# S29WS-S MirrorBit® Eclipse™ Flash Family

## S29WS01GS

**1024 Megabit (128 Megabyte)**
**16-bit Data Width, Burst Access, Simultaneous Read/Write,**
**1.8 Volt-only Flash Memory in 65 nm MirrorBit® Technology**

***Data Sheet*** *(Preliminary)*

**Notice to Readers:** This document states the current technical specifications regarding the Spansion product(s) described herein. Each product described herein may be designated as Advance Information, Preliminary, or Full Production. See *Notice On Data Sheet Designations* for definitions.

**Publication Number** S29WS-S_00 **Revision** 02 **Issue Date** April 16, 2009

# Notice On Data Sheet Designations

Spansion Inc. issues data sheets with Advance Information or Preliminary designations to advise readers of product information or intended specifications throughout the product life cycle, including development, qualification, initial production, and full production. In all cases, however, readers are encouraged to verify that they have the latest information before finalizing their design. The following descriptions of Spansion data sheet designations are presented here to highlight their presence and definitions.

## Advance Information

The Advance Information designation indicates that Spansion Inc. is developing one or more specific products, but has not committed any design to production. Information presented in a document with this designation is likely to change, and in some cases, development on the product may discontinue. Spansion Inc. therefore places the following conditions upon Advance Information content:

> "This document contains information on one or more products under development at Spansion Inc. The information is intended to help you evaluate this product. Do not design in this product without contacting the factory. Spansion Inc. reserves the right to change or discontinue work on this proposed product without notice."

## Preliminary

The Preliminary designation indicates that the product development has progressed such that a commitment to production has taken place. This designation covers several aspects of the product life cycle, including product qualification, initial production, and the subsequent phases in the manufacturing process that occur before full production is achieved. Changes to the technical specifications presented in a Preliminary document should be expected while keeping these aspects of production under consideration. Spansion places the following conditions upon Preliminary content:

> "This document states the current technical specifications regarding the Spansion product(s) described herein. The Preliminary status of this document indicates that product qualification has been completed, and that initial production has begun. Due to the phases of the manufacturing process that require maintaining efficiency and quality, this document may be revised by subsequent versions or modifications due to changes in technical specifications."

## Combination

Some data sheets contain a combination of products with different designations (Advance Information, Preliminary, or Full Production). This type of document distinguishes these products and their designations wherever necessary, typically on the first page, the ordering information page, and pages with the DC Characteristics table and the AC Erase and Program table (in the table notes). The disclaimer on the first page refers the reader to the notice on this page.

## Full Production (No Designation on Document)

When a product has been in production for a period of time such that no changes or only nominal changes are expected, the Preliminary designation is removed from the data sheet. Nominal changes may include those affecting the number of ordering part numbers available, such as the addition or deletion of a speed option, temperature range, package type, or $V_{IO}$ range. Changes may also include those needed to clarify a description or to correct a typographical error or incorrect specification. Spansion Inc. applies the following conditions to documents in this category:

> "This document states the current technical specifications regarding the Spansion product(s) described herein. Spansion Inc. deems the products to have been in sufficient production volume such that subsequent versions of this document are not expected to change. However, typographical or specification corrections, or modifications to the valid combinations offered may occur."

Questions regarding these document designations may be directed to your local sales office.

# S29WS-S MirrorBit® Eclipse™ Flash Family

## S29WS01GS

**1024 Megabit (128 Megabyte)**
**16-bit Data Width, Burst Access, Simultaneous Read/Write,**
**1.8 Volt-only Flash Memory in 65 nm MirrorBit® Technology**

***Data Sheet** (Preliminary)*

## Distinctive Characteristics

- **Big, Fast, NOR Flash**
  - 128 MBytes of XIP code and data storage
  - 115 ns max. random access to any location, up to 108 MHz burst read
  - 10 times faster program than a traditional NOR Flash
  - 7 times faster erase than traditional NOR
  - Traditional NOR Flash interface
- **Single supply 1.8 V read/program/erase (1.70 V – 1.95 V)**
- **16-bit (Word) data bus width**
- **Address and Data Interface:**
  - Address and Data in Parallel (ADP)—S29WS-S Family
- **Asynchronous read mode**
  - with 16 word intra-cache line read option - S29WS-S Family
- **Simultaneous Read/Write (SRW) operation**
  - Program or Erase in one bank while reading from any other bank
  - Memory array is divided into 16 equal size banks
- **Programmable burst read modes**
  - Linear 8, or 16 word (16 or 32 Byte) burst with wrap-around
  - Linear continuous burst
- **RDY output for data transfer flow control**
- **Sector Erase**
  - Four 32 Kbyte sectors at top of memory array (models 20, 30)
  - All other sectors (models) are Uniform sectors of 128 Kbyte
  - 0.7 MB/s erase rate
- **Page Programming up to 512-byte groups**
  - 2.0 MB/s program rate when programming full 512 Byte buffer
  - Bit-field programming for bit resolution programming
- **Suspend and Resume commands for Program and Erase operations**
- **Write operation status register bits indicate program and erase operation completion**
- **Program-Erase Endurance**
  - 100,000 cycles per sector (typical)
- **Data Protection**
  - Secure Silicon Region of 1KBytes
    - One Time Program (OTP) area of 512 bytes each for factory and customer
  - Hardware Sector Protection (via ACC pin)
    - Selected sectors protected when ACC input is at $V_{IL}$
  - Boot code controlled sector protection
    - A range of sectors may be protected to prevent program and erase until the next hardware reset or power is removed from the device
  - Dynamic sector protection
    - All sectors are unprotected at power on for simplified system production test & programming
    - A single command is used to protect all sectors from program or erase
    - A single sector at a time may be unprotected by a command to enable programming or erase.
  - 10-year data retention (typical)
- **Common Flash Interface (CFI) data structure**
- **Wireless Temperature range (-25°C to +85°C)**
- **Offered Packages**
  - Universal Footprint: 186-ball FBGA (11 mm x 13 mm)

**Publication Number** S29WS-S_00 **Revision** 02 **Issue Date** April 16, 2009

# Performance Characteristics

| Read Access Times (maximum values) | |
|---|---|
| **Speed Option (MHz)** | **108** |
| Synch. Internal Access, ns ($t_{IA}$) | 115 |
| Synch. Burst Access, ns ($t_{BACC}$) | 7.0 |
| Asynch. Access Time, ns ($t_{ACC}$) | 115 |
| Intra cache line read, ns ($t_{ICACC}$) | 20 |

| Current Consumption (typical values) | |
|---|---|
| Continuous Burst Read @ 108 MHz | 34 mA |
| Simultaneous Operation @ 108 MHz | 107 mA |
| Program / Erase | 65 mA |
| Standby / Sleep Mode | 30 µA |

| Program & Erase Rates (typical values) | |
|---|---|
| Page Programming ($V_{CC}$) | 2.0 MBytes/Sec |
| Sector Erase | 0.7 MBytes/Sec |

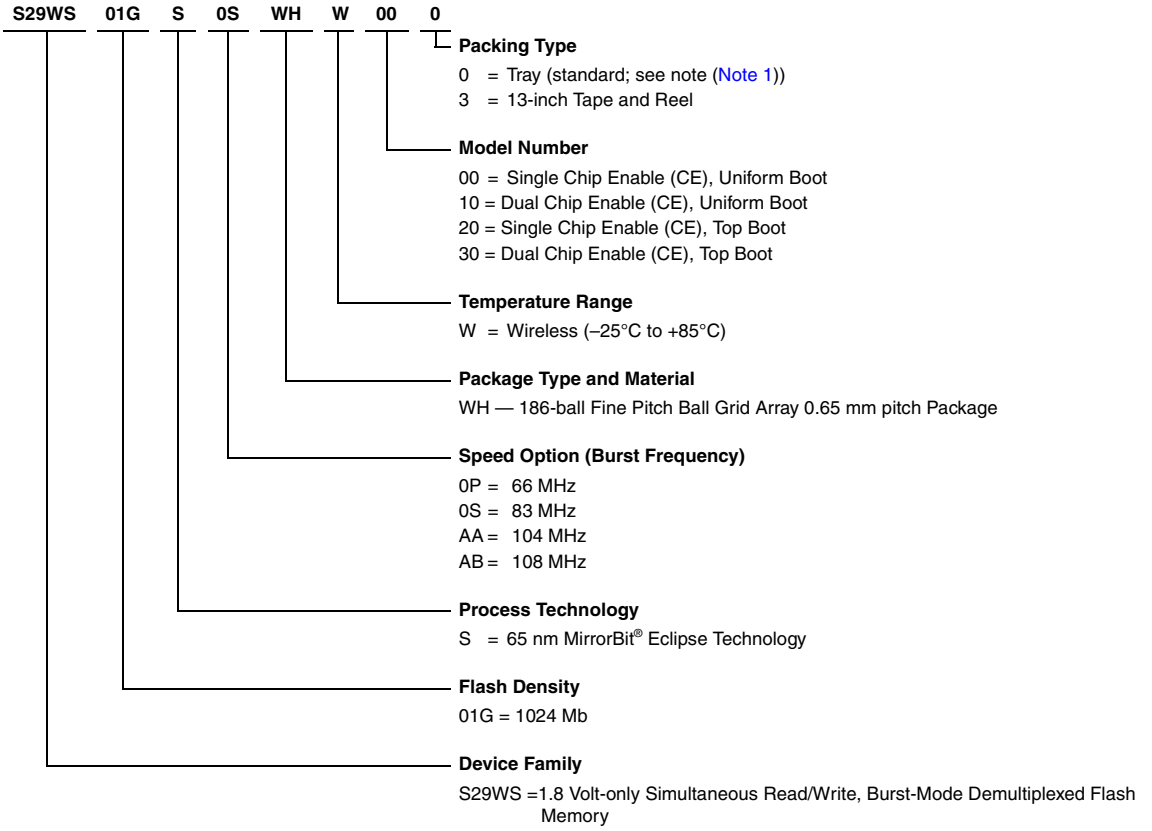
# Table of Contents

# Figures

# Tables

# 1. Ordering Information

The ordering part number is formed by a valid combination of the following:




## 1.1 Valid Combinations

Valid Combination list configurations are planned to be supported in volume for this device. Consult your local sales office to confirm availability of specific valid combinations and to check on newly released combinations.

| Valid Combinations | | | | | |
|---|---|---|---|---|---|
| Base Ordering Part Number | Speed Option | Package Type, Material, and Temperature Range | Packing Type | Model Numbers | Package Type (2) |
| S29WS01GS | 0P, 0S AA, AB | WHW (Lead (Pb) Free, Low Halogen) | 0, 3 (1) | 00, 10, 20, 30 | 11 mm x 13 mm, 186-ball |

***Notes:***

1. *Type 0 is standard. Specify other options as required.*
2. *BGA package marking omits leading* S29 *and packing type designator from ordering part number.*

# 2. Input/Output Descriptions

Table 2.1identifies the input and output package connections provided on the device.

**Table 2.1** Input/Output Descriptions

| Symbol | Type | Description |
|---|---|---|
| RESET# | Input | Hardware Reset. Low = device resets and returns to reading array data. |
| CLK | Input | Clock Input. In burst mode, after the initial word is output, subsequent active edges of CLK increment the internal address counter. Must not be floating but can be at $V_{IL}$ or $V_{IH}$ or continue cycling while in asynchronous mode; clock is ignored in asynchronous mode. |
| CE1# | Input | Chip Enable 1. Asynchronous relative to CLK. In single CE# model this signal is used to select the entire device address space. In dual CE# model this signal is used to select the first half of the Flash device address space that can be directly selected by one host chip enable signal. |
| CE2# | Input | Chip Enable 2. Asynchronous relative to CLK. In dual CE# model this signal is used to select a second half of the Flash device address space that can be directly selected by one host chip enable signal. Should be used when the host chip enable address space is less than the device address space and a second host chip enable is available to select the second half of the Flash device address space thus doubling the address space that may be directly accessed by the host system. |
| AVD# | Input | Address Valid. Indicates to device that the valid address is present on the address/data inputs. When low during asynchronous mode, indicates valid address; when low during burst mode, causes starting address to be latched at the next active clock edge. |
| $A_{max}$ – A16 | Input | High order address lines. $A_{Max}$ = A25 for 1 Gb. |
| A15 - A0 | Input | Low order address lines used only in S29WS-S family devices. |
| DQ15 – DQ0 | I/O | Data input/output. |
| OE# | Input | Output Enable. Asynchronous relative to CLK. |
| WE# | Input | Write Enable. |
| RDY | Output | Ready. Indicates when valid burst data is ready to be read. |
| $V_{CC}$ | Supply | Main power supply to the device. |
| ACC | Input | Acceleration Supply used by some Spansion memories to accelerate program and erase speed. Acceleration is not supported in this specific device. In this device it is used to disable program and erase in the entire memory array. At $V_{IL}$, the input disables all program and erase functions on all sectors. For more details see *Hardware Data Protection Methods* *on page 50* Should be at $V_{IH}$ for all other conditions can not be left floating or unconnected. |
| $V_{SS}$ | Supply | Ground. |
| NC | No Connect | Not connected internally. May be connected to $V_{SS}$ or leave floating |
| RFU | Reserved for Future Use | Signal connection which may be used in the future. It is recommended that printed circuit board signals not be routed through this signal connection in the package footprint. |
| DNU | Do Not Use | No signal should be connected to this ball in the package footprint. |

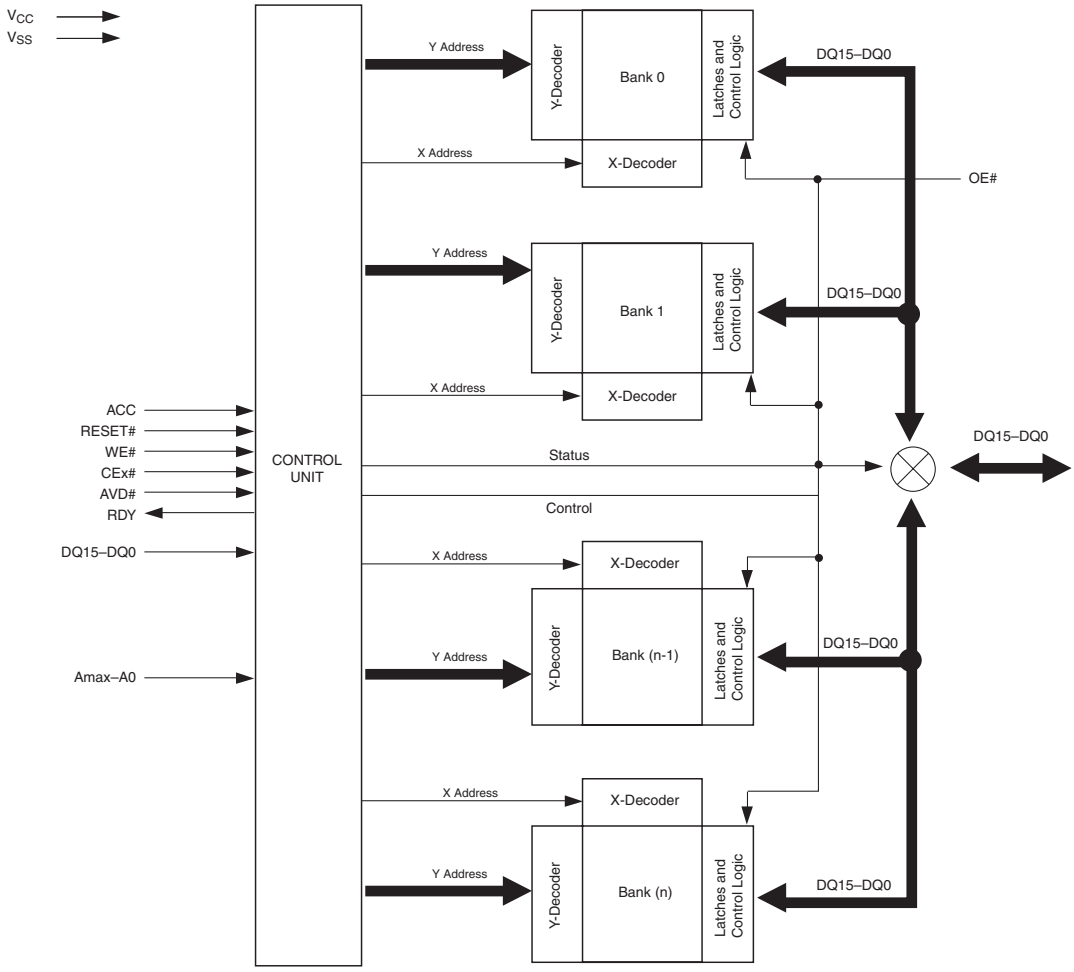
## 2.1 Dual CE Option

Devices with model number *00* and *20* use Amax in combination with CE1# to select any location in the device. For these models, CE2# is left unconnected internally, effectively it is an RFU signal.

Devices with model number *10*, and *30* use CE2# as a second chip enable that selects the upper half of the device address space. This allows host memory controllers with less than the full address range of the device per CE1# to use the device model with a second chip enable to reach all locations in the device. For this model, CE2# is connected to a second host CE# and Amax is left unconnected, effectively Amax is an RFU signal. If not used by the host system, CE2# must be at $V_{IH}$. CE1# and CE2# will be ignored if both are at $V_{IL}$ at the same time.

# 3. Block Diagram

**Figure 3.1** Simultaneous Operation Circuit

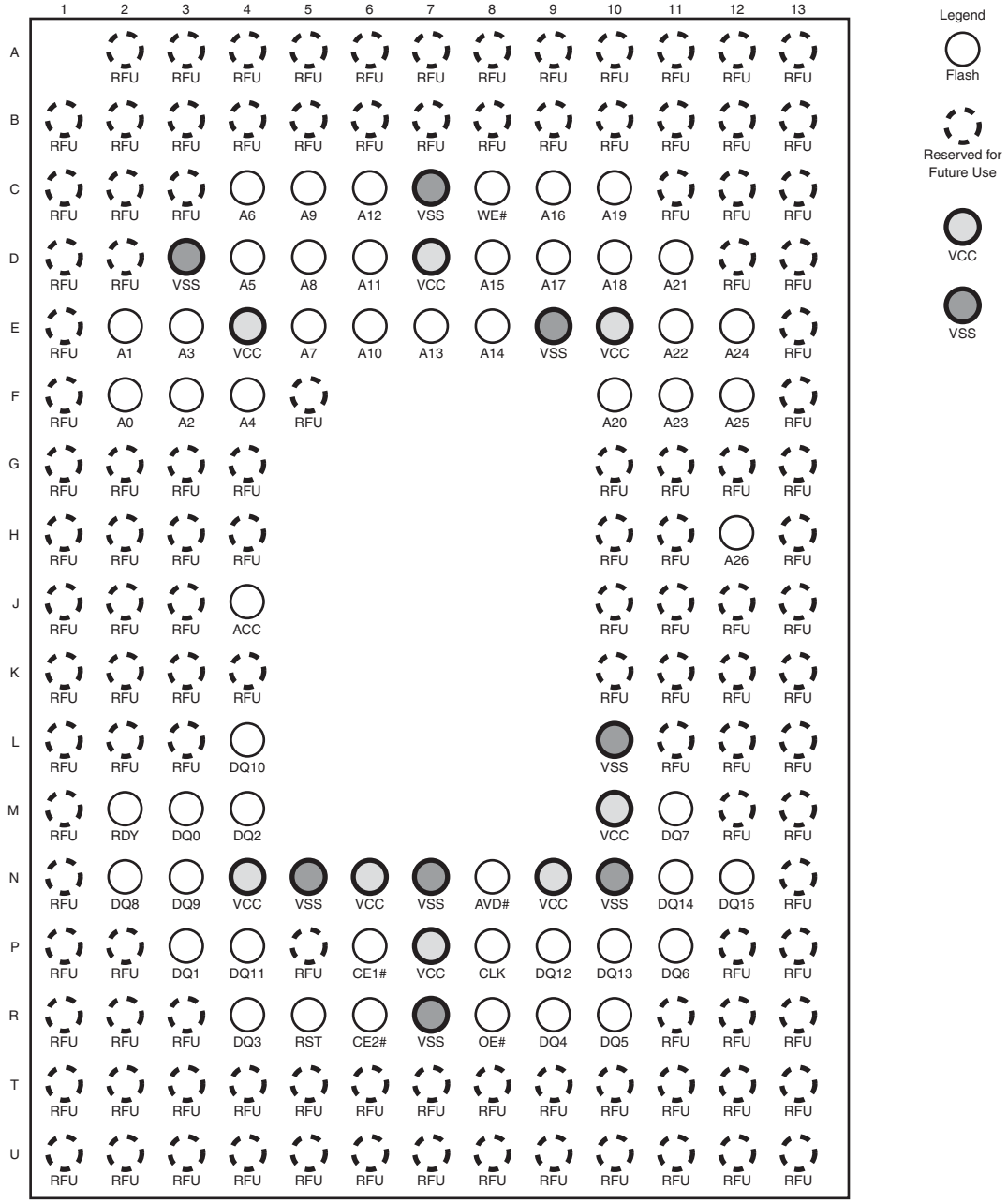



***Note:***
*1. Bank (n) = 15.*

# 4. Physical Dimensions/Connection Diagrams

This section shows the I/O designations and package specifications.

**Figure 4.1** Connection Diagram




***Notes***

1. *If selected device model number is 00 then Ball R6 CE2# is RFU and Amax = A25 for S29WS01GS.*
2. *If selected device model number is 10 then CE2# is valid and Amax = A24 for S29WS01GS.*

## 4.1 Related Documents

The following documents contain information related to this family of Flash devices. Click on the title or go to www.spansion.com to download the PDF file, or request a copy from your sales office.

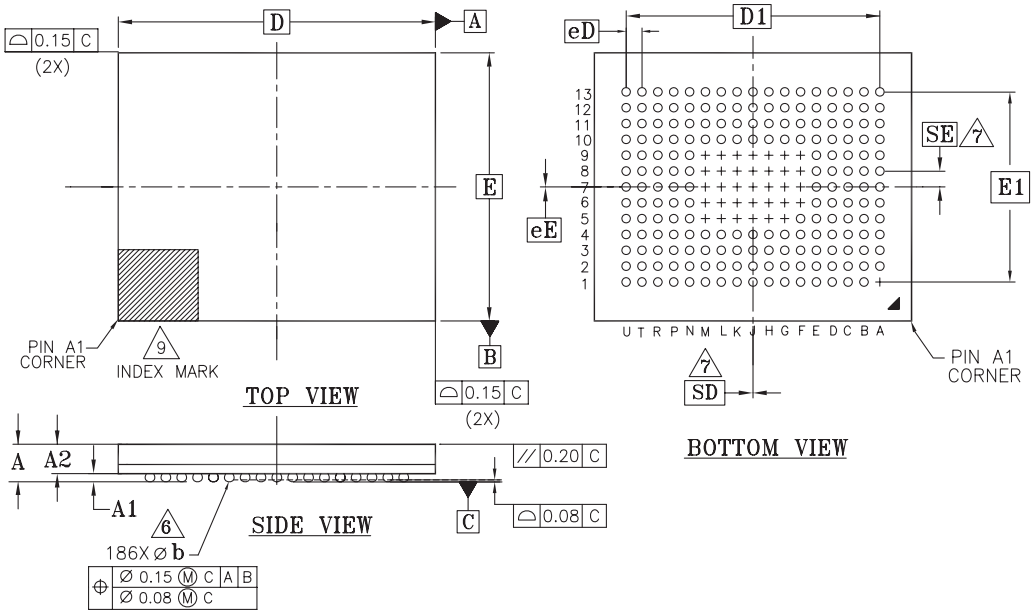
- Considerations for X-ray Inspection of Surface-Mounted Flash Integrated Circuits

## 4.2 Special Handling Instructions for FBGA Package

Special handling is required for Flash Memory products in FBGA packages.

Flash memory devices in FBGA packages may be damaged if exposed to ultrasonic cleaning methods. The package and/or data integrity may be compromised if the package body is exposed to temperatures above 150°C for prolonged periods of time.

### 4.2.0.1 AYY186—186-ball Fine Pitch Ball Grid Array 11 x 13 mm Package




| PACKAGE | Ayy 186 | | | |
|---|---|---|---|---|
| JEDEC | N/A | | | |
| D x E | 13.00 mm x 11.00 mm PACKAGE | | | |
| SYMBOL | MIN | NOM | MAX | NOTE |
| A | --- | --- | 1.20 | PROFILE |
| A1 | 0.20 | --- | --- | BALL HEIGHT |
| A2 | 0.79 | --- | 0.95 | BODY THICKNESS |
| D | | 13.00 BSC | | BODY SIZE |
| E | | 11.00 BSC | | BODY SIZE |
| D1 | | 10.40 BSC | | MATRIX FOOTPRINT |
| E1 | | 7.80 BSC | | MATRIX FOOTPRINT |
| MD | | 17 | | MATRIX SIZE D DIRECTION |
| ME | | 13 | | MATRIX SIZE E DIRECTION |
| n | | 186 | | BALL COUNT |
| Ø b | 0.325 | 0.375 | 0.425 | BALL DIAMETER |
| eE | | 0.65 BSC | | BALL PITCH |
| eD | | 0.65 BSC | | BALL PITCH |
| SD SE | | 0.00 BSC | | SOLDER BALL PLACEMENT |
| | A1,F6,F7,F8,F9,G5,G6,G7,G8,G9 H5,H6,H7,H8,H9,J5,J6,J7,J8,J9, K5,K6,K7,K8,K9,L5,L6,L7,L8,L9, M5,M6,M7,M8,M9 | | | DEPOPULATED SOLDER BALLS |

NOTES:

1. DIMENSIONING AND TOLERANCING METHODS PER ASME Y14.5M-1994.
2. ALL DIMENSIONS ARE IN MILLIMETERS.
3. BALL POSITION DESIGNATION PER JEP95, SECTION 4.3, SPP-010.
4. e REPRESENTS THE SOLDER BALL GRID PITCH.
5. SYMBOL "MD" IS THE BALL MATRIX SIZE IN THE "D" DIRECTION.

   SYMBOL "ME" IS THE BALL MATRIX SIZE IN THE "E" DIRECTION.

   n IS THE NUMBER OF POPULTED SOLDER BALL POSITIONS FOR MATRIX SIZE MD X ME.

6. DIMENSION "b" IS MEASURED AT THE MAXIMUM BALL DIAMETER IN A PLANE PARALLEL TO DATUM C.
7. SD AND SE ARE MEASURED WITH RESPECT TO DATUMS A AND B AND DEFINE THE POSITION OF THE CENTER SOLDER BALL IN THE OUTER ROW.

   WHEN THERE IS AN ODD NUMBER OF SOLDER BALLS IN THE OUTER ROW SD OR SE = 0.000.

   WHEN THERE IS AN EVEN NUMBER OF SOLDER BALLS IN THE OUTER ROW, SD OR SE = e/2

8. "+" INDICATES THE THEORETICAL CENTER OF DEPOPULATED BALLS.
9. A1 CORNER TO BE IDENTIFIED BY CHAMFER, LASER OR INK MARK, METALLIZED MARK INDENTATION OR OTHER MEANS.

3673 \ 16-038.ayy \ 1.11.8

# 5. Product Overview

The Eclipse family devices have a 16 bit (word) wide data bus. All read accesses provide 16 bits of data on each bus transfer cycle. All writes take 16 bits of data from each bus transfer cycle.

The Eclipse family combines the best features of eXecute In Place (XIP) and Data Storage Flash memories. The family has the fast random access of XIP Flash along with the high density and fast program speed of Data Storage Flash. Programming is 10 times faster and Erase is 7 times faster, than traditional NOR Flash.

Random access to any location takes only 115 ns max. After the first word is available the remaining words within a 16- word (32-Byte) aligned group (Cache Line) around the random access location can be delivered in a high-speed sequential burst. Linear read burst lengths of 8, or 16-words with wrap around and continuous burst read are supported.

A wrapped burst begins at the initial location and continues to the end of an 8, or 16 word aligned group then “wraps-around” to continue at the beginning of the 8, or 16-word aligned group. The burst completes with the last word before the initial location. Word wrap around burst is generally used for processor cache line fill.

Continuous burst delivers data from the initial word followed by sequential higher address words until the burst is terminated. At each 16 word (32 byte) aligned boundary additional wait states are inserted as the next group of 16-words is read. Wait states are indicated by the RDY signal being Low.

Asynchronous mode fast read of other locations within a Cache Line (intra cache line read) is also supported in S29WS-S family.

| Device | Mbits | Mbytes | Mwords | Banks | Mbytes / Bank |
|---|---|---|---|---|---|
| S29WS01GS | 1024 | 128 | 64 | 16 | 8 |

The Flash memory array is divided into banks as shown in the above table. A bank is the address range within which one program, or erase operation may be in progress at the same time as one read operation is in progress in any other bank of the memory. This multiple bank structure enables Simultaneous Read and Write (SRW) so that a group of data may be programmed or erased as a background task in one bank, while code may be executed, or data read, from any other bank.

