INTEL® DATA STREAMING ACCELERATOR ARCHITECTURE SPECIFICATION

Order Number: 341204-004US
Revision: 2.0
September 2022

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, with the sole exception that you may publish an unmodified copy. You may create software implementations based on this document and in compliance with the foregoing that are intended to execute on the Intel product(s) referenced in this document. No rights are granted to create modifications or derivatives of this document.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm.

Table of Contents

List of Figures

List of Tables

Revision History

Date	Revision	Description
November 2019	Rev 1.0	- Initial release of the document.
October 2020	Rev 1.1	- Addressed errata and omissions in Rev 1.0. - Added guarantee of descriptor ordering under certain conditions. - Added Command Capabilities register (CMDCAP). - Added Dummy Portal. - Added WQ ATS Disable. - Added constraint on the value of Global Bandwidth Token Limit. - Added Release Interrupt Handle command. - Added information on Performance Monitoring. - Added details on Create Delta Record and Apply Delta Record. - Added details on CRC and DIF operations. - Removed Interrupt Handle Request capability. Instead, the Command Capabilities register is used to indicate support for the Request Interrupt Handle command. - Clarified use of the Request Interrupt Handle command and described interrupt handle revocation. - Changed description of Abort All command to require that no descriptors be submitted to the device while it is being processed. - Changed Command register to write-only. - Clarified intended use of unlimited portals for SWQs. - Clarified behavior of IMS portals when IMS is not available. - Clarified behavior of Ignore field in MSI-X Permissions and IMS.
October 2021	Rev 1.2	- Addressed errata and omissions in Rev 1.1. - Added Interrupt Handles Revoked in INTCAUSE. - Changed the process of interrupt handle revocation and added pseudocode describing the software sequence to support it. - Changed the operand type for the Release Interrupt Handle command. - Renamed Bandwidth Tokens to Read Buffers, including renaming the associated fields in GRPCAP, GENCFG, and GRPCFG. (Note, there are not change bars for this name change.) - Clarified the behavior and usage of Read Buffer controls. - Moved the table of Administrative Command Error Codes from section 9.2.13 to section 5.7.3. - Clarified that the Strict Ordering flag in a descriptor guarantees ordering of memory writes both within the device and without. - Clarified that software must not rely on the values of fields in completion records for error codes where the fields do not have specified meanings. - Clarified that bits 11:0 of the Fault Address field in a completion record or the SWERROR register may be reported as 0. - Clarified that the Cache Control flag in descriptors is reserved if the corresponding Cache Control Support field in GENCAP is 0.

September 2022	Rev 2.0	- Added Inter-domain operations and related registers including Inter-Domain Capabilities Register, Permissions Table, Submitter bitmap, etc. - Extensions to Fill, CRC, and Data Integrity Field (DIF) operations. - Support for Event Log. - Support for per WQ OPCFG. - Support for Translation Fetch descriptor. - Support for engine pipeline depth control. - Error code changes, and updates to SWERROR register and Completion Record Fault Info. - Support for WQ PRS Disable control. - Removed Steering Tag Support. - Removed support for Cache Flush with write-back semantics. - Perfmon updates. - Corrected typographical errors.

Glossary

Acronym	Term	Description
ATS	Address Translation Services	A protocol defined by the PCI Express specification to support address translations by a device and to issue ATC invalidations.
ATC	Address Translation Cache	A structure in the device that stores translated addresses. Also known as Device TLB.
BD	Batch descriptor	A descriptor that refers to an array of descriptors in memory, to allow submitting multiple work descriptors at once.
	Completion Record	A 32-byte data structure in memory that is written by the device when an operation completes.
	Dedicated Mode	A mode that allows a single software client to submit work without unnecessary overhead.
	Descriptor	A 64-byte data structure written to the device to specify work to be performed.
DWQ	Dedicated Work Queue	A work queue used by a single software client to submit work.
DMWr	Deferrable Memory Write	A type of PCI Express transaction that allows the device to defer (temporarily refuse) the write request.
	Engine	An independent operational unit within the Intel DSA device.
ENQCMD	Enqueue Command	An Intel® 64 CPU instruction to enqueue a command to a shared work queue using Deferrable Memory Write (DMWr).
ENQCMDS	Enqueue Command Supervisor	An Intel® 64 CPU instruction to enqueue a command with Supervisor permissions (from privileged software) to a shared work queue using Deferrable Memory Write (DMWr).
IDPT	Inter-Domain Permissions Table	Table to manage inter-domain operations.
IDPTE	Inter-Domain Permissions Table Entry	Any entry in the IDPT.
IMS	Interrupt Message Storage	A Scalable I/O Virtualization feature used to store MSI messages in a device-specific manner.
IOMMU	I/O Memory Management Unit	DMA Remapping Hardware Unit as defined by Intel® Virtualization Technology for Directed I/O.
	Group	A configurable set of work queues and engines.
MMIO	Memory-Mapped I/O	
MOVDIR64B	Move 64-Bytes as Direct Store	An Intel® 64 CPU instruction used to enqueue a command to a dedicated work queue using a 64-byte memory write.

Acronym	Term	Description
MSI	Message Signaled Interrupt	A memory write operation to a pre-defined address to generate an interrupt.
MSI-X		A PCI Express feature used to configure Message Signaled Interrupts.
PASID	Process Address Space Identifier	A value used in memory transactions to convey the address space on the host of an address used by the device.
PM	Persistent Memory	Memory that retains state when power is removed, such as battery-backed DRAM or Intel® Optane™ DC persistent memory.
PRS	Page Request Service	A protocol defined by the PCI Express specification for a device to report recoverable page-faults and receive page-fault responses.
RSVD	Reserved	Any field that is described as reserved in this specification must be written as 0 by software. Generally, hardware reports an error if a reserved field is non-zero, but it may not do so in all cases. If software sets a reserved field to a non-zero value and no error is reported, behavior is undefined.
SoC	System-on-chip	An integrated chip composed of host processors, accelerators, memory, and I/O agents.
SR-IOV	Single Root I/O Virtualization	A PCI Express standard for virtualizing PCI Express endpoint device interfaces.
SVM	Shared Virtual Memory	Ability for an accelerator I/O device to operate in the same virtual memory space of applications on host processors. It also implies ability to operate from page-able memory, avoiding functional requirements to pin memory for DMA operations.
	Shared Mode	A mode that allows multiple software clients to concurrently submit work.
SWQ	Shared Work Queue	A work queue that allows multiple software clients to concurrently submit work.
TC	Traffic Class	A PCI Express feature that allows differentiation of transactions to apply appropriate servicing policies.
VDCM	Virtual Device Composition Module	A software component that is part of a VMM, which composes a virtual device and makes it available to a VM.
VDEV	Virtual Device	A virtual device implemented by VDCM.
WD	Work Descriptor	A descriptor that specifies a DMA operation.
WQ	Work Queue	A queue in the device used to store descriptors submitted by software until they can be dispatched.

§

1 Introduction

This document describes the architecture of the Intel® Data Streaming Accelerator (Intel® DSA), including the extensions in the 2nd generation of Intel DSA. Intel DSA is a high-performance data copy and transformation accelerator that will be integrated in future Intel® processors, targeted for optimizing streaming data movement and transformation operations common with applications for high-performance storage, networking, persistent memory, and various data processing applications.

Intel DSA replaces Intel® QuickData Technology, which is a part of Intel® I/O Acceleration Technology.

1.1 Audience

The intended audience for this specification is hardware engineers and SoC architects building compliant hardware implementations, device driver software developers programming the device, virtualization software providers efficiently enabling sharing and virtualization of the device, and application or library developers utilizing Intel DSA operations.

1.2 References

Description
Intel® 64 and IA-32 Architectures Software Developer's Manuals https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html
Intel® Architecture Instruction Set Extensions Programming Reference https://software.intel.com/content/www/us/en/develop/download/intel-architecture-instruction-set-extensions-programming-reference.html
PCI Express Base Specification 4.0 http://www.pcisig.com/specifications/pciexpress
Intel® Virtualization Technology for Directed I/O Specification https://software.intel.com/content/www/us/en/develop/download/intel-virtualization-technology-for-directed-io-architecture-specification.html
Intel® Scalable I/O Virtualization Technical Specification https://software.intel.com/content/www/us/en/develop/download/intel-scalable-io-virtualization-technical-specification.html
Intel® I/O Acceleration Technology https://www.intel.com/content/www/us/en/wireless-network/accel-technology.html
RFC 3720, Internet Small Computer Systems Interface http://www.ietf.org/rfc/rfc3720.txt

Table 1-1: References

§

2 Overview

The goal of Intel DSA is to provide higher overall system performance for data mover and transformation operations, while freeing up CPU cycles for higher level functions. Intel DSA hardware supports high-performance data mover capability to/from volatile memory, persistent memory, memory-mapped I/O, and through a Non-Transparent Bridge (NTB) in the SoC to/from remote volatile and persistent memory on another node in a cluster. It provides a PCI Express compatible programming interface to the Operating System and can be controlled through a device driver.

In addition to performing basic data mover operations, Intel DSA is designed to perform some number of higher-level transformation operations on memory. For example, it can generate and test CRC checksum or Data Integrity Field (DIF) on the memory region to support usages typical with storage and networking applications. It supports a memory compare operation for equality, generates a delta record, and applies a delta record to a buffer. These are compared and the delta generate/merge functions may be utilized by applications such as VM migration, VM fast check-pointing, and software managed memory deduplication usages.

Intel DSA may also be used for data movement between different address spaces by using the Inter-Domain capabilities of the device. This may have application in networking, for example with a virtual switch implementation to efficiently copy data between virtual machines, or to speed up inter-process communication (IPC) primitives in the OS or VMM. It may also be used for message and data passing between processes in application domains like HPC and Machine Learning.

2.1 High Level Usages

This section summarizes some of the envisioned data movement and transformation usages for Intel DSA.

- **Datacenter**: As a data movement offload engine to reduce datacenter tax for memory copying, zeroing, etc., to free up CPU cycles from mundane infrastructure work.
- **Storage**: For data movement in storage appliances, both within the node and across nodes using Non-Transparent Bridge (NTB); and for CRC generation and Data Integrity Field (DIF) generation, with or without simultaneously moving data.
- **Networking**: For data copy in packet processing pipelines. An example usage is virtual switch offload for inter-VM packet switching.
- **Deduplication**: For comparing memory pages for equality to support memory deduplication.
- **VM Migration and Fast Checkpointing**: VM fast checkpointing and VM migration flows require the VMM to identify a VM's modified pages and send them efficiently to the destination machine, with minimal network traffic and latency. Intel DSA delta operations generate diffs of pages, enabling the VMM to send only the modified data to the destination, reducing network traffic.
- **Data movement between peer devices**: May be used for data movement between a peer accelerator device and host memory or between two peer devices to free up CPU cycles from such infrastructure work.

- **Data Movement to/from/between virtual machines**: To free up CPU cores from performing routine infrastructure tasks including moving data between virtual machines, containers, and bare metal hosts.

2.2 Intel® DSA Features

Intel DSA features include 1) infrastructure features, which are basic features to help with programmability, performance, and efficiency; 2) data operations, which are the actual data DMA and other transformation operations; and 3) control operations. The following sections give an overview of these features.

2.2.1 Infrastructure Features

The following infrastructure features are supported by Intel DSA.

- **Shared Virtual Memory (SVM):** SVM allows user level applications to submit commands to the device directly, with virtual addresses in the descriptors. It supports translating virtual addresses to physical addresses using IOMMU including handling page faults. The virtual address ranges referenced by a descriptor may span multiple pages. Intel DSA also supports the use of physical addresses, as long as each data buffer specified in the descriptor is contiguous in physical memory.
- **Partial descriptor completion:** With SVM, an operation may encounter a page fault during address translation. Software can control whether the device is to continue processing after waiting for resolution of a page fault or terminate processing of a descriptor that encounters a page fault and proceed to the next descriptor. If processing of a descriptor is terminated, the completion record indicates to software the amount of work completed and information about the page fault so that software can resolve the fault and restart the operation from the point where it stopped.
- **Block on fault:** As an alternative to partial descriptor completion, when the device encounters a page fault it can coordinate with system software to resolve the fault and continue the operation transparently to the software that submitted the descriptor.
- **Batch processing:** A Batch descriptor points to an array of work descriptors (i.e., descriptors with actual data operations). When processing a batch descriptor, the device fetches the work descriptors from the specified virtual memory address and processes them.
- **Stateless device:** Descriptors are designed so that all information required for processing the descriptor comes in the descriptor itself. This allows the device to store little client specific state, which improves its scalability. The only exception is the completion interrupt message, when used, because it must be configured by trusted software.
- **Cache allocation control:** This allows applications to specify whether output data is allocated in the cache or is sent to memory without cache allocation. Completion records are always allocated in the cache.
- **Shared Work Queue (SWQ) support:** Shared Work Queues (SWQ) enable scalable work submission using Deferrable Memory Write transactions, which indicate whether the work was accepted into the WQ.
- **Dedicated Work Queue (DWQ) support:** Dedicated Work Queues (DWQ) enable high-throughput work submission using 64-byte Memory Write transactions.

- **QoS support:** Intel DSA supports several features that allow the kernel driver to separately control access to device resources by different guests and applications.
- **Intel® Scalable IOV support:** Intel Scalable IO Virtualization improves scalability of device assignment, allowing a VMM to share the device across many more VMs than would be possible using SR-IOV.
- **Persistent Memory features:** Configuration registers and descriptor flags allow software to indicate writes to durable memory (such as Intel® Optane™ DC persistent memory) and specify the durability and ordering semantics to the SoC.
- **Inter-domain support:** Intel DSA supports features that allow a single descriptor to perform certain data operations spanning different address space domains.

2.2.2 Data Operations

The following data operations are supported by Intel DSA. See chapter 8 for details on these operations.

Operation	Type	Description
Move	Memory Move	Transfer data from a source address to destination address. Source and destination ranges can be either in main memory or MMIO.
	CRC Generation	Generate CRC checksum on the transferred data.
	DIF/DIX	Data Integrity Field (DIF) check. DIF insert, strip, or update while transferring data. Compute DIF for each block of source data and write to the destination address.
	Dualcast	Copy data simultaneously to two destination locations.
Fill	Memory Fill	Fill memory range with a fixed pattern.
Compare	Memory Compare	Compare two source buffers and return whether the buffers are identical.
	Delta Record Create	Create a delta record containing the differences between the original and modified buffers. The size of the delta record is bounded, and the device signals an overflow if the differences exceed the bound.
	Delta Record Merge	Merge a delta record with the original source buffer to produce a copy of the modified buffer at the destination location.
	Pattern/Zero Detect	Special case of compare where instead of the second input buffer, an 8-byte pattern is specified. Pattern may be zero.
Flush	Cache Flush	Evict all lines in a given address range from all levels of CPU caches.
Inter-Domain	Data Operations	Data operations across address domains initiated by privileged or unprivileged software.
	Update Window	Control parameters of memory windows used by inter-domain operations.

Table 2-1: Intel® DSA Data Operations

2.2.3 Control Operations

The following control operations are supported by Intel DSA. Some of these commands are issued using descriptors and some are issued using the Command register. See sections 9.2.12 and 8.3 for details.

Operation	Type	Description
Enable / Disable / Reset	Device	Manage the device as a whole.
	WQ	Manage individual WQs.
Drain	Current client	Drain all in-flight work requests from the current client.
Drain / Abort	Specified client	Drain or abort in-flight work requests from the specified client.
	Work Queue	Drain or abort in-flight work requests in specified work queue.
	All	Drain all in-flight work requests in the device.
No-op	Null operation	Performs no operation but can signal completion.

Table 2-2: Intel® DSA Control Operations

§

3 Intel® Data Streaming Accelerator Architecture

This chapter describes the Intel DSA architecture in detail. Each SoC may support any number of Intel DSA device instances. A multi-socket server platform may support multiple such SoCs. From a software perspective, each instance is exposed as a single Root Complex Integrated Endpoint. Each instance is under the scope of a DMA Remapping hardware unit (also called an IOMMU). Each Intel DSA instance is behind a single DMA Remapping hardware unit, but depending on the SoC design, different device instances can be behind the same or different DMA Remapping hardware units.

Intel DSA supports an Address Translation Cache (ATC) and interacts with DMA Remapping hardware using the PCI-SIG-defined Address Translation Services (ATS), Process Address Space ID (PASID), and Page Request Services (PRS) capabilities. The PASID TLP prefix is added to upstream requests to support both Shared Virtual Memory (SVM) and Intel Scalable I/O Virtualization (Intel Scalable IOV). The device utilizes the DMA Remapping hardware to translate DMA addresses to host physical addresses. Depending on the usage, a DMA address can be a Host Virtual Address (HVA), Guest Virtual Address (GVA), Guest Physical Address (GPA), or I/O Virtual Address (IOVA). Intel DSA supports additional PCI Express capabilities, including Advanced Error Reporting (AER) and MSI-X.

The Intel DSA architecture is designed to support Intel Scalable I/O Virtualization. The device can be shared directly with multiple VMs in a secure and isolated manner to achieve high throughput. Sections 3.16 and 7.3 describe the virtualization features in more detail.

Figure 3-1 illustrates the high-level blocks within the Intel DSA device at a conceptual level. Downstream work requests from clients are received on the I/O fabric interface. Upstream read, write, and address translation operations are sent on that interface. The device includes configuration registers, Work Queues (WQ) to hold descriptors submitted by software, arbiters used to implement QoS and fairness policies, processing engines, an address translation and caching interface, and a memory read/write interface. The batch processing unit processes Batch descriptors by reading the array of descriptors from memory. The work descriptor processing unit has stages to read memory, perform the requested operation on the data, generate output data, and write output data, completion records, and interrupt messages.

The WQ configuration allows software to configure each WQ either as a Shared Work Queue (SWQ) that can be shared by multiple software components, or as a Dedicated Work Queue (DWQ) that is assigned to a single software component at a time. The configuration also allows software to control which WQs feed into which engines and the relative priorities of the WQs feeding each engine.

3.1 Register and Software Programming Interface

Intel DSA is software compatible with the standard PCI Express configuration mechanism and implements a PCI header and extended space in its configuration-mapped register set.

Memory-mapped I/O registers provide status and control of device operation. Capability, configuration, and work submission registers (portals) are accessible through the MMIO regions defined by the BAR0 and BAR2 registers described in section 9.1.1. Each portal is on a separate 4K page so that they may be independently mapped into different address spaces (clients) using CPU page tables.

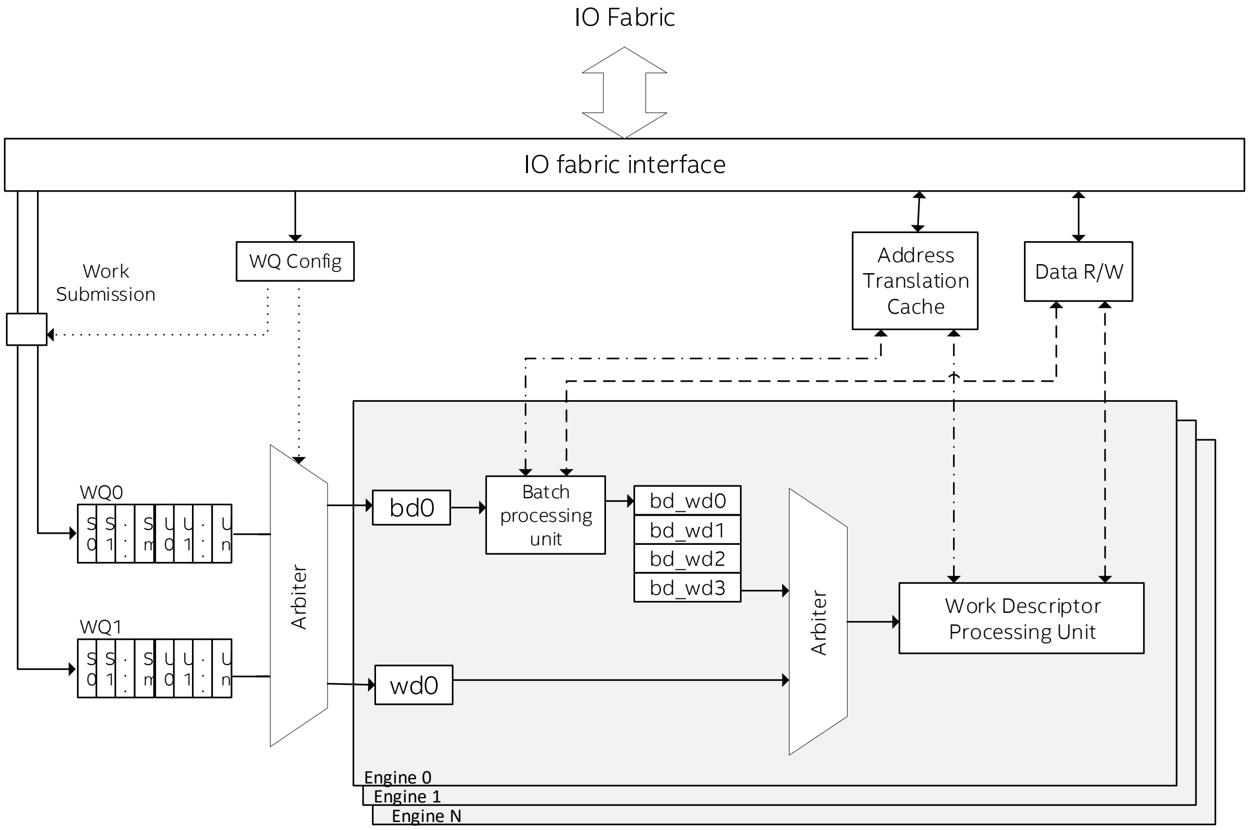


Figure 3-1: Abstracted Internal Block Diagram of Intel® DSA

3.2 Descriptors

Software specifies work for the device using descriptors. Descriptors specify the type of operation for the device to perform, addresses of data and status buffers, immediate operands, completion attributes, etc. See chapter 8 for descriptor formats and details. The completion attributes specify the address to write the completion record, and optionally, the information needed to generate a completion interrupt.

Intel DSA avoids maintaining client specific state on the device. All information to process a descriptor comes in the descriptor itself. This improves shareability of the device among user-mode applications, as well as among different virtual machines or machine containers in a virtualized system.

A descriptor may contain an operation and associated parameters (called a Work descriptor), or it can contain the address of an array of work descriptors (called a Batch descriptor). Software prepares the descriptor in memory and submits the descriptor to a Work Queue (WQ) of the device. The device dispatches descriptors from the work queues to the engines for processing. When an engine completes a descriptor or encounters certain faults or errors that result in an abort, it notifies the host software by either writing to a completion record in host memory, issuing an interrupt, or both.

3.3 Work Queues

Work queues (WQs) are on-device storage to contain descriptors that have been submitted to the device. The WQ Capability register indicates the number of work queues and the amount of work queue storage available on the device. Software configures how many work queues are enabled and divides the available WQ space among the active WQs.

The WQ Configuration Table is used to configure the WQs. Prior to enabling the device, software configures the size of each WQ. Unused WQs have a size of 0. Other parameters of each WQ can be configured later, prior to enabling the WQ. In some configurations, the WQ size and other aspects of the WQ configuration are read-only. See section 9.2.24 for more details on configuring WQs.

Each work queue can be configured to run in one of two modes, Dedicated or Shared. The WQ Capability register indicates support for Dedicated and Shared modes. Controls in the WQ Configuration Table allow software to configure the mode of each WQ. The mode of a WQ can only be changed while the WQ is Disabled. See the specifications for the WQ Capability register, the WQ Configuration Table, and the Command register in section 9.2 for details on configuring and enabling Work Queues.

Descriptors are submitted to work queues via special registers called portals. Each portal is in a separate 4 KB page in device MMIO space. There are four portals per WQ:

- Unlimited MSI-X Portal
- Limited MSI-X Portal
- Unlimited IMS Portal
- Limited IMS Portal

The address of the portal used to submit a descriptor allows the device to determine which WQ to place the descriptor in, whether the portal is limited or unlimited, and which interrupt table to use for the completion interrupt.

See section 3.3.1, “Shared Work Queue,” for the usage of limited and unlimited portals. For Dedicated WQs, there is no difference between the limited and unlimited portals.

See section 3.7, “Interrupts,” for the usage of MSI-X and IMS portals. For a descriptor that does not request an interrupt, it doesn’t matter whether it is submitted to an MSI-X portal or an IMS portal. The IMS portals do not exist if IMS is not supported, so a descriptor written to an address that would normally correspond to an IMS portal is discarded without reporting an error. If the descriptor was submitted with DMWr, a Retry response is returned.

3.3.1 Shared Work Queue (SWQ)

A Shared Work Queue accepts work submission using the PCIe-defined Deferrable Memory Write Request (DMWr). DMWr is a 64-byte non-posted write that waits for a response from the device before completing. The device returns Success if the descriptor is accepted into the work queue, or Retry if the descriptor is not accepted due to WQ capacity or QoS. This allows multiple clients to directly and simultaneously submit descriptors to the same work queue. Since the device provides this feedback, the clients can tell whether their descriptors were accepted. On Intel CPUs, DMWr is generated using the ENQCMD or ENQCMDS instructions. The ENQCMD and ENQCMDS instructions return the status of the command submission in EFLAGS.ZF flag; 0 indicates Success, and 1 indicates Retry.

A Shared WQ can be configured to reserve some of the WQ capacity by setting the WQ Threshold field in the WQCFG register. Work submission via a limited portal is accepted until the number of descriptors in the SWQ reaches the configured threshold. Work submission via an unlimited portal is accepted unless the SWQ is completely full. The unlimited portals are intended to be used only by privileged software when a work submission to the corresponding limited portal returns Retry. User-mode and guest software typically only have access to limited portals.

If DMWr returns Success, the descriptor has been accepted by the device and queued for processing. If DMWr returns Retry, software can try re-submitting the descriptor to the SWQ, or if it was a user-mode client using a limited portal, it can request that the kernel-mode driver submit the descriptor on its behalf using an unlimited portal. This helps avoid denial of service and provide forward progress guarantees. See chapter 7 for more information on software use of the limited and unlimited portals.

Clients are identified by the device using a 20-bit ID called Process Address Space ID (PASID). The PASID capability must be enabled to use SWQs. The PASID is used by the device to look up addresses in the Address Translation Cache and to send address translation or page requests to the IOMMU. In Shared mode, the PASID to be used with each descriptor is contained in the PASID field of every descriptor. The ENQCMD instruction copies the PASID of the current thread from the IA32_PASID MSR into the descriptor while ENQCMDS allows supervisor mode software to copy the PASID into the descriptor. For additional details on the use of PASID and the ENQCMD and ENQCMDS instructions, refer to the Intel® Architecture Instruction Set Extensions Programming Reference, listed in the References in section 1.2.

3.3.2 Dedicated Work Queue (DWQ)

To submit work to a Dedicated Work Queue, software uses a 64-byte memory write transaction with write atomicity. This transaction may complete faster than DMWr due to the posted nature of the write operation. The device depends on software to provide flow control based on the number of slots in the work queue. Software is responsible for tracking the number of descriptors submitted and completed, to detect a work queue full condition. If software erroneously submits a descriptor to a dedicated WQ when there is no space in the work queue, the descriptor is dropped. (The error is reported in the Software Error Register.)

On Intel CPUs, work submission to a DWQ is performed using the MOVDIR64B instruction, which generates a non-torn 64-byte write. For information about the MOVDIR64B instruction, refer to the Intel® 64 and IA-32 Architectures Software Developer's Manuals, listed in the References in section 1.2.

With dedicated WQs, the use of PASID is optional. If the PCI Express PASID capability is not enabled, PASID is not used. If the PASID capability is enabled, the WQ PASID Enable field of the WQ Configuration register controls whether PASID is used for each DWQ. Since the MOVDIR64B instruction does not fill in the PASID as the ENQCMD or ENQCMDS instructions do, the PASID field in the descriptor is ignored. When PASID is enabled for a DWQ, the device uses the WQ PASID field of the WQ Configuration register to do address translation. The WQ PASID field must be set by the driver before enabling a work queue in dedicated mode.

Although dedicated mode doesn't support the sharing of a single DWQ by multiple clients, Intel DSA can be configured to have multiple DWQs and each of the DWQs can be independently assigned to clients. DWQs can be configured to have the same or different QoS levels.

3.4 Engines and Groups

An engine is an operational unit within an Intel DSA device. A group is a set of work queues and engines. Software configures WQs and engines into groups using the Group Configuration registers. Each group contains one or more WQs and one or more engines. Any engine in a group may be used to process a descriptor posted to any WQ in the group. Each WQ and each engine may be in only one group.

Although the Intel DSA architecture allows great flexibility in configuring work queues, groups, and engines, the hardware is designed with the intent to be used in specific configurations. Example configurations are shown in Figures 3-2 and 3-3. In the configuration shown in Figure 3-2, hardware uses either engine in a group to process descriptors from any work queue in the group. If one engine has a stall due to a high-latency memory address translation or page fault, the other engine can continue to operate and maximize the throughput of the overall device.

Figure 3-2 shows example Traffic Class (TC) values for the two groups. In Group 0 both TC values are 0, while in Group 1, TC-B is 1. This example configuration might be used when Group 0 is used solely for operations that access DRAM, and Group 1 is used for operations that access both DRAM and persistent memory. The TC Selector flags in descriptors submitted to Group 1 indicate whether each address in the descriptor refers to DRAM or persistent memory. See chapter 4 for information on Traffic Classes and how they can be used to control QoS.

Figure 3-2 shows two work queues in each group, but there may be any number up to the maximum number of WQs supported. The WQs in a group may be shared WQs with different priorities, or one shared WQ and the others dedicated WQs, or multiple dedicated WQs with the same or different priorities.

Figure 3-3 shows another example configuration, in which each engine is placed in a separate group. Software may choose this configuration when it wants to reduce the likelihood that latency-sensitive operations become blocked behind other operations. In this configuration, software submits latency-sensitive operations to the work queue connected to one engine, and other operations to the work queues connected to another engine. If the group used for latency sensitive operations is idle when a descriptor is submitted, the descriptor will be dispatched to an engine immediately.

Software can also mix these two, with some engines in a single group and the others in groups by themselves.

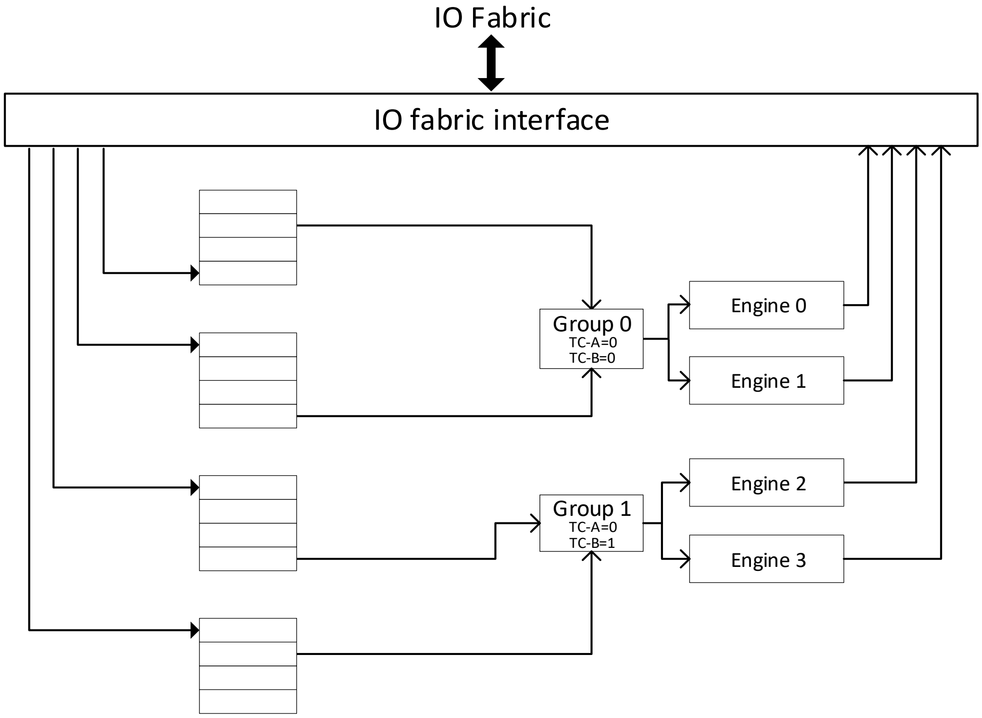


Figure 3-2: Sample Group Configuration 1

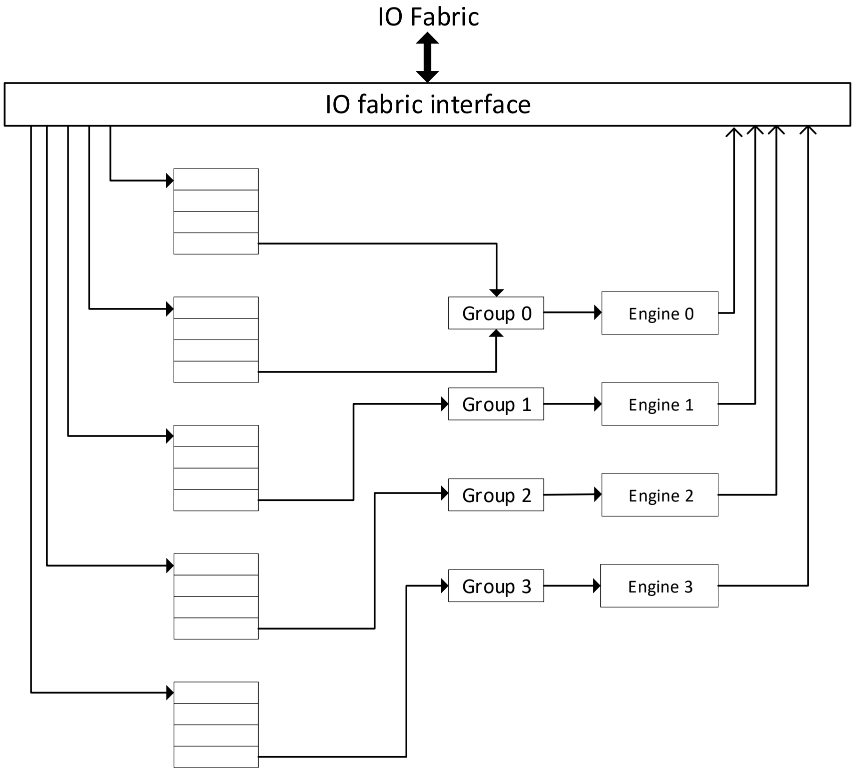


Figure 3-3: Sample Group Configuration 2

3.5 Descriptor Processing

As each descriptor reaches the head of the work queue, it is available to be dispatched by the group arbiter to an available engine in the group. The arbiter for each group dispatches descriptors from the WQs in the group according to their priority, while ensuring that the higher priority WQs don't starve lower priority WQs. See section 4.1 for information about work dispatch priority.

For a Batch descriptor, which refers to work descriptors in memory, the engine fetches the array of work descriptors from memory. Each work descriptor is passed to the work descriptor processing unit. The work descriptor processing unit uses the Address Translation Cache and IOMMU for completion record, source, and destination address translations; reads source data; performs the specified operation; and writes the destination data back to memory. When the operation is complete, the engine writes the completion record to the pre-translated completion address and generates an interrupt, if requested by the work descriptor.

3.6 Descriptor Completion

Descriptors contain three flags and two other fields that allow software to control completion notifications. The three flags are: Completion Record Address Valid, Request Completion Record, and Request Completion Interrupt. The two fields are Completion Record Address and Completion Interrupt Handle.

The completion record is a 32-byte aligned structure in memory that the device writes when the operation is complete or encounters an error. The completion record contains completion status. If the operation completed successfully, the completion record may contain the result of the operation, if any, depending on the type of operation. If the operation did not complete successfully, the completion record contains fault or error information.

Generally, all descriptors should have a valid Completion Record Address and the Completion Record Address Valid flag should be 1. (Exceptions to this rule are described later.)

The first byte of the completion record is the status byte. Status values written by the device are all non-zero. Software should initialize the status field of the completion record to 0 before submitting the descriptor to be able to tell when the device has written to the completion record. (Initializing the completion record also ensures that it is mapped, so the device is less likely to encounter a page fault when accessing it.)

The Request Completion Record flag indicates to the device that it should write the completion record even if the operation completed successfully. If this flag is not set, the device writes the completion record only if there is an error.

Descriptor completion can be detected by software using any of the following methods:

1. Poll the completion record, waiting for the status field to become non-zero.
2. Use the UMONITOR/UMWAIT instructions on the completion record address to block until it is written or until timeout. Software should then check whether the status field is non-zero to determine whether the operation has completed.
3. Request an interrupt when the operation is completed. For user-mode descriptors, this method requires the kernel to forward the notification to the application.
4. If the descriptor is in a batch, set the Fence flag in a subsequent descriptor in the same batch. Completion of the descriptor with the Fence or any subsequent descriptor in the same batch indicates completion of all descriptors that precede the Fence.
5. If the descriptor is in a batch, completion of the Batch descriptor that initiated the batch indicates completion of all descriptors in the batch.
6. Issue a Drain descriptor or a Drain command and wait for it to complete.

If the completion status indicates a partial completion due to a page fault, the completion record indicates how much processing was completed (if any) before the fault was encountered, and the virtual address where the fault was encountered. Software may choose to fix the fault (by touching the faulting address from the CPU) and resubmit the rest of the work in a new descriptor or complete the rest of the work in software. Faults on descriptor list and completion record addresses are handled differently and are described in more detail in section 3.13.

3.7 Interrupts

Intel DSA supports only message signaled interrupts. It provides two types of interrupt message storage: (1) an MSI-X table, enumerated through the MSI-X capability; and (2) a device-specific Interrupt Message Storage (IMS) table, as described by the Intel Scalable IOV Architecture Specification. For more information on IMS, refer to section 9.2.28, and to the Intel® Scalable I/O Virtualization Technical Specification, listed in the References in section 1.2.

Interrupts can be generated for six types of events: 1) completion of a descriptor; 2) WQ occupancy below programmed limit; 3) completion of an administrative command; 4) an error posted in the Software Error

Register or written to the Event Log in memory[1]; 5) performance monitoring counter overflow; and 6) interrupt handle revocation. For each type of event, there is a separate interrupt enable. Interrupts for types 3–6 are generated using entry 0 in the MSI-X table. The Interrupt Cause Register may be read by software to determine the reason for the interrupt.

For completion of a descriptor that requests a completion interrupt, the interrupt message used is dependent on the portal the descriptor was submitted to and the Completion Interrupt Handle in the descriptor. As described in section 3.3, each WQ has both MSI-X portals and IMS portals. For a descriptor submitted via an MSI-X portal, the Completion Interrupt Handle field in the descriptor selects an entry in the MSI-X table. For a descriptor submitted via an IMS portal, the Completion Interrupt Handle field in the descriptor selects an entry in the Interrupt Message Storage. Descriptors in a batch are treated as if they had been submitted via the same portal as the Batch descriptor.

Event	Submission Register	Interrupt Message Used
Error posted in SWERROR register or written to the Event Log	N/A	MSI-X table entry 0.
Completion of an administrative command	Command register	MSI-X table entry 0.
Perfmon counter overflow	N/A	MSI-X table entry 0.
WQ Occupancy below limit	WQ Occupancy Interrupt register	MSI-X or IMS entry programmed in WQ Occupancy Interrupt register.
Descriptor completion	MSI-X portal	MSI-X table entry specified by Completion Interrupt Handle field in descriptor.
	IMS portal	Interrupt Message Storage entry specified by Completion Interrupt Handle field in descriptor.
Interrupt handle revocation	N/A	MSI-X table entry 0.

Table 3-1: Interrupt Delivery

When the Request Interrupt Handle command is not supported (as indicated by the Command Capabilities register), the Completion Interrupt Handle is the index of the desired entry in the MSI-X table or the IMS. When the Request Interrupt Handle command is supported, software must use the command to obtain a handle to use for the interrupt. Software specifies in the Request Interrupt Handle command which interrupt table entry it wants a handle for. The response to the command contains the handle that software should place in the Completion Interrupt Handle field of the descriptor to request that interrupt.

An interrupt handle obtained using the Request Interrupt Handle command may be revoked. After an interrupt handle is revoked, any use of the handle will result in an Invalid Interrupt Handle error. When one or more interrupt handles are revoked, the device sets the Interrupt Handles Revoked bit in the Interrupt Cause register and generates an interrupt using MSI-X table entry 0. This interrupt cause can only occur if the Request Interrupt Handle command has been used to obtain interrupt handles. Software should use the Request Interrupt Handle command to obtain new handles for all MSI-X and/or IMS entries in use. Software should then resubmit any descriptors that failed with an Invalid Interrupt Handle error using the

[1] Any subsequent references to the SWERROR register should be treated as applying to either the SWERROR register or to the Event Log while it is enabled.

new handles. See section 7.5.4 for a description of interrupt virtualization including details of the steps software should perform to support interrupt handle revocation.

The MSI-X table defined by the PCIe specification is augmented in Intel DSA by the MSI-X Permissions Table, detailed in section 9.2.22. Each MSI-X Permissions Table entry has several fields that control generation of interrupts using that table entry. Each IMS entry contains the same control fields. The PASID Enable and PASID fields of the selected interrupt table entry are checked before the descriptor is executed, as detailed in section 5.4. The Ignore and Mask fields are checked when the descriptor completes. If the Ignore field is 1, no interrupt is generated. If the Ignore field is 0, the Mask and Pending fields behave as specified by PCIe. If the Mask field is 1, the Pending field is set to 1 and no interrupt is generated. If Ignore and Mask are both 0, the interrupt is generated. For interrupts other than descriptor completions, the PASID Enable, PASID, and Ignore fields are not used.

Interrupts generated by Intel DSA are processed through the Interrupt Remapping and Posting hardware as configured by the kernel or VMM software.

3.8 Batch Descriptor Processing

Intel DSA supports submitting multiple descriptors at once. A Batch descriptor contains the address of an array of work descriptors in host memory and the number of elements in the array. The array of work descriptors is called the “batch.” Use of Batch descriptors allows software to submit multiple work descriptors using a single work submission operation and can potentially improve overall throughput, especially when using descriptors with small transfer sizes.

Intel DSA enforces a limit on the number of work descriptors in a batch. There is an overall limit, indicated by the Maximum Supported Batch Size field in the General Capabilities register, and also a separate limit for each work queue, set by the WQ Maximum Batch Size field for each WQ in the WQ Configuration Table. A batch must contain at least 2 work descriptors.

Batch descriptors are submitted to work queues in the same way as other work descriptors. When a Batch descriptor is processed by the device, the device reads the array of work descriptors from memory and then processes each of the work descriptors. The work descriptors are not necessarily processed in order. (See section 3.9 for information on how software can control ordering of descriptors in a batch.)

The PASID and the Priv fields of a Batch descriptor are used for all descriptors in the batch.[1] The PASID and Priv fields in the descriptors in a batch are ignored.

Each work descriptor in a batch can specify a completion record address and/or a completion interrupt, just as with directly submitted work descriptors. The completion record and completion interrupt for the Batch descriptor (if requested) are generated after completion of all the descriptors in the batch and generation of their completion records (if requested). No readback is performed before the Batch descriptor completion record is generated. To maintain ordering of the completion record for the Batch behind all writes from descriptors in the batch, either the Batch descriptor should use the same TC for its completion record as the prior writes, or the Destination Readback flag should be set in each of the descriptors in the

[1] For a Batch descriptor submitted to a dedicated work queue, the PASID and Priv fields of the Batch descriptor and all the work descriptors in the batch come from the WQ Configuration register.

batch. To maintain ordering of the completion record for the Batch after the completion records of the descriptors in the batch, the same TC should be used for all of the completion records.

The completion record for the Batch descriptor contains an indication of whether any of the descriptors in the batch completed with Status not equal to Success. This allows software to avoid examining all the completion records for the descriptors in the batch, in the usual case where all the descriptors in the batch completed successfully. In some cases, if a descriptor in the batch encountered a page fault on the completion record address, hardware may indicate a possible error even though the completion record page fault was resolved successfully. Software can examine the completion records for descriptors in the batch to determine whether there were truly any failures.

A Batch descriptor may not be included in a batch. Nested or chained descriptor arrays are not supported. See section 8.3.2 for details on the format of Batch descriptors.

3.9 Ordering and Fencing

Descriptors may generally be processed by the device in any order. However, descriptors are guaranteed to be executed in the order that they are received by the device when all of the following conditions are met:

- Descriptors are submitted to a group with only one engine.
- Descriptors are all submitted to the same WQ using the same portal address.
- Descriptors are all Batch descriptors, or they are all not Batch descriptors.
- Descriptors all use the same Destination TC Selector.

Only write ordering is guaranteed. Reads by a subsequent descriptor can pass writes from a previous descriptor. If an error occurs in a descriptor, subsequent descriptors will continue to execute. Thus, software cannot necessarily rely on data transfers from earlier descriptors completing before those from later descriptors. The order in which completion records become visible to software is not guaranteed.

Even when these conditions are met, the order of descriptors within a batch is not guaranteed unless the Fence flag is set as described below.

If more control of the ordering of descriptors is required, software may use one of the following methods:

- Submit a descriptor, wait for the completion record or interrupt from the descriptor to ensure completion, and then submit the next descriptor.
- Use a Drain descriptor or Drain command to wait for preceding descriptors to complete, and then submit the following descriptors.
- Within a batch, use the Fence flag.

Enforcing ordering may increase both the CPU time used to submit a descriptor and the latency for the descriptor to begin execution within the device.

To control ordering for descriptors in a batch specified by a Batch descriptor, each work descriptor has a Fence flag. When set, Fence guarantees that processing of that descriptor will not start until all previous descriptors in the same batch are completed. This allows a descriptor with Fence to consume data produced by a previous descriptor in same batch. A descriptor consuming data from a previous descriptor in the batch should use the same Traffic Class as the descriptor producing the data. If software cannot ensure this, then software must set the Destination Readback flag in the descriptor that produces the data to ensure the required ordering.

If any descriptor in a batch completes with Status not equal to Success, for example if it is partially completed due to a page fault, a subsequent descriptor with the Fence flag equal to 1 and any following descriptors in the batch are abandoned. The completion record for the Batch descriptor that was used to submit the batch indicates how many descriptors were processed prior to the Fence.

The completion record write for a descriptor is ordered after all data writes produced by the descriptor if:

- the descriptor is fully completed; or
- the completion record TC Selector in the descriptor is the same as the destination TC Selector(s).

Otherwise, the completion record may be observed by software before some of the data writes produced by the descriptor. A completion interrupt (if requested) is ordered after the completion record write.

If a Batch descriptor does not request a completion record (e.g., Completion Record Address Valid is 0 or Request Completion Record is 0), the ordering of the completion interrupt for the Batch descriptor (if requested) relative to completion record writes for descriptors in the batch is not guaranteed if the completion record writes specify a TC Selector that selects a non-zero TC. In this case, software can set the Request Completion Record flag in the Batch descriptor to ensure correct ordering of the completion interrupt for the Batch descriptor.

The Destination Readback flag causes Intel DSA to perform a zero-length read, using the final destination address of the descriptor, prior to writing the completion record. If the destination target is different from the completion record target, then the Destination Readback flag may be set to ensure that writes have propagated to the destination before the completion record is written. For example, this flag may be used in descriptors that target NTB to ensure that data written by the descriptor has propagated across the NTB link. Destination readback is performed only if the descriptor is completed successfully. If the descriptor is partially completed, the readback is not performed. If a follow-up descriptor to complete the operation writes to the same destination using the same TC, sets the Destination Readback flag, and completes successfully, then the readback performed by the follow-up descriptor also ensures completion of memory writes performed by the prior descriptor(s).

3.10 Persistent Memory Support

Intel DSA provides several features intended to improve its utility with persistent memory such as Intel® Optane™ DC persistent memory. When the Destination Address of a descriptor is in persistent memory, software should use the following flags to ensure that data has become persistent at the time the descriptor completes.

- The Cache Control flag in the descriptor controls whether writes are directed to cache or to memory. For use with persistent memory, the Cache Control flag should be 0.
- The Destination Readback flag should be 1 to ensure that data written by the descriptor has become persistent at the time the descriptor completes.

As described in section 2.2.1, since completion record writes are always directed to cache, there is no device-supported mechanism to ensure persistence of completion record writes.

The Strict Ordering flag does not guarantee that writes become persistent in order.

3.11 Cache Control

The Cache Control flag in the descriptor is a hint indicating whether destination addresses targeted by the descriptor should be written to the last level cache or to memory. If the flag is 0, it hints that data written by the descriptor be directed to memory. If a write operation targets a cache line that is present in the cache, it may be removed from the cache. If the flag is 1, it hints that cache entries be allocated to contain data written by the descriptor. The hint may be ignored by an implementation. Because processors are free to speculatively fetch data into the caches or evict data from the caches at any time, the effect of this flag is not guaranteed, even when it is supported.

The Cache Control flag is reserved for operations that do not write to memory, including Cache Flush and Inter-Domain Cache Flush. The Cache Control flag is reserved for all operation types if the Cache Control Support field in GENCAP is 0.

3.12 Drain Descriptor

A Drain descriptor waits for completion of certain preceding descriptors in the WQ that the Drain descriptor is submitted to. If a Drain descriptor is submitted to a dedicated WQ, it waits for completion of all descriptors in the WQ. If a Drain descriptor is submitted to a shared WQ, it waits for descriptors in the WQ that were submitted with the same PASID as the Drain descriptor. To wait for all descriptors with a particular PASID, software should submit a separate Drain descriptor to every WQ that the PASID was used with. To wait for all descriptors in a WQ regardless of PASID, software may use the Drain WQ command described in section 3.13.3.

A Drain descriptor may be used during normal shutdown by a process that has been using the device. It can be used like a Fence operation for the entire PASID. It can be used to request a single completion record and/or interrupt for the completion of multiple descriptors. A Drain descriptor may not be included in a batch. (A Fence flag may be used in a batch to wait for prior descriptors in the batch to complete.) Software should execute a fencing instruction such as SFENCE or MFENCE before submitting a Drain descriptor to ensure that the Drain descriptor is received by the device after the descriptors it is intended to drain.

For the purpose of Drain, a preceding descriptor is completed after all writes generated by the operation are globally observable; after destination readback, if requested; after the write to the completion record is globally observable, if needed; and after generation of the completion interrupt, if requested. To ensure this, prior to Drain descriptor completion, hardware normally issues an implicit readback for each supported Traffic Class using an address determined by hardware. The implicit readbacks ensure that all previous writes to the Root Complex have completed.

Software can control the default behavior by setting readback flags in the Drain descriptor that can be used to suppress the implicit readbacks and/or request explicit readbacks to software-controlled addresses. Previous writes to a peer device (i.e., non-Root Complex) may not be flushed by a Drain descriptor implicit readback but can be flushed using explicit readbacks. The Drain descriptor allows software to specify up to 2 explicit readback addresses in the descriptor. If specified, hardware will issue readbacks to each explicit readback address using the Traffic Class specified by the corresponding TC Selector flag in the descriptor. See section 8.3.3 for details of the Drain descriptor.

3.13 Address Translation

Intel DSA supports the use of either physical or virtual addresses. The use of virtual addresses that are shared with processes running on the CPU is called shared virtual memory (SVM). To support SVM, the device provides a PASID when performing address translations and it handles page faults that occur when no translation is present for an address. However, the device itself doesn't distinguish between virtual and physical addresses; this distinction is controlled by the programming of the IOMMU.

Intel DSA supports the PCI Express Address Translation Service (ATS) and Page Request Service (PRS) capabilities. ATS describes the device behavior during address translation. When a descriptor enters a descriptor processing unit, the device requests translations for the addresses in the descriptor. If there is a hit in the Address Translation Cache, the device uses the corresponding HPA. If there is a miss or permission fault, the device sends an address translation request to IOMMU for the translation. The IOMMU finds the translation by walking the appropriate page tables and returns an address translation response that contains the translated address and the effective permissions. The device stores the translation in the Address Translation Cache and uses the corresponding HPA for the operation. If IOMMU can't find the translation in the page tables, it returns an address translation response that indicates no translation is available. When the IOMMU response indicates no translation or indicates effective permissions that don't include the permission required by the operation, it is considered a page fault.

3.13.1 Work Queue Address Translation Controls

Some implementations support the ability to control ATS and PRS capabilities for each enabled work queue in the device. Support for work-queue granular ATS and PRS controls is indicated by the WQ ATS Support and WQ PRS Support fields in WQCAP. If supported, software can use the WQ ATS Disable and WQ PRS Disable controls in WQCFG to disable the corresponding capabilities for each work queue. The device operation for descriptors submitted to a work queue that has either one of the capabilities disabled is the same as when the corresponding PCI Express control is 0.

If ATS is disabled, either by the PCIe ATS Control register or by the WQ ATS Disable flag in WQCFG, Intel DSA does not receive any notification of address translation faults for memory write transactions. The completion record for an operation may indicate success even if a page fault occurred for a write transaction. If ATS is disabled, software should ensure that no memory addresses accessed by a descriptor can cause a page fault. Software may use the Destination Readback flag to detect some errors writing to the destination address.

3.13.2 Page Fault Handling with PRS Enabled

PRS is enabled when the PCI Express PRS Enable Control is 1 and either WQ PRS Support in WQCAP is 0 or WQ PRS Disable in WQCFG is 0. In addition, each descriptor has a Block On Fault flag that specifies whether PRS is enabled for the source and destination buffer addresses in the descriptor.

When a page fault occurs and PRS is enabled, the fault is reported as a PRS request to the IOMMU for servicing by the OS page fault handler. The IOMMU notifies the OS through an interrupt. The OS validates the address and upon successful checks, creates a mapping in the page table and returns a PRS response through the IOMMU. The descriptor encountering the fault is blocked until the PRS response is received. Other operations behind the descriptor with the fault may also be blocked. If the OS was not able to create a mapping, it returns an error response and the descriptor is completed with an error. The error reporting is the same as page fault reporting when PRS is disabled, described in the next section.

When Block On Fault is 0 and a page fault is encountered on a source or destination buffer address, the device stops the operation and writes the partial completion status along with the faulting address and progress information into the completion record. (See sections 8.1 and 8.2 for more details.) The Block On Fault Support field in the General Capabilities register (GENCAP) indicates device support for enabling PRS for source and destination buffers, and the Block On Fault Enable field for each WQ in the WQ Configuration Table allows the VMM or kernel driver to control which applications are allowed to enable PRS for source and destination buffers. These registers are described in section 9.1.4.

Device page faults are relatively expensive, higher than the cost of servicing CPU page faults. Hence, for best performance, it is desirable for software to minimize device page faults without incurring the overheads of pinning and unpinning.

Batch descriptor lists and source data buffers are typically produced by software right before submitting them to the device. Hence, these addresses are not likely to incur faults due to temporal locality. Completion records and destination data buffers, however, are more likely to incur faults if they are not touched by software before submitting to the device. Such faults can be minimized by software explicitly "write touching" these pages before submission.

3.13.3 Page Fault Handling with PRS Disabled

PRS is disabled for all address translations when the PCI Express PRS Enable Control is 0 or the WQ PRS Disable field in WQCFG is 1. PRS is disabled only for the source and destination buffer addresses in a descriptor when the Block on Fault flag in the descriptor is 0.

