# Working Draft American National Standard

# Project T10/2240-D

Revision 6e
21 December 2012

# Information technology - PCI Express® Queuing Interface (PQI)

T10 Technical Editor:
Ie Wei Njoo
PMC-Sierra
1380 Bordeaux Drive
Sunnyvale, CA 94089
USA

Telephone: (408) 239-8273
Facsimile: (408) 492-9431
Email: iewei_njoo@pmc-sierra.com

**Reference number**
**ISO/IEC 14776-171:20xx**
**ANSI INCITS ***-20xx**

## Points of Contact

**InterNational Committee for Information Technology Standards (INCITS) T10 Technical Committee**

**T10 Chair**
John B. Lohmeyer
LSI Logic
4420 Arrows West Drive
Colorado Springs, CO 80907-3444
USA

Telephone: (719) 533-7560
Email: lohmeyer@t10.org

**T10 Vice-Chair**
Mark S. Evans
Western Digital Corporation
5863 Rue Ferrari
San Jose, CA 95138
USA

Telephone:(408) 363-5257
Email: mark.evans@wdc.com

T10 Web Site: http://www.t10.org

T10 E-mail reflector:
Server: majordomo@t10.org
To subscribe send e-mail with “subscribe” in message body
To unsubscribe send e-mail with “unsubscribe” in message body

**INCITS Secretariat**
1101 K Street, NW
Suite 610
Washington, DC 20005
USA

Telephone: 202-737-8888
Web site: http://www.incits.org
Email: incits@itic.org

**Information Technology Industry Council**
Web site: http://www.itic.org

**Document Distribution**
INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105
USA

Web site: http://www.techstreet.com/incitsgate.tmpl
Telephone: (734) 302-7801 or (800) 699-9277

Global Engineering Documents, an IHS Company
15 Inverness Way East
Englewood, CO 80112-5704
USA

Web site: http://global.ihs.com
Telephone: (303) 397-7956 or (303) 792-2181 or (800) 854-7179

American National Standard
for Information Technology

# PCI Express® Queuing Interface (PQI)

Secretariat
**Information Technology Industry Council**

Approved mm.dd.yy

American National Standards Institute, Inc.

## ABSTRACT

This standard defines a circular queue interface for transferring information between a PQI host and a PQI device over the PCI Express architecture, and defines a scatter gather list (SGL) format that is used to describe data buffers.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

**CAUTION NOTICE:** This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.
As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by
**American National Standards Institute**
**11 W. 42nd Street, New York, New York 10036**

Printed in the United States of America

# Revision Information

## R.1 Revision pqi-r00 (14 July 2011)

This is revision 0 of this working draft taken from 11-292r0.

## R.2 Revision pqi-r00a (25 July 2011)

The following T10 approved proposals were incorporated in PQI in this revision:

a) 11-166r6 - PQI Scatter Gather List and Descriptors (Steve Johnson, LSI Corp).

## R.3 Revision pqi-r01 (26 September 2011)

The following T10 approved proposals were incorporated in PQI in this revision:

a) 11-285r5 PQI object locations (Rob Elliott, HP);
b) 11-308r2 SOP PQI Bit and byte ordering conventions (Rob Elliott, HP); and
c) 11-209r4 PQI Minimizing host SGL copies (Rob Elliott, HP).

## R.4 Revision pqi-r02 (23 November 2011)

Incorporated these changes per the November 2011 SOP PQI WG (11-504r0) and T10 plenary (11-482r1):

a) 11-467r0 PQI defined registers (Ie Wei Njoo, PMC-Sierra);
b) 11-414r5 Create and delete Admin queue (Ie Wei Njoo, PMC-Sierra); and
c) 11-473r1 PQI Controller Specific SGL Type (David L. Black, EMC).

## R.5 Revision pqi-r02a (23 December 2011)

Incorporated these changes per the November - December 2011 SOP PQI WG (11-513r0):

a) 11-384r5 PQI interrupts (Rob Elliott, HP). Changed the COALESCING TRIGGER bit to WAIT FOR REARM bit for clarity and added editor note for possible future changes;
b) 11-469r3 PQI IU common format (Ie Wei Njoo, PMC-Sierra);
c) 12-008r1 PQI signature register (Rob Elliott, HP).

Other changes include the reorganization of sub clauses in clause 4 and clause 6. Moved the model for the circular queue from clause 6 into clause 4, so it is ahead of the register definitions in clause 5 that support it.

These proposals have not yet been approved by a T10 plenary.

## R.6 Revision pqi-r03 (20 January 2012)

Incorporated these changes per T10 plenary (12-048r0):

a) 12-009r0 PQI administrator NULL IU (Rob Elliott, HP). This change was listed in rev 2a revision history but the content did not make it into the document;
b) 11-464r2 PQI operation codes (Rob Elliott, HP);
c) 12-006r1 Circular queue CI and PI sizes and updates (Rob Elliott, HP);
d) 11-470r5 Add and remove PQI elements (Ie Wei Njoo, PMC-Sierra); and
e) 11-372r6 PQI device states (Ie Wei Njoo, PMC-Sierra).

## R.7 Revision pqi-r03a (17 February 2012)

Incorporated these changes per the January - February 2012 SOP PQI WG (12-099r0):

a) 12-007r4 PQI - REPORT LIST OF IQ and REPORT LIST OF OQ admin functions (Ie Wei Njoo, PMC-Sierra).

This proposal has not yet been approved by a T10 plenary.

## R.8 Revision pqi-r04 (16 March 2012)

Incorporated these changes per T10 plenary (12-155r0):

a) update the pqi-r03a revision to pqi-r04 as approved by T10 plenary.

This draft revision also includes editorial changes for clarification and proper T10 document styles.

## R.9 Revision pqi-r04a (21 March 2012)

Incorporated these changes per the March 8 2012 SOP PQI WG (12-154r0):

a) 11-294r5 PQI - Operational queues termination (DELETE OPERATIONAL IQ function and DELETE OPERATIONAL OQ function) (Ie Wei Njoo, PMC-Sierra);
b) 12-013r3 PQI - REPORT MANUFACTURING INFORMATION function (Ie Wei Njoo, PMC-Sierra);
c) 12-110r2 PQI - Update to PQI request IU format (Ie Wei Njoo, PMC-Sierra);
d) 12-111r1 PQI - CONFIGURE PROPERTY OPERATIONAL IQ function (Ie Wei Njoo, PMC-Sierra);
e) 12-133r0 PQI - Update STATUS field definition in response IU (Ie Wei Njoo, PMC-Sierra); and
f) 12-138r1 PQI - 16-bit ELEMENT LENGTH field (Ie Wei Njoo, PMC-Sierra).

This draft revision includes editorial changes for clarification and proper T10 document styles.

## R.10 Revision pqi-r04b (20 April 2012)

Incorporated these changes per the April 10-12 2012 SOP PQI WG (12-215r0):

a) 11-295r8 PQI - REPORT PQI DEVICE CAPABILITY function (Ie Wei Njoo, PMC-Sierra);
b) 12-200r0 PQI- CONFIGURE PROPERTY OPERATIONAL OQ response (Ie Wei Njoo, PMC-Sierra);
c) 12-206r1 SOP PQI - SGL segment clarifications (Rob Elliott, HP);
d) 12-180r2 PQI circular queue address field names (Rob Elliott, HP);
e) 12-183r2 PQI SOP circular queue UML model (Rob Elliott, HP);
f) 12-181r1 PQI overview clause rewrite (Rob Elliott, HP);
g) 12-210r1 PQI Queue priority (Mark Evans, WDC); and
h) 12-216r0 PQI Rename PCI Memory BAR (Richard Solomon, LSI).

This draft revision includes editorial changes for clarification and proper T10 document styles.

## R.11 Revision pqi-r05 (14 May 2012)

Incorporated these changes per T10 plenary (12-239r0):

a) update the pqi-r04b revision to pqi-r05 as approved by T10 plenary.

## R.12 Revision pqi-r05a (17 May 2012)

Incorporated these changes per the May 10-11 2012 SOP PQI WG (12-262r0):

a) 12-202r2 PQI SGL type support for PQI IUs and other IUs (Ie Wei Njoo, PMC-Sierra);
b) 12-254r0 PQI Queue ID field offset fix (Ie Wei Njoo, PMC-Sierra);
c) 11-169r4 SOP PQI SPC-4 overflow and underflow handling (Rob Elliott, HP); and
d) 12-195r2 PQI sleep, shutdown, and reboot notification (Rob Elliott, HP);

This draft revision includes editorial changes for clarification and proper T10 document styles. Including in these editorial changes are more than 200 comments from Rob Elliott.

The NUMBER OF ELEMENT fields in the administrator IUs and parameters data have been adjusted from a 32-byte field to a 16-byte field to match the intended goal of the approved proposal 12-183. Shortening the field caused some fields to shift rather than just marked the unused bytes as reserved.

Some sub-sections in section 7 and section 8 have been rearranged for a cleaner flow.

## R.13 Revision pqi-r05b (2 July 2012)

Incorporated these changes per the June 13-15 2012 SOP PQI WG (T10/12-288r0):

a) 12-292r0 (12-194r9 roll-over) PQI Device Status register and Error Data register (Ie Wei Njoo, PMC-Sierra);
b) 12-271r2 PQI administrator function error handling with SOP coordination (Rob Elliott, HP);
c) 12-276r2 PQI Interrupt coalescing timer granularity (Satish Vasudeva, PMC-Sierra);
d) 12-291r1 PQI CI and PI initialization during queue creation (Stephen Finch, Western Digital);
e) 12-286r2 PQI Adding vendor specific field in the administrator request IU and parameter data (Ie Wei Njoo, PMC-Sierra);
f) 11-397r2 PQI PCI Express Miscellanea (Richard Solomon, LSI); and
g) 12-212r4 PQI Memory Alignments (Brad Besmer, LSI).

Some sub-sections in section 8 have been moved into a new section 9 for a cleaner document organization.

This draft revision includes editorial changes for clarification and proper T10 document styles.

The CONFIGURE OPERATIONAL IQ PROPERTIES name has been changed to CHANGE OPERATIONAL IQ PROPERTIES, and the CONFIGURE OPERATIONAL OQ PROPERTIES name has been change to CHANGE OPERATIONAL OQ PROPERTIES.

## R.14 Revision pqi-r06 (27 July 2012)

Incorporated these changes per the SOP-PQI Working Group Meeting - 18-19 July 2012 (T10/12-339r0) and T10 plenary (12-327r0):

a) 12-038r8 PQI Interrupt Coalescing (David Geddes, Marvell);
b) 12-088r9 SOP PQI Reset (Ie Wei Njoo, PMC-Sierra);
c) 12-295r1 PQI changes to CONFIGURE OPERATIONAL OQ PROPERTIES request (Ie Wei Njoo, PMC-Sierra); and
d) 12-340r1 PQI Memory Alignment for SGL Segment Descriptors (Brad Besmer, LSI).

This draft revision includes editorial changes during the 7/19/2012 work group meeting and other editorial changes for clarification and proper T10 document styles.

## R.15 Revision pqi-r06a (11 September 2012)

Revision 6a is the first revision in the resolution process for the letter ballot held on revision 6. It coordinates with letter ballot revision document 12-373r0 (12-373r0.pdf, 12-373r0.xls, and 12-373r0.fdf).

Change bars indicate changes from revision 6. Change bars are cumulative for the entire letter ballot comments resolution process. To make the document more readable, the track text edit is enabled only to track changes from one minor revision to the next minor revision.

## R.16 Revision pqi-r06b (28 September 2012)

Revision 6b coordinates with letter ballot revision document 12-373r1 (12-373r1.pdf, 12-373r1.xls, and 12-373r1.fdf).

## R.17 Revision pqi-r06c (9 October 2012)

Revision 6c coordinates with letter ballot revision document 12-373r2 (12-373r2.pdf, 12-373r2.xls, and 12-373r2.fdf).

## R.18 Revision pqi-r06d (2 November 2012)

Revision 6d coordinates with letter ballot revision document 12-373r3 (12-373r3.pdf, 12-373r3.xls, and 12-373r3.fdf). The content of 12-388r4 (not officially voted) has also be incorporated.

## R.19 Revision pqi-r06e (21 December 2012)

Revision 6e coordinates with letter ballot revision document 12-373r4 (12-373r4.pdf, 12-373r4.xls, and 12-373r4.fdf). The content of 12-426r3 (not officially voted) has also been incorporated.This revision also includes more than two hundreds additional late letter ballot comments from Rob Elliott (HP).

## Contents

# Tables

## Figures

# FOREWORD (This foreword is not part of this standard)

This standard defines a queuing interface for transferring information units over PCI Express® architecture. This standard may be used in conjunction with the SCSI over PCI Express architecture (SOP) standard.

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, International Committee for Information Technology Standards, Information Technology Industry Council, Suite 610, 1101 K Street, NW, Washington, DC 20005.

This standard was processed and approved for submittal to ANSI by the International Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, INCITS had the following members:

INCITS Technical Committee T10 on SCSI Storage Interfaces, which developed and reviewed this standard, had the following members:

John B. Lohmeyer, Chair
Mark Evans, Vice-Chair
Ralph O. Weber, Secretary

# INTRODUCTION

The PCI Express® Queuing Interface (PQI) standard is divided into the following clauses:

Clause 1 (AMERICAN NATIONAL STANDARD INCITS ***-20xx) describes the relationship of this standard to the SCSI standards and PCI Express standards and specifications.

Clause 2 (Normative references) provides references to other standards and documents.

Clause 3 (Definitions, symbols, abbreviations, and conventions) defines terms and conventions used throughout this standard.

Clause 4 (General) describes architecture, queuing model, OQ service notification methods, PD (PQI device) state machine, register based error information and PQI reset.

Clause 5 (PCI Express requirements and PQI device registers PCI Express architecture usage by this standard) describes the register access and PQI device registers.

Clause 6 (Queuing layer) describes the PQI queuing layer.

Clause 7 (SGL (scatter gather list)) describes the SGL format ~~supported by this standard~~.

Clause 8 (Information unit layer) describes the PQI information unit layer.

Clause 9 (Administrator IUs and administrator functions) describes the administrator functions.

Annex A (Alternative SGL segment) describes the alternative SGL segment.

Annex B (Operating system suggestions) describes the operating system suggestions for the power actions.

Figure 0 shows a layered view of the clauses of this standard.

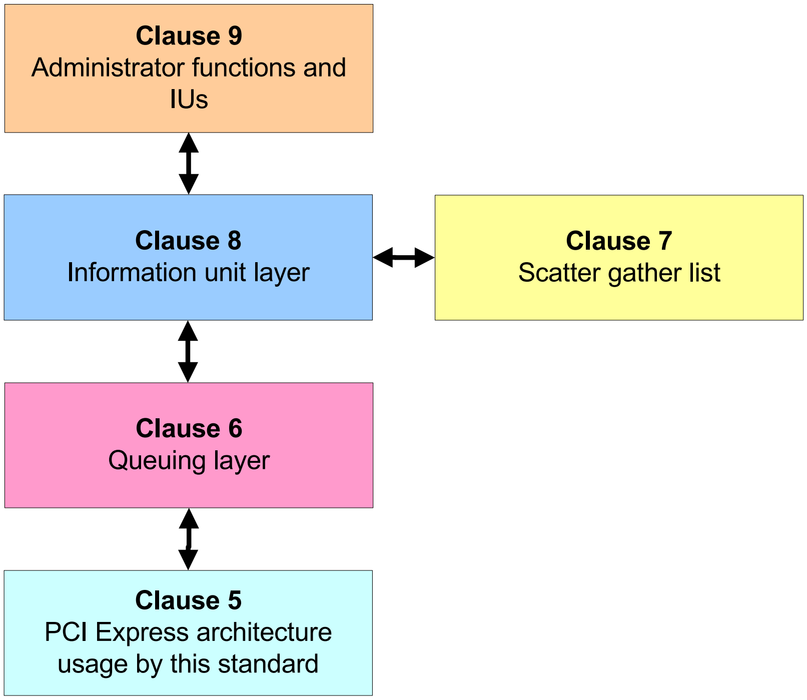


**Figure 0 — Layered view of the clauses in this standard**

## AMERICAN NATIONAL STANDARDINCITS ***-20xx

## American National Standard
## for Information Technology -

## PCI Express® Queuing Interface (PQI)

# 1 Scope

The SCSI family of standards provides for different transport protocols that define the methods for exchanging information between SCSI devices. This standard defines the transport methods for exchanging information between SCSI devices using a PCI Express interconnect. This standard defines a queuing layer, used by SOP. Other SCSI transport protocol standards define the methods for exchanging information between SCSI devices using other interconnects. Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards.

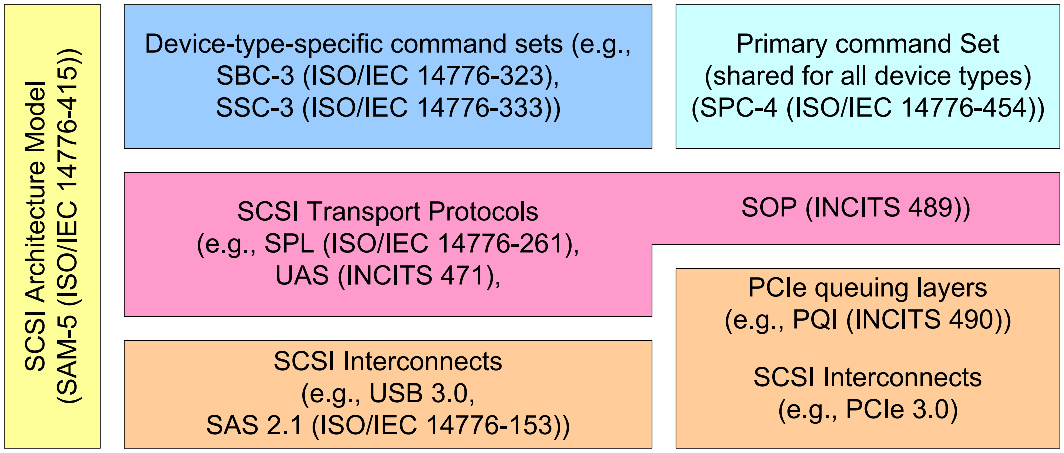

SCSI Architecture Model
(SAM-5 (ISO/IEC 14776-415))
Device-type-specific command sets (e.g.,
SBC-3 (ISO/IEC 14776-323),
SSC-3 (ISO/IEC 14776-333))
Primary command Set
(shared for all device types)
(SPC-4 (ISO/IEC 14776-454))
SCSI Transport Protocols
(e.g., SPL (ISO/IEC 14776-261),
UAS (INCITS 471),
SOP (INCITS 489))
PCIe queuing layers
(e.g., PQI (INCITS 490))
SCSI Interconnects
(e.g., USB 3.0,
SAS 2.1 (ISO/IEC 14776-153))
SCSI Interconnects
(e.g., PCIe 3.0)

**Figure 1 — SCSI document relationships**

PCI Express is used as the interconnect for this standard. This standard defines a generic queuing model to allow transporting IUs.

# 2 Normative references

## 2.1 Normative references overview

Referenced standards and specifications contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents may be obtained from ANSI:

a) approved ANSI standards;
b) approved and draft international and regional standards (ISO, IEC, CEN/ELEC, ITU-T); and
c) approved and draft foreign standards (including BSI, JIS, and DIN).

For further information, contact ANSI Customer Service Department at 212-642-4980 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

Additional availability contact information is provided below as needed.

## 2.2 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated.

ISO/IEC 14776-415, *SCSI Architecture Model - 5 (SAM-5)* (T10/2104-D)

ISO/IEC 14776-454, *SCSI Primary Commands - 4 (SPC-4)* (T10/1731-D)

ISO/IEC 14776-271, *SCSI Over PCI~~e~~ Express® ~~architecture~~ (SOP)* (T10/2239-D)

NOTE 1 - For more information on the current status of these documents, contact the INCITS Secretariat at 202-737-8888 (phone), 202-638-4922 (fax) or via Email at incits@itic.org. To obtain copies of these documents, contact Global Engineering at 15 Inverness Way, East Englewood, CO 80112-5704 at 303-792-2181 (phone), 800-854-7179 (phone), or 303-792-2192 (fax) or see http://www.incits.org.

## 2.3 Other references

For information on the current status of the listed documents, or regarding availability, contact the indicated organization.

*PCI Local Bus Specification Revision 3.0 (PCI), 3 February ~~3,~~ 2004*

*PCI Express® Base Specification Revision 3.0 (PCIe), 10 November ~~10,~~ 2010*

*PCI Code and ID Assignment Specification (PCI-ID) Revision 1.2, 15 March~~,~~ 2012*

*PCI Bus Power Management Interface Specification (PCI-PM) revision 1.2, 3 March ~~3,~~ 2004*

*PCI Express Card Electromechanical Specification (PCIe-CEM) revision 3.0, version 0.9, 23 May ~~23,~~ 2011*

NOTE 2 - For information on the current status of PCI documents, contact the Peripheral Component Interconnect Special Interest Group (PCI-SIG) (see http://www.pcisig.com).

*Advanced Configuration and Power Interface Specification (ACPI). Revision 5.0, 6 December 2011*

NOTE 3 - For information on the current status of ACPI documents, contact the ACPI Promoters (see http://www.acpi.info).

# 3 Definitions, symbols, abbreviations, and conventions

## 3.1 Definitions

### 3.1.1 administrator function (see 9.2)

unit of work to be performed by a PQI management device server

### 3.1.2 administrator queue pair

administrator IQ and administrator OQ

### 3.1.3 aggregation (see 3.9.4)

when used in class diagrams, ~~a~~ form of association that defines a whole-part relationship between the whole (i.e., aggregate) and its parts

### 3.1.4 alternative SGL segment (see annex A)

data structure in a contiguous region of memory space describing all or part of a data buffer

### 3.1.5 association

when used in class diagrams, ~~a~~ relationship between two or more classes that specifies connections among their objects (i.e., relationship that specifies that objects of one class are connected to objects of another class)~~.~~

### 3.1.6 attribute

when used in class diagrams, ~~a~~ named property of a class that describes ~~the~~a range of values that the class or its objects may hold; when used in object diagrams, ~~a~~ named property of an instance of a class~~.~~

### 3.1.7 big-endian (see 3.7)

format for storage of binary data in which the most significant byte ~~appears~~is first

### 3.1.8 byte

sequence of eight contiguous bits considered as a unit

### 3.1.9 circular queue (see 4.3.2)

data structure with an element array (see 4.3.2.1), consumer index ~~(see 3.1.13)~~, and producer index ~~(see 3.1.60)~~, that is used as if the first element follows the last element

### 3.1.10 configuration space (see PCI and PCIe)

byte addressable physical address space defined by PCI used to access registers of PCI functions (see PCI), addressed using ID-based routing (i.e., bus number, device number, if any, and function number), and accessible with configuration transactions

### 3.1.11 configuration transaction (see PCIe)

Request (see PCIe) and one or more Completions (see PCIe) used to access configuration space

### 3.1.12 consumer

entity that reads an occupied element from a circular queue and increments the CI

### 3.1.13 consumer index (CI) (see 4.3.2.1)

index of a circular queue element array that points to the next element to be read by the consumer

### 3.1.14 constraint

when used in class diagrams and object diagrams, ~~a~~ mechanism for specifying semantics or conditions that are maintained as true between entities (e.g., a required condition between associations)

### 3.1.15 class

description of a set of objects that share the same attributes, operations, relationships, and semantics; classes may have attributes and may support operations

**3.1.16 data block**
contiguous region of memory space

**3.1.17 data buffer**
one or more data blocks

**3.1.18 Data Buffer**
Data-In Buffer or Data-Out Buffer

**3.1.19 Data-In Buffer**
buffer specified by a PQI management application client to receive data from ~~the~~a PQI management device server

**3.1.20 Data-Out Buffer**
buffer specified by a PQI management application client to supply data that is transferred to ~~the~~a PQI management device server

**3.1.21 element**
set of contiguous bytes in memory space ~~(e.g., a part of a circular queue)~~ that are part of a circular queue

**3.1.22 element array**
set of contiguous elements

**3.1.23 error code**
a combination of the ERROR CODE field (see 4.6) and the ERROR CODE QUALIFIER field (see 4.6)

**3.1.24 field**
group of one or more contiguous bits that are part of a larger structure such as an IU

**3.1.25 generalization**
when used in class diagrams, ~~a~~ relationship among classes where one class (i.e., superclass) shares the attributes and/or operations on one or more classes (i.e., subclasses)

**3.1.26 inbound IU**
IU ~~(see 3.1.28)~~ contained in an IQ ~~(see 3.1.27)~~

**3.1.27 inbound queue (IQ)** (see 4.3.2.2)
circular queue that is used to transfer IUs from ~~the~~a PQI host to ~~the~~a PQI device

**3.1.28 information unit (IU)** (see 8.1)
delimited and sequenced set of information in a format appropriate for transport by the service delivery subsystem

**3.1.29 interrupt coalescing** (see 4.4.2)
technique for reducing the number of interrupts

**3.1.30 IQ ID** (see 4.1)
identifier assigned to an IQ

**3.1.31 least significant bit (LSB)**
in a binary code, ~~the~~ bit or bit position with the smallest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 0001b, the bit that is set to one)

**3.1.32 legacy INTx** (see PCI and PCIe)
interrupt mechanism defined by PCI and supported by PCI Express using virtual INTx wires

**3.1.33 little-endian** (see 3.7)
format for storage of binary data in which the least significant byte ~~appears~~is first

**3.1.34 memory read transaction** (see PCIe)
memory transaction used to read from memory space

**3.1.35 memory space** (see PCI and PCIe)
64-bit byte-addressable physical address space defined by PCI, addressed using address based routing (i.e., a 64-bit address), and accessible with memory transactions ~~(see 3.1.36)~~

**3.1.36 memory transaction** (see PCIe)
Request (see PCIe) and one or more Completions (see PCIe) used to access memory space ~~(see 3.1.35)~~

**3.1.37 memory write transaction** (see PCIe)
memory transaction used to write to memory space

**3.1.38 most significant bit (MSB)**
in a binary code, ~~the~~ bit or bit position with the largest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 1000b, the bit that is set to one)

**3.1.39 MSI-X Table** (see PCI and PCIe)
structure containing MSI-X vector control, message address, and message data registers

**3.1.40 MSI-X Pending Bit Array (PBA)** (see PCI)
structure containing MSI-X pending bits

**3.1.41 multiplicity**
when used in class diagrams, ~~an~~ indication of the range of allowable instances that a class or an attribute may have

**3.1.42 non-data administrator function**
administrator function that does not transfer data

**3.1.43 object**
entity with a well-defined boundary and identity that encapsulates state and behavior; all objects are instances of classes (i.e., a concrete manifestation of a class is an object)

**3.1.44 occupied element**
element that contains valid data

**3.1.45 operation**
when used in class diagrams, ~~a~~ service that may be requested from any object of the class in order to effect behavior

**3.1.46 OQ ID** (see 4.1)
identifier assigned to an OQ

**3.1.47 outbound IU**
IU ~~(see 3.1.28)~~ contained in an OQ ~~(see 3.1.48)~~

**3.1.48 outbound queue (OQ)** (see 4.3.2.3)
circular queue that is used to transfer IUs from ~~the~~a PQI device ~~(see 3.1.53)~~ to ~~the~~a PQI host ~~(see 3.1.56)~~

**3.1.49 PCI Express hierarchy domain** (see PCIe)
the part of an I/O system (e.g., PCI Express switches and PCI Express links) that allows a PQI host and a PQI device to communicate

### 3.1.50 PCI Express reset (see PCIe)

PCI Express cold reset (see PCIe), PCI Express function-level reset (see PCIe), PCI Express hard reset (see PCIe), PCI Express hot reset (see PCIe), PCI Express soft reset (see PCIe), or PCI Express warm reset (see PCIe)

### 3.1.51 PD function

unit of work to be performed by a PQI device that is specified in the Administrator Queue Configuration Function register (see 5.2.5)

### 3.1.52 PQI reset (see 4.7)

PQI soft reset (see 4.7.2), PQI firm reset (see 4.7.3), or PQI hard reset (see 4.7.4)

### 3.1.53 PQI device (see 4.2.3.2)

kind of PCI function (see PCI) that is compliant with this standard

### 3.1.54 PQI device memory space

memory space (see 3.1.35) in a PQI device whose address is specified in the first memory BAR in the PQI device's configuration space

### 3.1.55 PQI domain (see 4.2.1)

one or more PQI hosts, one or more PQI devices, and the PCI Express service delivery subsystem through which they communicate

### 3.1.56 PQI host (see 4.2.2.2)

kind of PCI Express host or PCI Express device that communicates with a PQI device

### 3.1.57 PQI management application client ~~(see 9.1.3 and 9.1.4)~~(see 4.2.2.3)

object in a PQI host that ~~specifies~~requests management tasks to be performed by PQI management device servers

### 3.1.58 PQI management device server ~~(see 9.1.3 and 9.1.4)~~(see 4.2.3.4)

object in a PQI device that performs management tasks ~~specified~~requested by PQI management application clients

### 3.1.59 producer

entity that writes to ~~an~~a vacant element in a circular queue and increments the PI

### 3.1.60 producer index (PI) (see 4.3.2.1)

index of a circular queue's element array where the new entries are ~~created~~written by the producer

### 3.1.61 PCI Express Queuing Interface (PQI)

queuing layer defined by this standard

### 3.1.62 PCI memory base address register (BAR) (see PCI)

~~PCI~~ BAR (see PCI) that describes a region of memory space

### 3.1.63 read administrator function

administrator function for which a PQI management device server transfers data to a PQI management application client using a Data-In Buffer

### 3.1.64 role

when used in class diagrams and object diagrams, label at the end of an association or aggregation that defines a relationship to the class on the other side of the association or aggregation

### 3.1.65 scatter gather list (SGL) (see clause 7)

data structure used to describe a data buffer

### 3.1.66 SGL segment

standard SGL segment ~~(see 3.1.68)~~ or alternative SGL segment ~~(see 3.1.4)~~

### 3.1.67 spanned IU

series of two or more elements that contain an IU that is larger than the size of one element (see 8.1.3)

### 3.1.68 standard SGL segment (see 7.2)

data structure in a contiguous region of memory space describing all or part of a data buffer and the next ~~segment of the SGL~~SGL segment, if any

### 3.1.69 vacant element

element that does not contain valid data

### 3.1.70 virtual INTx wire (see PCIe)

virtual wire controlled by PCI Express Assert_INTx and Deassert_INTx Messages for emulation of legacy INTx interrupts

### 3.1.71 write administrator function

administrator function for which a PQI management device server transfers data from a PQI management application client using a Data-Out Buffer

## 3.2 Symbols and abbreviations

Units and abbreviations used in this standard:

| Abbreviation | Meaning |
|---|---|
| XOR | exclusive logical OR |
| ^ | exclusive logical OR |
| / | ~~division~~divide |
| ≠ or NE | not equal |
| ≤ or LE | less than or equal to |
| ± | plus or minus |
| ≈ | approximately |
| x | multiply |
| + | add |
| - | subtract |
| < or LT | less than |
| = or EQ | equal |
| > or GT | greater than |
| ≥ or GE | greater than or equal to |
| ACPI | Advanced Configuration and Power Interface (see 2.3) |
| ID | identifier |
| ARI | Alternative Routing-ID Interpretation (see PCIe) |
| B | byte |
| BAR | base address register (see PCI) |
| CI | consumer index (see 3.1.13) |
| INTx | interrupt A, B, C, or D (see PCI) |
| IQ | inbound queue (see 3.1.27) |
| IU | information unit (see 3.1.28) |

| | |
|---|---|
| LUN | Logical unit number |
| LSB | least significant bit (see 3.1.31) |
| ms | millisecond (i.e., $10^{-3}$ seconds) |
| MSB | most significant bit (see 3.1.38) |
| MSI-X | Message Signaled Interrupt Extended (see PCI) |
| ns | nanosecond (i.e., $10^{-9}$ seconds) |
| OQ | outbound queue (see 3.1.48) |
| PBA | Pending Bit Array (see 3.1.40 and PCI) |
| PCI | Peripheral Component Interconnect (see 2.3) |
| PCIe | PCI Express® Base Specification (see 2.3) |
| PD | PQI device state machine (see 4.5) |
| PI | producer index (see 3.1.60) |
| POST | power on self test |
| PQI | PCI Express Queuing Interface (i.e., this standard) |
| SAM-5 | SCSI Architecture Model-5 (see 2.2) |
| SGL | scatter gather list (see 3.1.65) |
| SOP | SCSI over PCIe® Architecture |
| SPC-4 | SCSI Primary Commands-4 (see 2.2) |

## 3.3 Keywords

### 3.3.1 invalid

a keyword used to describe an illegal or unsupported bit, byte, word, field or code value; receipt of an invalid bit, byte, word, field or code value shall be reported as an error

### 3.3.2 mandatory

a keyword indicating an item that is required to be implemented as defined in this standard

### 3.3.3 may

a keyword that indicates flexibility of choice with no implied preference (equivalent to “may or may not”)

### 3.3.4 may not

keywords that indicate flexibility of choice with no implied preference (equivalent to “may or may not”)

### 3.3.5 obsolete

a keyword indicating that an item was defined in prior standards but has been removed from this standard

### 3.3.6 optional

a keyword that describes features that are not required to be implemented by this standard; however, if any optional feature defined in this standard is implemented, then it shall be implemented as defined in this standard

### 3.3.7 reserved

a keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization; a reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard; recipients are not required to check reserved bits, bytes, words or fields for zero values; receipt of reserved code values in defined fields shall be reported as an error

### 3.3.8 RsvdC

bits or bytes that are set aside for future standardization that shall be set to zero or in accordance with a future extension to this standard, and shall be checked for zero values by the recipient

### 3.3.9 RsvdZ

bit, byte, or field in a PQI device register that are set aside for future standardization that should be set to zero by the PQI host during memory writes, shall be ignored by the PQI device during memory writes, and shall be set to zero by the PQI device during memory reads

### 3.3.10 restricted

a keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes; a restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field in the context where the restricted designation appears

### 3.3.11 shall

a keyword indicating a mandatory requirement; designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard

### 3.3.12 should

a keyword indicating flexibility of choice with a strongly preferred alternative (equivalent to “is strongly recommended”)

### 3.3.13 vendor specific

something (e.g., a bit, field, or code value) that is not defined by this standard and may be used differently in various implementations

## 3.4 Editorial conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in the glossary or in the text where they first appear.

Upper case is used when referring to the name of a numeric value defined in this specification or a formal attribute possessed by an entity. When necessary for clarity, names of objects, procedure calls, arguments or discrete states are capitalized or set in bold type. Names of fields are identified using small capital letters (e.g., NACA bit).

Names of procedure calls are identified by a name in bold type (e.g., **Execute Command**). Names of arguments are denoted by capitalizing each word in the name (e.g., Sense Data is the name of an argument in the **Execute Command** procedure call). For more information on procedure calls see 3.8.

Quantities having a defined numeric value are identified by large capital letters (e.g., CHECK CONDITION). Quantities having a discrete but unspecified value are identified using small capital letters. (e.g., TASK COMPLETE, indicates a quantity returned by the **Execute Command** procedure call). Such quantities are associated with an event or indication whose observable behavior or value is specific to a given implementation standard.

Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 - The following list shows no relationship between the named items:

a) red (i.e., one of the following colors):
   A) crimson: or
   B) amber;
b) blue; or
c) green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 -The following list shows an ordered relationship between the named items:

1) top;
2) middle; and
3) bottom.

If a conflict arises between text, tables, or figures, the order of precedence to resolve the conflicts is text; then tables; and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values.

Notes and examples do not constitute any requirements for implementors and notes are numbered consecutively throughout this standard.

## 3.5 Numeric and character conventions

### 3.5.1 Numeric conventions

A binary number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B_FD8C_FA23h).

A decimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

A range of numeric values is represented in this standard in the form “a to z”, where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation “0h to 3h” includes the values 0h, 1h, 2h, and 3h).

This standard uses the following conventions for representing decimal numbers:

a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space;
c) the thousands separator is used in both the integer portion and the fraction portion of a number; and
d) the decimal representation for a year is 1999 not 1 999.

Table 1 shows some examples of decimal numbers using various conventions.

**Table 1 — Numbering conventions**

| French | English | This standard |
|---|---|---|
| 0,6 | 0.6 | 0.6 |
| 3,141 592 65 | 3.14159265 | 3.141 592 65 |
| 1 000 | 1,000 | 1 000 |
| 1 323 462,95 | 1,323,462.95 | 1 323 462.95 |

### 3.5.2 Units of measure

This standard represents values using both decimal units of measure and binary units of measure. Values are represented by the following formats:

a) for values based on decimal units of measure:
   1) numerical value (e.g., 100);
   2) space;
   3) prefix symbol and unit:
      1) decimal prefix symbol (e.g., M) (see table 2); and
      2) unit abbreviation (e.g., B);

   and

b) for values based on binary units of measure:
   1) numerical value (e.g., 1 024);
   2) space;
   3) prefix symbol and unit:
      1) binary prefix symbol (e.g., Gi) (see table 2); and
      2) unit abbreviation (e.g., b).

Table 2 compares the prefix, symbols, and power of the binary and decimal units.

**Table 2 — Comparison of decimal prefixes and binary prefixes**

| Decimal | | | Binary | | |
|---|---|---|---|---|---|
| **Prefix name** | **Prefix symbol** | **Power (base-10)** | **Prefix name** | **Prefix symbol** | **Power (base-2)** |
| kilo | k | $10^{3}$ | kibi | Ki | $2^{10}$ |
| mega | M | $10^{6}$ | mebi | Mi | $2^{20}$ |
| giga | G | $10^{9}$ | gibi | Gi | $2^{30}$ |
| tera | T | $10^{12}$ | tebi | Ti | $2^{40}$ |
| peta | P | $10^{15}$ | pebi | Pi | $2^{50}$ |
| exa | E | $10^{18}$ | exbi | Ei | $2^{60}$ |
| zetta | Z | $10^{21}$ | zebi | Zi | $2^{70}$ |
| yotta | Y | $10^{24}$ | yobi | Yi | $2^{80}$ |

### 3.5.3 Byte encoded character strings conventions

When this standard requires one or more bytes to contain specific encoded characters, the specific characters are enclosed in single quotation marks. The single quotation marks identify the start and end of the characters that are required to be encoded but are not themselves to be encoded. The characters that are to be encoded are shown in the case that is to be encoded.

An ASCII space character (i.e., 20h) may be represented in a string by the character '¬' (e.g., 'SCSI¬device').

The encoded characters and the single quotation marks that enclose them are preceded by text that specifies the character encoding methodology and the number of characters required to be encoded.

> EXAMPLE - Using the notation described in this subclause, stating that eleven ASCII characters 'SCSI device' are to be encoded would be the same writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

### ~~3.5.4 Data field requirements~~

#### ~~3.5.4.1 ASCII data field requirements~~

~~ASCII data fields shall contain only ASCII printable characters (i.e., code values 20h to 7Eh) and may be terminated with one or more ASCII null (00h) characters.~~

~~ASCII data fields described as being left-aligned shall have any unused bytes at the end of the field (i.e., highest offset) and the unused bytes shall be filled with ASCII space characters (20h).~~

~~ASCII data fields described as being right-aligned shall have any unused bytes at the start of the field (i.e., lowest offset) and the unused bytes shall be filled with ASCII space characters (20h).~~

#### ~~3.5.4.2 Null data field termination and zero padding requirements~~

~~A data field that is described as being null-terminated shall have one byte containing an ASCII or UTF-8 null (00h) character in the last used byte (i.e., highest offset) of the field and no other bytes in the field shall contain the ASCII/UTF-8 null character.~~

~~A data field may be specified to be a fixed length. The length specified for a data field may be greater than the length required to contain the contents of the field. A data field may be specified to have a length that is a multiple of a given value (e.g., a multiple of four bytes). When such fields are described as being null-padded, the bytes at the end of the field that are not needed to contain the field data shall contain ASCII or UTF-8 null (00h) characters. When such fields are described as being zero-padded, the bytes at the end of the field that are not needed to contain the field data shall contain zeros.~~

NOTE 4 - ~~There is no difference between the pad byte contents in null-padded and zero-padded fields. The difference is in the format of the other bytes in the field.~~

~~A data field that is described as being both null-terminated and null-padded shall have at least one byte containing an ASCII or UTF-8 null (00h) character in the end of the field (i.e., highest offset) and may have more than one byte containing ASCII or UTF-8 null characters, if needed, to meet the specified field length requirements. If more than one byte in a null-terminated, null-padded field contains the ASCII or UTF-8 null character, then all the bytes containing the ASCII or UTF-8 null character shall be at the end of the field (i.e., only the highest offsets).~~

## 3.6 State machine conventions

### 3.6.1 State machine conventions overview

Figure 2 shows how state machines are described.

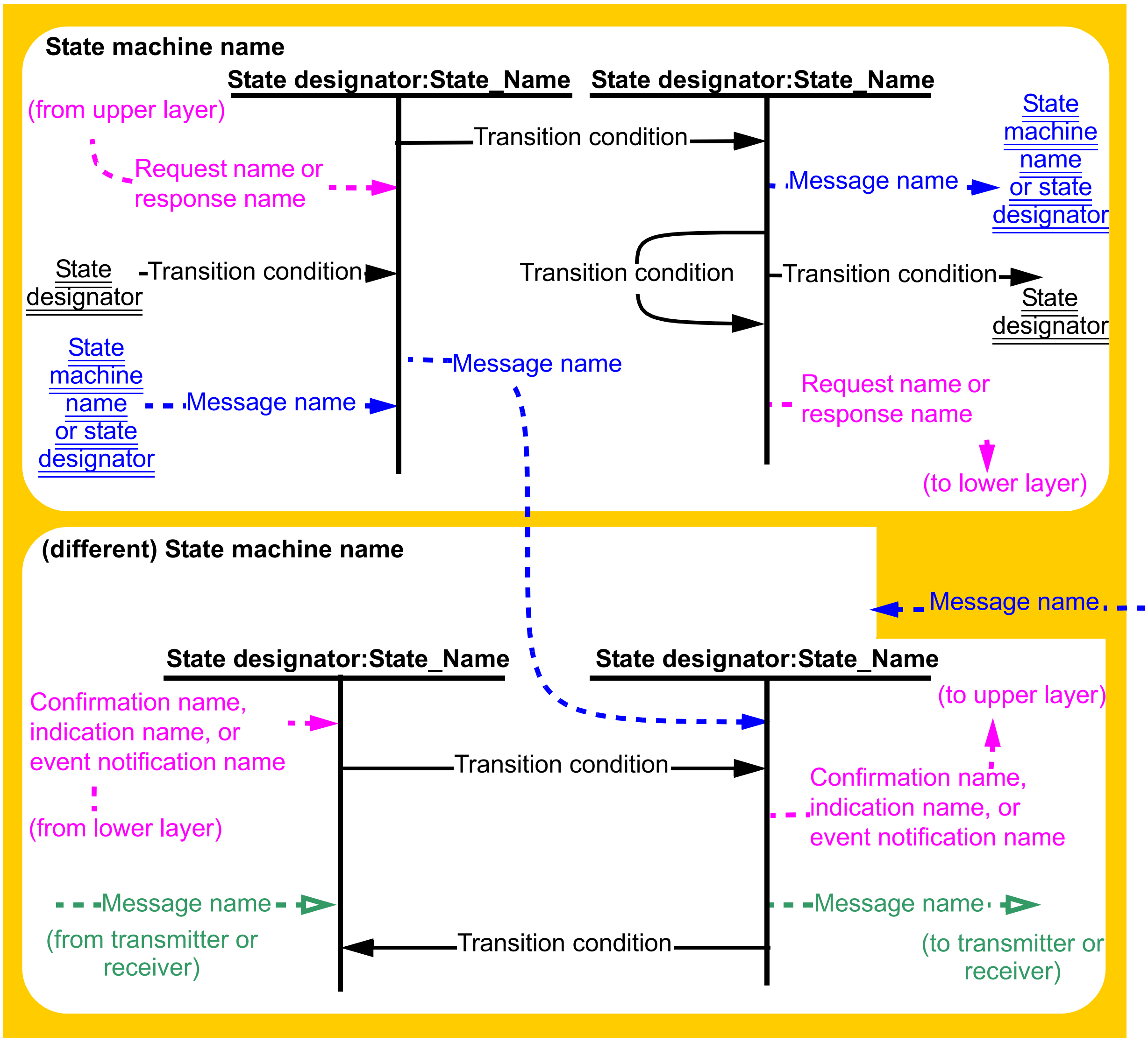

State machine name
State designator:State_Name
State designator:State_Name
(from upper layer)
Request name or response name
Transition condition
Message name
State machine name or state designator
State designator
Transition condition
Transition condition
Transition condition
State designator
State machine name or state designator
Message name
Message name
Request name or response name
(to lower layer)
(different) State machine name
Message name
State designator:State_Name
State designator:State_Name
Confirmation name, indication name, or event notification name
(from lower layer)
(to upper layer)
Transition condition
Confirmation name, indication name, or event notification name
Message name
(from transmitter or receiver)
Message name
(to transmitter or receiver)
Transition condition

**Figure 2 — State machine conventions**

When multiple state machines are present in a figure, they are enclosed in boxes with rounded corners.

Each state is identified by a state designator and a state name. The state designator (e.g., SL1) is unique among all state machines in this standard. The state name (e.g., Idle) is a brief description of the primary action taken during the state, and the same state name may be used by other state machines. Actions taken while in each state are described in the state description text.

### 3.6.2 Transitions

Transitions between states are shown with solid lines, with an arrow pointing to the destination state. A transition may be labeled with a transition condition label, a brief description of the event or condition that causes the transition to occur.

If the state transition leaves the page, the transition label goes to or from a state designator label with double underlines rather than to or from a state.

The conditions and actions are described fully in the transition description text. In case of a conflict between a figure and the text, the text shall take precedence.

Upon entry into a state, all actions to be processed in that state are processed. If a state is re-entered from itself, all actions to be processed in the state are processed again. A state may be entered and exited in zero time if the conditions for exiting the state are valid upon entry into the state. Transitions between states are instantaneous.

### 3.6.3 Messages, requests, indications, confirmations, responses, and event notifications

Messages passed between state machines are shown with dashed lines labeled with a message name. When messages are passed between state machines within the same layer of the protocol, they are identified by either:

a) a dashed line to or from a state machine name label with double underlines and/or state name label with double underlines, if the destination is in a different figure from the source;
b) a dashed line to or from a state in another state machine in the same figure; or
c) a dashed line from a state machine name label with double underlines to a “(to all states)” label, if the destination is every state in the state machine.

The meaning of each message is described in the state description text.

Requests, indications, confirmations, responses, and event notifications are shown with curved dashed lines originating from or going to the top or bottom of the figure. Each request, indication, confirmation, response, and event notification is labeled. The meaning of each request, indication, confirmation, response, and event notification is described in the state description text.

Messages with unfilled arrowheads are passed to or from the state machine’s transmitter or receiver, not shown in the state machine figures, and are directly related to data being transmitted on or received from the physical link.

### 3.6.4 State machine counters, timers, and variables

State machines may contain counters, timers, and variables that affect the operation of the state machine. The scope of state machine counters, timers, and variables is the state machine itself. They are created and deleted with the state machines with which they are associated. State machine transitions specify the initialization and modification of state machine timers, counters, and variables. Transitions out of a state may be conditioned upon specific criteria regarding the current value of a state machine counter, timer, or variable. State machine timers may continue to run while a state machine is in a given state, and a timer may cause a state transition upon reaching a defined threshold value (e.g., zero for a timer that counts down).

### 3.6.5 State machine arguments

State machines may contain one or more arguments received in a message or confirmation as state machine arguments. The following properties apply to state machine arguments:

a) the state machine that sends an argument owns that argument’s value;
b) the state machine that receives an argument shall not modify that argument’s value;
c) the state machine that sends an argument may resend that argument with a different value;
d) the scope of a state machine argument is the state machine itself; and
e) state machine argument usage is described in the state descriptions and the transition descriptions.

## 3.7 Bit and byte ordering

In this standard, data structures may be defined by a table. A table defines a complete ordering of elements (i.e., bits, bytes, fields, and dwords) within the structure. The ordering of elements within a table does not in itself constrain the order of storage or transmission of the data structure, but in combination with other normative text in this standard, may constrain the order of storage or transmission of the structure.

Tables defining data structures are shown with one row per byte and one column per bit. The lowest byte offset is at the top and the highest byte offset is at the bottom. The least significant bit (LSB) of each byte is numbered 0 and is shown on the right, and the most significant bit (MSB) of each byte is numbered 7 and shown on the left.

