LEGAL NOTICE:

LEGAL DISCLAIMER:

THIS DOCUMENT AND THE INFORMATION CONTAINED HEREIN IS PROVIDED ON AN "**AS IS**" BASIS. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, NVM EXPRESS, INC. (ALONG WITH THE CONTRIBUTORS TO THIS DOCUMENT) HEREBY DISCLAIM ALL REPRESENTATIONS, WARRANTIES AND/OR COVENANTS, EITHER EXPRESS OR IMPLIED, STATUTORY OR AT COMMON LAW, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, VALIDITY, AND/OR NONINFRINGEMENT.

All product names, trademarks, registered trademarks, and/or servicemarks may be claimed as the property of their respective owners.

NVM Express Workgroup
c/o Virtual, Inc.
401 Edgewater Place, Suite 600
Wakefield, MA 01880
info@nvmexpress.org

NVM Express Technical Proposal for New Feature

Technical Proposal ID	**4000a – Persistent Memory Region (PMR)**
Change Date	**11/7/2017**
Builds on Specification	**NVM Express 1.3**

Technical Proposal Author(s)

Name	Company
Stephen Bates	Microsemi Corporation
Peter Onufryk	Microsemi Corporation
Christoph Hellwig	Western Digital Corporation

This feature defines a Persistent Memory Region (PMR), which is a PCIe memory region similar to a Controller Memory Buffer, but whose contents persist across power cycles and resets. This new feature is optional.

The 4000a version includes errata. Specifically:

- PMRSTS modified to show Health Status is a 3-bit field.

Revision History

Revision Date	Change Description
2/28/2017	Initial version
3/8/2017	• Updated Health Status (HSTS) field in the Persistent Memory Region Status (PMRSTS) register. Added not persistent status. Added more details about the error status. Changed encoding. • Added text that describes TLP data poisoning and provided a reference to the PCI Express specification for more information. • Added Section 4.7 Controller Memory buffer and made necessary minor changes. • Minor edits to the wording of text.
3/15/2017	• Added WD (Christoph) as a sponsor. • Removed PMRFLUSH register and incorporated this functionality into PMRSTS register. • Fixed Ready (RDY) field. Changed field name from Not Ready to Ready

	and corrected polarity inconsistency. • Incorporated feedback from Michael Allison. • Clarified what ready and not ready means. Ready is EN and RDY are both equal to one. If either is zero, then the PMR is not ready. • Added text that clarifies that PMRSTS.HSTS polling may be used to ensure that prior reads have completed without error. • Added PMR behavior to Global Data Erased bit. • Added text and error code that prohibits a sanitize operation while the Persistent Memory Region is enabled.
3/21/2017	• Fixed PMRCAP.BIR so that it is three bits. • Added text that clarifies that the contents of a PMR persist across PMR disables. • Clarified PMRCTL.EN behavior when PMRCTL.EN and PMRSTS.RDY are not equal. • Clarified that PMRSTS.ERR behavior reflects previous write status when the PMR is ready and operating normally. • Updated SSAT (global data erased) field with text from NVMe 1.3 release candidate 1. • Changed Sanitize Prohibited While the Persistent Memory Region is Enabled code to TBD since code needs to be assigned using new resource assignment process. • Added text to PMRSTS.HSTS field that once set to a non-zero value, the field is cleared by disabling and re-enabling the PMR. • Added “shalls” to section 8.15.1. • Added text that clarifies how PMRSTS reads may be used to determine that reads may have returned in valid data. • Added text that PMRSTS.ERR is always zero if flushing using the PMRSTS register is not supported. • Fixed Persistent spelling typo. • Added text to PMRSTS.HSTS to clarity that a PMR has special behavior during a sanitize operation.
3/22/2017	• Removed 0h from being a valid value for PMRCAP.BIR.
04/05/2017	• Changed base address of PMR from D000h to E000h. • Added back the WDS, RDS, LISTS, CQS, and SQS bits from the CMB to the PMR. These bits were there originally, but were removed. Having only the WDS and RDS bits has caused confusion. • Moved PMR Capability (PMRCAP) bit to existing CAP register. • Shifted bits around in PMRCAP to compress used bits. • Removed Not Persistent health status due to feedback that this was adding complexity and not generally useful. • Added PMR has become unreliable indication to the SMART / Health Information Log. • Defined behavior of a not-ready PMR in the way required for Sanitize. Removed sanitize specific text. • Normal PMR operation following an enable/ready error shall resume following a PCI Function reset.
04/06/2017	• Removed LISTS, CQS and CQS bit from the PMRCAP register. Putting SGLs, PRPs, CQs, or SQs in the PMR is not supported by the TP. • Added restore error to the Health Status field in the PMRSTS register. The purpose of this field is to report that a controller has failed to read the non-volatile data for a PMR, but can operate normally going forward. • Added read-only transition to critical warnings.
6/28/2017	• Major rewrite after phase 2 to 3 vote.
7/8/2017	• Added back the footnote that was in earlier drafts that explained that what makes a write persistent is implementation specific. • Replaced PMRFLUSHI field with PMRWBM. Updated write barrier text. • Made minor edits and fixes.
7/18/2017	• Made write flush associated with a PMR read optional.

	• Made the value returned by a PMR read when the PMR is not enabled undefined. Added text that this undefined value shall not expose user data using a sanitize operation.
7/27/2017	• Assigned value to Sanitize Prohibited While Persistent Memory Region is Enabled status code in Figure 33. • Clarified that the undefined value returned by a PMR read following a sanitize operation is such that recovery of any previous user data from any cache or the non-volatile media is not possible. • Fixed typo in PMRCAP bits
8/2/2017	• Corrected PMRCAP field bit numbering error • Typo corrections and minor wording changes to improve readability
8/3/2017	• Typo corrections and minor wording changes to improve readability
11/7/2017	• Modified PMRSTS to show Health Status is 3 bit field.
1/24/2018	• Minor errata (4000a) ratified in 1/8/2018 Board meeting.

