AP-646

APPLICATION NOTE

Common Flash Interface (CFI) and Command Sets

December 1997

Order Number: 292204-003

Information in this document is provided in connection with Intel products. No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Intel literature, may be obtained from:

Intel Corporation
P.O. Box 5937
Denver, CO 80217-9808

or call 1-800-548-4725
or visit Intel's Website at http://www.intel.com

CONTENTS

REVISION HISTORY

Number	Description
-001	Original Version
-002	Added ANSI ‘C’ code to Appendix A
-003	Removed routine codes from the Appendices

1.0 INTRODUCTION

This application note defines Common Flash Interface (CFI), Basic Command Set (BCS), and Scaleable Command Set (SCS) as well as discusses their benefits and details how best to use them.

Common Flash Interface (CFI) is a published, standardized data structure that may be read from a flash memory device. CFI allows system software to query the installed device (on board component, PC (PCMCIA) Card, or Miniature Card) to determine configurations, various electrical and timing parameters, and functions supported by the device.

The Basic Command Set (BCS) is a group of commands that have been used for years on Intel's and other vendors' legacy products. This command set is also commonly referred to as the 28F008 or simply the 008 command set. These commands include Read Array, Read ID, Read Status Register, Clear Status Register, Program (Write), Block Erase, Erase Suspend, and Confirm/Resume. The BCS is the "Standard Command Set" used by Intel in its CFI implementations.

Scaleable Command Set (SCS) is the "Extended Command Set" that Intel uses to control the functions of most CFI enabled flash devices. CFI allows the vendor to specify a command set that should be used with the component. SCS is the command set that will be used by Intel on most of its CFI enabled devices. SCS includes all commands available in the BCS as well as some new advanced commands that have been designed to take advantage of Intel's next generation optimized flash devices. These new commands include Set and Clear Lock Bits, CFI Query, Write to Buffer, Program Suspend, Status Configuration, and Full Chip Erase. With many new capabilities being designed into flash products today, these new commands were necessary to take full advantage of the improvements.

CFI is used to allow the system to learn how to interface to the flash device most optimally. The BCS and SCS are used to then command the device to perform the desired flash functions.

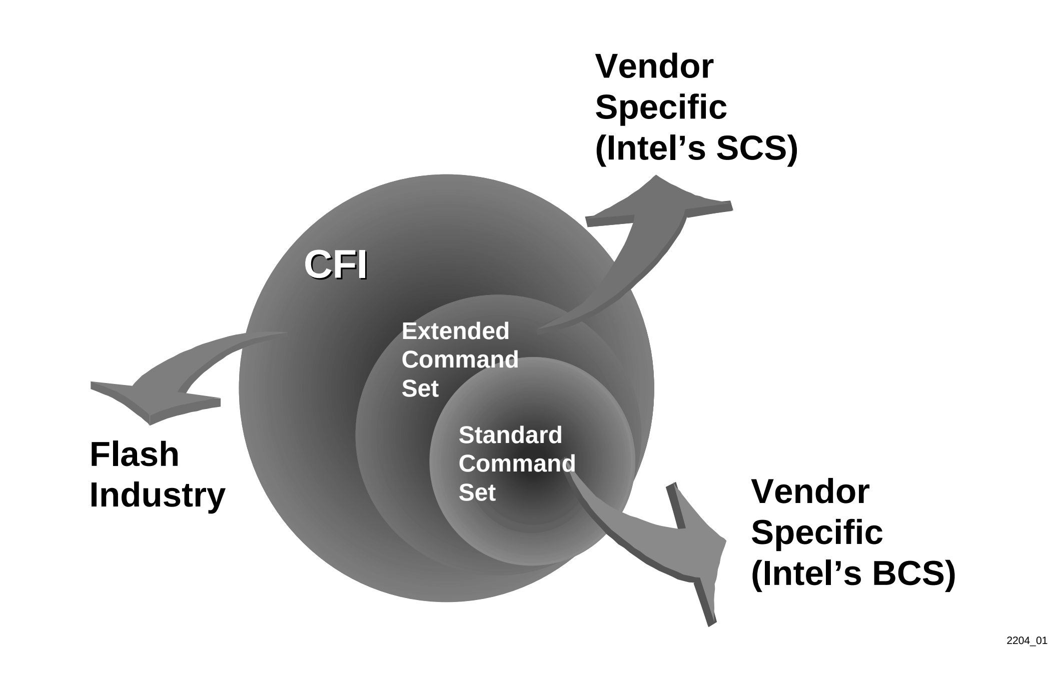


Figure 1. How CFI, SCS, and BCS Fit Together

Removable
Media
Embedded
Products
Future
Upgrade
Compatibility among
removable media
Silicon updates
without software
changes
Mebius
NOKIA

Figure 2. CFI Allows Easy Upgrades and Use of Second Sources

2.0 BENEFITS OF CFI

The two primary benefits of using CFI are ease of upgrading and second source availability. Both are concerns when an OEM or end-user (the consumer) purchases a product.

2.1 Upgrades

In order to take advantage of increased densities (or speeds, etc.) on memory devices and cards, an easy upgrade path is desirable. Care is generally taken to ensure that hardware footprints are pin-for-pin compatible or that flexible layouts may be used when upgrading a product. Thought is seldom given to software compatibility, however. CFI allows many new and improved products to be used in place of their older versions without modifications of system software.

Because CFI allows the system to learn about the features, parameters, and timings of a flash device, the system can take full advantage of these improvements. For instance, if the timeout for a block erase to occur was cut in half, the system software could take advantage of that fact by changing its internal timers. Also, a 32-Mbit device can be replaced by a 64-Mbit device and vice versa because the device can tell the system what size it is.

With CFI, when upgrading a flash memory design, it is no longer necessary to re-optimize low level software drivers to take advantage of the new features. Simply program the system initially to accept CFI enabled devices, and allow the software to upgrade itself.

2.2 Second Sources

Particularly in the card environment, second sourcing is a primary concern. Because the end-user of a PC Card or Miniature Card could be a consumer, care must be taken to ensure compatibility among all flash cards that may be installed into the same sockets. For instance, when purchasing a replacement or spare memory card for their digital camera, a consumer does not want to have to worry that they can only purchase a certain vendor's card or a particular version of what seems to be a similar card.

This is analogous to a consumer purchasing floppy disks for their computer or film for their camera. Any vendor's product works. That is the goal of CFI—complete and simple interchange between vendors in a card application. The hardware inside the PC Card or Miniature Card does not have to operate identically; the software takes care of the differences as long as the devices are CFI-compliant.

CFI allows the system to determine the manufacturer of the card, its operating parameters, its configuration, and any special command codes that the card may accept. With this knowledge, the system can optimize its use of the card by using appropriate timeout values, optimal voltages, and commands necessary to use the card to its full advantage.