Each bank is divided into sectors (also referred to as blocks). A sector is the minimum address range of data which can be erased to an all Ones state. A top boot option device has four 32 KByte sectors which are located at the top of the memory array on the device model purchased. These are called boot sectors because they are often used for holding boot code or parameters that need to be protected or erased separately from other data in the Flash array. All other sectors are a uniform size of 128 KBytes. There is also an ordering option (model) in which all sectors are of 128 KBytes uniform size.

Programming is done via a 512 Byte write buffer. Each 512-Byte aligned group of 512 Bytes is called a page. It is possible to program from one bit to 512 Bytes, anywhere within a page, in one programming operation. Each page in a sector may be programmed once before an erase of the sector is required to enable reprogramming of the same page in the sector.

A bit field programming method may be used to program bit resolution fields within a page. The method may be used multiple times to incrementally program different bit positions. Bits may only be programmed from a One (High) to a Zero (Low) state. An erase of the sector where the bit field is located is required to return any, and all, bits in the sector to the One (erased) state.

# 6. Address Space Maps

There are five address spaces within each device:

- A Non-Volatile Flash Memory Array used for storage of data that may be randomly accessed by asynchronous or synchronous read operations.
- A Read Only Memory Array used for factory programmed permanent device characteristics information. This area contains the Device Identification (ID) and Common Flash Interface (CFI) information.
- A One Time Programmable (OTP) Non-volatile Flash array used for factory programmed permanent data, and customer programmable permanent data. This is called the Secure Silicon Region (SSR).
- An OTP location used to permanently protect the SSR. This is call the SSR Lock.
- A volatile register set used to configure device behavior options. This is called the Configuration Register set.

The main Flash Memory Array is the primary and default address space but, it may be partially overlaid by the other four address spaces with one alternate address space available at any one time. The location where the alternate address space is overlaid is defined by the address provided in the command that enables each overlay. The portion of the command address that is sufficient to select a sector is used to select the sector that is overlaid by an alternate Address Space Overlay (ASO).

Any address range, within the overlaid sector, not defined by an overlay address map, is reserved for future use. All read accesses outside of an ASO address map within the selected sector, return non-valid data. The locations will display actively driven data but the meaning of whatever ones or zeros appear are not defined.

There are three operation modes for each bank that determine what portions of the address space are readable at any given time:

- Read Mode
- Embedded Algorithm (EA) Mode
- Address Space Overlay (ASO) Mode

Each bank of the device can be in any operation mode but, only one bank can be in EA or ASO mode at any one time.

In Read Mode a Flash Memory Array bank may be directly read by asynchronous or synchronous accesses from the host system bus. The Control Unit (CU) puts all banks in Read mode during Power-on, a Hardware Reset, after a Command Reset, or after a bank is returned to Read mode from EA mode. A bank with a suspended EA is considered to be returned to Read mode even though some or all of the data in the bank may be in an invalid state and thus not useful if read.

In EA mode the Flash memory array data in a bank is stable but undefined, and effectively unavailable for read access from the host system. While in EA mode the bank is used by the CU in the execution of commands. Typical EA mode operations are programming or erasing of data in the Flash array. All other banks are available for read access while the one bank is in EA mode. This ability to read from one bank while another bank is used in the execution of a command is called Simultaneous Read and Write (SRW) and allows for continued operation of the system via the reading of data or execution of code from other banks while one bank is programming or erasing data as a relatively long time frame background task.

In ASO mode, one of the overlay address spaces are overlaid in a bank (entered). That bank is in ASO mode and no other bank may be in EA or ASO mode. All EA activity must be completed before entering any ASO mode. A command for entering an EA or ASO mode while another bank is in EA or ASO mode will be ignored.

After issuing one of the ASO enter commands there is a delay required before the ASO is readable. The delay is generally less than 200 ns. It is recommended to issue a Status Read command to check the busy or ready state of the memory before reading from the ASO. This will take sufficient time as to ensure that the ASO data is available for read.

While an ASO mode is active (entered) in a bank, a read for Flash array data to any other bank is allowed. ASO mode selects a specific sector for the overlaid address space. Other sectors in the ASO bank still provide Flash array data and may be read during ASO mode. An ASO may overlay any sector in any bank.

While SSR Lock, SSR, or Configuration Register is overlaid only the SSR Lock, SSR, or Configuration Register respectively may be programmed in the overlaid sector.

While the SSR or SSR Lock ASO areas are being programmed all banks switch to EA mode. During EA mode no valid data may be read from a bank. This means that Simultaneous Read / Write is not supported for SSR or SSR Lock programming. Code controlling the SSR or SSR Lock programming operation must be executed from another memory, typically the code for this rarely performed programming operation is copied to RAM memory for execution.

While a Configuration Register is being programmed only the bank containing the ASO switches to EA mode. The ID/CFI and factory portion of the SSR ASO is not customer programmable.

During programming of any of these ASO areas, the Read Status Register command may be used with an address that selects the sector address where the ASO is active. This will cause the status register contents to overlay the ASO sector during the one read cycle for status that follows the Status Register Read command. After the one status information read access, the ASO returns to the EA mode condition of no valid data being readable. When the EA operation in the ASO is completed, the ASO information is again readable.

The Status Register Clear command may not be used while an ASO is entered. It is necessary to exit an ASO before using the Status Register Clear command.

## 6.1 Data Address & Quantity Nomenclature

A Bit is a single One or Zero data value. A Byte is a group of 8 bits aligned on an 8 bit boundary. A Word is a group of 16 bits aligned on a 16 bit boundary.

Throughout this document **quantities of data are generally expressed in terms of byte units**. Example: most sectors have 128 Kilo Bytes of data and is written as 128 KBytes or 128KB. **Addresses are also expressed in byte units.** A 128 KByte sector has an address range from 00000h to 1FFFFh Byte locations. Byte units are used because most host systems and software for these systems use byte resolution addresses. Software & hardware developers most often calculate code and data sizes in terms of bytes, so this is more familiar terminology than describing data sizes in bits or words. In general, data units will not be abbreviated if possible so that full unit names of Byte, Word, or bit are used. However, there may be cases where capital B is used for byte units and lower case b is used for bit units, in situations where space is limited such as in table column headers.

In some cases data quantities will also be expressed in word or bit units in addition to the quantity shown in bytes. This may be done as an aid to readers familiar with prior device generation documentation which often provided only word or bit unit values. Word units may also be used to emphasize that, in the memory devices described in this documentation, data is always exchanged with the host system in word units. Each bus cycle transfer of read or write data on the host system bus is a transfer 16 bits of data. A read bus cycle is always a16 bit wide transfer of data to the host system whether the host system chooses to look at all the bits or not. A write bus cycle is always a transfer of 16 bits to the memory device and the device will store all 16 bits to a register. In the case of a program operation all 16 bits of each word to be programmed will be stored in the Flash array.

The address nomenclature used in this document is a shorthand form that shows addresses are formed from a concatenation of high order bits, sufficient to select a Sector Address (SA = Flash Amax through A12), with low order bits to select a location within the sector. When in Read mode and reading from the Flash Array the entire address is used to select a specific word for asynchronous read or the starting word address of a burst read.

Because data is always transferred in word units, the memory devices being discussed use only the address signals from the system necessary to select words. The host system byte address uses system address a0 to select bytes and a1 to select words. Flash memories with word wide data paths have traditionally started their address signal numbering with A0 being the selector for words because a byte select input is not needed. So, system address a-maximum to a1 are connected to Flash A-maximum to A0 (the documentation convention here is to use lower case for system address signal numbering and upper case for Flash address signals). Example: Flash word address A22 to A0 is connected to system byte address a23 to a1.

Many commands to the Flash device use an address which is a merge (concatenation) of the upper address signals selecting the target sector with a specific bit pattern on the lower address signals. The specific bit pattern changes for different write cycles in a command sequence. The specific sequence of bit patterns reduce the chance that one or more misdirected or unintended write accesses will be interpreted as a valid command sequence, as random writes are unlikely to have the correct bit patterns, on the correct address signals, in the correct sequence. The bit patterns used are two complementary patterns of alternating ones and zeros often referred to by the hex characters used to represent the patterns - AAAh and 555h.

Because the number of address signals monitored for these patterns do not always match three hex characters (12 bits) and the patterns shift by one bit depending on whether the point of reference is word or byte address, there has been past confusion from the documentation of the correct address to use in each command. To reduce the chance for future confusion each address pattern has been given a specific name and binary documentation as follows:

CAP1 = Command Address Pattern 1 which is a merge (concatenation) of the upper address bits selecting the target sector with a first bit pattern. The Flash word address signals Amax to A12 must contain the target sector address and A11 to A0 must contain a binary bit pattern of 0101_0101_0101. In terms of a system byte address (typically used in software) the upper address bits a-max to a13 are merged with a binary bit pattern of 0_1010_1010_1010 on a12 to a0.

CAP2 = Command Address Pattern 2 where the Flash word address signals Amax to A12 must contain the target sector address and A11 to A0 must contain a binary bit pattern of 1010_1010_1010. In terms of a system byte address the upper address bits a-max to a13 are merged with a binary bit pattern of 1_0101_0101_0100 on a12 to a0. The LSB of byte address must be zero to ensure a word aligned address to place the command data in the lower order byte lane.

CAP3 = Command Address Pattern 3 where the Flash word address signals Amax to A12 must contain the sector address and A11 to A0 must contain a binary bit pattern of xxxx_0101_0101 (upper bits are don't care). In terms of a system byte address the upper address bits a-max to a13 are merged with a binary bit pattern of x_xxxx_1010_1010 on a12 to a0.

**Table 6.1** Command Address Patterns

| Name | Upper Address | Lower Address | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Word Address** | **A-max to A12** | **A11** | **A10** | **A9** | **A8** | **A7** | **A6** | **A5** | **A4** | **A3** | **A2** | **A1** | **A0** | |
| **Byte Address** | **a-max to a13** | **a12** | **a11** | **a10** | **a9** | **a8** | **a7** | **a6** | **a5** | **a4** | **a3** | **a2** | **a1** | **a0** |
| CAP1 | Target Sector | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| CAP2 | | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| CAP3 | | X | X | X | X | X | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

## 6.2 Flash Memory Array

The Non-Volatile Flash Memory Array is organized as shown in the following tables. Devices have either all uniform size sectors or small blocks at the top of memory map.

**Table 6.2** S29WS01GS 16 Bank Memory Map - Uniform Sectors

| Bank Size (Mbit) | Sector Count | Sector Size (KByte) | Bank | Sector Range | Address Range (word) | Address Range (byte) | Notes |
|---|---|---|---|---|---|---|---|
| 64 | 1024 | 128 | 0 | SA000–SA063 | 000000h–3FFFFFh | 000000h–7FFFFFh | Sector Starting Address – Sector Ending Address |
| | | | 1 | SA064–SA127 | ⋮ | ⋮ | |
| | | | 2 | SA128–SA191 | ⋮ | ⋮ | |
| | | | 3 | SA192–SA255 | ⋮ | ⋮ | |
| | | | 4 | SA256–SA319 | ⋮ | ⋮ | |
| | | | 5 | SA320–SA383 | ⋮ | ⋮ | |
| | | | 6 | SA384–SA447 | ⋮ | ⋮ | |
| | | | 7 | SA448–SA511 | 1C00000h–1FFFFFFh | 3800000h–3FFFFFFh | |
| | | | 8 | SA512–SA575 | ⋮ | ⋮ | |
| | | | 9 | SA576–SA639 | ⋮ | ⋮ | |
| | | | 10 | SA640–SA703 | ⋮ | ⋮ | |
| | | | 11 | SA704–SA767 | ⋮ | ⋮ | |
| | | | 12 | SA768–SA831 | ⋮ | ⋮ | |
| | | | 13 | SA832–SA895 | | | |
| | | | 14 | SA896–SA959 | 3800000h–3BFFFFFh | 7000000h–77FFFFFh | |
| | | | 15 | SA960–SA1023 | 3C00000h–3FFFFFFh | 7800000h–7FFFFFFh | |

***Note***

*This table has been condensed to show sector-related information for an entire device on a single page. Sectors and their address ranges that are not explicitly listed (such as SA008–SA009) have sector starting and ending addresses that form the same pattern as all other sectors of that size. For example, all 128 KB sectors have the byte address pattern x000000h–x1FFFFh.*

**Table 6.3** S29WS01GS 16 Bank Memory Map - Top Boot

| Bank Size (Mbit) | Sector Count | Sector Size (KByte) | Bank | Sector Range | Address Range (word) | Address Range (byte) | Notes |
|---|---|---|---|---|---|---|---|
| 64 | 960 | 128 | 0 | SA000–SA063 | 000000h–3FFFFFh | 000000h–7FFFFFh | Sector Starting Address – Sector Ending Address |
| | | | 1 | SA064–SA127 | ⋮ | ⋮ | |
| | | | 2 | SA128–SA191 | ⋮ | ⋮ | |
| | | | 3 | SA192–SA255 | ⋮ | ⋮ | |
| | | | 4 | SA256–SA319 | ⋮ | ⋮ | |
| | | | 5 | SA320–SA383 | ⋮ | ⋮ | |
| | | | 6 | SA384–SA447 | ⋮ | ⋮ | |
| | | | 7 | SA448–SA511 | 1C00000h–1FFFFFFh | 3800000h–3FFFFFFh | |
| | | | 8 | SA512–SA575 | ⋮ | ⋮ | |
| | | | 9 | SA576–SA639 | ⋮ | ⋮ | |
| | | | 10 | SA640–SA703 | ⋮ | ⋮ | |
| | | | 11 | SA704–SA767 | ⋮ | ⋮ | |
| | | | 12 | SA768–SA831 | ⋮ | ⋮ | |
| | | | 13 | SA832–SA895 | | | |
| | | | 14 | SA896–SA959 | 3800000h–3BFFFFFh | 7000000h–77FFFFFh | |
| 64 | 63 | 128 | 15 | SA960 | 3C00000h–3C0FFFFh | 7800000h–781FFFFh | |
| | | | | ⋮ | ⋮ | ⋮ | |
| | | | | SA1022 | 3FE0000h–3FEFFFFh | 7FC0000h–7FDFFFFh | |
| | 4 | 32 | | SA1023 | 3FF0000–3FF3FFFh | 7FE0000–7FE7FFFh | |
| | | | | SA1024 | 3FF4000–3FF7FFFh | 7FE8000–7FEFFFFh | |
| | | | | SA1025 | 3FF8000–3FFBFFFh | 7FF0000–7FF7FFFh | |
| | | | | SA1026 | 3FFC000–3FFFFFFh | 7FF8000–7FFFFFFh | |

***Note***
*All tables have been condensed to show sector-related information for an entire device on a single page. Sectors and their address ranges that are not explicitly listed (such as SA008–SA009) have sector starting and ending addresses that form the same pattern as all other sectors of that size. For example, all 128 KB sectors have the byte address pattern x000000h–x1FFFFh.*

## 6.3 Device ID and CFI (ID-CFI) ASO

There are two traditional methods for systems to identify the type of Flash memory installed in the system. One has traditionally been called Autoselect and is now referred to as Device Identification (ID). A command is used to enable an address space overlay where up to 16 word locations can be read to get JEDEC manufacturer identification (ID), and device ID from the Flash memory. The system can use the manufacturer and device IDs to select the appropriate driver software to use with the Flash device. The other method is called Common Flash Interface (CFI). It also uses a command to enable an address space overlay where an extendable table of standard information about how the Flash memory is organized and behaves can be read. With this method the driver software does not have to be written with the specifics of each possible memory device in mind. Instead the driver software is written in a more general way to handle many different devices but adjusts the driver behavior based on the information in the CFI table stored in the Flash memory. Traditionally these two address spaces have used separate commands and were separate overlays. However, the mapping of these two address spaces are non-overlapping and so can be combined in to a single address space and appear together in a single overlay. Either of the traditional commands used to access (enter) the Autoselect (ID) or CFI overlay will cause the now combined ID-CFI address map to appear.

A write at any sector address using CAP3, with xx98h or xx90h data, switches the addressed sector to an overlay of the ID-CFI address map. These are called ID-CFI Enter commands and are only valid when written to a bank that is in read mode. The ID-CFI address map appears within, and replaces Flash Array data of, the selected sector address range. The ID-CFI enter commands use the same address and data values used on previous generation memories to access the JEDEC Manufacturer ID (Autoselect) and Common Flash Interface (CFI) information, respectively. While the ID-CFI address space is overlaid, any write with xxF0h data to the device will remove the overlay and return the selected sector to showing Flash memory array

data. Thus, the ID-CFI address space and commands are backward compatible with standard memory discovery algorithms.

Within the ID-CFI address map there are two subsections:

**Table 6.4** ID-CFI Address Map Overview

| Byte Address | Description | Size Allocated (Max. Bytes) | Read/Write |
|---|---|---|---|
| (SA) + 00000h to 0001Fh | JEDEC ID<br>(traditional Autoselect values) | 32 | Read Only |
| (SA) + 00020h to 00BEh | CFI data structure | 480 | Read Only |

For the complete address map see Tables in Section 11.2, *Device ID and Common Flash Memory Interface Address Map on page 73*.

### 6.3.1 JEDEC Device ID

The Joint Electron Device Engineering Council (JEDEC) standard JEP106 defines a method for reading the manufacturer ID and device ID of a compliant memory. This information is primarily intended for programming equipment to automatically match a device with the corresponding programming algorithm.

The JEDEC ID information is structured to work with any memory data bus width e.g. x8, x16, x32. The code values are always byte wide but are located at bus width address boundaries such that incrementing the device address inputs will read successive byte, word, or double word locations with the codes always located in the least significant byte location of the data bus. Because the data bus is word wide each code byte is located in the lower half of each word location and the high order byte is always zero.

### 6.3.2 Common Flash Memory Interface

The Common Flash Interface (CFI) specification defines a standardized data structure that may be read from a flash memory device, which allows vendor-specified software algorithms to be used for entire families of devices. The data structure contains information for system configuration such as various electrical and timing parameters, and special functions supported by the device. Software support can then be device-independent, JEDEC ID-independent, and forward-and-backward-compatible for the specified flash device families.

The system can read CFI information at the addresses within the selected sector as shown in Section 11.2, *Device ID and Common Flash Memory Interface Address Map on page 73*.

Like the JEDEC Device ID information, CFI information is structured to work with any memory data bus width e.g. x8, x16, x32. The code values are always byte wide but are located at data bus width address boundaries such that incrementing the device address reads successive byte, word, or double word locations with the codes always located in the least significant byte location of the data bus. Because the data bus is word wide each code byte is located in the lower half of each word location and the high order byte is always zero.

For further information, please refer to the Spansion CFI Version 1.4 (or later) Specification (see also JEDEC publications JEP137 and JESD68.01). Please contact JEDEC (http://www.jedec.org) for their standards and the Spansion CFI Publications may be found at the Spansion Web site (http://www.spansion.com/CFI_v1.4_VendorSpec_Ext_A0.pdf at the time of this document's publication).

## 6.4 Secure Silicon Region ASO

The Secure Silicon Region (SSR) provides an extra Flash memory area that can be programmed once and permanently protected from further changes i.e. it is a One Time Program (OTP) area. The SSR cannot be erased. The SSR is 1Kbytes in length. It consists of 512 bytes for factory data and 512 bytes for customer-secured data.

The SSR is overlaid in the sector address specified by the SSR enter command.

**Table 6.5** Secure Silicon Region

| Byte Address Range | Secure Silicon Region | Size |
|---|---|---|
| (SA) + 0000h to 01FFh | Factory | 512 Bytes |
| (SA) + 0200h to 03FFh | Customer | 512 Bytes |

Page or Bit-field programming commands may be used to write data into the SSR. The Bit-field commands allow for incremental programming of bits, bytes, or words in the customer portion until that section is locked. However, when programming with these bit oriented programs a special format is used in which half of the SSR customer page is reserved and may not be programmed. This is explained later in the Bit-field programming command description.

## 6.5 SSR Lock ASO

The SSR Lock is a single word at offset zero in the ASO sector. The SSR Lock consists of two bits. The Customer Secure Silicon Region Protection Bit is in location 0. The Factory Secure Silicon Region Protection Bit is in location 1. The SSR Lock location may be read to determine the lock status of the Factory and Customer portions of the SSR. All other bits in this location return “1” if read. The Customer SSR Protection Bit may be programmed by the SSR Lock programming command to permanently lock the Customer portion of the SSR.

If the Customer SSR Protection Bit is programmed to “0”, the Customer SSR is protected and can not be programmed. If this bit is read as “0” the Customer SSR is locked. If this bit is read as “1,” the Customer SSR is available for programming. Once the Customer SSR has been programmed, the SSR Lock bit 0 should be programmed to “0.” to protect the Customer SSR from further programming.

Similarly the Factory Secure Silicon Region Protection Bit is set to “0” when protected. This area is always programmed and protected at the Spansion factory.

## 6.6 Configuration Register ASO

The Configuration Register Enter command is only valid when written to a bank that is in Read mode. The configuration register mode address map appears within, and replaces Flash Array data of, the selected sector address range. The meaning of the configuration register bits is defined in the configuration register operation description. In configuration register mode a write of 00F0h to any address will return the sector to Read mode. Each word (separate configuration register) of the Configuration Register ASO must be programmed by a separate programming operation i.e. a single word is loaded into the write buffer for each programming operation on the Configuration Register ASO.

# 7. Device Operations

This section describes the read and write bus operations, program, erase, simultaneous read/write, handshaking, and reset features of the Flash devices.

The address space of the Flash Memory Array is divided into banks. There are three operation modes for each bank:

- Read Mode
- Embedded Algorithm (EA) Mode
- Address Space Overlay (ASO) Mode

Each bank of the device can be in any operation mode but, only one bank can be in EA or ASO mode at any one time.

In Read Mode a Flash Memory Array bank may be read by simply selecting the memory, supplying the address, and taking read data when it is ready. This is done by asynchronous or synchronous accesses from the host system bus. The CU puts all banks in Read mode during Power-on, after a Hardware Reset, after a Command Reset (when an ASO is entered), or after a bank is returned to Read mode from EA mode.

During a synchronous read access valid read data is indicated by the RDY signal being in the Ready state (this may be High or Low depending on the RDY polarity setting in the configuration register). When RDY is in the not Ready state burst read data is not valid and wait states must be added. The use of the RDY signal to indicate when valid data is transferred on the system data bus is called handshaking or flow control.

EA and ASO modes are initiated by writing specific address and data patterns into command registers (see Section 11.1, *Command Definitions on page 71*). The command registers do not occupy any memory locations; they are loaded by write bus cycles with the address and data information needed to execute a command. The contents of the registers serve as input to the Control Unit (CU) and the CU dictates the function of the device. Writing incorrect address and data values or writing them in an improper sequence may place the device in an unknown state, in which case the system must write the reset command to return all banks to Read mode.

The Flash memory array data in a bank that is in EA mode, is stable but undefined, and effectively unavailable for read access from the host system. While in EA mode the bank is used by the CU in the execution of commands. Typical command operations are programming or erasing of data in the Flash array. All other banks are available for read access while the one bank is in EA mode. This ability to read from one bank while another bank is used in the execution of a command is called Simultaneous Read and Write (SRW) and allows for continued operation of the system via the reading of data or code from other banks while one bank is programming or erasing data as a relatively long time frame background task. Only a status register read command can be used in a bank in EA mode to retrieve the EA status.

While any one of the overlay address spaces are overlaid in a bank (entered) that bank is in ASO mode and no other bank may be in EA or ASO mode. All EA activity must be completed or suspended before entering any ASO mode. A command for entering an EA or ASO mode while another bank is in EA or ASO mode will be ignored.

While an ASO mode is active (entered) in a bank, a read for Flash array data to any other bank is allowed. ASO mode selects a specific sector for the overlaid address space. Other sectors in the ASO bank still provide Flash array data and may be read during ASO mode.

While SSR Lock, SSR, or Configuration Register is overlaid only the SSR Lock, SSR, or Configuration Register respectively may be programmed in the overlaid sector. While any of these ASO areas are being programmed the ASO bank switches to EA mode. The ID/CFI and factory portion of the SSR ASO is not customer programmable. An attempt to program in these areas will fail.

## 7.1 Bus Operations

Table 7.1 describes the required state of each input signal for each bus operation.

**Table 7.1** Bus Operations

| Operation | CE# | OE# | WE# | CLK | AVD# | A28-A16 | A15-A0 | DQ 15-DQ0 | RESET# |
|---|---|---|---|---|---|---|---|---|---|
| **Standby & Reset** | | | | | | | | | |
| Standby (CE# deselect) | H | X | X | X | X | X | X | High-Z | H |
| Hardware Reset | X | X | X | X | X | X | X | High-Z | L |
| **Asynchronous Mode Operations** | | | | | | | | | |
| Asynchronous Address Latch | L | H | X | X | ↑ | Addr In | Addr In | X | H |
| Asynchronous Read | L | L | H | X | H | X | X | Data Output Valid | H |
| Asynchronous Write Latched Data | L | H | ↑ | X | H | X | X | Data Input Valid | H |
| **Synchronous Mode Operations** | | | | | | | | | |
| Latch Starting Burst Address by CLK | L | H | H | ↑ | L | Addr In | Addr In | X | H |
| Burst Read and advance to next address (See Note) | L | L | H | ↑ | H | X | X | Data Output Valid | H |
| Terminate current Burst cycle | ↑ | X | X | X | X | X | | High-Z | H |

***Legend***
*L = Logic 0, H = Logic 1, X = can be either $V_{IL}$ or $V_{IH}$,* ↑ = rising edge.

***Note***
*Data is delivered by a read operation only after the burst Initial Access Cycle count has been satisfied.*

## 7.2 Asynchronous Read

In order to use Asynchronous Read Mode the configuration register bit 15 must be set to 1. This is the default after power-on or hardware reset.

To read data from the memory array, the system must first assert a valid address or address portions while driving AVD# and CE# to $V_{IL}$. WE# must remain at $V_{IH}$. CLK may toggle or remain at $V_{IL}$ or $V_{IH}$.

The rising edge of AVD# latches the full address or AVD# may remain low while address remains stable. The data appears on DQ15–DQ0 when CE# is Low, OE# is Low, AVD# is High (or AVD# remains Low while address remains stable), and the asynchronous access time is satisfied. AVD# must not be tied low. For systems that do not provide an AVD# signal, the Flash AVD# input may be tied to CE#. Burst mode is not supported if AVD# is tied to CE# because burst mode requires AVD# to be pulsed for one cycle to start the initial access of a burst.

Address access time ($t_{ACC}$) is equal to the delay from stable addresses to valid output data. The chip enable access time ($t_{CE}$) is the delay from stable CE# to valid data at the outputs. See $t_{ACES}$ in 10.7.2, *AC Characteristics–Asynchronous Read on page 61*.

### 7.2.1 Asynchronous Intra Cache Line Read

The device is capable of fast read of additional locations within a cache line after the initial access of the line. This provides fast ($t_{ICCC}$) random read access speed for multiple locations within a cache line. Address bits Amax–A4 select a 16-word cache line, and address bits A3 – A0 select a specific word within that cache line. This is an asynchronous operation with the microprocessor supplying the specific word location. During intra cache line read AVD# must remain low during the access if multiple words are selected during the access. Address input must be valid and stable while AVD# is low during the intra cache line read.

The random or initial page access is $t_{ACC}$ or $t_{CE}$ and subsequent page read accesses (as long as the locations specified falls within that cache line) is equivalent to $t_{ICCC}$. When CE# or AVD# is deasserted (=$V_{IH}$), the reassertion of CE# or AVD# for subsequent access has access time of $t_{ACC}$ or $t_{CE}$. See Figure 10.13 on page 63.

#### 7.2.1.1 Cache Line Persistence

The last Cache Line (CL) read stays in the Cache Line read buffer until another cache line is read. The CL data will stay in the CL read buffer even if CE# is de-asserted between read operations.

## 7.3 Synchronous Burst Read Mode and Configuration Register

The device is capable of synchronous, continuous sequential burst operation and linear burst operation of a preset length.

The device natively operates on a 32-byte aligned group of 32-bytes, which is called a Cache Line (CL). Every read operation accesses a CL in parallel then delivers to the data bus either as a single word from the CL, when the bus is in asynchronous mode or, as a sequence of words from the CL in synchronous mode.

In continuous sequential burst read, when a cache line boundary is crossed, wait states, equal to the programmable Initial Access Cycles value minus one, are inserted between the last word of a cache line and the first word of the next cache line. The wait states are indicated by the RDY signal going Low.

