INTEL® DATA STREAMING ACCELERATOR PRELIMINARY ARCHITECTURE SPECIFICATION

Order Number: 341204-001US
Revision: 1.0
November 2019

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.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

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.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

This document contains information on products in the design phase of development.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at http://www.intel.com.

Intel® 64 architecture requires a system with a 64-bit enabled processor, chipset, BIOS and software. Performance will vary depending on the specific hardware and software you use. Consult your PC manufacturer for more information. For more information, visit http://www.intel.com/info/em64t.

Intel® Virtualization Technology requires a computer system with an enabled Intel® processor, BIOS, and virtual machine monitor (VMM). Functionality, performance or other benefits will vary depending on hardware and software configurations. Software applications may not be compatible with all operating systems. Consult your PC manufacturer. For more information, visit http://www.intel.com/go/virtualization.

The products described in this document 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.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

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	Preliminary draft release.

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.
	Engine	An independent operational unit within the Intel DSA device.
ENQCMD	Enqueue Command	A CPU instruction to enqueue a command to a shared work queue.
ENQCMDS	Enqueue Command as Supervisor	A CPU instruction to enqueue a command to a shared work queue with Supervisor permissions (from privileged software).
IMS	Interrupt Message Storage	A Scalable I/O Virtualization feature used to store MSI messages in a device-specific manner.
Intel® I/OAT	Intel® I/O Acceleration Technology	A component of Intel® Virtualization Technology for Connectivity, improves data flow across the platform to enhance system performance.
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 Direct 64-Byte	A CPU instruction used to enqueue a command to a dedicated work queue.
MSI	Message Signaled Interrupt	A memory write operation to a pre-defined address to generate an interrupt.

Acronym	Term	Description
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). 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 the 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 to build compliant hardware implementations, device driver software developers to program the device, virtualization software providers to efficiently enable 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/en-us/articles/intel-sdm
Intel® Architecture Instruction Set Extensions Programming Reference https://software.intel.com/en-us/download/intel-architecture-instruction-set-extensions-programming-reference
PCI Express* Base Specification 4.0 http://www.pcisig.com/specifications/pciexpress
Intel® Virtualization Technology for Directed I/O Specification https://software.intel.com/en-us/download/intel-virtualization-technology-for-directed-io-architecture-specification
Intel® Scalable I/O Virtualization Technical Specification https://software.intel.com/en-us/download/intel-scalable-io-virtualization-technical-specification
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. Intel DSA hardware provides a PCI Express* compatible programming interface to the Operating System and can be controlled through a device driver.

Besides doing basic data mover operations, Intel DSA is designed to perform some number of higher-level transformation operations on memory. For example, Intel DSA can generate and test CRC checksum or Data Integrity Field (DIF) on the memory region, to support usages typical with storage and networking applications. Intel DSA supports a memory compare operation for equality, generates a delta record, and applies a delta record to a buffer. These compare 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.

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**: Storage appliances use data movement (including CRC generation and Data Integrity Field (DIF) generation) within the node and across nodes using Non-Transparent Bridge (NTB).
- **Networking**: Packet processing pipelines use Intel DSA for data copy. An example usage is virtual switch (vSwitch) offload for inter-VM packet switching.
- **Deduplication**: Memory deduplication requires comparing memory pages for equality, which can be done using Intel DSA memory compare operations.
- **VM Migration and Fast Checkpointing**: VM fast checkpointing and VM migration flows require the VMM to identify a VM’s dirty 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 delta record to the destination, reducing network bandwidth.

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 Intel® DSA Infrastructure Features

The following infrastructure features are supported by Intel DSA.

- **Shared Virtual Memory (SVM):** Intel DSA supports SVM which allows user level applications to submit commands to Intel DSA directly with virtual addresses in the descriptors. Intel DSA 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 support in Intel DSA, it is possible for an operation to 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 Intel DSA 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:** Intel DSA supports submitting descriptors in a batch. A Batch descriptor points to a set of virtually contiguous work descriptors (i.e., descriptors with actual data operations). When processing a batch descriptor, Intel DSA fetches the work descriptors from the specified memory and processes them.
- **Stateless device:** Intel DSA 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:** Intel DSA supports scalable work submission through Shared Work Queues (SWQ) using the ENQCMD and ENQCMDS instructions.
- **Dedicated Work Queue (DWQ) support:** Intel DSA supports high-throughput work submission through Dedicated Work queues (DWQ) using MOVDIR64B instruction.
- **QoS support:** Intel DSA has several features that allow the kernel driver to separately control access to Intel DSA resources by different guests and applications.
- **Scalable IOV support:** Intel Scalable IO Virtualization improves scalability of device assignment, allowing a VMM to share Intel DSA 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.

2.2.2 Intel® DSA 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	Data Integrity Field (DIF) check. DIF insert, strip, or update while transferring data.
	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 Intel DSA 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 given address range from all levels of CPU caches.

Table 2-1: Intel® DSA Data Operations

2.2.3 Intel® DSA 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 Intel DSA instance is exposed as a single Root Complex Integrated Endpoint. Each Intel DSA 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 Intel DSA 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. Intel DSA uses the PASID TLP prefix in upstream requests to support both Shared Virtual Memory (SVM) and Scalable I/O Virtualization (Scalable IOV). Intel DSA 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 Scalable I/O virtualization. The device can be shared directly to multiple VMs in a secure and isolated manner to achieve high throughput. Sections 3.13 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. Intel DSA uses the I/O fabric interface for receiving downstream work requests from clients and for upstream read, write, and address translation operations. Intel DSA 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 receives descriptors using non-posted ENQCMD/S instructions or as a Dedicated Work Queue (DWQ) that receives descriptors using posted MOVDIR64B instructions. The configuration also allows software to control which WQs feed into which engines and the relative priorities of the WQs feeding each engine.

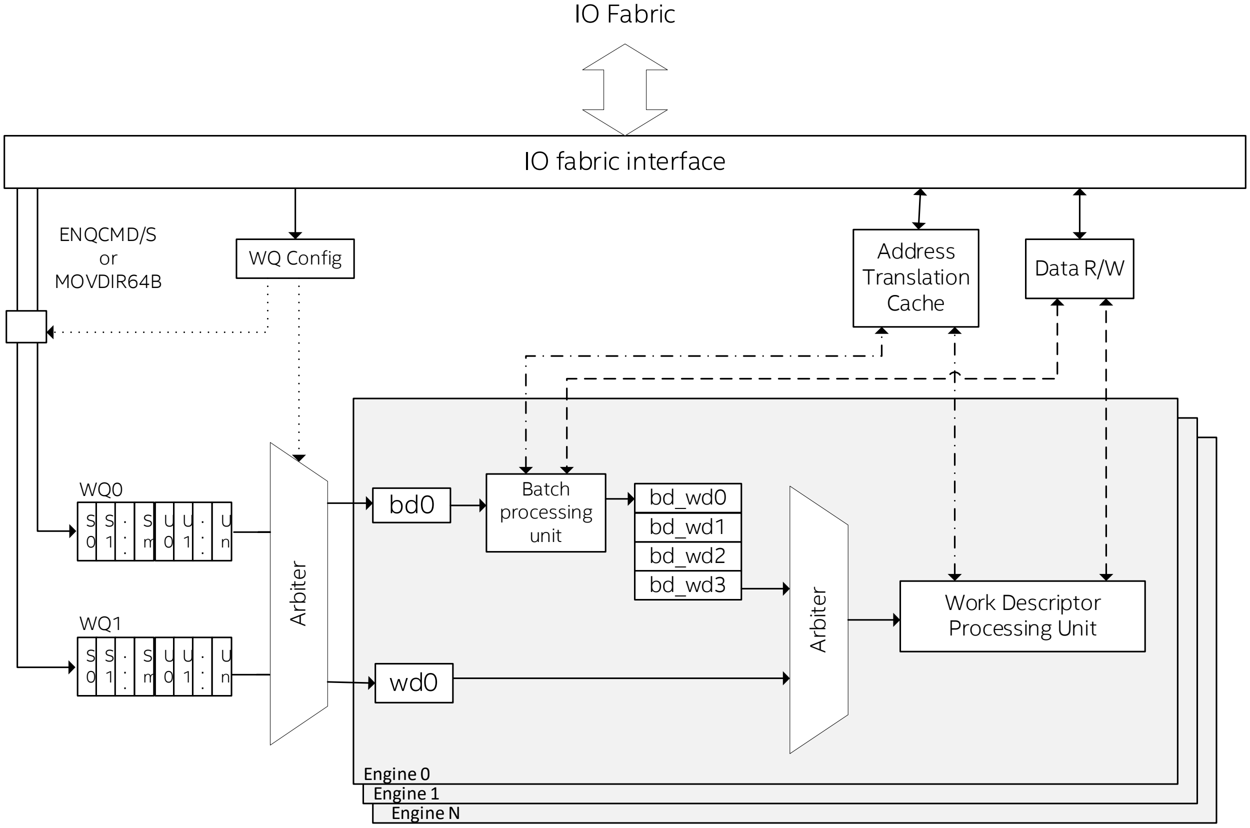


Figure 3-1: Abstracted Internal Block Diagram of Intel® DSA

3.1 Intel® DSA 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.

The Intel DSA device uses memory-mapped registers for controlling its 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.

