XR9240

Compression and Security Coprocessor

Data Sheet

Licensing and Government Use

Any Exar software ("Licensed Programs") based on Hifn Technology described in this document is furnished under a license and may be used and copied only in accordance with the terms of such license and with the inclusion of this copyright notice. Distribution of this document or any copies thereof and the ability to transfer title or ownership of this document's contents are subject to the terms of such license.

Such Licensed Programs and their documentation may contain public open-source software that would be licensed under open-source licenses. Refer to the applicable product release notes for open-source licenses and proprietary notices. Use, duplication, disclosure, and acquisition by the U.S. Government of such Licensed Programs is subject to the terms and definitions of their applicable license.

Disclaimer

Exar reserves the right to make changes to its products, including the contents of this document, or to discontinue any product or service without notice. Exar advises its customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied upon is current. Every effort has been made to keep the information in this document current and accurate as of the date of this document's publication or revision.

Limited Warranty

Exar warrants Products based on the Hifn Technology, including cards, against defects in materials and workmanship for a period of twelve (12) months from the delivery date. Exar's sole liability shall be limited to either, replacing, repairing or issuing credit, at its option, for the Product if it has been paid for. Exar will not be liable under this provision unless: (a) Exar is promptly notified in writing upon discovery of claimed defects by Buyer; (b) The claimed defective Product is returned to Exar, insurance and transportation charges prepaid, by Buyer; (c) The claimed defective Product is received within twelve (12) months from the delivery date; and (d) Exar's examination of the Product discloses to its satisfaction that the alleged defect was not caused by misuse, neglect, improper installation, repair, alteration, accident or other hazard. THIS WARRANTY DOES NOT COVER PRODUCT DAMAGE WHICH RESULTS FROM ACCIDENT, MISUSE, ABUSE, IMPROPER LINE VOLTAGE, FIRE, FLOOD, LIGHTNING OR OTHER ACTS OF GOD OR DAMAGE RESULTING FROM ANY MODIFICATIONS, REPAIRS OR ALTERATIONS PERFORMED OTHER THAN BY EXAR OR EXAR'S AUTHORIZED AGENT OR RESULTING FROM FAILURE TO STRICTLY COMPLY WITH EXAR'S WRITTEN OPERATING AND MAINTENANCE INSTRUCTIONS. BUYER ACKNOWLEDGES THAT THE PRODUCT ARE HIGHLY SENSITIVE ELECTRONIC PRODUCT REQUIRING SPECIAL HANDLING AND THAT THIS WARRANTY DOES NOT APPLY TO IMPROPERLY HANDLED PRODUCT. PRODUCT MANUFACTURED TO MEET BUYER'S SPECIFIC PERFORMANCE SPECIFICATIONS ACCEPTED BY EXAR ARE WARRANTED ONLY TO PERFORM IN CONFORMITY WITH SUCH SPECIFICATIONS, AND ARE WARRANTED ONLY AGAINST DEFECTS NOT RELATED TO SUCH SPECIFICATIONS IN ACCORDANCE WITH THE TERMS AND CONDITIONS SET FORTH HEREIN ABOVE.

Life Support Policy

Exar's Product are not authorized for use as critical components in life support devices or systems. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury or death to human life. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness. Buyer agrees to indemnify, defend and hold Exar harmless for any cost, loss, liability, or expense (including without limitation attorneys' fees and other costs of litigation or threatened litigation) arising out of violation of the above prohibition by Buyer or any person or entity receiving Exar's Product through Buyer.

Patent Infringement - Indemnification

Exar agrees, at its own expense, to defend Buyer from and against any claim, suit or proceeding, and to pay all judgments and costs finally awarded against Buyer by reason of claim, suit or proceeding insofar as it is based upon an allegation that the Product as furnished by Exar infringes any United States letter patent, provided that Exar is notified promptly of such claim in writing and is given authority and full and proper information and assistance (at Exar's expense) for defense of same. In case such Product are finally constituted an infringement and the use of Product is enjoined, Exar shall at its sole discretion and at its own expense: (1) procure for Buyer the right to continue using the Product; (2) replace or modify the same so that it becomes non-infringing; or (3) remove such Product and grant Buyer a credit for the depreciated value of the same.

Buyer shall have the right to employ separate counsel in any claim, suit or proceeding and to participate in the defense thereof, but the fees and expenses of Buyer's counsel shall not be borne by Exar unless: (1) Exar specifically so agrees; or (2) Exar, after written request and without cause, does not assume such defense. Exar shall not be liable to indemnify Buyer for any settlement effected without Exar's written consent, unless Exar failed, after notice and without cause, to defend such claim, suit or proceeding.

The indemnification shall not apply and Buyer shall indemnify Exar and hold it harmless from all liability or expense (including costs of suit and attorney's fees) if the infringement arises from, or is based upon Exar's

compliance with particular requirements of Buyer or Buyer's customer that differ from Exar's standard specifications (Custom Product) for the Product, or modifications or alterations of the Product, or a combination of the Product with other items not furnished or manufactured by Exar.

Buyer agrees that Exar shall not be liable for any collateral, incidental or consequential damages arising out of patent infringement.

The foregoing states the entire liability of Exar for patent infringement.

Motorola

The use of this product in stateful compression protocols (for example, PPP or multi-history applications) with certain configurations may require a license from Motorola. In such cases, a license agreement for the right to use Motorola patents (US05,245,614, US05,130,993) may be obtained directly from Motorola.

Patents

May include one or more of the following United States patents: 4,930,142; 4,996,690; 4,701,745; 5,003,307; 5,016,009; 5,126,739; 5,146,221; 5,414,425; 5,414,850; 5,463,390; 5,506,580; 5,532,694; 6,320,846; 6,816,459; 6,651,099; 6,665,725; 6,771,646; 6,789,116; 6,954,789; 6,839,751; 7,299,282; 7,260,558. Other patents pending.

Trademarks

Hi/fn®, MeterFlow®, MeterWorks®, and LZS®, are registered trademarks of Exar Corporation. Hifn™, Hifn Technology, FlowThrough™, BitWackr, and the Hifn logo are trademarks of Hi/fn, Inc. All other trademarks and trade names are the property of their respective holders.

IBM, IBM Logo, and IBM PowerPC are trademarks of International Business Machines Corporation in the United States, or other countries.

Microsoft, Windows, Windows XP, Windows Vista, Windows Server 2003, Windows Server 2008 and the Windows logo are trademarks of Microsoft Corporation in the United States, and/or other countries.

Intel QuickAssist is a trademark of Intel Corporation in the United States and in other countries.

Exporting

This product may only be exported from the United States in accordance with applicable Export Administration Regulations. Diversion contrary to United States laws is prohibited.

Exar Confidential

If you have signed a Exar Confidential Disclosure Agreement that includes this document as part of its subject matter, please use this document in accordance with the terms of the agreement. If not, please destroy the document.

Table of Contents

List of Figures

List of Tables

Abbreviations

Term	Definition
AAD	Additional Authentication Data
AER	Advanced Error Reporting
AES	Advanced Encryption Standard
ARI	Alternative Routing-ID Interpretation
BAR	PCIe Base Address Registers
BE	Best effort
CAB	Control Access Bus
CBC	Cipher Block Chaining
CM	Channel Manager
CML	Current Mode Logic signal type
COS	Class of Service
CPLD	Complex Programming Logic Device
CPR	Command Pointer Ring
CRC	Cyclic Redundancy Check
CTR	Counter Mode
DES	Data Encryption Standard
DH	Diffie-Hellman key exchange protocol
DPC	Data Process Channel
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
ECB	Electronic Codebook
ECC	Error correction code
ECDH	Elliptic-Curve Diffie-Hellman
ECDSA	Elliptic-Curve Digital Signature Algorithm
ECRC	End-to-End cyclic Redundancy Check
eLZS	Enhanced LZS
ESP	Encapsulating Security Payload
FLR	Function Level Reset
GMAC	Galois Message Authentication Code
GZIP	GNU ZIP
HMAC	Hash Message Authentication Code
IDC	Inbound Data Controller
IDO	ID based Ordering

Term	Definition
IHV	Initial Hash Value
IOV	Input/Output Virtualization
IPAD	Inner Padding
IPPCP	IP Payload Compression Protocol
IPsec	IP Security Protocol
IV	Initial Vector
JTAG	Joint Test Action Group
LVCMOS	Low Voltage CMOS signal type
LZS	Lempel-Ziv Stac
MAC	Message Authentication Code
MSI	Message Signaled Interrupts
ODC	Outbound Data Controller
OPAD	Outer Padding
PF	Physical Function
PHY	Physical-Layer interface - usually for PCI express
PKP	Public Key Processor
PLL	Phase-Locked Loop
PP	Packet Processor
PRF	Pseudo-Random Function
PTP	Post Transform Processing
RC	PCIe Root Complex
RNG	Random Number Generator
RSA	Ron Rivest, Adi Shamir and Leonard Adleman
RTV	Real Time Verification
S0	Initial Value for Authentication Engine GCM-MAC and GMAC calculation
SBOX	Substitution Box
SPI	Serial Peripheral Interface
SSL	Security Socket Layer
TC	Transform Channel
TE	Transform Engine
TLP	Transaction Layer Packet
TLS	Transport Layer Security
TPH	TLP processing hint

Term	Definition
VF	Virtual Function
WDRR	Weighted Deficit Round Robin
XTS	XES-based Tweaked CodeBook mode with Cipher Text Stealing

Preface

Welcome to the Data Sheet for Exar's XR9240 device. The XR9240 is a a high performance, low power, compression/encryption/authentication coprocessor. This document describes the XR9240 operation, data structures, data flow, and signal definition.

Audience

This document is intended for:

- Project managers
- System engineers
- Hardware and software development engineers
- Marketing and product managers

Prerequisite

Before proceeding, you should generally understand:

- PCI Express 3.0
- Compression, encryption, and authentication algorithms
- General networking concepts

Document Organization

This document is organized as follows:

Chapter 1, "Product Description" provides an overview of the XR9240 coprocessor.

Chapter 2, "Operation" describes key features of the XR9240 operations.

Chapter 3, "Data Structures" defines the format of the data structures used by the XR9240.

Chapter 4, "XR9240Command Operation" gives examples of the typical command sequences from the host point of view.

Chapter 5, "Modules" describes the internal XR9240 modules in more detail.

Chapter 6, "PCIe BAR0 Register Definition" details the syntax and usage of the XR9240 registers that are accessed through PCIe BAR0.

Chapter 7, "PCIe BAR3 Register Definition" details the syntax and usage of the XR9240 registers that are accessed through PCIe BAR3.

Chapter 8, "PCIe Configuration Register Definition", defines the XR9240 default values for the PCIe configuration and capability registers.

Chapter 9, "Signal Definition" lists the XR9240 external interfaces.

Chapter 10, "Error Handling" describes the XR9240 error handling features.

Chapter 11, "DC Specifications" defines the XR9240 DC specifications.

Chapter 12, "AC Specifications" defines the XR9240 AC specifications.

Chapter 13, "Thermal Specifications" defines the XR9240 thermal specifications.

Chapter 14, "Package Specifications" defines the XR9240 package specifications.

Appendix A lists the major differences between the previous generation device and the XR9240.

Related Documents

The following documents may be used as a reference to this document.

APN-0001 *XR9240 FPGA Design Guidelines Application Note*

USR-0032 *XR9240 Hardware Design User Guide*

ERR-0005 *XR9240 Hardware Errata*

Customer Support

For technical support about this product, please contact your local Exar sales office, representative, or distributor.

For general information about Exar and Exar products refer to: www.exar.com

1 Product Description

Exar's XR9240 compression and security coprocessor delivers best in class performance, scalability, and flexibility to address the requirements of today's enterprise data analytics, big data, storage and networking applications. The XR9240 provides hardware acceleration of compression, encryption and authentication algorithms including gzip/zlib/Deflate, LZS/ eLZS, AES, 3DES, RC4, SHA, HMAC, GMAC and public key algorithms such as DSA, DH, RSA, ECDSA and ECDH, and is designed to optimize SSL/IPsec/SRTP packet processing. . Real time verification and CRC protection ensure end-to-end data integrity. The XR9240 integrates an eight lane PCIe Gen 3 interface for maximum throughput, and supports Single Root I/O Virtualization (SR-IOV) with 128 virtual functions. Class of service is supported with eight priority queues for compression and bulk crypto, and four priority queues for public key processing.

Figure 1-1 shows a typical XR9240 application example where the XR9240 is employed on a half-height PCIe card. A FPGA may be optionally attached to the XR9240 Interlaken interface to support custom algorithms, or to allow for external pre or post processing relative to the XR9240 hardware algorithm acceleration.

The XR9240 also has a SPI interface that may be used to connect to a non-volatile device to store user information such as card serial and version numbers.

Refer to "Appendix A: Summary of 820x/XR9240 Differences" for more information.

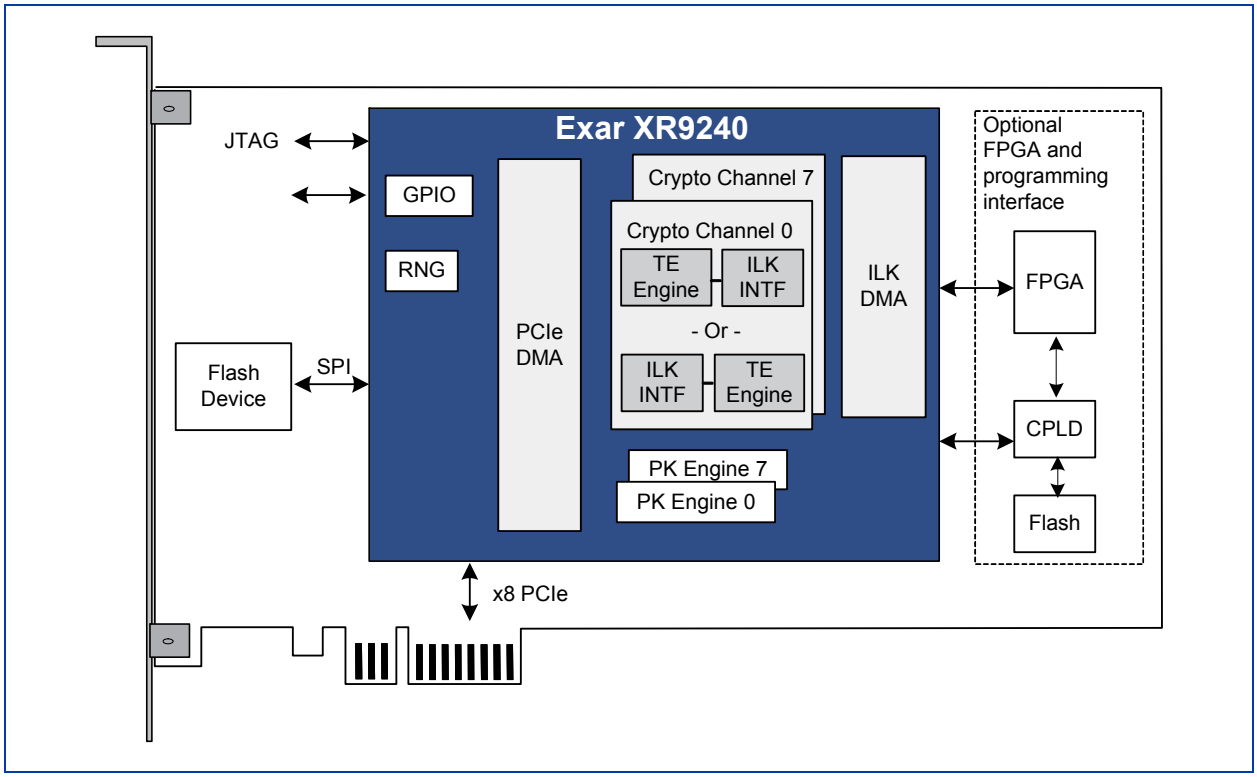


Figure 1-1. Application Example

1.1 Features

1.1.1 High Performance

- Eight channels of pipelined Compression, Encryption, Pad, and Authentication engines provide high performance parallel processing
- Eight Public Key engines accelerate secure key exchanges
- FPGA Interlaken interface allows for custom algorithms or offloads host pre/post-processing
- Random Number Generator
- Compression, encryption, authentication, and padding in a single pass optimizes IPsec and SSL applications
- Multiple algorithm padding for encode operations, and padding removal for decode operations
- Enhanced IV generation algorithm and optimized logic for SSL/IPsec/SRTP packets for networking applications

1.1.2 Engine Features

Table 1-1. XR9240 Engine Features

Engine	Features
Authentication	Algorithms supported: SHA1, SHA224, SHA256, SHA384, SHA512, MD5, HMAC-SHA1, HMAC-224, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512 and HMAC-MD5, GMAC, XCBC-MAC, SSL3.0-MAC, P_hash, CMAC Ability to insert and remove MAC into/from the data stream when not in skip mode Stateful hash and MAC operation
Compression	gzip/zlib/Deflate (stateful and stateless) Complies with RFC 1951 (Deflate), RFC 1952 (gzip) and RFC 1950 (zlib) Supports Static Huffman algorithm Supports stateful compression/decompression Supports Dynamic Huffman algorithm for high compression ratio Industry standard LZS algorithm Enhanced LZS (eLZS) algorithm with anti-expansion compression
Encryption	Algorithms supported: AES-GCM, AES-CBC, AES-CTR, and AES-ECB, AES-F8, AES-XTS, 3DES-CBC Stream cipher: RC4 Supports stateful encryption operations
Wireless applications	UEA1/UIA1 (KASUMI) EEA1/EIA1 (SNOW3G) EEA2/EIA2 (AES) EEA3/EIA3 (ZUC)

Table 1-1. XR9240 Engine Features

Engine	Features
IV generation	Enhanced IV generation algorithm
Pad	Adds padding to multiple algorithms for encode operations Removes padding in decode operations
Public Key	RSA (1K, 2K, 4K, 8K) DSA (1K, 2K, 3K) DH (1K, 2K, 4K, 8K) ECDSA, ECDH (192-521) Supports up to 8K-bits modular arithmetic and exponentiation

1.1.3 PCI Express 3.0 Compatible Bus Interface

The XR9240 host bus interface is compatible with the PCI Express Base Specification, Revision 3.0, and supports the following features:

- 32-bit and 64-bit address support
- x8, x4, x2, and x1 Gen3 link support
- Strong and relaxed transaction ordering rules
- 1KB maximum payload size
- PCI Express message support
- Legacy and MSI/MSIX interrupt support
- Baseline and advanced error reporting support
- IOV with bandwidth allocation scheme support
- Alternative Routing-ID Interpretation (ARI) support
- Function Level Reset (FLR) support
- TLP processing hint (TPH) support
- ID based ordering support
- D0/D3 device power state and L0, L0s, L1, L3 link power state support

1.1.4 Interlaken Expansion Interface

- Attach FPGA for custom algorithm or pre/post-processing
- May be configured with any combination of 1-8 lanes
- Lane speed may be configured to 3.125G or 6.25G
- Aggregate data throughput of up to 40 Gbps (accounting for overhead)
- FPGA image may be programmed through the XR9240 field programmable interface

1.1.5 Data Integrity

- Command-based CRC protection for all operations
- ECRC data protection through PCIe bus (if supported by host root complex)
- Real time verification of compression, encryption and authentication engine operations
- Error handling and reporting mechanism
- Memory ECC protection for internal memories

1.1.6 Other features

- Multiple power domains and static and dynamic clock gating for all processing engines ensures low power consumption
- JTAG support
- ACJTAG support for PCIe interface
- 32-bit GPIO interface
- SPI interface to access a non-volatile device
- Device temperature monitoring
- External fan monitoring
- 896 ball FCBGA package (31 x 31 mm body, 1 mm ball pitch)

1.2 Interlaken Interface Applications

The Interlaken interface provides a communication channel to expand the functionality of the XR9240 through an external FPGA. The FPGA enables applications that could support custom algorithms, pre-processing offload, or look-aside co-processing.

