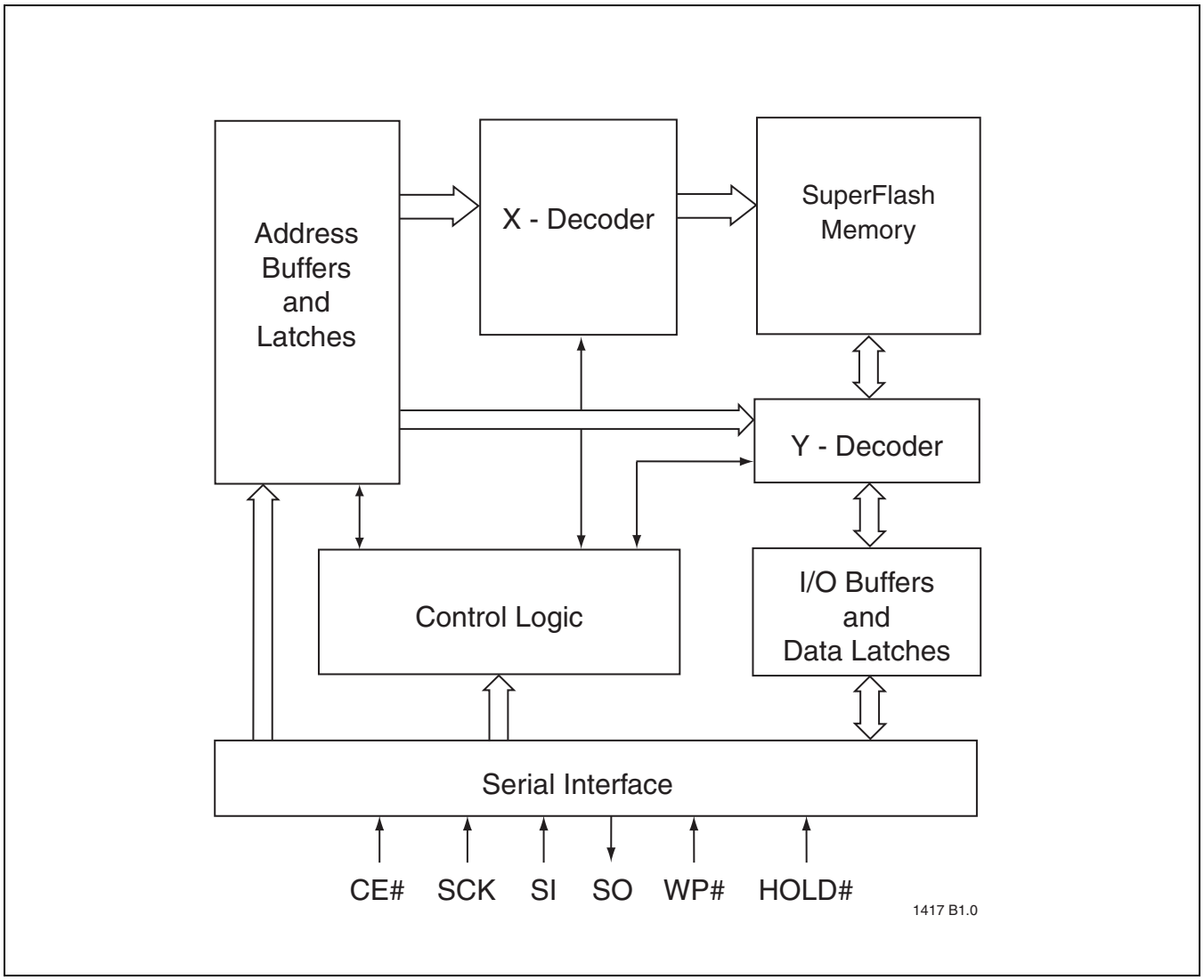

FIGURE 1: Functional Block Diagram

PIN DESCRIPTION

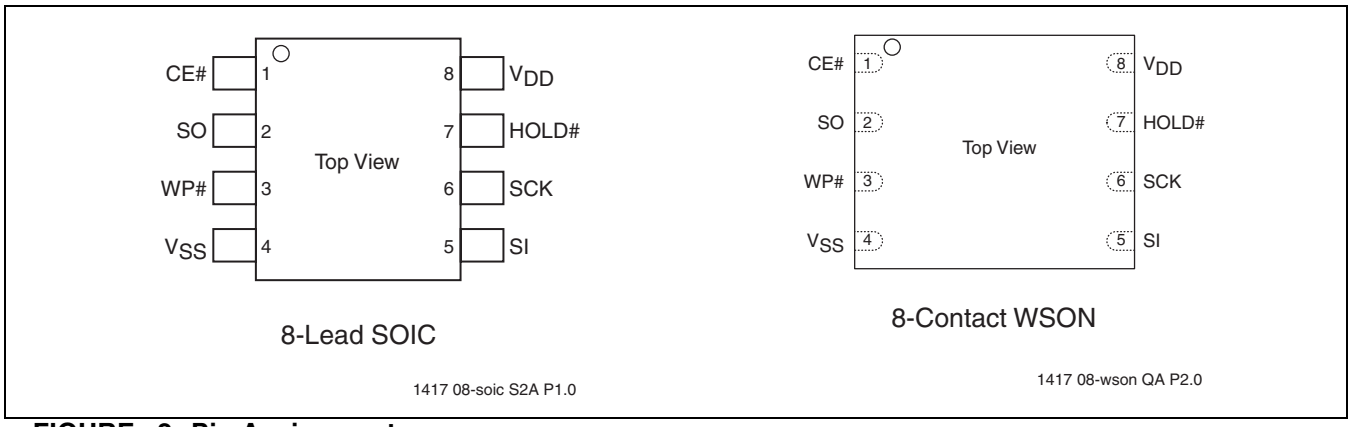

FIGURE 2: Pin Assignments

TABLE 1: Pin Description

Symbol	Pin Name	Functions
SCK	Serial Clock	To provide the timing of the serial interface. Commands, addresses, or input data are latched on the rising edge of the clock input, while output data is shifted out on the falling edge of the clock input.
SI	Serial Data Input	To transfer commands, addresses, or data serially into the device. Inputs are latched on the rising edge of the serial clock.
SO	Serial Data Output	To transfer data serially out of the device. Data is shifted out on the falling edge of the serial clock. Outputs Flash busy status during AAI Programming when reconfigured as RY/BY# pin. See "Hardware End-of-Write Detection" on page 12 for details.
CE#	Chip Enable	The device is enabled by a high to low transition on CE#. CE# must remain low for the duration of any command sequence.
WP#	Write Protect	The Write Protect (WP#) pin is used to enable/disable BPL bit in the status register.
HOLD#	Hold	To temporarily stop serial communication with SPI flash memory without resetting the device.
V_{DD}	Power Supply	To provide power supply voltage: 2.7-3.6V for SST25VF020B
V_{SS}	Ground	

T1.0 1417

MEMORY ORGANIZATION

The SST25VF020B SuperFlash memory array is organized in uniform 4 KByte erasable sectors with 32 KByte overlay blocks and 64 KByte overlay erasable blocks.

