



#### **LEGAL NOTICE:**

© **Copyright 2007 to 2021 NVM Express™, Inc. ALL RIGHTS RESERVED.**

This erratum to the NVM Express Management Interface revision 1.1 specification is proprietary to the NVM Express, Inc. (also referred to as "Company") and/or its successors and assigns.

**NOTICE TO USERS WHO ARE NVM EXPRESS, INC. MEMBERS:** Members of NVM Express, Inc. have the right to use and implement this erratum to the NVM Express Management Interface revision 1.1 specification subject, however, to the Member's continued compliance with the Company's Intellectual Property Policy and Bylaws and the Member's Participation Agreement.

**NOTICE TO NON-MEMBERS OF NVM EXPRESS, INC.:** If you are not a Member of NVM Express, Inc. and you have obtained a copy of this document, you only have a right to review this document or make reference to or cite this document. Any such references or citations to this document must acknowledge NVM Express, Inc. copyright ownership of this document. The proper copyright citation or reference is as follows: "© **2007 to 2021 NVM Express, Inc. ALL RIGHTS RESERVED.**" When making any such citations or references to this document you are not permitted to revise, alter, modify, make any derivatives of, or otherwise amend the referenced portion of this document in any way without the prior express written permission of NVM Express, Inc. Nothing contained in this document shall be deemed as granting you any kind of license to implement or use this document or the specification described therein, or any of its contents, either expressly or impliedly, or to any intellectual property owned or controlled by NVM Express, Inc., including, without limitation, any trademarks of NVM Express, Inc.

#### **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.

The NVM Express® design mark is a registered trademark of NVM Express, Inc.

NVM Express Workgroup  
c/o VTM, Inc.  
3855 SW 153<sup>rd</sup> Drive  
Beaverton, OR 97003  
USA  
info@nvmexpress.org

## NVM Express™ Technical Errata

<b>Errata ID</b>	<b>003</b>
<b>Revision Date</b>	<b>2021-02-03</b>
<b>Affected Spec Ver.</b>	<b>NVM Express Management Interface 1.1b</b>
<b>Corrected Spec Ver.</b>	<b>NVM Express Management Interface 1.1b+</b>

### Errata Author(s)

<b>Name</b>	<b>Company</b>
Myron Loewen, Mike Allison	Intel

### Errata Overview

- Eliminated use of master & slave terminology wherever possible
- Eliminated the use of “execute” terminology.
- Implement some of the older pending ECN notes

## Revision History

Revision Date	Author	Change Description
2020-09-17	Myron Loewen	<ul style="list-style-type: none"><li>Initial draft.</li><li>Eliminated master &amp; slave terminology wherever possible</li></ul>
2020-09-25	Mike Allison	<ul style="list-style-type: none"><li>Filled in the navigation pane information</li><li>Eliminated the word “execute”</li></ul>
2020-10-12	Mike Allison	<ul style="list-style-type: none"><li>Update section 2.2</li></ul>
2020-11-2	Myron Loewen	<ul style="list-style-type: none"><li>Implemented 4 of the old ECN notes</li></ul>
2020-11-20	Myron Loewen	<ul style="list-style-type: none"><li>Incorporated feedback from NVMe-MI workgroup review</li><li>Moved changes for replay during process to new TPAR6031 Progress Detect During Paused Process</li></ul>
2020-12-02	Myron Loewen	<ul style="list-style-type: none"><li>Moved clarifications for DOFST &amp; DLEN to TP6025A</li></ul>
2020-12-08	Myron Loewen	<ul style="list-style-type: none"><li>Incorporated feedback on “Invalid Message Size” and eliminated another case of Master Mode in SMBus Reset</li></ul>
2021-01-27	N/A	<ul style="list-style-type: none"><li>Integrated into the NVMe Management Interface Specification, Revision 1.1.</li></ul>
2021-02-01	Mike Allison	<ul style="list-style-type: none"><li>Removed comma from “or” with 2 conditions</li></ul>
2021-02-02	Austin Bolen	<ul style="list-style-type: none"><li>Corrected language from “larger or less” to “larger or smaller”</li></ul>
2021-02-03	Mike Allison	<ul style="list-style-type: none"><li>Accepted all changes and removed comments.</li></ul>

## Incompatible Changes

- none.

## Markup Conventions:

Black:	Unchanged (however, hot links are removed)
<del>Red Strikethrough:</del>	Deleted
Blue:	New
Blue Highlighted:	TBD values, anchors, and links to be inserted in new text.
<Green Bracketed>:	Notes to editor

## Description of Specification Changes

*Modify section 2.2 as follows:*

### 2.2 SMBus/I2C

...

SMBus/I2C elements that support ARP should be implemented as ~~Default Slave Address (DSA)~~ devices as defined by the SMBus specification. These devices should not issue “Notify ARP Master” commands.

*Modify figure 26 in section 4.1.2 as follows:*

### 4.1.2 Response Messages

...