In a field in a table consisting of more than one bit that contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left, bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of eight or fewer bits and is contained within one row. The MSB and LSB are labeled if the field consists of more than eight bits, crosses a row, and has no internal structure defined.

In a big-endian field, the byte containing the MSB is at the lowest byte offset and the byte containing the LSB is at the highest byte offset. The bits in big-endian fields are not shaded.

In a little-endian field, the byte containing the MSB is at the highest byte offset and the byte containing the LSB is at the lowest byte offset. The bits in little-endian fields are shaded.

In a field in a table consisting of more than one byte that contains multiple fields each with their own values (e.g., a descriptor), there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels for that field. The MSB and LSB of each subfield may be shown in another table.

In a field containing a text string (e.g., ASCII or UTF-8), only the MSB of the first character and the LSB of the last character are labeled.

Multiple byte fields are represented with three rows, with the non-sequentially increasing byte numbers separated by a row labeled '...'.

Table 3 shows how this standard depicts a 32-bit big-endian field.

**Table 3 — Example of a 32-bit big-endian field**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ••• | Other field(s), if any | | | | | | | |
| n | (MSB) | Field name | | | | | | |
| ••• | | | | | | | | |
| n + 3 | | | | | | | | (LSB) |
| ••• | Other field(s), if any | | | | | | | |

Table 4 shows the bit numbers for the field shown in table 3.

**Table 4 — Bit assignments in a 32-bit big-endian field**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ••• | Other field(s), if any | | | | | | | |
| n | (MSB)<br>Bit 31 | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 |
| n + 1 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
| n + 2 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| n + 3 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0<br>(LSB) |
| ••• | Other field(s), if any | | | | | | | |

EXAMPLE 1 - If the field in table 3 and table 4 contains a value of 00010203h, then:

a) byte n contains 00h;
b) byte n+1 contains 01h;
c) byte n+2 contains 02h; and

d) byte n+3 contains 03h.

Table 5 shows how this standard depicts a 32-bit little-endian field.

**Table 5 — Example of a 32-bit little-endian field**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ••• | Other field(s), if any | | | | | | | |
| n | Field name | | | | | | | (LSB) |
| ••• | | | | | | | | |
| n + 3 | (MSB) | | | | | | | |
| ••• | Other field(s), if any | | | | | | | |

Table 6 shows the bit numbers for the field shown in table 5.

**Table 6 — Bit numbers for a 32-bit little-endian field**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ••• | Other field(s), if any | | | | | | | |
| n | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | (LSB)<br>Bit 0 |
| n + 1 | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| n + 2 | Bit 23 | Bit 22 | Bit 21 | Bit 20 | Bit 19 | Bit 18 | Bit 17 | Bit 16 |
| n + 3 | Bit 31<br>(MSB) | Bit 30 | Bit 29 | Bit 28 | Bit 27 | Bit 26 | Bit 25 | Bit 24 |
| ••• | Other field(s), if any | | | | | | | |

EXAMPLE 2 - If the field in table 5 and table 6 contains a value of 00010203h, then:

a) byte n contains 03h;
b) byte n+1 contains 02h;
c) byte n+2 contains 01h; and
d) byte n+3 contains 00h.

## 3.8 Notation for procedure calls

In this standard, the model for functional interfaces between entities is a procedure call. Such interfaces are specified using the following notation:

**[Result =] Procedure Name (IN ( [input-1] [,input-2] …]), OUT ( [output-1] [,output-2] … ))**

Where:

Result:A single value representing the outcome of the procedure call.

Procedure Name:A descriptive name for the function modeled by the procedure call.

Input-1, Input-2, …:A comma-separated list of names identifying caller-supplied input arguments.

Output-1, Output-2, …:A comma-separated list of names identifying output arguments to be returned by the procedure call.

“[…]”:Brackets enclosing optional or conditional arguments.

This notation allows arguments to be specified as inputs and outputs. The following is an example of a procedure call specification:

**Found** = **Search (**IN (Pattern, Item List), OUT ([Item Found]))

Input arguments:

**Pattern:** Argument containing the search pattern.

**Item List:** **Item<NN>** contains the items to be searched for a match.

Output arguments:

**Item Found:** Item located by the search procedure call. This argument is only returned if the search succeeds.

~~In this standard, the model for functional interfaces between entities is a procedure call. Such interfaces are specified using the following notation:~~

~~**[Result =] Procedure Name (IN ( [input-1] [,input-2] …), OUT ( [output-1] [,output-2] … ))**~~

~~Where:~~

~~Result: A single value representing the outcome of the procedure call.~~

~~Procedure Name:A descriptive name for the function modeled by the procedure call.~~

~~Input-1, Input-2, …:A comma-separated list of names identifying caller-supplied input arguments.~~

~~Output-1, Output-2, …:A comma-separated list of names identifying output arguments to be returned by the procedure call.~~

~~"[ …]": Brackets enclosing optional or conditional arguments.~~

~~This notation allows arguments to be specified as inputs and outputs. The following is an example of a procedure call specification:~~

~~Found = **Search** (IN (Pattern, Item List), OUT ([Item Found]))~~

~~Where:~~

~~**Found = Flag**~~

~~**Flag**, if set to one, indicates that a matching item was located.~~

~~Input Arguments:~~

~~**Pattern = …**~~ ~~/* Definition of **Pattern** argument */~~

~~Argument containing the search pattern.~~

~~**Item List = Item<NN>**~~ ~~/* Definition of **Item List** as an array of NN **Item** arguments*/~~

~~Contains the items to be searched for a match.~~

~~Output Arguments:~~

~~**Item Found = Item …**~~ ~~/* Item located by the search procedure call */~~

~~This argument is only returned if the search succeeds.~~

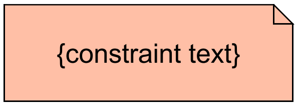
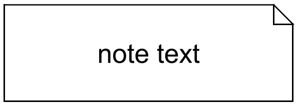
## 3.9 Notation for UML figures

### 3.9.1 Introduction

This standard contains class diagram figures that use notation that is based on UML.

Some class diagrams contain constraints or notes that use the notion shown in table 7.

**Table 7 — Class diagram constraints and notes notation**

| Notation | Description |
|---|---|
|  | The presence of the curly brackets defines constraint that is a normative requirement. An example of a constraint is shown in figure 4. |
|  | The absence of curly brackets defines a note that is informative. An example of a note is shown in figure 5. |

The notation used to denote multiplicity in class diagrams is shown in table 8.

**Table 8 — Class diagram multiplicity notation**

| Notation | Description |
|---|---|
| not specified | The number of instances of an attribute is not specified. |
| 1 | One instance of the class or attribute exists. |
| 0..* | Zero or more instances of the class or attribute exist. |
| 1..* | One or more instances of the class or attribute exist. |
| 0..1 | Zero or one instance of the class or attribute exists. |
| n..m | n to m instances of the class or attribute exist (e.g., 2..8). |
| x,n..m | Multiple disjoint instances of the class or attribute exist (e.g., 2, 8..15). |

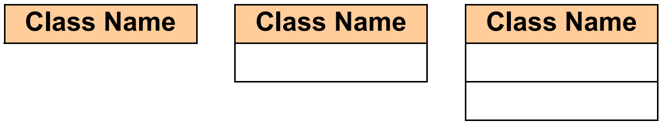
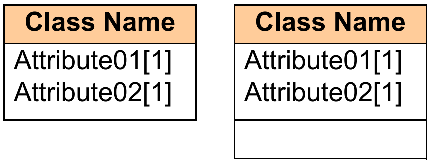
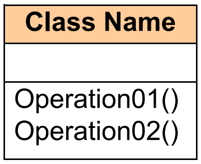
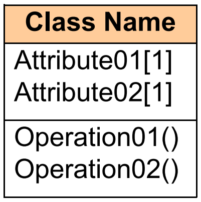
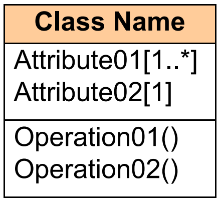
Class diagrams show:

a) two or more classes (see 3.9.2); and
b) one or more of the following relationships between them:
   A) association (see 3.9.3);
   B) aggregation (see 3.9.4);
   C) generalization (see 3.9.5); and
   D) dependency (see 3.9.6).

### 3.9.2 Class notation

The notation used for classes is shown in table 9.

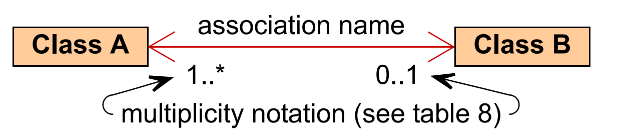
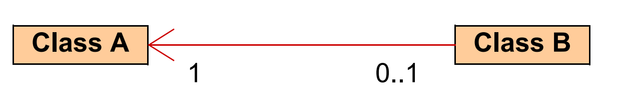
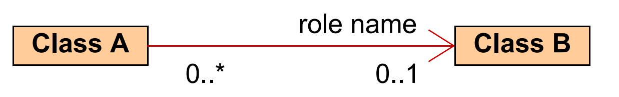
**Table 9 — Class diagram notation for classes**

| Notation | Description |
|---|---|
|  | A class with no attributes or operations |
|  Class Name<br>Attribute01[1]<br>Attribute02[1]<br>Class Name<br>Attribute01[1]<br>Attribute02[1] | A class with attributes and no operations |
|  | A class with operations and no attributes |
|  Class Name<br>Attribute01[1]<br>Attribute02[1]<br>Operation01()<br>Operation02() | A class with attributes and operations |
|  Class Name<br>Attribute01[1..*]<br>Attribute02[1]<br>Operation01()<br>Operation02() | A class with attributes that have a specified multiplicity (see table 8 in 3.9.1) and operations |

### 3.9.3 Class association relationships notation

The notation used to denote association (i.e., "knows about") relationships between classes is shown in table 10. Unless the two classes in an association relationship also have an aggregation relationship (see 3.9.4), association relationships have multiplicity notation (see table 8 in 3.9.1) at each end of the relationship line.

**Table 10 — Class diagram notation for associations**

| Notation | Description |
|---|---|
|   Class A<br>association name<br>Class B<br>1..*<br>0..1<br>multiplicity notation (see table 8) | Class A knows about Class B (i.e., read as "Class A association name Class B") and Class B knows about Class A (i.e., read as "Class B association name Class A") |
|  | Class B knows about Class A (i.e., read as "Class B knows about Class A") but Class A does not know about Class B |
|  | Class A knows about Class B (i.e., read as "Class A uses the role name attribute of Class B") but Class B does not know about Class A |
| Note: The use of role names and association names are optional. | |

Several example association relationships between classes are shown in figure 3.

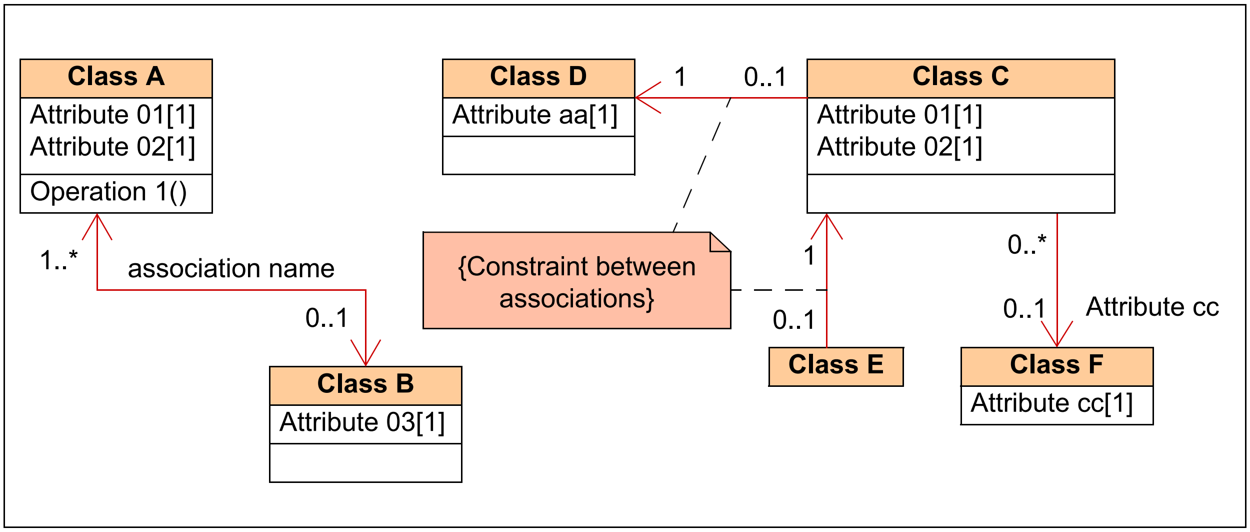
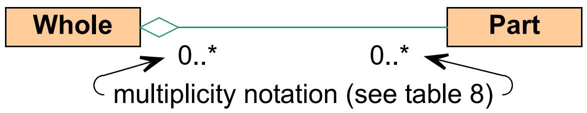
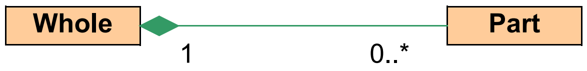


**Figure 3 — Example class association relationships**

### 3.9.4 Class aggregation relationships notation

The aggregation relationship is a specific type of association. The notation used to denote aggregation (i.e., "is a part of" or "contains") relationships between classes is shown in table 11. Aggregation relationships always include multiplicity notation (see table 8 in 3.9.1) at each end of the relationship line.

**Table 11 — Class diagram notation for aggregations**

| Notation | Description |
|---|---|
|  Whole<br>Part<br>0..*<br>0..*<br>multiplicity notation (see table 8) | The Part class is part of the Whole class and may continue to exist even if the Whole class is removed (i.e., read as "the Whole contains the Part.") |
|  | The Part class is part of the Whole class, shall only belong to one Whole class, and shall not continue to exist if the Whole class is removed (i.e., read as "the Whole contains the Part.") |

Several example aggregation relationships between classes are shown in figure 4.

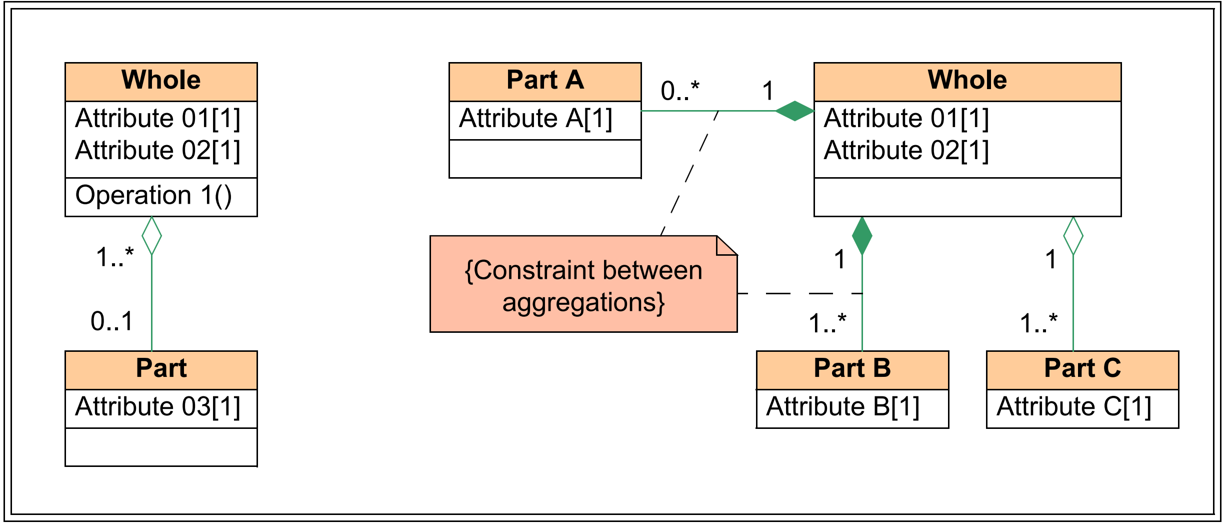
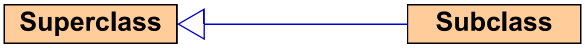


**Figure 4 — Example class aggregation relationships**

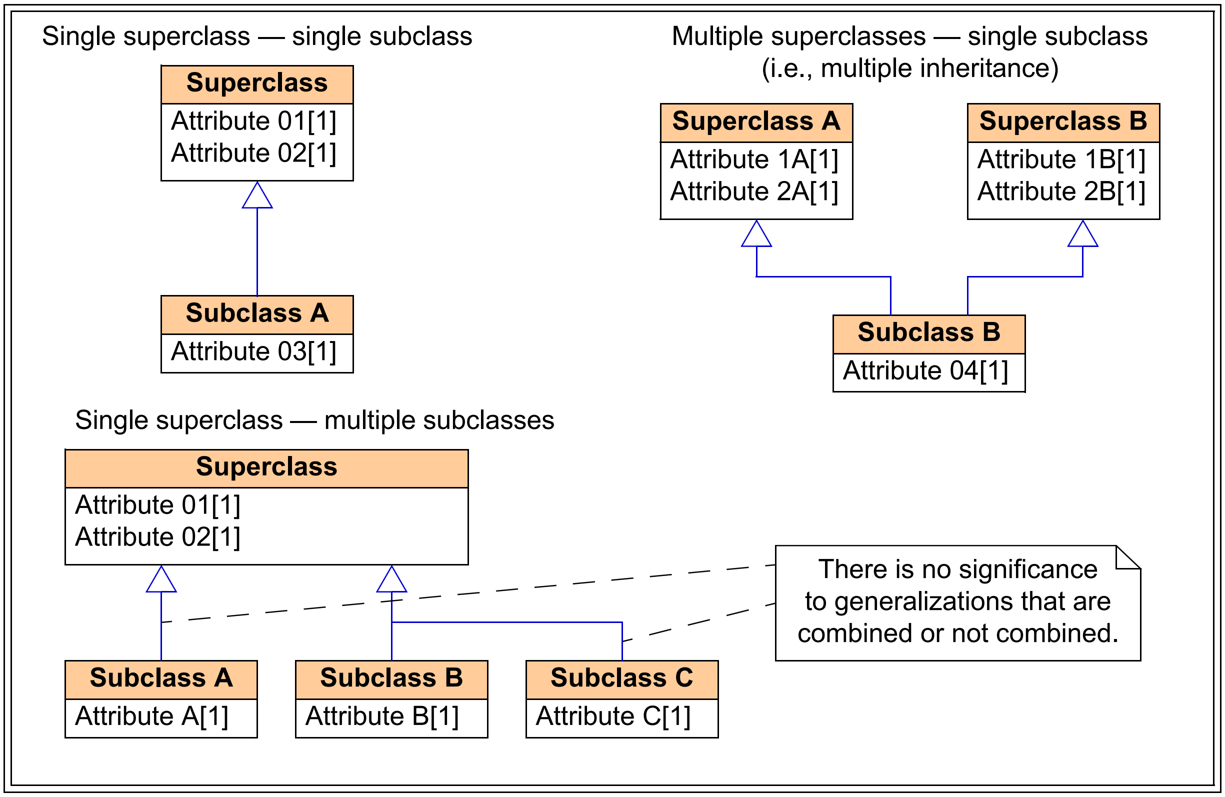
### 3.9.5 Class generalization relationships notation

The notation used to denote generalization (i.e., "is a kind of") relationships between classes is shown in table 12.

**Table 12 — Class diagram notation for generalizations**

| Notation | Description |
|---|---|
|  | Subclass is a kind of superclass. A subclass shares all the attributes and operations of the superclass (i.e., the subclass inherits from the superclass). |

Several example generalization relationships between classes are shown in figure 5.


Single superclass — single subclass

Multiple superclasses — single subclass (i.e., multiple inheritance)

Single superclass — multiple subclasses

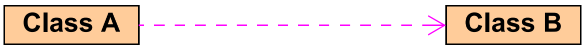
There is no significance to generalizations that are combined or not combined.

**Figure 5 — Example class generalization relationships**

### 3.9.6 Class dependency relationships notation

The notation used to denote dependency (i.e., "depends on") relationships between classes is shown in table 13.

**Table 13 — Class diagram notation for dependencies**

| Notation | Description |
|---|---|
|  | Class A depends on class B. A change in class B may cause a change in class A. |

An example dependency relationship between classes is shown in figure 6.

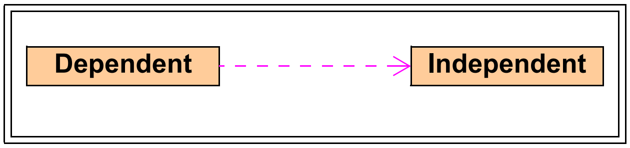


**Figure 6 — Example class dependency relationships**

### 3.9.7 Object notation

The notation used for objects is shown in table 14.

**Table 14 — Notation for objects**

| Notation | Description |
|---|---|
| label : Class Name | Notation for a named object with no attributes |
| label : Class Name<br>Attribute01 = x<br>Attribute02 = y | Notation for a named object with attributes |
| : Class Name | Notation for an anonymous object with no attributes |
| : Class Name<br>Attribute01 = x<br>Attribute02 = y | Notation for an anonymous object with attributes |

# 4 General

## 4.1 General overview

This standard defines:

a) an interface for transferring information between a PQI host and a PQI device over ~~the~~a PCI Express ~~architecture~~service delivery subsystem in a PQI domain; and
b) a scatter gather list (SGL) (see clause 7) format that is used to describe data buffers.

Certain types of information (e.g., command/functions information and response information) are packaged into information units (IUs) and transferred using circular queues. Other types of information (e.g., data) are transferred using memory transactions.

Each circular queue is unidirectional:

a) an inbound queue (IQ) is used to transfer IUs from the PQI host to the PQI device; and
b) an outbound queue (OQ) is used to transfer IUs from the PQI device to the PQI host.

Figure 7 shows a PQI host, a PQI device, a PCI Express service delivery subsystem, IQs, and OQs in a PQI domain.

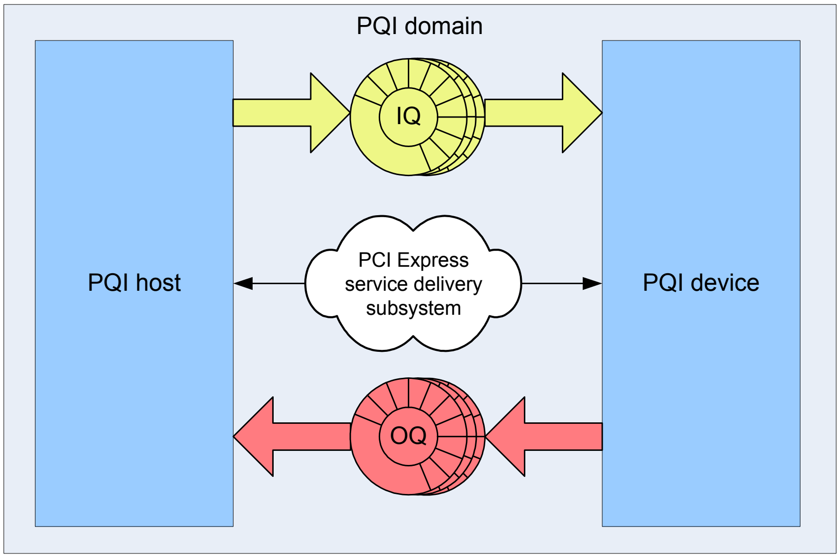


**Figure 7 — PQI device, PQI host, PCI Express service delivery subsystem, IQs and OQs**

This standard defines two types of queues:

a) administrator queues; and
b) operational queues.

Administrator queues are used for administrator IUs. Only one administrator IQ and only one administrator OQ (i.e., an administrator queue pair) are supported. Operational queues are used for operational IUs for other information unit layers (e.g., SOP). The queuing model applies to both types, although the method of creating queues of the two types differ.

An IQ identifier (IQ ID) is specified when an operational IQ is created and is used to identify the IQ. An OQ identifier (OQ ID) is specified when an operational OQ is created and is used to identify the OQ. The IQ IDs assigned to operational IQs are independent from the OQ IDs assigned to operational OQs. The administrator IQ does not have an IQ ID and the administrator OQ does not have an OQ ID.

Administrator IUs:

a) are used for managing the PQI device and are defined in this standard (see clause 9); and
b) are transferred using an administrator IQ or an administrator OQ, which is created and configured with PQI registers (see 5.2).

Operational IUs:

a) are defined in other standards (e.g., SOP); and
b) are transferred using an operational IQ or an operational OQ, which is created and configured with administrator IUs.

## 4.2 PQI classes

### 4.2.1 Classes related to the PQI domain

Figure 8 describes the classes related to the PQI domain.

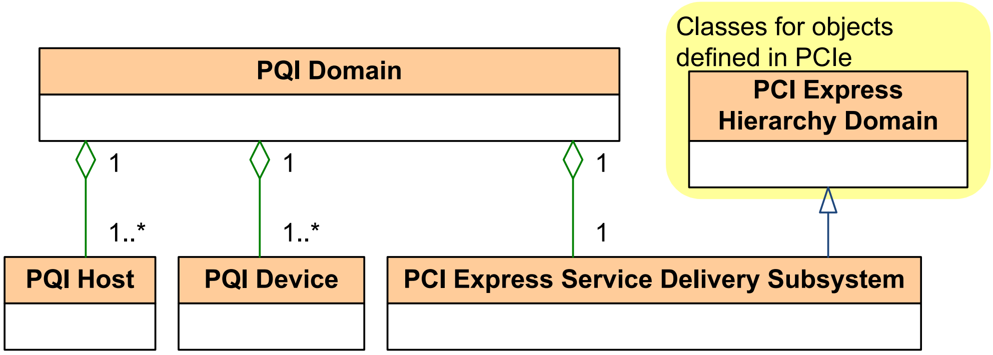

Classes for objects defined in PCIe
PQI Domain
PCI Express Hierarchy Domain
1
1
1
1..*
1..*
1
PQI Host
PQI Device
PCI Express Service Delivery Subsystem

**Figure 8 — PQI Domain class diagram**

The PQI Domain class represents the PQI domain, which contains the set of PQI host(s), PQI device(s), and the service delivery subsystem through which they communicate.

The PQI Host class is defined in 4.2.2.1.

The PQI Device class is defined in 4.2.3.1.

The PCI Express Service Delivery Subsystem class represents the PCI Express hierarchy domain (e.g., PCI Express switches and links) through which the PQI host(s) and PQI device(s) communicate.

### 4.2.2 Classes related to the PQI host

#### 4.2.2.1 Classes related to the PQI host overview

Figure 9 describes the classes related to the PQI host.

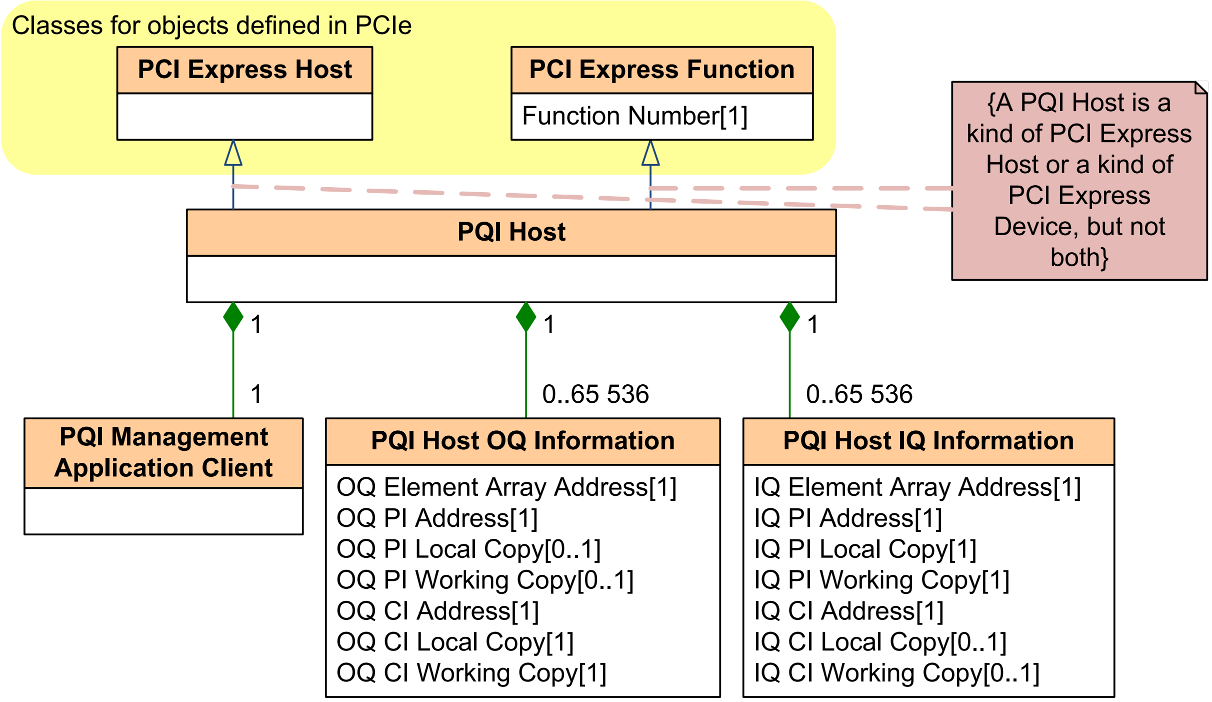


**Figure 9 — PQI Host class diagram**

#### 4.2.2.2 PQI Host class

The PQI Host class (see figure 9) represents the PQI host.

#### 4.2.2.3 PQI Management Application Client class

The PQI Management Application Client class (see figure 9) represents the PQI management application client, which requests management tasks to be performed by PQI management application clients.

NOTE 5 - The multiplicity of 65 536 represents one administrator OQ plus 65 535 operational OQs.

#### 4.2.2.4 PQI Host OQ Information class

The PQI Host OQ Information class (see figure 9) holds additional information about the OQ (see 4.3.2.3) maintained by the PQI device.

NOTE 6 - The multiplicity of 65 536 represents one administrator OQ plus 65 535 operational OQs.

The OQ Element Array Address attribute contains the OQ element array address.

The OQ PI Address attribute contains the OQ PI address.

The OQ CI Address attribute contains the OQ CI address.

The OQ PI Local Copy attribute contains the OQ PI local copy.

The OQ PI Working Copy attribute contains the OQ PI working copy.

The OQ CI Local Copy attribute contains the OQ CI local copy.

The OQ CI Working Copy attribute contains the OQ CI working copy.

#### 4.2.2.5 PQI Host IQ Information class

The PQI Host IQ Information class (see figure 9) holds additional information about the IQ (see 4.3.2.2) maintained by the PQI device.

The IQ Element Array Address attribute contains the IQ element array address.

The IQ PI Address attribute contains the IQ PI address.

The IQ PI Local Copy attribute contains the IQ PI local copy.

The IQ PI Working Copy attribute contains the IQ PI working copy.

The IQ CI Address attribute contains the IQ CI address.

The IQ CI Local Copy attribute contains the IQ CI local copy.

The IQ CI Working Copy attribute contains the IQ CI working copy.

### 4.2.3 Classes related to the PQI device

#### 4.2.3.1 Classes related to the PQI device overview

Figure 10 describes the classes related to the PQI device.

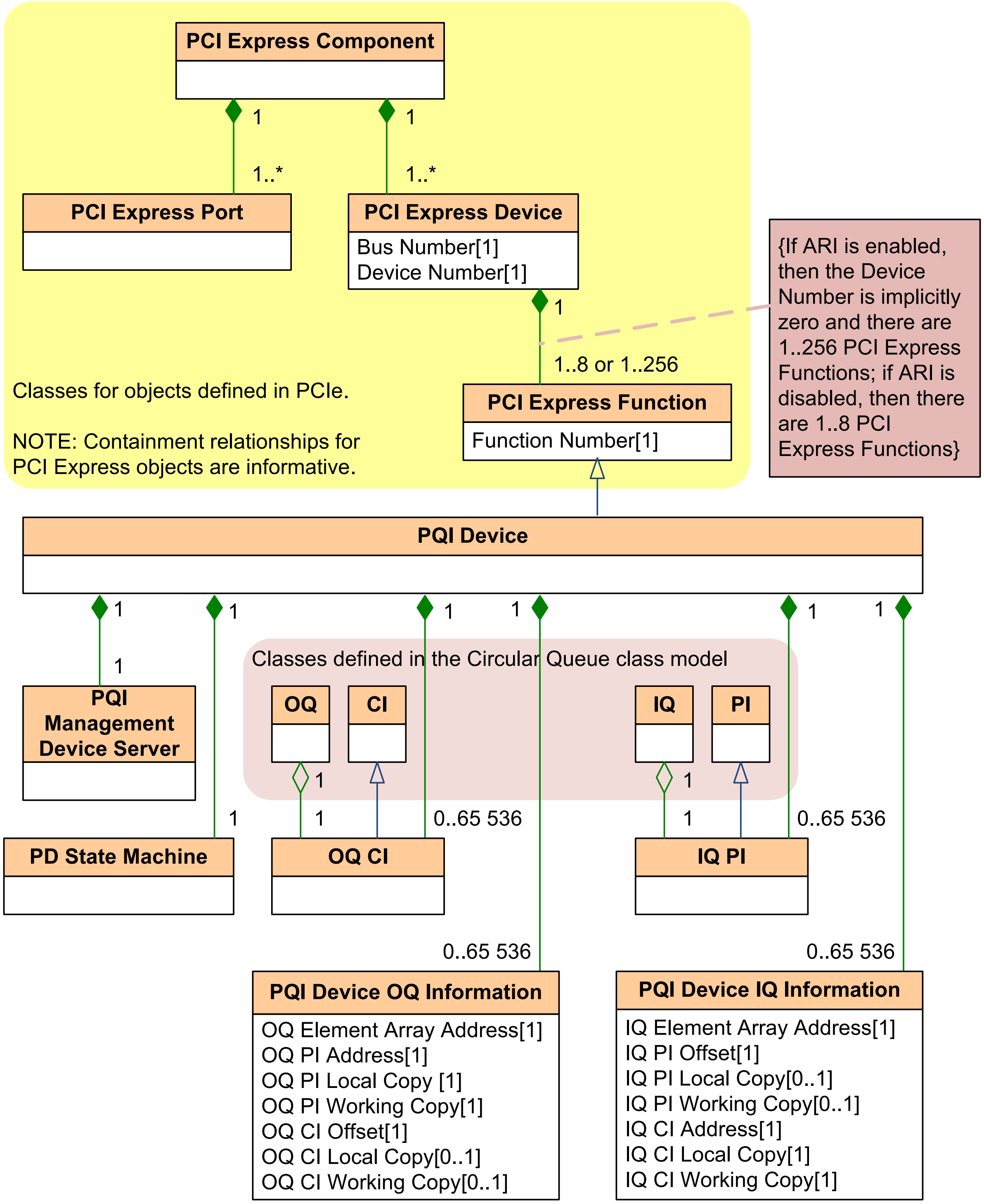


**Figure 10 — PQI Device class diagram**

#### 4.2.3.2 PQI Device class

The PQI Device class (see figure 10) represents the PQI device.

#### 4.2.3.3 PD State Machine class

The PD State Machine class (see figure 10) represents the PD state machine (see 4.5).

#### 4.2.3.4 PQI Management Device Server class

The PQI Management Device Server class (see figure 10) represents the PQI management device server, which performs management tasks requested by PQI management application clients.

#### 4.2.3.5 OQ CI class

The OQ CI class (see figure 10) represents the OQ CI (see 4.3.2.3).

NOTE 7 - The multiplicity of 65 536 represents one administrator OQ plus 65 535 operational OQs.

#### 4.2.3.6 IQ PI class

The IQ PI class (see figure 10) represents the IQ PI (see 4.3.2.2).

NOTE 8 - The multiplicity of 65 536 represents one administrator OQ plus 65 535 operational OQs.

#### 4.2.3.7 PQI Device OQ Information class

The PQI Device OQ Information class (see figure 10) holds additional information about the OQ (see 4.3.2.3) maintained by the PQI device.

NOTE 9 - The multiplicity of 65 536 represents one administrator OQ plus 65 535 operational OQs.

The OQ Element Array Address attribute contains the OQ element array address.

The OQ PI Address attribute contains the OQ PI address.

The OQ PI Local Copy attribute contains the OQ PI local copy.

The OQ PI Working Copy attribute contains the OQ PI working copy.

The OQ CI Offset attribute contains the OQ CI offset in PQI device memory space.

The OQ CI Local Copy attribute contains the OQ CI local copy.

The OQ CI Working Copy attribute contains the OQ CI working copy.

#### 4.2.3.8 PQI Device IQ Information class

The PQI Device IQ Information class (see figure 10) holds additional information about the IQ (see 4.3.2.2) maintained by the PQI device.

NOTE 10 - The multiplicity of 65 536 represents one administrator OQ plus 65 535 operational OQs.

The IQ Element Array Address attribute contains the IQ element array address.

The IQ CI Address attribute contains the IQ CI address.

The IQ PI Offset attribute contains the IQ PI offset in PQI device memory space.

The IQ CI Local Copy attribute contains the IQ CI local copy.

The IQ CI Working Copy attribute contains the IQ CI working copy.

The IQ PI Local Copy attribute contains the IQ PI local copy.

The IQ PI Working Copy attribute contains the IQ PI working copy.

## 4.3 Queuing model

### 4.3.1 Queuing model overview

The queuing model defines:

a) circular queue configuration and operations (see 4.3.2.1);
b) circular queue producer indexes (PIs) (see 4.3.2.1);
c) circular queue consumer indexes (CIs) (see 4.3.2.1);
d) circular queue elements (see 4.3.2.1); and
e) IQ priority (see 4.3.5).

### 4.3.2 Circular queue

#### 4.3.2.1 Circular queue overview

A circular queue (see figure 11) consists of the following:

a) element array;
b) CI; and
c) PI.

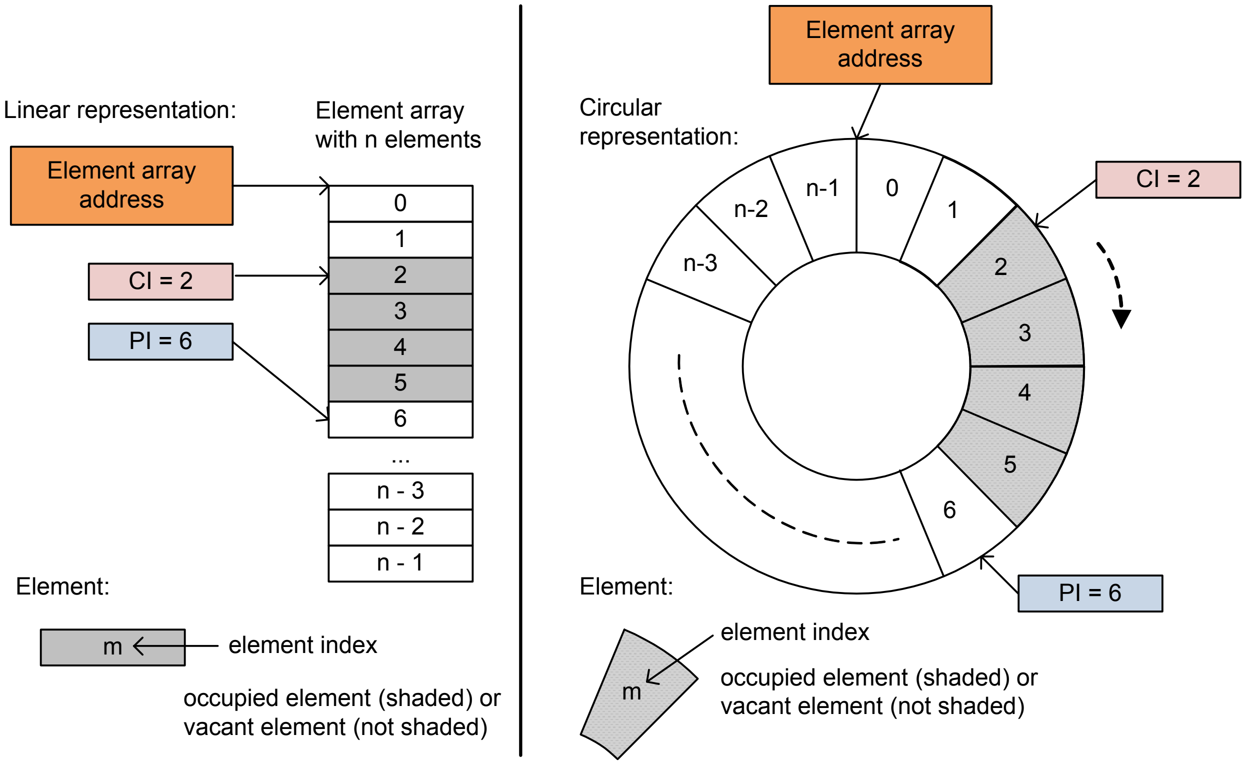


**Figure 11 — Circular queue**

The element array contains a number of elements of fixed size. The size and the number of elements in an element array are configured when the circular queue is created. For an element array of size n, the elements are indexed from zero to (n-1).

The element array address is the address of the start of the first element in the element array. The address of an element is determined by multiplying an index value by the ~~size of an~~ element size and adding that value to the element array address.

~~Elements are written to the element array by a producer and read from the element array by a consumer.~~

A producer writes to vacant elements in an element array and a consumer reads from occupied elements in an element array.

The PI contains the index of the next element to be written by the producer. The initial value of the PI is zero.

~~The~~If the CI does not equal the PI, then the CI contains the index of the next occupied element to be read by the consumer. If the CI equals the PI, then the CI is invalid. The initial value of the CI is zero.

~~The PI contains the index of the next element to be written by the producer. The initial value of the PI is zero.~~

An element within a circular queue is ~~a valid~~ an occupied element if:

a) the CI does not equal the PI; and
b) the index of the element:
   A) is equal to the index contained in the CI;
   B) is greater than the CI and less than the PI and the CI is less than the PI;
   C) is greater than the CI and the CI is greater than the PI; or
   D) is less than the PI and the CI is greater than the PI.

All other elements within the circular queue are ~~invalid~~vacant elements.

The circular queue is empty ~~when~~while the CI and PI are equal (i.e., all elements are vacant).

Figure 12 shows an example of a full circular queue and an empty circular queue. The circular queue is full when the PI is one behind the CI (e.g., PI=4 and CI=5, or an example ~~for~~of an element array with 64 elements, PI=63 and CI=0). There is always at least one ~~invalid~~vacant element in the element array. The maximum number of ~~valid~~occupied elements in an element array of size n is n-1.

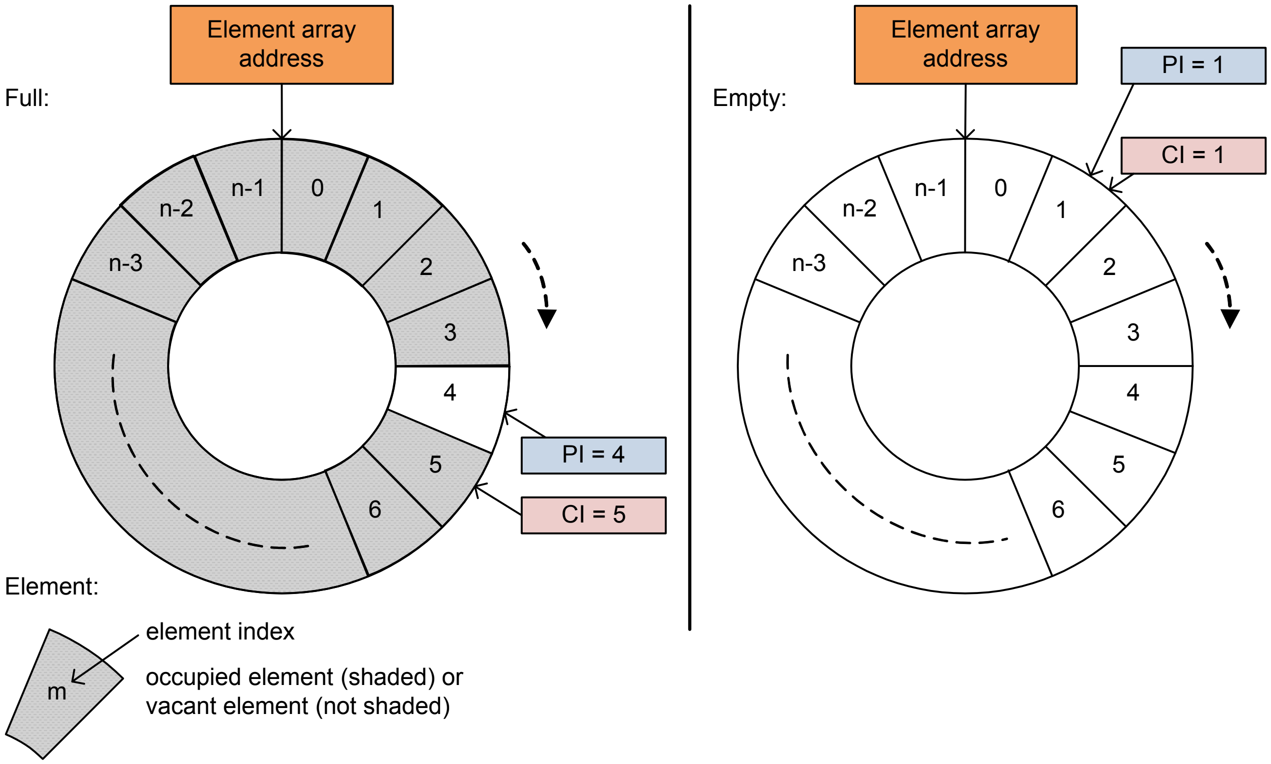


**Figure 12 — Example of a full circular queue and an empty circular queue**

The producer writes one or more IUs to vacant elements starting at the element indicated by the PI and continuing sequentially up to and including the element indicated by (CI - ~~1~~2), wrapping at the end of the element array if necessary. After writing one or more IUs to the element array, the producer increments PI by the number of occupied elements produced, modulo the number of elements in the array.

While writing to vacant elements ~~to~~in a circular queue, the producer ~~shall maintain~~maintains two copies of the PI:

a) ~~local copy of the PI~~a PI local copy (i.e., a mirror of the value last written to the PI); and
b) ~~working copy of the PI~~a PI working copy ~~(i.e., representing the elements written to the element array)~~(i.e., containing the index of the next vacant element in the element array).

~~While writing one or more elements to the circular queue, the producer compares the working copy of the PI to the CI. The producer:~~

a) ~~may write an updated value to the PI after producing each element; and~~
b) ~~should write an updated value to the PI after producing an element if the local copy of the PI is approaching the CI (i.e., the circular queue is almost empty).~~

NOTE 11 - ~~Regular PI updates keep the consumer from detecting that the circular queue is empty and allow it to continue reading from the circular queue. The algorithm for determining the threshold is vendor specific.~~

~~A memory write transaction used to send an update to the PI should use the same Traffic Class (see PCIe) as the memory write transaction used to write the elements of the queue and shall not be sent until after the updates to the elements have been completed.~~

~~The consumer reads elements from element indexes starting with CI and continuing sequentially through (PI - 1), wrapping at the end of the element array. After reading one or more elements from the element array, the consumer increments CI by the number of elements consumed, wrapping at the element array size.~~

While writing to one or more vacant elements in a circular queue, the producer compares the PI working copy to the CI. The producer:

a) may update the PI and the PI local copy to the value of the PI working copy after producing each occupied element; and
b) should update the PI and the PI local copy to the value of the PI working copy if the CI is approaching or equal to the PI local copy (i.e., the consumer considers the circular queue to be almost empty or to be empty).

NOTE 12 - Since the consumer is consuming occupied elements from the circular queue at the same time as the producer is producing occupied elements to the circular queue, regular PI updates keep the consumer from detecting that the circular queue is empty and allow it to continue consuming from the circular queue. The algorithm for determining the threshold is vendor specific.

NOTE 13 - There is a benefit in reducing the number of memory write transactions by combining IUs and updating the IQ PI with a single memory write transaction.

A producer should use the same Traffic Class for memory write transactions to the PI as it uses for memory write transactions to the element array.

To read IUs, the consumer reads occupied elements starting from the element indicated by the CI and continues sequentially wrapping at the end of the element array. After reading from one or more occupied elements in the element array, the consumer increments CI by the number of occupied elements consumed, wrapping at the element array size.

While reading from elements ~~from~~in a circular queue, the consumer ~~shall maintain~~maintains two copies of the CI:

a) ~~local copy of the CI~~a CI local copy (i.e., a mirror of the value last written to the CI); and
b) ~~working copy of the CI~~a CI working copy (i.e., representing the elements read from the element array).