DEVICE OPERATION

The SST25VF020B is accessed through the SPI (Serial Peripheral Interface) bus compatible protocol. The SPI bus consist of four control lines; Chip Enable (CE#) is used to select the device, and data is accessed through the Serial Data Input (SI), Serial Data Output (SO), and Serial Clock (SCK).

The SST25VF020B supports both Mode 0 (0,0) and Mode 3 (1,1) of SPI bus operations. The difference between the two modes, as shown in Figure 3, is the state of the SCK signal when the bus master is in Stand-by mode and no data is being transferred. The SCK signal is low for Mode 0 and SCK signal is high for Mode 3. For both modes, the Serial Data In (SI) is sampled at the rising edge of the SCK clock signal and the Serial Data Output (SO) is driven after the falling edge of the SCK clock signal.

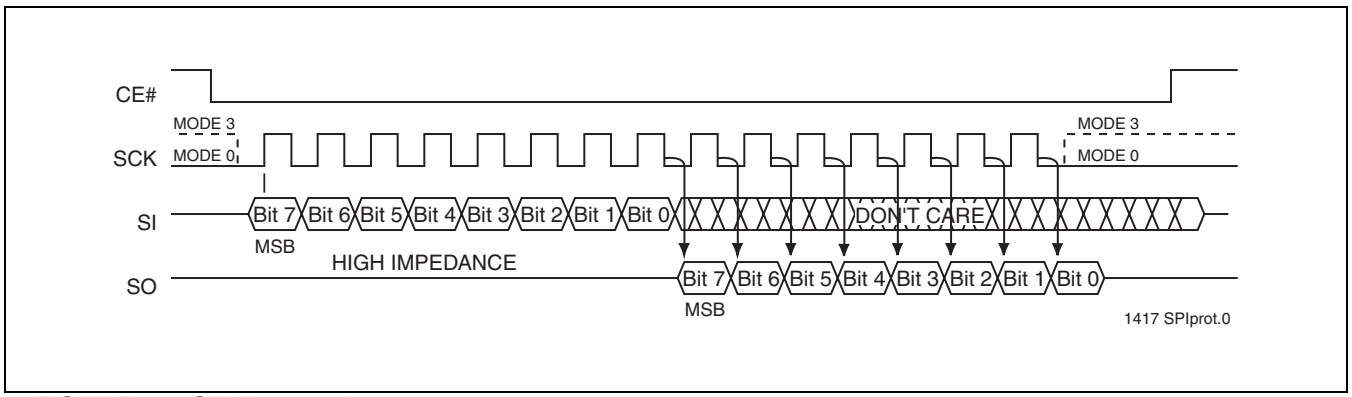

FIGURE 3: SPI Protocol

Hold Operation

The HOLD# pin is used to pause a serial sequence underway with the SPI flash memory without resetting the clocking sequence. To activate the HOLD# mode, CE# must be in active low state. The HOLD# mode begins when the SCK active low state coincides with the falling edge of the HOLD# signal. The HOLD mode ends when the HOLD# signal's rising edge coincides with the SCK active low state.

If the falling edge of the HOLD# signal does not coincide with the SCK active low state, then the device enters Hold mode when the SCK next reaches the active low state. Similarly, if the rising edge of the HOLD# signal does not coincide with the SCK active low state, then the device exits in Hold mode when the SCK next reaches the active low state. See Figure 4 for Hold Condition waveform.

Once the device enters Hold mode, SO will be in high-impedance state while SI and SCK can be V_{IL} or V_{IH}.

If CE# is driven high during a Hold condition, the device returns to Standby mode. As long as HOLD# signal is low, the memory remains in the Hold condition. To resume communication with the device, HOLD# must be driven active high, and CE# must be driven active low. See Figure 4 for Hold timing.

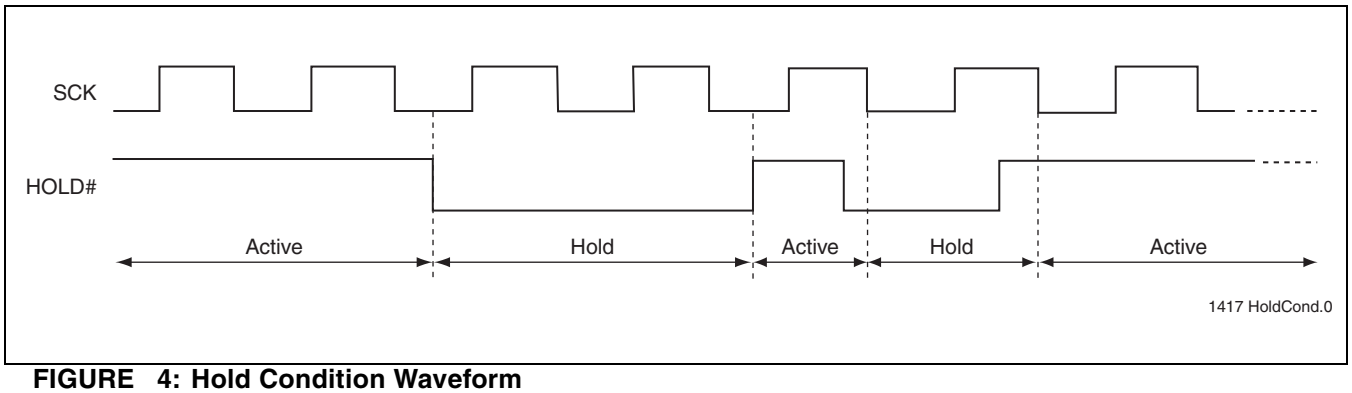

FIGURE 4: Hold Condition Waveform

Write Protection

SST25VF020B provides software Write protection. The Write Protect pin (WP#) enables or disables the lock-down function of the status register. The Block-Protection bits (BP1, BP0, and BPL) in the status register, and the Top/Bottom Sector Protection Status bits (TSP and BSP) in Status Register 1, provide Write protection to the memory array and the status register. See Table 5 for the Block-Protection description.

Write Protect Pin (WP#)

The Write Protect (WP#) pin enables the lock-down function of the BPL bit (bit 7) in the status register. When WP# is driven low, the execution of the Write-Status-Register (WRSR) instruction is determined by the value of the BPL bit (see Table 2). When WP# is high, the lock-down function of the BPL bit is disabled.

