NVM Express®

NVMe® over PCIe®

Transport Specification

Revision 1.0

May 18th, 2021

Please send comments to info@nvmexpress.org

NVMe® over PCIe® Transport Specification is available for download at http://nvmexpress.org.

SPECIFICATION DISCLAIMER

LEGAL NOTICE:

This NVMe over PCIe Transport Specification revision 1.0 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 NVMe over PCIe Transport Specification revision 1.0 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: **"© 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 153rd Drive
Beaverton, OR 97003 USA
info@nvmexpress.org

Table of Contents

Table of Figures

1 Introduction

1.1 Overview

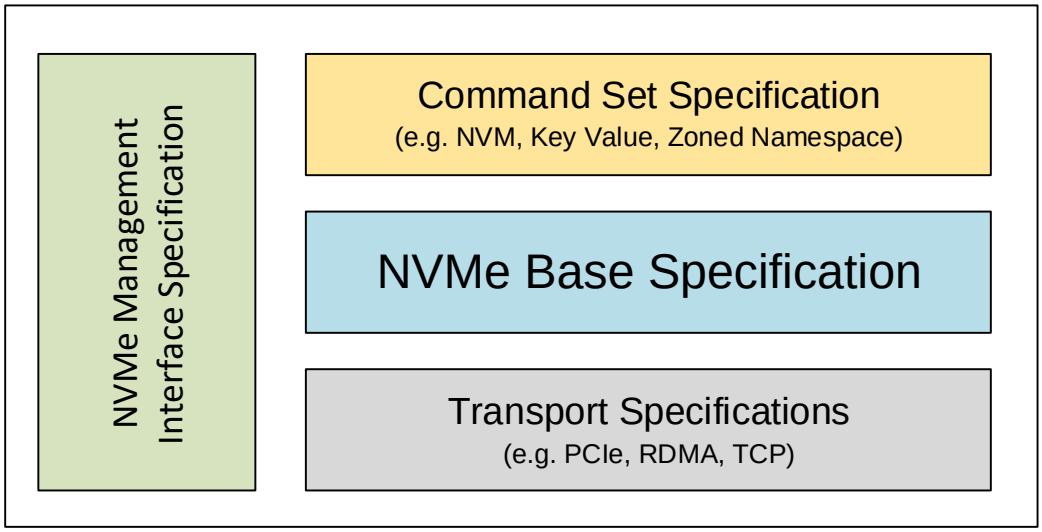
NVM Express® (NVMe®) Base specification defines an interface for host software to communicate with non-volatile memory subsystems over a variety of memory-based transports and message-based transports.

This document defines mappings of extensions defined in the NVMe Base Specification to a specific NVMe Transport: PCI Express®.

1.2 Scope

Figure 1 shows the relationship of the NVM Command Set Specification to other specifications within the NVMe Family of Specifications.

Figure 1: NVMe Family of Specifications



This specification supplements the NVMe Base Specification. This specification defines additional data structures, features, log pages, commands, and/or status values. This specification also defines extensions to existing data structures, features, log pages, commands, and/or status values. This specification defines requirements and behaviors that are specific to the PCIe transport. Functionality that is applicable generally to NVMe or that is applicable across multiple NVMe transports is defined in the NVMe Base specification.

If a conflict arises among requirements defined in different specifications, then a lower-numbered specification in the following list shall take precedence over a higher-numbered specification:

1. Non-NVMe specifications
2. NVMe Base specification
3. NVMe transport specifications
4. NVMe I/O command set specifications
5. NVMe-MI specification

1.3 Conventions

This specification conforms to the Conventions section of the NVMe Base Specification with the following exception.

Inside sections that reference registers or properties, the following terms and abbreviations are different from the NVMe Base Specification:

Reset This column indicates the value of the field after a reset as defined by the appropriate PCI or PCI Express specifications.

1.4 Definitions

1.4.1 Definitions from the NVMe Base Specification

This specification uses the definitions in the NVMe Base Specification.

1.5 References

NVM Express® (NVMe®) Base Specification revision 2.0. Available from http://www.nvmexpress.org

PCI Express Base Specification, Revision 4.0. Available from http://www.pcisig.com.

PCI-to-PCI Bridge Architecture Specification, Revision 1.2. Available from http://www.pcisig.com.

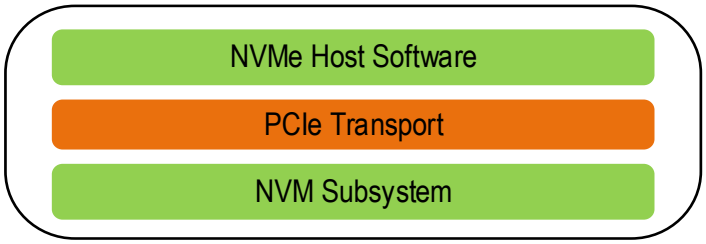
2 Transport Overview

The PCIe transport provides reliable mechanisms for memory mapped data transfer of Admin and I/O command data through memory mapped I/O transactions. The PCIe transport uses common PCIe capabilities such as:

- Memory mapped I/O for data transfer and register access;
- PCIe configuration space and
- PCIe message-based interrupts.

Refer to the PCIe specifications for a description of the details of PCIe.

Figure 2: Example of Transport Protocol Layers



3 Transport Binding

3.1 Setup & Initialization

This section describes the PCI Express register values when PCI Express is the transport.

3.1.1 PCI Device Requirements

This section details how the PCI Header, PCI Capabilities, and PCI Express Extended Capabilities should be constructed for an NVM Express controller. The fields shown are duplicated from the appropriate PCI or PCI Express specifications. The PCI documents are the normative specifications for these registers and this section details additional requirements for an NVM Express controller. Figure 3 lists the PCI Express defined register structures. Additional NVM Express requirements are defined in section 3.8.

Figure 3: PCI Express Registers

Start	End	Name	Type
00h	3Fh	PCI Header	
PMCAP	PMCAP+7h	PCI Power Management Capability	PCI Capability
MSICAP	MSICAP+9h	Message Signaled Interrupt Capability	PCI Capability
MSIXCAP	MSIXCAP+Bh	MSI-X Capability	PCI Capability
PXCAP	PXCAP+29h	PCI Express Capability	PCI Capability
AERCAP	AERCAP+47h	Advanced Error Reporting Capability	PCI Express Extended Capability

MSI-X is the recommended interrupt mechanism. However, some systems may not support MSI-X. As a result, devices may choose to support both the MSI Capability and the MSI-X Capability.

The PCI and PCI Express registers described in section 3.8 are initialized based on the system configuration. This includes configuration of power management features. A single interrupt (e.g., pin-based, single MSI, or single MSI-X) should be used until the number of I/O Queues is determined.

3.1.2 Transport Specific Controller Properties

This section details the register definitions within the transport specific section of the Controller Properties table (refer to the Controller Properties section in the NVMe Base Specification). Usage of these registers is described in section 3.3.

The PCIe transport supports Controller Properties as memory mapped registers that are located in the address range specified in the MLBAR/MUBAR registers (PCI BAR0 and BAR1). NVM Express defined registers for the PCI Express transport start at the offset defined in Figure 4. All controller registers shall be mapped to a memory space that supports in-order access and variable access widths. For many computer architectures, specifying the memory space as uncacheable produces this behavior. The host shall not issue locked accesses to registers. The host shall access registers in their native width or aligned 32-bit accesses. Violation of either of these host requirements results in undefined behavior.

Accesses that target any portion of two or more registers are not supported.

All reserved registers and all reserved bits within registers are read-only and return 0h when read.

Figure 4: PCI Express Specific Controller Property Definitions

Start	End	Symbol	Controller[1]	Name
1000h	1003h	SQ0TDBL	M	Submission Queue 0 Tail Doorbell (Admin)
1000h + (1 * (4 << CAP.DSTRD))	1003h + (1 * (4 << CAP.DSTRD))	CQ0HDBL	M	Completion Queue 0 Head Doorbell (Admin)
1000h + (2 * (4 << CAP.DSTRD))	1003h + (2 * (4 << CAP.DSTRD))	SQ1TDBL	O	Submission Queue 1 Tail Doorbell
1000h + (3 * (4 << CAP.DSTRD))	1003h + (3 * (4 << CAP.DSTRD))	CQ1HDBL	O	Completion Queue 1 Head Doorbell

Figure 4: PCI Express Specific Controller Property Definitions

Start	End	Symbol	Controller[1]	Name
1000h + (4 * (4 << CAP.DSTRD))	1003h + (4 * (4 << CAP.DSTRD))	SQ2TDBL	O	Submission Queue 2 Tail Doorbell
1000h + (5 * (4 << CAP.DSTRD))	1003h + (5 * (4 << CAP.DSTRD))	CQ2HDBL	O	Completion Queue 2 Head Doorbell
…	…	…	…	…
1000h+ (2y * (4 << CAP.DSTRD))	1003h + (2y * (4 << CAP.DSTRD))	SQyTDBL	O	Submission Queue y Tail Doorbell
1000h + ((2y + 1) * (4 << CAP.DSTRD))	1003h + ((2y + 1) * (4 << CAP.DSTRD))	CQyHDBL	O	Completion Queue y Head Doorbell
Notes: 1. O = Optional, M = Mandatory, R = Reserved. Applicable to all controller types.				

3.1.2.1 Offset (1000h + ((2y) * (4 << CAP.DSTRD))): SQyTDBL – Submission Queue y Tail Doorbell

This register defines the doorbell register that updates the Tail entry pointer for Submission Queue y. The value of y is equivalent to the Queue Identifier. This indicates to the controller that new commands have been submitted for processing.

The host should not read the doorbell registers. If a doorbell register is read, the value returned is vendor specific. Writing to a non-existent Submission Queue Tail Doorbell has undefined results.

Figure 5: Offset (1000h + ((2y) * (4 << CAP.DSTRD))): SQyTDBL – Submission Queue y Tail Doorbell

Bits	Type	Reset	Description
31:16	RO	0h	Reserved
15:00	RW	0h	**Submission Queue Tail (SQT):** Indicates the new value of the Submission Queue Tail entry pointer. This value shall overwrite any previous Submission Queue Tail entry pointer value provided. The difference between the last SQT entry pointer write and the current SQT entry pointer write indicates the number of commands added to the Submission Queue. Note: Submission Queue rollover needs to be accounted for.

3.1.2.2 Offset (1000h + ((2y + 1) * (4 << CAP.DSTRD))): CQyHDBL – Completion Queue y Head Doorbell

This register defines the doorbell register that updates the Head entry pointer for Completion Queue y. The value of y is equivalent to the Queue Identifier. This indicates Completion Queue entries that have been processed by host software.

The host should not read the doorbell registers. If a doorbell register is read, the value returned is vendor specific. Writing to a non-existent Completion Queue Head Doorbell has undefined results.

Host software should continue to process Completion Queue entries within Completion Queues regardless of whether there are entries available in a particular or any Submission Queue.

Figure 6: Offset (1000h + ((2y + 1) * (4 << CAP.DSTRD))): CQyHDBL – Completion Queue y Head Doorbell

Bits	Type	Reset	Description
31:16	RO	0h	Reserved

Figure 6: Offset (1000h + ((2y + 1) * (4 << CAP.DSTRD))): CQyHDBL – Completion Queue y Head Doorbell

Bits	Type	Reset	Description
15:00	RW	0h	**Completion Queue Head (CQH):** Indicates the new value of the Completion Queue Head entry pointer. This value shall overwrite any previous Completion Queue Head value provided. The difference between the last CQH entry pointer write and the current CQH entry pointer write indicates the number of entries that are now available for re-use by the controller in the Completion Queue. Note: Completion Queue rollover needs to be accounted for.

