NVM Express™ Technical Errata

Errata ID	**012**
Revision Date	**5/15/2014**
Affected Spec Ver.	**NVM Express™ 1.0 and NVM Express 1.1a and NVMe 1.1 ECN 008**
Corrected Spec Ver.	

Errata Author(s)

Name	Company
Peter Onufryk	PMC-Sierra
Jim Hatfield	Seagate
Neal Galbo	Micron
Ronnie Huang	CNEX-Labs
Dave Landsman	SanDisk
Judy Brock	Samsung

Errata Overview

The impact of the Atomic Write Unit Power Fail and Atomic Write Unit Normal fields with respect to write errors is clarified.

The commands that are support Time Limited Error Recovery is clarified.

The PMCSR register settings are updated to reflect the PCI Power Management specification.

The Number of Namespaces in Identify is corrected to show that namespaces do not have to be allocated in order.

Revision History

Revision Date	Change Description
2/11/2014	First draft captured.
3/5/2014	Added changes for Security Receive/Send to allow for pointer to PRP List.
3/12/2014	Added Get Log Page allowance for PRP List pointer and clarified namespaces do not have to be packed sequentially.
4/30/2014	Clarified AWUN and AWUPF concepts.
5/6/2014	Included AWUN and AWUPF concepts in final draft for review.
5/8/2014	Included updates for torn writes definition and other minor editorial updates made in 5/8/2014 Technical Workgroup.
5/15/2014	Added editorial clarification on Atomic Compare & Write Unit.
6/30/2014	Ratified by NVMe Promoters.

Description of Specification Changes

Modify section 9.2 as shown below:

9.2 Media and Data Error Handling

In the event that the requested operation could not be performed to the NVM media, the particular command is completed with a media error indicating the type of failure using the appropriate status code.
< ADD LINE BREAK >
If ~~During~~ a read error occurs during the processing of ~~within~~ a command, (e.g. End-to-end Guard Check ~~failure~~ Error, Unrecovered Read Error~~, etc~~), the controller may ~~choose to~~ either stop the DMA transfer into the system memory or transfer the erroneous data to the system memory. The host shall ignore the data in the system memory locations for commands that complete with such error conditions.

If a write error occurs during the processing of a command, (e.g., an internal error, End-to-end Guard Check Error, End-to-end Application Tag Check Error), the controller may either stop or complete the DMA transfer. If the write size is less than or equal to the Atomic Write Unit Power Fail size, then subsequent reads for the associated logical blocks shall return data from the previous successful write operation. If the write size is larger than the Atomic Write Unit Power Fail size, then subsequent reads for the associated logical blocks may return data from the previous successful write operation or this failed write operation.

Based on the value of the Limited Retry bit, the controller may apply all available error recovery means to complete the command.

Modify bytes 529:526 in Figure 82 as shown below:

527:526	M	**Atomic Write Unit Normal (AWUN):** This field indicates the atomic write size for the controller during normal operation. This field is specified in logical blocks and is a 0's based value. If a write command is submitted with size less than or equal to the AWUN value, ~~of this size or less,~~ the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands ~~operations~~. If a write command is submitted with size greater than the AWUN value, ~~that is greater than this size,~~ then there is no guarantee of command atomicity. AWUN does not have any applicability to write errors caused by power failure (refer to Atomic Write Unit Power Fail). A value of FFFFh indicates all commands are atomic as this is the largest command size. It is recommended that implementations support a minimum of 128KB (appropriately scaled based on LBA size).
529:528	M	**Atomic Write Unit Power Fail (AWUPF):** This field indicates the atomic write size for the controller during a power fail or error condition. This field is specified in logical blocks and is a 0's based value. The AWUPF value shall be less than or equal to the AWUN value. < INSERT BLANK LINE > If a write command is submitted with size less than or equal to the AWUPF value, ~~of this size or less,~~ the host is guaranteed that the write is atomic to the NVM with respect to other read or write commands ~~operations~~. If a write command is submitted that is greater than this size, there is no guarantee of command atomicity. If the write size is less than or equal to the AWUPF value and the write command fails, then subsequent read commands for the associated logical blocks shall return data from the previous successful write command. If a write command is submitted with size greater than the AWUPF value, then there is no guarantee of data returned on subsequent reads of the associated logical blocks.

Modify the second paragraph of section 6.2.1 as shown below:

Note: To ensure the Compare and Write is an atomic operation in a multi-host environment, host software should ensure that the size of a Compare and Write fused operation is no larger than the ~~atomic write unit size~~ Atomic Compare & Write Unit (ACWU). Controllers may abort a Compare and Write fused operation that is larger than the ~~atomic write unit size~~ Atomic Compare & Write Unit (ACWU).