TABLE 2: Conditions to execute Write-Status-Register (WRSR) Instruction

WP#	BPL	Execute WRSR Instruction
L	1	Not Allowed
L	0	Allowed
H	X	Allowed

T2.0 1417

Status Register

The software status register provides status on whether the flash memory array is available for any Read or Write operation, whether the device is Write enabled, and the state of the Memory Write protection. During an internal Erase or Program operation, the status register may be read only to determine the completion of an operation in progress. Table 3 describes the function of each bit in the software status register.

TABLE 3: Software Status Register

Bit	Name	Function	Default at Power-up	Read/Write
0	BUSY	1 = Internal Write operation is in progress 0 = No internal Write operation is in progress	0	R
1	WEL	1 = Device is memory Write enabled 0 = Device is not memory Write enabled	0	R
2	BP0	Indicates current level of block write protection (See Table 5)	1	R/W
3	BP1	Indicates current level of block write protection (See Table 5)	1	R/W
4:5	RES	Reserved for future use	0	N/A
6	AAI	Auto Address Increment Programming status 1 = AAI programming mode 0 = Byte-Program mode	0	R
7	BPL	1 = BP1, BP0 are read-only bits 0 = BP1, BP0 are read/writable	0	R/W

T3.0 1417

Software Status Register 1

The Software Status Register 1 is an additional register that contains Top Sector and Bottom Sector Protection bits. These register bits are read/writable and determine the lock and unlock status of the top and bottom sectors. Table 4 describes the function of each bit in the Software Status Register 1.

TABLE 4: Software Status Register 1

Bit	Name	Function	Default at Power-up	Read/Write
0:1	RES	Reserved for future use	0	N/A
2	TSP	Top Sector Protection status 1 = Indicates highest sector is write locked 0 = Indicates highest sector is Write accessible	0	R/W
3	BSP	Bottom Sector Protection status 1 = Indicates lowest sector is write locked 0 = Indicates lowest sector is Write accessible	0	R/W
4:7	RES	Reserved for future use	0	N/A

T4.0 1417

Busy

The Busy bit determines whether there is an internal Erase or Program operation in progress. A “1” for the Busy bit indicates the device is busy with an operation in progress. A “0” indicates the device is ready for the next valid operation.

Write Enable Latch (WEL)

The Write-Enable-Latch bit indicates the status of the internal memory Write Enable Latch. If the Write-Enable-Latch bit is set to “1”, it indicates the device is Write enabled. If the bit is set to “0” (reset), it indicates the device is not Write enabled and does not accept any memory Write (Program/Erase) commands. The Write-Enable-Latch bit is automatically reset under the following conditions:

- Power-up
- Write-Disable (WRDI) instruction completion
- Byte-Program instruction completion
- Auto Address Increment (AAI) programming is completed or reached its highest unprotected memory address
- Sector-Erase instruction completion
- Block-Erase instruction completion
- Chip-Erase instruction completion
- Write-Status-Register instructions

Auto Address Increment (AAI)

The Auto Address Increment Programming-Status bit provides status on whether the device is in AAI programming mode or Byte-Program mode. The default at power up is Byte-Program mode.

Block Protection (BP1, BP0)

The Block-Protection (BP1, BP0) bits define the size of the memory area, as defined in Table 5, to be software protected against any memory Write (Program or Erase) operation. The Write-Status-Register (WRSR) instruction is used to program the BP1 and BP0 bits as long as WP# is high or the Block-Protect-Lock (BPL) bit is 0. Chip-Erase can only be executed if Block-Protection bits are all 0. After power-up, BP1 and BP0 are set to 1.

Block Protection Lock-Down (BPL)

WP# pin driven low (V_{IL}), enables the Block-Protection-Lock-Down (BPL) bit. When BPL is set to 1, it prevents any further alteration of the BPL, BP1, and BP0 bits of the status register and BSP and TSP of Status Register 1. When the WP# pin is driven high (VIH), the BPL bit has no effect and its value is “Don’t Care”. After power-up, the BPL bit is reset to 0.

TABLE 5: Software Status Register Block Protection FOR SST25VF020B[1]

	Status Register Bit[2]		Protected Memory Address
Protection Level	**BP1**	**BP0**	**2 Mbit**
0	0	0	None
1 (1/4 Memory Array)	0	1	030000H-03FFFFH
1 (1/2 Memory Array)	1	0	020000H-03FFFFH
1 (Full Memory Array)	1	1	000000H-03FFFFH

T5.0 1417

1. X = Don’t Care (RESERVED) default is “0
2. Default at power-up for BP1 and BP0 is ‘11’. (All Blocks Protected)

Top-Sector Protection/Bottom-Sector Protection

The Top-Sector Protection (TSP) and Bottom-Sector Protection (BSP) bits independently indicate whether the highest and lowest sector locations are Write locked or Write accessible. When TSP or BSP is set to ‘1’, the respective sector is Write locked; when set to ‘0’ the respective sector is Write accessible. If TSP or BSP is set to '1' and if the top or bottom sector is within the boundary of the target address range of the program or erase instruction, the initiated instruction (Byte-Program, AAI-Word Program, Sector-Erase, Block-Erase, and Chip-Erase) will not be executed. Upon power-up, the TSP and BSP bits are automatically reset to ‘0’.

Instructions

Instructions are used to read, write (Erase and Program), and configure the SST25VF020B. The instruction bus cycles are 8 bits each for commands (Op Code), data, and addresses. Prior to executing any Byte-Program, Auto Address Increment (AAI) programming, Sector-Erase, Block-Erase, Write-Status-Register, or Chip-Erase instructions, the Write-Enable (WREN) instruction must be executed first. The complete list of instructions is provided in Table 6. All instructions are synchronized off a high to low transition of CE#. Inputs will be accepted on the rising edge of SCK starting with the most significant bit. CE# must be driven low before an instruction is entered and must be driven high after the last bit of the instruction has been shifted in (except for Read, Read-ID, and Read-Status-Register instructions). Any low to high transition on CE#, before receiving the last bit of an instruction bus cycle, will terminate the instruction in progress and return the device to standby mode. Instruction commands (Op Code), addresses, and data are all input from the most significant bit (MSB) first.