Description of Specification Changes

Modify a portion of Section 3.1 (Register Definition) as shown below:

Start	End	Symbol	Description
00h	07h	CAP	Controller Capabilities
08h	0Bh	VS	Version
0Ch	0Fh	INTMS	Interrupt Mask Set
10h	13h	INTMC	Interrupt Mask Clear
14h	17h	CC	Controller Configuration
18h	1Bh	Reserved	Reserved
1Ch	1Fh	CSTS	Controller Status
20h	23h	NSSR	NVM Subsystem Reset (Optional)
24h	27h	AQA	Admin Queue Attributes
28h	2Fh	ASQ	Admin Submission Queue Base Address
30h	37h	ACQ	Admin Completion Queue Base Address
38h	3Bh	CMBLOC	Controller Memory Buffer Location (Optional)
3Ch	3Fh	CMBSZ	Controller Memory Buffer Size (Optional)
40h	43h	BPINFO	Boot Partition Information (Optional)
44h	47h	BPRSEL	Boot Partition Read Select (Optional)
48h	4Fh	BPMBL	Boot Partition Memory Buffer Location (Optional)
50h	~~EFF~~DFFh	Reserved	Reserved
E00h	E03h	PMRCAP	Persistent Memory Capabilities (Optional)
E04h	E07h	PMRCTL	Persistent Memory Region Control (Optional)
E08h	E0Bh	PMRSTS	Persistent Memory Region Status (Optional)
E0Ch	EFFh	Reserved	Reserved
F00h	FFFh	Reserved	Command Set Specific
1000h	1003h	SQ0TDBL	Submission Queue 0 Tail Doorbell (Admin)
1000h + (1 * (4 << CAP.DSTRD))	1003h + (1 * (4 << CAP.DSTRD))	CQ0HDBL	Completion Queue 0 Head Doorbell (Admin)
1000h + (2 * (4 << CAP.DSTRD))	1003h + (2 * (4 << CAP.DSTRD))	SQ1TDBL	Submission Queue 1 Tail Doorbell
1000h + (3 * (4 << CAP.DSTRD))	1003h + (3 * (4 << CAP.DSTRD))	CQ1HDBL	Completion Queue 1 Head Doorbell
1000h + (4 * (4 << CAP.DSTRD))	1003h + (4 * (4 << CAP.DSTRD))	SQ2TDBL	Submission Queue 2 Tail Doorbell
1000h + (5 * (4 << CAP.DSTRD))	1003h + (5 * (4 << CAP.DSTRD))	CQ2HDBL	Completion Queue 2 Head Doorbell
…	…	…	…
1000h+ (2y * (4 << CAP.DSTRD))	1003h + (2y * (4 << CAP.DSTRD))	SQyTDBL	Submission Queue y Tail Doorbell
1000h + ((2y + 1) * (4 << CAP.DSTRD))	1003h + ((2y + 1) * (4 << CAP.DSTRD))	CQyHDBL	Completion Queue y Head Doorbell
			Vendor Specific (Optional)

Modify a portion of Section 3.1.1 (Register Definition) as shown below:

3.1.1 Offset 00h: CAP – Controller Capabilities

This register indicates basic capabilities of the controller to host software.

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>63:56~~67~~</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>56</td><td>RO</td><td></td><td>Persistent Memory Region Supported (PMRS): This field indicates whether the Persistent Memory Region is supported. This field is set to ‘1’ if the Persistent Memory Region is supported. This field is cleared to ‘0’ if the Persistent Memory Region is not supported.</td></tr>
<tr><td>55:52</td><td>RO</td><td>Impl Spec</td><td>Memory Page Size Maximum (MPSMAX): This field indicates the maximum host memory page size that the controller supports. The maximum memory page size is (2 ^ (12 + MPSMAX)). The host shall not configure a memory page size in CC.MPS that is larger than this value.</td></tr>
<tr><td>51:48</td><td>RO</td><td>Impl Spec</td><td>Memory Page Size Minimum (MPSMIN): This field indicates the minimum host memory page size that the controller supports. The minimum memory page size is (2 ^ (12 + MPSMIN)). The host shall not configure a memory page size in CC.MPS that is smaller than this value.</td></tr>
<tr><td>47:46</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>45</td><td>RO</td><td>Impl Spec</td><td>Boot Partition Support (BPS): This field indicates whether the controller supports Boot Partitions. If this field is set to '1‘, the controller supports Boot Partitions. If this field is cleared to ‘0‘, the controller does not support Boot Partitions. Refer to section 8.13.</td></tr>
<tr><td>44:37</td><td>RO</td><td>Impl Spec</td><td>Command Sets Supported (CSS): This field indicates the I/O Command Set(s) that the controller supports. A minimum of one command set shall be supported. The field is bit significant. If a bit is set to ‘1’, then the corresponding I/O Command Set is supported. If a bit is cleared to ‘0’, then the corresponding I/O Command Set is not supported.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>37</td><td>NVM command set</td></tr>
<tr><td>38</td><td>Reserved</td></tr>
<tr><td>39</td><td>Reserved</td></tr>
<tr><td>40</td><td>Reserved</td></tr>
<tr><td>41</td><td>Reserved</td></tr>
<tr><td>42</td><td>Reserved</td></tr>
<tr><td>43</td><td>Reserved</td></tr>
<tr><td>44</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>36</td><td>RO</td><td>Impl Spec</td><td>NVM Subsystem Reset Supported (NSSRS): This field indicates whether the controller supports the NVM Subsystem Reset feature defined in section 7.3.1. This field is set to '1' if the controller supports the NVM Subsystem Reset feature. This field is cleared to ‘0' if the controller does not support the NVM Subsystem Reset feature.</td></tr>
<tr><td>35:32</td><td>RO</td><td>Impl Spec</td><td>Doorbell Stride (DSTRD): Each Submission Queue and Completion Queue Doorbell register is 32-bits in size. This register indicates the stride between doorbell registers. The stride is specified as (2 ^ (2 + DSTRD)) in bytes. A value of 0h indicates a stride of 4 bytes, where the doorbell registers are packed without reserved space between each register. Refer to section 8.6.</td></tr>
<tr><td>31:24</td><td>RO</td><td>Impl Spec</td><td>Timeout (TO): This is the worst case time that host software shall wait for CSTS.RDY to transition from:
a) ‘0’ to ‘1’ after CC.EN transitions from ‘0’ to ‘1’; or
b) ‘1’ to ‘0’ after CC.EN transitions from ‘1’ to ‘0’.