If the user application requires a custom algorithm that is not one of the standard algorithms on the XR9240, that custom algorithm could be implemented in an external FPGA and integrated into the data flow through the Interlaken interface. The custom algorithm may be configured before or after the standard XR9240 algorithms in the datapath.

The Interlaken interface could also be used by a user application to offload host pre/post processing such as packet parsing to the FPGA. In this case, the FPGA preprocesses the data, and then creates or modifies the internal packet header to instruct the XR9240 to operate on the command according to the revised packet header. Using the FPGA to parse an IPsec packet header and form the command packets for the XR9240 is an example of a network application that would benefit from offloading the host pre-processing.

In addition, the XR9240 device has the flexibility to be used as a lookaside coprocessor to the FPGA device. In this scenario, the FPGA would initiate all packet flows over the ILK interface to the XR9240 device. This application has the potential to increase small packet processing performance.

2 Operation

2.1 Architecture Overview

Figure 2-1 shows a high level block diagram of the XR9240. Each block is defined in Table 2-1.

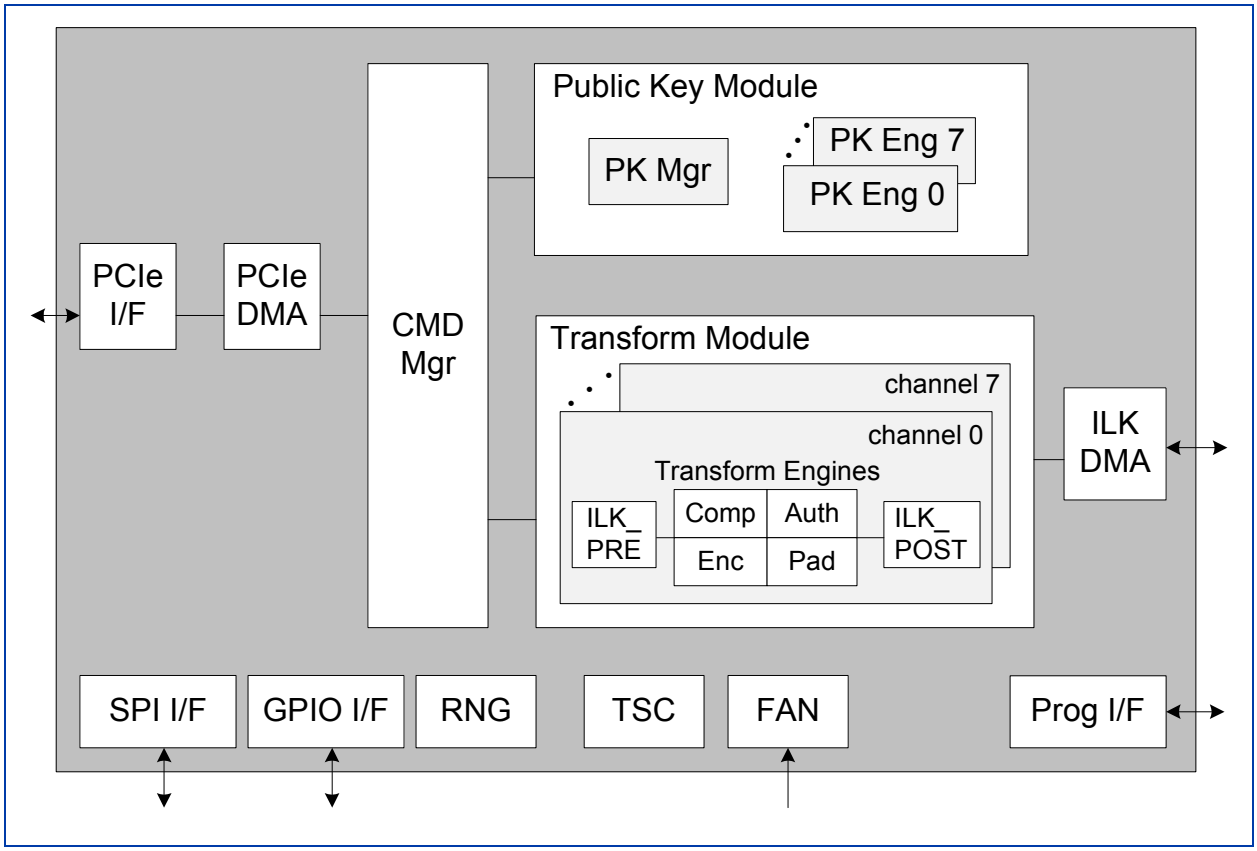


Figure 2-1. XR9240 Block Diagram

Table 2-1. Description of XR9240 Major Blocks

Block Name	Description
CMD Mgr	Command Manager. The Command Manager arbitrates among the command queues and dispatches commands to the Public Key module or Transform module.
FAN	Fan Monitor. Provides host software access to the fan sensor parameters.
GPIO I/F	General Purpose IO Interface.
ILK DMA	Interlaken DMA. Controls access to an optional external FPGA through the Interlaken interface.

Table 2-1. Description of XR9240 Major Blocks

Block Name	Description
ILK_PRE, ILK_POST	Interlaken Interface. Each of the eight Transform Channels contains two Interlaken Interface modules. The ILK Interface module may be ahead of the Transform Engine (ILK_PRE) or after the Transform Engine (ILK_POST) in the data path.
PCIe DMA	PCI Express DMA. Controls PCIe access and PCIe features such as TPH, IOV, power management, messages and interrupts.
PCIe I/F	PCI Express Interface. PCI Express end point controller.
PK Eng	Public Key Engine. There are eight PK engines in the PK module.
PK Mgr	Public Key Manager. Controls instruction and operand data fetching from host memory to the PK engines, and transmits the calculated result from the PK engines to host memory.
Prog I/F	Field Programming Interface. May be connected to a CPLD to program the attached FPGA.
Public Key Module	Public Key Module. The PK module consists of a Public Key manager and eight Public Key engines.
RNG	Random Number Generator.
SPI I/F	SPI interface. Used to connect the XR9240 device to a supported external non-volatile device such as an EEPROM or Flash.
TE	Transform Engine. Each of the eight Transform Channels contains a set of Transform Engines (TE). A Transform Engine consists of a Compression/Decompression engine, Encryption/Decryption engine, Authentication engine, and Pad/Depad engine.
Transform Module	Transform Module. The Transform Module contains eight Transform Channels (TC). Each Transform Channel contains two Interlaken Interfaces modules and a set of Transform Engines.
TSC	Temperature Sensor Controller. Provides host software access to the temperature sensor parameters.

2.2 Data Integrity

The XR9240 provides robust data integrity that protects the data path and validates the transform operations.

A combination of data integrity features create end-to-end protection of the data path. The data integrity features may be independently enabled or disabled, however, enabling all features ensures the highest level of data protection. The data integrity features have minimal impact on performance and latency.

Figure 2-2 illustrates the data path integrity for the Transform Engines.

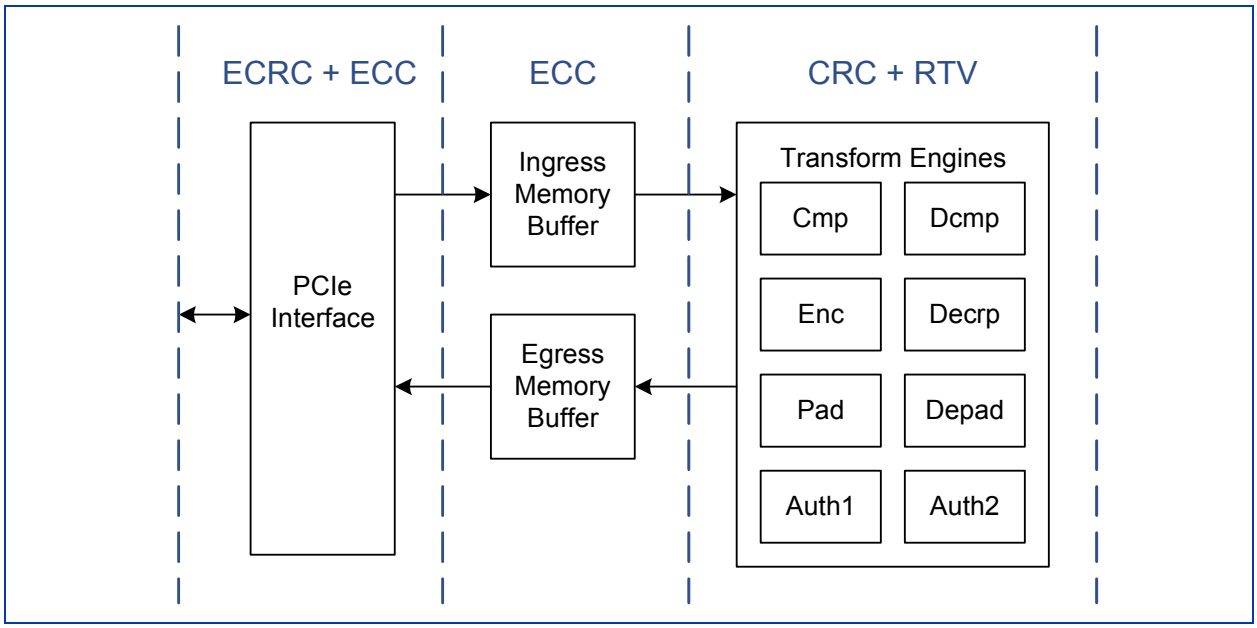


Figure 2-2. Transform Engine Data Integrity

The data path through the PCIe link and interface are protected by both ECRC and ECC. The ECRC is generated and verified at the boundary of the PCIe interface. The XR9240 ingress and egress memory buffers are protected with ECC.

The data in the Transform Engines is protected with embedded CRCs. Computations within the Transform Engines are validated with a combination of Real Time Verification (RTV) and CRC. In the encode direction, computations are the Transform Engine generates a CRC that is validated using a Real Time Verification (RTV). In the decode direction, the data in the Transform Engines are validated with the embedded CRC. See "Real Time Verification" for more details.

Figure 2-3 illustrates the data path integrity for the Public Key Engines.

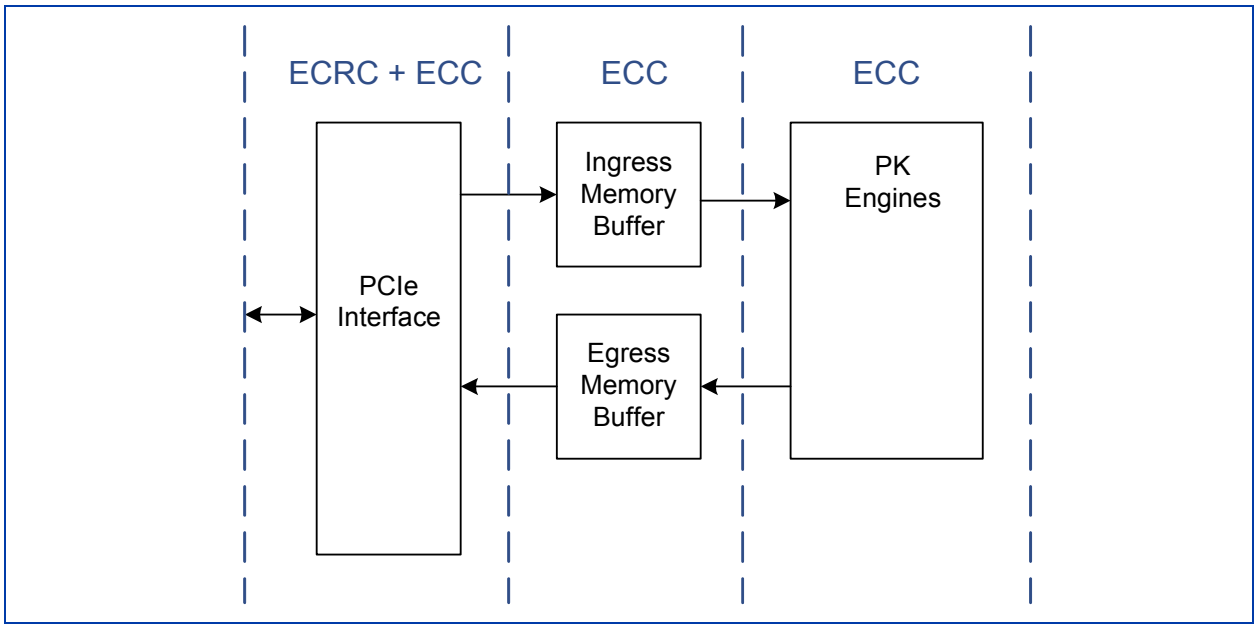


Figure 2-3. Public Key Integrity

The data paths through the PCIe link, PCIe interface and the XR9240 ingress and egress memory buffers are protected in the same way as the Transform Engine data path. The data in the Public Key Engines is protected with ECC.

Typically, storage applications will be able to enable all data integrity features. Networking applications that need to conform to security protocols may not be able to enable the embedded CRC. But as network data is protected by a MAC value, the XR9240 will validate the MAC value to verify networking operations.

2.2.1 PCIe Link Protection

The host may configure the PCIe Advanced Error Capabilities and Control Register to either enable/disable ECRC protection. If enabled, the PCIe Interface will generate the ECRC on the fly to protect data on the PCIe bus.

2.2.2 SRAM Protection

ECC is used to protect internal data paths on SRAMs that are not already protected by either RTV or an embedded CRC. ECC protection is always enabled.

The host will be notified of all ECC errors (see "Error ID Encoding"). 1-bit ECC memory errors, except for PK memories and some gzip memories, are correctable. 2-bit ECC memory errors are detectable but not correctable.

Correctable 1-bit ECC errors are logged as an ECC warning, whereas uncorrectable 2-bit ECC errors are logged as ECC errors. If more than 256 correctable errors are logged to a particular register, a DATA_ERR interrupt (see "PCIe Interrupt Source Register") will be

asserted, if enabled. The host can then determine the memory and error type by reading the XR9240 "DMA 1-bit ECC Error Counter Register". Uncorrectable 1-bit errors in the PK memories and some gzip memories are handled the same as 2-bit uncorrectable errors.

2.2.3 Source/Destination CRC Data Protection

The XR9240 supports a 32-bit embedded CRC that protects the source and destination data of a storage application session. If enabled, the embedded CRC will be verified by the XR9240, and may be sent to the host or stripped from the data stream. Two bits, CRC_IN_EN and CRC_OUT_EN, defined in the "Session Control Word 0" define the input CRC verification and output CRC generation.

Table 2-2. CRC Behavior

CRC_IN_EN	CRC_OUT_EN	Input	Output	XR9240 Operation
0	0	CRC input not present	CRC output not present	No CRC verification or generation
0	1	CRC input not present	CRC output present	No CRC verification, CRC output generated
1	0	CRC input present	CRC output not present	Input CRC verified, CRC is stripped from output
1	1	CRC input present	CRC output present	Input CRC verified, CRC appended to end of packet

The CRC implementation is based on the Network Working Group RFC 1662 documentation. The algorithm initial value is 0xFFFFFFFF. The equation to calculate the 32-bit CRC is:

$$CRC = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

where x is the current data byte.

Note that CRC verification and generation is always performed on the "raw" data. That is, for an encode command, the CRC is computed on the input data before encode; for a decode command, the CRC is computed on the output data after decode.

