JEDEC STANDARD

Serial Flash Discoverable Parameters (SFDP)

JESD216B

(Revision of JESD216A, July 2013)

MAY 2014

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION

NOTICE

JEDEC standards and publications contain material that has been prepared, reviewed, and approved through the JEDEC Board of Directors level and subsequently reviewed and approved by the JEDEC legal counsel.

JEDEC standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for use by those other than JEDEC members, whether the standard is to be used either domestically or internationally.

JEDEC standards and publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC standards or publications.

The information included in JEDEC standards and publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC standard or publication may be further processed and ultimately become an ANSI standard.

No claims to be in conformance with this standard may be made unless all requirements stated in the standard are met.

Inquiries, comments, and suggestions relative to the content of this JEDEC standard or publication should be addressed to JEDEC at the address below, or refer to www.jedec.org under Standards and Documents for alternative contact information.

Published by
©JEDEC Solid State Technology Association 2014
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

PRICE: Contact JEDEC

Printed in the U.S.A.

SERIAL FLASH DISCOVERABLE PARAMETERS (SFDP) STANDARD

Contents

Foreword

This standard was prepared by the JEDEC SFDP Task Group authorized by the JC-42.4 Committee Chairman. It was derived from prior work done by Intel on their 'Serial Flash Discoverable Parameters Guidelines' document.

The intended audience is serial flash vendors and engineers writing device drivers for SFDP compliant serial flash devices.

The participating SFDP TG members were volunteers from AMD, ASPEED, Emulex, HP, Intel, Macronix, Micron, Microchip, Sanyo, Spansion, and Winbond.

Introduction

The Serial Flash Discoverable Parameter (SFDP) standard provides a consistent method of describing the functional and feature capabilities of serial flash devices in a standard set of internal parameter tables. These parameter tables can be interrogated by host system software to enable adjustments needed to accommodate divergent features from multiple vendors.

The SFDP standard defines a common parameter table describing important device characteristics and serial access methods used to read the parameter table data. Special Function parameter tables for erase sector address map and 4-byte address instructions are added in this revision of this standard. Additional parameter headers and tables can be specified by future revisions of this standard or by flash vendors and are optional.

SERIAL FLASH DISCOVERABLE PARAMETERS (SFDP) STANDARD

(From JEDEC Board Ballot JCB-14-08, formulated under the cognizance of the JC-42.4 Committee on Nonvolatile Memory.)

1 Scope

The SFDP standard defines the structure of the SFDP database within the memory device and methods used to read its data.

The JEDEC-defined header with Parameter ID FF00h and the related Basic Parameter Table is mandatory. This header and table provide basic information for a Serial Peripheral Interface (SPI) protocol memory. Additional headers and tables are optional.

The read command protocol using various I/O modes and standard clock rate are specified. The device electrical parameters are not specified.

2 Normative reference

The following normative documents contain provisions that, through reference in this text, constitute provisions of this standard. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

1. JEP106, Standard Manufacturers Identification Code (contact jedec.org for the latest revision of this document)

2. NIST SP800-147, BIOS Protection Guidelines (http://csrc.nist.gov/publications/nistpubs/)

3 Terms and definitions

For the purposes of this standard, the following terms and definitions apply:

00b: The ‘b’ suffix indicates the ‘00’ digits are a binary representation of the number.

00h: The ‘h’ suffix indicates the ‘00’ digits are a hexadecimal representation of the number.

0x00: The ‘0x’ prefix indicates the ‘00’ digits are a hexadecimal representation of the number. This form is used in the ‘C’ sample code in Annexes.

Address: The three or four byte value following some instructions that is used to select a location within an address space of the flash memory.

3 Terms and definitions (cont'd)

Basic Parameter Table: The table pointed to by Parameter ID FF00h. Contains general information about the flash device's capabilities.

Block: A group of contiguous sectors.

Command: The combination of the instruction, address, optional mode bits, wait states, and data cycles used to initiate functions or transfer information between the controller and the serial flash.

Controller: The serial bus master

Double Transfer Rate (DTR): Instruction, address, and/or data may be input or output on both the rising and falling edges of the clock.

Dummy Cycles: Clock cycles during which no data is transferred to or from a memory.

DWORD: Four consecutive 8-bit bytes used as the basic 32-bit building block for headers and parameter tables.

Instruction: The one byte code used to initiate a function in the serial flash or identify the type of information transfer between the controller and the serial flash.

Mode Bits: Optional control bits that follow the address bits. These bits are driven by the controller if they are specified.

Wait States: Required clock cycles between the address bits or optional mode bits and the start of data when reading from the flash device. Some device data sheets describe these as dummy cycles because no information is transferred between the controller and memory during these cycles. Neither controller nor memory are required to drive the data lines during these cycles.

Read Latency: On flash read instructions, the total number of clocks between end of address and the start of data. The sum of clocks for mode bits and clocks for wait states equals the Read Latency.

Sector: The minimum granularity - size and alignment - of an area that can be erased in the data array of a flash memory device. Different areas within the address range of the data array may have a different minimum erase granularity (sector size).

(x-y-z): Command mode nomenclature used to indicate the number of active pins used for the instruction (x), address (y), and data (z). At the present time, the only valid Read SFDP command modes are: (1-1-1), (2-2-2), and (4-4-4)

4 Read SFDP Command Protocol

4.1 Instruction

The Read SFDP instruction code is 5Ah.

4.2 Address

Indicates the starting byte address in the SFDP area and is always expressed as a three byte field.

4.3 Wait States

Following the address, eight clocks are required before valid data is clocked out.

4.4 Clock Rate

SFDP compliant devices must support 50 MHz operation for the Read SFDP command (instruction 5Ah). Devices may support a wider frequency range, but a controller can always run SFDP cycles at 50 MHz or less and get valid results.

4.5 Command Modes

The Read SFDP command can be used with device supported modes of (1-1-1), (2-2-2), or (4-4-4), but the instruction (5Ah), address (24 bits), eight wait states, and 50 MHz requirements remain the same. Support for SFDP does not imply or require that the flash device support 2-2-2 or 4-4-4 mode. If the controller knows a priori the mode in which the flash device is configured, then it can issue the Read SFDP command in that mode. If the controller does not know, then a suggested algorithm is to try to read the SFDP signature (see 6.1) in 4-4-4 mode, if that fails try 2-2-2 mode, and if that fails try 1-1-1 mode.

Timing Diagram Signal Definitions:

- S# = Select, low active. Memory device selection signal also often referred to as Chip Select or Chip Enable
- C = Clock. Serial clock to the memory also often referred to as SCLK.
- DQ0 = Data input or output zero. The least significant memory data input or output also often referred to as IO0 or Serial Input (SI) when not used for two or four bit data I/O.
- DQ1 = Data input or output one. The next most significant memory data input or output above DQ0, also often referred to as IO1 or Serial Output (SO) when not used for two or four bit data I/O.
- DQ2 = Data input or output two. The next most significant memory data input or output above DQ1, also often referred to as IO2 or Write Protect, low active (WP#) when not used for four bit data I/O.
- DQ3 = Data input or output three. The most significant memory data input or output, also often referred to as IO3 or Hold, low active (HOLD#) when not used for four bit data I/O.

4.5.1 Read SFDP (1-1-1) Mode

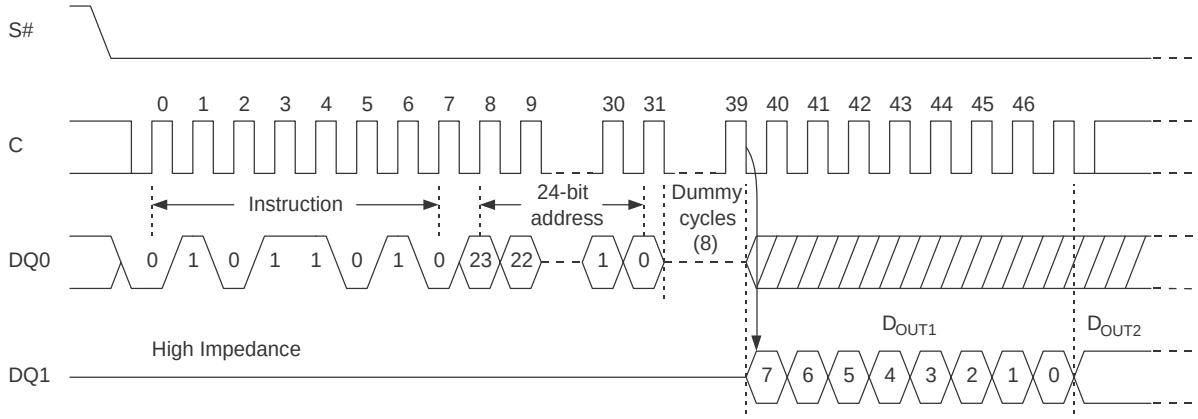

Figure 1 — Read SFDP (1-1-1) Mode Timing Diagram

4.5.2 Read SFDP (2-2-2) Mode

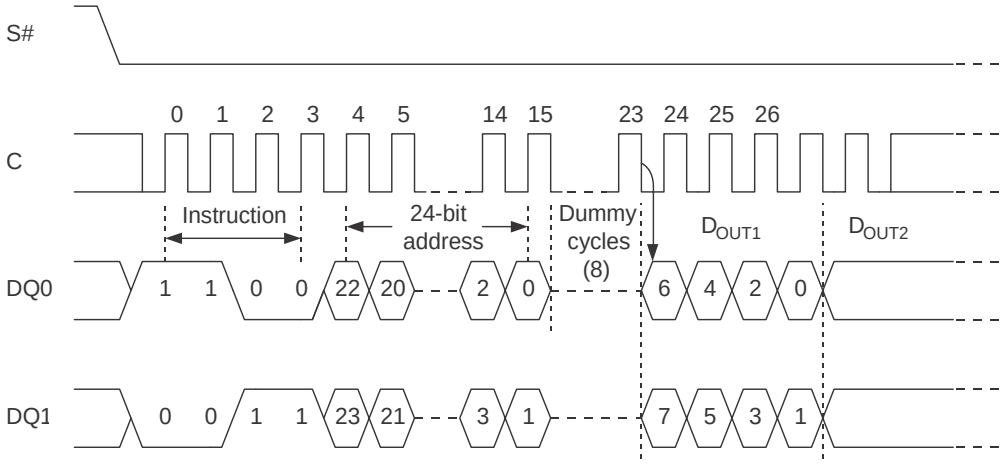

Figure 2 — Read SFDP (2-2-2) Mode Timing Diagram

4.5.3 Read SFDP (4-4-4) Mode

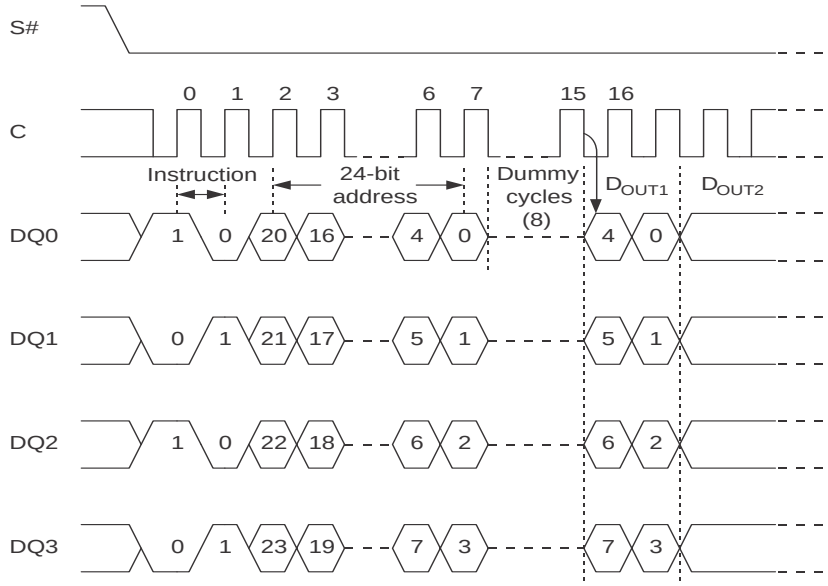

Figure 3 — Read SFDP (4-4-4) Mode Timing Diagram

5 Read SFDP Behavior

5.1 Security

The SFDP and flash memory address ranges must never overlap. This ensures that address range checking the controller may perform to prevent access to security keys or other sensitive information stored in flash cannot be bypassed. Also, for PC BIOS applications non-overlap is required to comply with NIST SP800-147.

Addresses beyond the end of the SFDP tables must not alias into the flash memory. Regardless of the implementation, writes to SFDP tables must be permanently disabled before the memory device is released to a customer by the memory vendor factory.

5.2 Reset and Hold Functions

Reset and Hold functionality will be available during the Read SFDP command if the memory device command mode supports these features.

5.3 Read Wrap

Not supported with the Read SFDP command--even when a memory device defaults to Read Wrap-around mode for other read commands. Only continuous (sequential) read is supported with the Read SFDP command.