This worst case time may be experienced after events such as an abrupt shutdown or activation of a new firmware image; typical times are expected to be much shorter. This field is in 500 millisecond units.</td></tr>
<tr><td>23:19</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
</table>

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>18:17</td><td>RO</td><td>Impl Spec</td><td>Arbitration Mechanism Supported (AMS): This field is bit significant and indicates the optional arbitration mechanisms supported by the controller. If a bit is set to ‘1’, then the corresponding arbitration mechanism is supported by the controller. Refer to section 4.11 for arbitration details.

<table><tr><th>Bit</th><th>Definition</th></tr><tr><td>17</td><td>Weighted Round Robin with Urgent Priority Class</td></tr><tr><td>18</td><td>Vendor Specific</td></tr></table>
The round robin arbitration mechanism is not listed since all controllers shall support this arbitration mechanism.</td></tr>
<tr><td>16</td><td>RO</td><td>Impl Spec</td><td>Contiguous Queues Required (CQR): This field is set to ‘1’ if the controller requires that I/O Submission Queues and I/O Completion Queues are required to be physically contiguous. This field is cleared to ‘0’ if the controller supports I/O Submission Queues and I/O Completion Queues that are not physically contiguous. If this field is set to ‘1’, then the Physically Contiguous bit (CDW11.PC) in the Create I/O Submission Queue and Create I/O Completion Queue commands shall be set to ‘1’.</td></tr>
<tr><td>15:00</td><td>RO</td><td>Impl Spec</td><td>Maximum Queue Entries Supported (MQES): This field indicates the maximum individual queue size that the controller supports. For NVMe over PCIe implementations, this value applies to the I/O Submission Queues and I/O Completion Queues that the host creates. For NVMe over Fabrics implementations, this value applies to only the I/O Submission Queues that the host creates. This is a 0’s based value. The minimum value is 1h, indicating two entries.</td></tr>
</table>

Modify Section 3.1.12 as shown below

3.1.12 Offset 3Ch: CMBSZ – Controller Memory Buffer Size

This optional register defines the size of the Controller Memory Buffer (refer to section 4.7). If the controller does not support the Controller Memory Buffer feature then this register shall be cleared to 0h.

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:12</td><td>RO</td><td>Impl Spec</td><td>Size (SZ): Indicates the size of the Controller Memory Buffer available for use by the host. The size is in multiples of the Size Unit. If the Offset + Size exceeds the length of the indicated BAR, the size available to the host is limited by the length of the BAR.</td></tr>
<tr><td>11:8</td><td>RO</td><td>Impl Spec</td><td>Size Units (SZU): Indicates the granularity of the Size field.
<table>
<tr><th>Value</th><th>Granularity</th></tr>
<tr><td>0h</td><td>4 KB</td></tr>
<tr><td>1h</td><td>64 KB</td></tr>
<tr><td>2h</td><td>1 MB</td></tr>
<tr><td>3h</td><td>16 MB</td></tr>
<tr><td>4h</td><td>256 MB</td></tr>
<tr><td>5h</td><td>4 GB</td></tr>
<tr><td>6h</td><td>64 GB</td></tr>
<tr><td>7h – Fh</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>7:5</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>4</td><td>RO</td><td>Impl Spec</td><td>Write Data Support (WDS): If this bit is set to ‘1’, then the controller supports data and metadata in the Controller Memory Buffer for commands that transfer data from the host to the controller (e.g., Write). If this bit is cleared to ‘0’, then ~~all~~ data and metadata for commands that transfer data from the host to the controller shall not be transferred from ~~host memory~~the Controller Memory Buffer.</td></tr>
<tr><td>3</td><td>RO</td><td>Impl Spec</td><td>Read Data Support (RDS): If this bit is set to ‘1’, then the controller supports data and metadata in the Controller Memory Buffer for commands that transfer data from the controller to the host (e.g., Read). If this bit is cleared to ‘0’, then ~~all~~ data and metadata for commands that transfer data from the controller to the host shall not be transferred ~~from host memory~~ to the Controller Memory Buffer.</td></tr>
<tr><td>2</td><td>RO</td><td>Impl Spec</td><td>PRP SGL List Support (LISTS): If this bit is set to ‘1’, then the controller supports PRP Lists in the Controller Memory Buffer. If this bit is set to ‘1’ and SGLs are supported by the controller, then the controller supports Scatter Gather Lists in the Controller Memory Buffer. If this bit is set to ‘1’, then the Submission Queue Support bit shall be set to ‘1’. If this bit is cleared to ‘0’, then ~~all~~ PRP Lists and SGLs shall not be placed in ~~host memory~~the Controller Memory Buffer.</td></tr>
<tr><td>1</td><td>RO</td><td>Impl Spec</td><td>Completion Queue Support (CQS): If this bit is set to ‘1’, then the controller supports Admin and I/O Completion Queues in the Controller Memory Buffer. If this bit is cleared to ‘0’, then ~~all~~ Completion Queues shall not be placed in ~~host memory~~the Controller Memory Buffer.</td></tr>
<tr><td>0</td><td>RO</td><td>Impl Spec</td><td>Submission Queue Support (SQS): If this bit is set to ‘1’, then the controller supports Admin and I/O Submission Queues in the Controller Memory Buffer. If this bit is cleared to ‘0’, then ~~all~~ Submission Queues shall not be placed in ~~host memory~~the Controller Memory Buffer.</td></tr>
</table>