3.0 HOW TO USE CFI EFFECTIVELY

To use CFI effectively, system software must be written to take advantage of the flexibility provided by the specification. The software must be capable of modifying timeouts, adjusting to different memory sizes, accommodating varying block erase characteristics, and branching to vendor-specific code sections. The following paragraphs outline several steps which system software must transition through to read a CFI enabled device. Flow charts are included in *Appendix A1*.

3.1 Read Query String

Not all devices installed into a flash memory socket will be CFI-enabled. To determine if a device is CFI-capable, the system software must write a 98h to location 55h within the memory (see *CFI Query Flowchart* included in *Appendix A1*). The flash device may or may not have an address sensitive query command; the Intel devices do not. The low-level driver, however, should supply the 55h address even though the flash device may choose to ignore the address bus and enter the query mode if 98h is on the data bus only. If three consecutive maximum device bus width reads beginning at location 10h in the flash array return the ASCII equivalent "Q," "R," and "Y," then the device is CFI-compliant.

Although there are other configuration possibilities, there are currently three CFI array configurations that are of primary interest. These configurations must be tested for and accounted for in the software. These configurations are:

- single chip operating in a x16 mode (16-bit data bus)—chips may be capable of 8 bit accesses, but are operating only with 16-bit bus accesses
- two chips each capable of 8- and 16-bit data bus accesses, but each only operating in a x8 mode (8-bit data bus on each chip with a total array bus width of 16 bits)
- two chips each only capable of 8-bit data bus accesses operating only in a x8 mode (8-bit data bus per chip with a total array bus width of 16 bits)

Each of these configurations are shown in Figure 3. Table 1 indicates the addressing necessary to read the CFI query table for each of these configurations along with some other possible device configurations. The table also includes what the query data will look like to the host processor in byte or word addressing. Note that the query data (ASCII "Q", "R", and "Y" as well as the electronic databook information discussed in the next section) may be doubled or even quadrupled depending on the array configuration. The software must be able to determine the correct array configuration based on the number of "Q"s returned to accurately calculate the array size and read and write to the array properly. The *CFI Query Flowchart* is located in *Appendix A1*. The QueryCFI routine heuristically determines the configuration of the array, calculates the appropriate data, and indicates to the higher level routines how to communicate with the CFI-enabled devices.

If the device does not respond with the "QRY" string, the device is not CFI-compliant and the software must then attempt to read the device's JEDEC ID. (See the *Memory Heuristics Flowchart* included in *Appendix A1*.) The software must write a 90h to the first location in the device. If the device returns a Manufacturer's ID and Component ID, the flash device may be accessed as it has been in the past, based on the Manufacturer and Component ID. If the device does not return a Manufacturer and Component ID, then the device is not a flash memory and other routines are necessary to determine what type of device is installed. (See the *Memory Heuristics Flowchart* included in *Appendix A1*.)

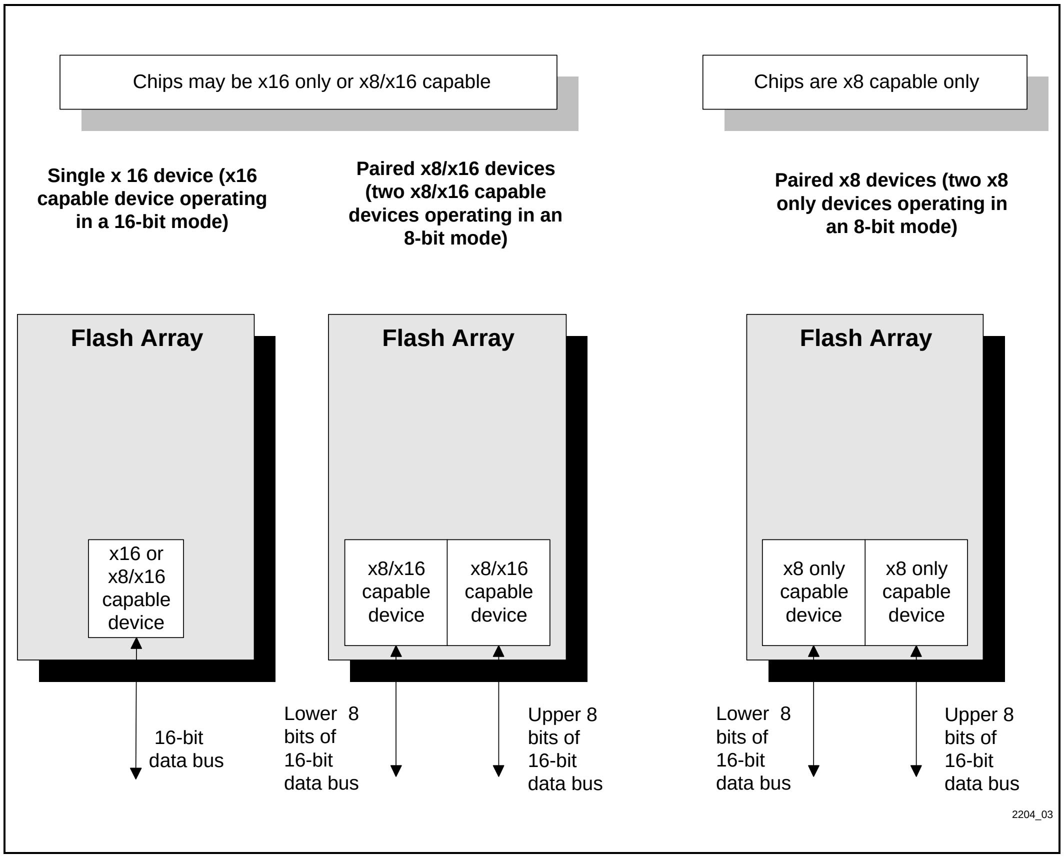

Chips may be x16 only or x8/x16 capable
Chips are x8 capable only
Single x 16 device (x16 capable device operating in a 16-bit mode)
Paired x8/x16 devices (two x8/x16 capable devices operating in an 8-bit mode)
Paired x8 devices (two x8 only devices operating in an 8-bit mode)
Flash Array
Flash Array
Flash Array
x16 or x8/x16 capable device
x8/x16 capable device
x8/x16 capable device
x8 only capable device
x8 only capable device
16-bit data bus
Lower 8 bits of 16-bit data bus
Upper 8 bits of 16-bit data bus
Lower 8 bits of 16-bit data bus
Upper 8 bits of 16-bit data bus
2204_03