5.4 SFDP Address Boundary Wrap

Device behavior when the Read SFDP command crosses the SFDP structure boundary is not defined except for the security restriction specified in 5.1. There is no requirement for the address counter to wrap back to the beginning of the structure and the data read after that point is not specified.

5.5 Reserved SFDP Locations

The content of reserved SFDP locations (memory within the SFDP address space that has not yet been defined or used) is not specified, but recommended to be all FFh.

6 SFDP Database

6.1 SFDP Header Structure

The format of the SFDP header is shown in Figure 4.

	[31:24]	[23:16]	[15:8]	[7:0]	Hex byte location
SFDP Header	Serial Flash Discoverable Parameters (SFDP) Signature = 50444653h				[3h:0h]
	Byte 3 = "P"	Byte 2 = "D"	Byte 1 = "F"	Byte 0 = "S"	
	Unused (set to FFh)	Number of Parameter Headers (NPH)	SFDP Major Revision	SFDP Minor Revision	[7h:4h]
1st Parameter Header	Parameter Length (in double words)	Parameter Major Revision	Parameter Minor Revision	Parameter ID LSB JEDEC ID (00h)	[Bh:8h]
	Parameter ID MSB JEDEC ID (FFh)	Parameter Table Pointer (byte address)			[Fh:Ch]
2nd Parameter Header (optional)	Parameter Length (in double words)	Parameter Major Revision	Parameter Minor Revision	Parameter ID LSB	[13h:10h]
	Parameter ID MSB	Parameter Table Pointer (byte address)			[17h:14h]
		...			
Nth Parameter Header (optional)	Parameter Length (in double words)	Parameter Major Revision	Parameter Minor Revision	Parameter ID LSB	
	Parameter ID MSB	Parameter Table Pointer (byte address)			

Figure 4 — Overall Header Structure

6.2 SFDP Header

The SFDP Header is located at address 000000h of the SFDP data structure. It identifies the SFDP Signature, the number of parameter headers, and the SFDP revision numbers.

Both the SFDP header and the individual parameter table headers include Major and Minor revision numbers.

Major revisions require code (BIOS/firmware) or hardware change to get previously defined discoverable parameters. For example, changes that reorganize previously defined fields.

Minor revisions are changes that define previously reserved fields, add fields to the end, or that clarify definitions of existing fields.

JESD216B maintains backwards compatibility with JESD216A, defines previously reserved bits in DWORD 15 of the Basic Flash Parameter Table and adds optional Function Specific table definitions.

The revision numbers of vendor-defined and Function Specific tables follow the same guidelines as the Basic Parameter Table.

Major vs. Minor Revisions

Major revisions indicate that the table structure has changed and may not be backwards compatible with software written for an earlier revision of SFDP.

Increments of minor revisions indicate that fields or DWORDS have been added but the definition of existing fields is unchanged. Software should accept any minor revision number and read only the number of parameter table fields that the software requires. Users should accept any minor reviion greater than or equal to the current revision. All fields in current tables will be unchange in subsequent minor revisions.

6.2.1 SFDP Header: 1st DWORD

Bits	Description
31:0	**SFDP Signature** Allows a user to know that the information is valid. Signature[31:0]: 50444653h

6.2 SFDP Header (cont'd)

6.2.2 SFDP Header: 2nd DWORD

Bits	Description
31:24	**Unused** Contains FFh and can never be changed.
23:16	**Number of Parameter Headers (NPH)** Specifies the number of parameter headers in the SFDP data structure. This number is 0-based. Therefore, 0 indicates 1 parameter header. The value of this field is not defined by this standard. It is dependent on the number of tables a vendor implements.
15:8	**SFDP Major Revision Number** This 8-bit field indicates the major revision number of this standard. The value in this field is 1 for devices which implement the JESD216B revision. NOTE The value of this field may only be changed by an update to the JESD216 standard.
7:0	**SFDP Minor Revision Number** This 8-bit field indicates the minor revision number of this standard. The value in this field is 6 for devices which implement the JESD216B revision. NOTE The value of this field may only be changed by an update to the JESD216 standard.

6.3 Parameter Headers

Each Parameter Header identifies the size, location, revision, ownership and function of their associated parameter tables. Parameter table ownership will be either JEDEC (via this standard) or an individual vendor (via vendor specific documentation).

Multiple parameter headers can be specified with each parameter header being 2 DWORDs (64-bits). The first parameter header is mandatory, is defined by this standard, and starts at byte offset 08h. If a vendor chooses to include multiple revisions of the Basic Parameter Table they may do so provided the table headers are in order starting with the oldest version. The total number of parameter headers is specified in the NPH field of the SFDP header, see 6.2. All subsequent parameter headers need to be contiguous and may be specified by JEDEC or by vendors using the same structure (shown in Figure 4). Minor revisions may overlap earlier revisions by starting at the same address as an earlier revision but have additional length for parameters added in the later revision. This allows the use of legacy parameters without the need to repeat them in each new minor revision.

6.3 Parameter Headers (cont'd)

6.3.1 Parameter Header: 1st DWORD

Bits	Description
31:24	**Parameter Table Length** This field specifies how many DWORDs are in the Parameter table. NOTE This field is 1's based. Therefore, 1 indicates 1 DWORD.
23:16	**Parameter Table Major Revision Number*** This 8-bit field indicates the major revision number of the associated parameter table. NOTE Major Revision starts at 01h. The Major Revision of JEDEC defined parameter tables can only be modified by updates to this standard. The Major Revision of a vendor-specified table is controlled by that vendor.
15:8	**Parameter Table Minor Revision Number*** This 8-bit field indicates the minor revision number of the associated parameter table. NOTE Minor Revision starts at 00h. The Minor Revision of the JEDEC owned parameter tables can only be modified by updates to this standard. The Minor Revision of a vendor-specified table is controlled by that vendor.
7:0	**Parameter ID LSB:** Refer to Definition of Parameter ID Field in 6.3.3.
* See "Major vs. Minor Revisions" in 6.2	

6.3.2 Parameter Header: 2nd DWORD

Bits	Description
31:24	**Parameter ID MSB** Refer to Definition of Parameter ID Field in 6.3.3.
23:0	**Parameter Table Pointer (PTP)** This address specifies the start of this header's Parameter Table in the SFDP structure. This is a byte address and must be DWORD-aligned.

6.3.3 Definition of Parameter ID Field

The Parameter ID indicates the parameter table ownership and type.

Parameter ID MSB	Parameter ID LSB	type	owner
01h – 7Fh	odd parity	Vendor specific	Vendor
01h – 7Fh	even parity	Function specific	Vendor
80h – FFh	even parity	Function specific	JEDEC JC42.4
FFh	00h	Basic Parameter Table	JEDEC JC42.4

Function Specific tables may be defined by the JC-42.4 Committee or a manufacturer (vendor). The purpose of the Function Specific tables is to allow development of features and associated parameter tables common to multiple manufacturers, prior to the parameter tables being incorporated into the next revision of JESD216. Allocation of IDs for Function Specific tables is requested through the JEDEC office, see Annex C.

6.3.3 Definition of Parameter ID Field (cont'd)

Vendor Specific table structure is defined by the identified device vendor. The parameter table ID field identifies the vendor that owns the table definition.

The original JESD216 specification used only a one byte ID field to identify the parameter table owner. JESD216 revsion A expanded the ID field to two bytes, MSB and LSB, because a single byte is insufficient to uniquely identify all manufacturers (vendors). The original single byte parameter ID is now referred to as the parameter ID LSB.

The Parameter ID LSB value of 00h is reserved for the Basic Parameter Table defined by this standard. For backwards compatibility the MSB is FFh when the LSB is 00h and LSB 00h shall not be used when the MSB is any value other than FFh. This is because some legacy systems using the original JESD216 may ignore the MSB and assume any parameter ID with LSB of 00h is the Basic Parameter Table.

Parameter IDs with:

- An MSB of 01h though 7Fh indicates a Vendor Owned table and provides the bank number of a JEDEC JEP106 assigned Manufacturer ID.
  - 00h is reserved because JEP106 bank numbering begins at 01h
  - The JEP106 Manufacturer's Identification Code LSB is an eight (8) bit field, consisting of seven (7) data bits plus one (1) odd parity bit in the most significant bit position. A Parameter ID LSB with odd parity signifies a Manufacturer's Identification Code and a Parameter ID LSB with even parity signifies a Function Specific table.
- An MSB of 80h through FFh indicates a Function Specific or Basic Parameter Table defined by the JEDEC 42.4 committeeThe LSB is used to identify the Function Specific table type
  - Any parameter ID with LSB 00h identifies the SPI protocol Basic Parameter Table.
  - LSB values with even parity are Function Specific tables that are defined by the JC-42.4 committee.
  - LSB values with odd parity are illegal to prevent any confusion with JEP106 Manufacturer ID values.

6.3.3.1 Function Specific parameter table ID assignments:

Description	ID (Hex)
Basic SPI protocol	FF00
Sector map	FF81
Replay Protected Monotonic Counters	FF03
4-byte Address Instruction Table	FF84
Reserved for next Function Specific Table assignment	FF05

6.3.4 Example SFDP Headers (Note the definition and discussion of Parameter Headers in 6.3)

Figure 5 shows an example of a basic SFDP Header (major revision 1, minor revision 6): one Parameter Header, Parameter Table length of 16 DWORDs, 1st Parameter Header Revision 1.6, JEDEC ID of FF00h, and the Parameter Table Pointer pointing to location 000010h. This header is backwards compatible with previous generations of this standard, therefore devices are not required to contain headers with prior revision numbers.

6.3.4 Example SFDP Headers (cont'd)

In Figure 6, the first parameter header points to a version 1.0 format Basic Parameter Table of 9 DWORDs starting at location 100h and the second parameter headers both points to a separate version 1.6 format Basic Parameter Table of 16 DWORDs starting at location 200h.

Figure 7 adds one of the Optional Function Specific headers introduced in JESD216B. This third header indicates that this device supports 4-byte address instructions functionality.

.

In Figure 6 and Figure7, devices that do not have the functionality of these features may not contain these additional parameters.

Example calculation using the Parameter Table Pointer (PTP):

- The PTP is a byte address. The fields within this document are defined in terms of DWORDS.
- To calculate the byte address of a particular field, given a PTP of 100h:
- The SFDP byte address of DWORD 3 = 100h + ((3-1) * 4) = 100h + 2 *4 = 100h + 8 = 108h

	[31:24]	[23:16]	[15:8]	[7:0]	Hex Byte Location
SFDP Header	50h	44h	46h	53h	< [3h:0h]
	FFh	00h	01h	06h	< [7h:4h]
1st Parameter Header	10h	01h	06h	00h	< [Bh:8h]
	FFh	00h	00h	10h	< [Fh:Ch]

Figure 5 — Example SFDP Header with single Basic Parameter Table

	[31:24]	[23:16]	[15:8]	[7:0]	Hex Byte Location
SFDP Header	50h	44h	46h	53h	< [3h:0h]
	FFh	01h	01h	06h	< [7h:4h]
1st Parameter Header	09h	01h	00h	00h	< [Bh:8h]
	FFh	00h	01h	00h	< [Fh:Ch]
2nd Parameter Header	10h	01h	06h	00h	< [13h:10h]
	FFh	00h	02h	00h	< [17h:14h]

Figure 6 — Example SFDP Header with two Basic parameter Tables

6.3.4 Example SFDP Headers (cont'd)

	[31:24]	[23:16]	[15:8]	[7:0]	Hex Byte Location
SFDP Header	50h	44h	46h	53h	< [3h:0h]
	FFh	02h	01h	06h	< [7h:4h]
1st Parameter Header	09h	01h	00h	00h	< [Bh:8h]
	FFh	00h	01h	00h	< [Fh:Ch]
2nd Parameter Header	10h	01h	06h	00h	< [13h:10h]
	FFh	00h	02h	00h	< [17h:14h]
3rd Parameter Header	02h	01h	00h	84h	< [1Bh:18h]
	FFh	00h	02h	80h	< [1Fh:1Ch]

Figure 7 — Example SFDP Header with two basic Parameter Tables and one optional (4-Byte Address) Function Specific Table

6.4 JEDEC Basic Flash Parameter Header and Table

Parameter tables contain coded information describing the features and capabilities of the serial flash. The first parameter table as defined by JEDEC is mandatory and its starting address is specified by the PTP field of the 1st Parameter Header. This table identifies some of the basic features of SPI protocol flash memory devices.

6.4.1 JEDEC Basic Flash Parameter Header: 1st DWORD

Bits	Description
31:24	**Parameter Table Length** This field specifies how many DWORDs are in the Parameter table. NOTE This field is 1's based. Therefore, 1 indicates 1 DWORD.
23:16	**Parameter Table Major Revision Number** This 8-bit field indicates the major revision number of the parameter table. The value in this field is 1 for this table defined by JESD216B revision. NOTE The Major Revision of JEDEC defined parameter tables can only be modified by updates to this standard.
15:8	**Parameter Table Minor Revision Number** This 8-bit field indicates the minor revision number of the Sector Map parameter table. The value in this field is 6 for this table defined by JESD216B revision. NOTE The Minor Revision of the JEDEC owned parameter tables can only be modified by updates to this standard.
7:0	**Parameter ID LSB** The JEDEC Basic Flash Parameter Table is assigned the ID LSB of 00h.