Insert the following sections between Sections 3.1.15 and 3.1.16 as shown below:

3.1.16 E00h: PMRCAP – Persistent Memory Region Capabilities

This register indicates capabilities of the Persistent Memory Region. If the controller does not support the Persistent Memory Region feature, then this register shall be cleared to 0h.

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:24</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>23:16</td><td>RO</td><td>Impl Spec</td><td>Persistent Memory Region Timeout (PMRTO): This field contains the minimum amount of time that host software should wait for the Persistent Memory Region to become ready or not ready after PMRCTL.EN is modified. The time in this field is expressed in Persistent Memory Region time units (refer to PMRCAP.PMRTU).</td></tr>
<tr><td>15:14</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>13:10</td><td>RO</td><td>Impl Spec</td><td>Persistent Memory Region Write Barrier Mechanisms (PMRWBM): This field lists mechanisms that may be used to ensure that previous writes to the Persistent Memory Region have completed and are persistent when the Persistent Memory Region is ready and operating normally. A bit in this field is set to ‘1’ if the corresponding mechanism to ensure persistence is supported. A bit in this field is cleared to ‘0’ if the corresponding mechanism to ensure persistence is not supported.

At least one bit in this field shall be set to ‘1’.
<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>0</td><td>The completion of a memory read from any Persistent Memory Region address ensures that all prior writes to the Persistent Memory Region have completed and are persistent.</td></tr>
<tr><td>1</td><td>The completion of a read to the PMRSTS register shall ensure that all prior writes to the Persistent Memory Region have completed and are persistent.</td></tr>
<tr><td>3:2</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>9:8</td><td>RO</td><td>Impl Spec</td><td>Persistent Memory Region Time Units (PMRTU): Indicates Persistent Memory Region time units
<table>
<tr><th>Value</th><th>Persistent Memory Region Time Units</th></tr>
<tr><td>00b</td><td>500 milliseconds</td></tr>
<tr><td>01b</td><td>minutes</td></tr>
<tr><td>01b – 11b</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>7:5</td><td>RO</td><td>Impl Spec</td><td>Base Indicator Register (BIR): This field indicates the Base Address Register (BAR) that specifies the address and size of the Persistent Memory Region. Values 2h, 3h, 4h, and 5h are valid.</td></tr>
<tr><td>4</td><td>RO</td><td>Impl Spec</td><td>Write Data Support (WDS): If this bit is set to ‘1’, then the controller supports data and metadata in the Persistent Memory Region for commands that transfer data from the host to the controller (e.g., Write). If this bit is cleared to ‘0’, then data and metadata for commands that transfer data from the host to the controller shall not be transferred to the Persistent Memory Region.</td></tr>
<tr><td>3</td><td>RO</td><td>Impl Spec</td><td>Read Data Support (RDS): If this bit is set to ‘1’, then the controller supports data and metadata in the Persistent Memory Region for commands that transfer data from the controller to the host (e.g., Read). If this bit is cleared to ‘0’, then all data and metadata for commands that transfer data from the controller to the host shall not be transferred from the Persistent Memory Region.</td></tr>
<tr><td>2:0</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
</table>

3.1.17 Offset E04h: PMRCTL – Persistent Memory Region Control

This optional register controls the operation of the Persistent Memory Region. If the controller does not support the Persistent Memory Region feature, then this register shall be cleared to 0h.

Bit	Type	Reset	Description
31:1	RO	0h	Reserved
0	RW	0	**Enable (EN):** When set to '1', then the Persistent Memory Region is ready to process PCI Express memory read and write requests once PMRSTS.NRDY is cleared to '0'. When cleared to '0', then the Persistent Memory Region is disabled and PMRSTS.NRDY shall be set to '1' once the Persistent Memory Region is ready to be re-enabled.

3.1.18 Offset E08h: PMRSTS – Persistent Memory Region Status

This optional register provides the status of the Persistent Memory Region. If the controller does not support the Persistent Memory Region feature, then this register shall be cleared to 0h.

<table>
<tr><th>Bit</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:12</td><td>RO</td><td>0h</td><td>Reserved</td></tr>
<tr><td>11:9</td><td>RO</td><td>0h</td><td>Health Status (HSTS): If the Persistent Memory Region is ready, then this field indicates the health status of the Persistent Memory Region. This field is always set to 000b when the Persistent Memory Region is not ready.