~~While reading one or more elements from the circular queue, the consumer compares the working copy of the CI to the PI. The consumer:~~

a) ~~may write an updated value to the CI after consuming each element; and~~
b) ~~should write an updated value to the CI after consuming an element if the working copy of the CI is approaching the PI (i.e., the circular queue is almost full).~~

While reading from one or more occupied elements in the circular queue, the consumer compares the CI working copy to the PI. The consumer:

a) may update the CI and the CI local copy to the value of the CI working copy after consuming each occupied element; and
b) should update CI and the CI local copy to the value of the CI working copy after consuming an occupied element if the PI is approaching the CI local copy (i.e., the producer considers the circular queue to be almost full or to be full).

NOTE 14 - Since the producer is producing occupied elements to the circular queue at the same time as the consumer is consuming occupied elements from the circular queue, regular CI updates keep the producer from detecting that the circular queue is full and allow it to continue producing to the circular queue. The algorithm for determining the threshold is vendor-specific.

~~A memory write transaction used to send an update to the CI should use the same Traffic Class as the memory read transactions used to read the elements of the circular queue being released~~

A consumer should use the same Traffic Class (see PCIe) for memory write transactions to the CI as it uses for memory read transactions from the element array.

~~A memory read transaction should not be sent to read invalid elements~~The consumer should not perform memory read transactions of a vacant element.

#### 4.3.2.2 IQ

An IQ is a circular queue used to transfer IUs from a PQI host to a PQI device.

For an IQ:

a) the IQ element array starts in the memory space at the IQ element array address (i.e., in PQI ~~device~~host memory space and not in PQI device memory space);
b) the IQ CI dword (see 6.1.1) specifies an IQ CI contained in the memory space that starts at the IQ CI address; and
c) the IQ PI dword (see 6.1.2) specifies an IQ PI contained in the memory space that starts at the IQ PI address.

The PQI host maintains ~~a local copy of the~~ an IQ PI local copy.

The PQI device maintains ~~a local copy of the~~ an IQ CI local copy.

The PQI device:

a) shall not use PCI memory transactions to access the IQ PI;
b) does not use PCI memory transactions to access an IQ element array or an IQ CI that is located in PQI device memory space; or
c) does not use PCI memory transactions to access the ~~memory space provided by the PQI device~~PQI device memory space.

Figure 13 shows an IQ example where:

a) ~~an IQ where~~ the IQ element array is in the memory space of one PCI function;~~,~~
b) the IQ CI is in the memory space of another PCI function;~~,~~
c) ~~and~~ the IQ PI ~~are~~ is in the PQI device memory space~~s of different PCI functions(see PCI)~~; and
d) the PQI host is separated from all of them.

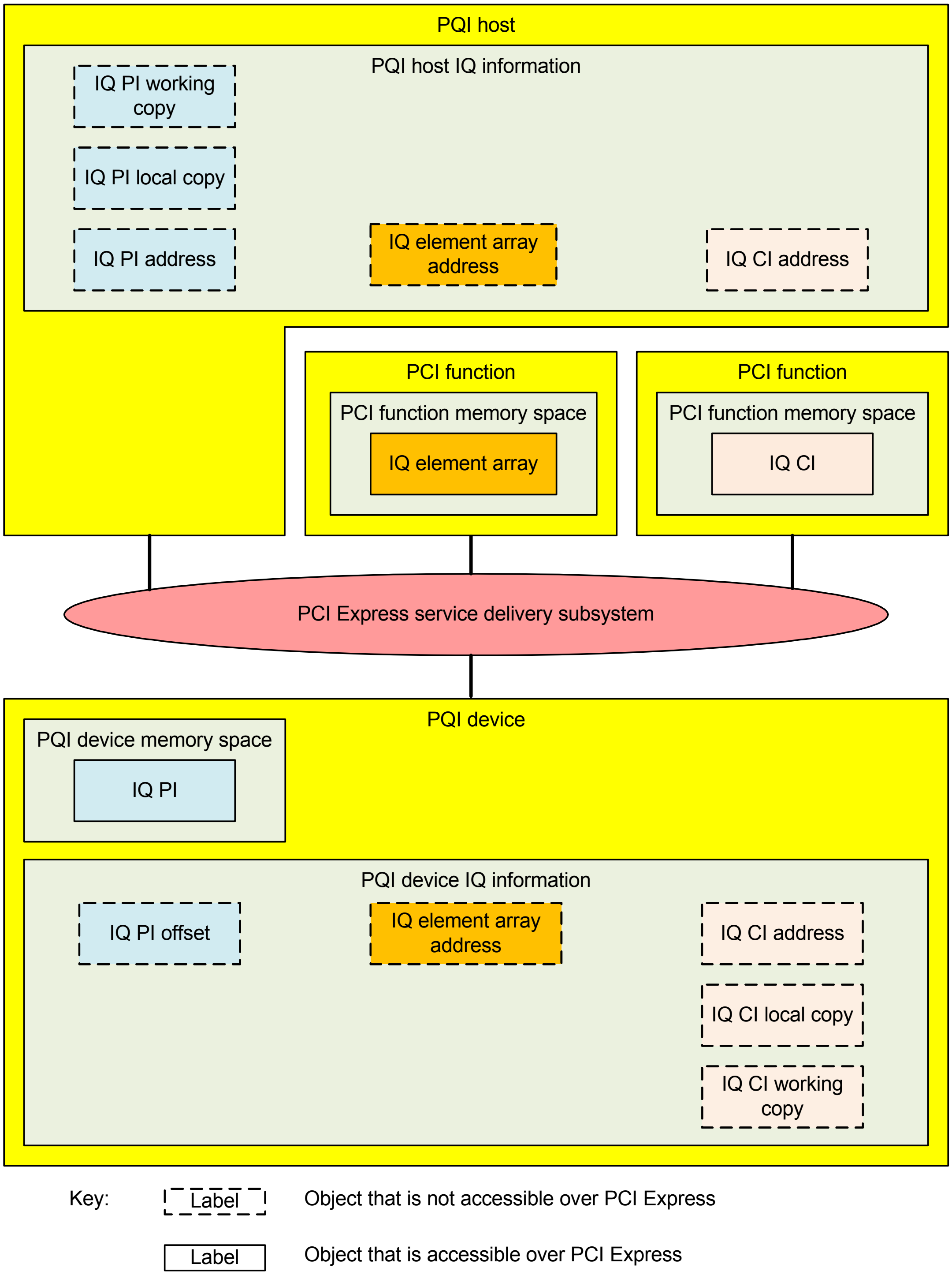


**Figure 13 — Example where IQ object locations are separated**

~~Figure 14 shows an IQ where the IQ element array and IQ CI are in the PQI host memory space.~~

Figure 14 shows an IQ example where:

a) the IQ element array is in the PQI host memory space;
b) the IQ CI is in the PQI host memory space; and
c) the IQ PI is in the PQI device memory space.

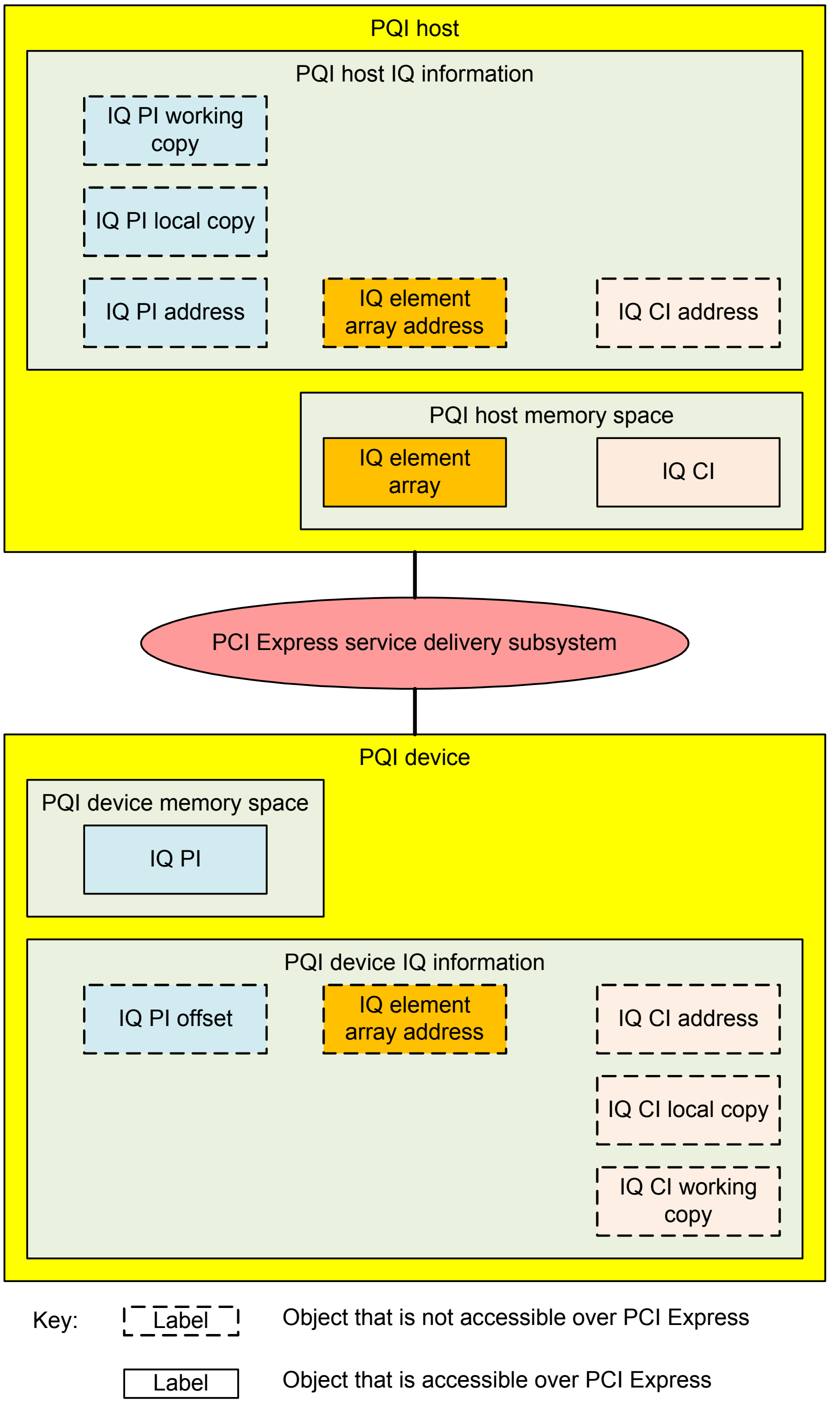


**Figure 14 — Example of IQ object locations - typical**

A PQI host shall not use the Relaxed Ordering ~~(see PCIe)~~ ordering type (see PCIe) in memory transactions accessing the IQ element array, IQ CI, or IQ PI.

A PQI device shall not use the Relaxed Ordering ordering type in memory transactions accessing the IQ element array or IQ CI.

~~A PQI device shall not use the ID-Based Ordering ordering type in memory transactions accessing the IQ element array or IQ CI unless the IQ element array and IQ CI are both located in PQI host memory space.~~

A PQI device shall use the ID-Based Ordering ordering type (see PCIe) in memory transactions accessing the IQ element array and IQ CI unless the IQ element array and IQ CI are not both located in PQI host memory space. The mechanism for determining the location of the IQ element array and IQ CI is outside the scope of this standard.

### 4.3.2.3 OQ

An OQ is a circular queue used to transfer IUs from a PQI device to a PQI host.

For an OQ:

a) the OQ element array starts in the memory space at the OQ element array address (i.e., in PQI ~~device~~host memory space and not in PQI device memory space);
b) the OQ CI dword (see 6.1.3) specifies an OQ CI contained in the memory space that starts at the OQ CI address; and
c) the OQ PI dword (see 6.1.4) specifies an OQ PI contained in the memory space that starts at the OQ PI address.

The PQI host maintains ~~a local copy of the~~ an OQ CI local copy.

The PQI device maintains ~~a local copy of the~~ an OQ PI local copy.

The PQI device:

a) shall not use memory transactions to access the OQ CI;
b) does not use memory transactions to access an OQ element array or an OQ PI that is located in PQI device memory space; or
c) does not use memory transactions to access the ~~memory space provided by the PQI device~~PQI device memory space.

~~Figure 15 shows:~~

a) ~~an OQ where the OQ element array, OQ PI, and OQ CI are in the memory spaces of different PCI functions (see PCI); and~~
b) ~~the PQI host is separate from all of them.~~

Figure 15 shows an OQ example where:

a) the OQ element array is in the memory space of one PCI function;
b) the OQ PI is in the memory space of another PCI function;
c) the OQ CI is in the PQI device memory space; and
d) the PQI host is separated from all of them.

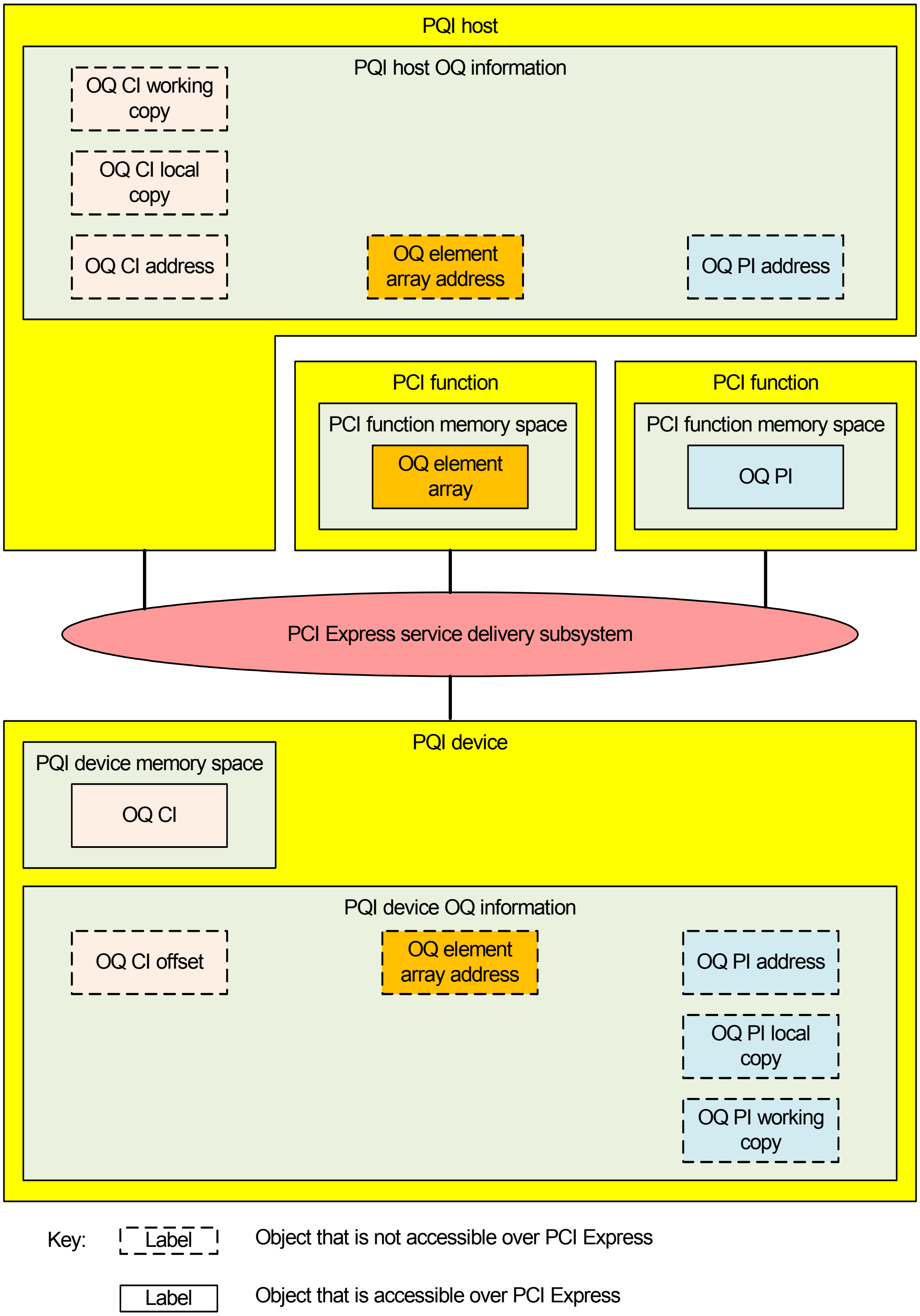


**Figure 15 — Example of OQ object locations are separated**

~~Figure 16 shows an OQ where the OQ element array and OQ PI are in the PQI host memory space.~~

Figure 16 shows an OQ example where:

a) the OQ element array is in the PQI host memory space;
b) the OQ PI is in the PQI host memory space; and
c) the OQ CI is in the PQI device memory space.

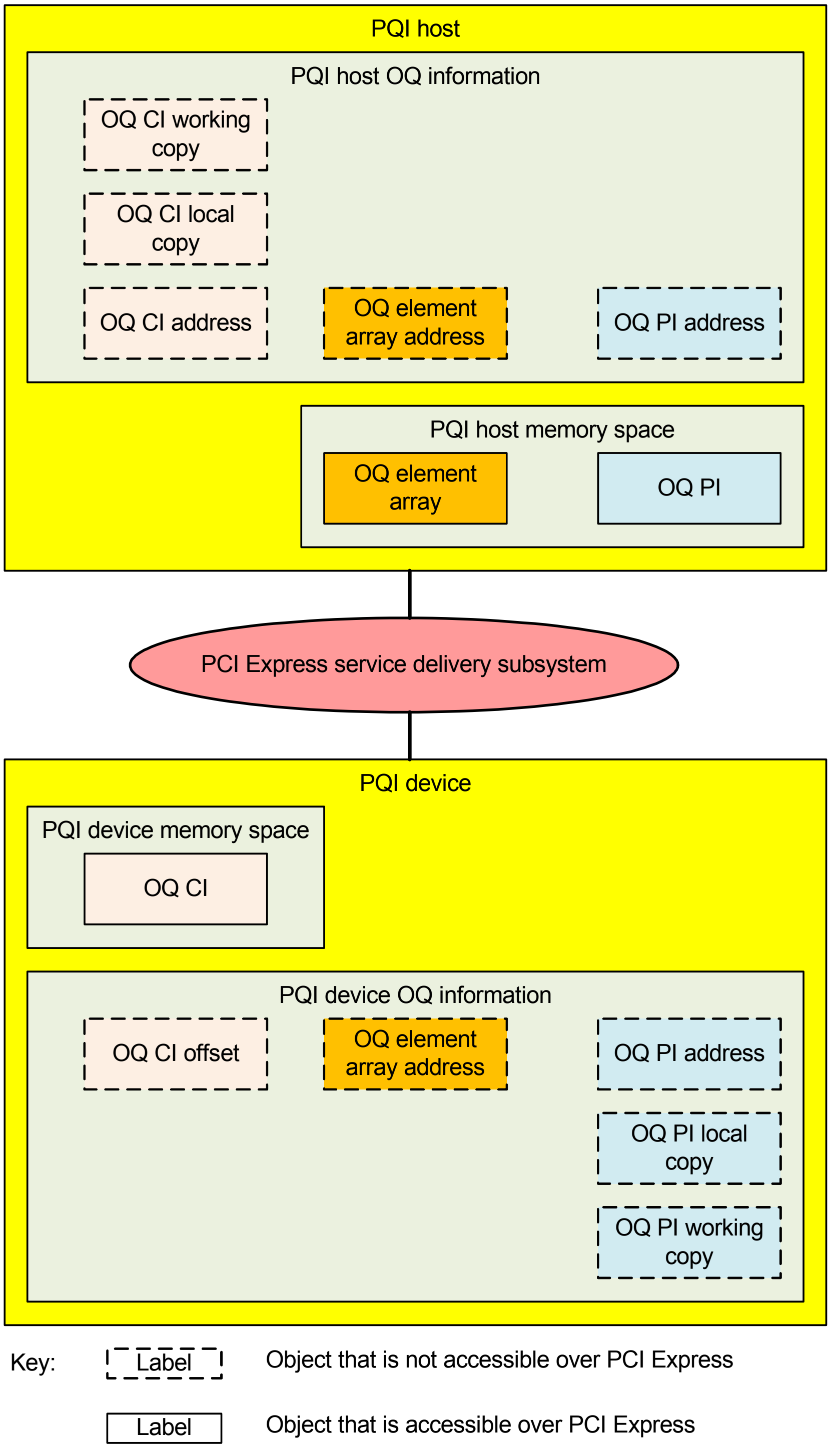


**Figure 16 — Example of OQ object locations - typical**

A PQI host should not use the Relaxed Ordering ~~(see PCIe)~~ ordering type (see PCIe) in memory transactions accessing the OQ element array, OQ CI, or OQ PI.

A PQI device shall not use the Relaxed Ordering ordering type in memory transactions accessing the OQ element array or OQ PI.

~~A PQI device shall not use the ID-Based Ordering (see PCIe) ordering type in memory transactions accessing the OQ element array or OQ PI unless the OQ element array and OQ PI are both in PQI host memory space.~~

A PQI device shall use the ID-Based Ordering ordering type (see PCIe) in memory transactions accessing the OQ element array and OQ PI unless the OIQ element array and OQ PI are not both located in PQI host memory space. The mechanism for determining the location of the OQ element array and OQ PI is outside the scope of this standard.

### 4.3.2.4 Enqueuing elements and dequeuing elements

#### 4.3.2.4.1 PQI host enqueuing IU(s) to an IQ

The PQI host determines the number of ~~available elements of an IQ~~vacant IQ elements available to hold new IUs by comparing the IQ CI and the ~~PQI host local copy~~ of IQ PI local copy. Figure 17 shows the typical location of the IQ CI and the ~~PQI host local copy~~ of IQ PI local copy used by the PQI host to determine the number of ~~available~~vacant IQ elements.

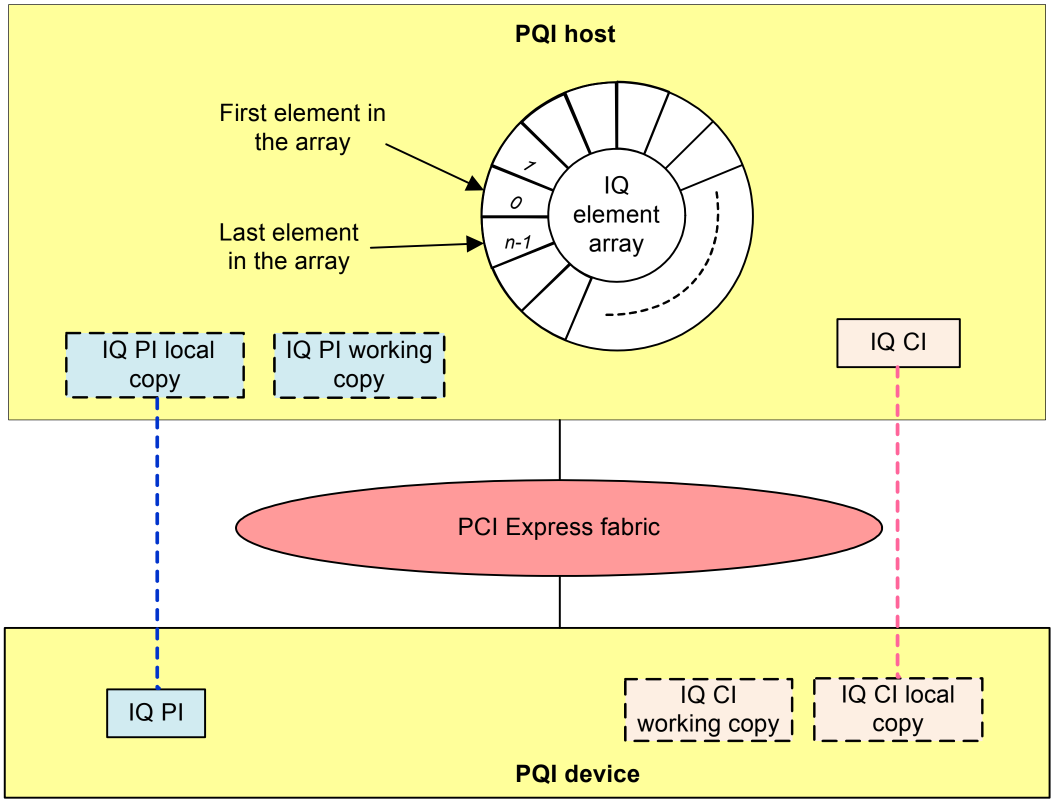


**Figure 17 — Typical location of IQ CI and PQI host IQ PI local copy**

NOTE 15 - There is a benefit in reducing the number of memory write transactions by combining IUs and updating the IQ PI with a single memory write transaction. This approach is optional~~ and a PQI host may enqueue one IU at a time~~.

The PQI host may enqueue one or more IUs to the IQ if the total number of elements required by the IUs to be enqueued is less than~~ or equal to~~ the number of ~~available~~vacant IQ elements. The PQI host should use the following steps to enqueue IUs to the IQ:

1) for each IU to be enqueued:
    1) write the IU into the IQ element array beginning at the address indicated by the ~~working copy of~~ IQ PI working copy, using wrapping ~~rules~~as described in 4.3.2.1; and
    2) update the ~~working copy of~~ IQ PI working copy;
2) update the ~~local copy of~~ IQ PI local copy with the ~~working copy of~~ IQ PI working copy; and
3) update the IQ PI with the ~~working copy of~~ IQ PI working copy.

If the PQI device supports IUs spanning multiple elements (see 8.1.3) in an IQ, then the IU LENGTH field (see 8.2) value plus four may be larger than the IQ element length (see 4.3.6.2.3). If the PQI device does not support IUs spanning multiple elements in an IQ, then the number of bytes in an IU indicated by the IU LENGTH field plus four shall be less than or equal to the IQ element length.

The PQI device handles errors in the IU LENGTH field when dequeuing IUs as described in 4.3.2.4.3.

#### 4.3.2.4.2 PQI host dequeuing IU(s) from an OQ

The PQI host determines the number of ~~available~~occupied ~~valid~~ OQ elements by comparing the OQ PI and the ~~PQI host local copy of~~ OQ CI local copy. ~~The PQI host may dequeue IUs from OQ following an interrupt from the PQI device.~~

Figure 18 shows the typical location of the OQ PI and the ~~PQI host~~ ~~local copy of~~ OQ CI local copy used by the PQI host to determine the number of ~~available~~occupied OQ elements.

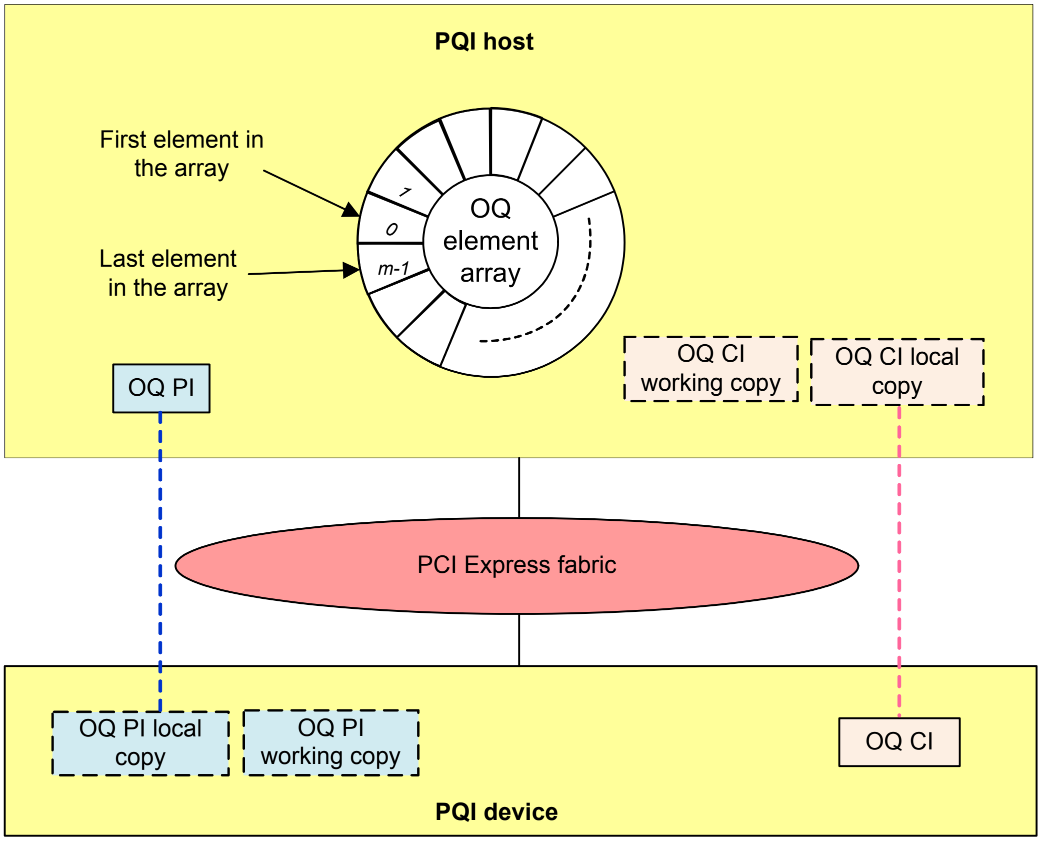


**Figure 18 — Typical location of OQ PI and PQI host OQ CI local copy**

The PQI host may dequeue one or more IUs from the OQ using the following steps:

1) for each IU to be dequeued:
    1) read the IU from the OQ element array beginning at the address indicated by the ~~working copy of~~ OQ CI working copy, using wrapping ~~rules~~as described in 4.3.2.1; and
    2) update the ~~working copy of~~ OQ CI working copy;
2) update the ~~local copy of~~ OQ CI local copy with the ~~working copy of~~ OQ CIworking copy; and
3) update the OQ CI with the ~~working copy of~~ OQ CI working copy.

If the PQI device supports IUs spanning multiple elements in an OQ, then the IU LENGTH field (see 8.2) value plus four may be larger than the OQ element length. If the PQI device does not support IUs spanning multiple elements in an OQ, then the number of bytes in an IU indicated by the IU LENGTH field plus four shall be less than or equal to the OQ element length (i.e., excludes the count of the IU header fields (see 8.2)).

During the processing of an IU, if the IU LENGTH field is invalid (see 8.2), then the PQI host processes the error using a method that is outside the scope of this standard.

#### 4.3.2.4.3 PQI device dequeuing IU(s) from an IQ

The PQI device determines the number of ~~available~~occupied elements of an IQ by comparing the IQ PI and the ~~local copy of~~ IQ CI local copy. The PQI device checks the validity of IUs and the ~~IU~~ IU LENGTH field (see 8.2) after the IQ elements are ~~read~~moved from the IQ into the PQI device internal resource.

Prior to dequeuing any IQ elements, the PQI device determines the number of elements to be dequeued by comparing the number of occupied elements ~~and~~with the number of available PQI internal resources to hold the elements for processing. Available PQI internal resources that receive the inbound IUs may ~~determine~~limit the maximum number of inbound IUs that are read from the IQ. The implementation of the PQI device internal resource to receive the inbound IUs is not defined by this standard.

The PQI device processes the inbound IU by parsing the content ~~of~~from the occupied IQ element(s) that ~~were already~~have been moved to the PQI device internal resource. For the administrator IU, if any IU parameters (i.e., including the IU LENGTH field (see 8.2)) are invalid, the PQI device shall process the error as defined in 8.2. For other information unit layer (e.g., SOP), ~~If~~if any IU parameters ~~plus~~(including the IU LENGTH field ~~(see 8.2)~~ ~~is~~are invalid, then the PQI device reports ~~the~~an error to the PQI host using a method defined in ~~other~~the information unit layer standard~~s~~ (e.g., SOP).

Figure 19 shows the typical location of the IQ PI and the PQI device ~~local copy of~~ IQ CI local copy used by the PQI device to determine the number of available IQ elements.

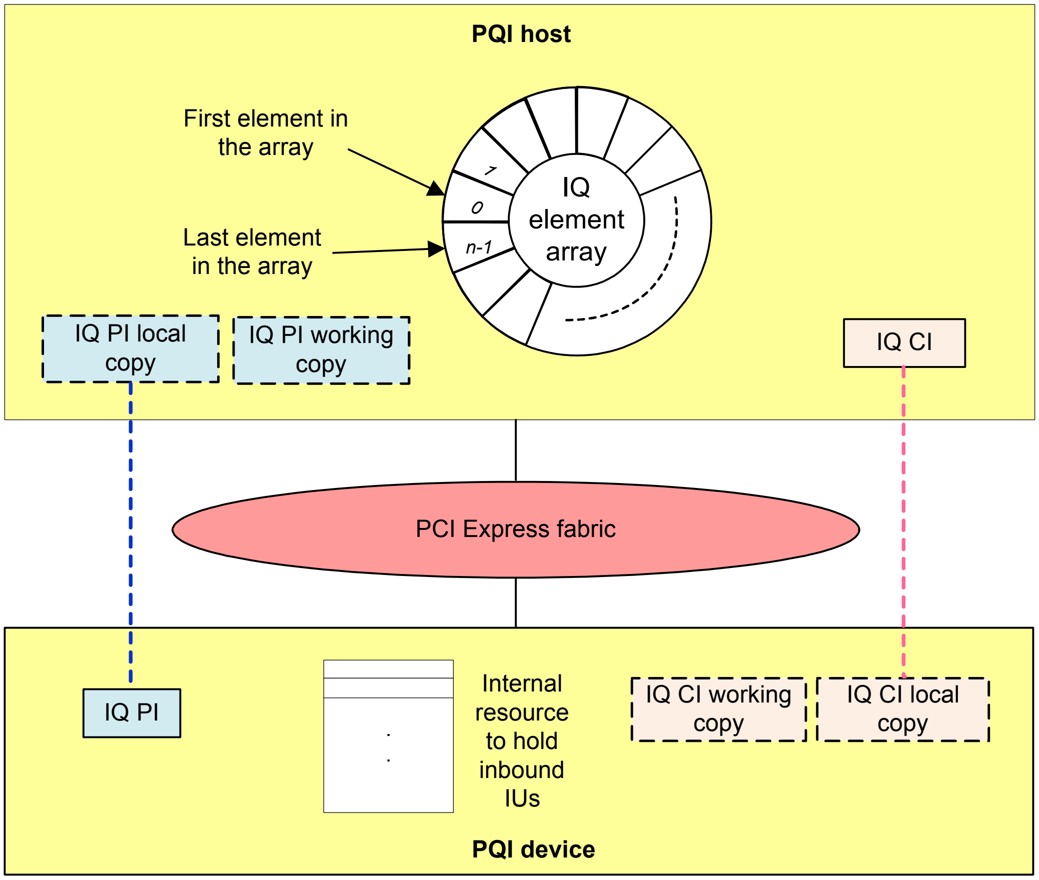


**Figure 19 — Typical location of IQ PI, PQI device IQ CI local copy, and PQI device internal resource to receive IUs**

The PQI device may dequeue one or more IUs from the IQ if the total number of elements required by the IUs to be dequeued is less than or equal to the number of PQI internal resource to hold the elements. The PQI device shall use the following steps to dequeue IUs from the IQ:

1) for each IU to be dequeued:
    1) read the IU from the IQ element array beginning at the address indicated by the ~~working copy of~~ IQ CI working copy, using wrapping ~~rules~~as described in 4.3.2.1; and
    2) update the ~~working copy of~~ IQ CI working copy;
2) update the ~~local copy of~~ IQ CI local copy with the ~~working copy of~~ IQ CI working copy; and
3) update the IQ CI with the ~~working copy of~~ IQ CI working copy.

If the PQI device supports IUs spanning multiple elements in an IQ, then the IU LENGTH field (see 8.2) value plus four may be larger than the IQ element length. If the PQI device does not support IUs spanning multiple elements in an IQ, then the number of bytes in an IU indicated by the IU LENGTH field plus four shall be less than or equal to the IQ element length (i.e., excludes the count of the IU header fields (see 8.2)).

If the PQI device does not support IUs spanning multiple elements in an operational IQ and the PQI device receives an operational IU with the IU LENGTH field value larger than the operational IQ element length, then the PQI device shall:

1) stop consuming from the operational IQ;
2) set the OP IQ ERROR bit to one in the PQI Device Status register (see 5.2.10); and
3) set the IQ ERROR bit to one in the Operational IQ property descriptor (see 9.2.10.3).

A PQI device shall not support IUs spanning multiple elements in an administrator IQ. If the PQI device receives an administrator IU with the IU LENGTH field value larger than the administrator IQ element length, then the PQI device shall respond as an error and follow the step as defined in 9.1.1.

#### 4.3.2.4.4 PQI device enqueuing IU(s) to an OQ

The PQI device determines the number of ~~available~~vacant OQ elements available to hold new IUs by comparing the OQ CI and the ~~local copy of~~ OQ PI local copy. The number of elements available to hold new IUs is equal to the number of vacant elements in the IQ minus one.

The PQI device ~~available~~ PQI internal resource that is available for sending the outbound IUs may ~~determine~~limit the maximum number of outbound IUs that are written to the OQ. The implementation of the PQI device internal resource for sending the outbound IUs is not defined by this standard.

Figure  shows the typical location of the OQ CI and the PQI device  OQ PI local copy used by the PQI device to determine the number of vacant OQ elements.

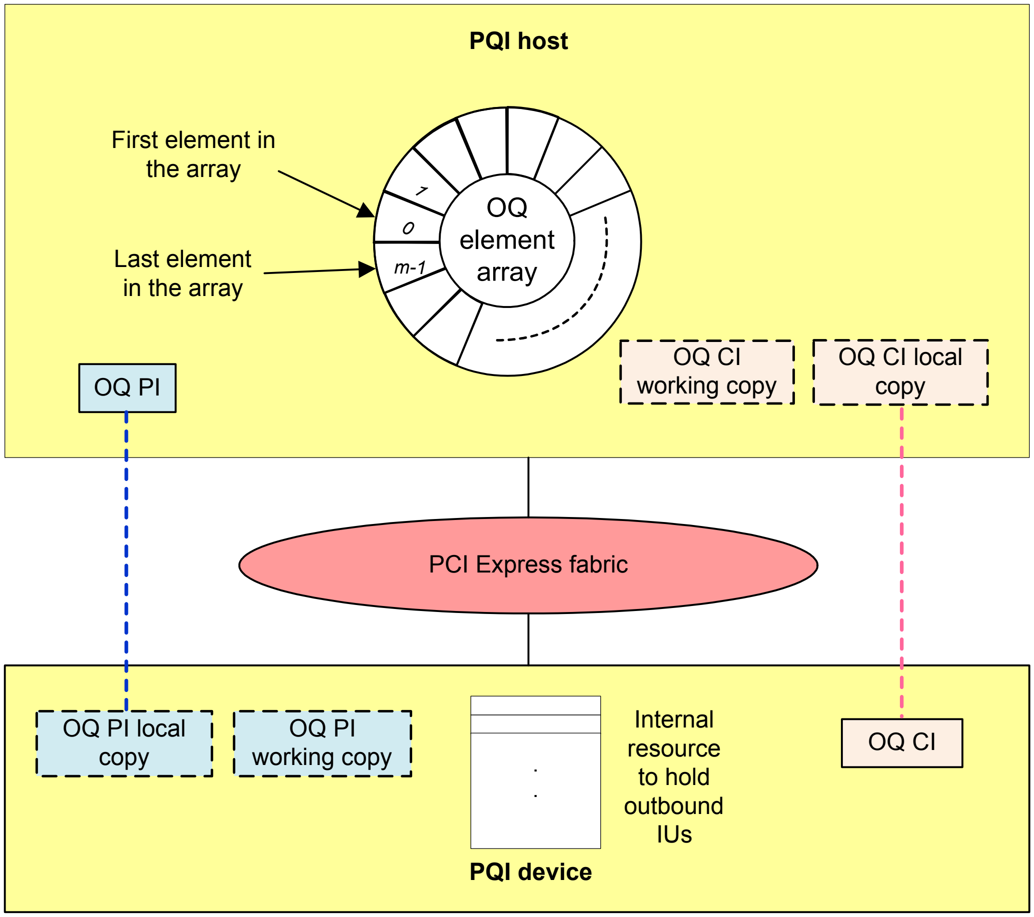


**Figure 20 — Typical location of OQ CI, PQI device OQ PI local copy and, PQI device internal resource for sending IUs**

The PQI device may enqueue one or more IUs to the OQ. If the total number of elements required by the IUs to be enqueued is less than or equal to the number of ~~available~~vacant OQ elements minus one, then the PQI device shall use the following steps to enqueue IUs to the OQ:

1) for each IU to be enqueued:
    1) write the IU into the OQ element array beginning at the address indicated by the ~~working copy of~~ OQ PI working copy, using wrapping ~~rules~~as described in 4.3.2.1; and
    2) update the ~~working copy of~~ OQ PI working copy;

2) update the ~~local copy of~~ OQ PI local copy with the ~~working copy of~~ OQ PI working copy; and
3) update the OQ PI with the ~~working copy of~~ OQ PI working copy.

If the PQI device supports IUs spanning multiple elements in an OQ, then the IU LENGTH field (see 8.2) value plus four may be larger than the OQ element length. If the PQI device does not support IUs spanning multiple elements in an OQ, then the number of bytes in an IU indicated by the IU LENGTH field plus four shall be less or equal to the OQ element length (i.e., excludes the count of the IU header fields (see 8.2)).

If the PQI device does not support IUs spanning multiple elements in an operational OQ and the PQI device receives from the information unit layer an operational IU with the IU LENGTH field value larger than the operational OQ element length, then the PQI device shall:

1) stop producing to the operational OQ;
2) set the OP OQ ERROR bit set to one in the PQI Device Status register (see 5.2.10);
3) set the OQ ERROR bit to one in the Operational OQ property descriptor (see 9.2.11.3).

A PQI device shall not support IUs spanning multiple elements in an administrator OQ. If the PQI device receives an administrator IU with the IU LENGTH field value larger than the administrator OQ element length, then the PQI device shall transition to the PD4:Error state (see 4.5.6) and set the error code to INVALID IU LENTH IN ADMINISTRATOR REQUEST IU.

This standard does not define specific reasons for a PQI device to stop producing to an OQ as a result of errors.

Editor's Note 3: Similar statement needs to be added in SOP.

### 4.3.3 Creating circular queues

#### 4.3.3.1 Creating circular queues overview

The PQI host requests that the administrator queue pair and the operational queues be created in the following order:

1) use ~~PQI device registers~~the CREATE ADMINISTRATOR QUEUE PAIR PD function ~~(see 5.2)~~ to create the administrator queue pair (i.e., administrator IQ and administrator OQ); and
2) use administrator functions (see 9.2) over the administrator queue pair to create operational queues.

#### 4.3.3.2 Creating the administrator queue pair

A PQI device shall support one administrator IQ and one administrator OQ (i.e., an administrator queue pair).

If the FUNCTION AND STATUS CODE field is set to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register (see 5.2.5) and the PQI DEVICE STATE field is set to 2h (i.e., PD2:~~Register_Ready~~All_Registers_Ready) in the PQI Device Status register (see 5.2.10), then the PQI host may create an administrator queue pair using the following steps:

1) read the PQI ~~device~~Device Capability register to determine the element length and the maximum number of ~~the~~ elements in both the administrator IQ and administrator OQ;
2) allocate the memory spaces using a mechanism outside the scope of this standard for the following:
   A) administrator IQ element array at an address with the required alignment (see 5.2.13);
   B) administrator OQ element array at an address with the required alignment (see 5.2.14);
   C) administrator IQ CI at an address with the required alignment (see 5.2.15); and
   D) administrator OQ PI at an address with the required alignment (see 5.2.16);
3) set:
   A) the administrator IQ CI to zero;
   B) the administrator OQ PI to zero;
   C) the ~~local copy of~~ the administrator IQ PI local copy to zero; and
   D) the ~~local copy of~~ the administrator OQ CI local copy to zero;
4) set the PQI device registers as shown in table 15;

5) set the value of the FUNCTION AND STATUS CODE field to 01h (i.e., CREATE ADMINISTRATOR QUEUE PAIR) in the Administrator Queue Configuration Function register;
6) read the Administrator Queue Configuration Function register until:
   A) the FUNCTION AND STATUS CODE field is set to 00h (i.e., IDLE); or
   B) 100 ms has elapsed after step 5);
7) if the FUNCTION AND STATUS CODE field is not set to 00h (i.e., IDLE) and 100 ms has elapsed, then read the Administrator Queue Configuration Function register;
8) if the FUNCTION AND STATUS CODE field is set to 00h, then:
   A) read the administrator IQ PI offset from the Administrator IQ PI Offset register and save the value in PQI host local memory; and
   B) read the administrator OQ CI offset from the Administrator OQ CI Offset register and save the value in PQI host local memory;
9) if the FUNCTION AND STATUS CODE field is not set to 00h and PD state machine is in PD4:Error state, then read the PQI Device Status register, the PQI Device Error register, and the PQI Device Error ~~Data~~Details register, and report an error in a vendor specific manner; and
10) if the FUNCTION AND STATUS CODE field is not 00h and the PD state machine is not in PD4:Error state, then report an error in a vendor specific manner.

**Table 15 — PQI device registers to be ~~set~~written during administrator queue pair creation**

| Register | Field | Value |
|---|---|---|
| Administrator IQ Element Array Address (see 5.2.13) | ADMINISTRATOR IQ ELEMENT ARRAY ADDRESS | Administrator IQ element array address |
| Administrator OQ Element Array Address (see 5.2.14) | ADMINISTRATOR OQ ELEMENT ARRAY ADDRESS | Administrator OQ element array address |
| Administrator IQ CI Address ~~register~~ (see 5.2.15) | ADMINISTRATOR IQ CI ADDRESS | Administrator IQ CI address |
| Administrator OQ PI Address ~~register~~ (see 5.2.16) | ADMINISTRATOR OQ PI ADDRESS | Administrator OQ PI address |
| Administrator Queue Parameter (see 5.2.17) | NUMBER OF ADMINISTRATOR IQ ELEMENTS | The number of elements in the administrator IQ |
| | NUMBER OF ADMINISTRATOR OQ ELEMENTS | The number of elements in the administrator OQ |
| | INTERRUPT MESSAGE NUMBER | The MSI-X Table entry used to generate the interrupt message for OQ PI updates to the administrator OQ if MSI-X is enabled |

If the FUNCTION AND STATUS CODE field is not set to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register (see 5.2.5) or the PQI DEVICE STATE field is not set to 2h (i.e., PD2:~~Register_Ready~~All_Registers_Ready) in the PQI Device Status register (see 5.2.10), then the PQI device shall fail the creation of the administrator queue pair and the PD state machine (see 4.5) ~~shall~~ transitions to the PD4:Error state (see 4.5.6).

When the PQI device is requested to create the administrator queue pair, the PQI device shall:

1) allocate PQI device memory space for the administrator IQ PI;
2) set the ADMINISTRATOR IQ PI OFFSET field to the administrator IQ PI offset in the Administrator IQ PI Offset register (see 5.2.11);

3) allocate PQI device memory space for the administrator OQ CI;
4) set the ADMINISTRATOR OQ CI OFFSET field to the administrator OQ CI offset in the Administrator OQ CI Offset register (see 5.2.12); and
5) initialize:
   A) the administrator IQ PI to zero;
   B) the administrator OQ CI to zero;
   C) the administrator IQ CI local copy to zero; and
   D) the administrator OQ PI local copy to zero.

If the PQI device successfully completes creating the administrator queue pair, then the PQI device shall set:

1) the FUNCTION AND STATUS CODE field to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register; and
2) transition the PD state machine (see 4.5) to the PD3:Administrator_Queue_Pair_Ready state.

~~If the PQI device successfully completes creating the administrator queue pair, then the PQI device shall set:~~

c) ~~the administrator IQ PI to zero;~~
d) ~~the administrator OQ CI to zero;~~
e) ~~the local copy of the administrator IQ CI to zero;~~
f) ~~the local copy of the administrator OQ PI to zero;~~
g) ~~the FUNCTION AND STATUS CODE field to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register; and~~
h) ~~transition the PD state machine (see 4.5) to PD3:Administrator_Queue_Pair_Ready state.~~

If the PQI device fails to complete creating the administrator queue pair, then the PQI device shall:

a) not change the FUNCTION AND STATUS CODE field from the current value of 01h (i.e., CREATING ADMINISTRATOR QUEUE PAIR) in the Administrator Queue Configuration Function register;
b) set the error code to ~~ADMINISTRATOR QUEUE PAIR CREATE ERROR~~ERROR CREATING ADMINISTRATOR QUEUE PAIR in the PQI Device Error register (see 5.2.18); and
c) ~~set the BYTE POINTER field to 0h, the BIT POINTER field to 00b, and the BPV bit to zero in the PQI Device Error register (see 5.2.18); and~~
d) transition the PD state machine (see 4.5) to the PD4:Error state (see 4.5.6).

