To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 11 December 2003
Subject: 03-388r0 SBC-2 Nonvolatile caches

Revision history
Revision 0 (11 December 2003) First revision

Related documents
sbc2r11 - SCSI Block Commands - 2 revision 11

Overview
SBC-2 has several tools that understand the concept of caches:

a) a force unit access (FUA) bit in each read and write command
b) LOCK UNLOCK CACHE, PRE-FETCH, PREVENT ALLOW MEDIUM REMOVAL, and SYNCHRONIZE CACHE commands which control cache usage
c) Caching mode page

However, it is does not comprehend the concept of a non-volatile cache. RAID controllers often employ battery-backed caches that can preserve the data for hours or days.

In many cases, when software wants to flush the cache with SYNCHRONIZE CACHE because it is concerned about temporary power loss, it is not necessary for the RAID controller to do anything; data is secure (at least for for a few days). In other cases, however, it does want the RAID controller flush its non-volatile cache to media.

For write commands, if force unit access is enabled, sometimes software really means the the media must be accessed, while other times it is sufficient to guarantee that it is in a non-volatile cache.

Forcing data to the media is useful when disk drives in a RAID set are going to be moved to another controller. They need to hold a coherent set of data - some of the data cannot be left in a write cache. It also is appropriate when shutting down a server for extended periods of time. It is also useful for removeable media.

Allowing data to remain in non-volatile cache is acceptable if short power loss is expected or when a system is being rebooted. It may also be useful for metadata writes. An unexpected power loss causes more trouble if it loses metadata than if it loses data itself (may disrupt one file vs. many files).

Some operating systems overuse SYNCHRONIZE CACHE and force unit access, which hinders performance as the caches get larger and larger.

Proposal

Define the current behavior as:

a) SYNCHRONIZE CACHE only needs to flush to non-volatile cache, not to the physical medium
b) FUA bit set to 1 on reads and writes forces access to the non-volatile cache, not to the physical medium

Add a bit to SYNCHRONIZE CACHE to force synchronization to the physical medium.

Add a FUA_PHYS bit next to the FUA bit that forces data to the physical medium.

Add a way to report how much battery life remains for a non-volatile cache (if known) and what the maximum possible battery life is.

Add a bit to the Caching mode page to enable/disable write caching for volatile and non-volatile caches separately.

Suggested changes

3.1.3 cache memory: A temporary (and often volatile) data storage area outside the user-accessible area that may contain a subset of the data stored in the non-volatile data storage area. A cache memory is usually

faster to access than the medium and thus has the effect of increasing data throughput by reducing the number of accesses to the medium.

3.1.16 non-volatile medium: A physical storage medium that retains data written to it for a subsequent read operation through power off/on cycles. An example of this is a disk within a device that stores data as magnetic field changes that do not require device power to exist.

3.1.29 volatile medium: Medium that does not retain data written to it for a subsequent read operation through power off/on cycles. An example of this is a silicon memory device that loses data written to it if device power is lost.

0.0.1 nonvolatile cache memory: Cache memory that retains data through power cycles.

0.0.2 volatile cache memory: Cache memory that does not retain data through power cycles.

4.1 SCSI block device models

4.1.1 Direct-access device type model

4.1.1.1 Direct-access device type model overview

Direct-access block devices store blocks of data for later retrieval. Each block of data is stored at a unique logical block address. An application client issues WRITE commands to store the blocks of data (write operations) and READ commands to retrieve the blocks of data (read operations). Other commands issued by the application client may also cause write and read operations to occur. A write operation causes one or more blocks of data to be written on the medium. A read operation causes one or more blocks of data to be read from the medium. A verify operation confirms that one or more blocks of data were correctly written and can be read without error from the medium.

Blocks of data are stored by a process that causes localized changes or transitions within the medium. The changes made to the medium to store the blocks of data may be volatile (i.e., not retained through power cycles) or non-volatile (i.e., retained through power cycles). The medium may be divided in parts that are used for data blocks, parts that are reserved for defect management, and parts that are reserved for use by the controller for the management of the block device.

4.1.1.2 Removable medium

4.1.1.2.1 Removable medium overview

The medium may be removable (e.g., used in a floppy disk device) or non-removable (e.g., used in a fixed disk device). The removable medium may be contained within a cartridge (or jacket) to prevent damage to the recording surfaces. The combination of medium and cartridge is often called a removable volume.

