NVM Express Technical Errata

Errata ID	**013**
Change Date	**6/2/2011**
Affected Spec Ver.	**NVM Express 1.0a**
Corrected Spec Ver.	

Submission info

Name	Company	Date
Judy Brock	Samsung	5/24/2011
Kevin Marks	Dell	5/24/2011

This erratum makes editorial changes and clarifications throughout the document.

Insertion and removal of entries to/from queues is clarified to avoid issues with wrap-around conditions.

Description of the specification technical flaw

Modify section 1.1 as shown below:

NVM Express (NVMe) is a register level interface that allows host software to communicate with a non-volatile memory subsystem. This interface is optimized for Enterprise and Client solid state drives, typically attached to the PCI Express interface.

Note: During development, this specification was referred to as Enterprise NVMHCI. However, the name was modified to NVM Express prior to specification completion ~~since this interface may also be used in Client systems~~. This interface is targeted for use in both Client and Enterprise systems.

Modify the first paragraph of section 1.4 as shown below:

NVM Express is a scalable host controller interface designed to address the needs of Enterprise and Client systems that utilize PCI Express based solid state drives. The interface provides an optimized command issue and completion path ~~beyond~~. It includes support for parallel operation by supporting up to 64K I/O Queues with up to 64K commands per I/O Queue. Additionally, support has been added for many Enterprise capabilities like end-to-end data protection (compatible with T10 DIF and SNIA DIX standards), enhanced error reporting, and virtualization.

Modify the seventh paragraph of section 1.4 as shown below:

An I/O Command Set is used with an I/O queue pair. This specification defines one I/O Command Set, named the NVM Command Set. The host selects one I/O Command Set that is used for all I/O queue pairs.

Modify Figure 12 as shown below:

Figure 12: Completion Queue Entry: DW 2

Bit	Description
31:16	**SQ Identifier (SQID):** Indicates the Submission Queue that the associated command was issued to. This field is used by software when more than one Submission Queue shares a single Completion Queue to uniquely determine the command completed in combination with the Command Identifier (CID).
15:00	**SQ Head Pointer (SQHD):** Indicates the current Submission Queue Head pointer for the Submission Queue indicated in the SQ Identifier field. This is used to indicate to the host the Submission Queue entries that have been consumed and may be re-used for new entries. Note: The value returned is the value of the SQ Head pointer when the completion entry was created. By the time software consumes the completion entry, the controller may have an SQ Head pointer that has advanced beyond the value indicated.

Modify section 4.1 as shown below:

The Head and Tail entry pointers correspond to the Completion Queue Head Doorbells and the Submission Queue Tail Doorbells defined in section 3.1.10 and 3.1.11. The doorbell registers are updated by host software.

The submitter of entries to a queue uses the current Tail entry pointer to identify the next open queue entry space. The submitter increments the Tail entry pointer after submitting the new entry to the open queue entry space. If the Tail entry pointer increment exceeds the queue size, the Tail entry shall roll to zero. The submitter may ~~can~~ continue to submit entries to the queue as long as the Full queue condition is not met (refer to section 4.1.2) ~~the Tail entry is less than the Current Head entry pointer~~. Note: The submitter shall take queue wrap conditions into account.

The consumer of entries on a queue uses the current Head entry pointer to identify the next entry to be pulled off the queue. The consumer increments the Head entry pointer after retrieving the next entry from the queue. If the Head entry pointer increment exceeds the queue size, the Head entry pointer shall roll to zero. The consumer may continue to remove entries from the queue as long as the Empty queue condition is not met (refer to section 4.1.1) ~~the Head entry pointer is greater than the Tail entry pointer~~. Note: The consumer shall take queue wrap conditions into account.

Modify the first paragraph of section 4.3 as shown below:

A physical region page (PRP) entry is a pointer to a physical memory page. PRPs are used as a scatter/gather mechanism for data transfers between the controller and system memory. To enable efficient out of order data transfers between the device and the host, PRP entries are a fixed size. (ADD RETURN)

The size of the physical memory page is configured by software in CC.MPS. **Error! Reference source not found.** shows the layout of a PRP entry that consists of a Page Base Address and an Offset. The size of the Offset field is determined by the physical memory page size configured in CC.MPS.