2.2.4 Real Time Verification

The Compression, Padding, Encryption and Authentication engines contain internal real time verification logic that ensures transforms are completed successfully and any detected errors are reported prior to the command completing. Real time verification is a powerful tool to detect data corruption during data processing, particularly when the source data is encoded.

Real time verification may be enabled or disabled per command via the RTV bit in the "Session Control Word 0". If disabled, real time verification will slightly increase the performance of small packet (< 2K bytes) LZS and gzip/Deflate compression operations. All other transforms will be unaffected. Disabling real time verification will slightly decrease the XR9240 power consumption.

Real time verification cannot be enabled for the following algorithms:

- UEA1/UIA1
- EEA1/EIA1
- EEA3/EIA3
- RC4
- P_hash
- Padding modes without length field (bit 3=0)

Real time verification will automatically be disabled by the XR9240 if two hash algorithms are selected in a single command.

2.2.4.1 Compression, Padding and Encryption Real Time Verification

The Compression, Padding and Encryption modules share the same real time verification data path. If RTV is enabled for an encode operation, the XR9240 will compute an initial RTV CRC on the raw data. Figure 2-4 illustrates real time verification in the Compression, Padding and Encryption modules for encode operations.

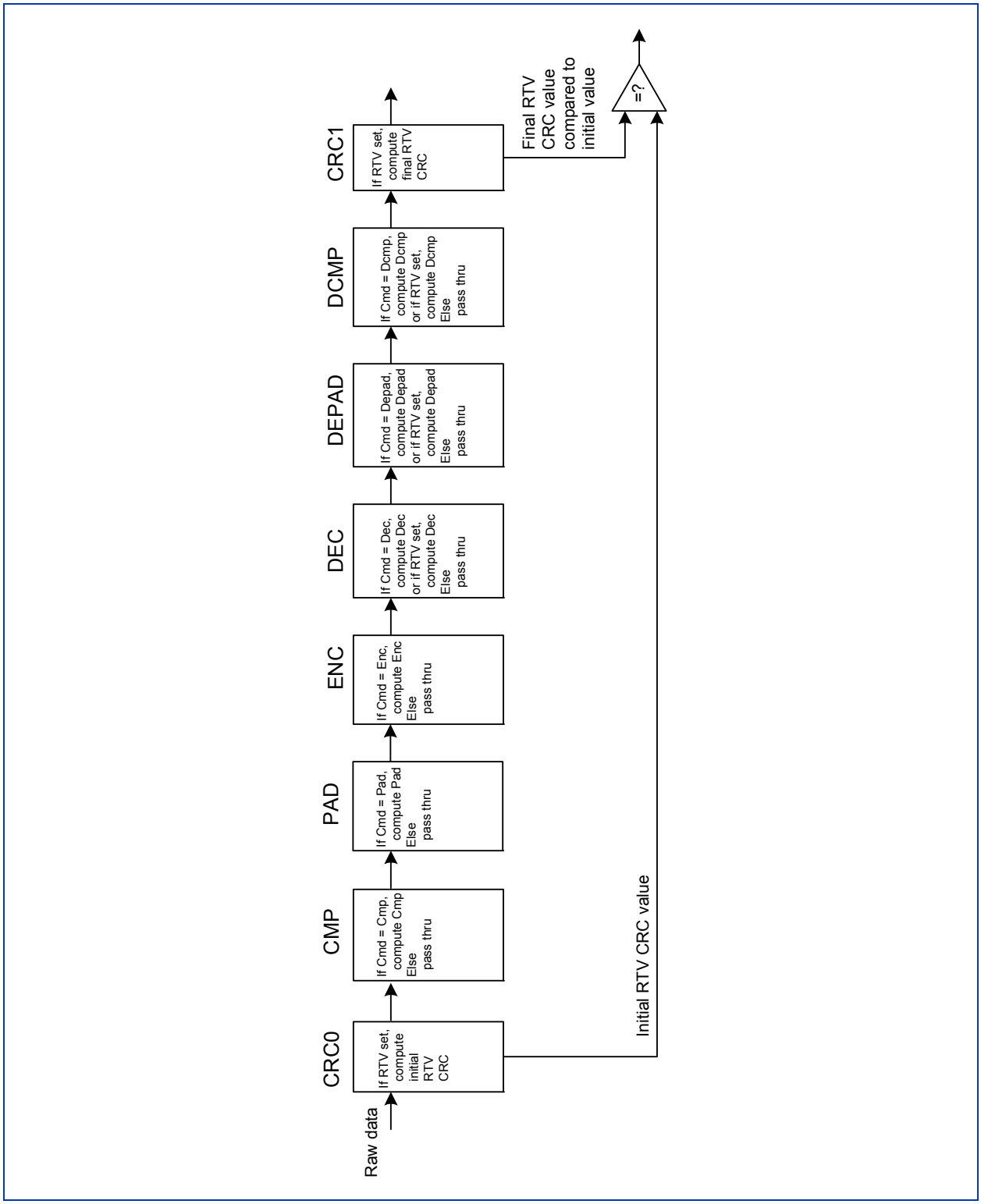


Figure 2-4. Compression, Pad, Encryption Real Time Verification for Encode Operations

Data for compression operations with RTV enabled will be automatically decompressed to verify that the decompressed CRC matches the original CRC of the raw data. If the CRCs do not match, the CRC1 module will report a RTV error to the host.

Data for padding operations with RTV enabled will be automatically depadded to verify that the depadded CRC matches the original CRC of the raw data. If the CRCs do not match, the CRC1 module will report a RTV error to the host.

Data for encryption operations with RTV enabled will be automatically decrypted to verify that the final CRC matches the original CRC of the raw data. If the CRCs do not match, the CRC1 module will report a RTV error to the host.

Note that if a RTV error occurs, the XR9240 is not able to identify the exact engine that failed among the compression, padding and encryption engines.

2.2.4.2 Authentication Real Time Verification

The Authentication module contains dual authentication engines that may either be used to operate on two authentication algorithms simultaneously or to implement real time verification of the raw data. Note that if two authentication algorithms are enabled in a single command, real time verification will automatically be disabled.

Data for authentication operations with RTV enabled will be automatically sent to both authentication engines. If the results do not match, the Authentication module will report a RTV error to the host.

For a single File hash or Slice hash operation, both engines calculate the hash/MAC simultaneously and the two results are compared. If an error is detected it will be reported to the host. The XR9240 does not support a combined File hash + Slice hash operation in a single pass.

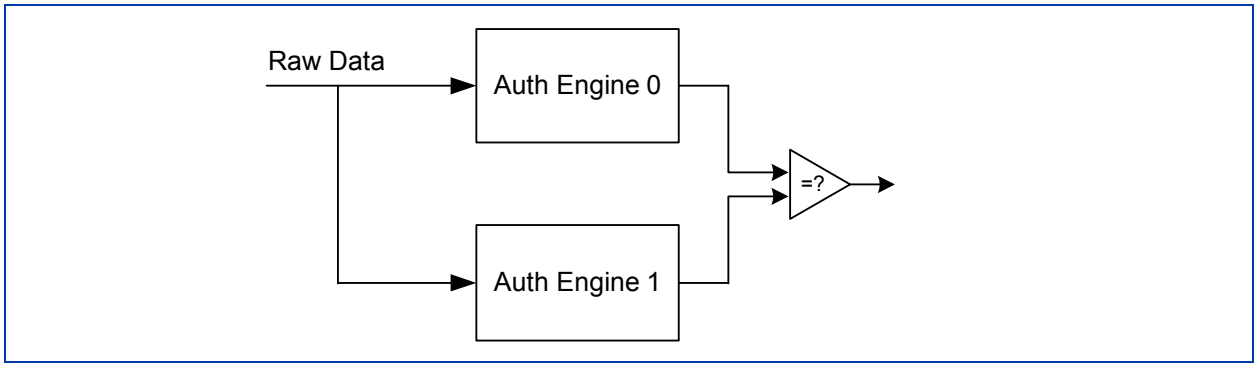


Figure 2-5. Authentication Engine Real Time Verification

2.2.4.3 Real Time Verification for Commands with Multiple Algorithms

For commands that contain a mixture of compression, padding, encryption, and authentication algorithms, the real time verification flows described in "Compression, Padding and Encryption Real Time Verification" and "Authentication Real Time Verification" apply.

3 Data Structures

This section describes the XRXR9240 data structures for the Command Pointer Ring, Result Pointer Ring, Free Buffer Pointer ring, and the Transform Channel and Public Key source buffer formats.

Chapter 4, "XR9240Command Operation" describes how these data structures are used in the command operation sequence.

3.1 Command Pointer Ring

The command pointer ring is circular ring of command pointers which point to the command structures in host memory. The XR9240 uses indirect command addressing to execute commands, thereby allowing the commands to be located anywhere in host memory.

The host software must maintain the command pointer ring and ensure that the command pointer ring base address is 8-byte aligned. The entries in the command pointer ring should be entered in consecutive order.

In 32-bit addressing mode, every command pointer is 4 bytes; in 64-bit addressing mode, every command pointer is 8 bytes.

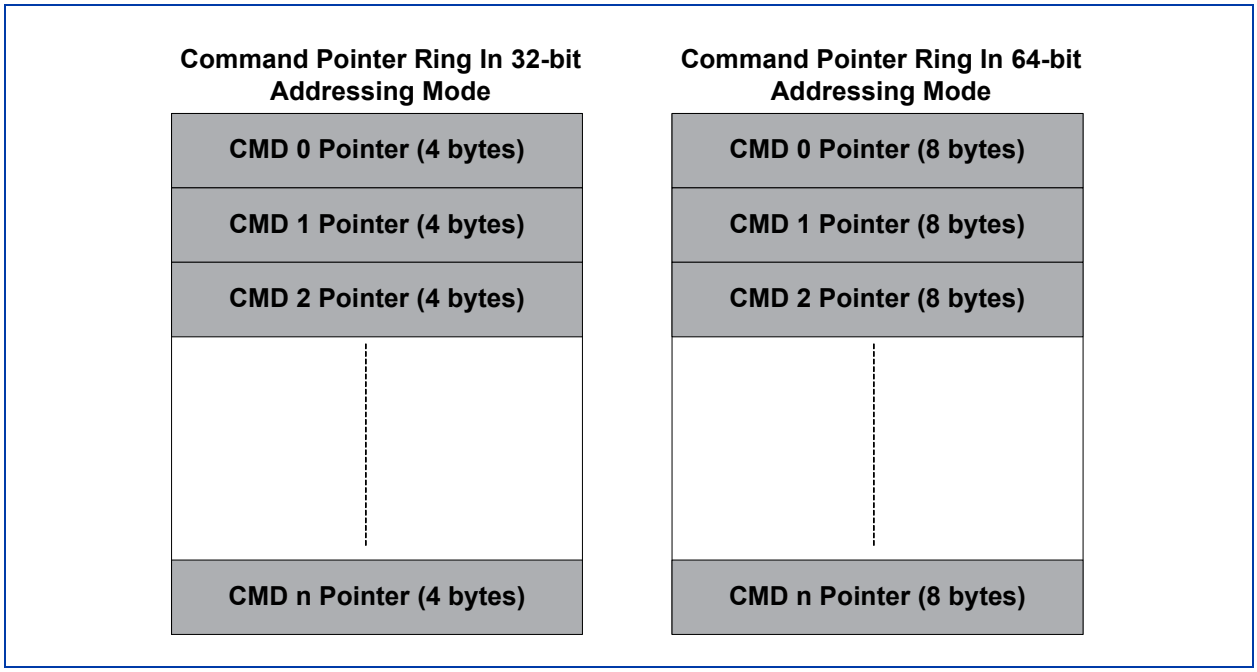


Figure 3-1. Command Pointer Ring Format

The minimum number of entries in the command pointer ring is 32 entries. The length of the command pointer ring must be a exponential power of 2 that is greater than or equal to 32. The maximum number of command pointers in the command pointer ring is configurable by the host using the CPR_SZ[3:0] field in the "Ring Configuration Register".

Figure 3-2 illustrates a command pointer ring and the associated command structures in host memory.

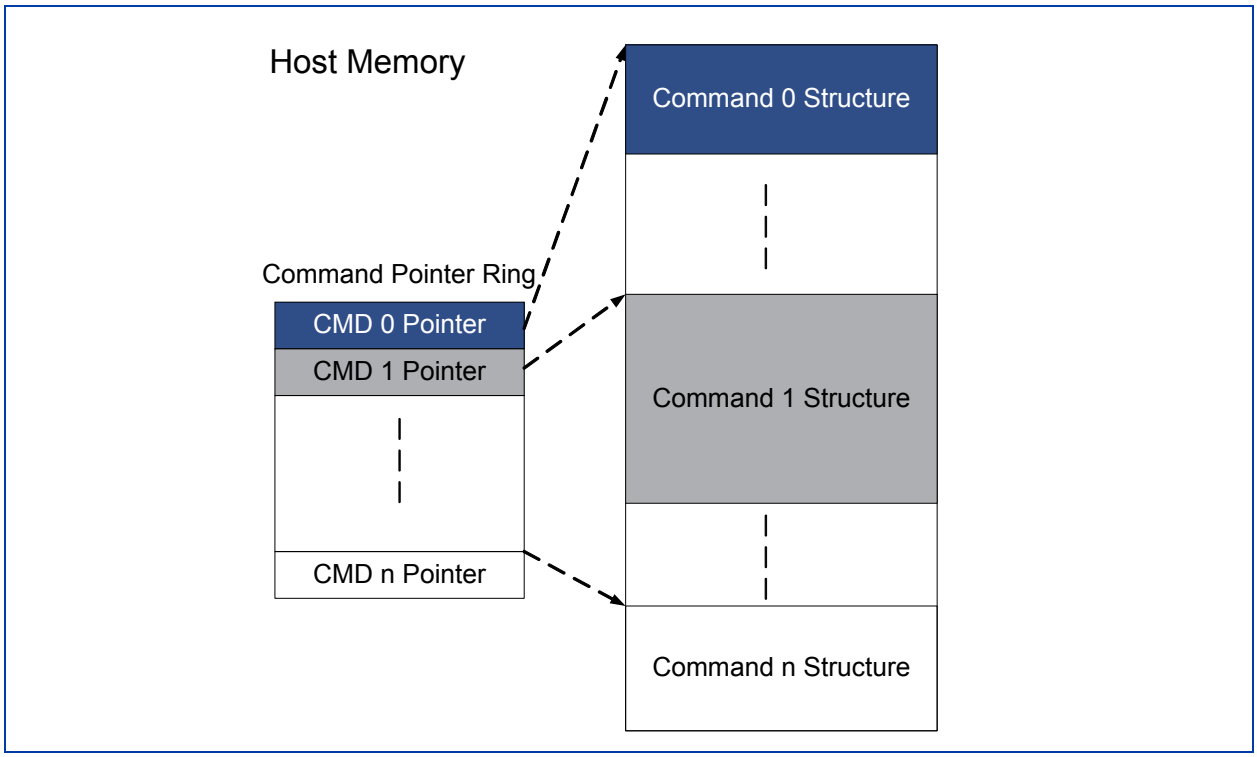


Figure 3-2. Command Pointer Ring Example

Command pointer rings are implemented in PCIe host memory. The host writes a complete command structure to any location in memory and then writes the pointer to that command structure as an entry in the command pointer ring. The XR9240 reads the command structure and interprets its various fields to perform the appropriate operations.

The XR9240 does not maintain a "full" bit for the command pointer ring; it is the responsibility of the host not to overflow the command ring.

3.1.1 Command Structure

A command structure consists of a linear array of descriptors. The size of one command structure set by the CMD_STRUCT_SZ[3:0] field in the "Ring Configuration Register" and is determined by the amount of host contiguous physical memory. Each command will have a unique command structure.

A command structure includes one command descriptor, and several pairs of source and destination descriptors. For all descriptors, reserved bits must be written as zeroes.

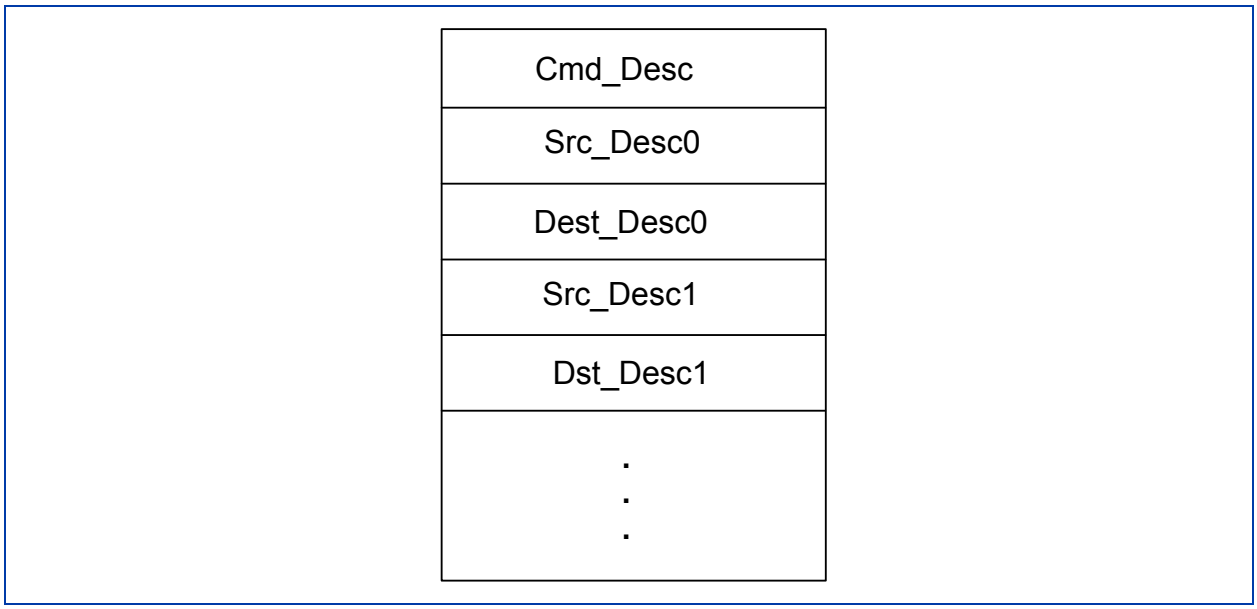


Figure 3-3. Command Structure

The starting address of each command structure must fall on a 128-byte boundary, regardless of the number of descriptor pairs for each command structure. This constraint will avoid the PCIe 4K boundary read request limitation because the XR9240 maximum command read request is 128 bytes.