The health status values are defined as:

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>Normal Operation – The Persistent Memory Region is operating normally.</td></tr>
<tr><td>001b</td><td>Restore Error – The Persistent Memory Region is operating normally and is persistent; however, the contents of the Persistent Memory Region may not have been restored correctly (i.e., may not contain the contents prior to the last power cycle, NVM subsystem reset, controller reset, or Peristent Memory Region disable).</td></tr>
<tr><td>010b</td><td>Read Only – The Persistent Memory Region is read only. PCI Express memory write requests do not update the Persistent Memory Region. PCI Express memory read requests to the Persistent Memory Region return correct data.</td></tr>
<tr><td>011b</td><td>Unreliable – The Persistent Memory Region has become unreliable. PCI Express memory reads may return invalid data or generate poisoned PCI Express TLP(s). Persistent Memory Region memory writes may not update memory or may update memory with undefined data. The Persistent Memory Region may also have become non-persistent.</td></tr>
<tr><td>100b - 111b</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>8</td><td>RO</td><td>0</td><td>Not Ready (NRDY): This bit indicates if the Persistent Memory Region is ready for use. If this bit is cleared to ‘0’ and the PMRCTL.EN is set to ‘1’, then the Persistent Memory Region is ready to accept and process PCI Express memory read and write requests. If this bit is set to ‘1’ or the PMRCTL.EN bit is cleared to ‘0’, then the Persistent Memory Region is not ready to process PCI Express memory read and write requests.</td></tr>
<tr><td>7:0</td><td>RO</td><td>0h</td><td>Error (ERR): When the Persistent Memory Region is ready and operating normally, this field indicates whether previous memory writes to the Persistent Memory Region have completed without error. If this field is cleared to ‘0’, then previous writes to the Persistent Memory Region have completed without error and that the values written are persistent. A non-zero value in this field indicates the occurrence of an error that may have caused one or more of the previous writes to not have completed successfully. The meaning of any particular non-zero value is vendor specific.

Once this field takes on a non-zero value, it maintains a non-zero value until the PCI Function is reset.</td></tr>
</table>

Modify Section 4.6.1.2.2 as shown below:

4.6.1.2.2 Command Specific Errors Definition

Completion queue entries with a Status Code Type of Command Specific Errors indicate an error that is specific to a particular command opcode. Status codes of 0h to 7Fh are for Admin command errors. Status codes of 80h – BFh are specific to the selected I/O command set.

Figure 33: Status Code – Command Specific Status Values

Value	Description	Commands Affected
00h	Completion Queue Invalid	Create I/O Submission Queue
01h	Invalid Queue Identifier	Create I/O Submission Queue, Create I/O Completion Queue, Delete I/O Completion Queue, Delete I/O Submission Queue
02h	Invalid Queue Size	Create I/O Submission Queue, Create I/O Completion Queue
03h	Abort Command Limit Exceeded	Abort
04h	Reserved	
05h	Asynchronous Event Request Limit Exceeded	Asynchronous Event Request
06h	Invalid Firmware Slot	Firmware Commit
07h	Invalid Firmware Image	Firmware Commit
08h	Invalid Interrupt Vector	Create I/O Completion Queue
09h	Invalid Log Page	Get Log Page
0Ah	Invalid Format	Format NVM, Namespace Management
0Bh	Firmware Activation Requires Conventional Reset	Firmware Commit
0Ch	Invalid Queue Deletion	Delete I/O Completion Queue
0Dh	Feature Identifier Not Saveable	Set Features
0Eh	Feature Not Changeable	Set Features
0Fh	Feature Not Namespace Specific	Set Features
10h	Firmware Activation Requires NVM Subsystem Reset	Firmware Commit
11h	Firmware Activation Requires Reset	Firmware Commit
12h	Firmware Activation Requires Maximum Time Violation	Firmware Commit
13h	Firmware Activation Prohibited	Firmware Commit
14h	Overlapping Range	Firmware Commit, Firmware Image Download, Set Features
15h	Namespace Insufficient Capacity	Namespace Management
16h	Namespace Identifier Unavailable	Namespace Management
17h	Reserved	
18h	Namespace Already Attached	Namespace Attachment
19h	Namespace Is Private	Namespace Attachment
1Ah	Namespace Not Attached	Namespace Attachment
1Bh	Thin Provisioning Not Supported	Namespace Management
1Ch	Controller List Invalid	Namespace Attachment
1Dh	Device Self-test In Progress	Device Self-test
1Eh	Boot Partition Write Prohibited	Firmware Commit
1Fh	Invalid Controller Identifier	Virtualization Management
20h	Invalid Secondary Controller State	Virtualization Management
21h	Invalid Number of Controller Resources	Virtualization Management
22h	Invalid Resource Identifier	Virtualization Management
23h	Sanitize Prohibited While Persistent Memory Region is Enabled	Sanitize
24h – 6Fh	Reserved	
70h – 7Fh	Directive Specific	NOTE 1
80h – BFh	I/O Command Set Specific	NOTE 2
C0h – FFh	Vendor Specific	

NOTES:
1. The Directives Specific range defines Directives specific status values. Refer to section 9.
2. The I/O Command Set Specific range in NVMe over Fabrics defines Fabrics command specific status values.

Modify Section 4.7 as shown below:

4.7 Controller Memory Buffer

The Controller Memory Buffer (CMB) is a region of general purpose read/write memory on the controller that may be used for a variety of purposes. The controller indicates which purposes the memory may be used for by setting support flags in the CMBSZ register.

Submission Queues in host memory require the controller to perform a PCI Express read from host memory in order to fetch the queue entries. Submission Queues in controller memory enable host software to directly write the entire Submission Queue Entry to the controller's internal memory space, avoiding one read from the controller to the host. This approach reduces latency in command execution and improves efficiency in a PCI Express fabric topology that may include multiple switches. Similarly, PRP Lists or SGLs require separate fetches across the PCI Express fabric, which may be avoided by writing the PRP or SGL to the Controller Memory Buffer. Completion Queues in the Controller Memory Buffer may be used for peer to peer or other applications. For writes of small amounts of data, it may be advantageous to have the host write the data and/or metadata to the Controller Memory Buffer rather than have the controller fetch it from host memory.