A removable volume has an attribute of being mounted or de-mounted on a suitable transport mechanism. A removable volume is mounted when the direct access block device is capable of performing write or read operations to the medium. A mounted removable volume may not be accessible by an initiator if it is reserved by another initiator. A removable volume is de-mounted at any other time (e.g., during loading, unloading, or storage).

An application client may check whether a removable volume is mounted by issuing a TEST UNIT READY command. A volume that is loaded may need a START STOP UNIT command issued to become accessible for write or read operations.

The PREVENT ALLOW MEDIUM REMOVAL command allows an application client to restrict the demounting of the removable volume. This is useful in maintaining system integrity. If the direct-access block device implements cache memory, it ensures that all logical blocks of the medium contain the most recent data prior to permitting demounting of the removable volume. If the application client issues a START STOP UNIT command to eject the removable volume, and the direct-access block device is prevented from demounting by the PREVENT ALLOW MEDIUM REMOVAL command, the START STOP UNIT command is rejected by the device server.

4.2.1.5 Initialization

Direct-access block devices may require initialization prior to write or read operations. This initialization is performed by a FORMAT UNIT command (see 5.2.2). Parameters related to the geometry and performancecharacteristics may be set with the MODE SELECT command prior to the format operation. Some block devices are initialized by means not specified in this standard. The time when the initialization occurs is specific to the implementation of the direct-access block device.

Block devices using a non-volatile medium may save the parameters and only need to be initialized once. However, some mode parameters may need to be initialized after each logical unit reset. A catastrophic failure of the direct-access block device may require the FORMAT UNIT command to be reissued.

Block devices that use a volatile medium may need to be initialized after each logical unit reset prior to the execution of read or write operations. Mode parameters may also need initialization after logical unit resets.

4.2.1.8 Cache memory

Some direct-access block devices implement cache memory. A cache memory is usually an area of temporary storage in the direct-access block device with a fast access time that is used to enhance performance. It exists separately from the blocks of stored data and is not directly accessible by the application client. Use of cache memory for write or read operations may reduce the access time to a logical block and can increase the overall data throughput.

Cache memory may be volatile or non-volatile. Volatile caches do not retain data through power cycles. Nonvolatile caches retain data through power cycles. There may be a limit on the amount of time a nonvolatile cache is able to retain data.

During read operations, the direct-access block device uses the cache memory to store blocks of data that the application client may request at some future time. The algorithm used to manage the cache memory is not part of this standard. However, parameters are provided to advise the device server about future requests, or to restrict the use of cache memory for a particular request.

During write operations, the direct-access block device uses the cache memory to store data that is written to the medium at a later time. This is called write-back caching. The command may complete prior to blocks of data being written to the medium. As a result of using a write-back caching there is a period of time when the data may be lost if power to the device is lost or a hardware failure occurs. There is also the possibility of an error occurring during the subsequent write operation. If an error occurred during the write, it may be reported as a deferred error on a later command. The application client may request that write-back caching be disabled to prevent detected write errors from being reported by deferred errors. Even with write-back caching disabled undetected write errors may occur. In order to detect these errors, verify commands are provided.

When the cache memory fills up with blocks of data that are being kept for possible future access, new blocks of data that are to be kept replace those currently in cache memory. The disable page out (DPO) bit allows the application client to influence the replacement of logical blocks in the cache. For write operations, setting this bit to one advises the device server to not replace existing blocks in the cache memory with the write data. For read operations, setting this bit to one causes blocks of data that are being read to not replace existing ones in the cache memory.

Sometimes the application client may want to have the blocks of data read from the medium instead of from the cache memory. The force unit access (FUA) bit is used to indicate that the device server shall access the physical medium. For a write operation, setting FUA to one causes the device server to complete the data write to the physical medium before completing the command. For a read operation, setting FUA to one causes the logical blocks to be retrieved from the physical medium.

When the DPO and FUA bits are both one, write and read operations, in effect, bypass the cache memory.

When a VERIFY command is executed, a forced unit access is implied, since the blocks of data stored on the medium are being verified. Furthermore, a SYNCHRONIZE CACHE operation is also implied to write unwritten blocks of data still in the cache memory. These blocks of data are stored on the medium before the verify operation begins. The DPO bit is provided since the VERIFY command may cause the replacement of blocks in the cache. The caching rules also applies to the WRITE AND VERIFY command.