Modify the second paragraph of section 5.12.1.7 and Figure 82 as shown below:

The number of queues allocated is returned in Dword 0 of the command completion entry. The definition is shown in Figure 82.

Note: The value allocated may be smaller or larger than the number of queues requested, often in virtualized implementations. The controller may not have as many queues to allocate as are requested. Alternatively, the controller may have an allocation unit of queues (e.g. power of two) and may supply more queues to host software to satisfy its allocation unit.

Figure 82: Number of Queues – Dword 0 of command completion entry

Bit	Description
31:16	**Number of I/O Completion Queues Allocated (NCQA):** Indicates the number of I/O Completion Queues allocated by the controller. A minimum of one shall be allocated, reflecting that the minimum support is for one I/O Completion Queue. The value may not match ~~be larger than~~ the number requested by host software. This is a 0's based value.
15:00	**Number of I/O Submission Queues Allocated (NSQA):** Indicates the number of I/O Submission Queues allocated by the controller. A minimum of one shall be allocated, reflecting that the minimum support is for one I/O Submission Queue. The value may not match ~~be larger than~~ the number requested by host software. This is a 0's based value.

Modify section 4.1 as shown below:

Figure 30: Asynchronous Event Information – Error Status

Value	Description
0h	**Invalid Submission Queue:** Software wrote the doorbell of a queue that was not created.
1h	**Invalid Doorbell Write Value:** Software attempted to write an invalid doorbell value. Some possible causes of this error are: • the value written was out of range of the corresponding queue's base address and size, • the value written is the same as the previously written doorbell value, • software attempts to add a command to a full Submission Queue, and • software attempts to remove a completion entry from an empty Completion Queue.
2h	**Diagnostic Failure:** A diagnostic failure was detected. This may include a self test operation.
3h	**Persistent Internal Device Error:** A failure occurred within the device that is persistent or the device is unable to isolate to a specific set of commands. If this error is indicated, then the CSTS.CFS bit may be set to '1' and the host should perform a reset as described in section 7.3.
4h	**Transient Internal Device Error:** A transient error occurred within the device that is specific to a particular set of commands and operation may continue.
5h	**Firmware Image Load Error:** The firmware image could not be loaded. The controller reverted to the previously active firmware image or a baseline read-only firmware image.
6h - FFh	Reserved

Modify steps 2-4 of section 8.1 as shown below:

2. After the firmware is downloaded to the controller, the next step is for the host to issue a Firmware Activate command. The Firmware Activate command verifies that the last firmware image downloaded is valid and commits that image to the firmware slot indicated for future use. A firmware image that does not start at offset zero, contains gaps, or contains overlapping regions is considered invalid. A controller may employ additional vendor specific means (e.g., checksum, CRC, cryptographic hash or a digital signature) to determine the validity of a firmware image.
    a. The Firmware Activate command may also be used to activate a firmware image associated with a previously activated firmware slot. ~~Using the Firmware Activate command to activate a firmware slot that was not previously successfully activated after the Firmware Image Download process concluded results in an Invalid Firmware Slot command completion error.~~
3. The last step is to perform a reset that then causes the firmware image ~~indicated~~ specified in the Firmware Activate command to be applied. The reset may be a Conventional Reset, Function Level Reset, Controller Reset (CC.EN transitions from '1' to '0'), or a power cycle of the controller.
4. After the reset has completed, host software ~~shall~~ re-initializes the controller. This includes re-allocating I/O Submission and Completion Queues. Refer to section 7.6.1.

Modify the third paragraph of section 8.1. as shown below:

If the firmware is not able to be successfully loaded following a reset, then the controller shall revert to the previously active firmware image or the baseline read-only firmware image, if available, and indicate the failure as an asynchronous event with a Firmware Image Load Error.

Modify section 8.2 as shown below:

The controller may support metadata per logical block. Metadata is additional data allocated on a per ~~LBA~~ logical block basis. There is no requirement for how the host makes use of the metadata area. One of the most common usages for metadata is to convey end-to-end protection information.

The metadata may be transferred by the controller to or from the host in one of two ways. The mechanism used is selected as part of the Format NVM command.