When a page fault occurs on a Completion Record Address, the error is reported in the Event Log, if enabled, or in the SWERROR register. Completion record faults reported in the event log are recoverable. The descriptor is completed and the event log entry contains the completion record address and the contents of the completion record. After handling the page fault, system software should copy the completion record to the completion record address. See Section 5.9 for information about the event log and section 7.3 for information about software handling of completion record faults reported in the event log. Completion record faults reported in SWERROR may not be recoverable: faults may be lost since only a single error can be logged; and the descriptor is not completed and must be restarted, since partial completion information is not recorded.

When a page fault occurs for an address in a descriptor other than a Completion Record Address, the device stops the operation and writes the partial completion status along with the faulting address and progress information into the completion record. See sections 8.1 and 8.2 for more details. When the client software receives a completion record indicating partial completion, it has the option to fix the fault on CPU (by touching the page, for example) and submit a new work descriptor with the remaining work. Alternatively, software can complete the remaining work on the CPU.

Operating with PRS disabled and Event Log enabled allows applications using Intel DSA to avoid having significant impact on other applications in the presence of page faults. Device page faults can be expensive even when PRS is disabled because they require software intervention to service the page fault and resubmit the work. It is desirable for software to minimize device page faults as described in the previous section.

3.14 Inter-Domain Operations

Every work descriptor submitted to Intel DSA is associated with a default address space, which corresponds to the address space of the work submitter. While the PASID capability is enabled, the default address space is explicitly specified, either by the PASID carried in the work descriptor submitted to a shared work queue, or by the PASID configured in the WQCFG register for a dedicated work queue (as described in section 9.2.24). Memory accesses and IOMMU requests are tagged with this PASID value. While the PASID capability is disabled, the default address space is implicitly specified to the IOMMU via the PCIe requester ID (bus, device, function) of the device. We refer to this default address space for a work submitter as the descriptor PASID, and this is the address space that descriptors from that submitter normally use for memory accesses and IOMMU requests from the device. While the PASID capability is enabled, certain operations allow a submitter to select an alternate address space for either the source addresses, destination addresses, or both source and destination addresses specified in a work descriptor. The alternate address space is usually that of a cooperating process[1]. Henceforth, we refer to this process as the owner of that alternate address space. The term domain is used to denote an address space, and we refer to the set of operations that allow selection of an alternate address space as inter-domain operations.

Inter-Domain operations require the PASID capability to be enabled. Support for inter-domain operations is indicated by the Inter-Domain Support field in the GENCAP register (described in section 9.2.2). When this field is 1, inter-domain capabilities are reported in the IDCAP register (described in section 9.2.18). The set of inter-domain operations supported by an implementation is reported in the OPCAP register and may be used only if the inter-domain capability is supported. Selection of PASIDs used in each operation is done using appropriate descriptor fields. Details of the inter-domain operations supported, along with a description of the corresponding descriptor fields, can be found in section 8.3.20.

If a work submitter does not explicitly select an alternate PASID for an address in a descriptor, the descriptor PASID is used for memory accesses and IOMMU requests pertaining to that address. If a descriptor selects an alternate PASID for an address, that PASID is used instead of the descriptor PASID, if the submitter has appropriate permissions to do so. When used in this manner, we refer to the alternate PASID as the access PASID for the corresponding address in a descriptor. The device uses the access PASID to perform memory accesses and IOMMU requests pertaining to that address. The descriptor PASID is always used to write the completion record, to generate interrupts, and to verify that a submitter has adequate permissions to the specified access PASID, as described below.

An inter-domain operation may involve two or three PASIDs depending on the use case. Some of the typical use cases are listed below.

1. Data read or write by one or more user-mode submitters from or to a memory region exported by a user-mode owner.
2. Data read or write by a kernel-mode submitter from or to a memory region of a user-mode process.
3. Data read or write by a kernel-mode submitter between memory regions of two distinct user-mode processes.
4. Data read or write by a kernel-mode submitter from or to a memory region of another kernel-mode process.

[1] The term process is used to refer to an OS process, a virtual-machine, container, or other similar software entity that could be associated with a PASID.

5. Data read or write by a privileged submitter between memory regions of two distinct guest OSes.
6. Any of the above executed within a guest OS.

Use case (1) above requires an owner to explicitly grant access to a portion of its memory space to one or more submitters. The memory region that an owner grants access to is referred to as a memory window. A memory window is only accessible using the owner's PASID as the access PASID. Use cases (2) to (6) involve privileged software accessing memory regions of other user-mode or kernel-mode processes within that OS domain.

3.14.1 Inter-Domain Permissions Table (IDPT)

If inter-domain operations are supported, Intel DSA implements an Inter-Domain Permissions Table to allow software to manage 1) the association between a descriptor PASID and an access PASID that a work submitter is allowed to access; 2) attributes of a memory region in an access PASID's memory space that a submitter is allowed to access; and 3) controls to manage the life cycle of such association. The Inter-Domain Permissions Table is managed by the host kernel-mode driver and may be configured to support uses for both kernel-mode and user-mode applications, in a host or guest OS.

Each entry in the IDPT contains the following:

1. An entry type as described below.
2. One or more submitter PASID values allowed to use that entry and a mechanism to validate them.
3. Depending on the entry type, an access PASID to be used for memory accesses.
4. Memory window address range and attributes.
5. Permissions and other control information.

Each Inter-Domain Permissions Table entry (IDPTE) may be configured in one of the following ways as indicated by the Type field described below and summarized in section 9.2.29):

- **Type 0–Single Access, Single Submitter (SASS)**: The IDPTE specifies a single access PASID and a single submitter PASID. For example, a process that wants to expose a memory window to a peer process may request the driver to set up a SASS entry with its own PASID as the access PASID and the PASID of its peer as the submitter PASID.
- **Type 1–Single Access, Multiple Submitter (SAMS):** The IDPTE specifies a single access PASID. The submitter PASID field in the entry is unused. Instead, the IDPTE points to a bitmap in memory that specifies the set of submitter PASIDs allowed to use the entry. A bit set to 1 in the bitmap indicates that the corresponding PASID is allowed to submit an inter-domain operation using the IDPTE. For example, a process that wants to allow multiple submitters to access a window in its address space requests a SAMS entry to be set up.

Type	Mnemonic	Description	Access PASID Obtained From	Submitter PASID Matched Against
00	SASS	Single Access, Single-submitter entry (1 access PASID, 1 submitter PASID)	IDPT entry	IDPT entry
01	SAMS	Single Access, Multi-submitter entry (1 access PASID, N submitter PASIDs)	IDPT entry	Bitmap

Figure 3-4: Inter-Domain Permissions Table Entry Types

A descriptor refers to an IDPTE entry using a handle in the descriptor. The Inter-Domain Permissions Table Size field in IDCAP specifies the maximum size of the IDPT table supported by the device. Software allocates IDPT entries with index less than this maximum size. In addition, if the Request IDPT Handle field in CMDCAP is 0, the handle is the index of the desired entry in the IDPT. If the Request IDPT Handle field in CMDCAP is 1, software must use the Request IDPT Handle command to obtain the handle to use corresponding to the allocated index in the IDPT. Software specifies in the Request IDPT Handle command the index of the PASID table entry it wants a handle for, and the response to the command contains the handle that software should place in the descriptor. See section 7.5.6 for a description of IDPT virtualization.

An inter-domain descriptor may contain more than one handle, depending on the type of operation. A separate handle may be specified for each distinct source and/or destination address in a descriptor. Each handle in a descriptor is used by hardware to look up the corresponding IDPTE to 1) validate access permissions for the submitter, 2) identify the access PASID and privilege to be used for memory access, 3) compute the effective memory address, and 4) verify that the access conforms to the memory window and permissions granted by the IDPTE.

An IDPTE may be referenced by:

- An inter-domain descriptor while the Usable bit in the IDPTE is 1. The hardware checks that the descriptor PASID matches a submitter PASID value in the specified IDPTE.
- An Update Window descriptor while the Allow Update bit in the IDPTE is 1. The hardware checks that the descriptor PASID matches the access PASID value in the specified IDPTE.

If the PASID values do not match, then memory accesses using that entry are disallowed for that descriptor, and the descriptor is completed with an error.

Details of the operations are described in section 8.3.20.

3.14.2 Submitter Bitmap

As mentioned above, a type 1 SAMS IDPTE points to a submitter bitmap in memory, with one bit for every possible PASID value. The bitmap is indexed by the PASID value to be checked against the bitmap. Access is allowed only if the bit corresponding to the checked PASID is 1 in the bitmap. For a SAMS IDPTE, hardware checks the descriptor PASID against the bitmap prior to allowing any memory access using the table entry. A type 1 SAMS entry specifies a 4KB aligned virtual or physical address, referred to as the Submitter Bitmap Address. Privileged software like the kernel-mode driver is responsible for setting up and maintaining the bitmap in memory. The maximum size of a submitter bitmap is 2^{20} bits, i.e., 128KB. Each IDPTE that requires a bitmap may point to a distinct submitter bitmap in memory. Software may also choose to share a submitter bitmap between multiple IDPTEs, if appropriate.

The IDBR (described in section 9.2.19) controls whether hardware should use a PASID value for submitter bitmap reads. If enabled, the IDBR specifies the PASID value and privilege to be used for bitmap reads. Although each submitter bitmap is mapped to a contiguous virtual address range in the corresponding PASID space, it may be mapped into discontiguous physical pages in system memory. Software is also not required to map the bitmap entirely into system memory at a given time; different bitmap pages may be mapped as needed. If a page of a bitmap is inaccessible, all bits on that page are treated as 0.[1] The IDBR also specifies the traffic class to be used for bitmap reads.

Bitmap reads may be done by hardware in an implementation-specific manner. An implementation may issue bitmap reads as either Translated or Untranslated accesses. Hardware may read a single byte or dword or a cache line or larger region of a bitmap, corresponding to the PASID to be checked. For example, for a PASID value of p to be checked against a bitmap, an implementation that uses cache line reads of the bitmap would read the cache line at (PPBA + ((p >> 3) & 0xFFFFFFC0)) and examine the bit corresponding to the PASID to be checked.

An implementation is allowed to cache portions of a bitmap in order to avoid repeated memory reads. This capability, if supported, is indicated by the Invalidate Submitter Bitmap Cache field in CMDCAP. If this capability is 1, software must issue the Invalidate Submitter Bitmap Cache command (described in section 9.2.12) after it modifies any portion of a bitmap in memory or modifies the mapping of any page of the bitmap. In the latter case, software must perform the bitmap invalidation after it performs any required invalidations normally associated with page mapping modifications.

3.14.3 Memory Window

A memory window is a region of memory in an owner's address space that it allows one or more submitters to access. It is defined by the window base address, window size, window mode, and access permissions fields in the IDPTE. The window attributes are initialized at the same time that an IDPTE is allocated by the kernel-mode driver to an owner or to a privileged submitter. The Window Enable field in an IDPTE controls whether a memory window is active for that IDPTE.

If Window Enable is 0, hardware does not perform an address range check when using that entry. A validated submitter is allowed to access any address in the address space, and the Window Mode, Window Base, and Window Size fields are reserved.

If Window Enable is 1, hardware checks that the memory region in a descriptor referencing the IDPTE falls within the memory window. The memory window must not wrap around the 2^{64} address boundary. The Window Mode field controls the interpretation of the address in a descriptor referencing the IDPTE.

Two window modes are supported:

- Address Mode: Hardware checks that the memory region in the descriptor that references the IDPTE lies within the window, i.e., between the window base address and the sum of window base address and window size.
- Offset Mode: The address of the memory region in the descriptor is treated as an offset from the window base address. The effective start of the memory region is computed as the sum of the window base address and the address in the descriptor referencing that IDPTE. The effective end of the memory region is the sum of the effective start address and region size. The effective start and end of the memory region must lie within the window.

[1] Depending on the IOMMU configuration, faults may be reported on pages that are not mapped. To avoid these faults, software can pin all memory pages corresponding to the bitmap.

An IDPTE specifies read and write permissions for memory accesses using that entry. If the requested permissions do not match the granted permissions, the access is denied.[1]

3.14.4 Memory Window Modification

For a SASS or SAMS IDPTE, if the Allow Update bit in the IDPTE is 1, the owner may modify the memory window attributes using an Update Window descriptor (section 8.3.20.8). Only the process whose PASID matches the access PASID in the IDPTE is allowed to issue the Update Window. If the descriptor PASID does not match the access PASID, the Update Window descriptor is completed with an error.

While Allow Update is 0 for an IDPTE, the entry may be modified by the kernel-mode driver using MMIO writes while the IDPTE is not usable. See section 9.2.29 for details on when different fields in an IDPTE may be modified.

An Update Window descriptor atomically changes only the values of Window Base, Window Size, Window Mode, Read and Write permissions, and the Window Enable field in the IDPTE. Since the update is done atomically by hardware, any inter-domain descriptor referencing the IDPTE at the same time, is guaranteed to see either the old value or the new value of the window attributes. After the atomic update is done, an Update Window descriptor also performs an implicit drain to flush out any in-flight descriptors that are still using pre-update window attributes of that IDPTE. This ensures that when an Update Window operation is completed, any descriptors referencing that IDPTE have also completed. As described in section 8.3.20.8, an Update Window descriptor also allows for the implicit drain to be suppressed, if necessary.

3.15 Administrative Commands

Administrative commands are submitted to the device by writing to the Command register. Administrative commands are used to enable and disable the device, enable and disable WQs, and drain and abort descriptors.

Only one command may be submitted at a time. Software must wait for a prior command to complete before submitting another command. To determine when a command has completed, software may poll the Command Status register or request an interrupt by setting the Request Completion Interrupt field to 1 when it issues the command.

If Command Capabilities Support in GENCAP is 1, the Command Capabilities register (described in section 9.2.14) indicates which of the administrative commands are supported.

See the description of the Command register in section 9.2.12 for details on how to submit these commands. See the description of the Command Capabilities register in section 9.2.14 for information about which administrative commands are supported.

Enable Device	Check the device configuration and enable the device. The device must be enabled before enabling any WQs.
Disable Device	Stop accepting descriptors to all WQs, wait for completion of all descriptors, disable all WQs, and disable the device.

[1] Even if an IDPTE grants read or write permissions, the memory access may still fail if it is disallowed by the IOMMU address translation.

Enable WQ	Check the WQ configuration and enable the WQ. Once the command has successfully completed, descriptors may be submitted to the WQ.
Disable WQ	Stop accepting descriptors to the specified WQs, wait for completion of all descriptors that had been queued to the WQs, and disable the WQs.
Drain All	Wait for all descriptors in all WQs and all engines that were submitted prior to the Drain All command. The device may start work on new descriptors while the command is waiting for prior descriptors to complete; thus, descriptors submitted after the command may be in progress at the time the command completes.
Abort All	Abandon and/or wait for all descriptors in all WQs and all engines that were submitted prior to the Abort All command. Software must ensure that no descriptors are submitted to any WQs after the command is submitted and before it completes; otherwise, the behavior is undefined.
Drain WQ Abort WQ	Wait for all descriptors submitted to the specified WQs. Software must ensure that no descriptors are submitted to any of the specified WQs after the command is submitted and before it completes; otherwise, the behavior is undefined. Abort WQ may abandon some or all descriptors in the WQ instead of completing them.
Drain PASID Abort PASID	Wait for all descriptors associated with the specified PASID in all WQs and all engines. When the command completes, there are no more descriptors for the PASID in the device. Software must ensure that no descriptors with the specified PASID are submitted to the device after the command is submitted and before it completes; otherwise, the behavior is undefined. Abort PASID may abandon some or all of the descriptors instead of completing them.
Reset Device	Stop accepting descriptors on all WQs, abort all descriptors in the device, wait for any operations in flight, disable all WQs, disable the device, and clear the entire device configuration to power-on values, except for the Command, Command Status, and Software Error registers; the MSI-X table; and the IMS. If the device is already disabled, only clear the device configuration. See Table 9-3 for the initial values of device registers.
Reset WQ	Stop accepting descriptors to the specified WQs, abort all descriptors in the WQs, wait for any operations in flight, and disable the WQs. Then reset the WQ configuration registers of the specified WQs to initial values, except the WQ Size fields that are not modified. For any of the specified WQs that are already disabled, only reset the WQ configuration registers. This command allows specification of multiple work queues. See Table 9-3 for the initial values of device registers.
Request Interrupt Handle	Request an interrupt handle that can be used in descriptors to request completion interrupts. If this command is supported (as indicated by the Command Capabilities register), software must use this command to obtain interrupt handles. The result of this command may be an error if no additional interrupt handles are available. See section 3.7 for more information.
Release Interrupt Handle	Release an interrupt handle previously returned by the Request Interrupt Handle command. This command may be used to free a handle that is no longer needed.

	The released handle may not be used to request interrupts once this command has been issued. If any previously submitted descriptors using the released handle have not yet completed, the behavior is undefined.
Request IDPT Handle	Request a handle that can be used in descriptors to reference the specified IDPT entry in the Inter-Domain Permissions Table. If the Request IDPT Handle field in CMDCAP is 1, software must use this command to obtain IDPT handles. If the Request IDPT Handle field in CMDCAP is 0, this command code is reserved, and software uses the index of the entry in the Inter-Domain Permissions Table as the handle.
Release IDPT Handle	Release an IDPT handle previously returned by the Request IDPT Handle command. Software may use this command to release a handle for an entry in the Inter-Domain Permissions Table that is no longer in use. When the Request IDPT Handle command is supported, hardware may revoke an IDPT handle at any time. Software should use this command to release the expired handle after acquiring a new handle for the IDPT entry. See section 7.5.6 for more information.
Invalidate Submitter Bitmap Cache	Invalidate all or a portion of the submitter bitmap cache. If the Invalidate Submitter Bitmap Cache field in CMDCAP is 1, software must issue this command after updating any part of a submitter bitmap or modifying the address mapping of any page of the bitmap. If the Invalidate Submitter Bitmap Cache field in CMDCAP is 0, this command code is reserved, and no bitmap cache invalidation is required.

Drain and Abort Commands

Upon completion of any command that waits for or abandons descriptors, hardware guarantees that no further address translations, memory reads, memory writes, or interrupts will be generated due to any of the affected descriptors. Depending on the implementation, any drain command may wait for completion of other descriptors in addition to the descriptors that it is required to wait for.

When any type of abort command is issued, the hardware may either abandon or complete any of the affected descriptors. Some descriptors may be completed while others are abandoned. If a descriptor is completed, all the associated memory accesses, completion record, and completion interrupt are performed. If a descriptor is abandoned, no completion record is written and no completion interrupt is generated for that descriptor, but some or all of the other memory accesses may occur. Since the abort and reset commands are not guaranteed to abandon operations that have already started, they are not effective to terminate operations that are taking longer than expected. The maximum size of operations may be limited using the WQ Maximum Transfer Size and WQ Maximum Batch Size configuration registers.

Software Usage of Drain and Abort Commands

When an application or VM that is using Intel DSA is suspended, it may have outstanding descriptors submitted to the device. This work must be completed so the client is in a coherent state that can be resumed later. The Drain PASID and Drain All commands are used by the OS or VMM to wait for any outstanding descriptors. The Drain PASID command is used for an application or a VM that was using a single PASID. The Drain All command is used for a VM using multiple PASIDs.

When an application that is using the device exits or is terminated by the OS, the OS needs to ensure that there are no outstanding descriptors before it can free up or re-use address space, allocated memory, and the PASID. To clear out any outstanding descriptors, the OS uses the Abort PASID command with the PASID of the client being killed. On receiving this command, the device discards all descriptors belonging to the specified PASID without further processing.

3.16 Virtualization

The Intel DSA architecture is designed to be easy and efficient to virtualize. Intel DSA supports the Intel Scalable I/O Virtualization model. For more details on the Intel Scalable IOV architecture, refer to the Intel® Scalable I/O Virtualization Technical Specification, listed in the References in section 1.2.

This section describes the Intel DSA features designed to support efficient virtualization. The design of software to use these features is described in section 7.3.

- **Directly accessible MMIO registers:** MMIO space lays out performance critical registers (i.e., portals) in separate 4K pages to allow direct mapping to VMs using CPU Extended Page Tables (EPT).
- **Minimize client specific state:** The architecture has been designed to store minimal client specific state on the device to increase scalability. For example, the descriptors have been designed so that the information required to process the descriptors is included in the descriptors themselves.
- **Capabilities:** Software reads capability registers to detect support for features such as block-on-fault. Through capability virtualization, the VMM can expose a subset of the device's capabilities to VMs, which helps in VM image deployment and VM migration across multiple generations of Intel DSA devices with different capabilities.
- **Intel Scalable IOV:** The Intel Scalable IO Virtualization architecture reduces virtualization complexity and allows the device to be shared across a large number of VMs.
- **Guest OS interrupts:** Intel DSA defines a command for a guest to request Completion Interrupt Handles to use in its descriptors to request completion interrupts. Each handle denotes an entry in the Interrupt Message Storage that has been configured as an interrupt for that guest. The device uses the IMS entry to send interrupts to the VM.

§

4 Quality of Service Control

4.1 Work Dispatch Priority

Intel DSA provides WQ priorities to control quality of service for dispatching work from multiple WQs in the same group. The priority of each WQ is specified in its WQ Configuration register, described in 9.2.24. WQ priority levels range from 1 to 15. The WQ priority is relative to other WQs in the same group. Work queues in a group may have the same or different priorities.

The arbiter for each group dispatches descriptors from the WQs in the group according to their priority using the following procedure: Each WQ has a counter that is initialized to the WQ's priority level and decremented each time a descriptor is dispatched from the WQ. The arbiter for each group iterates through the WQs in the group, dispatching one descriptor from each WQ that has a descriptor available and has a non-zero counter. Once the counter for a WQ reaches zero, no more descriptors are dispatched from that WQ until the counter is reset. Once all counters reach zero for all WQs in a group that have pending descriptors, the counters for all WQs in the group are reinitialized to the respective WQ's priority level.

Thus, for example, a WQ with a priority of 6 will issue 3 times as many descriptors to the engine as a WQ with a priority of 2 (assuming that both WQs have descriptors available at all times).

When software submits a descriptor to a WQ that was previously empty, the descriptor will be processed at that WQ's next turn, regardless of the WQ's priority level, if the WQ's counter is non-zero.

There is no delay caused by the arbiter checking empty WQs to see if they have descriptors available. Descriptors can be issued to the engines in the group at the rate the engines can process them, even if the only WQ(s) with descriptors available have low priority.

4.2 Traffic Classes

Intel DSA includes support for Traffic Classes as defined in PCIe, if the PCIe VC capability is present. Traffic Classes may be used by the platform outside of the device to control QoS for memory transactions initiated by the device.

Traffic Classes are also used within the device to segregate traffic destined for low bandwidth memory. Each platform has one or more designated traffic class values that should be used for accesses to low bandwidth memory. See section 4.5 for information on configuring traffic classes for use with low bandwidth memory.

There are two traffic class registers in each Group Configuration register. Each descriptor has flags to select which of the two traffic classes to use for each buffer referenced by the descriptor. For best results, software should ensure that operations with dissimilar QoS characteristics are issued to different groups.

4.3 Read Buffer Allocation

The Intel DSA device uses read buffers to hide memory read latency. Software can control how these read buffers are allocated, which affects the read bandwidth available to certain guests or applications.

Read Buffers are resources within the Intel DSA implementation that are allocated to engines to support memory read operations. The total number of Read Buffers supported is fixed by the implementation and is reported in the GRPCAP register. Limiting the number of Read Buffers available to a group can restrict the read bandwidth usable by engines in the group. The relationship between Read Buffers and actual bandwidth is dependent on instantaneous system memory latency and varies dynamically as system utilization changes. Read Buffers are internal to the design of Intel DSA and are not related to other resources in the SoC that also affect the bandwidth available to the device.

The policy by which Read Buffers are allocated to groups is based on two fields in the Group Configuration registers. The Read Buffers Reserved field indicates the number of Read Buffers set aside for the exclusive use of engines in the group. The Read Buffers Allowed field indicates the maximum number of Read Buffers that may be in use at one time by all engines in the group. (Read Buffers allocated to a group may also be limited by the Global Read Buffer Limit, as described in section 9.2.8.)

Setting the Read Buffers Reserved field to a non-zero value ensures that engines in the group are able to acquire Read Buffers without being starved by engines in other groups. However, reserving Read Buffers for a group limits the number of Read Buffers available to other groups, potentially limiting their ability to efficiently utilize the bandwidth capability of the device. The sum of the Read Buffers reserved for all groups must be no greater than the total number of Read Buffers available (as reported in GRPCAP).

For each group, the Read Buffers Allowed field must be greater than or equal to 4 times the number of engines in the group. It must also be no greater than the value of the Read Buffers Reserved field for that group plus the number of non-reserved Read Buffers. The number of non-reserved Read Buffers is defined as the total number of Read Buffers supported minus the number of Read Buffers reserved for all groups combined. For example, if the device supports 74 Read Buffers and 3 are reserved for each of the four groups, then 62 Read Buffers remain non-reserved, and the maximum value of Read Buffers Allowed for each group would be 65.

There is a system-specific value for the Read Buffers Allowed field (dependent on the read latency of the system) that allows the group to utilize the full bandwidth of the device. This value can be calibrated by software. There is no advantage to setting the Read Buffers Allowed field to a greater value. Setting this field to a smaller value limits the Read Buffers that can be allocated to engines in the group, thereby limiting their impact on the performance of engines in other groups.

4.4 Latency Control

Intel DSA provides controls for software to reduce the latency impact of multiple outstanding descriptors being in progress in the device. Within an engine, multiple descriptors may be processed concurrently, in a pipelined manner. If the Descriptors in Progress Limit Supported field in GRPCAP is 1, the Maximum Work Descriptors in Progress and Maximum Batch Descriptors in Progress fields in ENGCAP specify the maximum number that may be concurrently in process in an engine at a given time. Software can use the Work Descriptors in Progress Limit and Batch Descriptors in Progress Limit fields in GRPCFG to limit it to a value smaller than the maximum supported by the implementation. The GRPFLAGS register allows software to configure the Descriptors in Progress Limit and Batch Descriptors in Progress Limit fields as a fraction of the corresponding maximum permissible value. The values specified apply to each engine in the group. Specifying a value smaller than the maximum can sometimes result in a lower achievable bandwidth depending on system latency conditions. Software can set the limit fields judiciously to achieve the right balance between peak performance and quality of service considerations.

The arbiter for each group dispatches descriptors from a WQ to an engine in the group up to the maximum value configured for the engine. Once the number of descriptors dispatched to an engine reaches the maximum value, further dispatch to the engine is stalled until one or more descriptors are completed. During this time, additional descriptors may queue up in the WQ and eventually result in back pressure to software attempting to submit additional descriptors. Software can use this control along with the WQ size, maximum transfer size, and maximum batch size controls in WQ Configuration to limit the maximum latency that a descriptor may experience due to prior descriptors. The actual realizable latency benefit is implementation and system dependent. Depending on prevailing system latency values, use of this control could result in lower effective memory bandwidth from an engine.

4.5 Low Bandwidth Memory

Intel DSA includes features to improve system performance when accessing memory with lower bandwidth or higher latency than DRAM, such as Intel® Optane™ DC persistent memory. When Intel DSA is used to read and/or write to these types of memory, software should take these steps to limit the impact to the throughput of other operations, both within the device and throughout the platform.

1. Set the Global Read Buffer Limit field in GENCFG to a suitable value for the bandwidth available. (This value is platform dependent and can be calibrated by software.)
2. Create one or more groups that will be used with descriptors that access low bandwidth memory.
3. Set the Use Global Read Buffer Limit field to 1 in the Group Configuration register for those groups.
4. Set TC-B field in those groups to a Traffic Class value that is designated for use with low bandwidth memory.
5. Each descriptor should set the TC Selector flags to indicate which of its source and destination addresses refer to low bandwidth memory.

Software must take care to submit work to a suitable group and to correctly classify each buffer address in a descriptor and set the TC Selector flags. If a descriptor referencing low bandwidth memory is submitted to a group that is not configured to support low bandwidth memory, or if a TC Selector flag in a descriptor incorrectly indicates that the corresponding address is not in low bandwidth memory, the memory transaction may be blocked by the platform. If the memory transaction is a write operation, software will not be notified.

If software cannot correctly classify its buffers, for example, if the memory allocation strategy of system software mixes normal and low bandwidth memory in such a way that an application cannot tell which type of memory it has received, then both the TC-A and TC-B fields of GRPCFG should be set to TC values that are suitable for low bandwidth memory.

The number of Read Buffers specified by Global Read Buffer Limit is shared by all descriptors executing in all groups for which Use Global Read Buffer Limit is 1. The engine executing the descriptor is also limited by the Read Buffers Allowed field in GRPCFG.

§

5 Error Handling

The primary goals of Intel DSA error detection and handling are:

- Avoid writing incorrect data or writing to an incorrect address.
- Avoid errors due to one client from affecting work for other clients.
- Avoid misinterpreting an erroneous operation descriptor.
- Report errors to the client that submitted the work when possible.
- Provide enough information to continue an operation that was partially completed.
- Provide enough information to help diagnose the cause of the error.

Errors associated with the processing of a descriptor are reported in the completion record of the descriptor (if the completion record address is valid).

Hardware errors are reported via PCI Express Advanced Error Reporting. Hardware errors include errors in the fabric and errors internal to the device. If a hardware error is associated with a descriptor and the Completion Record Address in the descriptor is valid, the error is also reported in the completion record.

Errors on the completion record of a descriptor or during processing of a descriptor that does not have a valid completion record address are reported in the Software Error Register or event log. A synopsis of software error handling is shown in Table 5-1.

5.1 Device Enable Checks

The device performs the following checks at the time the Enable Device command is issued to the Command Register:

- Bus Master Enable is 1.
- The sum of the WQ Size fields of all the WQCFG registers is not greater than Total WQ Size.
- For each GRPCFG register:
  - The WQs and Engines fields are either both zero or both non-zero.
  - Bits in the WQs field beyond the number of WQs are 0.
  - Bits in the Engines field beyond the number of Engines are 0.
  - For Group Configuration registers beyond the number of groups, all fields are zero.
- Each WQ for which the Size field in the WQCFG register is non-zero is in exactly one group.
- Each WQ for which the Size field in the WQCFG register is zero is not in any group.
- Each engine is in no more than one group.
- If the Global Read Buffer Limit Supported field in GRPCAP is 0, then the Use Global Read Buffer Limit field is 0 in every GRPCFG register.
- If the Global Read Buffer Limit Supported field in GRPCAP is 1, then the Global Read Buffer Limit in GENCFG is less than or equal to the Total Read Buffers field in GRPCAP.
- If the Use Global Read Buffer Limit field is 1 in any GRPCFG register, then the Global Read Buffer Limit in GENCFG is at least 4 times the total number of engines in all groups that have the Use Global Read Buffer Limit set to 1.
- If the Read Buffer Controls Supported field in GRPCAP is 1, then the sum of the Read Buffers Reserved fields, for all groups that have engines assigned, are less than or equal to the Total Read Buffers field in GRPCAP.
- If the Read Buffer Controls Supported field in GRPCAP is 1, then for each group that has engines assigned to it, Read Buffers Allowed is:
  - Greater than or equal to 4 times the number of engines in the group;

  - Greater than or equal to the Read Buffers Reserved field for the group; and
  - Less than or equal to the sum of the Read Buffers Reserved field and the number of non-reserved Read Buffers.
- If the Enable bit in PRSCTL is 1, then the number of Outstanding Page Requests Allowed in PRSREQALLOC is non-zero and is less than or equal to the maximum number of Page Requests supported in PRSREQCAP.
- If Event Log Enable in GENCFG is 1:
  - Event Log Support in GENCAP is not 0.
  - Event Log Size ≥ 64.
  - Event Log Base Address + Event Log Size × Event Log entry size ≤ 2^{64}.
- In cases where the platform has indicated a requirement to either use ATS or not to use ATS, the setting of ATS Enable matches the requirement.
- If the PASID Enable bit in EVLCFG is 1, the PASID Enable bit in PASIDCTL must also be 1.
- If the PASID Enable bit in IDBR is 1, the PASID Enable bit in PASIDCTL must also be 1.

Category	Error Type	Intel® DSA Handling
Descriptor submission	Posted write to SWQ. Posted write to WQ that is not Enabled. Non-64-byte write to any portal.	Ignored.
	DMWr to DWQ. DMWr to WQ that is not Enabled. DMWr to non-WQ address.	Returns Retry.
Descriptor errors	Misaligned completion record address.	Reported in SWERROR or event log.
	Failure translating completion record address.	
	Descriptor decode error: invalid operation, invalid flags, non-zero reserved field, etc.	Completion Record Address Valid = 1: Reported in completion record. Completion Record Address Valid = 0: Reported in SWERROR register or event log.
	Error in descriptor processing (e.g., PRS failure, IDPT permission violation).	
	Invalid interrupt handle.	Completion Record Address Valid = 1: Reported in completion record and in SWERROR[1] or event log. Completion Record Address Valid = 0: Reported in SWERROR register or event log.
Configuration errors	Invalid device configuration when Enable Device command is issued.	Reported in Command Status register. Device is not enabled.
	Invalid work queue configuration when WQ Enable command is issued.	Reported in Command Status register. WQ is not enabled.
	Unsupported change to PCI configuration while device is not Disabled (including BME, ATS, PASID, and PRS).	Device enters Halt state. Reported in SWERROR register.

Table 5-1: Handling of Software Errors

[1] In some implementations, an invalid interrupt handle error that is reported in the completion record is not also reported in SWERROR.

If any of the device enable checks fail, the device is not enabled and the error is reported in the Command Status register. These checks may be performed in any order. Thus, an indication of one type of error does not imply that there are not also other errors. The same configuration errors may result in different error codes at different times or with different versions of the device.

If none of the checks fail, the device is enabled and the Command Status register is set to indicate successful completion of the Enable Device command. If Event Log Enable in GENCFG is 1, the Event Log Status register is set to 0.

5.2 WQ Enable Checks

The device performs the following checks at the time the Enable WQ command is issued to the Command Register:

- The device is Enabled.
- The WQ parameter is less than the number of work queues.
- The WQ is Disabled.
- The WQ Size field is non-zero.
- The WQ Mode field selects a supported mode. That is, if the Shared Mode Support field in WQCAP is 0, WQ Mode is 1; or if the Dedicated Mode Support field in WQCAP is 0, WQ Mode is 0. If both the Shared Mode Support and Dedicated Mode Support fields are 1, either value of WQ Mode is allowed.
- If WQ Priority Support is 1, the WQ Priority field is non-zero.
- If the Block on Fault Support field in GENCAP is 0 or the Enable field of the PCIe Page Request Control register is 0, the WQ Block on Fault Enable field is 0.
- If the WQ Mode field is 0, the WQ PASID Enable field is 1.
- If the PASID Enable field of the PCI Express PASID capability is 0, the WQ PASID Enable field is 0. (This rule, in combination with the above rule, means that Shared WQs cannot be used when the PASID capability is disabled.)
- If the WQ Mode field is 1, WQ PASID Enable is 1, and the Privileged Mode Enable field of the PCI Express PASID capability is 0, then the WQ Priv field must be 0.
- The WQ Maximum Transfer Size field is not greater than the Maximum Supported Transfer Size field in GENCAP.
- The WQ Maximum Batch Size field is greater than 0 and not greater than the Maximum Supported Batch Size field in GENCAP.
- If the IMS Support field in the SIOV capability is 0 or if the SIOV capability is not present, WQ Occupancy Interrupt Table is 0.
- If the Request Interrupt Handle command is not supported, then WQ Occupancy Interrupt Handle is less than the size of the selected interrupt table (the MSI-X table if WQ Occupancy Interrupt Table is 0; the IMS table if WQ Occupancy Interrupt Table is 1).[1]
- If the Request Interrupt Handle command is supported, then WQ Occupancy Interrupt Handle must be a handle returned by that command.
- If WQ Operations Configuration Support in WQCAP is 1, the OPCFG field in WQCFG must not have any bits set to 1 that are not set to 1 in OPCAP. That is, WQOPCFG AND ~OPCAP must be 0.
- If WQ ATS Support in WQCAP is 0, WQ ATS Disable is 0.
- If WQ PRS Support in WQCAP is 0 or Event Log Enable in GENCFG is 0 or WQ Block on Fault Enable is 1, then WQ PRS Disable is 0.

[1] The device may discard upper bits of the WQ Occupancy Interrupt Handle before performing this check.

If any of the WQ enable checks fail, the WQ is not enabled and the error is reported in the Command Status register. These checks may be performed in any order. Thus, an indication of one type of error does not imply that there are not also other errors. The same configuration errors may result in different error codes at different times or with different versions of the device.

If none of the checks fail, the WQ is enabled and the Command Status register is set to indicate successful completion of the Enable WQ command.

5.3 Descriptor Submission Checks

The device performs the following checks in order when a descriptor is received. Except as noted, if any of these checks fail, the descriptor is discarded, and if the descriptor was submitted with DMWr, a Retry response is returned.

- The WQ identified by the portal address used to submit the descriptor is Enabled.
- If the descriptor was submitted to a shared WQ:
  - It was submitted with DMWr (e.g., using the ENQCMD or ENQCMDS instruction).
  - If the descriptor was submitted via a limited portal, the current queue occupancy is less than the WQ Threshold.[1]
  - If the descriptor was submitted via an unlimited portal, the current queue occupancy is less than WQ Size.
- If the descriptor was submitted to a dedicated WQ:
  - It was submitted with a posted, aligned 64-byte write (e.g., using the MOVDIR64B instruction).
  - The queue occupancy is less than WQ Size. If this check fails, the descriptor is discarded, and the error is recorded in SWERROR.

Note that if MOVDIR64B is used to write to a disabled WQ, a shared WQ, or an invalid portal address, the write is discarded without notification to software.

5.4 Descriptor Checks

The device performs the following checks on each descriptor when it is processed:

- If the Completion Record Address Valid flag is 1, the Completion Record Address is 32-byte aligned.
- The value in the operation code field corresponds to a supported operation (as indicated in OPCAP).
- The operation is valid in the context in which it was submitted. Batch and Drain operations are not supported inside a batch and are treated as invalid operation codes.
- No reserved flags are set. This includes flags for which the corresponding capability bit in the GENCAP register is 0.
- No unsupported flags in the Flags field are set. This includes flags that are reserved for use with certain operations. For example, the Fence bit is reserved in descriptors that are enqueued directly rather than as part of a batch. It also includes flags that are disabled in the configuration, such as the Block On Fault flag, which is reserved when the Block On Fault Enable field in the WQCFG register is 0. See Table 5-3 and Table 5-4 for details.
- Required flags in the Flags field are set. For example, the Request Completion Record flag must be 1 in a descriptor for the Compare operation. See Table 5-5 for details.

[1] If WQ Threshold is greater than WQ Size, it is treated as if it is equal to WQ Size.

- Reserved fields (other than flags) are 0. This includes any fields that have no defined meaning for the specified operation. Some implementations may not check all reserved fields, but software should take care to clear all unused fields for maximum compatibility.
- In a descriptor submitted to a shared WQ, if the Privileged Mode Enable field of the PCI Express PASID capability is 0, the Priv field is 0.
- In a Batch descriptor, the Descriptor Count field is greater than 1 and is not greater than the value specified by the WQ Maximum Batch Size field in the WQ Config register.
- The Transfer Size (if applicable for the descriptor type) is greater than 0 and not greater than the value specified by the WQ Maximum Transfer Size field in the WQ Config register.
- In a Create Delta Record or Apply Delta Record descriptor, the Transfer Size is not greater than the allowed value (0x80000 bytes, or 512 KB, as described in section 8.3.8).
- In a Create Delta Record or Apply Delta Record descriptor, the Maximum Delta Record Size or Delta Record Size (as applicable for the descriptor type) is not greater than the value specified by the WQ Maximum Transfer Size field in the WQ Config register.
- In a Create Delta Record descriptor, the Maximum Delta Record Size is greater than or equal to 80 bytes.
- In an Apply Delta Record descriptor, the Delta Record Size is greater than or equal to 10 bytes.
- In a Memory Copy with Dualcast descriptor, bits 11:0 of the two destination addresses are the same.
- Destination buffers do not overlap source buffers or other destination buffers. (This check is not performed for a Memory Move operation if the Overlapping Copy Support capability is 1.)
- If the Request Completion Interrupt flag is 1, the Completion Interrupt Handle is valid according to Table 5-2.
- If the Request Completion Interrupt flag is 1, the PASID Enable field in the selected interrupt table entry equals the WQ PASID Enable control for the work queue the descriptor was submitted to. Furthermore, if the PASID Enable field is 1, the PASID field in the selected interrupt table entry equals the PASID of the descriptor.
- The Traffic Classes selected by descriptor flags have the corresponding bits set in the TC/VC map of a VC Resource Control register, and the VC Enable bit in that register is 1.
- The value in the operation code field corresponds to a supported operation as indicated in WQ OPCFG, if WQ Operations Configuration Support in WQCAP is 1; or as indicated in OPCAP.
- The operation is valid in the context in which it was submitted. Batch, Drain, and Update Window operations are not supported inside a batch and are treated as invalid operation codes. A Translation Fetch operation submitted to a WQ with WQ ATS Disable set to 1 in the WQ configuration is treated as an invalid operation code.
- In a Translation Fetch descriptor, the Region Size is non-zero, and the sum of Address and Region Size is less than or equal to 2^{64}.

Submission Portal	Request Interrupt Handle Command Available	Completion Interrupt Handle Check
MSI-X	No	Less than the size of the MSI-X table.
IMS	No	Less than Interrupt Message Storage Size.
MSI-X	Yes	A valid MSI-X handle returned by the Request Interrupt Handle command that has not been revoked.
IMS	Yes	A valid IMS handle returned by the Request Interrupt Handle command that has not been revoked.

Table 5-2: Completion Interrupt Handle Checks

- In a Translation Fetch descriptor, the Region Stride is greater than or equal to 4096 and is a power of 2.
- In an Inter-Domain Memory Copy or Inter-Domain Compare descriptor, at least one of the Use Alternate PASID flags is 1.
- In an Inter-Domain Memory Fill or Inter-Domain Cache Flush descriptor, the Use Alternate Destination PASID flag is 1.
- In an Inter-Domain Compare Pattern descriptor, the Use Alternate Source PASID flag is 1.
- For an inter-domain or Update Window descriptor, the WQ PASID Enable control for the work queue the descriptor was submitted to is 1.
- If the Request IDPT Handle command is not supported, any IDPT handle in an inter-domain or Update Window descriptor is less than the Inter-Domain Permissions Table size.
- If the Request IDPT Handle command is supported, any IDPT handle in an inter-domain or Update Window descriptor is a handle returned by that command.
- In an Update Window descriptor, if the Window Enable flag is 1, the Window Size field is non-zero.

If the Completion Record Address Valid flag is 0 and any of these checks fail, the error is reported in the Software Error register or event log.

If the Completion Record Address Valid flag is 1 and the Completion Record Address is misaligned or cannot be translated, or the completion record TC is invalid, then the descriptor is discarded and an error is reported in the Software Error Register or event log.

Otherwise, if any of these checks fail, the completion record is written with the Status field indicating the type of check that failed and Bytes Completed set to 0. If one of the flags checks fails, the Invalid Flags field of the completion record indicates flags that are invalid. A completion interrupt is generated, if requested, unless a check related to completion interrupt delivery failed. If an invalid interrupt handle was specified, the error is also reported in the Event Log, if enabled, or in the Software Error register.[1]

These checks may be performed in any order. Thus, an indication of one type of error in the completion record does not imply that there are not also other errors. The same invalid descriptor may report different error codes at different times or with different versions of the device.

5.5 Descriptor Reserved Field Checking

Reserved fields in descriptors fall into three categories: fields that are always reserved; fields that are reserved under some conditions (e.g., based on a capability, configuration field, how the descriptor was submitted, or values of other fields in the descriptor itself); and fields that are reserved based on the operation type. For additional details on descriptor formats, see chapter 8.

Table 5-3 lists the flags and fields that are allowed and reserved for each operation type. Flags not listed are allowed for all operation types. Flag bits 5, 6, and 15 are reserved for all operation types. Flag bits 23:16 are operation specific and are reserved except when the operation description describes their use. Table 5-4 lists additional conditions under which certain flags and fields are reserved. Additional operation-specific reserved fields and flags are described with the respective descriptor details in section 8.3. Table 5-5 lists the operation types that require certain flags to be set to 1.

[1] In some implementations, an invalid interrupt handle error that is reported in the completion record is not also reported in SWERROR.

Operation	Allowed Flags									Reserved Fields
	Block on Fault	Check Result	Cache Control	Strict Ordering	Destination Readback	Address 1 TC Selector	Address 2 TC Selector	Address 3 TC Selector	Fence	
No-op									●	Bytes 16-35; 38-63
Drain						●	●			Bytes 32-35; 38-63
Memory Move	●		●	●	●	●	●		●	Bytes 38-63
Fill	●		●	●	●		●		●	Bytes 38-63
Compare	●	●				●	●		●	Bytes 38-39; 41-63
Compare Pattern	●	●				●			●	Bytes 38-39; 41-63
Create Delta Record	●	●	●	●	●	●	●	●	●	Bytes 38-39; 52-55; 57-63
Apply Delta Record	●		●	●	●	●	●		●	Bytes 38-39; 44-63
Dualcast	●		●	●	●	●	●	●	●	Bytes 38-39; 48-63
CRC Generation	●					●		●	●	Bytes 24-31; 38-39; 44-63
Copy with CRC Generation	●		●	●	●	●	●	●	●	Bytes 38-39; 44-63
DIF Check	●					●			●	Bytes 24-31; 38-39; 41; 43-47; 56-63
DIF Insert	●		●	●	●	●	●		●	Bytes 38-39; 40; 43-55
DIF Strip	●		●	●	●	●	●		●	Bytes 38-39; 41; 43-47; 56-63
DIF Update	●		●	●	●	●	●		●	Bytes 38-39; 43-47
Cache flush	●			●	●		●		●	Bytes 16-23; 38-63
Batch						●				Bytes 24-31; 38-63
Inter-Domain Copy	●		●	●	●	●	●		●	Bytes 38-59
Inter-Domain Fill	●		●	●	●		●		●	Bytes 38-39; 48-61
Inter-Domain Compare	●	●				●	●		●	Bytes 38-39; 41-59
Inter-Domain Compare Pattern	●	●				●			●	Bytes 38-39; 41-59; 62-63
Inter-Domain Cache flush	●			●	●		●		●	Bytes 16-23; 38-61
Update Window										Bytes 32-35; 38-60
Translation Fetch	●								●	Bytes 24-31; 38-47; 52-63

Table 5-3: Supported Flags and Reserved Fields by Operations

Reserved Field	Conditions Under Which Field is Reserved
Request Completion Interrupt	User-mode Interrupts Enable = 0 and WQ PASID Enable = 1 and Priv = 0.
Completion Interrupt Handle	Request Completion Interrupt = 0.
Fence	Descriptor submitted directly to WQ (not in a batch).
Block On Fault	WQ Block On Fault Enable = 0.
Destination Readback	GENCAP Destination Readback Support = 0.
Address 1 TC Selector	In Drain descriptor, when Readback Address 1 Valid is 0.
Address 2 TC Selector	In Drain descriptor, when Readback Address 2 Valid is 0.
Completion Record Address	Completion Record Address Valid = 0.
Request Completion Record	Completion Record Address Valid = 0.
Completion Record TC Selector	Completion Record Address Valid = 0.
Cache Control	For operations that write to memory, if GENCAP Cache Control Support = 0.
Readback Address 1 valid in Drain descriptor	Drain Descriptor Readback Address Support = 0.
Readback Address 2 valid in Drain descriptor	Drain Descriptor Readback Address Support = 0.
Source/Source1 IDPT Handle	In Inter-Domain descriptors, if Use Alternate Source/Source1 PASID = 0.
Destination/Source2 IDPT Handle	In Inter-Domain descriptors, if Use Alternate Destination/Source2 PASID = 0.
Window Mode	In an Update Window descriptor, if Window Enable is 0 or if Offset Mode Support in IDCAP is 0.
Window Base Address	In an Update Window descriptor, if Window Enable is 0.
Window Size	In an Update Window descriptor, if Window Enable is 0.

Table 5-4: Conditional Reserved Field Checking

Operation	Required Flags (Must be 1)
Drain	Either Request Completion Record or Request Completion Interrupt must be set to 1.
Compare	Completion Record Address Valid and Request Completion Record flags must be 1.
Compare Pattern	
Create Delta Record	
CRC Generation	
Copy with CRC Generation	
DIF Check	
Inter-Domain Copy	Either Use Alternate Source PASID must be 1 or Use Alternate Destination PASID must be 1.
Inter-Domain Fill	Use Alternate Destination PASID must be 1.
Inter-Domain Compare	Either Use Alternate Source1 PASID must be 1 or Use Alternate Source2 PASID must be 1.
Inter-Domain Compare Pattern	Use Alternate Source PASID must be 1.
Inter-Domain Cache Flush	Use Alternate Destination PASID must be 1.

Table 5-5: Operation Types with Required (Must be 1) Flags

5.6 Inter-Domain Permissions Table Entry Checks

The device performs the following checks while processing a descriptor that references an IDPT entry:

- Any IDPTE referenced in a descriptor must satisfy the following:
    - Reserved fields are 0 as described in section 9.2.29. (This includes fields that are conditionally reserved).
    - The type field in the IDPTE must be one of the supported types in IDCAP.
    - If Window Enable is 1:
        - Window Base plus Window Size must be less than or equal to 2^{64} (i.e., the window must not wrap around the 2^{64} address boundary).
        - Window Size is not zero.
- For any inter-domain descriptor:
    - Each referenced IDPTE must have the Usable field as 1.
    - An IDPTE corresponding to a source address must have read permission as 1.
    - An IDPTE corresponding to a destination address must have write permission as 1.
    - For an Inter-Domain Cache Flush operation, either R or W must be 1.
    - If type is 0, the descriptor PASID must match the submitter PASID field in the IDPTE.
    - If type is 1, the bit corresponding to the descriptor PASID must be 1 in the submitter bitmap in memory.
    - If Window Enable in the IDPTE is 1 and Window Mode is 0, then the address range in the descriptor referencing the IDPTE must lie within the range specified by the window base and size; specifically:
        - Address must be greater than or equal to Window Base, and less than Window Base plus Window Size.
        - Address plus transfer size must be greater than Window Base, and less than or equal to Window Base plus Window Size.
    - If Window Enable in the IDPTE is 1 and Window Mode is 1, then address must be less than Window Size and address plus transfer size must be less than or equal to Window Size.

- For an Update Window descriptor:
    - The Allow Update bit in the IDPTE must be 1.
    - The descriptor PASID must match the access PASID field in the IDPTE.

5.7 Device Halt State

In addition to its normal states of operation, Intel DSA has a halt state to deal with various error or unsupported conditions and reset transitions. Software can find out the current device state by reading the Device State field of the General Status register. GENSTS also indicates the type of reset required to recover from the device halt condition. Based on this, software determines how to reset the device and bring it to a Normal mode of operation. If the Halt State Interrupt Enable field in GENCTRL is 1, an interrupt using entry 0 of the MSI-X table is generated when the device enters halt state. The Halt State field in the INTCAUSE register is set to 1 to indicate the interrupt cause to software. Some of the causes that may result in the device entering the halt state include:

- Unsupported PCIe configuration changes (for example, setting BME to 0).
- Parity error on a register write or certain internal buffers.
- Severe I/O fabric error (e.g., parity error encountered on transaction received over internal I/O fabric).

Note that not all errors result in the device entering this state, and most errors are handled without causing the device to Halt. It may also be noted that parity errors on data are normally reported and handled via PCIe AER mechanism and are not considered a severe I/O error.

In this state, the device typically stops sending upstream reads and writes. Depending on the severity of error, the device may continue to send completions for non-posted requests (e.g., register reads). New descriptor submissions via ENQCMD or ENQCMDS receive a retry response. The device typically continues to send invalidation completions unless it has encountered a severe I/O fabric error or is actively going through a PCIe reset. An implementation may treat configuration registers that are read-write while the device is Disabled as read-write in the Halt state also.

This state requires some level of reset to restore the device to normal operation. The type of reset needed (Reset device command, Function-level reset, warm reset, or cold reset) is indicated by the Reset Type Needed to Recover field in the GENSTS register (See section 9.2.10).

5.8 Error Codes

5.8.1 Operation Status Codes

The operation status code for a descriptor is written to the Status field of the completion record for the descriptor if a valid completion record is available for the descriptor. If the operation status is 0x1a, 0x1b, or 0x1d, or if the Completion Record Address Valid Flag is 0 and the operation status is not equal to 0x01, 0x02, or 0x05, then the operation status code is instead written either to the SWERROR register or to the Event Log if enabled.

0x01	Success.
0x02	Success with false predicate.
0x03	Partial completion due to page fault, when the Block on Fault flag in the descriptor is 0.
0x04	Partial completion due to an Invalid Request response to a Page Request.
0x05	One or more operations in the batch completed with Status not equal to Success. This value is used only for a Batch descriptor.
0x06	Partial completion of batch due to page fault while translating the Descriptor List Address in a Batch descriptor and: - Page Request Services are disabled; - WQ PRS Disable is 1; or - An Invalid Request response was received for the Page Request for the Descriptor List Address. This value is used only for a Batch descriptor.
0x07	Offsets in the delta record were not in increasing order. This value is used only for an Apply Delta Record operation.
0x08	An offset in the delta record was greater than or equal to the Transfer Size of the descriptor. This value is used only for an Apply Delta Record operation.
0x09	DIF error. This value is used for the DIF Check, DIF Strip, and DIF Update operations.
0x0a – 0x0f	Unused.
0x10	Unsupported operation code.
0x11	Invalid flags. One or more flags in the descriptor Flags field contain an unsupported or reserved value.
0x12	Non-zero reserved field (other than a flag in the Flags field).
0x13	Invalid Transfer Size.
0x14	Descriptor Count out of range (less than 2 or greater than the maximum batch size for the WQ).
0x15	Maximum Delta Record Size or Delta Record Size out of range.
0x16	Overlapping buffers.
0x17	Bits 11:0 of the two destination buffers differ in Memory Copy with Dualcast.
0x18	Misaligned Descriptor List Address.
0x19	Invalid Completion Interrupt Handle. - If the Request Interrupt Handle command is not supported: o The handle is out of range of the MSI-X or IMS table. - If the Request Interrupt Handle command is supported: o The interrupt handle was not returned by the Request Interrupt Handle command. o The interrupt handle has been revoked. See section 3.7. - The PASID Enable and PASID fields in the selected interrupt table entry don't match those of the descriptor.