Figure 3-4 shows an example of the command structure for the first command in a command ring. Note that while the command pointer ring is a linear array in contiguous host memory, the command structure and source buffers pointed to by the source descriptors may be scattered throughout host memory.

The XR9240 reads the command pointer, fetches the command descriptor which defines the type of command and any immediate parameters required to execute it. The XR9240 then continues fetching the source and destination descriptor pointers from the command structure. The source data is read by the XR9240 using the source descriptor pointers, and the output data is stored back to host memory using the destination buffers provided by the destination descriptor pointers.

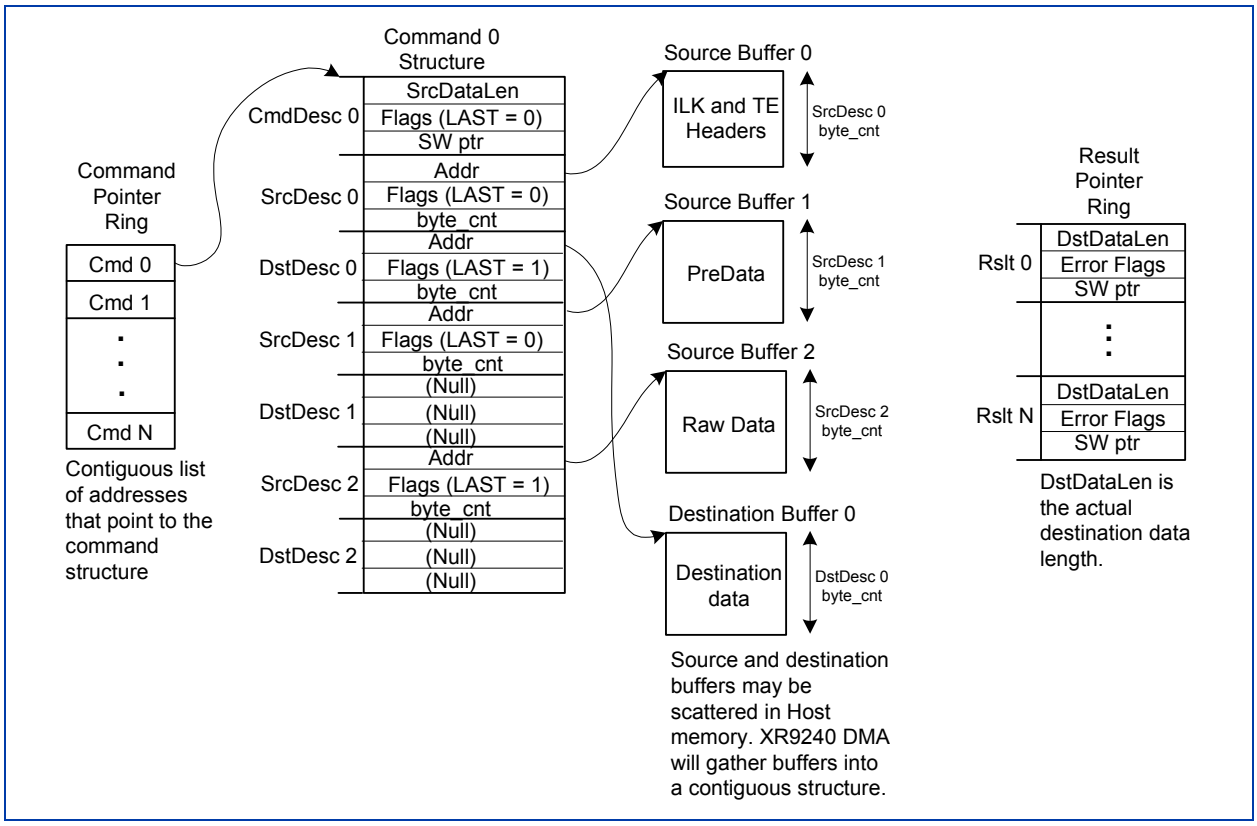


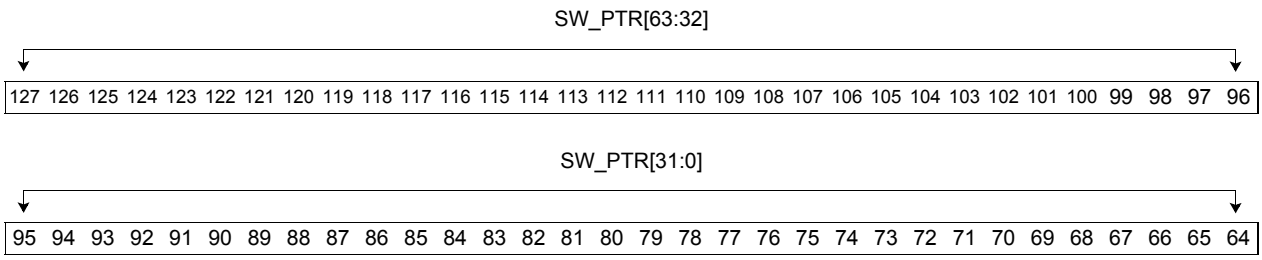
Figure 3-4. Command Structure Example

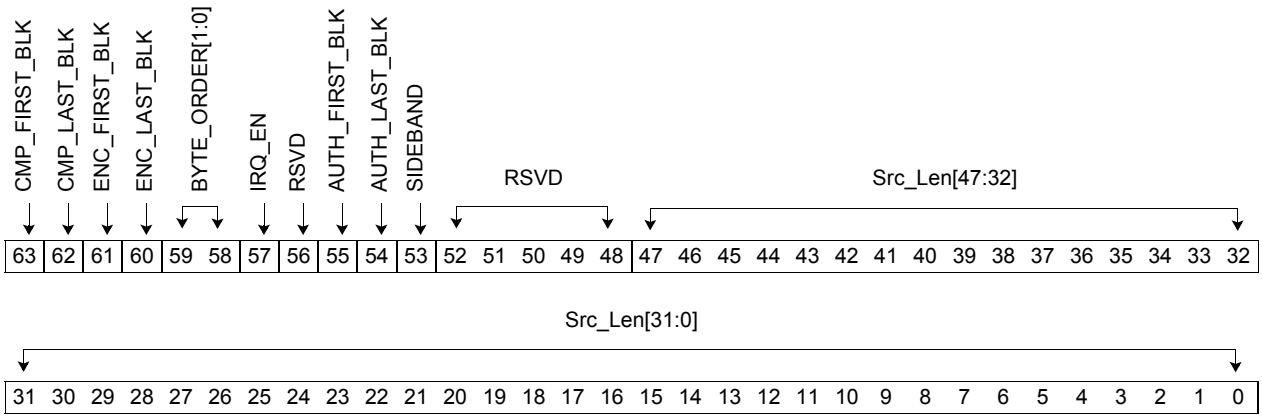
3.1.2 Command Structure Format

A command structure consists of one command descriptor and multiple source and destination descriptor pairs.

3.1.2.1 Command Descriptor

This section defines the command descriptor format. The command descriptor format is the same (16 bytes) for either 32-bit or 64-bit addressing modes.





Field Name	Bits	Description
SW_PTR[63:0]	127:64	Software Pointer. This field is a virtual address pointer to the command structure that is used by software and will be copied to an entry in the Result Pointer Ring. The software uses this pointer to determine the corresponding command structure of the entry in the Result Pointer Ring. With this pointer, an entry in command pointer ring could be overwritten immediately after the command pointer is read by the XR9240.
CMP_FIRST_BLK	63	Compression First Block Flag. This bit should be set for the first block of a stateful compression/decompression operation. 0 Current block is not the first block 1 Current block is the first block
CMP_LAST_BLK	62	Compression Last Block Flag. This bit should be set for the last block of a stateful compression/decompression operation. 0 Current block is not the last block 1 Current block is the last block
ENC_FIRST_BLK	61	Encryption First Block Flag. This bit should be set for the first block of a stateful encryption/decryption operation. 0 Current block is not the first block 1 Current block is the first block
ENC_LAST_BLK	60	Encryption Last Block Flag. This bit should be set for the last block of a stateful encryption/decryption operation. 0 Current block is not the last block 1 Current block is the last block

Field Name	Bits	Description
BYTE_ORDER[1:0]	59:58	Source and Destination Data Byte Order. Sets the byte order format for the source and destination data. Refer to Figure 3-5 for an illustration of the swap options. This field should not be confused with the CMD_EF field in the global control register that specifies the endian format of the host command pointer ring, command structure, result pointer ring and free buffer pointer ring. 00 no swap 01 byte swap within word 10 word swap, no byte swap within word 11 word swap and byte swap within word
IRQ_EN	57	Interrupt Enable Flag. This bit sets the command completion interrupt behavior. 0 Do not send interrupt to host when the command completes 1 Send interrupt to host when the command completes
RSVD	56	Reserved.
AUTH_FIRST_BLK	55	Authentication First Block Flag. This bit should be set for the first block of a stateful authentication operation. 0 Current block is not the first block 1 Current block is the first block
AUTH_LAST_BLK	54	Authentication Last Block Flag. This bit should be set for the last block of a stateful authentication operation. 0 Current block is not the last block 1 Current block is the last block
SIDEBAND	53	Sideband Flag. This bit is used to define whether the XR9240 should use a sideband buffer to store output data. 0 Sideband disabled 1 Sideband enabled
RSVD	52:48	Reserved.
SRC_LEN[47:0]	47:0	Source Data Length. The total length of the source data that is used to determine the COS arbitration. The length represents either the payload data length or the source data length including the session, user descriptor, payload data and pre-data fields such as the IV. Refer to "Transform Channel Source Buffer Format" and "Public Key Source Buffer Format" for more information.

Figure 3-5. Endian Format Field Swap Options

3.1.2.2 Source and Destination Descriptors

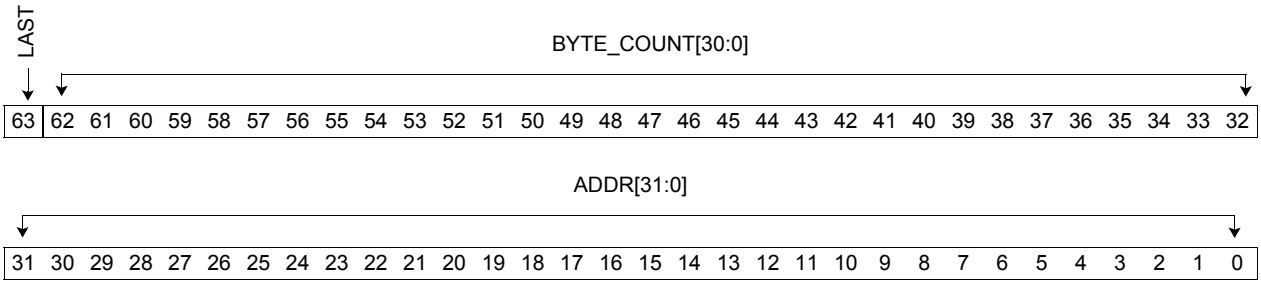
The source and destination descriptors share the same format.

The source data descriptors point to source data buffers. The source buffer address and length may be an arbitrary number of bytes and have no alignment restrictions.

The destination data descriptors point to the destination buffers. The destination buffer address and length have no alignment restrictions. Usually, the destination buffers are used to store the result data, and must be fully consumed in order. Note that the destination descriptor field BYTE_CNT[30:0] reserves a block of memory for the result data *before* the command executes. After the command completes, the actual length of the result data is saved to the "Result Pointer Ring".

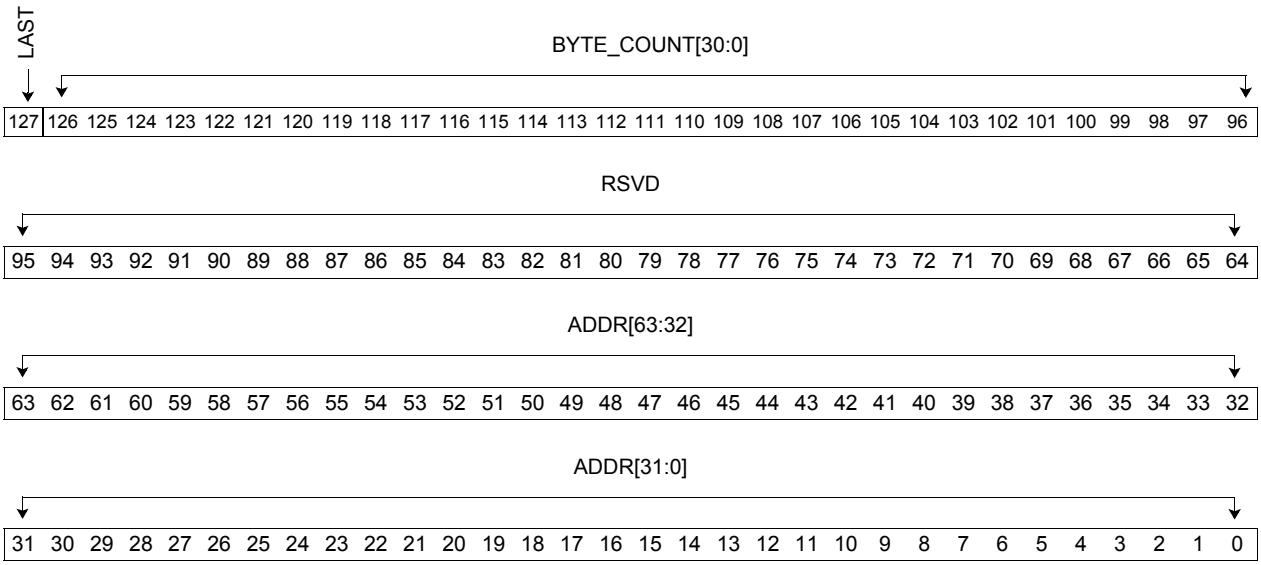
The source and destination descriptors contain 8 bytes in the 32-bit addressing mode and 16 bytes in the 64-bit addressing mode.

32-bit Addressing Mode Format



Field Name	Bits	Description
LAST	63	Last Descriptor Flag. Used to identify the current descriptor as the last valid descriptor in the command. The host software must set this bit for the last source or destination descriptor. Although the source and destination descriptors must be in pairs, the LAST flag need NOT be set in the same pair. See the Figure 3-6 below.
BYTE_CNT[30:0]	62:32	Byte Count. The length of the source or destination buffer for this descriptor in bytes. For the source descriptor, the byte count is an arbitrary number of bytes.
ADDR[31:0]	31:0	Address. The address pointer to the source or destination buffer.

64-bit Addressing Mode Format:



Field Name	Bits	Description
LAST	127	Last Descriptor Flag. Used to identify the current descriptor as the last valid descriptor in the command. The host software must set this bit for the last source or destination descriptor. Although the source and destination descriptors must be in pairs, the LAST flag need NOT be set in the same pair. See the example below.
BYTE_CNT[30:0]	126:96	Byte Count. The size of the source or destination buffer in bytes. It indicates the size of the data block defined by this descriptor. For the source descriptor, the byte count is an arbitrary number of bytes.
RSVD	95:64	Reserved.
ADDR[63:0]	63:0	Address. The address pointer to the source or destination data.

To conform to the format of the command structure, source and destination descriptors must be entered as pairs. If a command has three source descriptors and one valid destination descriptor, as shown in Figure 3-6, the host must allocate null destination descriptors to balance the source descriptors. The LAST bit in the descriptor is used to indicate the last descriptor in the command structure.

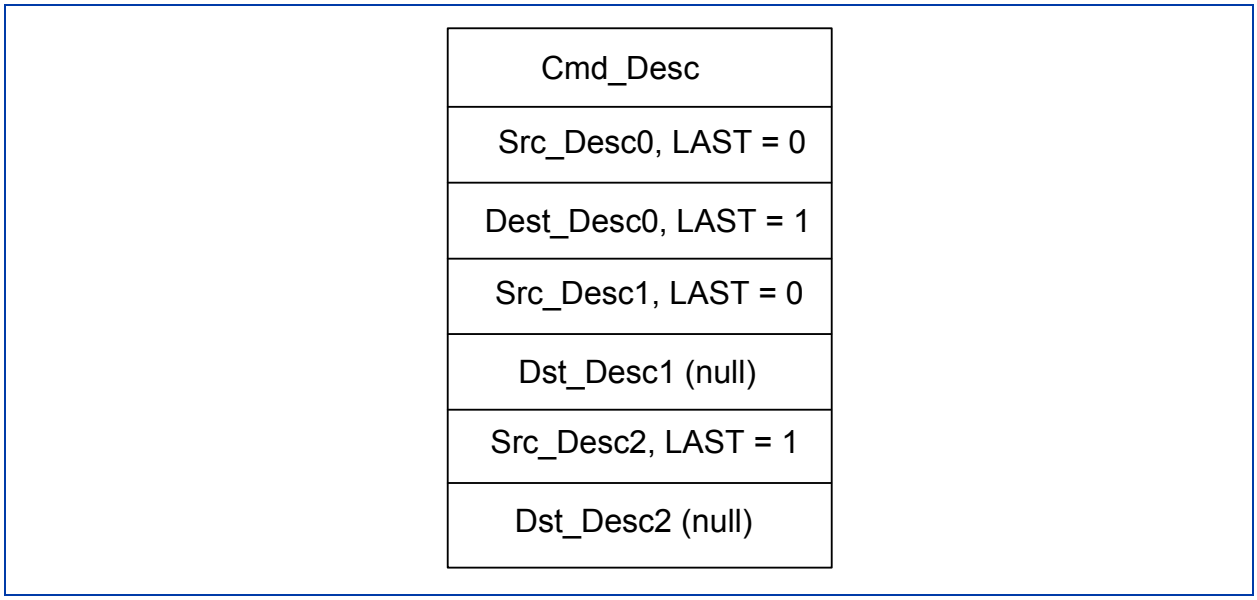


Figure 3-6. Command Structure Controlled by Last Bit

In the example above, the XR9240 will continue to parse the command structure until it recognizes the LAST bit set in the source and destination descriptors. The XR9240 recognizes that destination descriptor 0 is the last destination descriptor, but will continue

reading in the descriptors until reaching source descriptor 2 with the LAST bit set. Although the destination descriptors 1 and 2 are null, placeholders must be allocated for them in the command structure.