3.2 Descriptors

Software specifies work for Intel DSA using descriptors. Descriptors specify the type of operation for Intel DSA 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 its shareability 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 9.2.17 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 9.2.17 below 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 unlimited portals are intended to be used by kernel-mode software, and the limited portals are intended to be used by user-mode and guest software. 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 3.7, “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.

Descriptor submissions to a Shared WQ using a limited portal are subject to the occupancy threshold of the WQ, as configured using the WQCFG register. Descriptor submissions using an unlimited portal are not subject to the threshold. Dedicated WQs do not use the threshold, so there is no difference between the limited and unlimited portals for a DWQ.

3.3.1 Shared Work Queue (SWQ)

In Shared mode, an Intel DSA client uses the ENQCMD or ENQCMDS instruction to submit descriptors to the work queue. ENQCMD and ENQCMDS use a 64-byte non-posted write and wait for a response from the device before completing. Intel DSA 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. The ENQCMD and ENQCMDS instructions return the status of the command submission in EFLAGS.ZF flag (0 indicates Success, and 1 indicates Retry). Using the ENQCMD and ENQCMDS instructions, multiple clients can 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.

In Shared mode, Intel DSA can be configured to reserve some SWQ capacity for submissions via the unlimited portals for kernel-mode clients. Work submission via the limited portals is accepted until the number of descriptors in the SWQ reaches the threshold configured for the SWQ. Work submission via the unlimited portals is accepted until the SWQ is full.

If the ENQCMD or ENQCMDS instruction returns Success, the descriptor has been accepted by the device and queued for processing. If the instruction 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 the kernel-mode driver to submit the descriptor on its behalf using an unlimited portal. This helps avoid denial of service and provide forward progress guarantees.

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. ENQCMD copies the PASID of current thread from 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 (see Table 1-1: References).

3.3.2 Dedicated Work Queue (DWQ)

In Dedicated mode, an Intel DSA client uses the MOVDIR64B instruction to submit descriptors to the device work queue. MOVDIR64B uses a 64-byte posted write and the instruction completes faster due to the posted nature of the write operation. For dedicated work queues, Intel DSA exposes the total number of slots in the work queue and depends on software to provide flow control. 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.)

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, Intel DSA uses the WQ PASID field of the WQ Configuration register to do address translation. The WQ PASID field must be set by the Intel DSA driver before enabling a work queue in dedicated mode.

Although dedicated mode doesn't support sharing 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 Intel DSA 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. Intel DSA may use any engine in a group to process a descriptor posted to any WQ in the group. Each WQ and each engine may be in only one group.

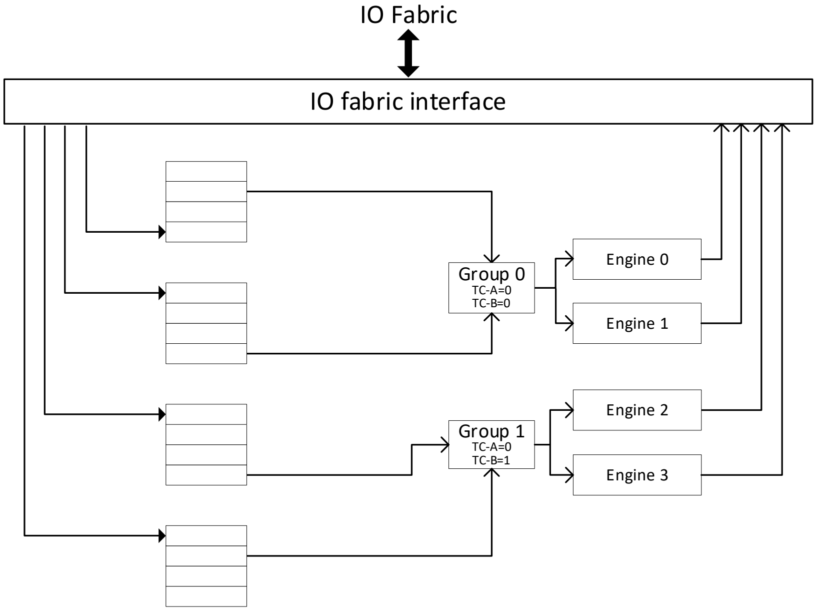


Figure 3-2: Sample Group Configuration 1

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. An example configuration is shown in Figure 3-2 with two groups, each provisioned with two engines. Hardware uses either engine in a group to process descriptors from any work queue in the group. In this configuration, 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.

Another example configuration illustrated in Figure 3-3 places each engine 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.

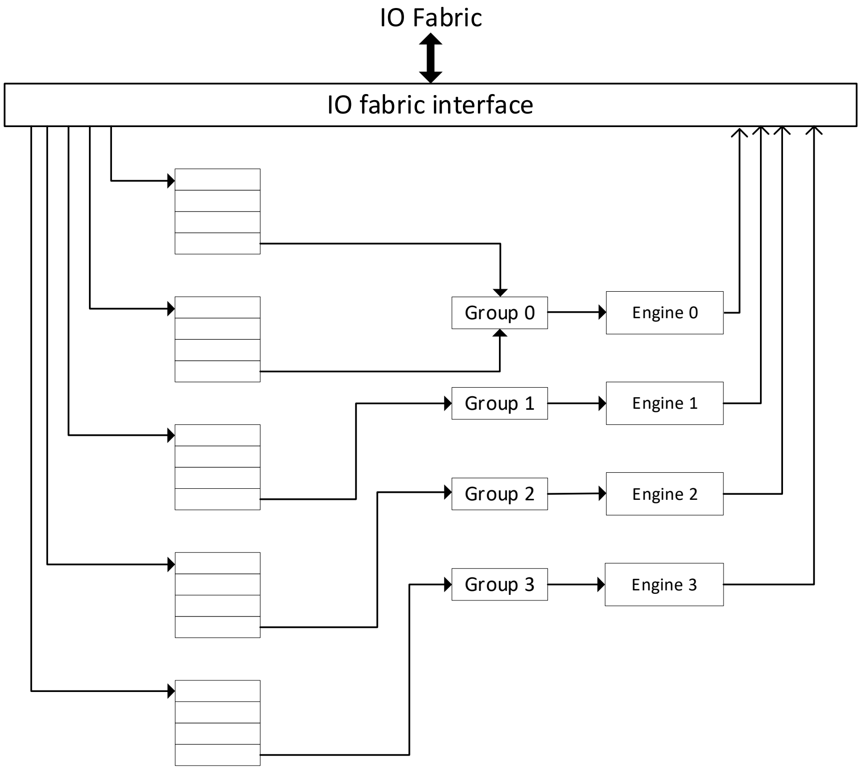


Figure 3-3: Sample Group Configuration 2

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 3.11.

3.7 Interrupts

Intel DSA supports only message signaled interrupts. Intel DSA provides two types of interrupt message storage: (a) an MSI-X table, enumerated through the MSI-X capability, which stores interrupt messages used by the host driver; and (b) a device-specific Interrupt Message Storage (IMS) table, as described by the Scalable IOV architecture specification, which stores interrupt messages used by guest drivers. Refer to the Scalable I/O Virtualization Architecture Specification (see Table 1-1: References) and section 9.2.21 for more details on IMS.

Interrupts can be generated for five 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; and 5) performance monitoring counter overflow. For each type of event there is a separate interrupt enable. Interrupts of types 3, 4 and 5 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.

When the Interrupt Handle Request capability is 0, the Completion Interrupt Handle is the index of the desired entry in the MSI-X table or the IMS. When the Interrupt Handle Request capability is 1, software uses the Request Interrupt Handle command to obtain the handle to use for the interrupt. Software specifies in the Request Interrupt Handle command which interrupt table entry it wants a handle for, and 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. See section 7.3.3 for a description of interrupt virtualization.

Event	Submission register	Interrupt message used
Error posted in SWERROR register	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.

Table 3-1: Interrupt Delivery

The MSI-X table defined by the PCIe* specification is augmented in Intel DSA by the MSI-X Permissions Table, detailed in section 9.2.15. 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.

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 Intel DSA clients to submit multiple work descriptors using a single ENQCMD, ENQCMDS, or MOVDIR64B instruction 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 the Batch descriptor are used for all descriptors in the batch.[1] The PASID and Priv fields in the descriptors in the batch are ignored.

Each work descriptor in the 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 batch.

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.

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.

[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.

3.9 Ordering and Fencing

By default, Intel DSA doesn't guarantee any ordering while executing work descriptors. Descriptors can be dispatched and completed in any order the device sees fit to maximize throughput. If ordering is required, there are several ways for software to enforce it:

- 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 Intel DSA.

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 completed. Any descriptors that were partially completed and generated a completion record are counted as completed. Only the abandoned descriptors are considered not completed.