#### 4.3.3.3 Creating operational queues

Operational queues are created by  the CREATE OPERATIONAL IQ function (see 9.2.4) and the CREATE OPERATIONAL OQ function (see 9.2.5) using the administrator queue pair.

~~Prior to enqueuing a CREATE OPERATIONAL IQ request to the administrator IQ, the PQI host should set:~~

a) ~~the operational IQ CI to zero; and~~
b) ~~the PQI host's local copy of the operational IQ PI to zero.~~

Prior to enqueuing a CREATE OPERATIONAL IQ request to the administrator IQ, the PQI host should:

1) use the REPORT PQI DEVICE CAPABILITY administrator function (see 9.2.2) to determine the supported address alignments, element length, and number of elements;
2) allocate PCI memory spaces using a mechanism outside the scope of this standard for the following:
   A) the operational IQ element array, at an address with the alignment indicated by the OPERATIONAL IQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3); and
   B) the operational IQ CI, at an address with the alignment indicated by the OPERATIONAL IQ CI ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data;

   and

3) initialize the values in:
   A) the operational IQ CI to zero; and
   B) the operational IQ PI local copy to zero.

When the PQI device processes a CREATE OPERATIONAL IQ request, the PQI device shall:

1) allocate PQI device memory space for the administrator IQ PI; and
2) initialize:
   - A) the operational IQ PI to zero; and
   - B) the operational IQ CI local copy to zero.

If the PQI device successfully completes creating the operational IQ, then the PQI device shall enqueue a CREATE OPERATIONAL IQ response with the STATUS field set to 00h (i.e., GOOD) (see 9.1.4.1).

If the PQI device fails to complete creating the operational IQ, then the PQI device shall enqueue a CREATE OPERATIONAL IQ response with the STATUS field and the additional status descriptor set to a value that indicates the error (see 9.1.4.1).

~~Prior to enqueuing a CREATE OPERATIONAL IQ response with the STATUS field set to 00h (i.e., GOOD), the PQI device shall set:~~

a) ~~the operational IQ PI to zero; and~~
b) ~~the PQI device's local copy of the operational IQ CI to zero.~~

~~Prior to enqueuing a CREATE OPERATIONAL OQ request to the administrator IQ, the PQI host should set:~~

a) ~~the operational OQ PI to zero; and~~
b) ~~the PQI host's local copy of the operational OQ CI to zero.~~

Prior to enqueuing a CREATE OPERATIONAL OQ request to the administrator IQ, the PQI host should:

1) use the REPORT PQI DEVICE CAPABILITY administrator function to determine the supported address alignments, element length, and number of elements;
2) allocate PCI memory spaces using a mechanism outside the scope of this standard for the following:
   - A) the operational OQ element array, at an address with the alignment indicated by the OPERATIONAL OQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data; and
   - B) the operational OQ PI, at an address with the alignment indicated by the OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data;

   and
3) initialize the values in:
   - A) the operational OQ PI to zero; and
   - B) the operational OQ CI local copy to zero.

When the PQI device processes a CREATE OPERATIONAL OQ request, the PQI device shall:

1) allocate PQI device memory space for the administrator OQ CI; and
2) initialize:
   - A) the operational OQ CI to zero; and
   - B) the operational OQ PI local copy to zero.

If the PQI device successfully completes creating the operational OQ, then the PQI device shall enqueue a CREATE OPERATIONAL OQ response with the STATUS field set to 00h (i.e., GOOD) (see 9.1.4.1).

If the PQI device fails to complete creating the operational IQ, then the PQI device shall enqueue a CREATE OPERATIONAL OQ response with the STATUS field and the additional status descriptor set to a value that indicates the error (see 9.1.4.1).

~~Prior to enqueuing a CREATE OPERATIONAL OQ response with the STATUS field set to 00h (i.e., GOOD), the PQI device shall set:~~

a) ~~the operational OQ CI to zero; and~~
b) ~~the PQI device's local copy of the operational OQ PI to zero.~~

### 4.3.4 Deleting circular queues

#### 4.3.4.1 Deleting circular queues overview

The PQI host requests that the administrator queue pair and the operational queues be deleted in the following order:

1) delete all of the operational IQs and operational OQs using administrator functions (see 9.2); and
2) delete the administrator queue pair using ~~PQI device registers~~the DELETE ADMINISTRATOR QUEUE PAIR PD function~~(see 5.2)~~ (see 5.2.5).

#### 4.3.4.2 Deleting the administrator queue pair

All operational queues of a PQI device should be deleted prior to deleting the administrator queue pair using the DELETE ADMINISTRATOR QUEUE PAIR PD function (see 5.2.5). If all operational queues have not been deleted and the PQI device receives the delete administrator queue pair request, then the PQI device shall fail the deletion of the administrator queue pair and the PD state machine (see 4.5) ~~shall~~ transitions to the PD4:Error state (see 4.5.6).

If the FUNCTION AND STATUS CODE field ~~in the Administrator Queue Configuration Function register~~ is set to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register (see 5.2.5) and the ~~PQI Device Status register~~ PQI DEVICE STATE field is set to 3h (i.e., PD3:Administrator_Queue_Pair_Ready) in PQI Device Status register (see 5.2.10), then the PQI host may delete the administrator queue pair using the following steps:

1) set the value of the FUNCTION AND STATUS CODE field to 02h (i.e., DELETE ADMINISTRATOR QUEUE PAIR) in the Administrator Queue Configuration Function register;
2) read the Administrator Queue Configuration Function register until:
   A) the FUNCTION AND STATUS CODE field is set to 00h (i.e., IDLE); or
   B) 100 ms has elapsed;
3) if the FUNCTION AND STATUS CODE field is not set to 00h (i.e., IDLE)~~ and 100 ms has elapsed~~, then read the Administrator Queue Configuration Function register; ~~and~~
4) if the FUNCTION AND STATUS CODE field is not set to 00h (i.e., IDLE), then:
   A) read the PQI Device Status register;
   B) read the PQI Device Error register (see 5.2.18); and
   C) report an error in a vendor specific manner.

   and

5) deallocate the PCI memory spaces using a mechanism outside the scope of this standard for the following:
   A) administrator IQ element array;
   B) administrator OQ element array;
   C) administrator IQ CI; and
   D) administrator OQ PI.

If the FUNCTION AND STATUS CODE field in the Administrator Queue Configuration Function register is not set to 00h (i.e., IDLE) (see 5.2.5) or the PQI Device Status register PQI DEVICE STATE field is not set to 3h (i.e., PD3:Administrator_Queue_Pair_Ready) (see 5.2.10), then the PQI device shall fail the deletion of the administrator queue pair and the PD state machine (see 4.5) ~~shall~~ transitions to the PD4:Error state (see 4.5.6).

When the PQI device is requested to delete the administrator queue pair, the PQI device shall:

1) deallocate PQI device memory space for the administrator IQ PI;
2) set the ADMINISTRATOR IQ PI OFFSET field to zero in the Administrator IQ PI Offset register (see 5.2.11);
3) deallocate PQI device memory space for the administrator OQ CI; and
4) set the ADMINISTRATOR OQ CI OFFSET field to zero in the Administrator OQ CI Offset register (see 5.2.12).

If the PQI device successfully completes deleting the administrator queue pair, then the PQI device shall:

1) set the FUNCTION AND STATUS CODE field to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register; and

2) transition the PD state machine (see 4.5) to the PD2:~~Register_Ready~~All_Registers_Ready state (see 4.5.4).

If the PQI device fails to complete the deletion of the administrator queue pair, then the PQI device shall:

a) not change the FUNCTION AND STATUS CODE field from the current value of 02h (i.e., DELETING ADMINISTRATOR QUEUE PAIR) in the Administrator Queue Configuration Function register;
b) set the error code to ~~ADMINISTRATOR QUEUE PAIR DELETE ERROR~~ERROR DELETING ADMINISTRATOR QUEUE PAIR in the PQI Device Error register (see 5.2.18); and
c) ~~set the BYTE POINTER field to 0h, the BIT POINTER field to 000b and the BPV bit to zero in the PQI Device Error register (see 5.2.18)~~;
d) transition the PD state machine (see 4.5) to the PD4:Error state (see 4.5.6).

A PQI device shall delete the administrator queue pair during a PQI reset or a PCI Express reset.

After a PQI reset or a PCI Express reset, the PQI host may deallocate the PCI memory spaces using a mechanism outside the scope of this standard for the following:

a) administrator IQ element array;
b) administrator OQ element array;
c) administrator IQ CI; and
d) administrator OQ PI.

### 4.3.4.3 Deleting operational queues

Operational queues are deleted by the DELETE OPERATIONAL IQ function (see 9.2.6) and the DELETE OPERATIONAL OQ function (see 9.2.7) using the administrator queue pair.

The PQI management application client should wait until the corresponding IQ is empty before issuing the DELETE OPERATIONAL IQ function. The results of deleting an operational IQ when it is not empty is not defined by this standard.

The PQI management application client should make sure that the corresponding operational OQ is not used before issuing the DELETE OPERATIONAL OQ function. The results of deleting an operational OQ when the operational OQ is still being used is not defined by this standard.

After dequeuing a DELETE OPERATIONAL IQ response from the administrator OQ, the PQI host should:

1) deallocate the PCI memory spaces using a mechanism outside the scope of this standard for the following:
   A) operational IQ element array; and
   B) operational IQ CI.

When the PQI device is requested to delete an operational IQ, the PQI device shall deallocate PQI device memory space for the operational IQ PI.

After dequeuing a DELETE OPERATIONAL OQ response from the administrator OQ, the PQI host should:

1) deallocate the PCI memory spaces using a mechanism outside the scope of this standard for the following:
   A) operational OQ element array; and
   B) operational OQ PI.

When the PQI device is requested to delete an operational OQ, the PQI device shall deallocate PQI device memory space for the operational OQ CI.

A PQI device shall delete all operational IQs and all operational OQs during a PQI reset or a PCI Express reset.

After a PQI reset or a PCI Express reset, the PQI host may deallocate the PCI memory spaces using a mechanism outside the scope of this standard for the following:

a) operational IQ element arrays;
b) operational OQ element arrays;
c) operational IQ CIs; and
d) operational OQ PIs.

#### 4.3.5 IQ priority

A PQI device shall ~~remove~~consume IUs from the IQs based on the vendor-specific priority level assigned to each IQ. A PQI device shall support at least two priority levels for IQs:

a) the administrator IQ shall be assigned the highest priority level; and
b) each operational IQ shall be assigned a lower priority level than the administrator IQ.

The PQI device ~~shall~~should ~~remove~~consume all ~~available~~ IUs from the administrator IQ (i.e., the administrator IQ is empty) before ~~removing~~consuming IUs from any operational IQ.

#### 4.3.6 PQI class diagrams

#### 4.3.6.1 Circular Queue class

#### 4.3.6.1.1 Circular Queue class overview

Figure 18 shows:

a) the Circular Queue class;
b) the Element Array class (see 4.3.6.2);
c) the Element class (see 4.3.6.3);
d) the PI class (see 4.3.6.4);
e) the CI class (see 4.3.6.5);
f) the IQ class (see 4.3.6.6);
g) the Administrator IQ class (see 4.3.6.7);
h) the Operational IQ class (see 4.3.6.8);
i) the OQ class (see 4.3.6.9);
j) the Administrator OQ class (see 4.3.6.10); and
k) the Operational OQ class (see 4.3.6.11).

~~The Circular Queue class (see figure 21) contains:~~

a) ~~the Element Array class (see 4.3.6.2), which contains:~~
   A) ~~the Element class (see 4.3.6.3);~~
b) ~~the PI class (see 4.3.6.4); and~~
c) ~~the CI class (see 4.3.6.5).~~

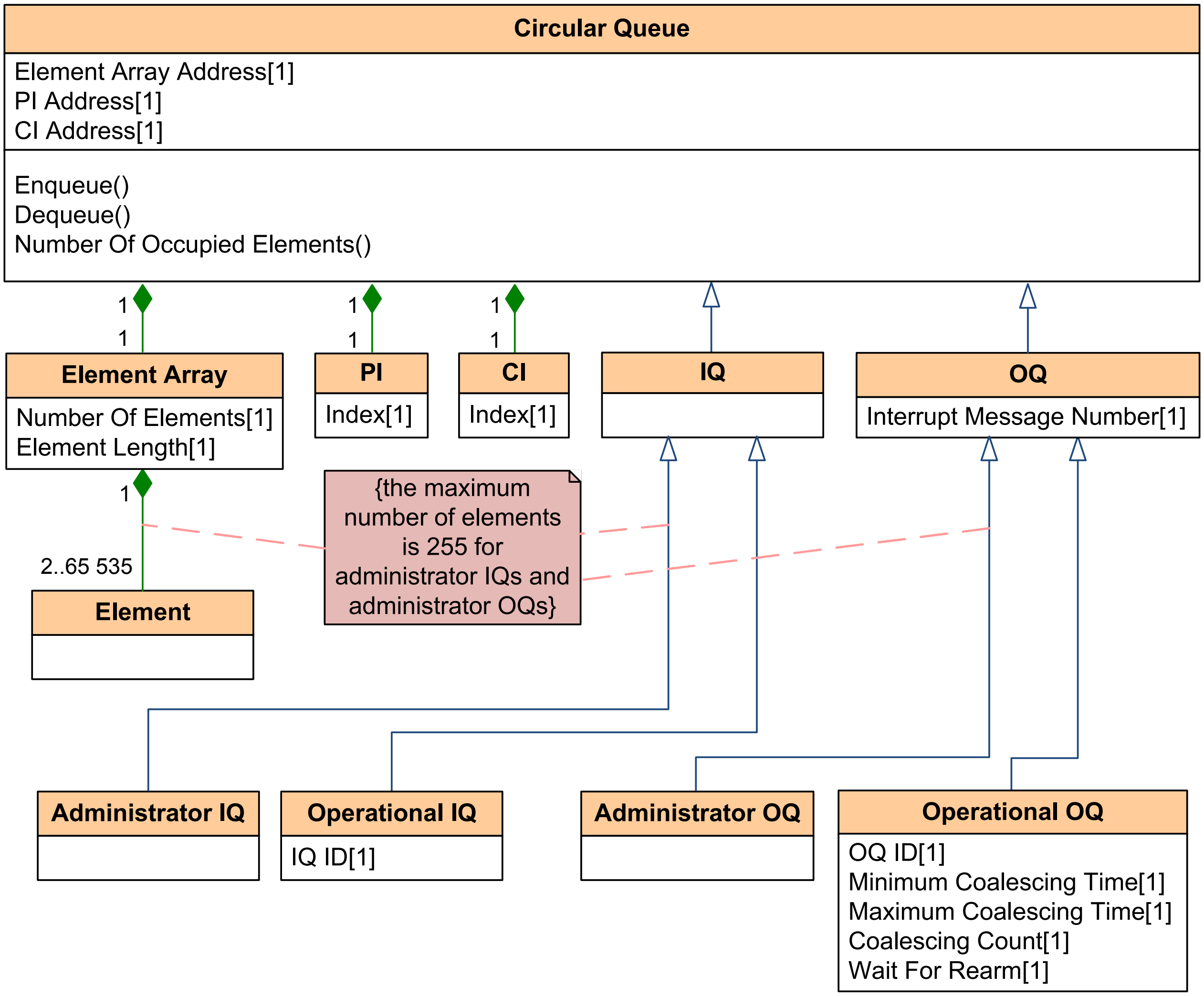


**Figure 21 — Circular Queue class and related classes**

~~An administrator IQ (see 4.3.6.7) is a kind of circular queue.~~

~~An administrator OQ (see 4.3.6.10) is a kind of circular queue.~~

~~An operational IQ (see 4.3.6.8) is a kind of circular queue.~~

~~An operational OQ (see 4.3.6.11) is a kind of circular queue.~~

~~Each instance of a Circular Queue class shall contain the following objects:~~

a) ~~one element array;~~
b) ~~one PI; and~~
c) ~~one CI.~~

~~Each instance of an Element Array class shall contain the following objects:~~

a) ~~2 elements to 65 535 elements.~~

~~Each instance of an Element class shall contain from 16 to 4 080 bytes and shall be an integer multiple of 16.~~

#### 4.3.6.1.2 Element Array Address attribute

This attribute contains the ~~64-bit~~ memory space address of the element array.

For the operational IQ, the minimum address alignment of the element array is indicated by the OPERATIONAL IQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3).

For the administrator IQ, the minimum address alignment of the element array is 64 bytes.

For the operational OQ, the minimum address alignment of the element array is indicated by the OPERATIONAL OQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3).

For the administrator OQ, the minimum address alignment of the element array is 64 bytes.

For an administrator IQ, this attribute is specified and indicated in the ADMINISTRATOR IQ ELEMENT ARRAY ADDRESS field in the Administrator IQ Element Array Address register (see 5.2.13).

For an administrator OQ, this attribute is specified and indicated in the ADMINISTRATOR OQ ELEMENT ARRAY ADDRESS field in the Administrator OQ Element Array Address register (see 5.2.14).

For an operational IQ, this attribute is:

a) specified in the IQ ELEMENT ARRAY ADDRESS field in the CREATE OPERATIONAL IQ request (see 9.2.4.1); and
b) indicated in the IQ ELEMENT ARRAY ADDRESS field in the operational IQ property descriptor in the REPORT OPERATIONAL IQ LIST parameter data (see 9.2.10.3).

For an operational OQ, this attribute is:

a) specified in the OQ ELEMENT ARRAY ADDRESS field in the CREATE OPERATIONAL OQ request (see 9.2.5.1); and
b) indicated in the OQ ELEMENT ARRAY ADDRESS field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3).

#### 4.3.6.1.3 PI Address attribute

This attribute contains the memory space address of the PI.

For the operational OQ, the minimum address alignment of the PI is indicated by the OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3).

For the administrator OQ, the minimum address alignment of the PI is 64 bytes.

For an administrator IQ, this attribute is specified and indicated in the ADMINISTRATOR IQ PI OFFSET field in the Administrator IQ PI Offset register (see 5.2.11).

For an administrator OQ, this attribute is specified and indicated in the ADMINISTRATOR OQ PI OFFSET field in the Administrator OQ PI Address register (see 5.2.16).

For an operational IQ, this attribute is:

a) specified by the PQI device;~~and~~
b) indicated in the IQ PI OFFSET field in the CREATE OPERATIONAL IQ response (see 9.2.4.2); and
c) indicated in the IQ PI OFFSET field in the operational IQ property descriptor in the REPORT OPERATIONAL IQ LIST parameter data (see 9.2.10.3).

For an operational OQ, this attribute is:

a) specified in the OQ PI ADDRESS field in the CREATE OPERATIONAL OQ request (see 9.2.5.1); and
b) indicated in the OQ PI ADDRESS field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3).

#### 4.3.6.1.4 CI Address attribute

This attribute contains the memory space address of the CI.

For the operational IQ, the minimum address alignment of the CI is indicated by the OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3).

For the administrator IQ, the minimum address alignment of the CI is 64 bytes.

For an administrator IQ, this attribute is specified and indicated in the ADMINISTRATOR IQ CI ADDRESS field in the Administrator IQ CI Address register (see 5.2.15).

For an administrator OQ, this attribute is indicated in the ADMINISTRATOR OQ CI ADDRESS field in the Administrator OQ CI Offset register (see 5.2.12).

For an operational IQ, this attribute is:

a) specified in the IQ CI ADDRESS field in the CREATE OPERATIONAL IQ request (see 9.2.4.1); and
b) indicated in the IQ CI ADDRESS field in the operational IQ property descriptor in the REPORT OPERATIONAL IQ LIST parameter data (see 9.2.10.3).

For an operational OQ, this attribute is:

a) specified by the PQI device; ~~and~~
b) indicated in the OQ CI OFFSET field in the CREATE OPERATIONAL OQ response (see 9.2.5.2); and
c) indicated in the OQ CI OFFSET field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3).

#### 4.3.6.1.5 Enqueue operation

This operation has the following procedure call:

**Enqueue (IN (Elements, Number of Elements))**

~~This operation adds one or more elements to the element array and increments the PI. This operation waits for the queue to have enough available elements to hold the requested number of elements (see 4.3.2.4.1 and 4.3.2.4.4).~~

This operation adds one or more occupied elements to the element array and increments the PI wrapping at the end of the element array. This operation waits for the circular queue to have the specified number of vacant elements (see 4.3.2.4.1 and 4.3.2.4.4).

#### 4.3.6.1.6 Dequeue operation

This operation has the following procedure call:

**Elements = Dequeue (IN (Number of Elements))**

~~This operation returns one or more elements from the element array and increments the CI. This operation waits for the queue to have enough available elements to hold the requested number of elements (see 4.3.2.4.2 and 4.3.2.4.3).~~

This operation returns one or more occupied elements from the element array and increments the CI wrapping at the end of the element array. This operation waits for the circular queue to have the specified number of occupied elements (see 4.3.2.4.2 and 4.3.2.4.3).

~~4.2.6.1.7 Peek operation~~

~~This operation has the following procedure call:~~

~~**Elements = Peek (Number of Elements)**~~

~~This operation returns one or more elements from the element array without incrementing the CI.~~

~~4.2.6.1.8 Number Of Valid Elements operation~~

~~This operation has the following procedure call:~~

~~**Elements = Number Of Valid Elements ()**~~

~~This operation returns the number of valid elements (i.e., PI.Index - CI.Index, accounting for wrapping).~~

### 4.3.6.2 Element Array class

#### 4.3.6.2.1 Element Array ~~Address~~ class overview

The Element Array class contains:

a) the Element class (see 4.3.6.3).

#### 4.3.6.2.2 Number Of Elements attribute

This attribute contains the number of elements in the element array.

For an administrator IQ, this attribute is indicated in the NUMBER OF ADMINISTRATOR IQ ELEMENTS field in the Administrator Queue Parameter register (see 5.2.17).

For an administrator OQ, this attribute is indicated in the NUMBER OF ADMINISTRATOR OQ ELEMENTS field in the Administrator Queue Parameter register (see 5.2.17).

For an operational IQ, this attribute is:

a) specified in the NUMBER OF ELEMENTS field in the CREATE OPERATIONAL IQ request (see 9.2.4.1); and
b) indicated in the NUMBER OF ELEMENTS field in the operational IQ property descriptor in the REPORT OPERATIONAL IQ LIST parameter data (see 9.2.10.3).

For an operational OQ, this attribute is:

a) specified in the NUMBER OF ELEMENTS field in the CREATE OPERATIONAL OQ request (see 9.2.5.1); and
b) indicated in the NUMBER OF ELEMENTS field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3).

For an administrator IQ, the minimum value is two and the maximum value is indicated in the MAXIMUM ADMINISTRATOR IQ ELEMENTS field in the PQI Device Capability register (see 5.2.6). The maximum value is less than or equal to 255.

For an administrator OQ, the minimum value is two and the maximum value is indicated in the MAXIMUM ADMINISTRATOR OQ ELEMENTS field in the PQI Device Capability register (see 5.2.6). The maximum value is less than or equal to 255.

For an operational IQ, the minimum value is two and the maximum value is indicated in the MAXIMUM OPERATIONAL IQ ELEMENTS field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3). The maximum value is less than or equal to 65 535.

For an operational OQ, the minimum value is two and the maximum value is indicated in the MAXIMUM OPERATIONAL OQ ELEMENTS field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3). The maximum value is less than or equal to 65 535.

#### 4.3.6.2.3 Element Length attribute

This attribute contains the length in bytes of an element in the element array (i.e., the element length).

For an administrator IQ, this attribute is indicated in the ADMINISTRATOR IQ ELEMENT LENGTH field in the PQI Device Capability register (see 5.2.6).

For an administrator OQ, this attribute is indicated in the ADMINISTRATOR OQ ELEMENT LENGTH field in the PQI Device Capability register (see 5.2.6).

For an operational IQ, this attribute is:

a) specified in the ELEMENT LENGTH field in the CREATE OPERATIONAL IQ request (see 9.2.4.1); and
b) indicated in the ELEMENT LENGTH field in the operational IQ property descriptor in the REPORT OPERATIONAL IQ LIST parameter data (see 9.2.10.3).

For an operational OQ, this attribute is:

a) specified in the ELEMENT LENGTH field in the CREATE OPERATIONAL OQ request (see 9.2.5.1); and
b) indicated in the ELEMENT LENGTH field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3).

The element length shall be a multiple of 16 bytes. The minimum value is 16 bytes and the maximum value is 1 048 560 (i.e., 16 × 65 535) bytes.

#### 4.3.6.3 Element class

This class represents an element in an element array.

Each instance of an Element class shall contain from 16 to 4 080 bytes and shall be an integer multiple of 16.

~~4.2.6.3.1 Element Length attribute~~

~~This attribute contains the element length of an element in the element array.~~

~~For an administrator IQ, this attribute is indicated in the ADMINISTRATOR IQ ELEMENT LENGTH field in the Capability register (see 5.2.6).~~

~~For an administrator OQ, this attribute is indicated in the ADMINISTRATOR OQ ELEMENT LENGTH field in the Capability register (see 5.2.6).~~

NOTE 16 - ~~The Element Length attribute contains the number of bytes, while the ADMINISTRATOR IQ ELEMENT LENGTH field and the ADMINISTRATOR OQ ELEMENT LENGTH field in the Capability register is in 16 byte units.~~

~~For an operational IQ, this attribute is specified in the ELEMENT LENGTH field in the CREATE OPERATIONAL IQ request (see 9.2.4.1) and indicated in the ELEMENT LENGTH field in the operational IQ property descriptor in the REPORT OPERATIONAL IQ LIST parameter data (see 9.2.10.3).~~

~~For an operational OQ, this attribute is specified in the ELEMENT LENGTH field in the CREATE OPERATIONAL OQ request (see 9.2.5.1) and indicated in the ELEMENT LENGTH field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3).~~

~~The minimum value is 16 and the maximum value is 4 080 (i.e., 16 × 255).~~

NOTE 17 - ~~The Element Length attribute contains the number of bytes, while the ELEMENT LENGTH field in the administrator IUs is in 16 byte units.~~

#### 4.3.6.4 PI class

#### 4.3.6.4.1 PI class overview

This class represents the PI (see 4.3.2.1).

#### 4.3.6.4.2 Index attribute

This attribute indicates the contents of the PI.

#### 4.3.6.5 CI class

#### 4.3.6.5.1 CI class overview

This class represents the CI (see 4.3.2.1).

#### 4.3.6.5.2 Index attribute

This attribute indicates the contents of the CI.

#### 4.3.6.6 IQ class

~~4.2.6.6.1 IQ class overview~~

This class represent the IQ (see 4.3.2.2).

#### 4.3.6.7 Administrator IQ class

~~4.2.6.6.2.1 Administrator IQ class overview~~

The administrator IQ is described in 4.1 and 4.3.2.2.

#### 4.3.6.8 Operational IQ class

##### 4.3.6.8.1 Operational IQ class overview

The operational IQ is described in 4.1 and 4.3.2.2.

##### 4.3.6.8.2 IQ ID attribute

This attribute contains the IQ ID (see 4.1).

This attribute is specified in the IQ ID field in the CREATE OPERATIONAL IQ request (see 9.2.4.1) and indicated in the IQ ID field in the operational IQ property descriptor in the REPORT OPERATIONAL IQ LIST parameter data (see 9.2.10.3).

The minimum value is one and the maximum value is 65 535.

#### 4.3.6.9 OQ class

##### 4.3.6.9.1 OQ class overview

This class represent the IQ (see 4.3.2.3).

##### 4.3.6.9.2 Interrupt Message Number attribute

This attribute contains the interrupt message number.

The minimum value is 0 and the maximum value is 2 047.

#### 4.3.6.10 Administrator OQ class

##### 4.3.6.10.1 Administrator OQ class overview

The administrator OQ is described in 4.1 and 4.3.2.3.

##### 4.3.6.10.2 Interrupt Message Number attribute

~~This attribute contains the interrupt message number~~

This attribute is inherited from the OQ class.

This attribute is:

a) specified in the INTERRUPT MESSAGE NUMBER field in the Administrator Queue Parameter register (see 5.2.17) when the administrator queue pair are created; and
b) indicated in the INTERRUPT MESSAGE NUMBER field in the Administrator Queue Parameter register when the PQI device is in state PD3:Administrator_Queue_Pair_Ready state (see 4.5.5).

~~The minimum value is 0 and the maximum value is 2 047.~~

#### 4.3.6.11 Operational OQ class

##### 4.3.6.11.1 Operational OQ class overview

The operational OQ is described in 4.1 and 4.3.2.3.

##### 4.3.6.11.2 OQ ID attribute

This attribute contains the OQ ID (see 4.1).

This attribute is:

a) specified in the OQ ID field in the CREATE OPERATIONAL OQ request (see 9.2.5.1); and
b) indicated in the OQ ID field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3).

The minimum value is one and the maximum value is 65 535.

##### 4.3.6.11.3 Minimum Coalescing Time attribute

This attribute contains the minimum coalescing time in 100 ns intervals.

This attribute:

a) is specified in the MINIMUM COALESCING TIME field in the CREATE OPERATIONAL OQ request (see 9.2.5.1);
b) is indicated in the MINIMUM COALESCING TIME field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3); and
c) may be modified as specified in the MINIMUM COALESCING TIME field in the CHANGE OPERATIONAL OQ PROPERTIES request (see 9.2.9.1).

The minimum value is zero and the maximum value is 65 535.

##### 4.3.6.11.4 Maximum Coalescing Time attribute

This attribute contains the maximum coalescing time in 100 ns intervals.

This attribute:

a) is specified in the MAXIMUM COALESCING TIME field in the CREATE OPERATIONAL OQ request (see 9.2.5.1);
b) is indicated in the MAXIMUM COALESCING TIME field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3); and
c) may be modified as specified in the MAXIMUM COALESCING TIME field in the CHANGE OPERATIONAL OQ PROPERTIES request (see 9.2.9.1);

The minimum value is zero and the maximum value is 65 535.

##### 4.3.6.11.5 Coalescing Count attribute

This attribute contains the coalescing count.

This attribute:

a) is specified in the COALESCING COUNT field in the CREATE OPERATIONAL OQ request (see 9.2.5.1);
b) is indicated in the COALESCING COUNT field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3); and
c) may be modified as specified in the COALESCING COUNT field in the CHANGE OPERATIONAL OQ PROPERTIES request (see 9.2.9.1).

The minimum value is zero and the maximum value is 65 535.

#### 4.3.6.11.6 Wait For Rearm attribute

This attribute indicates the wait for rearm enable.

This attribute:

a) is specified in the WAIT FOR REARM bit in the CREATE OPERATIONAL OQ request (see 9.2.5.1);
b) is indicated in the WAIT FOR REARM bit in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3); and
c) may be modified as specified in the WAIT FOR REARM bit in the CHANGE OPERATIONAL OQ PROPERTIES request (see 9.2.9.1);

The minimum value is 0 and the maximum value is 1.

#### 4.3.6.11.7 Interrupt Message Number attribute

~~This attribute contains the interrupt message number~~

This attribute is inherited from the OQ class.

This attribute is:

a) specified in the INTERRUPT MESSAGE NUMBER field in the CREATE OPERATIONAL OQ request (see 9.2.5.1); and
b) indicated in the INTERRUPT MESSAGE NUMBER field in the operational OQ property descriptor in the REPORT OPERATIONAL OQ LIST parameter data (see 9.2.11.3);

~~The minimum value is 0 and the maximum value is 2 047.~~

## 4.4 OQ service notification methods

### 4.4.1 OQ service notification methods overview

Table 16 lists the OQ service notification methods that may be used by the PQI host to determine when an entry is available in an OQ.

If MSI-X (see PCI) is enabled, then MSI-X mode is enabled. If MSI-X is not enabled then legacy INTx (see PCI) may be used (i.e., legacy INTx mode is used). If the PQI host masks off the generation of interrupts by the PQI device and uses other methods (e.g., reading OQ PI)  to determine when an event has occurred, then polled mode is being used.

Table 16 — OQ service notification methods

| Mode | PQI device | PQI host |
|---|---|---|
| MSI-X | 4.4.2 | 4.4.5 |
| Legacy INTx | 4.4.3 | 4.4.6 |
| Polled | 4.4.4 | 4.4.7 |

### 4.4.2 Sending interrupts in MSI-X mode

#### 4.4.2.1 Sending interrupts in MSI-X mode overview

A memory write transaction used to send an MSI-X interrupt should use the same Traffic Class (see PCIe) as used to write the IU to the ~~operational~~ OQ and to update the OQ PI for the ~~operational~~ OQ and shall not be sent until any update to the OQ PI that caused the generation of the MSI-X interrupt has been sent.

### 4.4.2.2 Sending interrupts for an administrator OQ

In MSI-X mode, the PQI device interrupt ~~events~~ shall be sent each time after the administrator OQ ~~CI~~PI is written with a new value. If the administrator OQ is mapped to the same MSI-X interrupt vector as one or more operational OQs in a PQI device, then interrupt coalescing for the MSI-X interrupt associated with that MSI-X interrupt vector is vendor specific.

### 4.4.2.3 Sending interrupts for an operational OQ

In MSI-X mode, the PQI device generates an MSI-X interrupt (i.e., performs a memory write transaction using an address and data specified in the MSI-X Table entry associated with the interrupt source) (see PCI and PCIe) every time an interrupt event (see table 17) occurs.

In MSI-X mode, the PQI device interrupt events may occur when an operational OQ contains one or more ~~valid~~occupied elements.

A PQI device implements a coalescing timer for each operational OQ. If reset, a coalescing timer is set to a value of zero. If started, the coalescing timer increases as time elapses. If stopped, the coalescing timer retains its current value.

For each operational OQ, the PQI device maintains the following attributes related to the generation of MSI-X interrupts:

a) Minimum Coalescing Time attribute (see 4.3.6.11.3);
b) Maximum Coalescing Time attribute (see 4.3.6.11.4);
c) Coalescing Count attribute (see 4.3.6.11.5);
d) Wait For Rearm attribute (see 4.3.6.11.6); and
e) Interrupt Message Number attribute (see 4.3.6.11.7).

The Minimum Coalescing Time attribute, Maximum Coalescing Time attribute, Coalescing Count attribute, Wait For Rearm attribute, and Interrupt Message Number attribute are created by the CREATE OPERATIONAL OQ function.

The Minimum Coalescing Time attribute, Maximum Coalescing Time attribute, Coalescing Count attribute, and Wait For Rearm attribute may be modified by the CHANGE OPERATIONAL OQ PROPERTIES function.

For an operational OQ:

~~valid~~occupied element count = (n + OQ PI – OQ CI) mod n

where:

a) ~~valid~~occupied element count is the number of ~~valid~~occupied elements in the OQ; and
b) n is the number of elements in the OQ.

A PQI device determines the ~~valid~~occupied element count based on OQ CI and its ~~local copy of~~ OQ PI local copy.

For an operational OQ an interrupt event occurs if:

a) the condition shown in table 17 is true; and
b) the associated event in table 17 occurs.

If an interrupt event occurs, then the PQI device should send the MSI-X interrupt designated for that operational OQ. If more than one event in table 17 occurs at the same time for an operational OQ, then the PQI device shall only generate one MSI-X interrupt.

Table 17 — Interrupt generation conditions and events

| Condition | Associated event |
|---|---|
| The ~~valid~~occupied element count is greater than or equal to the Coalescing Count attribute and Minimum Coalescing Time attribute is non-zero. | The coalescing timer equals the Minimum Coalescing Time attribute. |
| The ~~valid~~occupied element count is greater than or equal to one and Maximum Coalescing Time attribute is non-zero. | The coalescing timer equals the Maximum Coalescing Time attribute. |
| The coalescing timer is greater than or equal to the Minimum Coalescing Time attribute. | PQI device writes to the operational OQ PI causing the ~~valid~~occupied element count to be greater than or equal to the Coalescing Count attribute. |

The point in time at which a PQI device sends the interrupt is vendor specific (e.g., it may be before the MSI-X memory write transaction occurs on the PCI Express interface).

After sending an interrupt for an operational OQ:

a) if the Wait For Rearm attribute is set to zero, then the PQI device shall reset and start the coalescing timer; or
b) if the Wait For Rearm attribute is set to one, then the PQI device shall reset and stop the coalescing timer.

If:

a) a memory write transaction to the ~~dword containing the operational OQ CI~~operational OQ CI dword is received with the REARM INTERRUPT bit set to one~~ in the OQ PI dword~~ (see 6.1.3); and
b) the Wait For Rearm attribute is set to one,

then the PQI device shall reset and start the coalescing timer.

If more than one operational OQ is mapped to a MSI-X interrupt vector in a PQI device, then interrupt coalescing for the MSI-X interrupt associated with that MSI-X interrupt vector is vendor specific.

Figure 22 shows an example of interrupt coalescing with the WAIT FOR REARM bit set to zero.

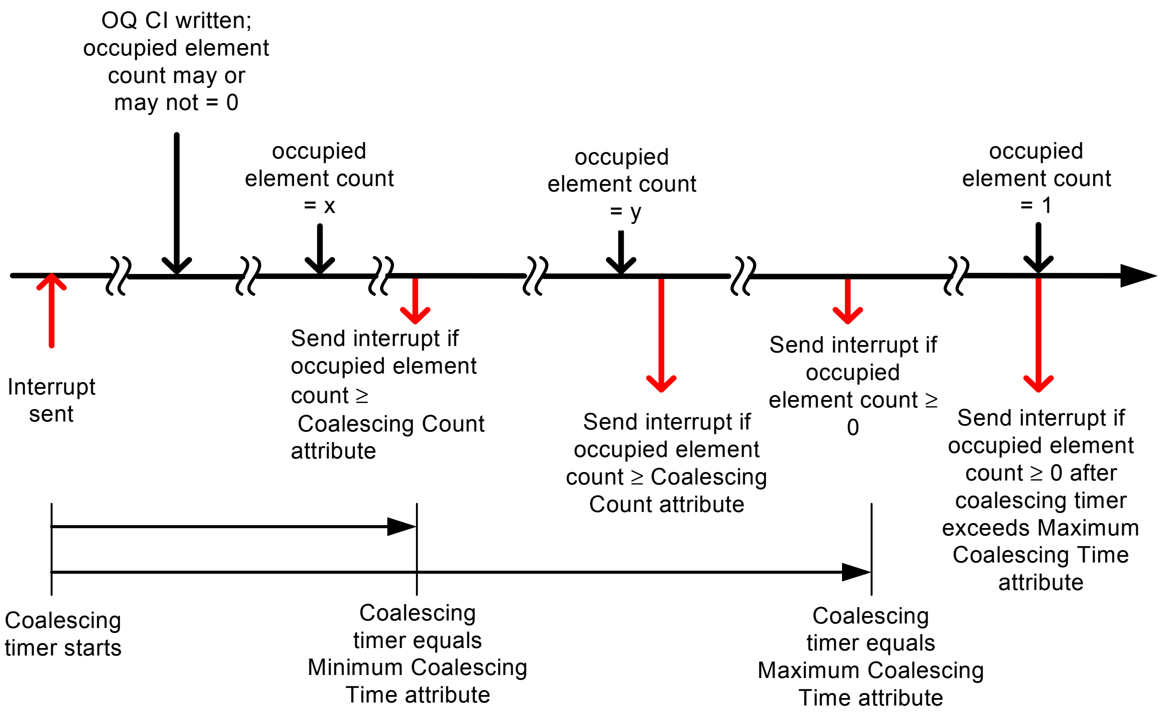

OQ CI written; occupied element count may or may not = 0
occupied element count = x
occupied element count = y
occupied element count = 1
Interrupt sent
Send interrupt if occupied element count ≥ Coalescing Count attribute
Send interrupt if occupied element count ≥ Coalescing Count attribute
Send interrupt if occupied element count ≥ 0
Send interrupt if occupied element count ≥ 0 after coalescing timer exceeds Maximum Coalescing Time attribute
Coalescing timer starts
Coalescing timer equals Minimum Coalescing Time attribute
Coalescing timer equals Maximum Coalescing Time attribute

**Figure 22 — Interrupt coalescing example with the WAIT FOR REARM bit set to zero**

Figure 23 shows an example of interrupt coalescing with the WAIT FOR REARM bit set to one.

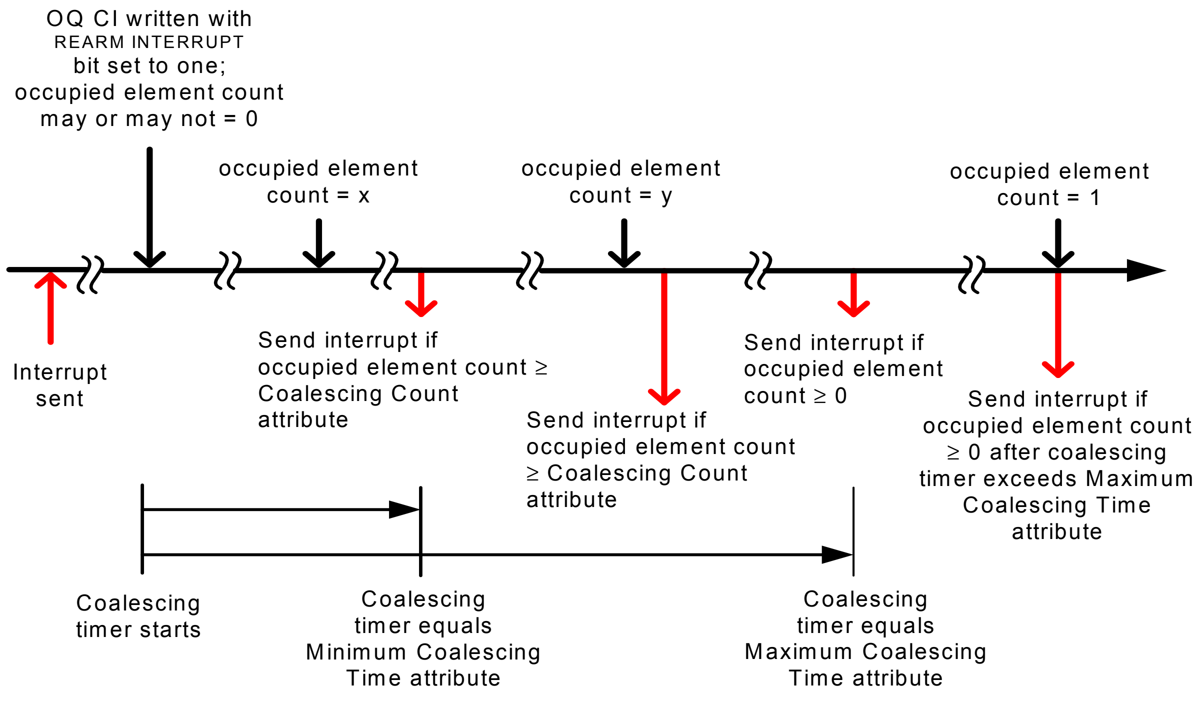

OQ CI written with REARM INTERRUPT bit set to one; occupied element count may or may not = 0
occupied element count = x
occupied element count = y
occupied element count = 1
Interrupt sent
Send interrupt if occupied element count ≥ Coalescing Count attribute
Send interrupt if occupied element count ≥ Coalescing Count attribute
Send interrupt if occupied element count ≥ 0
Send interrupt if occupied element count ≥ 0 after coalescing timer exceeds Maximum Coalescing Time attribute
Coalescing timer starts
Coalescing timer equals Minimum Coalescing Time attribute
Coalescing timer equals Maximum Coalescing Time attribute

**Figure 23 — Interrupt coalescing example with the WAIT FOR REARM bit set to one**

### 4.4.3 Sending interrupts in legacy INTx mode

In legacy INTx mode, the PQI device sends Assert_INTx and Deassert_INTx Messages (see PCIe) to maintain a virtual wire that reflects whether interrupt sources are asserted or deasserted (i.e., level triggered).

In legacy INTx mode, any OQ containing one or more ~~valid~~occupied elements is an interrupt source.

Figure 24 shows the legacy INTx interrupt sources and masks.

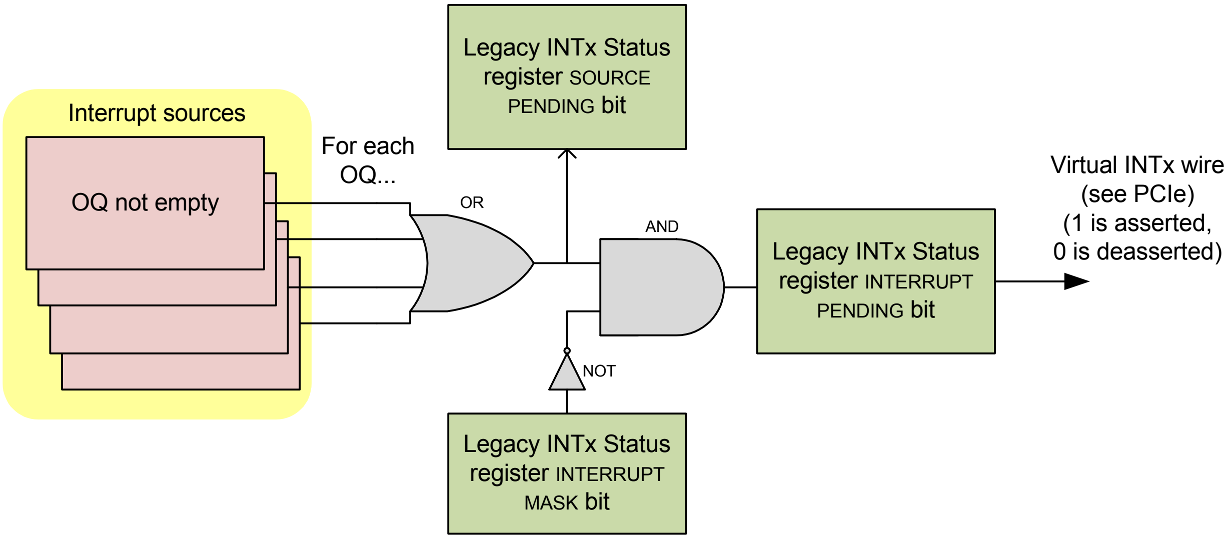

Interrupt sources
OQ not empty
For each OQ...
OR
Legacy INTx Status register SOURCE PENDING bit
AND
Legacy INTx Status register INTERRUPT PENDING bit
Virtual INTx wire (see PCIe) (1 is asserted, 0 is deasserted)
NOT
Legacy INTx Status register INTERRUPT MASK bit

**Figure 24 — Legacy INTx sources and masks**

The PQI device shall assert the virtual INTx wire whenever:

a) any of its interrupt sources is asserted; and
b) the Legacy INTx Interrupt Mask register INTERRUPT MASK bit is set to zero.

The PQI device shall deassert the virtual INTx wire whenever:

a) all of its interrupt sources are not asserted; or
b) the Legacy INTx Interrupt Mask register INTERRUPT MASK bit is set to one.

The OQ interrupt source is cleared when the OQ CI equals the OQ PI (i.e., OQ is empty).

### 4.4.4 Sending interrupts in polled mode

In polled mode, the PQI device shall not send interrupts.

### 4.4.5 Servicing interrupts in MSI-X mode

In MSI-X mode, if the WAIT FOR REARM bit (see 4.4.2 and 9.2.5.1) is set to one, then after reading one or more elements from the OQ, the PQI host should set the REARM INTERRUPT bit (see 6.1.3 and 9.2.5.1) to one before exiting the interrupt service routine (e.g., after consuming all ~~valid~~occupied elements in the OQ). The interrupt coalescing timer shall be reset and started regardless of whether or not the OQ is empty.

If the PQI host receives an interrupt assigned to more than one OQ, then the PQI host should check the OQ PI of ~~all OQs~~each OQ assigned to that interrupt.

### 4.4.6 Servicing interrupts in legacy INTx mode

In legacy INTx mode, if the PQI host receives an interrupt, then the PQI host should:

1) if the legacy INTx interrupt is shared with other PCI functions (see PCI), then read the Legacy INTx Interrupt Status register (see 5.2.7); and

NOTE 18 - The read from the Legacy INTx Status register ensures that all memory writes from the PQI device have completed.

2) if the Legacy INTx Interrupt Status register INTERRUPT PENDING bit is set to one (i.e., the PQI device is a source of the interrupt), then determine the PQI device's interrupt source(s) by checking the OQ PI of each OQ to determine if the OQ is not empty.

