Single Root I/O Virtualization and Sharing Specification

Revision 1.0

September 11, 2007

Revision	Revision History	Date
1.0	Initial release.	9/11/2007

Membership Services
http://www.pcisig.com

E-mail:	administration@pcisig.com
Phone:	503-619-0569
Fax:	503-644-6708

Technical Support
techsupp@pcisig.com

DISCLAIMER

Contents

Figures

Tables

Objective of the Specification

The purpose of this document is to specify PCI™ I/O virtualization and sharing technology. The specification is focused on single root topologies; e.g., a single computer that supports virtualization technology.

This document is to be used in conjunction with, and does not supersede, the terms and conditions specified in the *PCI-SIG® Trademark and Logo Usage Guidelines* document.

Document Organization

Chapter 1 provides an architectural overview of single root I/O virtualization (SR-IOV) technology and the associated methods used in this specification.

Chapter 2 specifies SR IOV initialization and resource allocation.

Chapter 3 specifies SR IOV configuration.

Chapter 4 specifies SR IOV error handling.

Chapter 5 specifies interrupts.

Chapter 6 specifies event handling.

Chapter 7 specifies power management.

Documentation Conventions

Capitalization

Some terms are capitalized to distinguish their definition in the context of this document from their common English meaning. Words not capitalized have their common English meaning. When terms such as "memory write" or "memory read" appear completely in lower case, they include all transactions of that type.

Register names and the names of fields and bits in registers and headers are presented with the first letter capitalized and the remainder in lower case.

Numbers and Number Bases

Hexadecimal numbers are written with a lower case "h" suffix, e.g., 0FFFFh and 80h. Hexadecimal numbers larger than four digits are represented with a space dividing each group of four digits, as in 1E FFFF FFFFh. Binary numbers are written with a lower case "b" suffix, e.g., 1001b and 10b. Binary numbers larger than four digits are written with a space dividing each group of four digits, as in 1000 0101 0010b.

All other numbers are decimal.

Reference Information

Reference information is provided in various places to assist the reader and does not represent a requirement of this document. Such references are indicated by the abbreviation "(ref)." For example, in some places, a clock that is specified to have a minimum period of 400 ps also includes the reference information maximum clock frequency of "2.5 GHz (ref)." Requirements of other specifications also appear in various places throughout this document and are marked as reference information. Every effort has been made to guarantee that this information accurately reflects the referenced document; however, in case of a discrepancy, the original document takes precedence.

Implementation Notes

Implementation Notes should not be considered to be part of this specification. They are included for clarification and illustration only.

Terms and Abbreviations

ARI	Alternative Routing ID Interpretation as per the *PCI Express Base Specification*.
FLR	Function Level Reset as per the *PCI Express Base Specification*.
Multi-Root I/O Virtualization (MR-IOV)	A Function that supports the MR-IOV capability. See the *Multi-Root I/O Virtualization and Sharing Specification* for additional information.
PCIe®	PCI Express®
Physical Function (PF)	A PCI Function that supports the SR-IOV capabilities defined in this specification. A PF contains the SR-IOV capability structure.
RC	Root Complex per the *PCI Express Base Specification*
RP	Root Port per the *PCI Express Base Specification*
Routing ID	Either the Requester ID or Completer ID that identifies a PCI Express Function.
System Image (SI)	A software component running on a virtual system to which specific Functions, PF, and VF can be assigned. Specification of the behavior and architecture of an SI is outside the scope of this specification. Examples of SIs include guest operating systems and shared/non-shared protected domain device drivers.
Single Root I/O Virtualization (SR-IOV)	A function that supports the SR-IOV capability defined in this specification.
SR-IOV Capability	The SR-IOV capability structure is used to discover and configure a PF's virtualization capabilities. These virtualization capabilities include the number of Virtual Functions (VF) the PCIe Device will associate with a PF and the type of BAR mechanism supported by those VFs.
Single Root PCI Manager (SR-PCIM)	Software responsible for configuration and management the SR-IOV capability and PF/VF as well as dealing with associated error handling. Multiple implementation options exist; therefore, SR-PCIM implementation is outside the scope of this specification.

Virtual Function (VF)	A Function that is associated with a Physical Function. A VF shares one or more physical resources, such as a Link, with the Physical Function and other VFs that are associated with the same PF.
Virtualization Intermediary (VI)	A software component supporting one or more SIs–colloquially known as a hypervisor or virtual machine monitor. Specification of the behavior and architecture of the VI is outside the scope of this specification.

Reference Documents

PCI Local Bus Specification, Revision 3.0

PCI Bus Power Management Interface Specification, Revision 1.2

PCI Express Base Specification, Revision 1.1

PCI Express Base Specification, Revision 2.0

Multi-Root I/O Virtualization and Sharing Specification, Revision 1.0

1. Architectural Overview

Within the industry, significant effort has been expended to increase the effective hardware resource utilization (i.e., application execution) through the use of virtualization technology. The *Single Root I/O Virtualization and Sharing Specification* (SR-IOV) defines extensions to the PCI Express (PCIe) specification suite to enable multiple System Images (SI) to share PCI hardware resources.

To illustrate how this technology can be used to increase effective resource utilization, consider the generic platform configuration illustrated in Figure 1-1.

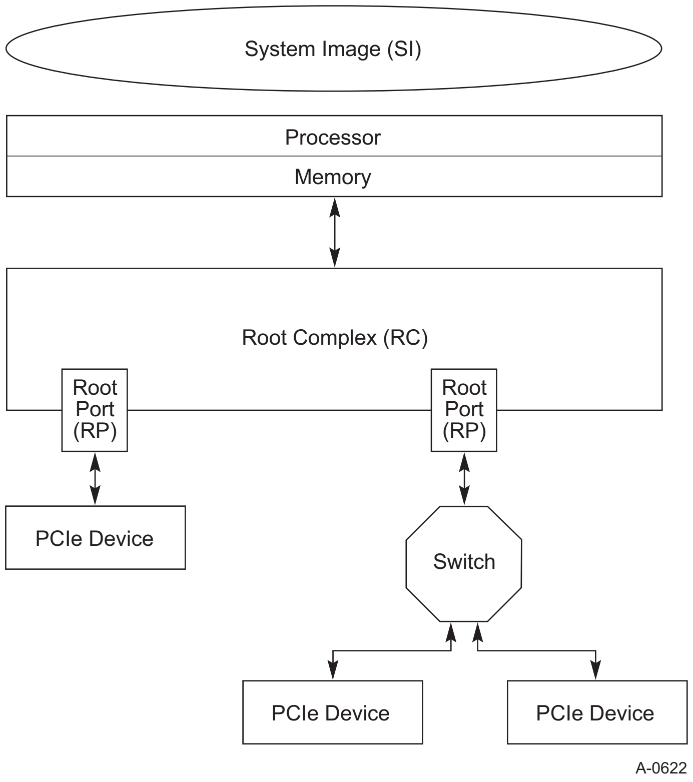

Figure 1-1: Generic Platform Configuration

The generic platform configuration is composed of the following components:

- ❑ Processor – general purpose, embedded, or specialized processing element
- ❑ Memory – general purpose or embedded
- ❑ PCIe Root Complex (RC) – one or more RC can be supported per platform

- ❑ PCIe Root Port (RP) – one or more RP can be supported per RC. Each RP represents a separate hierarchy per the *PCI Express Base Specification.* Each hierarchy is referred to a single root hierarchy to delineate it from the multiple hierarchy technology defined within the *Multi Root I/O Virtualization Specification.*
- ❑ PCIe Switch – provides I/O fan-out and connectivity
  - PCIe Device – multiple I/O device types, e.g., network, storage, etc.
  - System Image – software such an operating system that is used to execute applications or trusted services, e.g., a shared or non-shared I/O device driver.

In order to increase the effective hardware resource utilization without requiring hardware modifications, multiple SI can be executed. Software termed a Virtualization Intermediary (VI) is interposed between the hardware and the SI as illustrated in Figure 1-2.

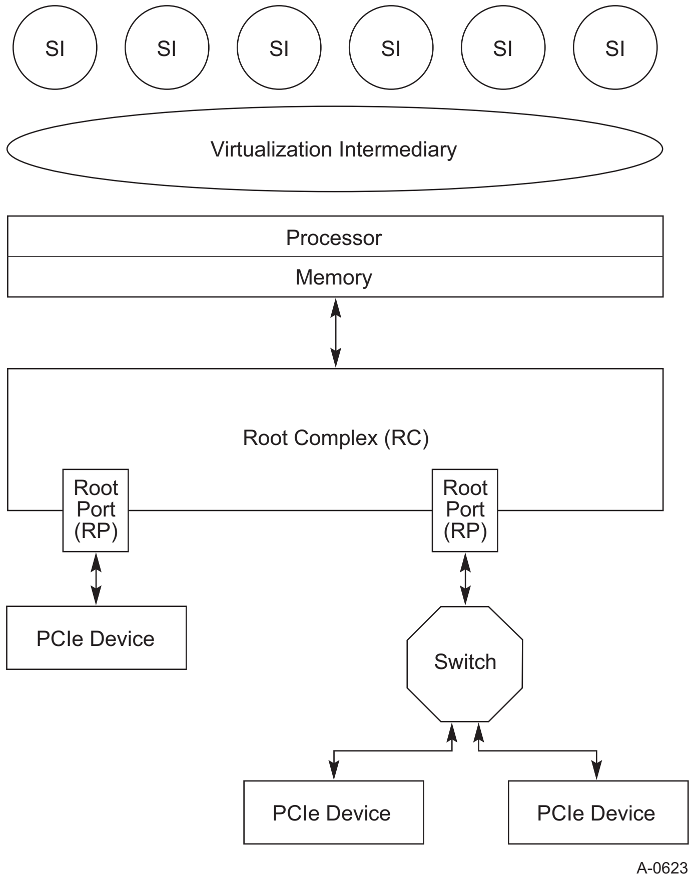

Figure 1-2: Generic Platform Configuration with a VI and Multiple SI

The VI takes sole ownership of the underlying hardware. Using a variety of methods outside of the scope of this specification, the VI abstracts the hardware to present each SI with its own virtual system. The actual hardware resources available to each SI can vary based on workload or customer-specific policies. While this approach works well for many environments, I/O intensive workloads can suffer significant performance degradation. Each I/O operation – inbound or outbound – must be intercepted and processed by the VI adding significant platform resource overhead.

To reduce platform resource overhead, PCI-SIG® developed the SR-IOV technology contained within this specification. The benefits of SR-IOV technology are:

- ❑ The ability to eliminate VI involvement in main data movement actions – DMA, Memory space access, interrupt processing, etc. Elimination of VI interception and processing of each I/O operation can provide significant application and platform performance improvements.
- ❑ Standardized method to control SR-IOV resource configuration and management through Single Root PCI Manager (SR-PCIM).
  - Due to a variety of implementation options – system firmware, VI, operating system, I/O drivers, etc. – SR-PCIM implementation is outside the scope of this specification.
- ❑ The ability to reduce the hardware requirements and associated cost with provisioning potentially a significant number of I/O Functions within a device.
- ❑ The ability to integrate SR-IOV with other I/O virtualization technologies such as Address Translation Services (ATS), Address Translation and Protection Table (ATPT) technologies, and interrupt remapping technologies to create a robust, complete I/O virtualization solutions.

Figure 1-3 illustrates an example SR-IOV capable platform.

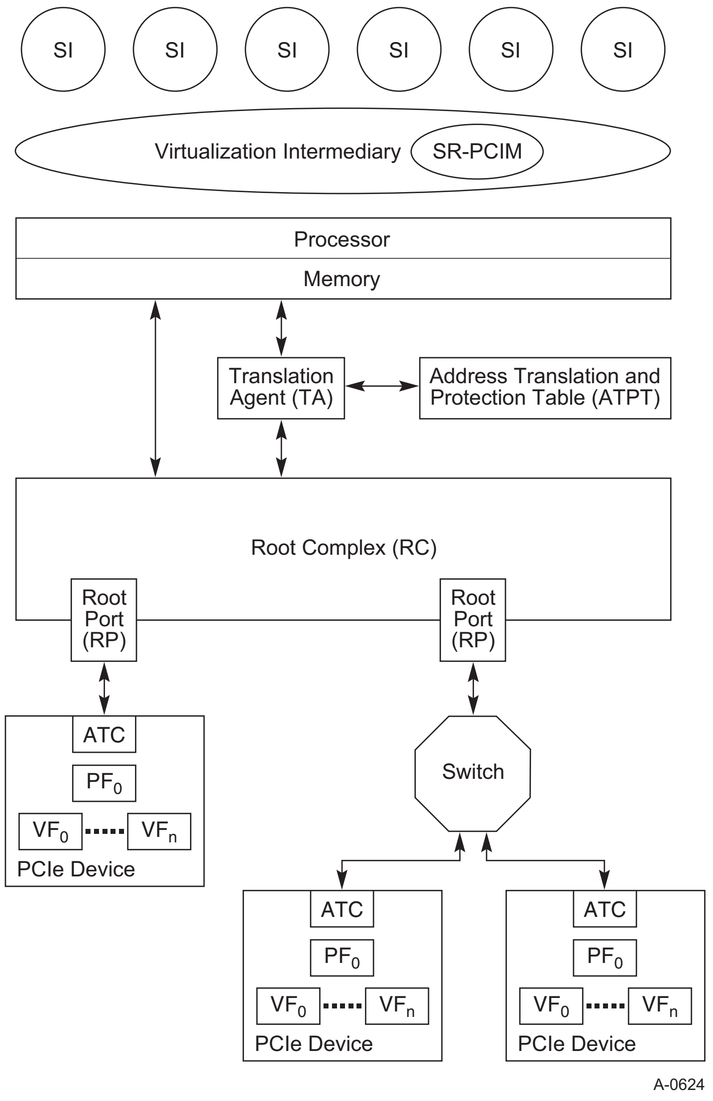

Figure 1-3: Generic Platform Configuration with SR-IOV and IOV Enablers

The SR-IOV generic platform configuration is composed of the following additional functional elements:

- ❑ Single Root PCI Manager (SR-PCIM) – Software responsible for the configuration of the SR-IOV capability, management of Physical Functions and Virtual Functions, and processing of associated error events and overall device controls such as power management and hot-plug services.

- ❑ Optional Translation Agent (TA) – A TA is hardware or a combination of hardware and software responsible for translating an address within a PCIe transaction into the associated platform physical address. A TA may contain an Address Translation Cache (ATC) to accelerate translation table access. A TA may also support the PCI-SIG *Address Translation Services Specification* which enables a PCIe Function to obtain address translations *a priori* to DMA access to the associated memory. See the associated specification for more details on ATS benefits and operation.

- ❑ Optional Address Translation and Protection Table (ATPT) – An ATPT contains the set of address translations accessed by a TA to process PCIe requests – DMA Read, DMA Write, or interrupt requests. See the *Address Translation Services Specification* for additional details.
    - In PCIe, interrupts are treated as memory write operations. Through the combination of a Requester Identifier and the address contained within a PCIe transaction, an interrupt can be routed to any target (e.g., a processor core) transparent to the associated I/O Function.
    - DMA Read and Write requests are translated through a combination of the Routing ID and the address contained within a PCIe transaction.
- ❑ Optional Address Translation Cache (ATC) – An ATC can exist in two locations within a platform – within the TA which can be integrated within or sit above an RC – or within a PCIe Device. Within an RC, the ATC enables accelerated translation look ups to occur. Within a Device, the ATC is populated through ATS technology. PCIe transactions that indicate they contain translated addresses may bypass the platform's ATC in order to improve performance without compromising the benefits associated with ATPT technology. See the *Address Translation Services Specification* for additional details.
- ❑ Physical Function (PF) – A PF is a PCIe Function (per the *PCI Express Base Specification*) that supports the SR-IOV capability and is accessible to an SR-PCIM, a VI, or an SI.
- ❑ Virtual Function (VF) – A VF is a "light-weight" PCIe Function (per this specification) that is directly accessible by an SI.
    - Minimally, resources associated with the main data movement of the Function are available to the SI. Configuration resources should be restricted to a trusted software component such as a VI or SR-PCIM.
    - A VF can be serially shared by different SI, i.e., a VF can be assigned to one SI and then reset and assigned to another SI.
    - A VF can be optionally migrated from one PF to another PF. The migration process itself is outside the scope of this specification but is facilitated through configuration controls defined within this specification.
- ❑ All VFs associated with a PF must be the same device type as the PF, e.g., the same network device type or the same storage device type.

To compare and contrast a PCIe Device with a PCIe SR-IOV capable device, examine the following set of figures. Figure 1-4 illustrates an example PCIe Device compliant with the *PCI Express Base Specification.*

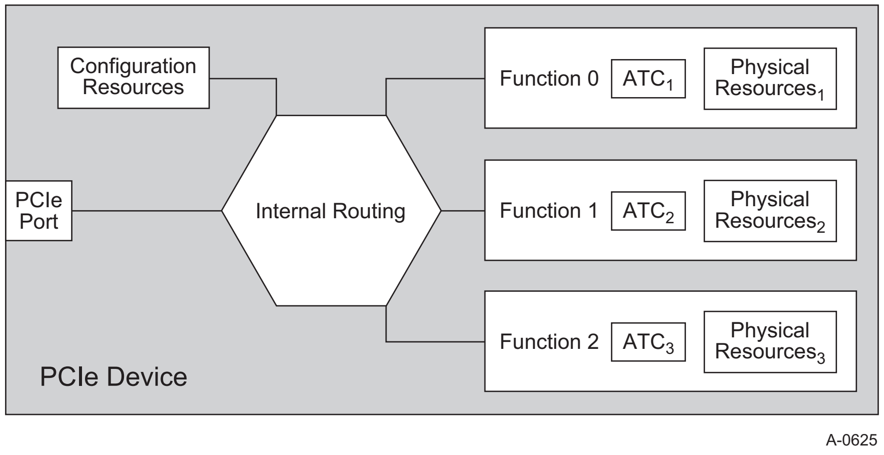

Figure 1-4: Example Multi-Function Device

This figure illustrates an example multi-Function PCIe Device with the following characteristics:

- ❑ The PCIe Device shares a common PCIe Link. Per the *PCI Express Base Specification*, the Link and PCIe functionality shared by all Functions is managed through Function 0.
    - While this figure illustrates only three Functions, with the use of the Alternative Routing Identifier (ARI) capability, a PCIe Device can support up to 256 Functions.
    - All Functions use a single Bus Number captured through the PCI enumeration process.
- ❑ In this example, each PCIe Function supports the ATS capability and therefore has an associated ATC to manage ATS obtained translated addresses.
- ❑ Each PCIe Function has a set of unique physical resources including a separate configuration space and BAR.
- ❑ Each PCIe Function can be assigned to an SI. To prevent one SI from impacting another, all PCIe configuration operations should be intercepted and processed by the VI.