6.4.2 JEDEC Basic Flash Parameter Header: 2nd DWORD

Bits	Description
31:24	**Parameter ID MSB** The JEDEC Basic Flash Parameter Table is assigned the ID MSB of FFh.
23:0	**Parameter Table Pointer (PTP)** This address specifies the start of this header's Parameter Table in the SFDP structure. The address is in terms of bytes and must be DWORD-aligned.

6.4.3 JEDEC Basic Flash Parameter Table Overview

DWORD	Description
1	Uniform 4KB Sectors, Write Buffer Size, Volatile Status Register, Fast Read Support (1-1-2) (1-2-2) (1-4-4)(1-1-4), Number of Address Bytes, DTR Support
2	Memory Density
3	Fast Read (1-4-4) (1-1-4): Wait States, Mode Bit Clocks, Instruction
4	Fast Read (1-1-2) (1-2-2): Wait States, Mode Bit Clocks, Instruction
5	Fast Read (2-2-2) (4-4-4) Support
6	Fast Read (2-2-2): Wait States, Mode Bit Clocks, Instruction
7	Fast Read (4-4-4): Wait States, Mode Bit Clocks, Instruction
8	Erase Type 1 & 2 Size and Instruction
9	Erase Type 3 & 4 Size and Instruction
10	Erase Type (1:4) Typical Erase Times and Multiplier Used To Derive Max Erase Times
11	Chip Erase Typical Time, Byte Program and Page Program Typical Times, Page Size
12	Erase/Program Suspend/Resume Support, Intervals, Latency, Keep Out Area Size
13	Program/Erase Suspend/Resume Instructions
14	Deep Powerdown and Status Register Polling Device Busy
15	Hold and WP Disable Function, Quad Enable Requirements, 4-4-4 Mode Enable/Disable Sequences, 0-4-4 Entry/Exit Methods and Support
16	32-bit Address Entry/Exit Methods and Support, Soft Reset and Rescue Sequences, Volatile and Nonvolatile Status Register Support

6.4.4 JEDEC Basic Flash Parameter Table: 1st DWORD

Bits	Description
31:23	**Unused** Contains FFh and can never be changed.
22	**Supports (1-1-4) Fast Read** Device supports single input instruction & address and quad output data Fast Read. 0: (1-1-4) Fast Read NOT supported. 1: (1-1-4) Fast Read supported.
21	**Supports (1-4-4) Fast Read** Device supports single input instruction, quad input address, and quad output data Fast Read. 0: (1-4-4) Fast Read NOT supported. 1: (1-4-4) Fast Read supported.
20	**Supports (1-2-2) Fast Read** Device supports single input instruction, dual input address, and dual output data Fast Read. 0: (1-2-2) Fast Read NOT supported. 1: (1-2-2) Fast Read supported.
19	**Supports Double Transfer Rate (DTR) Clocking** Indicates the device supports some type of double transfer rate clocking. 0: DTR NOT supported 1: DTR Clocking supported
18:17	**Address Bytes** Number of bytes used in addressing flash array read, write and erase: 00b: 3-Byte only addressing 01b: 3- or 4-Byte addressing (e.g., defaults to 3-Byte mode; enters 4-Byte mode on command) 10b: 4-Byte only addressing 11b: Reserved NOTE This field refers to the number of address bits/bytes that are clocked in for any command requiring an address except for SFDP Header or Table accesses. All SFDP accesses use 3-byte addressing. Examples: Read, Fast Read, Write, 4 kilobyte Erase.
16	**Supports (1-1-2) Fast Read** Device supports single input instruction & address and dual output data Fast Read with 8 wait states. 0: (1-1-2) Fast Read NOT supported. 1: (1-1-2) Fast Read supported.
15:8	**4 Kilobyte Erase Instruction** NOTE If 4 kilobyte erase is not supported, then enter FFh. This instruction must also be included in one of the Erase Types in 6.4.8 or 6.4.12
7:5	**Unused**

Bits	Description
	Contains 111b and can never be changed.
4	**Write Enable Instruction Select for Writing to Volatile Status Register** This bit only applies if bit 3 is 1. 0: flash device requires instruction 50h as the write enable prior to performing a volatile write to the status register 1: flash device requires instruction 06h as the write enable prior to performing a volatile write to the status register. NOTE If target flash status register is nonvolatile, then bits 3 and 4 must be set to 00b. This bit definition is maintained for legacy compatibility only. New system implementations should refer to 6.4.1 for a full definition of volatile and non-volatile behavior.
3	**Volatile Status Register Block Protect bits** 0: Block Protect bits in device's status register are solely non-volatile or may be programmed either as volatile using the 50h instruction for write enable or non-volatile using the 06h instruction for write enable. 1: Block Protect bits in device's status register are solely volatile. NOTE If target flash register is nonvolatile, then bits 3 and 4 must be set to 00b. This bit definition is maintained for legacy compatibility only. New devices should refer to 6.4.16 for a full definition of volatile and non-volatile behavior.
2	**Write Granularity** 0: 1 Byte – Use this setting for single byte programmable devices or buffer programmable devices when the buffer is less than 64 bytes (32 Words). 1: Use this setting for buffer programmable devices when the buffer size is 64 bytes (32 Words) or larger. This bit definition is maintained for legacy compatibility only. New system implementations should refer to 6.4.14for the buffer (page) size. The legacy minimum write granularity is a single byte within any size programming buffer.
1:0	**Block/Sector Erase Sizes** Identifies if the device supports uniform 4k erase blocks. This erase size information must also be included one of the EraseTypes in 6.4.8 or 6.4.12. 00b: Reserved 01b: 4 kilobyte Erase is supported throughout the device 10b: Reserved 11b: Use this setting only if uniform 4 kilobyte erase is unavailable. NOTE This is a legacy field. Refer to sections 6.4.8 and 6.4.12 for information on what erase sizes are supported.

6.4.5 JEDEC Basic Flash Parameter Table: 2nd DWORD

Bits	Description
31:0	**Flash Memory Density** For densities 2 gigabits or less, bit-31 is set to 0b. The field 30:0 defines the size in bits. Example: 00FFFFFFh = 16 megabits For densities 4 gigabits and above, bit-31 is set to 1b. The field 30:0 defines 'N' where the density is computed as 2^N bits (N must be >= 32)**.** Example: 80000021h = 2^33 = 8 gigabits

6.4.6 JEDEC Basic Flash Parameter Table: 3rd DWORD

Bits	Description
31:24	**(1-1-4) Fast Read Instruction** Instruction for single input instruction & address and quad output data Fast Read.
23:21	**(1-1-4) Fast Read Number of Mode Clocks** This field will be 000b if Mode Bits are not supported. NOTE This field should be counted in clocks not number of bits received by the serial flash. The master drives the bus during "mode bits" cycles; the master tri-states the bus during "dummy" cycles. **Example:** If 4 mode bits are needed with a single input address phase command, this field would be 100b.
20:16	**(1-1-4) Fast Read Number of Wait states (dummy clocks) needed before valid output** This field will be 00000b if wait states/dummy clocks are not supported. (The number of dummy clocks should be > 0 to avoid contention on bi-directional pins.) **Example:** If 8 bits are needed with a single input address phase command, this field would be 01000b.
15:8	**(1-4-4) Fast Read Instruction** Instruction for single input instruction, quad input address, and quad output data Fast Read.
7:5	**Quad Input Address Quad Output (1-4-4) Fast Read Number of Mode Clocks** This field will be 000b if Mode bits are not supported, NOTE This field should be counted in clocks not number of bits received by the serial flash. The master drives the bus during "mode bits" cycles; the master tri-states the bus during "dummy" cycles. **Example:** If 8 mode bits are needed with a quad input address phase command, this field would be 010b.
4:0	**(1-4-4) Fast Read Number of Wait states (dummy clocks) needed before valid output** This field will be 00000b if wait states/dummy clocks are not supported. (The number of dummy clocks should be > 0 to avoid contention on bi-directional pins.) **Example:** If 16 bits are needed with a quad input address phase command, this field would be 00100b.

6.4.7 JEDEC Basic Flash Parameter Table: 4th DWORD

Bits	Description
31:24	**(1-2-2) Fast Read Instruction** Instruction for single input instruction, dual input address, and dual output data Fast Read.
23:21	**(1-2-2) Fast Read Number of Mode Clocks** This field will be 000b if Mode bits are not supported, NOTE This field should be counted in clocks not number of bits received by the serial flash. The master drives the bus during "mode bits" cycles; the master tri-states the bus during "dummy" cycles. **Example:** If 8 mode bits are needed with a dual input address phase command, this field would be 100b.
20:16	**(1-2-2) Fast Read Number of Wait states (dummy clocks) needed before valid output** This field will be 00000b if wait states/dummy clocks are not supported. (The number of dummy clocks should be > 0 to avoid contention on bi-directional pins.) **Example:** If 8 bits are needed with a dual input address phase command, this field would be 00100b.
15:8	**(1-1-2) Fast Read Instruction** Instruction for single input instruction& address and dual output data Fast Read. Note: The industry standard is 3Bh
7:5	**(1-1-2) Fast Read Number of Mode Clocks** This field will be 000b if Mode bits are not supported, NOTE This field should be counted in clocks not number of bits received by the serial flash. **Example:** If 4 mode bits are needed with a single input address phase command, this field would be 100b.
4:0	**(1-1-2) Fast Read Number of Wait states (dummy clocks) needed before valid output** This field should be programmed with 01000b for 8 clocks of dummy cycle. (The number of dummy clocks should be > 0 to avoid contention on bi-directional pins.) NOTE For legacy reasons, if dummy clocks for this instruction is not 01000b, then bit 16 in 6.4.1 (Supports (1-1-2) Fast Read with 8 wait states) must NOT be set to '1'.

6.4.8 JEDEC Basic Flash Parameter Table: 5th DWORD

Bits	Description
31:5	**Reserved**. These bits default to all 1's
4	**Supports (4-4-4) Fast Read** Device supports Quad input instruction & address and quad output data Fast Read. 0: (4-4-4) Fast Read NOT supported. 1: (4-4-4) Fast Read supported.
3:1	**Reserved**. These bits default to all 1's
0	**Supports (2-2-2) Fast Read** Device supports dual input instruction& address and dual output data Fast Read. 0: (2-2-2) Fast Read NOT supported. 1: (2-2-2) Fast Read supported.

6.4.9 JEDEC Basic Flash Parameter Table: 6th DWORD

Bits	Description
31:24	**(2-2-2) Fast Read Instruction** Instruction for dual input instruction& address and dual output data Fast Read.
23:21	**(2-2-2) Fast Read Number of Mode Clocks** This field will be 000b if Mode bits are not supported, NOTE This field should be counted in clocks not number of bits received by the serial flash. The master drives the bus during "mode bits" cycles; the master tri-states the bus during "dummy" cycles. **Example:** If 4 mode bits are needed with a (2-2-2) Fast Read command, this field would be 010b.
20:16	**(2-2-2) Fast Read Number of Wait states (dummy clocks) needed before valid output** This field will be 00000b if wait states/dummy clocks are not supported. (The number of dummy clocks should be > 0 to avoid contention on bi-directional pins.) **Example:** If 8 bits are needed with a (2-2-2) Fast Read command, this field would be 00100b.
15:0	**Reserved**. These bits default to all 1's

6.4.10 JEDEC Basic Flash Parameter Table: 7th DWORD

Bits	Description
31:24	**(4-4-4) Fast Read Instruction** Instruction for quad input instruction/address, quad output data Fast Read.
23:21	**(4-4-4) Fast Read Number of Mode Clocks** This field will be 000b if Mode bits are not supported, NOTE This field should be counted in clocks not number of bits received by the serial flash. The master drives the bus during "mode bits" cycles; the master tri-states the bus during "dummy" cycles. **Example:** If 8 mode bits are needed with a (4-4-4) Fast Read phase command, this field would be 010b.
20:16	**(4-4-4) Fast Read Number of Wait states (dummy clocks) needed before valid output** This field will be 00000b if wait states/dummy clocks are not supported. (The number of dummy clocks should be > 0 to avoid contention on bi-directional pins.) **Example:** If 16 bits are needed with a (4-4-4) Fast Read phase command, this field would be 00100b.
15:0	**Reserved**. These bits default to all 1's

6.4.11 JEDEC Basic Flash Parameter Table: 8th DWORD

NOTE If the device uses a 4k subsector size, that size and instruction must be included somewhere in DWORDs 8 or 9. This allows the user to discover the typical and maximum erase times for the 4k subsector by referencing DWORD 10.