### 4.4.7 Handling the lack of interrupts in polled mode

In polled mode, the PQI host should poll the OQ PI of each OQ to determine if the OQ is not empty.

## 4.5 PD (PQI device) state machine

### 4.5.1 PD (PQI device) state machine overview

The PD state machine describes the PQI device states and transitions for PCI configuration and device operation.

This state machine consists of the following states:

a) PD0:Power_On_And_Reset (see 4.5.2) (initial state);
b) PD1:~~Configuration_Space_Ready~~ PQI_Status_Available (see 4.5.3);
c) PD2:~~Register_Ready~~ All_Registers_Ready (see 4.5.4);
d) PD3:Administrator_Queue_Pair_Ready (see 4.5.5); and
e) PD4:Error (see 4.5.6).

The PD state machine shall start in the PD0:Power_On_And_Reset state after power on.

The PQI device state is indicated by the PQI DEVICE STATE field in the PQI Device Status register (see 5.2.10). The PQI device shall set the PQI DEVICE STATE field (see table 29) to the appropriate value upon entry into each state.

~~This state machine shall maintain a RegisterLock state machine variable to determine whether the administrator queue pair is established. The RegisterLock state machine variable is set to LOCKED while the administrator queue pair is established. The RegisterLock state machine variable is set to UNLOCKED while the administrator queue pair is not established. At power on, the initial value of the RegisterLock state machine variable shall be set to UNLOCKED.~~

~~If the RegisterLock state machine variable is set to LOCKED, then memory writes to the following registers shall be ignored:~~

a) ~~the Administrator IQ Element Array Address register (see 5.2.13);~~
b) ~~the Administrator OQ Element Array Address register (see 5.2.14);~~
c) ~~the Administrator IQ CI Address register (see 5.2.15);~~
d) ~~the Administrator OQ PI Address register (see 5.2.16); and~~
e) ~~the Administrator Queue Parameter register (see 5.2.17).~~

Figure 25 describes the PD state machine.

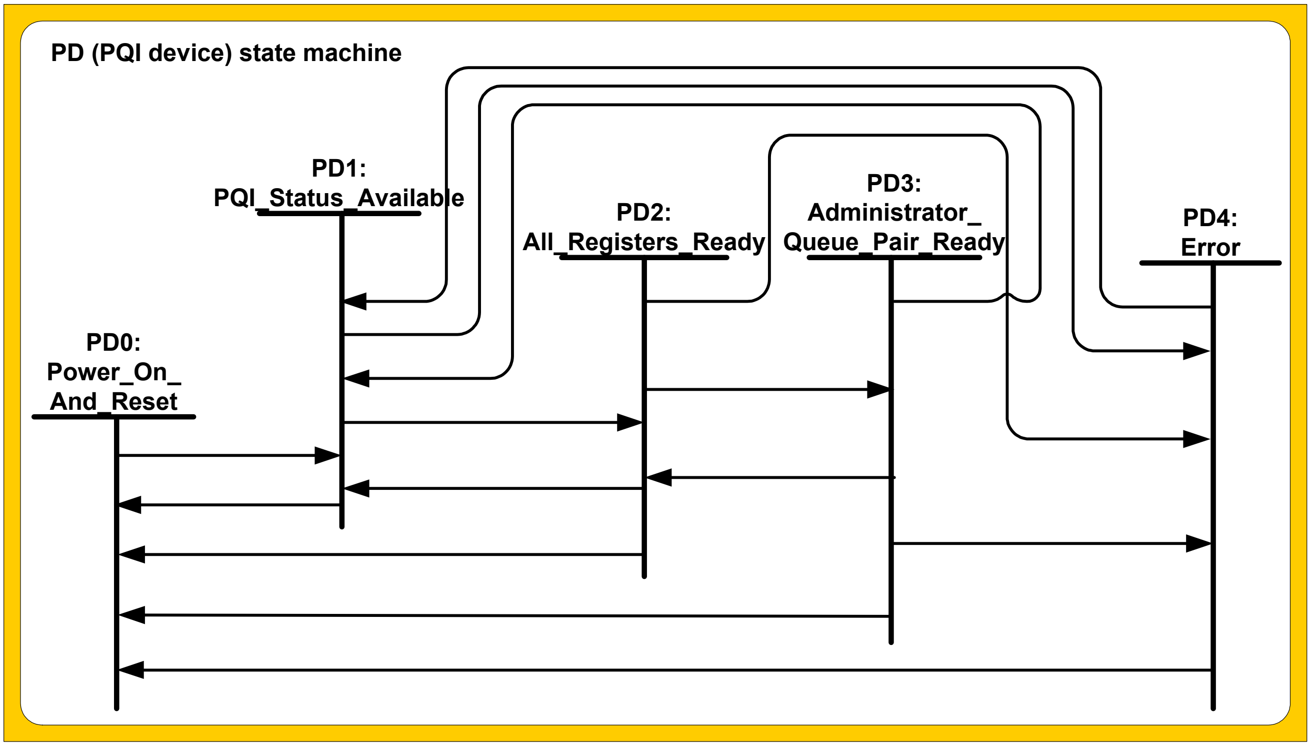

PD (PQI device) state machine
PD0: Power_On_And_Reset
PD1: PQI_Status_Available
PD2: All_Registers_Ready
PD3: Administrator_Queue_Pair_Ready
PD4: Error

**Figure 25 — PD (PQI device) state machine**

### 4.5.2 PD0:Power_On_And_Reset state

#### 4.5.2.1 PD0:Power_On_And_Reset state description

When in this state the PQI device is not initialized and is in the PCI Function D0 uninitialized state (see PCI PM).

The PQI device shall enter this state upon:

a) power on; or
b) PCI Express reset (see PCIe).

While in this state, the PQI device shall perform internal initialization (e.g., POST).

#### 4.5.2.2 Transition PD0:Power_On_And_Reset to the PD1:~~Configuration_Space_Ready~~ PQI_Status_Available

This transition shall occur after the PQI device:

a) reaches the D0 active power management state (see PCI PM) with the Memory Space Enable bit set to one in the Command register (see PCI).
a) ~~completes internal initialization; and~~
b) ~~is able to accept PCI configuration transactions.~~

If this state did not start POST, then this transition shall include a Post Not Running argument.

### 4.5.3 PD1:~~Configuration_Space_Ready~~ PQI_Status_Available state

#### 4.5.3.1 PD1:~~Configuration_Space_Ready~~ PQI_Status_Available state description

When in this state the PQI device is initialized by system software (see PCI) and the device is in the PCI Function D0 active state (see PCI PM).

Upon entry to this state PQI device initialization begins.

~~If this state was entered from PD0:Power_On_And_Reset state with a Post Not Running argument, then the PQI device may start POST.~~

While in this state:

a) if an error occurs during PQI device initialization, then the PQI device shall ~~indicate the error~~set the error code to ERROR DETECTED DURING INITIALIZATION in the PQI Device Error register (see 5.2.18);
b) all queues are deleted and their corresponding objects are set to their uninitialized state; and
c) register attributes (see PCIe) are defined in table 19.
d) ~~if POST is performed and an error occurs, then the PQI device shall set the ERROR DETECTED DURING POST error code in the PQI Device Error register;~~
e) ~~the PQI host may read the PQI defined PCIe registers; and~~
f) ~~only the Reset register (see 5.2.20) shall be writable and memory writes to other PQI defined registers shall be ignored by the PQI device.~~

#### 4.5.3.2 Transition PD1:~~Configuration_Space_Ready~~ PQI_Status_Available state to the PD0:Power_On_And_Reset

This transition shall occur if:

a) the PQI device detects a PCI Express reset.

#### 4.5.3.3 Transition PD1:~~Configuration_Space_Ready~~ PQI_Status_Available to the PD2:~~Register_Ready~~ All_Registers_Ready

This transition shall occur if the HOLD IN PD1 bit is set to zero in the PQI Device Reset register (see 5.2.20) and after:

a) PQI device initialization is complete;
b) POST, if any, is complete; and
c) the PQI device is ready to process ~~administrator functions~~PD functions.

#### 4.5.3.4 Transition PD1:~~Configuration_Space_Ready~~ PQI_Status_Available state to the PD4:Error

This transition shall occur if:

a) an error is detected during PQI initialization.~~; or~~
b) ~~an error is detected during POST.~~

### 4.5.4 PD2:~~Register_Ready~~ All_Registers_Ready state

#### 4.5.4.1 PD2:~~Register_Ready~~ All_Registers_Ready state description

When in this state the PQI device memory space is configured and is ready for PQI device standard registers. No circular queues are configured for PQI operations. The PQI device may be in any of the following PCI Function Power Management States (see PCI PM):

a) PCI Function D0 active state;
b) PCI Function D1 state; or
c) PCI Function D2 state;

~~This state writes to writable PQI defined registers that are enabled~~Register attributes (see PCIe) are defined in table 19.

While in this state the PQI host may create the administrator queue.

~~After the administrator queue pair is created, the PQI device shall set the RegisterLock state machine variable to LOCKED~~.

#### 4.5.4.2 Transition PD2:~~Register_Ready~~ All_Registers_Ready to the PD0:Power_On_And_Reset

This transition shall occur if:

a) the PQI device detects a PCI Express reset.

#### 4.5.4.3 Transition PD2:~~Register_Ready~~ All_Registers_Ready to the PD1:~~Configuration_Space_Ready~~ PQI_Status_Available

This transition shall occur if:

a) the PQI device detects a PQI reset.

#### 4.5.4.4 Transition PD2:~~Register_Ready~~ All_Registers_Ready to the PD3:Administrator_Queue_Pair_Ready

This transition shall occur if:

a) the FUNCTION AND STATUS CODE field is set to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register; and
b) the administrator queue pair has been created.

#### 4.5.4.5 Transition PD2:~~Register_Ready~~ All_Registers_Ready to the PD4:Error

This transition shall occur if the PQI device detects:

a) an invalid parameter used in PQI registers;
b) an administrator queue pair create error;
c) an administrator queue pair delete error; or
d) an internal error (e.g., device error).

### 4.5.5 PD3:Administrator_Queue_Pair_Ready state

#### 4.5.5.1 PD3:Administrator_Queue_Pair_Ready state description

When in this state the PQI device is initialized and ready to process any PCI transaction (see PCI PM). The PQI device may be in any of the following PCI Function Power Management States (see PCI PM):

a) PCI Function D0 active state;
b) PCI Function D1 state;
c) PCI Function D2 state;
d) PCI Function D3$_{hot}$ state; or
e) PCI Function D3$_{cold}$ state).

While in this state~~, the PQI device shall process~~:

a) the PQI device shall process the Administrator Queue Configuration Function register function (e.g., DELETE ADMINISTRATOR QUEUE PAIR);
b) the PQI device shall process administrator IUs using the administrator queue pair; and
c) if the operational IQ and the operational OQ have been created, then the PQI device shall process operational IUs using the operational IQ and the operational OQ.

~~If the administrator queue pair is deleted, then the PQI device shall:~~

a) ~~set the RegisterLock state machine variable to UNLOCKED.~~

#### 4.5.5.2 Transition PD3:Administrator_Queue_Pair_Ready to the PD0:Power_On_And_Reset

This transition shall occur if:

a) the PQI device detects a PCI Express reset.

#### 4.5.5.3 Transition PD3:Administrator_Queue_Pair_Ready to the PD1:~~Configuration_Space_Ready~~ PQI_Status_Available

This transition shall occur if:

a) the PQI device detects a PQI reset.

#### 4.5.5.4 Transition PD3:Administrator_Queue_Pair_Ready to the PD2:~~Register_Ready~~ All_Registers_Ready

This transition shall occur ~~if~~after:

a) the PQI device successfully completes deleting the administrator queue pair using the DELETE ADMINISTRATOR QUEUE PAIR PD function; and
b) the PQI device sets the FUNCTION AND STATUS CODE field to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register.
a) ~~the FUNCTION AND STATUS CODE field is set to 00h (i.e., IDLE) in the Administrator Queue Configuration Function register; and~~
b) ~~the administrator queue pair is deleted.~~

#### 4.5.5.5 Transition PD3:Administrator_Queue_Pair_Ready to the PD4:Error

This transition shall occur if the PQI device detects:

a) ~~an invalid parameter used in PQI registers~~ a reserved code value used in FUNCTION AND STATUS CODE field in the Administrator Queue Configuration Function register (see 5.2.5);
b) an invalid value in the Administrator Queue Parameter register (see 5.2.17);
c) an administrator queue pair delete error;
d) an internal error (e.g., device error);
e) an invalid IU TYPE field (see 8.2) in an inbound IU;
f) an invalid IU LENGTH field (see 8.2) in an inbound IU; or
g) an invalid RESPONSE OQ ID field (see 8.3) in an inbound IU.

### 4.5.6 PD4:Error state

#### 4.5.6.1 PD4:Error state description

The PQI device may be in any of the PCI Function Power Management States.

Upon entry to this state:

a) ~~the PQI DEVICE STATE field is set to 4h (i.e., PD4:Error state) in the PQI Device Status register;~~
a) the PQI Device Error register indicates the error (see 4.6);
b) all outstanding IU operations are aborted; and
c) the administrator queue pair and the operational queues are in an unknown state.

If an internal error occurs, then the PQI device shall set the error code to INTERNAL ERROR in the PQI Device Error register (see 4.6).

#### 4.5.6.2 Transition PD4:Error to the PD0:Power_On_And_Reset

This transition shall occur if:

a) the PQI device detects a PCI Express reset.

#### 4.5.6.3 Transition PD4:Error to the PD1:~~Configuration_Space_Ready~~ PQI_Status_Available

This transition shall occur if:

a) the PQI device detects a PQI reset.

## 4.6 Register based error information

Register based error information reports error information to the PQI host that is not able to be reported using an administrator outbound IU (i.e., the administrator OQ is not available or the administrator OQ is corrupted).

The register based error information is indicated by the ERROR CODE field, the ERROR CODE QUALIFIER field, the BYTE POINTER field, and the BIT POINTER field in the PQI Device Error register (see 5.2.18). If more than one error is detected, then the PQI device may report any of the errors.

The ERROR ~~DATA~~DETAILS REGISTER VALID bit in the PQI Device Error register indicates if valid error-specific data is available in the PQI Device Error ~~Data~~Details register (see 5.2.19).

Table 18 defines the register based error information.

**Table 18 — Register based error information structure** (part 1 of 2)

| ERROR CODE **field** | ERROR CODE QUALIFIER **field** | **Name** | BYTE POINTER **field and** BIT POINTER **field** | **PQI Device Error Details register contents** | **Reference** |
|---|---|---|---|---|---|
| 00h | 00h | NO ERROR | Invalid | Vendor-specific | 5.2.18 |
| | 01h to FFh | Reserved | | | |
| 01h | 00h | ERROR DETECTED DURING INITIALIZATION ~~POST~~ ~~[a]~~ | Invalid | Vendor-specific | 4.5.3.1 |
| | 01h to FFh | Reserved | | | |
| 02h | 00h | Reserved | | | |
| | ~~00~~01h | INVALID PD FUNCTION ~~PARAMETER IN PQI REGISTER~~ ~~[a]~~ | ~~Valid~~Invalid | Vendor-specific | 5.2.5 |
| | 02h | INVALID PARAMETER FOR PD FUNCTION | Valid | Vendor-specific | 5.2.17 |
| | ~~01~~03h to FFh | Reserved | | | |
| 03h | 00h | ~~ADMINISTRATOR QUEUE PAIR CREATE ERROR~~ ERROR CREATING ADMINISTRATOR QUEUE PAIR ~~[a]~~ | ~~Valid~~Invalid | Vendor-specific | 4.3.3.2 |
| | 01h | ~~ADMINISTRATOR QUEUE PAIR DELETE ERROR~~ERROR DELETING ADMINISTRATOR QUEUE PAIR ~~[a]~~ | ~~Valid~~Invalid | Vendor-specific | 4.3.4.2 |
| | 02h to FFh | Reserved | | | |

[a] ~~The PQI Device Error Data register contains vendor-specific information.~~

**Table 18 — Register based error information structure** (part 2 of 2)

| ERROR CODE **field** | ERROR CODE QUALIFIER **field** | **Name** | BYTE POINTER **field and** BIT POINTER **field** | **PQI Device Error Details register contents** | **Reference** |
|---|---|---|---|---|---|
| 04h | 00h | Reserved | | | |
| | 01h | INVALID IU TYPE IN ADMINISTRATOR REQUEST IU ~~[a]~~ | Invalid | Vendor-specific | 9.1.1 |
| | 02h | INVALID IU LENGTH IN ADMINISTRATOR REQUEST IU ~~[a]~~ | Invalid | Vendor-specific | 9.1.1 |
| | 03h | INVALID OQ ID IN ADMINISTRATOR REQUEST IU ~~[a]~~ | Invalid | Vendor-specific | 9.1.3 |
| | 04h to FFh | Reserved | | | |
| 05h | 00h | INTERNAL ERROR ~~[a]~~ | Invalid | Vendor-specific | ~~4.5.4.5 and 4.5.5.5~~ 4.5.6 |
| | 01h to FFh | Reserved | | | |
| 06h | 00h | Reserved | | | |
| | 01h | ERROR COMPLETING PQI SOFT RESET ~~[a]~~ | ~~Valid~~Invalid | Vendor-specific | ~~4.7.1,~~ 4.7.2 ~~and 5.2.20~~ |
| | 02h | ERROR COMPLETING PQI FIRM RESET ~~[a]~~ | ~~Valid~~Invalid | Vendor-specific | ~~4.7.1,~~ 4.7.3 ~~and 5.2.20~~ |
| | 03h | ERROR COMPLETING PQI HARD RESET ~~[a]~~ | ~~Valid~~Invalid | Vendor-specific | ~~4.7.1,~~ 4.7.4 ~~and 5.2.20~~ |
| | 04h to FFh | Reserved | | | |
| 07h to 7Fh | 00h to FFh | Reserved | | | |
| 80h to FFh | 00h to FFh | Vendor specific | | | |

[a] ~~The PQI Device Error Data register contains vendor-specific information.~~

## 4.7 PQI reset

### 4.7.1 PQI reset overview

PQI soft reset (see 4.7.2), PQI firm reset (see 4.7.3), and PQI hard reset (see 4.7.4) allow the PQI host to reset the PQI device without accessing the configuration space (see 5.1) of the PQI device.

The configuration space (see 5.1) of all PQI devices in the PCI Express device shall not be affected by any PQI reset. By preserving the configuration space, the PQI host may be able to resume communication with this PQI device without the need for the system software (see PCI) to re-scan the PCI bus to determine what PCI devices are actually present following the completion of the PQI soft reset.

The PQI host may specify that the PQI device remain in PD1:~~Configuration_Space_Ready~~PQI_Status_Available state (see 4.5.3) following a PQI reset (see 5.2.20).

The PQI host initiates a PQI reset using the following steps:

1) write to the PQI Device Reset register (see 5.2.20) with the RESET ACTION field set to 001b (i.e., ~~START~~ RESET), with a specific reset type in the RESET TYPE field, and the specific option in the HOLD IN PD1 bit;
2) wait for at least 100 ms;
3) repeatedly read the PQI Device Reset register (see 5.2.20) until:
    A) the RESET ACTION field is set to 010b (i.e., ~~START~~ RESET COMPLETED); or
    B) the timeout value indicated by the MAXIMUM TIMEOUT FOR PQI DEVICE RESET field in the PQI Device Capability register (see 5.2.6) has expired after step 2);
4) if the RESET ACTION field is not set to 010b (i.e., ~~START~~ RESET COMPLETED) ~~and the value indicated by the MAXIMUM TIMEOUT FOR RESET field in the Capability register (see 5.2.6) has expired~~, then read the PQI Device Status register (see 5.2.10) and report the error in a vendor specific manner; and
5) if the RESET ACTION field is set to 010b (i.e., ~~START~~ RESET COMPLETED), then the PQI reset is completed ~~successful~~without error, and:
    A) if the HOLD IN PD1 bit was set to zero in PQI Device Reset register in step 1), then the PD state machine ~~is allowed to transition~~transitions from PD1:~~Configuration_Space_Ready~~PQI_Status_Available to the PD2:~~Register_Ready~~All_Registers_Ready (see 4.5.4); and
    B) if the HOLD IN PD1 bit in PQI Device Reset register was set to one in step 1), then the PD state machine remains in the PD1:PQI_Status_Available. ~~is in the PD1:Configuration_Space_Ready state (i.e., prevent state transition from PD1:Configuration_Space_Ready to PD2:Register_Ready (see 4.5.4)).~~

To transition the PQI device out of the PD1:~~Configuration_Space_Ready~~PQI_Status_Available state when the HOLD IN PD1 bit ~~is set to 1b~~ in PQI Device Reset register was set to 1b on the previous PQI reset, the PQI host initiates the following steps:

1) write to the PQI Device Reset register (see 5.2.20) with the RESET ACTION field set to 001b (i.e., ~~START~~ RESET), with the RESET TYPE field set to 000b (i.e.,NO RESET) and the HOLD IN PD1 bit set to zero;
2) wait for at least 100 ms;
3) repeatedly read the PQI Device Reset register (see 5.2.20) until:
    A) the RESET ACTION field is set to 010b (i.e., ~~START~~ RESET COMPLETED); or
    B) the value indicated by the MAXIMUM TIMEOUT FOR PQI DEVICE RESET field in the PQI Device Capability register (see 5.2.6) has expired;
4) if the RESET ACTION field is not set to 010b (i.e., ~~START~~ RESET COMPLETED) ~~and the value indicated by the MAXIMUM TIMEOUT FOR RESET field in the Capability register (see 5.2.6) has expired~~, then read the PQI Device Status register (see 5.2.10), the PQI Device Error register (see 5.2.10), and the PQI Device Error ~~Data~~Details register (see 5.2.19), and report the error in a vendor specific manner; and
5) if the RESET ACTION field is set to 010b (i.e., ~~START~~ RESET COMPLETED), then the PQI reset is completed ~~successfully~~without error and the PD state machine ~~is allowed to transition~~transitions from PD1:~~Configuration_Space_Ready~~PQI_Status_Available to the PD2:~~Register_Ready~~All_Registers_Ready).

### 4.7.2 PQI soft reset

A PQI soft reset ~~shall~~ resets:

a) the PQI device standard registers for this PQI device (see 5.2.2);
b) the queuing layer for this PQI device (see 4.3 and clause 6); ~~and~~
c) the information unit layer (see clause 8 and SOP) content from only this PQI device;~~.~~
d) the administrator queue pair, if any (see 4.3.4.2), for this PQI device;
e) operational IQs, if any, and operational OQs, if any (see 4.3.4.3), for this PQI device; and
f) the PD state machine (see 4.5) for this PQI device.

If the PQI device fails to complete PQI soft reset, then the PQI device shall:

a) set the error code to ERROR COMPLETING PQI SOFT RESET in the PQI Device Error register (see 5.2.18).~~; and~~
b) ~~set the BYTE POINTER field to 0h, the BIT POINTER field to 00b, and the BPV bit to zero in the PQI Device Error register (see 5.2.18).~~

### 4.7.3 PQI firm reset

A PQI firm reset ~~shall~~ resets:

a) the PQI device registers for this PQI device (see 5.2);
b) the queuing layer for this PQI device ~~(see clause 6)~~(see 4.3 and clause 6); ~~and~~
c) the information unit layer ~~(see clause 8)~~(see clause 8 and SOP) content from all PQI devices in the PCI Express device;
d) the administrator queue pair, if any (see 4.3.4.2), for this PQI device;
e) operational IQs, if any, and operational OQs, if any (see 4.3.4.3), for this PQI device; and
f) the PD state machine (see 4.5) for this PQI device.

If the PQI device fails to complete PQI firm reset, then the PQI device shall:

a) set the error code to ERROR COMPLETING PQI FIRM RESET in the PQI Device Error register (see 5.2.18).~~; and~~
b) ~~set the BYTE POINTER field to 0h, the BIT POINTER field to 00b, and the BPV bit to zero in the PQI Device Error register (see 5.2.18).~~

### 4.7.4 PQI hard reset

A PQI hard reset shall reset:

a) the PQI device registers (see 5.2) for all PQI devices in the PCI Express device;
b) the queuing layer ~~(see clause 6)~~(see 4.3 and clause 6) for all PQI devices in the PCI Express device; ~~and~~
c) the information unit layer ~~(see clause 8)~~(see clause 8 and SOP) content from all PQI devices in the PCI Express device;
d) the administrator queue pair, if any (see 4.3.4.2), for all PQI devices in the PCI Express device;
e) operational IQs, if any, and operational OQs, if any (see 4.3.4.3), for all PQI devices in the PCI Express device; and
f) the PD state machine (see 4.5) for all PQI devices in the PCI Express device.

If the PQI device fails to complete PQI hard reset, then the PQI device shall:

a) set the error code to ERROR COMPLETING PQI HARD RESET in the PQI Device Error register (see 5.2.18)~~;~~.~~ and~~
b) ~~set the BYTE POINTER field to 0h, the BIT POINTER field to 00b, and the BPV bit to zero in the PQI Device Error register (see 5.2.18)~~

## 4.8 Data field requirements

### 4.8.1 ASCII data field requirements

ASCII data fields shall contain only ASCII printable characters (i.e., code values 20h to 7Eh) and may be terminated with one or more ASCII null (00h) characters.

ASCII data fields described as being left-aligned shall have any unused bytes at the end of the field (i.e., highest offset) and the unused bytes shall be filled with ASCII space characters (20h).

ASCII data fields described as being right-aligned shall have any unused bytes at the start of the field (i.e., lowest offset) and the unused bytes shall be filled with ASCII space characters (20h).

### 4.8.2 Null data field termination and zero padding requirements

A data field that is described as being null-terminated shall have one byte containing an ASCII or UTF-8 null (00h) character in the last used byte (i.e., highest offset for big endian and lowest offset for little endian) of the field and no other bytes in the field shall contain the ASCII/UTF-8 null character.

A data field may be specified to be a fixed length. The length specified for a data field may be greater than the length required to contain the contents of the field. A data field may be specified to have a length that is a multiple of a given value (e.g., a multiple of four bytes). When such fields are described as being null-padded, the bytes at the end of the field that are not needed to contain the field data shall contain ASCII or UTF-8 null (00h) characters. When such fields are described as being zero-padded, the bytes at the end of the field that are not needed to contain the field data shall contain zeros.

NOTE 19 - There is no difference between the pad byte contents in null-padded and zero-padded fields. The difference is in the format of the other bytes in the field.

A data field that is described as being both null-terminated and null-padded shall have at least one byte containing an ASCII or UTF-8 null (00h) character in the end of the field (i.e., highest offset for big endian and lowest offset for little endian) and may have more than one byte containing ASCII or UTF-8 null characters, if needed, to meet the specified field length requirements. If more than one byte in a null-terminated, null-padded field contains the ASCII or UTF-8 null character, then all the bytes containing the ASCII or UTF-8 null character shall be at the end of the field (i.e., only the highest offsets).

# 5 PCI Express requirements and PQI device registers ~~PCI Express architecture usage by this standard~~

## 5.1 PCI Express requirements

This clause describes information associated with PCIe configuration to support this standard.

A PQI device is a kind of PCI function (see PCI).

A PQI device shall:

a) be a PCI Express Endpoint (see PCIe) or a Root Complex Integrated Endpoint (see PCIe);
b) support the Advanced Error Reporting Extended Capability (see PCIe);
c) not require the use of Virtual Channels (see PCIe) or Traffic Classes (see PCIe) beyond the default VC0/TC0 (see PCIe); ~~and~~
d) support MSI-X (see PCIe); and
e) use a PCI class code assigned for an information unit layer protocol over PQI (see PCI-ID).

~~A PQI device shall set the Prefetchable bit (see PCIe) in its first memory BAR unless the PQI device has device-specific registers mapped to that region and those registers:~~

a) ~~have read side effects; or~~
b) ~~do not support write merging (see PCIe).~~

A PQI device that has device-specific registers mapped to its first memory BAR that:

a) have read side effects; or
b) do not tolerate write merging (see PCIe),

shall set the Prefetchable bit (see PCIe) to zero in its first memory BAR.

A PQI device that has device-specific registers mapped to its first memory BAR that:

a) do not have read side effects; and
b) tolerate write merging (see PCIe),

shall set the Prefetchable bit (see PCIe) to one in its first memory BAR.

A PQI device shall set its first memory BAR to at least 512 bytes in size, and should set it to at least 4 KiB in size.

> NOTE 20 - The 512 bytes size requirement allows for the PQI device registers plus at least one administrator queue pair. The 4 KiB size recommendation matches typical CPU page sizes.

A PQI device may support Advanced Error Reporting Multiple Header Recording (see PCIe).

If a PQI device is a function in a PCI Multi-Function device that contains more than eight PCI functions (see PCI), then the PCI Multi-Function device shall support the Alternative Routing-ID Interpretation (ARI) (see PCIe) (i.e., be an ARI Device, support the ARI Capability, and not support Phantom Functions).

A PQI device shall support the Power Budgeting Extended Capability (see PCIe).

A PQI device shall support 32-bit memory writes and 64-bit memory writes to PQI device registers, IQ PI dwords, and OQ CI dwords. 64-bit PQI device registers may be accessed by 32-bit memory writes in any order (e.g., offset 0 followed by offset 4, or offset 4 followed by offset 0). Handling of 8-bit memory writes and the 16-bit memory writes is outside the scope of this standard.

A PQI device shall support zero-length reads (see PCIe), 8-bit memory reads, 16-bit memory reads, 32-bit memory reads, and 64-bit memory reads from PQI device registers, IQ PI dwords, and OQ CI dwords.

## 5.2 PQI device ~~registers~~ memory space

### 5.2.1 PQI device ~~registers~~ memory space overview

PQI device memory space is the memory space described by the PQI device's first PCI memory BAR.

Figure 26 shows the PQI device memory space.

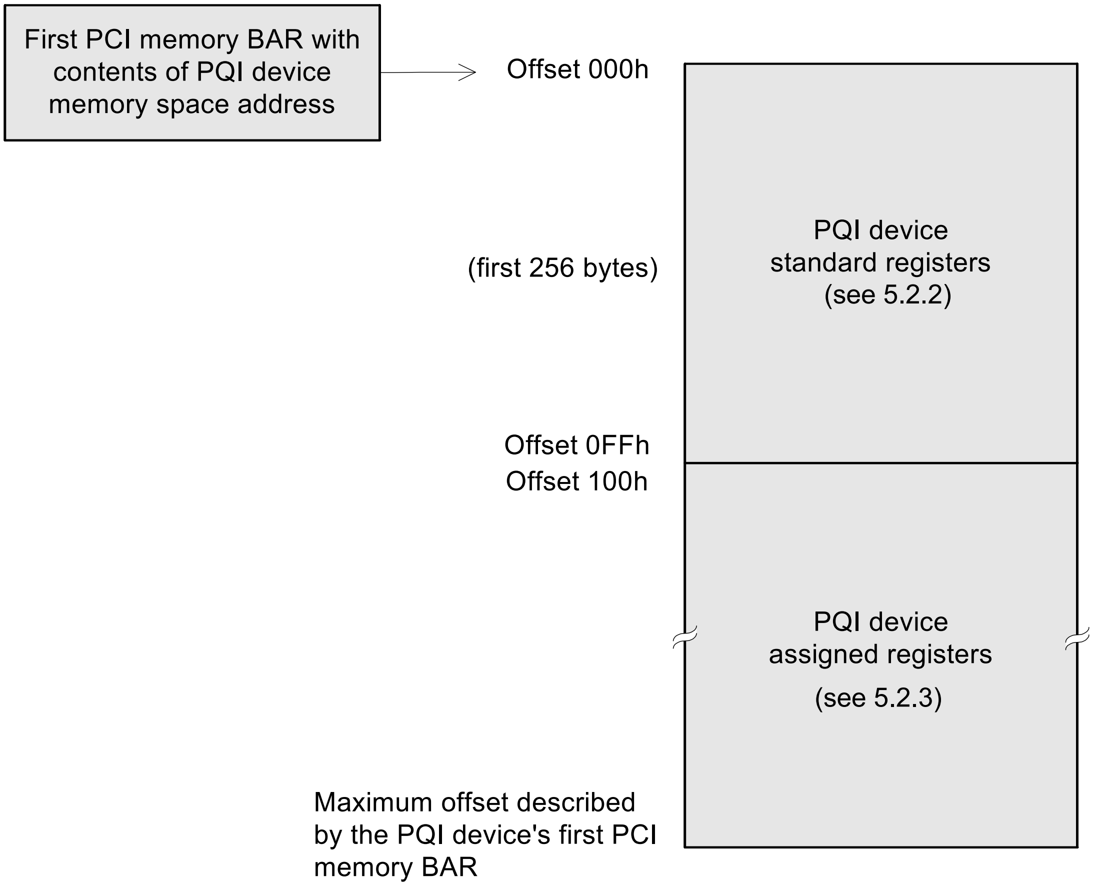

First PCI memory BAR with contents of PQI device memory space address
Offset 000h
(first 256 bytes)
PQI device standard registers (see 5.2.2)
Offset 0FFh
Offset 100h
PQI device assigned registers
(see 5.2.3)
Maximum offset described by the PQI device's first PCI memory BAR

**Figure 26 — PQI device memory space**

The first 256 bytes of the PQI device memory space contain PQI device standard registers as defined in 5.2.2.

The bytes after the first 256 bytes of ~~the memory space described by the first PCI memory BAR~~PQI device memory space are defined in 5.2.3.

### 5.2.2 PQI device standard registers

Table 19 defines the PQI device registers from PQI device memory space offsets 000h through 0FFh (i.e., the first 256 bytes).

**Table 19 — PQI device standard registers from offset 000h to offset 0FFh contained in PQI device memory space** (part 1 of 2)

| Offset | Size (bytes) | Register name | Type | Register Attribute (see PCIe) [a] based on PD state machine state | | | | Reference |
|---|---|---|---|---|---|---|---|---|
| | | | | PD1 | PD2 | PD3 | PD4 | |
| 000h | 8 | PQI Device Signature | M | RO | RO | RO | RO | 5.2.4 |
| 008h | 8 | Administrator Queue Configuration Function | M | RO | RW | RW | RO | 5.2.5 |
| 010h | 8 | PQI Device Capability | M | RO | RO | RO | RO | 5.2.6 |
| 018h | 4 | Legacy INTx Interrupt Status | M | RO | RO | RO | RO | 5.2.7 |
| 01Ch | 4 | Legacy INTx Interrupt Mask Set | M | RO | RW | RW | RO | 5.2.8 |
| 020h | 4 | Legacy INTx Interrupt Mask Clear | M | RO | RW | RW | RO | 5.2.9 |
| 024h | 28 | RsvdZ | | | | | | |
| 040h | 4 | PQI Device Status | M | RO | RO | RO | RO | 5.2.10 |
| 044h | 4 | RsvdZ | | | | | | |
| 048h | 8 | Administrator IQ PI Offset | M | RO | RO | RO | RO | 5.2.11 |
| 050h | 8 | Administrator OQ CI Offset | M | RO | RO | RO | RO | 5.2.12 |
| 058h | 8 | Administrator IQ Element Array Address | M | RO | RW | RO | RO | 5.2.13 |
| 060h | 8 | Administrator OQ Element Array Address | M | RO | RW | RO | RO | 5.2.14 |
| 068h | 8 | Administrator IQ CI Address | M | RO | RW | RO | RO | 5.2.15 |

Key:

M = Register implementation is mandatory.
O = Register implementation is optional.
R~~/~~W = Memory reads and memory writes are supported (see PCIe).
RO = Memory read are supported, and memory writes are ignored (see PCIe).
~~R/WL = Memory reads are supported, memory writes are supported while the administrator queue pair is not established, and memory writes are ignored while the administrator queue pair is established.~~

[a] Reading any register defined in this table shall not cause any side effects (e.g., change any register value).

**Table 19 — PQI device standard registers from offset 000h to offset 0FFh contained in PQI device memory space** (part 2 of 2)

| Offset | Size (bytes) | Register name | Type | Register Attribute (see PCIe) [a] based on PD state machine state | | | | Reference |
|---|---|---|---|---|---|---|---|---|
| | | | | PD1 | PD2 | PD3 | PD4 | |
| 070h | 8 | Administrator OQ PI Address | M | RO | RW | RO | RO | 5.2.16 |
| 078h | 4 | Administrator Queue Parameter | M | RO | RW | RO | RO | 5.2.17 |
| 080h | 4 | PQI Device Error | M | RO | RO | RO | RO | 5.2.18 |
| 084h | 4 | RsvdZ | | | | | | |
| 088h | 8 | PQI Device Error ~~Data~~Details | M | RO | RO | RO | RO | 5.2.19 |
| 090h | 4 | PQI Device Reset | M | RW | RW | RW | RW | 5.2.20 |
| 094h | 4 | Power Action | M | RO | R/W | R/W | RO | 5.2.21 |
| 098h | 104 | RsvdZ | | | | | | |

Key:

M = Register implementation is mandatory.
O = Register implementation is optional.
R~~/~~W = Memory reads and memory writes are supported (see PCIe).
RO = Memory read are supported, and memory writes are ignored (see PCIe).
~~R/WL = Memory reads are supported, memory writes are supported while the administrator queue pair is not established, and memory writes are ignored while the administrator queue pair is established.~~

[a] Reading any register defined in this table shall not cause any side effects (e.g., change any register value).

Reads to reserved bits within defined registers shall return zeros. Writes to RsvdZ bits within defined registers shall be ignored.

Writes of reserved values to defined fields within defined registers shall be ignored.

### 5.2.3 PQI device assigned registers

The PQI device memory space offset range from 100h to the maximum offset described by the first PCI memory BAR (see figure 26):

a) shall contain ~~dwords containing IQ PIs~~IQ PI dwords (see 6.1.2), if any;
b) shall contain ~~dwords containing OQ CIs~~OQ CI dwords (see 6.1.3), if any;
c) may contain MSI-X Table entries;
d) may contain MSI-X PBA entries; and
e) may contain vendor specific registers and vendor specific memory space.

### 5.2.4 PQI Device Signature register

The PQI Device Signature register contains a fixed value.

Table 20 defines the PQI Device Signature register.

**Table 20 — PQI Device Signature register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | SIGNATURE ('PQI DREG') | | | | | | | |
| 7 | | | | | | | | (LSB) |

The SIGNATURE field contains 8 bytes of ASCII data set as shown in table 20 for the PQI Device Signature register (i.e., 5051_4920_4452_4547h).

The PQI host may read this register to determine that the PQI device is present.

The PQI Device Signature register value after power on, PCI Express reset, and PQI reset (see 4.7) shall be set to 'PQI DREG'.

### 5.2.5 Administrator Queue Configuration Function register

The Administrator Queue Configuration Function register is used to ~~create or delete the administrator queue pair~~perform a PD function and to indicate the status of a PD function.

~~The Administrator Queue Configuration Function register value after power on, PCI Express, and PQI reset (see 4.7) shall be set to 00000000_00000000h.~~

The PQI device shall set to zero the fields defined in the Administrator Queue Configuration Function register after power on, PCI Express, or PQI reset (see 4.7).

Table 21 defines the Administrator Queue Configuration Function register.

**Table 21 — Administrator Queue Configuration Function register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | FUNCTION AND STATUS CODE | | | | | | | |
| 1 | RsvdZ | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |

The FUNCTION AND STATUS CODE field is used to specify the PD function to perform and to indicate the status of a PD function being performed.

Table 22 defines the FUNCTION AND STATUS CODE field for memory reads.

**Table 22 — FUNCTION AND STATUS CODE field for memory reads**

| Code | Name | Description |
|---|---|---|
| 00h | IDLE | The PQI device is not processing ~~an administrator~~ a PD function. |
| 01h | CREATING ADMINISTRATOR QUEUE PAIR | The PQI device is in the process of creating the administrator queue pair (see 4.3.3.2). |
| 02h | DELETING ADMINISTRATOR QUEUE PAIR | The PQI device is in the process of deleting the administrator queue pair (see 4.3.4.2). |
| All others | Reserved | |

Table 23 defines the ~~definition of~~ FUNCTION AND STATUS CODE field for memory writes.

**Table 23 — FUNCTION AND STATUS CODE field for memory writes**

| Code | Name | Description |
|---|---|---|
| 00h | NOP | Shall be ignored by the PQI device. |
| 01h | CREATE ADMINISTRATOR QUEUE PAIR | The PQI device shall create the administrator queue pair (see 4.3.3.2). |
| 02h | DELETE ADMINISTRATOR QUEUE PAIR | The PQI device shall delete the administrator queue pair (see 4.3.4.2). |
| All others [a] | Reserved | |

[a] ~~If the PQI host writes a reserved code value to the FUNCTION AND STATUS CODE field, then the PQI device shall set the PQI DEVICE STATE field to 4h (i.e., PD4:Error state) in the PQI Device Status register (see 5.2.10) and set error code to INVALID PARAMETER IN PQI REGISTERin the PQI Device Error register (see 5.2.18).~~ If the PQI host writes a reserved code value to the FUNCTION AND STATUS CODE field, then the PQI device transitions the PD state machine to the PD4:Error state (see 4.5.5.5) and shall set the error code to INVALID PD FUNCTION in the PQI Device Error register (see 5.2.18).

### 5.2.6 PQI Device Capability register

The PQI Device Capability register indicates the capabilities of the PQI device.

Table 24 defines the PQI Device Capability register.

**Table 24 — PQI Device Capability register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | MAXIMUM ADMINISTRATOR IQ ELEMENTS | | | | | | | |
| 1 | MAXIMUM ADMINISTRATOR OQ ELEMENTS | | | | | | | |
| 2 | ADMINISTRATOR IQ ELEMENT LENGTH | | | | | | | |
| 3 | ADMINISTRATOR OQ ELEMENT LENGTH | | | | | | | |
| 4 | MAXIMUM TIMEOUT FOR PQI DEVICE RESET | | | | | | | (LSB) |
| 5 | (MSB) | | | | | | | |
| 6 | RsvdZ | | | | | | | |
| 7 | | | | | | | | |

The MAXIMUM ADMINISTRATOR IQ ELEMENTS field indicates the maximum number of administrator IQ elements (see 4.3.6.2.2) supported by the PQI device. The PQI device shall support at least two administrator IQ elements.

The MAXIMUM ADMINISTRATOR OQ ELEMENTS field indicates the maximum number of administrator OQ elements (see 4.3.6.2.2) supported by the PQI device. The PQI device shall support at least two administrator OQ elements.

The ADMINISTRATOR IQ ELEMENT LENGTH field indicates the ~~element length in 16-byte increments of the administrator IQ~~ administrator IQ element length (see 4.3.6.2.3) in 16-byte increments (e.g., 01h means 16 bytes and FFh means 4 080 bytes). The length shall be greater than or equal to the size of the largest administrator inbound IU supported by the PQI device.

The ADMINISTRATOR OQ ELEMENT LENGTH field indicates the ~~element length in 16-byte increments of the administrator OQ~~administrator OQ element length (see 4.3.6.2.3) in 16-byte increments (e.g., 01h means 16 bytes and FFh means 4 080 bytes). The length shall be greater than or equal to the size of the largest administrator outbound IU supported by the PQI device.

The MAXIMUM TIMEOUT FOR PQI DEVICE RESET field indicates the timeout value in 100 ms~~seconds~~ for the PQI device to complete a PQI reset (see 4.7).

~~The Capability register shall be set to the value described by this subclause after power on, PCI Express reset, and PQI reset (see 4.7).~~

The PQI device shall set the PQI Device Capability register to the value described by this subclause after power on, PCI Express reset, and PQI reset (see 4.7).

### 5.2.7 Legacy INTx Interrupt Status register

~~When servicing a legacy INTx interrupt, the PQI host should read this register before reading the OQ element array or OQ PI to ensure that memory writes from the PQI device have completed.~~

~~The Legacy INTx Status register shall be set to 00000000h after power on, PCI Express reset, and PQI reset (see 4.7).~~

Table 25 defines the Legacy INTx Interrupt Status register.

**Table 25 — Legacy INTx Interrupt Status register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RsvdZ | | | | | SOURCE PENDING | INTERRUPT MASK | INTERRUPT PENDING |
| 1 | RsvdZ | | | | | | | |
| ••• | | | | | | | | |
| 3 | | | | | | | | |

A SOURCE PENDING bit set to one indicates that one or more interrupt sources are asserted. A SOURCE PENDING bit set to zero indicates that no interrupt source is asserted.

An INTERRUPT MASK bit set to one indicates that the virtual INTx wire mask is enabled. An INTERRUPT MASK bit set to zero indicates that the virtual INTx wire mask is disabled.

An INTERRUPT PENDING bit set to one indicates that the virtual INTx wire is asserted by the PQI device. An INTERRUPT PENDING bit set to zero indicates that the virtual INTx wire is deasserted by the PQI device.

The SOURCE PENDING bit and the INTERRUPT PENDING bit are each set to zero after power on, PCI Express, or PQI reset (see 4.7) as a result of all OQs being deleted. The INTERRUPT MASK bit shall be set to zero after power on, PCI Express, or PQI reset (see 4.7).

### 5.2.8 Legacy INTx Interrupt Mask Set register

~~The Legacy INTx Interrupt Mask Set register after power on, PCI Express reset, and PQI reset (see 4.7) shall be set to 00000000h.~~

The PQI device shall set the fields defined in the Legacy INTx Interrupt Mask Set register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 26 defines the Legacy INTx Interrupt Mask Set register.

**Table 26 — Legacy INTx Interrupt Mask Set register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RsvdZ | | | | | | | INTERRUPT MASK SET |
| 1 | RsvdZ | | | | | | | |
| ••• | | | | | | | | |
| 3 | | | | | | | | |

For memory writes:

a) an INTERRUPT MASK SET bit set to one specifies that the legacy INTx interrupt shall be masked (i.e., the virtual INTx wire (see PCIe) shall not be asserted by the PQI device); and
b) an INTERRUPT MASK SET bit set to zero shall be ignored.

For memory reads, the INTERRUPT MASK SET bit is the same as the Legacy INTx~~X~~ Interrupt Status register INTERRUPT MASK bit (see 5.2.7).

### 5.2.9 Legacy INTx Interrupt Mask Clear register

Table 27 defines the Legacy INTx Interrupt Mask Clear register.

Table 27 — Legacy INTx Interrupt Mask Clear register

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RsvdZ | | | | | | | INTERRUPT MASK CLEAR |
| 1 | RsvdZ | | | | | | | |
| ••• | | | | | | | | |
| 3 | | | | | | | | |

For memory writes:

a) an INTERRUPT MASK CLEAR bit set to one specifies that the legacy INTx interrupt shall be unmasked (i.e., the virtual INTx wire may be asserted by the PQI device); and
b) an INTERRUPT MASK CLEAR bit set to zero shall be ignored.

For memory reads, the INTERRUPT MASK CLEAR bit is the same as the INTERRUPT MASK bit in Legacy INTx~~X~~ Interrupt Status register (see 5.2.7).

~~The Legacy INTx Interrupt Mask Clear register shall be set to 00000000h after power on, PCI Express reset, and PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Legacy INTx Interrupt Mask Clear register to zero after power on, PCI Express, or PQI reset (see 4.7).

### 5.2.10 PQI Device Status register

The PQI Device Status register indicates the status of the PQI device.

Table 28 defines the PQI Device Status register.

**Table 28 — PQI Device Status register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RsvdZ | | | | PQI DEVICE STATE | | | |
| 1 | RsvdZ | | | | | | OP IQ ERROR | OP OQ ERROR |
| 2 | RsvdZ | | | | | | | |
| 3 | | | | | | | | |

Table 29 defines the PQI DEVICE STATE field.

**Table 29 — PQI DEVICE STATE field**

| Code | PD state machine state [a] |
|---|---|
| 0h | PD0:Power_On_And_Reset |
| 1h | PD1:~~Configuration_Space_Ready~~ PQI_Status_Available |
| 2h | PD2:~~Register_Ready~~ All_Registers_Ready |
| 3h | PD3:Administrator_Queue_Pair_Ready |
| 4h | PD4:Error |
| All others | Reserved |

[a] Refer to section 4.5.1 for the PD state machine

~~An OP IQ ERROR bit set to one indicates that one or more of the operational IQs are in error. An OP IQ ERROR bit set to zero indicates that all of the operational IQs are operating normally.~~ An OP IQ ERROR bit set to one indicates that the PQI device has stopped consuming from one or more operational IQs due to an error. An OP IQ ERROR bit set to zero indicates that the PQI device has not stopped consuming from one or more operational IQs due to an error. If all of the operational IQs are in error and have been deleted (see 4.3.4.3), then the PQI device shall set the OP IQ ERROR bit to zero.