Figure 1: Response Message Status Values

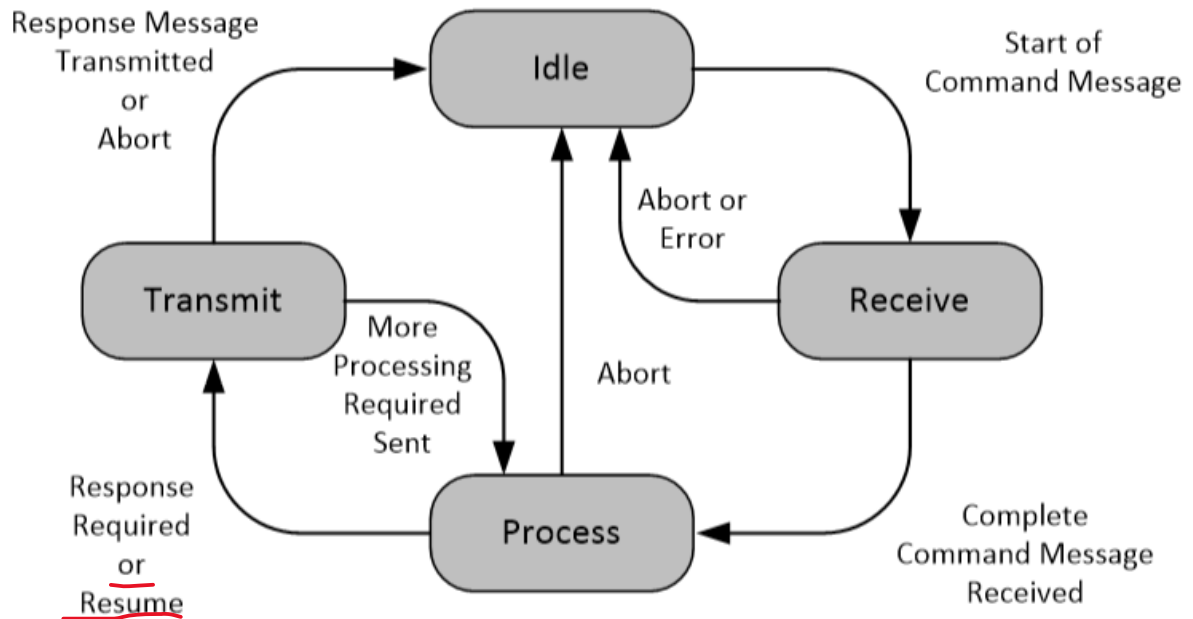
Value	Description	Error Response Format Section
00h	<b>Success:</b> The command completed successfully.	4.1.2.1
01h	<b>More Processing Required:</b> The Command Message is in progress and requires more time to complete processing. When this Response Message Status is used in a Response Message, a subsequent Response Message contains the result of the Command Message. This Response Message Status shall not be sent more than once per Request Message.	4.1.2.1
02h	<b>Internal Error:</b> The Request Message could not be processed due to a vendor specific internal error.	4.1.2.1
03h	<b>Invalid Command Opcode:</b> The associated command opcode field is not valid. Invalid opcodes include reserved and optional opcodes that are not implemented.	4.1.2.1
04h	<b>Invalid Parameter:</b> Invalid parameter field value. Request Messages received with reserved or unimplemented values in defined fields shall be completed with an Invalid Parameter Error Response. Other error conditions that result in Invalid Parameter Error Response are noted elsewhere in this specification.	4.1.2.2
05h	<b>Invalid Command Size:</b> The size of the Message Body of the <del>Command Request</del> Message was <del>larger or smaller</del> different than <del>that</del> expected due to a reason other than too much or too little Request Data (e.g., the <del>Command Request</del> Message did not contain all the required parameters or <del>Request Data was present when not expected</del> ). <del>no Request Data was expected but the Request Data is larger than that needed to contain the required parameters</del> .  The expected size of the Message Body <del>size</del> is determined by the NVMe-MI Message Type and opcode assuming no other errors are detected (e.g., Invalid Command Opcode or Invalid Parameter).	4.1.2.1

*Delete the 2 words “or resume” from fig. 30 in section 4.2 as follows to make the transition more generic for all cases:*

### 4.2 Out-of-Band Message Servicing Model

...

**Figure 30: Command Servicing State Diagram**



*Modify figure 112 in section 6 as follows:*

## 6 NVM Express Admin Command Set

...

**Figure 2: NVMe Admin Command Request Description**

Byte	Description
03:00	<b>NVMe-MI Message Header:</b> Refer to section <b>Error! Reference source not found..</b>
04	<b>Opcode (OPC):</b> This field specifies the opcode of the command <del>to be executed</del> . Refer to the NVMe Express specification.

*Modify SMBus Reset test in Section 9.3.4 as follows:*

### 9.3.4 SMBus Reset

...

If the SMBus/I2C element on an NVM Subsystem is transmitting a Response Message ~~in-master-mode~~, then an SMBus Reset shall cause it to generate a STOP condition as defined in the SMBus specification within or after the current data

...

*Modify portions of Appendix A as follows:*

## **Appendix A – Technical Note: NVM Express Basic Management Command**

...