3.1.3 Administrative Controller Support

The PCIe transport allows an Administrative controller to have a dedicated NVMe management driver loaded through the use of an explicit PCI programming interface value (refer to CC.PI field in Figure 15).

3.2 Queue Model Instantiation

The NVM Express use of the PCIe transport is based on a paired Submission and Completion Queue mechanism. Commands are placed by host software into a Submission Queue. Completions are placed into the associated Completion Queue by the controller.

The NVM Express use of the PCIe transport supports multiple Submission Queues to utilize the same Completion Queue. Submission and Completion Queues are allocated in host-addressable memory.

When instantiating an I/O Completion Queue using the Create I/O Completion Queue command, the host specifies if interrupts will be enabled using the Interrupts Enabled (IEN) field. When interrupts are enabled a transport specific field, Interrupt Vector, shall also be initialized. The following definition of the Interrupt Vector field is used for the PCIe transport:

Figure 7: Create I/O Completion Queue – Command Dword 11

Bits	Description
31:16	**Interrupt Vector (IV):** This field indicates interrupt vector to use for this Completion Queue. This corresponds to the MSI-X or multiple message MSI vector to use. If using single message MSI or pin-based interrupts, then this field shall be cleared to 0h. In MSI-X, a maximum of 2,048 vectors are used. This value shall not be set to a value greater than the number of messages the controller supports (refer to MSICAP.MC.MME or MSIXCAP.MXC.TS). If the value is greater than the number of messages the controller supports, the controller should return an error of Invalid Interrupt Vector.

3.3 Resets

3.3.1 Controller Level Reset

The following transport specific methods initiate a Controller Level Reset in addition to the methods described in the NVMe Base Specification:

- Conventional Reset (refer to the PCI Express Base Specification); and
- Function Level Reset (refer to the PCI Express Base Specification)

In all Controller Level Reset cases except a Controller Reset (refer to the Controller Level Reset section of the NVMe Base Specification), the PCI register space is reset as defined by the PCI Express Base Specification.

3.3.2 NVM Subsystem Reset

When an NVM Subsystem Reset occurs, all PCIe links in the NVM Subsystem transition to the LTSSM Detect state.

3.4 Data Transfer Model

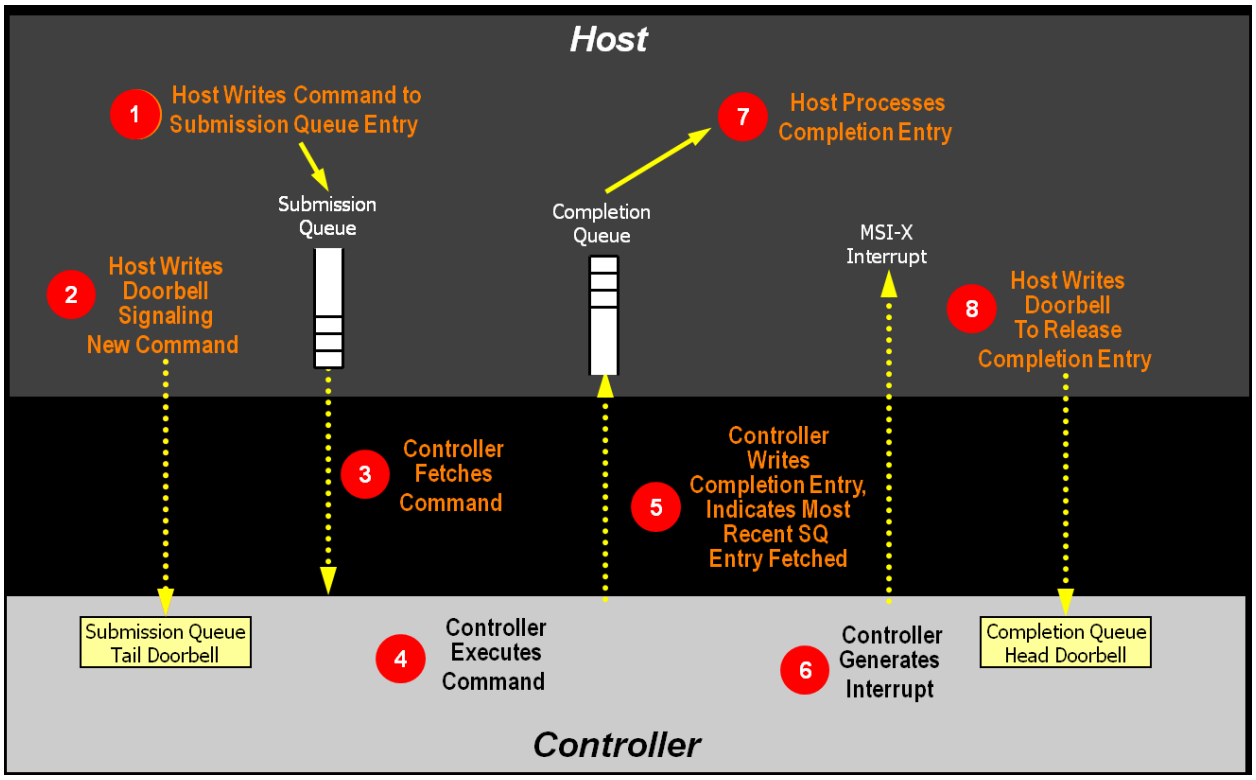
The PCIe transport is a memory-based transport that uses memory operations directly for data transfer as described in the Memory Based Theory of Operations section of the NVMe Base Specification.

3.4.1 Command Processing

This section describes command submission and completion processing for the PCIe transport. Figure 8 shows the steps to submit and complete a command. The steps are:

1. The host places one or more commands for execution in the next free Submission Queue slot(s) in memory;
2. The host updates the Submission Queue Tail Doorbell register with the new value of the Submission Queue Tail entry pointer. This indicates to the controller that a new command(s) is submitted for processing;
3. The controller transfers the command(s) in the Submission Queue slot(s) into the controller for future execution. Arbitration is the method used to determine the Submission Queue from which the controller starts processing the next candidate command(s), refer to the Command Arbitration section of the NVMe Base Specification;
4. The controller then proceeds with execution of the next command(s). Commands may complete out of order;
5. After a command has completed execution, the controller places a Completion Queue entry in the next free slot in the associated Completion Queue. As part of the Completion Queue entry, the controller indicates the most recent Submission Queue entry that has been consumed by advancing the Submission Queue Head pointer in the completion entry. Each new Completion Queue entry has a Phase Tag inverted from the previous entry to indicate to the host that this Completion Queue entry is a new entry;
6. The controller optionally generates an interrupt to the host to indicate that there is a new Completion Queue entry to consume and process. In the figure, the interrupt is shown as an MSI-X interrupt, however, it could also be a pin-based or MSI interrupt. Note that based on interrupt coalescing settings, an interrupt may or may not be generated for each new Completion Queue entry;
7. The host consumes and then processes the new Completion Queue entries in the Completion Queue. This includes taking any actions based on error conditions indicated. The host continues consuming and processing Completion Queue entries until a previously consumed entry with a Phase Tag inverted from the value of the current Completion Queue entries is encountered; and
8. The host writes the Completion Queue Head Doorbell register to indicate that the Completion Queue entry has been consumed. The host may consume many entries before updating the associated Completion Queue Head Doorbell register.

Figure 8: Command Processing



3.4.2 Command Related Resource Retirement

As part of reporting completions, the controller indicates the most recent Submission Queue entry that has been consumed. Submission Queue slots containing consumed Submission Queue entries are free and may be re-used by host software to submit new commands.

If a Completion Queue entry is posted for a command, then host software may re-use the associated PRP List(s) for that command and other resources (an exception is the PRP List for I/O Submission Queues and I/O Completion Queues).

3.5 Interrupts

The interrupt architecture allows for efficient reporting of interrupts such that the host may service interrupts through the least amount of overhead.

The specification allows the controller to be configured to report interrupts in one of four modes. The four modes are: pin-based interrupt, single message MSI, multiple message MSI, and MSI-X. It is recommended that MSI-X be used whenever possible to enable higher performance, lower latency, and lower CPU utilization for processing interrupts.

Interrupt aggregation, also referred to as interrupt coalescing, mitigates host interrupt overhead by reducing the rate at which interrupt requests are generated by a controller. This reduced host overhead typically comes at the expense of increased latency. Rather than prescribe a specific interrupt aggregation algorithm, this specification defines the mechanisms a host may use to communicate interrupt aggregation parameters to a controller and leaves the specific interrupt aggregation algorithm used by a controller as vendor specific. Interrupts associated with the Admin Completion Queue should not be delayed.

The Aggregation Threshold field in the Interrupt Coalescing feature (refer to the Interrupt Coalescing feature within the NVMe Base Specification) specifies the minimum interrupt aggregation threshold on a per vector basis. This value defines the number of Completion Queue entries that when aggregated on a per interrupt vector basis reduces host interrupt processing overhead below a host determined threshold. This value is

provided to the controller as a recommendation by the host and a controller is free to generate an interrupt before or after this aggregation threshold is achieved. The specific manner in which this value is used by the interrupt aggregation algorithm implemented by a controller is implementation specific.

The Aggregation Time field in the Interrupt Coalescing feature (refer to the Interrupt Coalescing feature within the NVMe Base Specification) specifies the maximum delay that a controller may apply to a Completion Queue entry before an interrupt is signaled to the host. This value is provided to the controller as a recommendation by the host and a controller is free to generate an interrupt before or after this aggregation time is achieved. A controller may apply this value on a per vector basis or across all vectors. The specific manner in which this value is used by the interrupt aggregation algorithm implemented by a controller is implementation specific.

Although support of the Get Features and Set Features commands associated with interrupt coalescing is required for PCIe-based NVMe controller implementations, the manner in which the Aggregation Threshold and Aggregation Time fields are used is implementation specific. For example, an implementation may ignore these fields and not implement interrupt coalescing.

3.5.1 Pin Based, Single MSI, and Multiple MSI Behavior

This is the mode of interrupt operation if any of the following conditions are met:

- Pin-based interrupts are being used – MSI (MSICAP.MC.MSIE='0') and MSI-X are disabled;
- Single MSI is being used – MSI is enabled (MSICAP.MC.MSIE='1'), MSICAP.MC.MME=000b, and MSI-X is disabled; or
- Multiple MSI is being used multiple message MSI is enabled (MSICAP.MC.MSIE='1'), MSICAP.MC.MME is set to a value between 001b and 101b inclusive, and MSI-X is disabled.

Within the controller there is an interrupt status register (IS) that is not visible to the host. In this mode, the IS register determines whether the PCI interrupt line shall be driven active or an MSI message shall be sent. Each bit in the IS register corresponds to an interrupt vector. The IS bit is set to '1' when the following conditions are true:

- There is one or more unacknowledged Completion Queue entries in a Completion Queue that utilizes this interrupt vector;
- The Completion Queue(s) with unacknowledged Completion Queue entries has interrupts enabled in the Create I/O Completion Queue command; and
- The corresponding INTM bit exposed to the host is cleared to '0', indicating that the interrupt is not masked.

