing. Therefore, the contents of the buffer will be altered from its previous state when this command is issued.

The Security Registers can be programmed in a similar fashion to the Program Array operation up to the maximum clock frequency specified by f_{CLK}. Before a Program Security Registers command can be started, the Write Enable command must have been previously issued to the device (see "Write Enable (06h)" on page 20) to set the Write Enable Latch (WEL) bit of the Status Register to a logical "1" state. To program the Security Registers, the $\overline{CS}$ pin must first be asserted and the opcode of 42h 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 program within the Security Register.

Table 9-2. Security Register Addresses for Program Security Registers Command

Address	A23-A16	A15-A8	A7-A0
Security Register 1	00H	01H	Byte Address
Security Register 2	00H	02H	Byte Address
Security Register 3	00H	03H	Byte Address

Figure 9-2. Program Security Registers



9.3 Read Security Registers (48h)

The Security Register can be sequentially read in a similar fashion to the Read Array operation up to the maximum clock frequency specified by f_{CLK}. To read the Security Register, the $\overline{CS}$ pin must first be asserted and the opcode of 48h 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 Security Register. Following the three address bytes, one dummy byte must be clocked into the device before data can be output.

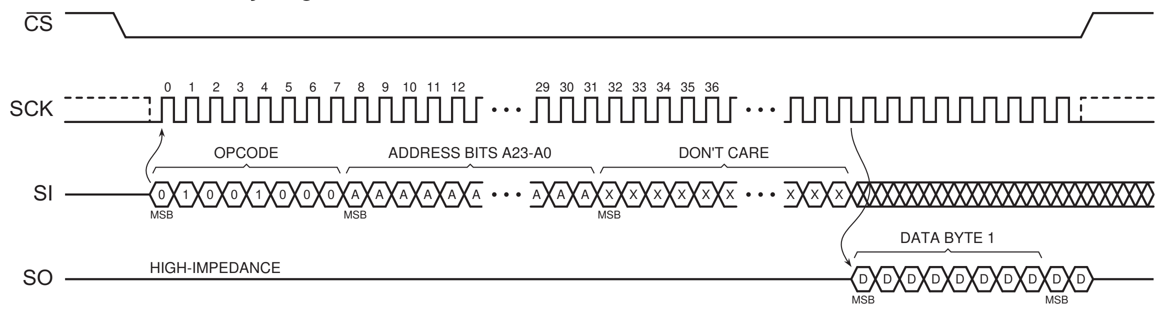
After the three address bytes and the dummy byte have been clocked in, additional clock cycles will result in Security Register data being output on the SO pin. When the last byte (0003FFh) of the Security Register has been read, the device will continue reading back at the beginning of the register (000000h). No delays will be incurred when wrapping around from the end of the register to the beginning of the register.

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

Table 9-3. Security Register Addresses for Read Security Registers Command

Address	A23-A16	A15-A8	A7-A0
Security Register 1	00H	01H	Byte Address
Security Register 2	00H	02H	Byte Address
Security Register 3	00H	03H	Byte Address

Figure 9-3. Read Security Registers



10. Status Register Commands

10.1 Read Status Register (05h and 35h)

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 Block Protection. The Status Register can be read at any time, including during an internally self-timed program or erase operation.

To read Status Register Byte 1, the $\overline{\text{CS}}$ pin must first be asserted and the opcode of 05h must be clocked into the device. After the opcode has been clocked in, the device will begin outputting Status Register Byte 1 data on the SO pin during every subsequent clock cycle. After the last bit (bit 0) of Status Register Byte 1 has been clocked out, the sequence will repeat itself starting again with bit 7 as long as the $\overline{\text{CS}}$ pin remains asserted and the clock pin is being pulsed. The data in the Status Register is constantly being updated, so each repeating sequence will output new data. Deasserting the $\overline{\text{CS}}$ pin will terminate the Read Status Register operation and put the SO pin into a high-impedance state. The $\overline{\text{CS}}$ pin can be deasserted at any time and does not require that a full byte of data be read.

To read Status Register Byte 2, the $\overline{\text{CS}}$ pin must first be asserted and the opcode of 35h must be clocked into the device. After the opcode has been clocked in, the device will begin outputting Status Register Byte 2 data on the SO pin during every subsequent clock cycle. After the last bit (bit 0) of Status Register Byte 2 has been clocked out, the sequence will repeat itself starting again with bit 7 as long as the $\overline{\text{CS}}$ pin remains asserted and the clock pin is being pulsed. The data in the Status Register is constantly being updated, so each repeating sequence will output new data. Deasserting the $\overline{\text{CS}}$ pin will terminate the Read Status Register operation and put the SO pin into a high-impedance state. The $\overline{\text{CS}}$ pin can be deasserted at any time and does not require that a full byte of data be read.