0x1a	A page fault occurred while translating a Completion Record Address and: - Page Request Services are disabled; - WQ PRS Disable is 1; or - An Invalid Request response was received for the Page Request for the completion record.
0x1b	Completion Record Address is not 32-byte aligned.
0x1c	Misaligned address, size, or stride field: - In a Create Delta Record or Apply Delta Record operation: Source1 Address, Source2 Address, Destination Address, or Transfer Size is not 8-byte aligned. - In a CRC Generation or Copy with CRC Generation operation: CRC Seed Address is not 4-byte aligned. - In a Translation Fetch operation: Region Stride is less than 4096 or is not a power of 2. - In a DIX Generate operation: Destination Address is not 8-byte aligned.
0x1d	In a descriptor submitted to an SWQ, Priv is 1 and the Privileged Mode Enable field of the PCI Express PASID capability is 0.
0x1e	Incorrect Traffic Class configuration: - A TC selected by the descriptor is not enabled in the TC/VC Map of any VC Resource Control register. - A TC selected by the descriptor is enabled in the TC/VC Map of a VC Resource Control register in which VC Enable is 0.
0x1f	A page fault occurred while translating a Readback Address in a Drain descriptor and: - Page Request Services are disabled; - WQ PRS Disable is 1; or - An Invalid Request response was received for the Page Request for the Drain Readback Address.
0x20	The operation failed due to a hardware error not covered by error code 0x21 or 0x22. This error code can occur due to a UR or CA response or completion timeout on a read operation, including an address translation fault when ATS is disabled, or a hardware error such as a parity error. Details of the hardware error are reported via PCIe Advanced Error Reporting (AER), if enabled.
0x21	Hardware error, including a UR or CA response or completion timeout, on a memory write[1] or destination readback operation. Error details are reported via PCIe Advanced Error Reporting (AER), if enabled.
0x22	An error occurred during address translation: - A UR or CA response or a completion timeout (CTO) on an ATS translation request. - A Response Failure response to a Page Request. The error is also recorded in SWERROR and in some cases, also via PCIe Advanced Error Reporting (AER), if enabled.
0x23 – 0x25	Unused.
0x26	Hardware may have encountered a page fault on the completion record for one or more descriptors prior to a Drain descriptor. This error is only reported in an Event Log entry for a Drain descriptor (described in Section 5.9).
0x27	Hardware may have encountered a page fault on the completion record for one or more descriptors in a batch. This error is only reported in an Event Log entry for a Batch descriptor (described in Section 5.9).

[1] Reporting of hardware errors on memory writes is implementation dependent.

0x28	An inter-domain or Update Window operation was submitted to a WQ that has WQ PASID Enable as 0.
0x29	Invalid IDPTE handle: An inter-domain operation referenced an IDPTE that is invalid, out-of-range, not usable, or inaccessible to the submitter. Refer to section 5.6 for a list of IDPT checks.
0x2a	An inter-domain operation failed due to insufficient permissions for the requested access. - A read operation using an IDPTE that did not grant read permissions. - A write operation using an IDPTE that did not grant write permissions.
0x2b	An inter-domain operation failed window address range checks.
0x2c	An Update Window operation referenced an IDPTE that is invalid, out-of-range, not modifiable, or not owned by the submitter. Refer to section 5.6 for a list of IDPT checks.
0x2d	Invalid window control fields in an Update Window descriptor - The Window Mode flag in the descriptor is set to enable Offset Mode while Offset Mode Support in IDCAP is 0. - The Window Enable flag in the descriptor is 1, but Window Size is 0. - Window Base plus Window Size is greater than 2^{64}.
0x2e	An inter-domain operation failed because the requested domain was inaccessible. Hardware reports this error code when terminating an inter-domain operation while executing an Abort command.
0x2f – 0x3f	Unused.

Table 5-6: Operation Status Codes

5.8.2 Other Software Error Codes

0x51	An unsupported change was made to one of the registers in PCI configuration space while the device was not Disabled. This causes the device to enter the Halt State. This error is only reported in the SWERROR register.
0x52	The Command register was written while the Active field of the Command Status register was 1. This error is reported either in the SWERROR register or in the Event Log if enabled.
0x53	A descriptor was submitted to a dedicated WQ that had no space to accept the descriptor. This error is only reported in the SWERROR register.
0x54	The device was unable to append an event to the Event Log because it was full and Event Log Overflow Support in GENCAP is 1. This error is only reported in the SWERROR register.

Table 5-7: Other Software Error Codes

5.8.3 Administrative Command Error Codes

These errors are reported in the Command Status register (described in Section 9.2.13).

Command	**Error codes**
Enable Device	**0x10:** Device is not Disabled. **0x11:** Unspecified error in configuration when enabling the device. **0x12:** Bus Master Enable is 0. **0x13:** PRSREQALLOC is configured with an unsupported value. **0x14:** Sum of WQCFG Size fields is out of range. **0x15:** Invalid Group configuration:

Command	Error codes
	- A Group Configuration register has one or more WQs and zero engines or has one or more engines and zero WQs. - A Group Configuration register beyond the number of groups contains non-zero fields. **0x16:** Invalid Group configuration: - A WQ is in more than one group. - An active WQ (with non-zero WQ Size) is not in a group. - An inactive WQ is in a group. - Reserved bits are set in the WQs field of a Group Configuration Register. **0x17:** Invalid Group configuration: - An engine is in more than one group. - Reserved bits are set in the Engines field of a Group Configuration Register. **0x18:** Invalid Read Buffers configuration: - Invalid value for Global Read Buffer Limit in GENCFG or Use Global Read Buffer Limit in GRPCFG. - Invalid value for Read Buffers Allowed or Read Buffers Reserved in GRPCFG. **0x19:** Invalid Event Log configuration: - Event Log Enable in GENCFG is 1 when Event Log Support in GENCAP is 0. - Event Log Size < 64. - Event Log Base Address + Event Log Size × Event Log entry size > 2^{64}. **0x1a**: Invalid ATS configuration: - ATS Enable is 0 and ATS is required by the platform; or - ATS Enable is 1 and ATS is not supported in the platform.
Enable WQ	**0x20:** Device is not Enabled. **0x21:** WQ is not Disabled. **0x22:** WQ Size is 0. Note: WQ Size out of range is diagnosed when the device is enabled. **0x23:** WQ Priority is 0. **0x24:** Invalid WQ mode: - WQ Mode = 0 and WQCAP Shared Mode Support = 0; or - WQ Mode = 1 and WQCAP Dedicated Mode Support = 0. **0x25:** WQ Block on Fault Enable = 1 and either the Block on Fault Support field in GENCAP is 0 or the Enable field of the PCIe Page Request Control register is 0. **0x26:** Invalid value for WQ PASID Enable: - WQ PASID Enable = 0 and WQ Mode = 0; or - WQ PASID Enable = 1 and PCI Express PASID capability Enable = 0. **0x27:** Invalid WQ Maximum Batch Size - WQ Maximum Batch Size less than 1; or - WQ Maximum Batch Size greater than Maximum Supported Batch Size.

Command	Error codes
	0x28: Invalid WQ Maximum Transfer Size - WQ Maximum Transfer Size greater than Maximum Supported Transfer Size. **0x2a:** WQ Mode = 1, WQ PASID Enable = 1, WQ Priv = 1, and the Privileged Mode Enable field of the PCI Express PASID capability = 0. **0x2b:** Invalid value for WQ Occupancy Interrupt Table or WQ Occupancy Interrupt Handle. **0x2c:** WQ ATS Disable = 1 and the WQ ATS Support field in WQCAP is 0. **0x2d:** Invalid WQ OPCFG; i.e., WQ OPCFG AND ~OPCAP is not 0. **0x2e:** WQ PRS Disable = 1 and: - WQ PRS Support field in WQCAP is 0; - Event Log Enable in GENCFG is 0; or - WQ Block on Fault Enable is 1.
Disable Device	**0x31:** Device is not Enabled.
Disable WQ Drain WQ Abort WQ	**0x32:** One or more of the specified WQs are not Enabled.
Reset WQ Reset Device Drain All Abort All Drain PASID Abort PASID	No error codes are defined for these commands.
Request Interrupt Handle	**0x41:** Invalid interrupt table index. **0x42:** No handle is available.
Release Interrupt Handle	**0x41:** Invalid interrupt table index.
Request IDPT Handle	**0x51:** Invalid Inter-Domain Permissions Table index. **0x52:** No handle is available.
Release IDPT Handle	**0x51:** Invalid Inter-Domain Permissions Table index.

Table 5-8: Administrative Command Error Codes

5.9 Event Log

Errors on the completion record of a descriptor or during processing of a descriptor that does not have a valid completion record address are normally reported in the Software Error Register. Occurrence of multiple such errors before software has processed the Software Error Register results in an overflow condition. As an alternative, if the value of the Event Log Support field in GENCAP is not 0, hardware supports logging of such events in an Event Log in memory. Software specifies the address and size of the Event Log region in the EVLCFG register. The Event Log is enabled if the Event Log Enable bit in GENCFG is 1 when the device is enabled.

The size of entries in the Event Log is specified by the Event Log Support field in GENCAP. The format of Event Log entries is described in section 5.9.1. An implementation may issue Event Log writes as either Translated or Untranslated accesses. Software must pin memory pages corresponding to the Event Log. Event Log writes are performed with a TC value of 0. If the PASID Enable field in EVLCFG is 1, Event Log writes are issued as writes with PASID using the PASID and privilege specified in EVLCFG. If Event Log is enabled, the device initializes the head and tail fields in EVLSTATUS when the device is enabled. Once enabled, hardware writes each event to the offset specified by the Event Log Tail field in EVLSTATUS and increments the tail value. When the tail reaches the end of the log, it wraps to 0. The next event to be processed by software is specified by the Event Log Head field in EVLSTATUS. Software updates the head field after processing one or more events at the head of the Event Log. The log is full when tail + 1 mod log-size = head.

At the time of writing an event to the log, if the Event Log Interrupt Enable field in GENCTRL is 1 and the Interrupt Pending bit in EVLSTATUS is 0, hardware sets the Interrupt Pending bit in EVLSTATUS to 1, sets the Event Log field of the Interrupt Cause register to 1, and generates an interrupt using MSI-X entry 0. No further interrupts are generated for additional log entries until software clears the Interrupt Pending bit.

If Event Log Overflow Support in GENCAP is 0 and the Event Log is full when hardware tries to append an event, hardware blocks until software updates the Event Log Head field after processing one or more events at the head of the log. Hence software must ensure that the Event Log region in memory is adequately sized, and that Event Log entries are processed in a timely manner. If Event Log Overflow Support in GENCAP is 1 and the Event Log is full when hardware tries to append an event, the event is dropped and hardware attempts to log an error in the SWERROR register to indicate the event log full condition. If the Valid bit in the SWERROR register is already 1, the behavior is as described in section 9.2.15.

If hardware encounters a page fault on a completion record address while PRS is disabled[1], it is reported as an error. If the Event Log is enabled, hardware writes an entry to the Event Log with an appropriate error code indicating the cause of the page fault; otherwise, it is reported via SWERROR. In the former case, hardware also writes the completion record for that descriptor to the Event Log entry. Completion records written to the Event Log have the same format as described in Section 8.3. Software responsible for processing the Event Log is expected to perform the necessary actions to propagate the completion record, where appropriate, to the software entity that submitted the faulting descriptor. Software should also generate the completion interrupt, if requested.

Event Log entries are written in the order in which descriptors are completed by the engines, as described in section 3.9. If the completion record for any descriptor in a batch is written to the Event Log due to a page fault on the completion record address, the completion for the corresponding Batch descriptor is also written to the Event Log, if either a completion record or a completion interrupt is required for the Batch descriptor. An Event Log entry for a Batch descriptor is written only after any completion records and Event Log entries for descriptors in the batch are written. In this case, the error code in the Event Log entry for the batch descriptor indicates that one or more descriptors in the batch have associated Event Log entries with completion records that must be processed by software.

Hardware generates a batch identifier value to allow software to correlate Event Log entries for descriptors within a batch and for the corresponding Batch descriptor and is reported in the Batch Identifier field of an Event Log entry. A batch identifier may be reused by hardware once the Event Log entry for the Batch descriptor has been written. An Event Log entry with the First Error in Batch flag as 1 identifies the first entry for that batch. This allows software to identify any stale Event Log entries with the same batch identifier. If

[1] Either PCI Express PRS Enable Control is 0 or WQ PRS Disable in WQCFG is 1.

software encounters an entry with this flag as 1, any outstanding page faults previously recorded for the same batch identifier should be discarded.

The completion for a Drain descriptor is written to the Event Log if it is enabled, and if either the Enable bit in PRSCTL is 0 or if WQ PRS Disable in WQCFG is 1.[1] If Event Log is not enabled, the Drain descriptor completion record is written to the completion record address.

5.9.1 Event Log Entry

The format of each entry is shown below. Any bytes in an Event Log entry beyond those specified here, and up to the entry size specified in GENCAP are reserved.

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td>▨</td><td colspan="2">PASID</td><td>Operation</td><td>Batch ID</td><td>WQ Index</td><td>Error</td><td>Flags</td><td>0</td></tr>
<tr><td colspan="4">Error Information</td><td colspan="2">▨</td><td colspan="2">Batch Index</td><td>8</td></tr>
<tr><td colspan="8">Event Log Fault Address</td><td>16</td></tr>
<tr><td colspan="8">▨</td><td>24</td></tr>
<tr><td colspan="8" rowspan="4">Completion Record</td><td>32</td></tr>
<tr><td>40</td></tr>
<tr><td>48</td></tr>
<tr><td>56</td></tr>
<tr><td colspan="8" rowspan="2">Reserved</td><td>...</td></tr>
<tr><td>N-8</td></tr>
</table>

Figure 5-1: Event Log Entry

Byte Offset	Bits	Size	Description
7:0	63:60	4 bits	Unused.
	59:40	20 bits	PASID The PASID field of the descriptor that caused the error.
	39:32	8 bits	Operation The Operation field of the descriptor that caused the error.
	31:24	8 bits	Batch Identifier Identifier used to correlate Event Log entries for descriptors in a batch with the Event Log entry for the Batch descriptor. This field is valid if the Descriptor Valid field is 1 and either the Batch Member field is 1 or the Operation field is 0x01 (Batch operation). Otherwise, this field is unused.
	23:16	8 bits	WQ Index Indicates which WQ that the descriptor was submitted to.
	15:8	8 bits	Error code See section 5.8 for the meaning of the value in this field.

[1] In some implementations, the completion for a Drain descriptor may be written to the Event Log even if PRS is not enabled.

<table>
<tr><th>Byte Offset</th><th>Bits</th><th>Size</th><th>Description</th></tr>
<tr><td></td><td>7</td><td>1 bit</td><td>Error Information Valid
0: Error Information field is valid only if the Error code is Invalid Flags (0x11).
1: Error Information field is valid and provides additional information pertaining to the error reported in the Error code field.</td></tr>
<tr><td></td><td>6</td><td>1 bit</td><td>Priv
The Priv field of the descriptor that caused the error.</td></tr>
<tr><td></td><td>5</td><td>1 bit</td><td>R/W
If the error is a page fault, this indicates whether the faulting access was a read or a write.
0: The faulting access was a read.
1: The faulting access was a write.
Page faults are indicated by error codes 0x03, 0x04, 0x06, 0x1a, and 0x1f. For other error code values, this field is unused.</td></tr>
<tr><td></td><td>4</td><td>1 bit</td><td>Batch Member
0: The descriptor was submitted directly.
1: The descriptor was submitted in a batch.</td></tr>
<tr><td></td><td>3</td><td>1 bit</td><td>WQ Index Valid
0: The WQ that the descriptor was submitted to is unknown. The WQ Index field is unused.
1: The WQ Index field indicates which WQ the descriptor was submitted to.</td></tr>
<tr><td></td><td>2</td><td>1 bit</td><td>Descriptor Valid
0: The descriptor that caused the error is unknown. The Batch Member, Operation, Batch Index, Priv, and PASID fields are unused.
1: The Batch Member, Operation, Batch Index, Priv, and PASID fields are valid.</td></tr>
<tr><td></td><td>1:0</td><td>2 bits</td><td>Unused.</td></tr>
<tr><td>15:8</td><td>63:32</td><td>32 bits</td><td>Error Information
This field reports additional information for the error codes listed below. Otherwise, this field is unused.
<table>
<tr><th>Error code</th><th>Error information</th></tr>
<tr><td>Invalid Flags (0x11)</td><td>63:32 – A bitmask of the flags that were found to be invalid. If a bit in this field is 1, it indicates that the flag at the corresponding bit position in the Flags field of the descriptor was invalid.</td></tr>
<tr><td>Invalid Interrupt Handle (0x19)</td><td>63:52 – Unused.
51 – First Error in Batch.
50 – Completion Record Required.
49 – Portal. 0: MSI-X portal; 1: IMS portal.
48 – Completion Interrupt Required. (Always 1.)
47:32 – Interrupt handle</td></tr>
</table></td></tr>
</table>

Byte Offset	Bits	Size	Description	
			Page Fault (0x03, 0x04)	63:61 – Operand Identifier. See Table 8-5 for a description of this field. For Inter-domain operations: 60:48 – Unused. 47:32 – The IDPT handle used with the faulting address. For other operation types, bits 60:32 are unused.
			Page Fault (0x06, 0x1f)	63:61 – Operand Identifier. See Table 8-5 for a description of this field. 60:32 – Unused.
			Page Fault (0x1a)	63:61 – Operand Identifier. See Table 8-5 for a description of this field. 60:52 – Unused. 51 – First Error in Batch. 50 – Completion Record Required. (Always 1.) 49 – Portal. 0: MSI-X portal; 1: IMS portal. Undefined if Completion Interrupt Required is 0. 48 – Completion Interrupt Required. 47:32 – Interrupt handle (if bit 48 is 1).
			Page Fault before Drain or in a batch (0x26, 0x27)	63:51 – Unused. 50 – Completion Record Required. 49 – Portal. 0: MSI-X portal; 1: IMS portal. Undefined if Completion Interrupt Required is 0. 48 – Completion Interrupt Required. 47:32 – Interrupt handle (if bit 48 is 1).
			Inter-Domain operation error (0x29-0x2c)	63:48 – Unused. 47:32 – The IDPT handle that caused the error.
	31:16	16 bits	Unused.	
	15:0	16 bits	Batch Index If the Descriptor Valid field is 1 and the Batch Member field is 1, this field contains the index of the descriptor within the batch. Otherwise, this field is unused.	
23:16	63:0	64 bits	Error Log Address If error code is 0x03, 0x04, 0x06, or 0x1f, this is the faulting address. Bits 11:0 may be reported as 0. If error code is 0x1a, 0x26, or 0x27, this is the completion record address, and all bits of the address are reported. Otherwise, this field is unused.	
31:24	63:0	64 bits	Unused.	

Byte Offset	Bits	Size	Description
63:32	255:0	256 bits	Completion Record If error code is 0x19, 0x1a, 0x26, or 0x27 and Completion Record Required is 1, this field contains the completion record for the operation. Otherwise, this field is unused.

Table 5-9 : Event Log Entry Format

§

6 Performance Monitoring

The purpose of the Intel DSA performance monitoring capability (perfmon) is to support collection of information about key events (architectural or micro-architectural) occurring during device execution, to aid performance tuning and debug. This can also be useful to understand usages of key features and operations supported by the device. The perfmon architecture comprises three parts:

- Ability to discover and enumerate perfmon capabilities supported by a given Intel DSA implementation.
- Set of configuration and data registers to enable and configure the device to monitor a subset of supported events.
- List of events and filters supported.

Details of the registers used to enumerate and configure the various perfmon capabilities can be found in section 9.2.25.

6.1 Perfmon Discovery and Enumeration

Software reads a set of capability registers to discover whether the device supports the perfmon capability, and if so, to enumerate details of the capability such as number of counters, counter width, event categories, filter support, etc. If an implementation does not support performance monitoring, then the performance capability (PERFCAP) register is reported as 0 and the other perfmon capability, configuration, and data registers are not supported.

Software configures a counter to monitor events by specifying two pieces of information: an Event Category and an Events field, in the counter configuration register (CNTRCFG). The Intel DSA perfmon architecture defines a set of architectural Event Categories along with a set of implementation-specific events for each Event Category. The Event Categories are defined in Table 6-1. Additional Event Categories may be added in future implementations.

Corresponding to each Event Category defined, there is an event capability register (EVNTCAP) to report the set of events supported in that category. In case an implementation does not support any events for a given Event Category, then the Events field in EVNTCAP is reported as zero, and software should not attempt to configure any counter with that category. Details of the events corresponding to each Event Category are in Appendix D. When enabling a counter to count events of a given category, bits corresponding to events not supported in the specified event category are ignored. Software should not rely on this behavior since a future implementation may support additional events in an event category, resulting in bits that are ignored in one implementation having a defined meaning in a different implementation.

The mapping of Event Categories and Events to Event Counters may be implementation specific. An implementation may allow any event to be counted by any counter. However, an implementation may also choose to restrict this by allowing only certain Event Categories to be counted by each Event Counter. In such cases, software can consult the per-counter capability register (CNTRCAP) to discover the Event Categories and sets of events supported by each counter.

Value	Event Category	Description
0	WQ	Specifies events pertaining to work submission to a shared or dedicated work queue.
1	Engine	Specifies events pertaining to dispatch of work descriptors from the WQs and execution of the descriptors in the corresponding engines.
2	Address Translation	Specifies events pertaining to address translation when processing descriptors (including ATS/PRS and invalidation related events).
3	Operations	Counts operations of a specified type.
4	Completions	Specifies events related to descriptor completion and interrupt generation.
5	Inter-Domain Operations	Counts inter-domain operations of a specified type.
6-15	Reserved	Reserved for future use.

Table 6-1: Event Categories

6.2 Perfmon Configuration Registers

When perfmon is supported, there is a set of configuration, status, and data registers that software can use to configure and control the perfmon hardware. This includes a set of global configuration and status registers, as well as per-counter configuration and data registers documented in section 9.2.25. Software can reset the state of all the supported counter control and data registers to the default initial values using the Reset Perfmon Configuration and Reset Perfmon Counter controls in PERFRST (see section 9.2.25.4). This may be done at any time. Reset of Perfmon Configuration results in all the counters being Disabled.

Each set of counter configuration and data registers operates independently, and software must configure and enable a counter before that counter can begin counting events. Software should program the Filter Configuration (FLTCFG) registers (see section 6.4) before enabling the corresponding counter configuration (CNTRCFG) register. While the Enable bit in CNTRCFG is set, writes to FLTCFG or other fields of CNTRCFG are ignored, and event monitoring continues as if the write did not happen. Writes to the counter data (CNTRDATA) registers are handled as described in section 6.3.

Software configures event counting by selecting an available counter register and programming the appropriate Event Category value and set of events to be monitored in the Events field in the corresponding counter configuration register (CNTRCFG). The device interprets the bits programmed in the Events field as corresponding to the specified Event Category. Hence, at any time, a given counter can only count events corresponding to a single Event Category. Software can configure a given counter to count multiple events belonging to the same Event Category by setting multiple bits in the Events field. In this case, the counter value reflects the sum of all occurrences of the specified events. If independent (non-additive) counts are required for some events, software needs to program different counters; one per event to be monitored.

Similarly, to count events corresponding to different Event Categories, software needs to configure multiple counters; at least one per Event Category desired, and with the corresponding Events value.

6.3 Event Counters

The perfmon architecture supports up to 32 event counters. The number of counters in a given implementation may be less than this. Software can discover the number of counters in a given implementation by reading the PERFCAP register. Each counter operates independently.

Software can read the counter data registers (CNTRDATA) at any time. Software can write to a CNTRDATA register prior to enabling the counter. If the Counters Writeable While Enabled field in PERFCAP is 1, writes to a CNTRDATA register are also allowed while the counter is enabled. Some usages of software writing to a counter data register include:

- Write to a counter while it is disabled to initialize the counter with a specific value.
- Write to a counter after it overflows to re-initialize the counter. Note that the counter may be enabled and currently counting (not frozen).
- Write to a currently frozen counter to re-initialize it.

The supported width of the counter data registers is discoverable by reading the capability (PERFCAP) register. While this may be less than 64 bits in a given implementation, software is still allowed to write a full 64-bit value to the register without causing an error to be triggered. However, bits above the specified width are ignored by hardware. If per-counter capability reporting is supported (as indicated by PERFCAP), then the supported width of each counter is the value specified in the corresponding CNTRCAP register. This allows an implementation to support counters of different widths.

When multiple events are enabled in a given counter register, if multiple events occur simultaneously, then the counter value is incremented by more than 1 in a given cycle.

6.3.1 Counter Overflow

While enabled to count events, if an event occurrence causes the counter value to increment and roll over to or past zero, this is termed as a counter overflow. Upon overflow, the corresponding bit in the overflow status register (OVFSTATUS) is set. If supported and enabled, an interrupt may also be generated (details below). Normally, the counter continues to count events and does not stop counting upon overflow. If supported, software can specify the Global Freeze on Overflow bit in the counter configuration register. If this bit is set for a counter, an overflow of that counter results in the freeze bits of all counters to be set in PERFFRZ. This forces all the counters to stop counting (freeze) and retain their current count value (until explicitly written or reset by software). The current freeze state of the counters is reported in PERFFRZ.

As mentioned above, since the counter data registers may be software writeable, software can treat the data register as a signed integer up to the supported width. To cause an overflow on the first occurrence of an event, software can write an initial count value of -1 (e.g., 0xFFFFFFFF for a 32-bit counter) prior to enabling the event counting. This causes the first occurrence of the specified event, after enabling the counter, to cause an overflow of the counter value and trigger an interrupt (if enabled).

6.3.2 Counter Stop and Resume

Software can stop a counter that is enabled and counting events, by writing a 1 to the corresponding bit in the PERFFRZ register. This is referred to as a freeze operation on that counter and causes it to stop counting further events. Likewise, a counter that was previously frozen may be resumed by writing a 0 to the corresponding bit in PERFFRZ. This is referred to as an unfreeze operation on the counter and causes it to resume counting of configured events. When unfrozen, the counter continues to increment, starting from the current counter value at the time of the unfreeze operation.

Current freeze/unfreeze status of a counter is reported in the PERFFRZ register. These bits are Readable and Writeable by software. Additionally, hardware sets the freeze bits of all counters when any of the counters that has the Global Freeze on Overflow bit set encounter an overflow.

6.4 Filter Support

The perfmon architecture allows software to specify a set of filters that can be used to constrain the counting of selected events based on one or more conditions specified in the Filter Configuration registers. When supported, there is a set of architecturally defined Filters as described in Table 6-2 and a corresponding set of Filter Configuration registers (one per Filter) for each perfmon counter. Software can discover support for filtering capability by querying the perfmon capability register (PERFCAP).

Each event might only support a subset of filter types or may not support filters at all. See Appendix D for information on which filters are allowed for each event. Software can specify one or more filters to apply to the events monitored by a given counter by programming the Filter Values in the corresponding Filter Configuration registers (FLTCFG) for that counter. An example use of filters might be to configure a counter to only count a specific event, e.g., number of drain descriptors (specified via the Event_Category and Events fields), from only a specific WQ (filter).

Software is allowed to specify multiple filters for a given counter. When multiple filters are configured for a counter, only the events that satisfy all the specified filters (i.e., logical AND of all the filter conditions) will be counted. See section D.3 for examples.

6.5 Event Programming Considerations

As mentioned in section 6.2, software can configure an event counter to count multiple events belonging to the same Event Category, by setting multiple bits in the Events field of the corresponding CNTRCFG register. To get meaningful event counts, software should ensure that when multiple events are to be monitored by a counter, the events are related in some way. For example, configuring a counter to count both number of cycles and number of operations may not be desirable. Similarly, all events within an Event Category may not support the same set of filters. Software should ensure that the filter values specified are compatible with the set of events configured for that counter. Not doing so may produce undesirable event counter values. Hardware does not perform error checks when programming the performance monitoring registers, and the onus is on software to ensure meaningful configuration.

6.6 Interrupt Generation

If the Interrupt on Overflow Support field in PERFCAP is 1, then the implementation supports generation of an MSI-X interrupt (using entry 0 of the MSI-X table) upon counter overflow. Software can use this facility to be notified when a counter overflows.

<table>
<tr><th>Filter</th><th>Encoding</th><th>Filter Value</th></tr>
<tr><td>WQ</td><td>0</td><td>Bitmask to select WQs to monitor.
(Bit 0 for WQ0, Bit 1 for WQ1, etc.)</td></tr>
<tr><td>Traffic Class (TC)</td><td>1</td><td>Bitmask to select which TCs to monitor.
(Bit 0 for TC0, Bit 1 for TC1, etc.)</td></tr>
<tr><td>Page Size</td><td>2</td><td>Bitmask to select which Page Sizes to monitor
<table>
<tr><td>Bit</td><td>Filter Value</td><td>Description.</td></tr>
<tr><td>0</td><td>0x1</td><td>4K</td></tr>
<tr><td>1</td><td>0x2</td><td>2M</td></tr>
<tr><td>2</td><td>0x4</td><td>1G</td></tr>
</table></td></tr>
<tr><td>Transfer Size</td><td>3</td><td>Bitmask to select range of transfer size values to monitor.
<table>
<tr><td>Bit</td><td>Filter Value</td><td>Description</td></tr>
<tr><td>0</td><td>0x1</td><td>0 ≤ size &lt; 512B</td></tr>
<tr><td>1</td><td>0x2</td><td>512B ≤ size &lt; 2KB</td></tr>
<tr><td>2</td><td>0x4</td><td>2KB ≤ size &lt; 4KB</td></tr>
<tr><td>3</td><td>0x8</td><td>4KB ≤ size &lt; 16KB</td></tr>
<tr><td>4</td><td>0x10</td><td>16KB ≤ size &lt; 1MB</td></tr>
<tr><td>5</td><td>0x20</td><td>1MB ≤ size &lt; 64MB</td></tr>
<tr><td>6</td><td>0x40</td><td>64MB ≤ size &lt; 1GB</td></tr>
<tr><td>7</td><td>0x80</td><td>1GB ≤ size &lt; 4GB</td></tr>
</table></td></tr>
<tr><td>Engine Number</td><td>4</td><td>Bitmask to select which Engines to monitor.</td></tr>
<tr><td>PASID</td><td>5</td><td>Select which submitter PASID to monitor:
Bits 21:20: Filter mode.
Bits 19:0: PASID value.

Filter mode:
0: Match requests with specified submitter PASID value.
1: Match requests with any PASID.
2: Match requests with no PASID.
3: Match any request. (This is the default.)</td></tr>
</table>

Table 6-2: Filter Types and Mask

Also, the INTCAUSE register indicates that a perfmon counter overflow caused the interrupt to be generated. Upon receiving the interrupt, software can read the global status register (OVFSTATUS) to identify which counters overflowed. It is possible for multiple bits to be set in this register (indicating multiple counter overflows). If Global Freeze on Overflow is enabled for the counter, software can check the current freeze state for all the counters in PERFFRZ and read the corresponding counter values from the CNTRDATA registers.

§

7 Reference Software Architecture

Software support for Intel DSA is expected to include the following elements:

- Kernel mode driver.
- User mode driver.
- Virtualization support.

7.1 Kernel Mode Driver

The Intel DSA kernel-mode driver (KMD) is responsible for initializing and managing the device. It can plug into the kernel DMA subsystem and provide services to any client using the internal OS-specific DMA APIs. It also exposes an interface to user space to support direct user level access for SVM services. KMD requests that the OS allocate/bind/unbind/free PASIDs based on user level requests. It maps limited portals to clients to allow them direct access for work submission.

For Shared WQs, KMD sets WQ Threshold to control how much of the WQ capacity is available for the limited portal. The remainder of the WQ capacity is reserved for work submission to the unlimited portal. KMD may change the threshold at any time. The threshold may be set to the WQ Size to not reserve any space and it may be set to 0 to prevent any work submission to the limited portal. When a limited portal returns Retry, the client can request that KMD submit work to the unlimited portal on its behalf. If the unlimited portal also returns Retry, KMD may reattempt the submission or take the following steps:

1. Reduce the threshold to prevent direct work submission.
2. Enable the WQ Occupancy interrupt to receive notification when there is space in the WQ.
3. When the notification arrives, submit the work that has been queued.
4. Restore the threshold.

In performing these steps, KMD may need to take care that descriptors are submitted to the device in the same order that they were attempted by the client, if the client relies on descriptors being executed in order. (See section 3.9 for information about descriptor ordering.)

7.2 User Mode Driver

The Intel DSA user-mode driver (UMD) is an optional component that is used to provide user-mode access to the device. UMD is used to make Intel DSA functions available to applications. It is linked with an application as a library and interfaces with the kernel-mode driver to request access to the device on behalf of the application. It exposes various device functions to the application by abstracting them in higher level APIs. It normally services application requests using ENQCMD to a limited portal. If the ENQCMD fails due to congestion, UMD may back off and retry the work submission or use a kernel-mode driver service to proxy the request to ensure forward progress. Additionally, UMD can service application requests using MOVDIR64B to a dedicated work queue portal.

7.3 Software Requirements for Handling Non-Blocking Page Faults

While PRS is disabled, Intel DSA handles page faults by stopping the operation and reporting a partial completion status in the completion record (described in Section 3.13). Additionally, page faults on completion record addresses are reported through the Event Log while it is enabled. KMD is responsible for configuring and enabling the Event Log. It must ensure that the Event Log is sufficiently large and must process entries in a timely manner to avoid subsequent Event Log writes from the device getting blocked due to log full conditions.

When processing an Event Log entry for a page fault on a completion record, KMD is expected to do the following:

1. If the First Error in Batch flag is 1, discard any previously recorded errors associated with the Batch Identifier. This can happen when a Batch completion is lost because of an Abort command or an internal hardware error. In the usual case, no errors will be recorded, and no action needs to be taken.
2. Attempt to fix the page fault corresponding to the Fault Address and PASID reported in the Event Log entry, and if successful, write the completion record to the Fault Address and generate the completion interrupt, if the Completion Interrupt Required field in the Event Log entry is 1.
3. If there is an error writing the completion record and the completion record is for a descriptor in a batch, KMD associates the error with the Batch Identifier of the Event Log entry and tracks it until the Event Log entry for the corresponding Batch descriptor is observed. KMD does not need to track successfully written completion records.

When processing an Event Log entry for a Batch descriptor with the error code indicating that one or more descriptors in the batch had Event Log entries and the Completion Record Required field in the Event Log entry is 1, KMD does the following before writing the Batch completion record to memory provided an error has been recorded with a matching Batch Identifier:

1. If the Status field of the completion record within the event log entry is 0x01, KMD changes it to 0x05 and changes the Result field to 1.
2. If Status is 0x06, change Result to 1.
3. If Status is any other value, the completion record should not be changed.
4. KMD should then clear the recorded error in preparation for the next batch with the same Batch Identifier.

If no error has been recorded for this batch (all completion records were written successfully), KMD writes the Batch completion record as-is. Software should then generate the completion interrupt, if the Completion Interrupt Required field in the Event Log entry is 1.

When an application or UMD receives a completion record indicating partial completion, it can choose to fix the page fault and resubmit a descriptor to the device to complete the remainder of the operation. In most cases, the original descriptor may need to be updated to adjust the Transfer Size field based on the amount of work already completed. For certain operations, additional updates to the original descriptor may be required. If Batch Continuation Support in GENCAP is 1, when resubmitting a Batch descriptor that was terminated early (as described in Section 8.3.2), software can set the Batch Error flag in the descriptor based on the Result field in the completion record to ensure that the Status and Result fields in the final completion record for the batch reflect the correct status across all the descriptors in the batch.

7.4 Software Requirements for Inter-Domain Operations

As described in section 3.13.3, each inter-domain or Update Window operation specifies at least one IDPT handle that references an entry in the Inter-Domain Permissions Table on that device. The IDPT controls the connection and communication between the different address spaces (PASIDs). Some of the key software considerations when using inter-domain or Update Window operations are as follows:

- Privileged software (e.g., KMD) is responsible for discovery and enumeration of inter-domain capabilities, and configuration of the IDPT in each device.
- For an inter-domain or Update Window operation to be successful, an IDPTE must be set up before the descriptor is submitted to the device.
- An IDPT is specific to a device; hence to use a given IDPTE, the owner and submitter processes (described in section 3.13.3) must subscribe to the same physical device.
- The owner and submitter may use the same SWQ or different SWQs/DWQs on the device. The owner does not need to have access to a WQ on the device if it does not need to use the Update Window operation.
- For a type 0 SASS and type 1 SAMS IDPTE:
    - The owner typically requests the KMD through a system call interface (e.g., ioctl) to set up an IDPTE of the appropriate type, with the necessary window parameters.
    - It is the responsibility of the owner to communicate the system file handle representing the IDPTE, IDPT handle and window parameters to the submitter. The exact mechanism used to exchange this information depends on the software implementation.
    - A software implementation may require each submitter to explicitly register itself with the KMD using a system call interface (e.g., ioctl) prior to first use of an IDPT handle in an inter-domain descriptor. Subsequent uses of the IDPT handle in inter-domain descriptors may be done without involving the KMD.
    - If the Allow Update field in the IDPTE is 1, the owner can issue an Update Window descriptor to modify window attributes of the IDPTE. It is the responsibility of software to coordinate changes to IDPTE window attributes with any concurrent access to the window by a submitter.

An example software sequence showing the communication between an owner, submitter, KMD, and hardware is shown in Figure 7-1.

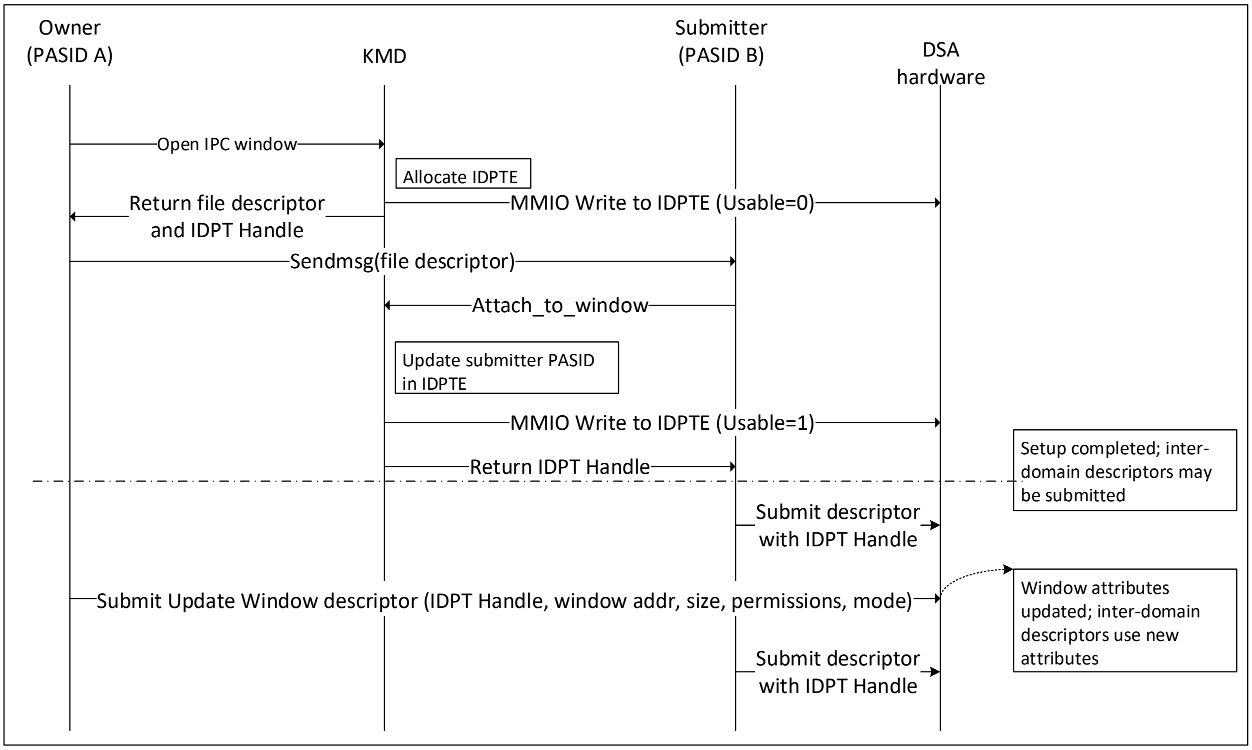

Owner
(PASID A)
KMD
Submitter
(PASID B)
DSA
hardware
Open IPC window
Allocate IDPTE
Return file descriptor
and IDPT Handle
MMIO Write to IDPTE (Usable=0)
Sendmsg(file descriptor)
Attach_to_window
Update submitter PASID
in IDPTE
MMIO Write to IDPTE (Usable=1)
Return IDPT Handle
Setup completed; inter-domain descriptors may be submitted
Submit descriptor
with IDPT Handle
Submit Update Window descriptor (IDPT Handle, window addr, size, permissions, mode)
Window attributes updated; inter-domain descriptors use new attributes
Submit descriptor
with IDPT Handle

Figure 7-1: Example Software Flow for an Inter-Domain Operation

7.5 Virtualization Software

Intel DSA is virtualized using the Intel Scalable IOV model, described in the Intel® Scalable I/O Virtualization Architecture Specification. The virtualization software architecture is shown in Figure 7-2. Virtualization of the device is supported by a software component called the Intel DSA Virtual Device Composition Module (VDCM), which composes a virtual Intel DSA device and exposes it to the guest. The VDCM is a VMM specific module and is responsible for communicating with the VMM to facilitate device virtualization. Depending on the host system software architecture, the VDCM may be developed as a user level module, as part of the kernel-mode driver, as a separate kernel module, or as part of the VMM.

The KMD in the Host OS is extended to support the VDCM operations required for virtualization. The KMD with virtualization extensions is called the host driver. The KMD in the Guest OS may run exactly like in a non-virtualization environment or it may be optimized to run in a VM. The KMD in the guest OS is called the guest driver. The host driver controls and manages the physical device and allows sharing of the device among multiple guest drivers. A single Intel DSA driver per OS may be developed to work in the non-virtualized OS, Host OS, and Guest OS.

7.5.1 Virtual Intel® DSA Device

The virtual device implemented by the VDCM, called VDEV, emulates the same interface as the physical Intel DSA device, so that the same device driver can run in both the host OS and the guest OS. The guest driver accesses the virtual device through MMIO registers using the same software interface as the physical device. The VDCM emulates the behavior of the virtual device and mediates guest subscription of the device through the host driver. Control path operations on the VDEV from the VM (e.g., dedicated WQ configuration) are trapped by the VMM and emulated by the VDCM, but fast path operations (descriptor submission and descriptor completion) are directly mapped to the VM.

Within a guest, some features of the device may not be supported. The capability registers indicate to the guest which features are available. For example, the number of work queues or groups available in the virtual device may differ from the number available in the physical device. Another example of a feature that may not be supported is interrupt message storage.

Some aspects of Group and WQ configuration are not modifiable by the guest, indicated by the Configuration Support capability in GENCAP. For example, the size of each WQ must be configured by the host driver before starting the device and may not be changed by a guest that the WQ is subsequently assigned to. To indicate this, the VDCM should always return the value 0 in the Configuration Support field in the GENCAP register of the VDEV.

If a WQ is assigned to multiple guests, it is configured as a Shared WQ by the host driver. None of the WQ configuration registers for such a WQ can be changed by the guest driver. This is indicated to the guest by the value 0 in the WQ Mode Support field of the WQCFG register.

If a WQ is assigned to a single guest, the guest driver may decide whether it is to be a Dedicated WQ or a Shared WQ. In this case, the guest driver may also configure the WQ Threshold, Priv, PASID Enable, and PASID. This is indicated to the guest by the value 1 in the WQ Mode Support field of the WQCFG register. See Table 9-7 for details of WQ configuration support.

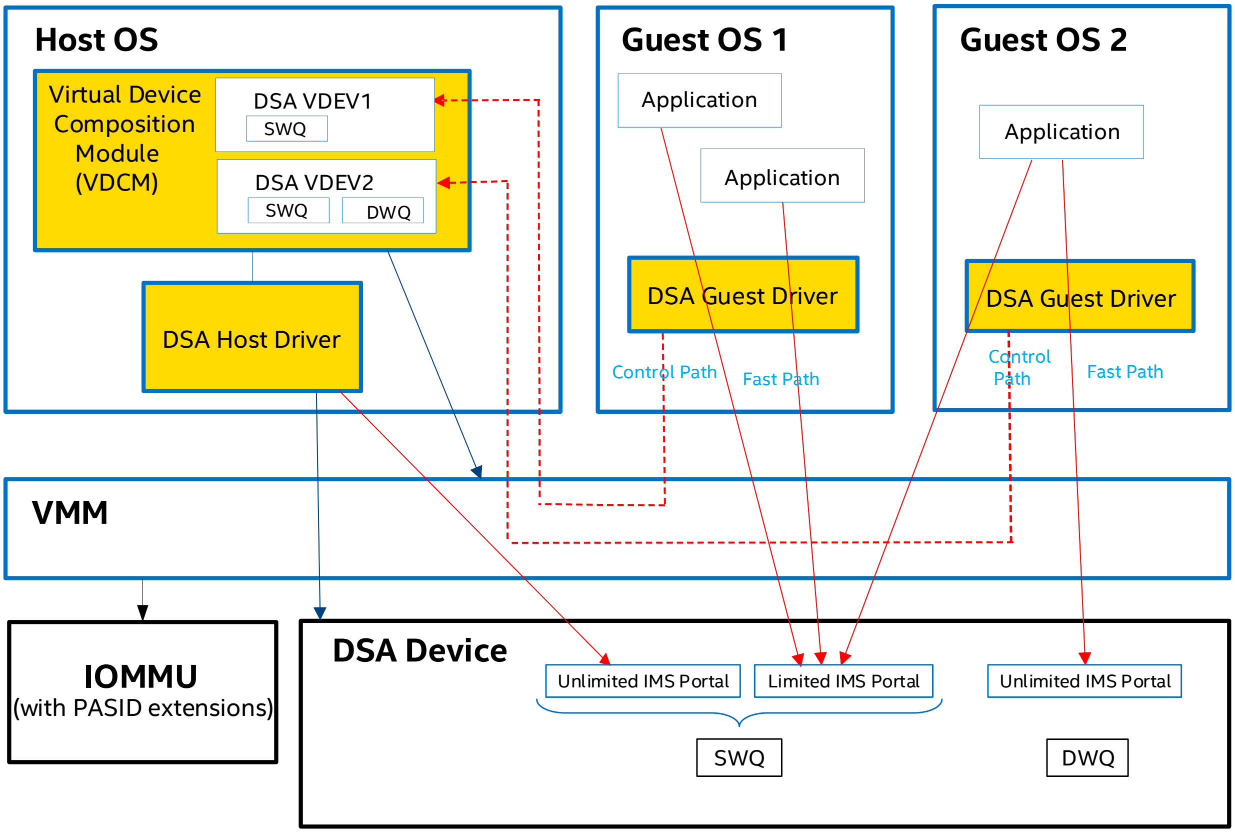


Figure 7-2: Intel® Scalable IOV for Intel® DSA

7.5.2 Portal Virtualization

For each WQ included in a VDEV, the VDCM directly maps some of the WQ's physical portals into the VM. For a WQ shared by multiple guests, the host driver retains control of the unlimited portal, and the VDCM maps only the limited portal into the guest. When a guest submits a descriptor using its unlimited portal address (after the guest's limited portal has returned Retry), the VMM traps on the portal write and the host driver submits the descriptor using the physical unlimited portal to provide forward progress to the guest. If the physical unlimited portal also returns Retry, the host driver may use the same approaches described in section 7.1.

For a WQ assigned to a single guest, the VDCM should map both the limited portal and the unlimited portal. That way, if the guest driver chooses to configure the WQ as a Shared WQ, it can set the WQ Threshold and manage forward progress assurance on the WQ itself by mapping the limited portal directly into its user-mode clients and using the unlimited portal for kernel-mode operations.

Figure 7-2 shows that VDCM has created VDEV1 for Guest 1 with one shared WQ (SWQ) and VDEV2 for Guest 2 with one SWQ and one dedicated WQ (DWQ). Guest 1 and Guest 2 share the same SWQ in the device. The DWQ can be assigned to only one VM. The corresponding SWQ and DWQ portals are directly mapped into the respective VMs for fast path operations. For the SWQ, the same limited portal is mapped into both VMs.

The VDCM maps only IMS portals into the guest. The MSI-X portals are reserved for host use. If the virtual device visible to the guest does not report support for IMS, the IMS portals are mapped into the guest's virtual BAR2 in place of the MSI-X portals and the dummy portal (described in section 9.2.21) may be mapped into the address ranges corresponding to the guest's virtual IMS portals. See section 7.5.4 for a description of interrupt virtualization.

7.5.3 SVM and PASID Virtualization

When a virtual Intel DSA device is assigned to a VM, all WQs used by the VM must be configured to use PASID. The VMM allocates a default Host PASID for the VM and configures the PASID table entry for that PASID in the IOMMU for second level address translation (GPA → HPA). This PASID is used when the guest configures a virtual WQ in dedicated mode with PASID disabled. For the guest to use the virtual device in this way, the VDEV need not support the PASID, ATS, and PRS PCIe capabilities (even though these capabilities are enabled in the physical Intel DSA device).

To support SVM in the guest, the VDEV includes support for the ATS, PASID, and PRS capabilities, and the VMM exposes a virtual IOMMU to the guest. The guest OS sets up PASID table entries in the virtual IOMMU's PASID table. Since guest software uses Guest PASIDs and the physical device uses Host PASIDs, the VMM must manage Guest PASID to Host PASID mapping.

Some VMMs may choose to use a para-virtualized or enlightened virtual IOMMU where the guest doesn't generate its own Guest PASIDs but instead requests Guest PASIDs from the virtual IOMMU. In this case, the VMM may use the same value for the Guest PASID as for the Host PASID for each requested Guest PASID, simplifying PASID management in the VMM. Otherwise, the guest OS allocates its own Guest PASIDs for its SVM operations and the VMM must allocate a Host PASID for each Guest PASID.

The method for setting up a Guest PASID to Host PASID mapping depends on whether the WQ is in dedicated or shared mode. If a WQ is assigned to a single VM, the guest driver can decide whether to configure it as a DWQ or as an SWQ to be shared across multiple applications within the VM. If a WQ is assigned to multiple VMs, then it is configured as an SWQ by the host driver and the guest cannot change the WQ Mode.

When the guest driver enables a WQ in dedicated mode with the WQ PASID Enable field in the VDEV equal to 1, the VMM creates a mapping for the Guest PASID in the WQ PASID field. If the WQ PASID Enable field is 0, the VMM uses the VM's default Host PASID. In either case, the host driver writes the proper Host PASID to the WQ PASID field of physical WQCFG register and writes 1 to the WQ PASID Enable field.

If a WQ is configured in shared mode, by either the host driver or the guest driver, the VMM enables the PASID Translation VMX execution control in the VMCS (VM Control Structure). The guest uses the ENQCMD or ENQCMDS instructions to submit descriptors. On the first submission for a Guest PASID, ENQCMD/S causes a VM exit since the PASID translation table doesn't have a mapping for the Guest PASID, and the VMM creates a mapping for it.

To create a mapping for a Guest PASID, the VMM looks at the PASID table entry for the Guest PASID in the virtual IOMMU's PASID table. If the Guest PASID is configured for first-level translation in the virtual IOMMU, the VMM allocates a new Host PASID, configures its PASID table entry for nested first-level (GVA to GPA) and second-level (GPA to HPA) translations, and sets up the VMCS PASID translation table to map the Guest PASID to the Host PASID. If the Guest PASID is not configured in the virtual IOMMU, the VMM sets up the VMCS PASID translation table to map the Guest PASID to the VM's default Host PASID, which is already configured in the physical IOMMU.

7.5.4 Interrupt Virtualization

The VDCM virtualizes interrupts by exposing a virtual MSI-X capability in the VDEV. The Interrupt Message Storage Support field in the Scalable IOV Capability in the VDEV may be 0. The VDCM requests that the host driver allocate an entry in the Interrupt Message Storage for each interrupt available to the VM. The VDCM maps the Limited IMS Portal for each WQ into the VM at the offset of both the Unlimited MSI-X Portal and the Limited MSI-X Portal. When the guest uses its MSI-X portal address to submit descriptors, it is actually using the physical IMS portal, so that guest interrupts are always generated using the IMS.

When the guest OS configures a virtual MSI-X entry, the VDCM or the host driver requests that the Host OS or VMM allocate a physical interrupt and program it into the IOMMU's interrupt posting structure using the vector and VCPU information from the virtual MSI-X table entry. The Host OS or VMM passes the physical interrupt address and data value to the host driver, which is responsible for configuring the physical interrupt into the allocated Interrupt Message Storage entry, including setting the IMS PASID field to the PASID of the guest.

The Command Capabilities register in the VDEV indicates support for the Request Interrupt Handle command, requiring the guest to use the Request Interrupt Handle command to obtain an interrupt handle associated with each MSI-X table entry. The VDCM responds to the command with the index in the IMS corresponding to the virtual MSI-X table entry. The guest places the interrupt handle in each descriptor that requests an interrupt. The physical device uses the handle to identify the Interrupt Message Storage entry to be used to generate the completion interrupt. It checks the PASID of the descriptor against the PASID field in the IMS entry. If a guest requests an interrupt using an interrupt handle that has not been assigned to it, the PASID won't match, so the interrupt will not be generated.

When migrating a VM or resuming a VM after it has been suspended, interrupt handles that were allocated to the VM may no longer be available. To inform the guest that one or more interrupt handles have been revoked, the VDCM sets the Interrupt Handles Revoked bit in the virtual INTCAUSE register and generates an interrupt to the guest, using MSI-X entry 0 in the VDEV. The guest clears the Interrupt Handles Revoked bit and then uses the Request Interrupt Handle command to obtain new handles for any MSI-X and/or IMS entries that are in use. After ensuring that all threads have stopped using the revoked handles, the guest submits a Drain descriptor using each new interrupt handle. The Drain waits for completion of any descriptors that were submitted using the revoked handle; these descriptors complete with Operation Status 0x19, Invalid Interrupt Handle. Upon completion of the Drain descriptor, Intel DSA hardware generates the expected completion interrupt for these descriptors, so that the errors in the completion records are recognized by software and the descriptors can be resubmitted using the new handle. See Figure 7-2 for pseudocode of the steps to be performed in the guest.

When the guest writes the Ignore or Mask bits of the virtual MSI-X table, the VDCM writes the corresponding IMS table entry. When the guest reads the virtual MSI-X Pending bit array, the VDCM constructs the value from the values of the Pending bits of the IMS table entries assigned to that guest.

The VDCM should provide one additional MSI-X table entry, used for errors and command completions. The VDCM itself is responsible for generating virtual interrupts for these events using the vector and VCPU information in the virtual MSI-X table entry.

If the Interrupt Message Storage Support capability in the VDEV is 1, the IMS is virtualized in much the same way as MSI-X.

When a guest is destroyed, after its PASIDs are drained, the PASID Enable field should be cleared in all the IMS entries allocated to the guest, to ensure that those entries cannot be improperly used by another guest when the PASIDs are reassigned.