For single and multiple MSI, the INTM register masks interrupt delivery prior to MSI logic. As such, an interrupt on a vector masked by INTM does not cause the corresponding Pending bit to assert within the MSI Capability Structure.

If MSIs are not enabled, IS[0] being set to '1' causes the PCI interrupt line to be active (electrical '0'). If MSIs are enabled, any change to the IS register that causes an unmasked status bit to transition from '0' to '1' or clearing of a mask bit to '0' whose corresponding status bit is set to '1' shall cause an MSI to be sent. Therefore, while in wire mode, a single wire remains active, while in MSI mode, several messages may be sent, as each edge triggered event on a port shall cause a new message.

In order to clear an interrupt for a particular interrupt vector, host software acknowledges all Completion Queue entries for Completion Queues associated with the interrupt vector.

Figure 9: Pin Based, Single MSI, and Multiple MSI Behavior

Status of IS Register	Pin-based Action	MSI Action
All bits '0' **Note:** May be caused by corresponding bit(s) in the INTM register being set to '1', masking the corresponding interrupt.	Wire inactive	No action

Figure 9: Pin Based, Single MSI, and Multiple MSI Behavior

Status of IS Register	Pin-based Action	MSI Action
One or more bits set to '1' **Note:** May be caused by corresponding bit(s) in the INTM register being cleared to '0', unmasking the corresponding interrupt.	Wire active	New message sent
One or more bits set to '1', new bit gets set to '1'	Wire active	New message sent
One or more bits set to '1', some (but not all) bits in the IS register are cleared to '0' (i.e., host software acknowledges some of the associated Completion Queue entries)	Wire active	New message sent
One or more bits set to '1', all bits in the IS register are cleared to '0' (i.e., host software acknowledges all associated Completion Queue entries)	Wire inactive	No action

3.5.1.1 Differences Between Pin Based and MSI Interrupts

Single MSI is similar to the pin-based interrupt behavior mode. The primary difference is the method of reporting the interrupt. Instead of communicating the interrupt through an INTx virtual wire, an MSI message is generated to the host. Unlike INTx virtual wire interrupts which are level sensitive, MSI interrupts are edge sensitive.

Pin-based and single MSI only use one interrupt vector. Multiple MSI may use up to 32 interrupt vectors.

For multiple MSI, the controller advertises the requested number of MSI interrupt vectors in the Multiple Message Capable (MMC) field in the Message Signaled Interrupt Message Control (MC) register. The MSICAP.MC.MMC field represents a power-of-2 wrapper on the number of requested vectors. For example, if three vectors are requested, then the MSICAP.MC.MMC field shall be '010' (four vectors).

Multiple-message MSI allows completions to be aggregated on a per vector basis. If sufficient MSI vectors are allocated, each Completion Queue(s) may send its own interrupt message, as opposed to a single message for all completions.

3.5.2 MSI-X Based Behavior

MSI-X is the preferred interrupt behavior to use. The following configuration describes this mode of interrupt operation:

- Multiple-message MSI is disabled (MSICAP.MC.MSIE is cleared to '0') and (MSICAP.MC.MME is cleared to 000b); and
- MSI-X is enabled.

MSI-X, similar to multiple-message MSI, allows completions to be aggregated on a per vector basis. However, the maximum number of vectors is 2KiB. MSI-X also allows each interrupt to send a unique message data corresponding to the vector.

MSI-X allows completions to be aggregated on a per vector basis. Each Completion Queue(s) may send its own interrupt message, as opposed to a single message for all completions.

When generating an MSI-X message, the following checks occur before generating the message:

- The Function Mask bit in the MSI-X Message Control register is cleared to '0'; and
- The corresponding vector mask in the MSI-X table structure is cleared to '0'.

If either of the mask bits are set to '1', the corresponding pending bit in the MSI-X PBA structure is set to '1' to indicate that an interrupt is pending for that vector. The MSI for that vector is later generated when both the mask bits are cleared to '0'.

It is recommended that the interrupt vector associated with the CQ(s) being processed be masked during processing of Completion Queue entries within the CQ(s) to avoid spurious and/or lost interrupts. The interrupt mask table defined as part of MSI-X should be used to mask interrupts.

3.6 Power Management

Power Management operates as defined in the NVMe Base Specification with the following specifics for the PCIe transport.

The host shall never select a power state that consumes more power than the PCI Express slot power limit control value expressed by the Captured Slot Power Limit Value (CSPLV) and Captured Slot Power Limit Scale (CSPLS) fields of the PCI Express Device Capabilities (PXDCAP) register. Hosts that do not dynamically manage power should set the power state to the lowest numbered state that satisfies the PCI Express slot power limit control value.

If a controller implements the PCI Express Dynamic Power Allocation (DPA) capability and that capability is enabled (i.e., the Substate Control Enable bit is set to '1'), then the maximum power that may be consumed by the NVM subsystem is equal to the minimum value specified by the DPA substate or the NVM Express power state, whichever is lower.

3.7 Error Handling Model

It is recommended that implementations support the Advanced Error Reporting Capability to enable more robust error handling.

3.8 Transport Specific Content

3.8.1 PCI Express Type 0/1 Common Configuration Space

Figure 10 summarizes the organization of the Type 0/1 Common Configuration Space defined in the PCI Express Base specification. The reference information in this section does not contain all PCI Express requirements. Refer to the PCI Express Base specification for more information.

Figure 10: PCI Express Type 0/1 Common Configuration Space

Start	End	Symbol	Name	Reference
00h	03h	ID	Identifiers	3.8.1.1
04h	05h	CMD	Command Register	3.8.1.2
06h	07h	STS	Device Status	3.8.1.3
08h	08h	RID	Revision ID	3.8.1.4
09h	0Bh	CC	Class Codes	3.8.1.5
0Ch	0Ch	CLS	Cache Line Size	3.8.1.6
0Dh	0Dh	MLT	Master Latency Timer	3.8.1.7
0Eh	0Eh	HTYPE	Header Type	3.8.1.8
0Fh	0Fh	BIST	Built-In Self Test (Optional)	3.8.1.9
10h	13h	MLBAR (BAR0)	Memory Register Base Address, lower 32-bits <BAR0>	3.8.1.10
14h	17h	MUBAR (BAR1)	Memory Register Base Address, upper 32-bits <BAR1>	3.8.1.11
18h	1Bh	BAR2	Index/Data Pair Register Base Address or Vendor Specific (Optional) <BAR2>	3.8.1.12
1Ch	1Fh	BAR3	Vendor Specific	3.8.1.13
20h	23h	BAR4	Vendor Specific	3.8.1.14
24h	27h	BAR5	Vendor Specific	3.8.1.15
28h	2Bh	CCPTR	CardBus CIS Pointer	3.8.1.16
2Ch	2Fh	SS	Subsystem Identifiers	3.8.1.17
30h	33h	EROM	Expansion ROM Base Address (Optional)	3.8.1.18
34h	34h	CAP	Capabilities Pointer	3.8.1.19
35h	3Bh	R	Reserved	
3Ch	3Dh	INTR	Interrupt Information	3.8.1.20
3Eh	3Eh	MGNT	Minimum Grant (Optional)	3.8.1.21
3Fh	3Fh	MLAT	Maximum Latency (Optional)	3.8.1.22

3.8.1.1 Offset 00h: ID - Identifiers

Figure 11: Offset 00h: ID - Identifiers

Bits	Type	Reset	Description
31:16	RO	Impl Spec	**Device ID (DID):** Indicates the device number assigned by the vendor. Specific to each implementation.
15:00	RO	Impl Spec	**Vendor ID (VID):** Indicates the company vendor, assigned by the PCI SIG.

3.8.1.2 Offset 04h: CMD - Command

Figure 12: Offset 04h: CMD - Command

Bits	Type	Reset	Description
15:11	RO	0h	Reserved by PCI-SIG
10	RW	0b	**Interrupt Disable (ID):** Disables the controller from generating pin-based INTx# interrupts. This bit does not have any effect on MSI or MSI-X operation.
09	RO	0b	**Fast Back-to-Back Enable (FBE):** Not supported by the NVM Express interface.
08	RW/RO	0b	**SERR# Enable (SEE):** Controls error reporting.
07	RO	0b	Reserved by PCI-SIG
06	RW/RO	0b	**Parity Error Response Enable (PEE):** When set to '1', the controller shall generate PERR# when a data parity error is detected. If parity is not supported, then this bit is read-only '0'.
05	RO	0b	**VGA Palette Snooping Enable (VGA):** Shall be cleared to zero for NVM Express use.
04	RO	0b	**Memory Write and Invalidate Enable (MWIE):** Shall be cleared to zero for NVM Express use.
03	RO	0b	**Special Cycle Enable (SCE):** Shall be cleared to zero for NVM Express use.
02	RW	0b	**Bus Master Enable (BME):** Enables the controller to act as a master for data transfers. When set to '1', bus master activity is allowed. When cleared to '0', the controller is not allowed to issue any Memory or I/O Requests.
01	RW	0b	**Memory Space Enable (MSE):** Controls access to the controller's register memory space.
00	RW	0b	**I/O Space Enable (IOSE):** Controls access to the controller's target I/O space.

3.8.1.3 Offset 06h: STS - Device Status

Figure 13: Offset 06h: STS – Device Status

Bits	Type	Reset	Description
15	RWC	0b	**Detected Parity Error (DPE):** Set to '1' by hardware when the controller detects a parity error on its interface.
14	RWC/RO	0b	**Signaled System Error (SSE):** Refer to the PCI SIG specifications.
13	RWC	0b	**Received Master-Abort (RMA):** Set to '1' by hardware when the controller receives a master abort to a cycle the controller generated.
12	RWC	0b	**Received Target Abort (RTA):** Set to '1' by hardware when the controller receives a target abort to a cycle the controller generated.
11	RO	0b	**Signaled Target-Abort (STA):** Not supported by the NVM Express interface.
10:09	RO	Impl Spec	**DEVSEL# Timing (DEVT):** Controls the device select time for the controller's PCI interface. This field is not applicable to PCI Express implementations.
08	RWC	0b	**Master Data Parity Error Detected (DPD):** Set to '1' by hardware when the controller, as a master, either detects a parity error or sees the parity error line asserted, and the Parity Error Response Enable bit (CMD.PEE) is set to '1'.
07	RO	Impl Spec	**Fast Back-to-Back Capable (FBC):** Shall be cleared to zero for NVM Express use.
06	RO	0b	Reserved by PCI-SIG
05	RO	Impl Spec	**66 MHz Capable (C66):** Shall be cleared to zero for NVM Express use.

Figure 13: Offset 06h: STS – Device Status

Bits	Type	Reset	Description
04	RO	1b	**Capabilities List (CL):** Indicates the presence of a capabilities list. The controller shall support the PCI Power Management capability as a minimum.
03	RO	0	**Interrupt Status (IS):** Indicates the interrupt status of the device ('1' = asserted).
02:00	RO	000b	Reserved by PCI-SIG

3.8.1.4 Offset 08h: RID - Revision ID

Figure 14: Offset 08h: RID - Revision ID

Bits	Type	Reset	Description
07:00	RO	Impl Spec	**Revision ID (RID):** Indicates stepping of the controller hardware.

3.8.1.5 Offset 09h: CC - Class Code

Fields in the Class Code register are described in the PCI Code and ID Assignment Specification.