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. Software may avoid partial completion by setting the Block on Fault flag in the descriptor. A completion interrupt (if requested) is ordered after the completion record write.

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 submitted to Intel DSA 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). Software may avoid the need for a follow-up descriptor by setting the Block on Fault flag in the initial descriptor to prevent partial completion.

3.10 Drain Descriptor

Drain is a normal descriptor that is submitted to a work queue and allows a client to wait for completion of preceding descriptors in the WQ that the Drain descriptor is submitted to. For the purpose of Drain, a 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.

If a Drain descriptor is submitted to a dedicated WQ, it waits for completion of all descriptors in the WQ. If Drain is submitted to a shared WQ, it waits for descriptors in the WQ that have the same PASID as the Drain descriptor. Drain can be used like a Fence operation for the entire PASID. A Drain descriptor can be used by software 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 the Drain descriptor to ensure that the Drain descriptor is received by Intel DSA after the descriptors it is intended to drain.

To ensure completion of descriptors in the WQ, prior to Drain descriptor completion, hardware normally issues an implicit readback for each supported Traffic Class, using an address determined by hardware. Software can control this default behavior by setting one or both of the Suppress flags in the Drain descriptor. If unsuppressed through flags, the Drain descriptor implicit readbacks ensure that all previous writes to the Root Complex, on the respective TCs, have completed. Previous writes to a peer device (i.e. non-Root Complex) may not get 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.

3.11 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, Intel DSA 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. Page fault handling is described below.

The Intel DSA device may encounter a page fault on one of: 1) a Completion Record Address; 2) the Descriptor List Address in a Batch descriptor; 3) Readback address in a Drain descriptor (See sections 3.10 and 8.3.3 for more details); or 4) a source buffer or destination buffer address. For the first three cases, the Intel DSA device blocks until the page fault is resolved, if PRS is enabled; otherwise it is reported as an error. For the fourth case, the Intel DSA device can either block until the page fault is resolved or prematurely complete the descriptor and return a partial completion to the client, as specified by software.

When Intel DSA blocks on a page fault it reports the fault 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. If the OS was not able to create a mapping, it returns an error response and Intel DSA completes the descriptor with an error.

Each descriptor has a Block On Fault flag which indicates whether Intel DSA should return a partial completion or block when a page fault occurs on a source or destination buffer address. When the Block On Fault flag is 1, and a fault is encountered, 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.

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.) 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.

The Block On Fault Support field in the General Capabilities register (GENCAP) indicates device support for this feature, 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 use the feature. These registers are described in section 9.1.4.

Device page faults are relatively expensive, higher than cost of servicing CPU page faults. Even if the device reports partial work completion instead of blocking on faults, it still incurs overheads because it requires software intervention to service the page-fault and resubmit the work. 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 descriptors 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.

3.12 Administrative Commands

Administrative commands are submitted to Intel DSA 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.

Intel DSA supports the following commands. See the description of the Command register for details on how to submit these commands.

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.
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 Abort All	Wait for all descriptors in all WQs and all engines that were submitted prior to the Drain All or Abort 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.
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 WQs after the command is submitted and before it completes; otherwise the behavior is undefined.
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.
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. 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, which are not modified. This command allows specification of multiple work queues. See Table 9-3 for the initial values of device registers.
Request Interrupt Handle	Requests an interrupt handle that can be used in descriptors to request completion interrupts. This command is usable only if the Interrupt Handle Request capability in GENCAP is 1; otherwise, it is not needed. The result of this command may be an error, if no additional interrupt handles are available.

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 Intel DSA. 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 Intel DSA 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, Intel DSA discards all descriptors belonging to the specified PASID without further processing.

3.13 Virtualization

The Intel DSA architecture is designed to be easy and efficient to virtualize. Intel DSA supports the Scalable I/O virtualization (IOV) model. Refer to the Scalable IO Virtualization Architecture Specification (see Table 1-1: References) for more details on the Scalable IOV architecture. 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:** Intel DSA 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 Intel DSA 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.
- **Intel DSA Capabilities:** Intel DSA provides capability registers to expose support for features such as block-on-fault to software. 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.
- **Scalable IOV:** Intel DSA supports the Scalable IO Virtualization architecture, which 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 9.2.17. 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. Traffic Classes may be used by the platform outside of Intel DSA to control QoS for memory transactions initiated by Intel DSA.

Traffic Classes are also used within Intel DSA 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 4.4 for information on configuring traffic classes for use with low bandwidth memory.

Intel DSA provides 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 arrange that operations with dissimilar QoS characteristics are issued to different groups.

4.3 Bandwidth Control

For software to control the bandwidth available to certain guests or applications, it may use the bandwidth tokens fields in the Group Configuration registers.

Bandwidth tokens represent resources within the Intel DSA implementation. These resources are allocated by engines to support operations. The total number of bandwidth tokens supported is fixed by the Intel DSA implementation and is reported in the GRPCAP register. The relationship between bandwidth tokens and actual bandwidth is dependent on instantaneous system memory latency and varies dynamically as system utilization changes. Bandwidth tokens 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 Intel DSA.

Bandwidth tokens are apportioned to groups based on two fields in the Group Configuration registers. The Bandwidth Tokens Reserved field indicates the number of bandwidth tokens set aside for the exclusive use of engines in the group. The Bandwidth Tokens Allowed field indicates the maximum number of bandwidth tokens that may be in use at one time by all engines in the group. (Bandwidth tokens for an operation may also be limited by the Global Bandwidth Token Limit, as described in section 4.4.)

Setting the Bandwidth Tokens Reserved field to a non-zero value ensures that engines in the group are always able to acquire some resources. However, if it is too high, it limits the ability of other engines to utilize the full bandwidth capability of the device. The sum of the bandwidth tokens reserved for all groups must be no greater than the total number of bandwidth tokens available (as reported in GRPCAP).

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

There is an implementation-specific value for the Bandwidth Tokens Allowed field that allows the group to utilize the full bandwidth of Intel DSA. This value will be determined experimentally. There is no advantage to setting the Bandwidth Tokens Allowed field to a greater value. Setting this field to a smaller value limits the bandwidth tokens the engines in the group can consume, thereby limiting their impact on the performance of engines in other groups.

4.4 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 access these types of memory, software should take these steps to limit the impact to the throughput of other operations, both within Intel DSA and throughout the platform.

1. Set the Global Bandwidth Token Limit field in GENCFG to a suitable value for the bandwidth available. (This value is platform dependent and will be determined experimentally.)
2. Create one or more groups that will be used with descriptors that access low bandwidth memory.
3. Set the Use Global Bandwidth Token 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. 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 field 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 bandwidth tokens specified by Global Bandwidth Token Limit is shared by all descriptors executing in groups for which Use Global Bandwidth Token Limit is 1. The engine executing the descriptor is also limited by the Bandwidth Tokens Allowed field in GRPCFG.

4.5 Persistent Memory Support

Intel DSA provides several features intended to improve its utility with persistent memory such as Intel Optane DC persistent memory.

- The Steering Tag Selector fields in the descriptor are used to select a platform-defined steering tag. Write operations generated by the descriptor are tagged with the selected steering tag from the TPH ST Table in the TPH Requester Capability. The steering tag is intended to indicate to the platform whether the write operation is destined for persistent memory. Use of the appropriate steering tag ensures that the data written has become persistent at the time the descriptor completes.
- The Strict Ordering flag causes Intel DSA to indicate to the platform that write operations generated by the descriptor may not be reordered. If the destination of the descriptor is in persistent memory, this flag allows software to rely on the guarantee that later write operations cannot become persistent while earlier write operations were lost.

4.6 Cache Control

The Cache Control flag in the descriptor is a hint indicating whether destination addresses targeted by the descriptor should reside in memory or in cache. The flag has two similar purposes, depending on the operation type. If the flag is 0, it hints that the destination buffer should be in memory. If the flag is 1, it hints that the destination buffer should be in cache. 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.

- For operations that write to memory, the Cache Control flag hints whether data written 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 Cache Flush operation writes modified data contained in the caches to memory. The Cache Control flag controls whether targeted cache lines are also removed from the caches. If the flag is 0, affected cache lines are invalidated from all cache levels. If the flag is 1, affected cache lines that are present in the caches are not evicted.

For operations other than these two categories, the Cache Control flag is reserved.

§

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.