As this figure illustrates, the hardware resources scale with the number of Functions provisioned. Depending upon the complexity and size of the device, the incremental cost per Function will vary. To reduce the incremental hardware cost, a device can be constructed using SR-IOV to support a single PF and multiple VFs as illustrated in Figure 1-5.

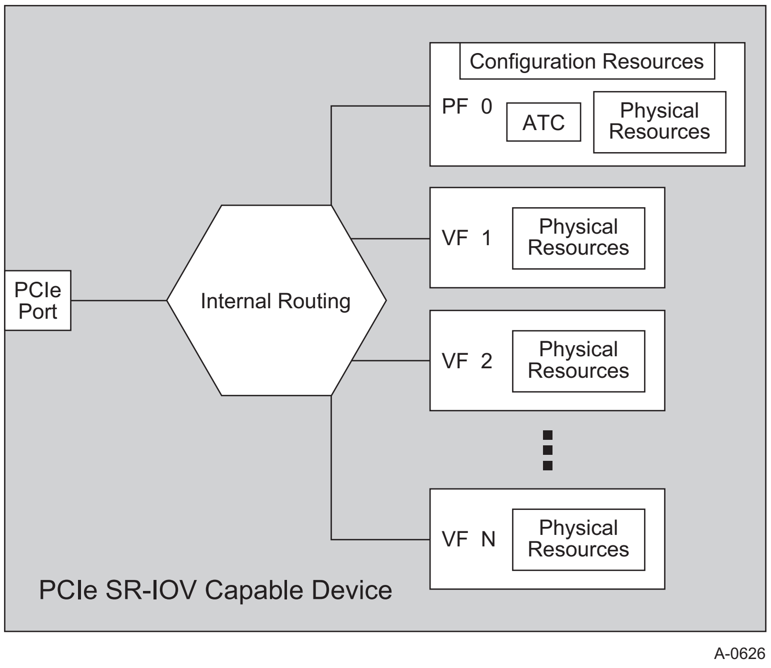

Figure 1-5: Example SR-IOV Single PF Capable Device

This example illustrates a single PF with three VFs. Key observations to note:

❑ The PF is a Function compliant with the *PCI Express Base Specification*.

- Initially and after a conventional reset, a PCIe Function that supports the SR-IOV capabilities defined in this specification shall have the SR-IOV capabilities disabled.
- To discover the page sizes supported by a PF and its associated VF, the Supported Page Sizes configuration field is read. For additional information on how this field can be used to align PF or VF Memory space apertures on a system page boundary, see Section 2.1.1.1.

❑ Each VF shares a number of common configuration space fields with the PF; i.e., where the fields are applicable to all VF and controlled through a single PF. Sharing reduces the hardware resource requirements to implement each VF.

- A VF uses the same configuration mechanisms and header types as a PF.
- All VFs associated with a given PF share a VF BAR set (see Section 3.3.14) and share a VF MSE bit in the SR-IOV extended capability (see Section 3.3.3.4) that controls access to the VF Memory space. That is, if the VF MSE bit is Clear, the memory mapped space allocated for all VFs is disabled.
- The InitialVFs and TotalVFs fields (see Section 3.3.6 and Section 3.3.5) are used to discover the maximum number of VFs that can be associated with a PF.
- If the Device does not support VF migration, TotalVFs and InitialVFs shall contain the same value. If the Device supports VF migration, when TotalVFs is read, the PF must return the number of VFs that can be assigned to the PF. For such a Device, when InitialVFs is read, the PF must return the initial number of VFs assigned to the PF.

- Each Function, PF, and VF is assigned a unique Routing ID. A Routing ID is constructed per default mechanism within the *PCI Express Base Specification* or through the mechanism enabled via the ARI capability.
- All PCIe and SR-IOV configuration access is assumed to be through a trusted software component such as a VI or an SR-PCIM.
- Each VF contains a non-shared set of physical resources required to deliver Function-specific services, e.g., resources such as work queues, data buffers, etc. These resources can be directly accessed by an SI without requiring VI or SR-PCIM intervention.
- One or more VF may be assigned to each SI. Assignment policies are outside the scope of this specification.
- While this example illustrates a single ATC within the PF, the presence of any ATC is optional. In addition, this specification does not preclude an implementation from supporting an ATC per VF within the Device.
- Internal routing is implementation specific.
- While many potential usage models exist regarding PF operation, a common usage model is to use the PF to bootstrap the device or platform strictly under the control of a VI. Once the SR-IOV capability is configured enabling VF to be assigned to individual SI, the PF takes on a more supervisory role. For example, the PF can be used to manage device-specific functionality such as internal resource allocation to each VF, VF arbitration to shared resources such as the PCIe Link or the Function-specific Link (e.g., a network or storage Link), etc. These policy, management, and resource allocation operations are outside the scope of this specification.

Another example usage model is illustrated in Figure 1-6. In this example, the device supports multiple PFs each with its own set of VFs.

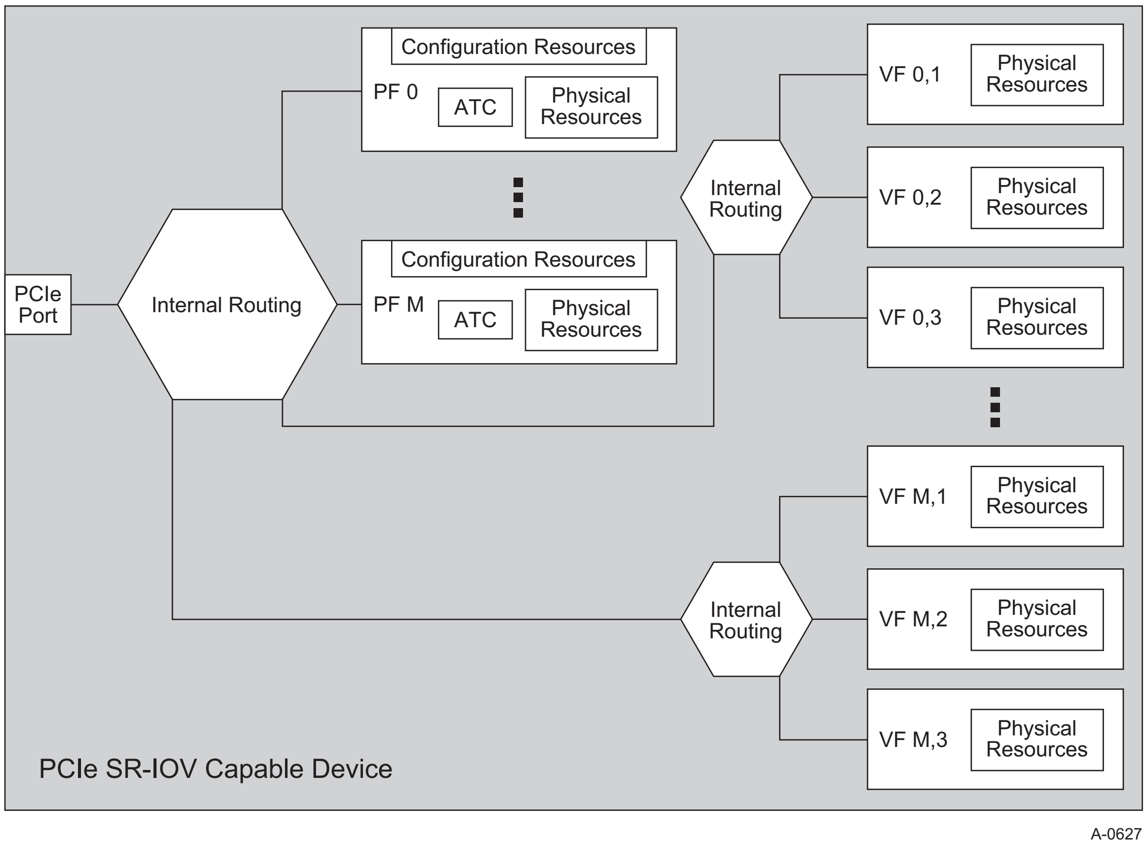

Figure 1-6: Example SR-IOV Multi-PF Capable Device

Key observations to note:

- ❑ Each PF can be assigned zero or more VFs. The number of VFs per PF is not required to be identical for all PFs within the device.
- ❑ The ARI capability enables Functions to be assigned to Function Groups and defines how Function Group arbitration can be configured. PFs and VFs can be assigned to Function Groups and take advantage of the associated arbitration capabilities. Within each Function Group, though, arbitration remains implementation specific.
- ❑ Internal routing between PFs and VFs is implementation specific.
- ❑ For some usage models, all PFs may be the same device type; e.g., all PFs deliver the same network device or all deliver the same storage device functionality. For other usage models, each PF may be represent a different device type; e.g., in Figure 1-6, one PF might represent a network device while another represents an encryption device.
  - In situations where there is a usage model dependency between device types, such as for each VF that is a network device type, each SI also requires a VF that is an encryption device type. The SR-IOV capability provides a method to indicate these dependencies. The policies used to construct these dependencies as well as assign dependent sets of VF to a given SI are outside the scope of this specification.

- VF nomenclature **VF *M,N*** designates the Nth VF associated with PF M. The Routing ID value for each VF is determined using the PF's Routing ID value and fields in the PF's SR-IOV Capability.

As seen in the prior example, the number of PF and VF can vary based on usage model requirements. To support a wide range of options, an SR-IOV Device can support the following number and mix of PF and VF:

- Using the Alternative Routing Identifier (ARI) capability, a device may support up to 256 PFs. As with the *PCI Express Base Specification*, Function Number assignment is implementation specific and may be sparse throughout the 256 Function Number space.
- A PF can only be associated with the Device's captured Bus Number as illustrated in Figure 1-7.
- SR-IOV Devices may consume more than one Bus Number. A VF can be associated with any Bus Number within the Device's Bus Number range – the captured Bus Number plus any additional Bus Numbers configured by software. See Section 2.1.2 for details.
    - Use of multiple Bus Numbers enables a device to support a very large number of VFs – up to the size of the Routing ID space minus the bits used to identify intervening busses.
    - If software does not configure sufficient additional Bus Numbers, then the VFs implemented for the additional Bus Numbers may not be visible.

IMPLEMENTATION NOTE

Function Co-location

The ARI capability enables a Device to support up to 256 Functions – Functions, PFs, or VFs in any combination – associated with the captured Bus Number. If a usage model does not require more than 256 Functions, implementations are strongly encouraged to co-locate all Functions, PFs, and VFs within the captured Bus Number and not require additional Bus Numbers to access VFs.

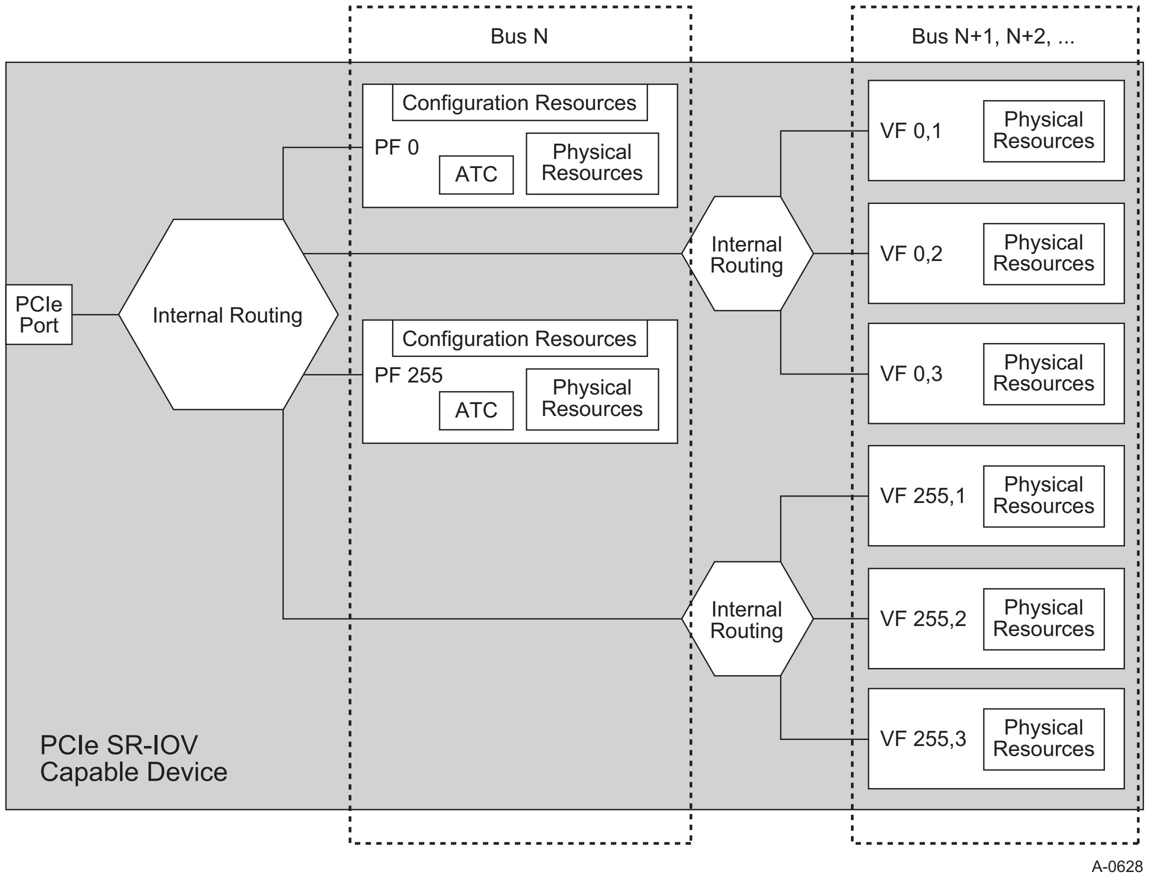

Figure 1-7: Example SR-IOV Device with Multiple Bus Numbers

In this last example, Figure 1-8, a device implementation may mix any number of Functions, PFs, and VFs.

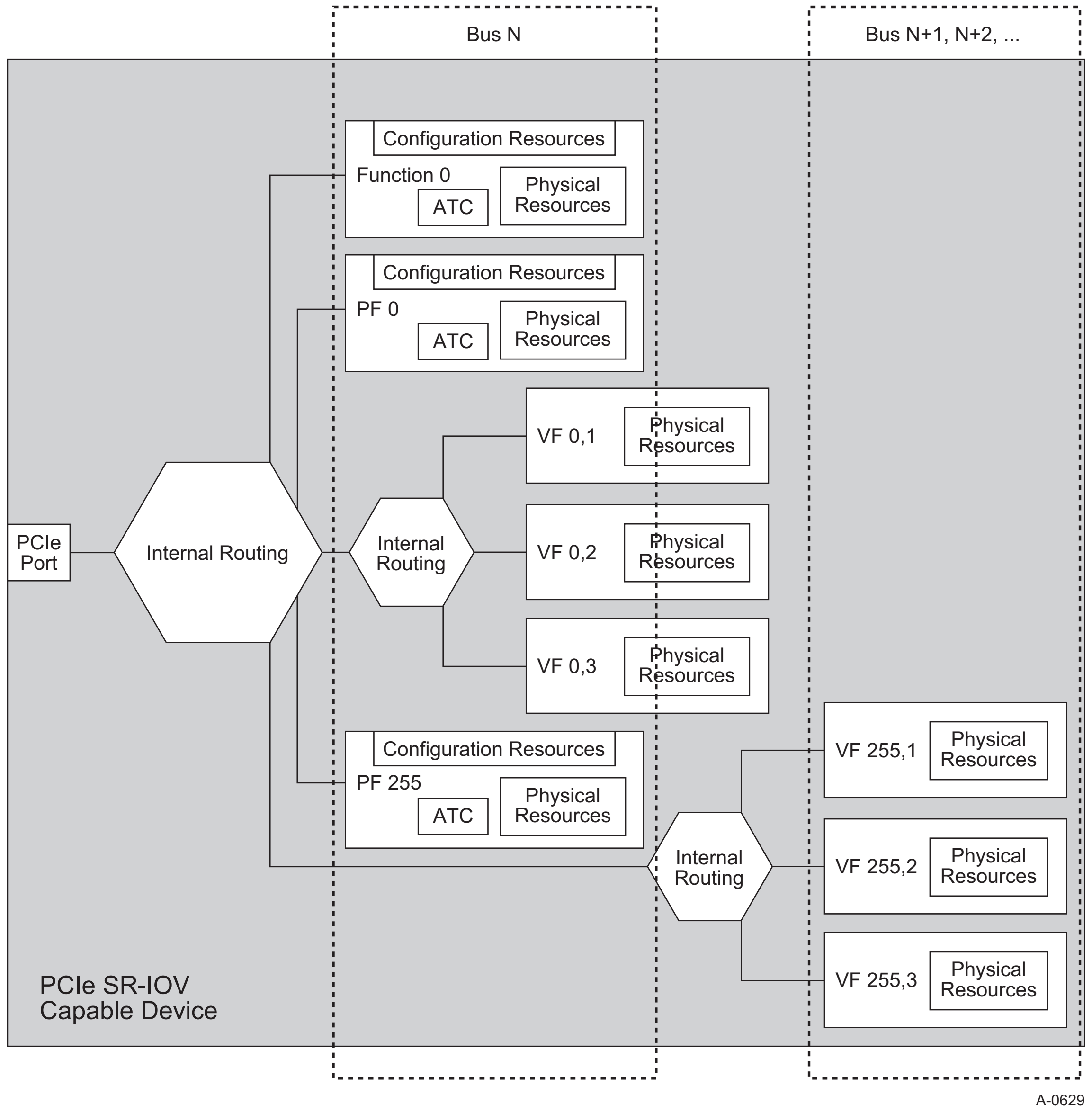

Figure 1-8: Example SR-IOV Device with a Mixture of Function Types

Key observations to note:

❑ Each Device must contain a Function 0. Function 0 may be a PF (i.e., it may include the SR-IOV extended capability).

❑ Any mix of Functions can be associated with the captured Bus Number.