Figure 15: Offset 09h: CC - Class Code

Bits	Type	Reset	Description
23:16	RO	01h	**Base Class Code (BCC):** Indicates the base class code as a mass storage controller.
15:08	RO	08h	**Sub Class Code (SCC):** Indicates the sub class code as a Non-Volatile Memory controller.
07:00	RO	02h or 03h	**Programming Interface (PI):** This field specifies that the controller uses the NVM Express programming interface. I/O Controllers shall report 02h and Administrative controllers shall report 03h as defined by the PCI Code and ID Assignment Specification.

3.8.1.6 Offset 0Ch: CLS – Cache Line Size

Figure 16: Offset 0Ch: CLS – Cache Line Size

Bits	Type	Reset	Description
07:00	RW	00h	**Cache Line Size (CLS):** Cache Line Size register is set by the system firmware or operating system to the system cache size.

3.8.1.7 Offset 0Dh: MLT – Master Latency Timer

Figure 17: Offset 0Dh: MLT – Master Latency Timer

Bits	Type	Reset	Description
07:00	RO	00h	**Master Latency Timer (MLT):** Indicates the number of clocks the controller is allowed to act as a master on PCI. For a PCI Express device, this register does not apply and shall be hardwired to '0'.

3.8.1.8 Offset 0Eh: HTYPE – Header Type

Figure 18: Offset 0Eh: HTYPE – Header Type

Bits	Type	Reset	Description
07	RO	Impl Spec	**Multi-Function Device (MFD):** Indicates whether the controller is part of a multi-function device.
06:00	RO	00h	**Header Layout (HL):** Indicates that the controller uses a target device layout.

3.8.1.9 Offset 0Fh: BIST – Built-In Self Test (Optional)

The following register is optional, but if implemented, shall look as follows. When not implemented, the register shall be read-only returning the value 0h.

Figure 19: Offset 0Fh: BIST – Built-In Self Test (Optional)

Bits	Type	Reset	Description
07	RO	Impl Spec	**BIST Capable (BC):** Indicates whether the controller has a BIST function.
06	RW	0b	**Start BIST (SB):** Host software sets this bit to '1' to invoke BIST. The controller clears this bit to '0' when BIST is complete.
05:04	RO	00b	Reserved
03:00	RO	0h	**Completion Code (CC):** Indicates the completion code status of BIST. A non-zero value indicates a failure.

3.8.1.10 Offset 10h: MLBAR (BAR0) – Memory Register Base Address, lower 32-bits

This register allocates space for the controller properties defined in the Controller Properties section of the NVMe Base specification.

Figure 20: Offset 10h: MLBAR (BAR0) – Memory Register Base Address, lower 32-bits

Bits	Type	Reset	Description
31:14	RW	0h	**Base Address (BA):** Base address of register memory space. For controllers that support a larger number of doorbell registers or have vendor specific space following the doorbell registers, more bits are allowed to be RO such that more memory space is consumed.
13:04	RO	0h	Reserved by PCI-SIG
03	RO	0b	**Prefetchable (PF):** Indicates that this range is not prefetchable.
02:01	RO	Impl Spec	**Type (TP):** Indicates where this range may be mapped. It is recommended to support mapping anywhere in 64-bit address space.
00	RO	0b	**Resource Type Indicator (RTE):** Indicates a request for register memory space.

3.8.1.11 Offset 14h: MUBAR (BAR1) – Memory Register Base Address, upper 32-bits

This register specifies the upper 32-bit address of the controller properties defined in the Controller Properties section of the NVMe Base specification.

Figure 21: Offset 14h: MUBAR (BAR1) – Memory Register Base Address, upper 32-bits

Bits	Type	Reset	Description
31:00	RW	0h	**Base Address (BA):** Upper 32-bits (bits 63:32) of the memory register base address.

Note: NVM Express implementations that reside behind PCI compliant bridges, such as PCI Express Endpoints, are restricted to having 32-bit assigned base address registers due to limitations on the maximum address that may be specified in the bridge for non-prefetchable memory. Refer to the PCI-to-PCI Bridge Architecture Specification 1.2 for more information on this restriction.

3.8.1.12 Offset 18h: BAR2 – Index/Data Pair Register Base Address or Vendor Specific (Optional)

If this register is configured as I/O space, then this register specifies the Index/Data Pair base address and is configured as shown in Figure 22. These registers are used to access the controller properties defined in defined in the Controller Properties section of the NVMe Base specification using I/O based accesses.

Figure 22: Offset 18h: BAR2 – Index/Data Pair Register Base Address or Vendor Specific (Optional)

Bits	Type	Reset	Description
31:03	RW	0h	**Base Address (BA):** Base address of Index/Data Pair registers that is 8 bytes in size.
02:01	RO	00b	Reserved
00	RO	1b	**Resource Type Indicator (RTE):** Indicates a request for register I/O space.

If this register is configured as memory space (Resource Type Indicator is cleared to '0'), then the BAR2 register is vendor specific. Vendor specific space may also be allocated at the end of the controller properties defined in the Controller Properties section of the NVMe Base specification.

3.8.1.13 Offset 1Ch to 1Fh: BAR3 – Vendor Specific

The BAR3 register is vendor specific. Vendor specific space may also be allocated at the end of the controller properties defined in the Controller Properties section of the NVMe Base specification.

3.8.1.14 Offset 20h to 23h: BAR4 – Vendor Specific

The BAR4 register is vendor specific. Vendor specific space may also be allocated at the end of the controller properties defined in the Controller Properties section of the NVMe Base specification.

3.8.1.15 Offset 24h to 27h: BAR5 – Vendor Specific

The BAR5 register is vendor specific. Vendor specific space may also be allocated at the end of the controller properties defined in the Controller Properties section of the NVMe Base specification.

3.8.1.16 Offset 28h: CCPTR – CardBus CIS Pointer

Figure 23: Offset 28h: CCPTR – CardBus CIS Pointer

Bits	Type	Reset	Description
31:00	RO	0h	Shall be cleared to zero for NVM Express use.

3.8.1.17 Offset 2Ch: SS - Subsystem Identifiers

Figure 24: Offset 2Ch: SS - Subsystem Identifiers

Bits	Type	Reset	Description
31:16	RO	HwInit	**Subsystem ID (SSID):** Indicates the subsystem identifier.
15:00	RO	HwInit	**Subsystem Vendor ID (SSVID):** Indicates the subsystem vendor identifier

3.8.1.18 Offset 30h: EROM – Expansion ROM (Optional)

If the register is not implemented, the register shall be read-only returning the value 0h.

Figure 25: Offset 30h: EROM – Expansion ROM (Optional)

Bits	Type	Reset	Description
31:00	RW	Impl Spec	**ROM Base Address (RBA):** Indicates the base address of the controller's expansion ROM. Not supported for integrated implementations.

3.8.1.19 Offset 34h: CAP – Capabilities Pointer

Figure 26: Offset 34h: CAP – Capabilities Pointer

Bits	Type	Reset	Description
7:0	RO	Impl Spec	**Capability Pointer (CP):** Indicates the first capability pointer offset.

3.8.1.20 Offset 3Ch: INTR - Interrupt Information

Figure 27: Offset 3Ch: INTR - Interrupt Information

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Interrupt Pin (IPIN):** This indicates the interrupt pin the controller uses.
07:00	RW	0h	**Interrupt Line (ILINE):** Host software written value to indicate which interrupt line (vector) the interrupt is connected to. No hardware action is taken on this register.

3.8.1.21 Offset 3Eh: MGNT – Minimum Grant

Figure 28: Offset 3Eh: MGNT – Minimum Grant

Bits	Type	Reset	Description
07:00	RO	0h	**Grant (GNT):** Not supported by the NVM Express interface.

3.8.1.22 Offset 3Fh: MLAT – Maximum Latency

Figure 29: Offset 3Fh: MLAT – Maximum Latency

Bits	Type	Reset	Description
07:00	RO	0h	**Latency (LAT):** Not supported by the NVM Express interface.

3.8.2 PCI Power Management Capabilities

Refer to the Offset 14h: CC – Controller Configuration section in the NVMe Base Specification for requirements when the PCI power management state changes.

Figure 30: PCI Power Management Capabilities

Start	End	Symbol	Name
PMCAP	PMCAP+1h	PID	PCI Power Management Capability ID
PMCAP+2h	PMCAP+3h	PC	PCI Power Management Capabilities
PMCAP+4h	PMCAP+5h	PMCS	PCI Power Management Control and Status

3.8.2.1 Offset PMCAP: PID - PCI Power Management Capability ID

Figure 31: Offset PMCAP: PID - PCI Power Management Capability ID

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Next Capability (NEXT):** Indicates the location of the next capability item in the list. This may be a capability pointer (such as Message Signaled Interrupts) or may be the last item in the list.
07:00	RO	1h	**Cap ID (CID):** Indicates that this pointer is a PCI Power Management capability.

3.8.2.2 Offset PMCAP + 2h: PC – PCI Power Management Capabilities

Figure 32: Offset PMCAP + 2h: PC – PCI Power Management Capabilities

Bits	Type	Reset	Description
15:11	RO	0h	**PME_Support (PSUP):** Not supported by the NVM Express interface.
10	RO	0b	**D2_Support (D2S):** Indicates support for the D2 power management state. Not recommended for implementation.
09	RO	0b	**D1_Support (D1S):** Indicates support for the D1 power management state. Not recommended for implementation.
08:06	RO	000b	**Aux_Current (AUXC):** Not supported by the NVM Express interface.
05	RO	Impl Spec	**Device Specific Initialization (DSI):** Indicates whether device specific initialization is required.
04	RO	0b	Reserved by PCI-SIG
03	RO	0b	**PME Clock (PMEC):** Indicates that PCI clock is not required to generate PME#.
02:00	RO	Impl Spec	**Version (VS):** Indicates support for revision 1.2 or higher revisions of the PCI Power Management Specification.

3.8.2.3 Offset PMCAP + 4h: PMCS – PCI Power Management Control and Status

Figure 33: Offset PMCAP + 4h: PMCS – PCI Power Management Control and Status

Bits	Type	Reset	Description
15	RWC	0b	**PME Status (PMES):** Refer to the PCI SIG specifications.
14:13	RO	00b	**Data Scale (DSC):** Refer to the PCI SIG specifications.
12:09	RO / RW	0h	**Data Select (DSE):** If PME is not supported, then this field is read-only '0'. Refer to the PCI SIG specifications.
08	RO / RW	0b	**PME Enable (PMEE):** If PME is not supported, then this bit is read-only '0'. Refer to the PCI SIG specifications.
07:04	RO	0h	Reserved by PCI-SIG
03	RO	1b	**No Soft Reset (NSFRST):** A value of '1' indicates that the controller transitioning from $D3_{hot}$ to D0 because of a power state command does not perform an internal reset.
02	RO	0b	Reserved by PCI-SIG
01:00	RW	00b	**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: 00b – D0 state 01b – D1 state 10b – D2 state 11b – $D3_{HOT}$ 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.