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.
	Non-posted write to DWQ Non-posted write to WQ that is not Enabled Non-posted write to non-WQ address	Returns Retry.
Descriptor errors	Misaligned completion record address	Reported in SWERROR.
	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.
	Error in descriptor processing (e.g., PRS failure)	
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

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.
- 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 Bandwidth Token Limit Supported field in GRPCAP is 0, then the Use Global Bandwidth Token Limit field is 0 in every GRPCFG register.
- If the Global Bandwidth Token Limit Supported field in GRPCAP is 1, then the Global Bandwidth Token Limit in GENCFG is less than or equal to the Total Bandwidth Tokens field in GRPCAP.
- If the Use Global Bandwidth Token Limit field is 1 in any GRPCFG register, then the Global Bandwidth Token Limit in GENCFG is greater than 0.
- If the Bandwidth Tokens Supported field in GRPCAP is 1, then the sum of the Bandwidth Tokens Reserved fields, for all groups that have engines assigned, is less than or equal to the Total Bandwidth Tokens field in GRPCAP.
- If the Bandwidth Tokens Supported field in GRPCAP is 1, then for each group that has engines assigned to it, Bandwidth Tokens Allowed is:
  - Greater than or equal to 4 times the number of engines in the group;
  - Greater than or equal to the Bandwidth Tokens Reserved field for the group; and
  - Less than or equal to the sum of the Bandwidth Tokens Reserved field and the number of non-reserved bandwidth tokens.
- 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 any of these 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.

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 any of these 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 a non-posted, aligned 64-byte write, 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 a non-posted, aligned 64-byte write (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 (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.

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

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 which 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.
- 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.
- 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.

Submission Portal	Interrupt Handle Request Capability	Completion Interrupt Handle Check
MSI-X	0	Less than the size of the MSI-X table.
IMS	0	Less than Interrupt Message Storage Size.
MSI-X	1	A valid MSI-X handle returned by the Request Interrupt Handle command.
IMS	1	A valid IMS handle returned by the Request Interrupt Handle command.

Table 5-2: Completion Interrupt Handle Checks

- 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.

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

If the Completion Record Address Valid flag is 1 and the Completion Record Address is misaligned or cannot be translated, the descriptor is discarded, and an error is reported in the Software Error Register.

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. In that case, the error is also reported in the Software Error register.

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 bit 6 is 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.

	Allowed Flags										
Operation	Block on Fault	Check Result	Cache Control	Strict Ordering	Destination Readback	Destination Steering Tag	Address 1 TC Selector	Address 2 TC Selector	Address 3 TC Selector	Fence	Reserved fields
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

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.
Destination Steering Tag Selector	TPH Requester Control Register ST Mode Select = 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.
Completion Record Steering Tag Selector	Completion Record Address Valid = 0 or TPH Requester Control Register ST Mode Select = 0.
Cache Control	For operations that write to memory, if GENCAP Cache Control Support (Memory) = 0. For the Cache Flush operation, if GENCAP Cache Control Support (Cache Flush) = 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.

Table 5-4: Conditional Reserved Field Checking

Table 5-5 below gives the list of operation types that require certain flags to be set to 1.

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	

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

5.6 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.

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).

5.7 Error Codes

5.7.1 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 written to the SWERROR register instead.

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 either: - Page Request Services are disabled; 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. Any flag contains an unsupported or reserved value.
0x12	Non-zero reserved field (other than a flag).
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.
0x1a	A page fault occurred while translating a Completion Record Address and either: - Page Request Services are disabled; 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: - 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.
0x1d	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 either: - Page Request Services are disabled; 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 other than a completion timeout or unsuccessful completion status for destination readback. Details of the hardware error are reported via PCIe Advanced Error Reporting (AER), if enabled.
0x21	Hardware error (completion timeout or unsuccessful completion status) on a destination readback operation. Error details are reported via PCIe Advanced Error Reporting (AER), if enabled.
0x22	An error occurred during address translation: - An UR or CA response to 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 – 0x3f	Unused.

Table 5-6: Operation Status Codes

5.7.2 Other Software Error Codes

These errors are reported in the SWERROR register.

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.
0x52	The Command register was written while the Active field of the Command Status register was 1.
0x53	A descriptor was submitted to a dedicated WQ that had no space to accept the descriptor.

Table 5-7: Other Software Error Codes

§

6 Performance Monitoring

The purpose of the performance monitoring architecture is to support collection of information about key events occurring during device execution, to aid performance tuning and debug. Details will be included in future revision of this document.

§

7 Reference Software Architecture

Software support for Intel DSA is envisioned to consist of the following:

- 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 Intel DSA 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 the OS to allocate/bind/unbind/free PASIDs based on user level requests. It maps limited portals to user-mode clients for direct access. Intel DSA KMD can service requests from kernel-mode or user-mode clients using ENQCMDS to an unlimited portal of a shared work queue. Additionally, it can service requests from kernel-mode clients using MOVDIR64B to a dedicated work queue portal.

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 applications as a library and interfaces with the kernel-mode driver to request access to Intel DSA hardware on behalf of the applications. It exposes various Intel DSA functions to the applications 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 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 Virtualization Software

The Intel DSA device is virtualized using the Scalable IOV model, described in the Intel® Scalable I/O Virtualization Architecture Specification. The virtualization software architecture is shown in Figure 7-1. Virtualization of the Intel DSA 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 Intel DSA virtualization. Depending on the host system software architecture, the VDCM may be developed as a user level module, as part of the Intel DSA driver, as a separate kernel module, or as part of the VMM. The Intel DSA KMD in the Host OS is extended to support the Intel DSA VDCM operations required for Intel DSA virtualization. The Intel DSA KMD with virtualization extensions is called the Intel DSA host driver. The Intel DSA KMD in the Guest OS may run exactly like in non-virtualization environment or it may be optimized to run in a VM. Intel DSA KMD in the guest OS is called Intel DSA 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. The sections below describe various aspects of Intel DSA virtualization.

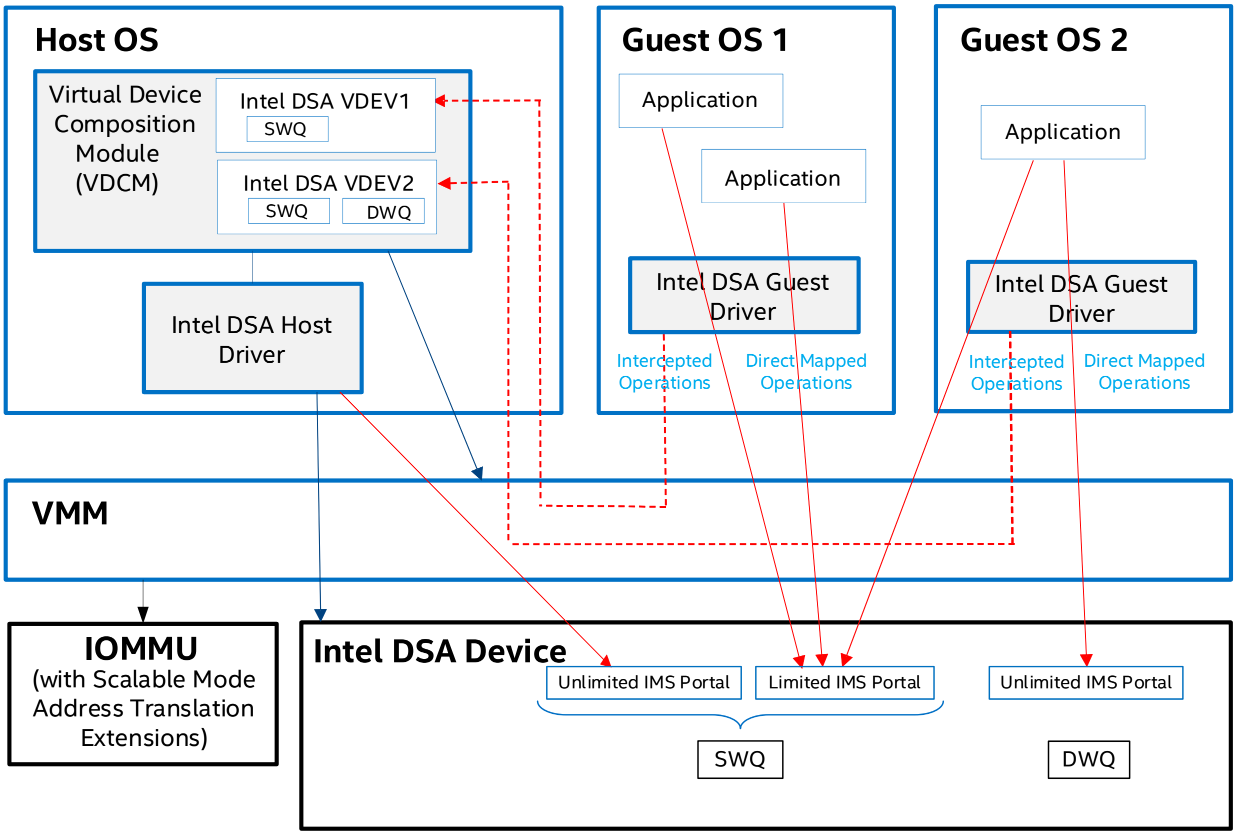


Figure 7-1: Scalable IOV for Intel® DSA

7.3.1 Virtual Intel® DSA Device

The virtual Intel DSA device implemented by the VDCM, called VDEV, emulates the same interface as the physical Intel DSA device, so that the same Intel DSA driver can run in both the host OS and the guest OS. The guest driver accesses the virtual Intel DSA 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.

Also, 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.