- Non-VFs can only be associated with the captured Bus Number.

❑ If the ARI capability is supported, Functions can be assigned to Function Groups. The assignment policy is outside the scope of this specification. If the ARI capability is not supported, Functions can still use the Function arbitration capabilities as defined in the *PCI Express Base Specification.*

1.1. PCI Technologies Interoperability

Establishing clear interoperability requirements is critical to the success of any technology. To this end, the PCI-SIG I/O virtualization specifications are organized to maximize the interoperability potential for compliant implementations. Conceptually, this is viewed as a set of concentric circles that define how functionality is layered to build an IOV capable component as illustrated in Figure 1-9.

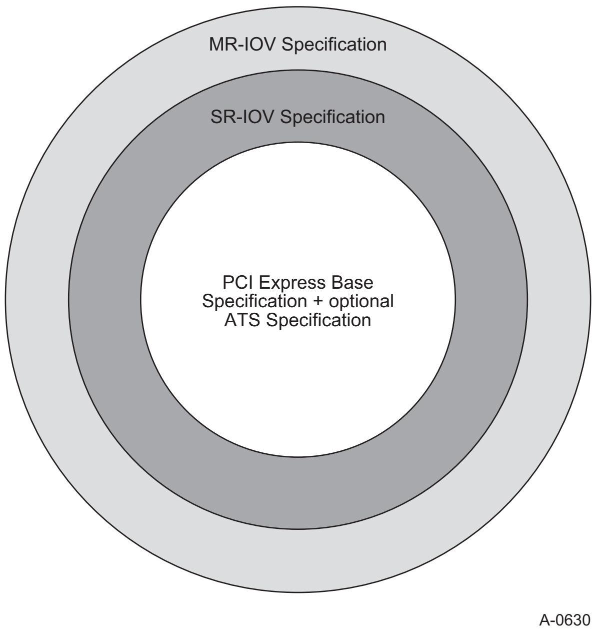

Figure 1-9: I/O Virtualization Interoperability

Key observations to note:

❑ At their core, I/O virtualization specifications build upon the *PCI Express Base Specification*. All IOV implementations must be compliant with the *PCI Express Base Specification, Revision 1.1* or later versions. Where applicable, the IOV specifications note relevant deltas between these versions.

- None of the IOV specifications touch the physical layer.
- The *Single Root I/O Virtualization and Sharing Specification* does not touch the data Link or transaction layers specified in the *PCI Express Base Specification*.
- The *Multi-Root I/O Virtualization and Sharing Specification* does not touch the transaction layer specified in the *PCI Express Base Specification*.
- All I/O virtualization capabilities are communicated through new PCI Express capabilities implemented in PCI Express extended configuration space.
- I/O virtualization specifications have no impact on PCI or PCI-X specifications.

- A hierarchy can be composed of a mix of PCI Express and PCI Express-to-PCI/PCI-X Bridges.
  - ♦ A PCI Express-to-PCI/PCI-X Bridge and PCI/PCI-X Devices can be serially shared by multiple SIs.

❑ The *Address Translation Specification* defines optional functionality applicable to any Function. ATS can be supported in SR-IOV components.

❑ To implement an SR-IOV Device, the *Single Root I/O Virtualization and Sharing Specification* requires the Device to be fully compliant with the *PCI Express Base Specification*.

- A hierarchy can be composed of a mix of SR IOV and non-SR IOV components. For example, a hierarchy may contain any mix of SR IOV and non-SR IOV Endpoint Devices.

❑ To implement an MR-IOV Device, the *Multi-Root I/O Virtualization and Sharing Specification* requires the Device to be fully compliant with the *Single Root I/O Virtualization and Sharing Specification* as well as the *PCI Express Base Specification*.

2

2. Initialization and Resource Allocation

2.1. SR-IOV Resource Discovery

The following sections describe how software determines that a Device is SR-IOV capable and subsequently identifies VF resources through Virtual Function Configuration Space.

2.1.1. Configuring SR-IOV Capabilities

This section describes the fields that must be configured before enabling a PF's IOV Capabilities. The VFs are enabled by Setting the PF's VF Enable bit (see Section 0) in the SR-IOV extended capability.

The NumVFs field (see Section 3.3.7) defines the number of VFs that are enabled when VF Enable is Set in the associated PF.

2.1.1.1. Configuring the VF BAR Mechanisms

This section describes how the VF BARs are configured to map memory space. VFs do not support I/O Space and thus VF BARs shall not indicate I/O Space.

The System Page Size field (see Section 3.3.13) defines the page size the system will use to map the VF's PCIe memory addresses when the PF's IOV Capabilities are enabled. The System Page Size field is used by the PF to align the Memory space aperture defined by each VF BAR to a system page boundary. The value chosen for the System Page Size must be one of the Supported Page Sizes (see Section 3.3.12) in the SR-IOV extended capability.

The behavior of VF BARs is the same as the *PCI Local Bus Specification, Revision 3.0* normal PCI Memory Space BARs, except that a VF BAR describes the aperture for each VF, whereas a PCI BAR describes the aperture for a single Function:

- ❑ The behavior described in the *PCI Local Bus Specification, Revision 3.0* for determining the memory aperture of a Function's BAR applies to each VF BAR. That is, the size of the memory aperture required for each VF BAR can be determined by writing all "1"s and then reading the VF BAR. The results read back must be interpreted as described in the *PCI Local Bus Specification, Revision 3.0*.
- ❑ The behavior for assigning the starting memory space address of each BAR associated with the first VF is also as described in the *PCI Local Bus Specification, Revision 3.0*. That is, the address written into each VF BAR is used by the Device for the starting address of the first VF.

- ❑ The difference between VF BARs and *PCI Local Bus Specification, Revision 3.0* BARs is that for each VF BAR, the memory space associated with the second and higher VFs is derived from the starting address of the first VF and the memory space aperture. For any given VF (*v*), the starting address of its Memory space aperture for any implemented BAR (*b*) is calculated according to the following formula:

  $$\text{BAR}_b\ \text{VF}_v \text{ starting address} = \text{VF BAR}_b + (v - 1) \times (\text{VF BAR}_\text{b} \text{ aperture size})$$

  where VF BAR_b aperture size is the size of VF BAR_b as determined by the usual BAR probing algorithm as described in Section 3.3.14.

  VF memory space is not enabled until both VF Enable and VF MSE have been Set (see Section 3.3.3.1 and Section 3.3.3.4). Note that changing System Page Size (see Section 3.3.13) may affect the VF BAR aperture size.

Figure 2-1 shows an example of the PF and VF Memory space apertures.

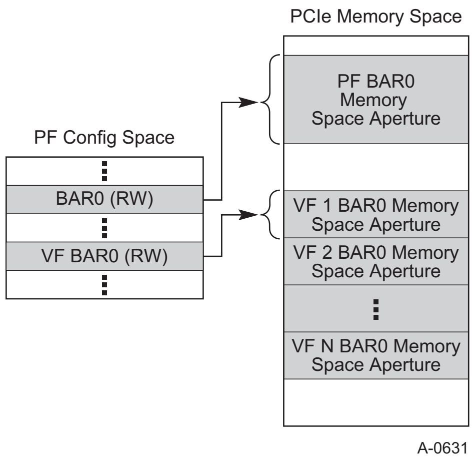
PCIe Memory Space
PF BAR0 Memory Space Aperture
PF Config Space
BAR0 (RW)
VF BAR0 (RW)
VF 1 BAR0 Memory Space Aperture
VF 2 BAR0 Memory Space Aperture
VF N BAR0 Memory Space Aperture
A-0631

Figure 2-1: BAR Space Example for Single BAR Device

2.1.2. VF Discovery

The First VF Offset and VF Stride fields in the SR-IOV extended capability are 16-bit Routing ID offsets. These offsets are used to compute the Routing IDs for the VFs with the following restrictions:

- ❑ The value in NumVFs (Section 3.3.7) affects the values in First VF Offset (Section 3.3.9) and VF Stride (Section 3.3.10).
- ❑ The value in ARI Capable Hierarchy (Section 3.3.3.5) in the lowest numbered PF of the Device (for example PF_0) may affect the values in First VF Offset and VF Stride in all PFs of the Device.
- ❑ NumVFs and ARI Capable Hierarchy may only be changed when VF Enable (Section 3.3.3.1) is Clear.

IMPLEMENTATION NOTE

NumVFs and ARI Capable Hierarchy

After configuring NumVFs and ARI Capable Hierarchy, software may read First VF Offset and VF Stride to determine how many Bus Numbers would be consumed by the PF's VFs. The additional Bus Numbers, if any, are not actually used until VF Enable is Set.

Table 2-1 describes the algorithm used to determine the Routing ID associated with each VF.

Table 2-1: VF Routing ID Algorithm

VF Number	VF Routing ID
VF 1	(PF Routing ID + VF Offset) Modulo 2^{16}
VF 2	(PF Routing ID + VF Offset + VF Stride) Modulo 2^{16}
VF 3	(PF Routing ID + VF Offset + 2 * VF Stride) Modulo 2^{16}
...	...
VF N	(PF Routing ID + VF Offset + (N-1) * VF Stride) Modulo 2^{16}
...	...
VF NumVFs (last one)	(PF Routing ID + VF Offset + (NumVFs-1) * VF Stride) Modulo 2^{16}

All arithmetic used in this Routing ID computation is 16-bit unsigned dropping all carries.

All VFs and PFs must have distinct Routing IDs. The Routing ID of any PF or VF must not overlap with the Routing ID of any other PF or VF given any valid setting of NumVFs across all PFs of a device.

VF Stride and First VF Offset are constants. Their values may be affected by NumVFs and the ARI Capable Hierarchy bit, but their values are not affected by other settings in this PF or any other PF of the Device.

VFs may reside on different Bus Number(s) than the associated PF. This can occur if, for example, First VF Offset has the value 0100h. A VF shall not be located on a Bus Number that is numerically smaller than its associated PF.

As in the *PCI Express Base Specification*, SR-IOV capable Devices capture the Bus Number from any Type 0 Configuration Request. SR-IOV capable Devices do not capture the Bus Number from any Type 1 Configuration Requests.

Switch processing of Bus Numbers is unchanged from base PCIe. In base PCIe, the switch sends all Configuration Requests in the range Secondary Bus Number (inclusive) to Subordinate Bus Number (inclusive) to the Device. Type 1 Configuration Requests addressing the Secondary Bus Number are converted into Type 0 Configuration Requests while Type 1 Configuration Requests addressing Bus Numbers between Secondary Bus Number (exclusive) and Subordinate Bus Number (inclusive) are forwarded on to the Device as Type 1 Configuration Requests.

Note: Bus Numbers are a constrained resource. Devices are strongly encouraged to avoid leaving "holes" in their Bus Number usage to avoid wasting Bus Numbers.

All PFs must be located on the Device's captured Bus Number.

IMPLEMENTATION NOTE

VFs Spanning Multiple Bus Numbers

As an example, consider an SR-IOV Device that supports a single PF. Initially, only PF 0 is visible. Software Sets ARI Capable Hierarchy. From the SR-IOV Capability it determines: InitialVFs is 600, First VF Offset is 1 and VF Stride is 1.

- ❑ If software sets NumVFs in the range [0 .. 255], then the Device uses a single Bus Number.
- ❑ If software sets NumVFs in the range [256 .. 511], then the Device uses two Bus Numbers.
- ❑ If software sets NumVFs in the range [512 .. 600], then the Device uses three Bus Numbers.

PF 0 and VF 0,1 through VF 0,255 are always on the first (captured) Bus Number. VF 0,256 through VF 0,511 are always on the second Bus Number (captured Bus Number plus 1). VF 0,512 through VF 0,600 are always on the third Bus Number (captured Bus Number plus 2).

Software should configure Switch Secondary and Subordinate Bus Number fields to route enough Bus Numbers to the Device. If sufficient Bus Numbers are not available, software should reduce a Device's Bus Number requirements by not enabling SR-IOV and/or reducing NumVFs for some or all PFs of the Device prior to enabling SR-IOV.

After VF Enable is Set in some PF *n*, the Device must Enable VF *n*,1 through VF *n*,NumVFs (inclusive). Each Enabled VF must respond to PCIe transactions that target VF Configuration space fields defined in Sections 3.4 to 3.7. Additionally, if VF MSE is Set, each Enabled VF must respond to PCIe Memory transactions addressing the memory space associated with that VF.

Functions that are not enabled (i.e., Functions for VFs above NumVFs) do not exist in the PCI Express fabric. As in the *PCI Express Base Specification*, Functions that do not exist respond with Unsupported Request (UR). This includes Functions on additional Bus Numbers.

2.1.3. Function Dependency Lists

PCI Devices may have vendor specific dependencies between Functions. For example, Functions 0 and 1 might provide different mechanisms for controlling the same underlying hardware. In such situations, the Device programming model might require that these dependent Functions be assigned to SIs as a set.

Function Dependency Lists are used to describe these dependencies (or to indicate that there are no Function dependencies). Software should assign PFs and VFs to SIs such that the dependencies are satisfied.

See Section 3.3.8 for details.

2.1.4. Interrupt Resource Allocation

PFs and VFs support either MSI, MSI-X interrupts, or both if interrupt resources are allocated. VFs shall not implement INTx. Interrupts are described in Chapter 5.

2.2. Reset Mechanisms

This section describes how PCI base defined reset mechanisms affect Devices that support SR-IOV. It also describes the mechanisms used to reset a single VF and a single PF with its associated VFs.

2.2.1. Conventional Reset

A Conventional Reset to a Device that supports this specification shall cause all Functions (including both PFs and VFs) to be reset to their original, power-on state per the rules defined in the *PCI Express Base Specification*. Chapter 3 describes this behavior for fields defined in this specification.

Note: Conventional Reset clears VF Enable in the PF. Thus, VFs no longer exist after a Conventional Reset.

2.2.2. FLR That Targets a VF

VFs must support Function Level Reset (FLR).

Note: Software may use FLR to reset a VF. FLR to a VF affects a VF's state but does not affect its existence in PCI Configuration Space or PCI Bus address space. The VFs BARn values (see Section 3.3.14) and VF MSE (see Section 3.3.3.4) in the PF's SR-IOV extended capability are unaffected by FLRs issued to the VF.

2.2.3. FLR That Targets a PF

PFs must support FLR.

FLR to a PF resets the PF state as well as the SR-IOV extended capability including VF Enable which means that VFs no longer exist.

2.3. IOV Re-initialization and Reallocation

If VF Enable is Cleared after having been Set, all of the VFs associated with the PF no longer exist and must no longer issue PCIe transactions or respond to Configuration Space or Memory Space accesses. VFs must not retain any state after VF Enable has been Cleared (including sticky bits).

Setting VF Enable after it has been previously Cleared results in the same VF state as if FLR had been issued to the VF.

2.4. VF Migration

VF Migration is an optional feature in the *Multi-Root I/O Virtualization and Sharing Specification*. Devices that do not support the MR-IOV Extended Capability do not support VF Migration functionality.

In Multi-Root systems, VFs can be migrated between Virtual Hierarchies.

For VF Migration to occur, both VF Migration Capable and VF Migration Enable must be Set. VF Migration Capable indicates to SR-PCIM that VF Migration Hardware is present and that MR-PCIM has enabled its use. Hardware support for VF Migration is optional. SR-PCIM support for VF Migration is also optional.

VF Migration Enable indicates to Device hardware and to MR-PCIM that SR-PCIM has also enabled VF Migration.

Supporting VF Migration places the following requirements on SR-PCIM:

- ❑ The need to determine if a VF is *Active, Dormant,* or *Inactive*. A VF that is Active is available for use by SR. A VF that is Dormant can be configured by SR but will not issue transactions. A VF that is Inactive is not available for use by SR.
- ❑ The need to participate in *Migrate In* operations. A Migrate In operation is used to offer a VF to SR. A Migrate In operation is initiated by MR-PCIM. SR-PCIM can accept a Migrate In operation and move a VF to the *Active.Available* state.
- ❑ The need to participate in *Migrate Out* operations. A Migrate Out operation is used to request the graceful removal of an Active VF from use by SR. SR-PCIM can accept the Migrate Out and move the VF to the *Inactive.Unavailable* state.
- ❑ The need to handle *Migrate In Retractions*. This is when an offer of a VF to SR is retracted by MR-PCIM and the VF moves back to the *Inactive.Unavailable* state.

2.4.1. Initial VF State

This section describes the initial values in the VF Migration State Array (see Section 3.3.15.1).

If InitialVFs (Section 3.3.5) is nonzero, VF_1 through $VF_{InitialVFs}$ are in the Active.Available state. If TotalVFs (Section 3.3.6) is greater than InitialVFs, $VF_{InitialVFs+1}$ through $VF_{TotalVFs}$ are in the Inactive.Unavailable state. If VF Migration Enable (Section 3.3.3.2) is Clear, VFs above InitialVFs are not used.

If InitialVFs is 0, no VFs are in the Active.Available state. If TotalVFs equals InitialVFs all VFs are in the Active.Available state. If TotalVFs is 0, no VFs are associated with this PF and there is no VF Migration State array.

Figure 2-2 describes this initial VF State.

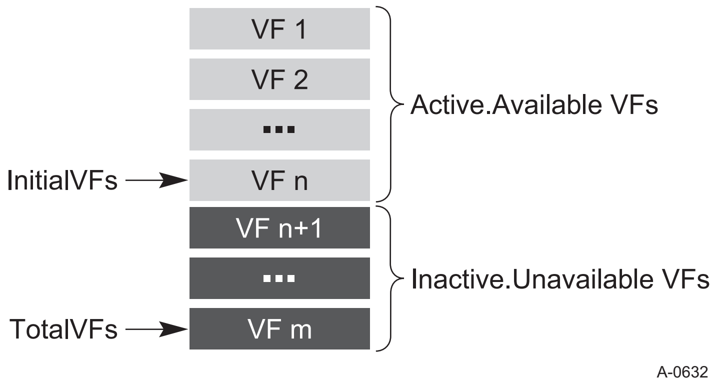

Figure 2-2: Initial VF Migration State Array

2.4.2. VF Migration State Transitions

VF migration follows the state diagram shown in Figure 2-3. The state values shown are contained in the VF State array entry associated with the VF. State transitions indicated by solid lines are initiated by MR-PCIM. State transitions indicated by dotted and dashed lines are initiated by SR-PCIM.