Commands may be implemented by the device server that allow the application client to control other behavior of the cache memory:

a) the LOCK UNLOCK CACHE command (see 5.2.3) controls whether certain logical blocks shall be held in the data cache for future use. Locking a logical block prevents its replacement by a future access. Unlocking a logical block exposes it to possible replacement by a future access;
b) the PRE-FETCH command (see 5.2.5) causes a set of logical blocks requested by the application client to be read into the data cache for possible future access. The blocks fetched are subject to later replacement unless they are locked;
c) the SYNCHRONIZE CACHE command (see 5.2.23) forces any pending write data in the requested set of logical blocks to be stored in the physical medium. This command may be used to ensure that the data was written and any detected errors reported;
d) the Caching mode page (see 6.1.3.2) writeable by the MODE SELECT command allows control of cache behavior and handles certain basic elements of cache replacement algorithms.

5.2.8 READ (10) command

The READ (10) command (see table 34) requests that the device server transfer data to the application client. Data includes user data and protection information, if any. The most recent data value written in the addressed logical block shall be returned.

Table 1 — READ (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	RDPROTECT			DPO	FUA	FUA_PHYS	Reserved	RELADR
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See 4.2.1.9 for reservation requirements for this command. See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the RELADR bit and the LOGICAL BLOCK ADDRESS field.

A disable page out (DPO) bit of zero indicates the priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page. A DPO bit of one indicates that the device server shall assign the logical blocks accessed by this command the lowest priority for being fetched into or retained by the cache. A DPO bit of one overrides any retention priority specified in the Caching mode page (see 6.1.3.2). All other aspects of the algorithm implementing the cache memory replacement strategy are not defined by this standard.

NOTE 1 - The DPO bit is used to control replacement of logical blocks in the cache memory when the host has information on the future usage of the logical blocks. If the DPO bit is one, the host is indicating that the logical blocks accessed by the command are not likely to be accessed again in the near future and should not be put in the cache memory nor retained by the cache memory. If the DPO bit is zero, the host is indicating that the logical blocks accessed by this command are likely to be accessed again in the near future.

~~A force unit access (FUA) bit of zero indicates that the device server may satisfy the command by accessing the cache memory. For read operations, any logical blocks that are contained in the cache memory may be transferred to the application client directly from the cache memory. For write operations, logical blocks may be transferred directly to the cache memory. GOOD status may be returned to the application client prior to writing the logical blocks to the medium. Any error that occurs after the GOOD status is returned is a deferred error, and information regarding the error is not reported until a subsequent command.~~

~~A (FUA) bit of one indicates that the device server shall access the media in performing the command prior to returning GOOD status. Read commands shall access the specified logical blocks from the media (i.e., the data is not directly retrieved from the cache). If the cache contains a more recent version of a logical block than the media, the logical block shall first be written to the media. Write commands shall not return GOOD status until the logical blocks have actually been written on the media (i.e., the data is not write cached). Read commands that cause data to be written to the media from cache and that encounter an error shall cause a deferred error to be reported. See SPC-3.~~

The force unit access (FUA and FUA_PHYS) bits are specified in table 7.

Table 2 — Force unit access for reads

FUA_PHYS	FUA	**Description**
0	0	The device server may read the logical blocks from volatile cache, non-volatile cache, and/or the physical medium. It should read the logical blocks from a cache if available.
0 or 1	1	The device server shall read the logical blocks from non-volatile cache or the physical medium. If a volatile cache contains a more recent version of a logical block, the device server shall first write the logical block to: a) the non-volatile cache, if a non-volatile cache is present; or b) the non-volatile cache, if a non-volatile cache is present, and the physical medium; before reading it.
1	0	The device server shall read the logical blocks from the physical medium. If a cache contains a more recent version of a logical block, the device server shall first write the logical block to the non-volatile cache (if present) and the physical medium before reading it.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred. A TRANSFER LENGTH of zero indicates that no logical blocks shall be transferred. This condition shall not be considered an error. Any other value indicates the number of logical blocks that shall be transferred. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.1.4.2).

NOTE 2 - For the READ (6) command, a TRANSFER LENGTH of zero indicates that 256 logical blocks are transferred.

[READ (12) - add FUA_PHYS bit to byte 1 bit 2]

[READ (16) - add FUA_PHYS bit to byte 1 bit 2]

[REBUILD (16) - command expected to be obsoleted. Otherwise, add FUA_PHYS bit.]

[REBUILD (32) - command expected to be obsoleted.]

[REGENERATE (16) - command expected to be obsoleted.]