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 should retain control of the unlimited portal, and the VDCM should map the limited portal into the guest in place of both the unlimited portal and the limited portal. 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 7-1 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.

7.3.2 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 host PASID table entry for that PASID 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 Intel DSA device in this way, the VDEV need not support the PASID 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.

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.

7.3.3 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 the host driver to 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 the Host OS or VMM to 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 Interrupt Handle Request capability in the VDEV is 1, indicating that the guest must use the Request Interrupt Handle command to obtain the 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. Intel DSA uses the handle to identify the Interrupt Message Storage entry to be used to generate the completion interrupt. Intel DSA 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 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.

7.3.4 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 Intel DSA capabilities to the VM through the virtual Intel DSA device's capability registers, allowing the virtual device to be compatible with multiple generations of Intel DSA devices. This capability virtualization enables a VM image with a guest Intel DSA 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 data center where the same VM image can be started or migrated.

7.3.5 Intel® DSA 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 the host driver to 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 Intel DSA 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 Intel DSA 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 Intel DSA state to it. Specifically, it configures IMS entries for interrupts that are configured in the virtual MSI-X table, assigns physical WQs as per the virtual Intel DSA's configuration to the VM, 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 Intel DSA driver may continue to use the old DWQ size. The capability virtualization described in section 3.13 ensures that the virtual Intel DSA device can work on multiple generations of Intel DSA devices.

§

8 Descriptor Formats

8.1 Common Descriptor Fields

Intel DSA descriptors are 64 bytes. Some descriptor fields are common to all operations 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 Intel DSA 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 Intel DSA clients.

Generic Descriptor Format

<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">Ignored</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="16" rowspan="3">Operation-specific fields</td><td>40</td></tr>
<tr><td>48</td></tr>
<tr><td>56</td></tr>
</table>

8.1.1 Trusted Fields

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

When a descriptor is submitted with ENQCMD, these fields in the source descriptor are ignored. 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 a descriptor is submitted with 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.

When a descriptor is submitted with MOVDIR64B, these fields in the descriptor are ignored. The device uses the WQ Priv and WQ PASID fields of the WQCFG register.

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	Ignored
19:0	**PASID** This field contains the Process Address Space ID of the requesting process.

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
0x10	CRC Generation
0x11	Copy with CRC generation
0x12	DIF Check
0x13	DIF Insert
0x14	DIF Strip
0x15	DIF Update
0x20	Cache flush

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. This field is reserved for all other operation types.

15	**Destination Steering Tag Selector** Selects a steering tag entry from the TPH ST Table in the TPH Requester Capability to be used for writes to Destination Address. The meaning of the steering tags is platform dependent, but is expected to be programmed as follows: **0:** Writes to the destination are not identified as writes to durable memory **1:** Writes to the destination are identified on the fabric as writes to durable memory. This field is reserved if the ST Mode Select field in the TPH Requestor Control Register is 0. For Memory Copy with Dualcast, this field selects the steering tag for Destination1. This field is reserved for operation types that do not write to memory: No-op, Drain, Batch, Compare, Compare Pattern, CRC Generation (without copy), and DIF Check.
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. If the descriptor is partially completed, the readback is not performed. 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: No-op, Drain, Batch, Compare, Compare Pattern, CRC Generation (without copy), and DIF Check.
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, so they become globally observable in the exact order issued by the device. This field is reserved for operation types that do not write to memory: No-op, Drain, Batch, Compare, Compare Pattern, CRC Generation (without copy), and DIF Check. 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: No-op, Batch, Compare Pattern, CRC Generation, and DIF Check.
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 (Memory) field in GENCAP is 0, this field is ignored in these descriptors. For the Cache Flush operation: **0:** Cache lines that contain modified data are written back to memory, and all affected cache lines are invalidated from every level of the processor caches. **1:** Cache lines that contain modified data are written back to memory but affected cache lines are not evicted from the processor caches. If the Cache Control Support (Cache Flush) field in GENCAP is 0, this field is ignored in Cache Flush descriptors. This field is reserved for operation types that do not write to memory: No-op, Drain, Batch, Compare, Compare Pattern, CRC Generation (without copy), and DIF Check.
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	**Reserved.** Must be 0.
5	**Completion Record Steering Tag Selector** Selects a steering tag entry from the TPH ST Table in the TPH Requester Capability to be used for writing the completion record. The meaning of the steering tags is platform dependent, but is expected to be programmed as follows: **0:** Writes to the completion record are not identified as writes to durable memory **1:** Writes to the completion record are identified on the fabric as writes to durable memory. This field is reserved if the ST Mode Select field in the TPH Requestor Control Register is 0 or if Completion Record Address Valid is 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 to 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 3.11. 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. **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 to the SWERROR register, 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 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 always contains enough information to continue the operation if it was partially completed due to a page fault.

Generic Completion Record

Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	bytes
Bytes Completed				Unused		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 5.7.1 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, or 0x1a) **0:** the faulting access was a read. **1:** the faulting access was a write.
6	Unused.
5:0	**Operation Status** See section 5.7.1 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. For operation types where this field has no meaning, it is 0.

8.2.3 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.

For some operation types, this field may also be used when the operation stopped before completion for some reason other than a fault.

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.

8.2.4 Fault Address

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

If the operation was partially completed due to a page fault, this field contains the address that caused the fault.

8.2.5 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 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 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 Status field of the Batch completion record indicates Success if all of the descriptors in the batch completed successfully; otherwise it indicates that one or more descriptors completed with Status not equal to Success. If Status is not equal to Success, 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.

See section 3.8 for details of batch processing.

Batch 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">Descriptor List Address</td><td>16</td></tr>
<tr><td colspan="8"></td><td>24</td></tr>
<tr><td colspan="2"></td><td colspan="2">Completion Interrupt Handle</td><td colspan="4">Descriptor Count</td><td>32</td></tr>
<tr><td colspan="8" rowspan="3">Reserved</td><td>40</td></tr>
<tr><td>48</td></tr>
<tr><td>56</td></tr>
</table>

Batch 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">Descriptors 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" rowspan="2">Unused</td><td>16</td></tr>
<tr><td>24</td></tr>
</table>

8.3.3 Drain

The Drain operation waits for completion of outstanding descriptors in the WQ that the Drain descriptor is submitted to, as described in section 3.10. This descriptor may be used during normal shutdown by a process that has been using the device. To wait for all descriptors associated with the PASID, software should submit a separate Drain operation to every WQ that the PASID was used with.

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

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

Table 8-5 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

<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">Readback Address 1</td><td>16</td></tr>
<tr><td colspan="16">Readback Address 2</td><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>

Drain 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></td><td>8</td></tr>
<tr><td></td><td>16</td></tr>
<tr><td></td><td>24</td></tr>
</table>

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 8-5: Drain Operation-specific Flags

8.3.4 Memory Move

The Memory Move operation 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
		Completion Interrupt Handle		Transfer Size				32
Reserved								40
								48
								56

Memory Move Completion Record

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

8.3.5 Fill

The Memory Fill operation fills memory at the Destination 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 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								16
Destination Address								24
		Completion Interrupt Handle		Transfer Size				32
Reserved								40
								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			Status	0
Fault Address								8
Unused								16
								24

8.3.6 Compare

The Compare operation 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">Reserved</td><td>Expected Result</td><td>40</td></tr>
<tr><td colspan="8" rowspan="2"></td><td>48</td></tr>
<tr><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 colspan="2">Unused</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 the table below. 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 8-6: Completion Status for Compare Descriptor

8.3.7 Compare Pattern

The Compare Pattern operation 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		Result	Status	0
Fault Address								8
Unused								16
								24

8.3.8 Create Delta Record

The Create Delta Record operation 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.

Create Delta Record 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">Reserved</td><td colspan="2">Completion Interrupt Handle</td><td colspan="4">Transfer Size</td><td>32</td></tr>
<tr><td colspan="8">Delta Record Address</td><td>40</td></tr>
<tr><td colspan="4"></td><td colspan="4">Maximum Delta Record Size</td><td>48</td></tr>
<tr><td colspan="7">Reserved</td><td>Expected Result Mask</td><td>56</td></tr>
</table>

Create Delta Record 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="2">Unused</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="4"></td><td colspan="4">Delta Record Size</td><td>16</td></tr>
<tr><td colspan="8">Unused</td><td>24</td></tr>
</table>

8.3.9 Apply Delta Record

The Apply Delta Record operation 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.

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				Unused			Status	0
Fault Address								8
Unused								16
								24

8.3.10 Memory Copy with Dualcast

The Memory Copy with Dualcast operation 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				Unused			Status	0
Fault Address								8
Unused								16
								24

Bits	Description
23:17	**Reserved:** Must be 0.
16	**Destination2 Steering Tag Selector** Selects a steering tag entry from the TPH ST Table in the TPH Requester Capability to use with writes to Destination2. The meaning of the steering tags is platform dependent, but is expected to be programmed as follows: **0:** Writes to the destination are not identified as writes to durable memory **1:** Writes to the destination are identified on the fabric as writes to durable memory. This field is reserved if the ST Mode Select field in the TPH Requestor Control Register is 0.