3.8.3 Message Signaled Interrupt Capability (Optional)

Figure 34: Message Signaled Interrupt Capability (Optional)

Start	End	Symbol	Name
MSICAP	MSICAP+1h	MID	Message Signaled Interrupt Capability ID
MSICAP+2h	MSICAP+3h	MC	Message Signaled Interrupt Message Control
MSICAP+4h	MSICAP+7h	MA	Message Signaled Interrupt Message Address
MSICAP+8h	MSICAP+Bh	MUA	Message Signaled Interrupt Upper Address
MSICAP+Ch	MSICAP+Dh	MD	Message Signaled Interrupt Message Data
MSICAP+10h	MSICAP+13h	MMASK	Message Signaled Interrupt Mask Bits (Optional)
MSICAP+14h	MSICAP+17h	MPEND	Message Signaled Interrupt Pending Bits (Optional)

3.8.3.1 Offset MSICAP: MID – Message Signaled Interrupt Identifiers

Figure 35: Offset MSICAP: MID – Message Signaled Interrupt Identifiers

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be a capability pointer or may be the last item in the list.
07:00	RO	5h	**Capability ID (CID):** Indicates this is a Message Signaled Interrupt (MSI) capability.

3.8.3.2 Offset MSICAP + 2h: MC – Message Signaled Interrupt Message Control

Figure 36: Offset MSICAP + 2h: MC – Message Signaled Interrupt Message Control

Bits	Type	Reset	Description
15:09	RO	0h	Reserved
08	RO	Impl Spec	**Per-Vector Masking Capable (PVM):** Specifies whether controller supports MSI per-vector masking.
07	RO	1b	**64 Bit Address Capable (C64):** Specifies whether the controller is capable of generating 64-bit messages. NVM Express controllers shall be 64-bit capable.
06:04	RW	000b	**Multiple Message Enable (MME):** Indicates the number of messages the controller should assert. Controllers that only support single message MSI may implement this field as read-only.
03:01	RO	Impl Spec	**Multiple Message Capable (MMC):** Indicates the number of messages the controller is requesting.
00	RW	0b	**MSI Enable (MSIE):** If set to '1', MSI is enabled. If cleared to '0', MSI operation is disabled.

3.8.3.3 Offset MSICAP + 4h: MA – Message Signaled Interrupt Message Address

Figure 37: Offset MSICAP + 4h: MA – Message Signaled Interrupt Message Address

Bits	Type	Reset	Description
31:02	RW	0h	**Address (ADDR):** Lower 32 bits of the system specified message address, always dword aligned.
01:00	RO	00b	Reserved by PCI-SIG

3.8.3.4 Offset MSICAP + 8h: MUA – Message Signaled Interrupt Upper Address

Figure 38: Offset MSICAP + 8h: MUA – Message Signaled Interrupt Upper Address

Bits	Type	Reset	Description
31:00	RW	0h	**Upper Address (UADDR):** Upper 32 bits of the system specified message address. This register is required when the MSI Capability is supported by the controller.

3.8.3.5 Offset MSICAP + Ch: MD – Message Signaled Interrupt Message Data

Figure 39: Offset MSICAP + Ch: MD – Message Signaled Interrupt Message Data

Bits	Type	Reset	Description
15:00	RW	0h	**Data (DATA):** This 16-bit field is programmed by system software if MSI is enabled. Its content is driven onto the lower word (PCI AD[15:0]) during the data phase of the MSI memory write transaction.

3.8.3.6 Offset MSICAP + 10h: MMASK – Message Signaled Interrupt Mask Bits (Optional)

Figure 40: Offset MSICAP + 10h: MMASK – Message Signaled Interrupt Mask Bits (Optional)

Bits	Type	Reset	Description
31:00	RW	0h	**Mask Bits (MASK):** For each Mask bit that is set to '1', the function is prohibited from sending the associated message.

3.8.3.7 Offset MSICAP + 14h: MPEND – Message Signaled Interrupt Pending Bits (Optional)

Figure 41: Offset MSICAP + 14h: MPEND – Message Signaled Interrupt Pending Bits (Optional)

Bits	Type	Reset	Description
31:00	RO	0h	**Pending Bits (PEND):** For each Pending bit that is set to '1', the function has a pending associated message.

3.8.4 MSI-X Capability (Optional)

Figure 42: MSI-X Capability (Optional)

Start	End	Symbol	Name
MSIXCAP	MSIXCAP+1h	MXID	MSI-X Capability ID
MSIXCAP+2h	MSIXCAP+3h	MXC	MSI-X Message Control
MSIXCAP+4h	MSIXCAP+7h	MTAB	MSI-X Table Offset and Table BIR
MSIXCAP+8h	MSIXCAP+Bh	MPBA	MSI-X PBA Offset and PBA BIR

Note: It is recommended that the host allocate a unique MSI-X vector for each Completion Queue.

The Table BIR and PBA BIR data structures may be allocated in either BAR0-1 or BAR4-5 in implementations. These tables should be 4 KiB aligned. The memory page(s) that comprise the Table BIR and PBA BIR shall not include other registers/structures. It is recommended that these structures be allocated in BAR0-1 following the Submission Queue and Completion Queue Doorbell registers. Refer to the PCI reference for more information on allocation requirements for these data structures.

3.8.4.1 Offset MSIXCAP: MXID – MSI-X Identifiers

Figure 43: Offset MSIXCAP: MXID – MSI-X Identifiers

Bits	Type	Reset	Description
15:08	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be a capability pointer or may be the last item in the list.
07:00	RO	11h	**Capability ID (CID):** Indicates this is an MSI-X capability.

3.8.4.2 Offset MSIXCAP + 2h: MXC – MSI-X Message Control

Figure 44: Offset MSIXCAP + 2h: MXC – MSI-X Message Control

Bits	Type	Reset	Description
15	RW	0b	**MSI-X Enable (MXE):** If set to '1' and the MSI Enable bit in the MSI Message Control register is cleared to '0', the function is permitted to use MSI-X to request service and is prohibited from using its INTx# pin (if implemented). If cleared to '0', the function is prohibited from using MSI-X to request service.
14	RW	0b	**Function Mask (FM):** If set to '1', all of the vectors associated with the function are masked, regardless of their per vector Mask bit states. If cleared to '0', each vector's Mask bit determines whether the vector is masked or not. Setting the MSI-X Function Mask bit to '1' or clearing to '0' has no effect on the state of the per vector Mask bits.
13:11	RO	000b	Reserved by PCI-SIG

Figure 44: Offset MSIXCAP + 2h: MXC – MSI-X Message Control

Bits	Type	Reset	Description
10:00	RO	Impl Spec	**Table Size (TS):** This value indicates the size of the MSI-X Table as the value *n*, which is encoded as *n* - 1. For example, a returned value of 3h corresponds to a table size of 4.

3.8.4.3 Offset MSIXCAP + 4h: MTAB – MSI-X Table Offset / Table BIR

Figure 45: Offset MSIXCAP + 4h: MTAB – MSI-X Table Offset / Table BIR

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:03</td><td>RO</td><td>Impl Spec</td><td>Table Offset (TO): Used as an offset from the address contained by one of the function's Base Address registers to point to the base of the MSI-X Table. The lower three Table BIR bits are masked off (cleared to 000b) by system software to form a 32-bit qword aligned offset.</td></tr>
<tr><td>02:00</td><td>RO</td><td>Impl Spec</td><td>Table BIR (TBIR): This field indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the function's MSI-X Table into system memory.
<table>
<tr><th>BIR Value</th><th>BAR Offset</th></tr>
<tr><td>0h</td><td>10h</td></tr>
<tr><td>1h</td><td>n/a</td></tr>
<tr><td>2h</td><td>n/a</td></tr>
<tr><td>3h</td><td>Reserved</td></tr>
<tr><td>4h</td><td>20h</td></tr>
<tr><td>5h</td><td>24h</td></tr>
<tr><td>6h</td><td>Reserved</td></tr>
<tr><td>7h</td><td>Reserved</td></tr>
</table>
For a 64-bit Base Address register, the Table BIR indicates the lower dword. With PCI-to-PCI bridges, BIR values 2h to 5h are also reserved.</td></tr>
</table>

3.8.4.4 Offset MSIXCAP + 8h: MPBA – MSI-X PBA Offset / PBA BIR

Figure 46: Offset MSIXCAP + 8h: MPBA – MSI-X PBA Offset / PBA BIR

<table>
<tr><th>Bits</th><th>Type</th><th>Reset</th><th>Description</th></tr>
<tr><td>31:03</td><td>RO</td><td>Impl Spec</td><td>PBA Offset (PBAO): Used as an offset from the address contained by one of the function's Base Address registers to point to the base of the MSI-X PBA. The lower three PBA BIR bits are masked off (cleared to 000b) by software to form a 32-bit qword-aligned offset.</td></tr>
<tr><td>02:00</td><td>RO</td><td>Impl Spec</td><td>PBA BIR (PBIR): This field indicates which one of a function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the function's MSI-X PBA into system memory.
<table>
<tr><th>BIR Value</th><th>BAR Offset</th></tr>
<tr><td>0h</td><td>10h</td></tr>
<tr><td>1h</td><td>n/a</td></tr>
<tr><td>2h</td><td>n/a</td></tr>
<tr><td>3h</td><td>Reserved</td></tr>
<tr><td>4h</td><td>20h</td></tr>
<tr><td>5h</td><td>24h</td></tr>
<tr><td>6h</td><td>Reserved</td></tr>
<tr><td>7h</td><td>Reserved</td></tr>
</table></td></tr>
</table>

3.8.5 PCI Express Capability

The PCI Express Capability definitions below are based on the PCI Express Base Specification Revision 2.1. Implementations may choose to base the device on a specification beyond the PCI Express Base

Specification Revision 2.1. In all cases, the PCI Express Base Specification is the normative reference for the PCI Express Capability registers.

Note: TLP poisoning is a mandatory capability for PCI Express implementations. There are optional features of TLP poisoning, such as TLP poisoning for a transmitter. When an NVM Express controller has an error on a transmission to the host (e.g., error for a Read command), the error should be indicated as part of the NVM Express command status and not via TLP poisoning.

Figure 47: PCI Express Capability

Start	End	Symbol	Name
PXCAP	PXCAP+1h	PXID	PCI Express Capability ID
PXCAP+2h	PXCAP+3h	PXCAP	PCI Express Capabilities
PXCAP+4h	PXCAP+7h	PXDCAP	PCI Express Device Capabilities
PXCAP+8h	PXCAP+9h	PXDC	PCI Express Device Control
PXCAP+Ah	PXCAP+Bh	PXDS	PCI Express Device Status
PXCAP+Ch	PXCAP+Fh	PXLCAP	PCI Express Link Capabilities
PXCAP+10h	PXCAP+11h	PXLC	PCI Express Link Control
PXCAP+12h	PXCAP+13h	PXLS	PCI Express Link Status
PXCAP+24h	PXCAP+27h	PXDCAP2	PCI Express Device Capabilities 2
PXCAP+28h	PXCAP+29h	PXDC2	PCI Express Device Control 2

3.8.5.1 Offset PXCAP: PXID – PCI Express Capability ID

Figure 48: Offset PXCAP: PXID – PCI Express Capability ID

Bits	Type	Reset	Description
15:8	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be a capability pointer or may be the last item in the list.
7:0	RO	10h	**Capability ID (CID):** Indicates that this capability structure is a PCI Express capability.

3.8.5.2 Offset PXCAP + 2h: PXCAP – PCI Express Capabilities

Figure 49: Offset PXCAP + 2h: PXCAP – PCI Express Capabilities

Bits	Type	Reset	Description
15:14	RO	00b	Reserved by PCI-SIG
13:9	RO	Impl Spec	**Interrupt Message Number (IMN):** This field indicates the MSI/MSI-X vector that is used for the interrupt message generated in association with any of the status bits of this capability structure. There are no status bits that generate interrupts defined in this capability within this specification, thus this field is not used.
8	RO	0b	**Slot Implemented (SI):** Not applicable for PCI Express Endpoint devices.
7:4	RO	0h	**Device/Port Type (DPT):** Indicates the specific type of this PCI Express function. This device shall be indicated as a PCI Express Endpoint.
3:0	RO	2h	**Capability Version (VER):** Indicates that this capability structure is a PCI Express capability structure.