[REGENERATE (32) - command expected to be obsoleted.]

5.2.24 START STOP UNIT command

5.2.24.1 START STOP UNIT description

The START STOP UNIT command provides an application client a method to control the power condition of a logical unit (see 4.2.4). This includes specifying that the device server enable or disable the block device for media access operations by controlling certain power conditions and timers.

Logical units that contain cache memory shall write all cached data to the medium for the logical unit (as a logical unit would do in response to a SYNCHRONIZE CACHE command (see 5.2.25)) prior to entering into

any power condition that prevents accessing the media (e.g., before a hard drive stops its spindle motor during transition to the stopped power condition).

5.2.25 SYNCHRONIZE CACHE (10) command

The SYNCHRONIZE CACHE (10) command (see table 3) ensures that logical blocks in the cache memory, within the specified range, have their most recent data value recorded on the physical medium. If a more recent data value for a logical block within the specified range exists in the cache memory than on the physical medium, then the logical block from the cache memory shall be written to the physical medium. Logical blocks may not be removed from the cache memory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard.

Table 3 — SYNCHRONIZE CACHE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (35h)							
1	Reserved					SYNC_NV	IMMED	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	(MSB)							
8	NUMBER OF BLOCKS							(LSB)
9	CONTROL							

See 4.2.1.9 for reservation requirements for this command.

The SYNC_NV bit specifies whether the device server is required to synchronize volatile and non-volatile caches and is described in table 4..

Table 4 — SYNC_NV bit

SYNC_NV	Device server requirement to synchronize	
	Volatile cache	Non-volatile cache
0	shall	should not
1	shall	shall

An immediate (IMMED) bit of zero indicates that the status shall not be returned until the operation has been completed. An IMMED bit of one indicates that the device server shall return status as soon as the command descriptor block has been validated. If the IMMED bit is one and the device server does not support the IMMED bit, the command shall terminate with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the RELADR bit and the LOGICAL BLOCK ADDRESS field.

The NUMBER OF BLOCKS field specifies the total number of contiguous logical blocks within the range. A number of blocks of zero indicates that all remaining logical blocks on the block device shall be within the range.

A logical block within the specified range that is not in cache memory is not considered an error.

5.2.26 SYNCHRONIZE CACHE (16) command

The SYNCHRONIZE CACHE (16) command (see table 5) ensures that logical blocks in the cache memory, within the specified range, have their most recent data value recorded on the physical medium. If a more recent data value for a logical block within the specified range exists in the cache memory than on the physical medium, then the logical block from the cache memory shall be written to the physical medium. Logical blocks may not be removed from the cache memory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard.

Table 5 — SYNCHRONIZE CACHE (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (91h)							
1	Reserved					SYNC_NV	IMMED	Reserved
2	(MSB)							
9	LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	NUMBER OF BLOCKS							(LSB)
14	Reserved							
15	CONTROL							

See 4.2.1.9 for reservation requirements for this command. See the SYNCHRONIZE CACHE (10) command (see 5.2.25) for a description of the fields in this command.

5.2.31 WRITE (10) command

The WRITE (10) command (see table 6) requests that the device server write the data transferred by the application client to the medium.

Table 6 — WRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	WRPROTECT			DPO	FUA	EBP	~~Reserved~~ FUA_PHYS	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	(MSB)							
8	TRANSFER LENGTH							(LSB)
9	CONTROL							

See 4.2.1.9 for reservation requirements for this command. See the READ (10) command (see 5.2.8) for a definition of the ~~cache control bits (~~DPO bit~~and FUA)~~. See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the RELADR bit and the LOGICAL BLOCK ADDRESS field.

The force unit access (FUA and FUA_PHYS) bits are specified in table 7.

Table 7 — Force unit access for writes

FUA_PM	FUA	Description
0	0	The device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the physical medium. If a cache is present and in write-back mode, it should not write the logical blocks to the physical medium.
0 or 1	1	The device server shall write the logical blocks to non-volatile cache and/or the physical medium. If a non-volatile cache is present and in write-back mode, it should not write the logical blocks to the physical medium.
1	0	The device server shall write the logical blocks to the physical medium, and shall not return GOOD status until the logical blocks have actually been written on the medium.

If logical blocks are transferred directly to a cache memory, GOOD status may be returned to the application client prior to writing the logical blocks to the physical medium. Any error that occurs after the GOOD status is returned is a deferred error, and information regarding the error is not reported until a subsequent command.