Table 8-7: Memory Copy with Dualcast Operation-specific Flags

8.3.11 CRC Generation

The CRC Generation operation computes the CRC on memory at the Source Address. 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.

If the Read CRC Seed flag is 1, the CRC seed is read from memory at the CRC Seed Address. The address must be 4-byte aligned. 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 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 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 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

<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"></td><td>24</td></tr>
<tr><td colspan="4" rowspan="2">Reserved</td><td colspan="4">Completion Interrupt Handle</td><td colspan="8">Transfer Size</td><td>32</td></tr>
<tr><td colspan="4"></td><td colspan="8">CRC Seed</td><td>40</td></tr>
<tr><td colspan="16">CRC Seed Address</td><td>48</td></tr>
<tr><td colspan="16">Reserved</td><td>56</td></tr>
</table>

CRC Generation 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="4"></td><td colspan="4">CRC Value</td><td>16</td></tr>
<tr><td colspan="8">Unused</td><td>24</td></tr>
</table>

Bits	Description
23:19	**Reserved:** Must be 0.
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.
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.
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 8-8: CRC Generation Operation-specific Flags

8.3.12 Copy with CRC Generation

The Copy with CRC Generation operation copies memory from the Source Address to the Destination Address and computes the CRC on the data copied. 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 8.3.11 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

<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="2">Reserved</td><td colspan="2">Completion Interrupt Handle</td><td colspan="4">Transfer Size</td><td>32</td></tr>
<tr><td colspan="2"></td><td colspan="4">CRC Seed</td><td>40</td></tr>
<tr><td colspan="8">CRC Seed Address</td><td>48</td></tr>
<tr><td colspan="8">Reserved</td><td>56</td></tr>
</table>

8.3.13 DIF Check

The DIF Check operation 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 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 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 8.3.16, DIF Update, for a description of DIF Flags, Source DIF Flags, and the fields in the completion record.

DIF Check 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">Reserved</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="5"></td><td>DIF Flags</td><td></td><td>Source DIF Flags</td><td>40</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>48</td></tr>
<tr><td colspan="8"></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 colspan="2">Unused</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>

8.3.14 DIF Insert

The DIF Insert operation 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 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.

See section 8.3.16, DIF Update, for a description of DIF Flags, Destination DIF Flags, and the fields in the completion record.

DIF Insert 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
					DIF Flags	Dest DIF Flags		40
								48
Application Tag Seed		Application Tag Mask		Reference Tag Seed				56

DIF Insert Completion Record

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

8.3.15 DIF Strip

The DIF Strip operation 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 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 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 8.3.16, DIF Update, for a description of DIF Flags, Source DIF Flags, and the fields in the completion record.

DIF Strip 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">Reserved</td><td colspan="2">Completion Interrupt Handle</td><td colspan="4">Transfer Size</td><td>32</td></tr>
<tr><td colspan="5">Reserved</td><td>DIF Flags</td><td>Reserved</td><td>Source DIF Flags</td><td>40</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>48</td></tr>
<tr><td colspan="8">Reserved</td><td>56</td></tr>
</table>

DIF Strip 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="2">Unused</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>

8.3.16 DIF Update

The DIF Update operation 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 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.

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		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

8.3.16.1 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

8.3.16.2 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.

8.3.16.3 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 Check, DIF Insert, and DIF Strip operations.
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 Check, DIF Insert, and DIF Strip operations.
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 Check, DIF Insert, and DIF Strip operations.
2:0	Reserved

8.3.16.4 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.

8.3.17 Cache Flush

The Cache Flush operation 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.

If the Cache Control flag is 0, affected cache lines are invalidated from every level of the cache hierarchy. If a cache line contains modified data at any level of the cache hierarchy, the data is written back to memory. This is similar to the behavior of the CLFLUSH and CLFLUSHOPT instructions in the CPU.

If the Cache Control flag is 1, modified cache lines are written to main memory, but are not evicted from the caches. This is like the behavior of the CLWB instruction in the CPU.

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				Unused			Status	0
Fault Address								8
Unused								16
								24

§

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 (Table 1-1: References) 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, using the ENQCMD, ENQCMDS, or MOVDIR64B instructions. ENQCMD or ENQCMDS must be used to access a work queue that is configured as shared (SWQ), and MOVDIR64B must be used to access a dedicated work queue (DWQ).

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.
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.
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.
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.
Write Only	WO	The field is only writeable by software. Reads return 0. Software should not rely on reading this field to determine 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 the PCI Express specification (Table 1-1: References) 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. It is typically twice the size of the Interrupt Message Storage. For example, if the device supports 1024 Interrupt Message Storage entries, the Interrupt Message Storage is 16 KB, and the size of BAR0 is at least 32 KB, but may be larger.

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 a non-posted write, a Retry response is returned. Any read operation to the BAR2 address space returns 0xFF for all bytes.

9.1.2 MSI-X Capability

MSI-X is the only PCI Express interrupt capability that Intel DSA provides. Intel DSA does not implement legacy PCI interrupts or MSI. Details of this register structure are in the PCI Express specification (see Table 1-1: References). 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 disabled 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 9.2.17.) 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 Intel DSA 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 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.

9.1.4 Scalable I/O Virtualization Capability

The Scalable I/O Virtualization capability is defined in the 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.

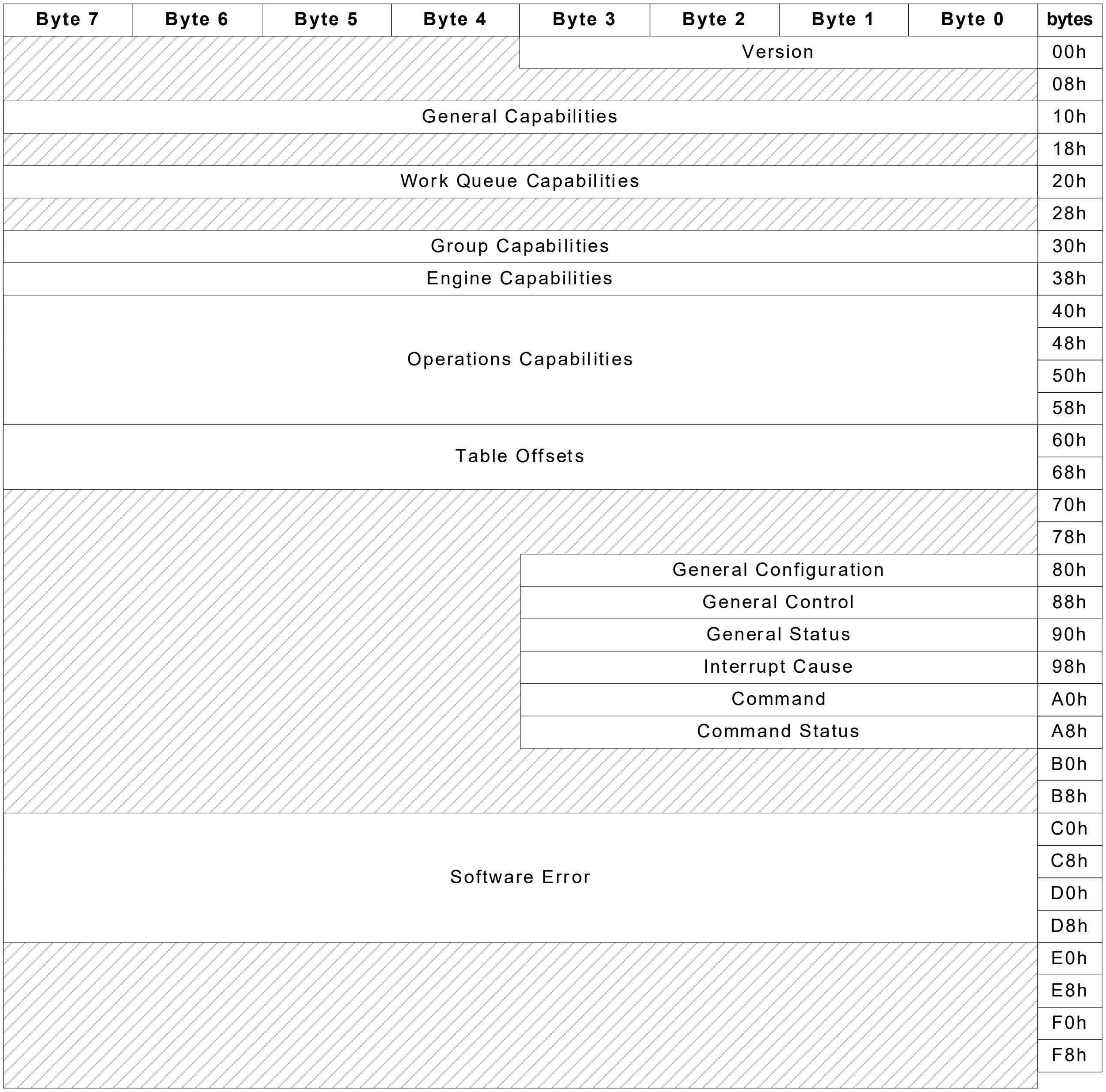
9.1.5 TPH Capability

Software configures the TPH Requester capability and the TPH ST table to specify the platform-specific steering tags to be used for writes to durable and non-durable memory regions. The use of steering tags to ensure durability of writes to local and remote (over NTB) persistent memory is described in section 4.5 above.