The device supports fixed length linear burst read operations of 8 or 16 word length with wrap around at the 8 or 16 word aligned boundary. A burst access may be terminated early by taking CE# High.

In order to use Burst Read Mode the configuration register bit 15 must be set to 0. Other Configuration Register values set various operational parameters associated with burst mode.

Prior to entering burst mode, the system should determine how many Initial Access Cycles (IAC) are needed for the initial word of each burst access (see table below), what mode of burst operation is desired, how the RDY signal transitions with valid data, and output drive strength. The system would then write the configuration register command sequence.

The Configuration Register can also be read while in Configuration Register mode. Section 6.6, *Configuration Register ASO on page 20* describes the register settings.

To burst read data from the memory array, the system must assert CE# to VIL, and provide a valid address while driving AVD# to VIL for one cycle. The data appears on DQ15–DQ0 when CE# remains Low, after OE# is Low and the synchronous access times are satisfied. The next data in the burst sequence is read on each clock cycle that OE# and CE# remain Low.

Initial access time ($t_{IA} + t_{BACC}$) is equal to the delay from the rising edge of CLK when OE# is High and AVD# is Low to valid output data. The chip enable access time ($t_{CE}$) is the delay from stable CE# to valid data at the outputs. See Section 10.7.1, *AC Characteristics–Synchronous Burst Read on page 59*.

A burst may be terminated early by taking CE# to $V_{IH}$ only after the first word of the burst is read.

The following table shows the number of Initial Access Cycles (IAC) needed at different burst frequencies. IAC are counted from the rising edge of clock when AVD# is Low, to the rising edge of clock when the first data word is valid on the data bus (see Figure 7.1, *Example of Programmable Initial Access Cycles on page 24*).

The formula for calculating maximum frequency is:

```
F-max = (Initial Access Cycles - 1) / Internal Access Time (tIA).
Example: 60,000,000 Hz = (7 IAC - 1) / 0.0000001 Sec.
```

Alternatively:

```
F-max (MHz) = 1000 x ((IAC -1) / Internal Access Time (tIA) (ns) ).
Example: 60 MHz = 1000 x ((7 IAC -1) / 100 ns).
```

**Table 7.2** Initial Access Cycles vs. Frequency

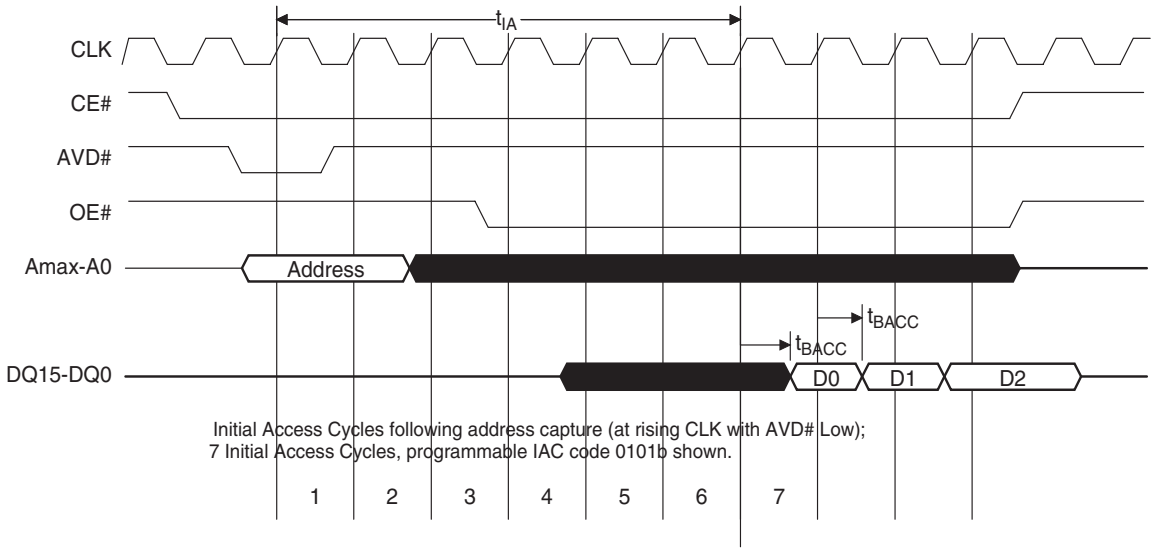
| Initial Access | Wait States | CR0[14:11] Settings | Cycles Frequency (MHz) |
|---|---|---|---|
| 9 | 8 | 0111 | 54 MHz < Frequency ≤ 66 MHz |
| 11 | 10 | 1001 | 67 MHz < Frequency ≤ 83 MHz |
| 13 | 12 | 1011 | 84 MHz < Frequency ≤ 104 MHz |
| 14 | 13 | 1100 | 105 MHz < Frequency ≤ 108 MHz |

***Note***

*Maximum frequency for each setting is limited to the supported maximum frequency of the related ordering information valid combination (speed bin).*

### 7.3.1 Initial Access Cycles Configuration Register Setup

**Figure 7.1** Example of Programmable Initial Access Cycles




| Configuration Register | Programmable Initial Access Cycles | | | |
|---|---|---|---|---|
| CR0[14:11] | 0000 = | initial data is valid on the | Reserved | rising CLK edge after addresses are latched |
| | 0001 = | | 3rd | |
| | 0010 = | | 4th | |
| | 0011 = | | 5th | |
| | 0100 = | | 6th | |
| | 0101 = | | 7th | |
| | 0110 = | | 8th | |
| | 0111 = | | 9th | |
| | 1000 = | | 10th | |
| | . . . | | . . . | |
| | 1100 = | | 14th | |
| | 1101 = | | 15th | |
| | 1110 = | Reserved | | |
| | 1111 = | | | |

***Note***
*Only the CR0[14:11] settings shown in Table 7.2 are tested.*

### 7.3.2 Continuous Burst

In continuous burst mode, the device continues to output sequential burst data from the memory array until the burst is terminated. The access continues until the system drives CE# high, RESET# low, or AVD# low in conjunction with a new address. See Table 7.1, *Bus Operations on page 22*. If the burst is not terminated at the highest addressable memory location related to the active CE#, the data output for locations beyond the highest addressable memory location is undefined.

As each 16 word (32 byte Cache Line) boundary is crossed another initial access delay is required to provide the first word of the next cache line. The wait states during which data is not valid are indicated by the RDY signal being in the not ready state.

### 7.3.3 8-, 16-Word Linear Burst with Wrap Around

**Table 7.3** Burst Address Groups

| Mode | Group Size | Group Byte Address Ranges |
|---|---|---|
| 8-word | 16 bytes | 0-Fh, 10-1Fh, 20-2Fh,... |
| 16-word | 32 bytes | 0-1Fh, 20-3Fh, 30-4Fh,... |

In a burst read, 16 or 32-Bytes are read from consecutive addresses a 16-bit word at a time. The burst read address sequence is determined by the initial word address within the 16 or 32-Byte aligned group in which the burst address falls.

For a 32-byte burst, if the starting byte address in the 16-word group is 3Ch, the address range to be read is 20-3Fh, and the burst sequence would be 3C-3E-20-22-24-26-28-2A-2C-2E-30-32-34-36-38-3Ah. The burst sequence begins with the starting address, and is followed by each higher address word up to the end of the 32-Byte aligned group, then the access wraps back to the first address in the selected 32-Byte aligned group, and continues until the maximum of 16 words is delivered. No additional wait states are required within the 16-word burst.

A 16-byte burst is similar except the wrap around point is at the 8 word aligned boundary.

### 7.3.4 Configuration Registers

Configuration register zero (CR0) sets various operational parameters associated with burst mode. Upon power-up or hardware reset, the device defaults to the idle state, and the configuration register settings are in their default state. The host system should determine the proper settings for the configuration register, and then execute the Set Configuration Register command sequence, before attempting burst operations. The Configuration Register can also be read using a command sequence (see 11.1, *Command Definitions on page 71*). The table below describes the register settings and indicates the default state of each bit after power-on or a hardware reset. The configuration register bits are not affected by a command reset.

**Table 7.4** Configuration Register 0 (CR0)

| SICR Bit | Function | Settings (Binary) | | |
|---|---|---|---|---|
| CR0.15 | Device Read Mode | 0 = Synchronous Read Mode | | |
| | | 1 = Asynchronous Read Mode (Default) | | |
| CR0.14<br>CR0.13<br>CR0.12<br>CR0.11 | Programmable Initial Access Cycles | Initial data is valid on the | | rising CLK edge after address is latched |
| | | 0011 = Reserved | | |
| | | . | . | |
| | | . | . | |
| | | . | . | |
| | | 0111 = | 9th | |
| | | 1000 = Reserved | | |
| | | 1001 = | 11th | |
| | | 1010 = Reserved | | |
| | | 1011 = | 13th (Default) | |
| | | 1100 = | 14th | |
| | | 1101 = | 15th | |
| | | 1110 = Reserved | | |
| | | 1111 = Reserved | | |
| CR0.10 | RDY Polarity | 0 = RDY signal is active low | | |
| | | 1 = RDY signal is active high (Default) | | |
| CR0.9 | Reserved | 0 = Reserved | | |
| | | 1 = Reserved (Default) | | |
| CR0.8 | RDY Timing | 0 = RDY active one clock cycle before data | | |
| | | 1 = RDY active with data (Default) | | |
| CR0.7 | Output Drive Strength | 0 = Full Drive= Current Driver Strength (Default) | | |
| | | 1 = Half Drive | | |
| CR0.6 | Reserved | 0 = Reserved | | |
| | | 1 = Reserved (Default) | | |
| CR0.5 | Reserved | 0 = Reserved (Default) | | |
| | | 1 = Reserved | | |
| CR0.4 | Reserved | 0 = Reserved (Default) | | |
| | | 1 = Reserved | | |
| CR0.3 | Reserved | 0 = Reserved | | |
| | | 1 = Reserved (Default) | | |
| CR0.2<br>CR0.1<br>CR0.0 | Read Burst Length | 000 = Continuous (Default) | | |
| | | 010 = 8 word (16-Byte) Linear Burst with wrap around | | |
| | | 011 = 16 word (32-Byte) Linear Burst with wrap around | | |
| | | (All other bit settings are reserved) | | |

### 7.3.4.1 Device Read Mode

Configuration Register zero bit 15 (CR0.15) controls whether read accesses via the bus interface are in asynchronous or synchronous mode. Asynchronous mode is the default after power-on or hardware reset. Write accesses are always conducted with asynchronous mode timing, independent of the read mode.

### 7.3.4.2 Initial Access Cycles

Configuration Register zero bits 14 to 11 (CR0.[14:11]) define the total number of cycles after the AVD# Low cycle that captures the initial address through the cycle that read data is valid. The bits from 14 to 11 are in most to least significant order. The random address access at the beginning of each read burst takes longer than the subsequent read cycles. The memory bus interface must be told how many cycles to wait before driving valid data then advancing to the next data word. The number of initial cycles will vary with the memory clock rate. The minimum number of cycles is 3 and the maximum is 15. The default after power-on or hardware reset is 13 cycles.

When the appropriate number of Initial Access Cycles minus one have occurred, data is output after the rising edge of the CLK. Valid data is then available after $t_{BACC}$ and may be captured at the next rising edge. Subsequent words are output $t_{BACC}$ after the rising edge of each successive clock cycle, which automatically increments the internal address counter.

### 7.3.4.3 RDY Polarity

Configuration Register Zero, bit 10 (CR0.10), controls whether the RDY signal indicates valid data when High or when Low. When this bit is zero the RDY signal indicates data is valid when the signal is Low. When this bit

is one the RDY signal indicates data is valid when the signal is High. The default for this bit is set to one after power-on or a hardware reset.

#### 7.3.4.4 RDY Timing

Configuration Register Zero, bit 8 (CR0.8), controls whether the RDY signal indicates valid data on the same cycle that data is valid or one cycle before data is valid. When this bit is one the RDY signal indicates data is valid in the same cycle the data is valid. When this bit is zero the RDY signal is asserted the cycle before data is valid. The default for this bit is set to one after power-on or a hardware reset.

#### 7.3.4.5 Output Drive Strength

Configuration Register Zero, bit 7 (CR0.7), controls whether the data outputs drive with full or half strength. When this bit is zero the data outputs drive with full strength. When this bit is one the data outputs drive with half strength. The default for this bit is cleared to zero after power-on or a hardware reset.

#### 7.3.4.6 Burst Length

Configuration Register Zero bits 2 to 0 (CR0.[2:0]) define the length of burst read accesses. The bits from 2 to 0 are in most to least significant order. See the register table for code meaning & default value.

#### 7.3.4.7 CR0 Default Value

The value in CR0 after power-on or hardware reset is DF48h.

## 7.4 Status Register

The status of program and erase operations is provided by a single 16 bit status register. The Status Register Read command is written followed by one read access of the status register information. The contents of the status register overlays the sector selected by the Status Register Read command. The overlay is in effect for one read access, specifically the next read access that follows the Status Register Read command, in the sector selected by the command. Read accesses to other sectors return normal memory array or ASO data. After the one status register access, the Status Register Read command selected sector returns to providing memory array or ASO data when read.

The status register can be read in synchronous or asynchronous bus access mode. If read in synchronous burst mode for more than one access cycle, the same status results will appear in each of the read cycles of the burst access until the burst is terminated.

The status register contains bits related to the results - success or failure - of the most recently completed Embedded Algorithms (EA): Erase Status (bit 5), and Program Status (bit 4), and Sector Locked Status (bit 1), and, bits related to the current state of any in process EA: Device Busy (bit 7), Erase Suspended (bit 6), Program Suspended (bit 2), and Bank Status (bit 0). The current state bits indicate whether an EA is in process, suspended, or completed.

The upper 8 bits (bits 15:8) are reserved. These have undefined high or low value that may change from one status read to another. These bit should be treated as don't care and ignored by any software reading status.

The Clear Status Register Command will clear to Zero the results related bits of the status register but will not affect the current state bits.

**Table 7.5** Status Register Reset State

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 1 at Reset | 0 at Reset | 0 at Reset | 0 at Reset | 0 at Reset | 0 at Reset | 0 at Reset | 0 at Reset |

***Notes***

1. *Bit 7 reflects the device status.*
2. *If the device is busy, Bit 0 is used to check whether the addressed bank is busy or some other bank is busy.*
3. *All the other bits reflect the status of the device.*

**Table 7.6** Status Register - Bit 7

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 0<br>Device busy programming or erasing | Invalid | Invalid | Invalid | Invalid | Invalid | Invalid | VALID |
| 1<br>Device ready | VALID | VALID | VALID | VALID | VALID | VALID | VALID |

***Notes***

1. *Bit 7 is set when there is no erase or program operation in progress in the device.*
2. *Bits 1 thru 6 are valid if and only if Bit 7 is set.*

**Table 7.7** Status Register - Bit 6

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 1<br>Bits 6:1 only valid when Bit 7 = 1 | 0<br>No Erase in Suspension | X | X | X | X | X | X |
| 1<br>Bit 6:1 only valid when Bit 7 = 1 | 1<br>Erase in Suspension | X | X | X | X | X | X |

***Notes***

1. *Upon issuing the "Erase Suspend" Command, the user must continue to read status until DRB becomes 1 before accessing another sector within the same bank.*
2. *Cleared by "Erase Resume" Command.*

**Table 7.8** Status Register - Bit 5

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 1<br>Bits 6:1 only valid when Bit 7 = 1 | X | 0<br>Erase/Blank Check successful | X | X | X | X | X |
| 1<br>Bit 6:1 only valid when Bit 7 = 1 | X | 1<br>Erase/Blank Check error | X | X | X | X | X |

***Notes***

1. *ESB bit reflects "success" or "failure" of the most recent erase or blank check operation.*
2. *Cleared by "Clear Status Register" Command as well as by hardware reset.*

**Table 7.9** Status Register - Bit 4

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 1<br>Bits 6:1 only valid when Bit 7 = 1 | X | X | 0<br>Program successful | X | X | X | X |
| 1<br>Bit 6:1 only valid when Bit 7 = 1 | X | X | 1<br>Program fail | X | X | X | X |

***Notes***

1. *PSB bit reflects "success" or "failure" of the most recent program operation.*
2. *Cleared by "Clear Status Register" Command as well as by hardware reset.*

**Table 7.10** Status Register - Bit 3

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 1<br>Bits 6:1 only valid when Bit 7 = 1 | X | X | X | X | X | X | X |

***Notes***

1. *This Register is reserved for future use.*

**Table 7.11** Status Register - Bit 2

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 1<br>Bits 6:1 only valid when Bit 7 = 1 | X | X | X | X | 0<br>No Program in suspension | X | X |
| 1<br>Bit 6:1 only valid when Bit 7 = 1 | X | X | X | X | 1<br>Program in suspension | X | X |

***Notes***

1. *Upon issuing the "Program Suspend" Command, the user must continue to read status until DRB becomes 1 before accessing another sector within the same bank.*
2. *Cleared by "Program Resume" Command.*

**Table 7.12** Status Register - Bit 1

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 1<br>Bits 6:1 only valid when Bit 7 = 1 | X | X | X | X | X | 0<br>Sector not locked during operation | X |
| 1<br>Bit 6:1 only valid when Bit 7 = 1 | X | X | X | X | X | 1<br>Sector locked error | X |

***Notes***

1. *SLSB indicates that a program or erase operation failed to program or erase because the sector was locked or the operation was attempted on the protected Secure Silicon Region.*
2. *SLSB reflects the status of the most recent program or erase operation.*
3. *SLSB is cleared by "Clear Status Register" or by hardware reset.*

**Table 7.13** Status Register - Bit 0

| Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|
| Device Ready Bit<br>Overall status | Erase Suspend Status Bit | Erase Status Bit | Program Status Bit | RFU | Program Suspend Status Bit | Sector Lock Status Bit | Bank Status Bit |
| DRB | ESSB | ESB | PSB | RFU | PSSB | SLSB | BSB |
| 0<br>Bits 6:1 only valid when Bit 7 = 1 | X | X | X | X | X | X | 0<br>Program or Erase op. in addressed Bank |
| 0<br>Bits 6:1 only valid when Bit 7 = 1 | X | X | X | X | X | X | 1<br>No Program or Erase op. in addressed Bank |
| 1<br>Bit 6:1 only valid when Bit 7 = 1 | X | X | X | X | X | X | 0<br>No active Program or Erase op. |
| 1<br>Bit 6:1 only valid when Bit 7 = 1 | X | X | X | X | X | X | 1<br>invalid |

***Notes***

1. *BSB is used to check if a program or erase operation is in progress in the current bank.*

## 7.5 Simultaneous Read/Write

The simultaneous read/write (SRW) feature allows the host system to read data from one bank of memory while an Embedded Algorithm (EA) such as programming or erasing is operating in another bank of memory. Sometimes this feature is also referred to as Read While Write (RWW). Section 10.7.4, *Back to Back Read and Write Combinations on page 65* shows how read and write cycles may be alternated in order to initiate an EA, for simultaneous operation, with zero latency between the read and write accesses. Refer to the *DC Characteristics on page 56* table for the read-while-write current specification.

## 7.6 Blank Check

The Blank Check command will confirm if the selected sector is erased.

The Blank Check command does not allow for reads to the array during the Blank Check. Reads to the array while this command is executing will return unknown data.

- To initiate a Blank Check on Sector X, write 33h to address CAP1 in Sector X when no other EA is active or suspended.
- The Blank Check command may not be written while the device is actively programming or erasing. Blank Check does not support simultaneous operations.
- Use the Status Register read to confirm if the device is still busy and when compete if the sector is blank or not.
- Bit 5 of the Status Register will be cleared to zero if the sector is erased and set to one if not erased.
- Bit 7 & Bit 0 of the Status Register will show if the device is performing a Blank Check (similar to an erase operation).
- As soon as any bit is found to not be erased, the device will halt the operation and report the results.
- Once the Blank Check is completed, the device will to return to the Idle State.

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Blank Check Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0033h |

### Sample Code

```
/* Example: Blank Check Command */

UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */

*CAP1 = 0x0033; /* Write Blank Check command */
/* poll for completion */
```

## 7.7 Writing Commands/Command Sequences

The device accepts Asynchronous write bus operations. During an asynchronous write bus operation, the system must drive CE# and WE# to $V_{IL}$, and OE# to $V_{IH}$ when providing an address and data. When latching an address, AVD# must be driven to $V_{IL}$. Addresses are latched on the rising edge of AVD#, while data is latched on the rising edge of WE#. See the Table 7.1, *Bus Operations on page 22* for the signal combinations that define each phase of a write bus operation to the device.

All write bus operations ignore the clock (CLK) input and all signal transitions must satisfy the asynchronous write timing requirements. However, write bus operations may be performed while the device remains in synchronous read mode as set in the Configuration register. Control, address, and data signal transitions may be related to a clock, i.e. the system may remain in a synchronous bus mode so long as the signal transitions to the device meet the asynchronous write timing requirements. But, the device does not support burst write accesses. Each write must convey a single address and data word pair to the device. If the host system uses a burst write bus protocol the burst should be terminated at a length of one word as the device will not capture multiple words from a burst write access. Synchronous read burst in combination with synchronous write operations (meeting the asynchronous write timing requirements) are illustrated in Section 10.7.4, *Back to Back Read and Write Combinations on page 65*.

Each write is a command or part of a command sequence to the device. The address provided in each write operation may be a bit pattern used to help identify the write as a command to the device. The upper portion of the address may also select the bank or sector the command operation is to be performed. A *Bank Address* (BA) is the set of address bits required to uniquely select a bank. A *Sector Address* (SA) includes Amax through A12 Flash address bits (system byte addresses a-max through a1). The data in each write identifies the command operation to be performed or supplies information needed to perform the operation. See 11.1, *Command Definitions on page 71* for a listing of the commands accepted by the device. $I_{CC2}$ in *DC Characteristics on page 56* represents the active current specification for a write (Embedded Algorithm) operation.

## 7.8 Program/Erase Operations

To initiate any program and/or erase operation, as with writing any command sequence, the system must drive AVD# and CE# to $V_{IL}$, and OE# to $V_{IH}$ when providing an address to the device, and drive WE# and CE# to $V_{IL}$, and OE# to $V_{IH}$ when writing commands or data.

Addresses are latched on the rising edge of AVD# during asynchronous writes. Data is latched on the rising edge of WE# during asynchronous writes.

Note the following:

- A *0* cannot be programmed back to a *1*. A succeeding read shows that the data is still *0*. Only erase operations can convert a *0* to a *1*.

| | |
|---|---|
| old data | 0011 |
| new data | 0101 |
| results | 0001 |

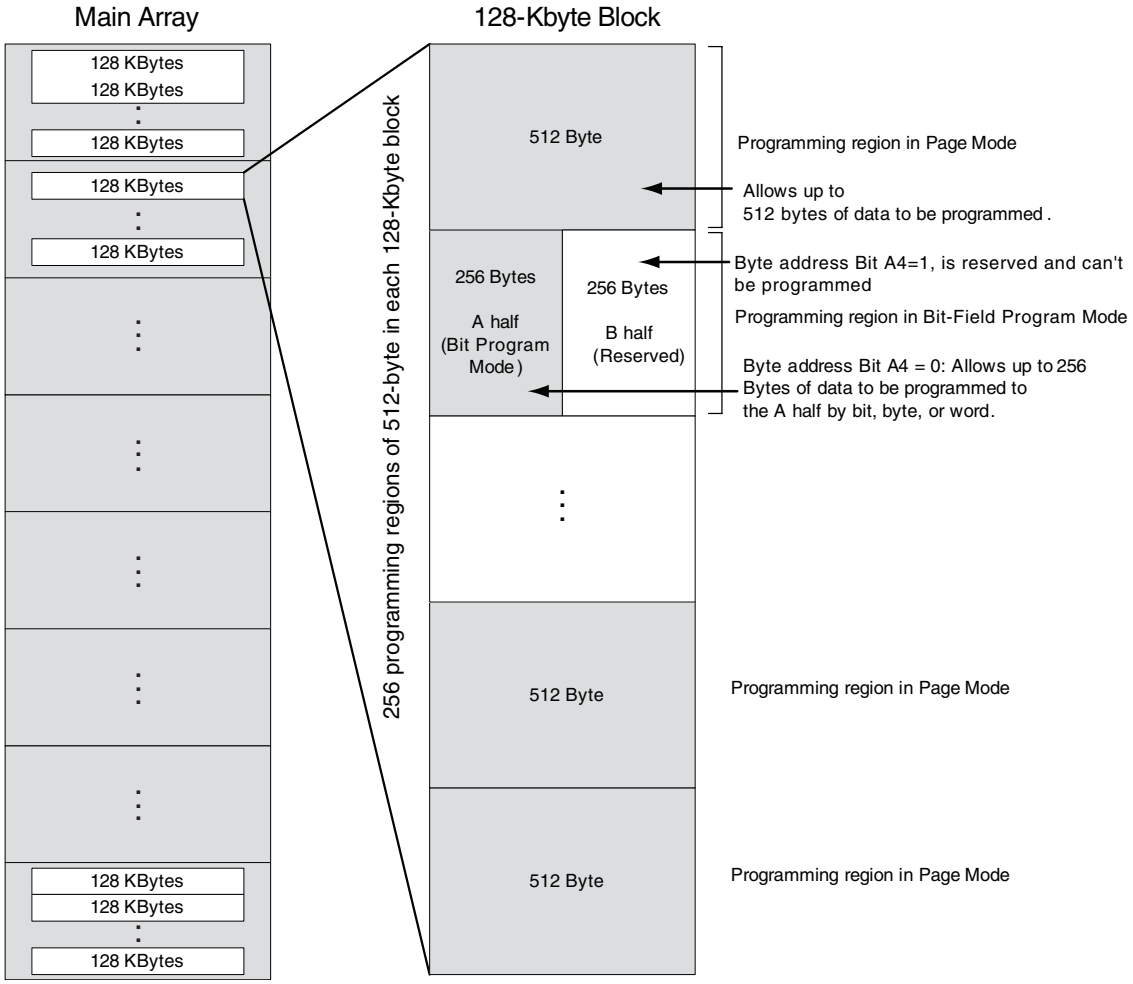
- The duration of program and erase operations is shown in Section 10.7.5, *Erase and Programming Performance on page 70*.
- Program and erase operations may be suspended. An erase operation may be suspended to allow either programming or reading of another sector (but not in the erase sector) in any bank. No other erase operation can be started during an erase suspend. A program operation may be suspended to allow reading of another location in any bank. Bits being programmed by the suspended program operation will be undefined during the suspend, all other locations are readable. No other program or erase operation may be started during a suspended program operation - program or erase commands will be ignored during a suspended program operation. After an intervening program operation or read access is complete the suspended erase or program operation may be resumed. Program and Erase operations may be interrupted as often as necessary but in order for a program or erase operation to progress to completion there must be some periods of time between resume and the next suspend commands greater than or equal to $t_{PRS}$ or $t_{ERS}$ in Section 10.7.5, *Erase and Programming Performance on page 70*.
- When the Embedded Program algorithm is complete, the device returns to the calling routine (Erase Suspend, SSR Lock, Secure Silicon Region, or Idle State).
- The system can determine the status of the program operation by reading the Status Register. Refer to *Status Register on page 27* for information on these status bits.
- Any commands written to the device during the Embedded Program Algorithm are ignored except the Program Suspend, and Status Read command. Any commands written to the device during the Embedded Erase Algorithm are ignored except Erase Suspend and Status Read command.
- A hardware reset immediately terminates any in progress program/erase operation, thus the terminated operation should be reinitiated once the device has returned to the idle state, to ensure data integrity.

### 7.8.1 Program Methods

There are two methods of programming that may be used, Page or Bit-Field. The Page method programs an entire 512 Byte group with any bit pattern but is limited to programming that page once before an erase is required. The Bit-Field method uses a specific format for the data in a page that limits the available space in the page to 256 Bytes but allows for the same page to be programmed multiple times to enable incremental programming of bits in the page. Each method affects the entire page it is used on. Each page may be programmed by either method such that pages programmed by the different methods may be mixed within any sector. Any memory sector can have a combination of any number of dedicated page-program, bit-field program or blank pages.

See Figure 7.2 for the memory array partition under Bit-Field and Page programming methods.