An erase by-pass (EBP) bit of zero indicates that the block device shall default to the normal write operation. An EBP bit of one indicates that the device server is allowed to by-pass the erase operation prior to writing the data. For direct access block devices and write-once block devices, the EBP bit shall be considered reserved.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred. A TRANSFER LENGTH of zero indicates that no logical blocks shall be transferred. This condition shall not be considered an error and no data shall be written. Any other value indicates the number of logical blocks that shall be transferred. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.1.4.2).

NOTE 3 - For the WRITE (6) command, a TRANSFER LENGTH of zero indicates that 256 logical blocks are transferred.

[WRITE (12) - add FUA_PHYS to byte 1 bit 1, point to WRITE (10)]

[WRITE (16) - add FUA_PHYS to byte 1 bit 1, point to WRITE (10)]

[WRITE AND VERIFY commands - already have an implicit force unit access]

[XDWRITE (16) - add FUA_PHYS, point to WRITE (10)]

[XDWRITE (32) - add FUA_PHYS, point to WRITE (10)]

[XDWRITEREAD (32) - add FUA_PHYS, point to WRITE (10) and READ (10)]

[XDWRITE EXTENDED (16) - command expected to be obsoleted. Otherwise, add FUA_PHYS, point to WRITE (10)]

[XDWRITE EXTENDED (32) - command expected to be obsoleted.]

[XDWRITE EXTENDED (64) - command expected to be obsoleted.]

[XPWRITE (10) - add FUA_PHYS to byte 1 bit 1, point to WRITE (10)]

[XPWRITE (32) - add FUA_PHYS to byte 10 bit 1, point to WRITE (10)]

[READ UPDATED BLOCK - command expected to be obsoleted. Otherwise, add FUA_PHYS, point to READ (10)]

6.1.2 Log parameters

6.1.2.1 Log parameters overview

This subclause defines the descriptors and pages for log parameters used with direct-access block devices. See SPC-3 for a detailed description of logging operations. The log page codes for direct-access block devices are defined in table 8.

Table 8 — Log page codes

Log page code	Description	Reference
00h	Supported log pages	SPC-3
01h	Buffer Overrun/Underrun log page	SPC-3
02h	Write Error Counter log page	SPC-3
03h	Read Error Counter log page	SPC-3
04h	Reserved	
05h	Verify Error Counter log page	SPC-3
06h	Non-Medium Error log page	SPC-3
07h	Last N Error Events log page	SPC-3
08h	Format Status log page	6.1.2.2
09h	Nonvolatile Cache log page	6.1.2.x
~~09h~~ 0Ah	Reserved	
0Bh	Last N Deferred Error Events log page	SPC-3
0Ch	Reserved	
0Dh	Temperature log page	SPC-3
0Eh	Start-Stop Cycle Counter log page	SPC-3
0Fh	Application Client log page	SPC-3
10h	Self-Test Results log page	SPC-3
11h - 2Eh	Reserved	
2Fh	Information Exceptions log page	SPC-3
30h - 3Eh	Vendor-specific log pages	
3Fh	Reserved	

6.1.2.2 Nonvolatile Cache log page [new section]

The Nonvolatile Cache log page defined in table 9 indicates the status of battery backup for a nonvolatile cache.

Table 9 — Nonvolatile Cache log page

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (xxh)							
1	Reserved							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
	Nonvolatile cache log parameters							
4	First nonvolatile cache log parameters							
	Reserved							
	Last nonvolatile cache log parameters							
n								

Table 10 defines the parameter codes.

Table 10 — Nonvolatile Cache log parameters

Code	Description
0000h	Remaining nonvolatile time
0001h	Maximum nonvolatile time
All others	Reserved

The Remaining Nonvolatile Time parameter has the format shown in table 11.

Table 11 — Nonvolatile Cache Remaining Nonvolatile Time parameter data

Byte\Bit	7	6	5	4	3	2	1	0
0	PARAMETER LENGTH (3h)							
1	(MSB)	REMAINING NONVOLATILE TIME						
3								(LSB)

The REMAINING NONVOLATILE TIME field is defined in table 12 .

Table 12 — Nonvolatile Cache log parameters

Code	Description
000000h	Cache is volatile (either permanently or temporarily, e.g., if batteries need to be recharged).
000001h to FFFFFEh	Cache is expected to remain nonvolatile for the number of seconds indicated (e.g., battery-backed random access memory).
FFFFFFh	Cache is indefinitely nonvolatile.