Bits	Description
31:24	**Erase Type 2 Instruction** Instruction used to erase the number of bytes specified by Erase Type 2 Size (bits 23-16).
23:16	**Erase Type 2 Size:** This field will be 00h if this erase type does not exist. NOTE This field specifies 'N' and is used to calculate erase type size = 2^N bytes **Example:** If the erase type size is 32 kilobytes, this field would 0Fh.
15:8	**Erase Type 1 Instruction** Instruction used to erase the number of bytes specified by Erase Type 1 Size (bits 7-0).
7:0	**Erase Type 1 Size** NOTE This field specifies 'N' and is used to calculate erase type size = 2^N bytes **Example:** If the erast type size is 4 kilobytes, this field would 0Ch.

6.4.12 JEDEC Basic Flash Parameter Table: 9th DWORD

Bits	Description
31:24	**Erase Type 4 Instruction** Instruction used to erase the number of bytes specified by Erase Type 4 Size (bits 23-16).
23:16	**Erase Type 4 Size** This field will be 00h if this erase type does not exist. NOTE This field specifies 'N' and is used to calculate erase type size = 2^N bytes **Example:** If the erase type size is 256 kilobytes, this field would 12h.
15:8	**Erase Type 3 Instruction** Instruction used to erase the number of bytes specified by Erase Type 3 Size (bits 7-0).
7:0	**Erase Type 3 Size** This field will be 00h if this erase type does not exist. NOTE This field specifies 'N' and is used to calculate erase type size = 2^N bytes **Example:** If the erase type size is 64 kilobytes, this field would 10h.

6.4.13 JEDEC Basic Flash Parameter Table: 10th DWORD

Bits	Description
31:25	**Erase Type 4 Erase, Typical time** Time the device *typically* takes to erase a Erase Type 4 size, see 6.4.12. User must poll device busy to determine if the operation has completed. This field has no meaning if the Erase Type 4 size is 00h. 31:30 units (00b: 1 ms, 01b: 16 ms, 10b: 128 ms, 11b: 1 s) 29:25 count Formula: typical time = (count + 1)*units **Example:** If count=2 and units=10b, then typical time is (2+1)*128ms = 384 ms The range is 1ms to 32 seconds in four groups: 1 ms to 32 ms, 16 ms to 512 ms, 128 ms to 4096 ms, 1 s to 32 s
24:18	**Erase Type 3 Erase, Typical time** Time the device *typically* takes to erase a Erase Type 3 size, see 6.4.12. User must poll device busy to determine if the operation has completed. This field has no meaning if the Erase Type 3 size is 00h. 24:23 units (00b: 1 ms, 01b: 16 ms, 10b: 128 ms, 11b: 1 s) 22:18 count Formula: typical time = (count + 1)*units **Example:** If count=1 and units=10b, then typical time is (1+1)*128ms = 256 ms The range is 1ms to 32 seconds in four groups: 1 ms to 32 ms, 16 ms to 512 ms, 128 ms to 4096 ms, 1 s to 32 s

17:11	**Erase Type 2 Erase, Typical time** Time the device *typically* takes to erase a Erase Type 2 size, see 6.4.8. User must poll device busy to determine if the operation has completed. This field has no meaning if the corresponding Erase Type size is 00h. 17:16 units (00b: 1 ms, 01b: 16 ms, 10b: 128 ms, 11b: 1 s) 15:11 count Formula: typical time = (count + 1)*units **Example:** If count=0 and units=10b, then typical time is (0+1)*128ms = 128 ms The range is 1ms to 32 seconds in four groups: 1 ms to 32 ms, 16 ms to 512 ms, 128 ms to 4096 ms, 1 s to 32 s
10:4	**Erase Type 1 Erase, Typical time** Time the device *typically* takes to erase a Erase Type 1 size, see 6.4.8. User must poll device busy to determine if the operation has completed. This field has no meaning if the corresponding Erase Type size is 00h. 10:9 units (00b: 1 ms, 01b: 16 ms, 10b: 128 ms, 11b: 1 s) 8:4 count Formula: typical time = (count + 1)*units **Example:** If count=1 and units=10b, then typical time is 1*128ms = 128 ms The range is 1ms to 32 seconds in four groups: 1 ms to 32 ms, 16 ms to 512 ms, 128 ms to 4096 ms, 1 s to 32 s
3:0	**Multiplier from typical erase time to maximum erase time** 3:0 count Formula: EraseType n (or Chip) erase maximum time = 2 * (count + 1) * EraseType n (or Chip) erase typical time **Example:** If count = 9, then Erase Type n (or Chip) erase maximum time is 20 * Sector Type n (or Chip) erase typical time NOTE 1 'n' = 1, 2, 3, or 4 NOTE 2 This multiplier applies to all erase types and the chip erase. The maximum time is intended to be used as a watchdog timeout for an error or failure condition. Since a common scale factor is used across all erase sizes, any particular maximum time may only approximate the datasheet maximum time.

6.4.14 JEDEC Basic Flash Parameter Table: 11th DWORD

Bits	Description
31	**Reserved**
30:24	**Chip Erase, Typical time** Typical time to erase one chip (die). User must poll device busy to determine if the operation has completed. For a device consisting of multiple die, that are individually accessed, the time is for each die to which a chip erase command is applied. 30:29 units (00b: 16 ms, 01b: 256 ms, 10b: 4 s, 11b: 64 s) 28:24 count Formula: typical time = (count + 1)*units **Example:** If count=5 and units=10b, then typical time is: 5*4 s = 20 s The range of this field is 16ms to 2048 seconds in four groups: 16 ms to 512 ms, 256 ms to 8192 ms, 4 s to 128 s, 64 s to 2048 s
23:19	**Byte Program Typical time, additional byte** Time the device *typically* takes to write each additional byte after the first. User must poll device busy to determine if the operation has completed. 23 units (0: 1 us, 1: 8 us) 22:19 count Formula: additional byte time = (count + 1)*units/byte **Example:** If units = 1 and count =4, then each additional byte typically adds (4+1)*8 us = 40 us to the programming time. For 16 bytes, the additional time would be 16 * 40 us = 640 us The range is 1 us to 128 us in two groups: 1 us to 16 us and 8 us to 128 us. NOTE The programming time for small numbers of bytes does not scale linearly up to a full page programming time. When the number of bytes being programmed exceeds ½ of a page size, users should base estimates on the Page Program typical time in this DWORD.
18:14	**Byte Program Typical time, first byte** Time the device *typically* takes to write the first byte in a sequence. User must poll device busy to determine if the operation has completed. 18 units (0: 1 us, 1: 8 us) 17:14 count Formula: first byte typical time = (count + 1)*units **Example:** If units = 0 and count = 7, then typical time is (7+1)*1 us = 8 us The range is 1 us to 128 us in two groups: 1 us to 16 us and 8 us to 128 us
13:8	**Page Program Typical time** Time the device *typically* takes to write a full page. User must poll device busy to determine if the operation has completed. The user may scale this by ½ or ¼ to determine approximate times for ½ and ¼ page program operations 13 units (0: 8 us, 1: 64 us) 12:8 count Formula: typical page program time = (count + 1)*units The range is 8 us to 2048 us in two groups: 8 us to 256 us and 64 us to 2048 us
7:4	**Page Size** This field specifies 'N' and is used to calculate page size = 2^N bytes.
3:0	**Multiplier from typical time to max time for Page or byte program** 3:0 count Formula: maximum time = 2 * (count + 1)*typical time NOTE This multiplier applies to all page or byte typical program times. The maximum time is intended to be used as a watchdog timeout for an error or failure condition. Since a common scale factor is used across all program sizes, any particular maximum time may only approximate the datasheet maximum time.

6.4.15 JEDEC Basic Flash Parameter Table: 12th DWORD

Bits	Description
31	**Suspend / Resume supported** The device supports suspend and resume of both program and erase operations. 0: supported 1: not supported
30:24	**Suspend in-progress erase max latency** Maximum time required by the flash device to suspend an in-progress erase and be ready to accept another command which accesses the flash array. This time does not apply to the read status command. See also *Suspend in-progress program in this DWORD.* 30:29 units (00b: 128ns, 01b: 1us, 10b: 8us, 11b: 64us) 28:24 count Formula: erase max latency = (count + 1)*units **Example:** if units =01b and count =19, then erase max latency = (19+1)*1 us = 20 us The range is 128 ns to 2048 us in four groups: 128 ns to 4**.** 096 us, 1 us to 32 us, 8 us to 256 us, 64 us to 2048 us
23:20	**Erase Resume to Suspend Interval** The device requires this typical amount of time to make progress on the erase before allowing another suspend. It is possible to immediately suspend again after a resume -- there is no required minimum time between resuming an operation and suspending the operation again. However, the device requires some average amount of active operation time, after a resume, to make progress on the operation, before another suspend. This parameter recommends an average interval of time that should be allowed between a resume and the next suspend in order for the operation to eventually complete. If there are some intervals less than the recommended value there should be a similar number of intervals that are longer than the recommended value. If the interval is consistently less than the recommended value the operation may never finish. 23:20 count of fixed units of 64us Formula: erase resume to suspend interval = (count + 1)*64 us **Example:** if count = 7, the erase resume to suspend interval = (7+1)*64 us = 512 us The range is 64 us to 1024 us
19:13	**Suspend in-progress program max latency** Maximum time required by the flash device to suspend an in-progress program and be ready to accept another command which accesses the flash array. This time does not apply to the read status command. See also *Suspend in-progress erase in this DWORD.* 19:18 units (00b: 128ns, 01b: 1us, 10b: 8us, 11b: 64us) 17:13 count Formula: suspend in-progress program max latency = (count+1)*units **Example:** if units = 01b and count = 4, then suspend in-progress program max latency = (4+1)*1 us = 5 us The range is 128 ns to 2048 us in four groups: 128 ns to 4.096 us, 1 us to 32 us, 8 us to 256 us, and 64 us to 2048 us.

12:9	**Program Resume to Suspend Interval** The device requires this typical amount of time to make progress on the program operation before allowing another suspend. It is possible to immediately suspend again after a resume -- there is no required minimum time between resuming an operation and suspending the operation again. However, the device requires some average amount of active operation time, after a resume, to make progress on the operation, before another suspend. This parameter recommends an average interval of time that should be allowed between a resume and the next suspend in order for the operation to eventually complete. If there are some intervals less than the recommended value there should be a similar number of intervals that are longer than the recommended value. If the interval is consistently less than the recommended value the operation may never finish. 12:9 count of fixed units of 64us Formula: program resume to suspend interval = (count + 1)*64 us **Example:** if count = 15, the erase resume to suspend interval = (15+1)*64 us = 1024 us The range is 64 us to 1024 us
8	**Reserved**
7:4	**Prohibited Operations During Erase Suspend** xxx0b: May not initiate a new erase anywhere (erase nesting not permitted) xxx1b: May not initiate a new erase in the erase suspended erase type size xx0xb: May not initiate a page program anywhere xx1xb: May not initiate a page program in the erase suspended erase type size x0xxb: Refer to vendor datasheet for read restrictions x1xxb: May not initiate a read in the erase suspended erase type size 0xxxb: Additional erase or program restrictions apply 1xxxb: The erase and program restrictions in bits 5:4 are sufficient NOTE This list is not comprehensive. Consult the device datasheet for a full list of allowed and prohibited operations.
3:0	**Prohibited Operations During Program Suspend** xxx0b: May not initiate a new erase anywhere (erase nesting not permitted) xxx1b: May not initiate a new erase in the program suspended page size xx0xb: May not initiate a new page program anywhere (program nesting not permitted) xx1xb: May not initiate a new page program in the program suspended page size x0xxb: Refer to vendor datasheet for read restrictions x1xxb: May not initiate a read in the program suspended page size 0xxxb: Additional erase or program restrictions apply 1xxxb: The erase and program restrictions in bits 1:0 are sufficient NOTE This list is not comprehensive. Consult the device datasheet for a full list of allowed and prohibited operations.

6.4.16 JEDEC Basic Flash Parameter Table: 13th DWORD

Bits	Description
31:24	**Suspend Instruction** Instruction used to suspend a write or erase type operation.
23:16	**Resume Instruction** Instruction used to resume a write or erase type operation.
15:8	**Program Suspend Instruction** Instruction used to suspend a program operation. (If the device requires a unique instruction to suspend a "program" command then that instruction is listed here. Otherwise this field contains the same value as the "Suspend Instruction" field above.)
7:0	**Program Resume Instruction** Instruction used to resume a program operation. (If the device requires a unique instruction to resume a "program" command then that instruction is listed here. Otherwise this field contains the same value as the "Resume Instruction" field above.)