Submitter Thread(s) using intr table entry *idx*

```
atomic_inc(intr_handle_users(idx))
// Check for revoked interrupt handle
while ((temp = intr_handle(idx)) == REVOKED) {
   atomic_dec(intr_handle_users(idx))
   // Wait for new handle to be available.
   while (intr_handle(idx) == REVOKED)
      yield()
   atomic_inc(intr_handle_users(idx))
}
dsa_desc.intr_handle = temp
...
enqcmd(dsa_desc) or movdir64b(dsa_desc)
atomic_dec(intr_handle_users(idx))
```

Interrupt Handle Revocation Handler

```
Clear INTCAUSE.Interrupt_Handles_Revoked
for each idx in MSI-X table and IMS table {
   new_intr_handle = dsa_request_intr_handle(idx)
   if (new_intr_handle == intr_handle(idx))
      continue // Interrupt handle did not change
   intr_handle(idx) = REVOKED
   // Wait for submitters to complete submission
   // of any descriptors using the revoked handle.
   while (intr_handle_users(idx) ≠ 0)
      yield()
   intr_handle(idx) = new_intr_handle
   Drain_descriptor(new_intr_handle)
}
```

Figure 7-3: Guest Steps to Handle Interrupt Handle Revocation

7.5.5 Capability Virtualization

Intel DSA exposes its capabilities to software via capability registers, described in section 9.2. This enables VDCM to expose a subset of device capabilities to the VM through the virtual device's capability registers, allowing the virtual device to be compatible with multiple generations of devices. This capability virtualization enables a VM image with a guest driver to be started on or migrated to physical machines containing different generations of Intel DSA devices. This allows creation of pools of compatible physical machines in a datacenter where the same VM image can be started or migrated.

7.5.6 Virtualization of Inter-Domain Features

If the PASID capability is not enabled for a guest, the VDCM does not expose inter-domain capabilities to that guest. If the PASID capability is enabled for a guest, the VDCM virtualizes inter-domain operations by exposing a virtual IDPT capability in the VDEV. The VM allocates entries in the virtual IDPT and configures them using MMIO writes to the corresponding virtual IDPTE. The VDCM translates Guest PASID values in the virtual IDPTE into the corresponding Host PASID values and requests the host driver to allocate a physical IDPTE for each virtual IDPTE allocated by the VM. The Request IDPT Handle field in CMDCAP of the VDEV is 1, indicating that the guest must use the Request IDPT Handle command to obtain the IDPT handle associated with each IDPTE in the VDEV. The VDCM responds to the command with the index of the physical IDPTE corresponding to the virtual IDPTE. When a guest submits descriptors with the IDPT handle, it is actually using the physical IDPTE containing the Host PASID values associated with that guest. The guest KMD can maintain an association of the handle with the virtual IDPTE.

When migrating a VM or resuming a VM after it has been suspended, IDPT handles that were allocated to the VM may no longer be available. An inter-domain or Update Window descriptor submitted by the VM that references an unavailable IDPT handle completes with an error code as described in section 5.8.1. Upon receiving a completion with an invalid IDPT handle error, the application in the VM uses an appropriate system call interface (e.g., ioctl) to request a new IDPT handle from the guest KMD. The guest KMD uses the Request IDPT Handle command to obtain a new IDPT handle as mentioned above. The VDCM allocates a new entry in the physical IDPT and initializes the entry appropriately based on the values in the corresponding virtual IDPTE. The VDCM responds to the command with the index of the physical IDPTE as the new handle to use. The guest KMD provides the new handle to the application and may record the new handle as the one associated with the virtual IDPTE.

The VDCM may map the region of memory corresponding to the virtual IDPT into the VM using read-only mapping. When a guest writes to a virtual IDPTE, the VDCM writes the corresponding location in the DRAM backed region for the virtual IDPT and requests the host driver to write the corresponding values to the physical IDPTE. When the guest reads the virtual IDPTE, it reads the values directly from the DRAM region without a VM exit.

If the VDCM exposes a type 1 SAMS IDPTE to a guest, it must virtualize the submitter bitmap associated with that IDPTE. The VM allocates memory for the virtual bitmap and configures the virtual IDPTE with the bitmap address. The VDCM allocates memory for a corresponding physical bitmap and updates the physical IDPTE. The guest can read or write to the virtual bitmap region without a VM exit. The Invalidate Submitter Bitmap Cache capability in the VDEV is 1, indicating that the guest must use the Invalidate Submitter Bitmap Cache command after any changes to the bitmap to ensure that the updated values are used by the VDEV. When the VM issues an Invalidate Submitter Bitmap Cache command specifying the portion of the bitmap that has been updated, the VDCM updates the corresponding bits in the physical bitmap. If the guest fails to issue an Invalidate Submitter Bitmap Cache command, the physical bitmap will not be updated by the VDCM, and any subsequent descriptors referencing that IDPTE may not complete as intended.

When a guest is destroyed, after its PASIDs are drained, any IDPTEs assigned to that VM must be marked unusable to ensure that those entries cannot be improperly used by another guest when the PASIDs are reassigned.

7.5.7 State Migration During VM Migration

Intel DSA virtualization supports live migration of VMs. During the final phase of live VM migration, the VMM suspends the VM and then issues a suspend command to all the virtual devices of the VM and waits for suspend to complete. The VMM then saves the virtual device state, migrates it along with the rest of the VM state, and restores the virtual device state on the destination machine.

To suspend the virtual Intel DSA device, the VDCM requests that the host driver drain all the Host PASIDs assigned to the VM. The host driver issues a Drain PASID command for each assigned PASID or it may issue Drain All if a large number of PASIDs are assigned to the VM. After completion of the Drain commands, the virtual device reaches the suspended state. If there are pending interrupts for the VM in the interrupt posting structure of the IOMMU, they are delivered to the virtual APIC. The virtual device state is transferred to the destination machine along with the rest of the state of the VM.

On the destination machine, the VMM creates a new virtual Intel DSA device for the VM and restores the virtual device state to it. Specifically, it configures IMS entries for interrupts that are configured in the virtual MSI-X table, assigns physical WQs to the VM according to the virtual device configuration, and sets up the physical IOMMU for DMA remapping and interrupt remapping/posting. For a Dedicated WQ, the destination DWQ must be the same or larger size compared to the original DWQ since the guest driver may continue to use the old DWQ size. The capability virtualization described in section 3.16 ensures that the virtual device can work on multiple generations of Intel DSA devices.

See section 7.5.4 for a description of interrupt handle revocation after VM migration. See section 7.5.6 for a description of IDPT handle revocation after VM migration.

7.5.8 Virtualization of Event Log

The Event Log capability may be exposed to a guest independently of whether Event Log is enabled in hardware. When the Event Log capability is exposed to a guest, the guest may or may not enable the feature. If a guest does not enable Event Log, the VDCM must use the virtual SWERROR to report errors to the guest. If multiple errors are present to be injected into the guest, the VDCM may set the Overflow field in the virtual SWERROR register or it may report errors to the guest one at a time, waiting for the guest to clear each error in the virtual SWERROR before the VDCM reports the next one.

If a guest enables the Event Log, the VDCM must use it to report errors to the guest instead of using SWERROR. The remainder of this section covers the case when Event Log is enabled by both the host and the guest.

When a guest enables the Event Log in the virtual Intel DSA device, it allocates space for its event log in its own memory and writes the base address and size to the EVLCFG fields in the virtual Intel DSA device.

While the host driver is processing the Event Log, it determines for each event log entry whether it is specific to a guest. The VDCM translates the PASID, WQ Index, and portal identifier from the host event log entry to guest values, writes the modified event log entry into the guest event log in guest memory, updates the Event Log Tail field in the virtual EVLSTATUS register, and then generates an interrupt into the guest if required. This does not require a VM exit from the guest if the host event log processing is performed on a separate thread.

When the guest receives an event log interrupt, it reads the Event Log entries from its own memory. No VM exits are required to read the Event Log, but VM exits may be required to read Event Log Tail, write Event Log Head, and clear Interrupt Pending in the virtual EVLSTATUS register.

When writing event log entries into guest memory, the VDCM performs address translation using the fields in the virtual EVLCFG register and the translation tables configured in the guest's virtual IOMMU.

- If there is no virtual IOMMU in the guest or it is disabled, the Event Log Base Address in the virtual EVLCFG register is a guest physical address. The PASID Enable, PASID, and Priv fields in the virtual EVLCFG are ignored.
- If the virtual IOMMU is in legacy mode, then the Guest PASID Enable must be 0 in the virtual EVLCFG. Otherwise, writing to the guest event log causes a fault in the virtual IOMMU. (Refer to the Intel Virtualization Technology for Directed I/O Specification for details on the error code reported.)
- If the virtual IOMMU is in scalable mode,
    - If the Guest PASID Enable is 0, translation is performed using RID_PASID in the virtual IOMMU context entry.
    - If the Guest PASID Enable is 1, translation is performed using the PASID and Priv fields in the virtual EVLCFG register.

Handling Guest Event Log Full Condition

As described in section 5.9, when the physical hardware needs to write to the Event Log and the Event Log is full, hardware waits until software writes to the Event Log Head to make space available in the event log. This delays completion of any descriptors that cause a write to the event log and may also delay execution of subsequent descriptors.

As described in section 9.2.2, the Event Log Overflow Support field in GENCAP indicates whether the device drops new events if the Event Log is full or if it blocks until software has updated the Event Log Head.

A VDCM may report the Event Log Overflow Support field in the virtual GENCAP register as 0 and mimic the blocking behavior of the physical hardware. In this case, to avoid impacting other uses of the device, the host driver should continue to process the physical event log even if a guest's event log is full. As the host driver processes the physical event log and forwards entries into the appropriate guests' event logs, if a guest event log is full, the VDCM stores that guest's event log entries in an internal VDCM Event Queue associated with the guest. As the guest frees entries in the event log by moving the head, VDCM transfers event log entries from the VDCM Event Queue into the guest event log.

Alternately, a VDCM may configure the virtual hardware to report Event Log Overflow Support in the virtual GENCAP register as 1. In this case if the virtual Event Log is full, the VDCM may discard events pertaining to that virtual event log and log an error indicating the Event Log full condition in the virtual SWERROR register.

If an Event Log full error is logged in SWERROR, or if SWERROR reports an overflow while the Event Log is full, guest software must assume that one or more events may have been dropped and take appropriate steps to notify and/or terminate any impacted applications as needed. A VDCM may also choose to implement a fixed size VDCM Event Queue per guest to log events when the virtual Event Log is full and begin discarding events for a guest only if the corresponding Event Queue becomes full.

If the VDCM Event Queue is of a fixed size, the VDCM can size it appropriately to make an overflow situation less likely (based on descriptors in flight, with no additional descriptors being submitted), by setting the queue size as the sum of the following:

- Number of WQs × WQ size × (Maximum Batch Size + 1).
- Number of Engines × Maximum Batch Descriptors in Progress × (Maximum Batch Size + 1).
- Number of Engines × Maximum Work Descriptors in Progress.

For example, if a guest is assigned the following:

- 1 WQ in a group with 1 engine.
- WQ Size = 16.
- WQ Maximum Batch Size = 32.

VDCM Event Queue Size = 1 × 16 × (32+1) + 1 × 16 × (32+1) + 1 × 128 = 1184 entries or about 76 KB.

§

8 Descriptor Formats

8.1 Common Descriptor Fields

Intel DSA descriptors are 64 bytes. Some descriptor fields are common to all operation types and some fields are dependent on the operation type. This section describes the fields that are common to most operation types. The diagram for each operation type indicates which of the common fields are used for that operation type and what the operation-specific fields are.

Common fields include both trusted fields and untrusted fields. Trusted fields are always trusted by the device since they are populated by the CPU or by privileged (ring 0 or VMM) software on the host. The untrusted fields are directly supplied by client software.

Generic Descriptor Format

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source Address								16
Destination Address								24
		Completion Interrupt Handle		Transfer Size				32
Operation-specific fields								40
								48
								56

8.1.1 Trusted Fields

Offset: 0; Size: 4 bytes (32 bits)

When a descriptor is submitted to an SWQ, these fields carry the Privilege and PASID of the software entity that submitted the descriptor. When a descriptor is submitted to a DWQ, these fields in the descriptor are ignored; the device uses the WQ Priv and WQ PASID fields of the WQCFG register.

On Intel CPUs, when software submits a descriptor to an SWQ using ENQCMD, these fields in the source descriptor are reserved. The value of IA32_PASID MSR is placed in the PASID field and the Priv field is set to 0 before the descriptor is sent to the device. When software uses ENQCMDS, these fields in the source descriptor must be initialized appropriately by software. If the Privileged Mode Enable field of the PCI Express PASID capability is 0, the Priv field must be 0.

These fields are ignored for any descriptor in a batch. The corresponding fields of the Batch descriptor are used for every descriptor in the batch.

Bits	Description
31	Priv (User/Supervisor) **0:** The descriptor is a user-mode descriptor submitted directly by a user-mode client or submitted by the kernel on behalf of a user-mode client. **1:** The descriptor is a kernel-mode descriptor submitted by kernel-mode software.
30:20	Reserved
19:0	PASID This field contains the Process Address Space ID of the requesting process and indicates the default address space for this descriptor.

Table 8-1: Descriptor Trusted Fields

8.1.2 Operation

Offset: 7; Size: 1 byte (8 bits)

This field specifies the operation to be executed.

0x00	No-op
0x01	Batch
0x02	Drain
0x03	Memory Move
0x04	Fill
0x05	Compare
0x06	Compare Pattern
0x07	Create Delta Record
0x08	Apply Delta Record
0x09	Memory Copy with Dualcast
0x0A	Translation Fetch
0x0B – 0x0F	Reserved
0x10	CRC Generation
0x11	Copy with CRC Generation
0x12	DIF Check
0x13	DIF Insert
0x14	DIF Strip
0x15	DIF Update
0x16	Reserved
0x17	DIX Generate
0x18 – 0x1F	Reserved
0x20	Cache flush
0x21	Update Window
0x22	Reserved
0x23	Inter-Domain Memory Copy
0x24	Inter-Domain Fill
0x25	Inter-Domain Compare
0x26	Inter-Domain Compare Pattern
0x27	Inter-Domain Cache Flush
0x28 – 0xFF	Reserved

Table 8-2: Operation Types

8.1.3 Flags

Offset: 4; Size: 3 bytes (24 bits)

Bits	Description
23:16	**Operation-Specific Flags** See the descriptions of the following operation types for the meaning of this field: Drain, CRC Generation, Copy with CRC Generation, and Memory Copy with Dualcast, Translation Fetch, Update Window and Inter-Domain operations. This field is reserved for all other operation types.
15	Reserved. Must be 0.
14	**Destination Readback** **0:** No readback is performed. **1:** After all writes to the destination have been issued by the device, a read of the final destination address is performed before the operation is completed. The readback is performed only if the descriptor is completed successfully. This field is reserved if the Destination Readback Support field in GENCAP is 0. This field is reserved for operation types that do not write to memory.
13	**Strict Ordering** **0:** Default behavior: writes to the destination can become globally observable out of order. The completion record write has strict ordering, so it always completes after all writes to the destination are globally observable. **1:** Forces strict ordering of all memory writes produced by the device and ensures that they become globally observable in that order. This field is reserved for operation types that do not write to memory. If the Enable Relaxed Ordering field in the PCIe config Device Control register is 0, this field is ignored, and all memory writes use strict ordering. Note that this flag has nothing to do with the order in which descriptors are executed. It only affects ordering of the writes generated by this descriptor.
12	**Completion Record TC Selector** This field selects the Traffic Class value used for writing the completion record. It selects one of the two TC values in the Group Configuration Register corresponding to the WQ that the descriptor was submitted to. See section 4.2 for information on the use of Traffic Classes. **0:** Use TC-A in the Group Configuration Register. **1:** Use TC-B in the Group Configuration Register. This field is reserved when Completion Record Address Valid is 0.
11	**Address 3 TC Selector** This field selects one of the two Traffic Class values in the Group Configuration Register corresponding to the WQ that the descriptor was submitted to. **0:** Use TC-A in the Group Configuration Register. **1:** Use TC-B in the Group Configuration Register. For Memory Copy with Dualcast, this field selects the TC value used for writes to Destination2 Address. For CRC Generation and Copy with CRC Generation, this field selects the TC value used for reading the CRC Seed. It is reserved if the Read CRC Seed field is 0. For Create Delta Record, this field selects the TC value for writes to Delta Record Address. This field is reserved for all other operation types.

10	**Address 2 TC Selector** This field selects one of the two Traffic Class values in the Group Configuration Register corresponding to the WQ that the descriptor was submitted to. **0:** Use TC-A in the Group Configuration Register. **1:** Use TC-B in the Group Configuration Register. For most operation types this field selects the TC value used for writes to Destination Address. For Memory Copy with Dualcast, this field selects the TC value for writes to Destination1 Address. For Compare and Create Delta Record, this field selects the TC value for reads from Source2 Address. For Drain, this field selects the TC value used for readback from Readback Address 2. This field is reserved for operation types that do not use Destination Address, Destination1 Address, or Source2 Address.
9	**Address 1 TC Selector** This field selects one of the two Traffic Class values in the Group Configuration Register corresponding to the WQ that the descriptor was submitted to. **0:** Use TC-A in the Group Configuration Register. **1:** Use TC-B in the Group Configuration Register. For most operation types this field selects the TC value used for reads from Source Address. For Batch, this field selects the TC value used for reading the descriptor list. For Compare and Create Delta Record, this field selects the TC value used for reads from Source1 Address. For Drain, this field selects the TC value used for readback from Readback Address 1. For Apply Delta Record, this field selects the TC value for reads from Delta Record Address. This field is reserved for the following operation types: No-op, Fill, and Cache Flush.
8	**Cache Control** For operations that write to memory: **0:** Hint to direct data writes to memory. **1:** Hint to direct data writes to CPU cache. This hint does not affect writing to the completion record, which is always directed to cache. If the Cache Control Support field in GENCAP is 0, this field is reserved. This field is reserved in Cache Flush and Inter-Domain Cache Flush descriptors. This field is reserved for operation types that do not write to memory.
7	**Check Result** **0:** Result of operation does not affect the Status field of the completion record. **1:** Result of operation affects the Status field of the completion record, if the operation is successful. Status is set to either Success or Success with false predicate, depending on the result of the operation. See the description of each operation for the possible results and how they affect the Status. This field is used for Compare, Compare Pattern, and Create Delta Record. It is reserved for all other operation types.
6:5	**Reserved.** Must be 0.

4	**Request Completion Interrupt** **0:** No interrupt is generated when the operation completes. **1:** An interrupt is generated when the operation completes. If both a completion record and a completion interrupt are generated, the interrupt is always generated after the completion record is written. See section 3.7 for information regarding the interrupt to be generated. This field is reserved if User-mode Interrupts Enable is 0 and Priv is 0 (indicating a user-mode descriptor). If WQ PASID Enable control is 0, this field is not-reserved, independent of the setting of the User-mode Interrupts Enable control (see section 9.2.8).
3	**Request Completion Record** **0:** A completion record is written only if the operation status is not equal to 0x01, 0x02, or 0x05. **1:** A completion record is always written at the completion of the operation. This flag must be 1 for any operation that yields a result, such as Compare. This flag must be 0 if Completion Record Address Valid is 0.
2	**Completion Record Address Valid** **0:** The completion record address is not valid. **1:** The completion record address is valid. This flag must be 1 for any operation that yields a result, such as Compare. It should be 1 for any operation that uses virtual addresses, because of the possibility of a page fault, which must be reported via the completion record. For best results, this flag should be 1 in all descriptors, because it allows the device to report errors to the software that submitted the descriptor. If this flag is 0 and an unexpected error occurs, the error is reported in the event log, if enabled, or in the SWERROR register, and the software that submitted the request may not be notified of the error. Notwithstanding the above caveats, if the descriptor uses physical addresses or uses virtual addresses that software guarantees are present (pinned), and software has no need to receive notification of any other types of errors, this flag may be 0.
1	**Block On Fault** **0:** Page faults cause partial completion of the descriptor. **1:** The device waits for page faults to be resolved and then continues the operation. This flag does not affect the handling of page faults on Completion Record Address, Descriptor List Address, or Drain Readback Address, all of which always block on fault. See section 3.13. This field is reserved if the Block on Fault Enable field in WQCFG is 0. This field is reserved for certain operation types: No-op, Drain, and Batch.
0	**Fence** **0:** This descriptor may be executed in parallel with other descriptors in the batch. **1:** The device waits for previous descriptors in the same batch to complete before beginning work on this descriptor. If any previous descriptor completed with Status not equal to Success, this descriptor and all subsequent descriptors in the batch are abandoned. This field may only be set in descriptors that are in a batch. It is reserved in descriptors submitted directly to a Work Queue.

Table 8-3: Descriptor Flags

8.1.4 Completion Record Address

Offset 8; Size 8 bytes (64 bits)

This field specifies the address of the completion record. The completion record is 32 bytes and must be aligned on a 32-byte boundary. If the Completion Record Address Valid flag is 0, this field is reserved.

If the Request Completion Record flag is 1, a completion record is written to this address at the completion of the operation. If Request Completion Record is 0, a completion record is written to this address only if there is a page fault or error.

For any operation that yields a result, such as Compare, the Completion Record Address Valid and Request Completion Record flags must both be 1 and the Completion Record Address must be valid.

For any operation that uses virtual addresses, the Completion Record Address should be valid, whether or not the Request Completion Record flag is set, so that a completion record may be written in case there is a page fault or error.

For best results, this field should be valid in all descriptors, because it allows the device to report errors to the software that submitted the descriptor. Otherwise, if an unexpected error occurs, the error is reported in the SWERROR register or event log, and the software that submitted the request may not be notified of the error.

8.1.5 Source Address

Offset: 16; Size: 8 bytes (64 bits)

For operations that read data from memory, this field specifies the address of the source data. There is no alignment requirement for the source address for most operation types. Exceptions are noted in the operation descriptions. If the Source Address and Transfer Size are not both aligned to a multiple of 64 bytes, an implementation may read more source data than required by the descriptor. For example, source data may be read in aligned 32-byte chunks. The excess data is discarded.

8.1.6 Destination Address

Offset: 24; Size: 8 bytes (64 bits)

For operations that write data to memory, this field specifies the address of the destination buffer. There is no alignment requirement for the destination address for most operation types. Exceptions are noted in the operation descriptions.

For some operation types, this field is used as the address of a second source buffer.

8.1.7 Transfer Size

Offset: 32; Size: 4 bytes (32 bits)

This field indicates the number of bytes to be read from the source address to perform the operation.

The maximum allowed transfer size is dependent on the WQ that the descriptor was submitted to. It is specified by the WQ Maximum Transfer Size field for the WQ in the WQ Configuration Table (which is, in turn, limited by the Maximum Supported Transfer Size field in the General Capabilities Register). The Create Delta Record operation has an additional limitation on the maximum allowed transfer size, noted in the description of that operation.

For a Batch operation, this field contains the Descriptor Count. Descriptor Count must be greater than 1. The maximum allowed descriptor count is specified by the WQ Maximum Batch Size field for the WQ in the WQ Configuration Table (which is, in turn, limited by the Maximum Supported Batch Size field in the General Capabilities Register).

Transfer Size must not be 0. For most operation types, there is no alignment requirement for the transfer size. Exceptions are noted in the operation descriptions.

8.1.8 Completion Interrupt Handle

Offset: 36; Size: 2 bytes (16 bits)
This field specifies the interrupt table entry to be used to generate a completion interrupt, as described in section 3.7.

This field is reserved if the Request Completion Interrupt flag is 0.

8.2 Completion Record

The completion record is a 32-byte structure in memory that the device writes when the operation is complete or encounters an error. A completion record address is in each descriptor. The completion record address must be 32-byte aligned. See section 3.6 for more information.

This section describes fields of the completion record that are common to most operation types. Additional operation-specific fields are described in the detailed operation descriptions in section 8.3. The completion record is always 32 bytes even if not all fields are needed. The completion record contains enough information to continue the operation if it was partially completed due to a page fault. Page faults are indicated by Operation Status codes 0x03, 0x04, 0x06, and 0x1f, described in Table 5-6. Software should not depend on the value of unused fields (including fields that are unused for specific operation types).

Generic Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info	Result	Status	0
Fault Address								8
Operation-specific fields					Invalid Flags			16
								24

8.2.1 Status

Offset: 0; Size: 1 byte (8 bits)
This field reports the completion status of the descriptor. Hardware never writes 0 to this field. Software should initialize this field to 0 so it can detect when the completion record has been written. See section 5.8.1 for a list of the operation status codes and their meanings.

Bits	Description
7	**R/W** (Not used unless Operation Status indicates a translation fault – code 0x03, 0x04, 0x06, 0x1a, or 0x1f) **0:** The faulting access was a read. **1:** The faulting access was a write.
6	Unused.

Bits	Description
5:0	**Operation Status** See section 5.8.1 for the meaning of the value in this field.

Table 8-4: Completion Record Status field

8.2.2 Result

Offset: 1; Size: 1 byte (8 bits)

For some operation types, the Result field contains information about the result of the operation. The description of each operation type includes the possible values and meaning of this field. Software should not depend on the value of this field for operation types where no meaning is specified.

8.2.3 Fault Info

Offset: 2; Size: 1 byte (8 bits)

If the operation was partially completed due to a page fault and Completion Record Fault Info Support in GENCAP is 1, this field contains additional information about the fault encountered.

Bits	Description
7:4	**Unused**
3:1	**Operand Identifier** **0:** Unknown **1:** Source, Source1, Descriptor List, Delta Record Address (Apply Delta Record), Drain Readback Address1, or Translation Fetch **2:** Source2, CRC Seed Address or Drain Readback Address2 **3:** Destination, Destination1, or Delta Record Address (Create Delta Record) **4:** Destination2 Address **5:** Completion Record Address **6-7:** Reserved
0	**Fault Address Masked** **0:** The fault address field contains the address that caused the fault. **1:** The fault address is masked or not available.

Table 8-5: Completion Record Fault Info

8.2.4 Bytes Completed

Offset: 4; Size: 4 bytes (32 bits)

If the operation was partially completed due to a page fault, this field contains the number of source bytes processed before the fault occurred. All of the source bytes represented by this count were fully processed and the result written to the destination address, as needed according to the operation type. Page faults are indicated by Operation Status codes 0x03, 0x04, 0x06, and 0x1f, described in Table 5-6. For other errors, this field is undefined.

For some operation types, this field may also be used when the operation stopped before completion for some reason other than a fault. These uses are described in the section specific to each operation type.

If the operation fully completed, this field is 0.

For operation types where the output size is not readily determinable from this value, the completion record also contains the number of bytes written to the destination address.

8.2.5 Fault Address

Offset: 8; Size: 8 bytes (64 bits)

If the operation was partially completed due to a page fault and Completion Record Fault Info Support in GENCAP is 1, the Fault Info field specifies if the Fault Address is available. If Completion Record Fault Info Support in GENCAP is 0, this field always contains the address that caused the fault. Bits 11:0 may be reported as 0. Page faults are reported as Operation Status codes 0x03, 0x04, 0x06, or 0x1f, described in Table 5-6. For other errors, this field is undefined.

8.2.6 Invalid Flags

Offset: 16; Size: 3 bytes (24 bits)

If the Operation Status is Invalid flags, this field contains a bitmask of the flags that were found to be invalid, to aid in debugging. If a bit in this field is 1, it indicates that the flag at the corresponding bit position in the Flags field of the descriptor was invalid. The implementation is not obligated to indicate every invalid flag that may be present in the descriptor, but it must indicate at least one any time it reports an Invalid flags error code.

If the operation status is anything other than Invalid Flags, this field may be used for operation-specific information, or it may be unused, depending on the operation type. See the description of the completion record for each operation type for more information.

8.3 Descriptor Types

8.3.1 No-op

The No-op operation, 0x00, performs no DMA operation. It may request a completion record and/or completion interrupt. If it is in a batch, it may specify the Fence flag to ensure that the completion of the No-op descriptor occurs after completion of all previous descriptors in the batch.

No-op Descriptor

<table>
<tr><th colspan="2">Byte 7</th><th colspan="2">Byte 6</th><th colspan="2">Byte 5</th><th colspan="2">Byte 4</th><th colspan="2">Byte 3</th><th colspan="2">Byte 2</th><th colspan="2">Byte 1</th><th colspan="2">Byte 0</th><th>bytes</th></tr>
<tr><td colspan="2">Operation</td><td colspan="6">Flags</td><td>Priv</td><td colspan="2">Reserved</td><td colspan="5">PASID</td><td>0</td></tr>
<tr><td colspan="16">Completion Record Address</td><td>8</td></tr>
<tr><td colspan="16" rowspan="2"></td><td>16</td></tr>
<tr><td>24</td></tr>
<tr><td colspan="4"></td><td colspan="4">Completion Interrupt Handle</td><td colspan="8"></td><td>32</td></tr>
<tr><td colspan="16" rowspan="3">Reserved</td><td>40</td></tr>
<tr><td>48</td></tr>
<tr><td>56</td></tr>
</table>

No-op Completion Record

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td colspan="7" rowspan="4">Unused</td><td>Status</td><td>0</td></tr>
<tr><td rowspan="3"></td><td>8</td></tr>
<tr><td>16</td></tr>
<tr><td>24</td></tr>
</table>

8.3.2 Batch

The Batch operation, 0x01, queues multiple descriptors at once. The Descriptor List Address is the address of a contiguous array of work descriptors to be processed. Each descriptor in the array is 64 bytes. Descriptor List Address must be 64-byte aligned. Descriptor Count is the number of descriptors in the array. The set of descriptors in the array is called the "batch." Descriptor Count must be greater than 1. The maximum number of descriptors allowed in a batch is specified by the WQ Maximum Batch Size field for the WQ in the WQ Configuration Table (which is, in turn, limited by the Maximum Supported Batch Size field in the General Capabilities Register).

The PASID and the Priv flag associated with the Batch descriptor are used for all descriptors in the batch. The PASID and Priv fields in the descriptors in the batch are ignored.

The Descriptors Completed field of the completion record contains the total number of descriptors in the batch that were processed, whether they were successful or not. Descriptors Completed may be less than Descriptor Count if there is a Fence in the batch or if an unrecoverable translation failure occurred while reading the batch.

The Status field of the Batch completion record indicates Success if all of the descriptors in the batch completed successfully; otherwise, it indicates if there was a page fault on the Descriptor List Address or if one or more descriptors in the batch completed with Status not equal to Success.

If Batch Continuation Support in GENCAP is 1, the Result field in the completion record is 1 if any of the operations in the batch completed with Status is not equal to Success. In some cases, Result may be 1 if a descriptor in the batch encountered a page fault on the completion record address, even though the completion record page fault was resolved successfully. Software can examine the completion records for descriptors in the batch to determine whether there were truly any failures.

Batch Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Descriptor List Address								16
Reserved								24
		Completion Interrupt Handle		Descriptor Count				32
								40
								48
								56

Batch Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Descriptors Completed				Unused	Fault Info	Result	Status	0
Fault Address								8
Unused								16
								24

If software continues execution of a batch after a fault on the Descriptor List Address, it should copy the Result field from the completion record of the partial completion into the Batch Error field of the continuation descriptor

See section 3.8 for details of batch processing.

Bits	Description
23:17	**Reserved:** Must be 0.
16	**Batch Error** **0:** This is either a new batch or all operations already completed in this batch completed with Status equal to Success. **1:** One or more operations in the batch previously completed with Status not equal to Success. Hardware uses this information in processing of the Fence flag and to determine the Status and Result fields in the Batch completion record. This flag must be 0 if Batch Continuation Support in GENCAP is 0.

Table 8-6 : Batch Operation-Specific Flags

8.3.3 Drain

The Drain operation, 0x02, waits for completion of certain outstanding descriptors in the WQ that the Drain descriptor is submitted to, as described in section 3.10.

A Drain descriptor may not be included in a batch; it is treated as an unsupported operation type.

Drain must specify Request Completion Record or Request Completion Interrupt. Completion notification is made after the other descriptors have completed.

Table 8-7 lists the operation-specific flags allowed with the Drain operation. The Readback Address 1 Valid and Readback Address 2 Valid flags are reserved if the Drain Descriptor Readback Address Support capability bit is 0.

The flags Address 1 TC Selector, and Address 2 TC Selector are conditionally allowed in the Drain descriptor. Address 1 TC Selector is reserved when Readback Address 1 Valid is 0. Address 2 TC Selector is reserved when Readback Address 2 Valid is 0.

Drain Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Readback Address 1								16
Readback Address 2								24
Reserved		Completion Interrupt Handle						32
								40
								48
								56

Drain Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
							Status	0
Fault Address								8
Unused								16
								24

Bits	Description
23:20	**Reserved:** Must be 0.
19	**Suppress TC-B Implicit Readback** **0:** Hardware may perform implicit readback on TC-B. **1:** Hardware will not perform implicit readback on TC-B. Note that this flag does not affect readbacks to the explicit Readback Addresses.
18	**Suppress TC-A Implicit Readback** **0:** Hardware may perform implicit readback on TC-A. **1:** Hardware will not perform implicit readback on TC-A. Note that this flag does not affect readbacks to the explicit Readback Addresses.
17	**Readback Address 2 Valid** **0:** Readback Address 2 field is reserved. **1:** Readback Address 2 field is valid and hardware will perform a readback to this address on the TC specified by the Address 2 TC Selector flag. Note that the destination readback flag is reserved for Drain descriptors.
16	**Readback Address 1 Valid** **0:** Readback Address 1 field is reserved. **1:** Readback Address 1 field is valid and hardware will perform a readback to this address on the TC specified by the Address 1 TC Selector flag. Note that the destination readback flag is reserved for Drain descriptors.

Table 8-7: Drain Operation-Specific Flags

8.3.4 Memory Move

The Memory Move operation, 0x03, copies memory from the Source Address to the Destination Address. The number of bytes copied is given by Transfer Size. There are no alignment requirements for the memory addresses or the transfer size.

If the source and destination regions overlap, the behavior depends on the value of the Overlapping Copy Support field in GENCAP. If Overlapping Copy Support is 1, the memory copy is done as if the entire source buffer is copied to temporary space and then copied to the destination buffer. (This may be implemented by reversing the direction of the copy when the beginning of the destination buffer overlaps the end of the source buffer.) If Overlapping Copy Support is 0, it is an error.

If the operation is partially completed due to a page fault, the Result field of the completion record contains the direction of the copy. It is 0 if the copy was performed starting at the beginning of the source and destination buffers; it is 1 if the direction of the copy was reversed. If Overlapping Copy Support is 0, Result is always 0.

To resume the operation after a partial completion, if Result is 0, the Source and Destination Address fields in the continuation descriptor should be increased by Bytes Completed, and the Transfer Size should be decreased by Bytes Completed. If Result is 1, the Transfer Size should be decreased by Bytes Completed, but the Source and Destination Address fields should be the same as in the original descriptor. Note that if a subsequent partial completion occurs, the Result field is not necessarily the same as it was for the first partial completion.

Memory Move Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved	PASID			0
Completion Record Address								8
Source Address								16
Destination Address								24
Reserved		Completion Interrupt Handle		Transfer Size				32
Reserved								40
								48
								56

Memory Move Descriptor Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info	Result	Status	0
Fault Address								8
Reserved								16
								24

8.3.5 Fill

The Memory Fill operation, 0x04, fills memory at the Destination Address with the value in the pattern field. The pattern size is specified by the Pattern Size flag. When the pattern size is 8 bytes, the pattern is specified in the Pattern Lower field. When the pattern size is 16 bytes, the first 8 bytes are in Pattern Lower and the next 8 bytes are in Pattern Upper. (To use a smaller pattern, software must replicate the pattern in the descriptor.) The number of bytes written is given by Transfer Size. The transfer size does not need to be a multiple of the pattern size. There are no alignment requirements for the destination address or the transfer size. If the operation is partially completed due to a page fault, the Bytes Completed field of the completion record contains the number of bytes written to the destination before the fault occurred.

Fill Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Pattern Lower								16
Destination Address								24
		Completion Interrupt Handle		Transfer Size				32
Pattern Upper								40
Reserved								48
								56

Fill Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info		Status	0
Fault Address								8
Unused								16
								24

Bits	Description
23:19	**Reserved:** Must be 0.
18	**Pattern Size** **0:** Pattern size is 8B and specified in the Pattern Lower field. The Pattern Upper field is reserved. **1:** Pattern size is 16B and specified by the Pattern Lower and Pattern Upper fields. This field must be 0 if Fill16 Support in GENCAP is 0.
17:16	**Reserved:** Must be 0.

Table 8-8 : Fill Operation-Specific Flags

8.3.6 Compare

The Compare operation, 0x05, compares memory at Source1 Address with memory at Source2 Address. The number of bytes compared is given by Transfer Size. There are no alignment requirements for the memory addresses or the transfer size. The Completion Record Address Valid and Request Completion Record flags must be 1 and the Completion Record Address must be valid. The result of the comparison is written to the Result field of the completion record: a value of 0 indicates that the two memory regions match, and a value of 1 indicates that they do not match. If Result is 1, the Bytes Completed field of the completion record indicates the byte offset of the first difference. If the operation is partially completed due to a page fault, Result is 0. (If a difference had been detected, the difference would be reported instead of the page fault.)

Compare Descriptor

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td>Operation</td><td colspan="3">Flags</td><td>Priv</td><td>Reserved</td><td colspan="2">PASID</td><td>0</td></tr>
<tr><td colspan="8">Completion Record Address</td><td>8</td></tr>
<tr><td colspan="8">Source1 Address</td><td>16</td></tr>
<tr><td colspan="8">Source2 Address</td><td>24</td></tr>
<tr><td colspan="2"></td><td colspan="2">Completion Interrupt Handle</td><td colspan="4">Transfer Size</td><td>32</td></tr>
<tr><td colspan="7" rowspan="3">Reserved</td><td>Expected Result</td><td>40</td></tr>
<tr><td></td><td>48</td></tr>
<tr><td></td><td>56</td></tr>
</table>

Compare Completion Record

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td colspan="4">Bytes Completed</td><td>Unused</td><td>Fault Info</td><td>Result</td><td>Status</td><td>0</td></tr>
<tr><td colspan="8">Fault Address</td><td>8</td></tr>
<tr><td colspan="8" rowspan="2">Unused</td><td>16</td></tr>
<tr><td>24</td></tr>
</table>

If the operation is successful and the Check Result flag is 1, the Status field of the completion record is set according to Result and Expected Result, as shown in Table 8-9. This allows a subsequent descriptor in the same batch with the Fence flag to continue or stop execution of the batch based on the result of the comparison. Bits 7:1 of Expected Result are ignored.

Check Result flag	Expected Result bit 0	Result	Status
0	X	X	Success
1	0	0	Success
1	0	1	Success with false predicate
1	1	0	Success with false predicate
1	1	1	Success

Table 8-9: Completion Status for Compare Descriptor

8.3.7 Compare Pattern

The Compare Pattern operation, 0x06, compares memory at Source Address with the value in the pattern field. The pattern size is always 8 bytes. (To use a smaller pattern, software must replicate the pattern in the descriptor.) The number of bytes compared is given by Transfer Size. The transfer size does not need to be a multiple of the pattern size. The Completion Record Address Valid and Request Completion Record flags must be 1 and the Completion Record Address must be valid. The result of the comparison is written to the Result field of the completion record; a value of 0 indicates that the memory region matches the pattern, and a value of 1 indicates that it does not match. If Result is 1, the Bytes Completed field of the completion record indicates the location of the first difference. (It may not be the exact byte location, but it is guaranteed to be no greater than the first difference.) If the operation is partially completed due to a page fault, Result is 0. (If a difference had been detected, the difference would be reported instead of the page fault.)

The completion record format for Compare Pattern and the behavior of Check Result and Expected Result are identical to Compare.

Compare Pattern Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source Address								16
Pattern								24
		Completion Interrupt Handle		Transfer Size				32
Reserved							Expected Result	40
								48
								56

Compare Pattern Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info	Result	Status	0
Fault Address								8
Unused								16
								24

8.3.8 Create Delta Record

The Create Delta Record operation, 0x07, compares memory at Source1 Address with memory at Source2 Address and generates a delta record that contains the information needed to update source1 to match source2. The number of bytes compared is given by Transfer Size. The transfer size is limited by the maximum offset that can be stored in the delta record, as described below, in addition to the usual WQ-specific limit on transfer size. Source1 Address, Source2 Address, and Transfer Size must be aligned to a multiple of 8. The Completion Record Address Valid and Request Completion Record flags must be 1 and the Completion Record Address must be valid.

The maximum size of the delta record is given by Maximum Delta Record Size. The maximum delta record size should be a multiple of the delta size (10 bytes), must not be less than the maximum number of deltas that can be generated from a single cache line (80 bytes), and must be no greater than the value allowed by the WQ Maximum Transfer Size in the WQ Configuration Table of the WQ that this descriptor was submitted to. If the maximum-size delta record overlaps either of the source buffers, it is an error. The actual size of the delta record that is generated depends on the number of differences detected between source1 and source2; this size is written to the Delta Record Size field of the completion record. If the space needed in the delta record exceeds the maximum delta record size specified in the descriptor, the operation completes with a partial delta record.

The result of the comparison is written to the Result field of the completion record. If the two regions match exactly, then Result is 0, Delta Record Size is 0, and Bytes Completed is 0. If the two regions don't match, and a complete set of deltas was written to the delta record, then Result is 1, Delta Record Size contains the total size of all the differences found, and Bytes Completed is 0. If the two regions don't match, and the space needed to record all the deltas exceeded the maximum delta record size, then Result is 2, Delta Record Size contains the size of the set of deltas written to the delta record (typically equal or nearly equal to the Maximum Delta Record Size specified in the descriptor), and Bytes Completed contains the number of bytes compared before space in the delta record was exceeded.

Create Delta Record Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source1 Address								16
Source2 Address								24
Reserved		Completion Interrupt Handle		Transfer Size				32
Delta Record Address								40
				Maximum Delta Record Size				48
Reserved							Expected Result Mask	56

Create Delta Record Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info	Result	Status	0
Fault Address								8
				Delta Record Size				16
Unused								24

If the operation is partially completed due to a page fault, Result is set to 0 if no deltas were written prior to the page fault, and Result is set to 1 if any deltas were written prior to the page fault. This behavior is the same whether the page fault is on one of the source buffers or on the delta record buffer. Bytes Completed contains the number of bytes compared before the page fault occurred, and Delta Record Size contains the space used in the delta record before the page fault occurred. If the operation fails due to any other error, these fields are undefined. To ensure that software can resume the operation without losing any deltas, if the fault occurred on the delta record, Bytes Completed does not include the bytes where the difference was found that was not written to the delta record.

The format of the delta record is shown below. The delta record contains an array of deltas. Each delta contains a 2-byte offset and an 8-byte block of data from Source2 that is different from the corresponding 8 bytes in Source1. The 2-byte offset field is stored in memory with the low byte at the lower address (little-endian). The total size of the delta record is a multiple of 10. Since the offset is a 16-bit field representing a multiple of 8 bytes, the maximum offset that can be expressed is 0x7FFF8, so the maximum Transfer Size is 0x80000 bytes (512 KB).

Byte 0	1	2	3	4	5	6	7	8	Byte 9
Offset		Data							
LSB	MSB	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7

If the operation is successful and the Check Result flag is 1, the Status field of the completion record is set according to Result and Expected Result Mask. This allows a subsequent descriptor in the same batch with the Fence flag to continue or stop execution of the batch based on the result of the delta record creation. Status is set as follows: If the value of Result is X and bit X of the Expected Results Mask is 1, Status is set to Success. If bit X is 0, Status is set to Success with false predicate. Since the value of Result is 0, 1, or 2, bits 7:3 of Expected Result Mask are ignored. Note that if bits 2:0 of Expected Result Mask are 0, Status will always be set to Success with false predicate, and if bits 2:0 of Expected Result Mask are all 1, Status will always be set to Success.

If the operation is successful and the Check Result flag is 0, the Expected Result Mask is ignored and Status is set to Success.

8.3.9 Apply Delta Record

The Apply Delta Record operation, 0x08, applies a delta record to the contents of memory at Destination Address. Delta Record Address is the address of a delta record that was created by a Create Delta Record operation that completed with Result equal to 1. Delta Record Size is the size of the delta record, as reported in the completion record of the Create Delta Record operation. Destination Address is the address of a buffer that contains the same contents as the memory at the Source1 Address when the delta record was created. Transfer Size is the same as the Transfer Size used when the delta record was created. After the Apply Delta Record operation completes, the memory at Destination Address will match the contents that were in memory at the Source2 Address when the delta record was created. Destination Address and Transfer Size must be aligned to a multiple of 8. If the delta record overlaps the destination buffer, it is an error.

If a page fault is encountered during the Apply Delta Record operation, the Bytes Completed field of the completion record contains the number of bytes of the delta record that were successfully applied to the destination. If software chooses to submit another descriptor to resume the operation, the continuation descriptor should contain the same Destination Address as the original. The Delta Record Address should be increased by Bytes Completed (so it points to the first unapplied delta), and the Delta Record Size should be reduced by Bytes Completed.

If the offset fields in the delta record are not in ascending order, or if any offset field is greater than or equal to Transfer Size, an error is reported and the Bytes Completed field of the completion record contains the number of bytes of the delta record that were successfully applied to the destination prior to the error.

See section 8.3.8 for a description of the format of the delta record.

Apply Delta Record Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Delta Record Address								16
Destination Address								24
Reserved		Completion Interrupt Handle		Transfer Size				32
Reserved				Delta Record Size				40
								48
								56

Apply Delta Record Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed					Fault Info		Status	0
Fault Address								8
Unused								16
								24

Figure 8-1 shows the usage of the Create Delta Record and Apply Delta Record operations. First, the Create Delta Record operation is performed. It reads the two source buffers and writes the delta record, recording the actual delta record size in its completion record. The Apply Delta Record operation takes the content of the delta record that was written by the Create Delta Record operation, along with its size and a copy of the Source1 data, and updates the destination buffer to be a duplicate of the original Source2 buffer.

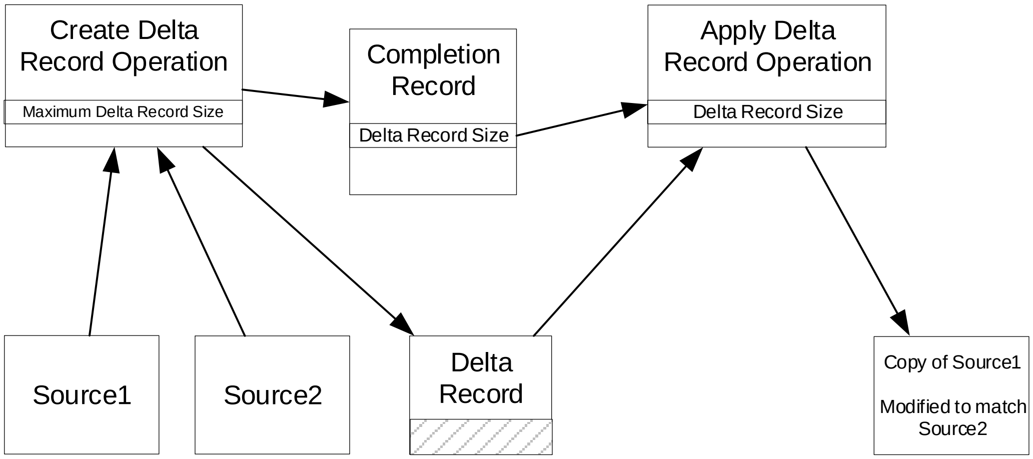

Create Delta Record Operation
Maximum Delta Record Size
Completion Record
Delta Record Size
Apply Delta Record Operation
Delta Record Size
Source1
Source2
Delta Record
Copy of Source1
Modified to match Source2

Figure 8-1: Delta Record Usage

8.3.10 Memory Copy with Dualcast

The Memory Copy with Dualcast operation, 0x09, copies memory from the Source Address to both Destination1 Address and Destination2 Address. The number of bytes copied is given by Transfer Size. There are no alignment requirements for the source address or the transfer size. Bits 11:0 of the two destination addresses must be the same.

If the source region overlaps with either of the destination regions or if the two destination regions overlap, it is an error. If the operation is partially completed due to a page fault, the copy operation stops after having written the same number of bytes to both destination regions.

Memory Copy with Dualcast Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source Address								16
Destination1 Address								24
Reserved		Completion Interrupt Handle		Transfer Size				32
Destination2 Address								40
Reserved								48
								56

Memory Copy with Dualcast Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed					Fault Info		Status	0
Fault Address								8
Unused								16
								24

8.3.11 Translation Fetch

The Translation Fetch operation, 0x0A, fetches address translations for the address range specified in the descriptor by issuing address translation (ATS) requests to the IOMMU. There is no data movement associated with this operation. The Region Size field specifies the size of the address range over which the translation requests may be issued. If the Use Stride flag is 1, the Region Stride field specifies the number of bytes to skip to compute the address for each subsequent ATS request. The Region Stride must be greater than or equal to 4096 and must be a power of 2. The descriptor execution terminates when the ATS request address is greater than or equal to the starting address plus region size. If Use Stride is 0, the device uses an implementation specific stride value. There are no alignment requirements for the address or the region size. However, the translation requests are always 4KB aligned and may be aligned to a multiple of the Region Stride.

The Region Size must be non-zero, and the sum of Address and Region Size in the descriptor must be less than or equal to 2^{64}. In case a page fault is encountered, the Block on Fault flag controls whether the operation is partially completed, or a page request is issued to the IOMMU. In case of partial completion due to a page fault, the fault address is reported in the completion record. The Bytes Completed field is undefined and should not be relied upon by software.

The operation may result in one or more of the address translations performed by the IOMMU, or translation structure entries used for address translation, or both, to be cached in an Address Translation Cache in the device or in the IOMMU.

Translation Fetch Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv	Reserved	PASID		0
Completion Record Address								8
Address								16
								24
		Completion Interrupt Handle		Region Size				32
Reserved								40
				Region Stride				48
								56

Translation Fetch Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Unused					Fault Info		Status	0
Fault Address								8
Unused								16
								24

Bits	Description
23:17	**Reserved:** Must be 0.
16	**Use Stride** **0:** The Region Stride field is reserved. Hardware uses an implementation specific value. **1:** The Region Stride field specifies the number of bytes to skip to compute the address for each subsequent ATS request. This field must be 0 if Translation Fetch Stride Support in GENCAP is 0.

Table 8-10 : Translation Fetch Operation-Specific Flags

8.3.12 CRC Generation

The CRC Generation operation, 0x10, computes the CRC on memory at the Source Address. See Appendix A for details of CRC Generation. The number of bytes used for the CRC computation is given by Transfer Size. There are no alignment requirements for the memory addresses or the transfer size. The Completion Record Address Valid and Request Completion Record flags must be 1 and the Completion Record Address must be valid. The computed CRC value is written to the completion record.

The CRC Generation operation-specific flags are shown in Table 8-11. The size of the CRC value computed, and the CRC seed used depends on the CRC Size flag. If CRC Size is 0, then a 32-bit CRC seed is used, and a 32-bit CRC value is computed. If CRC Size is 1, then a 64-bit CRC seed is used, and a 64-bit CRC value is computed.

If the Read CRC Seed flag is 1, the CRC seed is read from memory at the CRC Seed Address. The address must be naturally aligned to the size of the CRC value. If the Read CRC Seed flag is 0, the CRC Seed field in the descriptor is used for the seed. Unless this is a continuation of a partial CRC computation, the seed should be 0. If CRC Size is 0, bits 63:32 of CRC Seed are reserved.

If the operation is partially completed due to a page fault, the partial CRC result is written to the completion record along with the page fault information. If software corrects the fault and resumes the operation, it must use the partial CRC result as the seed of the continuation descriptor, either by copying it into the CRC Seed field or by setting the CRC Seed Address to the location of the partial CRC result and setting the Read CRC Seed flag to 1. If the operation fails due to any other error, or if Bytes Completed is 0, the CRC Value in the completion record is undefined and software should reuse the CRC Seed or CRC Seed Address from the descriptor.

If the Read CRC Seed flag is 0, the CRC Seed Address field is reserved. If the Read CRC Seed flag is 1, the CRC Seed field is reserved.

CRC Generation Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved	PASID			0
Completion Record Address								8
Source Address								16
Reserved								24
		Completion Interrupt Handle		Transfer Size				32
CRC Seed								40
CRC Seed Address								48
Reserved								56

CRC Generation Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info		Status	0
Fault Address								8
CRC Value								16
Unused								24

8.3.13 Copy with CRC Generation

The Copy with CRC Generation operation, 0x11, copies memory from the Source Address to the Destination Address and computes the CRC on the data copied. See Appendix A for details of CRC Generation. The number of bytes copied is given by Transfer Size. There are no alignment requirements for the memory addresses or the transfer size. If the source and destination regions overlap, it is an error. The Completion Record Address Valid and Request Completion Record flags must be 1 and the Completion Record Address must be valid. The computed CRC value is written to the completion record.

See the description of the CRC Generation operation in section 8.3.12 and Table 8-11 for a description of the CRC operation-specific flags, the CRC Seed field, and the CRC Seed Address field.

The completion record format for Copy with CRC Generation is identical to the format for CRC Generation.

Copy with CRC Generation Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source Address								16
Destination Address								24
		Completion Interrupt Handle		Transfer Size				32
CRC Seed								40
CRC Seed Address								48
Reserved								56

Bits	Description
23:20	**Reserved:** Must be 0.
19	**CRC Size** **0:** 32-bit CRC value is computed. Upper 4 bytes of CRC Seed field in the descriptor and the CRC Value field in the completion record are reserved. **1:** 64-bit CRC value is computed. This flag must be 0 if CRC64 Support in GENCAP is 0. See Appendix A for details of CRC computation.
18	**Bypass Data Reflection** **0:** Normal CRC operation: bit 0 of each data byte is the MSB in the CRC computation. **1:** Bit 7 of each data byte is the MSB in the CRC computation. See Appendix A for details of CRC computation.
17	**Bypass CRC Inversion and Reflection** **0:** Normal CRC operation: CRC seed and result are inverted and use standard CRC bit order. **1:** Bypass inversion and use reverse bit order for CRC seed and result. See Appendix A for details of CRC computation.
16	**Read CRC Seed** **0:** Use the CRC Seed field in the descriptor. **1:** Read the CRC seed from memory at the CRC Seed Address.

Table 8-11: CRC Generation Operation-Specific Flags

8.3.14 DIF Check

The DIF Check operation, 0x12, computes the Data Integrity Field (DIF) on the source data and compares the computed DIF to the DIF contained in the source data.

The number of source bytes read is given by Transfer Size. DIF computation is performed on each block of source data that is 512, 520, 4096, or 4104 bytes. The transfer size should be a multiple of the source block size plus 8 bytes for each source block. There is no alignment requirement for the source address.

If the operation completes successfully, the final Reference Tag and Application Tag are written to the completion record along with a Success completion status. If the operation is partially completed due to a page fault, updated values of Reference Tag and Application Tag are written to the completion record along with the page fault information. If software corrects the fault and resumes the operation, it may copy these fields into the continuation descriptor. If the operation fails due to any other error, these fields are undefined.

If an error is detected in the DIF in the source data, the operation stops. The Status field in the completion record is set to DIF Error, the DIF Status field is set to indicate the type of error, and the Bytes Completed field is set to the number of source bytes successfully processed. Bytes Completed does not include the block in which the error was detected. The Completion Record Address Valid and Request Completion Record flags must be 1 and the Completion Record Address must be valid.

See section 8.3.17, DIF Update, for a description of DIF Flags, Source DIF Flags, and the fields in the completion record. See Appendix B for details of DIF checking.