TABLE 6: Device Operation Instructions

Instruction	Description	Op Code Cycle[1]	Address Cycle(s)[2]	Dummy Cycle(s)	Data Cycle(s)
Read	Read Memory	0000 0011b (03H)	3	0	1 to ∞
High-Speed Read	Read Memory at higher speed	0000 1011b (0BH)	3	1	1 to ∞
4 KByte Sector-Erase[3]	Erase 4 KByte of memory array	0010 0000b (20H)	3	0	0
32 KByte Block-Erase[4]	Erase 32 KByte block of memory array	0101 0010b (52H)	3	0	0
64 KByte Block-Erase[5]	Erase 64 KByte block of memory array	1101 1000b (D8H)	3	0	0
Chip-Erase	Erase Full Memory Array	0110 0000b (60H) or 1100 0111b (C7H)	0	0	0
Byte-Program	To Program One Data Byte	0000 0010b (02H)	3	0	1
AAI-Word-Program[6]	Auto Address Increment Programming	1010 1101b (ADH)	3	0	2 to ∞
RDSR[7]	Read-Status-Register	0000 0101b (05H)	0	0	1 to ∞
RDSR1	Read-Status-Register 1	0011 0101b (35H)	0	0	1 to ∞
EWSR	Enable-Write-Status-Register	0101b 0000b (50H)	0	0	0
WRSR	Write-Status-Register	0000 0001b (01H)	0	0	1 or 2
WREN	Write-Enable	0000 0110b (06H)	0	0	0
WRDI	Write-Disable	0000 0100b (04H)	0	0	0
RDID[8]	Read-ID	1001 0000b (90H) or 1010 1011b (ABH)	3	0	1 to ∞
JEDEC-ID	JEDEC ID read	1001 1111b (9FH)	0	0	3 to ∞
EBSY	Enable SO to output RY/BY# status during AAI programming	0111 0000b (70H)	0	0	0
DBSY	Disable SO to output RY/BY# status during AAI programming	1000 0000b (80H)	0	0	0

T6.0 1417

1. One bus cycle is eight clock periods.
2. Address bits above the most significant bit of each density can be V_{IL} or V_{IH}.
3. 4KByte Sector Erase addresses: use A_{MS}-A_{12}, remaining addresses are don't care but must be set either at V_{IL} or V_{IH}.
4. 32KByte Block Erase addresses: use A_{MS}-A_{15}, remaining addresses are don't care but must be set either at V_{IL} or V_{IH}.
5. 64KByte Block Erase addresses: use A_{MS}-A_{16}, remaining addresses are don't care but must be set either at V_{IL} or V_{IH}.
6. To continue programming to the next sequential address location, enter the 8-bit command, ADH, followed by 2 bytes of data to be programmed. Data Byte 0 will be programmed into the initial address [A_{23}-A_1] with A_0=0, Data Byte 1 will be programmed into the initial address [A_{23}-A_1] with A_0=1.
7. The Read-Status-Register is continuous with ongoing clock cycles until terminated by a low to high transition on CE#.
8. Manufacturer's ID is read with A_0=0, and Device ID is read with A_0=1. All other address bits are 00H. The Manufacturer's ID and device ID output stream is continuous until terminated by a low-to-high transition on CE#.

Read (33 MHz)

The Read instruction, 03H, supports up to 33 MHz Read. The device outputs the data starting from the specified address location. The data output stream is continuous through all addresses until terminated by a low to high transition on CE#. The internal address pointer will automatically increment until the highest memory address is reached. Once the highest memory address is reached, the address pointer will automatically increment to the beginning (wrap-around) of the address space. Once the data from address location 3FFFFH has been read, the next output will be from address location 000000H.

The Read instruction is initiated by executing an 8-bit command, 03H, followed by address bits [A_{23}-A_0]. CE# must remain active low for the duration of the Read cycle. See Figure 5 for the Read sequence.

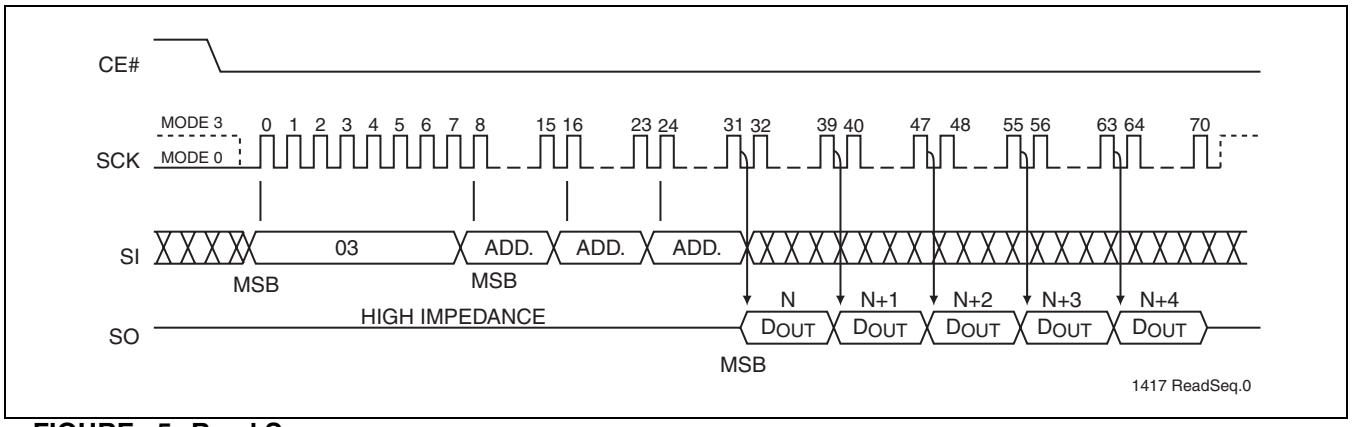

FIGURE 5: Read Sequence

High-Speed-Read (80 MHz)

The High-Speed-Read instruction supporting up to 80 MHz Read is initiated by executing an 8-bit command, 0BH, followed by address bits [A_{23}-A_0] and a dummy byte. CE# must remain active low for the duration of the High-Speed-Read cycle. See Figure 6 for the High-Speed-Read sequence.

Following a dummy cycle, the High-Speed-Read instruction outputs the data starting from the specified address location. The data output stream is continuous through all addresses until terminated by a low to high transition on CE#. The internal address pointer will automatically increment until the highest memory address is reached. Once the highest memory address is reached, the address pointer will automatically increment to the beginning (wrap-around) of the address space. Once the data from address location 3FFFH has been read, the next output will be from address location 00000H.