~~An OP OQ ERROR bit set to one indicates that one or more of the operational OQs are in error. An OP OQ ERROR bit set to zero indicates that all of the operational OQs are operating normally.~~ An OP OQ ERROR bit set to one indicates that the PQI device has stopped producing to one or more operational OQs due to an error. An OP OQ ERROR bit set to zero indicates that the PQI device has not stopped producing to one or more operational OQs due to an error. If all of the operational OQs are in error and have deleted (see 4.3.4.3), then the PQI device shall set the OP OQ ERROR bit to zero.

The OP IQ ERROR bit and OP OQ ERROR bit shall each be set to zero after power on, PCI Express reset, or PQI reset (see 4.7).

~~The PQI DEVICE STATE field shall be set to 0h after power on and PCI Express reset.~~

The PQI device shall set the PQI DEVICE STATE field to 0h (i.e., PD0:Power_On_And_Reset) after power on and PCI Express reset.

~~The PQI DEVICE STATE field shall be set to 1h after PQI reset (see 4.7).~~

The PQI device shall set the PQI DEVICE STATE field to 1h (i.e., PD1:PQI_Status_Available) after PQI reset completion (see 4.7).

~~The OP IQ ERROR bit and OP OQ ERROR bit shall each be set to zero after power on, PCI Express reset, or PQI reset (see 4.7).~~

### 5.2.11 Administrator IQ PI Offset register

~~The Administrator IQ PI Offset register is used during administrator queue pair creation (see 4.3.3.2).~~

The Administrator IQ PI Offset register is updated during administrator queue pair creation (see 4.3.3.2) and during administrator queue pair deletion (see 4.3.4.2).

~~The Administrator IQ PI Offset register shall be set to 00000000_00000000h after power on, PCI Express reset, and PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Administrator IQ PI Offset register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 30 defines the Administrator IQ PI Offset register.

**Table 30 — Administrator IQ PI Offset register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | (LSB) |
| ••• | | | | ADMINISTRATOR IQ PI OFFSET | | | | |
| 7 | (MSB) | | | | | | | |

~~The ADMINISTRATOR IQ PI OFFSET field indicates the offset in PQI device memory space of the administrator IQ PI (see 5.2.1) (i.e., the administrator IQ PI address is the address contained in the first PCI memory BAR plus the content of the ADMINISTRATOR IQ PI OFFSET field). Byte 0 bit 0 and byte 0 bit 1 shall each be set to zero.~~

The ADMINISTRATOR IQ PI OFFSET field indicates the 64-bit offset in PQI device memory space (see 5.2.1) of the administrator IQ PI (i.e., the administrator IQ PI address is the address contained in the first PCI memory BAR plus the administrator IQ PI offset). The lower two bits of the administrator IQ PI offset are zero.

### 5.2.12 Administrator OQ CI Offset register

~~The Administrator OQ CI Offset register is used during administrator queue pair creation (see 4.3.3.2).~~

The Administrator OQ CI Offset register is updated during administrator queue pair creation (see 4.3.3.2) and during administrator queue pair deletion (see 4.3.4.2).

~~The Administrator OQ CI Offset register shall each be set to 00000000_00000000h after power on, PCI Express reset, or PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Administrator OQ CI Offset register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 31 defines the Administrator OQ CI Offset register.

**Table 31 — Administrator OQ CI Offset register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | (LSB) |
| ••• | ADMINISTRATOR OQ CI OFFSET | | | | | | | |
| 7 | (MSB) | | | | | | | |

~~The ADMINISTRATOR OQ CI OFFSET field indicates the offset in PQI device memory space (see 5.2.1) of the administrator OQ CI (i.e., the administrator OQ CI address is the address contained in the first PCI memory BAR plus the content of the ADMINISTRATOR OQ CI OFFSET field). Byte 0 bit 0 and byte 0 bit 1 shall each be set to zero.~~

The ADMINISTRATOR OQ CI OFFSET field indicates the 64-bit offset in PQI device memory space (see 5.2.1) of the administrator OQ CI (i.e., the administrator OQ CI address is the address contained in the first PCI memory BAR plus the administrator OQ CI offset). The lower two bits of the administrator OQ CI offset are zero.

### 5.2.13 Administrator IQ Element Array Address register

~~The Administrator IQ Element Array Address register is used during administrator queue pair creation (see 4.3.3.2).~~

The Administrator IQ Element Array Address register is updated during administrator queue pair creation (see 4.3.3.2) and during administrator queue pair deletion (see 4.3.4.2).

~~The Administrator IQ Element Array Address register shall each be set to 00000000_00000000h after power on, PCI Express reset, or PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Administrator IQ Element Array Address register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 32 defines the Administrator IQ Element Array Address register.

**Table 32 — Administrator IQ Element Array Address register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | (LSB) | RsvdZ | | | | | |
| ••• | ADMINISTRATOR IQ ELEMENT ARRAY ADDRESS | | | | | | | |
| 7 | (MSB) | | | | | | | |

~~The ADMINISTRATOR IQ ELEMENT ARRAY ADDRESS field specifies the upper 58 bits of the administrator IQ element array address (see 5.2.1).~~

The ADMINISTRATOR IQ ELEMENT ARRAY ADDRESS field specifies the upper 58 bits of the 64-bit administrator IQ element array address. The lower bits of the administrator IQ element array address are zero.

~~If the RegisterLock state machine variable (see 4.5.1) is set to LOCKED, then memory writes to this register shall be ignored.~~ Register attributes (see PCIe) are defined in table 19

### 5.2.14 Administrator OQ Element Array Address register

~~The Administrator OQ Element Array Address register is used during administrator queue pair creation (see 4.3.3.2).~~

The Administrator OQ Element Array Address register is updated during administrator queue pair creation (see 4.3.3.2) and during administrator queue pair deletion (see 4.3.4.2).

~~The Administrator OQ Element Array Address register shall each be set to 00000000_00000000h after power on, PCI Express reset, or PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Administrator OQ Element Array Address register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 33 defines the Administrator OQ Element Array Address register.

**Table 33 — Administrator OQ Element Array Address register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 |  | (LSB) | RsvdZ |  |  |  |  |  |
| ••• | ADMINISTRATOR OQ ELEMENT ARRAY ADDRESS |  |  |  |  |  |  |  |
| 7 | (MSB) |  |  |  |  |  |  |  |

~~The ADMINISTRATOR OQ ELEMENT ARRAY ADDRESS field specifies the upper 58 bits of the administrator OQ element array address.~~

The ADMINISTRATOR OQ ELEMENT ARRAY ADDRESS field specifies the upper 58 bits of the 64-bit administrator OQ element array address (see 5.2.1). The lower bits of the administrator OQ element array address are zero.

~~If the RegisterLock state machine variable (see 4.5.1) is set to LOCKED, then memory writes to this register shall be ignored.~~ Register attributes (see PCIe) are defined in table 19.

### 5.2.15 Administrator IQ CI Address register

~~The Administrator IQ CI Address register is used during administrator queue pair creation (see 4.3.3.2).~~

The Administrator IQ CI Address register is updated during administrator queue pair creation (see 4.3.3.2) and during administrator queue pair deletion (see 4.3.4.2).

~~The Administrator IQ CI Address register shall be set to 00000000_00000000h after power on, PCI Express reset, PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Administrator IQ CI Address register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 34 defines the Administrator IQ CI Address register.

**Table 34 — Administrator IQ CI Address register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 |  | (LSB) | RsvdZ |  |  |  |  |  |
| ••• | ADMINISTRATOR IQ CI ADDRESS |  |  |  |  |  |  |  |
| 7 | (MSB) |  |  |  |  |  |  |  |

~~The ADMINISTRATOR IQ CI ADDRESS field specifies the upper 58 bits of the administrator IQ CI address.~~

The ADMINISTRATOR IQ CI ADDRESS field specifies the upper 58 bits of the 64-bit administrator IQ CI address. The lower bits of the administrator IQ CI address are zero.

~~If the RegisterLock state machine variable (see 4.5.1) is set to LOCKED, then memory writes to this register shall be ignored.~~ Register attributes (see PCIe) are defined in table 19.

### 5.2.16 Administrator OQ PI Address register

~~The Administrator OQ PI Address register is used during administrator queue pair creation (see 4.3.3.2).~~

The Administrator OQ PI Address register is updated during administrator queue pair creation (see 4.3.3.2) and during administrator queue pair deletion (see 4.3.4.2).

~~The Administrator OQ PI Address register shall each be set to 00000000_00000000h after power on, PCI Express reset, or PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Administrator OQ PI Address register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 35 defines the Administrator OQ PI Address register.

**Table 35 — Administrator OQ PI Address register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | | (LSB) | RsvdZ | | | | | |
| ••• | ADMINISTRATOR OQ PI ADDRESS | | | | | | | |
| 7 | (MSB) | | | | | | | |

~~The ADMINISTRATOR OQ PI ADDRESS field specifies the upper 58 bits of the administrator OQ PI address.~~

The ADMINISTRATOR OQ PI ADDRESS field specifies the upper 58 bits of the 64-bit administrator OQ PI address. The lower bits of the administrator OQ PI address are zero.

~~If the RegisterLock state machine variable (see 4.5.1) is set to LOCKED, then memory writes to this register shall be ignored.~~The register attributes (see PCIe) are defined in table 19.

### 5.2.17 Administrator Queue Parameter register

~~The Administrator Queue Parameter register is used during administrator queue pair creation (see 4.3.3.2).~~

The Administrator Queue Parameter register is updated during administrator queue pair creation (see 4.3.3.2) and during administrator queue pair deletion (see 4.3.4.2).

~~The Administrator Queue Parameter register shall be set to 00000000h after power on, PCI Express, or PQI reset (see 4.7).~~

The PQI device shall set the fields defined in the Administrator Queue Parameter register to zero after power on, PCI Express, or PQI reset (see 4.7).

Table 36 defines the Administrator Queue Parameter register.

**Table 36 — Administrator Queue Parameter register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | NUMBER OF ADMINISTRATOR IQ ELEMENTS | | | | | | | |
| 1 | NUMBER OF ADMINISTRATOR OQ ELEMENTS | | | | | | | |
| 2 | INTERRUPT MESSAGE NUMBER | | | | | | | (LSB) |
| 3 | RsvdZ | | | | | (MSB) | | |

The NUMBER OF ADMINISTRATOR IQ ELEMENTS field specifies the number of elements in the administrator IQ. This field shall be set to:

a) a value less than or equal to the value contained in the MAXIMUM ADMINISTRATOR IQ ELEMENTS field in the PQI Device Capability register (see 5.2.6); and
b) at least 02h.

If an invalid value is set to the NUMBER OF ADMINISTRATOR IQ ELEMENTS field, then the PQI device transitions to the PD4:Error state (see 4.5.5.5) and sets the error code to INVALID PARAMETER FOR PD FUNCTION in the PQI Device Error register (see 5.2.18).

The NUMBER OF ADMINISTRATOR OQ ELEMENTS field specifies the number of elements in the administrator OQ. This field shall be set to:

a) a value less than or equal to the value contained in the MAXIMUM ADMINISTRATOR OQ ELEMENTS field in the PQI Device Capability register (see 5.2.6); and
b) at least 02h.

If an invalid value is set to the NUMBER OF ADMINISTRATOR OQ ELEMENTS field, then the PQI device transitions to the PD4:Error state (see 4.5.5.5) and sets the error code to INVALID PARAMETER FOR PD FUNCTION in the PQI Device Error register (see 5.2.18).

The INTERRUPT MESSAGE NUMBER field specifies the MSI-X Table entry used to generate the interrupt message for ~~OQ PI updates to this OQ~~updates to the administrator OQ PI if MSI-X is enabled.

If the INTERRUPT MESSAGE NUMBER field is larger than the MSI-X Table, then the PQI device transitions to the PD4:Error state (see 4.5.5.5) and sets the error code to INVALID PARAMETER FOR PD FUNCTION in the PQI Device Error register (see 5.2.18).

~~If the RegisterLock state machine variable (see 4.5.1) is set to LOCKED, then memory writes to this register shall be ignored.~~ Register attributes (see PCIe) are defined in table 19.

### 5.2.18 PQI Device Error register

The PQI Device Error register indicates the cause or the source of an error (see 4.6) in the PQI device. The PQI Device Error register value is valid while the PD state machine is in PD4:Error state (see 4.5.6).

Table 37 defines the PQI Device Error register.

**Table 37 — PQI Device Error register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ERROR CODE | | | | | | | |
| 1 | ERROR CODE QUALIFIER | | | | | | | |
| 2 | BYTE POINTER | | | | | | | |
| 3 | ERROR DETAILS REGISTER VALID | BIT POINTER | | | RsvdZ | | | |
| ~~3~~ | ~~ERROR DATA VALID~~ | ~~BPV~~ | ~~BIT POINTER~~ | | | ~~RsvdZ~~ | | |

The ERROR CODE field indicates the generic information describing a reported condition (see table 18).

The ERROR CODE QUALIFIER field indicates further information (see table 18) related to the condition reported in the ERROR CODE field.

~~Table 18 (see 4.6) defines the content of the ERROR CODE field and the ERROR CODE QUALIFIER field.~~

If defined as valid for the error code (see table 18), the ~~The~~ BYTE POINTER field indicates the byte offset in the memory space of the PQI device registers described by the first PCI memory BAR that contains the field with the invalid value.

An ERROR ~~DATA~~DETAILS REGISTER VALID bit ~~of zero~~set to zero indicates that there is no valid error-specific data in the PQI Device Error ~~Data~~Details register (see 5.2.19). An ERROR ~~DATA~~DETAILS REGISTER VALID bit of one indicates that there is valid error-specific data in the PQI Device Error ~~Data~~Details register.

~~A BPV bit set to one indicates that the BIT POINTER field is valid. A BPV (bit pointer valid) bit set to zero indicates that the BIT POINTER field is not valid.~~

~~If the BPV bit is set to one, then the~~The BIT POINTER field indicates the offset in the PQI device register byte of the first bit (i.e., the lowest bit number) containing the field with the invalid value.

~~The ERROR CODE field, the ERROR CODE QUALIFIER field, the BYTE POINTER field, the ERROR DATA VALID bit, the BPV bit, and the BIT POINTER field after:~~

a) ~~power on shall be set to the values that indicate the status of the PQI device after the power on;~~
b) ~~PCI Express reset shall be set to the values that indicate the status of the PQI device after PCI Express reset; or~~
c) ~~PQI reset (see 4.7) shall be set to the values that indicate the status of the PQI device after the PQI reset.~~

The PQI device shall set the ERROR CODE field and the ERROR CODE QUALIFIER field to:

a) NO ERROR after power on;
b) NO ERROR after PCI Express reset; or
c) the error code that indicates the status of the PQI device after PQI reset (see 4.7).

### 5.2.19 PQI Device Error Details ~~Data~~ register

The PQI Device Error ~~Data~~Details register is used to describe error ~~specific data~~details (see 4.6) in combination with the PQI Device Error register (see 5.2.18). If the ERROR ~~DATA~~DETAILS REGISTER VALID bit is set to one in the PQI Device Error register, then the PQI Device Error ~~Data~~Details register contains valid error ~~specific data~~details.

If the ERROR ~~DATA~~DETAILS REGISTER VALID bit is set to zero in the PQI Device Error register, then the PQI Device Error ~~Data~~Details register does not contain error ~~specific data~~details.

Table 38 defines the PQI Device Error ~~Data~~Details register.

**Table 38 — PQI Device Error ~~Data~~Details register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ~~Error specific data~~Error details | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |

Table 18 defines the error ~~specific data~~details.

The PQI Device Error ~~Data~~Details register value along with the PQI Device Error register value ~~after power on or after PCI Express reset~~ shall be set to the values that indicate the error ~~specific data~~details of the PQI device after power on or after PCI Express reset ~~after the power on or after PCI Express reset~~.

The PQI Device Error ~~Data~~Details register value ~~along with the PQI Device Error register value after PQI reset (see 4.7)~~ shall be set to the values that indicate the ~~additional~~ error ~~specific data~~details of the PQI device after PQI reset (see 4.7) ~~after the PQI reset~~.

### 5.2.20 PQI Device Reset register

The PQI Device Reset register is used to initiate a ~~PQI soft reset (see 4.7.2), a PQI firm reset (see 4.7.3), or a PQI hard reset (see 4.7.4)~~PQI reset (see 4.7) and to provide PQI reset status.

Table 39 defines the PQI Device Reset register.

**Table 39 — PQI Device Reset register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RESET ACTION | | | RsvdZ | | RESET TYPE | | |
| 1 | RsvdZ | | | | | | | HOLD IN PD1 |
| 2 | RsvdZ | | | | | | | |
| 3 | | | | | | | | |

Table 40 defines the RESET ACTION field for memory writes.

**Table 40 — RESET ACTION field for memory writes**

| Code | Name | Description |
|---|---|---|
| 000b | NO ACTION | No action. |
| 001b | ~~START~~ RESET | Start processing the PQI reset action specified in the RESET TYPE field. |
| All others | Reserved | |

Table 41 defines the RESET ACTION field for memory reads.

**Table 41 — RESET ACTION field for memory reads**

| Code | Name | Description |
|---|---|---|
| 000b | NO ACTION | No reset action has been processed. |
| 001b | PROCESSING ~~START~~ RESET | The PQI device is processing the ~~start~~ reset action with the reset type specified in the RESET TYPE field. |
| 010b | ~~START~~ RESET COMPLETED | The PQI device has completed the ~~start~~ reset action with the reset type specified in the RESET TYPE field. |
| All others | Reserved | |

The RESET ACTION field shall be set to 000b (i.e., NO ACTION) after power on and after PCI Express reset. After a PQI reset, the ~~The~~ RESET ACTION field shall be set to:

a) 001b (i.e., PROCESSING ~~START~~ RESET) if the PQI reset does not complete normally; or
b) 010b (i.e., ~~START~~ RESET COMPLETED) if the PQI reset completes normally~~after PQI reset~~.

Table 42 defines the RESET TYPE field.

**Table 42 — RESET TYPE field**

| Code | Name | Description |
|---|---|---|
| 000b | NO RESET | No reset action~~.~~ [a] |
| 001b | SOFT RESET | PQI soft reset (see 4.7.2)~~.~~ |
| 010b | FIRM RESET | PQI firm reset (see 4.7.3)~~.~~ |
| 011b | HARD RESET | PQI hard reset (see 4.7.4)~~.~~ |
| All others | Reserved | |

[a] ~~The use of NO RESET action is limited only to allow the PD state machine to transition from PD1:Configuration_Space_Ready to PD2:Register_Ready when a previous PQI reset was done with HOLD IN PD1 bit set to one (see 4.7.1).~~The NO RESET reset type allows the PD state machine to transition from the PD1:PQI_Status_Ready state to the PD2:All_Registers_Ready state after a previous PQI reset was performed with the HOLD IN PD1 bit set to one (see 4.7.1).

The RESET TYPE field shall be set to 000b (i.e., NO RESET) after power on and after PCI Express reset. After a PQI reset, the~~The~~ RESET TYPE field shall be set to:

a) 001b (i.e., SOFT RESET) if the PQI host previously requested a PQI soft reset~~,~~;
b) 010b (i.e., FIRM RESET) if the PQI host previously requested a PQI firm reset~~,~~; or
c) 011b (i.e., HARD RESET) ~~after PQI reset~~if the PQI host previously requested a PQI hard reset.

A HOLD IN PD1 bit set to one specifies the PD state machine shall remain in the PD1:~~Configuration_Space_Ready~~ PQI_Status_Available state (see 4.5.3) following a PQI reset. A HOLD IN PD1 bit set to zero specifies the PD state machine is not held in the PD1:~~Configuration_Space_Ready~~ PQI_Status_Available state (i.e., is allowed ~~state~~to ~~-~~transition from PD1:~~Configuration_Space_Ready~~ PQI_Status_Available to the PD2:~~Register_Ready~~All_Registers_Ready) following a PQI reset.

~~The HOLD IN PD1 bit shall be set to zero after power on and after PCI Express reset. The HOLD IN PD1 field may be set to zero or one after PQI reset.~~

The PQI device shall set the HOLD IN PD1 bit to zero after power on and after PCI Express reset.

The PQI device shall set the HOLD IN PD1 bit to:

a) zero if the PQI host previously requested a PQI reset with the HOLD IN PD1 bit to zero; or
b) one if the PQI host previously requested a PQI reset with the HOLD IN PD1 bit to one.

### 5.2.21 PQI Device Power Action register

The PQI Device Power Action register is used to notify the PQI device about upcoming ~~system state transitions~~operating system-directed power management transitions and device power state transitions.

~~The Power Action register shall be set to 00000000h after power on, PCI Express, or PQI reset (see 4.7).~~

The PQI device shall set the PQI Device Power Action register to 00000000h after power on, PCI Express, or PQI reset (see 4.7).

Table 43 defines the PQI Device Power Action register.

**Table 43 — PQI Device Power Action register**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | POWER ACTION | | SYSTEM POWER ACTION | | | | | |
| 1 | RsvdZ | | | DEVICE POWER ACTION | | | | |
| 2 | RsvdZ | | | | | | | |
| 3 | | | | | | | | |

Table 44 defines the POWER ACTION field for memory writes.

**Table 44 — POWER ACTION field for memory writes**

| Code | Description |
|---|---|
| 00b | No action |
| 01b | Process the power action specified in the SYSTEM POWER ACTION field and the DEVICE POWER ACTION field.<br>If the PQI device receives a write specifying a different power action while processing a previous power action, then ~~it may abandon the previous power action and shall process the new power action~~ the results are vendor specific. |
| All others | Reserved |

Table 45 defines the POWER ACTION field for memory reads.

**Table 45 — POWER ACTION field for memory reads**

| Code | Description |
|---|---|
| 00b | No power action has been processed |
| 01b | The PQI device is processing the power action indicated in the SYSTEM POWER ACTION field and the DEVICE POWER ACTION field |
| 10b | The PQI device has completed processing the power action indicated in the SYSTEM POWER ACTION field and the DEVICE POWER ACTION field |
| 11b | Reserved |

Table 46 defines the SYSTEM POWER ACTION field.

**Table 46 — SYSTEM POWER ACTION field**

| Code | Description |
|---|---|
| General notifications (00h to 0Fh) | |
| 00h | No system power action notification (e.g., system remains in ACPI G0 (S0) (see ACPI)) |
| 01h | Operating system is going to shutdown and reboot (e.g., system remains in ACPI G0 (S0)) |
| 02h | Operating system is going to either:<br>a) shutdown and reboot (e.g., system remains in ACPI G0 (S0)); or<br>b) shutdown and enter ACPI G2 (see ACPI) (i.e., soft off) |
| 03h to 0Fh | Reserved |
| Notifications that the system is going to sleep (10h to 1Fh) | |
| 10h | System is going to enter ACPI G1 (S1, S2, or S3) (see ACPI) (i.e., sleeping) |
| 11h | System is going to enter ACPI G1 (S1) (i.e., sleeping) |
| 12h | System is going to enter ACPI G1 (S2) (i.e., sleeping) |
| 13h | System is going to enter ACPI G1 (S3) (i.e., sleeping) |
| 14h | System is going to enter ACPI G1 (S4) (i.e., hibernating) |
| 15h | System is going to enter ACPI G2 (S5) (i.e., soft off) |
| 16h to 1Fh | Reserved |
| Notifications that the system is back from sleep (20h to 2Fh) | |
| 20h | System is back from ACPI G1 (S1, S2, or S3) (i.e., sleeping) |
| 21h | System is back from ACPI G1 (S1) (i.e., sleeping) |
| 22h | System is back from ACPI G1 (S2) (i.e., sleeping) |
| 23h | System is back from ACPI G1 (S3) (i.e., sleeping) |
| 24h | System is back from ACPI G1 (S4) (i.e., hibernating) |
| 25h to 2Fh | Reserved |
| Other (30h to 3Fh) | |
| 30h to 3Fh | Reserved |

Table 47 defines the DEVICE POWER ACTION field.

**Table 47 — DEVICE POWER ACTION field**

| Code | Description |
|---|---|
| General notifications (00h to 0Fh) | |
| 00h | No device power action notification |
| 01h to 0Fh | Reserved |
| Notifications that the device is going to be requested to change device state (10h to 1Fh) | |
| 10h | System is going to request that the device enter D0 (see ACPI and PCI-PM) (i.e., on) |
| 11h | System is going to request that the device enter D1 (see ACPI and PCI-PM) |
| 12h | System is going to request that the device enter D2 (see ACPI and PCI-PM) |
| 13h | System is going to request that the device enter D3$_{hot}$ or D3 (i.e., off) (see ACPI and PCI-PM) |
| 14h to 1Fh | Reserved |

NOTE 21 - ~~The PowerState field in PCI Power Management Capabilities Power Management Control/Status register (i.e., PMCSR) (see PCI-PM) may be used to set the PQI device to a new device power state (e.g., D0, D1, D2, or D3$_{hot}$) after the device power action notification.~~

The PowerState field in the PCI Power Management Capabilities Power Management Control/Status register (i.e., PMCSR) (see PCI-PM) is used to set the PQI device to a new device power state (e.g., D0, D1, D2, or D3$_{hot}$) after the device power action notification.

This standard does not define the specific actions that a PQI device should take to prepare for the specified power actions (e.g., flush caches, terminate background tasks, spin down rotating media on a notification that the system is going to sleep, or resume background tasks and spin up rotating media if the system is back from sleep).

The PQI device shall continue to process administrator queues and operational queues after ~~a power action~~ being notified of an upcoming power action.

# 6 Queuing layer

## 6.1 IQ CI, IQ PI, OQ CI, and OQ PI structures

### 6.1.1 IQ CI dword

An IQ CI dword is a dword in memory space that contains an IQ CI (see 4.3.2.2).

Table 48 defines the structure of the IQ CI dword.

~~Table 48 defines the structure of the dword containing the IQ CI.~~

Table 48 — IQ CI dword

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IQ CI | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |

The IQ CI field contains the IQ CI. It is not an error to set the IQ CI field to the same value that it already contains.

### 6.1.2 IQ PI dword

An IQ PI dword is a dword in PQI device memory space that contains an IQ PI (see 4.3.2.2).

Table 49 defines the structure of the IQ PI dword.

~~Table 49 defines the structure of the dword containing the IQ PI.~~

~~This dword is in PQI device memory space.~~

Table 49 — IQ PI dword

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IQ PI | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |

The IQ PI field contains the IQ PI. It is not an error to set the IQ PI field to the same value that it already contains.

### 6.1.3 OQ CI dword

An OQ CI dword is a dword in PQI device memory space that contains an OQ CI (see 4.3.2.3).

Table 50 defines the structure of the OQ CI dword.

~~Table 50 defines the structure of the dword containing the OQ CI.~~

~~This dword is in PQI device memory space.~~

**Table 50 — OQ CI dword**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OQ CI | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | REARM INTERRUPT | Reserved | | | | | | |

The OQ CI field contains the OQ CI. It is not an error to set the OQ CI field to the same value that it already contains (e.g., to set the REARM INTERRUPT bit to one).

When written, a REARM INTERRUPT bit set to:

a) one specifies that the PQI device shall rearm the interrupt associated with the OQ; and
b) zero shall be ignored.

~~When read, the REARM INTERRUPT bit shall be set to zero.~~When read, the PQI device shall return the REARM INTERRUPT bit set to zero.

### 6.1.4 OQ PI dword

An OQ PI dword is a dword in memory space that contains an OQ PI (see 4.3.2.3).

Table 51 defines the structure of the OQ PI dword.

~~Table 51 defines the structure of the dword containing the OQ PI.~~

**Table 51 — OQ PI dword**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OQ PI | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |

The OQ PI field contains the OQ PI. It is not an error to set the OQ PI field to the same value that it already contains.

# 7 SGL (scatter gather list)

## 7.1 SGL overview

An SGL is a data structure used to describe a data buffer. A data buffer is either a source data buffer or a destination data buffer. An SGL contains one or more SGL segments, ending with a last SGL segment.

SGL errors are processed using a method defined in the information unit layer standard (e.g., SOP).

## 7.2 Standard SGL segment and ~~standard~~ last standard SGL segment

A standard SGL segment:

a) is a data structure in a contiguous region of memory space describing all, part, or none of a data buffer and the next SGL segment, if any; and
b) consists of an array of one or more SGL descriptors.

If the array has more than one SGL descriptor, then each SGL descriptor prior to the last SGL descriptor shall not be:

a) a~~n~~ ~~SGL~~ Standard SGL Segment descriptor (see 7.3.4);
b) a~~n~~ ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor (see 7.3.5); or
c) a~~n~~ ~~SGL Alternative Last Segment~~Last Alternative SGL Segment descriptor (see Annex A).

A ~~standard~~ last standard SGL segment is a standard SGL segment that does not contain:

a) a~~n~~ ~~SGL~~ Standard SGL Segment descriptor (see 7.3.4);
b) a~~n~~ ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor (see 7.3.5); or
c) a~~n~~ ~~SGL Alternative Last Segment~~Last Alternative SGL Segment descriptor (see Annex A) (e.g., the SGL segment is described by a~~n~~ ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor).

An SGL shall be processed as having an error if it contains:

a) a standard SGL segment that contains more than one of the following:
   A) a~~n~~ ~~SGL~~ Standard SGL Segment descriptor;
   B) a~~n~~ ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor; or
   C) a~~n SGL Alternative Last Segment~~Last Alternative SGL Segment descriptor;

or

b) a ~~standard~~ last standard SGL segment that contains one of the following:
   A) ~~SGL~~ Standard SGL Segment descriptor;
   B) ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor; or
   C) ~~SGL Alternative Last Segment~~Last Alternative SGL Segment descriptor.

NOTE 22 ~~-Refer to other sections or standards to determine how to process SGL errors.~~

NOTE 23 - ~~Refer to other sections or standards for segment and descriptor address alignment constraints.~~

Table 52 defines the standard SGL segment.

Table 52 — Standard SGL segment

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | SGL descriptor [first] (see table 53) | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| ••• | ••• | | | | | | | |
| n - 15 | SGL descriptor [last] (see table 53) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The standard SGL segment contains one or more SGL descriptors (see 7.3).

## 7.3 SGL descriptors

### 7.3.1 SGL descriptors overview

Table 53 defines the SGL descriptor format.

Table 53 — SGL descriptor format

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Descriptor type specific | | | | | | | |
| ••• | | | | | | | | |
| 14 | | | | | | | | |
| 15 | SGL DESCRIPTOR TYPE | | | | Descriptor type specific | | | |

The SGL DESCRIPTOR TYPE field (see table 54) specifies the SGL descriptor type. If the SGL DESCRIPTOR TYPE field is set to a reserved or unsupported value, then the SGL descriptor shall be processed as having an error.

Table 54 — SGL DESCRIPTOR TYPE field

| Code | Descriptor | Reference |
|---|---|---|
| 0h | ~~SGL~~ Data Block descriptor | 7.3.2 |
| 1h | ~~SGL~~ Bit Bucket descriptor | 7.3.3 |
| 2h | ~~SGL~~ Standard SGL Segment descriptor | 7.3.4 |
| 3h | ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor | 7.3.5 |
| 4h | ~~SGL Alternative Last Segment~~Last Alternative SGL Segment descriptor | Annex A |
| Fh | Vendor specific | |
| All others | Reserved | |

NOTE 24 - ~~An SGL descriptor set to all zeros is an SGL Data Block descriptor with the ADDRESS field set to 00000000_00000000h and the LENGTH field set to 00000000h and may be used as a NULL descriptor.~~

A SGL descriptor set to all zeros is a Data Block descriptor with the ADDRESS field set to 00000000_00000000h and the LENGTH field set to 00000000h and may be used to occupy space in the SGL without specifying an operation.

### 7.3.2 ~~SGL~~ Data Block descriptor

The ~~SGL~~ Data Block descriptor describes a data block.

Table 55 defines the ~~SGL~~ Data Block descriptor.

**Table 55 — ~~SGL~~ Data Block descriptor**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ADDRESS | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 7 | (MSB) | | | | | | | |
| 8 | LENGTH | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 11 | (MSB) | | | | | | | |
| 12 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 14 | | | | | | | | |
| 15 | SGL DESCRIPTOR TYPE (0h) | | | | ZERO | | | |

The ADDRESS field specifies the starting 64-bit memory byte address of the data block.

The LENGTH field specifies the length in bytes of the data block. A LENGTH field set to 00000000h specifies that no data be transferred. A~~n~~ ~~SGL~~ Data Block descriptor specifying that no data be transferred shall not be processed as having an error.

If the value in the ADDRESS field plus the value in the LENGTH field is greater than 1_00000000_00000000h, then the ~~SGL~~ Data Block descriptor shall be processed as having an error.

The ZERO field shall contain 0h. A~~n~~ ~~SGL~~ Data Block descriptor containing a ZERO field set to a value other than 0h shall be processed as having an error.

The SGL DESCRIPTOR TYPE field is defined in 7.3.1 and shall be set as shown in table 55 for the ~~SGL~~ Data Block descriptor.

### 7.3.3 ~~SGL~~ Bit Bucket descriptor

The ~~SGL~~ Bit Bucket descriptor is used to discard (i.e., skip over) parts of source data.

Table 56 defines the ~~SGL~~ Bit Bucket descriptor.

**Table 56 — ~~SGL~~ Bit Bucket descriptor**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | LENGTH | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 11 | (MSB) | | | | | | | |
| 12 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 14 | | | | | | | | |
| 15 | SGL DESCRIPTOR TYPE (1h) | | | | ZERO | | | |

If the SGL describes a destination data buffer, then the LENGTH field specifies the number of bytes of the source data not to be transferred (i.e., the number of bytes to be discarded). A LENGTH field set to 00000000h specifies that no source data shall be discarded. A~~n~~ ~~SGL~~ Bit Bucket descriptor specifying that no source data be discarded shall not be processed as having an error.

If the SGL describes a source data buffer, then the LENGTH field shall be ignored and no data shall be discarded from the source data or destination data. A~~n~~ ~~SGL~~ Bit Bucket descriptor specifying that no data be discarded shall not be processed as having an error.

The ZERO field shall contain 0h. A~~n~~ ~~SGL~~ Bit Bucket descriptor containing a ZERO field set to a value other than 0h shall be processed as having an error.

The SGL DESCRIPTOR TYPE field is defined in 7.3.1 and shall be set as shown in table 56 for the ~~SGL~~ Bit Bucket descriptor.

### 7.3.4 ~~SGL~~ Standard SGL Segment descriptor

The ~~SGL~~ Standard SGL Segment descriptor describes the next SGL segment,~~ which is not the last SGL segment~~ which is a standard SGL segment (see 7.2) and may or may not be the last standard SGL segment (see 7.2).

Table 57 defines the ~~SGL~~ Standard SGL Segment descriptor.

**Table 57 — ~~SGL~~ Standard SGL Segment descriptor**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ADDRESS | | | (LSB) | Reserved | | | |
| ••• | | | | | | | | |
| 7 | (MSB) | | | | | | | |
| 8 | LENGTH | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 11 | (MSB) | | | | | | | |
| 12 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 14 | | | | | | | | |
| 15 | SGL DESCRIPTOR TYPE (2h) | | | | ZERO | | | |

~~The ADDRESS field specifies the upper 60 bits of the starting 64-bit memory byte address of the next SGL segment, which is required to be a standard SGL segment (see 7.2).~~

The ADDRESS field specifies the upper 60 bits of the 64-bit memory space address of the next SGL segment, which is a standard SGL segment (see 7.2). The lower bits of the memory space address of the next SGL segment are zero.

The LENGTH field specifies the length in bytes of the next SGL segment. A LENGTH field set to zero or a value that is not a multiple of 16 shall be processed as an error.

If the value in the ADDRESS field plus the value in the LENGTH field is greater than 1_00000000_00000000h, then the ~~SGL~~ Standard SGL Segment descriptor shall be processed as having an error.

The ZERO field shall contain 0h. A~~n~~ ~~SGL~~ Standard SGL Segment descriptor containing a ZERO field set to a value other than 0h shall be processed as having an error.

The SGL DESCRIPTOR TYPE field is defined in 7.3.1 and shall be set as shown in table 57 for the ~~SGL~~ Standard SGL Segment descriptor.

### 7.3.5 ~~SGL Standard Last Segment~~ Last Standard SGL Segment descriptor

The ~~SGL Standard Last Segment~~ Last Standard SGL Segment descriptor describes the next SGL segment, which is ~~the last SGL segment~~a last standard SGL segment (see 7.2).

Table 58 defines the ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor.

**Table 58 — ~~SGL Standard Last Segment~~ Last Standard SGL Segment descriptor**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ADDRESS | | | (LSB) | Reserved | | | |
| ••• | | | | | | | | |
| 7 | (MSB) | | | | | | | |
| 8 | LENGTH | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 11 | (MSB) | | | | | | | |
| 12 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 14 | | | | | | | | |
| 15 | SGL DESCRIPTOR TYPE (3h) | | | | ZERO | | | |

~~The ADDRESS field specifies the upper 60 bits of the starting 64-bit memory byte address of the next SGL segment, which is the last SGL segment and is a standard SGL segment (see 7.2).~~

The ADDRESS field specifies the upper 60 bits of the 64-bit memory space address of the next SGL segment, which is the last SGL segment and is a standard SGL segment (see 7.2). The lower bits of the memory space address of the next SGL segment are zero.

The LENGTH field specifies the length in bytes of the SGL segment. A LENGTH field set to zero or a value that is not a multiple of 16 shall be processed as an error.

NOTE 25 - ~~A standard last SGL segment that contains an SGL Standard Segment descriptor, an SGL Standard Last Segment descriptor, or an SGL Alternative Last Segment descriptor is processed as having an error (see 7.2).~~

If the value in the ADDRESS field plus the value in the LENGTH field is greater than 1_00000000_00000000h, then the ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor shall be processed as having an error.

The ZERO field shall contain 0h. A~~n~~ ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor containing a ZERO field set to a value other than 0h shall be processed as having an error.

The SGL DESCRIPTOR TYPE field is defined in 7.3.1 and shall be set as shown in table 58 for the ~~SGL Standard Last Segment~~Last Standard SGL Segment descriptor.

## 7.4 SGL examples

### 7.4.1 Example of a data transfer from a source data stream to a destination data buffer

Figure 27 shows an example of a data transfer from a source data stream to a destination data buffer in memory space described by an SGL. The total length of the source data stream is 13 KiB and the total length of the destination data buffer is 11 KiB. The destination data buffer is described by an SGL that contains three standard SGL segments. The three standard SGL segments contain a total of three ~~SGL~~ Data Block descriptors with lengths of 3 KiB, 4 KiB, and 4 KiB respectively. Standard SGL segments 1 of the Destination SGL contains a~~n~~ ~~SGL~~ Bit Bucket descriptor (see 7.3.3) with a length of 2 KiB that specifies not transferring (i.e., discard) 2 KiB of data of the source data stream. SGL segment 1  also contains a ~~an SGL Standard Last Segment~~Last Standard SGL Segment descriptor (see 7.3.5) specifying that the standard SGL segment pointed to by the descriptor is the last standard SGL segment.

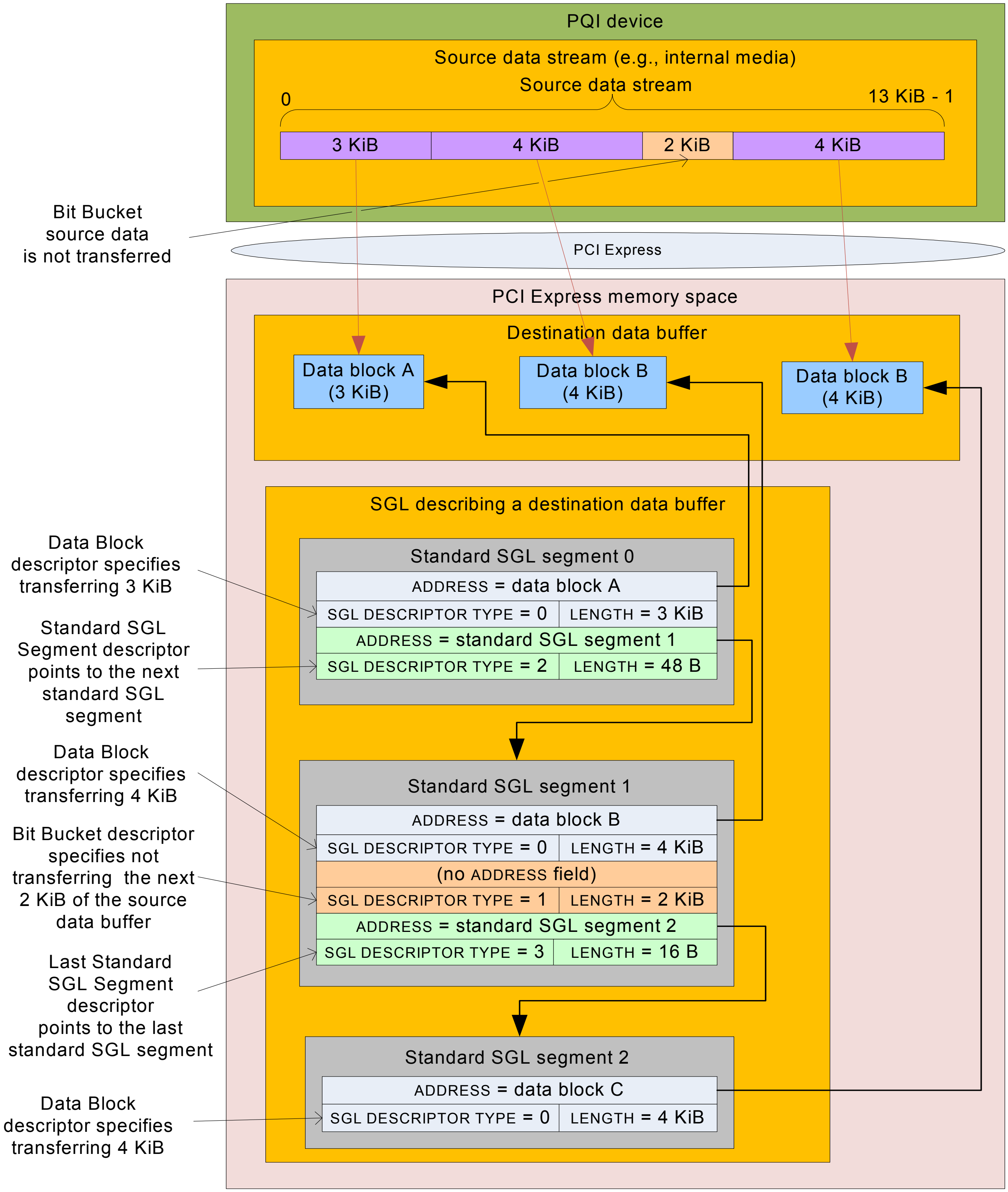

PQI device
Source data stream (e.g., internal media)
Source data stream
0
13 KiB - 1
3 KiB
4 KiB
2 KiB
4 KiB
Bit Bucket source data is not transferred
PCI Express
PCI Express memory space
Destination data buffer
Data block A (3 KiB)
Data block B (4 KiB)
Data block B (4 KiB)
SGL describing a destination data buffer
Standard SGL segment 0
ADDRESS = data block A
SGL DESCRIPTOR TYPE = 0
LENGTH = 3 KiB
ADDRESS = standard SGL segment 1
SGL DESCRIPTOR TYPE = 2
LENGTH = 48 B
Standard SGL segment 1
ADDRESS = data block B
SGL DESCRIPTOR TYPE = 0
LENGTH = 4 KiB
(no ADDRESS field)
SGL DESCRIPTOR TYPE = 1
LENGTH = 2 KiB
ADDRESS = standard SGL segment 2
SGL DESCRIPTOR TYPE = 3
LENGTH = 16 B
Standard SGL segment 2
ADDRESS = data block C
SGL DESCRIPTOR TYPE = 0
LENGTH = 4 KiB
Data Block descriptor specifies transferring 3 KiB
Standard SGL Segment descriptor points to the next standard SGL segment
Data Block descriptor specifies transferring 4 KiB
Bit Bucket descriptor specifies not transferring the next 2 KiB of the source data buffer
Last Standard SGL Segment descriptor points to the last standard SGL segment
Data Block descriptor specifies transferring 4 KiB

Figure 27 — SGL example of a data transfer from a source data stream to a destination data buffer

### 7.4.2 Example of a data transfer from a source data buffer to a destination data stream

Figure 28 shows an example of a data transfer from a source data buffer in memory space described by an SGL to a destination data stream. The total length of the destination data stream is 11 KiB and the total length of the source data buffer is 11 KiB. The source data buffer is described by an SGL that contains one standard SGL segment that contain three ~~SGL ~~Data Block descriptors with lengths of 3 KiB, 4 KiB, and 4 KiB respectively. The source SGL also contains a~~n~~ ~~SGL ~~Bit Bucket descriptor with a length of 2 KiB that is ~~ignored~~discarded because the SGL describes a source data buffer.

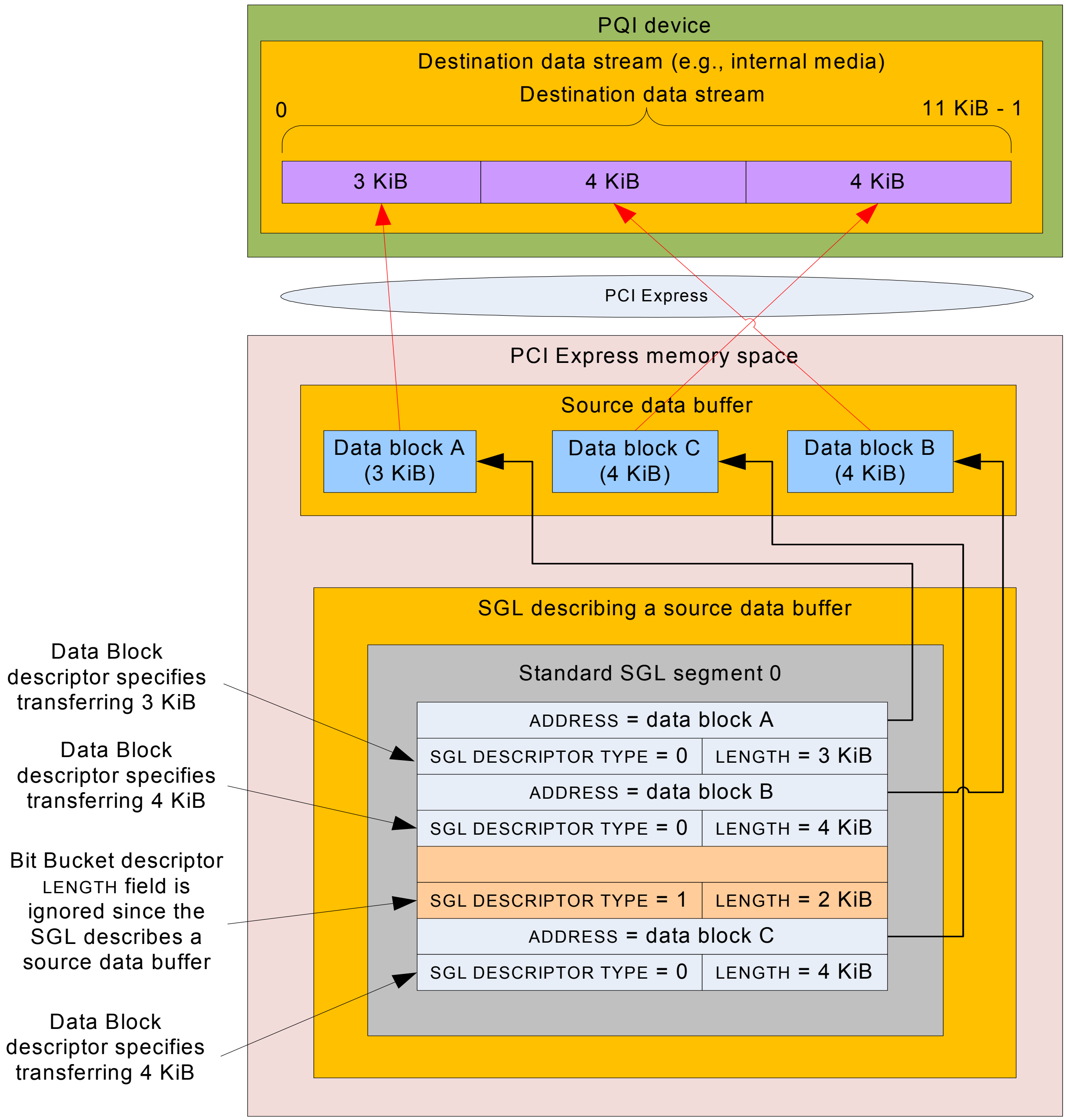


**Figure 28 — SGL example of a data transfer from a source data buffer to a destination data stream**

### 7.4.3 Example of a memory to memory data transfer

Figure 29 shows an example of a 12 KiB memory to memory data transfer from a source data buffer described by an SGL to a destination data buffer described by an SGL. The total length of the destination data buffer is 12 KiB and the total transferable length of the source data buffer is 12 KiB. The source data buffer is described by an SGL that contains one standard SGL segment that contains two ~~SGL~~ Data Block descriptors (see 7.3.2) with lengths of 6 KiB and 6 KiB respectively. The destination data buffer is described by an SGL made up of one standard SGL segment that contains three SGL Data Buffer descriptors with lengths of 4 KiB, 4 KiB, and 4 KiB respectively.