3.8.5.3 Offset PXCAP + 4h: PXDCAP – PCI Express Device Capabilities

Figure 50: Offset PXCAP + 4h: PXDCAP – PCI Express Device Capabilities

Bits	Type	Reset	Description
31:29	RO	000b	Reserved by PCI-SIG
28	RO	1b	**Function Level Reset Capability (FLRC):** A value of '1' indicates the Function supports the optional Function Level Reset mechanism. NVM Express controllers shall support Function Level Reset.
27:26	RO	00b	**Captured Slot Power Limit Scale (CSPLS):** Specifies the scale used for the Slot Power Limit Value.

Figure 50: Offset PXCAP + 4h: PXDCAP – PCI Express Device Capabilities

Bits	Type	Reset	Description
25:18	RO	0h	**Captured Slot Power Limit Value (CSPLV):** In combination with the Slot Power Limit Scale value, specifies the upper limit on power supplied by the slot. Power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field.
17:16	RO	00b	Reserved by PCI-SIG
15	RO	1b	**Role-based Error Reporting (RER):** When set to '1', indicates that the Function implements role-based error reporting. This functionality is required.
14:12	RO	000b	Reserved by PCI-SIG
11:9	RO	Impl Spec	**Endpoint L1 Acceptable Latency (L1L):** This field indicates the acceptable latency that the Endpoint is able to withstand due to a transition from the L1 state to the L0 state.
08:06	RO	Impl Spec	**Endpoint L0s Acceptable Latency (L0SL):** This field indicates the acceptable total latency that the Endpoint is able to withstand due to the transition from L0s state to the L0 state.
05	RO	Impl Spec	**Extended Tag Field Supported (ETFS):** This bit indicates the maximum supported size of the Tag field as a Requester.
04:03	RO	Impl Spec	**Phantom Functions Supported (PFS):** This field indicates the support for use of unclaimed Function Numbers to extend the number of outstanding transactions allowed by logically combining unclaimed Function Numbers with the Tag identifier.
02:00	RO	Impl Spec	**Max_Payload_Size Supported (MPS):** This field indicates the maximum payload size that the Function may support for TLPs.

3.8.5.4 Offset PXCAP + 8h: PXDC – PCI Express Device Control

Figure 51: Offset PXCAP + 8h: PXDC – PCI Express Device Control

Bits	Type	Reset	Description
15	RW	0b	**Initiate Function Level Reset:** A write of '1' initiates Function Level Reset to the Function. The value read by software from this bit shall always '0'.
14:12	RW/ RO	Impl Spec	**Max_Read_Request_Size (MRRS):** This field sets the maximum Read Request size for the Function as a Requester. The Function shall not generate Read Requests with size exceeding the set value.
11	RW/ RO	Impl Spec	**Enable No Snoop (ENS):** If this bit is set to '1', the Function is permitted to set the No Snoop bit in the Requestor Attributes of transactions the Function initiates that do not require hardware enforced cache coherency. This bit may be hardwired to '0' if a Function would never set the No Snoop attribute in transactions the Function initiates.
10	RW/ RO	0b	**AUX Power PM Enable (APPME):** If this bit is set to '1', enables a Function to draw AUX power independent of PME AUX power. Functions that do not implement this capability hardware this bit to '0'.
09	RW/ RO	0b	**Phantom Functions Enable (PFE):** If this bit is set to '1', enables a Function to use unclaimed Functions as Phantom Functions to extend the number of outstanding transaction identifiers. If this bit is cleared to '0', the Function is not allowed to use Phantom Functions.
08	RW/ RO	0b	**Extended Tag Enable (ETE):** If this bit is set to '1', enables a Function to use an 8-bit Tag field as a Requester. If this bit is cleared to '0', the Function is restricted to a 5-bit Tag field.
07:05	RW/ RO	000b	**Max_Payload_Size (MPS):** This field sets the maximum TLP payload size for the Function. As a receiver, the Function shall handle TLPs as large as the set value. As a transmitter, the Function shall not generate TLPs exceeding the set value. Functions that support only the 128-byte max payload size are permitted to hardwire this field to 0h.
04	RW/ RO	Impl Spec	**Enable Relaxed Ordering (ERO):** If this bit is set to '1', the Function is permitted to set the Relaxed Ordering bit in the Attributes field of transactions the Function initiates that do not require strong write ordering.
03	RW	0b	**Unsupported Request Reporting Enable (URRE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending error messages.

Figure 51: Offset PXCAP + 8h: PXDC – PCI Express Device Control

Bits	Type	Reset	Description
02	RW	0b	**Fatal Error Reporting Enable (FERE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending ERR_FATAL messages.
01	RW	0b	**Non-Fatal Error Reporting Enable (NFERE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending ERR_NONFATAL messages.
00	RW	0b	**Correctable Error Reporting Enable (CERE):** This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending ERR_COR messages.

3.8.5.5 Offset PXCAP + Ah: PXDS – PCI Express Device Status

Figure 52: Offset PXCAP + Ah: PXDS – PCI Express Device Status

Bits	Type	Reset	Description
15:06	RO	0h	Reserved by PCI-SIG
05	RO	0b	**Transactions Pending (TP):** When set to '1', this bit indicates that the Function has issued non-posted requests that have not been completed. This bit is cleared to '0' only when all outstanding non-posted requests have completed or have been terminated by the completion timeout mechanism. This bit shall also be cleared to '0' upon completion of a Function Level Reset.
04	RO	Impl Spec	**AUX Power Detected (APD):** Functions that require AUX power report this bit as set to '1' if AUX power is detected by the Function.
03	RWC	0b	**Unsupported Request Detected (URD):** When set to '1', this bit indicates that the Function received an Unsupported Request. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
02	RWC	0b	**Fatal Error Detected (FED):** When set to '1', this bit indicates the status of fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
01	RWC	0b	**Non-Fatal Error Detected (NFED):** When set to '1', this bit indicates the status of non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.
00	RWC	0b	**Correctable Error Detected (CED):** When set to '1', this bit indicates status of correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.

3.8.5.6 Offset PXCAP + Ch: PXLCAP – PCI Express Link Capabilities

Figure 53: Offset PXCAP + Ch: PXLCAP – PCI Express Link Capabilities

Bits	Type	Reset	Description
31:24	RO	HwInit	**Port Number (PN):** This field specifies the PCI Express port number for this device.
23	RO	0b	Reserved by PCI-SIG
22	RO	HwInit	**ASPM Optionality Compliance (AOC):** This bit specifies Active State Power Management (ASPM) support.
21	RO	0b	**Link Bandwidth Notification Capability (LBNC):** Not applicable to Endpoints.
20	RO	0b	**Data Link Layer Link Active Reporting Capable (DLLLA):** Not applicable to Endpoints.
19	RO	0b	**Surprise Down Error Reporting Capable (SDERC):** Not applicable to Endpoints.
18	RO	Impl Spec	**Clock Power Management (CPM):** If this bit is set to '1', the component tolerates the removal of any reference clock(s) via the "clock request" (CLKREQ#) mechanism when the Link is in the L1 and L2/L3 Ready Link states. If this bit is cleared to '0', the component does not have this capability and that reference clock(s) shall not be removed in these Link states.
17:15	RO	Impl Spec	**L1 Exit Latency (L1EL):** This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this port requires to complete transition from L1 to L0.

Figure 53: Offset PXCAP + Ch: PXLCAP – PCI Express Link Capabilities

Bits	Type	Reset	Description
14:12	RO	Impl Spec	**L0s Exit Latency (L0SEL):** This field indicates the L0s exit latency for the given PCI Express Link. The value reported indicates the length of time this port requires to complete transition from L0s to L0.
11:10	RO	Impl Spec	**Active State Power Management Support (ASPMS):** This field indicates the level of ASPM supported on the given PCI Express Link.
09:04	RO	Impl Spec	**Maximum Link Width (MLW):** This field indicates the maximum Link width (x*n* – corresponding to *n* lanes) implemented by the component.
03:00	RO	Impl Spec	**Supported Link Speeds (SLS):** This field indicates the supported Link speed(s) of the associated port.

3.8.5.7 Offset PXCAP + 10h: PXLC – PCI Express Link Control

Figure 54: Offset PXCAP + 10h: PXLC – PCI Express Link Control

Bits	Type	Reset	Description
15:10	RO	0h	Reserved by PCI-SIG
09	RW/ RO	0b	**Hardware Autonomous Width Disable (HAWD):** If set to '1', disables hardware from changing the Link width for reasons other than attempting to correct unreliable Link operation by reducing Link width. Components that do not implement the ability autonomously to change Link width are permitted to hardwire this bit to '0'.
08	RW	0b	**Enable Clock Power Management (ECPM):** If cleared to '0', clock power management is disabled and the device shall hold the CLKREQ# signal low. If set to '1', the device is permitted to use the CLKREQ# signal to power manage the Link clock according to the protocol defined for mini PCI Express.
07	RW	0b	**Extended Synch (ES):** If set to '1', this bit forces the transmission of additional Ordered Sets when exiting the L0s state and when in the Recovery state. This mode provides external devices (e.g., logic analyzers) monitoring the Link time to achieve bit and symbol lock before the Link enters the L0 state and resumes communication.
06	RW	0b	**Common Clock Configuration (CCC):** If set to '1', this bit indicates that this component and the component at the opposite end of this Link are operating with a distributed common reference clock. If cleared to '0', this component and the component at the opposite end of this Link are operating with asynchronous reference clocks.
05:04	RO	00b	Reserved: These bits are reserved on Endpoints.
03	RW	0b	**Read Completion Boundary (RCB):** Indicates the RCB value of the root port.
02	RO	0b	Reserved by PCI-SIG
01:00	RW	00b	**Active State Power Management Control (ASPMC):** This field controls the level of ASPM executed on the PCI Express Link.

3.8.5.8 Offset PXCAP + 12h: PXLS – PCI Express Link Status

Figure 55: Offset PXCAP + 12h: PXLS – PCI Express Link Status

Bits	Type	Reset	Description
15:13	RO	000b	Reserved by PCI-SIG
12	RO	Impl Spec	**Slot Clock Configuration:** If set to '1', this bit indicates that the component uses the same physical reference clock that the platform provides on the connector. If the device uses an independent clock irrespective of a reference on the connector, this bit shall be cleared to '0'.
11:10	RO	00b	Reserved by PCI-SIG
09:04	RO	n/a	**Negotiated Link Width (NLW):** This field indicates the negotiated Link width. This field is undefined when the Link is not up.
03:00	RO	n/a	**Current Link Speed (CLS):** This field indicates the negotiated Link speed. This field is undefined when the Link is not up.