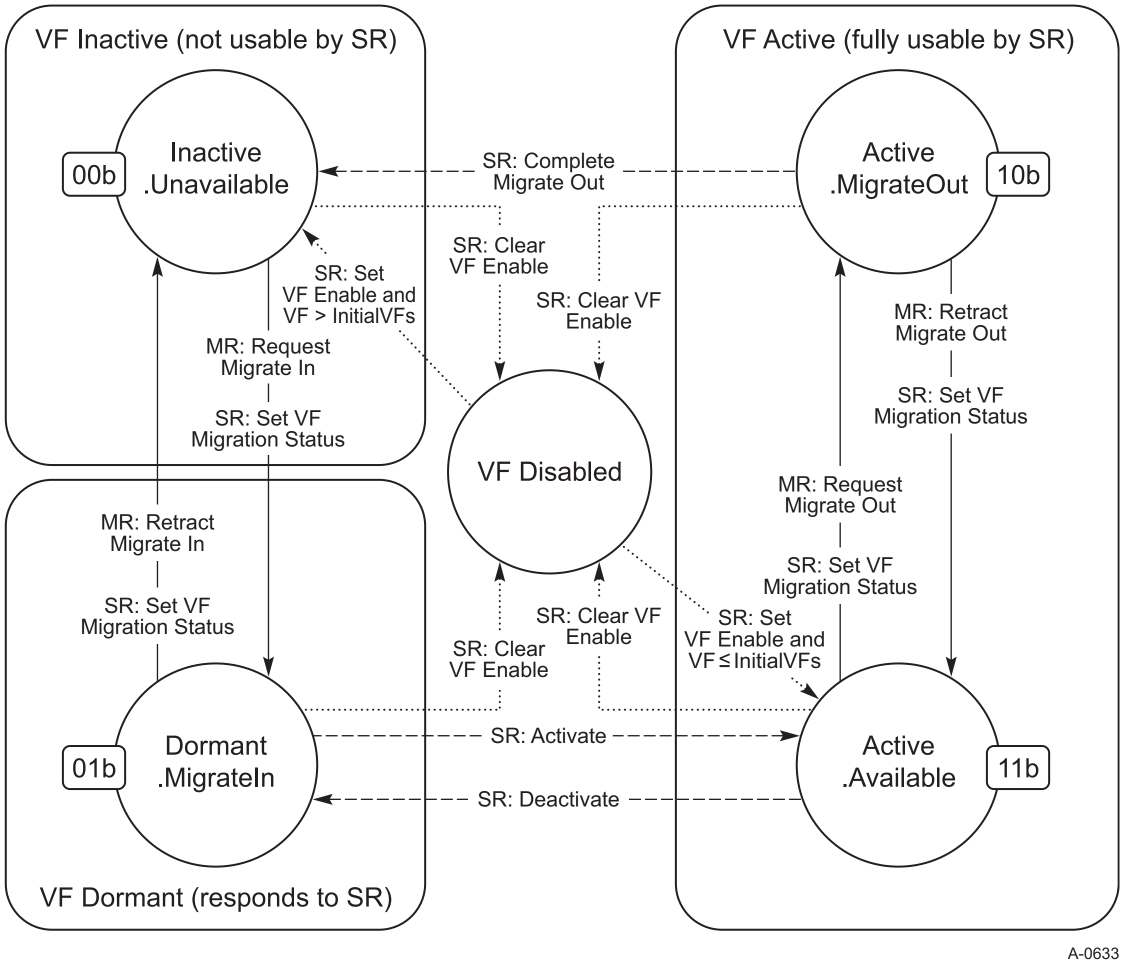

Figure 2-3: VF Migration State Diagram

SR-PCIM initiates a state transition by writing a new value to the VF Migration State array. Devices ignore write transactions and no state transitions occur when SR-PCIM attempts to initiate any state transition other than the following:

Current VF Enable	Current VF State	Written VF Enable	Written VF State	Meaning
1	Dormant.MigrateIn	1	Active.Available	SR Activate VF.
1	Active.Available	1	Dormant.MigrateIn	SR Deactivate VF.
1	Active.MigrateOut	1	Inactive.Unavailable	SR Complete Migrate Out.
1	Any	0	Any	SR Disables all VFs.
0	Any	1	Any	SR Enables all VFs. VFs transition to either Active.Available or Inactive.Unavailable based on the VF number and InitialVFs.

IMPLEMENTATION NOTE

Software State Migration Change Detection

SR-PCIM will typically need to verify that a state change took effect by re-reading VF Migration State after writing it.

VFs in states Inactive.Unavailable are not usable by software in any way. Requests targeting an Inactive VF return Unsupported Request (UR). Within 100 ms of transitioning to the Inactive or Dormant states, the Device must ensure that no new transactions will be issued using the indicated Routing ID.

MR-PCIM initiates state transitions by using a different data structure as specified in the *Multi-Root I/O Virtualization and Sharing Specification*. The effects of such transitions are visible in the VF Migration State array and in the VF Migration Status bit. All state transitions initiated by MR-PCIM cause the VF Migration Status bit to be Set.

This migration state machine exists for every VF that supports VF Migration. Migration state machines are not affected by Function Dependency Lists (see Section 2.1.3 and Section 3.3.8).

VF Migration State does not affect Function state. If VF state needs to be reset as part of a Migrate Out and/or Migrate In operation, SR-PCIM must issue an FLR to accomplish this. VF behavior when VF Migration occurs without an FLR is undefined.

IMPLEMENTATION NOTE

FLR and VF Migration

VF Migration from system A to system B will typically involve one FLR in system A before completing the MigrateOut operation and a second FLR in system B after accepting the MigrateIn operation but before using the VF. System A uses the first FLR to ensure that none of its data leaks out. System B uses the second FLR to ensure that it starts with a clean slate.

3. Configuration

3.1. Overview

This section provides additional requirements above the *PCI Express Base Specification* for implementing PFs and VFs.

PFs are discoverable in configuration space, as with all Functions. PFs contain the SR IOV Extended Capability described in Section 3.3. PFs are used to discover, configure, and manage the VFs associated with the PF and for other things described in this specification.

3.2. Configuration Space

PFs compliant with this specification shall implement the SR-IOV Capability as defined in the following sections. VFs compliant with this specification shall implement configuration space fields and capabilities as defined in the following sections.

3.3. SR-IOV Extended Capability

This specification defines a PCIe extended capability for each PF that supports Single Root IOV. This Capability is used to describe and control a PF's SR-IOV Capabilities.

For multi-Function devices, each PF that supports SR-IOV shall provide the Capability structure defined in this section.

Figure 3-1 shows the SR-IOV Capability structure.

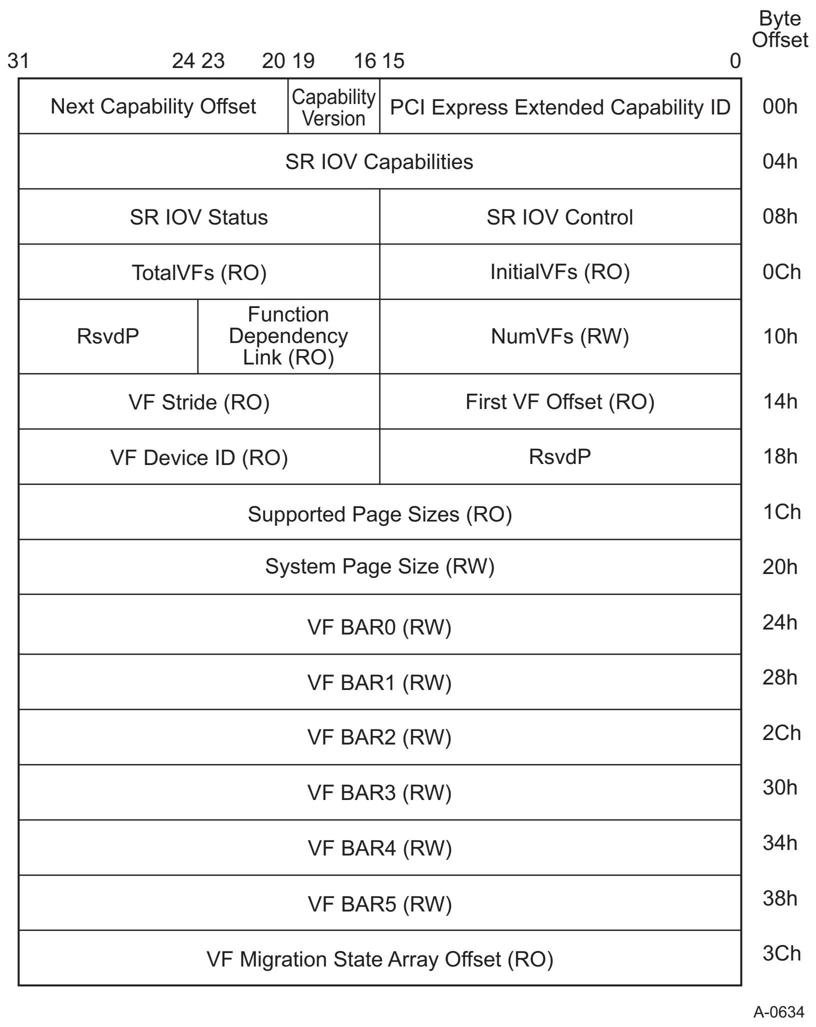

Figure 3-1: Single Root I/O Virtualization Extended Capabilities

3.3.1. SR-IOV Extended Capability Header (00h)

Table 3-1 defines the SR-IOV Extended Capability header. The Capability ID for the SR-IOV Extended Capability is 0010h.

Table 3-1: SR-IOV Extended Capability Header

Bit Location	Register Description	Attributes
15:0	**PCI Express Extended Capability ID** – This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. The Extended Capability ID for the SR-IOV Extended Capability is 0010h.	RO
19:16	**Capability Version** – This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification.	RO
31:20	**Next Capability Offset** – This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh.	RO

3.3.2. SR-IOV Capabilities (04h)

Table 3-2 defines the layout of the SR-IOV Capabilities field.

Table 3-2: SR-IOV Capabilities

Bit Location	Register Description	Attributes
0	**VF Migration Capable** – If Set, the PF is Migration Capable and operating under a Migration Capable MR-PCIM.	RO
20 .. 1	**Reserved** – These fields are currently reserved	RsvdP
31 .. 21	**VF Migration Interrupt Message Number** – Indicates the MSI/MSI-X vector used for migration interrupts. The value in this field is undefined if bit 0 of this Capability is Clear.	RO

3.3.2.1. VF Migration Capable

VF Migration Capable is Set to indicate that the PF supports VF Migration. If Clear, the PF does not support VF Migration.

VF Migration support is an optional feature of *Multi-Root I/O Virtualization and Sharing Specification*. If the PF does not implement hardware for VF Migration, this bit shall be implemented as RO value 0b. If the PF implements hardware for VF Migration, this bit is controlled by mechanisms defined in the *Multi-Root I/O Virtualization and Sharing Specification* and indicates the presence of support for VF Migration.

Devices that implement Single Root IOV only shall implement this field as RO value zero.

3.3.2.2. VF Migration Interrupt Message Number

VF Migration Interrupt Message Number must contain the MSI or MSI-X vector number used for the interrupt message generated in association with certain VF migration events.

For MSI, VF Migration Interrupt Message Number must indicate the MSI message number [0 .. 31] used to reference certain SR events described in this chapter. The Function is required to update this field so that it is correct if the number of MSI Messages assigned to the Function changes when software writes to the Multiple Message Enable field in the MSI Message Control register.

For MSI-X, VF Migration Interrupt Message Number must indicate the MSI-X Table entry [0 .. 2047] used to generate the interrupt message.

If both MSI and MSI-X are implemented, they are permitted to use different vectors, though software is permitted to enable only one mechanism at a time. If both MSI and MSI-X interrupts are enabled, this field is undefined.

If VF Migration Capable is Clear, this field is undefined.

IMPLEMENTATION NOTE

Migration and MSI

If a PF implements MSI and software sets Multiple Message Enable to a value less than Multiple Message Capable, some sharing of MSI vectors may be occurring. This can create complicated software structures since a single vector might need to be directed to both SR-PCIM and the PF Device Driver.

3.3.3. SR-IOV Control (08h)

Table 3-3 defines the layout of the SR-IOV Control fields.

Table 3-3: SR-IOV Control

Bit Location	Register Description	Attributes
0	**VF Enable** – Enables/Disables VFs. Default value is 0b.	RW
1	**VF Migration Enable** – Enables/Disables VF Migration Support. Default value is 0b.	RW
2	**VF Migration Interrupt Enable** – Enables/Disables VF Migration State Change Interrupt. Default value is 0b.	RW
3	**VF MSE** – Memory Space Enable for Virtual Functions. Default value is 0b.	RW
4	**ARI Capable Hierarchy** *PCI Express Endpoint:* The Device is permitted to locate VFs in Function numbers 8 to 255 of the captured Bus Number. Default value is 0b. This field is RW in the lowest numbered PF of the Device and is Read Only Zero in all other PFs. *Root Complex Integrated Endpoint:* Not applicable – must hardwire the bit to 0b.	RW or RO (see description)
15..5	**Reserved** – These fields are currently reserved.	RsvdP

3.3.3.1. VF Enable

VF Enable manages the assignment of VFs to the associated PF. If VF Enable is Set, the VFs associated with the PF are accessible in the PCI Express fabric. When Set, VFs respond to and may issue PCI Express transactions following the rules for PCI Express Endpoint Functions.

If VF Enable is Clear, VFs are disabled and not visible in the PCI Express fabric; VFs shall respond to Requests with UR and shall not issue PCI Express transactions.

To allow components to perform internal initialization, system software must wait for at least 100 ms after changing the VF Enable bit from a 0 to a 1, before it is permitted to issue Configuration Requests to the VFs which are enabled by that VF Enable bit. The Root Complex and/or system software must allow at least 1.0 s after Setting the VF Enable bit, before it may determine that a VF which fails to return a Successful Completion status for a valid Configuration Request is broken. After Setting the VF Enable bit the VFs enabled by that VF Enable bit are permitted to return a CRS status to configuration requests up to the 1.0 s limit, if they are not ready to provide a Successful Completion status for a valid Configuration Request.

Clearing VF Enable effectively destroys the VFs. Setting VF Enable effectively creates VFs. Setting VF Enable after it has previously been Cleared shall result in a new set of VFs in the $D0_{\text{ununitialized}}$ state.

If software Clears VF Enable, software must allow 1 second after VF Enable is Cleared before reading any field in the SR-IOV Extended Capability or the VF Migration State Array (see Section 3.3.15.1).

Section 3.3.7 NumVFs, Section 3.3.5 InitialVFs, Section 3.3.6 TotalVFs, Section 3.3.9 First VF Offset, Section 3.3.13 System Page Size, and Section 3.3.14 VF BARx describe additional semantics associated with this field.

3.3.3.2. VF Migration Enable

VF Migration Enable must be Set to allow VF Migration on this PF. See the *Multi-Root I/O Virtualization and Sharing Specification* for details.

If VF Migration Capable is Set, this bit is RW and the default value is 0b. Otherwise, this bit is RO Zero.

This field is RO when VF Enable is Set.

3.3.3.3. VF Migration Interrupt Enable

An MSI or MSI-X interrupt message must be sent every time the logical AND of the following conditions transitions from FALSE to TRUE:

- ❑ The associated vector is unmasked (not applicable if MSI does not support PVM).
- ❑ The VF Migration Interrupt Enable bit is Set to 1b.
- ❑ The VF Migration Status bit is Set.

The MSI or MSI-X vector used is indicated by the VF Migration Interrupt Message Number field.

Note: VF Migration events are described in Section 2.4.

3.3.3.4. VF MSE (Memory Space Enable)

VF MSE controls memory space enable for all Active VFs associated with this PF, as with the Memory Space Enable bit in a Function's PCI Command register. The default value for this bit is 0b.

When VF Enable is Set, VF memory space will respond only when VF MSE is Set. VFs shall follow the same error reporting rules as defined in the base specification if an attempt is made to access a Virtual Function's memory space when VF Enable is Set and VF MSE is Clear.

IMPLEMENTATION NOTE

VF MSE and VF Enable

VF memory space will respond with Unsupported Request when VF Enable is Clear. Thus, VF MSE is "don't care" when VF Enable is Clear; however, software may choose to Set VF MSE after programming the VF BAR*n* registers, prior to Setting VF Enable.

3.3.3.5. ARI Capable Hierarchy

ARI Capable Hierarchy is a hint to the Device that ARI has been enabled in the Root Port or Switch Downstream Port immediately above the Device. Software should set this bit to match the ARI Forwarding Enable bit in the Root Port or Switch Downstream Port immediately above the Device.

ARI Capable Hierarchy is only present in the lowest numbered PF of a Device (for example PF_0) and affects all PFs of the Device. ARI Capable Hierarchy is Read Only Zero in other PFs of a Device.

A Device may use the setting of ARI Capable Hierarchy to determine the values for First VF Offset (see Section 3.3.9) and VF Stride (see Section 3.3.10). The effect of changing ARI Capable Hierarchy is undefined if VF Enable is Set in any PF. This bit is not affected by FLR of any PF or VF.

ARI Capable Hierarchy does not apply to Root Complex Integrated Endpoints.

IMPLEMENTATION NOTE

ARI Capable Hierarchy

The Device has no way of knowing whether ARI has been enabled. If ARI is enabled, the Device can conserve Bus Numbers by assigning VFs to Function Numbers greater than 7 on the captured Bus Number. ARI is defined in the *PCI Express Base Specification.*

3.3.4. SR-IOV Status (0Ah)

Table 3-4 defines the layout of the SR-IOV Status field.

Table 3-4: SR-IOV Status

Bit Location	Register Description	Attributes
0	**VF Migration Status** – Indicates a VF Migration In or Migration Out Request has been issued by MR-PCIM. To determine the cause of the event, software may scan the VF State Array. Default value is 0b.	RW1C
15..1	**Reserved** – These fields are currently reserved.	RsvdZ

3.3.4.1. VF Migration Status

VF Migration Status is Set when VF Enable, VF Migration Capable, and VF Migration Enable are Set and certain state changes occur in a VF Migration State Array entry (see Section 2.4 and Section 3.3.15.1 for details).

This setting of VF Migration Status is not affected by the value of VF Migration Interrupt Enable or by any controls for MSI or MSI-X.