The contents of the Controller Memory Buffer are initially undefined. Host software should initialize any memory before it is referenced (e.g., a Completion Queue shall be initialized by host software in order for the Phase Tag to be used correctly).

A controller memory based queue is used in the same manner as a host memory based queue – the difference is the memory address used is located within the controller's own memory rather than in the host memory. The Admin or I/O Queues may be placed in the Controller Memory Buffer. For a particular queue, all memory associated with it shall reside in either the Controller Memory Buffer or host memory. For all queues in the Controller Memory Buffer, the queue shall be physically contiguous.

The controller may support PRPs and SGLs in the Controller Memory Buffer. For a particular PRP List or SGL associated with a single command, all memory associated with the PRP List or SGLs shall reside in either the Controller Memory Buffer or host memory. The PRPs and SGLs for a command may only be placed in the Controller Memory Buffer if the associated command is present in a Submission Queue in the Controller Memory Buffer.

The controller may support data and metadata in the Controller Memory Buffer. All data or metadata associated with a particular command shall be either entirely located in ~~either~~ the Controller Memory Buffer or ~~host memory~~ outside the Controller Memory Buffer.

If the requirements for the Controller Memory Buffer use are violated by the host, the controller shall fail the associated command with Invalid Use of Controller Memory Buffer status.

The address region allocated for the CMB shall be 4 KB aligned. It is recommended that a controller allocate the CMB on an 8 KB boundary. The controller shall support burst transactions up to the maximum payload size, support byte enables, and arbitrary byte alignment. The host shall ensure that all writes to the CMB that are needed for a command have been sent before updating the SQ Tail doorbell register. The Memory Write Request to the SQ Tail doorbell register shall not have the Relaxed Ordering bit set, to ensure that it arrives at the controller after all writes to the CMB.

Insert the following section after Section 4.7:

4.8 Persistent Memory Region

The Persistent Memory Region (PMR) is an optional region of general purpose PCI Express read/write persistent memory that may be used for a variety of purposes. The controller indicates support for the PMR by setting CAP.PMRS to '1' and indicates whether the controller supports command data and metadata transfers to or from the PMR by setting support flags in the PMRCAP register. When command data and metadata transfers to or from PMR are supported, all data and metadata associated with a particular command shall be either entirely located in the Persistent Memory Region or outside the Persistent Memory Region.

The PCI Express address range and size of the PMR is defined by the PCI Base Address register (BAR) indicated by PMRCAP.BIR. The PMR consumes the entire address region exposed by the BAR and supports

all the required features of the PCI express programming model (i.e., it in no way restricts what is otherwise permitted by PCI Express).

The contents of data written to the PMR while the PMR is ready persists across power cycles, NVM subsystem resets, controller resets, and disabling of the PMR. The mechanism used to make a write to the PMR persistent is implementation specific. For example, in one implementation this may mean that a write to non-volatile memory has completed while in another implementation this may mean that the write has been stored in a non-volatile write buffer and is written to non-volatile memory at some later point.

The host enables the PMR by setting PMRCTL.EN to '1'. Once enabled, the controller indicates that the PMR is ready by setting PMRSTS.NRDY to '0'. It is not necessary to enable the controller to enable the PMR. Restoring and saving the contents of the PMR may take time to complete. When the host modifies the value of PMRCTL.EN, the host should wait for at least the time interval specified in PMRCAP.PMRTO for PMRSTS.NRDY to reflect the change.

When the PMR is not ready, PMR reads complete successfully and return an undefined value while PMR writes complete normally, but do not update memory (i.e., the contents of the PMR address written remains unchanged). The undefined value returned by a PMR read following a sanitize operation is such that recovery of any previous user data from any cache or the non-volatile media is not possible.

When the PMR becomes read-only or unreliable, then a critical warning is reported in the SMART/Health Information Log which may be used to trigger an NVMe asynchronous event. Since reporting of an asynchronous event may occur an unspecified amount of time after the PMR health status has changed, the host should assume that all operations to the PMR have been affected since the last time normal operation was reported in PMRSTS.HSTS.

PMRCAP.PMRWBM enumerates supported PMR write barrier mechanisms. At least one mechanism shall be supported. An implementation may optionally support a mechanism where a PCI Express read of any size to the PMR, including a "zero-length read," ensures that all previous memory writes (i.e., Posted PCI Express requests) to the PMR have completed and are persistent. An implementation may optionally support a write barrier mechanism that utilizes a read of the PMRSTS register. When supported, a read of the PMRSTS register allows a host to:

- ensure that previously issued memory writes to the PMR have completed; and
- determine whether the PMR updates associated with those writes have completed without error and are persistent.

A PMR memory write error may be the result of a poisoned PCI Express TLP, an NVM subsystem internal error, or a PMR health status issue.

Regardless of the supported PMR write barrier mechanisms, a host may periodically read PMRSTS to ensure that reads to the PMR have returned valid data. For example, if a read to the PMRSTS register indicates that the PMR is operating normally is then followed by a series of reads, and finally a second read to the PMRSTS register that indicates the PMR is unreliable, then one or more of the reads between the two PMRSTS reads may have returned invalid data. Such polling of the PMRSTS register may be unnecessary if the host handles poisoned TLPs and/or poisoned TLP error reporting is enabled.