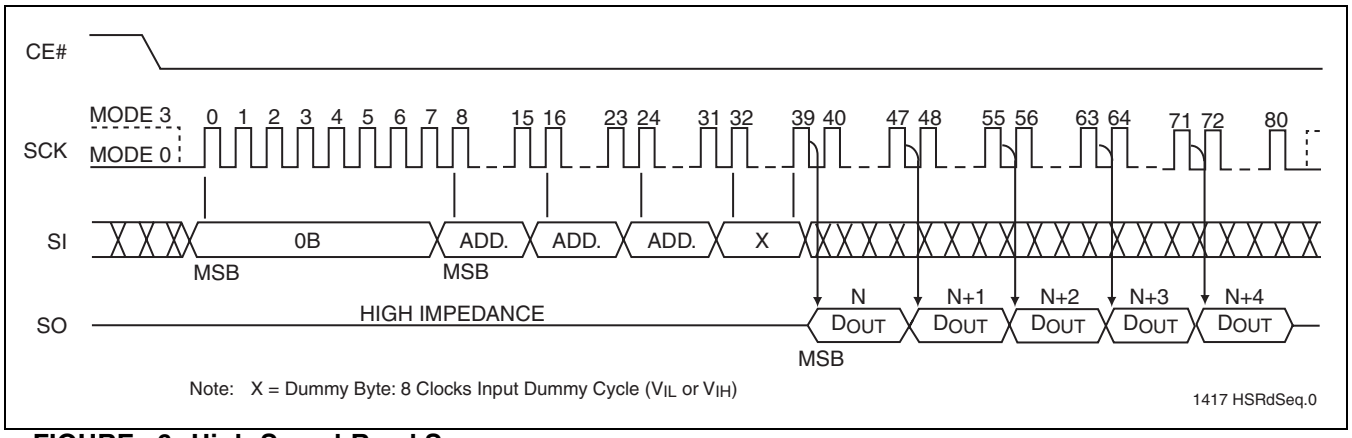

FIGURE 6: High-Speed-Read Sequence

Byte-Program

The Byte-Program instruction programs the bits in the selected byte to the desired data. The selected byte must be in the erased state (FFH) when initiating a Program operation. A Byte-Program instruction applied to a protected memory area will be ignored.

Prior to any Write operation, the Write-Enable (WREN) instruction must be executed. CE# must remain active low for the duration of the Byte-Program instruction. The Byte-Program instruction is initiated by executing an 8-bit command, 02H, followed by address bits [A_{23}-A_0]. Following the address, the data is input in order from MSB (bit 7) to LSB (bit 0). CE# must be driven high before the instruction is executed. The user may poll the Busy bit in the software status register or wait T_{BP} for the completion of the internal self-timed Byte-Program operation. See Figure 7 for the Byte-Program sequence.

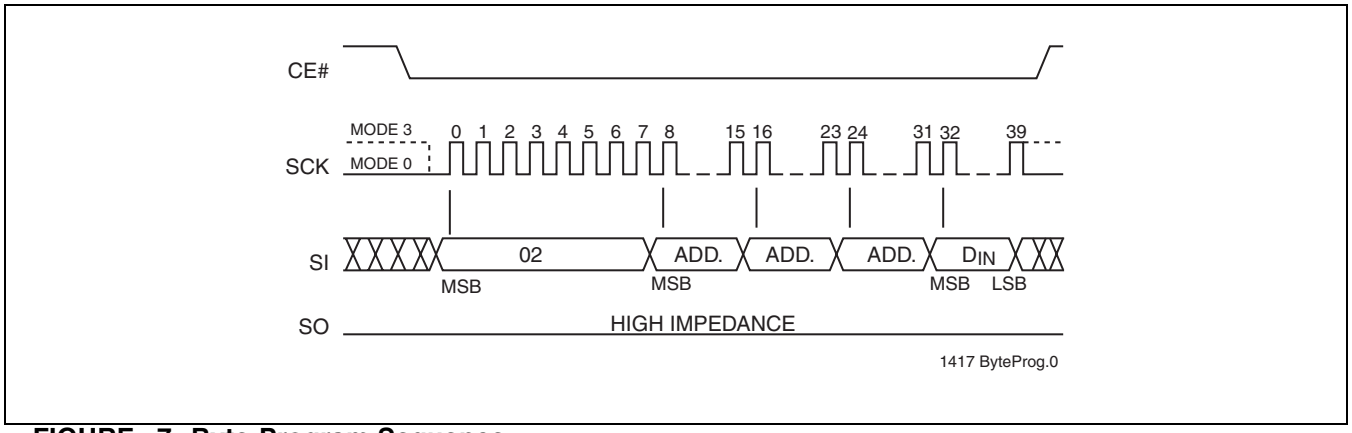

FIGURE 7: Byte-Program Sequence

Auto Address Increment (AAI) Word-Program

The AAI program instruction allows multiple bytes of data to be programmed without re-issuing the next sequential address location. This feature decreases total programming time when multiple bytes or entire memory array is to be programmed. An AAI Word program instruction pointing to a protected memory area will be ignored. The selected address range must be in the erased state (FFH) when initiating an AAI Word Program operation. While within AAI Word Programming sequence, the only valid instructions are AAI Word (ADH), RDSR (05H), or WRDI (04H). Users have three options to determine the completion of each AAI Word program cycle: hardware detection by reading the Serial Output, software detection by polling the BUSY bit in the software status register or wait T_{BP}. Refer to End-Of-Write Detection section for details.

Prior to any write operation, the Write-Enable (WREN) instruction must be executed. The AAI Word Program instruction is initiated by executing an 8-bit command, ADH, followed by address bits [A_{23}-A_0]. Following the addresses, two bytes of data is input sequentially, each one from MSB (Bit 7) to LSB (Bit 0). The first byte of data (D0) will be programmed into the initial address [A_{23}-A_1] with A_0=0, the second byte of Data (D1) will be programmed into the initial address [A_{23}-A_1] with A_0=1. CE# must be driven high before the AAI Word Program instruction is executed. The user must check the BUSY status before entering the next valid command. Once the device indicates it is no longer busy, data for the next two sequential addresses may be programmed and so on. When the last desired byte had been entered, check the busy status using the hardware method or the RDSR instruction and execute the Write-Disable (WRDI) instruction, 04H, to terminate AAI. User must check busy status after WRDI to determine if the device is ready for any command. See Figures 10 and 11 for AAI Word programming sequence.

There is no wrap mode during AAI programming; once the highest unprotected memory address is reached, the device will exit AAI operation and reset the Write-Enable-Latch bit (WEL = 0) and the AAI bit (AAI=0).

End-of-Write Detection

There are three methods to determine completion of a program cycle during AAI Word programming: hardware detection by reading the Serial Output, software detection by polling the BUSY bit in the Software Status Register or wait T_{BP}. The hardware end-of-write detection method is described in the section below.

Hardware End-of-Write Detection

The hardware end-of-write detection method eliminates the overhead of polling the Busy bit in the Software Status Register during an AAI Word program operation. The 8-bit command, 70H, configures the Serial Output (SO) pin to indicate Flash Busy status during AAI Word programming. (see Figure 8) The 8-bit command, 70H, must be executed prior to executing an AAI Word-Program instruction. Once an internal programming operation begins, asserting CE# will immediately drive the status of the internal flash status on the SO pin. A “0” indicates the device is busy and a “1” indicates the device is ready for the next instruction. De-asserting CE# will return the SO pin to tri-state.