3.3.5. InitialVFs (0Ch)

InitialVFs indicates to SR-PCIM the number of VFs that are initially associated with the PF.

The minimum value of InitialVFs is 0.

For Devices operating in Single Root mode, this field is HwInit and must contain the same value as TotalVFs.

For Devices operating in Multi-Root mode, the value of this field may be changed by MR-PCIM when VF Enable is Clear.

Note: The mechanism used by MR-PCIM to affect this field is described in the *Multi-Root I/O Virtualization and Sharing Specification.*

If VF Migration Enable is Set and VF Enable is Cleared and then Set, the value of InitialVFs may change. This is necessary since some VFs may have been migrated to other PFs and may no longer be available to this PF.

3.3.6. TotalVFs (0Eh)

TotalVFs indicates the maximum number of VFs that could be associated with the PF.

The minimum value of TotalVFs is 0.

For Devices operating in Single-Root mode, this field is HwInit and must contain the same value as InitialVFs.

For Devices operating in Multi-Root mode, the value of this field may be changed by MR-PCIM.

Note: The mechanism used by MR-PCIM to affect this field is described in the *Multi-Root I/O Virtualization and Sharing Specification.*

3.3.7. NumVFs (10h)

NumVFs controls the number of VFs that are available. SR-PCIM sets NumVFs as part of the process of creating VFs. This number of VFs shall be visible in the PCI Express fabric after both NumVFs is set to a valid value and VF Enable is Set. These VFs shall respond to PCI Express transactions targeting them, and shall follow all rules defined by this specification and the base specification.

The results are undefined if NumVFs is set to a value greater than TotalVFs.

NumVFs may only be written while VF Enable is Clear. If NumVFs is written when VF Enable is Set, the results are undefined.

The initial value of NumVFs is undefined.

3.3.8. Function Dependency Link (12h)

The programming model for a Device may have vendor specific dependencies between sets of Functions. The Function Dependency Link field is used to describe these dependencies.

This field describes dependencies between PFs. VF dependencies are the same as the dependencies of their associated PFs.

If a PF is independent from other PFs of a Device, this field shall contain its own Function Number.

If a PF is dependent on other PFs of a Device, this field shall contain the Function Number of the next PF in the same Function Dependency List. The last PF in a Function Dependency List shall contain the Function Number of the first PF in the Function Dependency List.

If PF *p* and PF *q* are in the same Function Dependency List, than any SI that is assigned VF *p,n* shall also be assigned to VF *q,n*.

IMPLEMENTATION NOTE

Function Dependency Link Example

Consider the following scenario:

SR-IOV Field	**PF 0**	**PF 1**	**PF 2**
Function Dependency Link	1	0	2
NumVFs	4	4	6
First VF Offset	4	4	4
VF Stride	3	3	3

Function Number	Description	Independent
0	PF 0	No
1	PF 1	No
2	PF 2	Yes
3	Function not present	
4	VF 0,1 (aka PF 0 VF 1)	No
5	VF 1,1 (aka PF 1 VF 1)	No
6	VF 2,1 (aka PF 2 VF 1)	Yes
7	VF 0,2	No
8	VF 1,2	No
9	VF 2,2	Yes
10	VF 0,3	No
11	VF 1,3	No
12	VF 2,3	Yes
13	VF 0,4	No
14	VF 1,4	No
15	VF 2,4	Yes
16 to 17	Functions not present	
18	VF 2,5	Yes
19 to 20	Functions not present	
21	VF 2,6	Yes
22 to 255	Functions not present	

In this example, Functions 4 and 5 must be assigned to the same SI. Similarly, Functions 7 and 8, 10 and 11, 13, and 14 must be assigned together. If PFs are assigned to SIs, Functions 0 and 1 must be assigned together as well. Functions 2, 6, 9, 12, 15, 18, and 21 are independent and may be assigned to SIs in any fashion.

All PFs in a Function Dependency List shall have the same values for the InitialVFs, TotalVFs, and VF Migration Capable fields.

SR-PCIM shall ensure that all PFs in a Function Dependency List have the same values for NumVFs, VF Enabled, VF Migration Enabled fields before any VF in that Function Dependency List is assigned to an SI.

VF Migration and VF Mapping operations occur independently for every VF. SR-PCIM shall not assign a VF to an SI until it can assign all dependent VFs. For example, using the scenario above, if both VF 0,2 (Function 7) and VF 1,2 (Function 8) are in the Inactive.Unavailable or Dormant.MigrateIn states, SR-PCIM shall not assign either VF to an SI until both VFs reach the Active.Available state.

Similarly, SR-PCIM shall not remove a VF from an SI until it can remove all dependent VFs. For example, using the scenario above, both VF 0,2 and VF 1,2 shall be removed from an SI only when they both reach the Active.MigrateOut state. SR-PCIM shall not transition the Functions to Inactive.Unavailable until the SI has stopped using all dependent Functions.

3.3.9. First VF Offset (14h)

First VF Offset is a constant and defines the Routing ID offset of the first VF that is associated with the PF that contains this Capability structure. The first VF's 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the PF containing this field ignoring any carry, using unsigned, 16-bit arithmetic.

A VF shall not be located on a Bus Number that is numerically smaller than its associated PF.

This field may change value when the ARI Capable Hierarchy or NumVFs values change.

3.3.10. VF Stride (16h)

VF Stride defines the Routing ID offset from one VF to the next one for all VFs associated with the PF that contains this Capability structure. The next VF's 16-bit Routing ID is calculated by adding the contents of this field to the Routing ID of the current VF, ignoring any carry, using unsigned 16-bit arithmetic.

This field may change value when the ARI Capable Hierarchy or NumVFs values change.

Note: VF Stride is unused if NumVFs is 0 or 1. If NumVFs is greater than 1, VF Stride must not be zero.

3.3.11. VF Device ID (1Ah)

This field contains the Device ID that should be presented for every VF to the SI.

VF Device ID may be different from the PF Device ID.

3.3.12. Supported Page Sizes (1Ch)

This field indicates the page sizes supported by the PF. This PF supports a page size of 2^{n+12} if bit n is Set. For example, if bit 0 is Set, the PF supports 4-KB page sizes. PFs are required to support 4-KB, 8-KB, 64-KB, 256-KB, 1-MB, and 4-MB page sizes. All other page sizes are optional.

The page size describes the minimum alignment requirements for VF BAR resources as described in Section 3.3.13.

IMPLEMENTATION NOTE

Non-pre-fetch Address Space

Non-pre-fetch address space is limited to addresses below 4 GB. Pre-fetch address space in 32-bit systems is also limited. Vendors are strongly encouraged to utilize the System Page Size feature to conserve address space while also supporting systems with larger pages.

3.3.13. System Page Size (20h)

This field defines the page size the system will use to map the VFs' memory addresses. Software must set the value of the System Page Size to one of the page sizes set in the Supported Page Sizes field (see Section 3.3.12). As with Supported Page Sizes, if bit *n* is Set in System Page Size, the VFs associated with this PF are required to support a page size of 2^{n+12}. For example, if bit 1 is Set, the system is using an 8-KB page size. The results are undefined if more than one bit is set in System Page Size. The results are undefined if a bit is Set in System Page Size that is not Set in Supported Page Sizes.

When System Page Size is set, the VF associated with this PF is required to align all BAR resources on a System Page Size boundary. Each VF BAR*n* or VF BAR*n* pair (see Section 3.3.14) shall be aligned on a System Page Size boundary. Each VF BAR*n* or VF BAR*n* pair defining a non-zero address space shall be sized to consume an integer multiple of System Page Size bytes. All data structures requiring page size alignment within a VF shall be aligned on a System Page Size boundary.

VF Enable must be zero when System Page Size is written. The results are undefined if System Page Size is written when VF Enable is Set.

Default value is 1h (i.e., 4 KB).

3.3.14. VF BAR0, VF BAR1, … VF BAR5 (24h … 38h)

These fields must define the VF's Base Address Registers (BARs). These fields behave as normal PCI BARs, as described in the *PCI Local Bus Specification, Revision 3.0*. They can be sized by writing all 1's and reading back the contents of the BARs as described in the *PCI Local Bus Specification, Revision 3.0*, complying with the low order bits that define the BAR type fields. The amount of address space decoded by each BAR shall be an integral multiple of System Page Size.

Each VF BAR*n*, when "sized" by writing 1's and reading back the contents, describes the amount of address space consumed and alignment required by a single Virtual Function, per BAR. When written with an actual address value, and VF Enable and VF MSE are Set, the BAR maps NumVFs BARs. In other words, the base address is the address of the first VF BAR*n* associated with this PF and all subsequent VF BAR*n* address ranges follow as described below.

VF BARs shall only support 32-bit and 64-bit memory space. PCI I/O Space is not supported in VFs. Bit 0 of any implemented VF BARx must be RO 0b except for a VF BARx used to map the upper 32 bits of a 64-bit memory VF BAR pair.

The alignment requirement and size read is for a single VF, but when VF Enable is Set and VF MSE is Set, the BAR contains the base address for all (NumVFs) VF BAR*n*.

The algorithm to determine the amount of address space mapped by a VF BAR*n* differs from the base specification as follows:

1. After reading the low order bits to determine if the BAR is a 32-bit BAR or 64-bit BAR pair, determine the size and alignment requirements by writing all 1's to VF BAR*n* (or VF BAR*n* and VF BAR*n+1* for a 64-bit BAR pair) and reading back the contents of the BAR or BAR pair. Convert the bit mask returned by the read(s) to a size and alignment value as described in the *PCI Local Bus Specification, Revision 3.0*. This value is the size and alignment for a single VF.
2. Multiply the value from step 1 by the value set in NumVFs to determine the total amount of space the BAR or BAR pair will map after VF Enable and VF MSE are Set.

For each VF BAR*n* field, *n* corresponds to one of the VFs BAR. Table 3-5 shows the relationship between *n* and a Function's BAR.

Table 3-5: BAR Offsets

n	BAR Offset in a Type 0 Header
0	10h
1	14h
2	18h
3	1Ch
4	20h
5	24h

3.3.15. VF Migration State Array Offset (3Ch)

If VF Migration Capable (Section 3.3.2.1) is Set and TotalVFs (Section 3.3.6) is not zero, this field shall contain a PF BAR relative pointer to the VF Migration State Array. The VF Migration State Array is defined in Section 3.3.15.1. The layout of the VF Migration State Array Offset is defined in Table 3-6.

Table 3-6: VF Migration State Array Offset

Bit Location	Register Description	Attributes
31..3	**VF Migration State Offset** – Used as an offset from the address contained by one of the Function's Base Address registers to point to the base of the VF Migration State Array. The lower three MVF Migration State BIR bits are masked off (set to zero) by software to form a 32-bit QWORD-aligned offset. This field is undefined if TotalVFs is 0.	RO
2..0	**VF Migration State BIR** – Indicates which one of a Function's Base Address registers, located beginning at 10h in Configuration Space, is used to map the Function's VF Migration State Array into Memory Space. **BIR Value** **BAR** 0 BAR0 10h 1 BAR1 14h 2 BAR2 18h 3 BAR3 1Ch 4 BAR4 20h 5 BAR5 24h 6 Reserved 7 Reserved For a 64-bit BAR, the VF Migration State BIR indicates the lower DWORD. This field is undefined if TotalVFs is 0.	RO

If a BAR that maps address space for the VF Migration State Array also maps other usable address space not associated with VF Migration, locations used in the other address space must not share any naturally aligned 8-KB address range with one where the VF Migration State Array resides.

This field is RO zero if VF Migration Capable is Clear.

3.3.15.1. *VF Migration State Array*

The VF Migration State Array is located using the VF Migration State Array Offset field of the SR-IOV Capability block.

The VF Migration State Array has a VF Migration State Entry for each VF. The total size of the VF State array is NumVFs bytes. The VF Migration State Array shall not exist if TotalVFs is 0. Table 3-7 defines the layout of a VF Migration State Array entry.

Table 3-7: VF Migration State Entry

Bit Location	Register Description	Attributes
1..0	**VF Migration State** – State of the associated VF. This field is undefined when VF Enable is Clear. The initial values of the VF Migration State Array are described in Section 2.4.1.	RW
7..2	**Reserved** – These fields are currently reserved.	RsvdP

VF Migration State contains the values shown in Table 3-8.

Table 3-8: VF Migration State Descriptions

VF State	VF Exists	Description
00b	No	**Inactive.Unavailable** – VF does not exist to SR nor is it being migrated in or out.
01b	No	**Dormant.MigrateIn** – VF is available for use by SR. VF exists but cannot initiate transactions.
10b	Yes	**Active.MigrateOut** – SR has been requested to relinquish use of the VF.
11b	Yes	**Active.Available** – Fully functional. Could be assigned to an SI.

The initial values of the VF Migration State array are described in Section 2.4.1. The VF Migration State array is returned to a configuration as described in Section 2.4.1 within 1 second after VF Enable is Cleared.

Software initiates a state transition by writing a new state value to the entry.

Valid state transitions are listed in Table 3-9 and Table 3-10 and shown in Figure 2-3. Only changes in Table 3-9 can be requested by SR. Changes in Table 3-10 are initiated by MR-PCIM using mechanisms described in the *Multi-Root I/O Virtualization and Sharing Specification* and have SR visible effects described here. Any transition issued by SR-PCIM and not listed in Table 3-9 is ignored and does not change the VF State.

Table 3-9: SR-PCIM Initiated VF Migration State Transitions

Current State	New State	Change Initiated By	SR Visible Effects of Change
Dormant.MigrateIn	Active.Available	SR-PCIM	**VF Activate** VF now exists.
Active.Available	Dormant.MigrateIn	SR-PCIM	**VF Deactivate** VF no longer exists.
Active.MigrateOut	Inactive.Unavailable	SR-PCIM	**VF Migrate Out Complete** VF no longer exists.

Table 3-10: MR-PCIM Initiated VF Migration State Transitions

Current State	New State	Change Initiated By	SR Visible Effects of Change
Active.Available	Active.MigrateOut	MR-PCIM	**VF Migrate Out Request** VF continues to exist. Sets VF Migration Status.
Inactive.Unavailable	Dormant.MigrateIn	MR-PCIM	**VF Migrate In Request** VF remains non-existent. Sets VF Migration Status.
Dormant.MigrateIn	Inactive.Unavailable	MR-PCIM	**VF Migrate In Retract** VF remains non-existent. Sets VF Migration Status.
Active.MigrateOut	Active.Available	MR-PCIM	**VF Migrate Out Retract** VF continues to exist. Sets VF Migration Status.

3.4. PF/VF Configuration Space Header

This section defines the requirements on PF and VF configuration space fields.

Devices compliant with this specification must comply with *PCI Express Base Specification, Revision 1.1* or later. The register definitions listed throughout this document establish the mapping between existing PCI-SIG specifications and the PF/VF definitions introduced in this specification.

Table 3-11 defines the attributes used to describe each field in subsequent tables.

Table 3-11: Register Attributes

Register Attribute	Description
LB 3.0	Attribute is same as specified in the *PCI Local Bus Specification, Revision 3.0*.
Base	Attribute is same as specified in the *PCI Express Base Specification.*
HwInit	Hardware Initialized: Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization and can only be reset (for write-once by firmware) with “Power Good Reset”.
RO	Read-only register: Register bits are read-only and cannot be altered by software. Register bits may be initialized by hardware mechanisms such as pin strapping or serial EEPROM.
RW	Read-Write register: Register bits are read-write and may be either Set or Cleared by software to the desired state.
RW1C	Read-only status, Write-1-to-clear status register: Register bits indicate status when read, a Set bit indicating a status event may be Cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
ROS	Sticky - Read-only register: Registers are read-only and cannot be altered by software. Registers are not initialized or modified by reset. Devices that consume AUX power must preserve sticky register values when AUX power consumption (either via AUX power or PME Enable) is enabled.
RWS	Sticky - Read-Write register: Registers are read-write and may be either Set or Cleared by software to the desired state. Bits are not initialized or modified by reset. Devices that consume AUX power must preserve sticky register values when AUX power consumption (either via AUX power or PME Enable) is enabled.

Register Attribute	Description
RW1CS	Sticky - Read-only status, Write-1-to-clear status register: Registers indicate status when read, a Set bit indicating a status event may be Cleared by writing a 1. Writing a 0 to RW1CS bits has no effect. Bits are not initialized or modified by reset. Devices that consume AUX power must preserve sticky register values when AUX power consumption (either via AUX power or PME Enable) is enabled.
RsvdP	Reserved and Preserved: Reserved for future RW implementations; software must preserve value read for writes to bits.
RsvdZ	Reserved and Zero: Reserved for future RW1C implementations; software must use 0 for writes to bits.

3.4.1. Type 0 Configuration Space Header

Figure 3-2 details allocation for register fields of PCI Express Type 0 Configuration Space Header.

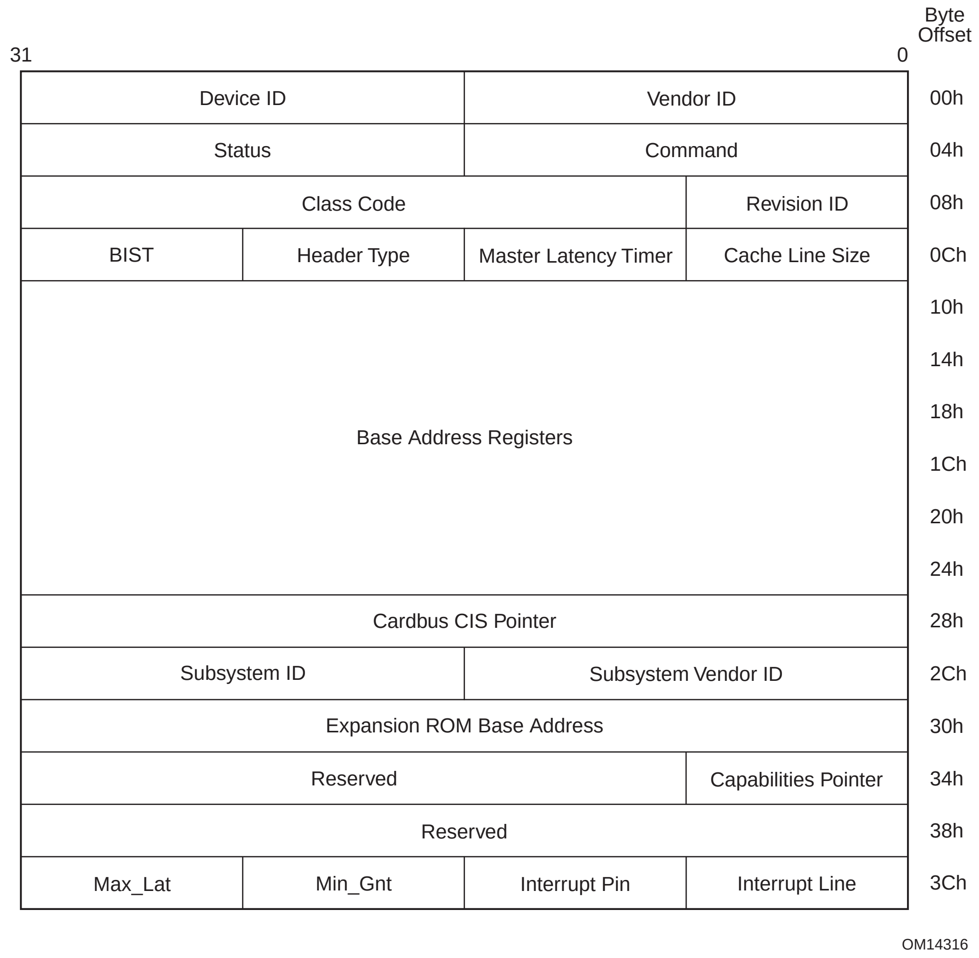

Figure 3-2: Type 0 Configuration Space Header

The PCI Express-specific interpretation of registers specific to PCI Express Type 0 Configuration Space Header is defined in this section.