Modify portion of figure 93 as shown below:

Figure 1: Get Log Page – SMART / Health Information Log

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>0</td><td>Critical Warning: This field indicates critical warnings for the state of the controller. Each bit corresponds to a critical warning type; multiple bits may be set. If a bit is cleared to ‘0’, then that critical warning does not apply. Critical warnings may result in an asynchronous event notification to the host. Bits in this field represent the current associated state and are not persistent.
<table>
<tr><th>Bit</th><th>Definition</th></tr>
<tr><td>0</td><td>If set to ‘1’, then the available spare space has fallen below the threshold.</td></tr>
<tr><td>1</td><td>If set to ‘1’, then a temperature is above an over temperature threshold or below an under temperature threshold (refer to section 5.22.1.4).</td></tr>
<tr><td>2</td><td>If set to ‘1’, then the NVM subsystem reliability has been degraded due to significant media related errors or any internal error that degrades NVM subsystem reliability.</td></tr>
<tr><td>3</td><td>If set to ‘1’, then the media has been placed in read only mode.</td></tr>
<tr><td>4</td><td>If set to ‘1’, then the volatile memory backup device has failed. This field is only valid if the controller has a volatile memory backup solution.</td></tr>
<tr><td>5</td><td>If set to ‘1’, then the Persistent Memory Region has become read-only or unreliable (refer to section 4.8).</td></tr>
<tr><td>7:~~5~~6</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

Modify Figure 104 as shown below:

Figure 104: Get Log Page – Sanitize Status Log

<table>
<tr><th>Bytes</th><th>Description</th></tr>
<tr><td>01:00</td><td>Sanitize Progress (SPROG): This field indicates the fraction complete of the sanitize operation. The value is a numerator of the fraction complete that has 65,536 (10000h) as its denominator. This value shall be set to FFFFh if the SSTAT field is not set to 010b.</td></tr>
<tr><td>03:02</td><td>Sanitize Status (SSTAT): This field indicates the status associated with the most recent sanitize operation.

Bits 15:9 are reserved.

Bit 8 (Global Data Erased) if set to ‘1’ then no namespace logical block ~~non-volatile storage~~ in the NVM subsystem has ~~not~~ been written to and no Persistent Memory Region in the NVM subsystem has been enabled:
a) since being manufactured and the NVM subsystem has never been sanitized; or
b) since the most recent successful sanitize operation.
If cleared to ‘0’, then a namespace logical block ~~non-volatile storage~~ in the NVM subsystem has been written to or a Persistent Memory Region in the NVM subsystem has been enabled:
a) since being manufactured and the NVM subsystem has never been sanitized; or
b) since the most recent successful sanitize operation of the NVM subsystem.

Bits 7:3 contains the number of completed passes if the most recent sanitize operation was an Overwrite. This field shall be cleared to 00000b if the most recent sanitize operation was not an Overwrite.

Bits 2:0 contains the status of the most recent sanitize operation as shown below.

<table>
<tr><th>Value</th><th>Definition</th></tr>
<tr><td>000b</td><td>The NVM subsystem has never been sanitized.</td></tr>
<tr><td>001b</td><td>The most recent sanitize operation completed successfully.</td></tr>
<tr><td>010b</td><td>A sanitize operation is currently in progress.</td></tr>
<tr><td>011b</td><td>The most recent sanitize operation failed.</td></tr>
<tr><td>100b-111b</td><td>Reserved</td></tr>
</table>
</td></tr>
</table>

Modify Section 5.21 as shown below:

5.21 Sanitize command

The Sanitize command is used to start a sanitize operation or to recover from a previously failed sanitize operation. The sanitize operation types that may be supported are Block Erase, Crypto Erase, and Overwrite. All sanitize operations are processed in the background (i.e., completion of the Sanitize command does not indicate completion of the sanitize operation). Refer to section 8.15 for details on the sanitize operation.

When a sanitize operation starts on any controller, all controllers in the NVM subsystem:

- Shall clear any outstanding Sanitize Operation Completed asynchronous event;
- Shall update the Sanitize Status log (refer to section 5.14.1.9.2;
- Shall abort any command (submitted or in progress) not allowed during a sanitize operation with a status of Sanitize In Progress (refer to section 8.15.1);
- Should suspend power management activities; and
- Shall release stream identifiers for any open streams.

While a sanitize operation is in progress, all controllers in the NVM subsystem shall abort any command not allowed during a sanitize operation with a status of Sanitize In Progress (refer to section 8.15.1) and the Persistent Memory Region shall behave as described in section 8.15.1.

After a sanitize operation fails, all controllers in the NVM subsystem shall abort any command not allowed during a sanitize operation with a status of Sanitize Failed (refer to section 8.15.1) and the Persistent Memory Region shall behave as described in section 8.15.1 until a subsequent sanitize operation is started or successful recovery from the failed sanitize operation occurs.

If the most recent failed sanitize operation was started in unrestricted completion mode (i.e. the AUSE bit was cleared to '0' in the Sanitize command), failure recovery requires the host to issue a subsequent Sanitize command in restricted or unrestricted completion mode or to issue a subsequent Sanitize command with the Exit Failure Mode action.

If the most recent failed sanitize operation was started in restricted completion mode (i.e. the AUSE bit was set to '1' in the Sanitize command), failure recovery requires the host to issue a subsequent Sanitize command in restricted completion mode. In the case of a sanitize operation failure in restricted completion mode, before starting another sanitize operation:

- any subsequent Sanitize command issued with the Exit Failure Mode action shall be aborted with a status of Sanitize Failed; and
- any Sanitize command issued in unrestricted completion mode shall be aborted with a status of Sanitize Failed.

The Sanitize Capabilities field in the Identify Controller data structure indicates the sanitize operation types supported. If an unsupported sanitize operation type is selected by a Sanitize command then the controller shall abort the command with a status of Invalid Field in Command.