Figure 3. Possible Flash Array Configurations

3.2 Read Electronic Databook Information

In a CFI-enabled device, following the “QRY” string is a list of device specific parameters and vendor-specific information, the “Electronic Databook.” Tables 2, 3, and 4 outline the data provided by the device during the CFI query. Software routine QueryCFI (the flowchart is included in *Appendix A1*) reads the following information from the device:

s

Table 1. CFI Query Read

Device type and data bus operating mode	Query start location in maximum device buswidth addresses	Query data with maximum device buswidth addressing (“x” = ASCII equivalent)	Query start address in bytes	Query data with byte addressing
x8 device operating in 8 bit mode	10h	10h: 51h “Q” 11h: 52h “R” 12h: 59h “Y”	10h	10h: 51h “Q” 11h: 52h “R” 12h: 59h “Y”
two x8 devices operating in 8 bit mode (paired chip configuration)	10h	10h: 0051h “Q” 11h: 0052h “R” 12h: 0059h “Y”	20h	20h: 51h “Q” 21h: 51h “Q” 22h: 52h “R” 23h: 52h “R” 24h: 59h “Y” 25h: 59h “Y”
x16 device operating in 16 bit mode	10h	10h: 0051h “Q” 11h: 0052h “R” 12h: 0059h “Y”	20h	20h: 51h “Q” 21h: 00h null 22h: 52h “R” 23h: 00h null
x16 device operating in 8 bit mode	N/A(1)	N/A(1)	20h	20h: 51h “Q” 21h: 51h “Q” 22h: 52h “R”
two x16 devices operating in 8 bit mode (paired chip configuration)	N/A(1)	N/A(1)	40h	40h: 51h “Q” 41h: 51h “Q” 42h: 51h “Q” 43h: 51h “Q” 44h: 52h “R” 45h: 52h “R”
x32 device operating in 32 bit mode	10h	10h: 00000051h “Q” 11h: 00000052h “R” 12h: 00000059h “Y”	40h	40h: 51h “Q” 41h: 00h null 42h: 00h null 43h: 00h null 44h: 52h “R”
x32 device operating in 8 bit mode	N/A(1)	N/A(1)	40h	40h: 51h “Q” 41h: 51h “Q” 42h: 51h “Q” 43h: 51h “Q” 44h: 52h “R”

NOTE:

1. The system must drive the lowest order addresses to access all the device’s array data when the device is configured in x8 mode. Therefore, word addressing where these lower addresses not toggled by the system is “Not Applicable” for x8-configured devices.

Table 2. CFI Query Identification String

Offset	Length (bytes)	Description
10h	03h	Query-unique ASCII string “QRY“
13h	02h	Primary Vendor Command Set and Control Interface ID Code 16-bit ID code defining specific Vendor-specified algorithm *[Refer to CFI Publication 100 for definition of the ID codes]*
15h	02h value = ***P***	Address for Primary Algorithm extended Query table Note: Address 0000h means that no extended table exists
17h	02h	Alternate Vendor Command Set and Control Interface ID Code second vendor-specified algorithm supported by the device *[Refer to CFI Publication 100 for definition of the ID codes]* Note: ID Code = 0000h means that no alternate algorithm is employed
19h	02h value = ***A***	Address for Alternate Algorithm extended Query table Note: Address 0000h means that no alternate extended table exists

NOTES:

1. Offset is the location in memory when using maximum device bus width addressing.
2. The CFI specification allows for replacement of all or part of the standard Query table contents. If the Vendor Primary (or Alternate) Algorithm extended Query table address (P or A) points to any address between 10h and the end of the Flash Geometry Table, the standard Query table contents from that point on are assumed to be replaced by the information defined by the Vendor Primary (or Alternate) Algorithm. Thus, some or all of the standard Query may be replaced. For example, a Vendor Primary (or Alternate) Algorithm extended Query table address of 27h means that the standard Device Geometry definition has been replaced by something which has been defined by the Vendor. The System Interface information at locations 1Bh to 26h may be assumed valid, but the ultimate definition must be described by the particular vendor algorithm. If the Vendor Primary (or Alternate) Algorithm extended Query table address points to an address beyond the end of the Flash Geometry Table, a new table of data is included at that address. The contents of this table are defined by the corresponding Vendor Primary (or Alternate) Algorithm.

Table 3. System Interface Information

Offset	Length (bytes)	Description
1Bh	01h	V_{CC} Logic Supply Minimum Program/Erase voltage bits 7–4 BCD value in volts bits 3–0 BCD value in 100 millivolts
1Ch	01h	V_{CC} Logic Supply Maximum Program/Erase voltage bits 7–4 BCD value in volts bits 3–0 BCD value in 100 millivolts
1Dh	01h	V_{PP} [Programming] Supply Minimum Program/Erase voltage bits 7–4 HEX value in volts bits 3–0 BCD value in 100 millivolts Note: This value must be 0000h if no V_{PP} pin is present
1Eh	01h	V_{PP} [Programming] Supply Maximum Program/Erase voltage bits 7–4 HEX value in volts bits 3–0 BCD value in 100 millivolts Note: This value must be 0000h if no V_{PP} pin is present
1Fh	01h	Typical timeout per single byte/word write (buffer write count = 1), 2^n µs
20h	01h	Typical timeout for maximum-size buffer write, 2^n µs (if supported; 00h = not supported)
21h	01h	Typical timeout per individual block erase, 2^n ms
22h	01h	Typical timeout for full chip erase, 2^n ms (if supported; 00h = not supported)
23h	01h	Maximum timeout for byte/word write, 2^n times typical (offset 1Fh)
24h	01h	Maximum timeout for buffer write, 2^n times typical (offset 20h) (00h = not supported)
25h	01h	Maximum timeout per individual block erase, 2^n times typical (offset 21h)
26h	01h	Maximum timeout for chip erase, 2^n times typical (offset 22h) (00h = not supported)

Table 4. Flash Geometry Information

Offset	Length (bytes)	Description
27h	01h	Device Size = 2^n in number of bytes.
28h	02h	Flash Device Interface description *[Refer to CFI Publication 100]*
2Ah	02h	Maximum number of bytes in buffer write = 2^n.
2Ch	01h	Number of Erase Block Regions within device **bits 7–0 = x** = number of Erase Block Regions Notes: 1. x = 0 means no erase blocking, i.e., the device erases at once in "bulk." 2. x specifies the number of regions within the device containing one or more contiguous Erase Blocks of the same size. For example, a 128-KB device (1 Mb) having blocking of 16 KB, 8 KB, four 2 KB, two 16 KB, and one 64 KB is considered to have 5 Erase Block Regions. Even though two regions both contain 16-KB blocks, the fact that they are not contiguous means they are separate Erase Block Regions. 3. By definition, symmetrically- blocked devices have only one blocking region.
2Dh	04h	Erase Block Region Information **bits 31–16 = z**, where the Erase Block(s) within this Region are (z) times 256 bytes in size. The value z = 0 is used for 128-byte block size. e.g., for 64-KB block size, z = 0100h = 256 => 256 * 256 = 64K **bits 15– 0 = y**, where y+1 = Number of Erase Blocks of identical size within the Erase Block Region: e.g., y = D15-D0 = FFFFh => y+1 = 64K blocks [maximum number] y = 0 means no blocking (# blocks = y+1 = "1 block") Note: y = 0 value must be used with # of block regions of one as indicated by (x) = 0
31h to (k-1)h	04h per entry	Additional Erase Block Region Information, 4 bytes per region Notes: 1. The total number of blocks times individual block size must add up to the device size. 2. The address K is next available Query address at end of the Device Geometry structure. It is the first possible starting address of the optional vendor-specific Query table(s) (i.e., Address "P," the Primary Vendor-specific extended Query table offset, must be ≥ k to not overwrite the existing tables). See note 2 under Table 2 for more information.