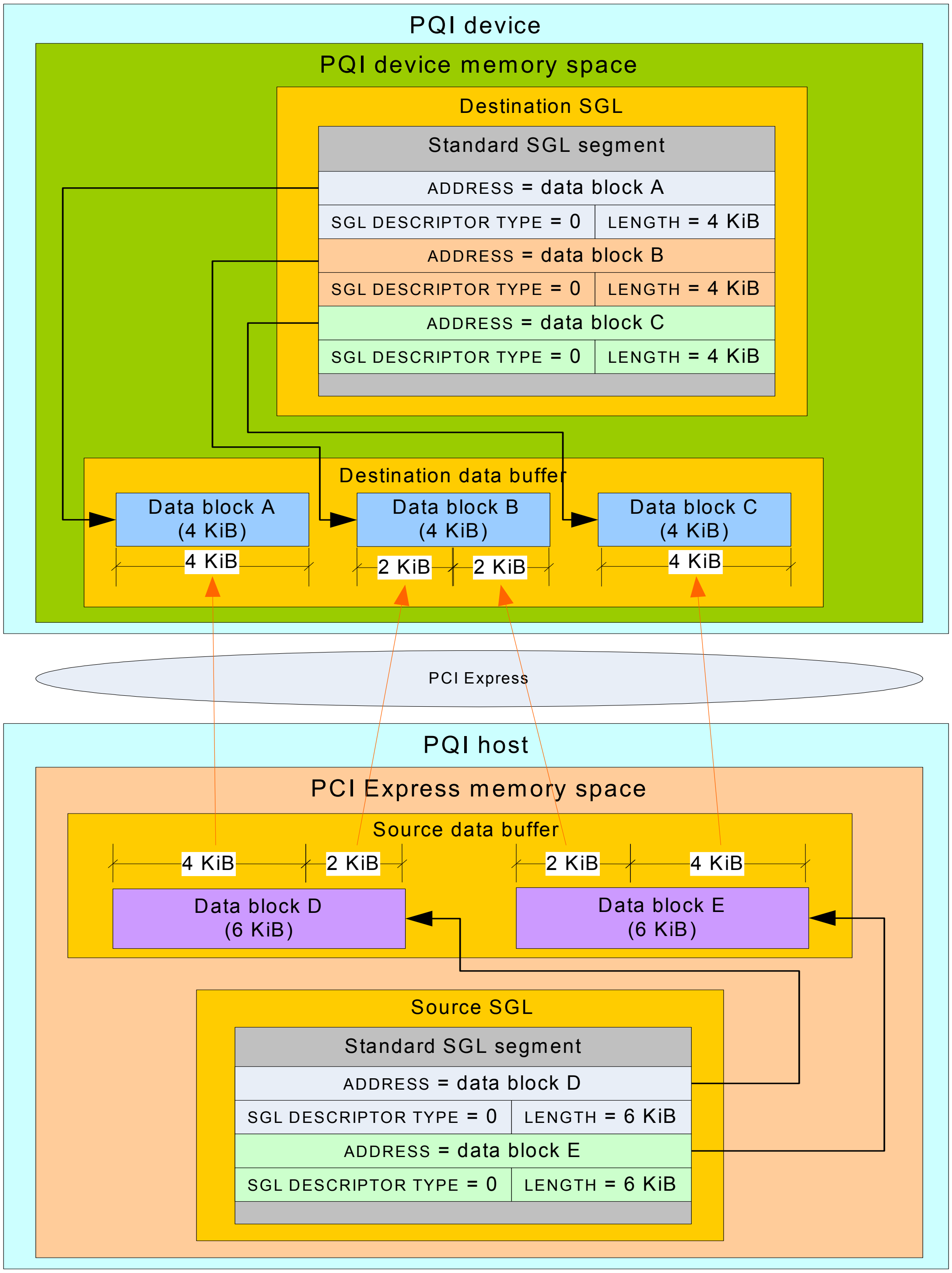


Figure 29 — SGL example of a memory to memory data transfer

# 8 Information unit layer

## ~~8.1.Operational IU usage and IU size~~

~~IU size is determined by the amount of information to be transferred. An IU that exceeds the size of an element shall be spanned across multiple consecutive elements.~~

~~The IU header indicates the number of elements that an IU occupies.~~

## 8.1 IUs and elements

### 8.1.1 IUs and elements overview

An IU contains a header (see table 59) followed by information defined by the IU type. The size of an IU is defined by the IU type. ~~If an IU is larger than the size of the element then the IU spans across multiple consecutive elements.~~If an IU is larger than the size of an element and IUs spanning multiple elements in the circular queue is not supported, then the producer shall not place the IU in the circular queue. If an IU is larger than the size of an element and IUs spanning multiple elements in the circular queue is supported, then the IU spans across multiple consecutive elements.

### 8.1.2 IU contained within a single element

Figure 30 shows an example of an IU that fits within a single element.

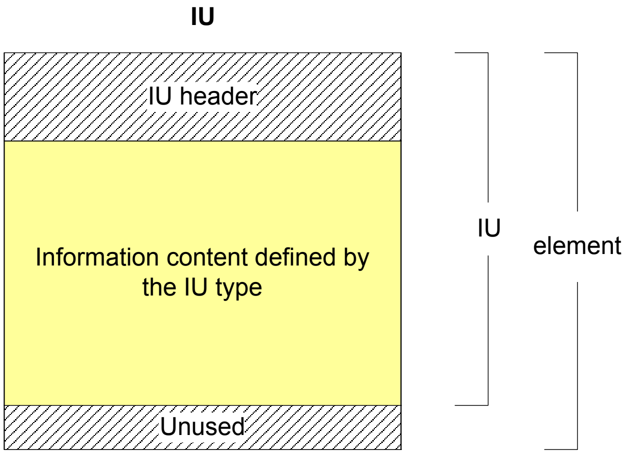


**Figure 30 — Example of an IU that fits within a single element ~~Single element containing an IU~~**

### 8.1.3 IU spanning multiple elements

When two or more elements are spanned to form a larger IU, only the first element includes the IU header. PQI host should not enqueue an IU larger than the size of number of elements minus 1.

Figure 31 shows an example of ~~a spanned IU~~an IU spanned across multiple elements.

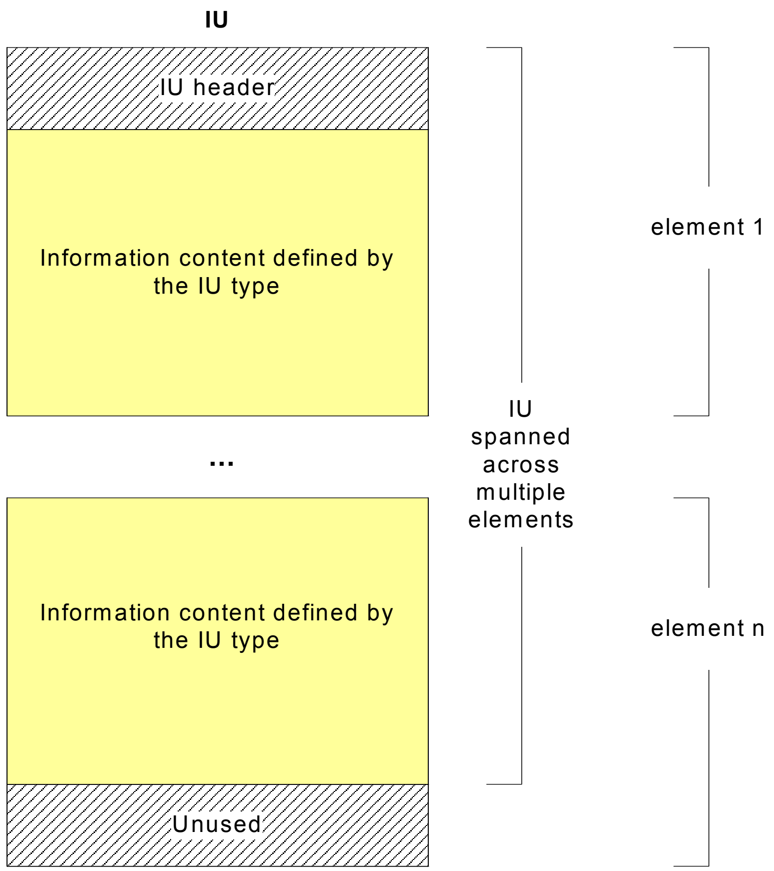

IU
IU header
Information content defined by the IU type
...
Information content defined by the IU type
Unused
IU spanned across multiple elements
element 1
element n

**Figure 31 — Example of an IU spanned multiple elements ~~Multiple elements containing a single IU~~**

### 8.1.4 IU with a field referencing an extended memory space

Figure  shows an example of an IU with a field referencing additional data in memory space to provide additional information for the IU.

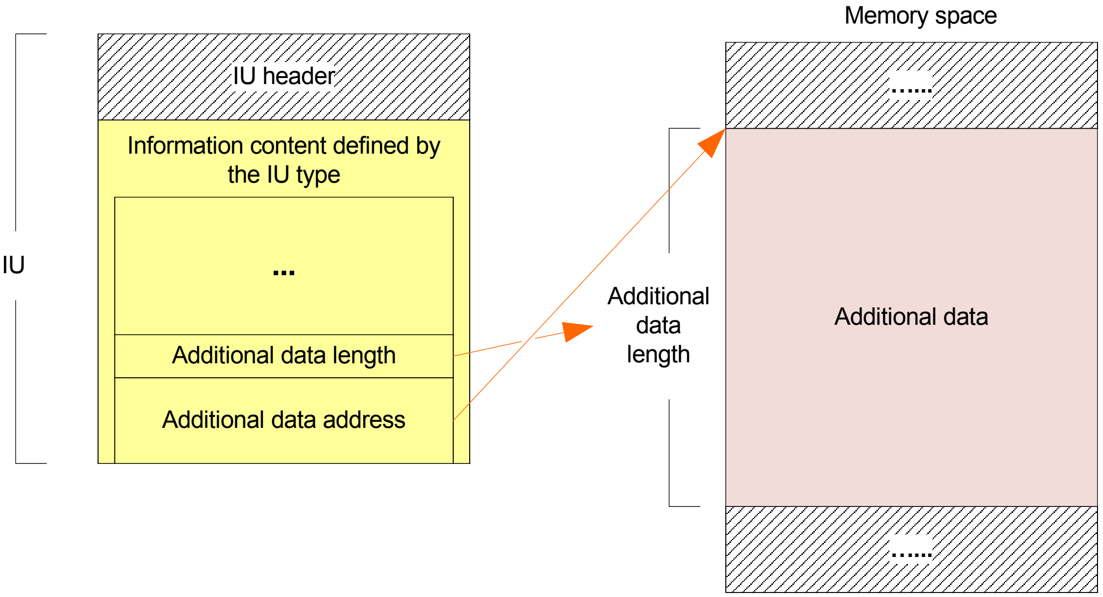


Figure 32 — Example of an IU with a field referencing additional data

## ~~8.3 OQ response data delivery~~

~~Information unit layers may utilize several methods of response data delivery, including:~~

a) ~~response data in an IU contained within a single OQ element;~~
b) ~~response data in an IU spanning multiple OQ elements; or~~
c) ~~response data in a data buffer referenced by an address specified in an IU.~~

~~Figure  shows a response data in an IU using a single OQ element.~~

~~Figure 30 - Response data in an IU using a single OQ element~~

~~Figure  shows a response IU with spanned OQ elements.~~

~~Figure 31 – Response data using spanned OQ elements~~

~~Figure  shows a response IU using a single OQ element that includes memory pointer to response data.~~

~~Figure 32 Response data by referenced PQI host memory address in response IU~~

## 8.2 ~~PQI~~ IU header common for all information unit layers

Table 59 defines the ~~PQI~~ IU header common for all information unit layers.

Byte labels for the ~~common~~ IU header are shaded in tables defining IUs.

**Table 59 — ~~PQI~~ IU header common for all information unit layers**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |

The IU TYPE field contains the type of IU and is defined by the information unit layer.

~~Table  defines the IU TYPE field.~~

~~IU TYPE field~~

| ~~Code~~ | ~~Type of IU~~ | ~~Originator~~ | ~~Reference~~ |
|---|---|---|---|
| ~~00h~~ | ~~NULL IU~~ | ~~any~~ | ~~9.1.2~~ |
| ~~Request IUs (01h to 7Fh)~~ | | | |
| ~~01h to 5Fh~~ | ~~Reserved~~ | | |
| ~~Administrator request IUs (60h to 6Fh)~~ | | | |
| ~~60h~~ | ~~Administrator request IU~~ | ~~PQI host~~ | ~~9.1.3~~ |
| ~~61h to 6Fh~~ | ~~Reserved~~ | | |
| ~~Vendor-specific administrator request IUs (70h to 7Fh)~~ | | | |
| ~~70h to 7Fh~~ | ~~Vendor specific~~ | | |
| ~~Response IUs (80h to FFh)~~ | | | |
| ~~80h to DFh~~ | ~~Reserved~~ | | |
| ~~Administrator response IUs (E0h to EFh)~~ | | | |
| ~~E0h~~ | ~~Administrator response IU~~ | ~~PQI device~~ | ~~9.1.4~~ |
| ~~E1h to EFh~~ | ~~Reserved~~ | | |
| ~~Vendor-specific administrator response IUs (F0h to FFh)~~ | | | |
| ~~F0h to FFh~~ | ~~Vendor specific~~ | | |
| ~~NOTE - Some information unit layers (e.g., SOP) do not use the 60h to 6Fh range and E0h to EFh range to avoid overlapping with the administrator request IU type codes and administrator response IU type codes.~~ | | | |

~~If the IU TYPE field is set to a reserved value or unsupported value, then the PQI device shall:~~

a) ~~stop consuming from the administrator IQ;~~
b) ~~transition to the PD4:Error state (see 4.5.6.1); and~~
c) ~~set the error code in the PQI Device Error register (see 5.2.18) to INVALID IU TYPE IN ADMINISTRATOR REQUEST IU.~~

The COMPATIBLE FEATURES field shall be set to the value shown in table 59 for the IU header. The recipient of the IU shall ignore the COMPATIBLE FEATURES field.

The IU LENGTH field contains the number of bytes that follow in the IU (i.e., the IU LENGTH field does not include the number of bytes in the IU header).

~~If the IU LENGTH field is set to a value other than the supported value (see 9.1), then the PQI device shall:~~

a) ~~stop consuming from the administrator IQ;~~
b) ~~transition to the PD4:Error state (see 4.5.6.1); and~~
c) ~~set the error code field in the PQI Device Error register (see 5.2.18) to INVALID IU LENGTH IN ADMINISTRATOR REQUEST IU.~~

## 8.3 Queuing interface ~~specific~~ descriptor definition

~~Table 60 defines the location of the queuing interface specific descriptor within the PQI IU.~~

For information unit layers that include a queuing interface descriptor in an IU, table 60 defines the location of the queuing interface descriptor.

The queuing interface ~~specific~~ descriptor contains IQ specific parameters and OQ specific parameters.

**Table 60 — ~~PQI IU queuing~~ Queuing interface ~~specific~~ descriptor location in an IU**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ~~PQI~~ IU header | | | | | | | |
| ••• | | | | | | | | |
| 3 | | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | IU specific | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

Table 61 defines the queuing interface ~~specific~~ descriptor for an inbound IU.

**Table 61 — Queuing interface ~~specific~~ descriptor definition for an inbound IU**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | RESPONSE OQ ID | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | WORK AREA | | | | | | | |
| 3 | | | | | | | | |

The RESPONSE OQ ID field specifies the OQ where the response IU is to be delivered.

~~For an administrator request IU, if the RESPONSE OQ ID field is not set to 0000h, then the PQI device shall:~~

a) ~~stop consuming from the administrator IQ;~~
b) ~~transition to the PD4:Error state (see 4.5.6.1); and~~
c) ~~set the error code in the PQI Device Error register (see 5.2.18) to INVALID OQ ID IN ADMINISTRATOR REQUEST IU.~~

~~The WORK AREA field shall be ignored by the recipient.~~

The WORK AREA field may be set to any value by the PQI host and shall be ignored by the PQI device.

Table 62 defines the queuing interface ~~specific~~ descriptor for an outbound IU.

**Table 62 — Queuing interface ~~specific~~ descriptor definition for an outbound IU**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| 1 | | | | | | | | |
| 2 | WORK AREA | | | | | | | |
| 3 | | | | | | | | |

~~The WORK AREA field shall be ignored by the recipient.~~

The WORK AREA field may be set to any value by the PQI device and should be ignored by the PQI host.

# 9 Administrator IUs and administrator functions

## 9.1 Administrator IU general formats

### 9.1.1 Administrator IU header

Table 63 defines the administrator IU header, which is compatible with the common IU header defined in 8.2.

**Table 63 — Administrator IU header**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |

Table 64 defines the IU TYPE field.

**Table 64 — IU TYPE field**

| Code | Type of IU | Queue type | Reference |
|---|---|---|---|
| 00h | Null IU | IO | 9.1.2 |
| • Request IUs (01h to 7Fh) | | | |
| 01h to 5Fh | Reserved | | |
| •• Administrator request IUs (60h to 6Fh) | | | |
| 60h | Administrator request IU | I | 9.1.3 |
| 61h to 6Fh | Reserved | | |
| •• Vendor-specific administrator request IUs (70h to 7Fh) | | | |
| 70h to 7Fh | Vendor specific | | |
| • Response IUs (80h to FFh) | | | |
| 80h to DFh | Reserved | | |
| •• Administrator response IUs (E0h to EFh) | | | |
| E0h | Administrator response IU | O | 9.1.4 |
| E1h to EFh | Reserved | | |
| •• Vendor-specific administrator response IUs (F0h to FFh) | | | |
| F0h to FFh | Vendor specific | | |
| **Key**:<br>I = inbound queue.<br>O = outbound queue.<br>IO = inbound queue and outbound queue. | | | |

If the IU TYPE field in an administrator request IU is set to a reserved value or unsupported value, then the PQI device:

a) shall stop consuming from the administrator IQ;
b) transitions to the PD4:Error state (see 4.5.6.1); and
c) shall set the error code  to INVALID IU TYPE IN ADMINISTRATOR REQUEST IU in the PQI Device Error register (see 5.2.18).

The COMPATIBLE FEATURES field shall be set to the value shown in table 63. The recipient of the IU shall ignore the COMPATIBLE FEATURES field.

The IU LENGTH field contains the number of bytes that follow in the IU. If the IU LENGTH field in an administrator request IU is set to a value other than the supported value, then the PQI device:

a) shall stop consuming from the administrator IQ;
b) transitions to the PD4:Error state (see 4.5.6.1); and

c) shall set the error code to INVALID IU LENGTH IN ADMINISTRATOR REQUEST IU in the PQI Device Error register (see 5.2.18).

### 9.1.2 ~~NULL~~Null IU

The ~~NULL~~Null IU shall be ignored by the recipient.

Table 65 defines the ~~NULL~~Null IU.

**Table 65 — ~~NULL~~Null IU**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (00h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (0000h) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI IU header~~IU header common for all information unit layers (see 8.2) and shall be set as shown in table 65 for the ~~NULL~~Null IU.

### 9.1.3 Administrator request IU format

An administrator request IU is sent by a PQI host to a PQI device to deliver a request from a PQI management application client that a PQI management device server perform an administrator function (see 9.2).

If:

a) a RsvdC bit or byte is set to a value other than zero in an administrator request IU after the IU header; or
b) a defined field is set to a reserved value or unsupported value in an administrator request IU after the IU header,

then the PQI management device server shall return an administrator response IU (see 9.1.4) with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

Table 66 defines the format for administrator request IUs.

**Table 66 — Administrator request IU format**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE | | | | | | | |
| 11 | Function specific fields | | | | | | | |
| ••• | | | | | | | | |
| 43 | | | | | | | | |
| 44 | DATA-IN BUFFER SIZE (if any),<br>DATA-OUT BUFFER SIZE (if any), or<br>function specific fields | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |
| 48 | SGL descriptor (if any) (see table 53) or function specific fields | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI IU header~~IU header common for all information unit layers (see 8.2) and shall be set as shown in table 66 for the ~~A~~administrator request IU format.

The queuing interface ~~specific~~ descriptor is defined in 8.3. Within the queuing interface descriptor, if the RESPONSE OQ ID field is not set to 0000h, then the PQI device shall:

a) stop consuming from the administrator IQ;
b) transition to the PD4:Error state (see 4.5.6.1); and
c) set the error code to INVALID OQ ID IN ADMINISTRATOR REQUEST IU in the PQI Device Error register (see 5.2.18).

The REQUEST IDENTIFIER field ~~contains a value that allows the PQI host to establish an identify for administrator functions~~specifies a request identifier that the PQI device shall use in the response IUs for this request IU. The request identifier allows the PQI host to determine the context for each response IU. The PQI management device server shall check the request identifiers for overlaps with administrator functions being processed. If the PQI management device server detects an overlap, then it shall:

a) abort the existing administrator function with that request identifier; and
b) return a single administrator response IU with:

A) the FUNCTION CODE field set to the function code specified in the new administrator request IU; and
B) the STATUS field set to OVERLAPPED REQUEST IDENTIFIER ATTEMPTED (see 9.1.4.1).

The FUNCTION CODE field specifies which administrator function is being requested and is defined in 9.2.1.

The function specific fields are defined by the specific administrator function being processed.

The DATA-IN BUFFER SIZE field, if any, is used by read administrator functions and specifies the size in bytes of the Data-In Buffer. The PQI management device server shall terminate transfers to the Data-In Buffer when the number of bytes specified by the DATA-IN BUFFER SIZE field have been transferred or when all available data has been transferred, whichever is less. A DATA-IN BUFFER SIZE field set to zero specifies that no data shall be transferred and is not an error unless otherwise specified for the administrator function. If the information being transferred to the Data-In Buffer includes fields containing the number of bytes to be transferred in some or all of the data, then the contents of these fields shall not be altered to reflect the truncation, if any, that results from an insufficient Data-In Buffer size.

The DATA-OUT BUFFER SIZE field, if any, is used by write administrator functions and specifies the size in bytes of the Data-Out Buffer. A DATA-OUT BUFFER SIZE field set to zero specifies that no data shall be transferred and is not an error unless otherwise specified for the administrator function.

The SGL descriptor, if any, contains the first standard SGL segment (see 7.3) of an SGL describing:

a) for a read administrator function, the Data-In Buffer as a destination data buffer; or
b) for a write administrator function, the Data-Out Buffer as a source data buffer.

Table 67 defines the SGL descriptor type support requirements for administrator request IUs.

**Table 67 — SGL descriptor type support requirements for administrator request IUs**

| SGL descriptor type [a] | Support ~~requirement~~ | Reference |
|---|---|---|
| 0h (i.e., ~~SGL~~ Data Block descriptor) | M | 7.3.2 |
| 1h (i.e., ~~SGL~~ Bit Bucket descriptor) | M | 7.3.3 |
| 2h (i.e., ~~SGL~~ Standard SGL Segment descriptor) | M | 7.3.4 |
| 3h (i.e., ~~SGL Standard~~ Last Standard SGL Segment descriptor) | M | 7.3.5 |
| 4h (i.e., ~~SGL Alternative Last Segment~~ Last Alternative SGL Segment descriptor) | O | Annex A |
| Fh (i.e., vendor specific) | ~~O~~ | |

Support ~~requirement~~ key:
M = Support is mandatory.
O = Support is optional.

[a] The REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3) reports the SGL descriptor types that are supported by a PQI device. The application client should only use mandatory SGL descriptor types until it retrieves the REPORT PQI DEVICE CAPABILITY parameter data (e.g., to send the REPORT PQI DEVICE CAPABILITY function itself).

NOTE 26 - ~~The REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3) reports the SGL descriptor types that are supported by a PQI device.~~

If the PQI management device server performs reserved field checking on the SGL descriptor contained in the IU and a reserved bit or byte in the ~~that~~ SGL descriptor is set to a value other than zero, then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

If the PQI management device server performs reserved field checking on an SGL descriptor in the SGL that is not contained in the IU and a reserved bit or byte in that SGL descriptor is set to a value other than zero, then the

PQI management device server shall return an administrator response IU with the STATUS field set to DATA BUFFER ERROR (see 9.1.4.1).

If the PQI management device server processes an SGL that has an error (see 7.2) or an SGL descriptor that has an error (see 7.3), then the PQI management device server shall return an administrator response IU with the STATUS field set to DATA BUFFER ERROR.

For a read administrator function, if the PQI management device server transfers fewer bytes than specified in the DATA-IN BUFFER SIZE field, then the PQI management device server shall return an administrator response IU with the STATUS field set to DATA-IN BUFFER UNDERFLOW (see 9.1.4.2). For a write administrator function, no indication is provided that the management device server transferred fewer bytes than specified in the DATA-OUT BUFFER SIZE field.

If the PQI management device server attempts to transfer beyond the length of the Data-In Buffer or Data-Out Buffer described by the SGL, then the PQI management device server shall return an administrator response IU with the STATUS field set to DATA BUFFER OVERFLOW (see 9.1.4.1).

While accessing the Data-In Buffer or Data-Out Buffer, ~~If~~if the PQI management device server encounters~~ one of the errors~~:

1) an error listed in table 68, then the PQI management device server shall return an administrator response IU with the STATUS field set to the corresponding value listed in table 68;
2) a PCI Express error (generic), then the PQI management device server shall return an administrator response IU with the STATUS field set to PCIE FABRIC ERROR; and
3) an generic error (e.g., the management device server is not able to determine if the error is a PCI Express error), then the PQI management device server shall return an administrator response IU with the STATUS field set to DATA BUFFER ERROR.

~~while accessing the Data-In Buffer or Data-Out Buffer,~~

~~then the PQI management device server shall return an administrator response IU with the STATUS field set to the corresponding value listed in table 68~~.

**Table 68 — Detail STATUS field values for errors while accessing the Data-In Buffer or Data-Out Buffer**

| Error | STATUS field |
|---|---|
| PCI Express completion timeout | PCIE COMPLETION TIMEOUT |
| PCI Express completer abort | PCIE COMPLETER ABORT |
| PCI Express poisoned TLP received | PCIE POISONED TLP RECEIVED |
| PCI Express ECRC check failed | PCIE ECRC CHECK FAILED |
| PCI Express unsupported request | PCIE UNSUPPORTED REQUEST |
| PCI Express ACS violation | PCIE ACS VIOLATION |
| PCI Express TLP prefix blocked | PCIE TLP PREFIX BLOCKED |
| ~~PCI Express error (generic)~~ | ~~PCIE FABRIC ERROR~~ |
| ~~Error (generic) (e.g., the management device server is not able to determine if the error is a PCI Express error)~~ | ~~DATA BUFFER ERROR~~ |

### 9.1.4 Administrator response IU format

#### 9.1.4.1 Administrator response IU format overview

An administrator response IU is sent by a PQI device to a PQI host to deliver a response for an administrator function (see 9.2) from a PQI management device server to a PQI management application client.

Table 69 defines the format for  administrator response IUs.

**Table 69 — Administrator response IU format**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Function specific fields | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI IU header~~IU header common for all information unit layers (see 8.2) and shall be set as shown in table 69 for the administrator response IU format.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field contains the request identifier of the administrator function for which this response is being returned.

The FUNCTION CODE field indicates the administrator function for which this response is being returned and is defined in 9.2.1.

Table 70 defines the STATUS field and additional status descriptor.

**Table 70 — STATUS field and additional status descriptor** (part 1 of 2)

| **STATUS field** | | **Additional status descriptor** |
|---|---|---|
| **Code** | **Name** | |
| Results indicating administrator function success (00h to 3Fh) | | |
| 00h | GOOD | Reserved |
| 01h | DATA-IN BUFFER UNDERFLOW | See 9.1.4.2 |
| 02h to 3Fh | Reserved | |
| Results indicating administrator function failure (40h to FFh) | | |
| • Errors accessing the Data Buffer (40h to 7Fh) | | |
| •• Miscellaneous errors accessing the Data Buffer (40h to 5Fh) | | |
| 40h | DATA BUFFER ERROR | Reserved |
| 41h | DATA BUFFER OVERFLOW | Reserved |
| 42h to 5Fh | Reserved | |
| •• PCI Express related errors accessing the Data Buffer (60h to 6Fh) | | |
| 60h | PCIE FABRIC ERROR | Reserved |
| 61h | PCIE COMPLETION TIMEOUT | Reserved |
| 62h | PCIE COMPLETER ABORT | Reserved |
| 63h | PCIE POISONED TLP RECEIVED | Reserved |
| 64h | PCIE ECRC CHECK FAILED | Reserved |
| 65h | PCIE UNSUPPORTED REQUEST | Reserved |
| 66h | PCIE ACS VIOLATION | Reserved |
| 67h | PCIE TLP PREFIX BLOCKED | Reserved |
| 68h to 6Fh | Reserved | |
| •• Other errors accessing the Data Buffer (70h to 7Fh) | | |
| 70h to 7Fh | Reserved | |

[a] The PQI management device server may or may not have transferred bytes beyond this offset.

**Table 70 — STATUS field and additional status descriptor** (part 2 of 2)

| STATUS field | | Additional status descriptor |
|---|---|---|
| Code | Name | |
| • Other errors (80h to EFh) | | |
| 80h | GENERIC ERROR | Reserved |
| 81h | OVERLAPPED REQUEST IDENTIFIER ATTEMPTED | Reserved |
| 82h | INVALID FIELD IN REQUEST IU | See 9.1.4.3 |
| 83h | INVALID FIELD IN DATA-OUT BUFFER | See 9.1.4.4 |
| 84h to EFh | Reserved | |
| • Vendor specific (F0h to FFh) | | |
| F0h to FFh | Vendor-specific | |

[a] The PQI management device server may or may not have transferred bytes beyond this offset.

The function specific fields are defined by the administrator function (see 9.2).

### 9.1.4.2 Additional status descriptor for a STATUS field set to DATA-IN BUFFER UNDERFLOW

Table 71 defines the additional status descriptor for a STATUS field set to DATA-IN BUFFER UNDERFLOW.

**Table 71 — Additional status descriptor for a STATUS field set to DATA-IN BUFFER UNDERFLOW**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | DATA TRANSFERRED | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 3 | (MSB) | | | | | | | |

The DATA TRANSFERRED field indicates the number of contiguous bytes starting with offset 0 in the Data-In Buffer that the PQI management device server transferred.

### 9.1.4.3 Additional status descriptor for a STATUS field set to INVALID FIELD IN REQUEST IU

Table 72 defines the additional status descriptor for a STATUS field set to INVALID FIELD IN REQUEST IU.

**Table 72 — Additional status descriptor for a STATUS field set to INVALID FIELD IN REQUEST IU**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | BYTE POINTER | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | ~~BPV~~<br>Reserved | BIT POINTER | | | Reserved | | | |

The BYTE POINTER field indicates the offset in the administrator request IU of the first byte (i.e., the lowest byte number) containing the field with the invalid value.

~~A BPV (bit pointer valid) bit set to one indicates that the BIT POINTER field is valid. A BPV bit set to zero indicates that the BIT POINTER field is not valid.~~

~~If the BPV bit is set to one, then the~~The BIT POINTER field indicates the offset in the byte ~~of~~in the administrator request IU pointed to by the BYTE POINTER field of the first bit (i.e., the lowest bit number) containing the field with the invalid value.

### 9.1.4.4 Additional status descriptor for a STATUS field set to INVALID FIELD IN DATA-OUT BUFFER

Table 73 defines the additional status descriptor for a STATUS field set to INVALID FIELD IN DATA-OUT BUFFER.

**Table 73 — Additional status descriptor for a STATUS field set to INVALID FIELD IN DATA-OUT BUFFER**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | BYTE POINTER | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 2 | (MSB) | | | | | | | |
| 3 | ~~BPV~~<br>Reserved | BIT POINTER | | | Reserved | | | |

The BYTE POINTER field indicates the offset in the Data-Out Buffer of the first byte (i.e., the lowest byte number) containing the field with the invalid value. If the byte that was in error is at an offset greater than FFFFFFh, then the BYTE POINTER field shall be set to FFFFFFh and the BIT POINTER field shall be set to 111b.

~~A BPV (bit pointer valid) bit set to one indicates that the BIT POINTER field is valid. A BPV bit set to zero indicates that the BIT POINTER field is not valid.~~

~~If the BPV bit is set to one, then the~~The BIT POINTER field indicates the offset in the byte of the Data-Out Buffer pointed to by the BYTE POINTER field of the first bit (i.e., the lowest bit number) containing the field with the invalid value.

## 9.2 Administrator functions

### 9.2.1 Administrator functions overview

Table 74 defines the administrator functions.

**Table 74 — Administrator functions (FUNCTION CODE field)** (part 1 of 2)

| Function code | Function | Description | Type | Support ~~requirement~~ | Reference |
|---|---|---|---|---|---|
| General functions (00h to 0Fh) | | | | | |
| 00h | REPORT PQI DEVICE CAPABILITY | Report PQI device capabilities | R | M | 9.2.2 |
| 01h | REPORT MANUFACTURER INFORMATION | Report PQI device manufacturing information | R | M | 9.2.3 |
| 02h to 0Fh | Reserved | | | | |
| Operational queue functions (10h to 1Fh) | | | | | |
| 10h | CREATE OPERATIONAL IQ | Create an operational IQ | N | M | 9.2.4 |
| 11h | CREATE OPERATIONAL OQ | Create an operational OQ | N | M | 9.2.5 |
| 12h | DELETE OPERATIONAL IQ | Delete an operational IQ | N | M | 9.2.6 |
| 13h | DELETE OPERATIONAL OQ | Delete an operational OQ | N | M | 9.2.7 |
| 14h | CHANGE OPERATIONAL IQ PROPERTIES | Change properties of an operational IQ | N | O | 9.2.8 |
| 15h | CHANGE OPERATIONAL OQ PROPERTIES | Change properties of an operational OQ | N | O | 9.2.9 |
| 16h | REPORT OPERATIONAL IQ LIST | Report list of configured operational IQs | R | M | 9.2.10 |
| 17h | REPORT OPERATIONAL OQ LIST | Report list of configured operational OQs | R | M | 9.2.11 |

**Key**:
M = Function implementation by the PQI device is mandatory.
O = Function implementation by the PQI device is optional.
R = read administrator function.
W = write administrator function.
N = non-data administrator function.

**Table 74 — Administrator functions (FUNCTION CODE field)** (part 2 of 2)

| Function code | Function | Description | Type | Support ~~requirement~~ | Reference |
|---|---|---|---|---|---|
| 18h to 1Fh | Reserved | | | | |
| Other (20h to DFh) | | | | | |
| 20h to DFh | Reserved | | | | |
| Vendor specific (E0h to FFh) | | | | | |
| E0h to FFh | Vendor specific | | | | |
| **Key**:<br>M = Function implementation by the PQI device is mandatory.<br>O = Function implementation by the PQI device is optional.<br>R = read administrator function.<br>W = write administrator function.<br>N = non-data administrator function. | | | | | |

### 9.2.2 REPORT PQI DEVICE CAPABILITY function

#### 9.2.2.1 REPORT PQI DEVICE CAPABILITY request

The REPORT PQI DEVICE CAPABILITY function requests that the PQI management device server return information about PQI device capabilities in the Data-In Buffer as defined in 9.2.2.3.

Table 75 defines the REPORT PQI DEVICE CAPABILITY request.

**Table 75 — REPORT PQI DEVICE CAPABILITY request**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (00h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 43 | | | | | | | | |
| 44 | DATA-IN BUFFER SIZE | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |
| 48 | SGL descriptor (see table 53) | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set to as shown in table 75 for the REPORT PQI DEVICE CAPABILITY request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in 9.2.1 and is set to the value shown in table 75 for the REPORT PQI DEVICE CAPABILITY request.

The DATA-IN BUFFER SIZE field is defined in 9.1.3.

The SGL descriptor is defined in 9.1.3.

NOTE 27 - ~~Only the mandatory SGL descriptor types should be used until the optional types are known.~~

### 9.2.2.2 REPORT PQI DEVICE CAPABILITY response

Table 76 defines the REPORT PQI DEVICE CAPABILITY response.

Table 76 — REPORT PQI DEVICE CAPABILITY response

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (00h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 76 for the REPORT PQI DEVICE CAPABILITY response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

~~The FUNCTION CODE field is set to the value shown in table 76.~~

The FUNCTION CODE field is defined in 9.2.1 and is set to the value shown in table 76 for the REPORT PQI DEVICE CAPABILITY response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

### 9.2.2.3 REPORT PQI DEVICE CAPABILITY parameter data

The format of the parameter data returned in the Data-In Buffer is shown in table 77.

**Table 77 — REPORT PQI DEVICE CAPABILITY parameter data (i.e., Data-In Buffer contents)** (part 1 of 3)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PARAMETER DATA LENGTH (003Eh) | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| Operational IQ capabilities | | | | | | | | |
| 16 | MAXIMUM OPERATIONAL IQS | | | | | | | (LSB) |
| 17 | (MSB) | | | | | | | |
| 18 | MAXIMUM OPERATIONAL IQ ELEMENTS | | | | | | | (LSB) |
| 19 | (MSB) | | | | | | | |
| 20 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 23 | | | | | | | | |
| 24 | MAXIMUM OPERATIONAL IQ ELEMENT LENGTH | | | | | | | (LSB) |
| 25 | (MSB) | | | | | | | |
| 26 | MINIMUM OPERATIONAL IQ ELEMENT LENGTH | | | | | | | (LSB) |
| 27 | (MSB) | | | | | | | |
| Operational OQ capabilities | | | | | | | | |
| 28 | Reserved | | | | | | | CIC |
| 29 | Reserved | | | | | | | |
| ~~28~~30 | MAXIMUM OPERATIONAL OQS | | | | | | | (LSB) |
| ~~29~~31 | (MSB) | | | | | | | |
| ~~30~~32 | MAXIMUM OPERATIONAL OQ ELEMENTS | | | | | | | (LSB) |
| ~~31~~33 | (MSB) | | | | | | | |
| ~~32~~ | ~~CIC~~ | ~~Reserved~~ | | | | | | |
| ~~33~~ | ~~Reserved~~ | | | | | | | |

Table 77 — REPORT PQI DEVICE CAPABILITY parameter data (i.e., Data-In Buffer contents) (part 2 of 3)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 34 | INTERRUPT COALESCING TIME GRANULARITY | | | | | | | (LSB) |
| 35 | (MSB) | | | | | | | |
| 36 | MAXIMUM OPERATIONAL OQ ELEMENT LENGTH | | | | | | | (LSB) |
| 37 | (MSB) | | | | | | | |
| 38 | MINIMUM OPERATIONAL OQ ELEMENT LENGTH | | | | | | | (LSB) |
| 39 | (MSB) | | | | | | | |
| Alignment | | | | | | | | |
| 40 | Reserved | | | | OPERATIONAL IQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT | | | |
| 41 | Reserved | | | | OPERATIONAL OQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT | | | |
| 42 | Reserved | | | | OPERATIONAL IQ CI ADDRESS ALIGNMENT EXPONENT | | | |
| 43 | Reserved | | | | OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT | | | |
| ~~40~~ | ~~OPERATIONAL IQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT~~ | | | | | | | |
| ~~41~~ | ~~OPERATIONAL OQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT~~ | | | | | | | |
| ~~42~~ | ~~OPERATIONAL IQ CI ADDRESS ALIGNMENT EXPONENT~~ | | | | | | | |
| ~~43~~ | ~~OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT~~ | | | | | | | |
| Other parameters | | | | | | | | |
| 44 | OPERATIONAL QUEUE PROTOCOL SUPPORT BITMASK | | | | | | | |
| ••• | | | | | | | | |
| 47 | | | | | | | | |
| 48 | ADMINISTRATOR SGL DESCRIPTOR TYPE SUPPORT BITMASK | | | | | | | |
| 49 | | | | | | | | |

Table 77 — REPORT PQI DEVICE CAPABILITY parameter data (i.e., Data-In Buffer contents) (part 3 of 3)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 50 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |
| Protocol specific descriptor list | | | | | | | | |
| 64 | Protocol specific descriptor [first] (see table 78) | | | | | | | |
| ••• | | | | | | | | |
| 79 | | | | | | | | |
| ••• | ••• | | | | | | | |
| m-15 | Protocol specific descriptor [last] (see table 78) | | | | | | | |
| ••• | | | | | | | | |
| m | | | | | | | | |

The PARAMETER DATA LENGTH field indicates the number of bytes that follow and shall be set as shown in ~~t~~table 77~~-for the REPORT PQI DEVICE CAPABILITY parameter data (data-in)~~.

The MAXIMUM OPERATIONAL IQS field indicates the maximum number of operational IQs supported by the PQI device.~~ The MAXIMUM OPERATIONAL IQS field shall not be set to 0000h.~~

The MAXIMUM OPERATIONAL IQ ELEMENTS field indicates the maximum number of elements in each of the operational IQs supported by the PQI device. The MAXIMUM OPERATIONAL IQ ELEMENTS field shall be set to a value ~~greater~~other than 0001h.

The MAXIMUM OPERATIONAL IQ ELEMENT LENGTH field indicates the maximum length of each operational IQ element in 16-byte increments (e.g., 0001h means 16 bytes and 00FFh means 4 080 bytes). The MAXIMUM OPERATIONAL IQ ELEMENT LENGTH field shall be greater than or equal to the MINIMUM OPERATIONAL OQ ELEMENT LENGTH field.

The MINIMUM OPERATIONAL IQ ELEMENT LENGTH field indicates the minimum length of each operational IQ element in 16-byte increments (e.g., 0001h means 16 bytes and 00FFh means 4 080 bytes) supported by the PQI device. The MINIMUM OPERATIONAL IQ ELEMENT LENGTH field shall not be set to 0000h.

A CIC (common interrupt coalescing) bit set to one indicates that the PQI device manages the values of the Coalescing Count attribute (see 4.3.6.11.5), the Minimum Coalescing Time attribute (see 4.3.6.11.3), the Maximum Coalescing Time attribute (see 4.3.6.11.4), and the Wait For Rearm attribute (see 4.3.6.11.6), such that:

a) the Coalescing Count attribute is the same value for all operational OQs;
b) the Minimum Coalescing Time attribute is the same value for all operational OQs;
c) the Maximum Coalescing Time attribute is the same value for all operational OQs; and
d) the Wait For Rearm attribute is the same value for all operational OQs.

A CIC bit set to zero indicates:

a) the Coalescing Count attribute may be a different value for each operational OQ;
b) the Minimum Coalescing Time attribute may be a different value for each operational OQ;
c) the Maximum Coalescing Time attribute may be a different value for each operational OQ; and
d) the Wait For Rearm attribute may be a different value for each operational OQ.

An INTERRUPT COALESCING TIME GRANULARITY field set to a value other than zero indicates the granularity supported for the coalescing timers (see 9.2.5.1) in 100 ns intervals. An INTERRUPT COALESCING TIME GRANULARITY field shall not be set to 0000h.

~~The MAXIMUM OPERATIONAL IQ ELEMENT LENGTH field indicates the maximum length of each operational IQ element in 16-byte increments (e.g., 01h means 16 bytes and FFh means 4 080 bytes). The MAXIMUM OPERATIONAL IQ ELEMENT LENGTH field shall be greater than or equal to the MINIMUM OPERATIONAL OQ ELEMENT LENGTH field and shall not be set to 0000h.~~

~~The MINIMUM OPERATIONAL IQ ELEMENT LENGTH field indicates the minimum length of each operational IQ element in 16-byte increments (e.g., 01h means 16 bytes and FFh means 4 080 bytes) supported by the PQI device. The MINIMUM OPERATIONAL IQ ELEMENT LENGTH field shall not be set to 0000h.~~

The MAXIMUM OPERATIONAL OQS field indicates the maximum number of operational OQs supported by the PQI device.~~ The MAXIMUM OPERATIONAL OQS field shall not be set to 0000h~~.

The MAXIMUM OPERATIONAL OQ ELEMENTS field indicates the maximum number of ~~queue ~~elements in each of the operational OQs supported by the PQI device. The MAXIMUM OPERATIONAL OQ ELEMENTS field shall be set to a value ~~greater~~other than 0001h.

The MAXIMUM OPERATIONAL OQ ELEMENT LENGTH field indicates the maximum length of each operational OQ element in 16-byte increments (e.g., 0001h means 16 bytes and 00FFh means 4 080 bytes) supported by the PQI device. The MAXIMUM OPERATIONAL OQ ELEMENT LENGTH field shall be greater than or equal to the MINIMUM OPERATIONAL OQ ELEMENT LENGTH field~~ and shall not be set to 0000h~~.

The MINIMUM OPERATIONAL OQ ELEMENT LENGTH field indicates the minimum length  of each operational OQ element in 16-byte increments (e.g., 0001h means 16 bytes and 00FFh means 4 080 bytes) supported by the PQI device. The MINIMUM OPERATIONAL OQ ELEMENT LENGTH field shall not be set to 0000h.

The OPERATIONAL IQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field indicates an exponent (i.e., x) for which the operational IQ element array address shall be an integer multiple of $2^x$ (e.g., if x equals 6, then the alignment is 64 bytes). PQI device shall set the OPERATIONAL IQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field to a value greater than or equal to 06h to satisfy the minimum alignment requirement defined by this standard (see 4.3.6.1.2).

The OPERATIONAL OQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field indicates an exponent (i.e., x) for which the operational OQ element array address shall be an integer multiple of $2^x$ (e.g., if x equals 6, then the alignment is 64 bytes). PQI device shall set the OPERATIONAL OQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field to a value greater than or equal to 06h to satisfy the minimum alignment requirement defined by this standard (see 4.3.6.1.2).

The OPERATIONAL IQ CI ADDRESS ALIGNMENT EXPONENT field indicates an exponent (i.e., x) for which the operational IQ CI address shall be an integer multiple of $2^x$ (e.g., if x equals 6, then the alignment is 64 bytes). PQI device shall set the OPERATIONAL IQ CI ADDRESS ALIGNMENT EXPONENT field to a value greater than or equal to 06h to satisfy the minimum alignment requirement defined by this standard (see 4.3.6.1.4).

The OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT field indicates an exponent (i.e., x) for which the operational OQ PI address shall be an integer multiple of $2^x$ (e.g., if x equals 6, then the alignment is 64 bytes). PQI device shall set the OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT field to a value greater than or equal to 06h to satisfy the minimum alignment requirement defined by this standard (see 4.3.6.1.3).

The OPERATIONAL QUEUE PROTOCOL SUPPORT BITMASK field indicates the operational queue protocols (see table 83) supported by the PQI device ~~(see table 83)~~. A bit set to one indicates that the corresponding operational queue protocol is supported. A bit set to zero indicates that the corresponding operational queue protocol is not supported. The first bit (i.e., byte 44 bit 0) corresponds to an operational queue protocol of 00h (i.e., SOP); the last bit (i.e., byte 47 bit 7) corresponds to an operational queue protocol of 1Fh (i.e., vendor specific).

The ADMINISTRATOR SGL DESCRIPTOR TYPE SUPPORT BITMASK field indicates the SGL descriptor types (see table 54) supported by the PQI device ~~(see table 54)~~ for administrator request IUs. A bit set to one indicates that the corresponding SGL descriptor type is supported. A bit set to zero indicates that the corresponding SGL descriptor type is not supported. The first bit (i.e., byte 48 bit 0) corresponds to an SGL descriptor type of 0h (i.e., the ~~SGL ~~Data Block descriptor); the last bit (i.e., byte 49 bit 7) corresponds to an SGL descriptor type of Fh. Bit 0, bit 1, bit 2, and bit 3 represent the mandatory SGL descriptor types for administrator IU (see table 67) and shall each be set to one.

The SGL descriptor type support bitmask for other protocol IUs (e.g., SOP) is not defined in this standard.

The protocol specific descriptor (see table 78) specifies the additional features supported by the operational queue protocol. The first protocol specific descriptor corresponds to an operational queue protocol of 00h (i.e., SOP); the last protocol specific descriptor corresponds to an operational queue protocol of 1Fh (i.e., vendor specific).