If any Persistent Memory Region is enabled in an NVM subsystem, then the controller shall abort any Sanitize command with a status of Sanitize Prohibited While Persistent Memory Region is Enabled. A sanitize operation is prohibited while the Persistent Memory Region is enabled.

If a firmware activation is pending, then the controller shall abort any Sanitize command with a status of Firmware Activation Requires NVM Subsystem Reset. Activation of new firmware is prohibited during a sanitize operation (refer to section 8.15.1).

Modify Section 5.21.1 as shown below:

5.21.1 Command Completion

When the command is complete, the controller shall post a completion queue entry to the Admin Completion Queue indicating the status for the command. All sanitize operations are performed in the background (i.e., completion of the Sanitize command does not indicate completion of the sanitize operation). If a sanitize operation is started, then the Sanitize Status log page shall be updated before posting the completion queue entry for the command that started that sanitize operation.

Sanitize command specific status values are defined in Figure TBD1.

Figure TBD1: Sanitize – Command Specific Status Values

Value	Description
23h	**Sanitize Prohibited While Persistent Memory Region is Enabled:** A sanitize operation is prohibited while the Persistent Memory Region is enabled.

Modify Section 7.3.1 as shown below:

7.3.1 NVM Subsystem Reset

An NVM Subsystem Reset is initiated when:

- Power is applied to the NVM subsystem,
- A value of 4E564D65h (“NVMe”) is written to the NSSR.NSSRC field, or
- A vendor specific event occurs.

When an NVM Subsystem Reset occurs, the entire NVM subsystem is reset. This includes the initiation of a Controller Level Reset on all controllers that make up the NVM subsystem, disabling of the Persistent Memory Region associated with all controllers that make up the NVM subsystem, and a transition to the Detect LTSSM state by all PCI Express ports of the NVM subsystem.

Modify Section 8.15 as shown below:

8.15 Sanitize Operations (Optional)

A sanitize operation alters all user data in the NVM subsystem such that recovery of any previous user data from any cache or the non-volatile media is not possible. If a portion of the user data was not altered and the sanitize operation completed successfully, then the NVM subsystem shall ensure permanent inaccessibility of that portion of the user data for any future use within the NVM subsystem (e.g., retrieval from NVM media or caches) and is permanent inaccessibility via any interface to the NVM subsystem, including management interfaces such as NVMe-MI.

The scope of a sanitize operation is all locations in the NVM subsystem that are able to contain user data, including caches, Persistent Memory Regions, and unallocated or deallocated areas of the media. Sanitize operations do not affect the Replay Protected Memory Block, Controller Memory Buffer, boot partitions, or other media and caches that do not contain user data. A sanitize operation also may alter log pages as necessary (e.g., to prevent derivation of user data from log page information). Once a sanitize operation is started, it cannot be aborted and continues after a Controller Level Reset including across power cycles.

The Sanitize command (refer to section 5.21) is used to start a sanitize operation or to recover from a previously failed sanitize operation. All sanitize operations are performed in the background (i.e., completion of the Sanitize command does not indicate completion of the sanitize operation). The completion of a sanitize operation is indicated in the Sanitize Status log page, and with the Sanitize Operation Completed asynchronous event (if an Asynchronous Event Request Command is outstanding). The Sanitize Capabilities field of the Identify Controller data structure indicates the sanitize operation types supported.

The sanitize operation types are:

- The Block Erase sanitize operation alters user data with a low-level block erase method that is specific to the media for all locations on the media within the NVM subsystem in which user data may be stored.
- The Crypto Erase sanitize operation alters user data by changing the media encryption keys for all locations on the media within the NVM subsystem in which user data may be stored.
- The Overwrite sanitize operation alters user data by writing a fixed data pattern or related patterns to all locations on the media within the NVM subsystem in which user data may be stored one or more times. Figure 284 defines the data pattern or patterns that are written.

The Overwrite sanitize operation is media specific and may not be appropriate for all media types. For example, if the media is NAND, multiple pass overwrite operations may have an adverse effect on media endurance.

Modify Section 8.15.1 as shown below:

8.15.1 Command Restrictions

While performing a sanitize operation and while a failed sanitize operation has occurred but successful recovery from that failure has not occurred, all enabled controllers and namespaces in the NVM subsystem are restricted to performing only a limited set of actions.

While a sanitize operation is in progress:

- All controllers in the NVM subsystem shall only process the Admin commands listed in Figure 284, subject to the additional restrictions stated in that figure;
- All I/O Commands shall be aborted with a status of Sanitize In Progress; ~~and~~
- Any command or command option that is not explicitly permitted in Figure 284 shall be aborted with a status of Sanitize in Progress if fetched by any controller in the NVM subsystem~~.~~; and
- The Persistent Memory Region shall be prevented from being enabled (i.e., setting PMRCTL.EN to '1' does not result in PMRSTS.NRDY being set to '0').

While a failed sanitize operation has occurred, a subsequent sanitize operation has not started and successful recovery from the failed sanitize operation has not occurred:

- All controllers in the NVM subsystem shall only process the Sanitize command (refer to section 5.21) and the Admin commands listed in Figure 284, subject to the additional restrictions noted in that figure;
- All I/O Commands are shall be aborted with a status of Sanitize Failed;
- The Sanitize command is permitted with action restrictions (refer to section 5.21) ; ~~and~~
- Aside from the Sanitize command, any other command or command option that is not explicitly permitted in Figure 284 shall be aborted with a status of Sanitize Failed if fetched by any controller in the NVM subsystem~~.~~; and
- The Persistent Memory Region shall be prevented from being enabled (i.e., setting PMRCTL.EN to '1' does not result in PMRSTS.NRDY being set to '0').