6.4.17 JEDEC Basic Flash Parameter Table: 14th DWORD

Bits	Description
31	**Deep Powerdown Supported** 0: supported 1: not supported
30:23	**Enter Deep Powerdown Instruction** Instruction used to enter deep powerdown
22:15	**Exit Deep Powerdown Instruction** Instruction used to exit deep powerdown
14:8	**Exit Deep Powerdown to next operation delay** Maximum time required by the flash device to exit Deep Powerdown and be ready to accept any command. (Note: Read status is not valid when exiting deep powerdown.) 14:13 units (00b: 128ns, 01b: 1us, 10b: 8us, 11b: 64us) 12:8 count Formula: exit Deep Powerdown to next operation delay = (count+1)*units **Example:** if units = 10b and count = 4, then delay = (4+1)*8 us = 40 us The range is 128 ns to 2048 us in four groups: 128 ns to 4.096 us, 1 us to 32 us, 8 us to 256 us, and 64 us to 2048 us
7:2	**Status Register Polling Device Busy** This bit field defines various ways the flash device's busy status may be polled. A zero in a bit position indicates that the device does not support the particular polling method. 1x_xxxxb: Reserved x1_xxxxb: Reserved xx_1xxxb: Reserved xx_x1xxb: Reserved xx_xx1xb: Bit 7 of the Flag Status Register may be polled any time a Program, Erase, Suspend/Resume command is issued, or after a Reset command while the device is busy. The read instruction is 70h. Flag Status Register bit definitions: bit[7]: Program or erase controller status (0=busy; 1=ready) xx_xxx1b: Use of legacy polling is supported by reading the Status Register with 05h instruction and checking WIP bit[0] (0=ready; 1=busy).
1:0	**Reserved**

6.4.18 JEDEC Basic Flash Parameter Table: 15th DWORD

Bits	Description
31:24	**Reserved**
23	**HOLD or RESET Disable** Defines whether HOLD or RESET may be disabled via a configuration register If driving DQ3 high during command phase HOLD or RESET do not need to be disabled. Device decodes instruction to determine functionality of HOLD/RESET vs. data 1: set bit 4 of the Non-Volatile Extended Configuration Register = 0 to disable HOLD or RESET 0: above feature is not supported
22:20	**Quad Enable Requirements (QER):** This field describes whether the device contains a Quad Enable (QE) bit used to enable 1-1-4 and 1-4-4 quad read or quad program operations. If QE exists, this field also identifies the bit location and method to set/clear the bit. In this standard, status register 1 refers to the first data byte transferred on a Read Status (05h) or Write Status (01h) command. Status register 2 refers to the byte read using instruction 35h. Status register 2 is the second byte transferred in a Write Status (01h) command. Bits are numbered from 7 to 0, where bit 7 is transferred first on the wire. NOTE Industry naming and definitions of these status registers may differ. The user will typically perform a read-modify-write sequence of operations to maintain the state of all other writable status register bits. For example read both status registers, set/clear QE, Write Status with both data bytes. 000b: Device does not have a QE bit. Device detects 1-1-4 and 1-4-4 reads based on instruction. DQ3/HOLD# functions as hold during instruction phase. 001b: QE is bit 1 of status register 2. It is set via Write Status with two data bytes where bit 1 of the second byte is one. It is cleared via Write Status with two data bytes where bit 1 of the second byte is zero. Writing only one byte to the status register has the side-effect of clearing status register 2, including the QE bit. The 100b code is used if writing one byte to the status register does not modify status register 2. 010b: QE is bit 6 of status register 1. It is set via Write Status with one data byte where bit 6 is one. It is cleared via Write Status with one data byte where bit 6 is zero.. 011b: QE is bit 7 of status register 2. It is set via Write status register 2 instruction 3Eh with one data byte where bit 7 is one. It is cleared via Write status register 2 instruction 3Eh with one data byte where bit 7 is zero. The status register 2 is read using instruction 3Fh. 100b: QE is bit 1 of status register 2. It is set via Write Status with two data bytes where bit 1 of the second byte is one. It is cleared via Write Status with two data bytes where bit 1 of the second byte is zero. In contrast to the 001b code, writing one byte to the status register does not modify status register 2. 101b: QE is bit 1 of the status register 2. Status register 1 is read using Read Status instruction 05h. Status register 2 is read using instruction 35h. QE is set via Write Status instruction 01h with two data bytes where bit 1 of the second byte is one. It is cleared via Write Status with two data bytes where bit 1 of the second byte is zero. other: reserved
19:16	**0-4-4 Mode Entry Method:** xxx1b: Mode Bits[7:0] = A5h Note: QE must be set prior to using this mode xx1xb: Read the 8-bit volatile configuration register with instruction 85h, set XIP bit[3] in the data read, and write the modified data using the instruction 81h, then Mode Bits [7:0] = 01h x1xxb: Mode Bit[7:0]=AXh 1xxxb: Reserved

15:10	**0-4-4 Mode Exit Method** xx_xxx1b: Mode Bits[7:0] = 00h will terminate this mode at the end of the current read operation xx_xx1xb: If 3-Byte address active, input Fh on DQ0-DQ3 for 8 clocks. If 4-Byte address active, input Fh on DQ0-DQ3 for 10 clocks. This will terminate the mode prior to the next read operation. xx_x1xxb: Reserved xx_1xxxb: Input Fh (mode bit reset) on DQ0-DQ3 for 8 clocks. This will terminate the mode prior to the next read operation. x1_xxxxb: Mode Bit[7:0] ≠ AXh 1x_xxxxb: Reserved
9	**0-4-4 mode supported** This mode is variously referred to as implied instruction, continuous read, execute in place, etc. 0: not supported 1: supported
8:4	**4-4-4 mode enable sequences** This field describes the supported methods to enter 4-4-4 mode from 1-1-1 mode. x_xxx1b: set QE per QER description above, then issue instruction 38h x_xx1xb: issue instruction 38h x_x1xxb: issue instruction 35h x_1xxxb: device uses a read-modify-write sequence of operations: read configuration using instruction 65h followed by address 800003h, set bit 6, write configuration using instruction 71h followed by address 800003h. This configuration is volatile. 1_xxxxb: **4-4-4 mode enable sequences** Device uses a read-modify-write sequence of operations: Read Volatile Enhanced Configuration Register using instruction 65h, no address is required, reset bit 7 to 0. Write Volatile Enhanced Configuration Register using instruction 61h, no address is required. This configuration is volatile. **4-4-4 mode disable sequences** Device uses a read-modify-write sequence of operations: Read Volatile Enhanced Configuration Register using instruction 65h, no address is required, set bit 7 to 1. Write Volatile Enhanced Configuration Register using instruction 61h, no address is required. This configuration is volatile. NOTE If device is in 0-4-4 mode, then this mode must be exited before the 4-4-4 enable sequence is issued.
3:0	**4-4-4 mode disable sequences** This field describes the supported methods to exit 4-4-4 mode. xxx1b: issue FFh instruction xx1xb: issue F5h instruction x1xxb: device uses a read-modify-write sequence of operations: read configuration using instruction 65h followed by address 800003h, clear bit 6, rrite configuration using instruction 71h followed by address 800003h. This configuration is volatile. 1xxxb: issue the Soft Reset 66/99 sequence, see 6.4.19 NOTE If device is in 0-4-4 mode, then this mode must be exited before the 4-4-4 disable sequence is issued.

6.4.19 JEDEC Basic Flash Parameter Table: 16th DWORD

Bits	Description
31:24	**Enter 4-Byte Addressing** This field defines the supported methods to enter 4-byte addressing mode or to use an extended address register with 3-byte addressing to access memory above 16 MBytes. xxxx_xxx1b: issue instruction B7h (preceding write enable not required) xxxx_xx1xb: issue write enable instruction 06h, then issue instruction B7h xxxx_x1xxb: 8-bit volatile extended address register used to define A[31:24] bits. Read with instruction C8h. Write instruction is C5h with 1 byte of data. Select the active 128 Mbit memory segment by setting the appropriate A[31:24] bits and use 3-Byte addressing. xxxx_1xxxb: 8-bit volatile bank register used to define A[30:A24] bits. MSB (bit[7]) is used to enable/disable 4-byte address mode. When MSB is set to '1', 4-byte address mode is active and A[30:24] bits are don't care. Read with instruction 16h. Write instruction is 17h with 1 byte of data. When MSB is cleared to '0', select the active 128 Mbit segment by setting the appropriate A[30:24] bits and use 3-Byte addressing. xxx1_xxxxb: A 16-bit nonvolatile configuration register controls 3-Byte/4-Byte address mode. Read instruction is B5h. Bit[0] controls address mode [0=3-Byte; 1=4-Byte]. Write configuration register instruction is B1h, data length is 2 bytes. xx1x_xxxxb: Supports dedicated 4-Byte address instruction set. Consult vendor data sheet for the instruction set definition. x1xx_xxxxb: Always operates in 4-Byte address mode 1xxx_xxxxb: Reserved
23:14	**Exit 4-Byte Addressing** xx_xxxx_xxx1b: issue instruction E9h to exit 4-Byte address mode (write enable instruction 06h is not required) xx_xxxx_xx1xb: issue write enable instruction 06h, then issue instruction E9h to exit 4-Byte address mode xx_xxxx_x1xxb: 8-bit volatile extended address register used to define A[31:A24] bits. Read with instruction C8h. Write instruction is C5h, data length is 1 byte. Return to lowest memory segment by setting A[31:24] to 00h and use 3-Byte addressing. xx_xxxx_1xxxb: 8-bit volatile bank register used to define A[30:A24] bits. MSB (bit[7]) is used to enable/disable 4-byte address mode. When MSB is cleared to '0', 3-byte address mode is active and A30:A24 are used to select the active 128 Mbit memory segment. Read with instruction 16h. Write instruction is 17h, data length is 1 byte. xx_xxx1_xxxxb: A 16-bit nonvolatile configuration register controls 3-Byte/4-Byte address mode. Read instruction is B5h. Bit[0] controls address mode [0=3-Byte; 1=4-Byte]. Write configuration register instruction is B1h, data length is 2 bytes. xx_xx1x_xxxxb: Hardware reset xx_x1xx_xxxxb: Software reset (see bits 13:8 in this DWORD) xx_1xxx_xxxxb: Power cycle x1_xxxx_xxxxb: Reserved 1x_xxxx_xxxxb: Reserved

13:8	**Soft Reset and Rescue Sequence Support** This field specifies how to return the device to its default power-on state. 00_0000b: no software reset instruction is supported xx_xxx1b: drive Fh on all 4 data wires for 8 clocks xx_xx1xb: drive Fh on all 4 data wires for 10 clocks if device is operating in 4-byte address mode xx_x1xxb: drive Fh on all 4 data wires for 16 clocks xx_1xxxb: issue instruction F0h x1_xxxxb: issue reset enable instruction 66h, then issue reset instruction 99h. The reset enable, reset sequence may be issued on 1, 2, or 4 wires depending on the device operating mode. 1x_xxxxb: exit 0-4-4 mode is required prior to other reset sequences above if the device may be operating in this mode. See 6.4.18, 0-4-4 Mode Exit
7	**Reserved**
6:0	**Volatile or Non-Volatile Register and Write Enable Instruction for Status Register 1** The instruction 01h is typically used to write status register 1 which contains Block Protection (BP) and other bits. Status register 1 is written by the first data byte following the instruction 01h. The protection bits must be written to zero to enable writes/erases to the device. This field describes how to modify the writable bits in status register 1 in either a volatile or non-volatile manner. Bits 1:0 in status register 1 are de-facto standard write enable and busy status and are excluded from the definitions below. xxx_xxx1b: Non-Volatile Status Register 1, powers-up to last written value, use instruction 06h to enable write xxx_xx1xb: Volatile Status Register 1, status register powers-up with bits set to "1"s, use instruction 06h to enable write xxx_x1xxb: Volatile Status Register 1, status register powers-up with bits set to "1"s, use instruction 50h to enable write xxx_1xxxb: Non-Volatile/Volatile status register 1 powers-up to last written value in the non-volatile status register, use instruction 06h to enable write to non-volatile status register. Volatile status register may be activated after power-up to override the non-volatile status register, use instruction 50h to enable write and activate the volatile status register. xx1_xxxxb: Status Register 1 contains a mix of volatile and non-volatile bits. The 06h instruction is used to enable writing of the register. x1x_xxxxb: Reserved 1xx_xxxxb: Reserved NOTE If the status register is read-only then this field will contain all zeros in bits 4:0.

6.5 JEDEC Sector Map Parameter Table

A sector is the minimum size and alignment (granularity) of an area that can be erased in the data array of a flash memory device. Different areas within the address range of the data array may have a different minimum erase granularity (sector size).

The Sector Map Parameter Table identifies the location and size of sectors within the main data array of the flash memory device and identifies which Erase Types are supported by each sector. This table is required when a memory device:

- Has sectors of more than one size
- Or, does not allow all Erase Type commands to be applied to all sectors.

When there is more than one sector size in a device, each contiguous group of sectors, that are of the same size, and support the same erase types, is called a region. A region may be as small as a single sector. There are one or more regions for each sector size. There is more than one region of a particular sector size when that sector size appears in more than one area of the address space and these areas are separated by one or more regions of a different sector size. For example: a region of 4KB sectors, followed by a region of 64KB sectors, followed by a region of 4KB sectors.