Modify section 6.3 as shown below:

6.3 Command Ordering Requirements ~~and Atomic Write Unit~~

For all NVMe commands which are not part of a fused operation (refer to section 4.7), or for which the write size is greater than AWUN, ~~Except for commands that are part of a fused operation,~~ each command is processed as an independent entity without reference to other commands submitted to the same I/O Submission Queue or to commands submitted to other I/O Submission Queues. Specifically, the controller is not responsible for checking the LBA of a Read or Write command to ensure any type of ordering between commands. For example, if a Read is submitted for LBA *x* and there is a Write also submitted for LBA *x*, there is no guarantee of the order of completion for those commands (the Read may finish first or the Write may finish first). < REMOVE BLANK LINE > If there are ordering requirements between these commands, host software or the associated application is required to enforce that ordering above the level of the controller.

The ordering requirements for fused operations are described in section 4.7.

(Note: Add new section after section 6.3 on Atomic Operations, including previous text in section 6.3 from the third paragraph on).

6.x Atomic Operations

The controller supports two values for atomic operations, Atomic Write Unit Normal (AWUN) and Atomic Write Unit Power Fail (AWUPF) that can affect command behavior and execution order based on write size.

AWUN controls the atomicity of command execution in relation to other commands. It imposes inter-command serialization of writing of blocks of data to the NVM and prevents blocks of data ending up on the NVM containing partial data from one new command and partial data from one or more other new commands.

AWUPF provides protection against torn writes. A torn write is a write operation where only some of the logical blocks that are supposed to be written contiguously are actually stored on the NVM, leaving the target logical blocks in an indeterminate state in which some logical blocks contain original data and some logical blocks contain new data from the write operation.

AWUN and AWUPF are specified in the Identify Controller data structure in Figure 82.

~~The controller supports an atomic write unit. The atomic write unit is the size of write operation guaranteed to be written atomically to the medium with respect to other read or write operations. The controller supports a value for normal operation that is potentially different than during a power fail condition, as reported in the Identify Controller data structure.~~ The host may indicate that the atomic write unit beyond a logical block size is not necessary by configuring the Write Atomicity feature, which may result in higher performance in some implementations.

6.x.1 AWUN

If enabled, AWUN specifies the execution behavior of write commands in relation to other read and write commands. If a write command is submitted with size less than or equal to the AWUN value, the host is guaranteed that the write command is atomic to the NVM with respect to other read or write commands. If a write command is submitted with size greater than the AWUN value, then there is no guarantee of command atomicity. AWUN does not have any applicability to write errors caused by power failure or other error conditions (refer to Atomic Write Unit Power Fail).

6.x.1.1 AWUN Example (Informative)

In this example, AWUN has a value of 2K (equivalent to four 512 byte logical blocks). The host issues two write commands, each with a length of 2K (i.e., four logical blocks). Command A writes LBAs 0-3 and command B writes LBAs 1-4.

Since the size of both command A and command B is less than or equal to the value of AWUN, the controller serializes these two write commands so that the resulting data in LBAs 0-4 reflects command A followed by command B, or command B followed by command A, but not an intermediate state where some of the logical blocks are written with data from command A and others are written with data from command B. Figure TBD1 shows valid results of the data in LBAs 0-4 and examples of invalid results (of which there are more possible combinations).

Figure TBD1: AWUN Example Results

	LBA 0	1	2	3	4	5	6	7
Valid Result	A	A	A	A	B			
Valid Result	A	B	B	B	B			
Invalid Result	A	A	B	B	B			
Invalid Result	A	B	A	A	B			

If the size of write commands A and B is larger than the AWUN value, then there is no guarantee of ordering. After execution of command A and command B, there may be an arbitrary mix of data from command A and command B in the LBA range specified.

6.x.2 AWUPF

AWUPF specifies the behavior of the controller if a power fail or other error condition interrupts a write operation. If a write command is submitted with size less than or equal to the AWUPF value, the controller guarantees that if the command fails due to a power failure or other error condition, then subsequent read commands for the logical blocks associated with the write command shall return one of the following:

- All old data (i.e. original data on the NVM in the LBA range addressed by the interrupted write), or
- All new data (i.e. all data to be written to the NVM by the interrupted write)

If a write command is submitted with size greater than the AWUPF value, then there is no guarantee of the data returned on subsequent reads of the associated logical blocks.

6.x.1.1 AWUPF Example (Informative)

In this example, AWUPF has a value of 1K (equivalent to two 512 byte logical blocks), AWUN has a value of 4K (equivalent to four 512 byte logical blocks). Command A writes LBAs 0-1. Figure TBD2 shows the initial state of the NVM.