**Figure 7.2** Configurable Programming Groups: Bit-Field Program and Page Program




#### 7.8.1.1 Page Programming Method

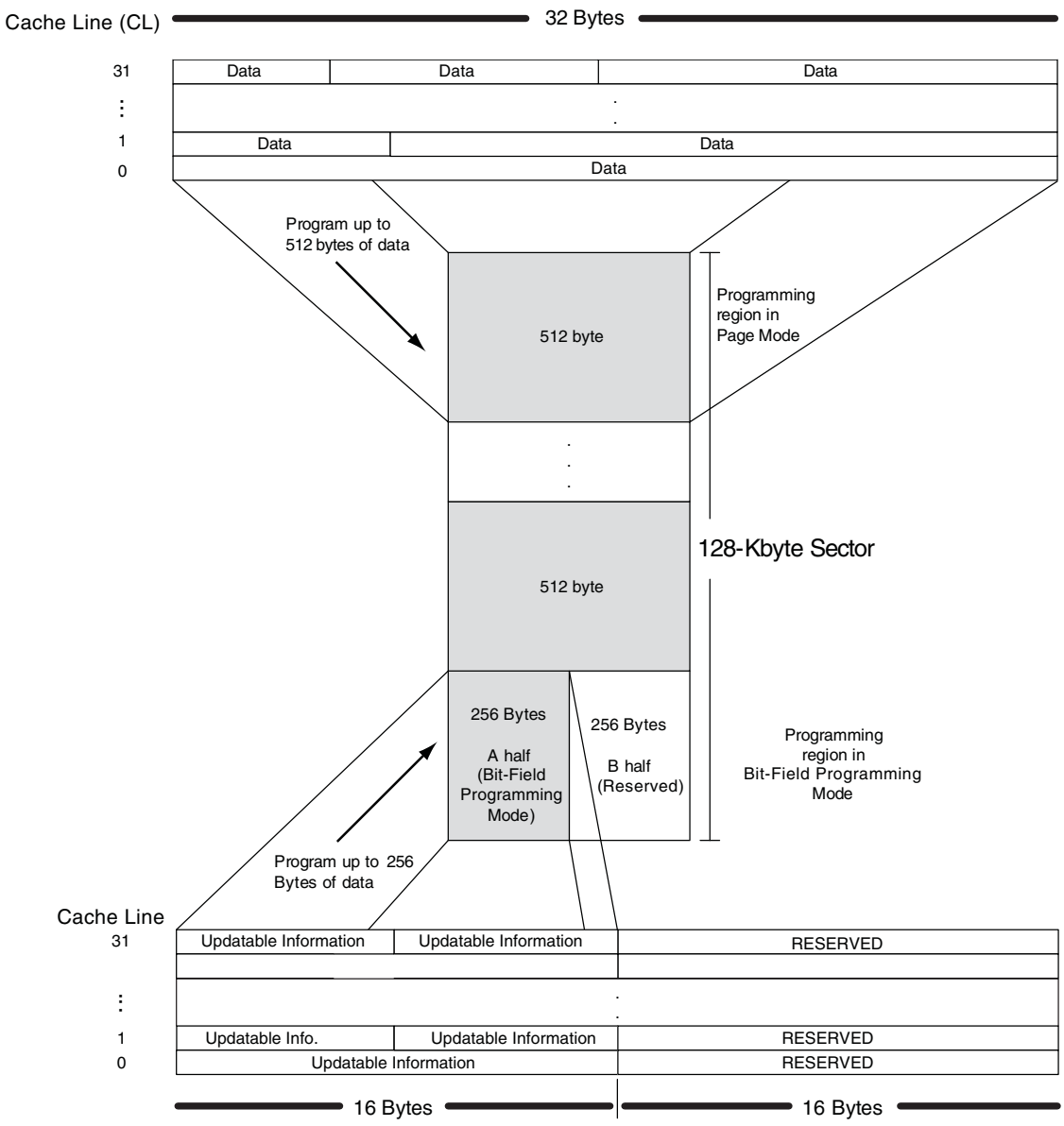
Each 512-Byte aligned group of 512 Bytes is called a page. Bits to be programmed may be located anywhere within a page. Using the Page programming method each page in a sector may be programmed only once before an erase of the sector is required to enable reprogramming of the same page in the sector. Typical device programming is done by loading a 512 Byte write buffer and issuing a programming command to move data from the buffer to the memory array. This sets an upper limit on the amount of data that may be programmed with a single programming command. The write buffer must also be aligned to each 512 Byte boundary. In addition, smaller groups of data may also be programmed by the Bit-Field programming method. It is possible to program from one bit up to 512 Bytes in each programming operation.

### 7.8.1.2 Bit-Field Programming Method

A Bit-Field programming method may be used to program individual bits within a page. With this method any combination of bits within the lower 16 bytes of a cache line (CL) may be programmed and the same bytes may be programmed multiple times in order to incrementally program different bits in those bytes. The remaining higher address 16 bytes of the CL are considered reserved. Reading these reserved locations within a page after bit-field programming is completed will provide stable but undefined data.

See Figure 7.3 for the Memory Array Configuration for Bit Field and Page Programming Methods.

**Figure 7.3** Programming Groups: Bit Program and Page Mode




Any word location within the user area, shown as A=half in Figure 7.3 (Flash word address bit A3=0 or system byte address bit a4=0) may be programmed (bits may be cleared to Zero). Any word location in the reserved area, shown as B-half in Figure 7.3 (Flash address A3=1 system address a4=1) must not have bits programmed to Zero. Any attempt to program a Zero to a One, of a bit that is already a Zero will be ignored and the bit will remain a Zero.

In addition, any attempt to clear to Zero of bits within the reserved area will not be done and will cause a program failure to be indicated in the status register. Bit-Field programming of the lower 16 bytes in a CL may

be repeated up to 128 times, allowing the lower 16 bytes in a CL to be programmed even a single bit at a time. When any portion of a CL is programmed by the Bit-Field programming command, all CL within the same 512-byte aligned, 512-byte group (page), containing the CL selected by the programming command, will be treated as being programmed by the Bit-Field programming command. This means all CL in the 512 Byte group will have the same format of user and reserved portions. Bit-Field programming is thus a finer resolution but lower density programming method as one half of the bytes within a 512 Byte group are available for programming and one half are reserved. The address mapping between the user and reserved locations is shown in Table 7.14.

**Table 7.14** Bit-Field Programming Page Map

| Page Byte Offset | Status | Page Byte Offset | Status |
|---|---|---|---|
| 000h - 00Fh | programmable | 010h - 01Fh | reserved |
| 020h - 02Fh | programmable | 030h - 03Fh | reserved |
| 040h - 04Fh | programmable | 050h - 05Fh | reserved |
| 060h - 06Fh | programmable | 070h - 07Fh | reserved |
| 080h - 08Fh | programmable | 090h - 09Fh | reserved |
| 0A0h - 0AFh | programmable | 0B0h - 0BFh | reserved |
| 0C0h - 0CFh | programmable | 0D0h - 0DFh | reserved |
| 0E0h - 0EFh | programmable | 0F0h - 0FFh | reserved |
| 100h - 10Fh | programmable | 110h - 11Fh | reserved |
| 120h - 12Fh | programmable | 130h - 13Fh | reserved |
| 140h - 14Fh | programmable | 150h - 15Fh | reserved |
| 160h - 16Fh | programmable | 170h - 17Fh | reserved |
| 180h - 18Fh | programmable | 190h - 19Fh | reserved |
| 1A0h - 1AFh | programmable | 1B0h - 1BFh | reserved |
| 1C0h - 1CFh | programmable | 1D0h - 1DFh | reserved |
| 1E0h - 1EFh | programmable | 1F0h - 1FFh | reserved |

### 7.8.1.3 Write Buffer (Page) Programming Command

Write Buffer Programming allows the system to write 1 to 256 words in one programming operation. The Write Buffer Programming command sequence is initiated by first writing the Write Buffer Load command to the CAP1 address. Next, the system writes the number of word locations minus 1 at the CAP2 address. This tells the device how many Write Buffer addresses are loaded with data and therefore when to expect the *Program Buffer to Flash* confirm command. The Sector Address must match during the Write Buffer Load command and during the Write Word Count command and the Sector must be unlocked or the operation will abort and return to the initiating state.

The Write Buffer Programming Command is used to program data within a 512 byte page aligned on a 512 byte boundary. Thus, a full page Write Buffer programming operation must be aligned on a page boundary. Programming operations of less than a full page may start on any word boundary but may not cross a page boundary. At the start of a Write Buffer programming operation all locations in the buffer are all ones (FFFFh words) thus any locations not loaded will not program any ones to zeros.

The system then writes the starting address/data combination. This starting address is the first address/data pair to be programmed, and selects the *write-buffer-page* address. The Sector address must match the Write Buffer Load Sector Address or the operation will abort and return to the initiating state. All subsequent address/data pairs must be in sequential order. All write buffer addresses must be within the same page. If the system attempts to load data outside this range, the operation will abort and return to the initiating state.

The counter decrements for each data load operation. Once the specified number of write buffer locations have been loaded, the system must then write the *Program Buffer to Flash* command at the CAP1 address. The device then goes busy as it begins the programming algorithm.

Until the *Program Buffer to Flash* command is issued, the bank where the write buffer operation will occur may be read. This allows an interrupt routine to have read access even while the write buffer is being loaded in order to minimize interrupt latency. After the *Program Buffer to Flash* command is issued, an interrupt routine must suspend the program operation before gaining read access to the same bank.

The Embedded Program algorithm automatically programs and verifies the data for the correct data pattern. The system is not required to provide any controls or timings during these operations. If the incorrect number of write buffer locations have been loaded and the *Program Buffer to Flash* command is issued, the operation will abort and return to the initiating state.

The Write Buffer embedded programming operation can be suspended using the Program Suspend command. When the Embedded Program algorithm is complete, the device then returns to Erase Suspend, SSR Lock, Secure Silicon Region, or Idle state. The system can determine the status of the program operation by reading the Status Register. *Status Register* *on page 27* for information on these status bits.

The Write Buffer Programming Sequence will be Aborted under the following conditions:

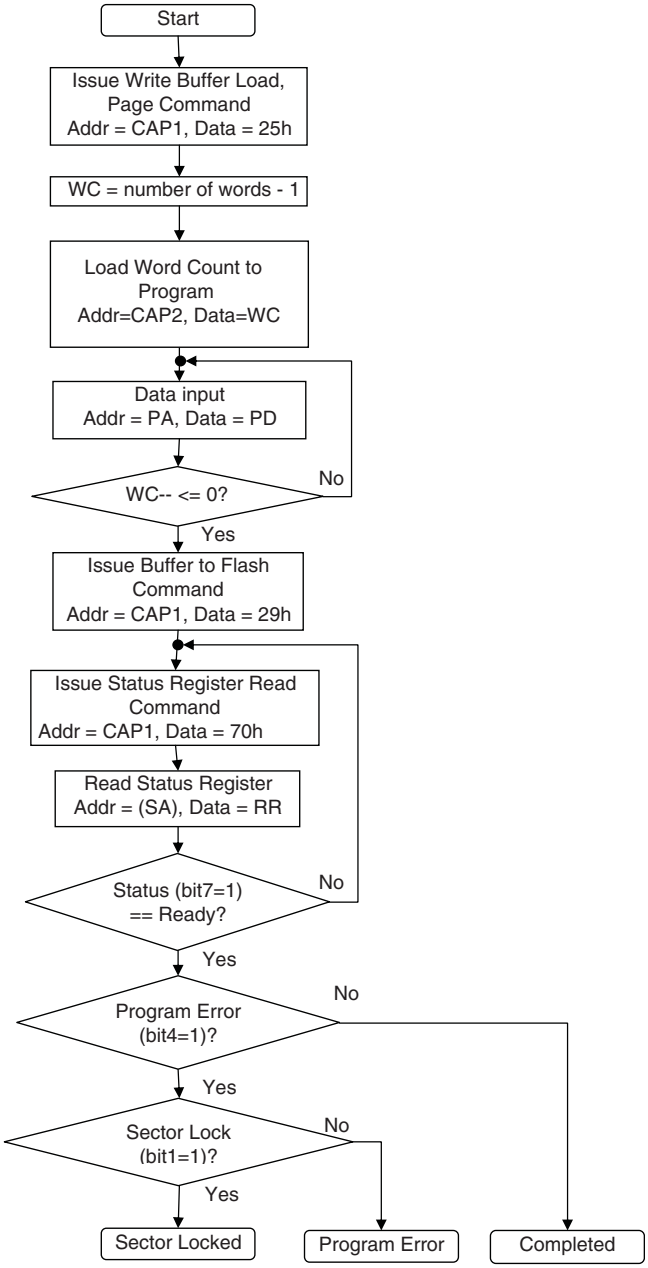
- Load a Word Count value that would cause the write buffer load sequence to attempt loading words past the end of the aligned 512 byte page. Note that only the lower eight bits of the Word Count data value are captured so no value > 255 can be recognized.
- Write an address that is outside the Page of the Starting Address during the write buffer data loading stage of the operation. Or, have a sector address not matching the one used in the initial command cycle of the page programming operation.
- The sector to be programmed is locked

When any of the conditions that cause an abort of write buffer command occur the abort will happen after the Program Buffer to Flash command is issued, and will indicate a Program Fail in the Status Register at bit location 4 (PSB). The next successful program operation will clear the failure status or a “Clear Status Register” may be issued to clear the PSB status bit.

The Write Buffer Programming Sequence can be stopped and reset by the following: Hardware Reset or Power cycle. However, using either of these methods may leave the area being programmed in an intermediate state with invalid or unstable data values. In this case the same area will need to be reprogrammed with the same data or erased to ensure data values are properly programmed or erased.

Page Programming should be done only once in any particular page before the sector in which the page resides is erased. Page Programming more than once in the same page between erase operations will not cause the operation to fail but may degrade the reliability of data in any Cache Line where data is changed, until the next erase of the related sector.

**Figure 7.4** Write Buffer Program Flowchart



### Software Functions and Sample Code

**Table 7.15** Page Program

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Buffer Load Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0025h |
| 2 | Write Word Count to CAP2 | Write | Sector Address+1554h | Sector Address+AAAh | Word Count (N-1) |
| 3 to 258 | **Number of words (N) loaded into the write buffer can be from 1 to 256** | | | | |
| Last | Write Buffer to flash Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0029h |

***Notes:***

1. *Last = Last cycle of write buffer program operation; depending on number of words written, the total number of cycles may be from 4 to 259.*
2. *For maximum efficiency, it is recommended that the write buffer be loaded with the highest number of words (N words) possible.*
3. *LLD Functions Used = lld_WriteToBufferCmd, lld_ProgramBufferToFlashCmd.*

The following is a C source code example of using the write buffer program function. Refer to the *Spansion Low Level Driver User's Guide* (available on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

```
/* Example: Write Buffer Programming Command     */
/* NOTES: page programming limited to 256 words. */
/* All addresses to be written to the flash in   */
/* one operation must be within the same flash   */
/* page. A flash page begins at addresses        */
/* evenly divisible by 0x200.                    */

UINT16 *src = source_of_data;      /* address of source data */
UINT16 *dst = destination_of_data; /* flash destination address */
UINT16 wc = words_to_program;      /* word count */
UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */
UINT16 *CAP2 = ((UINT16 *) sector_address + 0xAAA); /* Define CAP2 */

*CAP1 = 0x0025; /* write write buffer load command */
*CAP2 = wc - 1; /* write word count (minus 1)      */

do                 /* write all source data to destination */
{
    *dst = *src;   /* ALL dst MUST BE SAME PAGE      */
    dst++;         /* increment destination pointer  */
    src++;         /* increment source pointer       */
    wc--;          /* decrement word count           */
} while (wc != 0); /* loop until all words are moved */

*CAP1 = 0x0029; /* write confirm command */
/* poll for completion */
```

#### 7.8.1.4 Bit-Field Programming Command

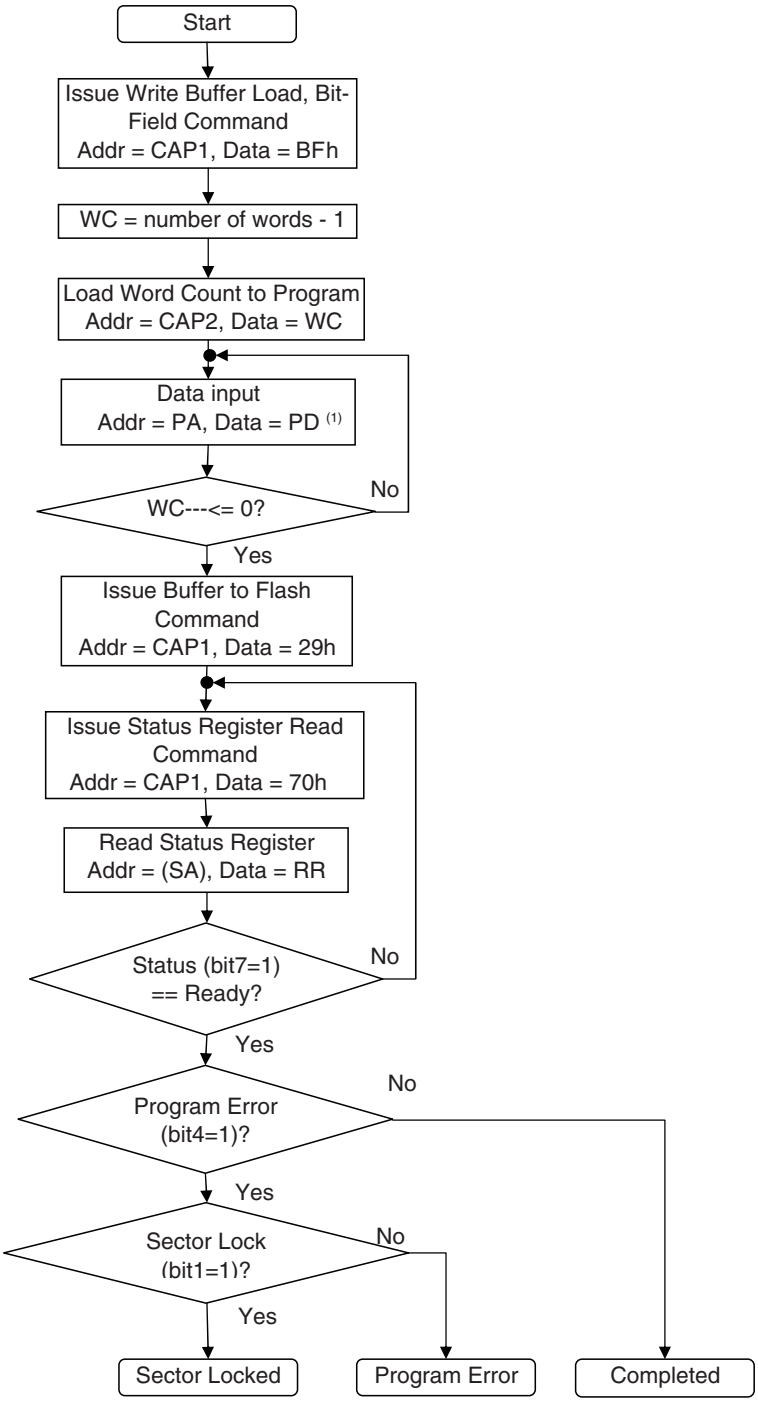
The Bit-Field Programming command follows the same format as Page programming but is limited to programming only locations in the write buffer with word address bit A3=0 (system byte address bit a4=0).

Locations with A3=1 are reserved and must be programmed only with FFFFh. Programming an A3=1 location with a bit pattern other than FFFFh will result in the program operation failing.

Only Zeros in the data are programmed. Ones are ignored. Programming within a CL that has been previously programmed will result in the logical AND of the prior CL data and the data being programmed. Bit-field programming may be done in a page that was previously programmed by the page programming command and will convert that page to a bit-field programmed page in which future bit-field programming operations are allowed. However, if the earlier page programming command caused bits in the reserved area to be programmed to zero, the bit-field programming command may fail. The reserved area is programmed in a special way by the bit-field programming command. If the reserved area has already had bits set to zero in

a manner that is incompatible with the pattern required by bit-field programming, the bit-field programming operation will fail.

**Figure 7.5** Bit Field Program Flowchart




***Note***

*1. For all address locations where A3=1, PD must be FFFFh*

### Software Functions and Sample Code

**Table 7.16** Bit Field Program

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Buffer Load Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 00BFh |
| 2 | Write Word Count to CAP2 | Write | Sector Address+1554h | Sector Address+AAAh | Word Count (N-1) |
| 3 to 258 | **Number of words (N) loaded into the write buffer can be from 1 to 256** | | | | |
| Last | Write Buffer to flash Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0029h |

***Notes***

1. *Last = Last cycle of write buffer program operation; depending on number of words written, the total number of cycles may be from 4 to 259.*
2. *For maximum efficiency, it is recommended that the write buffer be loaded with the highest number of words (N words) possible.*
3. *For all address locations where A3=1, PD must be FFFFh*

```
/* Example: Bit-Field Programming Command             */
/* NOTES: Bit-Field programming limited to 128 words. */
/* All addresses to be written to the flash in        */
/* one operation must be within the same flash        */
/* page. A flash page begins at addresses             */
/* evenly divisible by 0x200.                         */
/* The destination address must have A4=0.            */

UINT16 *src = source_of_data;      /* address of source data */
UINT16 *dst = destination_of_data; /* flash destination address */
UINT16 wc = words_to_program;      /* word count */
UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */
UINT16 *CAP2 = ((UINT16 *) sector_address + 0xAAA); /* Define CAP2 */

*CAP1 = 0x00BF; /* write bit-field load command */
*CAP2 = wc - 1; /* write word count (minus 1)   */

do /* write all source data to destination */
{
*dst = *src; /* ALL dst MUST BE SAME PAGE */
dst++; /* increment destination pointer */
src++; /* increment source pointer */
wc--; /* decrement word count */
} while (wc != 0); /* loop until all words are moved */
```

**Note:** The example above assumes that the data to be programmed using the Bit-Field Programming command is stored contiguously in source_of_data. If the data in source_of_data is stored with the reserved area taken into account. (16 bytes of data followed by 16 bytes of FF) Then the code example in the Page Programming command must be used with the 0x0025 (write buffer load command) rather than with 0x00BF.

**Note:** The source data buffer must contain FFs in locations corresponding to the reserved areas, and the word count (wc) should include the total number of words including the reserved areas.

## 7.8.2 Program Suspend/Program Resume Commands

The Program Suspend command allows the system to interrupt an embedded programming operation so that data can be read from any non-suspended sector. When the Program Suspend command is written during a programming process, the device halts the programming operation within $t_{PSL}$ (program suspend latency) and updates the status bits. Addresses are don't-cares when writing the Program Suspend command.

The system should read the Status Register to determine if the device is still actively programming or is program-suspended. Refer to *Status Register* *on page 27* for information on these status bits.

After the programming operation has been suspended, the system can read array data from any non-suspended sector (or from any bits not being programmed, in the case of programming). The Program

Suspend command may also be issued during a programming operation while an erase is suspended. In this case, data may be read from any sector not in Erase Suspend or bits not in Program Suspend Accesses and commands that are valid during Program Suspend are:

- Read of any other non-suspended bits
- Status Read command
- Program Resume command

After the Program Resume command is written, the device reverts to programming and the status bits are updated. The system can determine the status of the program operation by reading the Status Register, just as in the standard program operation. See *Status Register* *on page 27* for more information.

The system must write the Program Resume command to exit the Program Suspend mode and continue the programming operation. Further writes of the Program Resume command are ignored. Another Program Suspend command can be written after the device has resumed programming. Program operations may be interrupted as often as necessary but in order for a program operation to progress to completion there must be some periods of time between resume and the next suspend command greater than or equal to $t_{PRS}$ in Section 10.7.5, *Erase and Programming Performance on page 70*.

### Software Functions and Sample Code

**Table 7.17** Program Suspend
(LLD Function = lld_ProgramSuspendCmd)

| Cycle | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|
| 1 | Write Program Suspend Command | XXXX | XXXX | 0051h |

***Note***
*XXXX indicates a don't care address.*

The following is a C source code example of using the program suspend function. Refer to the *Spansion Low Level Driver User's Guide* (available on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

```
/* Example: Program Suspend Command */
*( (UINT16 *) base_address ) = 0x0051;
```

**Table 7.18** Program Resume
(LLD Function = lld_ProgramResumeCmd)

| Cycle | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|
| 1 | Write Program Resume Command | XXXX | XXXX | 0050h |

***Note***
*XXXX indicates a don't care address.*

The following is a C source code example of using the program resume function. Refer to the *Spansion Low Level Driver User's Guide* (available on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

```
/* Example: Program Resume Command */
*( (UINT16 *) base_address ) = 0x0050;
```

## 7.9 Erase Methods

### 7.9.1 Sector Erase

The sector erase function erases one sector in the memory array. (See 11.1, *Command Definitions on page 71*) The device does not require the system to preprogram prior to erase. The Embedded Erase algorithm automatically programs and verifies the selected sector with an all zero data pattern prior to electrical erase. After a successful sector erase, all locations within the erased sector contain FFFFh. The system is not required to provide any controls or timings during these operations.

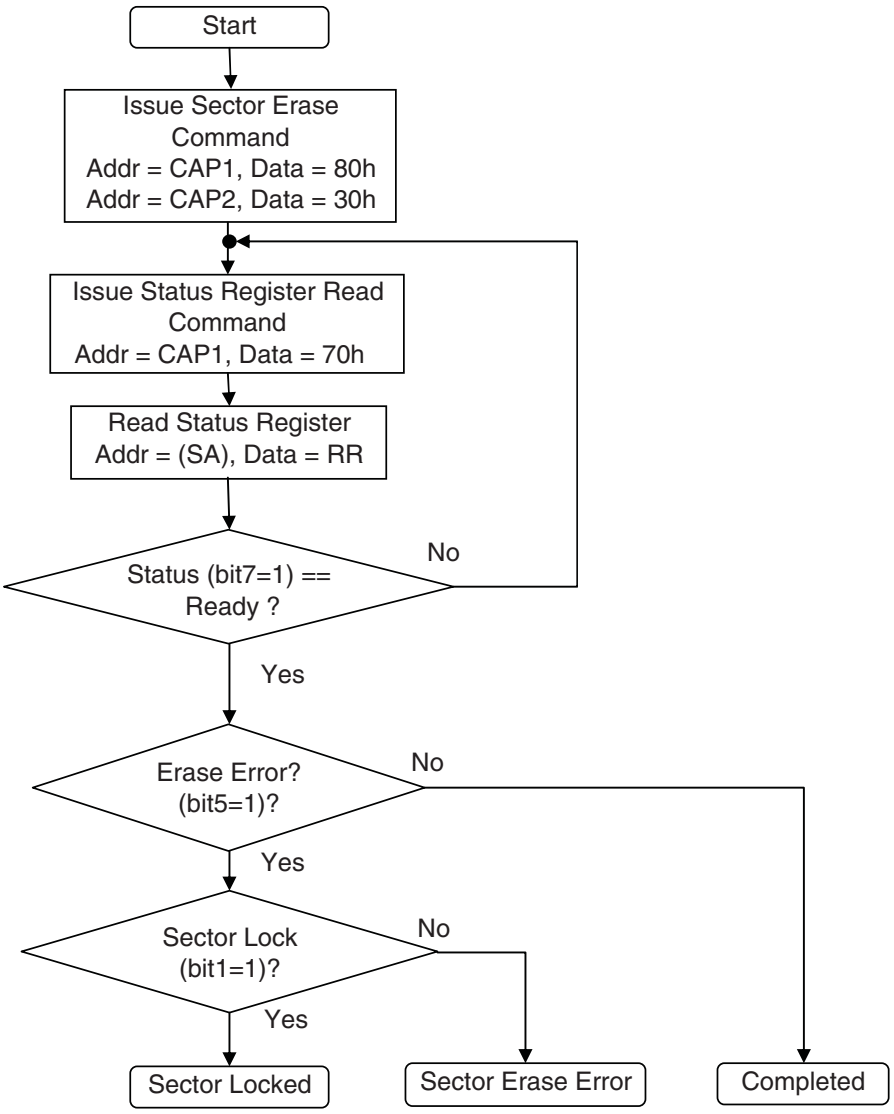
Sector Erase may require two commands; a command to unprotect the sector followed by the sector erase command itself. The desired sector must already be unlocked or must be unlocked by executing the Sector Unlock command, and the sector must not be locked by the Sector Lock Range command. In the erase command write sequence each of the Sector Addresses must match and the lower portion of each address with the address bit patterns must be correct.

When the Embedded Erase algorithm is complete, the bank returns to idle. Note that while the Embedded Erase operation is in progress, the system can read data from the non-erasing banks. The system can determine the status of the erase operation by reading the Status Register. See *Status Register on page 27* for information on these status bits.

Once the sector erase operation has begun, only reading from outside the erase bank, read of Status Register, and the Erase Suspend command are valid. All other commands are ignored. However, note that a hardware reset immediately terminates the erase operation. If that occurs, the sector erase command sequence must be reinitiated once the device has returned to idle state, to ensure data integrity.

See *Program/Erase Operations on page 32* for parameters and timing diagrams.