Note that all destination descriptors may be null in the command structure if all output is directed to the free buffer. Refer to "Free Buffer Pointer Ring" for its detailed usage.

An optional buffer, called the "Sideband Data" buffer, may be used to store sideband data such as the authentication result. The data structure of the sideband buffer is organized as a linked list so the last record in the sideband buffer can be managed by software. There is no address alignment restriction on the sideband buffer. If a sideband buffer is used, it must be pointed to by the first destination descriptor.

3.2 Transform Channel Source Buffer Format

The Transform Channel command structure source buffer format shown in Figure 3-7 contains a header and payload.

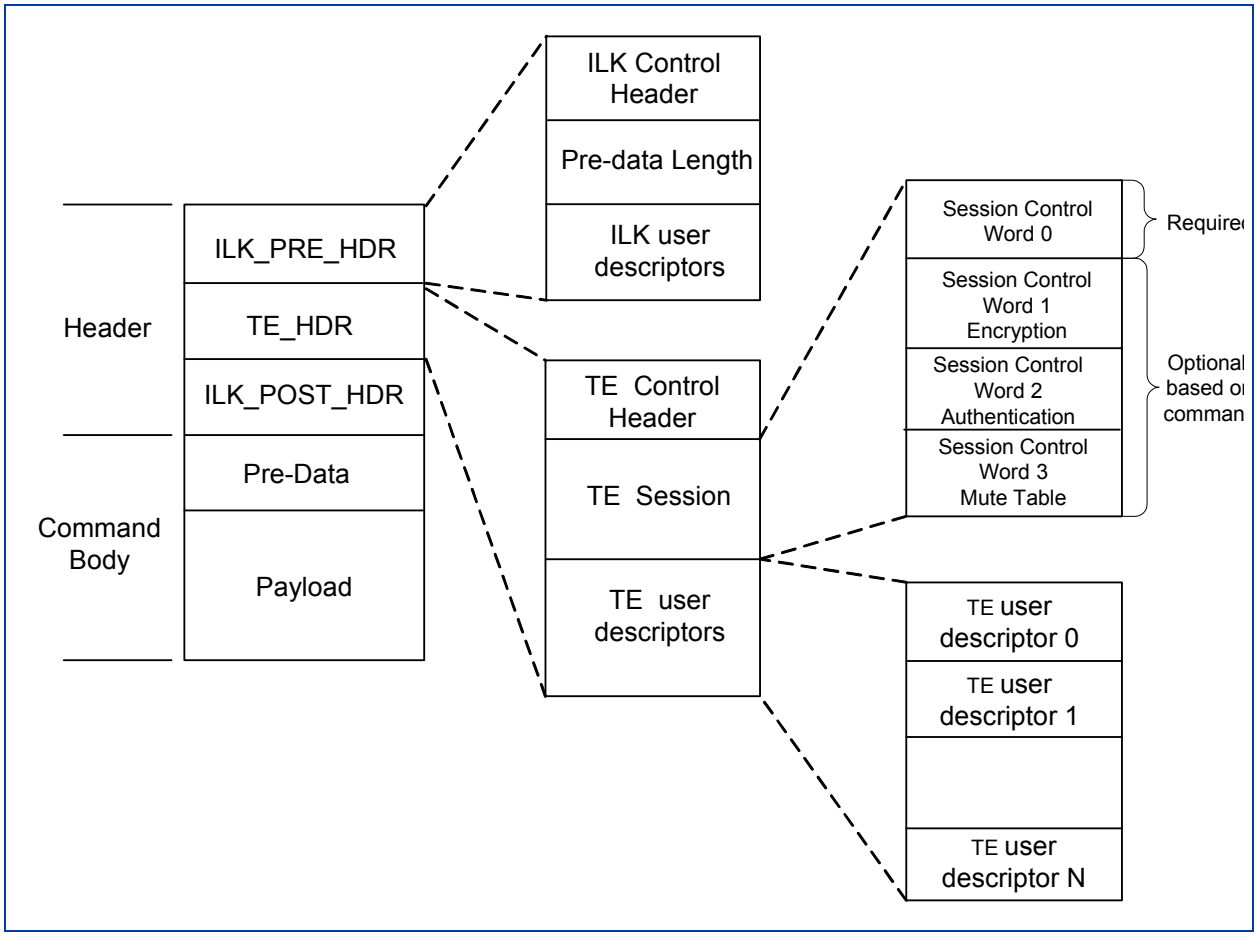


Figure 3-7. Transform Channel Source Buffer Format

There are four defined header types:

- ILK_PRE: An ILK_PRE header is defined as an ILK header that comes before an TE header. An ILK_PRE header is used for commands to be processed by the FPGA.
- TE: A TE header is used for commands to be processed by a Transform Engine.
- ILK_POST: An ILK_POST header is defined as an ILK header that comes after an TE header. An ILK_POST header is used for commands to be processed by the FPGA.
- Null: A Null header is used internally to identify the final processing results.

A transform channel source buffer must have at least one header and no more than three headers. The ILK_PRE and ILK_POST headers share the same format.

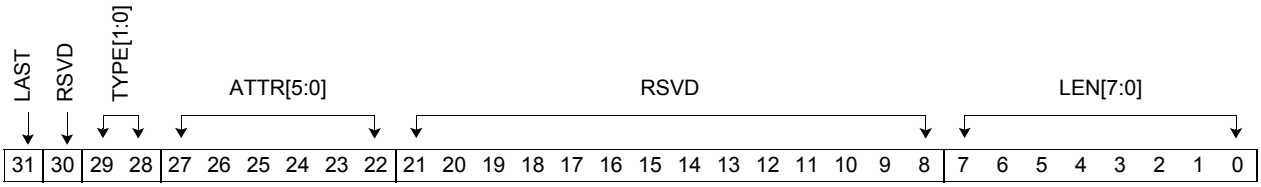
Refer to "Transform Processing Sequence" for an explanation of how the order of the headers controls the command processing.

3.2.1 Interlaken Header

The Interlaken header consists of the ILK control header and multiple ILK user descriptors. The format for the Interlaken header applies to both ILK_PRE and ILK_POST headers.

3.2.1.1 ILK Control Header

This section defines the ILK control header format.



Field Name	Bits	Description
LAST	31	Header Last Flag. This bit should be set for the last header. 0 Current header is not the last header 1 Current header is the last header
RSVD	30	Reserved.
TYPE[1:0]	29:28	Header Type. 00 Null header 01 TE header 10 ILK_PRE header 11 ILK_POST header
ATTR[5:0]	27:22	User Defined Attribute. This field applies only to ILK headers and is intended for user defined attributes that would specify FPGA operations, such as the programming the FPGA image. This field is ignored for a TE and Null header.
RSVD	21:8	Reserved.
LEN[7:0]	7:0	Header Length. This field represents the length of the header in words, including the header word. 0 256 words 1 1 word : 255 255 words

3.2.1.2 ILK Pre-Data Length Header

This 4 byte field is the length in bytes of the pre-data in the payload. This field is used by the FPGA to identify the starting location of the payload that is to be processed.

3.2.1.3 ILK User Descriptor Header

The ILK user descriptor format is defined by the user. Its structure will be based on the functional design in the FPGA.

3.2.2 Transform Engine Header

The TE header consists of the TE control header, TE session descriptor and multiple TE user descriptors.

3.2.2.1 Transform Engine Control Header

The TE control header has the same format as the ILK control header. See "Interlaken Header" for details.

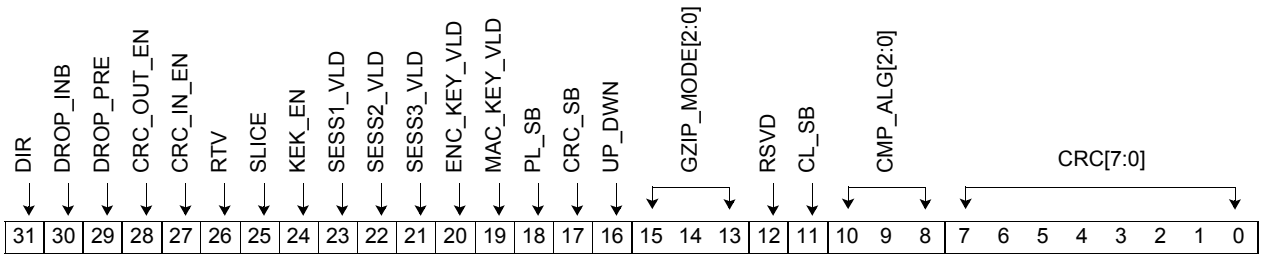
3.2.2.2 Transform Engine Session Descriptor

The TE session descriptor contains the session specific parameters for a group of commands in that session. An TE session consists of 1-4 session control words.

The first control word is required and contains flags that indicate whether the second, third and fourth control words are present. The second control word is used for encryption and padding operations. The third control word is used for authentication operations. The fourth control word is used for specifying the Mute tables for hash operations.

The compression, encryption, authentication and padding parameters defined in the session control words are described in more detail in Chapter 5, "Modules".

Session Control Word 0

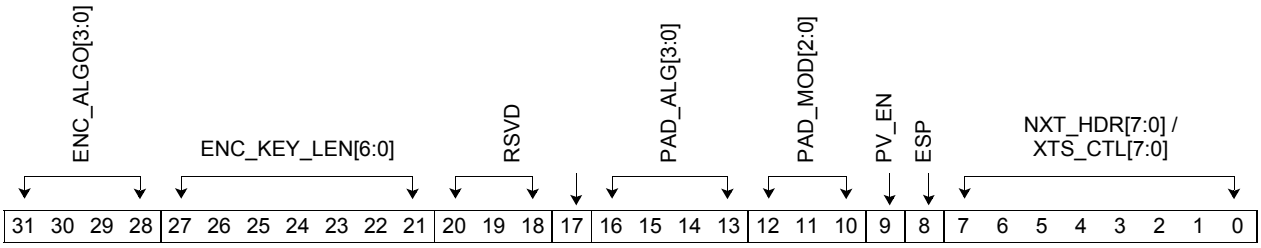


Field Name	Bits	Description
DIR	31	Encode/Decode Data Flow Direction. 0 Decode 1 Encode
DROP_INB	30	Drop Inband Data. If this bit is set, the inband data is not transformed and is discarded before being output. Some applications such as hash only require the hash result. 0 Inband data is not discarded 1 Inband data is discarded
DROP_PRE	29	Drop Predata. This bit controls whether the predata is included in the destination buffer. Network applications will typically include the predata to hold the IV or AAD. Storage applications will typically discard the predata. 0 Predata is not discarded 1 Predata is discarded
CRC_OUT_EN	28	Output CRC Enable. Refer to Table 2-2 for a description of how this bit is used in conjunction with CRC_IN_EN.
CRC_IN_EN	27	Input CRC Enable. Refer to Table 2-2 for a description of how this bit is used in conjunction with CRC_OUT_EN.
RTV	26	Real Time Verification. This bit should not be enabled when DIR = Decode. 0 Disable real time verification 1 Enable real time verification
SLICE	25	Slice Hash Operation. 0 Operation is not a slice hash operation 1 Slice hash operation
KEK_EN	24	Key Encryption Key Enable. 0 Disable KEK 1 Enable KEK
SESS1_VLD	23	Session Control Word 1 Valid. Session control word 1 is optional and includes the encryption and padding information. 0 Session control word 1 not present 1 Session control word 1 present
SESS2_VLD	22	Session Control Word 2 Valid. Session control word 2 is optional and includes the control fields for authentication operations. 0 Session control word 2 not present 1 Session control word 2 present

Field Name	Bits	Description
SESS3_VLD	21	Session Control Word 3 Valid. Session control word 3 is optional and includes the mute mask for hash operations. 0 Session control word 3 not present 1 Session control word 3 present
ENC_KEY_VLD	20	Encryption Key Valid. 0 Encryption key field is not included in the session 1 Encryption key field of the session is valid
MAC_KEY_VLD	19	MAC Key Valid. 0 MAC key field is not included in the session 1 MAC key field of the session is valid
PL_SB	18	Pad Length Sideband Output Control. 0 Do not output pad length to sideband 1 Output pad length to sideband
CRC_SB	17	CRC Sideband Output Control. 0 Do not output CRC 1 Output sideband CRC
UP_DWN	16	Direction. This field is only valid for wireless algorithms (UEA1/UIA1, EEA1/EIA1, EEA3/EIA3). 0 Uplink direction 1 Downlink direction
GZIP_MODE[2:0]	15:13	GZIP Mode. 000, 001 Raw data mode. Only stored mode allowed. 010 Static Huffman mode. 011 Static/Stored mode. The smaller block size of the static or stored mode will be used on a block by block basis. 100 Dynamic Huffman mode. 101 Dynamic/Stored mode. The smaller block size of the Dynamic Huffman or stored mode will be used on a block by block basis. 110 Dynamic/Static mode. The smaller block size of the Dynamic or Static Huffman mode will be used on a block by block basis. 111 Dynamic/Static/Stored mode. The smallest block size of the Dynamic or Static Huffman, or stored mode will be used on a block by block basis.
RSVD	12	Reserved. Must be set to zero.

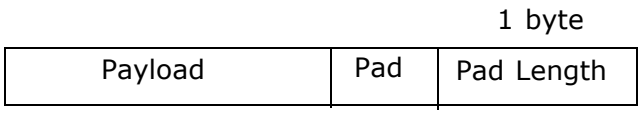
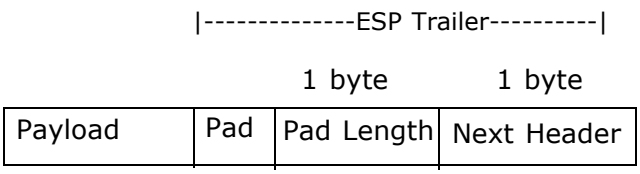
Field Name	Bits	Description
CL_SB	11	Compression Length Sideband. 0 Do not generate the compression length as sideband data 1 Generate the compression length as sideband data
CMP_ALG[2:0]	10:8	Compression Algorithm. 000 LZS algorithm 001 eLZS algorithm 010 GZIP algorithm 011 Deflate algorithm 100 zlib algorithm All other values are reserved.
CRC[7:0]	7:0	Session CRC. This field is the CRC for the session structure, including the session control words and key. The value in this field is cleared when calculating the session CRC. This field is optional and is not required by default. Session CRC may be enabled via the SESS_CRC bit in the "TE Control Register".

Session Control Word 1 - Encryption



Field Name	Bits	Description
ENC_ALG[3:0]	31:28	Encryption Algorithm. This field is used to set the encryption algorithm for the session. 0000AES-GCM 0001AES-CBC 0010AES-CTR / EEA2 0011AES-ECB 0100AES-XTS 0110AES-F8 01113DES-CBC 1000UEA1 1001EEA1 1011EEA3 1111ARC4 All other values reserved. Notes: If ENC_ALG[3:0] = AES-GCM, the authentication algorithm must be set to AES-GCM-MAC.
ENC_KEY_LEN[6:0]	27:21	Encryption Key Length. This field is used to set the encryption key length in words, and is only valid if ENC_KEY_VLD in Session Control Word 0 is set. The encryption key lengths of all algorithms are word aligned. Dummy bytes can be inserted at the end of the key. For ARC4, the last byte of the key is the key length. 0 128 words or 512 bytes 1 1 word or 4 bytes : 127127 words or 508 bytes
RSVD	20:17	Reserved.