9.1.6 VC Capability

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
				▨				B0h
								B8h
Software Error								C0h
								C8h
								D0h
								D8h
▨								E0h
								E8h
								F0h
								F8h

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

[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 Perfmon Capabilities Table Offsets	Contain read-only values indicating capabilities of the device.			
General Configuration General Control General Status Interrupt Cause WQ Configuration[1] MSI-X Pending Bit Array MSI-X Permissions Table	0	0	0	0
Group Configuration[1]	Bandwidth Tokens Allowed: initialized to Total Bandwidth Tokens All other fields: 0			
Command Command Status Software Error	0	0	0	Preserved
Perfmon	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:

Register	Conditions under which register is read-only
General Configuration Group Configuration	While device is not Disabled
WQ Configuration	See Table 9-7

Table 9-4: Read-only MMIO registers

[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.

9.2.1 Version Register (VERSION)

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

VERSION Base: BAR0			Offset: 0x0	Size: 4 bytes (32 bits)
Bit	Attr	Size	Description	
31:16	RO	16 bits	Unused.	
15:8	RO	8 bits	Major version	
7:0	RO	8 bits	Minor version	

9.2.2 General Capabilities Register (GENCAP)

GENCAP Base: BAR0			Offset: 0x10 Size: 8 bytes (64 bits)
Bit	Attr	Size	Description
63:40	RO	24 bits	Unused.
39:32	RO	8 bits	Maximum Descriptors in Progress The maximum number of descriptors that can be in progress in each engine.
31	RO	1 bit	Configuration Support 0: Group configuration and some fields of the Work Queue configuration registers are read-only and reflect the fixed configuration of the device. See section 9.2.17 for details about which WQ configuration fields are read-only. 1: 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.
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.
15:10	RO	6 bits	Unused.
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	Interrupt Handle Request 0: The Request Interrupt Handle command is not supported. The index of the desired MSI-X or IMS entry is used as the interrupt handle. 1: The Request Interrupt Handle command must be used to determine the interrupt handle to use in descriptors.
6:4	RO	3 bits	Unused.

GENCAP Base: BAR0			Offset: 0x10 Size: 8 bytes (64 bits)
Bit	Attr	Size	Description
3	RO	1 bit	Cache Control Support (Cache Flush) 0: Cache control for cache flush operations is not supported. The Cache Control flag in Cache Flush descriptors is ignored. 1: Cache control for cache flush operations is supported. Software can use the Cache Control flag in descriptors to control whether affected lines are evicted from the cache.
2	RO	1 bit	Cache Control Support (Memory) 0: Cache control for memory write operations is not supported. The Cache Control flag in descriptors that write to memory is ignored. 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 3.11 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
63:54	RO	10 bits	Unused.
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 9.2.17.
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	Unused.
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.
47:24	RO	24 bits	Unused.
23:16	RO	8 bits	Number of WQs
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
63:18	RO	46 bits	Unused.
17	RO	1 bit	Global Bandwidth Token Limit Supported 0: The Global Bandwidth Token Limit field of GENCFG and the Use Global Bandwidth Token Limit field in GRPCFG are reserved. 1: The Global Bandwidth Token Limit field of GENCFG and the Use Global Bandwidth Token Limit field in GRPCFG can be used by software to control bandwidth usage by selected groups, as described in chapter 4.
16	RO	1 bit	Bandwidth Tokens Supported 0: Bandwidth tokens are not supported. The value in the Total Bandwidth Tokens field is undefined. The Bandwidth Tokens Allowed and Bandwidth Tokens Reserved fields of GRPCFG are read-only and are unused. 1: Bandwidth tokens are supported as described in chapter 4.
15:8	RO	8 bits	Total Bandwidth Tokens Indicates the total number of bandwidth tokens 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><td>Bit</td><td>Attr</td><td>Size</td><td>Description</td></tr>
<tr><td>63:8</td><td>RO</td><td>56 bits</td><td>Unused.</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)

This register is a bitmask to specify the operation types supported by the device. 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). The bit is set if the operation is supported, and clear if the operation is not supported. 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. See section 8.1.2 for the values of the operation codes. 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 Intel DSA 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, and Performance Monitoring. 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
127:80	RO	48 bits	Unused.
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.

GENCFG Base: BAR0			Offset: 0x80 Size: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:13	RSVD	19 bits	Reserved.
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 Bandwidth Token Limit This field indicates the maximum number of bandwidth tokens that may be in use at one time by operations that access low bandwidth memory. This number of bandwidth tokens is shared by all descriptors accessing low bandwidth memory across the entire device. The default value is equal to the Total Bandwidth tokens reported in GRPCAP. The value in this field is used when the Use Global Bandwidth Token Limit field in any of the Group Configuration registers is 1. See section 4.4. If used, this value must be greater than 0. If the Global Bandwidth Token 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**
31:2	RSVD	30 bits	Reserved.
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.

<table>
<tr><td colspan="4">INTCAUSE
Base: BAR0 Offset: 0x98 Size: 4 bytes (32 bits)</td></tr>
<tr><th>Bits</th><th>Attr</th><th>Size</th><th>Description</th></tr>
<tr><td>31:5</td><td>RSVD</td><td>27 bits</td><td>Reserved.</td></tr>
<tr><td>4</td><td>RW1C</td><td>1 bit</td><td>Halt State</td></tr>
<tr><td>3</td><td>RW1C</td><td>1 bit</td><td>Perfmon Counter Overflow</td></tr>
<tr><td>2</td><td>RW1C</td><td>1 bit</td><td>WQ Occupancy Below Limit</td></tr>
<tr><td>1</td><td>RW1C</td><td>1 bit</td><td>Command Completion</td></tr>
<tr><td>0</td><td>RW1C</td><td>1 bit</td><td>Software Error</td></tr>
</table>

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.

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.

See section 3.12 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 the table of commands below for command codes. Undefined command codes are reserved.
19:0	WO	20 bits	Operand The meaning of this field depends on the command. See the table of commands below.

Command	Code	Operand	Operation
Enable Device	1	Unused	Enable the device.
Disable Device	2	Unused	Disable the device.
Drain All	3	Unused	Wait for all descriptors.
Abort All	4	Unused	Abandon and/or wait for all descriptors.
Reset Device	5	Unused	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	Request an interrupt handle for the specified interrupt table entry. If the Interrupt Handle Request capability in GENCAP is 0, this command code is reserved.

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.

[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.

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 any other command, this field is unused.
7:0	RO	8 bits	**Error Code** **0x00:** Successful completion. **0x01:** Invalid 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. **0x04-0x0f**: Unused. **0x10-0xff:** Command-specific error codes. See the table below.

Command-specific error codes

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: - A Group Configuration Register has one or more WQs and zero engines or has one or more engines and zero WQs. **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.

Command	Error codes
	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 Bandwidth Tokens configuration: - Invalid value for Use Global Bandwidth Token Limit or Global Bandwidth Token Limit in GENCFG. - Invalid value for Bandwidth Tokens Allowed or Bandwidth Tokens Reserved in GRPCFG.
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. 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.
Disable Device	0x31: Device is not Enabled.
Disable WQ Drain WQ Abort WQ Reset WQ	0x32: One or more of the specified WQs are not Enabled.
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.

Table 9-6: Administrative Command Error Codes

9.2.14 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 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 5.7 for the meaning of the value in this field.
	7	RO	1 bit	Unused.
	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. Otherwise, this field is unused.
	4	RO	1 bit	Batch 0: The descriptor was submitted directly. 1: The descriptor was submitted in a batch.
	3	RO	1 bit	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.

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

Byte offset	Bits	Attr	Size	Description
	2	RO	1 bit	Descriptor Valid 0: The descriptor that caused the error is unknown. The Batch, Operation, Batch Index, Priv, and PASID fields are unused. 1: The Batch, Operation, Batch Index, Priv, and PASID fields are valid.
	1	RW1C	1 bit	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.
	0	RW1C	1 bit	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.
15:8	63:32	RO	32 bits	Invalid flags If the Error Code field is Invalid flags, this field contains a bitmask of the flags that were found to be invalid. Otherwise this field is unused. 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.
	31:16	RO	16 bits	Unused.
	15:0	RO	16 bits	Batch Index If the Descriptor Valid field is 1 and the Batch field is 1, this field contains the index of the descriptor within the batch. Otherwise, this field is unused.
23:16	63:0	RO	64 bits	Address If the error is a page fault, this is the faulting address. Otherwise this field is unused.
31:24	63:0	RO	64 bits	Unused.

9.2.15 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.