DIF Check Descriptor

<table>
<tr><th colspan="2">Byte 7</th><th colspan="2">Byte 6</th><th colspan="2">Byte 5</th><th colspan="2">Byte 4</th><th colspan="2">Byte 3</th><th colspan="2">Byte 2</th><th colspan="2">Byte 1</th><th colspan="2">Byte 0</th><th>bytes</th></tr>
<tr><td colspan="2">Operation</td><td colspan="6">Flags</td><td>Priv</td><td colspan="2">Reserved</td><td colspan="5">PASID</td><td>0</td></tr>
<tr><td colspan="16">Completion Record Address</td><td>8</td></tr>
<tr><td colspan="16">Source Address</td><td>16</td></tr>
<tr><td colspan="16">Reserved</td><td>24</td></tr>
<tr><td colspan="4"></td><td colspan="4">Completion Interrupt Handle</td><td colspan="8">Transfer Size</td><td>32</td></tr>
<tr><td colspan="10"></td><td colspan="2">DIF Flags</td><td colspan="2"></td><td colspan="2">Source DIF Flags</td><td>40</td></tr>
<tr><td colspan="4">Application Tag Seed</td><td colspan="4">Application Tag Mask</td><td colspan="8">Reference Tag Seed</td><td>48</td></tr>
<tr><td colspan="16"></td><td>56</td></tr>
</table>

DIF Check Completion Record

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td colspan="4">Bytes Completed</td><td>Unused</td><td>Fault Info</td><td>DIF Status</td><td>Status</td><td>0</td></tr>
<tr><td colspan="8">Fault Address</td><td>8</td></tr>
<tr><td colspan="2">Application Tag</td><td colspan="2">Application Tag Mask</td><td colspan="4">Reference Tag</td><td>16</td></tr>
<tr><td colspan="8">Unused</td><td>24</td></tr>
</table>

8.3.15 DIF Insert

The DIF Insert operation, 0x13, copies memory from the Source Address to the Destination Address, while computing the Data Integrity Field (DIF) on the source data and inserting the DIF into the output data.

The number of source bytes copied is given by Transfer Size. DIF computation is performed on each block of source data that is 512, 520, 4096, or 4104 bytes. The transfer size should be a multiple of the source block size. The number of bytes written to the destination is the transfer size plus 8 bytes for each source block. There is no alignment requirement for the memory addresses. If the source and destination regions overlap, it is an error.

If the operation completes successfully, the final Reference Tag and Application Tag are written to the completion record along with a Success completion status. If the operation is partially completed due to a page fault, updated values of Reference Tag and Application Tag are written to the completion record along with the page fault information. If software corrects the fault and resumes the operation, it may copy these fields into the continuation descriptor. If the operation fails due to any other error, these fields are undefined.

See section 8.3.17, DIF Update, for a description of DIF Flags, Destination DIF Flags, and the fields in the completion record. See Appendix B for details of DIF computation.

DIF Insert Descriptor

<table>
<tr><th colspan="2">Byte 7</th><th colspan="2">Byte 6</th><th colspan="2">Byte 5</th><th colspan="2">Byte 4</th><th colspan="2">Byte 3</th><th colspan="2">Byte 2</th><th colspan="2">Byte 1</th><th colspan="2">Byte 0</th><th>bytes</th></tr>
<tr><td colspan="2">Operation</td><td colspan="6">Flags</td><td>Priv</td><td colspan="2">Reserved</td><td colspan="5">PASID</td><td>0</td></tr>
<tr><td colspan="16">Completion Record Address</td><td>8</td></tr>
<tr><td colspan="16">Source Address</td><td>16</td></tr>
<tr><td colspan="16">Destination Address</td><td>24</td></tr>
<tr><td colspan="4"></td><td colspan="4">Completion Interrupt Handle</td><td colspan="8">Transfer Size</td><td>32</td></tr>
<tr><td colspan="10"></td><td colspan="2">DIF Flags</td><td colspan="2">Dest DIF Flags</td><td colspan="2"></td><td>40</td></tr>
<tr><td colspan="16">Reserved</td><td>48</td></tr>
<tr><td colspan="4">Application Tag Seed</td><td colspan="4">Application Tag Mask</td><td colspan="8">Reference Tag Seed</td><td>56</td></tr>
</table>

DIF Insert Completion Record

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td colspan="4">Bytes Completed</td><td></td><td>Fault Info</td><td></td><td>Status</td><td>0</td></tr>
<tr><td colspan="8">Fault Address</td><td>8</td></tr>
<tr><td colspan="8">Unused</td><td>16</td></tr>
<tr><td colspan="2">Application Tag</td><td colspan="2">Application Tag Mask</td><td colspan="4">Reference Tag</td><td>24</td></tr>
</table>

8.3.16 DIF Strip

The DIF Strip operation, 0x14, copies memory from the Source Address to the Destination Address, removing the Data Integrity Field (DIF). It optionally computes the DIF on the source data and compares the computed DIF to the DIF contained in the source data.

The number of source bytes read is given by Transfer Size. DIF computation is performed on each block of source data that is 512, 520, 4096, or 4104 bytes. The transfer size should be a multiple of the source block size plus 8 bytes for each source block. The number of bytes written to the destination is the transfer size minus 8 bytes for each source block. There is no alignment requirement for the memory addresses. If the source and destination regions overlap, it is an error.

If the operation completes successfully, the final Reference Tag and Application Tag are written to the completion record along with a Success completion status. If the operation is partially completed due to a page fault, updated values of Reference Tag and Application Tag are written to the completion record along with the page fault information. If software corrects the fault and resumes the operation, it may copy these fields into the continuation descriptor. If the operation fails due to any other error, these fields are undefined.

If an error is detected in the DIF in the source data, the operation stops. The Status field in the completion record is set to DIF Error, the DIF Status field is set to indicate the type of error, and the Bytes Completed field is set to the number of source bytes successfully processed. Bytes Completed does not include the block in which the error was detected.

See section 8.3.17, DIF Update, for a description of DIF Flags, Source DIF Flags, and the fields in the completion record. See Appendix B for details of DIF checking.

DIF Strip Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source Address								16
Destination Address								24
Reserved		Completion Interrupt Handle		Transfer Size				32
Reserved					DIF Flags	Reserved	Source DIF Flags	40
Application Tag Seed		Application Tag Mask		Reference Tag Seed				48
Reserved								56

DIF Strip Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info	DIF Status	Status	0
Fault Address								8
Application Tag		Application Tag Mask		Reference Tag				16
Unused								24

8.3.17 DIF Update

The DIF Update operation, 0x15, copies memory from the Source Address to the Destination Address. It optionally computes the Data Integrity Field (DIF) on the source data and compares the computed DIF to the DIF contained in the data. It simultaneously computes the DIF on the source data using Destination DIF fields in the descriptor and inserts the computed DIF into the output data.

The number of source bytes read is given by Transfer Size. DIF computation is performed on each block of source data that is 512, 520, 4096, or 4104 bytes. The transfer size should be a multiple of the source block size plus 8 bytes for each source block. The number of bytes written to the destination is the same as the transfer size. There is no alignment requirement for the memory addresses. If the source and destination regions overlap, it is an error.

If the operation completes successfully, the final source and destination Reference Tags and Application Tags are written to the completion record along with a Success completion status. If the operation is partially completed due to a page fault, updated values of the source and destination Reference Tags and Application Tags are written to the completion record along with the page fault information. If software corrects the fault and resumes the operation, it may copy these fields into the continuation descriptor. If the operation fails due to any other error, these fields are undefined.

If an error is detected in the DIF in the source data, the operation stops. The Status field in the completion record is set to DIF Error, the DIF Status field is set to indicate the type of error, and the Bytes Completed field is set to the number of source bytes successfully processed (including generated DIF bytes). Bytes Completed does not include the block in which the error was detected.

See Appendix B for details of DIF computation and checking.

DIF Update Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved	PASID			0
Completion Record Address								8
Source Address								16
Destination Address								24
Reserved		Completion Interrupt Handle		Transfer Size				32
Reserved					DIF Flags	Dest DIF Flags	Source DIF Flags	40
Source Application Tag Seed		Source Application Tag Mask		Source Reference Tag Seed				48
Destination Application Tag Seed		Destination Application Tag Mask		Destination Reference Tag Seed				56

DIF Update Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused	Fault Info	DIF Status	Status	0
Fault Address								8
Source Application Tag		Source Application Tag Mask		Source Reference Tag				16
Destination Application Tag		Destination Application Tag Mask		Destination Reference Tag				24

8.3.17.1 DIF Flags

Bits	Description
7:4	Reserved.
3	**Invert CRC Result** **0:** Do not invert CRC result. **1:** Invert CRC result. (That is, invert each bit of the final CRC value.)
2	**Invert CRC Seed** **0:** The initial seed is 0. **1:** The initial seed is 0xffff.
1:0	**DIF Block Size** **00b:** 512 bytes **01b:** 520 bytes **10b:** 4096 bytes **11b:** 4104 bytes

8.3.17.2 Source DIF Flags

Bits	Description
7	**Source Reference Tag Type** This field denotes the type of operation to perform on the source DIF Reference Tag. **0:** Incrementing **1:** Fixed
6	**Reference Tag Check Disable** **0:** Enable Reference Tag field checking. **1:** Disable Reference Tag field checking.
5	**Guard Check Disable** **0:** Enable Guard field checking. **1:** Disable Guard field checking.
4	**Source Application Tag Type** This field denotes the type of operation to perform on the source DIF Application Tag. **0:** Fixed **1:** Incrementing Note that the meaning of the Application Tag Type is reversed compared to the Reference Tag Type. The default typically used in storage systems is for the Application Tag to be fixed and the Reference Tag to be incrementing.
3	**Application and Reference Tag F Detect** **0:** Disable F Detect for Application Tag and Reference Tag fields. **1:** Enable F Detect for Application Tag and Reference Tag fields. When all bits of both the Application Tag and Reference Tag fields are equal to 1, the Application Tag and Reference Tag checks are not done and the Guard field is ignored.
2	**Application Tag F Detect** **0:** Disable F Detect for the Application Tag field. **1:** Enable F Detect for the Application Tag field. When all bits of the Application Tag field of the source Data Integrity Field are equal to 1, the Application Tag check is not done and the Guard field and Reference Tag field are ignored.

Bits	Description
1	**All F Detect** **0:** Disable All F Detect. **1:** Enable All F Detect. When all bits of the Application Tag, Reference Tag, and Guard fields are equal to 1, no checks are performed on these fields. (The All F Detect Status is reported, if enabled.)
0	**Enable All F Detect Error** **0:** Disable All F Detect Error. **1:** Enable All F Detect Error. When all bits of the Application Tag, Reference Tag, and Guard fields are equal to 1, All F Detect Error is reported in the DIF Status field of the Completion Record. If All F Detect flag is 0, this flag is ignored.

8.3.17.3 Destination DIF Flags

Bits	Description
7	**Destination Reference Tag Type** This field denotes the type of operation to perform on the destination DIF Reference Tag. **0:** Incrementing **1:** Fixed
6	**Reference Tag Pass-through** **0:** The Reference Tag field written to the destination is determined based on the Destination Reference Tag Seed and Destination Reference Tag Type fields of the descriptor. **1:** The Reference Tag field from the source is copied to the destination. The Destination Reference Tag Seed and Destination Reference Tag Type fields of the descriptor are ignored. This field is ignored for the DIF Insert operation.
5	**Guard Field Pass-through** **0:** The Guard field written to the destination is computed from the source data. **1:** The Guard field from the source is copied to the destination. This field is ignored for the DIF Insert operation.
4	**Destination Application Tag Type** This field denotes the type of operation to perform on the destination DIF Application Tag. **0:** Fixed **1:** Incrementing Note that the meaning of the Application Tag Type is reversed compared to the Reference Tag Type. The default typically used in storage systems is for the Application Tag to be fixed and the Reference Tag to be incrementing.
3	**Application Tag Pass-through** **0:** The Application Tag field written to the destination is determined based on the Destination Application Tag Seed, Destination Application Tag Mask, and Destination Application Tag Type fields of the descriptor. **1:** The Application Tag field from the source is copied to the destination. The Destination Application Tag Seed, Destination Application Tag Mask, and Destination Application Tag Type fields of the descriptor are ignored. This field is ignored for the DIF Insert operation.
2:0	Reserved.

8.3.17.4 DIF Status

Completion Record Offset: 1; Size: 1 byte

This field reports the status of a DIF operation. This field is defined only for DIF Check, DIF Strip, and DIF Update operations and only if the Status field of the Completion Record is DIF Error. The values 0x01, 0x02, and 0x04 may be combined when more than one error is detected for a single block.

0x01	Guard mismatch. This value is reported under the following condition: - Guard Check Disable is 0; - F Detect condition is not detected; and - The guard value computed from the source data does not match the Guard field in the source Data Integrity Field.
0x02	Application Tag mismatch. This value is reported under the following condition: - Source Application Tag Mask is not equal to 0xFFFF; - F Detect condition is not detected; and - The computed Application Tag value does not match the Application Tag field in the source Data Integrity Field.
0x04	Reference Tag mismatch. This value is reported under the following condition: - Reference Tag Check Disable is 0. - F Detect condition is not detected; and - The computed Application Tag value does not match the Application Tag field in the source Data Integrity Field.
0x08	All F Detect Error. This value is reported under the following condition: - All F Detect is 1; - Enable All F Detect Error is 1; - All bits of the Application Tag, Reference Tag, and Guard fields of the source Data Integrity Field are equal to 1.

F Detect condition is detected when one of the following is true:

All F Detect = 1	All bits of the Application Tag, Reference Tag, and Guard fields of the source Data Integrity Field are equal to 1.
Application Tag F Detect = 1	All bits of the Application Tag field of the source Data Integrity Field are equal to 1.
Application and Reference Tag F Detect = 1	All bits of both the Application Tag and Reference Tag fields of the source Data Integrity Field are equal to 1.

8.3.18 DIX Generate

The DIX Generate operation, 0x17, computes the Data Integrity Field (DIF) on the source data and writes only the computed DIF for each source block to the Destination Address. The source data is not copied to the destination region.

The number of source bytes is given by Transfer Size. DIF computation is performed on each block of source data that is 512, 520, 4096, or 4104 bytes. The transfer size should be a multiple of the source block size. If the operation completes successfully, the number of bytes written to the destination is the number of source blocks multiplied by 8. There is no alignment requirement for the Source Address, but the Destination Address must be aligned to a multiple of 8. If the source and destination regions overlap, it is an error.

If the operation completes successfully, the final Reference Tag and Application Tag are written to the completion record along with a Success completion status. If the operation is partially completed due to a page fault, updated values of Reference Tag and Application Tag are written to the completion record along with the page fault information. If software corrects the fault and resumes the operation, it may copy these fields into the continuation descriptor. The Completion Record Address Valid and Request Completion Record flags must be 1 and the Completion Record Address must be valid.

See section 8.3.17, DIF Update, for a description of DIF Flags, Destination DIF Flags, and the fields in the completion record. See Appendix B for details of DIF computation.

DIX Generate Descriptor

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td>Operation</td><td colspan="3">Flags</td><td>Priv</td><td>Reserved</td><td colspan="2">PASID</td><td>0</td></tr>
<tr><td colspan="8">Completion Record Address</td><td>8</td></tr>
<tr><td colspan="8">Source Address</td><td>16</td></tr>
<tr><td colspan="8">Destination Address</td><td>24</td></tr>
<tr><td colspan="2" rowspan="3">Reserved</td><td colspan="2">Completion Interrupt Handle</td><td colspan="4">Transfer Size</td><td>32</td></tr>
<tr><td colspan="2" rowspan="2">Reserved</td><td>DIF Flags</td><td>Dest DIF Flags</td><td colspan="2">Reserved</td><td>40</td></tr>
<tr><td colspan="4">Reserved</td><td>48</td></tr>
<tr><td colspan="2">Application Tag Seed</td><td colspan="2">Application Tag Mask</td><td colspan="4">Reference Tag Seed</td><td>56</td></tr>
</table>

DIX Generate Completion Record

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td colspan="4">Bytes Completed</td><td colspan="3">Unused</td><td>Status</td><td>0</td></tr>
<tr><td colspan="8">Fault Address</td><td>8</td></tr>
<tr><td colspan="8">Unused</td><td>16</td></tr>
<tr><td colspan="2">Application Tag</td><td colspan="2">Application Tag Mask</td><td colspan="4">Reference Tag</td><td>24</td></tr>
</table>

8.3.19 Cache Flush

The Cache Flush operation, 0x20, flushes the processor caches at the Destination Address. The number of bytes flushed is given by Transfer Size. The transfer size does not need to be a multiple of the cache line size. There are no alignment requirements for the destination address or the transfer size. Any cache line that is partially covered by the destination region is flushed.

Cache Flush Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved	PASID			0
Completion Record Address								8
Reserved								16
Destination Address								24
Reserved		Completion Interrupt Handle		Transfer Size				32
								40
								48
								56

Cache Flush Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed					Fault Info		Status	0
Fault Address								8
Unused								16
								24

8.3.20 Inter-Domain Operations

This section describes the details of the descriptors for the Inter-Domain operations described in section 3.13.3. The following operations support Inter-Domain capabilities:

- Inter-Domain Operations
    - Inter-Domain Memory Copy
    - Inter-Domain Fill
    - Inter-Domain Compare
    - Inter-Domain Compare Pattern
    - Inter-Domain Cache Flush
- Update Window

8.3.20.1 Generic Descriptor Format

As shown in the figure below, the descriptor for an Inter-Domain operation allows software to specify an IDPT handle for each source or destination address. The IDPT handle must reference a type 0 SASS or type 1 SAMS IDPTE (described in section 3.14.1). There must be at least one valid IDPT handle indicated by the corresponding flags bit set to 1.

Restricted Inter-Domain Memory Op Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Address1								16
Address2								24
Reserved		Completion Interrupt Handle		Transfer Size				32
Reserved								40
								48
Address2 IDPT Handle		Address1 IDPT Handle						56

8.3.20.2 Completion Record

This section defines the completion record format for Inter-Domain operations.

If an Inter-Domain operation is partially completed due to a page fault, and the fault address references the descriptor PASID, the completion record reports the fault address where the page fault was encountered.

If the fault address references an alternate PASID, or if the operation fails because of an invalid IDPT handle, a misconfigured IDPTE or insufficient permissions, the completion record specifies the IDPT handle associated with the faulting address or error.

If the Window Mode field in the IDPTE is set to Offset Mode, the completion record does not report the fault address in the alternate PASID address space, and the Fault Address Masked bit in the Fault Info field is reported as 1. The Fault Info field indicates the operand that caused the fault (described in section 8.2.3), and software can use the bytes completed field to identify the location of the fault relative to the starting address or offset specified in the descriptor.

The submitter is expected to use software mechanisms to communicate with the process whose memory window experienced the page fault and resolve it prior to resuming the operation. Alternately the submitter can set the Block on Fault flag in the descriptor to cause the device to wait for the page fault to be resolved before continuing with the operation.

The Result field is used for Inter-Domain Compare and Compare Pattern. The Result field is not used for Inter-Domain Copy, Fill, or Cache Flush.

Restricted Inter-Domain Operation Completion Record

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td colspan="4">Bytes Completed</td><td>Unused</td><td>Fault Info</td><td>Result</td><td>Status</td><td>0</td></tr>
<tr><td colspan="8">Fault Address</td><td>8</td></tr>
<tr><td colspan="6" rowspan="2">Unused</td><td colspan="2">Fault IDPT Handle</td><td>16</td></tr>
<tr><td colspan="2"></td><td>24</td></tr>
</table>

8.3.20.3 Inter-Domain Copy

The Inter-Domain Copy operation, 0x23, copies memory from the Source Address to the Destination Address. At least one of Source or Destination Address must be qualified with an IDPT handle and it must reference an Inter-Domain Permissions Table entry of one of the types specified in section 3.14.1. Software does this by setting at least one of the flags described in the table below.

The number of bytes copied is given by Transfer Size. There are no alignment requirements for the memory addresses or the transfer size.

If the regions overlap in physical memory, the behavior is undefined.

The completion record format is described in section 8.3.20.2.

Restricted Inter-Domain Memory Copy Descriptor

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td>Operation</td><td colspan="3">Flags</td><td>Priv Reserved</td><td colspan="3">PASID</td><td>0</td></tr>
<tr><td colspan="8">Completion Record Address</td><td>8</td></tr>
<tr><td colspan="8">Source Address</td><td>16</td></tr>
<tr><td colspan="8">Destination Address</td><td>24</td></tr>
<tr><td colspan="2">Reserved</td><td colspan="2">Completion Interrupt Handle</td><td colspan="4">Transfer Size</td><td>32</td></tr>
<tr><td colspan="8" rowspan="2">Reserved</td><td>40</td></tr>
<tr><td>48</td></tr>
<tr><td colspan="2">Destination IDPT Handle</td><td colspan="2">Source IDPT Handle</td><td colspan="4"></td><td>56</td></tr>
</table>

Bits	Description
23:18	**Reserved:** Must be 0.
17	Use Alternate Destination PASID **0:** The Destination IDPT Handle field is not used. **1:** The Destination IDPT Handle field references an Inter-Domain Permissions Table entry which specifies the PASID used to access the Destination Address.
16	Use Alternate Source PASID **0:** The Source IDPT Handle field is not used. **1:** The Source IDPT Handle field references an Inter-Domain Permissions Table entry which specifies the PASID used to access the Source Address.

Table 8-12: Inter-Domain Copy Operation-Specific Flags

8.3.20.4 Inter-Domain Fill

The Inter-Domain Fill operation, 0x24, fills memory at the Destination Address with the value in the pattern field. A Destination IDPT handle must be specified, and it must reference an Inter-Domain Permissions Table entry of one of the types specified in section 3.14.1.

The completion record format is described in section 8.3.20.2.

The description of the pattern and other software requirements, along with reporting of completions are identical to the Fill operation described in section 8.3.5.

Restricted Inter-Domain Fill Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Pattern Lower								16
Destination Address								24
		Completion Interrupt Handle		Transfer Size				32
Pattern Upper								40
								48
Destination IDPT Handle		Reserved						56

Bits	Description
23:19	**Reserved:** Must be 0.
18	**Pattern Size** **0:** Pattern size is 8B and specified in the Pattern Lower field. The Pattern Upper field is reserved. **1:** Pattern size is 16B and specified by the Pattern Lower and Pattern Upper fields. This field must be 0 if Fill16 Support in GENCAP is 0.
17	**Use Alternate Destination PASID** This field must be 1. The Destination IDPT Handle field references an Inter-Domain Permissions Table entry which specifies the PASID used to access the Destination Address.
16	**Reserved:** Must be 0.

Table 8-13: Inter-Domain Fill Operation-Specific Flags

8.3.20.5 Inter-Domain Compare

The Inter-Domain Compare operation, 0x25, compares memory at Source1 Address with memory at Source2 Address. At least one of Source1 or Source2 Address must be qualified with an IDPT handle and it must reference an Inter-Domain Permissions Table entry of one of the types specified in section 3.14.1. Software does this by setting at least one of the flags described in the table below.

The completion record format is described in section 8.3.20.2.

The reporting of results from this operation is identical to the Compare operation described in section 8.3.6.

Restricted Inter-Domain Compare Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source1 Address								16
Source2 Address								24
		Completion Interrupt Handle		Transfer Size				32
				Reserved			Expected Result	40
								48
Source2 IDPT Handle		Source1 IDPT Handle						56

Bits	Description
23:18	**Reserved:** Must be 0.
17	**Use Alternate Source2 PASID** **0:** The Source2 IDPT Handle field is not used. **1:** The Source2 IDPT Handle field references an Inter-Domain Permissions Table entry which specifies the PASID used to access Source2 Address.
16	**Use Alternate Source1 PASID** **0:** The Source1 IDPT Handle field is not used. **1:** The Source1 IDPT Handle field references an Inter-Domain Permissions Table entry which specifies the PASID used to access Source1 Address.

Table 8-14: Inter-Domain Compare Operation-Specific Flags

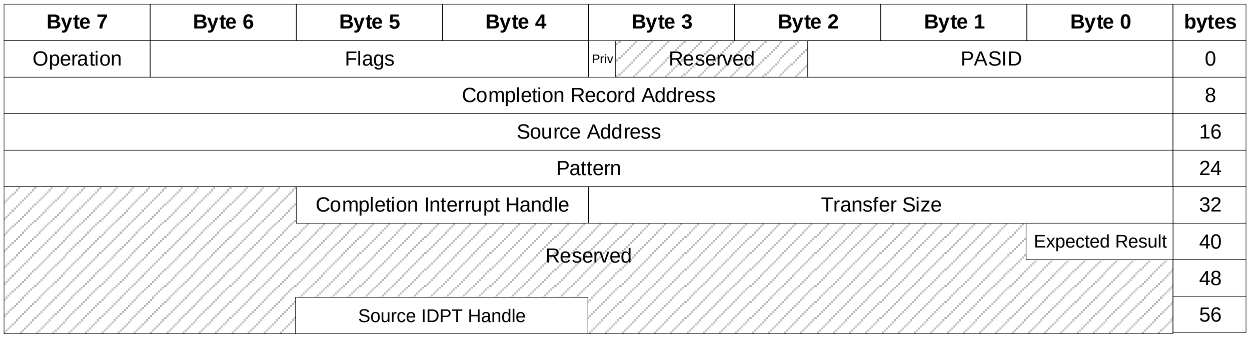
8.3.20.6 Inter-Domain Compare Pattern

The Inter-Domain Compare Pattern operation, 0x26, compares memory at Source Address with the value in the pattern field. The Source Address must be qualified with an IDPT handle and it must reference an Inter-Domain Permissions Table entry of one of the types specified in section 3.14.1.

The completion record format is described in section 8.3.20.2.

The description of the pattern in the descriptor, and the reporting of results from this operation are identical to the Compare Pattern operation described in section 8.3.7. The behavior of Check Result and Expected Result are identical to the Compare operation in section 8.3.6.

Restricted Inter-Domain Compare Pattern Descriptor



Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Source Address								16
Pattern								24
		Completion Interrupt Handle		Transfer Size				32
Reserved							Expected Result	40
								48
		Source IDPT Handle						56

Bits	Description
23:17	**Reserved:** Must be 0.
16	**Use Alternate Source PASID** This field must be 1. The Source IDPT Handle field references an Inter-Domain Permissions Table entry which specifies the PASID used to access the Source Address.

Table 8-15: Inter-Domain Compare Pattern Operation-Specific Flags

8.3.20.7 Inter-Domain Cache Flush

The Inter-Domain Cache Flush operation, 0x27, flushes the processor caches at the Destination Address. The Destination Address must be qualified with an IDPT handle and it must reference an Inter-Domain Permissions Table entry of one of the types specified in section 3.14.1. The referenced IDPTE may be configured with either read or write permission enabled.

The completion record format is described in section 8.3.20.2.

Other details of the operation are identical to the Cache Flush operation described in section 8.3.18.

Restricted Inter-Domain Cache Flush Descriptor

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Operation	Flags			Priv Reserved		PASID		0
Completion Record Address								8
Reserved								16
Destination Address								24
		Completion Interrupt Handle		Transfer Size				32
								40
								48
Destination IDPT Handle		Reserved						56

Bits	Description
23:18	**Reserved:** Must be 0.
17	**Use Alternate Destination PASID** This field must be 1. The Destination IDPT Handle field references an Inter-Domain Permissions Table entry which specifies the PASID used to access the Destination Address.
16	**Reserved:** Must be 0.

Table 8-16: Inter-Domain Cache Flush Operation-Specific Flags

8.3.20.8 Update Window

The Update Window operation, 0x21, atomically modifies attributes of the memory window associated with the specified Inter-Domain Permissions Table entry. The descriptor PASID must match the access PASID in the entry referenced by the handle, and the Allow Update bit in the entry must be 1. There are no alignment requirements for the Window Base Address or the Window Size fields. If the Window Enable field in Window Flags is 1, the sum of Window Base Address and Window Size in the descriptor must be less than or equal to 2^{64}. If Window Enable is 0, then the Window Mode, Window Base Address and Window Size fields in the descriptor must be 0.

As described in section 3.14.4, an implicit drain is performed to flush out any in-flight descriptors that are still using pre-update window attributes. Software can use the Suppress Drain flag to avoid the implicit drain if necessary.

An Update Window descriptor may not be included in a batch; it is treated as an unsupported operation type.

Update Window Descriptor

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td>Operation</td><td colspan="3">Flags</td><td>Priv</td><td>Reserved</td><td colspan="2">PASID</td><td>0</td></tr>
<tr><td colspan="8">Completion Record Address</td><td>8</td></tr>
<tr><td colspan="8">Window Base Address</td><td>16</td></tr>
<tr><td colspan="8">Window Size</td><td>24</td></tr>
<tr><td colspan="2"></td><td colspan="2">Completion Interrupt Handle</td><td colspan="4"></td><td>32</td></tr>
<tr><td colspan="8"></td><td>40</td></tr>
<tr><td colspan="8">Reserved</td><td>48</td></tr>
<tr><td colspan="2">IDPT Handle</td><td>Window Flags</td><td colspan="5"></td><td>56</td></tr>
</table>

Update Window Completion Record

<table>
<tr><th>Byte 7</th><th>Byte 6</th><th>Byte 5</th><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th><th>Byte 0</th><th>bytes</th></tr>
<tr><td colspan="7"></td><td>Status</td><td>0</td></tr>
<tr><td colspan="8" rowspan="3">Unused</td><td>8</td></tr>
<tr><td>16</td></tr>
<tr><td>24</td></tr>
</table>

Bits	Description
7:4	**Reserved:** Must be 0.
3	**Window Mode** **0:** Window operates in Address Mode. **1:** Window operates in Offset Mode. See section 3.14.3 for details. This field is reserved if Window Enable is 0 or if Offset Mode Support in IDCAP is 0.
2	**Window Enable** **0:** The window address range checks are disabled, and hardware will not perform range checks on the incoming address. **1:** The window address range checks are enabled, and hardware will perform range checks on the incoming address based on the window mode.
1	**Write Permissions** **0:** Disallows memory write using this entry. **1:** Allows memory writes using this entry.
0	**Read Permissions** **0:** Disallows memory reads using this entry. **1:** Allows memory reads using this entry.

Table 8-17: Window Flags

Bits	Description
23:17	**Reserved:** Must be 0.
16	**Suppress Drain** **0:** Drain any descriptors using prior values of the IDPTE fields modified by this descriptor. **1:** No drain is performed. This field is reserved if Update Window Suppress Drain Support in IDCAP is 0.

Table 8-18: Update Window Operation-Specific Flags

§

9 Register Descriptions

The programming interface for the Intel Data Streaming Accelerator consists of PCI configuration registers and MMIO registers, which include configuration and control registers and work submission portals. The base addresses for the MMIO registers and portals are specified by two Base Address Registers (BARs) in PCI config space.

PCI config space accesses must be performed as aligned 1-, 2-, or 4-byte accesses. See the PCI Express Base Specification listed in section 1.2 for rules on accessing unimplemented registers and reserved bits in PCI config space.

MMIO space accesses to the BAR0 region (capability, configuration, and status registers) must be performed as aligned 1-, 2-, 4- or 8-byte accesses. Software may use 8-byte accesses for any registers, including accessing two adjacent 32-bit registers with a single 8-byte access.

MMIO space accesses to the BAR2 region must be performed as 64-byte accesses, as described in section 9.3.

This chapter uses the following abbreviations for register attributes.

Attribute	Abbreviation	Description
Read/Write	RW	The field can be read and written by software. The value read always matches the value last written. Bits that are reserved or not supported by an implementation may be hardwired to 0.
Read/Write/Lock	RWL	The field is read-write at some times and read-only at other times. The specification of each register or field describes when it is read-only. The value read always matches the value last written. Bits that are reserved or not supported by an implementation may be hardwired to 0.
Read Only	RO	The field is set by the hardware and software can only read it. In some cases, the field has a fixed value (e.g., in a capability register), and in some cases the field reports status that can change during device operation. Writes to the field have no effect.
Write Only	WO	The field is only writeable by software. Reads return 0.
Read/Write-1-to-Clear	RW1C	The field can be read or cleared by software. To clear an RW1C bit, software writes a one to it. Writing a zero to an RW1C bit has no effect.
Read Only Sticky	ROS	The field reports status and software can only read it. Writes to the register have no effect. The field is not cleared on reset.
Read/Write-1-to-Clear Sticky	RW1CS	The field behaves the same as RW1C except that it is not cleared on reset.
Reserved	RSVD	Read as 0. Ignored on writes. Software must write 0 for compatibility with future expansion.
Read/Write/Volatile	RWV	The field can be read and written by software. The value may be changed by hardware, so the value read may not match the last value written.

Read/Write/Lock/Volatile	RWLV	The field is read-write at some times and read-only at other times. The specification of each register or field describes when it is read-only. The value may be changed by hardware, so the value read may not match the last value written.

Table 9-1: Register Attributes

9.1 PCI Configuration Space Registers

This section provides Intel DSA specific details about some of the PCI configuration registers. See Appendix C and the PCI Express specification listed in section 1.2 for a complete specification of these registers.

9.1.1 Base Address Registers (BAR)

Intel DSA PCI configuration space implements two 64-bit BARs.

9.1.1.1 BAR0 (Device Control Registers)

BAR0 is a 64-bit BAR that contains the physical base address of device control registers. These registers provide information about device capabilities, controls to configure and enable the device, and device status. These registers are described in more detail in the following sections. The size of the BAR0 region depends on the specific device implementation, and is at least 64KB.

9.1.1.2 BAR2 (Portals)

BAR2 is a 64-bit BAR that contains the physical base address of the portals that are used to submit descriptors to the device. Each portal is 64 bytes in size and is located on a separate 4 KB page. This allows the portals to be independently mapped into different address spaces using CPU page tables.

There are 4 portals per WQ, as described in section 3.3. So, for example, if the device supports 8 WQs, the size of BAR2 would be 8 × 4 × 4 KB = 128 KB. If the size is not a power of two, the total size of BAR2 is rounded up to the next power of two.

Any write to an address within the BAR2 region that does not correspond to a WQ portal is ignored; for DMWr, a Retry response is returned. Any read operation to the BAR2 address space returns either 0x00 or 0xFF for all bytes.

9.1.2 MSI-X Capability

MSI-X is the only PCI Express interrupt capability that Intel DSA provides. It does not implement legacy PCI interrupts or MSI. Details of this register structure are in the PCI Express specification. See section 3.7 for information on how the MSI-X table is used.

9.1.3 Address Translation Capabilities

Three PCI Express capabilities control address translation. If any of these capabilities are changed by software while the device is not Disabled, the device enters the Halt state and an error is reported in the Software Error register.

PASID	ATS	PRS	Operation
1	1	1	Addresses are translated with or without PASID, depending on the work queue configuration. (See section 9.2.24.) Recoverable page faults are supported. This is the recommended mode. This mode must be used to allow user-mode access to the device or to allow sharing among multiple guests in a virtualized system.
0	1	0	Addresses are translated using the BDF of the device. PASID is not used. Translation failures are not recoverable. This mode may be used when address translation is enabled in the IOMMU but the device is only used by the kernel or by a single guest kernel in a virtualized platform.
0	0	0	All memory accesses are Untranslated Accesses without PASID. The Address Translation Cache is not used. This mode is recommended only when IOMMU address translation is disabled.
1	0	0	All memory accesses are Untranslated Accesses, with or without PASID, depending on the WQ configuration. The Address Translation Cache is not used.
1	1	0	Addresses are translated with or without PASID, depending on the WQ configuration. Translation failures are not recoverable.
0	1	1	Addresses are translated using the BDF of the device. PASID is not used. Recoverable page faults are supported.
0	0	1	Page requests are never generated when ATS is disabled, so these modes are not useful; PRS Enable is ignored.
1	0	1	

Table 9-2: Address Translation Modes

9.1.3.1 PASID Capability

Software configures the PASID capability to control whether the device uses PASID to perform address translation. If PASID is disabled, shared virtual memory (SVM) is not supported, only dedicated WQs (DWQs) may be used, and the device cannot be shared across multiple VMs. PASID must always be enabled to use shared WQs (SWQs). If PASID is enabled, address translation is performed using PASID according to the IOMMU configuration.

9.1.3.2 ATS Capability

Software configures the ATS capability to control whether the device should translate addresses before performing memory accesses. If address translation is enabled in the IOMMU, ATS must be enabled in the device to obtain acceptable system performance. If the device ATS capability is enabled and if WQ ATS Support in WQCAP is 1, software can optionally disable the use of ATS independently for each WQ by setting WQ ATS Disable in WQCFG (described in Section 9.2.24). If address translation is not enabled in the IOMMU, ATS must be disabled. If ATS is disabled, all memory accesses are performed using Untranslated Accesses.

9.1.3.3 PRS Capability

Software configures the PRS capability to control whether the device can request a page when an address translation fails. If the device PRS capability is enabled and if WQ PRS Support in WQCAP is 1, software can optionally disable the use of PRS independently for each WQ by setting WQ PRS Disable in WQCFG (described in Section 9.2.24).

9.1.4 Scalable I/O Virtualization Capability

The Scalable I/O Virtualization capability is defined in the Intel® Scalable I/O Virtualization Architecture Specification. It indicates that Intel DSA supports Scalable IOV. The fields are filled in as follows:

Function Dependency Link	<self>
Flags	0
Supported page sizes	1
System Page Size	1
IMS	1 if IMS is supported; 0 otherwise.

9.1.5 VC Capability

If the PCIe VC capability is present, software configures the TC/VC mapping in the PCI Express VC capability to control the mapping of different Traffic Classes to the corresponding platform and internal I/O fabric resources. Use of traffic classes is described in more detail in section 4.2.

9.2 Configuration and Control Registers (BAR0)

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
▨	▨	▨	▨	Version				00h
▨	▨	▨	▨	▨	▨	▨	▨	08h
General Capabilities								10h
▨	▨	▨	▨	▨	▨	▨	▨	18h
Work Queue Capabilities								20h
▨	▨	▨	▨	▨	▨	▨	▨	28h
Group Capabilities								30h
Engine Capabilities								38h
Operations Capabilities								40h
								48h
								50h
								58h
Table Offsets								60h
								68h
▨	▨	▨	▨	▨	▨	▨	▨	70h
▨	▨	▨	▨	▨	▨	▨	▨	78h
▨	▨	▨	▨	General Configuration				80h
▨	▨	▨	▨	General Control				88h
▨	▨	▨	▨	General Status				90h
▨	▨	▨	▨	Interrupt Cause				98h
▨	▨	▨	▨	Command				A0h
▨	▨	▨	▨	Command Status				A8h
▨	▨	▨	▨	Command Capabilities				B0h
▨	▨	▨	▨	▨	▨	▨	▨	B8h
Software Error								C0h
								C8h
								D0h
								D8h
Event Log Configuration								E0h
								E8h
Event Log Status								F0h
▨	▨	▨	▨	▨	▨	▨	▨	F8h
Inter-PASID Capabilities								100h
▨	▨	▨	▨	Inter-PASID Bitmap Register				108h

Continued on the next page.

Figure 9-1: MMIO Register Map

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
▨				MSI-X Permissions Table[1]				400h
				...				408h
▨								
Group Configuration Table[1]								600h
...								640h
▨								
Work Queue Configuration Table[1]								800h
...								820h
▨								
Performance Monitoring Registers[1]								2000h
								2010h
▨								
MSI-X Table[2]								4000h
								4010h
...								
▨								
MSI-X Pending Bit Array[2]								5000h
▨								
IMS Table[1]								8000h
								8010h
...								
▨								
Inter-PASID Table[1]								9000h
								9008h
▨								
Admin Command Parameters								E000h
▨								E008h
Dummy Portal								F000h
								FFF8h

[1] The offset shown is an example. The actual offset of this table is given in the Table Offsets register.

[2] The offset shown is an example. The actual offset of this table is given in the PCIe MSI-X capability.

The initial values of MMIO-space registers are as follows:

Register	Initial Value			
	Power-On Reset	Warm Reset	Function-Level Reset	Software Reset
Version General Capabilities WQ Capabilities Group Capabilities Engine Capabilities Operations Capabilities Command Capabilities Perfmon Capabilities Inter-Domain Capabilities Table Offsets	Contain read-only values indicating capabilities of the device.			
General Configuration General Control General Status Interrupt Cause MSI-X Pending Bit Array MSI-X Permissions Table Event Log Configuration Command Parameter Inter-Domain Permissions Table Inter-Domain Bitmap	0	0	0	0
WQ Configuration[1]	OPCFG: Initialized to match the value in OPCAP All other fields: 0			
Group Configuration[1]	Read Buffers Allowed: initialized to Total Read Buffers All other fields: 0			
Command Command Status Software Error Event Log Status	0	0	0	Preserved
Perfmon	PASID Filter Configuration Registers: Initialized to 0x3FFFFF. Other Filter Configuration Registers: Initialized to 0xFFFF. All other registers: 0			
MSI-X Table	Message Data: 0 Message Address: 0 Mask: 1			Preserved
Interrupt Message Storage	Message Data: 0 Message Address: 00000000FEE00000 Mask: 1 PASID, PASID Enable, Ignore, Pending: 0			

Table 9-3: MMIO Register Initial Values

The following MMIO-space registers are read-only under the described conditions:

[1] If the Configuration Support field in GENCAP is 0, the initial values of the WQ Configuration and Group Configuration registers reflect the fixed configuration of the groups and WQs.

Register	Conditions Under Which Register is Read-Only
General Configuration Group Configuration	While device is not Disabled.
WQ Configuration	See Table 9-7.
Perfmon	See Table 9-8.
Inter-Domain Permissions Table	See Table 9-10.
Event Log Configuration	While device is not Disabled.

Table 9-4: Read-Only MMIO Registers

9.2.1 Version Register (VERSION)

The Version register reports the version of this architecture specification that is supported by the device.

<table>
<tr><td colspan="4">VERSION
Base: BAR0 Offset: 0x0 Size: 4 bytes (32 bits)</td></tr>
<tr><th>Bit</th><th>Attr</th><th>Size</th><th>Description</th></tr>
<tr><td>31:16</td><td>RO</td><td>16 bits</td><td>Unused.</td></tr>
<tr><td>15:8</td><td>RO</td><td>8 bits</td><td>Major version</td></tr>
<tr><td>7:0</td><td>RO</td><td>8 bits</td><td>Minor version</td></tr>
</table>

9.2.2 General Capabilities Register (GENCAP)

GENCAP Base: BAR0			Offset: 0x10 Size: 8 bytes (64 bits)
Bit	Attr	Size	Description
63:55	RO	9 bits	Unused
54	RO	1 bit	Strict Ordering Limitation for Peer Destinations 0: Strict ordering is supported for descriptors whose destination address references a peer device. 1: The Strict Ordering flag is reserved in descriptors whose destination address references a peer device.
53	RO	1 bit	Strict Ordering Limitation for Memory Destinations 0: Strict ordering is supported for descriptors whose destination address references memory. 1: The Strict Ordering flag is reserved in descriptors whose destination address references memory.
52:33	RO	20 bits	Unused
32	RO	1 bit	Event Log Overflow Support 0: Event Log does not overflow. If the Event Log is full at the time hardware attempts to append an entry, the device blocks until the Event Log Head field is updated by software. 1: Event Log may overflow. If the Event Log is full at the time hardware attempts to append an entry, hardware drops the event, and attempts to log an error in SWERROR to indicate the Event Log full condition. If the Valid bit in SWERROR is 1 and the Event Log is full at the time an event occurs, the Overflow bit is SWERROR is set to 1.
31	RO	1 bit	Configuration Support 0: General Configuration, Group Configuration, and some fields of the Work Queue Configuration registers are read-only and reflect the fixed configuration of the device. See section 9.2.24 for details about which WQ configuration fields are read-only. 1: General Configuration, Group Configuration, and Work Queue Configuration registers are read-write and can be used by software to set the desired configuration.
30:25	RO	6 bits	Interrupt Message Storage Size The number of entries in the Interrupt Message Storage is N × 256, where N is the value in this field. If the Interrupt Message Storage Support field in the Scalable IOV capability is 0, the value in this field is undefined.
24:21	RO	4 bits	Maximum Supported Batch Size The maximum number of descriptors that can be referenced by a Batch descriptor is independently controlled for each WQ. This field indicates the maximum value that each WQ can be configured with. The maximum supported batch size is 2^N, where N is the value in this field. If Batch descriptor is not supported (as indicated by bit 1 of OPCAP), this field is unused.

GENCAP Base: BAR0		Offset: 0x10	Size: 8 bytes (64 bits)
Bit	Attr	Size	Description
20:16	RO	5 bits	Maximum Supported Transfer Size The maximum transfer size that can be specified in a descriptor is independently controlled for each WQ. This field indicates the maximum value that each WQ can be configured with. The maximum supported transfer size is 2^N, where N is the value in this field.
15	RO	1 bit	Batch Continuation Support Indicates support for the Batch Error flag and the Result field in the Batch completion record.
14:13	RO	2 bits	Event Log Support 0: No Event Log Support. 1: 64-byte entries. 2: 128-byte entries. 3: 256-byte entries. See section 5.9 for more information on Event Log.
12	RO	1 bit	Completion Record Fault Info Support 0: Completion Record does not have a Fault Information field. 1: Completion Record has a Fault Information field as specified in section 8.2.3.
11	RO	1 bit	CRC64 Support 0: The CRC Size flag in a CRC Generation or Copy with CRC Generation descriptor is not supported. 1: The CRC Size flag in a CRC Generation or Copy with CRC Generation descriptor is supported.
10	RO	1 bit	Fill16 Support 0: The Pattern Size flag in a Fill or Inter-Domain Memory Fill descriptor is not supported. 1: The Pattern Size flag in a Fill or Inter-Domain Memory Fill descriptor is supported.
9	RO	1 bit	Drain Descriptor Readback Address Support 0: Hardware does not support specification of Readback Addresses in Drain descriptors and the Readback Address Valid flags and the Readback Address fields in the descriptor are reserved. 1: Hardware supports specification of Readback Addresses in Drain descriptors. If the corresponding Readback Address Valid flags are set, hardware will issue the corresponding readbacks.
8	RO	1 bit	Destination Readback Support 0: The Destination Readback flag in descriptors is not supported. 1: The Destination Readback flag in descriptors is supported.
7	RO	1 bit	Translation Fetch Stride Support 0: The Use Stride flag in a Translation Fetch descriptor is not supported. Hardware uses an implementation specific stride value. 1: The Use Stride flag in a Translation Fetch descriptor is supported.

GENCAP Base: BAR0			Offset: 0x10 Size: 8 bytes (64 bits)
Bit	Attr	Size	Description
6	RO	1 bit	Inter-Domain Support 0: Inter-Domain operations are not supported. 1: Inter-Domain operations are supported while the PASID capability is enabled. Inter-Domain capabilities are reported in the IDCAP register (section 9.2.18).
5	RO	1 bit	Unused.
4	RO	1 bit	Command Capabilities Support 0: The Command Capabilities register is not supported. The administrative commands supported are listed in Table 9-6. 1: The Command Capabilities register is supported and reports the set of administrative commands supported by the device. See section 9.2.14 for a description of the Command Capabilities register.
3	RO	1 bit	Unused.
2	RO	1 bit	Cache Control Support 0: Cache control for memory write operations is not supported. The Cache Control flag in descriptors that write to memory is reserved. 1: Cache control for write operations is supported. Software can use the Cache Control flag in descriptors to control the use of cache.
1	RO	1 bit	Overlapping Copy Support 0: Overlapping copies are not supported. If source and destination buffers overlap, it is an error. 1: Overlapping copies are supported by the Memory Move operation. See the description of the Memory Move operation for details of the behavior. Regardless of the value of this field, overlapping copies are not supported by any operation other than Memory Move.
0	RO	1 bit	Block on Fault Support 0: Block on fault is not supported. The Block On Fault Enable bit in the WQCFG registers and the Block On Fault flag in descriptors are reserved. If a page fault occurs on a source or destination memory access, the operation stops and the page fault is reported to software. 1: Block on fault is supported. Behavior on page faults depends on the values of the Block On Fault Enable bit in each WQCFG register and the Block on Fault flag in each descriptor. See section 3.13 for more information on page fault handling.

9.2.3 WQ Capabilities Register (WQCAP)

WQCAP
Base: BAR0 Offset: 0x20 Size: 8 bytes (64 bits)

Bit	Attr	Size	Description
63:56	RO	8 bits	Unused.
55	RO	1 bit	WQ PRS Support 0: The WQ PRS Disable control is not supported. 1: The WQ PRS Disable control may be used to control the use of PRS independently for each WQ.
54	RO	1 bit	WQ Operations Configuration Support 0: Configuration of per WQ operations is not supported. The OPCFG fields in WQCFG are reserved. All WQs support the operations specified in the OPCAP register. 1: Configuration of per WQ operations is supported. The OPCFG fields in WQCFG may be used to limit the operations allowed in each WQ.
53	RO	1 bit	WQ Occupancy Interrupt Support 0: WQ occupancy interrupts are not supported. The WQ Occupancy Limit and WQ Occupancy Interrupt Enable fields in WQCFG are reserved. 1: WQ occupancy interrupts are supported as described in section 9.2.24.
52	RO	1 bit	WQ Occupancy Support 0: The value of the WQ Occupancy field in WQCFG is undefined. 1: The WQ Occupancy field in WQCFG contains the current occupancy of the WQ.
51	RO	1 bit	WQ Priority Support 0: WQ priorities are not supported. The WQ Priority field in WQ configuration is ignored. 1: WQ priorities are supported as described in section 4.1.
50	RO	1 bit	WQ ATS Support 0: ATS is used for all WQs according to the setting of the Enable field in the PCIe ATS capability. 1: The WQ ATS Disable control may be used to control the use of ATS independently for each WQ.
49	RO	1 bit	Dedicated Mode Support 0: Dedicated mode is not supported. All WQs must be configured in shared mode. 1: Dedicated mode is supported.
48	RO	1 bit	Shared Mode Support 0: Shared mode is not supported. All WQs must be configured in dedicated mode. 1: Shared mode is supported.
47:28	RO	20 bits	Unused.
27:24	RO	4 bits	WQCFG Size Indicates the size of the WQCFG register for each WQ. The size of each WQCFG register is 2^{N+5} bytes, where N is the value in this field.
23:16	RO	8 bits	Number of WQs

WQCAP Base: BAR0			Offset: 0x20 Size: 8 bytes (64 bits)
Bit	Attr	Size	Description
15:0	RO	16 bits	Total WQ Size The total amount of work queue space in the device, which may vary in different implementations. Software uses the WQCFG registers to apportion this space among the WQs, to support multiple QoS levels and/or multiple dedicated work queues.

9.2.4 Group Capabilities Register (GRPCAP)

GRPCAP Base: BAR0			Offset: 0x30 Size: 8 bytes (64 bits)
Bit	Attr	Size	Description
63:19	RO	45 bits	Unused.
18	RO	1 bit	**Descriptors in Progress Limit Supported** **0:** The Descriptors in Progress Limit and Batch Descriptors in Progress Limit fields in GRPFLAGS are reserved. **1:** The Descriptors in Progress Limit and Batch Descriptors in Progress Limit fields in GRPFLAGS can be used by software to control the value for each engine in the group.
17	RO	1 bit	**Global Read Buffer Limit Supported** **0:** The Global Read Buffer Limit field of GENCFG and the Use Global Read Buffer Limit field in GRPCFG are reserved. **1:** Global Read Buffer Limit and Use Global Read Buffer Limit can be used by software to control bandwidth usage by selected groups, as described in chapter 4.
16	RO	1 bit	**Read Buffer Controls Supported** **0:** The Read Buffers Allowed and Read Buffers Reserved fields in GRPCFG are read-only and are unused. The value in the Total Read Buffers field is undefined. **1:** Read Buffers Allowed and Read Buffers Reserved are supported as described in chapter 4.
15:8	RO	8 bits	**Total Read Buffers** Indicates the total number of Read Buffers available. See chapter 4 for information on the meaning of this field.
7:0	RO	8 bits	**Number of Groups**

9.2.5 Engine Capabilities Register (ENGCAP)

<table>
<tr><td colspan="4">ENGCAP
Base: BAR0 Offset: 0x38 Size: 8 bytes (64 bits)</td></tr>
<tr><th>Bit</th><th>Attr</th><th>Size</th><th>Description</th></tr>
<tr><td>63:24</td><td>RO</td><td>40 bits</td><td>Unused.</td></tr>
<tr><td>23:16</td><td>RO</td><td>8 bits</td><td>Maximum Batch Descriptors in Progress
The maximum number of batch descriptors that each engine is capable of concurrently processing at any time. This field is undefined if the Descriptors in Progress Limit Supported field is GRPCAP is 0.</td></tr>
<tr><td>15:8</td><td>RO</td><td>8 bits</td><td>Maximum Work Descriptors in Progress
The maximum number of work descriptors that each engine is capable of concurrently processing at any time. This field is undefined if the Descriptors in Progress Limit Supported field is GRPCAP is 0.</td></tr>
<tr><td>7:0</td><td>RO</td><td>8 bits</td><td>Number of Engines</td></tr>
</table>

9.2.6 Operations Capabilities Register (OPCAP)

The Operations Capabilities register indicates which operation types are supported by the device. The register is a bitmask where each bit corresponds to the operation type with the same code as the bit position. For example, bit 0 of this register corresponds to the No-op operation (code 0). See section 8.1 for the values of the operation codes.

OPCAP Base: BAR0			Offset: 0x40 Size: 32 bytes (4 × 64 bits)
Bit	Attr	Size	Description
255:0	RO	256 bits	Each bit corresponds to an operation code and indicates whether that operation type is supported. If the bit is 1, the corresponding operation type is supported; if the bit is 0, the corresponding operation type is not supported. Bits corresponding to undefined operation codes are unused and are read as 0.

9.2.7 Table Offsets Register (OFFSETS)

Hardware implementations may place configuration tables in any otherwise unassigned address ranges within BAR0 MMIO space. This register indicates the offsets of these tables: Group Configuration, WQ Configuration, MSI-X Permissions, IMS, Performance Monitoring, and IDPT. Software must use the values in this register to determine the offsets of these tables, as the offsets may change between implementations.

OFFSETS Base: BAR0			Offset: 0x60 Size: 16 bytes (2 × 64 bits)
Bits	Attr	Size	Description
127:96	RO	32 bits	Unused.
95:80	RO	16 bits	Inter-Domain Permissions Table Offset Indicates the offset of the Inter-Domain Permissions Table. The offset is the value in this field times 0x100.
79:64	RO	16 bits	Perfmon Offset Indicates the offset of the Performance Monitoring Registers. The offset is the value in this field times 0x100.
63:48	RO	16 bits	IMS Offset Indicates the offset of the Interrupt Message Storage. The offset is the value in this field times 0x100. If the Interrupt Message Storage Support field in the Scalable IOV capability is 0, the value in this field is undefined.
47:32	RO	16 bits	MSI-X Permissions Offset Indicates the offset of the MSI-X Permissions Table. The offset is the value in this field times 0x100.
31:16	RO	16 bits	WQ Configuration Offset Indicates the offset of the WQ Configuration Table. The offset is the value in this field times 0x100.
15:0	RO	16 bits	Group Configuration Offset Indicates the offset of the Group Configuration Table. The offset is the value in this field times 0x100.