**Figure 7.6** Sector Erase Flowchart




## Software Functions and Sample Code

**Table 7.19** Sector Erase
(LLD Function = lld_SectorEraseCmd)

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Erase Setup Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0080h |
| 2 | Write Sector Erase Command to CAP2 | Write | Sector Address+1554h | Sector Address+AAAh | 0030h |

The following is a C source code example of using the sector erase function. Refer to the *Spansion Low Level Driver User's Guide* (available on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

```
/* Example: Sector Erase Command */
UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */
UINT16 *CAP2 = ((UINT16 *) sector_address + 0xAAA); /* Define CAP2 */

*CAP1 = 0x0080; /* write the setup command          */
*CAP2 = 0x0030; /* write the sector erase command   */

/* poll for completion */
```

### 7.9.2 Erase Suspend/Erase Resume Commands

The Erase Suspend command allows the system to interrupt a sector erase operation and then read data from, or program data to, the device. This command is valid only during the sector erase operation. The Erase Suspend command is ignored if written during the chip erase operation.

When the Erase Suspend command is written during the sector erase operation, the device requires a maximum of tESL (erase suspend latency) to suspend the erase operation and update the status bits to indicate the device is ready and in the suspend mode. The system should read the Status Register to determine if the device is still actively erasing or is erase-suspended. Refer to *Status Register* *on page 27* for information on these status bits.

After the erase operation has been suspended, the bank is in the erase-suspend mode. The system can read data from or program data to the device. Reading at any address within erase-suspended sectors produces undetermined data.

After an erase-suspended program operation is complete, the bank returns to the erase-suspend mode. The system can determine the status of the program operation by reading the Status Register, just as in the standard program operation.

Accesses and commands that are valid during Erase Suspend are:

- Read to any other non-suspended sector
- Program to any other non-suspended sector
- Status Read command
- Sector Lock/Unlock command
- Erase Resume command

To resume the sector erase operation, the system must write the Erase Resume command. The device will revert to erasing and the status bits will be updated. Further writes of the Resume command are ignored. Another Erase Suspend command can be written after the chip has resumed erasing. Erase operations may be interrupted as often as necessary but in order for an erase operation to progress to completion there must be some periods of time between resume and the next suspend command greater than or equal to $t_{ERS}$ in Section 10.7.5, *Erase and Programming Performance on page 70*.

### Software Functions and Sample Code

**Table 7.20** Erase Suspend
(LLD Function = lld_EraseSuspendCmd)

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Erase suspend Command | Write | XXXX | XXXX | 00B0h |

***Note***
*XXXX indicates a don't care address.*

The following is a C source code example of using the erase suspend function. Refer to the *Spansion Low Level Driver User's Guide* (available on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

```
/* Example: Erase Suspend Command */
*( (UINT16 *) base_address ) = 0x00B0
```

**Table 7.21** Erase Resume
(LLD Function = lld_EraseResumeCmd)

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Erase Resume Command | Write | XXXX | XXXX | 0030h |

***Note***
*XXXX indicates a don't care address.*

The following is a C source code example of using the erase resume function. Refer to the *Spansion Low Level Driver User's Guide* (available on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

```
/* Example: Erase Resume Command */
*( (UINT16 *) base_address ) = 0x0030;
```

## 7.10 Handshaking

The handshaking feature allows the host system to detect when data is ready to be read by simply monitoring the RDY (Ready) pin, which is a dedicated output controlled by CE#.

When either CE# input is Low, the RDY output signal is actively driven. When both of the CE# inputs are High the RDY output is high-impedance. When either CE# input and OE# input is Low, the A/DQ15-A/DQ0 output signals are actively driven. When both of the CE# inputs are High, or the OE# input is High, the A/DQ15-A/DQ0 outputs are high-impedance.

When the device is operated in synchronous mode, and OE# is low (active), the initial word of burst data becomes available after the rising edge of the RDY. CR0.8 in the Configuration Register allows the host to specify whether RDY is active at the same time that data is ready, or one cycle before data is ready (see Table 7.4, *Configuration Register 0 (CR0) on page 26*).

When the device is operated in asynchronous mode, RDY will be high when CE# is low (active).

## 7.11 Power On (Cold) Reset (POR)

When power is first applied, with supply voltage below 1V, then rising to reach operating range minimum, internal device configuration and warm reset activities are initiated. The device must not be accessed (CE# to go High within $t_{VCE}$ and remain High) for the duration of the POR operation ($t_{VCS}$). RESET# Low during this period is optional. If RESET# is driven Low during POR it must satisfy the Hardware Reset parameters $t_{RP}$ and $t_{VRPH}$. In which case the Reset operations will be completed at the later of $t_{VCS}$ or $t_{VRPH}$. During Reset operations the device will draw $I_{CC3}$ current.

At the end of POR the device conditions are:

- all internal configuration information is loaded
- the device is in read mode
- the Configuration Registers are at default values
- the Status Register is at default value
- the Sector Unlock Register is cleared
- the Sector Lock Range registers are cleared
- no sectors protected mode - however, ACC protection may be in effect
- the Write Buffer is loaded with all ones
- the AADM upper address register is cleared to zero
- the internal Control Unit is in the idle state

## 7.12 Hardware (Warm) Reset

The RESET# input provides a hardware method of resetting the device to idle state. When RESET# is driven low for at least a period of $t_{RP}$, the device immediately:

- terminates any operation in progress,
- exits any ASO,
- tristates all outputs,
- resets the Configuration Registers,
- resets the Status Register,
- clears the Sector Unlock Register,
- clears the Sector Lock Range registers,
- sets the no sectors protected mode; however, ACC protection may be in effect
- loads the Write Buffer with all ones,
- reloads all internal configuration information necessary to bring all banks in the device to Read mode,
- and resets the internal Control Unit to idle state.

The device must not be accessed (CE# to remain High) for the duration of the reset operation ($t_{RPH}$).

To ensure data integrity any operation that was interrupted should be reinitiated once the device is ready to accept another command sequence.

When RESET# is first asserted Low, the device draws $I_{CC4}$ during $t_{RPH}$. If RESET# continues to be held at $V_{SS}$ the device will draw CMOS standby current ($I_{CC3}$).

See Figure 10.7 for timing diagrams

## 7.13 Software (Command) Reset

Software reset is part of the command set (see 11.1, *Command Definitions on page 71*) that also returns the device to idle state and must be used for the following conditions:

1. Exit ID/CFI mode
2. Exit Secure Silicon Region mode
3. Exit Configuration Register mode
4. Exit SSR Lock mode

Software Reset does not affect EA mode. Reset commands are ignored once programming or erasure has begun, until the operation is complete. Software Reset does not affect outputs, or register values; it serves primarily to return to Read mode from ASO mode.

Software Reset may cause a return to Read mode from undefined states that might result from invalid command sequences. However, a Hardware Reset may be required to return to normal operation from some undefined states.

There is no software reset latency requirement. The reset command is executed during the $t_{WPH}$ period.

Reset commands are ignored once programming/erasure has begun until the operation is complete.

### Software Functions and Sample Code

**Table 7.22** Reset
(LLD Function = lld_ResetCmd)

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| Reset Command | Write Software Reset Command | Write | XXXX | XXXX | 00F0h |

***Note:***
*XXXX indicates a don't care address.*

The following is a C source code example of using the reset function. Refer to the *Spansion Low Level Driver User's Guide* (available on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

```
/* Example: Software Reset Command */
/* Return to read array mode from ASO mode */ *( (UINT16 *) base_address ) = 0x00F0;
```

# 8. Data Protection

The fundamental job of a non-volatile memory is to preserve the information (data) stored in the memory array. In particular, to preserve the information during long periods when no power is applied to the memory.

The original non-volatile memory type was Read Only Memory (ROM) which preserved information permanently by hard-wired connections of the memory array transistors. ROM provides great permanent data storage but, no ability to make any changes to the data that might be desired at a later time.

Non-volatile memories evolved to add various methods for modifying data when needed. Flash memory has become the most commonly used form of non-volatile memory because of its high density, high speed, and electrical re-programmability that allows data to be easily updated within the host system.

However, the ability to easily change the stored data can be a problem if the data can be unintentionally changed by power or signal anomalies (glitches), incorrect code execution, or even maliciously changed by intentionally destructive code.

Data Protection features are provided in the Eclipse Flash family to either permanently or temporarily disable programming or erase operations in selected portions of the memory. These data protection features enable multiple levels of control over when data can be changed. This allows a more flexible balance between the goal of preserving data and the ability to modify the data when desired.

This section describes the various methods of protecting data stored in the memory array. Some protection features are implemented through hardware methods and some are under the control of software. They are discussed in the order of protection precedence (highest to lowest priority).

The general principle used is that, in normal operation, data should be protected by default:

- Data in each Flash memory array sector is protected unless protection is explicitly removed
- Some sectors can be configured to not allow protection to be removed
- For those sectors where protection is allowed to be removed, protection can only be removed from one sector at a time
- Once protection is removed from a sector, multiple program and erase operations may be performed in the sector, until protection is restored for that sector

## 8.1 Secure Silicon Region

The Secure Silicon Region (SSR) provides an extra Flash memory region that may be programmed once and permanently protected from further programming or erase. The SSR is made visible via an Address Space Overlay (ASO) command. The protection of SSR is controlled by the SSR Lock ASO.

- Reads can be performed in the Asynchronous or Synchronous mode.
- Sector address supplied during the Secure Silicon Entry command selects the Flash memory array sector that is overlaid by the Secure Silicon Region address map.
- Continuous burst mode reads within Secure Silicon Region wrap from byte address 3FFh back to address 000h.
- Reads outside of the overlaid sector return memory array data.
- The Secure Silicon Region can't be entered when the device is executing an Embedded Algorithm (nor during Program Suspend, Erase Suspend, or while another AOS is active).
- See the Secure Silicon address map for address range of this area.
- Both Write Buffer Programming and Bit-Field programming may be used in the SSR.

### 8.1.1 Factory Secure Silicon Region

The Factory Secure Silicon Region is always protected when shipped from the factory and has the Factory SSR Lock Bit (bit 1) permanently set to a *Zero*. This prevents cloning of a factory locked part and ensures the security of the ESN and customer code once the product is shipped to the field.

### 8.1.2 Customer Secure Silicon Region

The Customer Secure Silicon Region is shipped unprotected, Customer SSR Lock Bit (bit 0) set to a *One*. allowing customers to utilize that sector in any manner they choose.

The Customer SSR can be read any number of times, but each portion of the page can be programmed only once and the region locked only once. Either the Page Program or the Bit-Field Program method may be used in the region. The Customer SSR Lock must be used with caution as once locked, there is no procedure available for unlocking the Customer SSR and none of the bits in the Customer SSR memory space can be modified in any way.

Once the Customer Secure Silicon Region area is protected, any further attempts to program in the area will fail with status indicating the area being programmed is protected.

### 8.1.3 Secure Silicon Region Entry and Exit Command Sequences

The system can access the Secure Silicon Region region by issuing the one-cycle Enter Secure Silicon Region Entry command sequence from the IDLE State. The device continues to have access to the Secure Silicon Region region until the system issues the Exit Secure Silicon Region command sequence, performs a Hardware RESET, or until power is removed from the device.

See Command Definition Table [Secure Silicon Region Command Table, Appendix 11.1, *Command Definitions on page 71*] for address and data requirements for both command sequences.

The Secure Silicon Region Entry Command allows the following commands to be executed

- Read customer and factory Secure Silicon Regions
- Program the customer Secure Silicon Region - program suspend is not allowed during SSR programming
- Read data out of all sectors not re-mapped to Secure Silicon Region
- Secure Silicon Region Exit

#### Software Functions and Sample Code

The following are C functions and source code examples of using the Secured Silicon Sector Entry, Program, and exit commands. Refer to the *Spansion Low Level Driver User's Guide* (available soon on www.spansion.com) for general information on Spansion Flash memory software development guidelines.

**Table 8.1** Secured Silicon Region Entry
(LLD Function = lld_SecSiSectorEntryCmd)

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Entry Cycle Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0088h |

***Note:***
*Base = Base Address.*

```
/* Example: Secured Silicon Region Entry Command */

UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */

*CAP1 = 0x0088; /* Write Secured Silicon Region Entry command */
```

**Table 8.2** Secured Silicon Region Program
(LLD Function = lld_WriteBufferProgramOp)

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Buffer Load Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0025h |
| 2 | Write Word Count to CAP2 | Write | Sector Address+1554h | Sector Address+AAAh | Word Count (N-1) |
| 3 to 258 | **Number of words (N) loaded into the write buffer can be from 1 to 256 (512 bytes)** | | | | |
| Last | Write Buffer to flash Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0029h |

```
/* Example: Secured Silicon Region Program Command     */

UINT16 *src = source_of_data;      /* address of source data */
UINT16 *dst = destination_of_data; /* flash destination address */
UINT16 wc = words_to_program;      /* word count */
UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */
UINT16 *CAP2 = ((UINT16 *) sector_address + 0xAAA); /* Define CAP2 */

*CAP1 = 0x0025; /* write write buffer load command */
*CAP2 = wc - 1; /* write word count (minus 1)      */

do                 /* write all source data to destination */
{
    *dst = *src;   /* ALL dst MUST BE SAME PAGE      */
    dst++;         /* increment destination pointer  */
    src++;         /* increment source pointer       */
    wc--;          /* decrement word count           */
} while (wc != 0); /* loop until all words are moved */

*CAP1 = 0x0029; /* write confirm command */
/* poll for completion */
```

**Table 8.3** Secured Silicon Region Exit
(LLD Function = lld_SecSiSectorExitCmd)

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write Secured Silicon Region Exit Command | Write | XXXX | XXXX | 00F0h |

***Note***
*XXXX indicates a don't care address.*

```
/* Example: Secured Silicon Region Exit Command */ *( (UINT16 *) base_address ) = 0x00F0;
```

## 8.2 Hardware Data Protection Methods

### 8.2.1 Power-Up Write Inhibit

If RESET# = $V_{IL}$ during power up, the device does not accept write commands. The Control Unit is automatically reset to Read mode during power-up or a Hardware Reset.

### 8.2.2 ACC Method

Depending on the device model selected, when the ACC input is at $V_{IL}$, one of the following occurs:

only the lowest two small sectors are protected in a top boot device;

or, only the lowest 128 KByte sector is protected in a uniform sector device.

ACC at $V_{IL}$ takes precedence over the POR Unlocked mode that is in effect after POR.

ACC does not prevent programming (writing) of the configuration register, Secure Silicon Region, or SSR Lock.

When ACC is at $V_{IH}$ there is no effect on sector protection.

### 8.2.3 POR Unlocked Mode

Following Power On Reset (POR) a special mode is made active that removes protection from all Flash memory array sectors that are not protected by ACC being at $V_{IL}$. This mode does not affect the protection of the Secure Silicon Region. This mode is intended to simplify programming of multiple sectors during system production in a factory environment. This mode is made inactive, leaving all sectors protected by default, when either the Sector Lock Range or Sector Lock/Unlock command is issued. Generally these commands are issued by boot code so that the POR Unlock mode is inactive during normal system operation, thus leaving all sectors protected by default.

## 8.3 Sector Lock Range (SLR) Command

This command allows a range of sectors to be protected from program or erase (locked) until a hardware reset or power is removed from the device. This command causes the Sector Lock/Unlock command to be ignored for the range of sectors.

This command is generally used by trusted boot code. After power on reset, boot code has the option to check for any need to update sectors before locking them for the remainder of power on time. Once boot code is satisfied with the content of sectors to be protected the Sector Lock Range command is used to lock selected sectors against any program or erase during normal system operation. This adds an extra layer of protection for critical data that must be protected against accidental or malicious corruption. Yet, maintains flexibility for trusted boot code to perform occasional updates of the data. It is important to issue the Sector Lock Range command even if no sectors are to be protected so that sectors that should remain available for update cannot be later locked by accidental or malicious code behavior.

Two cycles are first written: addresses are CAP1 and CAP2 with data 60h. During the third cycle, the SLR Low Address (SLA) and load sector address command (61h) is written. This cycle sets the lower sector address of the range. During the fourth cycle, a sector address (SA) and load sector address command (61h) is written. This cycle sets the upper sector address of the range. The addresses reference a large sector address range (128 KBytes). The sectors selected by the lower and upper address, as well as all sectors between these sectors, are protected from program and erase until a hardware reset or power is removed. If the lower and upper sector addresses are for the same sector then only that one sector is locked. If the first sector address cycle contains an address which is higher than the second sector address cycle, then the command sequence will be invalid and ignored.

If the address range includes an area where four small sectors are located all four sectors are protected. If the small sector area is not selected then the least significant 4 bits of the SLA control the lock status of the four small sectors. System address a1 (Flash A0) controls the highest small sector for top boot. System address a2 (Flash A1) controls the next higher or lower sector respectively and so on.

When one of the address inputs is a One (at $V_{IH}$) the related small sector is protected (locked), when it is a Zero the related small sector is unprotected. These address inputs are ignored in a device with uniform large sectors.

Flash address input A6 (system byte address bit a7) during both address cycles must be Zero (A6 = $V_{IL}$) for both of the large sector addresses to be accepted as valid. If A6 is One during both address cycles the SLR command is valid but, neither large sector address is valid so, no large sector is locked by the SLR command. If A6 is not Zero or One in both address cycles the command is invalid and ignored.

In either of the valid command cases - with A6 at Zero or One on both write cycles - the Flash addresses A3 to A0 are used to set the protection on the 4 small sectors as described earlier.

A valid Sector Lock Range command sequence is accepted only once after a Hardware Reset or initial power up. Additional Sector Lock Range commands will be ignored.

When the device is first powered up, all sectors are unlocked. Writing the SLR command will make the POR Unlock mode inactive which causes all sectors to be protected by default since no sector is explicitly unlocked.

**Table 8.4** Sector Lock Range Command

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write First Setup Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0060h |
| 2 | Write Second Setup Command to CAP2 | Write | Sector Address+1554h | Sector Address+AAAh | 0060h |
| 3 | Write lowest sector address | Write | Low Sector Address | Low Sector Address | 0061h |
| 4 | Write highest sector address | Write | High Sector Address | High Sector Address | 0061h |

***Note***
*To lock large sectors word address bit A6, byte address bit A7, must be 0. If word address bit A6, byte address bit A7 is one on both cycle 3 and 4 then no sectors are locked and no further Sector Lock Range commands are accepted. Bits A0-A3 control locking the small sectors as described earlier.*

### Software Function and Sample Code

```
/* Example: Sector Lock Range Command */
/* Lock a sector range  and not the small sectors */

UINT16 *CAP1 = ((UINT16 *) low_sector_address + 0x555); /* Define CAP1 */
UINT16 *CAP2 = ((UINT16 *) low_sector_address + 0xAAA); /* Define CAP2 */

*CAP1 = 0x0060;
*CAP2 = 0x0060;
*( (UINT16 *)(low_sector_address & 0xFFFFFF80h) ) = 61;
*( (UINT16 *)(high_sector_address & 0xFFFFFF80h) ) = 61;
```

## 8.4 Sector Lock/Unlock Command

The Sector Lock/Unlock command sequence allows the system to protect all sectors from accidental writes or, unprotect one sector to allow programming or erasing of the sector.

When the device is first powered up, all sectors are unlocked by the POR Unlock Mode. To lock all sectors (enter protected mode), a Sector Lock/Unlock command may be issued to any Sector Address with A6 = $V_{IL}$. Once this command is issued, only one sector at a time can be unlocked until power is cycled.

To unlock one sector, the system must write the Sector Lock/Unlock command sequence. Two cycles are first written: addresses are CAP1 and CAP2 with data 60h. During the third cycle, the sector address (SA) and unlock command (60h) are written, with address A6 = $V_{IH}$. The A6 = $V_{IH}$ indicates that the SA is a valid address and the SA identifies the one sector in the device that is unlocked. To again lock the selected sector, issue a Sector Lock/Unlock command to any Sector Address with A6 = $V_{IL}$. With A6 = $V_{IL}$ the SA is made invalid, thus the previously unlocked sector is locked. And, because the one sector unlock address is now invalid, by default all sectors are locked.

The SLR has higher priority than the Sector Lock/Unlock command. If a Sector Unlock command tries to unlock a Sector within the Sector Lock Range, the Sector will remain in locked state. Also, all sectors in the part will be locked. This is because attempting to unlock a sector in the SLR will not unlock the addressed sector and leaves no sector outside the SLR unlocked, thus all sectors remain locked by default.

Similarly, if a Sector that is currently unlocked by the Sector Unlock command is overlapped by a subsequent Sector Lock Range command, that sector will be locked and program or erase commands for that sector will be ignored. Also, all sectors in the part will be locked because no sector outside the SLR is unlocked, thus all sectors remain locked by default.

A Program or Erase operation will check the unlocked Sector Address only at the beginning of the Program or Erase operation. It is not necessary to keep the sector being Programmed or Erased unlocked during the operation. The system can change the unlocked Sector after programming or erasing the sector has begun. An Erase Resume or Program Resume command does not check whether the sector is still unlocked.

## Software Functions and Sample Code

**Table 8.5** Sector Lock Command

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write First Setup Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0060h |
| 2 | Write Second Setup Command to CAP2 | Write | Sector Address+1554h | Sector Address+AAAh | 0060h |
| 3 | Write Sector Lock Command | Write | (Sector Address) & FFFFFF7Fh | (Sector Address) & FFFFFFBFh | 0060h |

***Note***
& *indicates a bitwise AND operation.*

```
/* Example: Sector Lock Command */
UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */
UINT16 *CAP2 = ((UINT16 *) sector_address + 0xAAA); /* Define CAP2 */

*CAP1 = 0x0060;
*CAP2 = 0x0060;
*( (UINT16 *)(sector_address & 0xFFFFFFBF) ) = 60;
```

## Software Functions and Sample Code

**Table 8.6** Sector Unlock Command

| Cycle | Description | Operation | Byte Address | Word Address | Data |
|---|---|---|---|---|---|
| 1 | Write First Setup Command to CAP1 | Write | Sector Address+AAAh | Sector Address+555h | 0060h |
| 2 | Write Second Setup Command to CAP2 | Write | Sector Address+1554h | Sector Address+AAAh | 0060h |
| 3 | Write Sector Lock Command | Write | (Sector Address) \| 00000080h | (Sector Address) \| 00000040h | 0060h |

***Note***
| *indicates a bitwise OR operation.*

```
/* Example: Sector Unlock Command */
UINT16 *CAP1 = ((UINT16 *) sector_address + 0x555); /* Define CAP1 */
UINT16 *CAP2 = ((UINT16 *) sector_address + 0xAAA); /* Define CAP2 */

*CAP1 = 0x0060;
*CAP2 = 0x0060;
*( (UINT16 *)(sector_address | 0x00000040h) ) = 60;
```

# 9. Power Conservation Modes

## 9.1 Standby Mode

In the standby mode current consumption is greatly reduced, and the outputs (DQ15 - DQ0) are placed in the high impedance state, independent of the OE# input. The device enters the CMOS standby mode when the CE# and RESET# inputs are both held at $V_{CC} \pm 0.2$ V. The device requires standard access time ($t_{CE}$ or $t_{IA}$) for read access, before it is ready to read data. If the device is deselected during erasure or programming, the device draws active current until the operation is completed. $I_{CC3}$ in *DC Characteristics* *on page 56* is the standby current specification.

## 9.2 Automatic Sleep Mode

The automatic sleep mode minimizes Flash device energy consumption after read initial access time is completed and while the device is not performing a program or erase operation. The device automatically enables this mode when addresses remain stable for $t_{ACC}$ + 20 ns in asynchronous mode or after $t_{IA}$ of a synchronous burst read. The automatic sleep mode is independent of the CE#, WE#, and OE# control signals. Standard address access timings ($t_{ACC}$ or $t_{IA}$) provide new data when addresses are changed. While in asynchronous mode sleep, output data is latched and always available to the system. While in synchronous mode sleep, the burst data continues to be provided until the burst is terminated. $I_{CC6}$ in *DC Characteristics* *on page 56* is the automatic sleep mode current specification.

## 9.3 Output Disable (OE#)

When the OE# input is at $V_{IH}$, output (DQ15 - DQ0) from the device is disabled and placed in the high impedance state. RDY is not controlled by OE#.

# 10. Electrical Specifications

## 10.1 Absolute Maximum Ratings

| Parameter | Rating |
|---|---|
| Storage Temperature Plastic Packages | –65°C to +150°C |
| Ambient Temperature with Power Applied | –65°C to +125°C |
| Voltage with Respect to Ground: All Inputs and I/Os except as noted below (Note 1) | –0.5 V to $V_{CC}$ + 0.5 V |
| $V_{CC}$ | –0.5 V to +2.5 V |
| Output Short Circuit Current (Note 3) | 100 mA |

***Notes***

1. *Minimum DC voltage on input or I/Os is –0.5 V. During voltage transitions, inputs or I/Os may undershoot $V_{SS}$ to –2.0 V for periods of up to 20 ns. See Figure 10.1. Maximum DC voltage on input or I/Os is $V_{CC}$ + 0.5 V. During voltage transitions outputs may overshoot to $V_{CC}$ + 2.0 V for periods up to 20 ns. See Figure 10.2.*
2. *Minimum DC input voltage on pin ACC is -0.5V. During voltage transitions, ACC may overshoot $V_{SS}$ to –2.0 V for periods of up to 20 ns. See Figure 10.1. Maximum DC voltage on pin ACC is +9.5 V, which may overshoot to +10.5 V for periods up to 20 ns.*
3. *No more than one output may be shorted to ground at a time. Duration of the short circuit should not be greater than one second.*
4. *Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure of the device to absolute maximum rating conditions for extended periods may affect device reliability.*

**Figure 10.1** Maximum Negative Overshoot Waveform

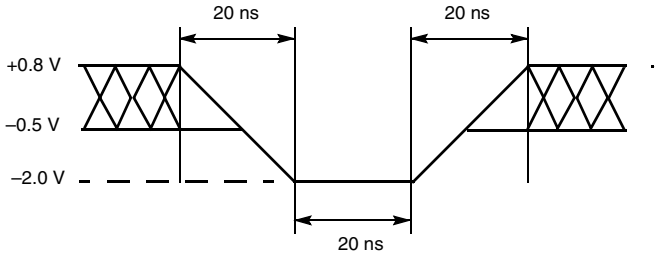
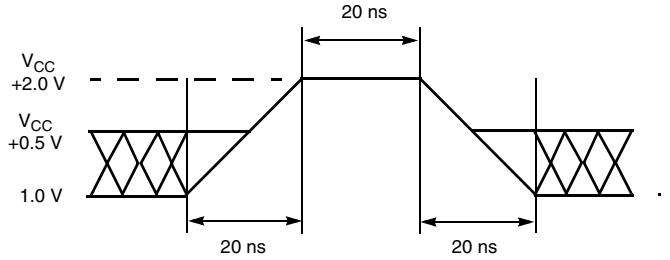



**Figure 10.2** Maximum Positive Overshoot Waveform




## 10.2 Operating Ranges

| Wireless (I) Devices | |
|---|---|
| Ambient Temperature ($T_A$) | –25°C to +85°C |
| **Supply Voltages** | |
| $V_{CC}$ Supply Voltages | +1.70V to +1.95V |

***Note***

1. *Operating ranges define those limits between which the functionality of the device is guaranteed.*

