NVM Express Technical Errata

Errata ID	**021**
Change Date	**10/5/2011**
Affected Spec Ver.	**NVM Express 1.0b**
Corrected Spec Ver.	

Submission info

Name	Company	Date
Kevin Marks	Dell	9/27/2011

This erratum makes editorial changes are made to sections 6 and 7.

Description of the specification technical flaw

Modify section 6.3 as shown below:

Except for commands that are part of a fused operation, each command is processed as an independent entity without reference to other commands issued to the same I/O Submission Queue or to commands issued 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 issued for LBA *x* and there is a Write also issued 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).

If there are ordering requirements between commands, host software or the associated application is required to enforce that ordering above the level of the controller.

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 ~~level~~ is not necessary by configuring the Write Atomicity feature, which may result in higher performance in some implementations.

After a write has completed, reads for that location that subsequently complete shall return the data from that write and not an older version of the data from a previous write.

Modify the first paragraph of section 6.4 as shown below:

The commands that include data transfer ~~optionally~~ may utilize end-to-end data protection. Within these commands, the protection information action and protection information check field is specified as defined in Figure 100.

Modify the first paragraph of section 6.5 as shown below:

The Compare command reads the logical blocks specified by the command from the medium ~~LBA locations indicated from the device~~ and compares the ~~read~~ data read to a comparison data buffer transferred as part of the command. If the data read from the device and the comparison data buffer are equivalent with no miscompares, then the command completes successfully. If there is any miscompare, the command completes with error ~~failure~~. If metadata is provided, then a comparison is also performed for the metadata.

Modify Figure 101 as shown below:

Figure 101: Compare – Metadata Pointer

Bit	Description
63:00	**Metadata Pointer (MD):** This field contains the address of a contiguous physical buffer of metadata, if applicable. This field shall be Dword aligned.

Modify Figure 104 as shown below:

Figure 104: Compare – Command Dword 10 and Command Dword 11

Bit	Description
63:00	**Starting LBA (SLBA):** This field ~~indicates~~ specifies the 64-bit address of the first logical block ~~LBA~~ to compare against as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63: 32.

Modify Figure 105 as shown below:

Figure 105: Compare – Command Dword 12

Bit	Description
31	**Limited Retry (LR):** If set to '1', the controller should apply limited retry efforts. If cleared to '0', the controller should apply all available error recovery means to retrieve the data for comparison.
30	**Force Unit Access (FUA):** This field ~~indicates~~ specifies that the data read shall be read ~~returned~~ from non-volatile media. ~~There is no implied ordering with other commands.~~
29:26	**Protection Information Field (PRINFO):** Specifies the protection information action and check field, as defined in Figure 100.
25:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field ~~indicates~~ specifies the number of logical blocks to be compared. This is a 0's based value.

Modify Figure 106 as shown below:

Figure 106: Compare – Command Dword 14

Bit	Description
31:00	**Expected Initial Logical Block Reference Tag (EILBRT):** This field ~~indicates~~ specifies the Initial Logical Block Reference Tag expected value. This field is ignored ~~only used~~ if the namespace is not formatted to use end-to-end protection information. Refer to section 8.2.

Modify Figure 107 as shown below:

Figure 107: Compare – Command Dword 15

Bit	Description
31:16	**Expected Logical Block Application Tag Mask (ELBATM):** This field ~~indicates~~ specifies the Application Tag Mask expected value. This field is ignored ~~only used~~ if the namespace is formatted to use end-to-end protection information. Refer to section 8.2.
15:00	**Expected Logical Block Application Tag (ELBAT):** This field ~~indicates~~ specifies the Application Tag expected value. This field is ignored ~~only used~~ if the namespace is formatted to use end-to-end protection information. Refer to section 8.2.

Modify the first paragraph of section 6.5.1 as shown below:

~~When~~ If the command is completed, then the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command. If there are any miscompares between the data read from the NVM media and the data buffer provided, then the command fails with a status code of Compare Failure.

Modify the first paragraph of section 6.7 as shown below:

The Flush command is used by the host to indicate that any data in volatile storage should be flushed to non-volatile ~~memory~~ media.

Modify section 6.7.1 as shown below:

~~The controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status of the completed command.~~ If the command is completed, then the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Modify the first paragraph of section 6.10 as shown below:

The Write Uncorrectable command is used to mark ~~an LBA~~ a logical block as invalid. When the specified ~~LBA(s)~~ logical block(s) are read after this operation, a failure is returned with Unrecovered Read Error status. To clear the invalid ~~LBA~~ logical block status, a ~~successful~~ write operation is performed on ~~shall be issued for~~ those logical blocks.

Modify Figure 134 as shown below:

Figure 134: Write Uncorrectable – Command Dword 10 and Command Dword 11

Bit	Description
63:00	**Starting LBA (SLBA):** This field ~~indicates~~ specifies the 64-bit address of the first ~~LBA~~ logical block to be marked as invalid as part of the operation. Command Dword 10 contains bits 31:00; Command Dword 11 contains bits 63: 32.

Modify Figure 135 as shown below:

Figure 135: Write Uncorrectable – Command Dword 12

Bit	Description
31:16	Reserved
15:00	**Number of Logical Blocks (NLB):** This field ~~indicates~~ specifies the number of logical blocks to be marked as invalid. This is a 0's based value.

Modify section 6.10.1 as shown below:

~~When~~ If the command is completed ~~with success or failure~~, then the controller shall post a completion queue entry to the associated I/O Completion Queue indicating the status for the command.

Modify section 7.1 as shown below:

Host software presents commands to the controller through pre-allocated Submission Queues. The controller is alerted to new commands to execute through SQ Tail Doorbell register writes. The difference between the previous doorbell register value and the current register write indicates the number of commands that are ready for processing by the controller.

The controller fetches the commands from the Submission Queue(s) and transmits them to the NVM subsystem for processing. Except for fused operations, there are no ordering restrictions for processing of the commands within or across Submission Queues. Host software ~~shall~~ should not place commands in the list that may not be re-ordered arbitrarily. Data may or may not be committed to the NVM media in the order that commands are received.

Host software issues commands of higher priorities to the appropriate Submission Queues. Priority is associated with the Submission Queue itself, thus the priority of the command is based on the Submission Queue it is issued through. The controller arbitrates across the Submission Queues based on fairness and priority according to the arbitration scheme specified in section 4.7.

Upon completion of the commands by the NVM subsystem, the controller presents completion queue entries to the host through the appropriate Completion Queues. If MSI-X or multiple message MSI is in use, then the interrupt vector indicates the Completion Queue(s) with possible new command completions for the host to process. If pin-based interrupts or single message MSI interrupts are used, host software interrogates the Completion Queue(s) for new completion queue entries. The host updates the CQ Head doorbell register to release Completion Queue entries to the controller and clear the associated interrupt.

There are no ordering restrictions for completions to the host. Each completion queue entry identifies the Submission Queue ~~ID~~ Identifier and Command ~~ID~~ Identifier of the associated command. Host software uses this information to correlate the completions with the commands issued to the Submission Queue(s).

Host software is responsible for creating all required Submission and Completion Queues prior to issuing commands to the controller. I/O Submission and Completion Queues are created using Admin commands defined in section 5.

Disposition log

9/27/2011	Erratum partially captured.
10/5/2011	Compare command language updated.
11/14/2011	Erratum ratified.