3.3 Read Vendor-Specific Extended Query Table

Using data from addresses 15h (Address for Primary Algorithm extended Query table) and possibly 19h (Address for Alternate Algorithm extended Query table), the system software can read more specific information about the flash device (see the *CFI Query Flowchart*, included in *Appendix A1*). Each vendor will have specific data that should be read from the extended query table. Intel defines this data with its SCS. Also, each vendor may locate this table in a different location, so it is important that the software reads the location of the tables from offsets 15h and 19h to determine where (if at all) the extended query data is stored. The Vendor Command Set definition (Intel's SCS) will indicate what data is stored in the extended query table. Table 5 shows the extended table for the Intel devices implementing CFI (all devices implementing CFI will use the extended query table regardless of the command set being used).

Table 5. Intel Primary Algorithm Extended Query Table

Offset	Length (bytes)	Description
(P)h	03h	Primary extended Query table unique ASCII string "PRI"
(P +3)h	01h	Major version number, ASCII
(P +4)h	01h	Minor version number, ASCII
(P +5)h	04h	Optional Feature & Command Support bit 0 Chip Erase Supported (1=yes, 0=no) bit 1 Suspend Erase Supported (1=yes, 0=no) bit 2 Suspend Program Supported (1=yes, 0=no) bit 3 Lock/Unlock Supported (1=yes, 0=no) bit 4 Queued Erase Supported (1=yes, 0=no) *bits 5–31 Reserved for future use; undefined bits should be "0"*
(P +9)h	01h	Supported functions after Suspend Read Array, Status, and Query are always supported during suspended Erase or Program operation. This field defines other operations supported. bit 0 Program supported after Erase Suspend (1=yes, 0=no) *bits 1–7 Reserved for future use; undefined bits should be "0"*
(P +A)h	02h	Block Status Register Mask Defines which bits in the Block Status Register section of Query are implemented. bit 0 Block Status Register Lock Bit [BSR.0] active (1=yes, 0=no) bit 1 Block Status Register Valid Bit [BSR.1] active (1=yes, 0=no) *bits 2–15 Reserved for future use; undefined bits should be "0"*
(P +C)h	01h	V_{CC} Logic Supply Optimum Program/Erase voltage (highest performance) bits 7–4 BCD value in volts bits 3–0 BCD value in 100 millivolts
(P +D)h	01h	V_{PP} [Programming] Supply Optimum Program/Erase voltage bits 7–4 HEX value in volts bits 3–0 BCD value in 100 millivolts *Note: This value must be 0000h if no Vpp pin is present*
(P +E)h	*reserved*	*Reserved for future versions of the SCS Specification*

3.4 Software Branch to Appropriate Routines

Using the information read from location 13h (Primary Vendor Command Set and Control Interface ID code) and possibly from 17h (Alternate Vendor Command Set and Control Interface ID Code), the system determines which set(s) of commands are recognizable by the flash device. Every vendor has defined at least one set of commands that their flash devices accept. Any commands other than those defined will either be rejected or cause unexpected behavior and should therefore be avoided.

Intel has two such sets of commands defined, BCS (Basic Command Set) and SCS (Scaleable Command Set). SCS includes all of the BCS commands plus some new enhanced ones. The low level reference code provided by Intel includes both of these command sets. By using this reference code, current and future flash devices may be controlled using the same system software drivers.

4.0 HOW TO USE COMMAND SETS EFFECTIVELY

Flowcharts for most of the driver software routines necessary to implement the Basic and Scaleable Command Sets are located in *Appendix A2* and *A3* respectively. All of the reference code may be downloaded from the Intel web site from the electronic tools catalog. To ensure compatibility with future CFI compliant devices, all functions of the reference code should be included in a design.