9.2.8 General Configuration Register (GENCFG)

This register is read-write while the device is Disabled and read-only otherwise. It is read-only at all times if the Configuration Support field in GENCAP is 0.

GENCFG Base: BAR0			Offset: 0x80 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:14	RSVD	18 bits	Reserved.
13	RWL	1 bit	Event Log Enable 0: Hardware writes any software error related events to SWERROR. 1: Any events including software error related ones are written to the event log in memory. (See section 5.9 for information on the event log.) This field is reserved when Event Log Support in GENCAP is 0.
12	RWL	1 bit	User-mode Interrupts Enable 0: User-mode descriptors are not allowed to request completion interrupts. 1: User-mode descriptors may request completion interrupts. An application is prevented from generating any interrupt that is not assigned to it by matching the PASID field of the interrupt table entry to the PASID of the descriptor. See section 5.4.
11:8	RSVD	4 bits	Reserved.
7:0	RWL	8 bits	Global Read Buffer Limit This field indicates the maximum number of Read Buffers that may be in use at one time by operations that access low bandwidth memory. This number of Read Buffers is shared by all descriptors accessing low bandwidth memory across the entire device. The default value is equal to the Total Read Buffers reported in GRPCAP. The value in this field is used when the Use Global Read Buffer Limit field in any of the Group Configuration registers is 1. See section 9.2.23. If used, this value must be at least 4 times the total number of engines in all groups that have the Use Global Read Buffer Limit set to 1. If the Global Read Buffer Limit Supported field in GRPCAP is 0, this field is reserved.

9.2.9 General Control Register (GENCTRL)

GENCTRL Base: BAR0			Offset: 0x88 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:3	RSVD	29 bits	Reserved.
2	RW	1 bit	Event Log Interrupt Enable 0: No interrupt is generated when an event is written to the event log. 1: The interrupt at index 0 in the MSI-X table is generated when an event is written to the event log. The Event Log field of the Interrupt Cause Register is set to 1.
1	RW	1 bit	Halt State Interrupt Enable 0: No interrupt is generated when device transitions to Halt state. 1: The interrupt at index 0 in the MSI-X table is generated when the device transitions to Halt state (see section 5.6). The Halt State field of the Interrupt Cause Register is set to 1.
0	RW	1 bit	Software Error Interrupt Enable 0: No interrupt is generated for software errors. 1: The interrupt at index 0 in the MSI-X table is generated when the Valid field in SWERROR changes from 0 to 1. The Software Error field of the Interrupt Cause Register is set to 1.

9.2.10 General Status Register (GENSTS)

GENSTS Base: BAR0			Offset: 0x90 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:4	RO	28 bits	Unused.
3:2	RO	2 bits	Reset Type Required 00: Software can issue a Reset Device command to the Command Register (see section 9.2.12) to recover the device. 01: Device requires a function-level reset (FLR) to recover from the current state. 10: Device requires a warm-reset to recover from the current state. 11: Device requires a cold-reset to recover. This is typically after a severe error that cannot be cleared with a function-reset (FLR) or warm reset. Note: This field indicates the minimum reset type needed to recover. Software can always choose to invoke a stronger type of reset to reinitialize the device. The mechanism used to trigger a reset may be platform-specific. It should be noted that when using Function Level Reset, software is expected to follow the app note in the PCIe specification, section 6.6.2.
1:0	RO	2 bits	Device State 00: Device is Disabled. No work is performed. All ENQ operations return Retry. 01: Device is Enabled. Work queues may be enabled, and descriptors may be submitted to enabled work queues. 10: Disable Device or Reset Device command is in progress. Descriptors are not accepted into any WQ. All Descriptors are being drained. 11: Halt State. The device is halted due to an error or unsupported condition that was encountered. Additional details related to this state and related software actions needed are described in section 5.6.

9.2.11 Interrupt Cause Register (INTCAUSE)

The Interrupt Cause Register is used to indicate the reason that an interrupt was generated using entry 0 in the MSI-X table. For interrupts generated using other MSI-X table entries or any of the IMS entries, no separate cause register exists. In the latter cases, software can identify the cause of the interrupt based on the interrupt vector or by reading the cause associated location, for example the completion record address or the WQ Occupancy register.

INTCAUSE Base: BAR0			Offset: 0x98 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31	RW1C	1 bit	Interrupt Handles Revoked
30:6	RSVD	25 bits	Reserved
5	RW1C	1 bit	Event Log
4	RW1C	1 bit	Halt State
3	RW1C	1 bit	Perfmon Counter Overflow
2	RW1C	1 bit	WQ Occupancy Below Limit
1	RW1C	1 bit	Command Completion
0	RW1C	1 bit	Software Error

9.2.12 Command Register (CMD)

The Command register is used to submit administrative commands. Before writing to this register, software must ensure that any command previously submitted via this register has completed by checking the Active field of the Command Status register. When a command is submitted, the Active field of the Command Status register is set to 1. The Active field changes to 0 when the command is complete. The other fields of the Command Status register indicate whether the command completed successfully. If the command register is written while Active is 1, the value written is discarded and an error is recorded in the SWERROR register. Reading the Command register returns unpredictable values.

When the command finishes, if the Request Completion Interrupt field of the Command register is 1, then the Command Completion field of the Interrupt Cause register is set to 1 and an interrupt is generated using entry 0 in the MSI-X table.

The Command Capabilities register (9.2.14) indicates which of the commands listed in Table 9-5 are supported by an implementation. If an undefined or unsupported command is written to the Command register, error code 0x01 is reported in the Command Status register.

See section 3.13.3 for details on the operation of commands submitted to the Command register.

CMD Base: BAR0			Offset: 0xA0 Size: 4 bytes (32 bits)
Bit	Attr	Size	Description
31	WO	1 bit	**Request Completion Interrupt** When this field is 1, upon completion of the command an interrupt is generated using entry 0 in the MSI-X table.
30:25	RSVD	6 bits	Reserved.
24:20	WO	5 bits	**Command Code** See Table 9-5 for command codes. Undefined command codes are reserved.
19:0	WO	20 bits	**Operand** The meaning of this field depends on the command. See Table 9-5.

Command	Code	Operand	Operation
Enable Device	1	Reserved	Enable the device.
Disable Device	2	Reserved	Disable the device.
Drain All	3	Reserved	Wait for all descriptors.
Abort All	4	Reserved	Abandon and/or wait for all descriptors.
Reset Device	5	Reserved	Disable the device and clear the device configuration.
Enable WQ	6	19:8 Reserved 7:0 Index of the WQ to enable	Enable the WQ.

Command	Code	Operand	Operation
Disable WQ	7	19:16 Group number[1] 15:0 Bitmap specifying which WQs in the group to operate on. See description below.	Disable the specified WQs.
Drain WQ	8		Wait for descriptors in the specified WQs.
Abort WQ	9		Abandon and/or wait for descriptors in the specified WQs.
Reset WQ	10		Disable the specified WQs and clear the WQ configurations.
Drain PASID	11	The PASID to drain.	Wait for descriptors using the specified PASID.
Abort PASID	12	The PASID to abort.	Abandon and/or wait for descriptors using the specified PASID.
Request Interrupt Handle	13	19:17 Reserved 16 0 = MSI-X table 1 = IMS 15:0 Table index	Return a handle for the specified interrupt table entry. If this command is supported, it must be used to obtain interrupt handles. See section 3.7 for more information.
Release Interrupt Handle	14	19:17 Reserved 16 0 = MSI-X table 1 = IMS 15:0 Table index	Release the handle that was returned by Request Interrupt Handle for the specified interrupt table entry.
Request IDPT Handle	15	CMD 19:16 Reserved 15:0 IDPT Index	Request a handle for the specified index in the Inter-Domain Permissions Table. If the Request IDPT Handle capability in CMDCAP is 1, software is required to use this command to obtain an IDPT handle. The returned handle may be used in an inter-domain or Update Window descriptor. If the Request IDPT Handle capability in CMDCAP is 0, this command code is reserved. The IDPT index may be used as the handle in an inter-domain or Update Window descriptor.
Release IDPT Handle	16	CMD 19:16 Reserved 15:0 IDPT Index	Release the IDPT handle that was returned by Request IDPT Handle for the specified IDPT entry. If the Release IDPT Handle capability in CMDCAP is 1, software may use this command to release an IDPT entry that is no longer in use. If the Release IDPT Handle capability in CMDCAP is 0, this command code is reserved.

[1] The term “group” in this section is not to be confused with the engine groups described in section 3.4. For issuing WQ commands, a group simply consists of the WQs for which bits 7:4 of the WQ number are the same.

Command	Code	Operand	Operation
Invalidate Submitter Bitmap Cache	17	CMD 19:17 Reserved 16:0 Size CMDPARAM 511:64 Unused 63:0 Address	Invalidate the specified address range if cached in the submitter bitmap cache. The start of the address range must be written to the CMDPARAM register prior to issuing this command. The Size field allows software to limit the address range to invalidate to only the portion of the bitmap that was modified. If the Invalidate Submitter Bitmap Cache field in CMDCAP is 1, hardware may cache any portion of a bitmap, and software is required to issue this command after any change to a bitmap region including page mapping changes, and after performing necessary invalidations for any pages that are part of a bitmap. If the Invalidate Submitter Bitmap Cache field in CMDCAP is 0, this command code is reserved, and software is not required to issue this command after any change to a bitmap. See section 3.14.2 for details related to bitmap caching.

Table 9-5: Administrative Commands

The Disable WQ, Drain WQ, Abort WQ, and Reset WQ commands can be applied to groups[1] of up to 16 WQs at the same time. The group number is specified in bits 19:16 of the operand field, corresponding to bits 7:4 of the WQ index. Bits 15:0 of the operand field contain a bitmask indicating which WQs in the group to operate on. In an implementation with no more than 16 WQs, the group number is always 0. For example, to drain WQs 1, 4, and 7, the Operand field would be set to 0x00092. To drain WQs 21 and 22, the Operand field would be set to 0x10060. It is not possible use a single command to disable or drain WQs in different groups.

9.2.13 Command Status Register (CMDSTATUS)

The Command Status register indicates the status of the last command submitted to the Command register. The Active field indicates that a command is in progress. The Active field is set to 1 when a command is written to the Command register. While the Active field is 1, the values of the other fields are unspecified. When the command completes, the Active field is set to 0 and the other fields of this register indicate whether the command completed successfully.

CMDSTATUS Base: BAR0			Offset: 0xA8 Size: 4 bytes (32 bits)
Bit	Attr	Size	Description
31	RO	1 bit	Active 0: Command is complete (or no command has been submitted). 1: Command is in progress.
30:24	RSVD	7 bits	Unused.
23:8	RO	16 bits	Command Result For the Request Interrupt Handle command, if the Error Code field is 0, this field contains the interrupt handle corresponding to the interrupt table entry specified in the command operand. If Error Code is non-zero, this field is unused. For the Request IDPT Handle command, if the Error Code field is 0, this field contains the IDPT handle corresponding to the IDPT entry specified in the command operand. If Error Code is non-zero, this field is unused. For any other command, this field is unused.
7:0	RO	8 bits	Error Code 0x00: Successful completion. 0x01: Undefined or unsupported command code. 0x02: Invalid WQ index. 0x03: Error Condition caused by a platform or internal hardware error. Software can read GENSTS register and PCIe AER logs for details and to determine further action. 0x04: Non-zero reserved field in command. 0x05: Command submitted while device is in halt or error state. 0x06-0x0f: Unused. 0x10-0xff: Command-specific error codes. See Table 5-8.

9.2.14 Command Capabilities Register (CMDCAP)

The Command Capabilities register indicates which administrative commands are supported by the Command register. This register is a bitmask where each bit corresponds to the command with the same command code as the bit position. For example, bit 1 of this register corresponds to the Enable Device command (command code 1). See Table 9-5 for the values of the command codes.

This register is present only if the Command Capabilities Support field in GENCAP is 1.

If this register indicates support for the Request Interrupt Handle command, then the command must be used to obtain interrupt handles to use for descriptor completions.

CMDCAP Base: BAR0		Offset: 0xB0	Size: 4 bytes (32 bits)
Bit	Attr	Size	Description
31:0	RO	32 bits	Each bit corresponds to a command code, and indicates whether that administrative command is supported. If the bit is 1, the corresponding command is supported; if the bit is 0, the corresponding command is not supported. Bits corresponding to undefined command codes are unused and are read as 0.

If Command Capabilities Support is 0, this register is not present and the following commands are supported:

Command	Code	Operation
Enable Device	1	Enable the device.
Disable Device	2	Disable the device.
Drain All	3	Wait for all descriptors.
Abort All	4	Abandon and/or wait for all descriptors.
Reset Device	5	Disable the device and clear the device configuration.
Enable WQ	6	Enable the WQ.
Disable WQ	7	Disable the specified WQs.
Drain WQ	8	Wait for descriptors in the specified WQs.
Abort WQ	9	Abandon and/or wait for descriptors in the specified WQs.
Reset WQ	10	Disable the specified WQs and clear the WQ configurations.
Drain PASID	11	Wait for descriptors using the specified PASID.
Abort PASID	12	Abandon and/or wait for descriptors using the specified PASID.

Table 9-6: Default Commands Supported

9.2.15 Software Error Register (SWERROR)

Several types of errors can be recorded in this register:

- An error in submitting a descriptor.
- An error translating a Completion Record Address in a descriptor.
- An error validating a descriptor, if the Completion Record Address Valid flag in the descriptor is 0.
- An error while processing a descriptor, such as a page fault, if the Completion Record Address Valid flag in the descriptor is 0.
- An unsupported change to device configuration while the device is not Disabled.

Details on the error checking that can result in these errors are covered in chapter 5.

Only one error at a time can be recorded in this register. When an error is recorded, Valid is set to 1. If Valid is 1 at the time an error occurs, Overflow is set to 1 and the error is not recorded. The Valid and Overflow fields are cleared by software writing 1. They are not cleared by hardware, other than by reset. When supported, the event log may be used for reporting multiple errors without overflow (as described in Section 5.9).

When Valid changes from 0 to 1, if the Software Error Interrupt Enable field in GENCTRL is 1, the Software Error field of the Interrupt Cause register is set to 1 and an interrupt is generated.

SWERROR
Base: BAR0 Offset: 0xC0 Size: 32 bytes (4 × 64 bits)

Byte offset	Bits	Attr	Size	Description
7:0	63:60	RO	4 bits	Unused.
	59:40	RO	20 bits	PASID The PASID field of the descriptor that caused the error.
	39:32	RO	8 bits	Operation The Operation field of the descriptor that caused the error.
	31:24	RO	8 bits	Unused.
	23:16	RO	8 bits	WQ Index Indicates which WQ the descriptor was submitted to.
	15:8	RO	8 bits	Error code See section 5.8 for the meaning of the value in this field.
	7	RO	1 bit	Error Information Valid 0: Error Information field is valid only if the Error code is Invalid Flags (0x11). 1: Error Information field is valid and provides additional information pertaining to the error reported in the Error code field.
	6	RO	1 bit	Priv The Priv field of the descriptor that caused the error.
	5	RO	1 bit	R/W If the error is a page fault, this indicates whether the faulting access was a read or a write. 0: The faulting access was a read. 1: The faulting access was a write. Page faults are indicated by error codes 0x03, 0x04, 0x06, 0x1a, and 0x1f. For other error code values, this field is unused.

<table>
<tr><td colspan="5">SWERROR
Base: BAR0 Offset: 0xC0 Size: 32 bytes (4 × 64 bits)</td></tr>
<tr><th>Byte offset</th><th>Bits</th><th>Attr</th><th>Size</th><th>Description</th></tr>
<tr><td></td><td>4</td><td>RO</td><td>1 bit</td><td>Batch Member
0: The descriptor was submitted directly.
1: The descriptor was submitted in a batch.</td></tr>
<tr><td></td><td>3</td><td>RO</td><td>1 bit</td><td>WQ Index Valid
0: The WQ that the descriptor was submitted to is unknown. The WQ Index field is unused.
1: The WQ Index field indicates which WQ the descriptor was submitted to.</td></tr>
<tr><td></td><td>2</td><td>RO</td><td>1 bit</td><td>Descriptor Valid
0: The descriptor that caused the error is unknown. The Batch Member, Operation, Batch Index, Priv, and PASID fields are unused.
1: The Batch Member, Operation, Batch Index, Priv, and PASID fields are valid.</td></tr>
<tr><td></td><td>1</td><td>RW1C</td><td>1 bit</td><td>Overflow
0: The last error recorded in this register is the most recent error.
1: One or more additional errors occurred after the last one recorded in this register.
This field is not cleared by hardware, except by reset. It is cleared by software writing 1.</td></tr>
<tr><td></td><td>0</td><td>RW1C</td><td>1 bit</td><td>Valid
0: No error is recorded. All of the other fields of the SWERROR register except Overflow are undefined.
1: An error has occurred and is recorded in this register.
This field is not cleared by hardware, except by reset. It is cleared by software writing 1.</td></tr>
<tr><td>15:8</td><td>63:32</td><td>RO</td><td>32 bits</td><td>Error Information
This field reports additional information for the error codes listed below. Otherwise, this field is unused.
<table>
<tr><th>Error code</th><th>Error information</th></tr>
<tr><td>Invalid Flags (0x11)</td><td>63:32 – A bitmask of the flags that were found to be invalid. If a bit in this field is 1, it indicates that the flag at the corresponding bit position in the Flags field of the descriptor was invalid.</td></tr>
<tr><td>Invalid Handle (0x19)</td><td>63:48 – Unused.
47:32 – Interrupt handle.</td></tr>
</table></td></tr>
</table>

<table>
<tr><td colspan="5">SWERROR
Base: BAR0 Offset: 0xC0 Size: 32 bytes (4 × 64 bits)</td></tr>
<tr><th>Byte offset</th><th>Bits</th><th>Attr</th><th>Size</th><th>Description</th></tr>
<tr><td></td><td></td><td></td><td></td><td>
<table>
<tr><td>Page Fault (0x03, 0x04)</td><td>63:61 – Operand Identifier. See Table 8-5 for a description of this field.
For Inter-domain operations:
60:48 – Unused.
47:32 – The IDPT handle used with the faulting address.
For other operation types, bits 60:32 are unused.</td></tr>
<tr><td>Page Fault (0x06, 0x1f)</td><td>63:61 – Operand Identifier. See Table 8-5 for a description of this field.
60:32 – Unused.</td></tr>
<tr><td>Inter-Domain Operation Error (0x29-0x2c)</td><td>63:48 – Unused.
47:32 – The IDPT handle that caused the error.</td></tr>
</table>
</td></tr>
<tr><td></td><td>31:16</td><td>RO</td><td>16 bits</td><td>Unused.</td></tr>
<tr><td></td><td>15:0</td><td>RO</td><td>16 bits</td><td><b>Batch Index</b>
If the Descriptor Valid field is 1 and the Batch Member field is 1, this field contains the index of the descriptor within the batch. Otherwise, this field is unused.</td></tr>
<tr><td>23:16</td><td>63:0</td><td>RO</td><td>64 bits</td><td><b>Address</b>
If the error is a page fault, this is the faulting address. Bits 11:0 may be reported as 0.
Otherwise, this field is undefined.</td></tr>
<tr><td>31:24</td><td>63:0</td><td>RO</td><td>64 bits</td><td>Unused.</td></tr>
</table>

9.2.16 Event Log Configuration Register (EVLCFG)

The Event Log Configuration register is used to configure and manage the region of memory used by the hardware to report several types of error events. Section 5.9 describes the functional details of the event log. The size of event log entries is indicated by the Event Log Support field in GENCAP.

This register is read-write while the device is Disabled and read-only otherwise.

EVLCFG Base: BAR0			Offset: 0xE0 Size: 16 bytes (2*64 bits)
Bits	Attr	Size	Description
127:100	RSVD	28 bits	Reserved.
99:80	RWL	20 bits	PASID If PASID Enable is 1, this field specifies the PASID value used for writes to the event log.
79:64	RWL	16 bits	Event Log Size Indicates the number of entries in the event log. The maximum number of entries that may be written to the event log without overflow is this value minus 1. If Event Log Enable is 1, the following constraints apply: - Event Log Size ≥ 64. - Event Log Base Address + Event Log Size × Event Log entry size ≤ 2^{64}.
63:12	RWL	52 bits	Event Log Base Address Base address of the event log. The address must be 4KB aligned.
11:2	RSVD	10 bits	Reserved.
1	RWL	1 bit	Priv The Priv flag used for writes to the event log. 0: The Priv flag is 0 on event log writes. 1: The Priv flag is 1 on event log writes. This field is reserved if PASID Enable is 0 or if the Privileged Mode Enable field of the PCI Express PASID capability is 0.
0	RWL	1 bit	PASID Enable Indicates whether PASID is used for writes to the event log. 0: PASID is not used for event log writes. 1: PASID is used for event log writes. The PASID value is specified by the PASID field in this register. This field is reserved when the PASID Enable field of the PCI Express PASID capability is 0.

9.2.17 Event Log Status Register (EVLSTATUS)

If Event Log Enabled in GENCFG is 1, this register is cleared upon successful completion of the Enable Device command.

EVLSTATUS Base: BAR0		Offset: 0xF0	Size: 8 bytes (64 bits)
Bits	Attr	Size	Description
63	RSVD	1 bit	Reserved.
62	RW1C	1 bit	Interrupt Pending Indicates that an event log interrupt has been generated. Hardware sets this bit when generating an event log interrupt. No further interrupts are generated for additional log entries while this bit is set. Software clears it after processing the interrupt by writing a 1 to this bit.
61:48	RSVD	14 bits	Reserved.
47:32	RO	16 bits	Event Log Tail Index of the event log entry to be written next by hardware. Hardware updates this register after each event written to the event log.
31:16	RSVD	16 bits	Reserved.
15:0	RW	16 bits	Event Log Head Index of the event log entry to be processed next by software. Software updates this register after processing one or more event log entries.

9.2.18 Inter-Domain Capabilities Register (IDCAP)

The IDCAP register describes capabilities related to inter-domain operations support in the device. This register is present if Inter-Domain support in GENCAP is 1.

IDCAP Base: BAR0			Offset: 0x100 Size: 8 bytes (64 bits)
Bits	Attr	Size	Description
63:28	RO	36 bits	Reserved.
27:26	RO	2 bits	IDPTE Size Indicates the size of an IDPTE. The size of each IDPT entry is 2^{N+5} bytes, where N is the value in this field.
25	RO	1 bit	Update Window Suppress Drain Support 0: Suppress Drain flag is not supported in the Update Window descriptor. 1: Software may set the Suppress Drain flag to avoid an implicit drain after an Update Window operation.
24	RO	1 bit	Offset Mode Support 0: Window Mode field in an IDPTE does not allow selection of Offset Mode. 1: Window Mode field in an IDPTE allows selection of Offset Mode.
23:8	RO	16 bits	Inter-Domain Permissions Table Size The number of entries in the Inter-Domain Permissions Table. If this field is 0, there is no Inter-Domain Permissions Table on the device.
7:2	RO	6 bits	Reserved.
1:0	RO	2 bits	Inter-Domain Permissions Table Entry Type Support Bitmask to specify the types of Inter-Domain Permissions Table entries supported by the device. If a bit is 1, then that type is supported.

9.2.19 Inter-Domain Bitmap Register (IDBR)

If the Inter-Domain Support field in GENCAP is 1, and bit 1 is set in the Type Support field in IDCAP, the Inter-Domain Bitmap register is used to specify the PASID and Privilege to be used to read bitmaps referenced by the IDPT. Otherwise, this register is reserved. This register is read-write while the device is disabled and read-only otherwise.

IDBR Base: BAR0			Offset: 0x108 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:28	RSVD	4 bits	Reserved.
27:8	RWL	20 bits	Bitmap PASID PASID value used to read the bitmap from memory.
7:5	RSVD	3 bits	Reserved.
4:2	RWL	3 bits	Bitmap TC Specifies the traffic class to use to read the bitmap from memory.
1	RWL	1 bit	Priv The privilege field used to read the bitmap from memory. This field is reserved if PASID Enable is 0 or if the Privileged Mode Enable field of the PCI Express PASID capability is 0.
0	RWL	1 bit	PASID Enable 0: Reads of the bitmap will not be tagged with PASID. 1: Reads of the bitmap will use the PASID specified in this register. If the PCI Express PASID Capability Enable is 0, this field is reserved.

9.2.20 Command Parameter Register (CMDPARAM)

The Command Parameter register is used to specify additional parameters for administrative commands. The use of this register depends on the specific command being issued. See section 9.2.12 for details of commands that use the CMDPARAM register. Hardware may only implement the bits in the register that are used by commands supported in CMDCAP. Software must not rely on the value of unimplemented bits. Read of unimplemented bits may return 0 and writes to unimplemented bits may be dropped.

<table>
<tr><td colspan="4">CMDPARAM
Base: BAR0   Offset: 0xE000   Size: 8 bytes (64 bits)</td></tr>
<tr><th>Bits</th><th>Attr</th><th>Size</th><th>Description</th></tr>
<tr><td>63:0</td><td>RW</td><td>64 bits</td><td>Command Parameters</td></tr>
</table>

9.2.21 Dummy Portal (DUMMY)

The Dummy Portal behaves like a portal for a WQ that is not enabled. For all addresses on the page, writes are ignored (returning Retry for DMWr) and reads return either 00 or FF for all bytes. See section 9.3 for more information about portals. See section 7.3 for how this register may be used for virtualization.

<table>
<tr><td colspan="2">DUMMY
Base: BAR0 Offset: 0xF000 Size: 0x1000 bytes</td></tr>
<tr><th>Size</th><th>Description</th></tr>
<tr><td>0x1000 bytes</td><td>Dummy Portal
Writes are ignored (returning Retry for DMWr) and reads return either 00 or FF for all bytes.</td></tr>
</table>

9.2.22 MSI-X Permissions Table (MSIXPERM)

The MSI-X Permissions Table is a set of 4-byte registers in BAR0 with the same number of entries as the MSI-X Table. The offset of the MSI-X Permissions Table is given by the MSI-X Permissions Offset field in the Table Offsets register. The number of entries is given by the PCIe-defined MSI-X capability. The individual registers in the table are on 8-byte boundaries.

Each register in the MSI-X Permissions Table corresponds to an entry in the MSI-X table and contains controls associated with that interrupt table entry. These controls are the same as those in the IMS, but these fields cannot be added to the MSI-X table itself, because it is defined by PCI-SIG.

MSIXPERM Base: BAR0			Offset: Table-offset + index × 8 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:12	RW	20 bits	PASID If PASID Enable is 1, this field is checked against the PASID field of the descriptor. See section 5.4.
11:4	RSVD	8 bits	Reserved.
3	RW	1 bit	PASID Enable This field is checked against the WQ PASID Enable field of the WQ the descriptor was submitted to. See section 5.4.
2	RW	1 bit	Ignore If this field is 1 when a descriptor completion interrupt references the corresponding MSI-X table entry, no interrupt is generated and the Pending field is not set. This field does not prevent delivery of an interrupt if Pending is 1 and Mask is cleared. This field does not affect delivery of interrupts due to causes other than descriptor completion.
1:0	RSVD	2 bits	Reserved.

9.2.23 Group Configuration Table (GRPCFG)

The Group Configuration Table is an array of registers in BAR0 that controls the mapping of work queues to engines. The offset of the Group Configuration Table is given by the Group Configuration Offset field in the Table Offsets register. The number of groups is given by the Number of Groups field in GRPCAP. Software may configure the number of groups that it needs. Group Configuration registers beyond the number of groups available are reserved and may not be implemented in hardware.

Each active group contains one or more work queues and one or more engines. Any unused group must have both the WQs field and the Engines field equal to 0. Descriptors submitted to any WQ in a group may be processed by any engine in the group. Each active work queue must be in a single group. (An active work queue is one for which the WQ Size field of the corresponding WQCFG register is non-zero.) Any engine that is not in a group is inactive. See section 3.4 for more information on engines and groups.

Each GRPCFG register is divided into three sub-registers.

These registers are read-write while the device is Disabled and read-only otherwise. They are read-only at all times if the Configuration Support field in GENCAP is 0.

GRPWQCFG
Base: BAR0 Offset: Table-offset + Group-ID × 64 + 0 Size: 256 bits (4 × 64 bits)

Bits	Attr	Size	Description
255:0	RWL	256 bits	WQs Each bit corresponds to a WQ and indicates that the corresponding WQ is in the group. Bits beyond the number of WQs available are reserved and may not be implemented in hardware. Each active WQ must be in exactly one group. Inactive WQs (those for which WQ Size is 0 in WQCFG) must not be in any group.

GRPENGCFG
Base: BAR0 Offset: Table-offset + Group-ID × 64 + 32 Size: 8 bytes (64 bits)

Bits	Attr	Size	Description
63:0	RWL	64 bits	Engines Each bit corresponds to an engine and indicates that the corresponding engine is in the group. Bits beyond the number of engines available are reserved and may not be implemented in hardware.

GRPFLAGS Base: BAR0			Offset: Table-offset + Group-ID × 64 + 40 Size: 8 bytes (64 bits)
Bits	Attr	Size	Description
63:38	RSVD	26 bits	Reserved.
37:36	RWL	2 bits	**Batch Descriptors in Progress Limit** This field controls the number of batch descriptors that can be concurrently processed by an engine in the group as a fraction of the Maximum Batch Descriptors in Progress value specified in ENGCAP. Note that exercising this control can cause bandwidth to be reduced in some cases, depending on prevailing system latency. This field is reserved if the Descriptors in Progress Limit Supported field is GRPCAP is 0. **00:** Allow up to the maximum value that an engine is capable of. **01:** Limit to $^{1}/_{2}$ of the maximum value. **10:** Limit to $^{1}/_{4}$ of the maximum value. **11:** Limit to $^{1}/_{8}$ of the maximum value.
35:34	RSVD	2 bits	Reserved.
33:32	RWL	2 bits	**Work Descriptors in Progress Limit** This field controls the number of work descriptors that can be concurrently processed by an engine in the group as a fraction of the Maximum Work Descriptors in Progress value specified in ENGCAP. Note that exercising this control can cause bandwidth to be reduced in some cases, depending on prevailing system latency. This field is reserved if the Descriptors in Progress Limit Supported field is GRPCAP is 0. **00:** Allow up to the maximum value that an engine is capable of. **01:** Limit to $^{1}/_{2}$ of the maximum value. **10:** Limit to $^{1}/_{4}$ of the maximum value. **11:** Limit to $^{1}/_{8}$ of the maximum value.
31:28	RSVD	4 bits	Reserved.
27:20	RWL	8 bits	**Read Buffers Allowed** This field indicates the maximum number of Read Buffers that may be in use at one time by all engines in the group. This value can be used to limit the maximum bandwidth used by engines in the group. This value must be: - greater than or equal to 4 times the number of engines in the group; - greater than or equal to the Read Buffers Reserved field for this group; and - less than or equal to the sum of the Read Buffers Reserved field and the number of non-reserved Read Buffers. (The number of non-reserved Read Buffers is the Total Read Buffers field in GRPCAP minus the total of the Read Buffers Reserved fields for all groups.) The default value of this field is the same as the value of the Total Read Buffers field in GRPCAP. If the Read Buffer Controls Supported field in GRPCAP is 0, this field is read-only and is unused.
19:16	RSVD	4 bits	Reserved.

15:8	RWL	8 bits	Read Buffers Reserved This field indicates the number of Read Buffers reserved for the use of engines in the group. This value can be used to reduce the possibility of contention with engines in other groups. However, if it is set to a non-zero value, it may reduce the overall performance of the device. The sum of the Read Buffers reserved for all groups must be less than or equal to the Total Read Buffers field in GRPCAP. If the Read Buffer Controls Supported field in GRPCAP is 0, this field is read-only and is unused.
7	RWL	1 bit	Use Global Read Buffer Limit 0: The Global Read Buffer Limit does not apply to this group. 1: The Global Read Buffer Limit programmed in the GENCFG register applies to descriptors processed by engines in this group. (The limit indicated by the Read Buffers Allowed field applies as well.) If the Global Read Buffer Limit Supported field in GRPCAP is 0, this field is reserved.
6	RSVD	1 bit	Reserved.
5:3	RWL	3 bits	TC-B Specifies the traffic class to use for memory accesses for which the traffic class selector in the descriptor is 1.
2:0	RWL	3 bits	TC-A Specifies the traffic class to use for memory accesses for which the traffic class selector in the descriptor is 0.

9.2.24 WQ Configuration Table (WQCFG)

The WQ Configuration Table is an array of registers in BAR0. The offset of the WQ Configuration Table is given by the WQ Configuration Offset field in the Table Offsets register. The number of WQs is given by the Number of WQs field in WQCAP. The size of the WQCFG register for each WQ is given by the WQCFG Size field in WQCAP. The size is 2^{N+5} bytes, where N is the value of the WQCFG Size field.

Each WQCFG register is divided into sub-registers, which may be read or written using aligned 1-, 2-, 4-, or 8-byte read or write operations. The fields of WQCFG are read-only or read-write at different times, depending on device state, WQ state, the Configuration Support field in GENCAP, and the WQ Mode Support field, as detailed in the table. Any writes to fields while they are read-only are ignored.

Field	Configuration Support		
	1	0	
		Mode Support=0	Mode Support=1
Mode Support	Read-only at all times		
Size	Read-write while device is Disabled; read-only otherwise	Read-only at all times	Read-only at all times
Threshold	Read-write at all times	Read-only at all times	Read-write at all times
Mode Priv PASID Enable PASID	Read-write while WQ is Disabled; read-only otherwise	Read-only at all times	Read-write while WQ is Disabled; read-only otherwise
Priority Block-on-Fault Enable Maximum Transfer Size Maximum Batch Size ATS Disable PRS Disable	Read-write while WQ is Disabled; read-only otherwise	Read-only at all times	Read-only at all times
Occupancy Interrupt Enable	Read-write at all times	Read-only at all times	Read-write at all times
Occupancy Limit	Read-only while Occupancy Interrupt Enable is 1	Read-only at all times	Read-only while Occupancy Interrupt Enable is 1
Occupancy Interrupt Table Occupancy Interrupt Handle	Read-write while WQ is Disabled; read-only otherwise	Read-only at all times	Read-write while WQ is Disabled; read-only otherwise
Operations Configuration	Read-write while WQ is Disabled; read-only otherwise	Read-only at all times	Read-only at all times

Table 9-7: Work Queue Configuration Support

The WQ Size fields of all the WQCFG registers must be set before the device is enabled. The sum of all the WQ Size fields must not be greater than Total WQ Size field in WQCAP. WQs for which the WQ Size field is 0 are inactive and cannot be enabled. The other configuration fields for inactive WQs are ignored.

At the time a WQ is enabled, consistency checks are performed on the fields of the WQCFG register. See section 5.2 for the checks that are performed.

WQCFG
Base: BAR0 Offset: Table-offset + WQ-ID × WQCFG-Size Size: WQCFG-Size bytes

Bytes	Bits	Attr	Size	Description
3:0	31:16	RSVD	16 bits	Reserved.
	15:0	[1]	16 bits	**WQ Size** The number of entries in the WQ storage allocated to this WQ. The sum of the WQ Size fields for all work queues must be less than or equal to the Total WQ Size field in WQCAP.
7:4	31:16	RSVD	16 bits	Reserved.
	15:0	[1]	16 bits	**WQ Threshold** The number of entries in this WQ that may be filled via a limited portal. If WQ Occupancy is greater than or equal to WQ Threshold, work submissions using a limited portal return Retry. The threshold applies only to shared work queues. If WQ Mode is 1 (dedicated mode), this field is ignored. If WQ Threshold is greater than WQ Size, it is treated as if it is equal to WQ Size.
11:8	31:30	RSVD	2 bits	Reserved.
	29	[1]	1 bit	**WQ Priv** The Priv flag to be used for descriptors submitted to this WQ when it is in dedicated mode. If the WQ is in dedicated mode, WQ PASID Enable is 1, and the Privileged Mode Enable field of the PCI Express PASID capability is 0, this field must be 0. If the WQ is in shared mode or WQ PASID Enable is 0, this field is ignored.
	28	[1]	1 bit	**WQ PASID Enable** Indicates whether PASID is used for address translation requests for descriptors from this WQ. If the PCI Express PASID capability is not enabled, this field must be 0. If WQ Mode is 0 (SWQ), this field must be 1.
	27:8	[1]	20 bits	**WQ PASID** The PASID to be used for descriptors submitted to this WQ when it is in dedicated mode. If the WQ is in shared mode or WQ PASID Enable is 0, this field is ignored.
	7:4	[1]	4 bits	**WQ Priority** If the WQ Priority Support field in WQCAP is 1, this field indicates the priority of this work queue relative to other WQs in the same group. This field must not be 0. See section 4.1 for a description of WQ priorities. If the WQ Priority Support field in WQCAP is 0, this field is ignored.

[1] Table 9-7 in this section describes when this field is RW and when it is RO.

WQCFG
Base: BAR0 Offset: Table-offset + WQ-ID × WQCFG-Size Size: WQCFG-Size bytes

Bytes	Bits	Attr	Size	Description
	3	[1]	1 bit	**WQ PRS Disable** **0:** PRS is used for descriptors submitted to this WQ according to the setting of the Enable field in the PCIe PRS capability, the WQ Block on Fault Enable field, and the Block on Fault flag in the descriptor. **1:** PRS is not used for descriptors submitted to this WQ even when the Enable field in the PCIe PRS capability is 1. The WQ Block on Fault Enable field must be 0. If WQ PRS Support is 0, this field is reserved and may be hardwired to 0. If Event Log Enable in GENCFG is 0, this field must be 0.
	2	[1]	1 bit	**WQ ATS Disable** **0:** ATS is used for descriptors submitted to this WQ according to the setting of the Enable field in the PCIe ATS capability. **1:** ATS is not used for descriptors submitted to this WQ even when the Enable field in the PCIe ATS capability is 1. If WQ ATS Support is 0, this field is reserved and may be hardwired to 0.
	1	[1]	1 bit	**WQ Block on Fault Enable** **0:** Block on fault is not allowed. The Block On Fault flag in descriptors submitted to this WQ is reserved. If a page fault occurs on a source or destination memory access, the operation stops and the page fault is reported to software. **1:** Block on fault is allowed. Behavior on page faults depends on the values of the Block on Fault flag in each descriptor. This field is reserved if the Block on Fault Support field in GENCAP is 0 or if the Enable field of the PCIe Page Request Control Register is 0.
	0	[1]	1 bit	**WQ Mode** **0:** WQ is in shared mode. **1:** WQ is in dedicated mode.
15:12	31:9	RSVD	23 bits	Reserved.
	8:5	[1]	4 bits	**WQ Maximum Batch Size** The maximum number of descriptors that can be referenced by a Batch descriptor submitted to this WQ is 2^N, where N is the value in this field. If Batch descriptor is supported (as indicated by bit 1 of OPCAP), this field must not be 0 and must not be greater than the Maximum Supported Batch Size field in GENCAP. Otherwise, this field is reserved. It is checked when the WQ is enabled. Software should set this field to the minimum size needed, to limit how long descriptors can block other descriptors behind them.

[1] Table 9-7 in this section describes when this field is RW and when it is RO.

WQCFG
Base: BAR0 Offset: Table-offset + WQ-ID × WQCFG-Size Size: WQCFG-Size bytes

Bytes	Bits	Attr	Size	Description
	4:0	[1]	5 bits	**WQ Maximum Transfer Size** The maximum transfer size that can be specified in a descriptor submitted to this WQ is 2^N, where N is the value in this field. This field must not be greater than the Maximum Supported Transfer Size field in GENCAP. It is checked when the WQ is enabled. Software should set this field to the minimum size needed, to limit how long descriptors can block other descriptors behind them.
19:16	31:17	RSVD	15 bits	Reserved.
	16	RWL	1 bit	**WQ Occupancy Interrupt Table** **0:** WQ Occupancy Interrupt Handle is a handle for the MSI-X table. **1:** WQ Occupancy Interrupt Handle is a handle for the IMS. This field is read-only except while the WQ is Disabled. If the IMS Support field in the SIOV capability is 0 or if the SIOV capability is not present, this field must be 0. If the WQ Occupancy Interrupt Support field in WQCAP is 0, this field is reserved.
	15:0	RWL	16 bits	**WQ Occupancy Interrupt Handle** An interrupt handle indicating which interrupt table entry to use to generate the interrupt. When the Interrupt Handle Request capability is 0, this field is the index of the desired entry in the MSI-X table or IMS. When the Interrupt Handle Request capability is 1, this is a handle returned by the Request Interrupt Handle command. This field is read-only except while the WQ is Disabled. If the WQ Occupancy Interrupt Support field in WQCAP is 0, this field is reserved.
23:20	31:17	RSVD	15 bits	Reserved.
	16	RW	1 bit	**WQ Occupancy Interrupt Enable** Setting this field to 1 causes the device to generate an interrupt when the WQ occupancy is at or less than the WQ Occupancy Limit. The device sets the Interrupt Generated field when the interrupt is generated. This field may be set to 1 at the same time the WQ Occupancy Limit field is set to the desired value. If this field is set to 1 with Limit ≥ the current WQ occupancy, the interrupt is generated immediately. If the WQ Occupancy Interrupt Support field in WQCAP is 0, this field is reserved. If this field is set to 1 while the WQ is Disabled, the interrupt will be delivered at the time the WQ is enabled.

[1] Table 9-7 in this section describes when this field is RW and when it is RO.

WQCFG Base: BAR0		Offset: Table-offset + WQ-ID × WQCFG-Size		Size: WQCFG-Size bytes
Bytes	Bits	Attr	Size	Description
	15:0	RWL	16 bits	WQ Occupancy Limit When the WQ Occupancy Interrupt Enable is 1 and the WQ occupancy is at or below the value in this field, an interrupt is generated. This field is read-only while WQ Occupancy Interrupt Enable is 1; however, it may be changed at the same time that WQ Occupancy Interrupt Enable is set to 1.[1] If the WQ Occupancy Interrupt Support field in WQCAP is 0, this field is reserved.
27:24	31:30	RO	2 bits	WQ State 00: WQ is Disabled. Descriptors are not accepted into the WQ. (ENQ operations to this WQ return Retry. Other write operations are ignored.) 01: WQ is Enabled. Descriptors may be submitted and processed. 10: Disable WQ, Reset WQ, Disable Device, or Reset Device command is in progress. Descriptors are not accepted into the WQ. Descriptors currently in the WQ are being drained. 11: Unused.
	29	RO	1 bit	WQ Mode Support When the Configuration Support field in GENCAP is 0, this field indicates whether certain WQ configuration fields are read-only. See the table in this section for the meaning of this field. When the Configuration Support field in GENCAP is 1, this field is unused.
	28:17	RSVD	12 bits	Reserved.
	16	RW1C	1 bit	WQ Occupancy Interrupt Generated 0: There are no WQ Occupancy Interrupts for this WQ that have not been acknowledged by software. Device is able to generate a WQ Occupancy Interrupt if the conditions are satisfied. 1: WQ Occupancy Interrupt was generated. Software should write a 1 to clear this bit. This bit must be cleared before another WQ Occupancy Interrupt can be generated for this WQ.
	15:0	RO	16 bits	WQ Occupancy The number of entries currently in this WQ. This number may change whenever descriptors are submitted to or dispatched from the queue, so it cannot be relied on to determine whether there is space in the WQ. If the WQ Occupancy Support field in WQCAP is 0, the value in this field is undefined.

[1] To change Limit when Enable is 1, software must first write 0 to Enable. It may then write a new value to Limit and set Enable back to 1 at the same time.

WQCFG
Base: BAR0 Offset: Table-offset + WQ-ID × WQCFG-Size Size: WQCFG-Size bytes

Bytes	Bits	Attr	Size	Description
31:28	31:0	RSVD	32 bits	Reserved.
63:32	255:0	[1]	256 bits	**WQ Operations Configuration** Each bit corresponds to an operation code and indicates whether that operation type is allowed in the WQ. If the bit is 1, the corresponding operation type is allowed to be used in the WQ; if the bit is 0, the corresponding operation type is not allowed to be used in the WQ. Bits corresponding to undefined operation codes are unused and are read as 0.

[1] Table 9-7 in this section describes when this field is RW and when it is RO.

9.2.25 Performance Monitoring Registers

The performance monitoring registers are a collection of registers in BAR0 to discover capabilities, configure and control the performance monitoring capabilities in Intel DSA. The capability registers include a global performance monitoring capability register (PERFCAP) and registers to describe per-event category (EVNTCAP) and optionally, per-counter (CNTRCAP) capabilities.

Perfmon Register	Conditions Under Which Register is Read-Only
CNTRCFG	All fields except Enable are read-only while the counter is enabled.
FLTCFG	Read-only while corresponding counter is enabled.
CNTRDATA	Read-only while corresponding counter is enabled, if the Counters Writeable while Enabled field in PERFCAP is 0.
PERFRST	Read-Write at all times.
OVFSTATUS	Read-Write at all times.
PERFFRZ	Read-Write at all times.

Table 9-8: Perfmon Register Read-Only Status

9.2.25.1 Performance Monitoring Capabilities Register (PERFCAP)

PERFCAP Base: BAR0			Offset: Table-Offset Size: 8 bytes (64 bits)
Bits	Attr	Size	Description
63:56	RO	8 bits	Unused
55	RO	1 bit	Interrupt on Overflow Support 0: Device does not support generation of interrupts upon counter overflow. 1: Device supports generation of interrupt upon counter overflow. Interrupt generation is controlled by the Interrupt on Overflow bit in the CNTRCFG registers.
54	RO	1 bit	Counter Freeze Support 0: Counter Freeze controls in PERFFRZ and CNTRCFG registers are not supported. 1: Counter Freeze controls in PERFFRZ and CNTRCFG are supported.
53	RO	1 bit	Counters Writeable While Enabled 0: Indicates that software is not allowed to write to a counter data register while that counter is enabled. Counter registers are always writeable while disabled. 1: Indicates that hardware supports writes to a counter data register while it is enabled.
52	RO	1 bit	Per Counter Capabilities Supported Indicates whether per counter capability registers are supported. 0: All supported counters have the same capabilities (i.e., can be used to monitor any of the supported events, can be used with any of the filter types etc.) and per counter capability registers are not supported. 1: Software should read the per-counter capability registers to identify the Event Categories, Events and Filters supported by each counter.
51:44	RO	8 bits	Unused

PERFCAP Base: BAR0			Offset: Table-Offset Size: 8 bytes (64 bits)
43:36	RO	8 bits	**Filters Supported** Bitmask indicating which Filters are supported in this implementation. If no filters are supported, then this field is 0. Note that even if this field is non-zero, not all filters may be supported for each Event. See Appendix D for information on which filters are supported for each Event. Table 6-2 describes the details for each of the filters supported. The number of Filter Configuration registers per counter corresponds to the number of bits set to 1 in this field.
35:20	RO	16 bits	**Global Event Categories Supported** Bitmask indicating the Event Categories that may be specified with any of the counters. If per-counter capabilities are supported, the value in CNTRCAP overrides the value specified here.
19:16	RO	4 bits	**Number of Event Categories Supported** The Event Categories are listed in Table 6-1. The EVNTCAP register corresponding to each supported Event Category indicates the events supported in that category.
15:8	RO	8 bits	**Counter Width** The number of bits supported per counter. If the value of this field is n, then each counter is an n-bit counter and the max value it can count is 2^n-1. If per-counter capabilities are supported, the counter width specified in the CNTRCAP registers overrides this value.
7:6	RO	2 bits	**Unused**
5:0	RO	6 bits	**Number of Performance Monitoring Counter Registers Supported** A value of 0 indicates that performance counters are not supported.

9.2.25.2 Performance Monitoring Event Capabilities Register (EVNTCAP)

Each EVNTCAP register corresponds to an Event Category and reports the set of events supported for that Event Category. The number of EVNTCAP registers corresponds to the number of Event Categories reported in PERFCAP. For example, if the number of Event Categories defined is 5, there will be five EVNTCAP registers, namely EVNTCAP_0, EVNTCAP_1 and so on, one for each of the Event Categories.

EVNTCAP_mmm Base: BAR0		Offset: Table-Offset + 0x80 + Event-Category-Num × 8	Size: 8 bytes (64 bits)
Bits	Attr	Size	Description
63:28	RSVD	36 bits	Reserved.
27:0	RO	28 bits	Events Bitmask of events supported for this Event Category. The Event Category that this register corresponds to, depends on the offset of this register. There is a separate EVNTCAP register for each Event Category supported in the implementation. Any bit that is 1 indicates that the corresponding event is supported. Note that the set of Events supported for any given Event Category is implementation-specific and may change in future implementations. When programming the CNTRCFG register with a particular Event Category value, if software sets Events bits not supported for that Event Category, those bits are ignored. If the implementation does not support any events for a given Event Category, this field is 0.

9.2.25.3 Performance Monitoring Counter Capabilities Register (CNTRCAP)

The CNTRCAP registers report the Event Categories and Events allowed for each counter. Implementations which do not have any restrictions on mapping of Event Categories to counters do not support these registers. These registers are present only if the Per Counter Capabilities Supported field in PERFCAP is 1. If present, the number of these capability registers corresponds to the number of Performance monitoring counter registers reported in PERFCAP. The values specified in each capability register apply only to the corresponding counter and override the values specified in PERFCAP.

CNTRCAP_nnn

Base: BAR0 Offset: Table-Offset + 0x800 + Counter-Num × 64 Size: 64 bytes (512 bits)

Bits	Attr	Size	Description
-	RO	28 bits	Events_k
-	RO	4 bits	Event Category_k
			...
95:68	RO	28 bits	Events_1
67:64	RO	4 bits	Event Category_1
63:36	RO	28 bits	Events_0 Specifies the Events supported in this counter register for the Event Category below.
35:32	RO	4 bits	Event Category_0 Specifies the first Event Category that can be enabled in this counter register.
31:28	RO	4 bits	Number of Event Entries This field indicates the number of records describing the Event Categories and Events supported by this counter register. For example, if a given counter can only count events corresponding to a single category (e.g., WQ related events), then this field will be 1 and there will be 1 pair of entries reported in the Event Category and Events fields in this register. If this field is 0, then this counter supports all Global Event Categories specified in PERFCAP.
27:8	RO	20 bits	Unused
7:0	RO	8 bits	Counter Width The value of this field represents the number of bits supported for this counter. If the value of this field is n, then the counter is an n-bit counter and the max value it can count is 2^n-1.

9.2.25.4 Performance Monitoring Reset Control Register (PERFRST)

The PERFRST register can be used by software to reset all the performance monitoring configuration and data registers to their default values.

PERFRST Base: BAR0			Offset: Table-Offset + 0x10 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:2	RSVD	30 bits	Reserved.
1	RWV	1 bit	**Reset Perfmon Counters** Software writes a 1 to this bit to reset all the Performance Monitoring Data registers. All the CNTRDATA registers are initialized to 0. Hardware clears this bit when all the counter data registers have been reset.
0	RWV	1 bit	**Reset Perfmon Configuration** Software writes a 1 to this bit to reset all the Performance Monitoring Configuration registers. All the CNTRCFG, FLTCFG, OVFSTATUS and PERFFRZ registers are initialized to default values. Hardware clears this bit when all the configuration registers have been reset.

9.2.25.5 Performance Monitoring Overflow Status Register (OVFSTATUS)

OVFSTATUS is a register used to indicate status across all the performance monitoring counters supported. Any bits beyond the number of counters reported in the PERFCAP register will be reported as 0 and should be ignored by software.

OVFSTATUS Base: BAR0			Offset: Table-Offset + 0x30 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:0	RW1C	32 bits	**Overflow Status** Bitmask with 1 bit per counter. Bit N indicates whether performance counter N has encountered an overflow condition. **0:** Counter has not encountered an overflow condition. **1:** Counter has encountered an overflow condition. Writing 1 clears the bit.

9.2.25.6 Performance Monitoring Freeze Register (PERFFRZ)

The PERFFRZ register can be used by software to control the freeze behavior and monitor the freeze status of all the performance monitoring counters. This register is present only if the Counter Freeze Support field in PERFCAP is 1.

PERFFRZ Base: BAR0			Offset: Table-Offset + 0x20 — Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:0	RWV	32 bits	**Freeze Control and Status** Bitmask with 1 bit per counter. Writing a 0 or 1 has the following impact on the corresponding counter: **0:** The counter is unfrozen and resumes counting unless CNTRCFG.Enable=0; in which case the counter remains disabled. If the counter is enabled but not currently frozen, it is unaffected and continues to count events. **1:** The counter, if enabled, gets frozen and stops counting further events, and retains its current value. If a counter is already frozen when this bit is set, it remains frozen. Reads return the current freeze status of each counter: **0:** The counter is currently not frozen. The counter may be disabled (CNTRCFG.Enable=0), or may be enabled and counting events. **1:** The counter is currently frozen and not counting events. It remains frozen until explicitly unfrozen by software. Bits corresponding to counters not supported by the hardware are ignored. Disabling a counter by setting CNTRCFG.Enable to 0 clears the freeze status for that counter.

9.2.25.7 Counter Configuration Register (CNTRCFG)

The CNTRCFG registers specify the set of events to be monitored by each counter. They also control interrupt generation behavior and the behavior upon overflow. The number of CNTRCFG registers corresponds to the number of counter registers specified in PERFCAP. The default value of these registers is 0. All fields except Enable are read-only while the counter is enabled.

CNTRCFG_nnn Base: BAR0			Offset: Table-Offset + 0x100 + Counter-Num × 8 — Size: 8 bytes (64 bits)
Bits	Attr	Size	Description
63:60	RSVD	4 bits	Reserved.
59:32	RWL	28 bits	**Events** Specifies the set of events to be monitored by this counter, corresponding to the Event Category selected. The set of supported events depends on the value of Event Category. Unsupported bits are ignored. The definition of some Events in each Event Category may be implementation specific.
31:12	RSVD	20 bits	Reserved.

<table>
<tr><td colspan="4">CNTRCFG_nnn
Base: BAR0   Offset: Table-Offset + 0x100 + Counter-Num × 8   Size: 8 bytes (64 bits)</td></tr>
<tr><td>11:8</td><td>RWL</td><td>4 bits</td><td>Event Category
Specifies the Event Category to associate with this counter. Based on the Event Category selected, different sets of events can be selected in the Events field.
<table>
<tr><td>Value</td><td>Event Category</td></tr>
<tr><td>0</td><td>WQ</td></tr>
<tr><td>1</td><td>Engine</td></tr>
<tr><td>2</td><td>Address Translation</td></tr>
<tr><td>3</td><td>Operations</td></tr>
<tr><td>4</td><td>Completions</td></tr>
<tr><td>5</td><td>Inter-Domain Operations</td></tr>
<tr><td>6-15</td><td>Reserved</td></tr>
</table>
</td></tr>
<tr><td>7:3</td><td>RSVD</td><td>5 bits</td><td>Reserved.</td></tr>
<tr><td>2</td><td>RWL</td><td>1 bit</td><td>Global Freeze on Overflow
0: No global freeze.
1: When an overflow is detected from this register, all counters in the device are frozen.
In either case, Overflow status is recorded in the OVFSTATUS register.
This bit is reserved if Counter Freeze Support in PERFCAP is 0.</td></tr>
<tr><td>1</td><td>RWL</td><td>1 bit</td><td>Interrupt on Overflow
0: No Interrupt is generated.
1: Generate a Performance monitoring Interrupt when this counter overflows.
This bit is reserved if Interrupt on Overflow Support in PERFCAP is 0.</td></tr>
<tr><td>0</td><td>RW</td><td>1 bit</td><td>Enable
0: This counter is disabled.
1: This counter is enabled to count events.</td></tr>
</table>

9.2.25.8 Filter Configuration Register (FLTCFG)

Each counter supports a set of Filter Configuration registers, one for each filter defined in Table 6-2. Software can program one or more Filter Configuration registers with the filter values to apply to that counter. For example, FLTCFG_WQ_0 selects the WQs to monitor for events in counter 0, FLTCFG_TC_2 selects the Traffic Classes to monitor for events in counter 2, and so on. Each FLTCFG register has a default value of all 1s which implies that no constraints are imposed by that filter. Table 9-9 shows an example set of register offsets for the set of Filter Configuration registers associated with each counter. This register is read-only while the corresponding counter is enabled.