The 8-bit command, 80H, disables the Serial Output (SO) pin to output busy status during AAI-Word-program operation and return SO pin to output Software Status Register data during AAI Word programming. (see Figure 9)

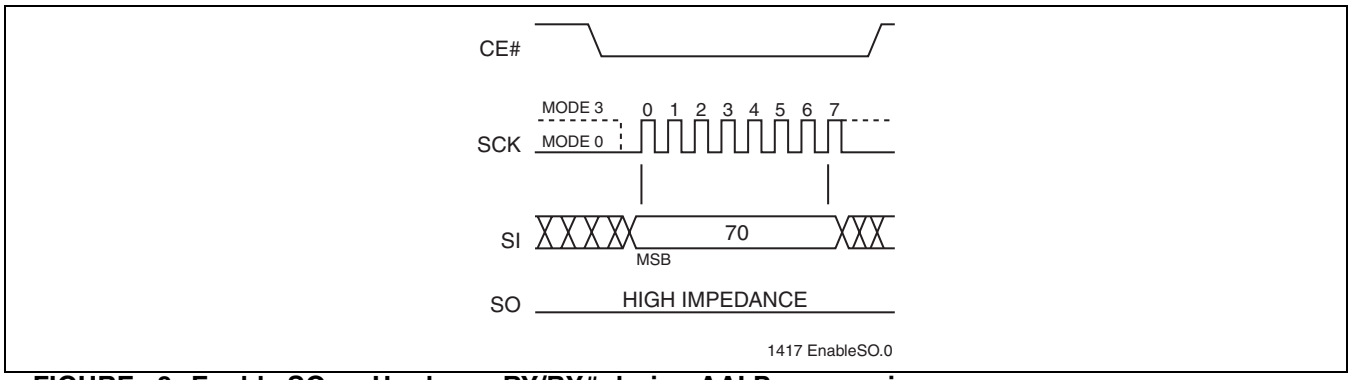

FIGURE 8: Enable SO as Hardware RY/BY# during AAI Programming

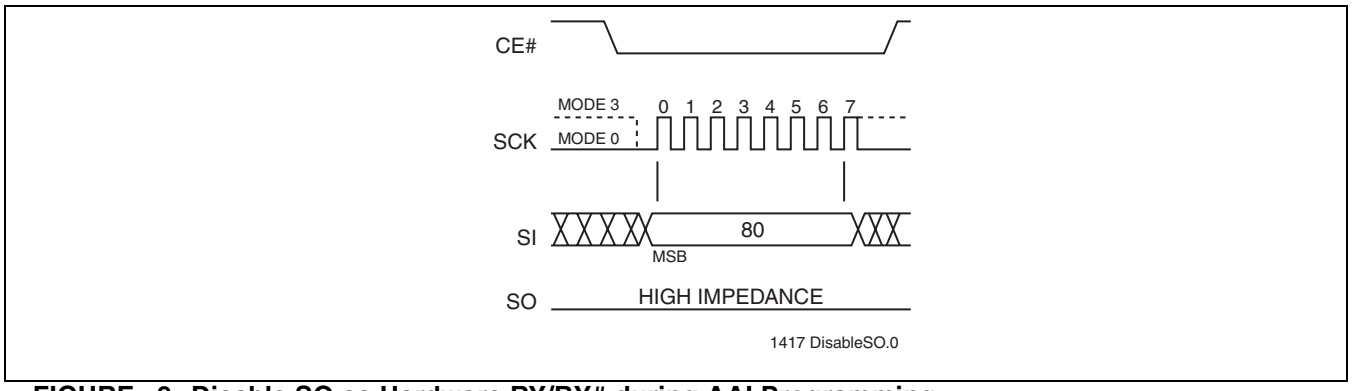

FIGURE 9: Disable SO as Hardware RY/BY# during AAI Programming

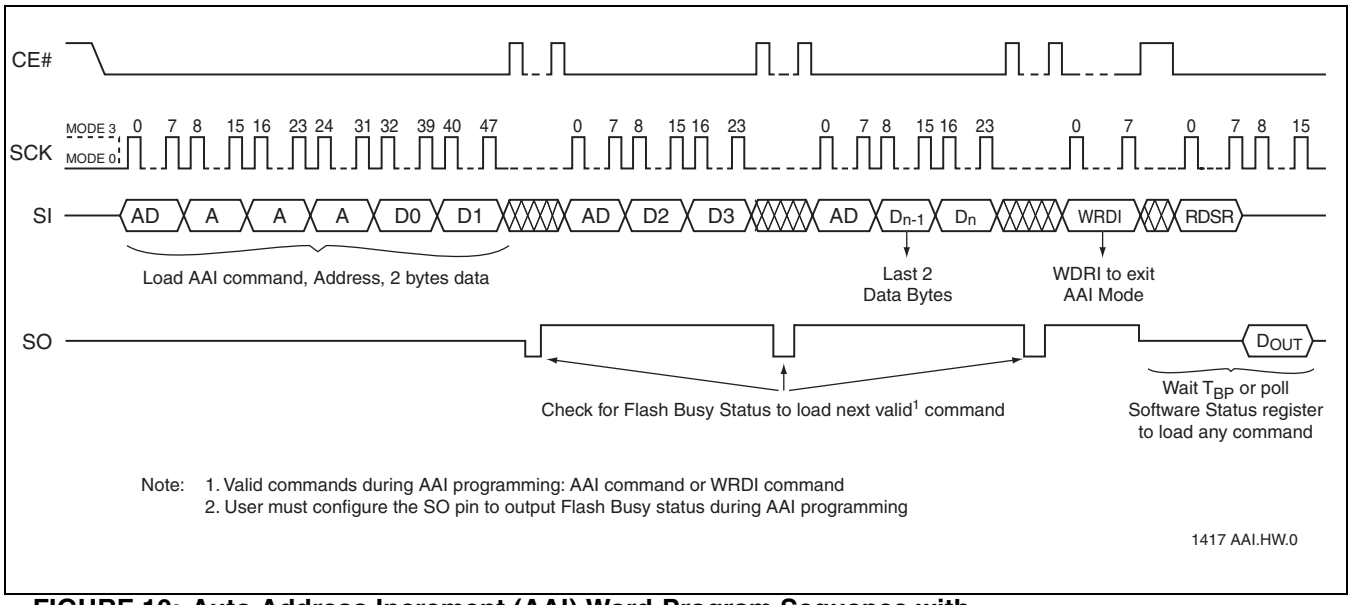

FIGURE 10: Auto Address Increment (AAI) Word-Program Sequence with Hardware End-of-Write Detection