Field Name	Bits	Description
PAD_ALG[3:0]	16:13	Padding Algorithm. The behavior of this field depends on the value of DIR in "Session Control Word 0" and the setting of the ESP bit. Refer to Table 3-1 for the values of this field. For all algorithms, the pad length is the number of bytes in the preceding Padding field, except for setting 1010 when the pad length is the number of bytes in the preceding Padding field plus the "pad length" byte. PAD_AG[3] indicates whether the pad length field is included in the padding. If PAD_AG[3] = 0, the pad length field is not included, and if PAD_AG[3] = 1, the pad length field is included. PAD_AG[3] = 0, without Length byte Payload data | Pad PAD_AG[3] = 1, with Length byte Payload data | Pad | Pad Length
PAD_MOD[2:0]	12:10	Pad Modulo. This field determines the modulo setting for the Pad engine. 000 MOD-4 The payload after padding is 4-byte aligned. 001 MOD-8 The payload after padding is 8-byte aligned. 010 MOD-16 The payload after padding is 16-byte aligned. 011 MOD-32 The payload after padding is 32-byte aligned. 100 MOD-64 The payload after padding is 64-byte aligned. 101 MOD-128 The payload after padding is 128-byte aligned. 110 MOD-256 The payload after padding is 256-byte aligned. 111 Reserved
PV_EN	9	Pad Verification Enable. This bit is used to enable or disable verification of the padding value. This bit is only valid in decode mode. A padding verification error will be recorded as a Depad Engine error (see "Transform Engine Errors"). Note that the padding value will be removed regardless of the state of this bit. 0 Depad engine does not verify padding value 1 Pad engine will verify the padding value

Field Name	Bits	Description
ESP	8	ESP Header Padding Enable. If DIR = Encode direction: 0 Next Header will not be inserted by Pad engine 1 Pad engine will pad Next Header with pad value and pad length (this setting is used for IPsec packet processing) If DIR = Decode direction: 0 Pad engine will extract Pad Length field from the data stream, and write the Pad Length to the host command structure after the command completes 1 Pad engine will extract Pad Length and Next Header fields from the data stream, and write these fields to the host command structure after the command completes Note: if ESP = 1, PAD_ALG[3] must be set to 1. ESP Header Format for ESP = 0:  ESP Header Format for ESP = 1: 

Field Name	Bits	Description
NXT_HDR[7:0] / XTS_CTL[7:0]	7:0	Next Header / XTS Control. This field has mixed usage, depending on the type of operation. If ESP = 1, this field represents the IPv4/IPv6 protocol number describing the format of the Payload data field. The software must set this field for IPsec packet processing. The Pad engine will append this field to the data stream if the ESP bit is set. If ENC_ALG[3:0] = XTS, this field contains the XTS control information listed below. See "AES-XTS Operation" for more information. Bit [7] DIF Enable 0DIF is not attached 1DIF is attached Bit [6] DIF Protection Mode. Only valid when DIF_EN = 1. 0Protect sector without DIF 1Protect sector with DIF Bit [5] DIF Format. Only valid when DIF_EN = 1. 0T10/03-310r0 1T10/08-044r1 Bits [4:0] XTS size. Only valid when EN_ALG = XTS. 0000064B 00001128B ... 1010064MB All other values reserved.

Table 3-1. PAD_AG[3:0] Field Decoding

Encode/ Decode	ESP	PAD_AG [3:0]	Description
Encode	0	0000	Padding value is 0 - 255. For example, if 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 00, 01, 02, 03, 04.
		0001	Padding value 1 - 255 - 0. For example, If padding 256 bytes, the padding will be 1, 2, 3, ... 255, 0.
		0010	Padding with same value, the value is the number of bytes inserted. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 05, 05, 05, 05, 05.
		0011	Padding will be all zeroes. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 00, 00, 00, 00, 00.
		0100	Padding with number of bytes minus one inserted. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 04, 04, 04, 04, 04.
		1000	Padding value 0 - 255, with pad length field. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 00, 01, 02, 03, 04.
		1001	Padding value 1 - 255 - 0, with pad length field. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 01, 02, 03, 04, 04.
		1010	Padding with same value, the value is the number of bytes inserted, with pad length field. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 05, 05, 05, 05, 05.
		1011	Padding with all zero, with pad length field. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 00, 00, 00, 00, 04.
		1100	Padding with number of bytes inserted minus 1, with pad length field. For example, If 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 04, 04, 04, 04, 04.
		All others	Reserved.

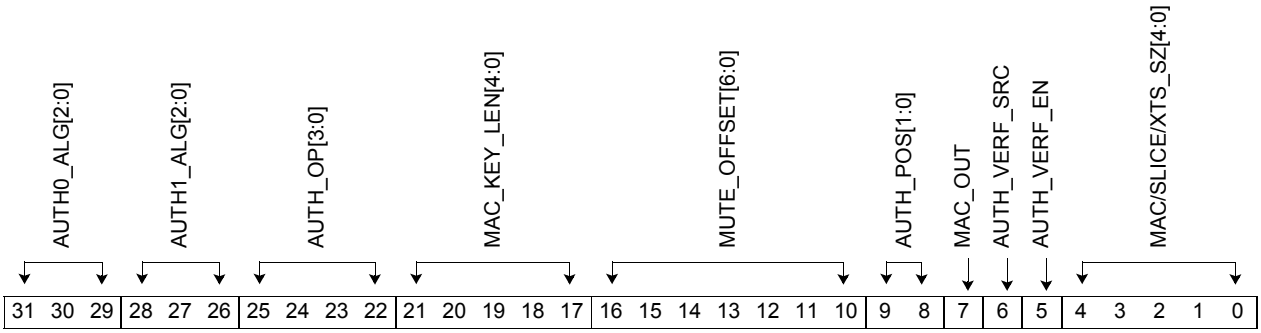
Table 3-1. PAD_AG[3:0] Field Decoding

Encode/ Decode	ESP	PAD_AG [3:0]	Description
Encode	1	0xxx	Not allowed
		1000	Padding value 0 - 255, with pad length field and next_header. For example, if 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 00, 01, 02, 03, <next_header>.
		1001	Padding value 1 - 255 - 0, with pad length field and next_header. For example, if 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 01, 02, 03, 03, <next_header>. Note: please refer to RFC 2406 (ESP) for this configuration.
		1010	Padding with same value, the value is the number of bytes inserted, with pad length field and next_header. For example, if 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 04, 04, 04, 04, <next_header>.
		1011	Padding with all zeroes, with pad length field and next_header. For example, if 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 00, 00, 00, 03, <next_header>.
		1100	Padding with number of bytes inserted minus 1, with pad length field and next_header. For example, if 5 bytes are required to pad out to modulo 8, 5 bytes will be inserted, the padding will be 03, 03, 03, 03, <next_header>.
		All others	Reserved.
Decode	0	0000	No verification and removal (no pad length field)
		0001	No verification and removal (no pad length field)
		0011	No verification and removal (no pad length field)
		0100	No verification and removal (no pad length field)
		1000	Verify the removed padding value 0 - 255 and the pad length field
		1001	Verify the removed padding value 1 - 255 - 0 and the pad length field
		1010	Verify the removed padding of number of bytes inserted, and the pad length field
		1011	Verify the removed padding value of all zeroes and the pad length field
		1100	Verify the removed padding of number of bytes inserted minus one, and the pad length field
		All others	Reserved.

Table 3-1. PAD_AG[3:0] Field Decoding

Encode/ Decode	ESP	PAD_AG [3:0]	Description
Decode	1	0xxx	Not valid.
		1000	Verify the removed padding value 0 - 255, pad length field; remove next_header
		1001	Verify the removed padding value 1 - 255 - 0, pad length field; remove next_header
		1010	Verify the removed padding of number of bytes inserted, pad length field; remove next_header
		1011	Verify the removed padding value of all zeroes, pad length field; remove next_header
		1100	Verify the removed padding value of all padding length value less one, pad length field; remove next_header
		All others	Reserved.

Session Control Word 2 - Authentication



Field Name	Bits	Description
AUTH0_ALG[2:0]	31:29	Authentication Engine 0 Algorithm. This field is used to set the authentication algorithm for the session. This field represents one of the dual Authentication engines; AUTH1_ALG[2:0] represents the other Authentication engine. AUTH0_ALG[2:0] is only valid if AUTH_OP[3:0] is set to a hash or HMAC operation. 000 MD5 001 SHA1 010 SHA-224 011 SHA-256 100 SHA-384 101 SHA-512 111 NOP All other values reserved.

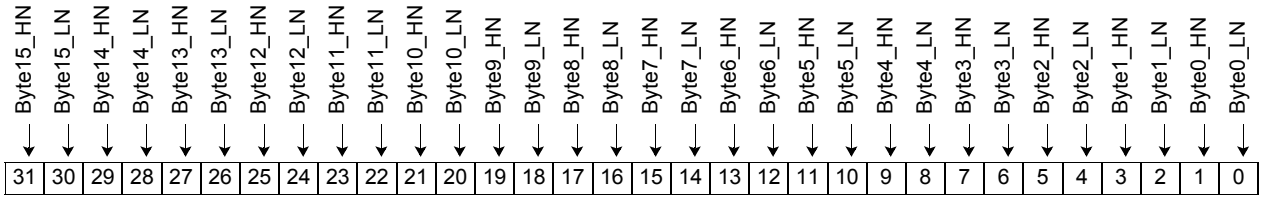
Field Name	Bits	Description
AUTH1_ALG[2:0]	28:26	Authentication Engine Algorithm 1. This field is used to set the authentication algorithm for the session. This field represents the one of the dual Authentication engines; AUTH0_ALG[2:0] represents the other Authentication engine. AUTH1_ALG[2:0] is only valid if AUTH_OP[3:0] is set to a hash or HMAC operation. 000 MD5 001 SHA1 010 SHA-224 011 SHA-256 100 SHA-384 101 SHA-512 111 NOP All other values reserved.
AUTH_OP[3:0]	25:22	Authentication Operation. This field is used to select the operation the Authentication engine will perform for the session. 0000 Hash calculation 0001 Reserved 0010 Reserved 0011 HMAC 0100 SSL3.0 MAC 0101 XCBC-MAC 0110 GMAC 0111 AES-GCM-MAC 1000 CMAC / EIA2 1001 UIA1 1010 EIA1 1011 EIA3 1100 P_hash All other values reserved. Notes: When calculating a stateful hash, the software must wait for the current command to complete before submitting the next command in the same session because next command's Initial Hash Vector (IHV) is included in the result of the current command. For decode operations, the Authentication engine calculates the MAC and compares the result to the MAC in the data stream.

Field Name	Bits	Description
MAC_KEY_LEN[4:0]	21:17	MAC Key Length. This field is used to set the MAC key length in words. The host may add dummy bytes if the length is not word-aligned. The XR9240 will discard the dummy bytes. The MACKEY value that is saved depends on the application. For SSL3.0 MAC, the MACKEY = MAC secret For XCBC MAC, the MACKEY = XCBC MAC key For all other modes, the MACKEY = IPAD and OPAD 0 128 words 1 1 word : 127127 words
MUTE_OFFSET[6:0]	16:10	Mute Offset. This field is used to set the offset from the authentication toggle token (see "Action Token Flags"). The mute offset has a range of 0 to 127 bytes.
AUTH_POS[1:0]	9:8	Authentication Engine Position. This field determines the position of the Authentication engine for the session, based on the encode direction. Refer to Figure 2-4 for an illustration of the Compression and Encryption engine's data path. If DIR = Encode direction: 00Authentication engine after CRC0 engine 01Authentication engine after Compression engine 10Authentication engine after Pad engine 11Authentication engine after Encryption engine If DIR = Decode direction: 00Authentication engine after Encryption engine 01Authentication engine after Decryption engine 10Authentication engine after Depad engine 11Authentication engine after Decompression engine
MAC_OUT_CTL	7	MAC Output Control. This bit controls how the XR9240 reports the MAC result. This field is only valid when AUTH_OP[2:0] is set to a MAC operation. Note that all authentication results (excluding MAC) are output to the sideband buffer. See "Transform Engine User Descriptor" for an explanation of tokens. If DIR = Encode direction: 0Insert MAC before AUTH toggle off token 1Output MAC to the sideband buffer If DIR = Decode direction: 0Remove MAC; the MAC should be adjacent and ahead of AUTH toggle off token 1Output MAC to the sideband buffer

Field Name	Bits	Description
AUTH_VERF_SRC	6	Authentication Verification Source. This bit is valid only when AUTH_VERF_EN is set. 0 Use source token 1 Use MAC value which is adjacent and ahead of the AUTH toggle off token
AUTH_VERF_EN	5	Authentication Verification Enable. This bit controls whether the XR9240 verifies the MAC for an authentication session. It should always be set for a decode session; it may be set for an encode session. 0 Do not verify authentication result 1 Verify authentication result
MAC_SZ[4:0] / SLICE_SZ[4:0]	4:0	MAC Size or Slice Size. This field represents MAC_SZ when AUTH_OP[2:0] is set to a type of MAC operation. 00000: Insert 32 half-words into the data stream ... 11111: Insert 1 half-word into the data stream The MAC value for MD5 is 16 bytes, for SHA1 is 20 bytes, SHA512 will generate 64 bytes. The XR9240 will truncate the MAC value according the configuration and insert it to the data stream. This field represents SLICE_SZ when SLICE in Control Word 0 is set to a slice hash operation. The range for the slice size is 64 to 64K bytes. 00000: 64 bytes 00001: 128 bytes ... 01010: 64K bytes All values > 01010 are reserved.

Session Control Word 3 - Mute Table

The Mute Table may be used by the host to inform the XR9240 to selectively ignore portions of the data byte stream. Muted data is passed through the XR9240 device unchanged. The mute table in this control word allows each nibble of the data stream to be muted.



Field Name	Bits	Description
Byte15_HN	31	Mute Data Stream Byte 15, bits [7:4] 0 Do not mute data stream byte 15, bits [7:4] 1 Mute data stream byte 15, bits [7:4]
Byte15_LN	30	Mute Data Stream Byte 15, bits [3:0] 0 Do not mute data stream byte 15, bits [3:0] 1 Mute data stream byte 15, bits [3:0]
. . .	. . .	. . .
Byte0_HN	1	Mute Data Stream Byte 0, bits [7:4] 0 Do not mute data stream byte 0, bits [7:4] 1 Mute data stream byte 0, bits [7:4]
Byte0_LN	0	Mute Data Stream Byte 0, bits [3:0]. 0 Do not mute data stream byte 0, bits [3:0] 1 Mute data stream byte 0, bits [3:0]

The mute offset, defined in "Session Control Word 2 - Authentication" as MUTE_OFFSET[6:0], may be used in conjunction with the mute table to delay the position at which the mute table takes effect relative to the Authentication token in the source data, as shown in the example below.

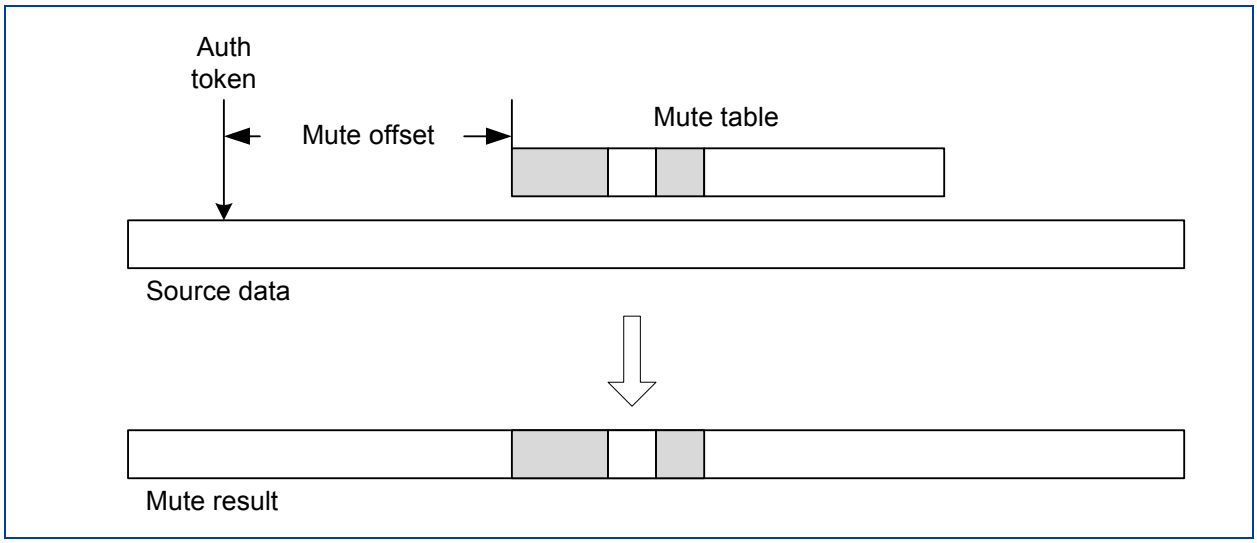


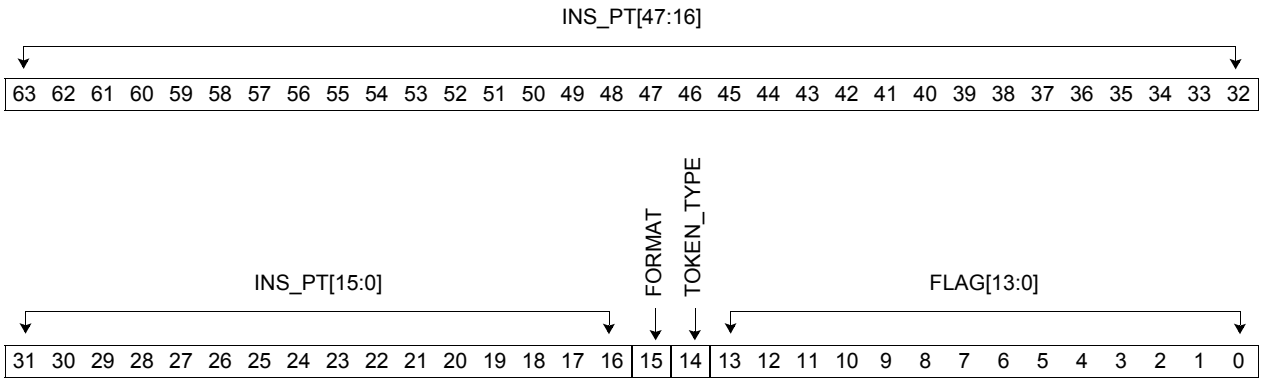
Figure 3-8. Effect of Mute Offset and Mute Table on Source Data

3.2.2.3 Transform Engine User Descriptor

The XR9240 translates the TE user descriptors into source tokens or action tokens. These tokens are then inserted into the source data stream as markers that instruct the computational engines how to process the payload data.