There may also be more than one region of the same size sector when the regions support different sets of Erase Types. For example: one region of 4KB sectors may support use of Erase Types for 4KB erase and 64KB erase but not 8KB erase or 32KB erase and an adjacent region of 4KB sectors may support all these Erase Types. The Sector Map Parameter Table is used to identify which Erase Type commands may be used within each region.

There may be more than one possible map of sector size, location, or Erase Type support. For example: a memory device may be user configurable to have some small sectors at the top or at the bottom of the address space, with all remaining sectors being a larger size. Because the small sectors may appear at either the top or the bottom of the address space, two sector maps are needed to describe the top or bottom location and size of the smaller sectors.

If there is more than one user selected sector map (configuration), this table includes the definition of instructions needed to determine which sector map configuration is in use. The number of sector map configuration detection commands is variable, the number of configurations is variable, and the number of regions in each configuration is variable, thus the size of this table is variable.

6.5.1 Sector Map Parameter Header: 1st DWORD

Bits	Description
31:24	**Parameter Table Length** This field specifies how many DWORDs are in the Parameter table. NOTE This field is 1's based. Therefore, 1 indicates 1 DWORD.
23:16	**Parameter Table Major Revision Number** This 8-bit field indicates the major revision number of the parameter table. The value in this field is 1 for this table defined by JESD216B revision. NOTE The Major Revision of JEDEC defined parameter tables can only be modified by updates to this standard.
15:8	**Parameter Table Minor Revision Number** This 8-bit field indicates the minor revision number of the Sector Map parameter table. The value in this field is 0 for this table defined by JESD216B revision. NOTE The Minor Revision of the JEDEC owned parameter tables can only be modified by updates to this standard.
7:0	**Parameter ID LSB** The Sector Map Function Specific Table is assigned the ID LSB of 81h.

6.5.2 Sector Map Parameter Header: 2nd DWORD

Bits	Description
31:24	**Parameter ID MSB** The Sector Map Function Specific Table is assigned the ID MSB of FFh.
23:0	**Parameter Table Pointer (PTP)** This address specifies the start of this header's Parameter Table in the SFDP structure. The address is in terms of bytes and must be DWORD-aligned.

The Sector Map table is built from a sequence of descriptors. There are two types of descriptors:
- A configuration detection command descriptor (command)
- A configuration sector map descriptor (map)

The command descriptors are optional. If there is a single configuration then no command descriptors are needed. If there are more than two configurations, more than one command descriptor is needed. If command descriptors are provided, they always precede map descriptors in the table.

Each configuration detection command is described by two Dwords. These Dwords provide:
- A bit indicating that the descriptor is a configuration detection command,
- a bit indicating whether this descriptor is the last command descriptor,
- the instruction code for the command,
- the number of address bytes for the command,
- the number of read latency cycles between the last address byte and the read data byte,
- a mask to select the bit of interest in the returned data byte,
- and the address value for the command.

It is assumed that each command is reading a configuration register with a single byte of return data and that this type of command does not provide mode bits. Each command selects a single bit from the byte of returned data. There will be a separate command descriptor and command sent for each bit of configuration selecting information that is needed to select the current Sector Map Configuration that is in use.

6.5.3 Configuration Detection Command Descriptor, DWORD 1

Bits	Description
31:24	**Read data mask** Eight bit field with a single bit = 1. The 8 bit field is logically ANDed with the one byte of data read by the configuration detection command specified in this Dword. This bit field is used as a mask to select one bit from the data byte that is read.
23:22	**Configuration detection command address length** Two bit field that defines the length of the address used in the configuration detection command. 00b: No address in the command. 01b: 3 byte address. 10b: 4 byte address 11:b: Variable address length (the current setting of the address length mode defines the address length) When the length is defined as variable, the software or hardware controlling the memory is aware of the address length mode last set in the memory device and this same length of address is used in sending the configuration detection command.
21:20	**Reserved** These bits default to all 1's.
19:16	**Configuration detection command read latency, in clock cycles** Four bit field indicating the number of cycles between the end of address and the beginning of returning read data. Range from 0 to 14 cycles of read latency (wait states). A value of Fh indicates the read latency is variable. The software or hardware controlling the memory is aware of the latency last set in the memory device and this same value is used in the configuration detection command.
15:8	**Detection command instruction.** Eight bit instruction for the sector map configuration detection command.
7:2	**Reserved** These bits default to all 1's.
1	**Descriptor Type** 0b: Command descriptor 1b: Map descriptor
0	**Descriptor Sequence End Indicator** 0b: Another descriptor of the same type follows this descriptor 1b: This is the last descriptor of this type.

6.5.4 Configuration Detection Command Descriptor, DWORD 2

Bits	Description
31:0	**Sector map configuration detection command address** Thirty two bit field providing up to 4 bytes of address. The number of address bytes in this field that are used by the command is determined by the address length in bits 23:22 of the first Dword.

The first DWORD of a command descriptor contains a field that identifies it as a command type descriptor. A single bit field in each command descriptor indicates whether it is the last command descriptor in the sequence. Each command descriptor defines the format of a command used to detect the value of one configuration bit in the memory device and has an eight bit mask value used to select a single bit from one data byte read by the command. The second DWORD of each command descriptor provides a 4 byte address that may optionally be used as part of the configuration detection command.

Each configuration bit value detected is part of the selection for the current configuration of the sector map. For example: if there are five to eight possible sector map configurations, at least three configuration detection commands will be needed to extract three bits of configuration selection information from the device in order to identify which configuration is currently in use. The configuration selector is limited to a maximum of 8 bits, allowing for a maximum of 256 possible configurations. However, there may not be a separate configuration needed for every possible value of the selector value. Each detected configuration bit is shifted left into the configuration selector value such that the last detected bit is in the least significant bit of the selector value. If there are no command descriptors provided, because there is a single configuration, the default value of the configuration selector is zero.

Each configuration map descriptor is described by two or more Dwords. These Dwords provide:

- A map header DWORD
  - A bit indicating that the descriptor is a sector map descriptor,
  - a bit indicating whether this descriptor is the last map descriptor,
  - a configuration ID,
  - a count of the regions in the map,

- a DWORD for for each region in the map
  - a value indicating the size of the region
  - bits that indicate which Erase Types are supported in the region

6.5.5 Configuration Map Descriptor Header DWORD

Bits	Description
31:24	**Reserved** These bits default to all 1's.
23:16	**Region count** The number of following region DWORDs minus 1
15:8	**Configuration ID** A value compared with the configuration selector value to determine if this map descriptor is for the currently selected sector map.
7:2	**Reserved** These bits default to all 1's.
1	**Descriptor Type** 0b: Command descriptor 1b: Map descriptor
0	**Descriptor Sequence End Indicator** 0b: Another descriptor of the same type follows this descriptor 1b: This is the last descriptor of this type.

6.5.6 Region DWORD

Bits	Description
31:8	**Region size** Region size as a multiple (count) of 256 Byte units. The Region size value is zero based, so a region of 256 Bytes has a size value = 0. Region size value = (count - 1) Region size = (value +1) * 256 bytes
7:4	**Reserved** These bits default to all 1's.
3	**Erase Type 4** 1b: Erase Type 4 erase command is supported in this region 0b: Erase Type 4 erase command is not supported in this region
2	**Erase Type 3** 1b: Erase Type 3 erase command is supported in this region 0b: Erase Type 3 erase command is not supported in this region
1	**Erase Type 2** 1b: Erase Type 2 erase command is supported in this region 0b: Erase Type 2 erase command is not supported in this region
0	**Erase Type 1** 1b: Erase Type 1 erase command is supported in this region 0b: Erase Type 1 erase command is not supported in this region

At least one map descriptor is required. The first DWORD of a map descriptor is the header for the map and contains a field that identifies it as a map type descriptor. A single bit field in each map descriptor header indicates whether it is the last map descriptor in the sequence. The map descriptor header has a configuration ID field that is matched against the configuration selector value. If the configuration selector matches the configuration ID, the rest of the map descriptor defines the current sector map in use. If the configuration values do not match, the next map descriptor is examined. If there are no more map descriptors and no configuration ID matched the configuration identifier, the sector address map is unknown.

6.5.6 Region DWORD (cont'd)

The map descriptor header contains a count of the number of regions in the map. Each region is described by a following DWORD. Each region DWORD indicates the size of sectors in the region and the supported Erase Types for that region. The region count indicates the number of DWORDs to skip when looking for the next map descriptor.

The first region starts at location zero of the data array in the flash device. Each additional region starts at the next higher location than the size of the previous region**.**

6.5.7 Sector Map Parameter Table – Example 1

The memory device in the following example is 256Mbit density and has three user setable sector map configurations. Two configuration detection commands are used to read the sector map related configuration control bits from registers. While there are four possible combinations of the two control bits, only three combinations are valid and only three configuration maps are provided. The three configurations are:

- Eight 4Kbyte sectors at the low address end (bottom) of the device address space, with all other sectors being 64Kbytes in size. Only Erase Type 1 for 4Kbyte sectors are supported in the region of 4Kbyte sectors and only Erase Type 2 for 64Kbyte erase is supported in the region containing 64Kbyte sectors.
- Eight 4Kbyte sectors at the high address end (top) of the device address space, with all other sectors being 64Kbytes in size. Only Erase Type 1 for 4Kbyte sectors are supported in the region of 4Kbyte sectors and only Erase Type 2 for 64Kbyte erase is supported in the region containing 64Kbyte sectors.
- Uniform 64Kbyte sectors with only Erase Type 2 for 64Kbyte sectors supported.

<table>
<tr><th rowspan="2">DWORD</th><th colspan="4">Bits</th><th colspan="2" rowspan="2">Notes</th></tr>
<tr><th>31:24</th><th>23:16</th><th>15:8</th><th>7:0</th></tr>
<tr><td>0</td><td>Read Data
Mask = 00001000b =

08h</td><td>Instruction Format
(Address length variable = 11b
[2 bits]
Reserved = 11b
[2 bits]
Latency cycles variable = 1111b
[4 bits]) =

FFh</td><td>Instruction =

65h</td><td>Reserved = 111111b
Descriptor type = command = 0b
Not the last command = 0b

FCh</td><td rowspan="2">Configuration Detect Command 1
(2 DWORDs)

Instruction 65h, variable address length, variable latency, address 00800004h, select bit 3.</td><td rowspan="4">Configuration Detection</td></tr>
<tr><td>1</td><td colspan="4">Address Value =

00800004h</td></tr>
<tr><td>2</td><td>Read Data
Mask = 00000100b =

04h</td><td>Instruction Format
(Address length zero = 00b [2 bits]
Reserved = 11b [2bits]
Latency cycles = 00000b [6 bits] =

20h</td><td>Instruction =

35h</td><td>Reserved = 111111b
Descriptor type = command = 0b
Last command = 1b

FDh</td><td rowspan="2">Configuration Detect Command 2
(2 DWORDs)

Instruction 35h, no address, zero latency, select bit 2.</td></tr>
<tr><td>3</td><td colspan="4">Address Value = FFFFFFFFh</td></tr>
</table>

DWORD	Bits				Notes	
	31:24	23:16	15:8	7:0		
4	Reserved = FFh	Region Count = 3 Regions = 02h	Configuration ID = 00h	Reserved = 111111b Descriptor type = map = 1b Last map = 0b FEh	Configuration Map Header	1st Configuration Address Map Bottom: 8x 4KB sectors at bottom, 1x overlaid 64KB sector at bottom, 511 uniform 64KB sectors
5	Region size as count-1 of 256 Byte units [24 bits] 8x 4KB sectors = 32KB, count = 32KB/256 = 128 value = count - 1 = 128 -1 = 127 = 00007Fh			Reserved = 1111b Supported Sector Type commands bit-field = 0001b Assuming 4KB sector type is assigned to sector type 1 F1h	<- Sector Region 0 only 4KB erase commands supported in this region	
6	Region size as count-1 of 256 Byte units [24 bits] 32KB region, count = 32KB/256 = 128 value = count - 1 = 128 -1 = 127 = 00007Fh			Reserved = 1111b Supported Sector Type commands bit-field = 0010b Assuming 64KB sector type is assigned to sector type 2 F2h	<- Sector Region 1 only 64KB erase commands supported in this region	
7	Region size as count-1 of 256 Byte units [24 bits] 511 x 64KB sectors, count = 33488896 B/256 = 130816 value = count - 1 = 130816 -1 = 130815 = 01FEFFh			Reserved = 1111b Supported Sector Type commands bit-field = 0010b Assuming 64KB sector type is assigned to sector type 2 F2h	<- Sector Region 2 only 64KB erase commands supported in this region	