## 10.3 DC Characteristics

| Parameter | Description | Test Conditions (Notes 1 & 2) | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| $I_{LI}$ | Input Load Current | $V_{IN} = V_{SS}$ to $V_{CC}$, $V_{CC} = V_{CC}$max | | | | ±1 | µA |
| $I_{LO}$ | Output Leakage Current | $V_{OUT} = V_{SS}$ to $V_{CC}$, $V_{CC} = V_{CC}$max | | | | ±1 | µA |
| $I_{CCB}$ | $V_{CC}$ Active burst Read Current | CE# = $V_{IL}$, OE# = $V_{IH}$, WE# = $V_{IH}$, burst length = 8 | 66 MHz | | 33 | 38 | mA |
| | | | 83 MHz | | 38 | 44 | mA |
| | | | 104 MHz | | 42 | 47 | mA |
| | | | 108 MHz | | 42 | 47 | mA |
| | | CE# = $V_{IL}$, OE# = $V_{IH}$, WE# = $V_{IH}$, burst length = 16 | 66 MHz | | 26 | 30 | mA |
| | | | 83 MHz | | 30 | 34 | mA |
| | | | 104 MHz | | 34 | 39 | mA |
| | | | 108 MHz | | 34 | 39 | mA |
| | | CE# = $V_{IL}$, OE# = $V_{IH}$, WE# = $V_{IH}$, burst length = Continuous | 66 MHz | | 26 | 30 | mA |
| | | | 83 MHz | | 30 | 35 | mA |
| | | | 104 MHz | | 34 | 39 | mA |
| | | | 108 MHz | | 34 | 39 | mA |
| $I_{IO}$ | $V_{IO}$ Standby | CE# = RESET# = $V_{CC}$ ± 0.2V | | | 2 | 3 | µA |
| $I_{CC1}$ | $V_{CC}$ Active Asynchronous Read Current | CE# = $V_{IL}$, OE# = $V_{IH}$, WE# = $V_{IH}$ | 1 MHz | | 11 | 15 | mA |
| | | | 5 MHz | | 51 | 56 | mA |
| | | | 8 MHz | | 66 | 75 | mA |
| $I_{CC2}$ | $V_{CC}$ Active Write Current (Note 2) | CE# = $V_{IL}$, OE# = $V_{IH}$, ACC = $V_{IH}$ | | | 65 | 72 | mA |
| $I_{CC3}$ | $V_{CC}$ Standby Current | CE# = RESET# = $V_{CC}$ ± 0.1 V | | | 30 | 50 | µA |
| $I_{CC4}$ | $V_{CC}$ Reset Current | RESET# = $V_{IL}$, CLK = $V_{IL}$ | | | 15 | 20 | mA |
| $I_{CC5}$ | $V_{CC}$ Active Current (Read While Write) | CE# = $V_{IL}$, OE# = $V_{IH}$, ACC = $V_{IH}$, burst length = 8 | 66 MHz | | 95 | 110 | mA |
| | | | 83 MHz | | 103 | 116 | mA |
| | | | 104 MHz | | 107 | 119 | mA |
| | | | 108 MHz | | 107 | 119 | mA |
| $I_{CC6}$ | $V_{CC}$ Sleep Current (Note 3) | CE# = $V_{IL}$, OE# = $V_{IH}$ | | | 30 | 50 | µA |
| $I_{ACC}$ | Accelerated Program Current | CE# = $V_{IL}$, OE# = $V_{IH}$, $V_{ACC}$ = 9.5 V | | | 8 | 13 | mA |
| $V_{IL}$ | Input Low Voltage | $V_{IO}$ = 1.8 V | | –0.2 | | 0.4 | V |
| $V_{IH}$ | Input High Voltage | $V_{IO}$ = 1.8 V | | $V_{IO}$ – 0.4 | | $V_{IO}$ + 0.4 | |
| $V_{OL}$ | Output Low Voltage | $I_{OL}$ = 100 µA, $V_{CC} = V_{CC\ min} = V_{IO}$ | | | | 0.1 | V |
| $V_{OH}$ | Output High Voltage | $I_{OH}$ = –100 µA, $V_{CC} = V_{CC\ min} = V_{IO}$ | | $V_{IO}$ – 0.1 | | | V |

***Notes***

1. *Maximum $I_{CC}$ specifications are tested with $V_{CC} = V_{CC}$max.*
2. *$I_{CC}$ active while Embedded Erase or Embedded Program is in progress.*
3. *Device enters automatic sleep mode in asynchronous mode when addresses are stable for $t_{ACC}$ + 20 ns. Device enters automatic sleep mode in synchronous burst mode at $t_{IA}$ until the next cycle with AVD# Low if no embedded operation is active. Note automatic sleep mode current does not include I/O related switching current.*

## 10.4 Capacitance

| Symbol | Description | Test Condition | | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $C_{IN}$ | Input Capacitance (Address, CE#, OE#, WE#, AVD#, WE#, CLK, RESET#) | $V_{IN}$ = 0 | Single Die | 4.5 | 6.0 | pF |
| | | | Dual Die | 9.0 | 12.0 | pF |
| $C_{OUT}$ | Output Capacitance (DQ, RDY) | $V_{OUT}$ = 0 | Single Die | 6.5 | 7.5 | pF |
| | | | Dual Die | 13.0 | 15.0 | pF |

***Notes***

1. *Test conditions $T_A$ = 25°C, f = 1.0 MHz*
2. *Sampled, not 100% tested.*

## 10.5 AC Test Conditions

| Parameter | Limit | Condition | | Value | Units |
|---|---|---|---|---|---|
| Input level range | | | | 0.0 to $V_{CC}$ | V |
| Input or CLK Rise Measurement | | | | $V_{IL}$ to $V_{IH}$ | V |
| Input or CLK Fall Measurement | | | | $V_{IH}$ to $V_{IL}$ | V |
| Input comparison level | | | | $V_{IO}/2$ | V |
| Output data comparison level | | | | $V_{IO}/2$ | V |
| Load capacitance (CL) | Max | | | 30 | pF |
| | | $f_{CLK}$ | $t_{CLK}$ | | |
| Input Transition time ($t_T$) (input rise and fall times = 20% of $t_{CLK}$) | Max | 66 MHz | 15 | 3.00 | ns |
| | Max | 83 MHz | 11.9 | 2.50 | ns |
| | Max | 104 MHz | 9.6 | 1.85 | ns |
| | Max | 108 MHz | 7.5 | 1.50 | ns |
| CLK Transition time ($t_{CLKRF}$) (CLK input rise and fall times = 20% of $t_{CLK}$) | Max | 66 MHz | 15 | 3.00 | ns |
| | Max | 83 MHz | 11.9 | 2.50 | ns |
| | Max | 104 MHz | 9.6 | 1.85 | ns |
| | Max | 108 MHz | 9.26 | 1.50 | ns |
| CLK High/Low time ($t_{CL}/t_{CH}$) (($t_{CLK}/2$ - $t_{CLKRF}$) * 0.9) | Min | 66 MHz | 15 | 4.1 | ns |
| | Min | 83 MHz | 11.9 | 3.20 | ns |
| | Min | 104 MHz | 9.6 | 2.6 | ns |
| | Min | 108 MHz | 9.26 | 2.0 | ns |

**Figure 10.3** Input Pulse and Test Point

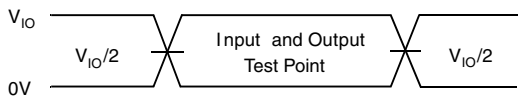



**Figure 10.4** CLK Parameters

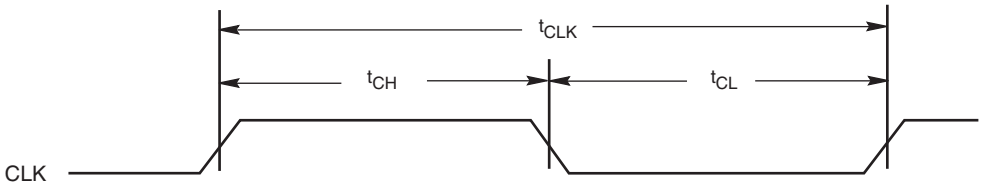
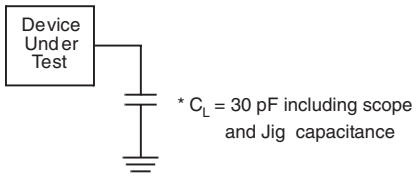



**Figure 10.5** Output Load




Output Load

## 10.6 Power On Reset (POR) and Warm Reset

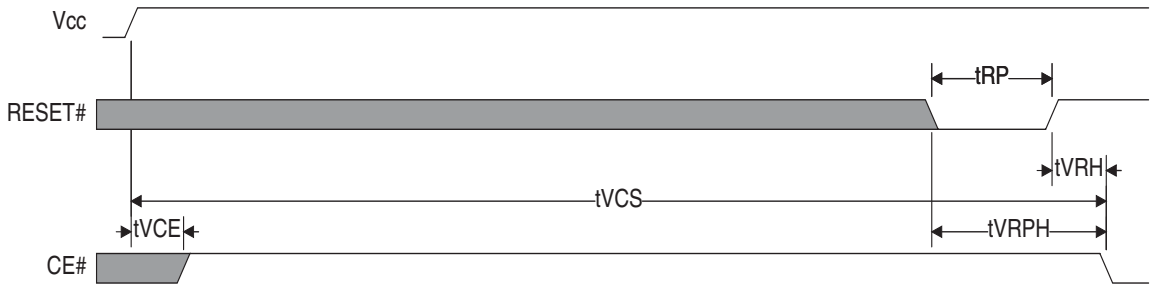
**Table 10.1** Power On and Reset Parameters

| Parameter | Description | Limit | Time | Unit |
|---|---|---|---|---|
| $t_{VCS}$ | $V_{CC}$ Setup Time to first access (1) | Min | 450 | µs |
| $t_{VCE}$ | CE# High after Vcc in operating range | Max | 25 | µs |
| $t_{VRPH}$ | RESET# Low to CE# Low | Min | 450 | µs |
| $t_{RP}$ | RESET# Pulse Width | Min | 200 | ns |
| $t_{VRH}$ | RESET# High to CE# Low | Min | 200 | ns |

***Notes***

1. *Timing measured from $V_{CC}$ reaching $V_{CC}$ minimum to $V_{IH}$ on Reset and $V_{IL}$ on CE#.*
2. *RESET# Low is optional during POR. If Reset# is asserted during POR, the later of $t_{VRPH}$ or $t_{VCS}$ will determine when CE# may go Low. Reset# must also be high $t_{RH}$ before CE# goes low.*
3. *$V_{CC}$ ramp rate can be non-linear but, $V_{CC}$ must reach $V_{CC}$ minimum within 25 mS from the start of $V_{CC}$ ramp and must not dip below a previous $V_{CC}$ value by more than 100 mV before reaching $V_{CC}$ minimum.*

**Figure 10.6** Power-up Diagram



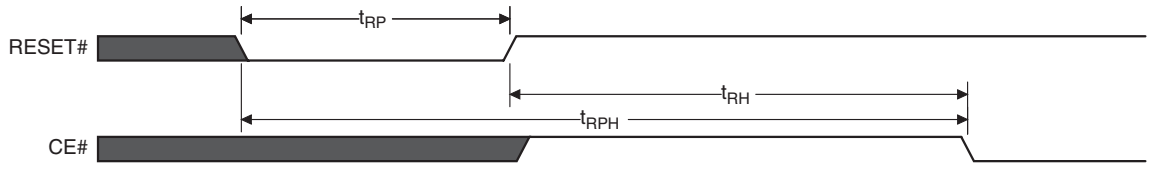

### 10.6.1 Hardware (Warm) Reset

Hardware Reset is initiated by the RESET# signal going to $V_{IL}$.

**Table 10.2** Hardware Reset Parameters

| Parameter | Description | Limit | Time | Unit |
|---|---|---|---|---|
| $t_{RPH}$ | RESET# Low to CE# Low | Min | 30 | µs |
| $t_{RP}$ | RESET# Pulse Width | Min | 200 | ns |
| $t_{RH}$ | RESET# High to CE# Low | Min | 200 | ns |

**Figure 10.7** Hardware Reset



## 10.7 AC Characteristics

### 10.7.1 AC Characteristics–Synchronous Burst Read

**Table 10.3** 1 Gbit

| Parameter (Notes) | Symbol | Limit | Value by Frequency (MHz) | | | | Unit |
|---|---|---|---|---|---|---|---|
| | | | 66 | 83 | 104 | 108 | |
| Clock Frequency | CLK | Min | 1 | | | | MHz |
| Clock Cycle | $t_{CLK}$ | Min | 15 | 12 | 9.6 | 9.26 | ns |
| Internal Access Time | $t_{IA}$ | Max | 115 | | | | ns |
| Burst Access Time Valid Clock to Output Delay | $t_{BACC}$ | Max | 11 | 9 | 7 | 6.75 | ns |
| AVD# Setup Time to CLK | $t_{AVDS}$ | Min | 5 | 4 | 3 | 3 | ns |
| AVD# Hold Time from CLK | $t_{AVDH}$ | Min | 2.5 | 2 | 2 | 2 | ns |
| Address Setup Time to CLK | $t_{ACS}$ | Min | 5 | 4 | 3 | 3 | ns |
| Address Hold Time from CLK when AVD# is Low | $t_{ACH}$ | Min | $t_{CLK}$ | | | | ns |
| Data Hold Time from Next Clock Cycle | $t_{BDH}$ | Min | 2.5 | | | 1.5 | ns |
| Output Enable to Data | $t_{OE}$ | Max | 15 | | | | ns |
| CE# low to RDY valid | $t_{CR}$ | Max | 15 | | | | ns |
| CE# Disable to Output High Z (2) | $t_{CEZ}$ | Max | 15 | | | | ns |
| OE# Disable to Output High Z (2) | $t_{OEZ}$ | Max | 7 | | | | ns |
| CE#, OE#, WE# Setup Time to CLK | $t_{CES}$ | Min | 5 | 4 | 3 | 3 | ns |
| CLK to RDY valid | $t_{RACC}$ | Max | 11 | 9 | 7 | 6.75 | ns |
| AVD# Pulse Width | $t_{AVDP}$ | Min | $t_{CLK}$ | | | | ns |

***Notes***

1. *Not 100% tested.*
2. *If OE# is disabled before CE# is disabled, the output goes to High-Z by $t_{OEZ}$.*
   *If CE# is disabled before OE# is disabled, the output goes to High-Z by $t_{CEZ}$.*
   *If CE# and OE# are disabled at the same time, the output goes to High-Z by $t_{OEZ}$.*
3. *Synchronous Access Time is calculated using the formula (# of WS)*(clock period) + ($t_{BACC}$) = $t_{IA}$ + $t_{BACC}$*
4. *AVD can not be low for 2 subsequent CLK cycles.*
5. *For faster read cycle time ($t_{RC}$ = 150 ns), operating voltage $V_{CC}$ must be greater than 1.75V.*

**Figure 10.8** Synchronous Read Wrapped Burst

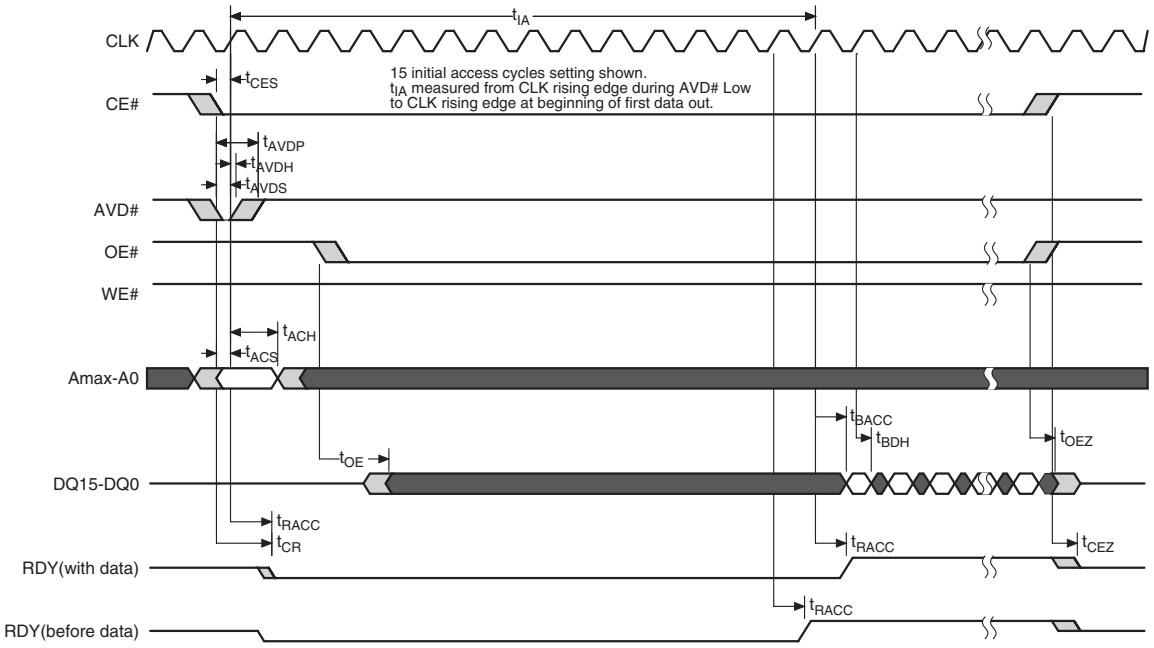
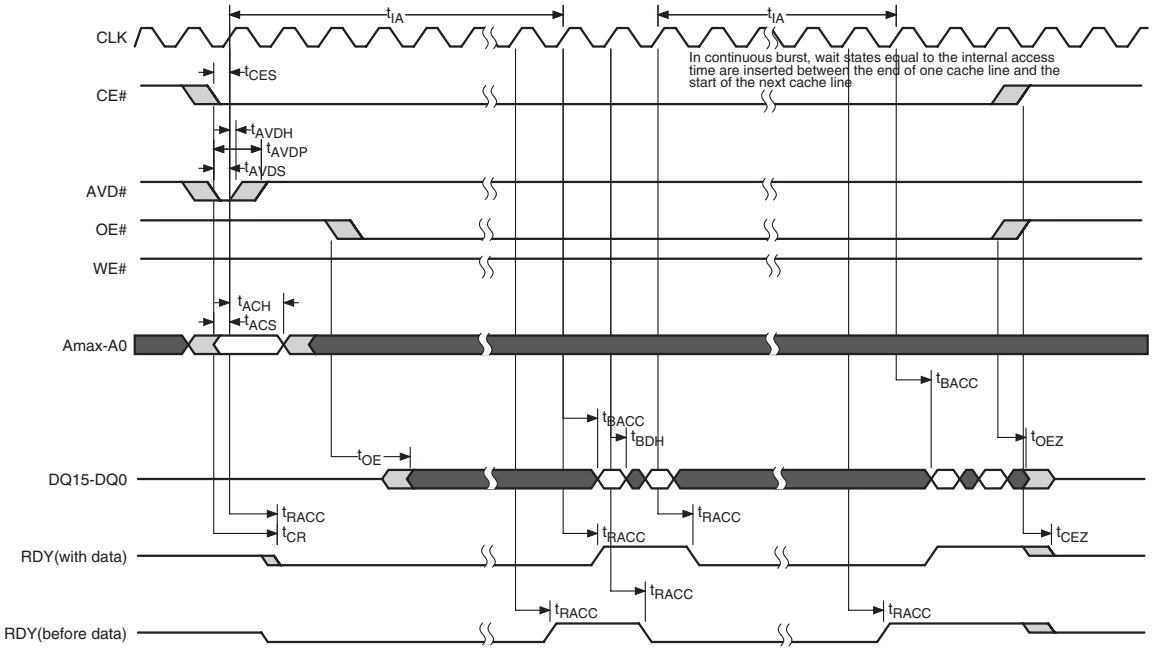



**Figure 10.9** Synchronous Read Continuous Burst



## 10.7.2 AC Characteristics–Asynchronous Read

| Parameter | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Access Time from CE# Low | $t_{CE}$ | – | 115 | ns |
| Asynchronous Access Time from address valid | $t_{ACC}$ | – | 115 (4) | |
| Read Cycle Time (5) | $t_{RC}$ | 210 (4) | – | |
| AVD# Low Time (5) | $t_{AVDP}$ | 9.6 | – | |
| Address Setup to rising edge of AVD# | $t_{AAVDS}$ | 5.5 | – | |
| Address Hold from rising edge of AVD# | $t_{AAVDH}$ | 5.5 | – | |
| Output Enable to Output Valid | $t_{OE}$ | – | 15 | |
| CE# Setup to AVD# falling edge | $t_{CAS}$ | 0 | – | |
| CE# Disable to Output & RDY High Z (1) (2) | $t_{CEZ}$ | – | 15 | |
| OE# Disable to Output High Z (1) (2) | $t_{OEZ}$ | – | 10 | |
| AVD# High to OE# Low (1) (3) | $t_{AVDO}$ | $t_{AAVDH}$ | – | |
| CE# low to RDY valid | $t_{CR}$ | – | 15 | |
| WE# Disable to AVD# Enable | $t_{WEA}$ | 4 | – | |
| WE# Disable to OE# Enable (Note 1) | $t_{OEH}$ | 0 | – | |
| Intra Cache Line Access Time | $t_{ICCC}$ | – | 20 | |

***Notes***

1. *Not 100% tested.*
2. *If OE# is disabled before CE# is disabled, the output goes to High-Z by $t_{OEZ}$.*
   *If CE# is disabled before OE# is disabled, the output goes to High-Z by $t_{CEZ}$.*
   *If CE# and OE# are disabled at the same time, the output goes to High-Z by $t_{OEZ}$.*
3. *In ADP interface address and data are separate so there is no required relationship between AVD# and OE#.*
4. *For faster read cycle time ($t_{RC}$ = 150 ns), operating voltage $V_{CC}$ must be greater than 1.75V, AVD signal toggled.*
5. *Read Cycle time ($t_{RC}$) in an asynchronous read is measured from the falling edge of CE# or AVD# or an address change , which ever occurs later, at the start of the read access, to the next falling edge of CE# or AVD# or address change, whichever occurs first, at the start of a following access.*

**Figure 10.10** Asynchronous Read AVD# Options

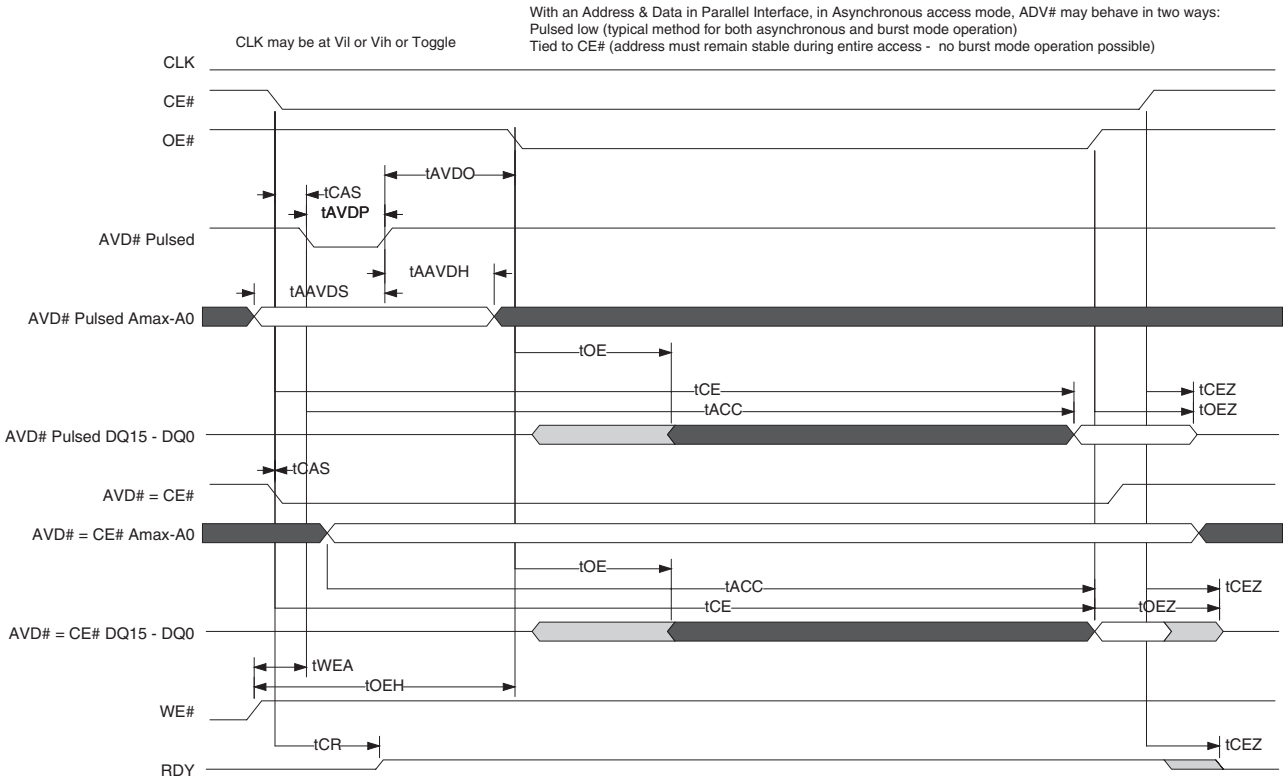


**Figure 10.11** Asynchronous Read

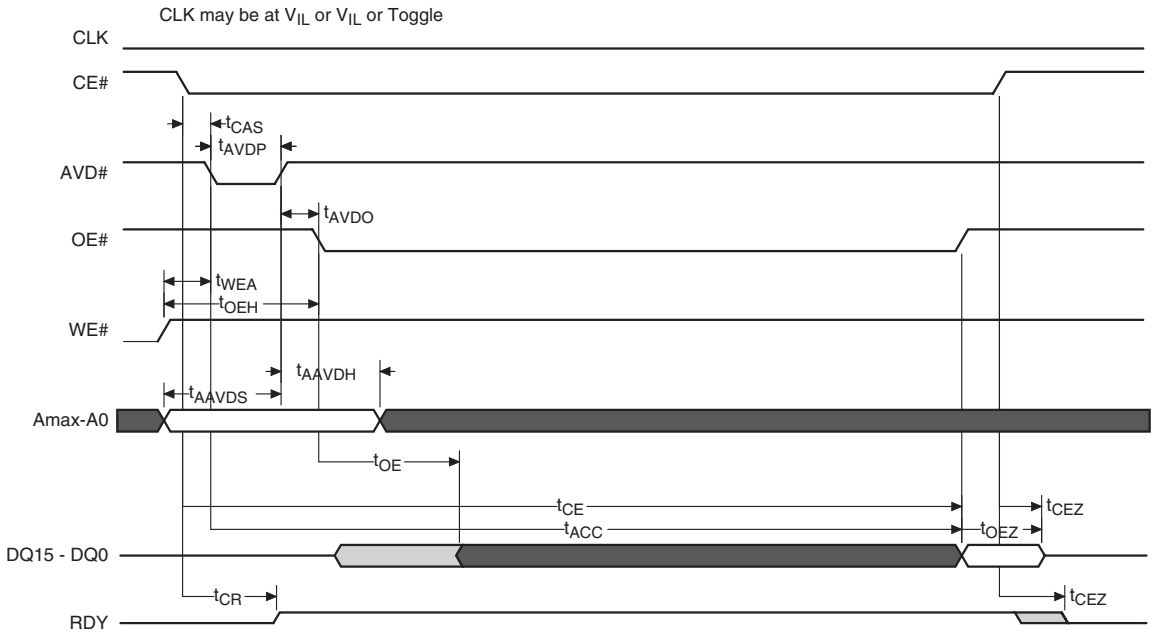



**Figure 10.12** $t_{RC}$ for Asynchronous Read

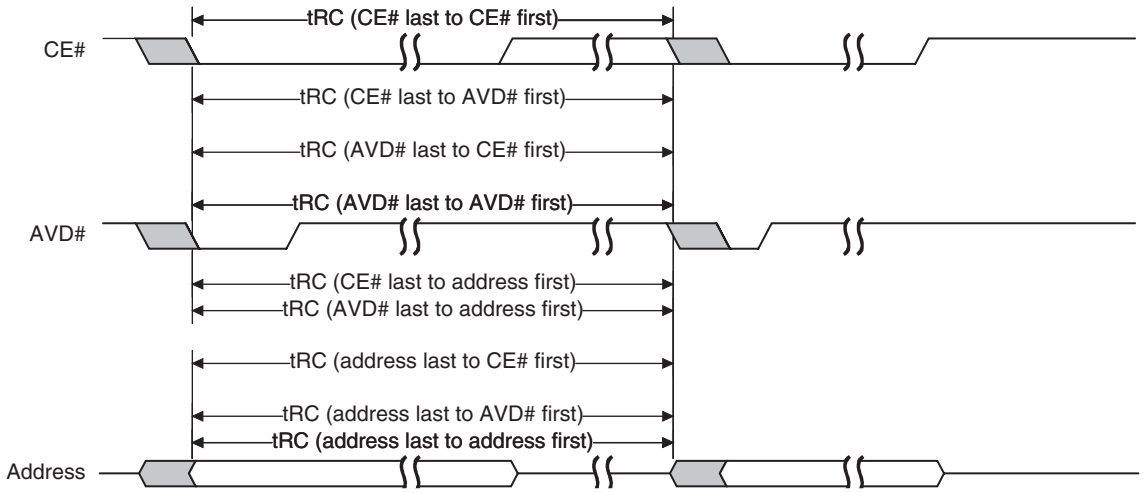
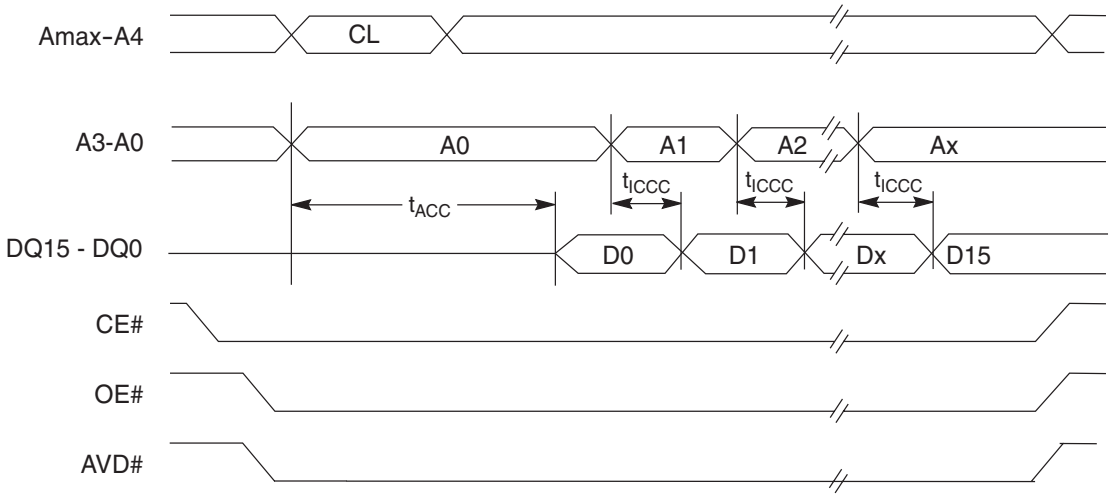