Error Reporting fields are described in more detail in Chapter 4.

3.4.1.1. Vendor ID (Offset 00h)

This Read Only field identifies the manufacturer of the Device.

This field in all VFs returns FFFFh when read. VI software should return the Vendor ID value from the associated PF as the Vendor ID value for the VF.

3.4.1.2. Device ID (Offset 02h)

This Read Only field identifies the particular Device.

This field in all VFs returns FFFFh when read. VI software should return the VF Device ID (see Section 3.3.11) value from the associated PF as the Device ID for the VF.

IMPLEMENTATION NOTE

Legacy PCI Probing Software

Returning FFFFh for Device ID and Vendor ID values allows some legacy software to ignore VFs. See the *PCI Local Bus Specification, Revision 3.0.*

3.4.1.3. Command (Offset 04h)

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-12. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 3-12: Command Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**I/O Space Enable** – Does not apply to VFs. Must be hardwired to 0b for VFs.	Base	0b
1	**Memory Space Enable** – Does not apply to VFs. Must be hardwired to 0b for VFs. VF Memory Space is controlled by the VF MSE bit in the VF Control register.	Base	0b
2	**Bus Master Enable** – Transactions for a VF that has its Bus Master Enable Set must not be blocked by transactions for VFs that have their Bus Master Enable Cleared.	Base	Base

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
6	**Parity Error Enable**	Base	RsvdP
8	**SERR Enable**	Base	RsvdP
10	**Interrupt Disable** – Does not apply to VFs.	Base	0b

3.4.1.4. Status (Offset 06h)

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-13.

Table 3-13: Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3	**Interrupt Status** – Does not apply to VFs.	Base	0b

3.4.1.5. Revision ID (Offset 08h)

This register specifies a device specific revision identifier.

The value reported in the VF may be different than the value reported in the PF.

3.4.1.6. Class Code (Offset 09h)

The Class Code register is read-only and is used to identify the generic function of the device and, in some cases, a specific register level programming interface. The field in the PF and associated VFs must return the same value when read.

3.4.1.7. Cache Line Size (Offset 0Ch)

This field is implemented by PCI Express devices as a read-write field for legacy compatibility purposes but has no effect on any PCI Express device behavior. Physical Functions continue to implement this field as RW. For Virtual Functions, this field is RO Zero.

3.4.1.8. Latency Timer (Offset 0Dh)

This field does not apply to PCI Express. This register must be RO Zero.

3.4.1.9. Header Type (Offset 0Eh)

This byte identifies the layout of the second part of the predefined header (beginning at byte 10h in Configuration Space) and also whether or not the device contains multiple Functions. Bit 7 in this register is used to identify a multi-Function device. For an SR-IOV device, bit 7 in this field is only Set if there are multiple Functions. VFs do not affect the value of bit 7. Bits 6 through 0 identify the layout of the second part of the predefined header. For VFs, this field must be RO Zero.

3.4.1.10. BIST (Offset 0Fh)

VFs shall not support BIST and must define this field as RO Zero.

If VF Enable is turned on in any PF of a Device, then software must not invoke BIST in any Function associated with that Device.

3.4.1.11. Base Address Registers (Offset 10h, 14h, ... 24h)

For VFs, the values in these registers are RO Zero.

Note: See also Section 2.1.1.1 and Section 3.3.14.

3.4.1.12. Cardbus CIS Pointer (Offset 28h)

For VFs, this register is not used and shall be RO Zero.

3.4.1.13. Subsystem Vendor ID (Offset 2Ch)

This Read Only field identifies the manufacturer of the subsystem. The field in the PF and associated VFs must return the same value when read.

3.4.1.14. Subsystem ID (Offset 2Eh)

This Read Only field identifies the particular subsystem. The Device may have a different value in the PF and the VF.

3.4.1.15. Expansion ROM BAR (Offset 30h)

The Expansion ROM BAR may be implemented in PFs. The expansion ROM BAR is RO Zero in VFs. The VI may choose to provide access to the PF Expansion ROM BAR for VFs via emulation.

ROM BAR shared decoding, defined in the *PCI Local Bus Specification, Revision 3.0*, Section 6.2.5.2, is not permitted in any Device compliant with this specification.

3.4.1.16. Capabilities Pointer (Offset 34h)

No differences from the *PCI Express Base Specification*.

3.4.1.17. Interrupt Line (Offset 3Ch)

This field does not apply to VFs. This field must be RO Zero.

3.4.1.18. Interrupt Pin (Offset 3Dh)

This field does not apply to VFs. This field must be RO Zero.

3.4.1.19. Min_Gnt/Max_Lat (Offset 3Eh/3Fh)

These registers do not apply to PCI Express. They must be RO Zero.

3.5. PCI Express Capability

PCI Express defines a Capability structure in *PCI Local Bus Specification, Revision 3.0* compatible configuration space (first 256 bytes) as shown in Figure 3-2 for identification of a PCI Express device and indicates support for new PCI Express features. The PCI Express Capability Structure is required for PCI Express devices. The Capability structure is a mechanism for enabling PCI software transparent features requiring support on legacy operating systems. In addition to identifying a PCI Express Device, the PCI Express Capability Structure is used to provide access to PCI Express specific Control/Status registers and related Power Management enhancements.

Figure 3-3 details allocation of register fields in the PCI Express Capability Structure. PFs and VFs are required to implement this capability as described in the *PCI Express Base Specification* subject to the exceptions and additional requirements described below.

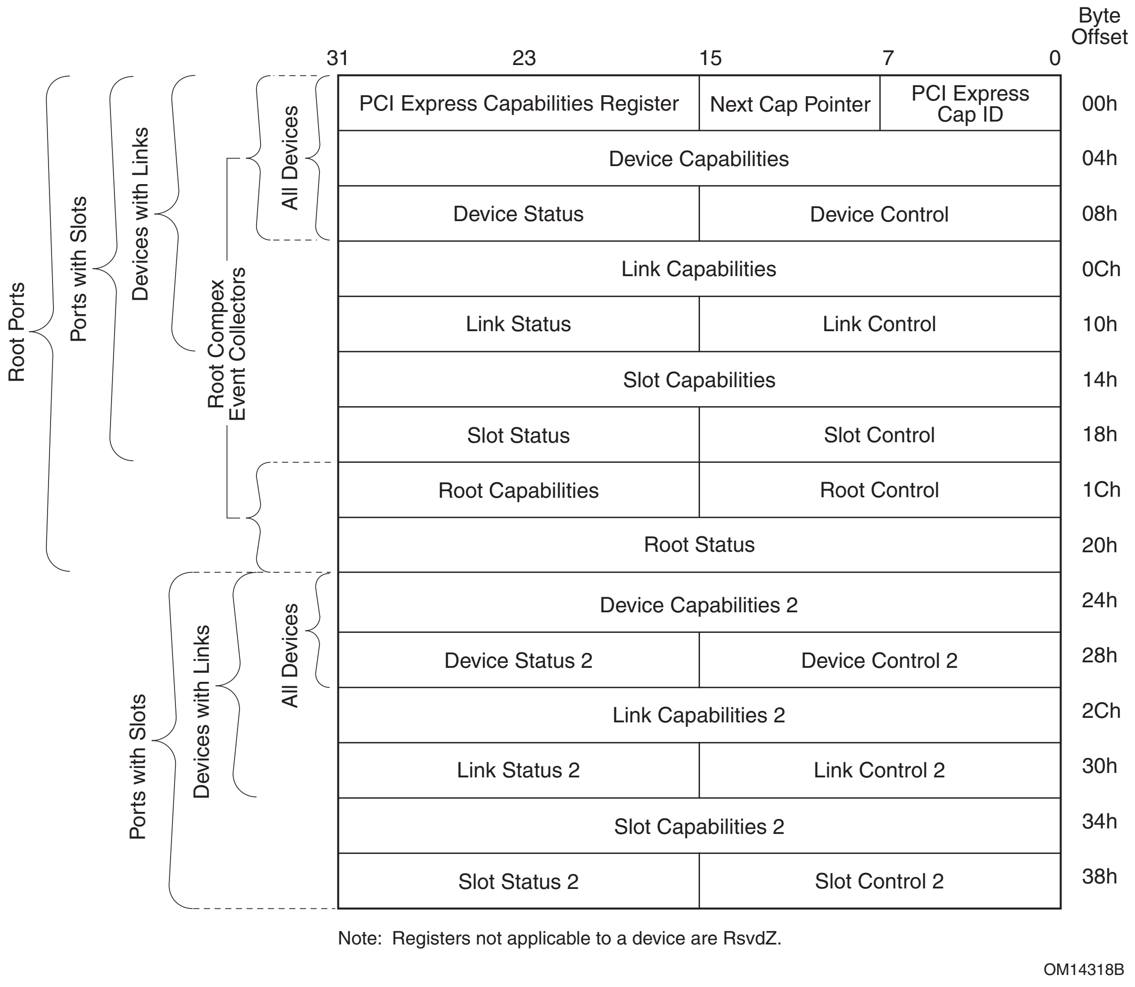

Figure 3-3: PCI Express Capability Structure

3.5.1. PCI Express Capability List Register (Offset 00h)

The PCI Express Capability List register enumerates the PCI Express Capability Structure in the *PCI Local Bus Specification, Revision 3.0* configuration space Capability list. Figure 3-4 details allocation of register fields in the PCI Express Capability List register. The PF and VF functionality is defined in the *PCI Express Base Specification*.

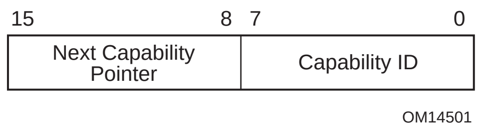

Figure 3-4: PCI Express Capability List Register

3.5.2. PCI Express Capabilities Register (Offset 02h)

The PCI Express Capabilities register identifies PCI Express device type and associated capabilities. Figure 3-5 details allocation of register fields in the PCI Express Capabilities register.

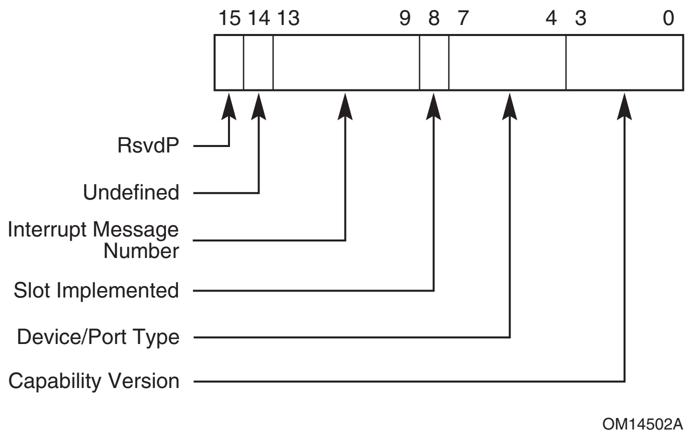

Figure 3-5: PCI Express Capabilities Register

The PF and VF functionality is defined in the *PCI Express Base Specification*.

3.5.3. Device Capabilities Register (Offset 04h)

The Device Capabilities register identifies PCI Express device specific capabilities. Figure 3-6 details allocation of register fields in the Device Capabilities register; Table 3-14 provides the respective bit definitions.

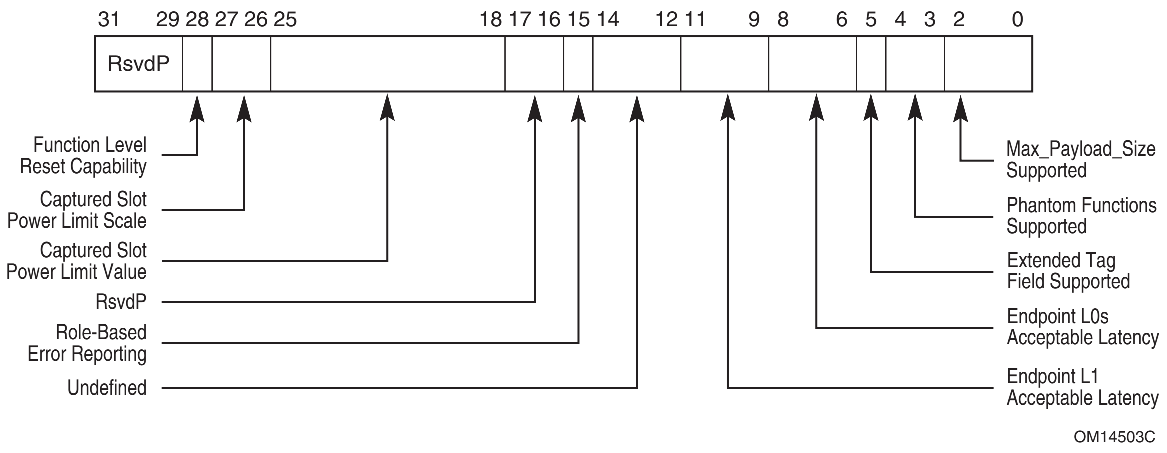
31
29 28 27 26 25
18 17 16 15 14
12 11
9 8
6 5 4 3 2
0
RsvdP
Function Level Reset Capability
Captured Slot Power Limit Scale
Captured Slot Power Limit Value
RsvdP
Role-Based Error Reporting
Undefined
Max_Payload_Size Supported
Phantom Functions Supported
Extended Tag Field Supported
Endpoint L0s Acceptable Latency
Endpoint L1 Acceptable Latency
OM14503C

Figure 3-6: Device Capabilities Register

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-14.

Table 3-14: Device Capabilities Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4:3	**Phantom Functions Supported** – When the VF Enable is Set, use of Phantom Function numbers by this PF and associated VFs is not permitted and this register must return 00b when read.	Base	00b
25:18	**Captured Slot Power Limit Value**	Base	undefined
27:26	**Captured Slot Power Limit Scale**	Base	undefined
28	**Function Level Reset Capability** – Required for PFs and for VFs.	1b	1b

3.5.4. Device Control Register (Offset 08h)

The Device Control register controls PCI Express device specific parameters. Figure 3-7 details allocation of register fields in the Device Control register; Table 3-15 provides the respective bit definitions.

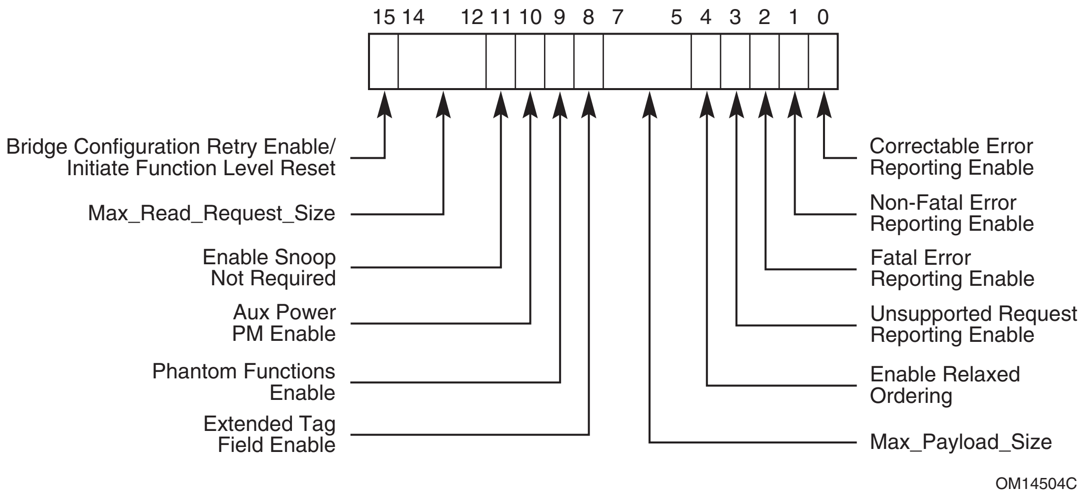
15 14 12 11 10 9 8 7 5 4 3 2 1 0
Bridge Configuration Retry Enable/ Initiate Function Level Reset
Max_Read_Request_Size
Enable Snoop Not Required
Aux Power PM Enable
Phantom Functions Enable
Extended Tag Field Enable
Correctable Error Reporting Enable
Non-Fatal Error Reporting Enable
Fatal Error Reporting Enable
Unsupported Request Reporting Enable
Enable Relaxed Ordering
Max_Payload_Size
OM14504C

Figure 3-7: Device Control Register

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-15. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 3-15: Device Control Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Correctable Error Reporting Enable** See Chapter 4 for details on error reporting.	Base	RsvdP
1	**Non-Fatal Error Reporting Enable** See Chapter 4 for details on error reporting.	Base	RsvdP
2	**Fatal Error Reporting Enable** See Chapter 4 for details on error reporting.	Base	RsvdP
3	**Unsupported Request Reporting Enable** See Chapter 4 for details on error reporting.	Base	RsvdP
4	**Enable Relaxed Ordering**	Base	RsvdP
7:5	**Max_Payload_Size**	Base	RsvdP
8	**Extended Tag Field Enable**	Base	RsvdP
9	**Phantom Functions Enable**	Base	RsvdP
10	**Auxiliary (AUX) Power PM Enable**	Base	RsvdP

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
11	**Enable No Snoop**	Base	RsvdP
14:12	**Max_Read_Request_Size**	Base	RsvdP
15	**Initiate Function Level Reset** – Required for PFs and for VFs. **Note:** Setting Initiate Function Level Reset in a PF resets VF Enable which means that VFs no longer exist after the FLR is complete.	Base	Base

3.5.5. Device Status Register (Offset 0Ah)

The Device Status register provides information about PCI Express device specific parameters. Figure 3-8 details allocation of register fields in the Device Status register; Table 3-16 provides the respective bit definitions.