Table 78 — Protocol specific descriptor

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | INBOUND SPANNING |
| 1 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | MAXIMUM INBOUND IU LENGTH | | | | | | | (LSB) |
| 7 | (MSB) | | | | | | | |
| 8 | Reserved | | | | | | | OUTBOUND SPANNING |
| 9 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 13 | | | | | | | | |
| 14 | MAXIMUM OUTBOUND IU LENGTH | | | | | | | (LSB) |
| 15 | (MSB) | | | | | | | |

An INBOUND SPANNING bit set to one indicates that the information unit layer (e.g., SOP) supports spanning an inbound IU across multiple elements. An INBOUND SPANNING bit set to zero indicates that the information unit layer (e.g., SOP) does not support spanning an inbound IU across multiple elements.

The MAXIMUM INBOUND IU LENGTH field indicates the maximum number of bytes in an inbound IU supported by the information unit layer.

An OUTBOUND SPANNING bit set to one indicates that the information unit layer (e.g., SOP) supports spanning an outbound IU across multiple elements. An OUTBOUND SPANNING bit set to zero indicates that the information unit layer (e.g., SOP) does not support spanning an outbound IU across multiple elements.

The MAXIMUM OUTBOUND IU LENGTH field indicates the maximum number of bytes in an outbound IU supported by the information unit layer.

For each of the protocol specific descriptors, if the PQI device does not support the protocol, the PQI device shall set:

a) the INBOUND SPANNING bit to zero;
b) the MAXIMUM INBOUND IU LENGTH field to zero;
c) the OUTBOUND SPANNING bit to zero; and
d) the MAXIMUM OUTBOUND IU LENGTH field to zero.

Editor's Note 4: Add in SOP Annex A the value for SOP.

### 9.2.3 REPORT MANUFACTURER INFORMATION function

#### 9.2.3.1 REPORT MANUFACTURER INFORMATION request

The REPORT MANUFACTURER INFORMATION function requests that the PQI management device server return information about the PQI device manufacturer in the Data-In Buffer as defined in 9.2.3.3.

Table 79 defines the REPORT MANUFACTURER INFORMATION request.

**Table 79 — REPORT MANUFACTURER INFORMATION request**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (01h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 43 | | | | | | | | |
| 44 | DATA-IN BUFFER SIZE | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |
| 48 | SGL descriptor (see table 53) | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set ss shown in table 79 for the REPORT MANUFACTURER INFORMATION request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

~~The FUNCTION CODE field is set to the value shown in table 79.~~

The FUNCTION CODE field is defined in 9.2.1 and is set to the value shown in table 79 for the REPORT MANUFACTURER INFORMATION request.

The DATA-IN BUFFER SIZE field is defined in 9.1.3.

The SGL descriptor is defined in 9.1.3.

### 9.2.3.2 REPORT MANUFACTURER INFORMATION response

Table 80 defines the REPORT MANUFACTURER INFORMATION response.

**Table 80 — REPORT MANUFACTURER INFORMATION response**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (01h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 80 for the REPORT MANUFACTURER INFORMATION response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

~~The FUNCTION CODE field is set to the value shown in table 80.~~

The FUNCTION CODE field is defined in 9.2.1 and is set to the value shown in table 80 for the REPORT MANUFACTURER INFORMATION response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

### 9.2.3.3 REPORT MANUFACTURER INFORMATION parameter data

The format of the parameter data returned in the Data-In Buffer is shown in table 81.

**Table 81 — REPORT MANUFACTURER INFORMATION parameter data (i.e., Data-In Buffer contents)**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PARAMETER DATA LENGTH (007Eh) | | | | | | | (LSB) |
| 1 | (MSB) | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | PCI VENDOR ID | | | | | | | (LSB) |
| 5 | (MSB) | | | | | | | |
| 6 | PCI DEVICE ID | | | | | | | (LSB) |
| 7 | (MSB) | | | | | | | |
| 8 | PCI REVISION ID | | | | | | | |
| 9 | PCI CLASS CODE | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 11 | (MSB) | | | | | | | |
| 12 | PCI SUBSYSTEM VENDOR ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | PCI SUBSYSTEM ID | | | | | | | (LSB) |
| 15 | (MSB) | | | | | | | |
| 16 | (MSB) | PRODUCT SERIAL NUMBER | | | | | | |
| ••• | | | | | | | | |
| 47 | | | | | | | | (LSB) |
| 48 | (MSB) | T10 VENDOR IDENTIFICATION | | | | | | |
| ••• | | | | | | | | |
| 55 | | | | | | | | (LSB) |
| 56 | (MSB) | PRODUCT IDENTIFICATION | | | | | | |
| ••• | | | | | | | | |
| 71 | | | | | | | | (LSB) |
| 72 | (MSB) | PRODUCT REVISION LEVEL | | | | | | |
| ••• | | | | | | | | |
| 87 | | | | | | | | (LSB) |
| 88 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 127 | | | | | | | | |

The PARAMETER DATA LENGTH field indicates the number of bytes that follow and shall be set as shown in table 81 ~~for the REPORT MANUFACTURER INFORMATION parameter data~~.

The PCI VENDOR ID field indicates the identification of the manufacturer of the PCI device and shall be identical to the ~~PCI~~ Vendor ID field in configuration space (see PCI).

The PCI DEVICE ID field indicates the identification of the PCI device allocated by the PCI device ~~vendor~~manufacturer and shall be identical to the ~~PCI~~ Device ID field in configuration space (see PCI).

The PCI REVISION ID field indicates the revision of the PCI device allocated by the PCI device ~~vendor~~manufacturer and shall be identical to the ~~PCI~~ Revision ID field in configuration space (see PCI).

The PCI CLASS CODE field indicates the identification of the generic function and register-level programming interface of the PCI device defined by PCI-ID and shall be identical to the ~~PCI~~ Class Code field in configuration space (see PCI).

The PCI SUBSYSTEM VENDOR ID field indicates the manufacturer of the add-in card or subsystem containing the PCI device and shall be identical to the Subsystem Vendor ID field in configuration space (see PCI).

The PCI SUBSYSTEM ID field indicates the identification of the add-in card or subsystem identification allocated by the subsystem ~~vendor~~manufacturer used to ~~uniquely~~ identify the add-in card or subsystem where the PCI device resides and shall be identical to the Subsystem ID field in ~~the~~ configuration space (see PCI).

The PRODUCT SERIAL NUMBER field indicates ASCII data that is a ~~vendor~~manufacturer defined serial number. If the product serial number is not available, then the PRODUCT SERIAL NUMBER field shall contain ASCII spaces (20h).

The T10 VENDOR IDENTIFICATION field indicates eight bytes of left-aligned ASCII data identifying the ~~vendor~~manufacturer of the PQI device. The T10 vendor identification shall be one assigned by INCITS. A list of assigned T10 vendor identifications is defined in SPC-4 and on the T10 web site (http://www.t10.org).

The PRODUCT IDENTIFICATION field indicates sixteen bytes of left-aligned ~~PQI device vendor ASCII data~~ASCII product identification data defined by the manufacturer.

The PRODUCT REVISION LEVEL field contains sixteen bytes of left-aligned ~~PQI device vendor ASCII data~~ASCII product revision level data defined by the manufacturer.

### 9.2.4 CREATE OPERATIONAL IQ function

#### 9.2.4.1 CREATE OPERATIONAL IQ request

The CREATE OPERATIONAL IQ function requests that the PQI device create a new operational IQ.

Table 82 defines the CREATE OPERATIONAL IQ request.

**Table 82 — CREATE OPERATIONAL IQ request**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (10h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| 12 | IQ ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | RsvdC | | | | | | | |
| 15 | | | | | | | | |
| 16 | IQ ELEMENT ARRAY ADDRESS | (LSB) | RsvdC | | | | | |
| ••• | | | | | | | | |
| 23 | (MSB) | | | | | | | |
| 24 | IQ CI ADDRESS | | | | | (LSB) | RsvdC | |
| ••• | | | | | | | | |
| 31 | (MSB) | | | | | | | |
| 32 | NUMBER OF ELEMENTS | | | | | | | (LSB) |
| 33 | (MSB) | | | | | | | |
| 34 | ELEMENT LENGTH | | | | | | | (LSB) |
| 35 | (MSB) | | | | | | | |
| 36 | Reserved | | OPERATIONAL QUEUE PROTOCOL | | | | | |
| 37 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 59 | | | | | | | | |
| 60 | Vendor specific | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set as shown in table 82 for the CREATE OPERATIONAL IQ request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 82 for the CREATE OPERATIONAL IQ request.

The IQ ID field specifies the IQ ID (see 4.3.6.8.2) to be assigned to the operational IQ. If:

a) the IQ ID field is set to 0000h;
b) the IQ ID field is set to a value that is already assigned to an operational IQ; or
c) the IQ ID field is set to a value greater than the MAXIMUM OPERATIONAL IQS field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3),

then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The IQ ELEMENT ARRAY ADDRESS field specifies the upper 58 bits of the 64-bit operational IQ element array address (see 4.3.2.1). The lower bits of the operational IQ element array address are zero. If the IQ ELEMENT ARRAY ADDRESS field is set to a value that does not meet the alignment requirement indicated by the OPERATIONAL IQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The IQ CI ADDRESS field specifies the upper 62 bits of the 64-bit operational IQ CI address (see 4.3.2.2). The lower bits of the operational IQ CI address are zero. If the IQ CI ADDRESS field is set to a value that does not meet the alignment requirement indicated by the OPERATIONAL IQ CI ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The NUMBER OF ELEMENTS field specifies the number of elements in the operational IQ element array (see 4.3.2.2). If:

a) the NUMBER OF ELEMENTS field is set to a value less than 0002h; or
b) the NUMBER OF ELEMENTS field is set to a value greater than the MAXIMUM OPERATIONAL IQ ELEMENTS field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3),

then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The ELEMENT LENGTH field specifies the element length in 16-byte increments (e.g., a value of 01h in the ELEMENT LENGTH field specifies an element length of 16 bytes). If the ELEMENT LENGTH field is set to a value:

a) less than the MINIMUM OPERATIONAL IQ ELEMENT LENGTH field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3); or
b) greater than the MAXIMUM OPERATIONAL IQ ELEMENT LENGTH field in the REPORT PQI DEVICE CAPABILITY parameter data,

then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The OPERATIONAL QUEUE PROTOCOL (see table 83) specifies the information unit layer protocol used by the operational queue (see 4.3.1).

**Table 83 — OPERATIONAL QUEUE PROTOCOL field**

| Code | Operational queue protocol | Reference |
|---|---|---|
| 00h | SOP | SOP |
| 01h to 0Fh | Reserved | |
| 10h to 1Fh | Vendor specific | |

### 9.2.4.2 CREATE OPERATIONAL IQ response

Table 84 defines the CREATE OPERATIONAL IQ response.

**Table 84 — CREATE OPERATIONAL IQ response**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (10h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | IQ PI OFFSET | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 23 | (MSB) | | | | | | | |
| 24 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 84 for the CREATE OPERATIONAL IQ response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 84 for the CREATE OPERATIONAL IQ response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

If the STATUS field is set to GOOD, then:

a) the IQ PI OFFSET field indicates the offset in PQI device memory space of the operational IQ PI (i.e., the IQ PI address is the memory address ~~defined~~contained ~~by~~in the first PCI memory BAR plus the IQ PI OFFSET field); and
b) the IQ PI OFFSET field shall be a multiple of four (i.e, byte 0 bit 0 set to zero and byte 0 bit 1 set to zero).

If the STATUS field is not set to GOOD, then the IQ PI OFFSET field is invalid.

### 9.2.5 CREATE OPERATIONAL OQ function

#### 9.2.5.1 CREATE OPERATIONAL OQ request

The CREATE OPERATIONAL OQ function requests that the PQI device create a new operational OQ.

Table 85 defines the CREATE OPERATIONAL OQ request.

**Table 85 — CREATE OPERATIONAL OQ request** (part 1 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (11h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| 12 | OQ ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | RsvdC | | | | | | | |
| 15 | | | | | | | | |
| 16 | OQ ELEMENT ARRAY ADDRESS | (LSB) | RsvdC | | | | | |
| ••• | | | | | | | | |
| 23 | (MSB) | | | | | | | |
| 24 | OQ PI ADDRESS | | | | | (LSB) | RsvdC | |
| ••• | | | | | | | | |
| 31 | (MSB) | | | | | | | |
| 32 | NUMBER OF ELEMENTS | | | | | | | (LSB) |
| 33 | (MSB) | | | | | | | |
| 34 | ELEMENT LENGTH | | | | | | | (LSB) |
| 35 | (MSB) | | | | | | | |
| 36 | Reserved | | | OPERATIONAL QUEUE PROTOCOL | | | | |
| 37 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 39 | | | | | | | | |

**Table 85 — CREATE OPERATIONAL OQ request** (part 2 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ~~36~~40 | INTERRUPT MESSAGE NUMBER | | | | | | | (LSB) |
| ~~37~~41 | WAIT FOR REARM | RsvdC | | | | (MSB) | | |
| ~~40~~42 | COALESCING COUNT | | | | | | | (LSB) |
| ~~41~~43 | (MSB) | | | | | | | |
| 44 | MINIMUM COALESCING TIME | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |
| 48 | MAXIMUM COALESCING TIME | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 51 | (MSB) | | | | | | | |
| ~~44~~ | ~~MAXIMUM COALESCING TIME~~ | | | | | | | ~~(LSB)~~ |
| ~~45~~ | ~~(MSB)~~ | | | | | | | |
| ~~38~~ | ~~RsvdC~~ | | | | | | | ~~WAIT FOR REARM~~ |
| ~~39~~ | ~~RsvdC~~ | | | | | | | |
| ~~42~~ | ~~MINIMUM COALESCING TIME~~ | | | | | | | ~~(LSB)~~ |
| ~~43~~ | ~~(MSB)~~ | | | | | | | |
| ~~46~~ | ~~Reserved~~ | | | | ~~OPERATIONAL QUEUE PROTOCOL~~ | | | |
| ~~47~~52 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 59 | | | | | | | | |
| 60 | Vendor specific | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set as shown in table 85 for the CREATE OPERATIONAL OQ request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 85 for the CREATE OPERATIONAL OQ request.

The OQ ID field specifies the OQ ID to be assigned to the operational OQ. If:

a) the OQ ID field is set to 0000h;

b) the OQ ID field is set to a value that is already assigned to an operational OQ; or
c) the OQ ID field is set to a value greater than the MAXIMUM OPERATIONAL OQS field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3),

then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The OQ ELEMENT ARRAY ADDRESS field specifies the upper 58 bits of the 64-bit operational OQ element array address (see 4.3.2.1). The lower bits of the operational OQ element array address are zero. If the OQ ELEMENT ARRAY ADDRESS field is set to a value that does not meet the alignment requirement indicated by the OPERATIONAL OQ ELEMENT ARRAY ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The OQ PI ADDRESS field specifies the upper 62 bits of the 64-bit operational OQ PI address (see 4.3.2.3). The lower bits of the operational OQ PI address are zero. If the OQ PI ADDRESS field is set to a value that does not meet the alignment requirement indicated by the OPERATIONAL OQ PI ADDRESS ALIGNMENT EXPONENT field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The NUMBER OF ELEMENTS field specifies the number of elements in the operational OQ element array (see 4.3.2.3). If:

a) the NUMBER OF ELEMENTS field is set to a value less than 0002h; or
b) the NUMBER OF ELEMENTS field is set to a value greater than the MAXIMUM OPERATIONAL OQ ELEMENTS field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3),

then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The ELEMENT LENGTH field specifies the element length in 16-byte increments (e.g., a value of 01h in the ELEMENT LENGTH field specifies an element length of 16 bytes). If the ELEMENT LENGTH field is set to a value:

a) less than the MINIMUM OPERATIONAL OQ ELEMENT LENGTH field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3); or
b) greater than the MAXIMUM OPERATIONAL OQ ELEMENT LENGTH field in the REPORT PQI DEVICE CAPABILITY parameter data,

then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

The OPERATIONAL QUEUE PROTOCOL field (see table 83) indicates the information unit layer protocol used by the operational queue (see 4.3.1).

The INTERRUPT MESSAGE NUMBER field specifies the MSI-X Table entry used to generate the interrupt message for ~~OQ PI updates to this OQ~~operational OQ PI updates if MSI-X is enabled (see 4.4.2). This field is not used in legacy INTx mode (see 4.4.3) or polled mode (see 4.4.4). If the INTERRUPT MESSAGE NUMBER field is set to value greater than the Table Size field in the Message Control register in the MSI-X Capability Structure (see PCI), then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

NOTE 28 - Since the Table Size ~~register~~field is 11 bits, the maximum interrupt message number that the PQI device is capable of supporting is less than or equal to 2 047.

The WAIT FOR REARM bit, the COALESCING COUNT field, the MINIMUM COALESCING TIME field, and the MAXIMUM COALESCING TIME field control interrupt coalescing in MSI-X mode (see 4.4.2).

If:

a) the CIC bit is set to one in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3);
b) one or more operational OQs already exist; and
c) one or more of the following conditions is true:
   A) the COALESCING COUNT field is set to a value that is not already in use for the Coalescing Count attribute by the existing operational OQs;

B) the MINIMUM COALESCING TIME field is set to a value that is not already in use for the Minimum Coalescing Time attribute by the existing operational OQs;
C) the MAXIMUM COALESCING TIME field is set to a value that is not already in use for the Maximum Coalescing Time attribute by the existing operational OQs; or
D) the WAIT FOR REARM bit is set to a value that is not equal to the value of the Wait For Rearm attribute for the existing operational OQs,

then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

~~A WAIT FOR REARM bit set to zero specifies that the interrupt coalescing timer is reset and started upon the sending of the prior interrupt.~~ A WAIT FOR REARM bit set to one specifies that the interrupt coalescing timer is reset and started upon receipt of the REARM INTERRUPT bit set to one (see 4.4.2). A WAIT FOR REARM bit set to zero specifies that the interrupt coalescing timer is reset and started upon the sending of the prior interrupt. The WAIT FOR REARM bit may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

The COALESCING COUNT field specifies a number of ~~valid~~occupied operational OQ element array entries used for interrupt coalescing (see 4.4.2). The COALESCING COUNT field may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

The MINIMUM COALESCING TIME field specifies a minimum coalescing time in 100 ns intervals (see 4.4.2). If the MINIMUM COALESCING TIME field is greater than the MAXIMUM COALESCING TIME field, then the minimum coalescing time shall be set to zero. If the MINIMUM COALESCING TIME field is set to a value other than zero that is not a multiple of INTERRUPT COALESCING TIME GRANULARITY field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then the minimum coalescing time in the PQI device shall be rounded up to the next multiple of INTERRUPT COALESCING TIME GRANULARITY field (e.g., if the MINIMUM COALESCING TIME field is set to 00000015h (i.e., 21) and the INTERRUPT COALESCING TIME GRANULARITY field is set to 000Ah (i.e.,10), then the minimum coalescing time shall be rounded up to 0000001Eh (i.e., 30)). The MINIMUM COALESCING TIME field may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

The MAXIMUM COALESCING TIME field specifies a maximum coalescing time in 100 ns intervals (see 4.4.2). If the MAXIMUM COALESCING TIME field is set to a value other than zero that is not a multiple of INTERRUPT COALESCING TIME GRANULARITY field in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then the maximum coalescing time in the PQI device shall be rounded up to the next multiple of INTERRUPT COALESCING TIME GRANULARITY field (e.g., if the MAXIMUM COALESCING TIME field is set to 00000035h (i.e., 53) and the INTERRUPT COALESCING TIME GRANULARITY field is set to 0000000Ah (i.e., 10), then the maximum coalescing time shall be rounded up to 003Ch (i.e., 60)). The MAXIMUM COALESCING TIME field may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

~~The OPERATIONAL QUEUE PROTOCOL field (see table 83) indicates the information unit layer protocol used by the operational queue (see 4.3.1).~~

### 9.2.5.2 CREATE OPERATIONAL OQ response

Table 86 defines the CREATE OPERATIONAL OQ response.

**Table 86 — CREATE OPERATIONAL OQ response**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (11h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | OQ CI OFFSET | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 23 | (MSB) | | | | | | | |
| 24 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 86 for the CREATE OPERATIONAL OQ response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 86 for the CREATE OPERATIONAL OQ response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

If the STATUS field is set to GOOD, then:

a) the OQ CI OFFSET field indicates the offset in PQI device memory space of the operational OQ CI (i.e., the operational OQ CI address is the memory address ~~defined~~contained by the first PCI memory BAR plus the OQ CI OFFSET field); and
b) byte 0 bit 0 and byte 0 bit 1 of the OQ CI OFFSET field shall each be set to zero.

If the STATUS is not set to GOOD, then the OQ CI OFFSET field is invalid.

### 9.2.6 DELETE OPERATIONAL IQ function

#### 9.2.6.1 DELETE OPERATIONAL IQ request

The DELETE OPERATIONAL IQ function requests that the PQI device delete the specified operational IQ.

See 4.3.4.3 for information on when it is safe to delete an operational IQ.Table 87 defines the DELETE OPERATIONAL IQ request.

**Table 87 — DELETE OPERATIONAL IQ request**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (12h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| 12 | IQ ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set as shown in table 87 for the DELETE OPERATIONAL IQ function requests.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 87 for the DELETE OPERATIONAL IQ function requests.

The IQ ID field specifies the IQ ID (see 4.3.6.8.2) of the operational IQ to be deleted. If the IQ ID field is set to an IQ ID that is not assigned, then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

### 9.2.6.2 DELETE OPERATIONAL IQ response

Table 88 defines the DELETE OPERATIONAL IQ response.

Table 88 — DELETE OPERATIONAL IQ response

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (12h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 88 for the DELETE OPERATIONAL IQ response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 88 for the DELETE OPERATIONAL IQ response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

### 9.2.7 DELETE OPERATIONAL OQ function

#### 9.2.7.1 DELETE OPERATIONAL OQ request

The DELETE OPERATIONAL OQ function requests that the PQI device delete the specified operational OQ.

See 4.3.4.3 for information on when it is safe to delete an operational OQ.

Table 89 defines the DELETE OPERATIONAL OQ request.

**Table 89 — DELETE OPERATIONAL OQ request**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | | IU LENGTH (003Ch) | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | REQUEST IDENTIFIER | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (13h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| 12 | | OQ ID | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set as shown in table 89 for the DELETE OPERATIONAL OQ request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 89 for the DELETE OPERATIONAL OQ request.

The OQ ID field specifies the OQ ID (see 4.3.6.11.2) of the operational OQ to be deleted. If the OQ ID field is set to an OQ ID that is not assigned, then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

### 9.2.7.2 DELETE OPERATIONAL OQ response

Table 90 defines the DELETE OPERATIONAL OQ response.

**Table 90 — DELETE OPERATIONAL OQ response**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 ••• 7 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (13h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 ••• 15 | Additional status descriptor | | | | | | | |
| 16 ••• 63 | Reserved | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 90 for the DELETE OPERATIONAL OQ response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 90 for the DELETE OPERATIONAL OQ response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

## 9.2.8 CHANGE OPERATIONAL IQ PROPERTIES function

### 9.2.8.1 CHANGE OPERATIONAL IQ PROPERTIES request

The CHANGE OPERATIONAL IQ PROPERTIES function requests that the PQI device change the properties of the specified operational IQ.

Table 91 defines the CHANGE OPERATIONAL IQ PROPERTIES request.

**Table 91 — CHANGE OPERATIONAL IQ PROPERTIES request**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (14h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| 12 | IQ ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 59 | | | | | | | | |
| 60 | Vendor specific | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set as shown in table 91 for the CHANGE OPERATIONAL IQ PROPERTIES request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 91 for the CHANGE OPERATIONAL IQ PROPERTIES request.

The IQ ID field specifies the IQ ID (see 4.3.6.8.2) of the operational IQ to be configured. If the IQ ID field is set to an IQ ID that is not assigned, then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

### 9.2.8.2 CHANGE OPERATIONAL IQ PROPERTIES response

Table 92 defines the CHANGE OPERATIONAL IQ PROPERTIES response.

**Table 92 — CHANGE OPERATIONAL IQ PROPERTIES response**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (14h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 92 for the CHANGE OPERATIONAL IQ PROPERTIES response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 92 for the CHANGE OPERATIONAL IQ PROPERTIES response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

## 9.2.9 CHANGE OPERATIONAL OQ PROPERTIES function

### 9.2.9.1 CHANGE OPERATIONAL OQ PROPERTIES request

The CHANGE OPERATIONAL OQ PROPERTIES function requests that the PQI device change the properties of the specified operational OQ.

Table 93 defines the CHANGE OPERATIONAL OQ PROPERTIES request.

**Table 93 — CHANGE OPERATIONAL OQ PROPERTIES request** (part 1 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (15h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| 12 | OQ ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| ~~17~~35 | | | | | | | | |
| 36 | Reserved | | | OPERATIONAL QUEUE PROTOCOL | | | | |
| 37 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 40 | | | | | | | | |
| 41 | WAIT FOR REARM | RsvdC | | | | | | |
| ~~20~~42 | COALESCING COUNT | | | | | | | (LSB) |
| ~~21~~43 | (MSB) | | | | | | | |
| 44 | MINIMUM COALESCING TIME | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |
| 48 | MAXIMUM COALESCING TIME | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 51 | (MSB) | | | | | | | |

Table 93 — CHANGE OPERATIONAL OQ PROPERTIES request (part 2 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ~~18~~ | ~~RsvdC~~ | | | | | | | ~~WAIT FOR REARM~~ |
| ~~19~~ | ~~RsvdC~~ | | | | | | | |
| ~~22~~ | ~~MINIMUM COALESCING TIME~~ | | | | | | | ~~(LSB)~~ |
| ~~23~~ | ~~(MSB)~~ | | | | | | | |
| ~~24~~ | ~~MAXIMUM COALESCING TIME~~ | | | | | | | ~~(LSB)~~ |
| ~~25~~ | ~~(MSB)~~ | | | | | | | |
| ~~26~~52 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 59 | | | | | | | | |
| 60 | Vendor specific | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and shall be set as shown in table 93 for the CHANGE OPERATIONAL OQ PROPERTIES function.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 93 for the CHANGE OPERATIONAL OQ PROPERTIES function.

If the CIC bit is set to zero in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then:

a) the OQ ID field specifies the OQ ID (see 4.3.6.11.2) of the operational OQ to be configured; and
b) if the OQ ID field is set to an OQ ID that is not assigned, then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3).

If the CIC bit is set to one in the REPORT PQI DEVICE CAPABILITY parameter data (see 9.2.2.3), then:

a) ~~If~~if no operational OQs exist, then the PQI management device server shall return an administrator response IU with the STATUS field set to INVALID FIELD IN REQUEST IU (see 9.1.4.3); or
b) ~~If~~if one or more operational OQs exists, then:
    A) the OQ ID field shall be ignored;
    B) the Coalescing Count attribute for all operational OQs shall be set to the value of the COALESCING COUNT field;
    C) the Minimum Coalescing Time attribute for all operational OQs shall be set to the value of the MINIMUM COALESCING TIME field;
    D) the Maximum Coalescing Time attribute for all operational OQs shall be set to the value of the MAXIMUM COALESCING TIME field; and
    E) the Wait For Rearm attribute for all operational OQs shall be set to value of the WAIT FOR REARM bit.

The OPERATIONAL QUEUE PROTOCOL field (see table 83) indicates the information unit layer protocol used by the operational queue (see 4.3.1).

The WAIT FOR REARM bit, the COALESCING COUNT field, the MINIMUM COALESCING TIME field, and the MAXIMUM COALESCING TIME field are defined in the CREATE OPERATIONAL OQ request (see 9.2.5.1).

### 9.2.9.2 CHANGE OPERATIONAL OQ PROPERTIES response

Table 94 defines the CHANGE OPERATIONAL OQ PROPERTIES response.

**Table 94 — CHANGE OPERATIONAL OQ PROPERTIES response**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | | IU LENGTH (003Ch) | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | REQUEST IDENTIFIER | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (15h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 94 for the CHANGE OPERATIONAL OQ PROPERTIES response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 94 for the CHANGE OPERATIONAL OQ PROPERTIES response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

### 9.2.10 REPORT OPERATIONAL IQ LIST function

#### 9.2.10.1 REPORT OPERATIONAL IQ LIST request

The REPORT IQ LIST function requests that the PQI device return a list of all existing operational IQs and their properties in the Data-In Buffer as defined in 9.2.10.3.

Table 95 defines the REPORT OPERATIONAL IQ LIST request.

Table 95 — REPORT OPERATIONAL IQ LIST request

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (16h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 43 | | | | | | | | |
| 44 | DATA-IN BUFFER SIZE | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |
| 48 | SGL descriptor (see table 53) | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~IU header for administrator requests (see 9.1.3) and shall be set as shown in table 95 for the REPORT OPERATIONAL IQ LIST request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 95 for the REPORT OPERATIONAL IQ LIST request.

The DATA-IN BUFFER SIZE field is defined in 9.1.3.

The SGL descriptor is defined in 9.1.3.

### 9.2.10.2 REPORT OPERATIONAL IQ LIST response

Table 96 defines the REPORT OPERATIONAL IQ LIST response.

**Table 96 — REPORT OPERATIONAL IQ LIST response**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (16h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 96 for the REPORT OPERATIONAL IQ LIST response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 96 for the REPORT OPERATIONAL IQ LIST response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

### 9.2.10.3 REPORT OPERATIONAL IQ LIST parameter data

The format of the parameter data returned in the Data-In Buffer is shown in table 97.

**Table 97 — REPORT OPERATIONAL IQ LIST parameter data (i.e., Data-In Buffer contents)**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | NUMBER OF OPERATIONAL IQ PROPERTY DESCRIPTORS | | | | | | | (LSB) |
| 7 | (MSB) | | | | | | | |
| Operational IQ property descriptor list | | | | | | | | |
| 8 | Operational IQ property descriptor [first] (see table 98) | | | | | | | |
| ••• | | | | | | | | |
| 135 | | | | | | | | |
| ••• | ••• | | | | | | | |
| m-127 | Operational IQ property descriptor [last] (see table 98) | | | | | | | |
| ••• | | | | | | | | |
| m | | | | | | | | |

The NUMBER OF OPERATIONAL IQ PROPERTY DESCRIPTORS field indicates the number of operational IQ property descriptors in the operational IQ property descriptor list.

Table 98 defines the operational IQ property descriptor.

**Table 98 — Operational IQ property descriptor** (part 1 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | |
| 12 | IQ ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | Reserved | | | | | | | IQ ERROR |
| 15 | Reserved | | | | | | | |
| 16 | IQ ELEMENT ARRAY ADDRESS | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 23 | (MSB) | | | | | | | |

**Table 98 — Operational IQ property descriptor** (part 2 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 24 | IQ CI ADDRESS | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 31 | (MSB) | | | | | | | |
| 32 | NUMBER OF ELEMENTS | | | | | | | (LSB) |
| 33 | (MSB) | | | | | | | |
| 34 | ELEMENT LENGTH | | | | | | | (LSB) |
| 35 | (MSB) | | | | | | | |
| 36 | Reserved | | | OPERATIONAL QUEUE PROTOCOL | | | | |
| 37 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 59 | | | | | | | | |
| 60 | Vendor specific | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |
| Bytes corresponding to the CREATE OPERATIONAL IQ response | | | | | | | | |
| 64 | IQ PI OFFSET | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 71 | (MSB) | | | | | | | |
| 72 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 127 | | | | | | | | |

The IQ ID field indicates the IQ ID (see 4.3.6.8.2) of the operational IQ being described.

~~An IQ ERROR bit set to one indicates that the operational IQ is in an error condition and the information unit layer has stopped consuming from the operational IQ (see the appropriate information unit standard). An IQ ERROR bit set to zero indicates the operational IQ is operating without error.~~ An IQ ERROR bit set to one indicates that the PQI device has stopped consuming from the operational IQ due to an error. An IQ ERROR bit set to zero indicates that the PQI device has not stopped consuming from the operational IQ due to an error.

The IQ ELEMENT ARRAY ADDRESS field indicates the operational IQ element array address.

The IQ CI ADDRESS field indicates the operational IQ CI address.

The NUMBER OF ELEMENTS field indicates the number of elements in the operational IQ element array.

The ELEMENT LENGTH field indicates the element length in 16-byte increments (e.g., a value of 01h in the ELEMENT LENGTH field specifies an element length of 16 bytes).

The OPERATIONAL QUEUE PROTOCOL field (see table 83) indicates the information unit layer protocol used by the operational ~~queue~~IQ (see 4.3.1).

The IQ PI OFFSET field indicates the offset in PQI device memory space of the operational IQ PI (i.e., the operational IQ PI address is the memory address ~~defined~~contained ~~by~~in the first PCI memory BAR plus the IQ PI OFFSET field).

### 9.2.11 REPORT OPERATIONAL OQ LIST function

#### 9.2.11.1 REPORT OPERATIONAL OQ LIST request

The REPORT OPERATIONAL OQ LIST function requests that the PQI device return a list of all existing operational OQs and their properties in the Data-In Buffer as defined in 9.2.11.3.

Table 99 defines the REPORT OPERATIONAL OQ LIST request.

**Table 99 — REPORT OPERATIONAL OQ LIST request**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (60h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | | IU LENGTH (003Ch) | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for inbound IUs (see table 61) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | REQUEST IDENTIFIER | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (17h) | | | | | | | |
| 11 | RsvdC | | | | | | | |
| ••• | | | | | | | | |
| 43 | | | | | | | | |
| 44 | | DATA-IN BUFFER SIZE | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |
| 48 | SGL descriptor (see table 53) | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator requests (see 9.1.3) and are set to the values shown in table 99 for the REPORT OPERATIONAL OQ LIST request.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.3.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 99 for the REPORT OPERATIONAL OQ LIST request.

The DATA-IN BUFFER SIZE field is defined in 9.1.3.

The SGL descriptor is defined in 9.1.3.

### 9.2.11.2 REPORT OPERATIONAL OQ LIST response

Table 100 defines the REPORT OPERATIONAL OQ LIST response.

Table 100 — REPORT OPERATIONAL OQ LIST response

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | IU TYPE (E0h) | | | | | | | |
| 1 | COMPATIBLE FEATURES (00h) | | | | | | | |
| 2 | IU LENGTH (003Ch) | | | | | | | (LSB) |
| 3 | (MSB) | | | | | | | |
| 4 | Queuing interface ~~specific~~ descriptor for outbound IUs (see table 62) | | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | REQUEST IDENTIFIER | | | | | | | (LSB) |
| 9 | (MSB) | | | | | | | |
| 10 | FUNCTION CODE (17h) | | | | | | | |
| 11 | STATUS | | | | | | | |
| 12 | Additional status descriptor | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| 16 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The IU TYPE field, the COMPATIBLE FEATURES field, and the IU LENGTH field are part of the ~~PQI~~ IU header for administrator responses (see 9.1.4) and shall be set as shown in table 100 for the REPORT OPERATIONAL OQ LIST response.

The queuing interface ~~specific~~ descriptor is defined in 8.3.

The REQUEST IDENTIFIER field is defined in 9.1.4.1.

The FUNCTION CODE field is defined in ~~table 74~~9.2.1 and is set to the value shown in table 100 the REPORT OPERATIONAL OQ LIST response.

The STATUS field and the additional status descriptor are defined in 9.1.4.

### 9.2.11.3 REPORT OPERATIONAL OQ LIST parameter data

The format of the parameter data returned in the Data-In Buffer is shown in table 101.

**Table 101 — REPORT OF OPERATIONAL OQ LIST parameter data (i.e., Data-In Buffer contents)**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | NUMBER OF OPERATIONAL OQ PROPERTY DESCRIPTORS | | | | | | | (LSB) |
| 7 | (MSB) | | | | | | | |
| Operational OQ property descriptor list | | | | | | | | |
| 8 | Operational OQ property descriptor [first] (see table 102) | | | | | | | |
| ••• | | | | | | | | |
| 135 | | | | | | | | |
| ••• | ••• | | | | | | | |
| m-127 | Operational OQ property descriptor [last] (see table 102) | | | | | | | |
| ••• | | | | | | | | |
| m | | | | | | | | |

The NUMBER OF OPERATIONAL OQ PROPERTY DESCRIPTORS field indicates the number of operational OQ property descriptors in the operational OQ property descriptor list.

Table 102 defines the operational OQ property descriptor.

**Table 102 — Operational OQ property descriptor** (part 1 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | |
| 12 | OQ ID | | | | | | | (LSB) |
| 13 | (MSB) | | | | | | | |
| 14 | Reserved | | | | | | | OQ ERROR |
| 15 | Reserved | | | | | | | |
| 16 | OQ ELEMENT ARRAY ADDRESS | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 23 | (MSB) | | | | | | | |
| 24 | OQ PI ADDRESS | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 31 | (MSB) | | | | | | | |
| 32 | NUMBER OF ELEMENTS | | | | | | | (LSB) |
| 33 | (MSB) | | | | | | | |
| 34 | ELEMENT LENGTH | | | | | | | (LSB) |
| 35 | (MSB) | | | | | | | |
| 36 | Reserved | | OPERATIONAL QUEUE PROTOCOL | | | | | |
| 37 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 39 | | | | | | | | |
| ~~36~~40 | INTERRUPT MESSAGE NUMBER | | | | | | | (LSB) |
| ~~37~~41 | WAIT FOR REARM | RevdC | | | | (MSB) | | |
| ~~40~~42 | COALESCING COUNT | | | | | | | (LSB) |
| ~~41~~43 | (MSB) | | | | | | | |
| 44 | MINIMUM COALESCING TIME | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 47 | (MSB) | | | | | | | |

Table 102 — Operational OQ property descriptor (part 2 of 2)

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 48 | MAXIMUM COALESCING TIME | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 51 | (MSB) | | | | | | | |
| ~~38~~ | ~~Reserved~~ | | | | | | | ~~WAIT FOR REARM~~ |
| ~~39~~ | ~~Reserved~~ | | | | | | | |
| ~~42~~ | ~~MINIMUM COALESCING TIME~~ | | | | | | | (LSB) |
| ~~43~~ | (MSB) | | | | | | | |
| ~~44~~ | ~~MAXIMUM COALESCING TIME~~ | | | | | | | (LSB) |
| ~~45~~ | (MSB) | | | | | | | |
| ~~46~~ | ~~Reserved~~ | | | ~~OPERATIONAL QUEUE PROTOCOL~~ | | | | |
| ~~47~~52 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 59 | | | | | | | | |
| 60 | Vendor specific | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |
| Bytes corresponding to the CREATE OPERATIONAL OQ response | | | | | | | | |
| 64 | OQ CI OFFSET | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 71 | (MSB) | | | | | | | |
| 72 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 127 | | | | | | | | |

The OQ ID field indicates the OQ ID (see 4.3.6.11.2) of the operational OQ being described.

~~An OQ ERROR bit set to one indicates that the operational OQ is in an error condition and the information unit layer has stopped producing to the operational OQ (see the appropriate information unit standard). An OQ ERROR bit set to zero indicates the OQ is operating without error.~~ An OQ ERROR bit set to one indicates that the PQI device has stopped producing to the operational OQ due to an error. An OQ ERROR bit set to zero indicates that the PQI device has not stopped producing to the operational OQ due to an error.

The OQ ELEMENT ARRAY ADDRESS field indicates the operational OQ element array address.

The OQ PI ADDRESS field indicates the operational QQ PI address.

The NUMBER OF ELEMENTS field indicates the number of elements in the operational OQ element array.

The ELEMENT LENGTH field indicates the ~~element length in 16-byte increments~~ length in 16-byte increments of elements in the operational OQ (e.g., a value of 01h in the ELEMENT LENGTH field specifies an element length of 16 bytes).

The OPERATIONAL QUEUE PROTOCOL field (see table 83) indicates the information unit layer protocol used by the operational queue (see 4.3.1).

The INTERRUPT MESSAGE NUMBER field indicates the MSI-X Table entry used to generate the interrupt message for ~~OQ PI updates to this operational OQ~~updates to the operational OQ PI if MSI-X is enabled (see PCI).

The WAIT FOR REARM bit, the COALESCING COUNT field, the MINIMUM COALESCING TIME field, and the MAXIMUM COALESCING TIME field ~~indicate the~~ control interrupt coalescing parameters in MSI-X mode (see 4.4.2.3).

A WAIT FOR REARM bit set to zero indicates that the interrupt coalescing timer is reset and started upon the sending of the prior interrupt. A WAIT FOR REARM bit set to one indicates that the interrupt coalescing timer is reset and started upon receipt of the REARM INTERRUPT bit set to one (see 4.4.2). The WAIT FOR REARM bit is specified when the operational OQ is created using the CREATE OPERATIONAL OQ PROPERTIES function (see 9.2.5). The WAIT FOR REARM bit may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

The COALESCING COUNT field indicates a number of valid operational OQ element array entries used for interrupt coalescing (see 4.4.2). The COALESCING COUNT field is specified when the operational OQ is created using the CREATE OPERATIONAL OQ PROPERTIES function (see 9.2.5). The COALESCING COUNT field may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

The MINIMUM COALESCING TIME field indicates ~~that~~ the minimum coalescing time, in 100 ns intervals. A MINIMUM COALESCING TIME field set to 0000FFFFh indicates the time is 6 553 500 ns or greater. The MINIMUM COALESCING TIME field is specified when the operational OQ is created using the CREATE OPERATIONAL OQ PROPERTIES function (see 9.2.5). The MINIMUM COALESCING TIME field may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

The MAXIMUM COALESCING TIME field indicates ~~that~~ the maximum coalescing time, in 100 ns intervals. A MAXIMUM COALESCING TIME field set to 0000FFFFh indicates the time is 6 553 500 ns or greater. The MAXIMUM COALESCING TIME field is specified when the operational OQ is created using the CREATE OPERATIONAL OQ PROPERTIES function (see 9.2.5). The MAXIMUM COALESCING TIME field may be changed using the CHANGE OPERATIONAL OQ PROPERTIES function (see 9.2.9).

The OQ CI OFFSET field indicates the offset in PQI device memory space of the operational OQ CI (i.e., the operational OQ CI address is the memory address ~~defined~~contained ~~by~~in the first PCI memory BAR plus the OQ CI OFFSET field).

# Annex A
(normative)

## Alternative SGL segment

An alternative SGL segment (see table A.2):

a) is a kind of SGL segment (see 7.2) pointed to by a~~n~~ ~~SGL Alternative Last Segment~~Last Alternative SGL Segment descriptor (see table A.1); and
b) ~~containing~~contains ~~SGL~~ Alternative Data Block descriptors (see table A.3).

The length and format of the ~~SGL~~ Alternative Data Block descriptor differs from that of the SGL descriptors defined in 7.3 (i.e., 20 bytes rather than 16 bytes).

An alternative SGL segment is always a last SGL segment.

Table A.1 defines the ~~SGL Alternative Last Segment~~ Last Alternative SGL Segment descriptor.

**Table A.1 — Last Alternative SGL Segment descriptor**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ADDRESS | | | | | (LSB) | Reserved | |
| ••• | | | | | | | | |
| 7 | (MSB) | | | | | | | |
| 8 | NUMBER OF DESCRIPTORS | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 11 | (MSB) | | | | | | | |
| 12 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 14 | | | | | | | | |
| 15 | SGL DESCRIPTOR TYPE (4h) | | | | Reserved | | | |

~~The ADDRESS field specifies the upper 62 bits of the starting 64-bit memory byte address of the next SGL segment, which is the last SGL segment and is an alternative SGL segment (see table A.2).~~

The ADDRESS field specifies the upper 62 bits of the 64-bit memory space address of the next SGL segment, which is the last SGL segment (see 7.2) and is an alternative SGL segment (see table A.2). The lower bits of the memory space address of the next SGL segment are zero.

The NUMBER OF DESCRIPTORS field specifies the number of ~~SGL~~ Alternative Data Block descriptors in the alternative SGL segment. If the NUMBER OF DESCRIPTORS field is set to zero, then the PQI management device server shall return an administrator response IU with the STATUS field set to DATA BUFFER ERROR (see 9.1.4.1).

If the value in the ADDRESS field plus the value in the NUMBER OF DESCRIPTORS field times 20 (i.e., the length in bytes of each ~~SGL~~ Alternative Data Block descriptor) is greater than 1_00000000_00000000h, then the ~~SGL~~ Alternative Data Block descriptor shall be processed as having an error.

The SGL DESCRIPTOR TYPE field is defined in 7.3.1 and is set to the value shown in table A.1 for the Last Alternative SGL Segment descriptor.

Table A.2 defines the alternative SGL segment.

**Table A.2 — Alternative SGL segment**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ~~SGL~~ Alternative Data Block descriptor [first] (see table A.3) | | | | | | | |
| ••• | | | | | | | | |
| 19 | | | | | | | | |
| ••• | ••• | | | | | | | |
| n - 19 | ~~SGL~~ Alternative Data Block descriptor [last] (see table A.3) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The alternative SGL segment shall contain at least one ~~SGL~~ Alternative Data Block descriptor.

Table A.3 defines the ~~SGL~~ Alternative Data Block descriptor.

**Table A.3 — ~~SGL~~ Alternative Data Block descriptor**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ADDRESS | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 7 | (MSB) | | | | | | | |
| 8 | LENGTH | | | | | | | (LSB) |
| ••• | | | | | | | | |
| 11 | (MSB) | | | | | | | |
| 12 | Vendor specific | | | | | | | |
| ••• | | | | | | | | |
| 19 | | | | | | | | |

The ADDRESS field specifies the starting 64-bit memory byte address of a data block.

The LENGTH field specifies the length in bytes of the data block. A LENGTH field set to 00000000h specifies that no data is transferred. A~~n~~ ~~SGL~~ Alternative Data Block descriptor specifying that no data is transferred shall not be processed as having an error.

If the value in the ADDRESS field plus the value in the LENGTH field is greater than 1_00000000_00000000h, then the ~~SGL~~ Alternative Data Block descriptor shall be processed as having an error.

# Annex B
(informative)

# Operating system suggestions

## B.1 Power actions

If a Windows® operating system Storport miniport driver's **HwStorBuildIo** function is invoked with an Srb parameter pointing to a SCSI_POWER_REQUEST BLOCK (i.e., the Function member is set to SRB_FUNCTION_POWER), then the miniport driver should set the fields in the ~~PQI device's~~PQI Device Power Action register (see 5.2.21) as follows:

a) set the SYSTEM POWER ACTION field as listed in table B.1; and

**Table B.1 — Windows PowerAction member to SYSTEM POWER ACTION field**

| PowerAction member | SYSTEM POWER ACTION field |
|---|---|
| 0 (i.e., StorPowerActionNone) | 00h (i.e., no action) |
| 1 (i.e., StorPowerActionReserved) | 00h (i.e., no action) |
| 2 (i.e., StorPowerActionSleep) | 10h (i.e., S1, S2, or S3) |
| 3 (i.e., StorPowerActionHibernate) | 14h (i.e., S4) |
| 4 (i.e., StorPowerActionShutdown) | 02h (i.e., S0 or S5) |
| 5 (i.e., StorPowerActionShutdownReset) | 01h (i.e., S0) |
| 6 (i.e., StorPowerActionShutdownOff) | 15h (i.e., S5) |
| 7 (i.e., StorPowerActionWarmEject) | 00h (i.e., no action) |

b) set the DEVICE POWER ACTION field as listed in table B.2.

**Table B.2 — Windows DevicePowerState member to DEVICE POWER ACTION field**

| DevicePowerState member | DEVICE POWER ACTION field |
|---|---|
| 0 (i.e., StorPowerDeviceUnspecified) | 00h (i.e., no action) |
| 1 (i.e., StorPowerDeviceD0) | 10h (i.e., D0) |
| 2 (i.e., StorPowerDeviceD1) | 11h (i.e., D1) |
| 3 (i.e., StorPowerDeviceD2) | 12h (i.e., D2) |
| 4 (i.e., StorPowerDeviceD3) | 13h (i.e., D3) |

NOTE 29 - Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

NOTE 30 - For information on Windows, see the Microsoft Developer Network web site (see http://msdn.microsoft.com).

## Bibliography

ISO/IEC 14776-323, *SCSI Block Commands - 3 (SBC-3) (T10/1799-D)*

ISO/IEC 14776-333, *SCSI Stream Commands - 3 (SSC-3) (T10/1611-D)*

ISO/IEC 14776-262, *SAS Protocol Layer - 2 (SPL-2) (T10/2228-D)*

ISO/IEC 14776-154, *Serial Attached SCSI - 3 (SAS-3) (T10/2212-D)*