There are two TE user descriptor formats, a short format and a long format. The short format supports a session and user descriptor length up to 64KB. The long format supports a session and user descriptor length up to 2^{48} bytes. The XR9240 allows up to 64 short format or 32 long format TE user descriptors. A command cannot mix the long or short descriptor formats.

The TE user descriptor format is shown below.



Field Name	Bits	Description
INST_PT[47:0] for long format	63:16 for long format	Insertion Point. This field represents the insertion point for the token. The insertion point is calculated from the first byte of the command body. If the INS_PT=0, the token is inserted before byte 0.
INST_PT[15:0] for short format	31:16 for short format	
FORMAT	15	Descriptor Format. This bit defines whether this TE user descriptor uses the short or long format. 0 Short 1 Long
TOKEN_TYPE	14	Type of Token. This bit sets the type of token for this TE user descriptor. 0 Source token 1 Action token
FLAG[13:0]	13:0	Flag. The value and format of this field depends on the value of TOKEN_TYPE. Refer to "Source Token Flags", or "Action Token Flags" below for details.

Figure 3-9 illustrates how the TE user descriptor fields are translated by the XR9240 into a logically contiguous packet. Note that the host and FPGA will not see tokens inserted in the data stream.

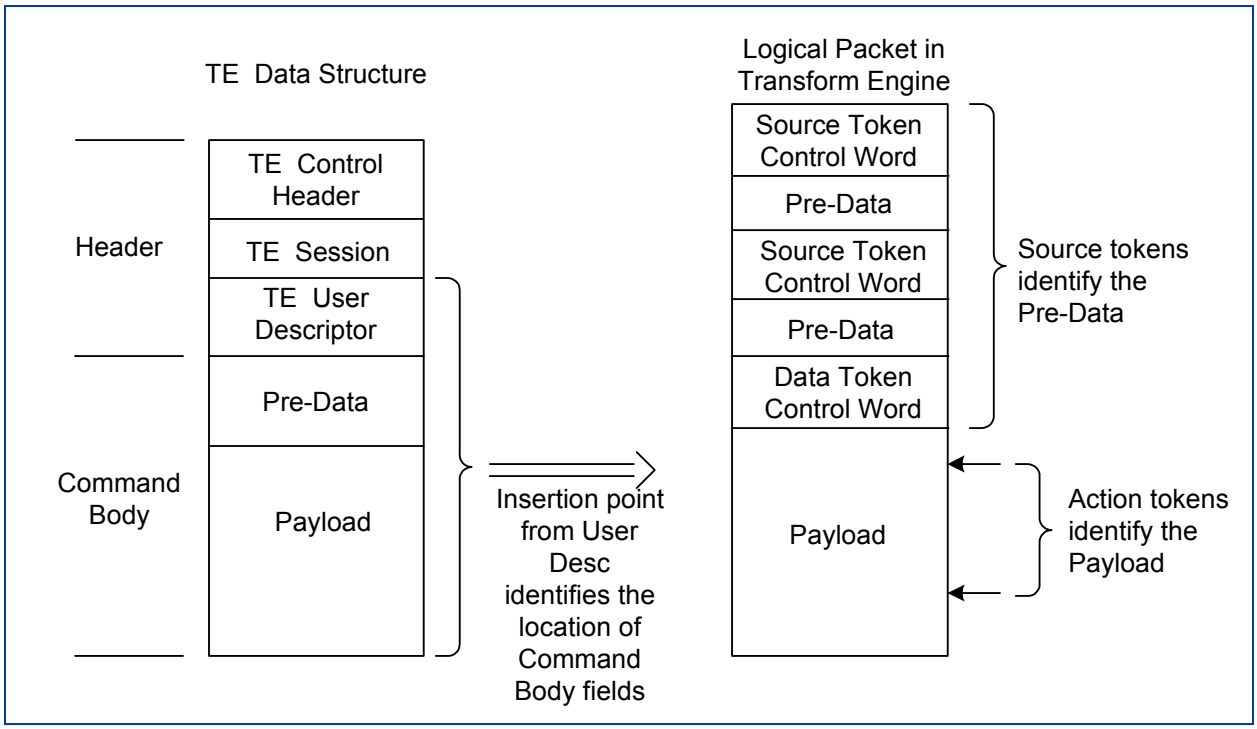


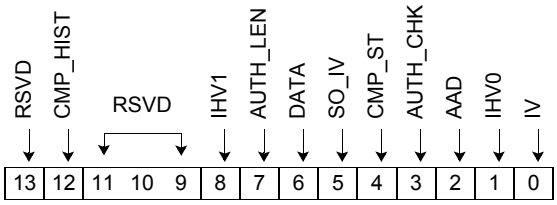
Figure 3-9. TE User Descriptor to XR9240 Logical Packet Translation

The source and action token flags are described in more detail below. Refer to "Using Tokens to Process Transform Commands" for an example of how tokens are used.

Source Token Flags

Source token flags apply to the pre-data portion of the command body. The source tokens identify which fields are present in the pre-data so that the engines correctly process the pre-data.

The Data token identifies the end of the pre-data and the beginning of the raw data. A Data token is required as the last source token unless there is no raw data.



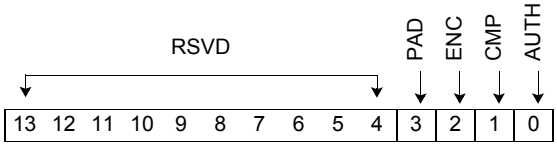
Field Name	Bits	Description
RSVD	13	Reserved.
CMP_HIST	12	Compression History. This token identifies whether the compression history is present. If present, the compression history field consists of ≤ 4+4K bytes, where the first 4 bytes are the field length and the remaining bytes are the compression history. 0 Compression history token not present 1 Compression history token present
RSVD	11:9	Reserved.
IHV1	8	IHV1 Flag. This bit is not valid if only one authentication algorithm is enabled for a command. This token flag provides the IHV for the second authentication engine if both authentication engines are processing two different algorithms. 0 IHV1 token is not present 1 IHV1 token present
AUTH_LEN	7	Authentication Length Flag. For all stateless Hash/HMAC/GCM/GMAC operations, this token is not used because the length may be deduced by the hardware. For stateful GCM operations, this token must be set for the last command; the first and middle commands do not require this token. When set, the source data will include 6 bytes for the source (cipher) length and 2 bytes for the AAD length. For all other stateful Hash/HMAC/GCM/GMAC operations, this token must be set for the last command; the first and middle commands do not require this token. When set, the source data will include 6 bytes for the AAD length. The AUTH_LEN token should always come before the AAD token. For all other algorithms, the file length is 6 bytes. 0 Authentication length token is not present 1 Authentication length token present
DATA	6	Data Flag. This flag indicates the starting point of the payload data. 0 Data token is not present 1 Data token present
S0_IV	5	S0 IV Flag. This token only applies to the last block of a stateful GCM/GMAC operation. When set, this token identifies the S0 IV used by the XR9240 to calculate the S0. For stateless GCM/GMAC operations, the S0_IV is the same as J0, which is specified by the IV token. 0 S0 IV token is not present 1 S0 IV token present

Field Name	Bits	Description
CMP_ST	4	Compression Stateful Flag. This token only applies to an intermediate block or last block of a stateful compression/decompression operation. 0 This token is not present 1 gzip CRC or the zlib ADLER32 token is present
AUTH_CHK	3	Authentication Check Flag. The value encapsulated in the token will be compared with authentication result (hash/MAC) in decode processing. 0 Authentication token not present 1 Authentication token present
AAD	2	AAD Flag. The data encapsulated in the token is only for GCM. GMAC operations do not use this token because the AAD is encapsulated into the data token. 0 AAD token not present 1 AAD token present
IHV0	1	IHV0 Flag. The data encapsulated in the token is the partial hash result for a stateful hash operation. 0 IHV0 token not present 1 IHV0 token present
IV	0	IV Delimiter Flag. This token is also used for RC4 SBOX. 0 IV token not present 1 IV token present

Action Token Flags

Action token flags operate on the payload. Action tokens define the regions of the source data that each computational engine (Compression, Encryption, Pad, Authentication) will operate on. Action tokens are defined as toggle tokens: the first token marks the start of the active region and the second token marks the end of the active region.

Action tokens may have multiple flags set at the same time. For example, for a combined compression + encryption command where the compression and encryption operation starts at the same byte, both compression and encryption token flags would be set in a single action token.



Field Name	Bits	Description
RSVD	13:5	Reserved.

Field Name	Bits	Description
PAD	3	Pad Toggle Flag. For an encode operation, this is the toggle flag for the Pad engine. If real time verification is enabled for an encode operation, this flag also toggles the Depad engine. For a decode operation, this is the toggle flag for the Depad engine. The first active toggle token marks the starting point of the data to be padded. The active second toggle token marks the end of the data to be padded. 0 Pad toggle flag not set 1 Pad toggle flag set
ENC	2	Encryption Toggle Flag. For an encode operation, this is the toggle flag for the Encryption engine. If real time verification is enabled for an encode operation, this flag also toggles the Decryption engine. For a decode operation, this is the toggle flag for the Decryption engine. The first active toggle token marks the starting point of the data to be encrypted. The active second toggle token marks the end of the data to be encrypted. 0 Encryption toggle flag not set 1 Encryption toggle flag set
CMP	1	Compression Toggle Flag. For an encode operation, this is the toggle flag for the Compression engine. If real time verification is enabled for an encode operation, this flag also toggles the Decompression engine. For a decode operation, this is the toggle flag for the Decompression engine. The first active toggle token marks the starting point of the data to be compressed. The active second toggle token marks the end of the data to be compressed. 0 Compression toggle flag not set 1 Compression toggle flag set
AUTH	0	Authentication Toggle Flag. The first active toggle token marks the starting point of the data to be authenticated. The active second toggle token marks the end of the data to be authenticated. 0 Authentication toggle flag not set 1 Authentication toggle flag set

3.2.3 Transform Engine Command Body

The command body for a TE command consists of optional pre-data and the source data.

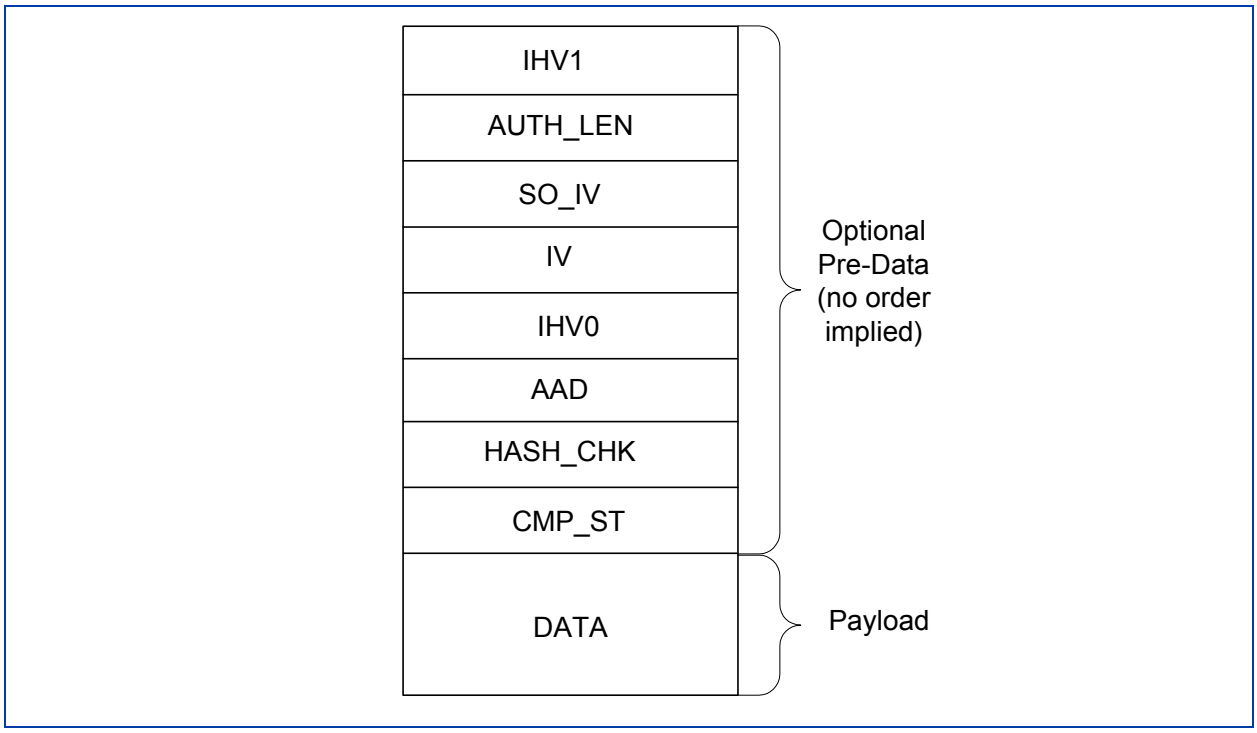


Figure 3-10. Command Body Example Format

3.2.3.1 Pre-Data Input Requirements

The predata is optional and based on the requirements of the command. The order of the fields is not fixed; the "Source Token Flags" identify the field's presence, order and length.

The position of the field is identified by the "insertion point" of the source token. The length of the field is deduced by the delta between the insertion point of the current token and the insertion point of the following source token.

Refer to "Using Tokens to Process Transform Commands" for general token usage rules.

3.2.3.2 Data Input Requirements

This sections defines the alignment and size requirement for the source data buffers that must be adhered to yield predictable output.

Table 3-2. Compression Source Buffer Data Requirements

Algorithm	FB	LB	CMP_ST	CMP_HIST	Data
Deflate	0	0		4B hist_len + ≤ 4KB hist	arbitrary
	0	1			
	1	0		Optional	
	1	1			

Table 3-2. Compression Source Buffer Data Requirements

Algorithm	FB	LB	CMP_ST	CMP_HIST	Data
gzip	0	0	4B CRC + 4B size	4B hist_len + ≤ 4KB hist	arbitrary
	0	1			
	1	0		Optional	
	1	1			
zlib	0	0	4B ADLER32	4B hist_len + ≤ 4KB hist	arbitrary
	0	1			
	1	0		Optional	
	1	1			
LZS / eLZS	X	X			arbitrary
Notes: FB = First Block LB = Last Block CMP_ST = Compression State CMP_HIST = Compression History					

Table 3-3. Decompression Source Buffer Data Requirements

Algorithm		LB	CMP_ST	CMP_HIST	Data
Deflate	0	0	1B (start_bit)	4B hist_len + ≤ 32KB hist	arbitrary
	0	1			
	1	0		Optional	
	1	1			
gzip	0	0	4B CRC + 1B (start_bit)	4B hist_len + ≤ 32KB hist	arbitrary
	0	1			
	1	0		Optional	
	1	1			
zlib	0	0	4B ADLER32 + 1B (start_bit)	4B hist_len + ≤ 32KB hist	arbitrary
	0	1			
	1	0		Optional	
	1	1			
LZS / eLZS	X	X			arbitrary
Notes: FB = First Block LB = Last Block CMP_ST = Compression State CMP_HIST = Compression History					

Table 3-4. Encryption Source Buffer Data Requirements

Algorithm	Packet Based Information		Session Based Information
	IV Size (Bytes)	Data Alignment	Key Size (Bytes)
3DES-CBC	8	64-bit	24
AES-128-CBC	16	128-bit	16, 16B key + 16B fkey for RTV
AES-128-CTR	16	arbitrary	16
AES-128-ECB	16	128-bit	16, 16B key + 16B fkey for RTV
AES-128-F8	16	arbitrary	16 k_e + 16 m (Refer to RFC 3711 SRTP protocol definition)
AES-192-CBC	16	128-bit	24, 24B key + 24B fkey for RTV
AES-192-CTR	16	arbitrary	24
AES192-ECB	16	128-bit	24, 24B key + 24B fkey for RTV
AES-192-F8	16	arbitrary	24 k_e + 24 m
AES-256-CBC	16	128-bit	32, 32B key + 32B fkey for RTV
AES-256-CTR	16	arbitrary	32
AES-256-ECB	16	128-bit	32, 32B key + 32B fkey for RTV
AES-256-F8	16	arbitrary	32 k_e + 32 m
AES-XTS-256	16	arbitrary	16B key1 + 16B key2, 16B key1+16B fkey1+16B key for RTV
AES-XTS-512	16	arbitrary	64, 32B key1 + 32B fkey1 + 32B key2 for RTV
SNOW3G-EEA1	16	arbitrary	16
ZUC-EEA3	16	arbitrary	16
KASUMI-UEA1	8	arbitrary	8
RC4	0 for first block, 258 for subsequent blocks	arbitrary	1-256 for first block, 0 for subsequent blocks
Notes: 1. Some algorithms require both the encryption key and the decryption key (fkey) for real time verification (RTV).			