DWORD	Bits				Notes	
	31:24	23:16	15:8	7:0		
8	Reserved = FFh	Region Count = 3 Regions = 02h	Configuration ID = 01h	Reserved = 111111b Descriptor type = map = 1b Last map = 0b FEh	Configuration Map Header	2nd Configuration Address Map Top: 511x uniform 64KB sectors, 1x overlaid 64KB sector, 8x 4KB sectors at top
9	Region size as count-1 of 256 Byte units [24 bits] 511 x 64KB sectors, count = 33488896 B/256 = 130816 value = count - 1 = 130816 -1 = 130815 = 01FEFFh			Reserved = 1111b Supported Sector Type commands bit-field = 0010b Assuming 64KB sector type is assigned to sector type 2 F2h	<- Sector Region 0 only 64KB erase commands supported in this region	
10	Region size as count-1 of 256 Byte units [24 bits] 32KB region, count = 32KB/256 = 128 value = count - 1 = 128 -1 = 127 = 00007Fh			Reserved = 1111b Supported Sector Type commands bit-field = 0010b Assuming 64KB sector type is assigned to sector type 2 F2h	<- Sector Region 1 only 64KB erase commands supported in this region	
11	Region size as count-1 of 256 Byte units [24 bits] 8x 4KB sectors = 32KB, count = 32KB/256 = 128 value = count - 1 = 128 -1 = 127 = 00007Fh			Reserved = 1111b Supported Sector Type commands bit-field = 0001b Assuming 4KB sector type is assigned to sector type 1 F1h	<- Sector Region 2 only 4KB erase commands supported in this region	
12	Reserved = FFh	Region Count = 1 Regions = 00h	Configuration ID = 02h	Reserved = 111111b Descriptor type = map = 1b Last map = 1b FFh	Configuration Map Header	3rd Configuration Address Map Uniform 64KB sectors
13	Region size as count-1 of 256 Byte units [24 bits] 512 x 64KB sectors, count = 32MB/256 = 131072 value = count - 1 = 131072 -1 = 131071 = 01FFFFh			Reserved = 1111b Supported Sector Type commands bit-field = 0010b Assuming 64KB sector type is assigned to sector type 2 F2h	<- Sector Region 0 only 64KB erase commands supported in this region	

6.5.8 Sector Map Parameter Table – Example 2

The memory device in the following example is 128Mbit density and has one sector map configuration. No configuration detection commands are needed because there is a single fixed sector map. The sector format is:

- Sixteen 4Kbyte sectors at the low address end (bottom) of the device address space, sixteen 4Kbyte sectors at the high address end (top) of the device address space,with all other sectors being 32Kbytes in size.

The Erase Types are defined as:

- Erase Type 1 = 4KB
- Erase Type 2 = 32KB
- Erase Type 3 = 64KB

All three Erase Types are supported in 4KB sector regions at the bottom and top of the address space. Only Erase Types 2 and 3 are supported in the region containing 32Kbyte sectors.

DW	Bits				Notes	
	31:24	23:16	15:8	7:0		
1	Reserved = FFh	Region Count = 3 Regions = 02h	Configuration ID = 00h	Reserved = 111111b Descriptor type = map = 1b Last map = 1b FFh	Configuration Map Header	Configuration Address Map Top and bottom 16x 4KB sectors, remainder uniform 32KB sector map
2	Region size as count-1 of 256 Byte units [24 bits] 16x 4KB sectors = 64KB, count = 64KB/256 = 256 value = count - 1 = 256 -1 = 255 = 0000FFh			Reserved = 1111b Supported Sector Type commands bit-field = 0111b F7h	<- Sector Region 0 4KB, 32KB, and 64KB supported in this region	
3	Region size as count-1 of 256 Byte units [24 bits] 16MB -128KB region, count = 16646144 B/256 = 65024 value = count - 1 = 65024 -1 = 65023 = 00FDFFh			Reserved = 1111b Supported Sector Type commands bit-field = 0110b F6h	<- Sector Region 1 32KB and 64KB erase commands supported in this region	
5	Region size as count-1 of 256 Byte units [24 bits] 16x 4KB sectors = 64KB, count = 64KB/256 = 256 value = count - 1 = 256 -1 = 255 = 0000FFh			Reserved = 1111b Supported Sector Type commands bit-field = 0111b F7h	<- Sector Region 2 4KB, 32KB, and 64KB supported in this region	

6.6 JEDEC 4-byte Address Instruction Table

Legacy SPI memory devices were limited to 128-Mbits (16-Mbytes) of address space by commands that provided only three bytes (24-bits) of address. Recent SPI memories that exceed 128-Mbits density provide various options for providing 4-bytes (32-bits) of address. One option is the use of commands that always provide 4-bytes of address. These commands in some cases have the same function as legacy 3-byte address commands but use a different instruction to indicate that 4-bytes of address follow the instruction. The 4-byte address instruction special function table indicates which 4-byte address command instructions are supported by the SPI memory. The table also provides the 4-byte address instructions for the four Erase Types defined in 8th DWORD of the Basic Flash Parameter Table. If a 4-byte address instruction is not supported for an Erase Type, the instruction for that type is shown in the table as FFh.

6.6.1 4-byte Address Instruction Parameter Header: 1st DWORD

Bits	Description
31:24	**Parameter Table Length** This field specifies how many DWORDs are in the Parameter table. NOTE This field is 1's based. Therefore, 1 indicates 1 DWORD.
23:16	**Parameter Table Major Revision Number** This 8-bit field indicates the major revision number of the parameter table. The value in this field is 1 for this table defined by JESD216B revision. NOTE The Major Revision of JEDEC defined parameter tables can only be modified by updates to this standard.
15:8	**Parameter Table Minor Revision Number** This 8-bit field indicates the minor revision number of the Sector Map parameter table. The value in this field is 0 for this table defined by JESD216B revision. NOTE The Minor Revision of the JEDEC owned parameter tables can only be modified by updates to this standard.
7:0	**Parameter ID LSB** The Sector Map Function Specific Table is assigned the ID LSB of 84h.

6.6.2 4-byte Address Instruction Parameter Header: 2nd DWORD

Bits	Description
31:24	**Parameter ID MSB** The Sector Map Function Specific Table is assigned the ID MSB of FFh.
23:0	**Parameter Table Pointer (PTP)** This address specifies the start of this header's Parameter Table in the SFDP structure. The address is in terms of bytes and must be DWORD-aligned.

6.6.3 4-byte Address Instuction Table, DWORD 1

Bits	Description
31:20	**Reserved**
19	**Support for non-volatile individual sector lock write command, *Instruction=E3h*** 0: Not supported 1: Supported
18	**Support for non-volatile individual sector lock read command, *Instruction=E2h*** 0: Not supported 1: Supported
17	**Support for volatile individual sector lock Write command, *Instruction=E1h*** 0: Not supported 1: Supported
16	**Support for volatile individual sector lock Read command, *Instruction=E0h*** 0: Not supported 1: Supported
15	**Support for (1-4-4) DTR_Read Command, *Instruction=EEh*** 0: Not supported 1: Supported
14	**Support for (1-2-2) DTR_Read Command, *Instruction=BEh*** 0: Not supported 1: Supported
13	**Support for (1-1-1) DTR_Read Command, *Instruction=0Eh*** 0: Not supported 1: Supported
12	**Support for Erase Command – *Type 4 size, Instruction lookup in next Dword*** 0: Not supported 1: Supported
11	**Support for Erase Command – *Type 3 size, Instruction lookup in next Dword*** 0: Not supported 1: Supported
10	**Support for Erase Command – *Type 2 size, Instruction lookup in next Dword*** 0: Not supported 1: Supported
9	**Support for Erase Command – *Type 1 size, Instruction lookup in next Dword*** 0: Not supported 1: Supported
8	**Support for (1-4-4) Page Program Command, *Instruction=3Eh*** 0: Not supported 1: Supported
7	**Support for (1-1-4) Page Program Command, *Instruction=34h*** 0: Not supported 1: Supported
6	**Support for (1-1-1) Page Program Command, *Instruction=12h*** 0: Not supported 1: Supported
5	**Support for (1-4-4) FAST_READ Command, *Instruction=ECh*** 0: Not supported 1: Supported

Bits	Description
4	**Support for (1-1-4) FAST_READ Command, *Instruction=6Ch*** 0: Not supported 1: Supported
3	**Support for (1-2-2) FAST_READ Command, *Instruction=BCh*** 0: Not supported 1: Supported
2	**Support for (1-1-2) FAST_READ Command, *Instruction=3Ch*** 0: Not supported 1: Supported
1	**Support for (1-1-1) FAST_READ Command, *Instruction=0Ch*** 0: Not supported 1: Supported
0	**Support for (1-1-1) READ Command, *Instruction=13h*** 0: Not supported 1: Supported

6.6.4 4-byte Address Instuction Table, DWORD 2

NOTE (informative) Industry common usage is:

21h 4 kbyte erase
5Ch 32 kbyte erase
DCh 64 kbyte erase
DCh 256 kbyte erase

Bits	Description
31:24	**Instruction for Erase Type 4** Erase Type is defined in 9th DWORD of the Basic Flash Parameter Table
23:16	**Instruction for Erase Type 3** Erase Type is defined in 9th DWORD of the Basic Flash Parameter Table
15:8	**Instruction for Erase Type 2** Erase Type is defined in 8th DWORD of the Basic Flash Parameter Table
7:0	**Instruction for Erase Type 1** Erase Type is defined in 8th DWORD of the Basic Flash Parameter Table

7 Rules for Header and Table Additions and Modifications

- Additional headers and parameter tables can be added by vendors without JEDEC approval.
- The first four DWORDs of the 6.4 JEDEC Flash Parameters Table can never be modified.
- New headers must be built using exactly two DWORDs and they must immediately follow the existing header(s).
- Minimum parameter table size is one DWORD. The maximum parameter table size is not specified.
- Parameter tables may be located anywhere in the SFDP space. They do not need to immediately follow the parameter headers.
- Overlapping parameter tables are permitted.

8 Legacy Compatibility

Prior to the release of this standard, Intel published SFDP guidelines with a four DWORD parameter table. The first four DWORDs of the JEDEC Basic Parameter Table are identical to the table in Intel's guidelines. Devices in production prior to the release of the initial JESD216 standard might only contain these four DWORDs.

Revision A increased the number of DWORDs from nine to sixteen. The first nine DWORDs in Revision A maintain backwards compatibility. Devices in production prior to the release of this revision may not contain all of the currently defined DWORDs.

Revision B continues to maintain backwards compatibility of the Basic Parameter Table. Optional Function Specific tables for Sector Map Parameters and 4-Byte address commands are added. See Annex D for revision history details.

Annex A (informative)Example SFDP Discovery Code

```
// C-syntax pseudo code for discovering SFDP table
// This code is provided as an example. It is not optimized.
// Code searches flash for the highest revision table that the driver
supports.
// Code assumes that driver can support all revisions up to and including
// some maximum supported revision.
// Use three functions:
// spi()            performs a SPI flash operation
// update_current() updates global Parameter Header variables from the
//                  data buffer of bytes read from flash
// find_table()     checks for a valid SFDP header in flash and sets
//                  global Parameter Header variables to the
//                  highest revision supported by the driver

#include <stdbool.h>
#define JEDEC_TABLE_ID 0
#define READ_SFDP      0x5A

int ms_major_rev = 1;       // maximum major revision supported by driver
int ms_minor_rev = 0;       // maximum minor revision supported by driver

int NPH = 0;                // Number of Parameter Headers
int sfdp_major_revision = 0;
int sfdp_minor_revision = 0;

int curr_major_rev = 0; // current Parameter Table revision
int curr_minor_rev = 0; // current Parameter Table revision
int curr_PTP = 0;       // current Parameter Table Pointer
int curr_length = 0;    // current parameter table length in dwords
int curr_address = 0;   // current address

bool table_found = false;

#define MAX_BYTES 128
unsigned char data[MAX_BYTES];

extern void
spi(int opcode, int address, int byte_count, unsigned char *buffer);
// the implementation of spi() is flash-controller dependent

void
update_current()
{
  curr_minor_rev = data[1];
  curr_major_rev = data[2];
  curr_length    = data[3];
  curr_PTP       = (data[6] << 16) | (data[5] << 8) | (data[4]);
}

bool
find_table()
{
```

```
  // Read the first 8 bytes of the SFDP header. If the device does not
  // support SFDP it will not drive any return data to the controller.
  // This example code does not make use of the SFDP major/minor revision.
  spi(READ_SFDP, curr_address, 8, data);

  // check signature
  if (!(data[0] == "S" &&
        data[1] == "F" &&
        data[2] == "D" &&
        data[3] == "P"))
  {
    return false;
  }

  NPH = data[6];
  sfdp_major_revision = data[5];
  sfdp_minor_revision = data[4];

  // search for highest revision JEDEC-standard table in flash device
  // loop over all parameter headers
  while (NPH >= 0)
  {
    curr_address = curr_address + 8;

    spi(READ_SFDP, curr_address, 8, data);

    if (data[0] == JEDEC_TABLE_ID)
    {
      // if the major revision is newer then minor revision is don't care
      if (data[2] >  curr_major_rev &&
          data[2] <= ms_major_rev     )
      {
        update_current();
        table_found = true;
      }
      // if the major revision is the same then use newer minor revision
      else if (data[2] == curr_major_rev &&
               data[2] <= ms_major_rev   &&
               data[1] >  curr_minor_rev &&
               data[1] <= ms_minor_rev     )
      {
        update_current();
        table_found = true;
      }
    } // end if JEDEC TABLE ID

    NPH = NPH - 1;
  } // end while NPH

  // read the parameter table into the data buffer, converting
  // dword count in curr_length to byte count
  if (table_found)
  {
    spi(READ_SFDP, curr_PTP, curr_length * 4, data);
  }
 return table_found;
}
```