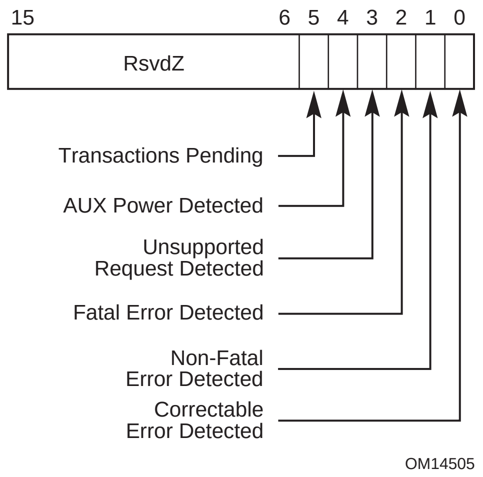

Figure 3-8: Device Status Register

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-16.

Table 3-16: Device Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	**AUX Power Detected**	Base	0b

3.5.6. Link Capabilities Register (Offset 0Ch)

The Link Capabilities register identifies PCI Express Link specific capabilities. Figure 3-9 details allocation of register fields in the Link Capabilities register.

PF and VF functionality is defined in the *PCI Express Base Specification*.

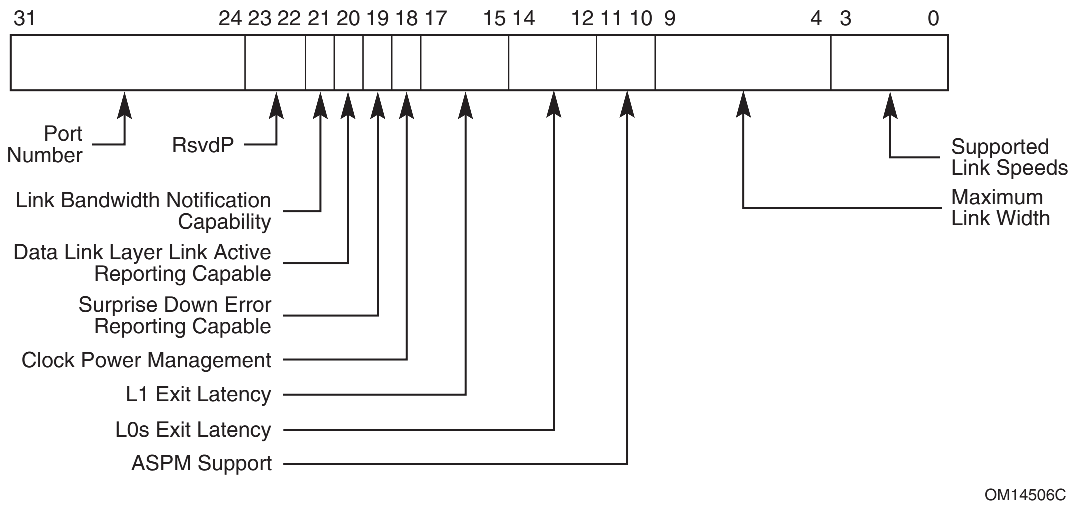

Figure 3-9: Link Capabilities Register

3.5.7. Link Control Register (Offset 10h)

The Link Control register controls PCI Express Link specific parameters. Figure 3-10 details allocation of register fields in the Link Control register.

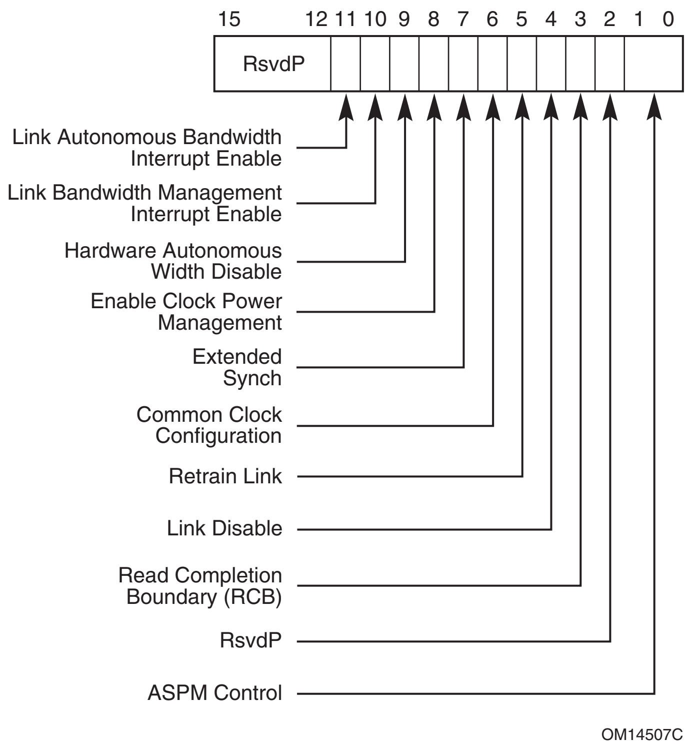
15
12 11 10 9 8 7 6 5 4 3 2 1 0
RsvdP
Link Autonomous Bandwidth Interrupt Enable
Link Bandwidth Management Interrupt Enable
Hardware Autonomous Width Disable
Enable Clock Power Management
Extended Synch
Common Clock Configuration
Retrain Link
Link Disable
Read Completion Boundary (RCB)
RsvdP
ASPM Control
OM14507C

Figure 3-10: Link Control Register

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-17. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 3-17: Link Control Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
1:0	**Active State Power Management (ASPM) Control**	Base	RsvdP
3	**Read Completion Boundary (RCB)** Must be hardwired to 0b for VFs.	Base	RsvdP
6	**Common Clock Configuration**	Base	RsvdP
7	**Extended Synch**	Base	RsvdP
8	**Enable Clock Power Management**	Base	RsvdP
9	**Hardware Autonomous Width Disable**	Base	RsvdP

3.5.8. Link Status Register (Offset 12h)

The Link Status register provides information about PCI Express Link specific parameters. Figure 3-11 details allocation of register fields in the Link Status register.

PF functionality is defined in the *PCI Express Base Specification*. For the VF, all fields in this register are reserved and the PF setting applies to the VF.

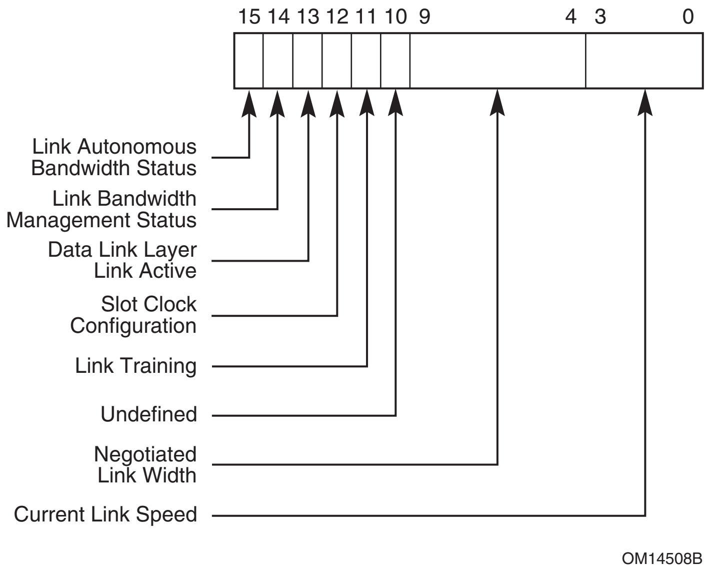

Figure 3-11: Link Status Register

3.5.9. Device Capabilities 2 Register (Offset 24h)

PF and VF functionality is defined in the *PCI Express Base Specification* except as noted in Table 3-18.

Table 3-18: Device Capabilities 2 Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3:0	**Completion Timeout Ranges Supported** VF value must be identical to PF value.	Base	Base
4	**Completion Timeout Disable Supported** VF value must be identical to PF value.	Base	Base

3.5.10. Device Control 2 Register (Offset 28h)

PF and VF functionality is defined in the *PCI Express Base Specification* except as noted in Table 3-19.

Table 3-19: Device Control 2 Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
3:0	**Completion Timeout Value** The PF value applies to all associated VFs.	Base	RsvdP
4	**Completion Timeout Disable** The PF value applies to all associated VFs.	Base	RsvdP

3.5.11. Device Status 2 Register (Offset 2Ah)

This register is not applicable to this specification.

3.5.12. Link Capabilities 2 Register (Offset 2Ch)

This register is not applicable to this specification.

3.5.13. Link Control 2 Register (Offset 30h)

PF and VF functionality is defined in the *PCI Express Base Specification*.

3.5.14. Link Status 2 Register (Offset 32h)

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-20. The VF fields marked RsvdP use the value of the associated PF.

Table 3-20: Device Control 2 Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Current De-emphasis Level**	Base	RsvdP

3.6. PCI Standard Capabilities

SR-IOV usage of PCI Standard Capabilities is described in Table 3-21. Items marked n/a are not applicable to PFs or VFs.

Table 3-21: SR-IOV Usage of PCI Standard Capabilities

Capability ID	Description	PF Attributes	VF Attributes
01h	PCI Power Management Interface	Base	Optional. See Chapter 6.
02h	AGP	n/a	n/a
03h	VPD	Base	Optional. See Section 3.6.1.
04h	Slot Identification	Base	n/a
05h	MSI	Base	See Chapter 5.
06h	CompactPCI* Hot Swap	n/a	n/a
07h	PCI-X™	n/a	n/a
08h	HyperTransport*	n/a	n/a
09h	Vendor Specific	Base	Base
0Ah	Debug Port	Base	Base
0Bh	CompactPCI Central Resource Control	n/a	n/a
0Ch	PCI Hot Plug	Base	n/a
0Dh	PCI Bridge Subsystem ID	n/a	n/a
0Eh	AGP 8x	n/a	n/a
0Fh	Secure Device	n/a	n/a
10h	PCI Express	Base	See Section 3.5.
11h	MSI-X	See Chapter 5.	See Chapter 5.
12h	Serial ATA Data/Index Configuration	n/a	n/a
13h	Advanced Features	n/a	n/a

3.6.1. VPD Capability

The VPD Capability is optional in PCI. It remains optional in SR-IOV.

VFs and PFs that implement the VPD Capability must ensure that there can be no “data leakage” between VFs and/or PFs via the VPD Capability.

3.7. PCI Express Extended Capabilities

SR-IOV usage of PCI Express Extended Capabilities is described in Table 3-22. Items marked n/a are not applicable to PFs or VFs (e.g., for Capabilities only present in Root Complexes or only present in Function 0).

Table 3-22: SR-IOV Usage of PCI Express Extended Capabilities

Extended Capability ID	Description	PF Attributes	VF Attributes
0001h	Advanced Error Reporting	Base	See Section 4.2.
0002h	Virtual Channel (02h)	Base	Not implemented. See Section 3.7.1.
0003h	Device Serial Number	Base	Not implemented
0004h	Power Budgeting	Base	Not implemented
0005h	Root Complex Link Declaration	n/a	n/a
0006h	Root Complex Internal Link Control	n/a	n/a
0007h	Root Complex Event Collector Endpoint Association	n/a	n/a
0008h	Multi Function Virtual Channel	Base	Not implemented. See Section 3.7.1.
0009h	Virtual Channel	Base	Not implemented. See Section 3.7.1.
000Ah	Root Complex Register Block Header Capability (RCRB)	n/a	n/a
000Bh	Vendor Specific	Base	Base
000Ch	Correlation Access Capability	n/a	n/a
000Dh	Access Control Services (ACS)	Base. See Section 3.7.2.	Base. See Section 3.7.2.
000Eh	Alternative Routing ID Interpretation (ARI)	See Section 3.7.3.	See Section 3.7.3.
000Fh	Address Translation Services (ATS)	See Section 3.7.4.	See Section 3.7.4.
0010h	Single Root I/O Virtualization (SR-IOV)	See Section 3.3.	Not implemented
0011h	Multi-Root I/O Virtualization (MR-IOV)	Not implemented. See Section 3.7.4.	Not implemented. See Section 3.7.4.

3.7.1. Virtual Channel/MFVC

VFs use the Virtual Channels of the associated PFs. As such, VFs do not contain any Virtual Channel Capabilities.

VFs shall not contain the MFVC Capability. This Capability is only present in Function 0 which shall never be a VF.

3.7.2. Access Control Services (ACS) Extended Capability

ACS is an optional extended capability. If an SR-IOV Capable Device implements internal peer-to-peer transactions, ACS is required with additional requirements described below.

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-23 and Table 3-24.

All Functions in SR-IOV Capable Devices (Devices that implement at least one PF) that support peer-to-peer transactions within the Device shall implement the ACS Egress Control Vector including the fields described in Table 3-23 and Table 3-24.

Table 3-23: ACS Capability Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
5	**ACS P2P Egress Control (E)** – Required if peer-to-peer transactions within the Device are supported.	Base	Base
15:8	**Egress Control Vector Size** – Required if peer-to-peer transactions within the Device are supported	Base	Base

Table 3-24: ACS Egress Control Vector

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
N-1:0	**Egress Control Vector** – Required if peer-to-peer transactions within the Device are supported	Base	Base

3.7.3. Alternative Routing ID Interpretation Extended Capability (ARI)

ARI is not applicable to Root Complex Integrated Endpoints; all other SR-IOV Capable Devices (Devices that include at least one PF) shall implement the ARI Capability in each Function. PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 3-25.

Table 3-25: ARI Capability Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**MFVC Function Groups Capability (M)** – Additional requirements described below.	Base	Base
1	**ACS Function Groups Capability (A)** – Additional requirements described below.	Base	Base
15:8	**Next Function Number** – VFs are located using First VF Offset (see Section 3.3.9) and VF Stride (see Section 3.3.10).	Base	Undefined

Any Device that implements the MFVC Capability with the optional Function Arbitration Table and consumes more than one Bus Number shall implement the MVFC Function Groups Enable (M) capability bit as 1b.

Any Device that implements the ACS Capability with the optional Egress Control Vector and consumes more than one Bus Number shall implement the ACS Function Groups Enable (A) capability bit as 1b.

3.7.4. Address Translation Services Extended Capability (ATS)

ATS support is optional in SR-IOV devices. If a VF implements an ATS capability, its associated PF must implement an ATS capability. The ATS Capabilities in VFs and their associated PFs may be enabled independently.

PF and VF functionality is defined in the *Address Translation Services Specification* except where noted in Table 3-26. Attributes shown as ATS are the same as specified in the *Address Translation Services Specification.*

Table 3-26: ATS Capability Register

Bit Location	PF and VF Register Differences From ATS	PF Attributes	VF Attributes
20:16	**Smallest Translation Unit (STU)** Hardwired to 0 for VFs. PF value applies to all VFs.	ATS	RO
28:24	**Invalidate Queue Depth** Hardwired to 0 for VFs. Depth of shared PF input queue.	ATS	RO

ATS behavior is specified in the *Address Translation Services Specification*. ATS requests target the Function being invalidated. ATS responses will have a Routing ID field matching the targeted Function. Each Function is required to implement sufficient queuing to ensure it can hold the maximum number of outstanding Invalidation Requests from a TA (using either input or output queuing).

However, all VFs associated with a PF share a single input queue in the PF. To implement Invalidation flow control, the TA must ensure that the total number of outstanding Invalidate Requests to the shared PF queue (targeted to the PF and its associated VFs) does not exceed the value in the PF Invalidate Queue Depth field.

3.7.5. MR-IOV

PFs and VFs shall not contain an MR-IOV Capability.

If present, the MR-IOV Capability is contained in other Functions of the Component. See the *Multi-Root I/O Virtualization and Sharing Specification* for details.

4

4. Error Handling

Native IOV devices utilize the Error Reporting mechanism defined in the *PCI Express Base Specification*. Errors that are defined as non-Function specific are only logged in the PF.

The *PCI Express Base Specification* defines two error reporting paradigms: the baseline Capability and the Advanced Error Reporting Capability. The baseline error reporting capabilities are required of all PCI Express devices and define the minimum error reporting requirements. The Advanced Error Reporting Capability is optional and is defined for more robust error reporting and is implemented with a specific PCI Express Capability structure.

4.1. Baseline Error Reporting

All Native IOV devices must support the baseline error reporting capabilities, with some modifications to account for the goal of reduced cost and complexity of implementation.

These control bits are only meaningful in the PF. VFs shall use the error reporting control bits in the associated PF when making decisions on generating error messages.

These following fields are RsvdP in VFs:

❑ Command register (see Section 3.4.1.3)

- SERR# Enable
- Parity Error Response

❑ Device Control register (see Section 3.5.4)

- Correctable Reporting Enable
- Non-Fatal Reporting Enable
- Fatal Reporting Enable
- UR Reporting Enable

Each VF shall implement a mechanism to provide error status independent of any other Function. This is necessary to provide SI isolation for errors that are Function specific.

The following baseline error reporting status bits must be implemented in each VF:

- ❑ Status register (see Section 3.4.1.4)
  - Master Data Parity Error
  - Signaled Target Abort
  - Received Target Abort
  - Received Master Abort
  - SERR# Asserted
  - Detected Parity Error
- ❑ Device Status register (see Section 3.5.5)
  - Correctable Error Detected
  - Non-Fatal Error Detected
  - Fatal Error Detected
  - Unsupported Request Detected

Each VF shall use its own Routing ID when signaling errors.

4.2. Advanced Error Reporting

The Advanced Error Reporting Capability is optional. If AER is not implemented in the PF, it must not be implemented in the associated VFs. If AER is implemented in the PF, it is optional in the VFs.

The *PCI Express Base Specification* classifies errors as Function Specific and non-Function Specific. Each Virtual Function that implements the Advanced Error Reporting Capability must maintain its own error reporting status for function-specific errors.