<table>
<tr><td colspan="3">MSIXPERM
Base: BAR0</td><td>Offset: Table-offset + index × 8   Size: 4 bytes (32 bits)</td></tr>
<tr><td>Bits</td><td>Attr</td><td>Size</td><td>Description</td></tr>
<tr><td>31:12</td><td>RW</td><td>20 bits</td><td>PASID
If PASID Enable is 1, this field is checked against the PASID field of the descriptor. See section 5.4.</td></tr>
<tr><td>11:4</td><td>RSVD</td><td>8 bits</td><td>Reserved</td></tr>
<tr><td>3</td><td>RW</td><td>1 bit</td><td>PASID Enable
This field is checked against the WQ PASID Enable field of the WQ the descriptor was submitted to. See section 5.4.</td></tr>
<tr><td>2</td><td>RW</td><td>1 bit</td><td>Ignore
When this field is 1, no descriptor completion interrupts are generated using the corresponding MSI-X table entry; the Pending field is not set to 1, and the Mask field is ignored.
This field does not affect delivery of interrupts due to causes other than descriptor completion.</td></tr>
<tr><td>1:0</td><td>RSVD</td><td>2 bits</td><td>Reserved.</td></tr>
</table>

9.2.16 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. 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.

This register is 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: 4 bytes (32 bits)
Bits	Attr	Size	Description
31:28	RSVD	4 bits	Reserved.
27:20	RWL	8 bits	Bandwidth Tokens Allowed This field indicates the maximum number of bandwidth tokens 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 Bandwidth Tokens Reserved field for this group; and - less than or equal to the sum of the Bandwidth Tokens Reserved field and the number of non-reserved bandwidth tokens. (The number of non-reserved bandwidth tokens is the Total Bandwidth Tokens field in GRPCAP minus the total of the Bandwidth Tokens Reserved fields for all groups.) The default value of this field is the same as the value of the Total Bandwidth Tokens field in GRPCAP. If the Bandwidth Tokens 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	Bandwidth Tokens Reserved This field indicates the number of bandwidth tokens 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 bandwidth tokens reserved for all groups must be less than or equal to the Total Bandwidth Tokens field in GRPCAP. If the Bandwidth Tokens Supported field in GRPCAP is 0, this field is read-only and is unused.
7	RWL	1 bit	Use Global Bandwidth Token Limit 0: The Global Bandwidth Token Limit does not apply to this group. 1: The Global Bandwidth Token Limit programmed in the GENCFG register applies to descriptors processed by engines in this group. (The limit indicated by the Bandwidth Tokens Allowed field applies as well.) If the Global Bandwidth Token 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.

9.2.17 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.

Each 32-byte WQCFG register is divided into eight 32-bit sub-registers, which may also be read or written using aligned 64-bit 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.

	Configuration Support		
		0	
Field	**1**	**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	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

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 × 32 Size: 32 bytes (8 × 32 bits)

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.
	3:2	RSVD	2 bits	Reserved.

[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 × 32 Size: 32 bytes (8 × 32 bits)

Bytes	Bits	Attr	Size	Description
	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. This field must not be 0 and must not be greater than the Maximum Supported Batch 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.
	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.

WQCFG
Base: BAR0 Offset: Table-offset + WQ-ID × 32 Size: 32 bytes (8 × 32 bits)

Bytes	Bits	Attr	Size	Description
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 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.

WQCFG
Base: BAR0 Offset: Table-offset + WQ-ID × 32 Size: 32 bytes (8 × 32 bits)

Bytes	Bits	Attr	Size	Description
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.
	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.

[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 × 32 Size: 32 bytes (8 × 32 bits)
Bytes	Bits	Attr	Size	Description
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.
31:28	31:0	RSVD	32 bits	Reserved.

9.2.18 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. Details will be included in future revision of this document.

9.2.19 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 (Table 1-1: References) for details of this table. The offset and number of entries are in the MSI-X capability. The suggested number of entries is the number of WQs plus 1. See section 3.7 for information on how the MSI-X table is used.

9.2.20 MSI-X Pending Bit Array

BAR0; Offset: given by the MSI-X capability; Size: ⌈number of entries ÷ 64⌉ × 64 bits. (Please 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 (Table 1-1: References) for details of this table. The offset and number of entries are in the MSI-X capability.

9.2.21 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 When this field is 1, no descriptor completion interrupts are generated using this IMS entry; the Pending field is not set to 1, and the Mask field is ignored.
	1	RO	1 bit	Pending This field is set to 1 when a descriptor completion specifies 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.

9.3 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 9-1.

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 ENQCMD or ENQCMDS. Any other write operation to an SWQ portal is ignored. Descriptor submissions to a DWQ must be performed using a 64-byte write operation. Software should use MOVDIR64B, as that is the only instruction that is architecturally-defined to generate a non-broken 64-byte write. An ENQCMD or ENQCMDS 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 0xFF in all bytes. See section 5.3 for more information on error checking and reporting related to portal accesses.

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

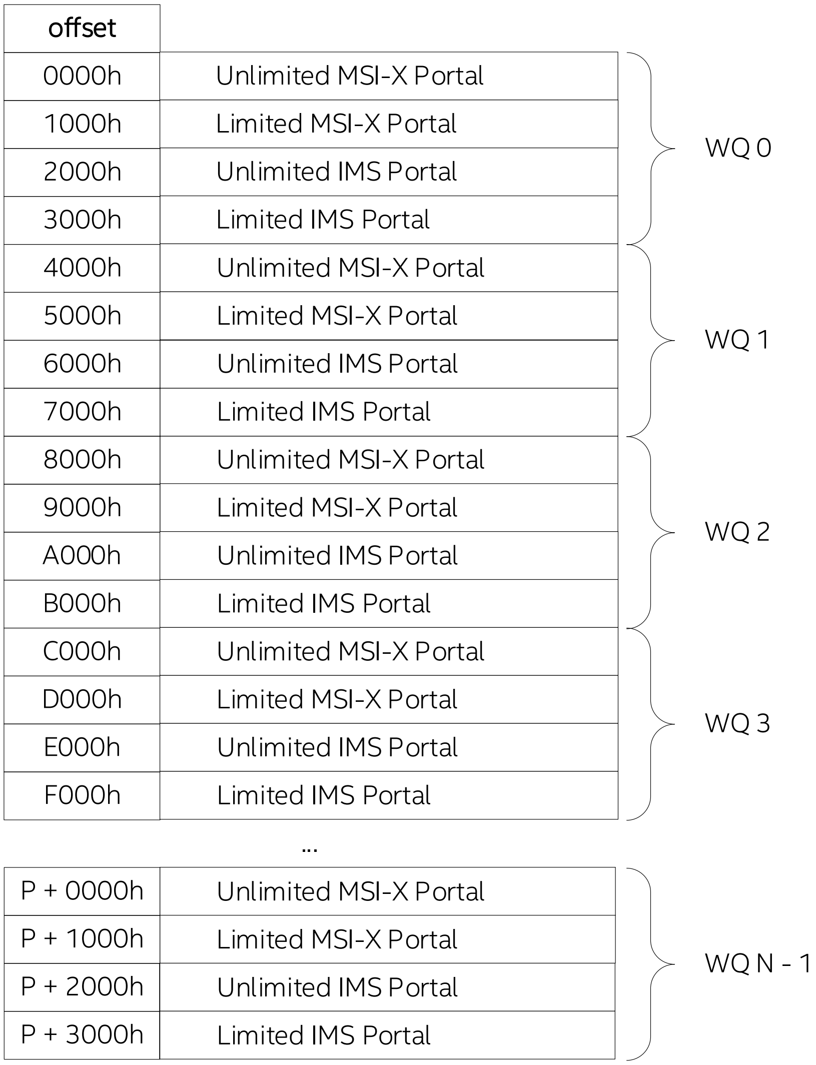
Table 9-8: Supported Portal Operations

Portal Virtualization

When Intel DSA is virtualized according to the Scalable I/O Virtualization architecture, as shown in Figure 7-1, the VMM maps only IMS portals into the guests. The MSI-X portals are reserved for host use. The IMS portals are mapped into the guest's virtual BAR2 in place of the MSI-X portals. No portals are mapped into the guests' virtual IMS portals since the virtual Intel DSA device visible to the guest does not report support for IMS.

For an SWQ available to a guest, the VMM maps the Limited IMS Portal into the guest's virtual BAR2 in place of both the Unlimited MSI-X Portal and the Limited MSI-X Portal. The VMM does not map unlimited portals for an SWQ into the guest, so that it can retain control of sharing of WQ space across guests.

For a WQ that is assigned to a single guest, the VMM maps the Unlimited IMS Portal into the guest's virtual BAR2 in place of the Unlimited MSI-X Portal and it maps the Limited IMS Portal into the guest in place of the Limited MSI-X Portal. The guest may then choose whether to use the WQ as an SWQ or a DWQ, and if it is used as an SWQ, the guest driver may use the unlimited portal itself to manage sharing of WQ space.



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

§