Annex B (informative)Example SFDP Sector Map Discovery Code

```
/* C-syntax pseudo code for discovering sector architecture from the SFDP
Sector Map Parameter Table. This code is provided as an example. It is not
optimized.
*/

#include <stdio.h>

/* The example of how to find a table has been given in JESD216 Annex A. This
example assumes the Parameter Table Pointer (PTP) and the table length are
already aquired by code like that in Annex A.
*/

unsigned int  sector_info_length;    // table length in DWORDs
unsigned int  sector_info_PTP;       // DWORD aligned address


/* Define the Sector Map data structure (region_info) that will be loaded
from the SFDP Sector Map Parameter Table by this pseudo code example. The SPI
Flash driver software may then use this region information in making
decisions that need informaion on the sector sizes, locations, and erase
types supported in each address region within the SPI memory.
*/

#define MAX_BYTES 128
#define MAX_REGION 4

typedef struct
{
   unsigned char region_no;       // region index in this device
   unsigned char supported_ET;    // bit field indicator of Erase Type
   unsigned int  region_size;     // size of the region
} region_info_t;

region_info_t region_info[MAX_REGION];
unsigned char number_of_regions;

/* Build an example Sector Map Table to be read by the example code. The
define selects which example Sector Map Parameter Table to load into the
sector_info array for testng the code.
*/

#if 1
// Example one
unsigned char sector_info[MAX_BYTES] =
{
   0xFC, 0x65, 0xFF, 0x08,
   0x04, 0x00, 0x80, 0x00,
   0xFD, 0x35, 0x20, 0x04,
   0xFF, 0xFF, 0xFF, 0xFF,
   0xFE, 0x00, 0x02, 0xFF,
   0xF1, 0x7F, 0x00, 0x00,
   0xF2, 0x7F, 0x00, 0x00,
```

```
   0xF2, 0xFF, 0xFE, 0x01,
   0xFE, 0x01, 0x02, 0xFF,
   0xF2, 0xFF, 0xFE, 0x01,
   0xF2, 0x7F, 0x00, 0x00,
   0xF1, 0x7F, 0x00, 0x00,
   0xFF, 0x02, 0x00, 0xFF,
   0xF2, 0xFF, 0xFF, 0x01
};

#else
// Example two
unsigned char sector_info[MAX_BYTES] =
{
   0xFF, 0x00, 0x02, 0xFF,
   0xF7, 0xFF, 0x00, 0x00,
   0xF6, 0xFF, 0xFD, 0x00,
   0xF7, 0xFF, 0x00, 0x00
};
#endif

/* Function to determine the address length needed by a Configuration
Detection Command
*/

// These are just pseudo values. They will be OS specific in real code
#define NO_ADDRESS         0
#define THREE_BYTE_ADDRESS 3
#define FOUR_BYTE_ADDRESS  4
#define USE_CURRENT        5

unsigned char get_address_cycle(unsigned char instruction_format)
{
   unsigned char address_length;

   switch ((instruction_format & 0xC0) >> 6)    // Take Bit7:6
   {
      case 0x0:
         address_length = NO_ADDRESS;           // No address cycle
         break;
      case 0x1:
         address_length = THREE_BYTE_ADDRESS;   // 3-byte address
         break;
      case 0x2:
         address_length = FOUR_BYTE_ADDRESS;    // 4-byte address
         break;
      case 0x3:
         address_length = USE_CURRENT;         // use current setting
         break;
      default:
         address_length = USE_CURRENT;
         break;
   }
   return address_length;
}
```

```
/* Function to read the Sector Map Parameter Table, execute Configuration
Detection commands if any are provided to detect more than one configuration
option, and read the selected Sector Map Configuration into a region
information data structure for later use by the Flash memory driver software.
*/

unsigned int populate_region_info()
{
   unsigned char current_table_index = 0;
   unsigned char region_info_index = 0;

   unsigned int  i;
   unsigned char instruction;
   unsigned int  address;
   unsigned char cycle;
   unsigned char latency_cycle;
   unsigned char data_byte;
   unsigned char data_mask;
   unsigned char current_bit;

   unsigned char read_count = 0;

/* It is assumed that there is an already defined function for reading from
the SPI memory = spi_read()the implementation of spi_read() is flash-
controller dependent.

To read the Sector Map Parameter Table from a real memory, the following
comments would be changed to exectutable code to declare the SPI read
function and read the entire Sector Map Parameter Table into an array for
parsing.
*/
/*
   extern void
   spi_read(int opcode, int address, int byte_count, unsigned char *buffer);
   #define READ_SFDP      0x5A
   spi_read(READ_SFDP, sector_info_PTP, sector_info_length*4, sector_info);
*/

   number_of_regions = 1;    // at least one region in any device

/* Determine if there is more than one Sector Map Configuration by first
looking for any Sector Map Configuration Detection descriptors. Examine the
first byte of the Sector Map Parameter Table and mask for bit location 1. If
bit location 1 is a zero, the first two DWORDs contain a Configuration
Detection Command Descriptor that has the information needed to create a
command to the SPI memory, that will read out one byte of status or
configuration register information, and mask to read one bit from that
register, to build an index value that is used to select the Sector Map
Configuration that is currently in use. If there is more than one
Configuration Detection descriptor the process is repeated by the While loop,
until the number of bits needed to select one Sector Map Configuration
Descriptor are accumulated into the map descriptor region_index value.
*/
```

```
   while ((sector_info[current_table_index] & 0x2) == 0x0)
   {
      instruction = sector_info[current_table_index+1];
      address = *( (unsigned int *)&sector_info[current_table_index+4]);
      cycle = get_address_cycle(sector_info[current_table_index+2]);
      latency_cycle  = sector_info[current_table_index+2] & 0xF;
      data_mask = sector_info[current_table_index+3];

      // read a byte of status or configuration register information from the
      // memory to build the region_index value
      // spi_read(instruction, address, cycle, latency_cycle, &data_byte);

      // for test only
      data_byte = 0;

      // construct the region_index value
      if ((data_byte&data_mask) == 0)
         current_bit = 0;
      else
         current_bit = 1;

      region_info_index = (region_info_index << 1) | current_bit;

      // move on to the next Dword
      current_table_index += 8;
   }

   // All Configuration Detection Command Decriptors have been processed,
   // if any were provided.

/* Search for the correct Sector Map Configuration Descriptor by comparing
the region_info_index to each Configuration Map Descriptor Configuration ID
and adjust the current_table_index to point to the selected Sector Map
Configuration Descriptor.
*/

   // finding the matching configuration map
   //
   // check to see if the Configuration ID matches the region_info_index
   while (sector_info[current_table_index+1] != region_info_index)
   {
      // check if this is the last map
      if ( (sector_info[current_table_index] & 0x1) == 0x1)
      {
         // last map reached, no map is found, return error
         return -1;
      }

      // increment the table index to the next map
      current_table_index += (sector_info[current_table_index+2]+2)*4;
   }

   // Map found. Now populate the data structure.
   //
   // Identify how many regions are in the sector configuraion currently in
```

```
   // use
   number_of_regions = sector_info[current_table_index+2]+1;

   // populate the region_info data structure from the selected
   // Sector Map Configuration Descriptor

   for (i=0;i<number_of_regions;i++)
   {
      current_table_index += 4;

      region_info[i].region_no = i;
      region_info[i].supported_ET = sector_info[current_table_index] & 0xF;
      region_info[i].region_size  = ((*( (unsigned int
*)&sector_info[current_table_index]) >> 8) + 1)*256;
   }

   return 0;
}

/* This is a test program only for reading the examples. This main() can be
removed when using a real memory and replaced by the Flash Driver code that
calls the above Sector Map extraction functions.
*/

int main()
{
   unsigned int  i;

   if (populate_region_info() != 0)
   {
      printf("Finding region info returns error\n");
      // should take some action here
   }

   printf("nubmer of regions : %d\n",  number_of_regions);
   for (i=0;i<number_of_regions;i++)
   {
      printf("Region %d : region_no=%d, supported_ET=0x%x, region_size=0x%x",
i,
            region_info[i].region_no, region_info[i].supported_ET,
region_info[i].region_size);
      printf("\n");
   }

}
```

Annex C (Informative) Procedure For Requesting Function Specific ID

The Function Specific ID list is not a fixed listing. Any company may request a Function Specific ID by making a request to the JEDEC office at juliec@jedec.org. Please include "Function Specific ID Request, JESD216" in the email subject line. Upon receipt of the email the request will be forwarded to the JC-42.4 Chair and the JC-42.4 TG Chair for committee consideration. The chair will then respond to the request. Updates to the list will be made periodically.

The SFDP Standard will allow Serial Flash vendors to describe the functions and features of their devices in a standard set of internal parameter tables. These internal parameter tables can be read by users to determine the characteristics of the device.

Annex D (informative) Differences between revisions

This annex briefly describes most of the changes made to entries that appear in this standard, JESD216B, comparded to its predecessors, JESD216A (July 2013) and ESD216 (April 2011).

D.1 Differnces between JESD216B and JESD216A

- Added a definition for Sector and Block to clarify terminology
- Changed Sector Type nomenclature to Erase Type, to clarify that the erase operation type is independent of the minimum erase granularity of the sectors affected by the erase operation e.g. several erase types (sizes of erase operations e.g. 8KB, 32KB, 64KB) may be applied to sectors of a smaller granularity than the erase operation (e.g. 4KB sectors).
- Clarified parameter ID LSB parity being located in the most significant Bit of the Byte
- Added the Function Specific Parameter Table ID assignments
- Added the Header for the JEDEC Basic Flash Parameter Table, separate from the general description for Parameter Table Headers. Removed references to the Basic Flash Parameter Table in the general Parameter Table Header description. Incremented the JEDEC Basic Flash Parameter Table Minor Revision to 6 as an indication that there has been an additional definition for a previously reserved bits: DWORD 15[18, 14, & 8].
- Removed use of the terms "sector" or "block" in relationship to erase operations, instead using the term Erase Type
- DW11, [31:24] Clarified the Chip Erase time applies separately to each die for multi-die devices in which the dice are individually accessed.
- DW12, [23:20], [12:9], Added comments that suspend can be issued at any time, there is no required minimum timing. However, … this parameter recommends an average resume to suspend interval so the operation can make progress
- DW15, [23], Updated description and replaced WP by RESET
- DW15, [8:4], added 4-4-4 enable option
- Added a section with the optional Function Specific Sector Map Parameter Tables
- Added a section with the optional Function Specific 4-Byte Address Instruction Tables
- Added Annex B code example for Sector Map Parameter Table reading and moved previous Annex B and Annex C to be Annex C and Annex D

Annex D (informative) Differences between revisions (cont'd)

D.2 Differnces between JESD216A and JESD216

- Extensive rewrite to clarify requirements, but functionality of JESD216 has been maintained.
- The JESD216A parameter table is marked with Major Revision 1 and Minor Revision 5. Using Minor Revision 5 instead of 1 is required to avoid version conflicts with legacy devices which implemented SFDP tables prior to JEDEC standardization.
- Increased the number of DWORDs in the Basic Parameters Table from nine to sixteen.
- Provided hooks for future expansion using Function Specific Tables.
- Added Annex A Example code for SFDP discovery
- Added Annex B Procedure for requesting a Function Specific ID

Standard Improvement Form JEDEC JESD216B

The purpose of this form is to provide the Technical Committees of JEDEC with input from the industry regarding usage of the subject standard. Individuals or companies are invited to submit comments to JEDEC. All comments will be collected and dispersed to the appropriate committee(s).

If you can provide input, please complete this form and return to:

JEDEC
Attn: Publications Department
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

Fax: 703.907.7583

1. I recommend changes to the following:

☐ Requirement, clause number _____

☐ Test method number _____ Clause number _____

The referenced clause number has proven to be:

☐ Unclear ☐ Too Rigid ☐ In Error

☐ Other _____

2. Recommendations for correction:

_____

_____

_____

_____

3. Other suggestions for document improvement:

_____

_____

_____

_____

Submitted by

Name: _____ Phone: _____

Company: _____ E-mail: _____

Address: _____

City/State/Zip: _____ Date: _____


JEDEC®