FLTCFG_F_nnn Base: BAR0		Offset: Table-Offset + 0x300 + Counter-Num × 32 + Filter-Num × 4	Size: 4 bytes (32 bits)
Bits	**Attr**	**Size**	**Description**
31:16	RSVD	16 bits	Reserved.
15:0	RWL	16 bits	**Filter Value** Specifies the filter value to be used for the Filter associated with this register. It defaults to all 1s implying that all values are allowed. Bits beyond the max value allowed for that filter are ignored. For example, for the WQ filter, bits beyond the number of enabled WQs are ignored.

FLTCFG_PASID_nnn Base: BAR0		Offset: Table-Offset + 0x300 + Counter-Num × 32 + 0x14	Size: 4 bytes (32 bits)
Bits	**Attr**	**Size**	**Description**
31:22	RSVD	10 bits	Reserved.
21:0	RWL	22 bits	**Filter Value** Specifies the filter value to be used for the Filter associated with this register. It defaults to all 1s implying that all values are allowed. This field is read-only when the corresponding counter is enabled for counting. Bits beyond the max value allowed for that filter are ignored.

Filter Configuration Register	BAR0 Offset
FLTCFG_WQ_0	0x1300
FLTCFG_TC_0	0x1304
FLTCFG_PGSZ_0	0x1308
FLTCFG_SZ_0	0x130C
FLTCFG_ENG_0	0x1310
FLTCFG_PASID_0	0x1314
Unused	0x1318 - 0x131F
FLTCFG_WQ_1	0x1320
FLTCFG_TC_1	0x1324
…	…
FLTCFG_ENG_N	0x1300 + N × 0x20 + 0x10
FLTCFG_PASID_N	0x1300 + N × 0x20 + 0x14

Table 9-9: Filter Configuration Register Offsets

9.2.25.9 Counter Data Register (CNTRDATA)

Each CNTRDATA register is an N-bit counter that is used to count occurrences of configured events, where N is the value of the Counter Width field in PERFCAP. Behavior of software reads and writes to these registers are described in section 6.3. Once written, the counter continues to increment from the written value. A freeze operation causes the counter to stop accumulating further events and to retain its value at the time of freeze. An unfreeze operation allows the counter to resume counting subsequent events.

<table>
<tr><td colspan="4"><b>CNTRDATA_nnn</b>
<b>Base: BAR0</b>   <b>Offset: Table-Offset + 0x200 + Counter-Num × 8</b>   <b>Size: 8 bytes (64 bits)</b></td></tr>
<tr><th>Bits</th><th>Attr</th><th>Size</th><th>Description</th></tr>
<tr><td>63:N</td><td>RSVD</td><td>64-N bits</td><td><b>Ignored</b></td></tr>
<tr><td>N-1:0</td><td>RWLV</td><td>N bits</td><td><b>Event Count Value</b>
N-bit performance event counter where N is the value of the Counter Width field in PERFCAP.
If the Counters Writeable while Enabled field in PERFCAP is 0, then this register is read-only while the counter is Enabled.</td></tr>
</table>

9.2.26 MSI-X Table

BAR0; Offset: given by the MSI-X capability; Size: 16 bytes × number of entries (2 × 64 bits × number of entries). See the PCI Express specification listed in section 1.2 for details of this table. The offset and number of entries are in the MSI-X capability. See section 3.7 for information on how the MSI-X table is used.

9.2.27 MSI-X Pending Bit Array

BAR0; Offset: given by the MSI-X capability; Size: ⌈number of entries ÷ 64⌉ × 64 bits. (Note the use of the ceiling function in the above equation to round up the result of the division to the nearest integer.) See the PCI Express specification listed in section 1.2 for details of this table. The offset and number of entries are in the MSI-X capability.

9.2.28 Interrupt Message Storage

If the Interrupt Message Storage Support field in the Scalable IOV capability is 1, the Interrupt Message Storage contains interrupt messages in addition to those in the MSI-X table defined in the PCI Express specification. The format of this table is like that of the MSI-X table, except that:

- The pending bit for each entry is in the Control field instead of in a separate pending bit array.
- Several additional controls are defined in the Control field.
- The size of the IMS table is not limited to 2048 entries. (However, the size of this table may vary between different Intel DSA implementations and may be less than 2048 entries.)

The offset of the Interrupt Message Storage is given by the IMS Offset field in the Table Offsets register. The number of entries is given by the Interrupt Message Storage Size field in GENCAP. See section 3.7 for information on how this table is used.

If the Interrupt Message Storage Support field in the Scalable IOV capability is 0, this table is not present.

The initial value of Message Address is 00000000FEE00000h. If the value written to the Message Address field of the IMS entry does not contain 00000000FEEh in the upper 44 bits, the value written is ignored. (The previously stored value is retained.) Bits 1:0 of the value written to Message Address are ignored.

IMS entry
Base: BAR0 Offset: Table-offset + index × 16 Size: 16 bytes (4 × 32 bits)

Bytes	Bits	Attr	Size	Description
7:0	63:0	RW	64 bits	Message Address See description for constraints on the value that may be written to this register.
11:8	31:0	RW	32 bits	Message Data
15:12	31:12	RW	20 bits	PASID If PASID Enable is 1, this field is checked against the PASID field of the descriptor. See section 5.4.
	11:4	RSVD	8 bits	Reserved.
	3	RW	1 bit	PASID Enable This field is checked against the WQ PASID Enable field of the WQ the descriptor was submitted to. See section 5.4.
	2	RW	1 bit	Ignore If this field is 1 when a descriptor completion interrupt references this IMS entry, no interrupt is generated and the Pending field is not set. This field does not prevent delivery of an interrupt if Pending is 1 and Mask is cleared, nor does it affect delivery of interrupts due to causes other than descriptor completion.
	1	RO	1 bit	Pending This field is set to 1 when an interrupt is raised using this IMS entry and the Mask field is 1. This field becomes 0 when the interrupt is generated.
	0	RW	1 bit	Mask When this field is 1, no interrupt is generated using this IMS entry. Instead, the Pending field is set to 1. If 0 is written to this field when the Pending field is 1, an interrupt is generated.

9.2.29 Inter-Domain Permissions Table (IDPT)

If the Inter-Domain Support field in GENCAP is 1, the Inter-Domain Permissions Table controls the mapping of work submitters and alternate access PASIDs, corresponding permissions, and the memory regions permitted to be accessed. The offset of the Inter-Domain Permissions Table is given by the Inter-Domain Permissions Table Offset field in the Table Offsets register. The number of entries is given by the Inter-Domain Permissions Table Size field in IDCAP. See section 3.14.1 for information on how this table is used. If Inter-Domain Support in GENCAP is 0, this table is not present. The initial value of each entry in the table is 0.

Each 32-byte Inter-Domain Permissions Table entry (IDPTE) is divided into four 64-bit sub-entries. The fields of an IDPTE are read-only or read-write at different times, depending on the Usable and Allow Update fields, as detailed in the table. Any writes to fields while they are read-only are ignored.

Field	Conditions Under Which Field is Read-Only
Submitter Bitmap Address	Read-only while Usable is 1.
Window Size	Read-only while Usable is 1 or Allow Update is 1.[1]
Window Base	Read-only while Usable is 1 or Allow Update is 1.[1]
Access PASID	Read-only while Usable is 1 or Allow Update is 1.
Window Mode	Read-only while Usable is 1 or Allow Update is 1.[1]
Window Enable	Read-only while Usable is 1 or Allow Update is 1.[1]
Access Privilege	Read-only while Usable is 1 or Allow Update is 1.
Write Permissions	Read-only while Usable is 1 or Allow Update is 1.[1]
Read Permissions	Read-only while Usable is 1 or Allow Update is 1.[1]
Submitter PASID	Read-only while Usable is 1.
Type	Read-only while Usable is 1 or Allow Update is 1.
Allow Update	Read-write at all times.
Usable	Read-write at all times.

Table 9-10: Inter-Domain Permissions Table Entry Read-Only Status

[1] While Allow Update is 1, this field may be updated using an Update Window descriptor even if it is read-only with respect to MMIO writes.

IDPT
Base: BAR0 Offset: Table-Offset + Index × 32 Size: 32 bytes (4*64bits)

Bytes	Bits	Attr	Size	Description
31:24	63:12	RWL	52 bits	Submitter Bitmap Address This field specifies the upper 52 address bits of a 4 KB aligned submitter bitmap region in memory. Bits 11:0 are not specified and are treated as 0. If Type is 1, a descriptor's PASID must have a corresponding bit set to 1 in the bitmap. If Type is 0, this field is reserved. This field is read-only while Usable is 1.
	11:0	RSVD	12 bits	Reserved.
23:16	63:0	RWL	64 bits	Window Size Size in bytes of the memory window. This field is read-only while Usable is 1 or Allow Update is 1. This field is reserved if Window Enable is 0.
15:8	63:0	RWL	64 bits	Window Base Base address of the memory window. This field is read-only while Usable is 1 or Allow Update is 1. This field is reserved if Window Enable is 0.
7:4	31:12	RWL	20 bits	Access PASID This is the PASID used to access memory. This field is read-only while Usable is 1 or Allow Update is 1.
	11:5	RSVD	7 bits	Reserved.
	4	RWL	1 bit	Window Mode 0: Address Mode. Address field in the descriptor must be greater than or equal to the window base and must be less than the sum of window base and window size. 1: Offset Mode. Address field in the descriptor is an offset and must be less than the window size. Descriptor address is added to the window base to compute the real address. This field is read-only while Usable is 1 or Allow Update is 1. This field is reserved if Window Enable is 0 or if Offset Mode Support in IDCAP is 0.
	3	RWL	1 bit	Window Enable 0: Memory window checking is disabled, and hardware does not perform address range checks. The IDPTE grants access to the entire address space of the access PASID. 1: Memory window checking is enabled; address range check is performed against the corresponding address in the descriptor. This field is read-only while Usable is 1 or Allow Update is 1.
	2	RWL	1 bit	Access Privilege This controls the privilege used to access memory. 0: Overrides the privilege of the descriptor. Memory access using this entry specifies user privilege (priv field equal to 0). 1: Memory access using this entry uses the privilege of the descriptor. This field is read-only while Usable is 1 or Allow Update is 1.

IDPT Base: BAR0			Offset: Table-Offset + Index × 32	Size: 32 bytes (4*64bits)
	1	RWL	1 bit	Write Permissions 0: Memory writes are not allowed using this entry. 1: Memory writes are allowed using this entry This field is read-only while Usable is 1 or Allow Update is 1.
	0	RWL	1 bit	Read Permissions 0: Memory reads are not allowed using this entry. 1: Memory reads are allowed using this entry. This field is read-only while Usable is 1 or Allow Update is 1.
3:0	31:12	RWL	20 bits	Submitter PASID If Type is 0, this field must match the PASID in the descriptor. If Type is 1, this field is reserved. This field is read-only while Usable is 1.
	11:4	RSVD	8 bits	Reserved.
	3:2	RWL	2 bits	Type 0: Single-access, single-submitter entry (SASS) 1: Single-access, multi-submitter entry (SAMS) 2: Reserved 3: Reserved This field is read-only while Usable is 1 or Allow Update is 1. See section 3.14.1 for details on the different types of entries.
	1	RW	1 bit	Allow Update 0: Window attributes cannot be modified using an Update Window descriptor. Window attributes may be modified by writing to the table entry while Usable is 0. 1: Window attributes may be modified using an Update Window descriptor.
	0	RW	1 bit	Usable 0: A handle in an inter-domain descriptor may not reference this entry. 1: A handle in an inter-domain descriptor may reference this entry.

9.3 Portals (BAR2)

Portals are used to submit descriptors to the device. Portals are located in the address space specified by BAR2. Each portal is 64 bytes in size and is located on a separate 4 KB page. This allows the portals to be independently mapped into different address spaces using CPU page tables and extended page tables.

There are four portals per WQ, as shown in Figure 9-2. Bits 5:0 of the portal address must be 0. Bits 11:6 are ignored; thus any 64-byte-aligned address on the page can be used with the same effect.

Descriptor submissions to a portal for an SWQ must be performed using 64-byte Deferrable Memory Write transactions (DMWr). Any other write operation to an SWQ portal is ignored. Descriptor submissions to a DWQ must be performed using a 64-byte write operation. On Intel CPUs, software should use the MOVDIR64B instruction to generate a non-torn 64-byte write. A DMWr transaction to a disabled or dedicated WQ portal returns Retry. Any other write operation to a DWQ portal is ignored. Any read operation to the BAR2 address space returns 0x00 or 0xFF in all bytes. Reads to the BAR 2 address space are not ordered with respect to any other transactions to the device and cannot be used to ensure that upstream write operations have completed. See section 5.3 for more information on error checking and reporting related to portal accesses.

	64-byte DMWr (ENQCMD or ENQCMDS)	64-byte posted write (MOVDIR64B)	Non-64-byte write	Read
Shared WQ	Supported	Ignored	Ignored	Returns 0x00 or 0xFF in all bytes
Dedicated WQ	Returns Retry	Supported		
Disabled WQ	Returns Retry	Ignored		

Table 9-11: Supported Portal Operations

offset		
0000h	Unlimited MSI-X Portal	WQ 0
1000h	Limited MSI-X Portal	
2000h	Unlimited IMS Portal	
3000h	Limited IMS Portal	
4000h	Unlimited MSI-X Portal	WQ 1
5000h	Limited MSI-X Portal	
6000h	Unlimited IMS Portal	
7000h	Limited IMS Portal	
8000h	Unlimited MSI-X Portal	WQ 2
9000h	Limited MSI-X Portal	
A000h	Unlimited IMS Portal	
B000h	Limited IMS Portal	
C000h	Unlimited MSI-X Portal	WQ 3
D000h	Limited MSI-X Portal	
E000h	Unlimited IMS Portal	
F000h	Limited IMS Portal	
	...	
P + 0000h	Unlimited MSI-X Portal	WQ N - 1
P + 1000h	Limited MSI-X Portal	
P + 2000h	Unlimited IMS Portal	
P + 3000h	Limited IMS Portal	

N = Number of WQs
P = (N – 1) × 4000h

Figure 9-2: Portals

§

Appendix A CRC Computation

Intel DSA computes CRC using a method that produces results matching the following description.

Intel DSA computes 32-bit CRC using the polynomial 0x11edc6f41 following the specification in the iSCSI Protocol (RFC 3720).

For 64-bit CRC, Intel DSA uses the polynomial 0x1ad93d23594c93659.

The following description is adapted from RFC 3720. ‘w’ is the CRC Size, either 32 or 64. ‘n’ is the number of data bits. When the Transfer Size is not a multiple of w bits, the source data is padded on the end with zeros to a multiple of w bits.

- The data bits are considered as the coefficients of a polynomial M(x) of degree n-1. The least significant bit (bit 0) of the first byte of the data is the coefficient of the most significant term (x^{n-1}), followed by bit 1 of the first byte, and so on through bit 7 of the highest numbered byte (x^0).
- The most significant w bits of the data are complemented.
- The polynomial is multiplied by x^w, then divided by G(x). The generator polynomial produces a remainder R(x) of degree $\leq w - 1$.
- The coefficients of R(x) are considered a w-bit sequence.
- The bit sequence is complemented, and the result is the CRC value.
- The bits of the CRC value are stored in the CRC Value field of the completion record as follows: The x^{w-1} coefficient is stored in the least significant bit (bit 0) of byte 0 of the field, followed by the x^{w-2} coefficient in bit 1, and so on through the x^0 coefficient in the most significant bit (bit 7) of the most-significant byte of the CRC Value field.

The CRC Seed field of the descriptor or the CRC seed read from memory follows the same byte/bit ordering described for the CRC Value field in the completion record.

Intel DSA computes CRC using the same polynomial as the CRC32 instruction described in the Intel® 64 and IA-32 Architectures Software Developer Manual. However, software that expects to use the two mechanisms interchangeably must pay attention to the CRC algorithm requirements for inversion and reflection of the CRC seed and result in order to obtain consistent results.

§

Appendix B Data Integrity Field (DIF)

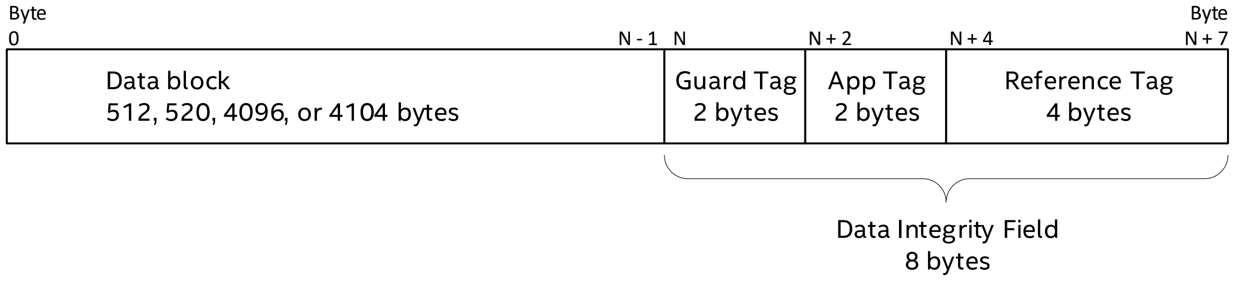
The Data Integrity Field (DIF) provides a system solution to protect the communication path between a host and storage device for end-to-end data integrity. Enterprise drives can be formatted with sector sizes that include an extra 8 bytes of information per sector which can be used to store integrity information. DIF was introduced as a way to use those extra bytes in an open standard.

Intel DSA performs DIF computation on a block of source data organized in 512B, 520B, 4096B, or 4104B blocks. It can check, strip, insert, or update the Guard Tag, Application Tag, and Reference Tag fields from source data and write the result to a destination buffer. The DIX generate operation may be used to generate the Guard Tag, Application Tag, and Reference Tag fields from source data, and write them to the destination address.

The 8 bytes of DIF information are divided up as follows:

- A 16-bit Guard Tag (CRC of the data, using the polynomial 0x18bb7).
- A 16-bit Application Tag.
- A 32-bit Reference Tag.

The guard tag protects the data portion of the sector. The application tag is opaque storage information. The reference tag protects against out-of-order and misdirected writes. Standardizing the contents of the protection information enables all nodes in the I/O path, including the disk itself, to verify the integrity of the data block.



The Guard Tag, Application Tag, and Reference Tag are stored in memory with the most-significant byte at the lowest address; that is, in big-endian format.

Reference Tag

The Reference Tag is initialized from the Reference Tag Seed field in the descriptor. The tag may be fixed or incrementing. If the tag is fixed, the seed in the descriptor is used for all blocks in the transfer. If the tag is incrementing, the seed is used for the first block in the transfer, and the value is incremented by one for each subsequent block in the transfer. If incrementing a tag value overflows the width of the tag, it wraps to 0. The final value is written to the completion record. (The final value is the value that would be used for the block after the last completed block.)

For the DIF Update operation, there are separate fields in the descriptor for the Source Reference Tag Seed and Destination Reference Tag Seed. There are separate flags to control whether the source and destination tags are fixed or incrementing. The source tag fields are used to determine the expected tag values in the source data, while the destination tag fields are used to determine the tag values to be written to the destination. However, there is a flag in the descriptor to force the Reference Tag values read from the source to be written to the destination. In this case, the destination tag fields in the descriptor are ignored.

Application Tag

The Application Tag is initialized from the Application Tag Seed field in the descriptor. The tag may be fixed or incrementing, and the final value is written to the completion record, similar to the Reference Tag. The Application Tag Mask is applied to the application tag value before using it. Bits in the tag value corresponding to 0 bits in the mask are retained, while bits in the tag value corresponding to 1 bits in the mask are forced to 0. If the application tag is incrementing, the mask is applied after incrementing the tag value. Depending on the bits set in the mask, the effect of incrementing the tag value may be masked off, resulting in the same tag value being used for multiple blocks. To avoid this, the application tag mask should typically be set to mask only higher-order bits. The Source Application Tag Mask may be set to 0xffff to disable application tag checking.

For the DIF Update operation, there are separate fields in the descriptor to determine the source and destination Application Tag values, just as there are for the Reference Tag. There are also separate fields for the Source Application Tag Mask and Destination Application Tag Mask. As for the Reference Tag, there is a flag in the descriptor to force the Application Tag values read from the source to be written to the destination.

Guard Tag

The Guard Tag is computed from the source data using the T10 CRC polynomial:

$$G(x) = x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

normally written as 0x18bb7.

The CRC algorithm treats the source data as a polynomial F(x), where each bit of the source data is considered the coefficient of the corresponding term of the polynomial. F(x) is divided by G(x) to find the remainder R(x).

$$\frac{F(x)}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

The CRC is created by concatenating the coefficients of each term of the remainder R(x).

The Guard Tag in the source data is checked using one of two methods:

1. Compute the CRC on the source data as described above and compare it to the Guard Tag in the source DIF.
2. Append the Guard Tag from the source DIF to the source data, compute the remainder, and check that the remainder is 0.

The two methods are mathematically equivalent.

The default value of the T10 CRC seed is 0. To provide flexibility to software to use a different seed, the Invert CRC Seed flag in the DIF Flags field of the descriptor causes 0xffff to be used as the CRC seed. The first 16 bits of the source data are XORed with the seed before performing the CRC computation.

The Invert CRC Result flag causes the computed CRC value to be inverted before comparing or storing the Guard Tag.

B.1 DIF Check

The DIF Check operation is used to check the validity of the Data Integrity Fields in the source data. When performing a DIF Check operation, Intel DSA performs the following actions on each block of source data and the associated DIF:

- Optionally calculate the Guard Tag and compare it to the Guard Tag field in the source DIF value.
- Optionally verify the Application Tag and Reference Tag in the source DIF value.
- Update the Application Tag and Reference Tag for the next block of data, based on the Source DIF Flags field of the descriptor.

At least one of the Guard Tag, Application Tag, or Reference Tag should be checked; otherwise, this operation does nothing.

B.2 DIF Insert

The DIF Insert operation is used to add Data Integrity Fields when the source data does not contain them. When performing a DIF Insert operation, the device performs the following actions on each block of source data:

- Calculate the Guard Tag.
- Combine the Guard Tag, Application Tag and Reference Tag into a DIF value.
- Write the source data to the destination, appending the DIF value.
- Update the Application Tag and Reference Tag for the next block of data, based on the Destination DIF Flags field of the descriptor.

For a DIF Insert operation, the destination buffer size is given by

$$\text{Destination Buffer Size} = TS + \left(\frac{TS}{BS}\right) \times 8$$

where TS is the Transfer Size (source data size) and BS is the DIF Block Size.

B.3 DIF Strip

The DIF Strip operation is used to remove Data Integrity Fields from the source data. Intel DSA can optionally check the validity of the fields as it removes them. When performing a DIF Strip operation, the device performs the following actions on each block of source data and the associated DIF:

- Optionally calculate the Guard Tag and compare it to the Guard Tag field in the source DIF value.
- Optionally verify the Application Tag and Reference Tag in the source DIF value.
- Write the source data (without the DIF) to the destination.
- Update the Application Tag and Reference Tag for the next block of data, based on the Source DIF Flags field of the descriptor.

For a DIF Strip operation, the destination buffer size is given by

$$\text{Destination Buffer Size} = TS - \left(\frac{TS}{BS + 8}\right) \times 8$$

where TS is the Transfer Size (source data size) and BS is the DIF Block Size.

B.4 DIF Update

The DIF Update operation is used to replace the Data Integrity Fields in the source data with fresh values. Intel DSA can optionally check the validity of the fields in the source data. When performing a DIF Update operation, the device performs the following actions on each block of source data and the associated DIF:

- Calculate the Guard Tag value.
- Optionally compare the computed Guard Tag value to the Guard Tag field in the source DIF value.
- Optionally verify the Source Application Tag and Source Reference Tag in the source DIF value.
- Combine the computed Guard Tag, the Destination Application Tag, and the Destination Reference Tag into a destination DIF value.
- Write the source data to the destination, with the source DIF value replaced by the destination DIF value.
- Update the Source Application Tag and Source Reference Tag for the next block of data, based on the Source DIF Flags field of the descriptor.
- Update the Destination Application Tag and Destination Reference Tag for the next block of data, based on the Destination DIF Flags field of the descriptor.

For a DIF Update operation, the destination buffer size is the same as the Transfer Size.

The required destination buffer size for various DIF operations can be computed as shown in this table. If a DIF operation does not fully complete, the bytes written to the destination can be computed from the Bytes Completed field of the completion record.

BS = DIF block size
N = number of blocks to process
M = number of blocks completed

	Transfer Size	Destination Buffer Size	Bytes Completed	Bytes Written to Destination (Can be computed by SW)
DIF-Check	(BS+8) × N	N/A	(BS+8) × M	N/A
DIF-Strip	(BS+8) × N	(BS) × N	(BS+8) × M	(BS) × M
DIF-Insert	(BS) × N	(BS+8) × N	(BS) × M	(BS+8) × M
DIF-Update	(BS+8) × N	(BS+8) × N	(BS+8) × M	(BS+8) × M

B.5 DIX Generate

The DIX Insert operation is used to compute Data Integrity Fields for the specified source data. When performing a DIX Generate operation, the device performs the following actions on each block of source data:

- Calculate the Guard Tag.
- Combine the Guard Tag, Application Tag, and Reference Tag into a DIF value.
- Write the DIF value to the destination.
- Update the Application Tag and Reference Tag for the next block of data, based on the Destination DIF Flags field of the descriptor.

For a DIX Generate operation, the destination buffer size is given by:

$$\text{Destination Buffer Size} = \left(\frac{TS}{BS}\right) \times 8$$

where TS is the Transfer Size (source data size) and BS is the DIF Block Size.

§

Appendix C PCIe Configuration Registers

This appendix provides details of PCIe configuration registers for version 1.0 of Intel DSA. The Version register is described in section 9.2.1.

Vendor ID (VID)

VENDOR ID (VID)
Identifies the manufacturer of the device.
Base: Rootbus CFG Offset: 0x0 Size: 2 bytes (16 bits)
Default Value: 0x8086

Bits	Attr	Size	Default Val	Description
15:0	RO	16	0x8086	**Vendor ID (VID)** Indicates Intel (8086h).

Device ID (DID)

DEVICE ID (DID)
Identifies the particular device.
Base: Rootbus CFG Offset: 0x2 Size: 2 bytes (16 bits)
Default Value: 0x0B25

Bits	Attr	Size	Default Val	Description
15:0	ROS	16	0x0B25	**Device ID (DID)** Allocated by the vendor.

PCI Command (PCICMD)

PCI COMMAND (PCICMD)
The Command register provides coarse control over a device's ability to generate and respond to PCI cycles. When a 0 is written to this register, the device is logically disconnected from the PCI bus for all accesses except configuration accesses.
Base: Rootbus CFG Offset: 0x4 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:11	RSVD	5	0x00	Reserved.
10	RO	1	0x0	**Interrupt Disable (INTD)** Controls the ability of the Function to generate INTx interrupts. This Function does not generate INTx interrupts, so this bit it hardwired to 0b.
9	RO	1	0x0	**Fast Back-to-Back Enable (FBE)** Does not apply to PCI Express and is hardwired to 0b.

PCI COMMAND (PCICMD)
The Command register provides coarse control over a device's ability to generate and respond to PCI cycles. When a 0 is written to this register, the device is logically disconnected from the PCI bus for all accesses except configuration accesses.
Base: Rootbus CFG Offset: 0x4 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
8	RW	1	0x0	**SERR# Enable (SEE)** When Set, this bit enables reporting of Non-Fatal and Fatal errors detected by the Function to the Root Complex. Note that errors are reported if enabled either through this bit or through the PCI Express specific bits in the Device Control register.
7	RO	1	0x0	**Wait Cycle Control (WCC)** Does not apply to PCI Express and is hardwired to 0b.
6	RW	1	0x0	**Parity Error Response Enable (PERE)** This bit controls the logging of poisoned TLPs in the Master Data Parity Error bit in the Status register.
5	RO	1	0x0	**VGA Palette Snoop Enable (VGAPSE)** Does not apply to PCI Express and is hardwired to 0b.
4	RO	1	0x0	**Memory Write and Invalidate Enable (MWIE)** Does not apply to PCI Express and is hardwired to 0b.
3	RO	1	0x0	**Special Cycle Enable (SCE)** Does not apply to PCI Express and is hardwired to 0b.
2	RW	1	0x0	**Bus Master Enable (BME)** Controls the ability of the endpoint to issue Memory Read/Write requests. When set, the Function is allowed to issue Memory Requests. When clear, the Function is not allowed to issue Memory Requests. Note that as interrupt messages are in-band memory writes, setting BME to 0b disables interrupt messages as well. Requests other than Memory Requests (e.g., Completion) are not controlled by this bit.
1	RW	1	0x0	**Memory Space Enable (MSE)** Controls the Function's response to Memory Space accesses. A value of 0 disables the response. A value of 1 allows the Function to respond to Memory Space accesses.

PCI COMMAND (PCICMD)
The Command register provides coarse control over a device's ability to generate and respond to PCI cycles. When a 0 is written to this register, the device is logically disconnected from the PCI bus for all accesses except configuration accesses.
Base: Rootbus CFG Offset: 0x4 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
0	RO	1	0x0	**I/O Space Enable (IOSE)** Controls the Function's response to I/O Space accesses. A value of 0 disables the response. Hardwired to 0 as this Function does not support I/O Space accesses.

PCI Status (PCISTS)

PCI STATUS (PCISTS)
The Status register is used to record status information for PCI bus related events.
Base: Rootbus CFG Offset: 0x6 Size: 2 bytes (16 bits)
Default Value: 0x0010

Bits	Attr	Size	Default Val	Description
15	RW1C	1	0x0	**Detected Parity Error (DPE)** This bit is Set by a Function whenever it receives a Poisoned TLP, regardless of the state the Parity Error Response bit in the Command register.
14	RW1C	1	0x0	**Signaled System Error (SSE)** This bit is Set when a Function sends an ERR_FATAL or ERR_NONFATAL Message, and the SERR# Enable bit in the Command register is 1.
13	RW1C	1	0x0	**Received Master Abort (RMA)** This bit is Set when a Requester receives a Completion with Unsupported Request Completion Status.
12	RW1C	1	0x0	**Received Target Abort (RTA)** This bit is Set when a Requester receives a Completion with Completer Abort Completion Status.
11	RW1C	1	0x0	**Signaled Target Abort (STA)** This bit is Set when a Function completes a Posted or Non-Posted Request as a Completer Abort error.
10:9	RO	2	0x0	**DEVSEL Timing (DT)** Does not apply to PCI Express and is hardwired to 00b.

PCI STATUS (PCISTS)
The Status register is used to record status information for PCI bus related events.
Base: Rootbus CFG Offset: 0x6 Size: 2 bytes (16 bits)
Default Value: 0x0010

Bits	Attr	Size	Default Val	Description
8	RW1C	1	0x0	**Master Data Parity Error (MDPE)** This bit is Set by an Endpoint Function if the Parity Error Response bit in the Command register is 1b it either receives a Poisoned Completion or transmits a Poisoned Request.
7	RO	1	0x0	**Fast Back-to-Back Transactions Capable (FBTC)** Does not apply to PCI Express and is hardwired to 0b.
6	RSVD	1	0x0	Reserved.
5	RO	1	0x0	**66 MHz Capable (C66)** Does not apply to PCI Express and is hardwired to 0b.
4	RO	1	0x1	**Capabilities List (CL)** Indicates the presence of an Extended Capability list item. Required by all PCI Express endpoints.
3	RO	1	0x0	**Interrupt Status (INTS)** When Set, indicates that an INTx emulation interrupt is pending internally in the Function. Hardwired to 0b as this Function does not support INTx.
2:0	RSVD	3	0x0	Reserved.

Revision ID (RID)

REVISION ID (RID)
This register specifies a device specific revision identifier.
Base: Rootbus CFG Offset: 0x8 Size: 1 byte (8 bits)
Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	ROS	8	0x00	**Revision ID (RID)** The value is chosen by the vendor. This field should be viewed as a vendor defined extension to the Device ID.

Class Code Register-Level Programming Interface (CCRLPI)

CLASS CODE REGISTER-LEVEL PROGRAMMING INTERFACE (CCRLPI)

The Class Code register is read-only and is used to identify the generic function of the device and, in some cases, a specific register-level programming interface. The lower byte identifies a specific register-level programming interface (if any) so that device independent software can interact with the device.

Base: Rootbus CFG Offset: 0x9 Size: 1 byte (8 bits)

Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x00	**Register-Level Programming Interface (RLPI)** Other system peripheral.

Class Code Sub-Class (CCSC)

CLASS CODE SUB-CLASS (CCSC)

The Class Code register is read-only and is used to identify the generic function of the device and, in some cases, a specific register-level programming interface. The middle byte is a sub-class code which identifies more specifically the function of the device.

Base: Rootbus CFG Offset: 0x0A Size: 1 byte (8 bits)

Default Value: 0x80

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x80	**Sub-Class (SC)** Other system peripheral.

Class Code Base Class (CCBC)

CLASS CODE BASE CLASS (CCBC)

The Class Code register is read-only and is used to identify the generic function of the device and, in some cases, a specific register-level programming interface. The upper byte is a base class code which broadly classifies the type of function the device performs.

Base: Rootbus CFG Offset: 0x0B Size: 1 byte (8 bits)

Default Value: 0x08

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x08	**Base Class (BC)** Generic system peripheral.

Cache Line Size (CLS)

CACHE LINE SIZE (CLS)
The Cache Line Size register is set by the system firmware or the operating system to system cache line size.
Base: Rootbus CFG Offset: 0x0C Size: 1 byte (8 bits)
Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RW	8	0x00	**Cache Line Size (CLS)** This field is implemented as a read-write field for legacy compatibility purposes but has no effect on any device behavior.

Latency Timer (LATTMR)

LATENCY TIMER (LATTMR)
This register is also referred to as Primary Latency Timer for Type 1 Configuration Space header Functions.
Base: Rootbus CFG Offset: 0x0D Size: 1 byte (8 bits)
Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x00	**Latency Timer (LATTMR)** The Latency Timer does not apply to PCI Express. This register is hardwired to 00h.

Header Type (HDR)

HEADER TYPE (HDR)
This register identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and also whether or not the Device might contain multiple Functions.
Base: Rootbus CFG Offset: 0x0E Size: 1 byte (8 bits)
Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7	RO	1	0x0	**Multi-Function Device (MFD)** When Clear, software must not probe for Functions other than Function 0. Hardwired to 0b as this is a single Function Device.
6:0	RO	7	0x00	**Header Type (HT)** Indicates Type 0 Configuration Space Header.

Built-in Self-Test (BIST)

BUILT-IN SELF-TEST (BIST)
This optional register is used for control and status of BIST.
Base: Rootbus CFG Offset: 0x0F Size: 1 byte (8 bits)
Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x00	**Revision ID (BIST)** Devices that do not support BIST must always return a value of 0.

Base Address 0 (BAR0)

BASE ADDRESS 0 (BAR0)
Size, type, and location of address range for control registers.
Base: Rootbus CFG Offset: 0x10 Size: 8 bytes (64 bits)
Default Value: 0x00000000_0000000C

Bits	Attr	Size	Default Val	Description
63:4	RW	60	0x0	**Address (ADDR)** Bits N-1:4 are hardwired to 0, where 2^N is the size of the BAR region.
3	RO	1	0x1	**Pre-Fetchable (PF)** This address map is pre-fetchable but assumes that the IP is integrated into a platform that does not do write merging beyond aligned 8-byte accesses.
2:1	RO	2	0x2	**BAR Type (BT)** Base register is 64 bits wide and can be mapped anywhere in the 64-bit address space.
0	RO	1	0x0	**Memory Space Indicator (MSI)** Base Address registers that map to Memory Space must return a 0 in bit 0.

Base Address 2 (BAR2)

BASE ADDRESS 2 (BAR2)
Size, type, and location of address range for portals.
Base: Rootbus CFG Offset: 0x18 Size: 8 bytes (64 bits)
Default Value: 0x00000000_0000000C

Bits	Attr	Size	Default Val	Description
63:4	RW	60	0x0	**Address (ADDR)** Bits N-1:4 are hardwired to 0, where 2^N is the size of the BAR region.
3	RO	1	0x1	**Pre-Fetchable (PF)** This address map is pre-fetchable but assumes that the IP is integrated into a platform that does not do write merging beyond aligned 8-byte accesses.

BASE ADDRESS 2 (BAR2)
Size, type, and location of address range for portals.
Base: Rootbus CFG Offset: 0x18 Size: 8 bytes (64 bits)
Default Value: 0x00000000_0000000C

Bits	Attr	Size	Default Val	Description
2:1	RO	2	0x2	**BAR Type (BT)** Base register is 64 bits wide and can be mapped anywhere in the 64-bit address space.
0	RO	1	0x0	**Memory Space Indicator (MSI)** Base Address registers that map to Memory Space must return a 0 in bit 0.

Sub-System Vendor ID (SSVID)

SUB-SYSTEM VENDOR ID (SSVID)
This register (along with SSID) is used to uniquely identify the subsystem where the PCI device resides.
Base: Rootbus CFG Offset: 0x2C Size: 2 bytes (16 bits)
Default Value: 0x8086

Bits	Attr	Size	Default Val	Description
15:0	RW	16	0x8086	**Sub-System Vendor ID (SSVID)** This field should be written by boot SW.

Sub-System ID (SSID)

SUB-SYSTEM ID (SSID)
This register (along with SSVID) is used to uniquely identify the subsystem where the PCI device resides.
Base: Rootbus CFG Offset: 0x2E Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:0	RW	16	0x0000	**Sub-System ID (SSID)** This field should be written by boot SW.

Capabilities Pointer (CAPPTR)

CAPABILITIES POINTER (CAPPTR)
This optional register is used to point to a linked list of new capabilities implemented by this device. This register is only valid if the Capabilities List bit in the Status Register is set.
Base: Rootbus CFG Offset: 0x34 Size: 1 byte (8 bits)
Default Value: 0x40

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x40	**CAPPTR (CAPPTR)** Points to PCI Express Capability.

Interrupt Line (INTL)

INTERRUPT LINE (INTL)

The Interrupt Line register communicates interrupt line routing information.

Base: Rootbus CFG Offset: 0x3C Size: 1 byte (8 bits)

Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x00	**Interrupt Line (INTL)** Hardwired to 00h as this Function does not use an Interrupt Pin.

Interrupt Pin (INTP)

INTERRUPT PIN (INTP)

The Interrupt Pin register is a read-only register that identifies the legacy interrupt Message(s) the Function uses.

Base: Rootbus CFG Offset: 0x3D Size: 1 byte (8 bits)

Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x00	**Interrupt Pin (INTP)** A value of 00h indicates that the Function uses no legacy interrupt Message(s).

Minimum Grant (MINGNT)

MINIMUM GRANT (MINGNT)

Does not apply to PCI Express.

Base: Rootbus CFG Offset: 0x3E Size: 1 byte (8 bits)

Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x00	**MIN_GNT (MINGNT)** Hardwired to 00h.

Maximum Latency (MAXLAT)

MAXIMUM LATENCY (MAXLAT)

Does not apply to PCI Express.

Base: Rootbus CFG Offset: 0x3F Size: 1 byte (8 bits)

Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:0	RO	8	0x00	**MAX_LAT (MAXLAT)** Hardwired to 00h.

PCI Express Capability List (PCIECAPLST)

PCI EXPRESS CAPABILITY LIST (PCIECAPLST)
Enumerates the PCI Express Capability Structure in the PCI Configuration list.
Base: Rootbus CFG Offset: 0x40 Size: 2 bytes (16 bits)
Default Value: 0x8010

Bits	Attr	Size	Default Val	Description
15:8	RO	8	0x80	**Next Capability Pointer (NXTCAP)** Offset to the next PCI Capability structure (MSI-X, in this case).
7:0	RO	8	0x10	**Capability ID (CAPID)** Indicates the PCI Express Capability structure.

PCI Express Capabilities (PCIECAP)

PCI EXPRESS CAPABILITIES (PCIECAP)
Identifies PCI Express device Function type and associated capabilities.
Base: Rootbus CFG Offset: 0x42 Size: 2 bytes (16 bits)
Default Value: 0x0092

Bits	Attr	Size	Default Val	Description
15:14	RSVD	2	0x0	Reserved.
13:9	RO	5	0x0	**Interrupt Message Number (INTMSGNUM)** Indicates which MSI-X vector is used for the interrupt message generated in association with any of the status bits in this Capability structure.
8	RO	1	0x0	**Slot Implemented (SLOTIMP)** No slot associated with this Function.
7:4	RO	4	1001b	**Device Type (DEVTYPE)** Indicates the specific type of this PCI Express Function. Root Complex Integrated Endpoint.
3:0	RO	4	0x2	**Capability Version (CAPVER)** Indicates the PCI-SIG defined PCI Express Capability structure version number.

Device Capabilities (DEVCAP)

DEVICE CAPABILITIES (DEVCAP)
Identifies PCI Express device Function specific capabilities.
Base: Rootbus CFG Offset: 0x44 Size: 4 bytes (32 bits)
Default Value: 0x10008022

Bits	Attr	Size	Default Val	Description
31:29	RSVD	3	0x0	Reserved.
28	RO	1	0x1	**Function Level Reset Capability (FLR)** Indicates support for the Function Level reset mechanism.
27:16	RSVD	12	0x000	Reserved.

DEVICE CAPABILITIES (DEVCAP)
Identifies PCI Express device Function specific capabilities.
Base: Rootbus CFG Offset: 0x44 Size: 4 bytes (32 bits)
Default Value: 0x10008022

Bits	Attr	Size	Default Val	Description
15	RO	1	0x1	**Role-Based Error Reporting (RBER)** Must be Set.
14:12	RSVD	3	0x0	Reserved.
11:9	RO	3	0x0	**Endpoint L1 Acceptable Latency (L1LAT)** Reserved.
8:6	RO	3	0x0	**Endpoint L0s Acceptable Latency (L0SLAT)** Reserved.
5	RO	1	0x1	**Extended Tag Field Supported (ETFS)** 8-bit tag field supported.
4:3	RO	2	0x0	**Phantom Functions Supported (PFS)** Phantom functions are not supported.
2:0	RO	3	Implementation defined	**Max Payload Size Supported (MPSS)** Indicates the maximum payload size that the Function can support for TLPs.

Device Control (DEVCTL)

DEVICE CONTROL (DEVCTL)
Controls PCI Express device specific parameters.
Base: Rootbus CFG Offset: 0x48 Size: 2 bytes (16 bits)
Default Value: 0x2910

Bits	Attr	Size	Default Val	Description
15	RW	1	0x0	**Initiate Function Level Reset (IFLR)** A write of 1b initiates Function Level Reset to the Function. The value read by software from this bit is always 0b.
14:12	RW	3	010b	**Max Read Request Size (MRRS)** This field sets the maximum Read Request size for the Function as a Requester.
11	RW	1	0x1	**Enable No Snoop (ENS)** If this bit is Set, the Function is permitted to Set the No Snoop bit in the Requester Attributes of transactions it initiates that do not require hardware enforced cache coherency.
10:9	RSVD	2	0x0	Reserved.
8	RW	1	0x1	**Extended Tag Field Enable (ETFE)** This bit, in combination with the 10-Bit Tag Requester Enable bit in the Device Control 2 register, determines how many Tag field bits a Requester is permitted to use.

DEVICE CONTROL (DEVCTL)
Controls PCI Express device specific parameters.
Base: Rootbus CFG Offset: 0x48 Size: 2 bytes (16 bits)
Default Value: 0x2910

Bits	Attr	Size	Default Val	Description
7:5	RW	3	Implementation defined	**Max Payload Size (MPS)** This field sets the maximum TLP payload size for the Function.
4	RW	1	0x1	**Enable Relaxed Ordering (ERO)** If this bit is Set, the Function is permitted to set the Relaxed Ordering bit in the Attributes field of transactions it initiates that do not require strong write ordering.
3	RW	1	0x0	**Unsupported Request Reporting Enable (URRE)** This bit, in conjunction with other bits, controls the signaling of unsupported Request Errors by sending error Messages.
2	RW	1	0x0	**Fatal Error Reporting Enable (FERE)** This bit, in conjunction with other bits, controls sending ERR_FATAL Messages.
1	RW	1	0x0	**Non-Fatal Error Reporting Enable (NERE)** This bit, in conjunction with other bits, controls sending ERR_NONFATAL Messages.
0	RW	1	0x0	**Correctable Error Reporting Enable (CERE)** This bit, in conjunction with other bits, controls sending ERR_COR Messages.

Device Status (DEVSTS)

DEVICE STATUS (DEVSTS)
Provides information about PCI Express device (Function) specific parameters.
Base: Rootbus CFG Offset: 0x4A Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:6	RSVD	10	0x000	Reserved.
5	RO	1	0x0	**Transactions Pending (TP)** When Set, this bit indicates that the Function has issued Non-Posted Requests that have not been completed. This bit is cleared only when all outstanding Non-Posted Requests have completed or have been terminated by the Completion Timeout mechanism. This bit will also be cleared upon the completion of an FLR.
4	RSVD	1	0x0	Reserved.

DEVICE STATUS (DEVSTS)
Provides information about PCI Express device (Function) specific parameters.
Base: Rootbus CFG Offset: 0x4A Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
3	RW1C	1	0x0	**Unsupported Request Detected (URD)** 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.
2	RW1C	1	0x0	**Fatal Error Detected (FED)** This bit indicates status of Fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register. Errors are logged in this register regardless of the settings of the AER Uncorrectable Error Mask register.
1	RW1C	1	0x0	**Non-fatal Error Detected (NED)** This bit indicates status of Non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register. Errors are logged in this register regardless of the settings of the AER Uncorrectable Error Mask register.
0	RW1C	1	0x0	**Correctable Error Detected (CED)** 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. Errors are logged in this register regardless of the settings of the AER Correctable Error Mask register.

Device Capabilities 2 (DEVCAP2)

DEVICE CAPABILITIES 2 (DEVCAP2)
Identifies additional PCI Express device Function specific capabilities.
Base: Rootbus CFG Offset: 0x64 Size: 4 bytes (32 bits)
Default Value: 0x10730810

Bits	Attr	Size	Default Val	Description
31	RSVD	1	0x0	Reserved.
30:29	RO	2	0x0	**DMWr Lengths Supported (DMWRLS)** Indicates the largest supported DMWr TLP.
28	RO	1	0x1	**DMWr Completer Supported (DMWRCS)** Indicates whether this function can serve as a DMWr Completer.
27:24	RSVD	4	0x0	Reserved.

DEVICE CAPABILITIES 2 (DEVCAP2)
Identifies additional PCI Express device Function specific capabilities.
Base: Rootbus CFG Offset: 0x64 Size: 4 bytes (32 bits)
Default Value: 0x10730810

Bits	Attr	Size	Default Val	Description
23:22	RO	2	0x1	**Max End-End TLP Prefixes (MEETLPP)** Indicates the maximum number of End-End TLP Prefixes supported by this Function.
21	RO	1	0x1	**End-End TLP Prefix Supported (EETLPPS)** Indicates Whether End-End TLP Prefix support is offered by a Function.
20	RO	1	0x1	**Extended Fmt Field Supported (EFFS)** If Set, the Function supports the 3-bit definition of the Fmt field. If Clear, the Function supports a 2-bit definition of the Fmt field.
19:18	RSVD	2	0x0	Reserved.
17	RO	1	0x1	**Ten-Bit Tag Requester Supported (TBTRS)** Indicates the Function supports 10-Bit Tag Requester capability.
16	RO	1	0x1	**Ten-Bit Tag Completer Supported (TBTCS)** Indicates the Function supports 10-Bit Tag Completer capability.
15:12	RSVD	4	0x0	Reserved.
11	RO	1	0x1	**LTR Mechanism Supported (LTRMS)** Indicates support for the Latency Tolerance Reporting (LTR) mechanism.
10:5	RSVD	6	0x00	Reserved.
4	RO	1	0x1	**Completion Timeout Disable Supported (CTDS)** Indicates support for the Completion Timeout Disable mechanism.
3:0	RO	4	0x0	**Completion Timeout Ranges Supported (CTRS)** Completion Timeout programming not supported. Function implements a timeout value in the range 50us to 50ms (closer to 20-40ms).

Device Control 2 (DEVCTL2)

DEVICE CONTROL 2 (DEVCTL2)
Controls additional PCI Express device specific parameters.
Base: Rootbus CFG Offset: 0x68 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:13	RSVD	3	0x0	Reserved.
12	RW	1	0x0	**Ten-Bit Tag Requester Enable (TBTRE)** When this bit is Set to 1b, the Requester is permitted to use 10-Bit tags.

DEVICE CONTROL 2 (DEVCTL2)
Controls additional PCI Express device specific parameters.
Base: Rootbus CFG Offset: 0x68 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
11	RSVD	1	0x0	Reserved.
10	RW	1	0x0	**LTR Mechanism Enable (LTRME)** When Set to 1b, this bit enables the Function to send LTR Messages.
9:5	RSVD	5	0x00	Reserved.
4	RW	1	0x0	**Completion Timeout Disable (CTD)** When Set, this bit disables the Completion Timeout mechanism.
3:0	RO	4	0x0	**Completion Timeout Value (CTV)** Completion Timeout Value programmability is not supported.

MSI-X Capability Header (MSIXCAPLST)

MSI-X CAPABILITY HEADER (MSIXCAPLST)
Enumerates the MSI-X Capability structure in the PCI Configuration Space Capability list.
Base: Rootbus CFG Offset: 0x80 Size: 2 bytes (16 bits)
Default Value: 0x9011

Bits	Attr	Size	Default Val	Description
15:8	RO	8	0x90	**Next Capability Pointer (NXTCAP)** Pointer to next capability (Power Management, in this case).
7:0	RO	8	0x11	**Capability ID (CAPID)** Indicates the MSI-X Capability structure.

MSI-X Message Control (MSIXMSGCTL)

MSI-X MESSAGE CONTROL (MSIXMSGCTL)
MSI-X controls. System SW can modify bits in this register. A device driver is not permitted to modify this register.
Base: Rootbus CFG Offset: 0x82 Size: 2 bytes (16 bits)
Default Value: 0x0008

Bits	Attr	Size	Default Val	Description
15	RW	1	0x0	**MSI-X Enable (MSIXEN)** If set, the Function is permitted to send MSI-X messages.
14	RW	1	0x0	**Function Mask (FCNMSK)** If set, all vectors associated with the Function are masked.
13:11	RSVD	3	0x0	Reserved.

MSI-X MESSAGE CONTROL (MSIXMSGCTL)
MSI-X controls. System SW can modify bits in this register. A device driver is not permitted to modify this register.
Base: Rootbus CFG Offset: 0x82 Size: 2 bytes (16 bits)
Default Value: 0x0008

Bits	Attr	Size	Default Val	Description
10:0	RO	11	0x008	**Table Size (TBLSZ)** MSI-X Table Size. Encoded as N-1 (N = 9 entries).

MSI-X Table (MSIXTBL)

MSI-X TABLE (MSIXTBL)
MSI-X Table Offset and Table BIR.
Base: Rootbus CFG Offset: 0x84 Size: 4 bytes (32 bits)
Default Value: 0x00002000

Bits	Attr	Size	Default Val	Description
31:3	RO	29	0x00000400	**Table Offset (OFFSET)** MSI-X Table Offset within BAR indicated by BIR. Entire register is used, masking BIR to form a 32-bit QWORD-aligned offset.
2:0	RO	3	0x0	**Table BIR (BIR)** Indicates the BAR used to map the MSI-X Table into Memory Space. BAR 0 at 10h.

MSI-X Pending Bit Array (MSIXPBA)

MSI-X PENDING BIT ARRAY (MSIXPBA)
MSI-X PBA Offset and PBA BIR.
Base: Rootbus CFG Offset: 0x88 Size: 4 bytes (32 bits)
Default Value: 0x00003000

Bits	Attr	Size	Default Val	Description
31:3	RO	29	0x00000600	**PBA Offset (OFFSET)** MSI-X PBA Offset within BAR indicated by BIR. Entire register is used, masking BIR to form a 32-bit QWORD-aligned offset.
2:0	RO	3	0x0	**PBA BIR (BIR)** Indicates the BAR used to map the MSI-X PBA into Memory Space. BAR 0 at 10h.

Power Management Capabilities (PMCAP)

POWER MANAGEMENT CAPABILITIES (PMCAP)
PCI Power Management Capability.
Base: Rootbus CFG Offset: 0x90 Size: 4 bytes (32 bits)
Default Value: 0x00030001

Bits	Attr	Size	Default Val	Description
31:27	RSVD	5	0x00	Reserved.
26	RO	1	0x0	**D2 Support (D2)** This Function does not support the D2 Power Management State.
25	RO	1	0x0	**D1 Support (D1)** This Function does not support the D1 Power Management State.
24:19	RSVD	6	0x00	Reserved.
18:16	RO	3	011b	**Version (VER)** Must be hardwired to 011b per PCIe spec.
15:8	RO	8	0x00	**Next Capability Pointer (NXTCAP)** Pointer to next capability (end of list, in this case).
7:0	RO	8	0x01	**Capability ID (CAPID)** Indicates PCI Power Management Capability.

Power Management Control/Status (PMCSR)

POWER MANAGEMENT CONTROL/STATUS (PMCSR)
This register is used to manage the PCI Function's power management state.
Base: Rootbus CFG Offset: 0x94 Size: 4 bytes (32 bits)
Default Value: 0x00000008

Bits	Attr	Size	Default Val	Description
31:4	RSVD	28	0x0000000	Reserved.
3	RO	1	0x1	**No Soft Reset (NSR)** This bit indicates the state of the Function after writing the Power State field to transition the Function from D3(hot) to D0.
2	RSVD	1	0x0	Reserved.
1:0	RW	2	0x0	**Power State (PS)** This field is used both to determine the current power state of a Function and to set the Function into a new power state. If an unsupported, optional state value is written, the data is discarded, and no state change occurs. 00b - D0, 01b - D1 (unsupported, 10b - D2 (unsupported), 11b - D3 (hot).