The Maximum Nonvolatile Time parameter has the format shown in table 13.

Table 13 — Nonvolatile Cache Maximum Nonvolatile Time parameter data

Byte\Bit	7	6	5	4	3	2	1	0
0	PARAMETER LENGTH (3h)							
1	(MSB)	MAXIMUM NONVOLATILE TIME						
3								(LSB)

The MAXIMUM NONVOLATILE TIME field is defined in table 12 .

Table 14 — Nonvolatile Cache log parameters

Code	Description
000000h	Cache is volatile
000001h to FFFFFEh	Cache is capable of being nonvolatile for the estimated number of seconds indicated.
FFFFFFh	Cache is indefinitely nonvolatile.

6.1.3 Mode parameters [direct-access block devices]

6.1.3.1 Mode parameters overview

This subclause defines the descriptors and pages for mode parameters used with direct-access device types.

The mode parameter list, including the mode parameter header and mode block descriptor are described in SPC-3.

The MEDIUM TYPE field is contained in the mode parameter header (see SPC-3). Table 104 defines this field for direct-access block devices.

...

The DEVICE-SPECIFIC PARAMETER field (see table 15) is contained in the mode parameter header (see SPC-3).

Table 15 — Device-specific parameter

Bit	7	6	5	4	3	2	1	0
	WP	Reserved		DPOFUA	Reserved			

When used with the MODE SELECT command the write protect (WP) bit is not defined.

When used with the MODE SENSE command a WP bit of zero indicates that the medium is write enabled. A WP bit of one indicates that the medium is write protected.

When used with the MODE SELECT command, the DPOFUA bit is not used and the field is reserved.

When used with the MODE SENSE command, a DPOFUA bit of zero indicates that the device server does not support the DPO and FUA bits. When used with the MODE SENSE command, a DPOFUA bit of one indicates that the device server supports the DPO and FUA bits and may support the FUA_PHYS bit (see 4.2.1.8).

The DENSITY CODE field is contained in the mode parameter block descriptor (see SPC-3). This field is reserved for direct-access block devices.

...

6.1.3.2 Caching mode page

The Caching mode page (see table 16) defines the parameters that affect the use of the cache.

Table 16 — Caching mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	Reserved	PAGE CODE (08h)					
1	PAGE LENGTH (12h)							
2	IC	ABPF	CAP	DISC	SIZE	WCE	MF	RCD
3	DEMAND READ RETENTION PRIORITY				WRITE RETENTION PRIORITY			
4	(MSB)	DISABLE PRE-FETCH TRANSFER LENGTH						
5								(LSB)
6	(MSB)	MINIMUM PRE-FETCH						
7								(LSB)
8	(MSB)	MAXIMUM PRE-FETCH						
9								(LSB)
10	(MSB)	MAXIMUM PRE-FETCH CEILING						
11								(LSB)
12	FSW	LBCSS	DRA	VS	VS	Reserved		
13	NUMBER OF CACHE SEGMENTS							
14	(MSB)	CACHE SEGMENT SIZE						
15								(LSB)
16	Reserved						NV_DIS	NV_SUP
17	(MSB)	NON CACHE SEGMENT SIZE						
19								(LSB)

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit of one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location. If the PS is one in MODE SENSE data then the mode page shall be savable by issuing a MODE SELECT command with the SP bit of one.

The initiator control (IC) enable bit, when one, requests that the device server use the number of CACHE SEGMENTS or CACHE SEGMENT SIZE fields, dependent upon the Size bit, to control the caching algorithm rather than the device server's own adaptive algorithm.

The abort pre-fetch (ABPF) bit, when one, with the DRA bit equal to zero, requests that the device server abort the pre-fetch upon receipt of a new command. The ABPF bit of one takes precedence over the Minimum Pre-fetch bytes. When the ABPF bit is zero, with the DRA bit equal to zero, the termination of any active pre-fetch is dependent upon Caching mode page bytes 4 through 11 and is operation and/or vendor-specific.

The caching analysis permitted (CAP) bit, when one, requests that the device server perform caching analysis during subsequent operations. When zero, CAP requests that caching analysis be disabled to reduce overhead time or to prevent nonpertinent operations from impacting tuning values.

The discontinuity (DISC) bit, when one, requests that the device server continue the pre-fetch across time discontinuities, such as across cylinders (or tracks in an embedded servo device), up to the limits of the buffer, or segment, space available for the pre-fetch. When zero, the DISC requests that pre-fetches be truncated (or wrapped) at time discontinuities.