**Figure 10.13** Asynchronous Intra Cache Line Read



### 10.7.3 AC Characteristics–Asynchronous Write Operation

| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| WE# Cycle Time | $t_{WC}$ | 60 | – | – | ns |
| AVD# low pulse width | $t_{AVDP}$ | 9.6 | – | – | ns |
| Address Setup to rising edge of AVD# | $t_{AAVDS}$ | 5.5 | – | – | ns |
| Address Hold from rising edge of AVD# | $t_{AAVDH}$ | 5.5 | – | – | ns |
| Data Setup to rising edge of WE# | $t_{DS}$ | 15 | – | – | ns |
| Read Recovery Time Before Write | $t_{GHWL}$ | 0 | - | - | ns |
| Data Hold from rising edge of WE# | $t_{DH}$ | 0 | – | – | ns |
| CE# Setup to falling edge of WE# | $t_{CS}$ | 4 | – | – | ns |
| CE# Hold from rising edge of WE# | $t_{CH}$ | 0 | – | – | ns |
| AVD# falling edge setup to falling edge of WE# | $t_{AWS}$ | 10 | – | – | ns |
| WE# Pulse Width | $t_{WP}$ | 25 | – | – | ns |
| WE# Pulse Width High | $t_{WPH}$ | 20 | – | – | ns |
| AVD# Disable to WE# Disable (Muxed) | $t_{VLWH}$ | 23.5 | – | – | ns |
| AVD# Disable to WE# Disable (Demuxed) | $t_{VLWH}$ | 10 | – | – | ns |
| WE# Disable to AVD# Enable | $t_{WEA}$ | 4 | – | – | ns |
| CE# low to RDY valid | $t_{CR}$ | – | – | 15 | ns |
| CE# Disable to Output High Z | $t_{CEZ}$ | – | – | 15 | ns |
| OE# Disable to WE# Enable | $t_{WEH}$ | 4 | – | – | ns |
| CE# Setup time to AVD# | $t_{CAS}$ | 0 | – | – | ns |

**Figure 10.14** Asynchronous Write AVD# Options

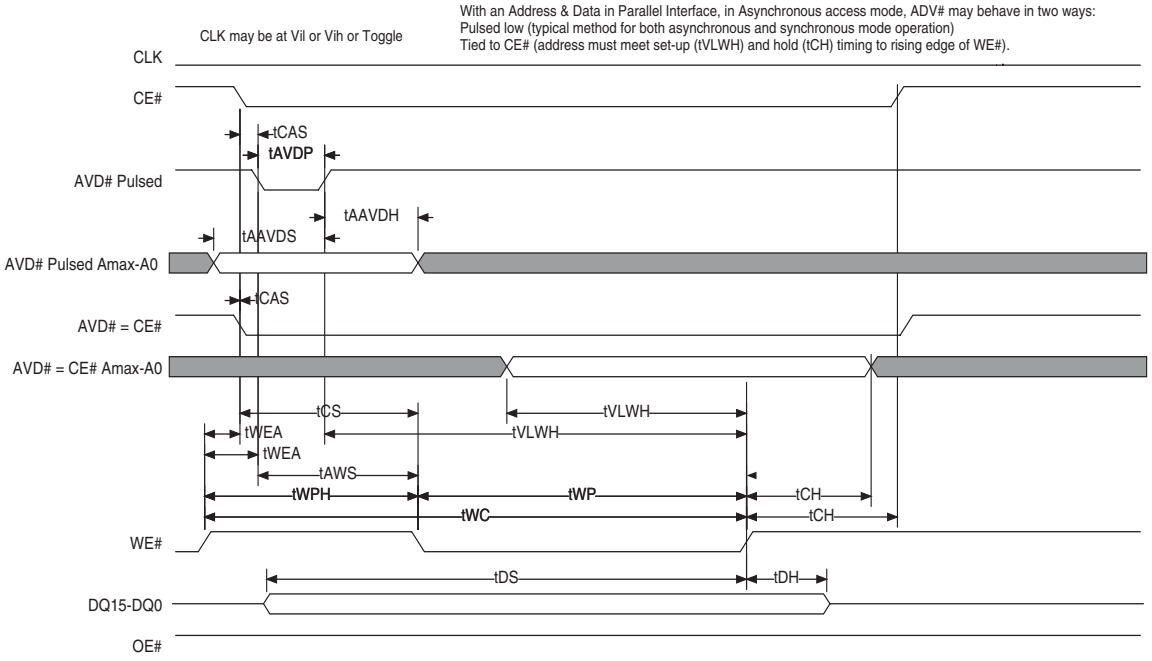
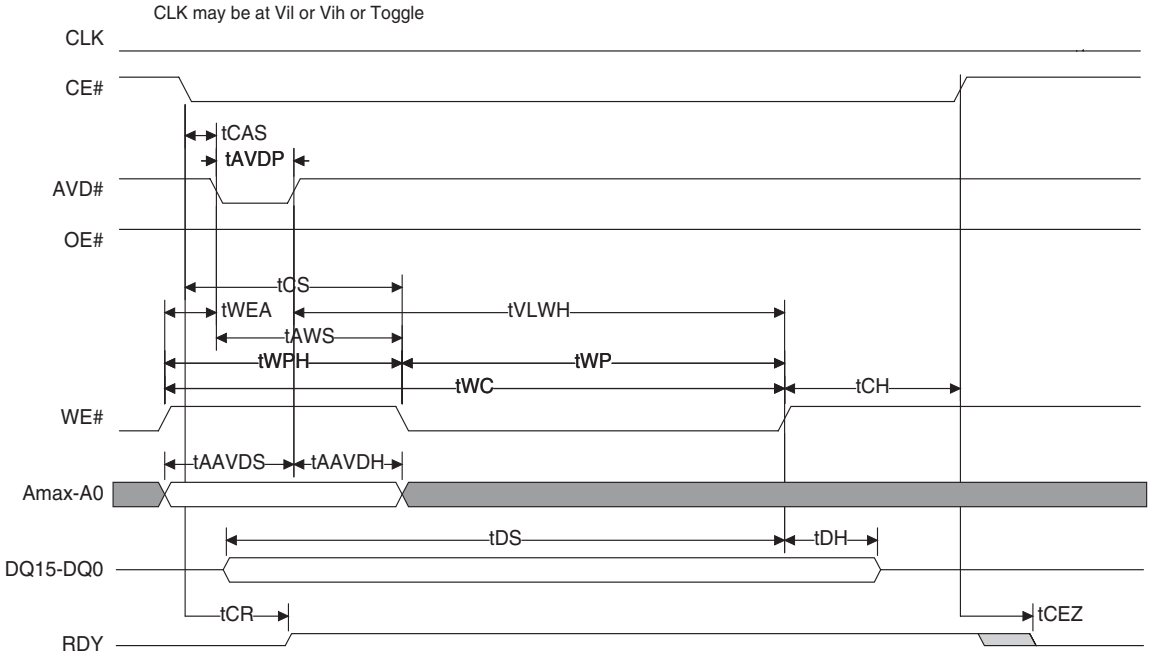


**Figure 10.15** Asynchronous Write




## 10.7.4 Back to Back Read and Write Combinations

**Figure 10.16** Synchronous Read Burst Followed By Read Burst

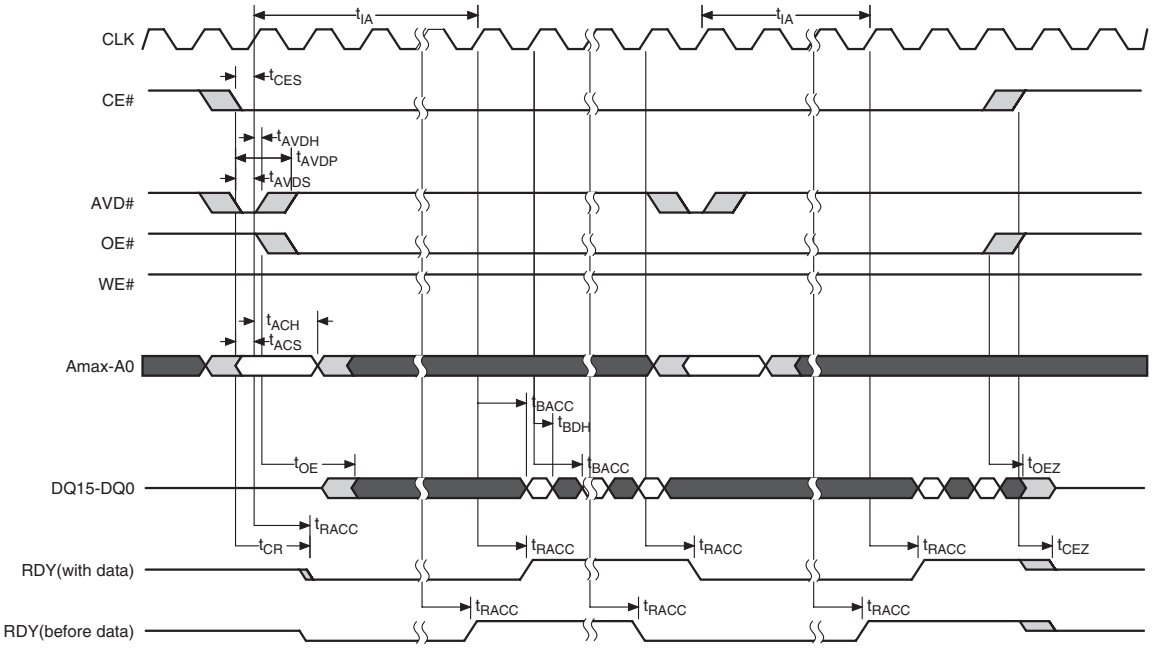


**Figure 10.17** Synchronous Read Burst Followed By Write

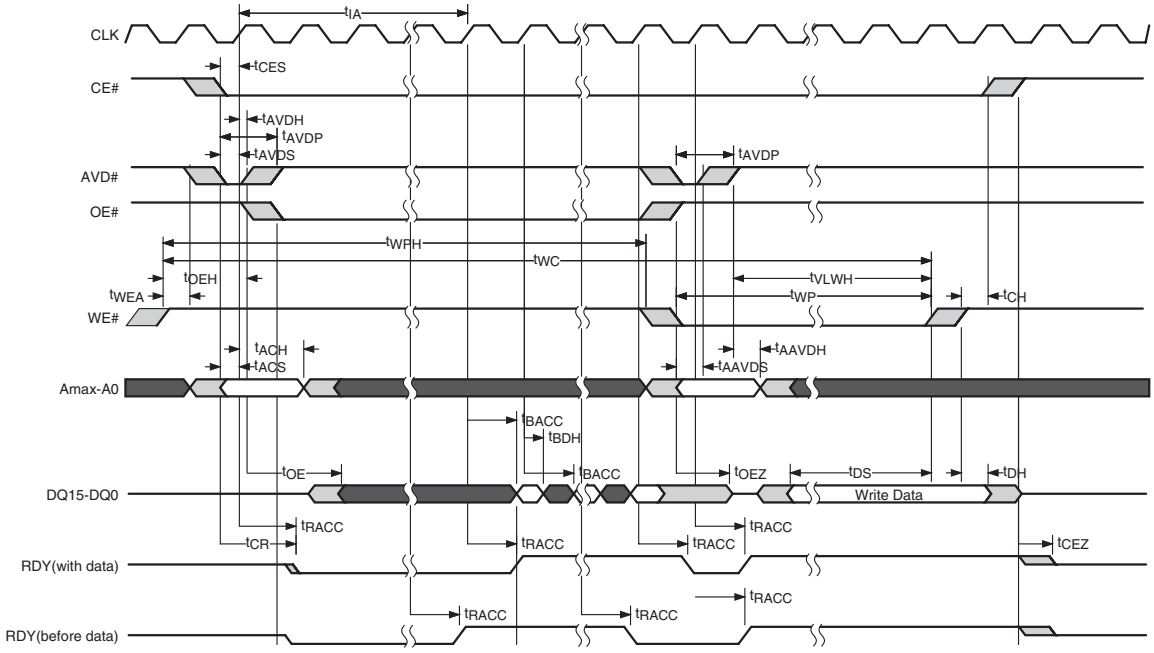



**Figure 10.18** Synchronous Write Followed By Read Burst

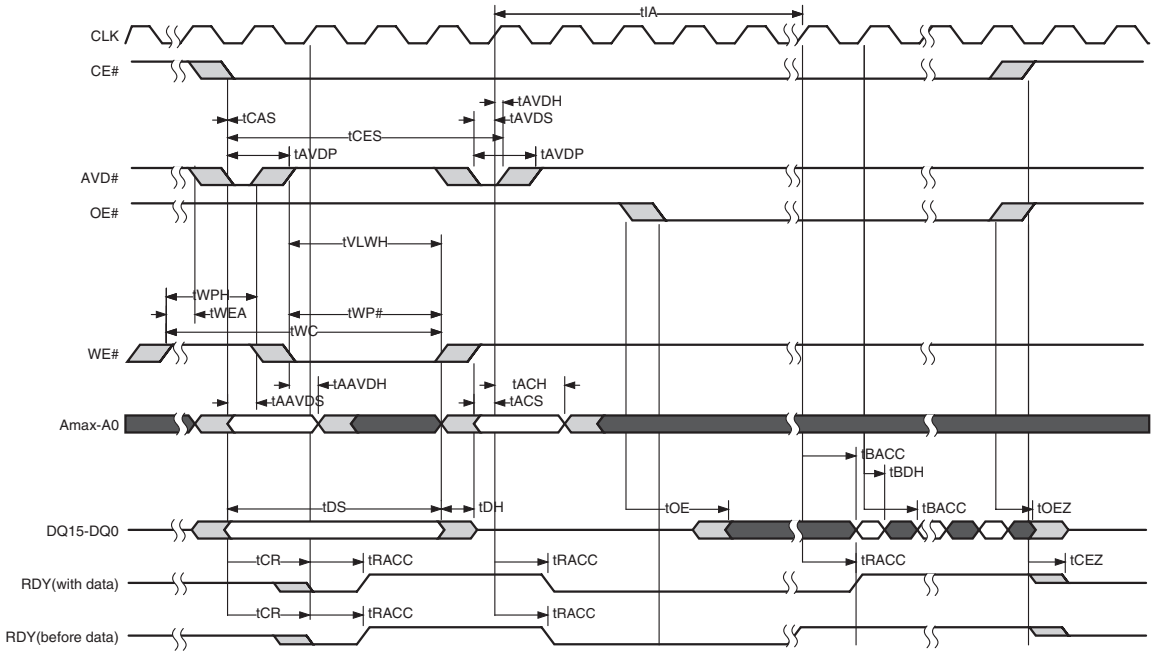


**Figure 10.19** Synchronous Write Followed By Write

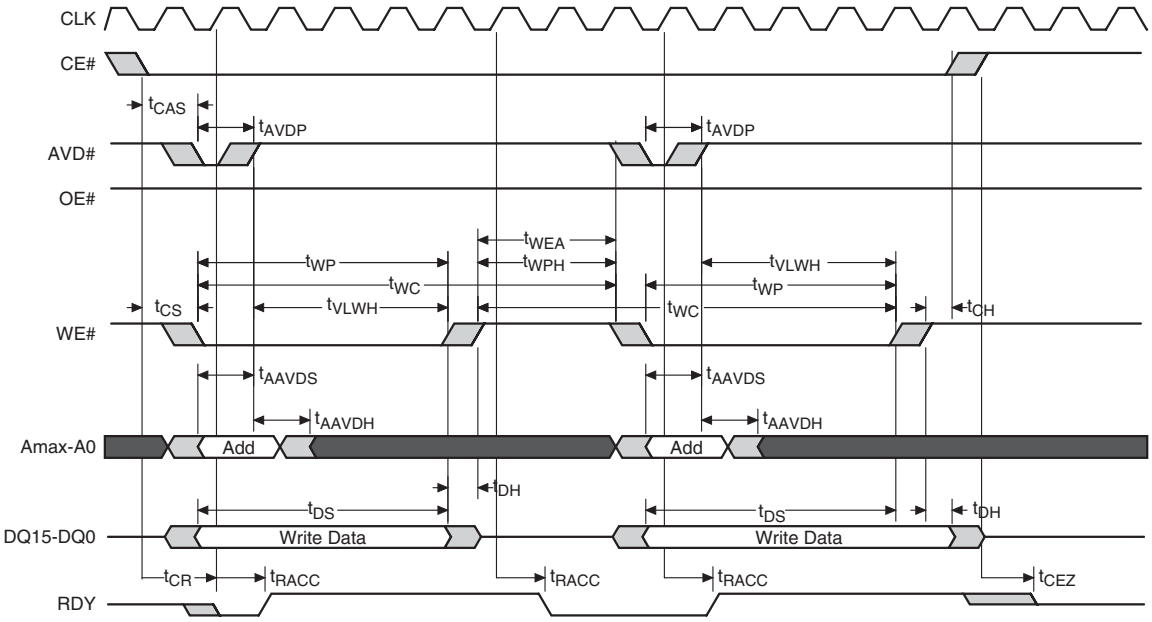



**Figure 10.20** Asynchronous Read Followed By Read

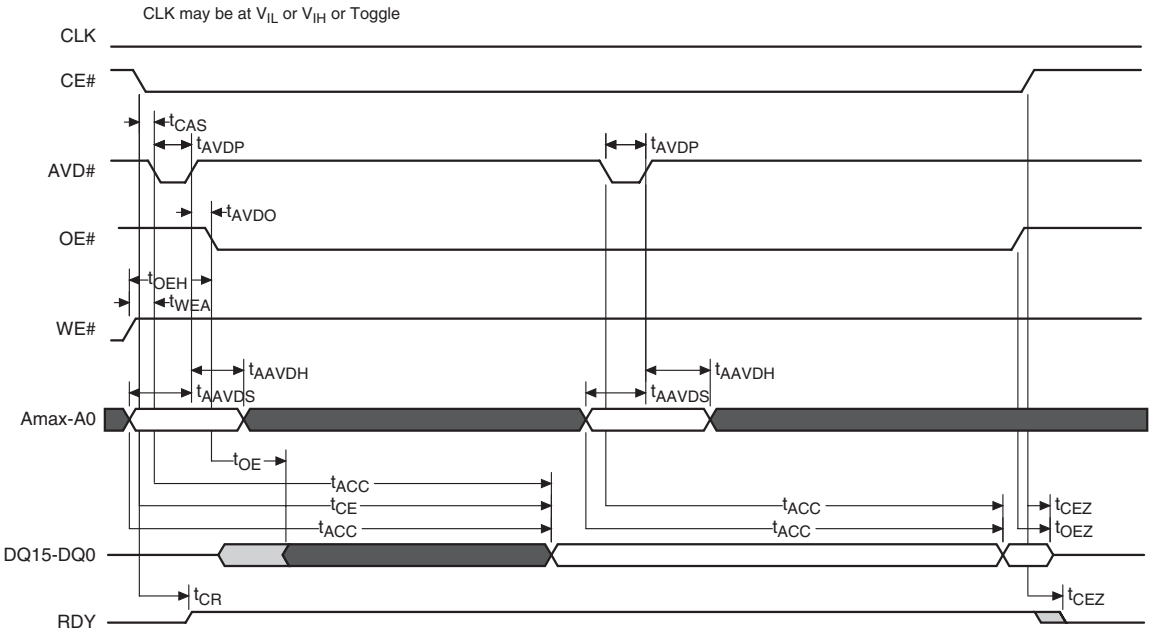


**Figure 10.21** Asynchronous Read Followed By Write

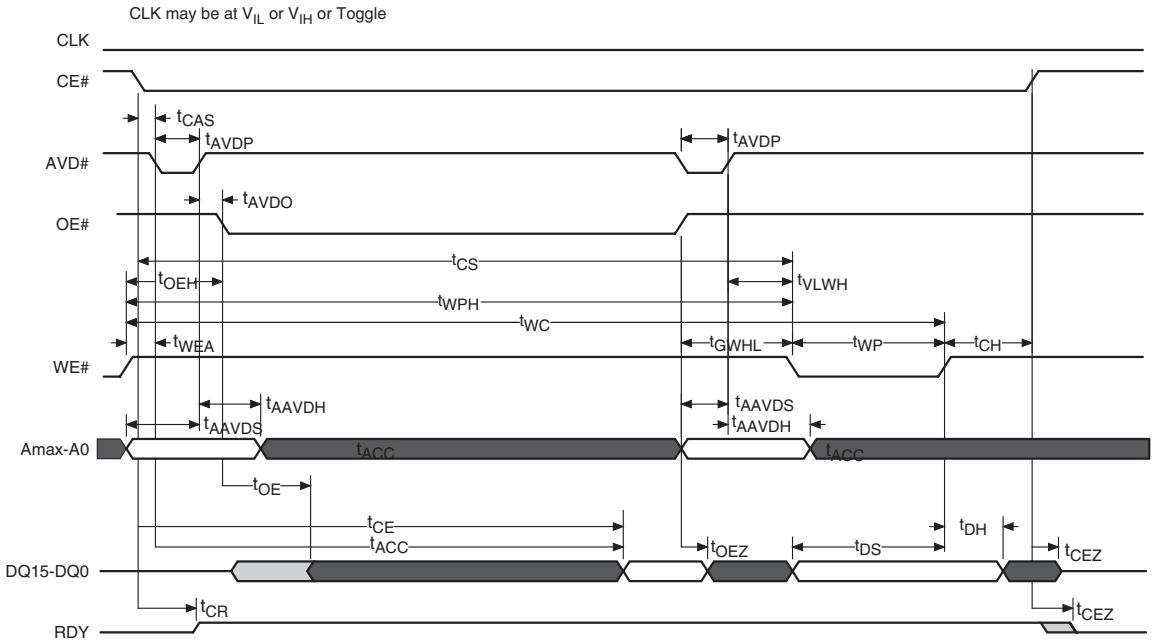



**Figure 10.22** Asynchronous Write Followed By Read

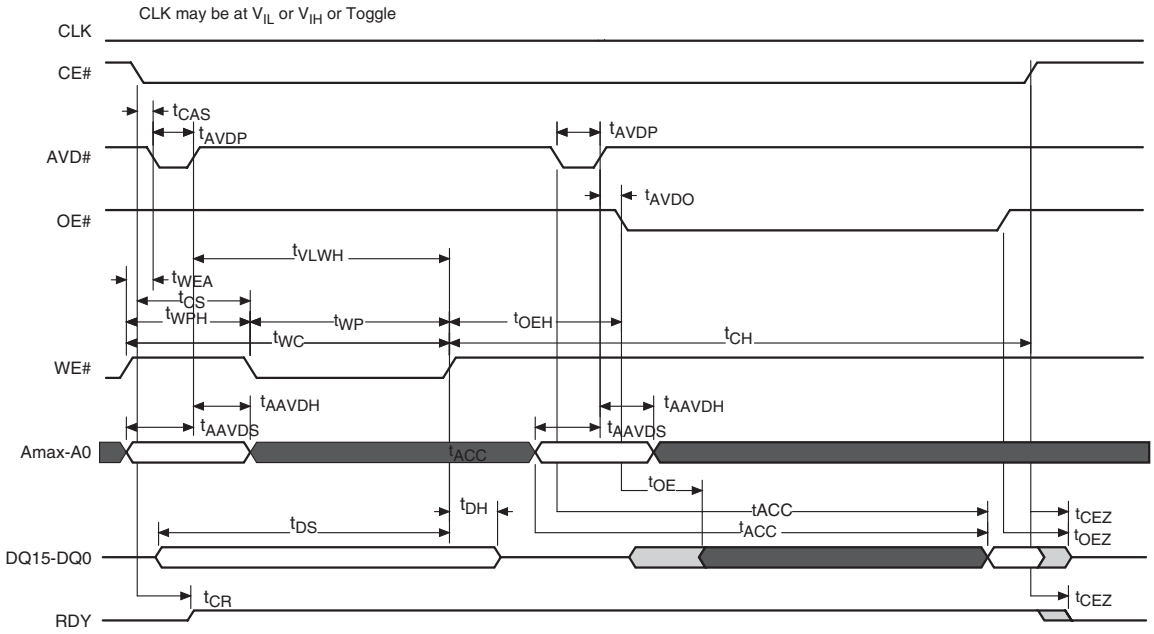
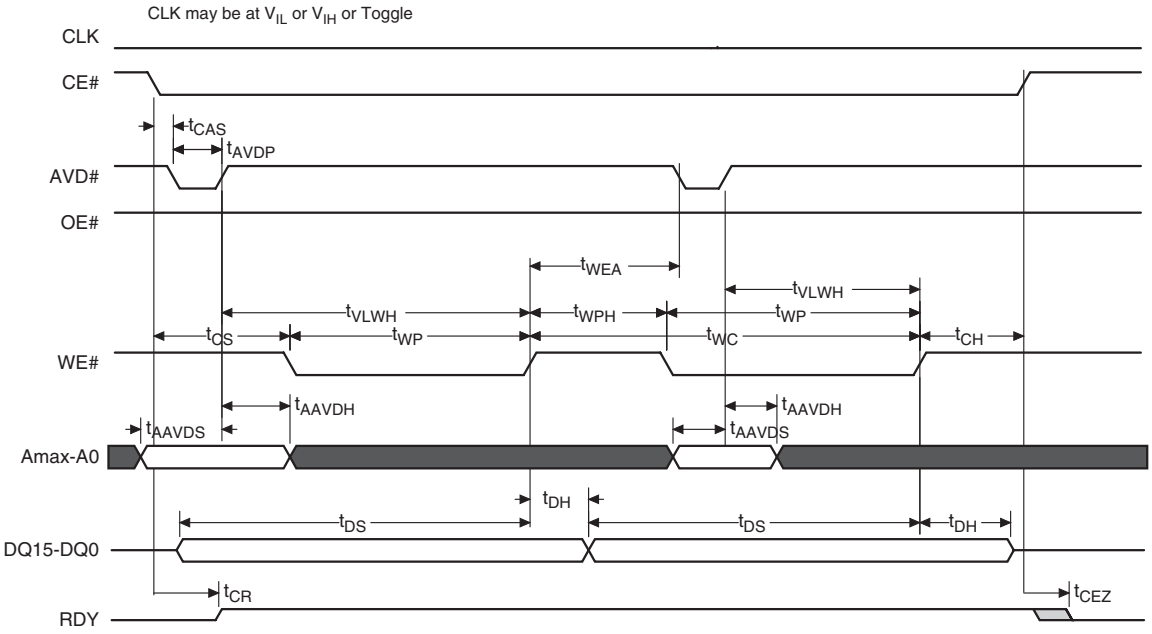


**Figure 10.23** Asynchronous Write Followed By Write



### 10.7.5 Erase and Programming Performance

| Parameter | | | Typ (1) | Max (2) | Unit | Comments |
|---|---|---|---|---|---|---|
| Sector Erase | 0.7 MBytes/s | 128 Kbyte | 180 | 300 | ms | Target Performance<br>Includes 00h programming prior to erasure (3) |
| | | 32 Kbyte | 90 | 150 | | |
| Write Buffer Programming | 2.0 MBytes/s | 512 byte | 244 | 732 | µs | Target Performance<br>Excludes system level overhead (4) |
| Bit-field Write Buffer Programming | | 1 byte | 30 | 90 | | |
| | | 16 byte (CL) | 50 | 150 | | |
| | 1.0 MBytes/s | 256 byte | 247 | 750 | | |
| Sector Programming Time | 2.0 MBytes/s | 128 Kbyte | 63 | 180 | ms | |
| | | 32 Kbyte | 32 | 100 | | |
| Chip Programming | 2.0 MBytes/s | 1 Gbit | 64 | 192 | S | |
| Erase Suspend/Latency ($t_{ESL}$) | | | | 35 | µs | |
| Program Suspend/Latency ($t_{PSL}$) | | | | 35 | µs | |
| Erase Resume to next Erase Suspend ($t_{ERS}$) | | | 100 | | µs | Minimum of 60 ns but ≥ typical periods are needed for Erase to progress to completion. |
| Program Resume to next Program Suspend ($t_{PRS}$) | | | 60 | | µs | Minimum of 60 ns but ≥ typical periods are needed for Program to progress to completion. |
| Blank Check | | | | 5 | ms | |

***Notes***

1. *Typical program and erase times are under the following conditions: 25°C, 1.8 V $V_{CC}$, 10,000 cycles, checkerboard pattern.*
2. *Maximum program and erase times are under the following conditions: 90°C, $V_{CC}$ = 1.70 V, 100,000 cycles, checkerboard pattern.*
3. *In the pre-programming step of the Embedded Erase algorithm, all words are programmed to 00h before erasure.*
4. *System-level overhead is the time required to execute the bus-cycle sequence for the program command. See 11.1, Command Definitions on page 71 for further information on command definitions.*
5. *The device has a minimum erase and program cycle endurance of 10,000 cycles.*

# 11. Appendix - Software Interface Reference

This section contains information relating to software control or interfacing with the Flash device.