AER Extended Capability Header (AEREXTCAP)

AER EXTENDED CAPABILITY HEADER (AEREXTCAP)
Extended Capability Header.
Base: Rootbus CFG Offset: 0x100 Size: 4 bytes (32 bits)
Default Value: 0x15020001

Bits	Attr	Size	Default Val	Description
31:20	RO	12	0x150	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x2	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x0001	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

Uncorrectable Error Status (ERRUNCSTS)

UNCORRECTABLE ERROR STATUS (ERRUNCSTS)
Indicates error detection status of individual errors on a PCI Express device Function.
Base: Rootbus CFG Offset: 0x104 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:23	RSVD	9	0x000	Reserved.
22	RW1CS	1	0x0	**Uncorrectable Internal (UI)** Set when an uncorrectable error not covered by AER occurs (internal parity error).
21	RSVD	1	0x0	Reserved.
20	RW1CS	1	0x0	**Unsupported Request (UR)** Set when this function receives an Unsupported Request response.
19	RSVD	1	0x0	Reserved.
18	RW1CS	1	0x0	**Malformed TLP (MTLP)** Set when this function receives a Malformed TLP (MPS violation).
17	RSVD	1	0x0	Reserved.
16	RW1CS	1	0x0	**Unexpected Completion (UC)** Set when this function receives a completion that does not correspond to a Non-posted it issued.
15	RW1CS	1	0x0	**Completer Abort (CA)** Set when this function sends a completion with Completer Abort status.
14	RW1CS	1	0x0	**Completion Timeout (CTO)** Set when a Non-posted requested by this function is terminated via the Completion Timeout mechanism.
13	RSVD	1	0x0	Reserved.

UNCORRECTABLE ERROR STATUS (ERRUNCSTS)
Indicates error detection status of individual errors on a PCI Express device Function.
Base: Rootbus CFG Offset: 0x104 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
12	RW1CS	1	0x0	**Poisoned TLP Received (PTLP)** Set when a TLP received by this function is marked as poisoned.
11:0	RSVD	12	0x000	Reserved.

Uncorrectable Error Mask (ERRUNCMSK)

UNCORRECTABLE ERROR MASK (ERRUNCMSK)
Controls reporting of individual errors. A masked error is not recoded or reported in the Header Log or First Error Pointer and is not reported to the PCI Express Root Complex.
Base: Rootbus CFG Offset: 0x108 Size: 4 bytes (32 bits)
Default Value: 0x00400000

Bits	Attr	Size	Default Val	Description
31:23	RSVD	9	0x000	Reserved.
22	RWS	1	0x1	**Uncorrectable Internal (UI)** When Set, prevents the logging and reporting of Uncorrectable Internal errors.
21	RSVD	1	0x0	Reserved.
20	RWS	1	0x0	**Unsupported Request (UR)** When Set, prevents the logging and reporting of Unsupported Request errors.
19	RSVD	1	0x0	Reserved.
18	RWS	1	0x0	**Malformed TLP (MTLP)** When Set, prevents the logging and reporting of Malformed TLP errors.
17	RSVD	1	0x0	Reserved.
16	RWS	1	0x0	**Unexpected Completion (UC)** When Set, prevents the logging and reporting of Unexpected Completion errors.
15	RWS	1	0x0	**Completer Abort (CA)** When Set, prevents the logging and reporting of Completer Abort errors.
14	RWS	1	0x0	**Completion Timeout (CTO)** When Set, prevents the logging and reporting of Completion Timeout errors.
13	RSVD	1	0x0	Reserved.
12	RWS	1	0x0	**Poisoned TLP Received (PTLP)** When Set, prevents the logging and reporting of Poisoned TLP errors.
11:0	RSVD	12	0x000	Reserved.

Uncorrectable Error Severity (ERRUNCSEV)

UNCORRECTABLE ERROR SEVERITY (ERRUNCSEV)

Controls whether an individual error is reported as a Non-fatal (bit is Clear) or Fatal (bit is Set) error.

Base: Rootbus CFG Offset: 0x10C Size: 4 bytes (32 bits)

Default Value: 0x00440000

Bits	Attr	Size	Default Val	Description
31:23	RSVD	9	0x000	Reserved.
22	RWS	1	0x1	**Uncorrectable Internal (UI)** When Set, Uncorrectable Internal errors are Fatal.
21	RSVD	1	0x0	Reserved.
20	RWS	1	0x0	**Unsupported Request (UR)** When Set, Unsupported Request errors are Fatal.
19	RSVD	1	0x0	Reserved.
18	RWS	1	0x1	**Malformed TLP (MTLP)** When Set, Malformed TLP errors are Fatal.
17	RSVD	1	0x0	Reserved.
16	RWS	1	0x0	**Unexpected Completion (UC)** When Set, Unexpected Completion errors are Fatal.
15	RWS	1	0x0	**Completer Abort (CA)** When Set, Completer Abort errors are Fatal.
14	RWS	1	0x0	**Completion Timeout (CTO)** When Set, Completion Timeout errors are Fatal.
13	RSVD	1	0x0	Reserved.
12	RWS	1	0x0	**Poisoned TLP Received (PTLP)** When Set, Poisoned TLP errors are Fatal.
11:0	RSVD	12	0x000	Reserved.

Correctable Error Status (ERRCORSTS)

CORRECTABLE ERROR STATUS (ERRCORSTS)

Reports error status of individual correctable error sources.

Base: Rootbus CFG Offset: 0x110 Size: 4 bytes (32 bits)

Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:15	RSVD	17	0x00000	Reserved.
14	RW1CS	1	0x0	**Corrected Internal (CI)** Set when a Corrected Internal error is detected.
13	RW1CS	1	0x0	**Advisory Non-Fatal (ANF)** Set when an Error is classified as Advisory Non-fatal.
12:0	RSVD	13	0x0000	Reserved.

Correctable Error Mask (ERRCORMSK)

CORRECTABLE ERROR MASK (ERRCORMSK)
Controls the reporting of individual correctable errors.
Base: Rootbus CFG Offset: 0x114 Size: 4 bytes (32 bits)
Default Value: 0x00002000

Bits	Attr	Size	Default Val	Description
31:15	RSVD	17	0x00000	Reserved.
14	RWS	1	0x1	**Corrected Internal (CI)** When Set, Corrected Internal errors are not reported.
13	RWS	1	0x1	**Advisory Non-Fatal (ANF)** When Set, Advisory Non-Fatal errors are not reported.
12:0	RSVD	13	0x0000	Reserved.

AER Capabilities and Control (AERCAPCTL)

AER CAPABILITIES AND CONTROL (AERCAPCTL)
More AER information.
Base: Rootbus CFG Offset: 0x118 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:12	RSVD	20	0x00000	Reserved.
11	ROS	1	0x0	**TLP Prefix Log Present (TLPPLP)** If Set and the First Error Pointer is valid, indicates that the TLP Prefix Log register contains valid information.
10:5	RSVD	6	0x00	Reserved.
4:0	ROS	5	0x0	**First Error Pointer (FEP)** Identifies the bit position of the first error reported in the Uncorrectable Error Status register.

Header Log DW1 (AERHDRLOG1)

HEADER LOG DW1 (AERHDRLOG1)
First DWORD of the header for the TLP corresponding to a detected error.
Base: Rootbus CFG Offset: 0x11C Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**Header Log (HDRLOG)** Header Log DW.

Header Log DW2 (AERHDRLOG2)

HEADER LOG DW2 (AERHDRLOG2)
Second DWORD of the header for the TLP corresponding to a detected error.
Base: Rootbus CFG Offset: 0x120 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**Header Log (HDRLOG)** Header Log DW.

Header Log DW3 (AERHDRLOG3)

HEADER LOG DW3 (AERHDRLOG3)
Third DWORD of the header for the TLP corresponding to a detected error.
Base: Rootbus CFG Offset: 0x124 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**Header Log (HDRLOG)** Header Log DW.

Header Log DW4 (AERHDRLOG4)

HEADER LOG DW4 (AERHDRLOG4)
Fourth DWORD of the header for the TLP corresponding to a detected error.
Base: Rootbus CFG Offset: 0x128 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**Header Log (HDRLOG)** Header Log DW.

TLP Prefix Log Register DW1 (AERTLPPLOG1)

TLP PREFIX LOG REGISTER DW1 (AERTLPPLOG1)
This register captures the End-End TLP Prefix (DW1) for the TLP corresponding to the detected error.
Base: Rootbus CFG Offset: 0x138 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**TLP Prefix Log (TLPPLOG)** TLP Prefix Log DW.

TLP Prefix Log Register DW2 (AERTLPPLOG2)

TLP PREFIX LOG REGISTER DW2 (AERTLPPLOG2)
This register captures the End-End TLP Prefix (DW2) for the TLP corresponding to the detected error.
Base: Rootbus CFG Offset: 0x13C Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**TLP Prefix Log (TLPPLOG)** TLP Prefix Log DW.

TLP Prefix Log Register DW3 (AERTLPPLOG3)

TLP PREFIX LOG REGISTER DW3 (AERTLPPLOG3)
This register captures the End-End TLP Prefix (DW3) for the TLP corresponding to the detected error.
Base: Rootbus CFG Offset: 0x140 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**TLP Prefix Log (TLPPLOG)** TLP Prefix Log DW.

TLP Prefix Log Register DW4 (AERTLPPLOG4)

TLP PREFIX LOG REGISTER DW4 (AERTLPPLOG4)
This register captures the End-End TLP Prefix (DW4) for the TLP corresponding to the detected error.
Base: Rootbus CFG Offset: 0x144 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:0	ROS	32	0x0	**TLP Prefix Log (TLPPLOG)** TLP Prefix Log DW.

LTR Extended Capability Header (LTREXTCAP)

LTR EXTENDED CAPABILITY HEADER (LTREXTCAP)
Extended Capability Header.
Base: Rootbus CFG Offset: 0x150 Size: 4 bytes (32 bits)
Default Value: 0x16010018

Bits	Attr	Size	Default Val	Description
31:20	RO	12	0x160	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x1	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x0018	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

Max Snoop Latency (MAXSNPLAT)

MAX SNOOP LATENCY (MAXSNPLAT)
Maximum Snoop Latency the function is permitted to request.
Base: Rootbus CFG Offset: 0x154 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:13	RSVD	3	0x0	Reserved.
12:10	RW	3	0x0	**Latency Value (SCALE)** Scale of value sent in LTR message (scale = 2^{5N} ns).
9:0	RW	10	0x000	**Latency Value (VALUE)** Value sent in LTR message.

Max No-Snoop Latency (MAXNSNPLAT)

MAX NO-SNOOP LATENCY (MAXNSNPLAT)
Maximum No-Snoop Latency the function is permitted to request.
Base: Rootbus CFG Offset: 0x156 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:13	RSVD	3	0x0	Reserved.
12:10	RW	3	0x0	**Latency Value (SCALE)** Scale of value sent in LTR message (scale = 2^{5N} ns).
9:0	RW	10	0x000	**Latency Value (VALUE)** Value sent in LTR message.

TPH Extended Capability Header (TPHEXTCAP)

TPH EXTENDED CAPABILITY HEADER (TPHEXTCAP)
Extended Capability Header.
Base: Rootbus CFG Offset: 0x160 Size: 4 bytes (32 bits)
Default Value: 0x17010017

Bits	Attr	Size	Default Val	Description
31:20	RO	12	0x170	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x1	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x0017	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

TPH Capability (TPHCAP)

TPH CAPABILITY (TPHCAP)
TPH Capabilities.
Base: Rootbus CFG Offset: 0x164 Size: 4 bytes (32 bits)
Default Value: 0x00010205

Bits	Attr	Size	Default Val	Description
31:27	RSVD	5	0x00	Reserved.
26:16	RO	11	0x001	**ST Table Size (STTBLSIZE)** Value indicates the maximum number of ST Table entries the Function may use. Software reads this field to determine the ST Table Size N, which is encoded as N-1.
15:11	RSVD	5	0x00	Reserved.
10:9	RO	2	0x1	**ST Table Location (STTBLLOC)** Value indicates if and where the ST Table is located.
8	RO	1	0x0	**Extended TPH Requester Supported (EXTTPHSUPP)** If set, indicates that the Function is capable of generating Requests with a TPH TLP Prefix.
7:3	RSVD	5	0x00	Reserved.
2	RO	1	0x0	**Device Specific Mode Supported (DEVSPECSUPP)** If set, indicates that the Function supports the Device Specific Mode of operation.
1	RO	1	0x0	**Interrupt Vector Mode Supported (INTVECSUPP)** If set, indicates that the Function supports the Interrupt Vector Modes of operation.
0	RO	1	0x1	**No ST Mode Supported (NOSTSUPP)** If set, indicates that the Function supports the No ST Mode of operation.

TPH Requester Control Register (TPHCTL)

TPH REQUESTER CONTROL REGISTER (TPHCTL)
TPH Requester Capabilities.
Base: Rootbus CFG Offset: 0x168 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:10	RSVD	22	0x000000	Reserved.
9	RO	1	0x0	**TPH Requester Enable [9:9] (TPHREQEN_9_9)** Controls the ability to issue Request TLPs using Extended TPH.
8	RW	1	0x0	**TPH Requester Enable [8:8] (TPHREQEN_8_8)** Controls the ability to issue Request TLPs using TPH.
7:3	RSVD	5	0x00	Reserved.

TPH REQUESTER CONTROL REGISTER (TPHCTL)
TPH Requester Capabilities.
Base: Rootbus CFG Offset: 0x168 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
2:0	RW	3	0x0	**ST Mode Select (STMODESEL)** Selects the ST Mode of operation.

TPH ST Table (TPHSTTBL0)

TPH ST TABLE (TPHSTTBL0)
TPH ST Table.
Base: Rootbus CFG Offset: 0x16C Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:8	RO	8	0x00	**ST Upper Entry (STUE)** If the Function's Extended TPH Requester Supported bit is Set, then this field contains the upper 8 bits of a Steering Tag.
7:0	RW	8	0x00	**ST Lower Entry (STLE)** This field contains the lower 8 bits of a Steering Tag.

TPH ST Table (TPHSTTBL1)

TPH ST TABLE (TPHSTTBL1)
TPH ST Table.
Base: Rootbus CFG Offset: 0x16E Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:8	RO	8	0x00	**ST Upper Entry (STUE)** If the Function's Extended TPH Requester Supported bit is Set, then this field contains the upper 8 bits of a Steering Tag.
7:0	RW	8	0x00	**ST Lower Entry (STLE)** This field contains the lower 8 bits of a Steering Tag.

VC Extended Capability Header (VCEXTCAP)

VC EXTENDED CAPABILITY HEADER (VCEXTCAP)
Virtual Channel Extended Capability Header.
Base: Rootbus CFG Offset: 0x170 Size: 4 bytes (32 bits)
Default Value: 0x20010002

Bits	Attr	Size	Default Val	Description
31:20	RO	12	0x200	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x1	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x0002	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

Port VC Capability Register 1 (PORTVCCAP1)

PORT VC CAPABILITY REGISTER 1 (PORTVCCAP1)
Port VC Capability Register 1.
Base: Rootbus CFG Offset: 0x174 Size: 4 bytes (32 bits)
Default Value: 0x00000011

Bits	Attr	Size	Default Val	Description
31:12	RSVD	20	0x00000	Reserved.
11:10	RO	2	0x0	**Port Arbitration Table Entry Size (PATES)** Indicates the size of Port Arbitration table entry in the Function. Does not apply to this Endpoint IP.
9:8	RO	2	0x0	**Reference Clock (REFCLK)** Indicates the reference clock for Virtual Channels that support time-based WRR Port Arbitration. Does not apply to this Endpoint IP.
7	RSVD	1	0x0	Reserved.
6:4	RO	3	0x1	**Low Priority Extended VC Count (LPEXTVCCNT)** Indicates the number of (extended) Virtual Channels in addition to the default VC belonging to the low-priority VC (LPVC) group.
3	RSVD	1	0x0	Reserved.
2:0	RO	3	0x1	**Extended VC Count (EXTVCCNT)** Indicates the number of (extended) Virtual Channels in addition to the default VC supported by the device.

Port VC Capability Register 2 (PORTVCCAP2)

PORT VC CAPABILITY REGISTER 2 (PORTVCCAP2)
Port VC Capability Register 2.
Base: Rootbus CFG Offset: 0x178 Size: 4 bytes (32 bits)
Default Value: 0x00000001

Bits	Attr	Size	Default Val	Description
31:24	RO	8	0x00	**VC Arbitration Table Offset (VCARBTO)** Indicates the location of the VC Arbitration Table.
23:8	RSVD	16	0x0000	Reserved.
7:0	RO	8	0x01	**VC Arbitration Capability (VCARBCAP)** Indicates the types of VC Arbitration supported by the Function for the LPVC.

Port VC Control Register (PORTVCCTL)

PORT VC CONTROL REGISTER (PORTVCCTL)
Port VC Control Register.
Base: Rootbus CFG Offset: 0x17C Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:4	RSVD	12	0x000	Reserved.
3:1	RW	3	0x0	**VC Arbitration Select (VCARBSEL)** Used by software to configure the VC arbitration by selecting one of the supported VC Arbitration schemes indicated by the VC Arbitration schemes indicated by the VC Arbitration Capability field in the Port VC Capability register 2.
0	RO	1	0x0	**Load VC Arbitration Table (LDVCARBTBL)** Used by software to update the VC Arbitration Table. Does not apply to this IP.

Port VC Status Register (PORTVCSTS)

PORT VC STATUS REGISTER (PORTVCSTS)
Port VC Status Register.
Base: Rootbus CFG Offset: 0x17E Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:1	RSVD	15	0x0000	Reserved.
0	RO	1	0x0	**VC Arbitration Table Status (VCARBTBLSTS)** Indicates the coherency status of the VC Arbitration Table. Does not apply to this IP.

VC Resource Capability Register (VC0CAP)

VC RESOURCE CAPABILITY REGISTER (VC0CAP)
VC Resource Capability Register.
Base: Rootbus CFG Offset: 0x180 Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:24	RO	8	0x00	**Port Arbitration Table Offset (PATO)** Indicates the location of the Port Arbitration Table associated with the VC resource. Does not apply to this Endpoint IP.
23	RSVD	1	0x0	Reserved.
22:16	RO	7	0x00	**Maximum Time Slots (MAXTIMSLT)** Indicates the maximum number of time slots (minus one) that the VC resource is capable of supporting when it is configured for time-based WRR Port Arbitration. Does not apply to this Endpoint IP.
15	RO	1	0x0	**Reject Snoop Transactions (REJSNPTXN)** When Clear, transactions with or without the No Snoop bit Set within the TLP header are allowed on this VC. Does not apply to this Endpoint IP.
14	RO	1	0x0	**Undefined (UNDEF)** The value read from this bit is undefined.
13:8	RSVD	6	0x00	Reserved.
7:0	RO	8	0x00	**Port Arbitration Capability (PORTARBCAP)** Indicates types of Port Arbitration supported by the VC resource. Does not apply to this Endpoint IP.

VC 0 Resource Control Register (VC0CTL)

VC 0 RESOURCE CONTROL REGISTER (VC0CTL)
VC Resource Control Register.
Base: Rootbus CFG Offset: 0x184 Size: 4 bytes (32 bits)
Default Value: 0x800000FF

Bits	Attr	Size	Default Val	Description
31	RO	1	0x1	**VC Enable (VCEN)** This bit, when Set, enables a Virtual Channel.
30:27	RSVD	4	0x0	Reserved.
26:24	RO	3	0x0	**VC ID (VCID)** This field assigns a VC ID to the VC resource.
23:20	RSVD	4	0x0	Reserved.
19:17	RO	3	0x0	**Port Arbitration Select (PORTARBSEL)** This field configures the VC resource to provide a particular Port Arbitration service. Does not apply to this Endpoint IP.

VC 0 RESOURCE CONTROL REGISTER (VC0CTL)
VC Resource Control Register.
Base: Rootbus CFG Offset: 0x184 Size: 4 bytes (32 bits)
Default Value: 0x800000FF

Bits	Attr	Size	Default Val	Description
16	RO	1	0x0	**Load Port Arbitration Table (LDPORTARBTBL)** When Set, this bit updates the Port Arbitration logic from the Port Arbitration Table for the VC resource. Does not apply to this Endpoint IP.
15:8	RSVD	8	0x00	Reserved.
7:1	RW	7	0x7F	**TC/VC Map [7:1] (TCVCMAP_7_1)** This field indicates the TCs that are mapped to the VC resource.
0	RO	1	0x1	**TC/VC Map [0:0] (TCVCMAP_0_0)** This field indicates the TCs that are mapped to the VC resource.

VC Resource Status Register (VC0STS)

VC RESOURCE STATUS REGISTER (VC0STS)
VC Resource Status Register.
Base: Rootbus CFG Offset: 0x18A Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:2	RSVD	14	0x0000	Reserved.
1	RO	1	0x0	**VC Negotiation Pending (VCNEGPEND)** This bit indicates whether the Virtual Channel negotiation is in pending state. Does not apply to this non-Link IP.
0	RO	1	0x0	**Port Arbitration Table Status (PORTARBTBLSTS)** This bit indicates the coherency status of the Port Arbitration Table associated with the VC resource. Does not apply to this Endpoint IP.

VC Resource Capability Register (VC1CAP)

VC RESOURCE CAPABILITY REGISTER (VC1CAP)
VC Resource Capability Register.
Base: Rootbus CFG Offset: 0x18C Size: 4 bytes (32 bits)
Default Value: 0x00000000

Bits	Attr	Size	Default Val	Description
31:24	RO	8	0x00	**Port Arbitration Table Offset (PATO)** Indicates the location of the Port Arbitration Table associated with the VC resource. Does not apply to this Endpoint IP.
23	RSVD	1	0x0	Reserved.

VC RESOURCE CAPABILITY REGISTER (VC1CAP) VC Resource Capability Register. Base: Rootbus CFG Offset: 0x18C Size: 4 bytes (32 bits) Default Value: 0x00000000				
Bits	**Attr**	**Size**	**Default Val**	**Description**
22:16	RO	7	0x00	**Maximum Time Slots (MAXTIMSLT)** Indicates the maximum number of time slots (minus one) that the VC resource is capable of supporting when it is configured for time-based WRR Port Arbitration. Does not apply to this Endpoint IP.
15	RO	1	0x0	**Reject Snoop Transactions (REJSNPTXN)** When Clear, transactions with or without the No Snoop bit Set within the TLP header are allowed on this VC. Does not apply to this Endpoint IP.
14	RO	1	0x0	**Undefined (UNDEF)** The value read from this bit is undefined.
13:8	RSVD	6	0x00	Reserved.
7:0	RO	8	0x00	**Port Arbitration Capability (PORTARBCAP)** Indicates types of Port Arbitration supported by the VC resource. Does not apply to this Endpoint IP.

VC 1 Resource Control Register (VC1CTL)

VC 1 RESOURCE CONTROL REGISTER (VC1CTL) VC Resource Control Register. Base: Rootbus CFG Offset: 0x190 Size: 4 bytes (32 bits) Default Value: 0x01000000				
Bits	**Attr**	**Size**	**Default Val**	**Description**
31	RW	1	0x0	**VC Enable (VCEN)** This bit, when Set, enables a Virtual Channel.
30:27	RSVD	4	0x0	Reserved.
26:24	RW	3	0x1	**VC ID (VCID)** This field assigns a VC ID to the VC resource.
23:20	RSVD	4	0x0	Reserved.
19:17	RO	3	0x0	**Port Arbitration Select (PORTARBSEL)** This field configures the VC resource to provide a particular Port Arbitration service. Does not apply to this Endpoint IP.
16	RO	1	0x0	**Load Port Arbitration Table (LDPORTARBTBL)** When Set, this bit updates the Port Arbitration logic from the Port Arbitration Table for the VC resource. Does not apply to this Endpoint IP.
15:8	RSVD	8	0x00	Reserved.

VC 1 RESOURCE CONTROL REGISTER (VC1CTL)
VC Resource Control Register.
Base: Rootbus CFG Offset: 0x190 Size: 4 bytes (32 bits)
Default Value: 0x01000000

Bits	Attr	Size	Default Val	Description
7:1	RW	7	0x00	**TC/VC Map [7:1] (TCVCMAP_7_1)** This field indicates the TCs that are mapped to the VC resource.
0	RO	1	0x0	**TC/VC Map [0:0] (TCVCMAP_0_0)** This field indicates the TCs that are mapped to the VC resource.

VC Resource Status Register (VC1STS)

VC RESOURCE STATUS REGISTER (VC1STS)
VC Resource Status Register.
Base: Rootbus CFG Offset: 0x196 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:2	RSVD	14	0x0000	Reserved.
1	RO	1	0x0	**VC Negotiation Pending (VCNEGPEND)** This bit indicates whether the Virtual Channel negotiation is in pending state. Does not apply to this non-Link IP.
0	RO	1	0x0	**Port Arbitration Table Status (PORTARBTBLSTS)** This bit indicates the coherency status of the Port Arbitration Table associated with the VC resource. Does not apply to this Endpoint IP.

Scalable IOV Extended Capability Header (SIOVEXTCAP)

SCALABLE IOV EXTENDED CAPABILITY HEADER (SIOVEXTCAP)
Extended Capability Header.
Base: Rootbus CFG Offset: 0x200 Size: 4 bytes (32 bits)
Default Value: 0x22010023

Bits	Attr	Size	Default Val	Description
31:20	RO	12	0x220	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x1	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x0023	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

Scalable IOV Designated Vendor-Specific Header 1 (SIOVDVSEC1)

SCALABLE IOV DESIGNATED VENDOR-SPECIFIC HEADER 1 (SIOVDVSEC1) Designated Vendor-Specific Header 1. Base: Rootbus CFG Offset: 0x204 Size: 4 bytes (32 bits) Default Value: 0x01808086				
Bits	**Attr**	**Size**	**Default Val**	**Description**
31:20	RO	12	0x018	**DVSEC Length (LENGTH)** This field indicates the number of bytes in the entire DVSEC structure, including the PCI Express Extended Capability header, the DVSEC Header 1, DVSEC Header 2, and DVSEC vendor-specific registers.
19:16	RO	4	0x0	**DVSEC Revision (REV)** This field is a vendor-defined version number that indicates the version of the DVSEC structure.
15:0	RO	16	0x8086	**DVSEC Vendor ID (VID)** This field is the Vendor ID associated with the vendor that defined the contents of this capability.

Scalable IOV Designated Vendor-Specific Header 2 (SIOVDVSEC2)

SCALABLE IOV DESIGNATED VENDOR-SPECIFIC HEADER 2 (SIOVDVSEC2) Designated Vendor-Specific Header 2. Base: Rootbus CFG Offset: 0x208 Size: 2 bytes (16 bits) Default Value: 0x0005				
Bits	**Attr**	**Size**	**Default Val**	**Description**
15:0	RO	16	0x0005	**DVSEC ID (DVSECID)** This field is a vendor-defined ID that indicates the nature and format of the DVSEC structure.

Scalable IOV Function Dependency Link (SIOVFDL)

SCALABLE IOV FUNCTION DEPENDENCY LINK (SIOVFDL) See Scalable IOV Arch Spec. Base: Rootbus CFG Offset: 0x20A Size: 1 byte (8 bits) Default Value: 0x00				
Bits	**Attr**	**Size**	**Default Val**	**Description**
7:0	RO	8	0x00	**Function Dependency Link (FDL)** Indicates dependencies between functions of a multi-function device.

Scalable IOV Flags (SIOVFLAGS)

SCALABLE IOV FLAGS (SIOVFLAGS)
See Scalable IOV Arch Spec.
Base: Rootbus CFG Offset: 0x20B Size: 1 byte (8 bits)
Default Value: 0x00

Bits	Attr	Size	Default Val	Description
7:1	RSVD	7	0x00	Reserved.
0	RO	1	0x0	**Homogeneous (H)** When Set, indicates that if any function of a multi-function device is enabled for scalable IOV operation, all functions of the device must be so enabled.

SIOV Supported Page Sizes (SIOVSUPPGSZ)

SIOV SUPPORTED PAGE SIZES (SIOVSUPPGSZ)
See Scalable IOV Arch Spec.
Base: Rootbus CFG Offset: 0x20C Size: 4 bytes (32 bits)
Default Value: 0x00000001

Bits	Attr	Size	Default Val	Description
31:0	RO	32	0x00000001	**Bits (BITS)** Indicates the supported system page size is 4KB.

SIOV System Page Size (SIOVSYSPGSZ)

SIOV SYSTEM PAGE SIZE (SIOVSYSPGSZ)
See Scalable IOV Arch Spec.
Base: Rootbus CFG Offset: 0x210 Size: 4 bytes (32 bits)
Default Value: 0x00000001

Bits	Attr	Size	Default Val	Description
31:0	RW	32	0x00000001	**Bits (BITS)** Indicates selected system page size is 4KB.

SIOV Capabilities (SIOVCAP)

SIOV CAPABILITIES (SIOVCAP)
See Scalable IOV Arch Spec.
Base: Rootbus CFG Offset: 0x214 Size: 4 bytes (32 bits)
Default Value: 0x00000001

Bits	Attr	Size	Default Val	Description
31:1	RSVD	31	0x0000	Reserved.
0	RO	1	0x1	**IMS Support (IMSS)** Indicates support for device-specific Interrupt Message Storage.

ATS Extended Capability Header (ATSEXTCAP)

ATS EXTENDED CAPABILITY HEADER (ATSEXTCAP)
Extended Capability Header.
Base: Rootbus CFG Offset: 0x220 Size: 4 bytes (32 bits)
Default Value: 0x2301000F

Bits	Attr	Size	Default Val	Description
31:20	RO	12	0x230	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x1	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x000F	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

ATS Capability (ATSCAP)

ATS CAPABILITY (ATSCAP)
ATS Capabilities.
Base: Rootbus CFG Offset: 0x224 Size: 2 bytes (16 bits)
Default Value: 0x0060

Bits	Attr	Size	Default Val	Description
15:7	RSVD	9	0x000	Reserved.
6	RO	1	0x1	**Global Invalidate Supported (GIS)** If Set, the Function supports Invalidation Requests that have the Global Invalidate bit Set.
5	RO	1	0x1	**Page Aligned Request (PAR)** When Set, indicates the Untranslated Address is always aligned to a 4096-byte boundary.
4:0	RO	5	0x00	**Invalidate Queue Depth (IQD)** Number of Invalidate Requests the Function can accept before back pressuring (00000b = 32).

ATS Control (ATSCTL)

ATS CONTROL (ATSCTL)
ATS Controls.
Base: Rootbus CFG Offset: 0x226 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15	RW	1	0x0	**Enable (EN)** When Set, function is enabled to cache translations.
14:5	RSVD	10	0x000	Reserved.

ATS CONTROL (ATSCTL) ATS Controls. Base: Rootbus CFG Offset: 0x226 Size: 2 bytes (16 bits) Default Value: 0x0000				
Bits	**Attr**	**Size**	**Default Val**	**Description**
4:0	RW	5	0x0	**Smallest Translation Unit (STU)** Minimum number of 4096-byte blocks that are indicated in a Translation Completion or Invalidate Request. Number of blocks = 2 ^ STU.

PASID Extended Capability Header (PASIDEXTCAP)

PASID EXTENDED CAPABILITY HEADER (PASIDEXTCAP) Extended Capability Header. Base: Rootbus CFG Offset: 0x230 Size: 4 bytes (32 bits) Default Value: 0x2401001B				
Bits	**Attr**	**Size**	**Default Val**	**Description**
31:20	RO	12	0x240	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x1	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x001B	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

PASID Capability (PASIDCAP)

PASID CAPABILITY (PASIDCAP) PASID-related capabilities. Base: Rootbus CFG Offset: 0x234 Size: 2 bytes (16 bits) Default Value: 0x1404				
Bits	**Attr**	**Size**	**Default Val**	**Description**
15:13	RSVD	3	0x0	Reserved.
12:8	RO	5	0x14	**Max PASID Width (MAXWID)** PASID width supported by the function.
7:3	RSVD	5	0x00	Reserved.
2	RO	1	0x1	**Privileged Mode Supported (PMS)** If Set, function supports sending requests with the Privileged Mode Requested bit Set.
1	RO	1	0x0	**Execute Permission Supported (EPS)** If Set, function supports sending TLPs that have the Execute Requested bit Set.
0	RSVD	1	0x0	Reserved.

PASID Control (PASIDCTL)

PASID CONTROL (PASIDCTL)
Controls for PASID-related functionality.
Base: Rootbus CFG Offset: 0x236 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:3	RSVD	13	0x0000	Reserved.
2	RW	1	0x0	**Privileged Mode Enable (PME)** If Set, function is permitted to send Requests with the Privileged Mode Requested bit Set.
1	RO	1	0x0	**Execute Permission Enable (EPE)** If Set, function is permitted to send Requests with the Execute Requested bit Set.
0	RW	1	0x0	**PASID Enable (PE)** Function is permitted to send and receive TLPs that contain the PASID TLP prefix.

Page Request Extended Capability Header (PRSEXTCAP)

PAGE REQUEST EXTENDED CAPABILITY HEADER (PRSEXTCAP)
Extended Capability Header.
Base: Rootbus CFG Offset: 0x240 Size: 4 bytes (32 bits)
Default Value: 0x00010013

Bits	Attr	Size	Default Val	Description
31:20	RO	12	0x000	**Next Capability Offset (NXTCAP)** Offset to the next PCI Express Capability structure.
19:16	RO	4	0x1	**Capability Version (CAPVER)** PCI-SIG defined version number indicating the version of the Capability structure.
15:0	RO	16	0x0013	**Extended Capability ID (EXTCAPID)** PCI-SIG defined ID number indicating the nature and format of the Extended Capability.

Page Request Control (PRSCTL)

PAGE REQUEST CONTROL (PRSCTL)
Controls for Page Request activities.
Base: Rootbus CFG Offset: 0x244 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
15:2	RSVD	14	0x0000	Reserved.
1	RW	1	0x0	**Reset (RST)** When written to 1b, clears Page Request credit counter and pending request state when Enable bit is cleared or being cleared.

PAGE REQUEST CONTROL (PRSCTL)
Controls for Page Request activities.
Base: Rootbus CFG Offset: 0x244 Size: 2 bytes (16 bits)
Default Value: 0x0000

Bits	Attr	Size	Default Val	Description
0	RW	1	0x0	**Enable (EN)** When Set, function is allowed to make Page Requests.

Page Request Status (PRSSTS)

PAGE REQUEST STATUS (PRSSTS)
Status of Page Requests.
Base: Rootbus CFG Offset: 0x246 Size: 2 bytes (16 bits)
Default Value: 0x8100

Bits	Attr	Size	Default Val	Description
15	RO	1	0x1	**PRG Response PASID Required (PRPR)** If Set, function expects a PASID on PRG Response Messages when corresponding Page Requests had a PASID.
14:9	RSVD	6	0x00	Reserved.
8	RO	1	0x1	**Stopped (STOP)** When Enable is Clear, indicates whether previously issued Page Requests have completed.
7:2	RSVD	6	0x00	Reserved.
1	RW1C	1	0x0	**Unexpected Page Request Group Index (UPRGI)** When Set, indicates the function has received a PRG Response Message containing a PRG index with no matching request.
0	RW1C	1	0x0	**Response Failure (RF)** When Set, indicates the function has received a PRG Response Message indicating a Response Failure.

Outstanding Page Request Capacity (PRSREQCAP)

OUTSTANDING PAGE REQUEST CAPACITY (PRSREQCAP)
Maximum Number of Outstanding Page Requests.
Base: Rootbus CFG Offset: 0x248 Size: 4 bytes (32 bits)
Default Value: 0x00000200

Bits	Attr	Size	Default Val	Description
31:0	RO	32	0x200	**Capacity (CAP)** How many Page Requests can the function issue.

Outstanding Page Request Allocation (PRSREQALLOC)

OUTSTANDING PAGE REQUEST ALLOCATION (PRSREQALLOC) Maximum Number of Outstanding Page Requests Allowed. Base: Rootbus CFG Offset: 0x24C Size: 4 bytes (32 bits) Default Value: 0x00000000				
Bits	**Attr**	**Size**	**Default Val**	**Description**
31:0	RW	32	0x0	**Enable (ALLOC)** How many Page Requests will system SW allow.

§

Appendix D Performance Monitoring Events

D.1 Architectural Performance Monitoring Events

A set of architecturally defined performance monitoring events is common across different Intel DSA implementations. Additional events may be added in future implementations.

The Intel DSA architecture defines the following performance monitoring events.

D.2.1 Version 1

Event Category 0: Work Queue (WQ)

Event Name	Event Encoding	Description	Supported Filters
EV_SWQ_SUCCESS_LIMPORTAL	0x1	Number of successful DMWr transactions submitted to limited portal.	WQ, PASID
EV_SWQ_RETRY_LIMPORTAL	0x2	Number of retries returned for DMWr transactions to limited portal.	WQ, PASID
EV_SWQ_SUCCESS_UNLIMPORTAL	0x4	Number of successful DMWr transactions submitted to unlimited portal.	WQ, PASID
EV_SWQ_RETRY_UNLIMPORTAL	0x8	Number of retries returned for DMWr transactions to unlimited portal.	WQ, PASID
EV_DWQ_SUCCESS	0x10	Number of successful posted writes to DWQ.	WQ, PASID
EV_DWQ_FULL	0x20	Number of posted writes to DWQ dropped because queue is full.	WQ, PASID

Event Category 1: Engine

Event Name	Event Encoding	Description	Supported Filters
EV_CL_PROCESSED	0x1	Total input data processed, in units of 32 bytes.	TC, Transfer size, Engine Number
EV_CL_WRITE	0x2	Total data written, in units of 32 bytes.	TC, Transfer size, Engine Number
EV_NUM_READ	0x4	Number of descriptors that read Source 1.	TC, Transfer size, Engine Number
EV_NUM_WRITE	0x8	Number of descriptors that write Destination 1.	TC, Transfer size, Engine Number
EV_NUM_DESC_FROM_BATCH	0x10	Number of work descriptors dispatched from batch descriptors.	WQ, Engine Number, PASID
EV_NUM_DESC_DISPATCH_WQ	0x20	Number of descriptors dispatched from WQs.	WQ, Engine Number, PASID

Event Category 2: Address Translation

Event Name	Event Encoding	Description	Supported Filters
EV_ATS_RSP_PASID_NO_PF	0x1	Number of Successful Translation completions with PASID and without page fault.	Page Size, Engine Number, PASID
EV_ATS_RSP_PASID_PF	0x2	Number of Successful Translation completions with PASID and with page fault.	Page size, Engine Number, PASID

Event Name	Event Encoding	Description	Supported Filters
EV_ATS_RSP_NO_PASID_NO_PF	0x4	Number of Successful Translation completions without PASID and without page fault.	Page Size, Engine Number, PASID
EV_ATS_RSP_NO_PASID_PF	0x8	Number of Successful Translation completions without PASID and with page fault.	Page size, Engine Number, PASID
EV_PRS_RSP_SUCCESS	0x10	Number of PRS Responses with Success.	PASID
EV_PRS_RSP_INVALID	0x20	Number of PRS Responses with Invalid Request.	PASID

Event Category 3: Operations

Event Name	Event Encoding	Description	Supported Filters
EV_DESC_NOOP	0x1	Number of No-op descriptors.	WQ, PASID
EV_DESC_BATCH	0x2	Number of Batch descriptors.	WQ, PASID
EV_DESC_DRAIN	0x4	Number of Drain descriptors.	WQ, PASID
EV_MEM_MOVE	0x8	Number of Memory Move descriptors.	WQ, PASID
EV_FILL	0x10	Number of Fill descriptors.	WQ, PASID
EV_COMPARE_MEM	0x20	Number of Compare descriptors.	WQ, PASID
EV_COMPARE_PAT	0x40	Number of Compare Pattern descriptors.	WQ, PASID
EV_CREATE_DELTA	0x80	Number of Create Delta Record descriptors.	WQ, PASID
EV_APPLY_DELTA	0x100	Number of Apply Delta Record descriptors.	WQ, PASID

Event Name	Event Encoding	Description	Supported Filters
EV_DUALCAST	0x200	Number of Memory Copy with Dualcast descriptors.	WQ, PASID
EV_CRC_GEN	0x400	Number of CRC Generation descriptors.	WQ, PASID
EV_COPY_CRC	0x800	Number of Copy with CRC Generation descriptors.	WQ, PASID
EV_DIF_CHK	0x1000	Number of DIF Check descriptors.	WQ, PASID
EV_DIF_INS	0x2000	Number of DIF Insert descriptors.	WQ, PASID
EV_DIF_STRIP	0x4000	Number of DIF Strip descriptors.	WQ, PASID
EV_DIF_UPD	0x8000	Number of DIF Update descriptors.	WQ, PASID
EV_CLFLUSH	0x10000	Number of Cache Flush descriptors.	WQ, PASID

Event Category 4: Completions

Event Name	Event Encoding	Description	Supported Filters
EV_NUM_MSIX	0x1	Number of MSI-X interrupts generated.	WQ, PASID
EV_NUM_IMS	0x2	Number of IMS interrupts generated.	WQ, PASID
EV_CPL_PARTIAL	0x4	Number of descriptors with partial completion.	WQ, PASID
EV_CPL_ERR	0x8	Number of descriptors with error completion.	WQ, PASID
EV_NUM_CPL_SUCC	0x10	Number of successful completions.	WQ, PASID
EV_NUM_CPL_WRITES	0x20	Number of completion writes.	WQ, PASID

D.2.2 Version 2

This section lists the additions to architecturally defined performance monitoring events in implementations where the Major version field in the VERSION register (described in Section 9.2.1) is 2.

Event Category 0: Work Queue (WQ)

Event Name	Event Encoding	Description	Supported Filters
EV_WQ_DISCARD	0x40	Number of writes to a WQ that are discarded because one or more of the descriptor submission checks fail.	WQ

Event Category 1: Engine

Event Name	Event Encoding	Description	Supported Filters
EV_NUM_IDPT_BITMAP_RD_PERM	0x40	Number of reads of IDPT bitmap where the submitter access check was successful.	Engine Number, PASID
EV_NUM_IDPT_BITMAP_RD_NOPERM	0x80	Number of reads of IDPT bitmap where the submitter access check was not successful.	Engine Number, PASID
EV_CL_READ	0x100	Total data read, in units of 32 bytes.	TC, Transfer size, Engine Number
EV_NUM_READ2	0x200	Number of descriptors that read Source 2.	TC, Transfer size, Engine Number
EV_NUM_WRITE2	0x400	Number of descriptors that write Destination 2.	TC, Transfer size, Engine Number

Event Category 3: Operations

Event Name	Event Encoding	Description	Supported Filters
EV_TRANS_FETCH	0x20000	Number of Translation Fetch descriptors.	WQ, PASID
EV_DIX_GENERATE	0x40000	Number of DIX Generate descriptors.	WQ, PASID

Event Category 5: Inter-Domain Operations

Event Name	Event Encoding	Description	Supported Filters
EV_INTER_DOMAIN_COPY	0x1	Number of Inter-Domain Copy descriptors.	WQ, Transfer size, PASID
EV_INTER_DOMAIN_FILL	0x2	Number of Inter-Domain Fill descriptors.	WQ, Transfer size, PASID
EV_INTER_DOMAIN_COMPARE_MEM	0x4	Number of Inter-Domain Compare descriptors.	WQ, Transfer size, PASID
EV_INTER_DOMAIN_COMPARE_PAT	0x8	Number of Inter-Domain Compare Pattern descriptors.	WQ, Transfer size, PASID
EV_INTER_DOMAIN_CLFLUSH	0x10	Number of Inter-Domain Cache Flush descriptors.	WQ, Transfer size, PASID
EV_UPD_WINDOW	0x400	Number of Update Window descriptors.	WQ, Transfer size, PASID

D.2 Model-Specific Performance Monitoring Events

Model-specific performance monitoring events may be supported in addition to the architectural events defined above. These events are subject to change and may or may not be supported across different implementations of Intel DSA.

D.2.3 Version 1

The following model-specific events are supported in implementations where the Major version field in the VERSION register (described in Section 9.2.1) is 1.

Event Category 0: Work Queue (WQ)

Event Name	Event Encoding	Description	Supported Filters
EV_CYC_NON_BATCH_DESC_RDY	0x40	Number of cycles when non-batch descriptor ready.	WQ
EV_CYC_BATCH_DESC_RDY	0x80	Number of cycles when batch descriptor ready.	WQ

Event Name	Event Encoding	Description	Supported Filters
EV_CYC_DESC_NOT_RDY	0x100	Number of cycles when descriptor not ready.	WQ

Event Category 1: Engine

Event Name	Event Encoding	Description	Supported Filters
EV_PIPEFULL_NO_DISPATCH	0x40	Number of cycles when engine unable to dispatch descriptor to work pipeline because pipeline full.	Engine Number
EV_STALL_NO_DESC_RDY	0x80	Number of cycles when no descriptors ready to dispatch to work pipeline.	Engine Number
EV_STALL_BATCH_FETCH_FULL	0x100	Number of cycles when batch fetch-queue is full.	Engine Number
EV_STALL_BATCH_EXEC_FULL	0x200	Number of cycles when batch exec-queue is full.	Engine Number

Event Category 2: Address Translation

Event Name	Event Encoding	Description	Supported Filters
EV_ATC_ALLOC	0x40	Number of Translation requests to ATC.	Engine Number
EV_ATC_NO_ALLOC	0x80	Number of times a translation request is unable to allocate an ATC entry.	Engine Number
EV_ATC_HIT_PREV	0x100	Number of times a translation request matches a valid ATC entry.	Engine Number
EV_CYC_INV_RSP	0x200	Number of cycles to respond to all the entries in the invalidation queue (i.e., number of cycles when invalidation queue is not empty).	None

Event Name	Event Encoding	Description	Supported Filters
EV_ATS_RSP_DROP	0x400	Number of Translation Completions discarded.	Page size, Engine Number
EV_CYC_ATC_IDLE	0x800	Number of cycles when ATC is idle (no new requests, no outstanding ATS, etc.).	None
EV_INV_PASID_Q_EMPTY	0x8000	Number of times an invalidation request with PASID is received when the invalidation queue is empty.	None
EV_INV_PASID_Q_NOT_EMPTY	0x10000	Number of times an invalidation request with PASID is received when invalidation queue is not empty.	None
EV_INV_NO_PASID_Q_EMPTY	0x20000	Number of times an invalidation request without PASID is received when the invalidation queue is empty.	None
EV_INV_NO_PASID_Q_NOT_EMPTY	0x40000	Number of times an invalidation request without PASID is received when invalidation queue is not empty.	None
EV_INV_Q_FULL	0x80000	Number of times an invalidation request received caused the invalidation queue to become full.	None

Event Category 3: Operations

Event Name	Event Encoding	Description	Supported Filters
EV_FENCE_NO_DROP	0x20000	Number of fence operations not abandoned.	WQ
EV_FENCE_DROP	0x40000	Number of fence operations abandoned.	WQ
EV_OVERLAP_MOV	0x80000	Number of Memory move descriptors with src-dest overlap.	WQ

Event Category 4: Completions

Event Name	Event Encoding	Description	Supported Filters
EV_NUM_IMPLICIT_READBACKS	0x40	Number of implicit readbacks issued.	TC

D.2.4 Version 2

The following model-specific events are supported in implementations where the Major version field in the VERSION register (described in Section 9.2.1) is 2.

Event Category 0: Work Queue (WQ)

Event Name	Event Encoding	Description	Supported Filters
EV_CYC_NON_BATCH_DESC_RDY	0x8000000	Number of cycles when non-batch descriptor ready.	WQ, PASID
EV_CYC_BATCH_DESC_RDY	0x4000000	Number of cycles when batch descriptor ready.	WQ, PASID
EV_CYC_DESC_NOT_RDY	0x2000000	Number of cycles when descriptor not ready.	WQ, PASID

Event Category 1: Engine

Event Name	Event Encoding	Description	Supported Filters
EV_PIPEFULL_NO_DISPATCH	0x8000000	Number of cycles when engine unable to dispatch descriptor to work pipeline because pipeline full.	Engine Number
EV_STALL_NO_DESC_RDY	0x4000000	Number of cycles when no descriptors ready to dispatch to work pipeline.	Engine Number
EV_STALL_BATCH_FETCH_FULL	0x2000000	Number of cycles when batch fetch-queue is full.	Engine Number
EV_STALL_BATCH_EXEC_FULL	0x1000000	Number of cycles when batch exec-queue is full.	Engine Number

Event Name	Event Encoding	Description	Supported Filters
EV_STALL_NOAT	0x800000	Number of cycles when engine stall due to pending Address Translation.	Engine Number

Event Category 2: Address Translation

Event Name	Event Encoding	Description	Supported Filters
EV_ATC_ALLOC	0x8000000	Number of Translation requests to ATC.	Engine Number, PASID
EV_ATC_NO_ALLOC	0x4000000	Number of times a translation request is unable to allocate an ATC entry.	Engine Number, PASID
EV_ATC_HIT_PREV	0x2000000	Number of times a translation request matches a valid ATC entry.	Engine Number, PASID
EV_CYC_INV_RSP	0x1000000	Number of cycles to respond to all the entries in the invalidation queue (i.e., number of cycles when invalidation queue is not empty).	None
EV_ATS_RSP_DROP	0x800000	Number of Translation Completions discarded.	Page size, Engine Number, PASID
EV_CYC_ATC_IDLE	0x400000	Number of cycles when ATC is idle (no new requests, no outstanding ATS, etc.).	None
EV_INV_PASID_Q_EMPTY	0x200000	Number of times an invalidation request with PASID is received when the invalidation queue is empty.	PASID
EV_INV_PASID_Q_NOT_EMPTY	0x100000	Number of times an invalidation request with PASID is received when invalidation queue is not empty.	PASID

Event Name	Event Encoding	Description	Supported Filters
EV_INV_NO_PASID_Q_EMPTY	0x80000	Number of times an invalidation request without PASID is received when the invalidation queue is empty.	None
EV_INV_NO_PASID_Q_NOT_EMPTY	0x40000	Number of times an invalidation request without PASID is received when invalidation queue is not empty.	None
EV_INV_Q_FULL	0x20000	Number of times an invalidation request received caused the invalidation queue to become full.	PASID
EV_PRS_NO_ALLOC	0x10000	Number of times unable to issue PRS request because of lack of credits (outstanding PRS==PRSREQALLOC).	PASID

Event Category 3: Operations

Event Name	Event Encoding	Description	Supported Filters
EV_FENCE_NO_DROP	0x8000000	Number of fence operations not abandoned.	WQ, PASID
EV_FENCE_DROP	0x4000000	Number of fence operations abandoned.	WQ, PASID
EV_OVERLAP_MOV	0x2000000	Number of Memory move descriptors with src-dest overlap.	WQ, PASID

Event Category 4: Completions

Event Name	Event Encoding	Description	Supported Filters
EV_NUM_IMPLICIT_READBACKS	0x8000000	Number of implicit readbacks issued.	TC
EV_NUM_EXPLICIT_READBACKS	0x4000000	Number of explicit readbacks issued.	TC, PASID

D.3 Event Configuration Examples

Some event monitoring examples are shown below.

- To count the total number of attempted or successful descriptor submissions using DMWr, software can use a single counter to aggregate counts of the following events in the WQ category:
    - EV_SWQ_SUCCESS_LIMPORTAL - Number of successful DMWr transactions submitted to limited portal.
    - EV_SWQ_RETRY_LIMPORTAL - Number of retries returned for DMWr transactions to limited portal.
    - EV_SWQ_SUCCESS_UNLIMPORTAL - Number of successful DMWr transactions submitted to unlimited portal.
    - EV_SWQ_RETRY_UNLIMPORTAL - Number of retries returned for DMWr transactions to unlimited portal.
    - Set CNTRCFG_0 to 0xF_00000003 (Enable=1, Interrupt on Overflow=1, Event Category=WQ, Events field set to monitor the events listed above).
    - All filters for counter 0 set to default value of 0xFFFF (no constraints).
- To count the number of descriptors writing memory on TC 1, from engine 1 or 2, with transfer size 4KB or higher, software can use the following event in the Engine category:
    - EV_NUM_WRITE - Number of writes issued.
    - Set FLTCFG_TC_1 to 0x2 (TC 1).
    - Set FLTCFG_SZ_1 to 0xF8 (any transfer size ≥ 4KB).
    - Set FLTCFG_ENG_1 to 0x6 (Engine 1 or 2).
    - Set CNTRCFG_1 to 0x8_00000103 (Enable=1, Interrupt on Overflow=1, Event Category=Engine, Events field set to monitor the event listed above).
    - Other filters for counter 1 set to default value of 0xFFFF (no constraints).
- To count the number of DIF operations submitted to WQ 1 or WQ 2, software can use events in the Operations event category:
    - EV_DIF_CHK – Number of DIF Check descriptors.
    - EV_DIF_INS – Number of DIF Insert descriptors.
    - EV_DIF_STRIP – Number of DIF Strip descriptors.
    - EV_DIF_UPD – Number of DIF Update descriptors.
    - Set FLTCFG_WQ_2 to 0x6 (WQ 1 or 2).
    - Set CNTRCFG_2 to 0xF000_00000303 (Enable=1, Interrupt on Overflow=1, Event Category=Operations, Events field set to monitor DIF operations).
    - Other filters for counter 2 set to default value of 0xFFFF (no constraints).
- To estimate the frequency of occurrence of an event, software needs to use 2 distinct counters. For example, to estimate frequency (expressed as a percentage) of ATC full condition, software can program counter 0 to count EV_ATC_ALLOC events and counter 1 to count EV_ATC_NO_ALLOC events. Software then computes the ratio to estimate the frequency of occurrence of the desired condition.

§

Appendix E Summary of Key Architecture Extensions

This section lists the key architecture extensions introduced in Revision 2.0 of the Intel DSA specification. Software should consult the capability registers in the device (as described in Section 9.2) to identify the presence of specific features in an implementation.

Inter-Domain Operations	New set of Inter-Domain operations that can operate on multiple address spaces (identified by PASID) with a single descriptor. New Inter-Domain Permissions Table to facilitate connections between different address spaces. Use with host OS/VMM, guest OS, and host or guest applications. (Refer to section 3.14 for details.)
64-bit CRC	Extend CRC operations to support 64-bit CRC (XP10 polynomial); Intel DSA 1.0 limited to CRC16/32. (Refer to section 8.3.12 and Appendix A.)
16-byte Fill	Extend Fill operation to support larger 16-byte pattern size; Intel DSA 1.0 limited to 8-byte pattern size. (Refer to section 8.3.5.)
Event Log	Support for a software-configurable event log in memory to report various types of software error events. (Refer to section 5.9.)
Changes in completion record	Additional field in completion record to identify operand causing page fault or other software error. (Refer to section 8.2.3.)
Performance Monitoring Changes	Support for new event category and filter type, and additional performance monitoring events.
Translation Fetch Descriptor	Addition of a new Translation Fetch descriptor to allow software to prefetch address translations into the device and warm up system IOTLBs to reduce address translation latency.
WQ OPCFG Support	Ability to control which operations are supported at a work queue granularity.
Engine Pipeline Depth Control	Ability to discover and control number of outstanding work descriptors and batch descriptors in each engine.
DIX Operations	Addition of new DIX Generate operation. Similar to existing DIF Insert operation, but with the DIF field written to a distinct buffer instead of being interspersed with the source data blocks.

§