4.2.1. VF Header Log

A Device that implements AER in the VFs may share Header Log Registers among VFs associated with a single PF. See Section 4.2.10 for details.

Header logging Registers for the PF are independent of its associated VFs and must be implemented with dedicated storage space.

When implementing a reduced set of Header Log Registers, a Function may not have room to log a header associated with an error. In this case, the Function shall update the Uncorrectable Error Status Register and Advanced Error Capabilities and Control register as required by the *PCI Express Base Specification*; however, when the Header Log Register is read, it shall return all 1's to indicate an overflow condition and no header was logged.

4.2.2. Advanced Error Reporting Capability

Figure 4-1 describes the AER extended capability structure.

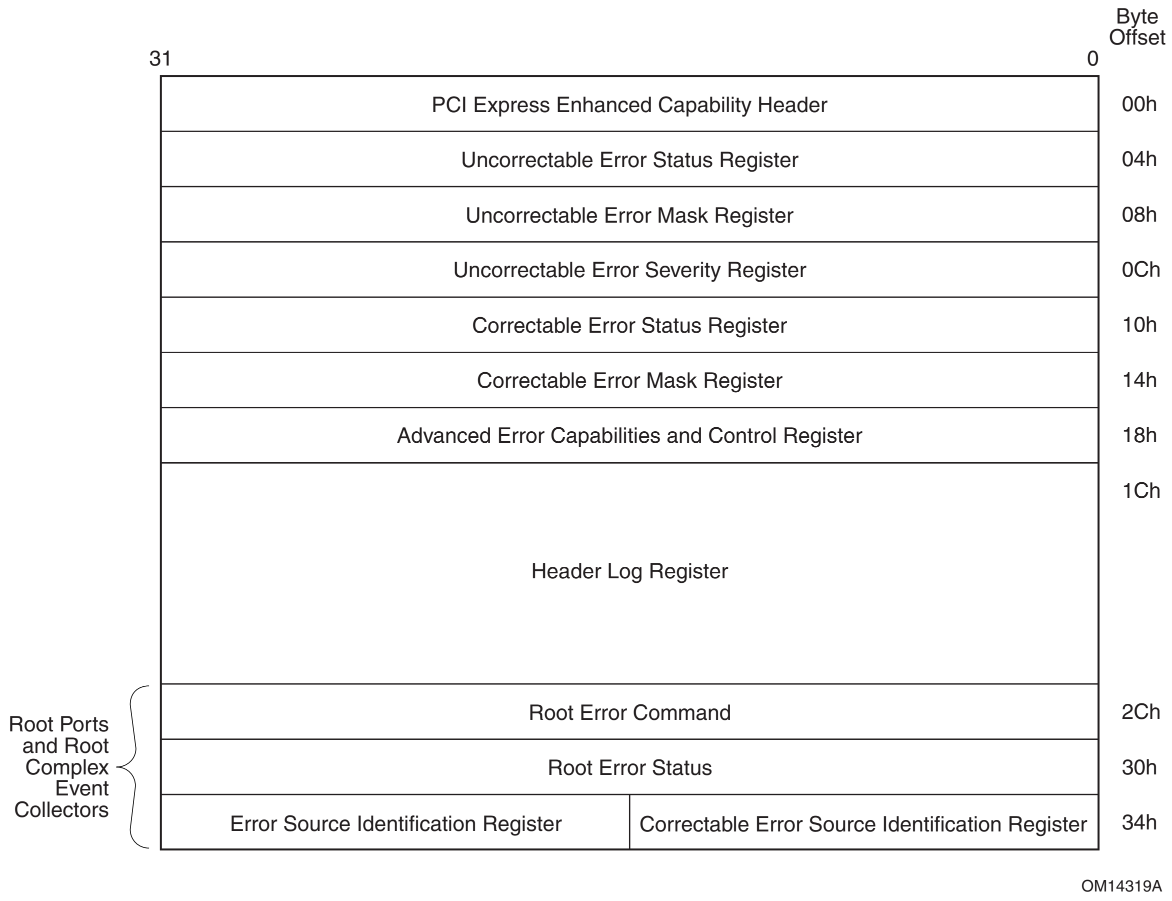
Byte Offset
31
0
PCI Express Enhanced Capability Header
00h
Uncorrectable Error Status Register
04h
Uncorrectable Error Mask Register
08h
Uncorrectable Error Severity Register
0Ch
Correctable Error Status Register
10h
Correctable Error Mask Register
14h
Advanced Error Capabilities and Control Register
18h
Header Log Register
1Ch
Root Ports and Root Complex Event Collectors
Root Error Command
2Ch
Root Error Status
30h
Error Source Identification Register
Correctable Error Source Identification Register
34h
OM14319A

Figure 4-1: Advanced Error Reporting Extended Capability Structure

4.2.3. Advanced Error Reporting Enhanced Capability Header (Offset 00h)

This register contains the PCI Express Extended Capability ID, Capability Version, and Next Capability Offset. These fields are unchanged from the *PCI Express Base Specification*.

4.2.4. Uncorrectable Error Status Register (Offset 04h)

The Uncorrectable Error Status register indicates error detection status of individual errors. Errors that are defined as non-Function specific are logged in the PF. Only Function-specific errors are logged in the VFs.

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 4-1.

Table 4-1: Uncorrectable Error Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	**Data Link Protocol Error Status**	Base	0b
5	**Surprise Down Error Status**	Base	0b
13	**Flow Control Protocol Error Status**	Base	0b
17	**Receiver Overflow Status**	Base	0b
18	**Malformed TLP Status**	Base	0b
19	**ECRC Error Status**	Base	0b

4.2.5. Uncorrectable Error Mask Register (Offset 08h)

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 4-2. For VF fields marked RsvdP, the PF setting applies to the VF. For VF fields marked 0b, the error is not applicable to a VF.

Table 4-2: Uncorrectable Error Mask Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	**Data Link Protocol Error Mask**	Base	0b
5	**Surprise Down Error Mask**	Base	0b
12	**Poisoned TLP Mask**	Base	RsvdP
13	**Flow Control Protocol Error Mask**	Base	0b
14	**Completion Timeout Error Mask**	Base	RsvdP
15	**Completer Abort Error Mask**	Base	RsvdP
16	**Unexpected Completion Error Mask**	Base	RsvdP
17	**Receiver Overflow Mask**	Base	0b
18	**Malformed TLP Mask**	Base	0b
19	**ECRC Error Mask**	Base	0b
20	**Unsupported Request Error Mask**	Base	RsvdP

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
21	**ACS Violation Mask**	Base	RsvdP

4.2.6. Uncorrectable Error Severity Register (Offset 0Ch)

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 4-3. For VF fields marked RsvdP, the PF setting applies to the VF. For VF fields marked 0b, the error is not applicable to a VF.

Table 4-3: Uncorrectable Error Severity Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
4	**Data Link Protocol Error Severity**	Base	0b
5	**Surprise Down Error Severity**	Base	0b
12	**Poisoned TLP Severity**	Base	RsvdP
13	**Flow Control Protocol Error Severity**	Base	0b
14	**Completion Timeout Error Severity**	Base	RsvdP
15	**Completer Abort Error Severity**	Base	RsvdP
16	**Unexpected Completion Error Severity**	Base	RsvdP
17	**Receiver Overflow Severity**	Base	0b
18	**Malformed TLP Severity**	Base	0b
19	**ECRC Error Severity**	Base	0b
20	**Unsupported Request Error Severity**	Base	RsvdP
21	**ACS Violation Severity**	Base	RsvdP

4.2.7. Correctable Error Status Register (Offset 10h)

The Correctable Error Status register indicates error detection status of individual correctable errors. Errors that are defined as non-Function specific are logged in the PF. Only Function-specific errors are logged in the VFs.

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 4-4.

Table 4-4: Correctable Error Status Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Receiver Error Status**	Base	0b
6	**Bad TLP Status**	Base	0b
7	**Bad DLLP Status**	Base	0b
8	**REPLAY_NUM Rollover Status**	Base	0b
12	**Replay Timer Timeout Status**	Base	0b

4.2.8. Correctable Error Mask Register (Offset 14h)

Correctable Error Mask bits are only implemented in the PF. The VFs will use the error reporting control bits in the associated PF.

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 4-5. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 4-5: Correctable Error Mask Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
0	**Receiver Error Mask**	Base	RsvdP
6	**Bad TLP Mask**	Base	RsvdP
7	**Bad DLLP Mask**	Base	RsvdP
8	**REPLAY_NUM Rollover Mask**	Base	RsvdP
12	**Replay Timer Timeout Mask**	Base	RsvdP
13	**Advisory Non-Fatal Error Mask**	Base	RsvdP

4.2.9. Advanced Error Capabilities and Control Register (Offset 18h)

PF and VF functionality is defined in the *PCI Express Base Specification* except where noted in Table 4-6. For VF fields marked RsvdP, the PF setting applies to the VF.

Table 4-6: Advanced Error Capabilities and Control Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
6	**ECRC Generation Enable**	Base	RsvdP
8	**ECRC Check Enable**	Base	RsvdP

4.2.10. Header Log Register (Offset 1Ch)

The Header Log register captures the header for the TLP corresponding to a detected error. See also Section 4.2.1.

A Device that implements AER in the VFs may share Header Log Registers among VFs associated with a single PF. A shared header log must have storage for at least one header.

Header logging Registers for the PF is independent of its associated VFs and must be implemented with dedicated storage space.

When an error is detected in a VF, the error shall be logged as specified in the PCI Express Base Specification. If a shared set of Header Log Registers is implemented, a VF may not have room to log a header. In this case, the VF shall update its Uncorrectable Error Status Register and Advanced Error Capabilities and Control register as required by the *PCI Express Base Specification*; however, when that VF's Header Log Register is read, it shall return all 1's to indicate an overflow condition.

The VF's header log entry shall be locked and remain valid while that VF's First Error Pointer is valid. As defined in the *PCI Express Base Specification*, the First Error Pointer register is valid when the corresponding bit of the Uncorrectable Error Status register is Set. While the header log entry is locked, additional errors shall not overwrite the locked entry for this or any other VF. When a header entry is unlocked, it shall be available to record a new error for any VF sharing the header logs.

Figure 4-2 and Table 4-7 describe the Header Log associated with the AER extended capability.

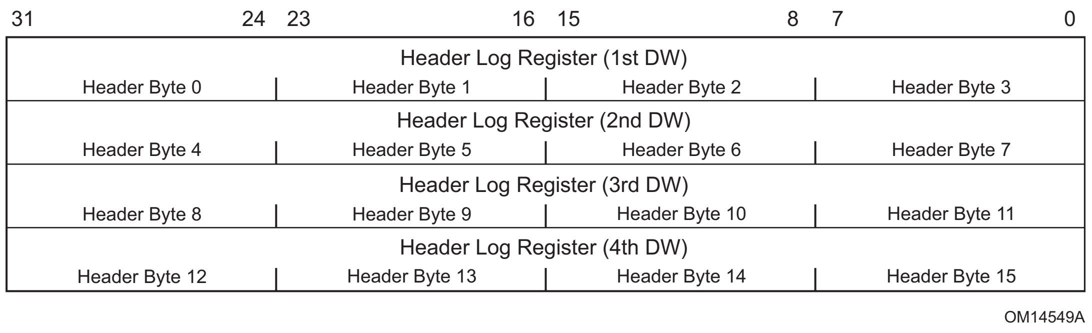

Figure 4-2: Header Log Register

Table 4-7: Header Log Register

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
127:0	**Header of TLP associated with error**	Base	Base (see above)

4.2.11. Root Error Command Register (Offset 2Ch)

This register is not applicable to Devices.

4.2.12. Root Error Status Register (Offset 30h)

This register is not applicable to Devices.

4.2.13. Correctable Error Source Identification Register (Offset 34h)

This register is not applicable to Devices.

4.2.14. Error Source Identification Register (Offset 36h)

This register is not applicable to Devices.

5. Interrupts

SR-IOV capable devices utilize the same Interrupt signaling mechanisms defined in the *PCI Express Base Specification.*

5.1. Interrupt Mechanisms

There are three methods of signaling interrupts:

❑ INTx

❑ MSI

❑ MSI-X

PFs may implement INTx per the *PCI Express Base Specification.* VFs must not implement INTx. PFs and VFs shall implement MSI or MSI-X or both if interrupt resources are requested. Each PF and VF must implement its own unique interrupt Capabilities.

5.1.1. MSI Interrupts

MSI Capability is defined in the *PCI Local Bus Specification, Revision 3.0.*

PF and VF functionality is defined in the PCI Express Base Specification except where noted in Table 5-1.

Table 5-1: MSI Capability: Message Control

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
8	**Per-Vector Masking Capable**	1b	1b

5.1.2. MSI-X Interrupts

The MSI-X Capability is defined in the *PCI Local Bus Specification, Revision 3.0* and depicted in Figure 5-1.

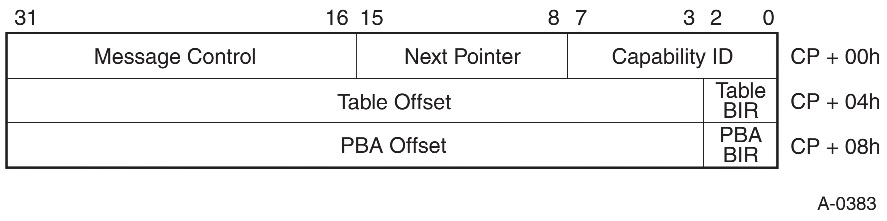

Figure 5-1: MSI-X Capability

PF and VF functionality is identical to that of a Function as defined in the *PCI Express Base Specification.*

Note that the Table Offset and PBA Offset values are relative to the VF's Memory address space.

5.1.3. Address Range Isolation

If a BAR that maps address space for the MSI-X Table or MSI-X PBA also maps other usable address space that is not associated with MSI-X structures, locations (e.g., for CSRs) used in the other address space must not share any naturally aligned System Page Size address range with one where either MSI-X structure resides. The MSI-X Table and MSI-X PBA are permitted to co-reside within a naturally aligned System Page Size address range, though they must not overlap with each other.

6

6. Power Management

This chapter defines the PCI Express SR-IOV power management capabilities and protocols.

The Power Management Capability is required for PFs as described in the *PCI Express Base Specification* and the *PCI Bus Power Management Interface Specification*.

For VFs, the Power Management Capability is optional.

6.1. VF Device Power Management States

If a VF does not implement the Power Management Capability, then the VF behaves as if it had been programmed into the equivalent power state of its associated PF.

If a VF implements the Power Management Capability, the functionality is defined in the *PCI Express Base Specification* except as noted in Section 6.4.

If a VF implements the Power Management Capability, the Device behavior is undefined if the PF is placed in a lower power state than the VF. Software should avoid this situation by placing all VFs in lower power state before lowering their associated PF's power state.

A VF in the D0 state is in the $D0_{active}$ state when any single or combination of the VF's Bus Master Enable bit (see Section 3.4.1.3) and the VF MSE bit in the SR-IOV Control (see Section 3.3.3) Extended Capability have been Set.

6.2. PF Device Power Management States

The PF's power management state (D-state) has global impact on its associated VFs. If a VF does not implement the Power Management Capability, then it behaves as if it is in an equivalent power state of its associated PF.

If a VF implements the Power Management Capability, the Device behavior is undefined if the PF is placed in a lower power state than the VF. Software should avoid this situation by placing all VFs in lower power state before lowering their associated PF's power state.

When the PF is placed into the $D3_{hot}$ state:

- ❑ If the No_Soft_Reset bit is Clear then the PF performs an internal reset on the $D3_{hot}$ to D0 transition and all its configuration state returns to the default values.

  Note: Resetting the PF resets VF Enable which means that VFs no longer exist and any VF specific context is lost after the $D3_{hot}$ to D0 transition is complete.

- ❑ If the No_Soft_Reset bit is Set then the internal reset does not occur. The SR-IOV extended capability retains state, and associated VFs remain enabled.

When the PF is placed into the $D3_{cold}$ state VFs no longer exist, any VF specific context is lost and PME events can only be initiated by the PF.

6.3. Link Power Management State

VF Power State does not affect Link Power State.

Link Power State is controlled solely by the setting in the PFs regardless of the VF's D-state.

6.4. VF Power Management Capability

The following tables list the requirements for PF and VFs Power Management Capability.

PF and VF functionality is defined in the *PCI Express Base Specification* and *PCI Power Management Interface Specification* except where noted in Table 6-1.

Table 6-1: Power Management Control/Status (PMCSR)

Bit Location	PF and VF Register Differences From Base	PF Attributes	VF Attributes
14:13	**Data_Scale**	Base	00b
12:9	**Data_Select**	Base	0000b
3	**No_Soft_Reset** – This value must be identical in the PF and its associated VFs.	Base	Base

Acknowledgements

The following people were instrumental in the development of the *Single Root I/O Virtualization and Sharing Specification*[1].

Boon Ang, VMware Corporation

Antonio Asaro, Advanced Micro Devices, Inc.

Shawn Clayton, Emulex Corporation

Eric DeHaemer, Intel Corporation

Robert Dickson, Sun Microsystems, Inc.

Jeff Fose, Emulex Corporation

Douglas Freimuth, IBM Corporation

Steve Glaser, NextIO, Inc.

Lucian Gozu, Neterion Corporation

Andrew Gruber, Advanced Micro Devices, Inc.

Mark Hummel, Advanced Micro Devices, Inc.

David Kahn, Sun Microsystems, Inc.

Kwok Kong, IDT Corporation

Michael Krause, Hewlett-Packard Company

Brian Langendorf, Nvidia Corporation

Mallik Mahalingam, VMware, Inc.

Paul Mattos, IBM Corporation

David Mayhew, Stargen, Inc.

Richard Moore, QLogic Corporation

Peter Onufyrk, IDT Corporation

Jake Oshins, Microsoft Corporation

Chris Pettey, NextIO, Inc.

Renato Recio, IBM Corporation

Jack Regula, PLX Corporation

Wesley Shao, Sun Microsystems, Inc.

Richard Solomon, LSI Corporation

[1] Company affiliation is at the time of specification contribution.

Steven Thurber, IBM Corporation

Mahesh Wagh, Intel Corporation

Jim Williams, Emulex Corporation

Theodore Willke, Intel Corporation

David Wooten, Microsoft Corporation

William Wu, Broadcom Corporation