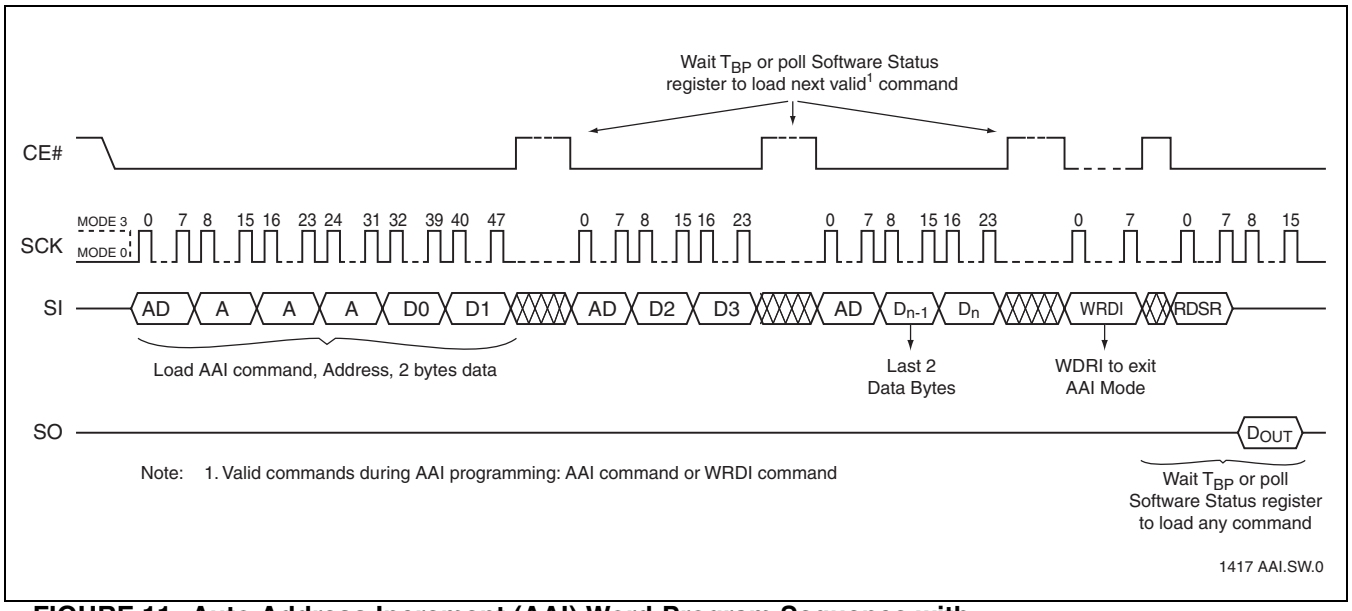

FIGURE 11: Auto Address Increment (AAI) Word-Program Sequence with Software End-of-Write Detection

4-KByte Sector-Erase

The Sector-Erase instruction clears all bits in the selected 4 KByte sector to FFH. A Sector-Erase instruction applied to a protected memory area will be ignored. Prior to any Write operation, the Write-Enable (WREN) instruction must be executed. CE# must remain active low for the duration of any command sequence. The Sector-Erase instruction is initiated by executing an 8-bit command, 20H, followed by address bits [A_{23}-A_0]. Address bits [A_{MS}-A_{12}] (A_{MS} = Most Significant address) are used to determine the sector address (SA_X), remaining address bits can be V_{IL} or V_{IH}. CE# must be driven high before the instruction is executed. The user may poll the Busy bit in the software status register or wait T_{SE} for the completion of the internal self-timed Sector-Erase cycle. See Figure 12 for the Sector-Erase sequence.

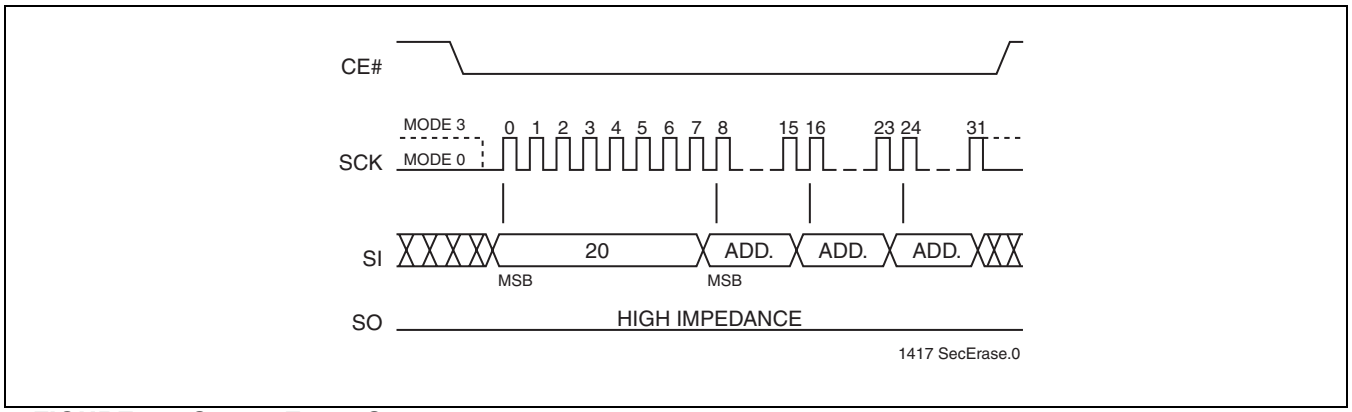

FIGURE 12: Sector-Erase Sequence

32-KByte and 64-KByte Block-Erase

The 32-KByte Block-Erase instruction clears all bits in the selected 32 KByte block to FFH. A Block-Erase instruction applied to a protected memory area will be ignored. The 64-KByte Block-Erase instruction clears all bits in the selected 64 KByte block to FFH. A Block-Erase instruction applied to a protected memory area will be ignored. Prior to any Write operation, the Write-Enable (WREN) instruction must be executed. CE# must remain active low for the duration of any command sequence. The 32-Kbyte Block-Erase instruction is initiated by executing an 8-bit command, 52H, followed by address bits [A_{23}-A_0]. Address bits [A_{MS}-A_{15}] (A_{MS} = Most Significant Address) are used to determine block address (BA_X), remaining address bits can be V_{IL} or V_{IH}. CE# must be driven high before the instruction is executed. The 64-Kbyte Block-Erase instruction is initiated by executing an 8-bit command D8H, followed by address bits [A_{23}-A_0]. Address bits [A_{MS}-A_{15}] are used to determine block address (BA_X), remaining address bits can be V_{IL} or V_{IH}. CE# must be driven high before the instruction is executed. The user may poll the Busy bit in the software status register or wait T_{BE} for the completion of the internal self-timed 32-KByte Block-Erase or 64-KByte Block-Erase cycles. See Figures 13 and 14 for the 32-KByte Block-Erase and 64-KByte Block-Erase sequences.