3.8.5.9 Offset PXCAP + 24h: PXDCAP2 – PCI Express Device Capabilities 2

Figure 56: Offset PXCAP + 24h: PXDCAP2 – PCI Express Device Capabilities 2

Bits	Type	Reset	Description
31:24	RO	0h	Reserved by PCI-SIG
23:22	RO	Impl Spec	**Max End-End TLP Prefixes (MEETP):** Indicates the maximum number of End-End TLP Prefixes supported by this Function. TLPs received by this Function that contain more End-End TLP Prefixes than are supported shall be handled as Malformed TLPs.
21	RO	Impl Spec	**End-End TLP Prefix Supported (EETPS):** Indicates whether End-End TLP Prefix support is offered by a Function. If cleared to '0', there is no support. If set to '1', the Function supports receiving TLPs containing End-End TLP Prefixes.
20	RO	Impl Spec	**Extended Fmt Field Supported (EFFS):** If set to '1', the Function supports the 3-bit definition of the Fmt field. If cleared to '0', the Function supports a 2-bit definition of the Fmt field.
19:18	RO	Impl Spec	**OBFF Supported (OBFFS):** This field indicates the level of support for OBFF.
17:14	RO	0h	Reserved by PCI-SIG
13:12	RO	Impl Spec	**TPH Completer Supported (TPHCS):** Defined encodings are listed in the following table. Value / Definition: 00b – TPH and Extended TPH Completer not supported 01b – TPH Completer supported; Extended TPH Completer not supported 10b – Reserved 11b – Both TPH and Extended TPH Completer supported
11	RO	Impl Spec	**Latency Tolerance Reporting Supported (LTRS):** If set to '1', then the latency tolerance reporting mechanism is supported.
10	RO	0b	**No RO-enabled PR-PR Passing (NPRPR):** Not applicable to the NVM Express interface.
09	RO	Impl Spec	**128-bit CAS Completer Supported (128CCS):** This bit shall be set to '1' if the Function supports this optional capability.
08	RO	Impl Spec	**64-bit AtomicOp Completer Supported (64AOCS):** Includes FetchAdd, Swap, and CAS AtomicOps. This bit shall be set to '1' if the Function supports this optional capability.
07	RO	Impl Spec	**32-bit AtomicOp Completer Supported (32AOCS):** Includes FetchAdd, Swap, and CAS AtomicOps. This bit shall be set to '1' if the Function supports this optional capability.
06	RO	0b	**AtomicOp Routing Supported (AORS):** Not applicable to the NVM Express interface.
05	RO	0b	**ARI Forwarding Supported (ARIFS):** Not applicable for the NVM Express interface.
04	RO	1b	**Completion Timeout Disable Supported (CTDS):** A value of '1' indicates support for the Completion Timeout Disable mechanism. The Completion Timeout Disable mechanism is required for Endpoints that issue requests on their own behalf.
03:00	RO	Impl Spec	**Completion Timeout Ranges Supported (CTRS):** This field indicates device function support for the optional Completion Timeout programmability mechanism.

3.8.5.10 Offset PXCAP + 28h: PXDC2 – PCI Express Device Control 2

Figure 57: Offset PXCAP + 28h: PXDC2 – PCI Express Device Control 2

Bits	Type	Reset	Description
31:15	RO	0h	Reserved by PCI-SIG
14:13	RW	Impl Spec	**OBFF Enable (OBFFE):** This field controls the capabilities enabled for OBFF.
12:11	RO	00b	Reserved by PCI-SIG
10	RW	0b	**Latency Tolerance Reporting Mechanism Enable (LTRME):** When set to '1', enables the LTR mechanism. When cleared to '0', the LTR mechanism is disabled.
09:05	RO	0h	Reserved by PCI-SIG

Figure 57: Offset PXCAP + 28h: PXDC2 – PCI Express Device Control 2

Bits	Type	Reset	Description
04	RW	0b	**Completion Timeout Disable (CTD):** When set to '1', this bit disables the Completion Timeout mechanism.
03:00	RW/RO	Impl Spec	**Completion Timeout Value:** Specifies the completion timeout value. If this feature is not supported in PXDCAP2, then this field is read-only 0h.

3.8.6 Advanced Error Reporting Capability (Optional)

The Advanced Error Reporting definitions below are based on the PCI Express Base Specification Revision 2.1. Implementations may choose to base the device on a specification beyond the PCI Express Base Specification Revision 2.1. In all cases, the PCI Express Base Specification is the normative reference for the Advanced Error Reporting registers.

Figure 58: Advanced Error Reporting Capability (Optional)

Start	End	Symbol	Name
AERCAP	AERCAP+3h	AERID	AER Capability ID
AERCAP+4h	AERCAP+7h	AERUCES	AER Uncorrectable Error Status Register
AERCAP+8h	AERCAP+Bh	AERUCEM	AER Uncorrectable Error Mask Register
AERCAP+Ch	AERCAP+Fh	AERUCESEV	AER Uncorrectable Error Severity Register
AERCAP+10h	AERCAP+13h	AERCES	AER Correctable Error Status Register
AERCAP+14h	AERCAP+17h	AERCEM	AER Correctable Error Mask Register
AERCAP+18h	AERCAP+1Bh	AERCC	AER Advanced Error Capabilities and Control Register
AERCAP+1Ch	AERCAP+2Bh	AERHL	AER Header Log Register
AERCAP+38h	AERCAP+47h	AERTLP	AER TLP Prefix Log Register (Optional)

3.8.6.1 Offset AERCAP: AERID – AER Capability ID

Figure 59: Offset AERCAP: AERID – AER Capability ID

Bits	Type	Reset	Description
31:20	RO	Impl Spec	**Next Pointer (NEXT):** Indicates the next item in the list. This may be a capability pointer or may be the last item in the list.
19:16	RO	Impl Spec	**Capability Version (CVER):** Indicates the version of the capability structure. Reset value may be 1h or 2h.
15:0	RO	1h	**Capability ID (CID):** Indicates that this capability structure is an Advanced Error Reporting capability.

3.8.6.2 Offset AERCAP + 4: AERUCES – AER Uncorrectable Error Status Register

This register indicates the error detection status of the individual errors on the controller. These bits are sticky – they are neither initialized nor modified during a hot reset or Function Level Reset (FLR).

Figure 60: Offset AERCAP + 4: AERUCES – AER Uncorrectable Error Status Register

Bits	Type	Reset	Description
31:26	RO	0h	Reserved by PCI-SIG
25	RWC/RO	0b	**TLP Prefix Blocked Error Status (TPBES)** (Optional)
24	RWC/RO	0b	**AtomicOp Egress Blocked Status (AOEBS)** (Optional)
23	RWC/RO	0b	**MC Blocked TLP Status (MCBTS)** (Optional)
22	RWC/RO	0b	**Uncorrectable Internal Error Status (UIES)** (Optional)
21	RWC/RO	0b	**ACS Violation Status (ACSVS)** (Optional)
20	RWC	0b	**Unsupported Request Error Status (URES)**
19	RWC/RO	0b	**ECRC Error Status (ECRCES)** (Optional)
18	RWC	0b	**Malformed TLP Status (MTS)**
17	RWC/RO	0b	**Receiver Overflow Status (ROS)** (Optional)
16	RWC	0b	**Unexpected Completion Status (UCS)**

Figure 60: Offset AERCAP + 4: AERUCES – AER Uncorrectable Error Status Register

Bits	Type	Reset	Description
15	RWC/RO	0b	**Completer Abort Status (CAS)** (Optional)
14	RWC	0b	**Completion Timeout Status (CTS)**
13	RWC/RO	0b	**Flow Control Protocol Error Status (FCPES)** (Optional)
12	RWC	0b	**Poisoned TLP Status (PTS)**
11:05	RO	0h	Reserved by PCI-SIG
04	RWC	0b	**Data Link Protocol Error Status (DLPES)**
03:00	RO	0h	Reserved by PCI-SIG

3.8.6.3 Offset AERCAP + 8: AERUCEM – AER Uncorrectable Error Mask Register

This register controls the reporting of the individual errors by the controller. A masked error is not reported in the Header Log register (AERHL), does not update the First Error Pointer (AERCC.FEP), and is not reported to the host. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Figure 61: Offset AERCAP + 8: AERUCEM – AER Uncorrectable Error Mask Register

Bits	Type	Reset	Description
31:26	RO	0h	Reserved by PCI-SIG
25	RW/RO	0b	**TLP Prefix Blocked Error Mask (TPBEM)** (Optional)
24	RW/RO	0b	**AtomicOp Egress Blocked Mask (AOEBM)** (Optional)
23	RW/RO	0b	**MC Blocked TLP Mask (MCBTM)** (Optional)
22	RW/RO	1b	**Uncorrectable Internal Error Mask (UIEM)** (Optional)
21	RW/RO	0b	**ACS Violation Mask (ACSVM)** (Optional)
20	RW	0b	**Unsupported Request Error Mask (UREM)**
19	RW/RO	0b	**ECRC Error Mask (ECRCEM)** (Optional)
18	RW	0b	**Malformed TLP Mask (MTM)**
17	RW/RO	0b	**Receiver Overflow Mask (ROM)** (Optional)
16	RW	0b	**Unexpected Completion Mask (UCM)**
15	RW/RO	0b	**Completer Abort Mask (CAM)** (Optional)
14	RW	0b	**Completion Timeout Mask (CTM)**
13	RW/RO	0b	**Flow Control Protocol Error Mask (FCPEM)** (Optional)
12	RW	0b	**Poisoned TLP Mask (PTM)**
11:05	RO	0h	Reserved by PCI-SIG
04	RW	0b	**Data Link Protocol Error Mask (DLPEM)**
03:00	RO	0h	Reserved by PCI-SIG

3.8.6.4 Offset AERCAP + Ch: AERUCESEV – AER Uncorrectable Error Severity Register

This register controls whether an individual error is reported as a non-fatal or a fatal error. An error is reported as fatal when the corresponding error bit in the severity register is set to ‘1’. If the bit is cleared to ‘0’, the corresponding error is considered non-fatal. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Figure 62: Offset AERCAP + Ch: AERUCESEV – AER Uncorrectable Error Severity Register

Bits	Type	Reset	Description
31:26	RO	0h	Reserved by PCI-SIG
25	RW/RO	0b	**TLP Prefix Blocked Error Severity (TPBESEV)** (Optional)
24	RW/RO	0b	**AtomicOp Egress Blocked Severity (AOEBSEV)** (Optional)
23	RW/RO	0b	**MC Blocked TLP Severity (MCBTSEV)** (Optional)
22	RW/RO	1b	**Uncorrectable Internal Error Severity (UIESEV)** (Optional)
21	RW/RO	0b	**ACS Violation Severity (ACSVSEV)** (Optional)
20	RW	0b	**Unsupported Request Error Severity (URESEV)**
19	RW/RO	0b	**ECRC Error Severity (ECRCESEV)** (Optional)
18	RW	1b	**Malformed TLP Severity (MTSEV)**

Figure 62: Offset AERCAP + Ch: AERUCESEV – AER Uncorrectable Error Severity Register

Bits	Type	Reset	Description
17	RW/RO	1b	**Receiver Overflow Severity (ROSEV)** (Optional)
16	RW	0b	**Unexpected Completion Severity (UCSEV)**
15	RW/RO	0b	**Completer Abort Severity (CASEV)** (Optional)
14	RW	0b	**Completion Timeout Severity (CTSEV)**
13	RW/RO	1b	**Flow Control Protocol Error Severity (FCPESEV)** (Optional)
12	RW	0b	**Poisoned TLP Severity (PTSEV)**
11:05	RO	0h	Reserved by PCI-SIG
04	RW	1b	**Data Link Protocol Error Severity (DLPESEV)**
03:00	RO	0h	Reserved by PCI-SIG

3.8.6.5 Offset AERCAP + 10h: AERCES – AER Correctable Error Status Register

This register reports error status of individual correctable error sources from the controller. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Figure 63: Offset AERCAP + 10h: AERCES – AER Correctable Error Status Register

Bits	Type	Reset	Description
31:16	RO	0h	Reserved by PCI-SIG
15	RWC/RO	0b	**Header Log Overflow Status (HLOS)** (Optional)
14	RWC/RO	0b	**Corrected Internal Error Status (CIES)** (Optional)
13	RWC	0b	**Advisory Non-Fatal Error Status (ANFES)**
12	RWC	0b	**Replay Timer Timeout Status (RTS)**
11:09	RO	000b	Reserved by PCI-SIG
08	RWC	0b	**REPLAY_NUM Rollover Status (RRS)**
07	RWC	0b	**Bad DLLP Status (BDS)**
06	RWC	0b	**Bad TLP Status (BTS)**
05:01	RO	0h	Reserved by PCI-SIG
00	RWC	0b	**Receiver Error Status (RES)**

3.8.6.6 Offset AERCAP + 14h: AERCEM – AER Correctable Error Mask Register

This register controls the reporting of the individual correctable errors by the controller. A masked error is not reported to the host. These bits are sticky – they are neither initialized nor modified during a hot reset or FLR.