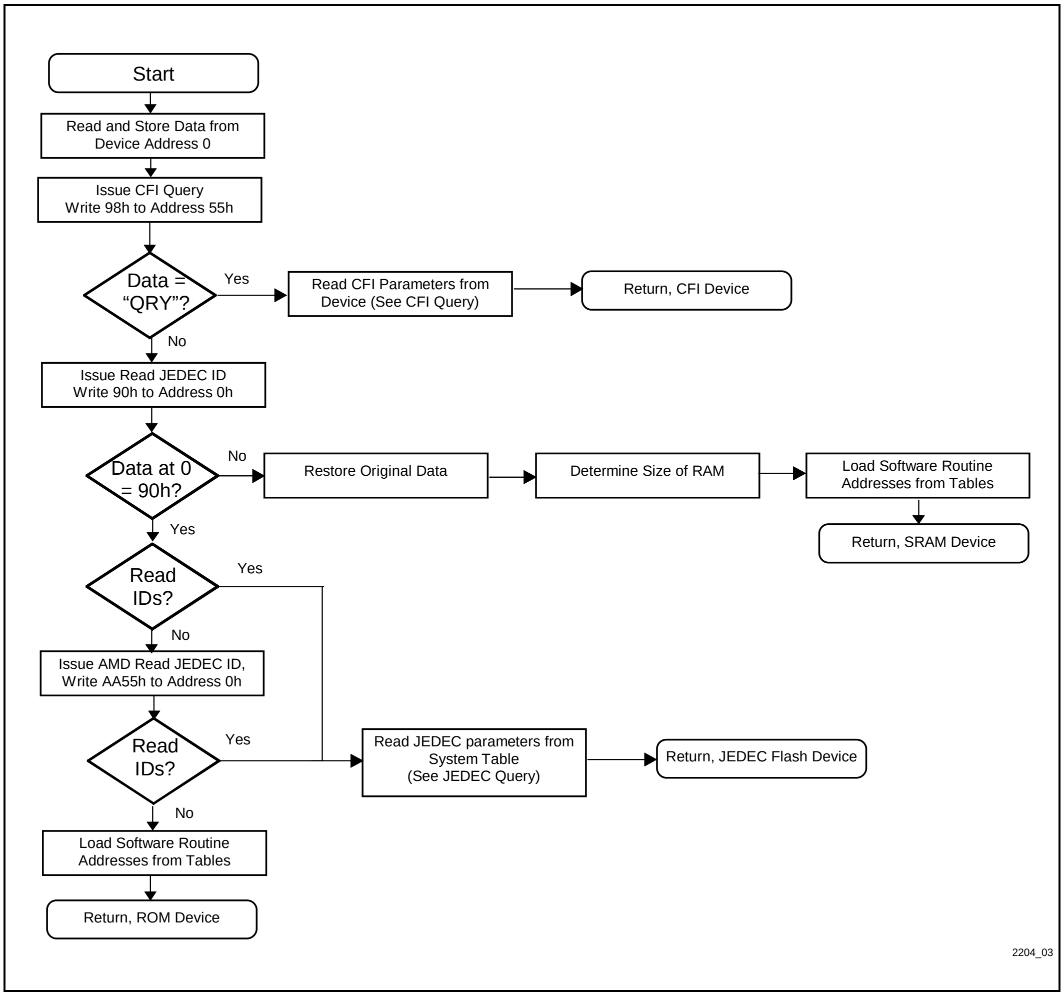
5.0 CONCLUSION

CFI and SCS have been created to allow a system designer the flexibility to design products now that can use both current and future flash memory devices as well as the security of knowing that second source products may be used without system software modifications.

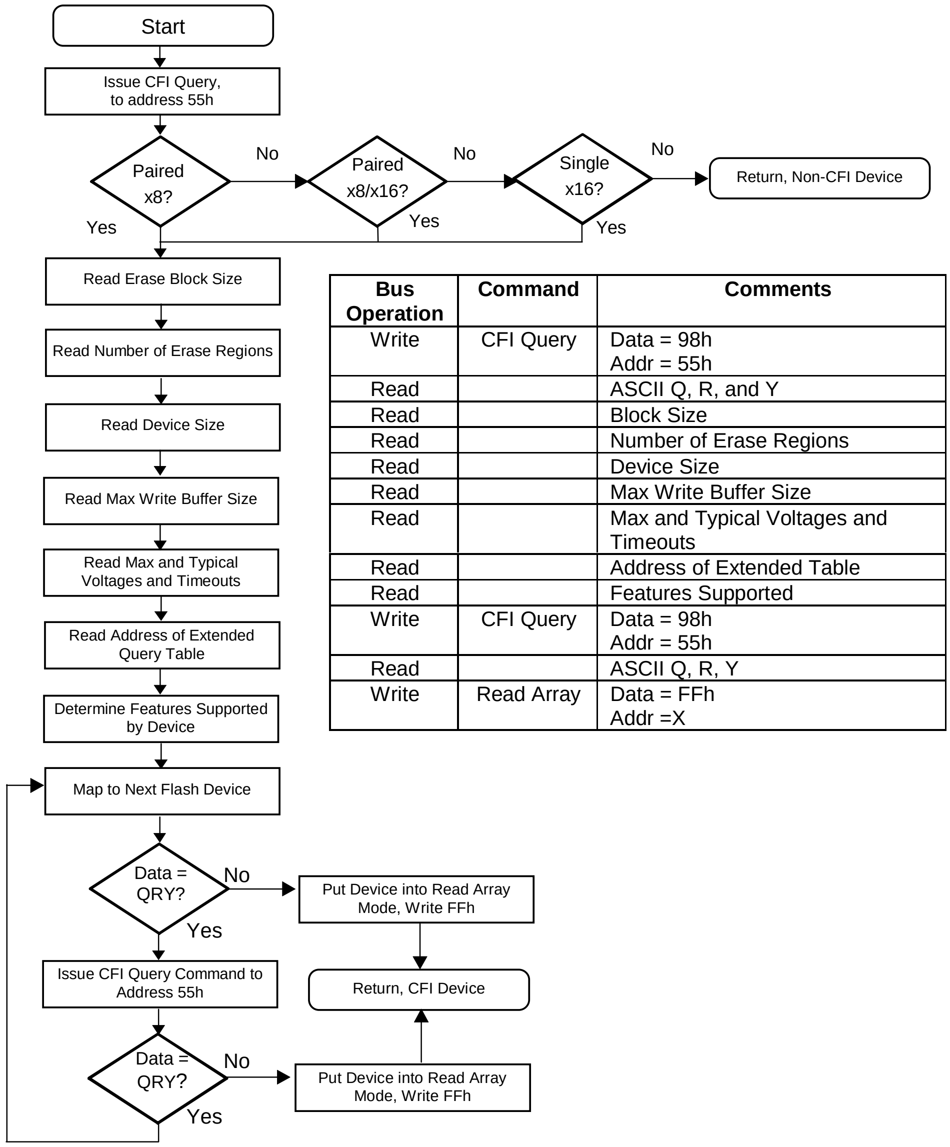
APPENDIX A
FLOWCHARTS

A1 MEMORY HUERISTICS

A1.1 Memory Hueristics Flowchart


Start
Read and Store Data from Device Address 0
Issue CFI Query Write 98h to Address 55h
Data = "QRY"?
Yes
Read CFI Parameters from Device (See CFI Query)
Return, CFI Device
No
Issue Read JEDEC ID Write 90h to Address 0h
Data at 0 = 90h?
No
Restore Original Data
Determine Size of RAM
Load Software Routine Addresses from Tables
Return, SRAM Device
Yes
Read IDs?
Yes
No
Issue AMD Read JEDEC ID, Write AA55h to Address 0h
Read IDs?
Yes
Read JEDEC parameters from System Table (See JEDEC Query)
Return, JEDEC Flash Device
No
Load Software Routine Addresses from Tables
Return, ROM Device
2204_03

A1.2 CFI Query Flowchart


Start
Issue CFI Query, to address 55h
Paired x8?
No
Yes
Paired x8/x16?
No
Yes
Single x16?
No
Yes
Return, Non-CFI Device
Read Erase Block Size
Read Number of Erase Regions
Read Device Size
Read Max Write Buffer Size
Read Max and Typical Voltages and Timeouts
Read Address of Extended Query Table
Determine Features Supported by Device
Map to Next Flash Device
Data = QRY?
No
Yes
Put Device into Read Array Mode, Write FFh
Return, CFI Device
Issue CFI Query Command to Address 55h
Data = QRY?
No
Yes
Put Device into Read Array Mode, Write FFh

Bus Operation	Command	Comments

Write	CFI Query	Data = 98h
		Addr = 55h

Read		ASCII Q, R, and Y
Read		Block Size
Read		Number of Erase Regions
Read		Device Size
Read		Max Write Buffer Size
Read		Max and Typical Voltages and Timeouts
Read		Address of Extended Table
Read		Features Supported