## 11.1 Command Definitions

**Table 11.1** Command Definitions (Sheet 1 of 2)

| Command Sequence | | Cycles | Bus Cycles (Notes 1–5) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | First | | Second | | Third | | Fourth | |
| | | | Addr | Data | Addr | Data | Addr | Data | Addr | Data |
| Read | | 1 | RA | RD | | | | | | |
| Reset | | 1 | XX | F0 | | | | | | |
| Write Buffer Load, Page (Note 9) | | 3-258 | CAP1 | 25 | CAP2 | WC | (SA) PA (Note 11) | PD | (SA) PA (Note 12) | PD |
| Write Buffer Load, Bit-Field (Note 9) | | 3-258 | CAP1 | BF | CAP2 | WC | (SA) PA (Note 11) | PD | (SA) PA (Note 12) | PD |
| Buffer to Flash | | 1 | CAP1 | 29 | | | | | | |
| Chip Erase | | 2 | CAP1 | 80 | CAP2 | 10 | | | | |
| Sector Erase | | 2 | CAP1 | 80 | CAP2 | 30 | | | | |
| Status Register Read | | 2 | CAP1 | 70 | (SA) | RR | | | | |
| Status Register Clear | | 1 | CAP1 | 71 | | | | | | |
| Program Suspend (Note 6) | | 1 | XXX | 51 | | | | | | |
| Program Resume (Note 6) | | 1 | XXX | 50 | | | | | | |
| Erase Suspend (Note 7) | | 1 | XXX | B0 | | | | | | |
| Erase Resume (Note 7) | | 1 | XXX | 30 | | | | | | |
| Blank Check | | 1 | CAP1 | 33 | | | | | | |
| Sector Lock/Unlock | | 3 | CAP1 | 60 | CAP2 | 60 | SA | 60 | | |
| Sector Lock Range | | 4 | CAP1 | 60 | CAP2 | 60 | SLA | 61 | SA | 61 |
| **ID/CFI Command Definitions** | | | | | | | | | | |
| ID/CFI | ID/CFI Entry (Note 8) (Note 10) | 1 | CAP3 | 90 or 98 | | | | | | |
| | ID/CFI Read | 1 | (SA) RA | data | | | | | | |
| | ID/CFI Exit | 1 | XXX | F0 | | | | | | |
| **Configuration Command Definitions** | | | | | | | | | | |
| Configuration Register | Configuration Register Entry (Note 8)(Note 10) | 1 | CAP1 | D0 | | | | | | |
| | Write Buffer Load | 3 | CAP1 | 25 | CAP2 | 0 | (SA) X00 or X02 | PD | | |
| | Buffer to Flash (Configuration Register) | 1 | CAP1 | 29 | | | | | | |
| | Configuration Register Read | 1 | (SA) X00 or X02 | RR | | | | | | |
| | Configuration Register Exit | 1 | XXX | F0 | | | | | | |
| **SSR Lock Command Definitions** | | | | | | | | | | |
| SSR Lock | SSR Lock Entry (Note 8)(Note 10) | 1 | CAP1 | 40 | | | | | | |
| | Write Buffer Load (Note 9) | 3 | CAP1 | 25 | CAP2 | 0 | (SA) 00 | PD | | |
| | Buffer to Flash | 1 | CAP1 | 29 | | | | | | |
| | SSR Lock Read | 1 | (SA) XXX | RR | | | | | | |
| | SSR Lock Exit | 1 | XXX | F0 | | | | | | |

**Table 11.1** Command Definitions (Sheet 2 of 2)

| Command Sequence | | Cycles | Bus Cycles (Notes 1–5) | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | First | | Second | | Third | | Fourth | |
| | | | Addr | Data | Addr | Data | Addr | Data | Addr | Data |
| **Secure Silicon Region Command Definitions** | | | | | | | | | | |
| Secure Silicon Region | Secure Silicon Region Entry (Note 8)(Note 10) | 1 | CAP1 | 88 | | | | | | |
| | Write Buffer Load (Note 9) | 3-258 | CAP1 | 25 or BF | CAP2 | WC | (SA) PA (Note 11) | PD | (SA) PA (Note 12) | PD |
| | Buffer to Flash | 1 | CAP1 | 29 | | | | | | |
| | Secure Silicon Region Read | 1 | (SA) RA | RD | | | | | | |
| | Secure Silicon Region Exit | 1 | XXX | F0 | | | | | | |

***Legend***

*X = Don't care*

*RA = Address of the location to be read.*

*RD = Read Data from location RA during read operation.*

*RR = Read Register value*

*PA = Address of the memory location to be programmed.*

*PD = Data to be programmed at location PA.*

*BA = Bank Address, bits sufficient to select a bank*

*SA = Sector Address, bits Amax through A12*

*CAP1 = Command Address Pattern 1 which is a merge (concatenation) of the upper address bits selecting the target sector with a first bit pattern. The Flash address signals Amax to A12 must contain the sector address and A11 to A0 must contain a binary bit pattern of 0101_0101_0101. Note that the Flash address signals express a word address. Flash address signals Amax to A0 are physically connected to system byte address signals amax to a1; example: Flash A22 to A0 connected to system a23 to a1. In terms of a system byte address (typically used in software) the upper address bits amax to a13 are merged with a binary bit pattern of 0_1010_1010_1010 on a12 to a0.*

*CAP2 = Command Address Pattern 2 which is a merge (concatenation) of the upper address bits selecting the target sector with a second bit pattern. The Flash address signals Amax to A12 must contain the sector address and A11 to A0 must contain a binary bit pattern of 1010_1010_1010. Note that the Flash address signals express a word address. Flash address signals Amax to A0 are physically connected to system byte address signals amax to a1; example: Flash A22 to A0 connected to system a23 to a1. In terms of a system byte address (typically used in software) the upper address bits amax to a13 are merged with a binary bit pattern of 1_0101_0101_0100 on a12 to a0. The LSB of byte address must be zero to ensure a word aligned address to place the command data in the lower order byte lane.*

*CAP3 = Command Address Pattern 3 which is a merge (concatenation) of the upper address bits selecting the target sector with a third bit pattern. The Flash address signals Amax to A12 must contain the sector address and A11 to A0 must contain a binary bit pattern of xxxx_0101_0101 (upper bits are don't care). Note that the Flash address signals express a word address. Flash address signals Amax to A0 are physically connected to system byte address signals amax to a1; example: Flash A22 to A0 connected to system a23 to a1. In terms of a system byte address (typically used in software) the upper address bits amax to a13 are merged with a binary bit pattern of x_xxxx_1010_1010 on a12 to a0.*

*SLA = SLR Low Address - Amax through A16 to select the lowest large sector address, A15 through A4 bits are don't care, and A3 to A0 to select the boot sectors to be locked.*

*WBL = Write Buffer Location. Address must be within the same write buffer page as PA.*

*WC = Word Count. Number of write buffer locations to load minus 1.*

***Notes***

1. *See Section 7., Device Operations on page 21 for description of bus operations.*
2. *All values are in hexadecimal.*
3. *Except for the following, all bus cycles are write cycle: read cycle during Read, ID/CFI Read (Manufacturing ID, Device ID, Indicator Bits), Configuration Register read, Secure Silicon Region Read, SSR Lock Read, and 2nd cycle of Status Register Read*
4. *Data bits DQ15–DQ8 are don't care in command sequences, except for RD, PD, and WD.*
5. *Writing incorrect address and data values or writing them in the improper sequence may place the device in an unknown state. The system must write the reset command to return the device to reading array data*
6. *The Program Resume command is valid only during the Program Suspend mode/state.*
7. *The Erase Resume command is valid only during the Erase Suspend mode/state.*
8. *Command is valid when all banks are ready to read array data.*
9. *The total number of cycles in the command sequence is determined by the number of words written to the write buffer.*
10. *Entry commands are needed to enter a specific mode to enable instructions only available within that mode.*
11. *Must be the lowest word address of the cache line being programmed within the 512-byte write buffer page. This is not necessarily the lowest address of the page. Data words are loaded into the write page buffer in sequential order from lowest to highest address.*
12. *Subsequent addresses must fall within the same Sector and Page as the initial starting address.*

## 11.2 Device ID and Common Flash Memory Interface Address Map

The Device ID fields occupy the first 32 bytes of address space followed by the Common Flash Interface data structure. The Common Flash Interface (CFI) specification defines a standardized data structure containing device specific parameter, structure, and feature set information, which allows vendor-specified software algorithms to be used for entire families of devices. Software support can then be device-independent, JEDEC ID-independent, and forward- and back-ward-compatible for the specified flash device families. Flash driver software can be standardized for long-term compatibility.

This device enters the ID/CFI mode when the system writes the ID/CFI Query command, 90h or 98h, to address CAP3 any time all banks are in read mode (the CU is in Idle State). The system can then read ID and CFI information at the addresses, within the selected sector, given in the following tables. To terminate reading ID/CFI, the system must write the reset command. The high order byte of each word is always zero per the JEDEC CFI specification for a 16 bit wide data bus device. Reserved fields have undefined but stable (valid high or low bit) values and are represented as 00xxh. Generally the reserved values are 00FFh but may have other values. Host system software should not depend on any particular value in reserved fields.

**Table 11.2** Device ID

| Word Offset Address | Byte Offset Address | Data | Description |
|---|---|---|---|
| (SA) + 00h | (SA) + 00h | 007Fh | JEDEC Manufacturer ID Continuation Code |
| (SA) + 01h | (SA) + 02h | 007Fh | JEDEC Manufacturer ID Continuation Code |
| (SA) + 02h | (SA) + 04h | 007Fh | JEDEC Manufacturer ID Continuation Code |
| (SA) + 03h | (SA) + 06h | 007Fh | JEDEC Manufacturer ID Continuation Code |
| (SA) + 04h | (SA) + 08h | 007Fh | JEDEC Manufacturer ID Continuation Code |
| (SA) + 05h | (SA) + 0Ah | 0067h | JEDEC Spansion Manufacturer ID |
| (SA) + 06h | (SA) + 0Ch | XS01GS = 0052h | High Order Device ID |
| (SA) + 07h | (SA) + 0Eh | 0001h | Low Order Device ID |
| (SA) + 08h | (SA) + 10h | 00xxh | Reserved |
| (SA) + 09h | (SA) + 12h | 00xxh | Reserved |
| (SA) + 0Ah | (SA) + 14h | 00xxh | Reserved |
| (SA) + 0Bh | (SA) + 16h | 00xxh | Reserved |
| (SA) + 0Ch | (SA) + 18h | 00xxh | Reserved |
| (SA) + 0Dh | (SA) + 1Ah | 00xxh | Reserved |
| (SA) + 0Eh | (SA) + 1Ch | 00xxh | Reserved |
| (SA) + 0Fh | (SA) + 1Eh | 00xxh | Reserved |

**Table 11.3** CFI Query Identification String

| Word Offset Address | Byte Offset Address | Data | Description |
|---|---|---|---|
| (SA) + 10h<br>(SA) + 11h<br>(SA) + 12h | (SA) + 20h<br>(SA) + 22h<br>(SA) + 24h | 0051h<br>0052h<br>0059h | Query Unique ASCII string "*QRY*" |
| (SA) + 13h<br>(SA) + 14h | (SA) + 26h<br>(SA) + 28h | 0000h<br>0003h | Primary OEM Command Set |
| (SA) + 15h<br>(SA) + 16h | (SA) + 2Ah<br>(SA) + 2Ch | 0040h<br>0000h | Address for Primary Extended Table (word offset value) |
| (SA) + 17h<br>(SA) + 18h | (SA) + 2Eh<br>(SA) + 30h | 0000h<br>0000h | Alternate OEM Command Set (00h = none exists) |
| (SA) + 19h<br>(SA) + 1Ah | (SA) + 32h<br>(SA) + 34h | 0000h<br>0000h | Address for Alternate OEM Extended Table (00h = none exists) |

**Table 11.4** System Interface String - 16 bank

| Word Offset Address | Byte Offset Address | Data | Description |
|---|---|---|---|
| | | 1G | |
| (SA) + 1Bh | (SA) + 36h | 0017h | $V_{CC}$ Min. (write/erase)<br>D7–D4: volt, D3–D0: 100 millivolt |
| (SA) + 1Ch | (SA) + 38h | 0019h | $V_{CC}$ Max. (write/erase)<br>D7–D4: volt, D3–D0: 100 millivolt |
| (SA) + 1Dh | (SA) + 3Ah | 0085h | $V_{PP}$ Min. voltage (00h = no $V_{PP}$ pin present)<br>D7–D4: volt, D3–D0: 100 millivolt |
| (SA) + 1Eh | (SA) + 3Ch | 0095h | $V_{PP}$ Max. voltage (00h = no $V_{PP}$ pin present)<br>D7–D4: volt, D3–D0: 100 millivolt |
| (SA) + 1Fh | (SA) + 3Eh | 0005h | Typical timeout per single byte/word write $2^N$ µs (00h = not supported) (<32 µs) |
| (SA) + 20h | (SA) + 40h | 0008h | Typical timeout for Min. size buffer write $2^N$ µs (00h = not supported) (<256 µs per 512-byte program) |
| (SA) + 21h | (SA) + 042h | 0008h | Typical timeout per individual block erase $2^N$ ms (<256 ms) |
| (SA) + 22h | (SA) + 44h | 0012h | Typical timeout for full chip erase $2^N$ ms (00h = not supported)<br>(< 32K, 64K, 128K ms) |
| (SA) + 23h | (SA) + 046h | 0002h | Max. timeout for byte/word write $2^N$ times typical (00h = not supported) <124 µs |
| (SA) + 24h | (SA) +048h | 0002h | Max. timeout for buffer write $2^N$ times typical (4x of 512 byte typical buffer write or <976 µs worst case) |
| (SA) + 25h | (SA) + 4Ah | 0001h | Max. timeout per individual block erase $2^N$ times typical (2x or <360 ms) |
| (SA) + 26h | (SA) + 4Ch | 0001h | Max. timeout for full chip erase $2^N$ times typical (2x or < 512K ms) |

**Table 11.5** Device Geometry Definition (16 Bank Top Boot)

| Word Offset Address | Byte Address | Data | Description |
|---|---|---|---|
| | | 1G | |
| (SA) + 27h | (SA) + 4Eh | 0001Bh | Device Size = $2^N$ byte |
| (SA) + 28h<br>(SA) + 29h | (SA) + 50h<br>(SA) + 52h | 0001h<br>0000h | Flash Device Interface description (refer to JEDEC JEP137)<br>01h = x16-only interface |
| (SA) + 2Ah<br>(SA) + 2Bh | (SA) + 54h<br>(SA) + 56h | 0009h<br>0000h | Max. number of bytes in multi-byte write = $2^N$ (00h = not supported) |
| (SA) + 2Ch | (SA) + 58h | 0002h | Number of Erase Block Regions within device |
| (SA) + 2Dh | (SA) + 5Ah | 00FEh | Erase Block Region 1 Information<br>(refer to the JEDEC JEP137)<br>Number of blocks in region minus 1, two bytes, little endian.<br>Size of blocks in 256 byte units, two bytes, little endian |
| (SA) + 2Eh | (SA) + 5Ch | 0003h | |
| (SA) + 2Fh | (SA) + 5Eh | 0000h | |
| (SA) + 30h | (SA) + 60h | 0002h | |
| (SA) + 31h | (SA) + 62h | 0003h | Erase Block Region 2 Information<br>(refer to the JEDEC JEP137)<br>Number of blocks in region minus 1, two bytes, little endian.<br>Size of blocks in 256 byte units, two bytes, little endian |
| (SA) + 32h | (SA) + 64h | 0000h | |
| (SA) + 33h | (SA) + 66h | 0080h | |
| (SA) + 34h | (SA) + 68h | 0000h | |

**Table 11.6** Device Geometry Definition (16 Bank Uniform Sectors)

| Word Offset Address | Byte Address | Data | Description |
|---|---|---|---|
| | | 1G | |
| (SA) + 27h | (SA) + 4Eh | 0001Bh | Device Size = $2^N$ byte |
| (SA) + 28h<br>(SA) + 29h | (SA) + 50h<br>(SA) + 52h | 0001h<br>0000h | Flash Device Interface description (refer to JEDEC JEP137)<br>01h = x16-only interface |
| (SA) + 2Ah<br>(SA) + 2Bh | (SA) + 54h<br>(SA) + 56h | 0009h<br>0000h | Max. number of bytes in multi-byte write = $2^N$ (00h = not supported) |
| (SA) + 2Ch | (SA) + 58h | 0001h | Number of Erase Block Regions within device |
| (SA) + 2Dh | (SA) + 5Ah | 00FFh | Erase Block Region 1 Information<br>(refer to the JEDEC JEP137)<br>Number of blocks in region minus 1, two bytes, little endian.<br>Size of blocks in 256 byte units, two bytes, little endian |
| (SA) + 2Eh | (SA) + 5Ch | 0003h | |
| (SA) + 2Fh | (SA) + 5Eh | 0000h | |
| (SA) + 30h | (SA) + 60h | 0002h | |

**Table 11.7** Device Geometry Definition (16 Bank Bottom Boot)

| Word Offset Address | Byte Address | Data | Description |
|---|---|---|---|
| | | 1G | |
| (SA) + 27h | (SA) + 4Eh | 0001Bh | Device Size = $2^N$ byte |
| (SA) + 28h<br>(SA) + 29h | (SA) + 50h<br>(SA) + 52h | 0001h<br>0000h | Flash Device Interface description (refer to JEDEC JEP137)<br>01h = x16-only interface |
| (SA) + 2Ah<br>(SA) + 2Bh | (SA) + 54h<br>(SA) + 56h | 0009h<br>0000h | Max. number of bytes in multi-byte write = $2^N$ (00h = not supported) |
| (SA) + 2Ch | (SA) + 58h | 0002h | Number of Erase Block Regions within device |
| (SA) + 2Dh | (SA) + 5Ah | 0003h | Erase Block Region 1 Information<br>(refer to the JEDEC JEP137)<br>Number of blocks in region minus 1, two bytes, little endian.<br>Size of blocks in 256 byte units, two bytes, little endian |
| (SA) + 2Eh | (SA) + 5Ch | 0000h | |
| (SA) + 2Fh | (SA) + 5Eh | 0080h | |
| (SA) + 30h | (SA) + 60h | 0000h | |
| (SA) + 31h | (SA) + 62h | 00FEh | Erase Block Region 2 Information<br>(refer to the JEDEC JEP137)<br>Number of blocks in region minus 1, two bytes, little endian.<br>Size of blocks in 256 byte units, two bytes, little endian |
| (SA) + 32h | (SA) + 64h | 0003h | |
| (SA) + 33h | (SA) + 66h | 0000h | |
| (SA) + 34h | (SA) + 68h | 0002h | |

**Table 11.8** Primary Vendor-Specific Extended Query -16 Bank

| Word Offset Address | Byte Address | Data | | | Description |
|---|---|---|---|---|---|
| | | 512 | 1G | 2G | |
| (SA) + 40h<br>(SA) + 41h<br>(SA) + 42h | (SA) + 80h<br>(SA) + 82h<br>(SA) + 84h | 0050h<br>0052h<br>0049h | | | Query-unique ASCII string *PRI*<br>Refer to the Spansion CFI version 1.4 or later document for full field definitions |
| (SA) + 43h | (SA) + 86h | 0031h | | | Major version number, ASCII |
| (SA) + 44h | (SA) + 88h | 0035h | | | Minor version number, ASCII |
| (SA) + 45h | (SA) + 8Ah | 0021h | | | Address Sensitive Unlock (Bits 1-0):<br>00b = Required<br>01b = Not required<br>Process Technology (Bits 5-2)<br>0011b = 130 nm Floating-Gate Technology<br>0100b = 110 nm MirrorBit™ Technology<br>0101b = 90 nm Floating-Gate Technology<br>0110b = 90 nm MirrorBit™ Technology<br>1000b = 65 nm MirrorBit™ Technology<br>1001b = 45 nm MirrorBit™ Technology |
| (SA) + 46h | (SA) + 8Ch | 0002h | | | Erase Suspend,<br>0 = Not Supported, 1 = To Read Only, 2 = To Read & Write |
| (SA) + 47h | (SA) + 8Eh | 0001h | | | Sector Protect,<br>0 = Not Supported, X = Number of sectors in per group |
| (SA) + 48h | (SA) + 90h | 0000h | | | Sector Temporary Unprotect<br>00 = Not Supported, 01 = Supported |
| (SA) + 49h | (SA) + 92h | 0009h | | | Sector Protect/Unprotect scheme<br>09h = Single-Sector Lock + Sector Lock Range |
| (SA) + 4Ah | (SA) + 94h | 0020h | 0040h | 0080h | Simultaneous Operation<br>Number of Sectors in all banks except boot bank |
| (SA) + 4Bh | (SA) + 96h | 0001h | | | Burst Mode Type<br>00 = Not Supported, 01 = Supported |
| (SA) + 4Ch | (SA) + 98h | 0004h | | | Page Mode Type,<br>00 = Not Supported, 01 = 4 Word Page, 02 = 8 Word Page, 04 = 16 Word Page |
| (SA) + 4Dh | (SA) + 9Ah | 0000h | | | ACC (Acceleration) Supply Minimum<br>(00h = Not Supported, D7-D4: Volt, D3-D0: 100 mV) |
| (SA) + 4Eh | (SA) + 9Ch | 0000h | | | ACC (Acceleration) Supply Maximum<br>(00h = Not Supported, D7-D4: Volt, D3-D0: 100 mV) |
| (SA) + 4Fh | (SA) + 9Eh | 0002h (Bottom Boot)<br>0003h (Top Boot)<br>0004h (Uniform Bottom Protect) | | | Top/Bottom Boot Sector Flag<br>00h = No Boot<br>01h = Dual Boot<br>02h = Bottom boot<br>03h = Top boot<br>04h = Uniform Bottom Protect<br>05h = Uniform Top Protect |
| (SA) + 50h | (SA) + A0h | 0001h | | | Program Suspend. 00h = not supported 01=Supported |
| (SA) + 51h | (SA) + A2h | 0000h | | | Unlock Bypass, 00 = Not Supported, 01=Supported |
| (SA) + 52h | (SA) + A4h | 0009h | | | Secure Silicon Region (Customer SSR Area) Size $2^N$ bytes |
| (SA) + 53h | (SA) + A6h | 000Eh | | | Hardware Reset Low Time-out during an embedded algorithm to read mode Maximum $2^N$ ns (<16 ms) |
| (SA) + 54h | (SA) + A8h | 000Eh | | | Hardware Reset Low Time-out not during an embedded algorithm to read mode Maximum $2^N$ ns (<16 ms) |
| (SA) + 55h | (SA) + AAh | 0005h | | | Erase Suspend Time-out Maximum $2^N$ µs (<32 µs) |
| (SA) + 56h | (SA) + ACh | 0005h | | | Program Suspend Time-out Maximum $2^N$ µs (<32 µs) |

**Table 11.9** Primary Vendor-Specific Extended Query 16 Bank Region Information

| Word Offset Address | Byte Address | Data | | | Description |
|---|---|---|---|---|---|
| | | 512 | 1G | 2G | |
| (SA) + 57h | (SA) + AEh | 0010h | | | Bank Organization: X = Number of banks |
| (SA) + 58h | (SA) + B0h | 0020h | 0040h | 0080h | Bank 0 Region Information. X = Number of 128KByte units in bank |
| (SA) + 59h | (SA) + B2h | 0020h | 0040h | 0080h | Bank 1 Region Information. X = Number of 128KByte units in bank |
| (SA) + 5Ah | (SA) + B4h | 0020h | 0040h | 0080h | Bank 2 Region Information. X = Number of 128KByte units in bank |
| (SA) + 5Bh | (SA) + B6h | 0020h | 0040h | 0080h | Bank 3 Region Information. X = Number of 128KByte units in bank |
| (SA) + 5Ch | (SA) + B8h | 0020h | 0040h | 0080h | Bank 4 Region Information. X = Number of 128KByte units in bank |
| (SA) + 5Dh | (SA) + BAh | 0020h | 0040h | 0080h | Bank 5 Region Information. X = Number of 128KByte units in bank |
| (SA) + 5Eh | (SA) + BCh | 0020h | 0040h | 0080h | Bank 6 Region Information. X = Number of 128KByte units in bank |
| (SA) + 5Fh | (SA) + BEh | 0020h | 0040h | 0080h | Bank 7 Region Information. X = Number of 128KByte units in bank |
| (SA) + 60h | (SA) + C0h | 0020h | 0040h | 0080h | Bank 8 Region Information. X = Number of 128KByte units in bank |
| (SA) + 61h | (SA) + C2h | 0020h | 0040h | 0080h | Bank 9 Region Information. X = Number of 128KByte units in bank |
| (SA) + 62h | (SA) + C4h | 0020h | 0040h | 0080h | Bank 10 Region Information. X = Number of 128KByte units in bank |
| (SA) + 63h | (SA) + C6h | 0020h | 0040h | 0080h | Bank 11 Region Information. X = Number of 128KByte units in bank |
| (SA) + 64h | (SA) + C8h | 0020h | 0040h | 0080h | Bank 12 Region Information. X = Number of 128KByte units in bank |
| (SA) + 65h | (SA) + CAh | 0020h | 0040h | 0080h | Bank 13 Region Information. X = Number of 128KByte units in bank |
| (SA) + 66h | (SA) + CCh | 0020h | 0040h | 0080h | Bank 14 Region Information. X = Number of 128KByte units in bank |
| (SA) + 67h | (SA) + CEh | 0020h | 0040h | 0080h | Bank 15 Region Information. X = Number of 128KByte units in bank |

# 12. Revision History

| Section | Description |
|---|---|
| **Revision 01 (March 23, 2009)** | |
| | Initial release |
| **Revision 02 (April 16, 2009)** | |
| Asynchronous Intra Cache Line Read | Add section 7.2.1.1 Cache Line Persistence |
| Secure Silicon Region | Add Write Buffer programming to be used in SSR |
| AC Characteristics - Synchronous Burst Read | Remove $t_{RC}$ (Read Cycle Time) from Table 10.3 1 Gbit<br>Remove Figure 10.10, $t_{RC}$ for Synchronous Read |
| Erase and Programming Performance | Revise chip and buffer programming performance from 1.3 MB/s to 2.0 MB/s<br>Change Write Buffer Programming typical value of 512 byte from 375 µs to 244 µs<br>Change sector programming time of 128 Kbyte from 96 ms to 63 ms<br>Change 1G chip programming time from 98 sec to 64 sec |
| Device ID and CFI Address Map | Delete 512 M and 2 G from CFI table<br>Update Table 11.2 Device ID:<br>to correct Data value for (SA) + 06h to XS01GS = 0052h only<br>Update Table 11.4 System Interface String - 16 bank:<br>to correct Data value for (SA) + 21h to 0008h and comments to < 256 ms<br>to correct Data value for (SA) + 22h to 0012<br>to correct comment for (SA) + 24h to 4x or <976 µs<br>to correct Data value for (SA) + 25h to 0001h and comments to 2x or <360 ms<br>to correct Data value for (SA) + 26h to 0000h<br>Add Table 11.7 Device Geometry Definition (16 Bank Bottom Boot)<br>Update Table 11.8 Primary Vendor-Specific Extended Query -16 Bank:<br>to correct Data value for (SA) + 4Ch to 0004h |

### *Colophon*

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

### *Trademarks and Notice*

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.