The size enable (SIZE) bit, when one, indicates that the CACHE SEGMENT SIZE is to be used to control caching segmentation. When SIZE equals zero, the application client requests that the NUMBER OF CACHE SEGMENTS is to be used to control caching segmentation. Simultaneous use of both the number of segments and the segment size is vendor-specific.

A writeback cache enable (WCE) bit of zero specifies that the device server shall return GOOD status for a WRITE command only after successfully writing all of the data to the medium. A WCE bit of one specifies that the device server may return GOOD status for a WRITE command after successfully receiving the data and prior to having successfully written it to the medium.

A multiplication factor (MF) bit of zero specifies that the device server shall interpret the MINIMUM and MAXIMUM PRE-FETCH fields in terms of the number of logical blocks for each of the respective types of pre-fetch. An MF bit of one specifies that the device server shall interpret the MINIMUM and MAXIMUM PRE-FETCH fields to be specified in terms of a scalar number that, when multiplied by the number of logical blocks to be transferred for the current command, yields the number of logical blocks for each of the respective types of pre-fetch.

A read cache disable (RCD) bit of zero specifies that the device server may return data requested by a READ command by accessing either the cache or media. A RCD bit of one specifies that the device server shall transfer all of the data requested by a READ command from the medium (i.e., data shall not be transferred from the cache).

The DEMAND READ RETENTION PRIORITY field (see table 17) advises the device server the retention priority to assign for data read into the cache that has also been transferred from the logical unit to the application client.

Table 17 — Demand read retention priority and write retention priority

Value	Description
0h	Indicates the device server should not distinguish between retaining the indicated data and data placed into the cache memory by other means (e.g., pre-fetch).
1h	Demand read retention priority; Data put into the cache via a READ command should be replaced sooner (has lower priority) than data placed into the cache by other means (e.g., pre-fetch). Write retention priority: Data put into the cache during a WRITE or WRITE AND VERIFY command should be replaced sooner (has lower priority) than data placed into the cache by other means (e.g., pre-fetch).
2h - Eh	Reserved
Fh	Demand read retention priority: Data put into the cache via a READ command should not be replaced if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and it may be replaced (i.e., it is not locked). Write retention priority: Data put into the cache during a WRITE or WRITE AND VERIFY command should not be replaced if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and it may be replaced (i.e., it is not locked).

The WRITE RETENTION PRIORITY field advises the device server the retention priority to assign for data written into the cache that has also been transferred from the cache memory to the medium.

An anticipatory pre-fetch occurs when data is placed in the cache that has not been requested. This may happen in conjunction with the reading of data that has been requested. All the following parameters give an indication to the device server how it should manage the cache based on the last READ command. An anticipatory pre-fetch may occur based on other information. All the remaining caching parameters are only recommendations to the device server and should not cause a CHECK CONDITION to occur if the device server is not able to satisfy the request.

The DISABLE PRE-FETCH TRANSFER LENGTH field specifies the selective disabling of anticipatory pre-fetch on long transfer lengths. The value in this field is compared to the number of blocks requested by the current READ command. If the number of blocks is greater than the disable pre-fetch transfer length, then an

anticipatory pre-fetch is not done for the command. Otherwise the device server should attempt an anticipatory pre-fetch. If the pre-fetch disable transfer length is zero, then all anticipatory pre-fetching is disabled for any request for data, including those for zero logical blocks.

The MINIMUM PRE-FETCH field indicates either a number of blocks or a scalar multiplier of the TRANSFER LENGTH, depending upon the setting of the MF bit. In either case, the resulting number of blocks is the number to pre-fetch regardless of the delays it might cause in executing subsequent commands.

The pre-fetching operation begins at the logical block immediately after the last logical block of the previous READ command. Pre-fetching shall always halt before exceeding the end of the media. Errors that occur during the pre-fetching operation shall not be reported to the application client unless the device server is unable to, as a result of the error, execute subsequent commands correctly. In this case the error may be reported either immediately as an error for the current READ command, or as a deferred error, at the discretion of the device server and according to the rules for reporting deferred errors.

If the pre-fetch has read more than the amount of data indicated by the MINIMUM PRE-FETCH then pre-fetching should be terminated whenever another command is ready to execute. This consideration is ignored when the MINIMUM PRE-FETCH is equal to the MAXIMUM PRE-FETCH.