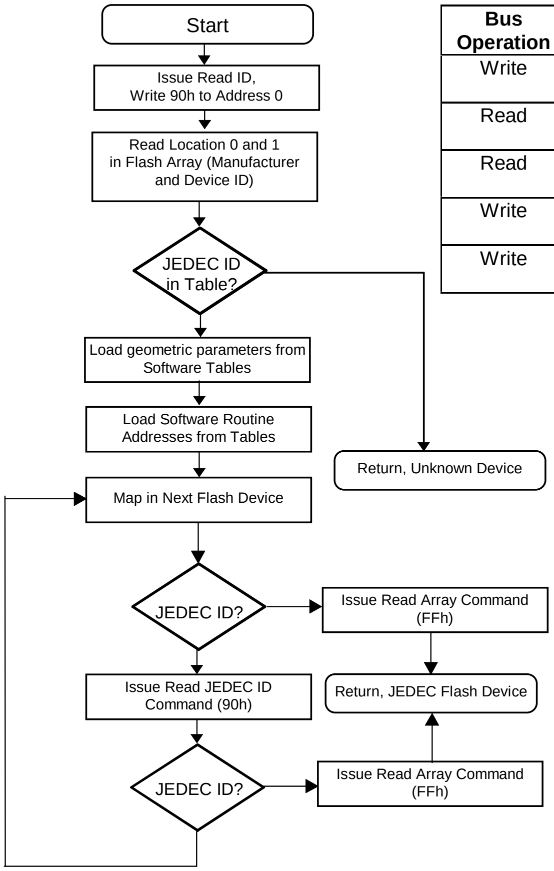
Write	CFI Query	Data = 98h
		Addr = 55h

Read		ASCII Q, R, Y

Write	Read Array	Data = FFh
		Addr =X

2204_04

A1.3 Read JEDEC ID Flowchart


Start
Issue Read ID, Write 90h to Address 0
Read Location 0 and 1 in Flash Array (Manufacturer and Device ID)
JEDEC ID in Table?
Load geometric parameters from Software Tables
Load Software Routine Addresses from Tables
Return, Unknown Device
Map in Next Flash Device
JEDEC ID?
Issue Read Array Command (FFh)
Issue Read JEDEC ID Command (90h)
Return, JEDEC Flash Device
JEDEC ID?
Issue Read Array Command (FFh)

Bus Operation	Command	Comments

Write	Read ID	Data = 90h
		Addr = 0

Read		Data = Manufacturer ID
		Addr = 0

Read		Data = Device ID
		Addr = 1

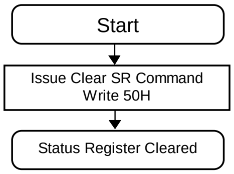
Write	Read ID	Data = 90h
		Addr = 0

Write	Read Array	Data = FFh
		Addr = X

2204_05

A2 BASIC COMMAND SET

A2.1 Clear Status Register Flowchart

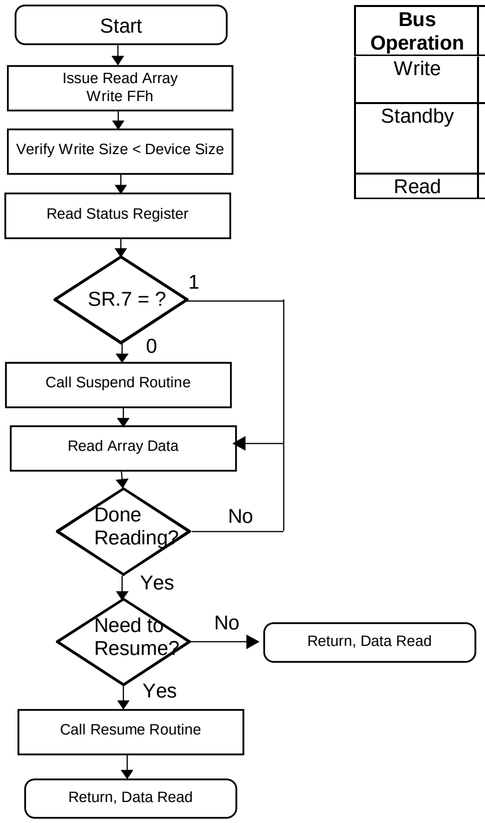


Bus Operation	Command	Comments

Write	Clear Status Register	Data = 50h
		Addr = X

2204_06

A2.2 Read Array Flowchart



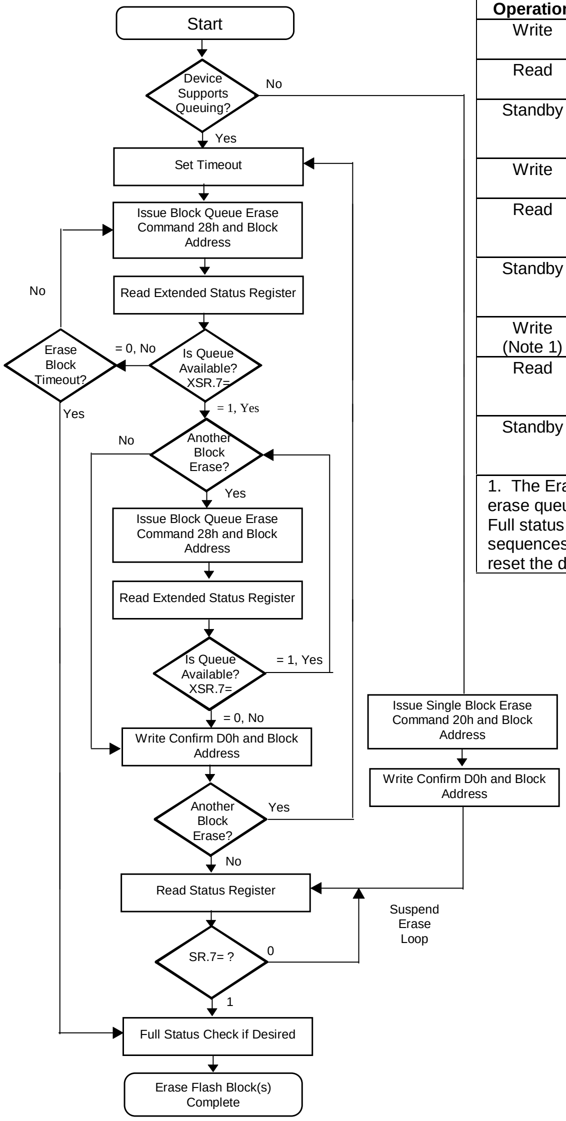
Bus Operation	Command	Comments

Write	Read Array	Data = FFh
		Addr = Address to read

Standby		Check SR.7 1 = WSM ready 0 = WSM busy
Read		Read array locations

2204_07

A2.3 Block Erase Flowchart


Start
Device Supports Queuing?
No
Yes
Set Timeout
Issue Block Queue Erase Command 28h and Block Address
Read Extended Status Register
No
Erase Block Timeout?
= 0, No
Is Queue Available? XSR.7=
= 1, Yes
Yes
No
Another Block Erase?
Yes
Issue Block Queue Erase Command 28h and Block Address
Read Extended Status Register
Is Queue Available? XSR.7=
= 1, Yes
= 0, No
Write Confirm D0h and Block Address
Another Block Erase?
Yes
No
Issue Single Block Erase Command 20h and Block Address
Write Confirm D0h and Block Address
Read Status Register
Suspend Erase Loop
SR.7= ?
0
1
Full Status Check if Desired
Erase Flash Block(s) Complete