Figure 64: Offset AERCAP + 14h: AERCEM – AER Correctable Error Mask Register

Bits	Type	Reset	Description
31:16	RO	0h	Reserved by PCI-SIG
15	RW/RO	1b	**Header Log Overflow Mask (HLOM)** (Optional)
14	RW/RO	1b	**Corrected Internal Error Mask (CIEM)** (Optional)
13	RW	1b	**Advisory Non-Fatal Error Mask (ANFEM)**
12	RW	0b	**Replay Timer Timeout Mask (RTM)**
11:09	RO	000b	Reserved by PCI-SIG
08	RW	0b	**REPLAY_NUM Rollover Mask (RRM)**
07	RW	0b	**Bad DLLP Mask (BDM)**
06	RW	0b	**Bad TLP Mask (BTM)**
05:01	RO	0h	Reserved by PCI-SIG
00	RW	0b	**Receiver Error Mask (REM)**

3.8.6.7 Offset AERCAP + 18h: AERCC – AER Capabilities and Control Register

Figure 65: Offset AERCAP + 18h: AERCC – AER Capabilities and Control Register

Bits	Type	Reset	Description
31:12	RO	0h	Reserved by PCI-SIG
11	RO	0b	**TLP Prefix Log Present (TPLP):** If this bit is set to '1' and FEP is valid, this indicates that the TLP Prefix Log register contains valid information. This bit is sticky and is neither initialized nor modified during a hot reset or FLR.
10	RW/RO	0b	**Multiple Header Recording Enable (MHRE):** If this bit is set to '1', this enables the controller to generate more than one error header. This bit is sticky – it is neither initialized nor modified during a hot reset or FLR. If the controller does not implement the associated mechanism, then this bit is cleared to '0'.
09	RW/RO	Impl Spec	**Multiple Header Recording Capable (MHRC):** If this bit is set to '1', indicates that the controller is capable of generating more than one error header.
08	RW/RO	0b	**ECRC Check Enable (ECE):** If this bit is set to '1', indicates that the ECRC checking is enabled. This bit is sticky – it is neither initialized nor modified during a hot reset or FLR. If the controller does not implement the associated mechanism, then this bit is cleared to '0'.
07	RO	Impl Spec	**ECRC Check Capable (ECC):** If this bit is set to '1', indicates that the controller is capable of checking ECRC.
06	RW/RO	0b	**ECRC Generation Enable (EGE):** If this bit is set to '1', indicates that the ECRC generation is enabled. This bit is sticky – it is neither initialized nor modified during a hot reset or FLR. If the controller does not implement the associated mechanism, then this bit is cleared to '0'.
05	RO	Impl Spec	**ECRC Generation Capable (EGC):** If this bit is set to '1', indicates that the controller is capable of generating ECRC.
04:00	RO	0h	**First Error Pointer (FEP):** This field identifies the bit position of the first error reported in the AERUCES register. This field is sticky – it is neither initialized nor modified during a hot reset or FLR.

3.8.6.8 Offset AERCAP + 1Ch: AERHL – AER Header Log Register

This register contains the header for the TLP corresponding to a detected error. This register is sticky – it is neither initialized nor modified during a hot reset or FLR.

Figure 66: Offset AERCAP + 1Ch: AERHL – AER Header Log Register

Byte	Type	Reset	Description
0	RO	0h	**Header Byte 3 (HB3)**
1	RO	0h	**Header Byte 2 (HB2)**
2	RO	0h	**Header Byte 1 (HB1)**
3	RO	0h	**Header Byte 0 (HB0)**
4	RO	0h	**Header Byte 7 (HB7)**
5	RO	0h	**Header Byte 6 (HB6)**
6	RO	0h	**Header Byte 5 (HB5)**
7	RO	0h	**Header Byte 4 (HB4)**
8	RO	0h	**Header Byte 11 (HB11)**
9	RO	0h	**Header Byte 10 (HB10)**
10	RO	0h	**Header Byte 9 (HB9)**
11	RO	0h	**Header Byte 8 (HB8)**
12	RO	0h	**Header Byte 15 (HB15)**
13	RO	0h	**Header Byte 14 (HB14)**
14	RO	0h	**Header Byte 13 (HB13)**
15	RO	0h	**Header Byte 12 (HB12)**

3.8.6.9 Offset AERCAP + 38h: AERTLP – AER TLP Prefix Log Register (Optional)

This register contains the End-End TLP prefix(es) for the TLP corresponding to a detected error. This register is sticky – it is neither initialized nor modified during a hot reset or FLR.

Figure 67: Offset AERCAP + 38h: AERTLP – AER TLP Prefix Log Register (Optional)

Byte	Type	Reset	Description
0	RO	0h	**First TLP Prefix Log Byte 3 (TPL1B3)**
1	RO	0h	**First TLP Prefix Log Byte 2 (TPL1B2)**
2	RO	0h	**First TLP Prefix Log Byte 1 (TPL1B1)**
3	RO	0h	**First TLP Prefix Log Byte 0 (TPL1B0)**
4	RO	0h	**Second TLP Prefix Log Byte 3 (TPL2B3)**
5	RO	0h	**Second TLP Prefix Log Byte 2 (TPL2B2)**
6	RO	0h	**Second TLP Prefix Log Byte 1 (TPL2B1)**
7	RO	0h	**Second TLP Prefix Log Byte 0 (TPL2B0)**
8	RO	0h	**Third TLP Prefix Log Byte 3 (TPL3B3)**
9	RO	0h	**Third TLP Prefix Log Byte 2 (TPL3B2)**
10	RO	0h	**Third TLP Prefix Log Byte 1 (TPL3B1)**
11	RO	0h	**Third TLP Prefix Log Byte 0 (TPL3B0)**
12	RO	0h	**Fourth TLP Prefix Log Byte 3 (TPL4B3)**
13	RO	0h	**Fourth TLP Prefix Log Byte 2 (TPL4B2)**
14	RO	0h	**Fourth TLP Prefix Log Byte 1 (TPL4B1)**
15	RO	0h	**Fourth TLP Prefix Log Byte 0 (TPL4B0)**

3.8.7 Other Capability Pointers

Though not mentioned in this specification, other capability pointers may be necessary, depending upon the implementation. Examples would be the PCI-X capability for PCI-X implementations, and potentially the vendor specific capability pointer.

These capabilities are beyond the scope of this specification.

Annex A. Host Considerations (Informative)

A.1 Submitting an NVMe Command with PCIe

Build a submission queue entry as described in the Host Annex of the NVMe Base Specification.

Host software writes the corresponding Submission Queue Tail Doorbell register (SQxTDBL) to submit one or more commands for processing.

The write to the Submission Queue Tail Doorbell register triggers the controller to consume one or more new commands contained in the submission queue entry. The controller indicates the most recent submission queue entry that has been consumed as part of reporting completions. Host software may use this information to determine when submission queue slots may be re-used for new commands.

A.2 Processing Completed Commands

Host software processes the interrupt generated by the controller for command completion(s). If MSI-X or multiple message MSI is in use, then the interrupt vector infers the Completion Queue(s) with new command completions for the host to process. If pin-based interrupts or single message MSI interrupts are used, then host software interrogates the Completion Queue(s) to determine if new Completion Queue entries are present for the host to process.

Once the host software determines the Completion Queue (CQy) that generated the interrupt:

1. Host software reads a Completion Queue entry from the specified Completion Queue;
2. Host software processes the CQ entry to identify the Submission Queue entry that generated this completion. DW2.SQID indicates the Submission Queue ID and DW3.CID indicates the command that generated the completion;
3. DW3.SF indicates the status of the completion;
4. Host software indicates available Completion Queue slots by updating the corresponding Completion Queue Head Doorbell register (CQyHDBL). By updating CQyHDBL, the associated interrupt is cleared; and
5. If the Status Field field in the completion queue entry specifies an error, then host software performs error recovery actions (refer to the Command and Queue Error Handling section of the NVMe Base specification).

A.3 Host Software Interrupt Handling

It is recommended that host software utilize the Interrupt Mask Set and Interrupt Mask Clear (INTMS/INTMC) registers to efficiently handle interrupts when configured to use pin-based or MSI messages. Specifically, within the interrupt service routine, host software should set the appropriate mask register bits to ‘1’ in the INTMS register to mask interrupts. In the deferred procedure call, host software should process all Completion Queue entries and acknowledge the Completion Queue entries that have been processed by writing the associated CQyHDBL doorbell registers. When all Completion Queue entries have been processed, host software should unmask interrupts by clearing the appropriate mask register bits to ‘0’ in the INTMC register.

It is recommended that the MSI interrupt vector associated with the CQ(s) being processed be masked during processing of Completion Queue entries within the CQ(s) to avoid spurious and/or lost interrupts. For single message or multiple message MSI, the INTMS and INTMC registers should be used to appropriately mask interrupts during Completion Queue entry processing.

A.3.1. Interrupt Example (Informative)

An example of the host software flow for processing interrupts is described in this section. This example assumes multiple message MSI is used and that interrupt vector 3 is associated with I/O Completion Queue 3:

1. The controller posts a Completion Queue entry to I/O Completion Queue 3. The controller sets IS[3] to ‘1’ in its internal IS register. The controller asserts an interrupt to the host;
2. The interrupt service routine (ISR) is triggered;

3. Host software scans all I/O Completion Queues associated with the asserted MSI vector to determine the location of new Completion Queue entries. In this case, a new Completion Queue entry has been posted to I/O Completion Queue 3;
4. Host software writes 08h to the INTMS register to mask interrupts for interrupt vector 3, which is associated with I/O Completion Queue 3;
5. The controller masks interrupt vector 3, based on the host write to the INTMS register;
6. Host software schedules a deferred procedure call (DPC) to process the completed command;
7. The deferred procedure call (DPC) is triggered;
8. Host software processes new Completion Queue entries for I/O Completion Queue 3, completing the associated commands to the OS. Host software updates CQyHDBL to acknowledge the processed Completion Queue entries and clear the interrupt associated with those Completion Queue entries. If all Completion Queue entries have been acknowledged by host software, the controller de-asserts interrupt vector 3; and
9. Host software unmasks interrupt vector 3 by writing 08h to the INTMC register.