This command does not provide any mechanism to modify or configure the NVMe device. ~~Modifying or configuring the NVMe device requires~~ Such features use the more capable MCTP protocol rather than this command's ~~simpler~~ SMBus Block Read. The host may reuse existing SMBus or FRU Information Device read subroutines for this read. ~~and is not required to switch the SMBus between master and slave modes as in MCTP.~~

The block read protocol is specified by the SMBus specification which is available online at [www.smbus.org](http://www.smbus.org). First ~~SMBus~~~~slave~~ address write and command code bytes are transmitted by the host, then a repeated start and finally a ~~SMBus~~~~slave~~ address read. The host keeps clocking as the drive ~~then~~ responds ~~in slave mode~~ with the selected data. The command code is used as a starting offset into the data block shown in **Error! Reference source not found.**, like an address on a serial EEPROM.

The offset value increments on every byte read and is reset to zero on a stop condition. A read command without a repeated start is permissible and starts transmission from offset zero. Reading more than the block length with an I2C read is also permissible and these reads continue into the first byte in the next block of data. The Packet Error Code (PEC) accumulates all bytes sent or received after the start condition and the current value is inserted whenever a PEC field is reached.

Blocks of data are packed sequentially. The first 2 blocks are defined by the NVMe-MI workgroup. The first block is the dynamic host health data. The second block includes the Vendor ID (VID) and serial number of the drive. Additional blocks of data may be defined by the owner of the VID. Reading past the end of the vendor defined blocks shall return zeros.

The SMBus ~~slave~~ address to read this data structure defaults to D4h. After the Management Controller successfully assigns the MCTP UDID to D4h using ARP, then the Basic Management Command may track and respond to ~~slave~~ reads at future ARP assigned MCTP addresses. This method of changing the Basic Command address is optional and does not persist through power cycles. Interleaved MCTP and block read traffic is permissible and neither command type shall disturb the state of the other commands.

...

The SMBus Arbitration bit may be used for simple arbitration on systems that have multiple drives on the same SMBus channel without ARP or muxes to separate them. To use this mechanism, the host follows this 3 step process to handle collisions for the same ~~SMBus~~~~slave~~ address:

1. The host does an SMBus byte write to send byte FFh which clears the SMBus Arbitration bit on all listening Management Endpoints at this ~~SMBus~~~~slave~~ address;
2. The host does an I2C read starting from offset 0h and continuing at least through the serial number in the second block. The drive transmitting a '0' when other drives sent a '1' wins arbitration and sets the arbitration bit to '1' upon read completion to give other drives priority on the next read;
3. Repeat step 2 until all drives are read, host receiving the Arbitration bit as a '1' indicates loop is done; and
4. Sort the responses by serial number since the order of drive responses varies with health status and temperatures.

...

**Figure 3: Subsystem Management Data Structure**

Command Code	Offset (byte)	Description
	01	... <b>SMBus Arbitration</b> – Bit 7 is set to '1' after an SMBus block read is completed all the way to the stop bit without bus contention and cleared to '0' if an SMBus Send Byte FFh is received on this SMBus <del>slave</del> address. ...
	07	... <b>PEC:</b> An 8 bit CRC calculated over the <del>SMBus</del> <del>slave</del> address, command code, second <del>SMBus</del> <del>slave</del> address, and returned data. The algorithm is defined in the SMBus specification.
8	08	<b>Length of identification:</b> Indicates number of additional bytes to read before encountering PEC. This value should always be 22 (16h) in implementations of this version of the spec.
	10:09	<b>Vendor ID:</b> The 2 byte vendor ID, assigned by the PCI-SIG. Should match VID in the Identify Controller command response. Note the MSB is transmitted first.
	11:30	<b>Serial Number:</b> 20 characters that match the serial number in the NVMe Identify Controller command response. Note the first character is transmitted first.
	31	<b>PEC:</b> An 8 bit CRC calculated over the <del>SMBus</del> <del>slave</del> address, command code, second <del>SMBus</del> <del>slave</del> address, and returned data. The algorithm is defined in the SMBus specification.
32+	32:255	Vendor Specific – These data structures shall not exceed the maximum read length of 255 specified in the SMBus version 3 specification. Preferably their lengths are not greater than 32 for compatibility with SMBus 2.0.

*Modify a portion of Appendix C as follows:*

## **Appendix C - Example NVMe-MI Messages over SMBus/I2C**

...

The first 4 bytes and the last byte of each packet (shown in orange in the examples below) are defined by the MCTP SMBus/I2C Transport Binding Specification. Bytes 4 to 7 of each packet and the Message Integrity Check (green) are defined by the MCTP Base Specification. The CRC-32C algorithm and the NVMe-MI Message Header (blue) are defined in section **Error! Reference source not found.**. Management Controller transmission bytes are shown in white blocks and Management Endpoint transmission bytes are shown in grey blocks. ~~All messages are sent in SMBus master mode and received in slave mode so both sides must reconfigure SMBus between commands and responses.~~ The MCTP endpoint sending the messages drives the clock pin so the signal direction changes between commands and responses as described in the MCTP binding specification.