Bus Operation	Command	Comments

Write	Erase Block	Data = 28h or 20h
		Addr = Block Address

Read		XSR.7 = valid
		Addr = X

Standby		Check XSR.7 1 = Erase queue available 0 = No Erase queue available

Write	Erase Block	Data = 28h
		Addr = Block Address

Read		SR.7 = valid; SR.6-0 = X
		CE# and OE# low updates SR
		Addr = X

Standby		Check XSR.7 1 = Erase queue available 0 = No erase queue available

Write (Note 1)	Erase Confirm	Data = D0h
		Addr = X

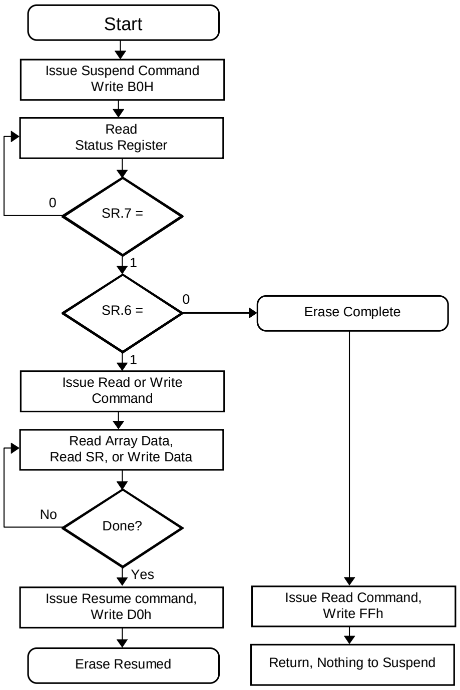
Read		Status Register data CE# and OE# low updates SR Addr = X
Standby		Check SR.7 1 = WSM ready 0 = WSM busy

1. The Erase Confirm byte must follow Erase setup when the erase queue status (XSR.7) = 0.

Full status check can be done after all Erase and Program sequences complete. Write FFh after the last operation to reset the device to read array mode.

2204_08

A2.4 Erase Suspend/Resume Flowchart



Bus Operation	Command	Comments

Write	Erase Suspend	Data = B0h
		Addr = X

Read		SR.7=valid; SR.6-0=X
		CE# & OE# low updates SR
		Addr = X

Standby		Check SR.7 1 = WSM ready 0 = WSM busy
Standby		Check SR.6 1 = Erase suspended 0 = Erase in progress or complete

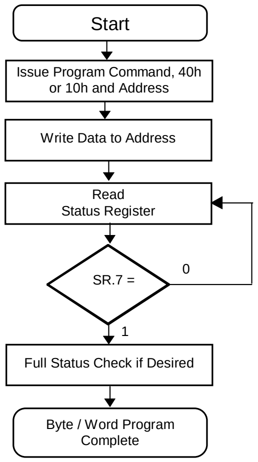
Write	Read Array, Read SR, or Write	Data = FFh
		Addr = X

Read/Write		Read/Write array locations other than those being erased or Read SR

Write	Resume	Data = D0h
		Addr = X

2204_09

A2.5 Single Byte Program (Write) Flowchart


Start
Issue Program Command, 40h or 10h and Address
Write Data to Address
Read Status Register
SR.7 =
0
1
Full Status Check if Desired
Byte / Word Program Complete

Bus Operation	Command	Comments

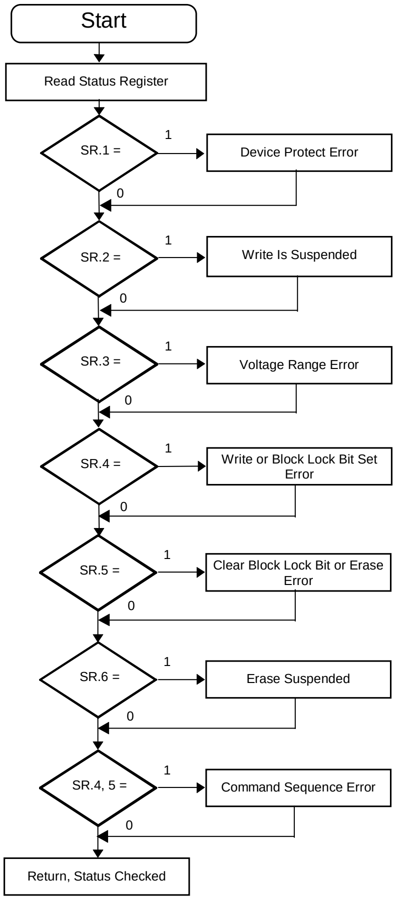
Write	Setup Byte / Word Program	Data = 40h or 10h
		Addr = Location to Be Programmed

Write	Byte / Word Program	Data = Data to Be Programmed
		Addr = Location to be Programmed

Read		Status Register Data
Standby		Check SR.7 1 = WSM Ready 0 = WSM Busy
Repeat for subsequent program operations. Full status register check can be done after each program operation or after a sequence of Programming. Write FFh after the last program operation to place the device in read array mode.		