Figure TBD2: AWUPF Example Initial State of NVM

	LBA 0	1	2	3	4	5	6	7
	C	B	B	B	B			

Command A begins executing but is interrupted by a power failure during the writing of the logical block at LBA 1. Figure TBD3 describes valid and invalid results.

Figure TBD3: AWUPF Example Final State of NVM

	LBA 0	1	2	3	4	5	6	7
Valid Result	A	A	B	B	B			
Valid Result	C	B	B	B	B			
Invalid Result	A	B	B	B	B			
Invalid Result	C	A	B	B	B			
Invalid Result	D	D	B	B	B			

If the size of write command A is larger than the AWUPF value, then there is no guarantee of the state of the data contained in the specified LBA range after the power fail or error condition.

After a write command has completed, reads for that location which are subsequently submitted shall return the data from that write command and not an older version of the data from ~~a~~ previous write commands with the following exception;

If all of the following conditions are met:

a) the controller supports a volatile write cache;
b) the volatile write cache is enabled;
c) the FUA bit for the write is not set;
d) no flush commands, associated with the same namespace as the write, successfully completed before shutdown; and
e) a controller shutdown occurs without completing the normal or abrupt shutdown procedure outlined in section 7.6.2

then subsequent reads for locations written to the volatile write cache that were not written to non-volatile storage may return older data.

Modify Figure 96 as shown below:

Figure 96: Error Recovery – Command Dword 11

Bit	Description
31:16	Reserved
15:00	**Time Limited Error Recovery (TLER):** Indicates a limited retry timeout value in 100 millisecond units. This applies to I/O ~~(e.g. Read, Write, etc)~~ commands that support the Limited Retry bit ~~indicate a time limit is required~~. The timeout starts when error recovery actions have started while processing the command. A value of 0h indicates that there is no timeout. Note: This mechanism is primarily intended for use by host software that may have alternate means of recovering the data.

Modify section 2.2.3 from NVMe 1.1 ECN 008 as shown below:

Bit	Type	Reset	Description
15	RWC	0	**PME Status (PMES):** Refer to the PCI SIG specifications.
14:13	~~RW~~ RO	0	**Data Scale (DSC):** Refer to the PCI SIG specifications.
12:09	~~RO~~ RO / RW	0	**Data Select (DSE):** If PME is not supported, then this field is read only '0'. Refer to the PCI SIG specifications.
08	~~RWS~~ RO / RW	0	**PME Enable (PMEE):** If PME is not supported, then this field is read only '0'. Refer to the PCI SIG specifications.
07:04	RO	0	Reserved
03	RO	1	**No Soft Reset (NSFRST):** A value of '1' indicates that the controller transitioning from D3hot to D0 because of a power state command does not perform an internal reset.
02	RO	0	Reserved
01:00	R/W	00	**Power State (PS):** This field is used both to determine the current power state of the controller and to set a new power state. The values are: 00 – D0 state 01 – D1 state 10 – D2 state 11 – $D3_{HOT}$ state When in the $D3_{HOT}$ state, the controller's configuration space is available, but the register I/O and memory spaces are not. Additionally, interrupts are blocked.

Modify the third paragraph of section 5.13 as shown below:

The Format NVM command shall fail if the controller is in an invalid security state. See the TCG SIIS reference. The Format NVM command may fail if there are outstanding ~~IO~~ I/O commands to the namespace specified to be formatted.

Modify Figure 77 as shown below:

Figure 1: Get Log Page – Reservation Notification Log

Bytes	Description
07:00	**Log Page Count:** This is a 64-bit incrementing Reservation Notification log page count, indicating a unique identifier for this notification. The count starts at 0h following a controller reset, is incremented with each unique log entry, and rolls over to zero when the maximum count is reached and a log page is created. A value of 0h indicates an empty log entry.
08	**Reservation Notification Log Page Type:** This field indicates the Reservation Notification type described by this log page. **Value** / **Definition** 0 – Empty Log Page: Get Log Page command was processed when no unread Reservation Notification log pages were available. All the fields of an empty log page shall have a value of zero. 1 – Registration Preempted 2 – Reservation Released 3 – Reservation Preempted 255:4 – Reserved
09	**Number of Available Log Pages:** This field indicates the number of additional available Reservation Notification log pages (i.e., the number of unread log pages not counting this one). If there are more than 255 additional available log pages, then a value of 255 is returned. A value of zero indicates that there are no additional available log pages.
11:10	Reserved
15:12	**Namespace ID:** This field indicates the namespace ID of the namespace associated with the Reservation Notification described by this log page.
~~63:12~~ 63:16	Reserved

Modify bytes 519:516 in Figure 82 as shown below:

519:516	M	**Number of Namespaces (NN):** This field defines the number of valid namespaces present for the controller. ~~Namespaces shall be allocated in order (starting with 1) and packed sequentially.~~