The first mechanism for transferring the metadata is as a contiguous part of the ~~LBA~~ logical block that it is associated with. The metadata is transferred at the end of the associated ~~LBA~~ logical block, forming an extended ~~LBA~~ logical block. This mechanism is illustrated in Figure 139. In this case, both the ~~LBA~~ logical block data and ~~LBA~~ logical block metadata are pointed to by the PRP1 and PRP2 ~~PRP List~~ pointers.

The second mechanism for transferring the metadata is as a separate contiguous buffer of data. This mechanism is illustrated in Figure 140. In this case, the metadata is pointed to with the Metadata Pointer, while the ~~LBA~~ logical block data is pointed to by the PRP1 and PRP2 ~~PRP List~~ pointers. The metadata is required to be physically contiguous in this case since there is only one Metadata Pointer.

One of the transfer mechanisms shall be selected for each namespace when it is formatted; transferring a portion of metadata with one mechanism and a portion with the other mechanism is not supported.

If ~~When~~ end-to-end data protection is used, then ~~the~~ eight bytes of metadata for each ~~LBA~~ logical block is used for the Protection Information field.

Modify the second paragraph of section 8.4 as shown below:

The number of power states implemented by a controller is returned in the Number of Power States Supported (NPSS) field in the ~~Indentify~~ Identify Controller data structure. A controller shall support at least one power state and may optionally support up to a total of 32 power states. Power states are contiguously numbered starting with zero such that each subsequent power state consumes less or equal power than the previous state. Thus, power state zero indicates the maximum power that the NVM subsystem is capable of consuming.

Modify the first paragraph of section 8.5 as shown below:

The PCI-SIG Single Root I/O Virtualization and Sharing Specification (SR-IOV) defines extensions to PCI Express that allow multiple System Images (SI), such as virtual machines running on a hypervisor, to share PCI hardware resources. The primary benefit of SR-IOV is that it eliminates the hypervisor from participating in I/O operations which may ~~can~~ be a significant factor limiting storage performance in some virtualized environments and allows direct SI access to PCI hardware resources.

Modify section 9.1 as shown below:

In the case of serious error conditions, like Completion Queue Invalid, the operation of the associated Submission Queue or Completion Queue may be compromised. In this case, host software should delete the associated Completion Queue and/or Submission Queue. The delete of a Submission Queue aborts all outstanding commands, and deletion of either queue type releases resources associated with that queue. Host software ~~Software~~ should recreate the Completion Queue and/or Submission Queue to then continue with operation.

In the case of serious error conditions for Admin commands, the entire controller should be reset using a Controller Reset. The entire controller should also be reset if a completion is not received for the deletion of a Submission Queue or Completion Queue.

For most command errors, there is not an issue with the Submission Queue and/or Completion Queue itself. Thus, host software and the controller should continue to process commands. It is at the discretion of host software whether to retry the failed command; the Retry bit in the Completion Queue Entry indicates whether a retry of the failed command may succeed.

Modify the second paragraph of section 9.2 as shown below:

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

Modify the first paragraph of section 9.3 as shown below:

System memory errors such as target abort, master abort, and parity may cause the controller to stop processing the currently executing command. These are serious errors that cannot be recovered from without host software intervention.

Modify section 9.4 as shown below:

Device specific errors such as a DRAM failure or power loss notification errors indicate that a controller level failure has occurred during the processing of a command. The status code of the completion queue entry ~~shall~~ should indicate an Internal Device Error status code (if multiple error conditions exist, the lowest numerical value is returned). Host software shall ignore any data transfer associated with the command. The host may choose to re-issue the command or indicate an error to the higher level software.

Modify the second paragraph of section 10.1.3 as shown below:

Host software ~~Software~~ should search the EFI Configuration Table for the NVM Express GUID. If a match is found then an NVM Express EFI module was or is currently loaded. The associated data structure may be read to obtain the version information.

Disposition log

5/24/2011	Erratum captured.
5/27/2011	Updates for number of queues returned and firmware image download.
6/2/2011	Removed confusing new sentence on firmware update process.
7/8/2011	Erratum ratified.