2204_10

A2.6 Full Status Check Flowchart

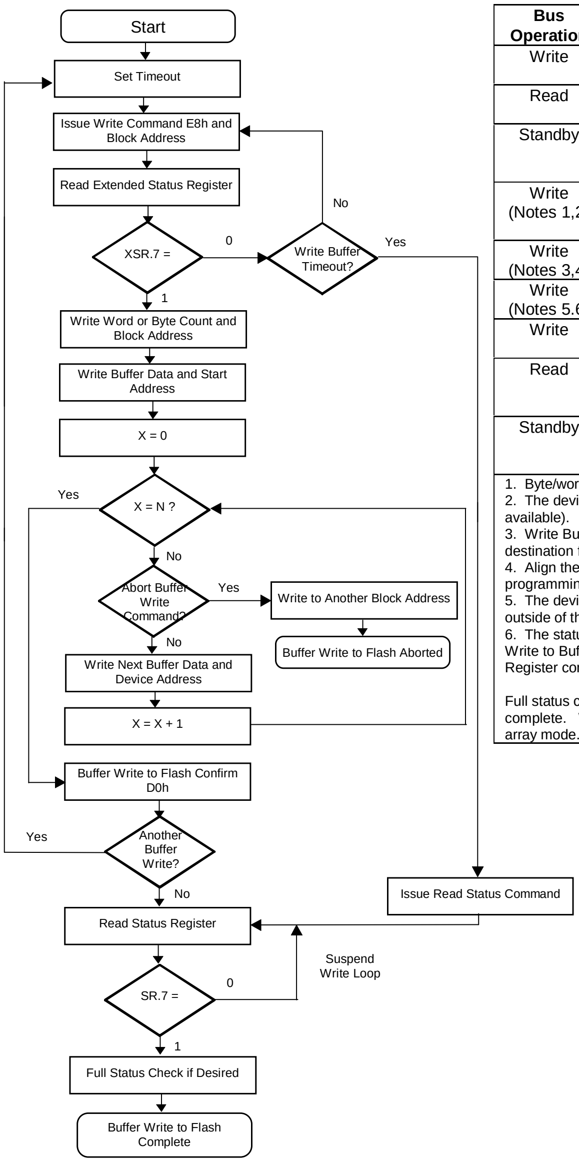


Bus Operation	Command	Comments
Standby		Check SR.3 1 = Programming Voltage Error Detect
Standby		Check SR.1 1 = Device Protect Detect
Standby		Check SR.4,5 Both 1 = Command Sequence Error
Standby		Check SR.5 1 = Clear Block Lock-Bits Error
SR.5, SR.4, SR.3, and SR.1 are only cleared by the Clear Status Register command. If error is detected, clear the status register before attempting retry or other error recovery.		

2204_11

A3 SCALEABLE COMMAND SET

A3.1 Write to Buffer Flowchart


Start
Set Timeout
Issue Write Command E8h and Block Address
Read Extended Status Register
XSR.7 =
0
No
Write Buffer Timeout?
Yes
1
Write Word or Byte Count and Block Address
Write Buffer Data and Start Address
X = 0
Yes
X = N ?
No
Abort Buffer Write Command?
Yes
Write to Another Block Address
Buffer Write to Flash Aborted
No
Write Next Buffer Data and Device Address
X = X + 1
Buffer Write to Flash Confirm D0h
Yes
Another Buffer Write?
No
Issue Read Status Command
Read Status Register
Suspend Write Loop
SR.7 =
0
1
Full Status Check if Desired
Buffer Write to Flash Complete

Bus Operation	Command	Comments

Write	Write to Buffer	Data = E8h
		Addr = Block Address

Read		XSR.7 = valid; XSR.6-0 = X;
		Addr = X

Standby		Check XSR.7 1 = Write buffer ready 0 = No write buffer ready

Write (Notes 1,2)		Data = N = word/byte count
		N = 0 corresponds to count = 1
		Addr = Block Address

Write (Notes 3,4)		Data = write buffer data
		Addr = device address

Write (Notes 5.6)		Data = write buffer data
		Addr = device address

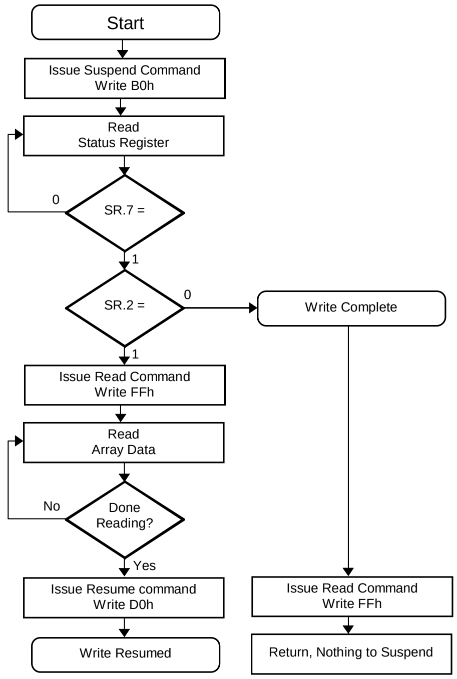
Write	Write Confirm	Data - D0h Addr = X
Read		Status Register Data CE# and OE# low updates SR Addr = X
Standby		Check SR.7 1 = WSM ready 0 = WSM busy

1. Byte/word count values on DQ_{0-7} are loaded into the count register.
2. The device now outputs the status register when read (XSR is no longer available).
3. Write Buffer contents will be programmed at the device start address or destination flash address.
4. Align the start address on a Write Buffer boundary for maximum programming performance.
5. The device aborts the Write to Buffer command if the current address is outside of the original block address.
6. The status register indicates an "improper command sequence" if the Write to Buffer command is aborted. Follow this with a Clear Status Register command.

Full status check can be done after all Erase and Write sequences complete. Write FFh after the last operation to reset the device to read array mode.

2204_12

A3.2 Program (Write) Suspend/Resume Flowchart



Bus Operation	Command	Comments

Write	Write Suspend	Data = B0h
		Addr = X

Read		SR.7=valid; SR.6-0=X
		CE# & OE# low updates SR
		Addr = X

Standby		Check SR.7 1 = WSM ready 0 = WSM busy
Standby		Check SR.2 1 = Write suspended 0 = Write complete

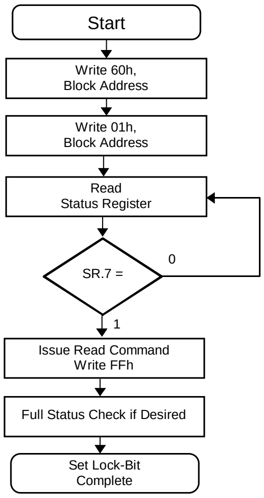
Write	Read Array	Data = FFh
		Addr = X

Read		Read array locations other than those being written

Write	Resume	Data = D0h
		Addr = X

2204_13

A3.3 Block Lock Bit Set Flowchart



Bus Operation	Command	Comments

Write	Set Block Lock-Bit Setup	Data = 60h
		Addr = Block Address (Block)

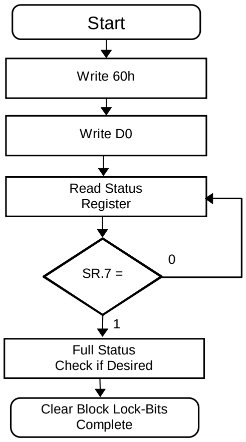
Write	Set Block Lock-Bit Confirm	Data = 01h
		Addr = Block Address (Block)

Read		Status Register Data
Standby		Check SR.7 1 = WSM Ready 0 = WSM Busy

Repeat for subsequent lock-bit set operations.
Full status check can be done after each lock-bit set operation or after a sequence of lock-bit set operations.
Write FFh after the last lock-bit set operation to place device in read array mode.

2204_14

A3.4 Block Lock Bit Reset Flowchart



Bus Operation	Command	Comments

Write	Clear Block Lock-Bits Setup	Data = 60h
		Addr = X

Write	Clear Block Lock-Bit Confirm	Data = DOh
		Addr = X

Read		Status Register Data
Standby		Check SR.7 1 = WSM Ready 0 = WSM Busy

Write FFh after the last lock-bit set operation to place device in read array mode.

2204_15