The MAXIMUM PRE-FETCH field indicates either a number of blocks or a scalar multiplier of the TRANSFER LENGTH, depending upon the setting of the MF bit. In either case, the resulting number of blocks is the number to pre-fetch if the pre-fetch does not delay executing subsequent commands.

The MAXIMUM PRE-FETCH field contains the maximum amount of data to pre- fetch into the cache as a result of one READ command. It is used in conjunction with the DISABLE PRE-FETCH TRANSFER LENGTH and MAXIMUM PRE- FETCH CEILING parameters to trade off pre-fetching new data with displacing old data already stored in the cache.

The MAXIMUM PRE-FETCH CEILING field specifies an upper limit on the number of logical blocks computed as the maximum pre-fetch. If this number of blocks is greater than the MAXIMUM PRE-FETCH, then the number of logical blocks to pre-fetch shall be truncated to the value stored in the MAXIMUM PRE-FETCH CEILING field.

> NOTE 4 - If the MF bit is one the MAXIMUM PRE-FETCH CEILING field is useful in limiting the amount of data to be pre-fetched.

The force sequential write (FSW) bit when one, indicates that multiple block writes are to be transferred over the SCSI bus and written to the media in an ascending, sequential, logical block order. When the FSW bit equals zero, the device server is allowed to reorder the sequence of writing addressed logical blocks in order to achieve a faster command completion.

The logical block cache segment size (LBCSS) bit when one, indicates that the CACHE SEGMENT SIZE field units shall be interpreted as logical blocks. When the LBCSS bit equals zero the CACHE SEGMENT SIZE field units shall be interpreted as bytes. The LBCSS shall not impact the units of other fields.

The disable read-ahead (DRA) bit, when one, requests that the device server not read into the buffer any logical blocks beyond the addressed logical block(s). When the DRA bit equals zero, the device server may continue to read logical blocks into the buffer beyond the addressed logical block(s).

The vendor-specific (VS) bits may optionally be used for vendor-specific purposes.

The NUMBER OF CACHE SEGMENTS advises the device server how many segments the host requests that the cache be divided into.

The CACHE SEGMENT SIZE field indicates the requested segment size in bytes. This standard defines that the CACHE SEGMENT SIZE field is valid only when the SIZE bit is one.

An NV_DIS bit set to one specifies that the device server shall disable a non-volatile cache and indicates that a non-volatile cache is supported but disabled. An NV_DIS bit set to zero specifies that the device server may use a non-volatile cache and indicates that a non-volatile cache may be present and enabled.

An NV_SUP bit set to one indicates that the device server supports a non-volatile cache. An NV_SUP bit set to zero indicates that the device server does not support a non-volatile cache. The NV_SUP bit shall be non-changeable.

If the NON CACHE BUFFER SIZE field is greater than zero, this field advises the device server how many bytes the application client requests that the device server allocate for a buffer function when all other cache segments are occupied by data to be retained. If the number is at least one, caching functions in the other segments need not be impacted by cache misses to perform the SCSI buffer function. The impact of the NON CACHE BUFFER SIZE equals 0 or the sum of this field plus the CACHE SEGMENT SIZE greater than the buffer size is vendor-specific.

6.2.4 Mode parameters [optical memory block devices]

6.2.4.1 Mode parameters overview

This subclause defines the descriptors and pages for mode parameters used with optical memory block devices.

The mode parameter list, including the mode parameter header and mode block descriptor, are defined in SPC-3.

The MEDIUM TYPE field is contained in the mode parameter header (see SPC-3). Table 129 defines this field for optical memory block devices.

...

The DEVICE-SPECIFIC PARAMETER field is contained in the mode parameter header (see SPC-3). Table 18 defines the device-specific parameter values used for optical memory block devices.

Table 18 — Optical memory block device-specific parameter

Bit	7	6	5	4	3	2	1	0
	WP	Reserved		DPOFUA	Reserved			EBC

When used with the MODE SELECT command the WP bit is not defined. When used with the MODE SENSE command, a write protected (WP) bit of zero shall indicate that the medium is write enabled. A WP bit of one shall indicate that the medium is write protected. For read-only media the WP bit is reserved.

When used with the MODE SELECT command, the DPOFUA bit is reserved. When used with the MODE SENSE command, a DPOFUA bit of one indicates that the device server supports the DPO and FUA bits and may support the FUA_PHYS bit (see 4.2.1.8).