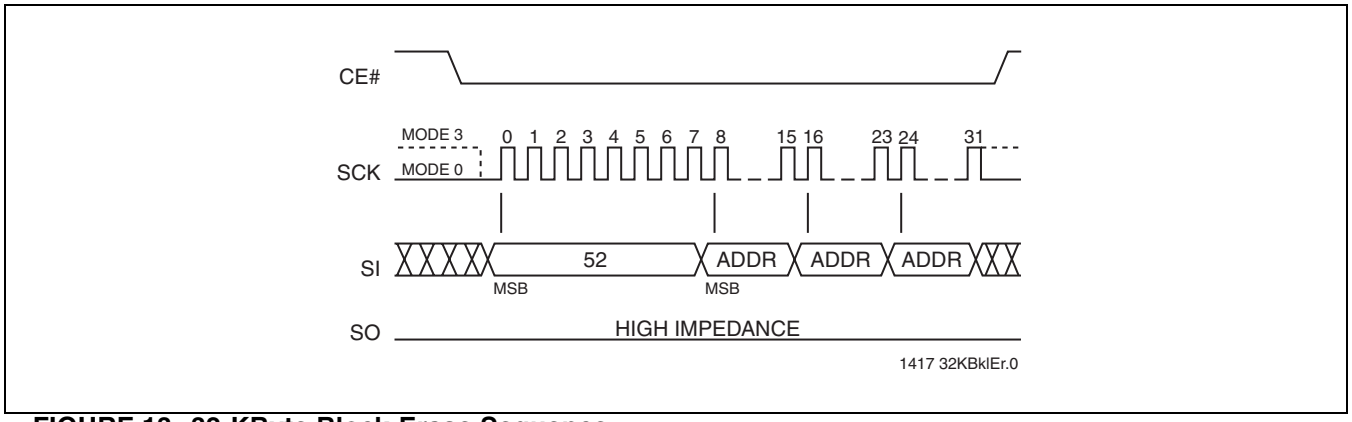

FIGURE 13: 32-KByte Block-Erase Sequence

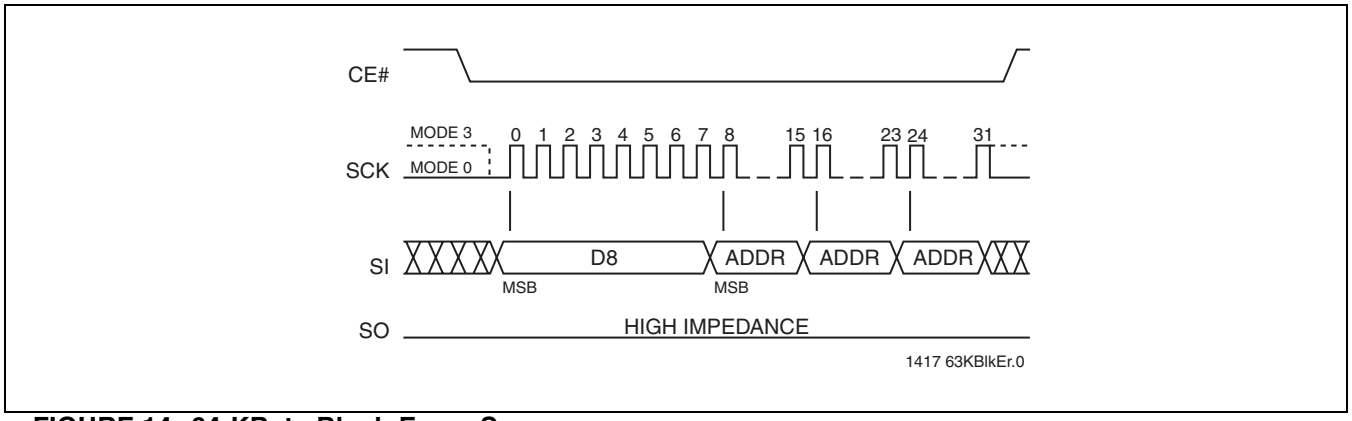

FIGURE 14: 64-KByte Block-Erase Sequence

Chip-Erase

The Chip-Erase instruction clears all bits in the device to FFH. A Chip-Erase instruction will be ignored if any of the memory area is protected. Prior to any Write operation, the Write-Enable (WREN) instruction must be executed. CE# must remain active low for the duration of the Chip-Erase instruction sequence. The Chip-Erase instruction is initiated by executing an 8-bit command, 60H or C7H. CE# must be driven high before the instruction is executed. The user may poll the Busy bit in the software status register or wait T_{CE} for the completion of the internal self-timed Chip-Erase cycle. See Figure 15 for the Chip-Erase sequence.

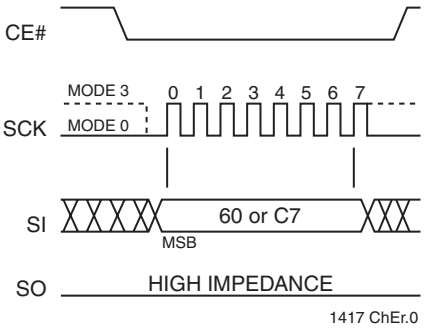

FIGURE 15: Chip-Erase Sequence

Read-Status-Register (RDSR)

The Read-Status-Register (RDSR) instruction allows reading of the status register. The Status Register may be read at any time even during a Write (Program/Erase) operation. When a Write operation is in progress, the Busy bit may be checked before sending any new commands to assure that the new commands are properly received by the device. CE# must be driven low before the RDSR instruction is entered and remain low until the status data is read. Read-Status-Register is continuous with ongoing clock cycles until it is terminated by a low to high transition of the CE#. See Figure 16 for the RDSR instruction sequence.

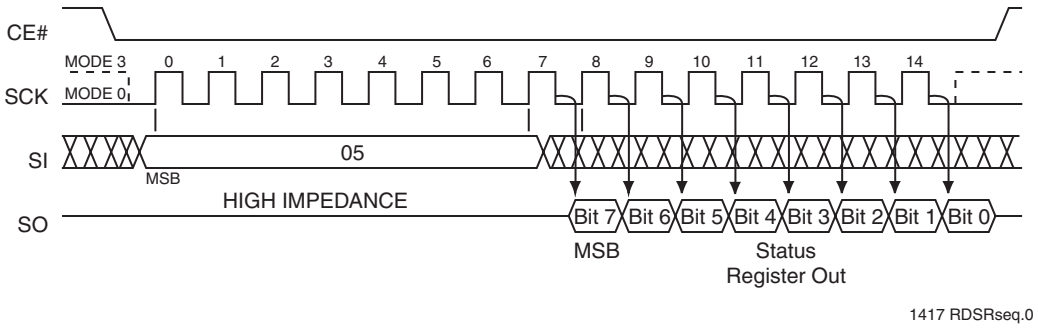

FIGURE 16: Read-Status-Register (RDSR) Sequence

Read-Status-Register (RDSR1)

The Read-Status-Register 1 (RDSR1) instruction allows reading of the status register 1. CE# must be driven low before the RDSR instruction is entered and remain low until the status data is read. Read-Status-Register 1 is continuous with ongoing clock cycles until it is terminated by a low to high transition of the CE#. See Figure 17 for the RDSR instruction sequence.