Table 3-5. Authentication Source Buffer Data Requirements

Algorithm				Packet Based Information					Session Based Information
	FB	LB	IHV Size (Bytes)	IV Size (Bytes)	SO-IV Size (Bytes)	AAD Size (Bytes)	Auth_len Size (Bytes)	Data Alignment	Key Size (Bytes)
MD5	0	0	16	X	X	X	X	512-bit	X
	0	1	16	X	X	X	X	arbitrary	X
	1	0	X	X	X	X	X	512-bit	X
	1	1	X	X	X	X	X	arbitrary	X
SHA1	0	0	20	X	X	X	X	512-bit	X
	0	1	20	X	X	X	X	arbitrary	X
	1	0	X	X	X	X	X	512-bit	X
	1	1	X	X	X	X	X	arbitrary	X
SHA224, SHA256	0	0	32	X	X	X	X	512-bit	X
	0	1	32	X	X	X	X	arbitrary	X
	1	0	X	X	X	X	X	512-bit	X
	1	1	X	X	X	X	X	arbitrary	X
SHA384, SHA512	0	0	64	X	X	X	X	1024-bit	X
	0	1	64	X	X	X	X	arbitrary	X
	1	0	X	X	X	X	X	1024-bit	X
	1	1	X	X	X	X	X	arbitrary	X
HMAC-MD5 SSLv3-MAC-MD5	0	0	16	X	X	X	X	512-bit	X
	0	1	16	X	X	X	6B file length	arbitrary	16B OPAD
	1	0	X	X	X	X	X	512-bit	16B IPAD
	1	1	X	X	X	X	X	arbitrary	16B IPAD + 16B OPAD
HMAC-SHA1	0	0	20	X	X	X	X	512-bit	X
	0	1	20	X	X	X	6B file length	512-bit	20B OPAD
	1	0	X	X	X	X	X	512-bit	20B IPAD
	1	1	X	X	X	X	X	512-bit	20B IPAD + 20B OPAD
SSLv3-MAC-SHA1	0	0	20	X	X	X	X	512-bit	20B
	0	1	20	X	X	X	6B file length	512-bit	20B
	1	0	X	X	X	X	X	512-bit	20B
	1	1	X	X	X	X	X	512-bit	20B

Table 3-5. Authentication Source Buffer Data Requirements

Algorithm				Packet Based Information					Session Based Information
	FB	LB	IHV Size (Bytes)	IV Size (Bytes)	SO-IV Size (Bytes)	AAD Size (Bytes)	Auth_len Size (Bytes)	Data Alignment	Key Size (Bytes)
HMAC-SHA224/256 SSLv3-MAC-SHA256	0	0	32	X	X	X	X	512-bit	X
	0	1	32	X	X	X	6B file length	arbitrary	32B OPAD
	1	0	X	X	X	X	X	512-bit	32B IPAD
	1	1	X	X	X	X	X	arbitrary	32B IPAD + 32B OPAD
HMAC-SHA384/512	0	0	64	X	X	X	X	1024-bit	X
	0	1	64	X	X	X	6B file length	arbitrary	64B OPAD
	1	0	X	X	X	X	X	1024-bit	64B IPAD
	1	1	X	X	X	X	X	arbitrary	64B IPAD + 64B OPAD
GMAC-128	0	0	16	X	X	any	X	128-bit	share 16B enc key
	0	1	16	X	16	any	6B file length + 2B AAD_len	arbitrary	
	1	0	X	16	X	any	X	128-bit	
	1	1	X	16	X	any	X	arbitrary	
GMAC-192	0	0	16	X	X	any	X	128-bit	share 24B enc key
	0	1	16	X	16	any	6B file length + 2B AAD_len		
	1	0	X	16	X	any	X		
	1	1	X	16	X	any	X		
GMAC-256	0	0	16	X	X	any	X	128-bit	share 32B enc key
	0	1	16	C	16	any	6B file length + 2B AAD_len		
	1	0	X	16	X	any	X		
	1	1	X	16	X	any	X		
X-CBC-MAC, CMAC	0	0	16	16	X	X	X	128-bit	16B MAC key
	0	1	16	16	16	X	X		
	1	0	X	16	X	X	X		
	1	1	X	16	X	X	X		

Table 3-5. Authentication Source Buffer Data Requirements

Algorithm				Packet Based Information					Session Based Information
	FB	LB	IHV Size (Bytes)	IV Size (Bytes)	SO-IV Size (Bytes)	AAD Size (Bytes)	Auth_len Size (Bytes)	Data Alignment	Key Size (Bytes)
EIA1/EIA3	X	X	16	X	X	X	X	≤ 2500 bytes	share 16B enc key with EEA1
UIA1	X	X	8	X	X	X		≤ 8192 bytes	share 16B enc key with UEA1
P_hash(x)	X	X	X	X	X	X	2B loop_num	≤ 77 bytes	Same as the HMAC(x)
Notes: FB = First Block LB = Last Block									

Table 3-6. AEAD Source Buffer Data Requirements

Algorithm				Packet Based Information					Session Based Information
	FB	LB	IHV Size (Bytes)	IV Size (Bytes)	SO-IV Size (Bytes)	AAD Size (Bytes)	Auth_len Size (Bytes)	Data Alignment	Enc Key Size (Bytes)
GCM-128	0	0	16	16	X	any	X	128	16
	0	1	16	16	16	any	6B file length + 2B AAD_len		
	1	0	X	16	X	any	X		
	1	1	X	16	X	any	X		
GCM-192	0	0	16	16	X	any	X	128	24
	0	1	16	16	16	any	6B file length + 2B AAD_len		
	1	0	X	16	X	any	X		
	1	1	X	16	X	any	X		

Table 3-6. AEAD Source Buffer Data Requirements

Algorithm				Packet Based Information					Session Based Information
	FB	LB	IHV Size (Bytes)	IV Size (Bytes)	SO-IV Size (Bytes)	AAD Size (Bytes)	Auth_len Size (Bytes)	Data Alignment	Enc Key Size (Bytes)
GCM-256	0	0	16	16	X	any	X	128	32
	0	1	16	16	16	any	6B file length + 2B AAD_len		
	1	0	X	16	X	any	X		
	1	1	X	16	X	any	X		
Notes: FB = First Block LB = Last Block									

3.3 Public Key Source Buffer Format

The Public Key source buffer format shown in Figure 3-11 contains a header and the command body. The order of the command body fields are flexible, however the CSR field must be the last field because it starts the PK calculation. The command body is optional.

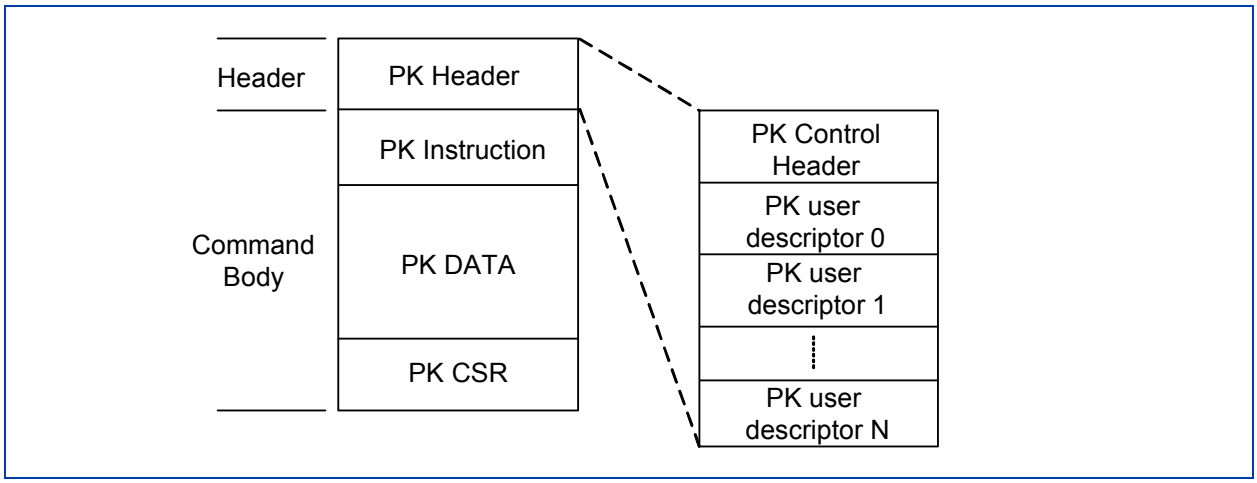
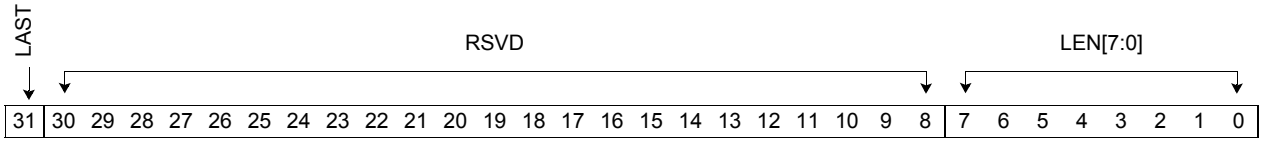


Figure 3-11. Detailed PK Source Buffer Format

3.3.1 PK Header

The PK header consists of a control header and up to 16 user descriptors. This section describes the format of these structures.

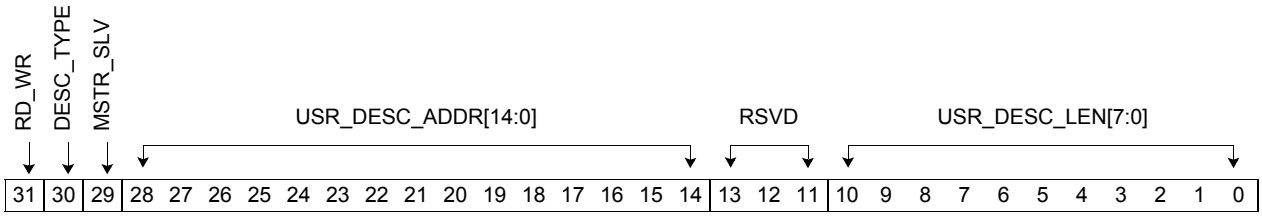
3.3.1.1 PK Control Header



Field Name	Bits	Description
LAST	31	Header Last Flag. This bit must always be set to one for a PK operation. 0 Current block is not the last block 1 Current block is the last block
RSVD	30:8	Reserved.

Field Name	Bits	Description
LEN[7:0]	7:0	Header Length. The field represents the length of the header in words, including the PK control header. The maximum value is 17 (1 PK_CTL_HDR + 16 PK_USR_DESC).

3.3.1.2 PK User Descriptor



Field Name	Bits	Default	Description
RD_WR	31	0	Read/Write Flag. This bit defines whether the PK operation is a read or write command. 0 Write 1 Read
DESC_TYPE	30	0	Descriptor Type. This bit defines the execution time for the PK operation. 0 Non-Blocking: Execute command immediately 1 Blocking: Wait until completion of PK command before executing command
MSTR_SLV	29	0	Master/Slave Flag. This bit indicates whether the PK command is for the PK engine master or slave. For most PK commands this bit should be set to Master. See "Public Key Module" for more information. 0 Master 1 Slave
USR_DESC_ADDR[14:0]	28:14	0	User Descriptor Address. The address pointer to the PK user descriptor.
RSVD	13:11	0	Reserved.
USR_DESC_LEN[10:0]	10:0	0	User Descriptor Length. This field represents the length of the user descriptor in words.

3.3.2 PK Payload

3.3.2.1 PK Instruction

The address range of the PK user descriptor defines the type of instruction access (see Table 6-8). The actual instruction is coded in firmware.

3.3.2.2 PK Data

The PK source data could include the LIR, BER, MMR, TSR, or FPR data. The data is written into the corresponding data registers of the PK engine if a write operation was specified in the user descriptor. See "Public Key Module" for more information.

3.3.2.3 PK Control and Status

The PK Control and Status register value will initiate the PK command. See "Public Key Module" for more information

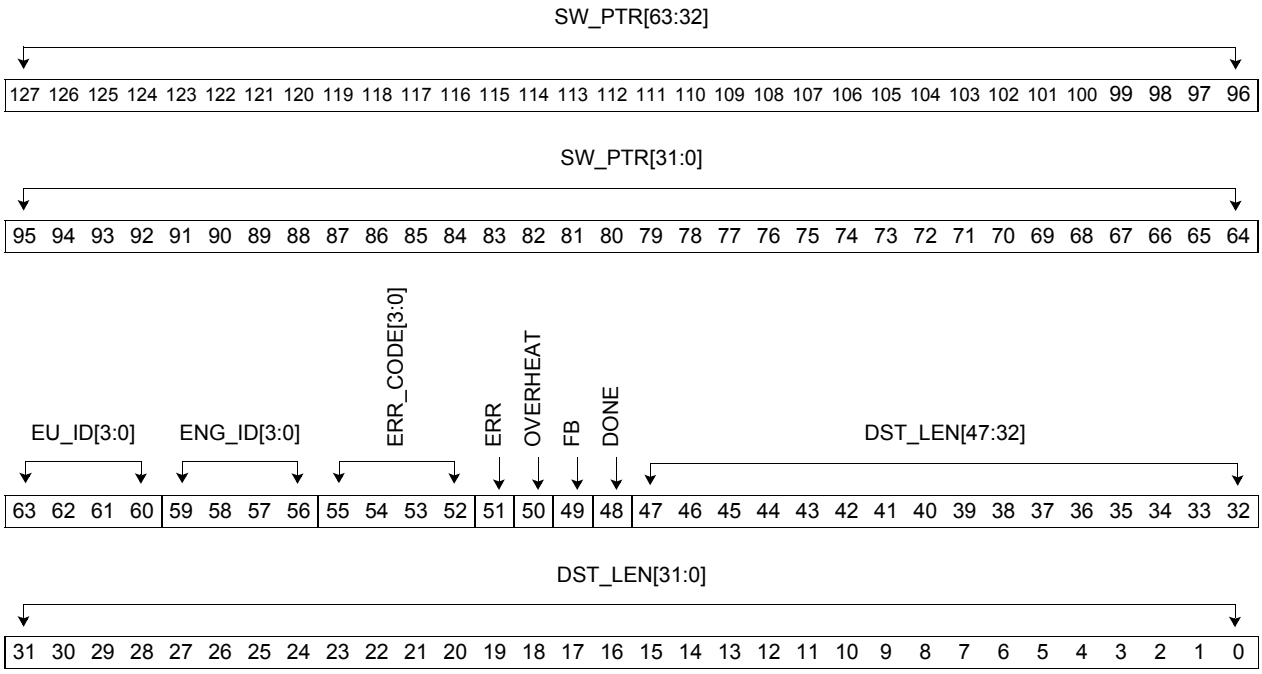
3.4 Result Pointer Ring

The result pointer ring contains the completed command's actual resultant data length and execution status (successful completion or errors occurred). The XR9240 writes to the result pointer ring and the host reads from the result pointer ring. The XR9240 will write to the result pointer ring when a command completes and then update its result ring write pointer appropriately.

Note that the result pointer ring is used in conjunction with the destination descriptor that is defined in the "Command Structure". The destination descriptor contains a pointer to the result data, whereas the result pointer ring contains the resultant data length and command execution status.

The number of entries in the result pointer ring may be larger than the number of entries in the command pointer ring. The base address of the result pointer ring must be aligned to a 16-byte boundary and the result pointer ring must reside entirely in a physically contiguous range of memory.

The error bits remain set until cleared by the host software.



Field Name	Bits	Description
SW_PTR[63:0]	127:64	Software Pointer. This field is a virtual address pointer whose usage is defined by software. Software uses this pointer to determine the corresponding command structure of the destination descriptor. With this pointer, an entry in the command pointer ring could be overwritten immediately after the command pointer is read by the XR9240. See Note 1 below.

Field Name	Bits	Description
EU_ID[3:0]	63:60	Execution Unit Identifier. For PK commands, this field will be set to zero. For TE commands, this field identifies the Transform engine that reported the error. This field is intentionally larger than required to allow for future expansion. 0000TE 0 | 0111TE 7 All other values are reserved.
ENG_ID[3:0]	59:56	Engine Identifier. This field identifies the engine that first reported the error. See "Error ID Encoding".
ERR_ID[3:0]	55:52	Error Identifier. This field contains the detailed error identifier code for the failed command. See "Error ID Encoding".
ERR	51	Error Flag. The XR9240 will set this bit if any type of error occurred during command processing. 0 No error occurred 1 Error occurred
OVERHEAT	50	Overheat Flag. This bit will be set whenever the alarm signal from temperature sensor is asserted. It is up to the host software to decide whether to discard the command. 0 Device temperature was within allowable range during processing of the command 1 Overheated condition detected during processing of the command
FB	49	Free Buffer Pointer Flag. The Free buffer enable bit in the "Ring Configuration Register" must be set to one by the host before the XR9240 may use the free buffer. If enabled and all destination buffers are consumed, the XR9240 will set this bit to indicate it used the free buffer to store the result data. When the XR9240 uses the free buffer, the OVFL bit will be set by the XR9240 if the free buffer space is consumed or the command structure cannot hold any more free buffer addresses. 0 XR9240 is not using the free buffer 1 XR9240 is using the free buffer
DONE	48	Done Flag. (See Note 1 below.) This bit may be polled by the host software to determine if the corresponding command has finished. When set, it indicates that there are valid entries in the result pointer ring. The DONE bit must be cleared by the host software after the destination data has been processed. 0 Command has not completed 1 Command has completed

Field Name	Bits	Description
DST_LEN[47:0]	47:0	Destination Data Length. The length of the data includes all fields in the destination buffer, including the free buffer but excluding the sideband data. This field is written to by the XR9240 after the command completes and is read by the host.
Note 1: On some hardware platforms, the DONE bit may be set before the error fields in the result entry have been written. To prevent a potential race condition, it is recommended that the host software clear the SW_PTR[63:0] field before submitting a new command. After the host software finds that the DONE bit has been set, the host software should then verify that the SW_PTR[63:0] field was modified.		

Figure 3-12 shows an example of how the result pointer ring and command pointer ring are used to process a command and its results. To simplify the illustration, this example shows a command pointer ring only with 4 entries, but the actual minimum command pointer ring size is 32 entries.

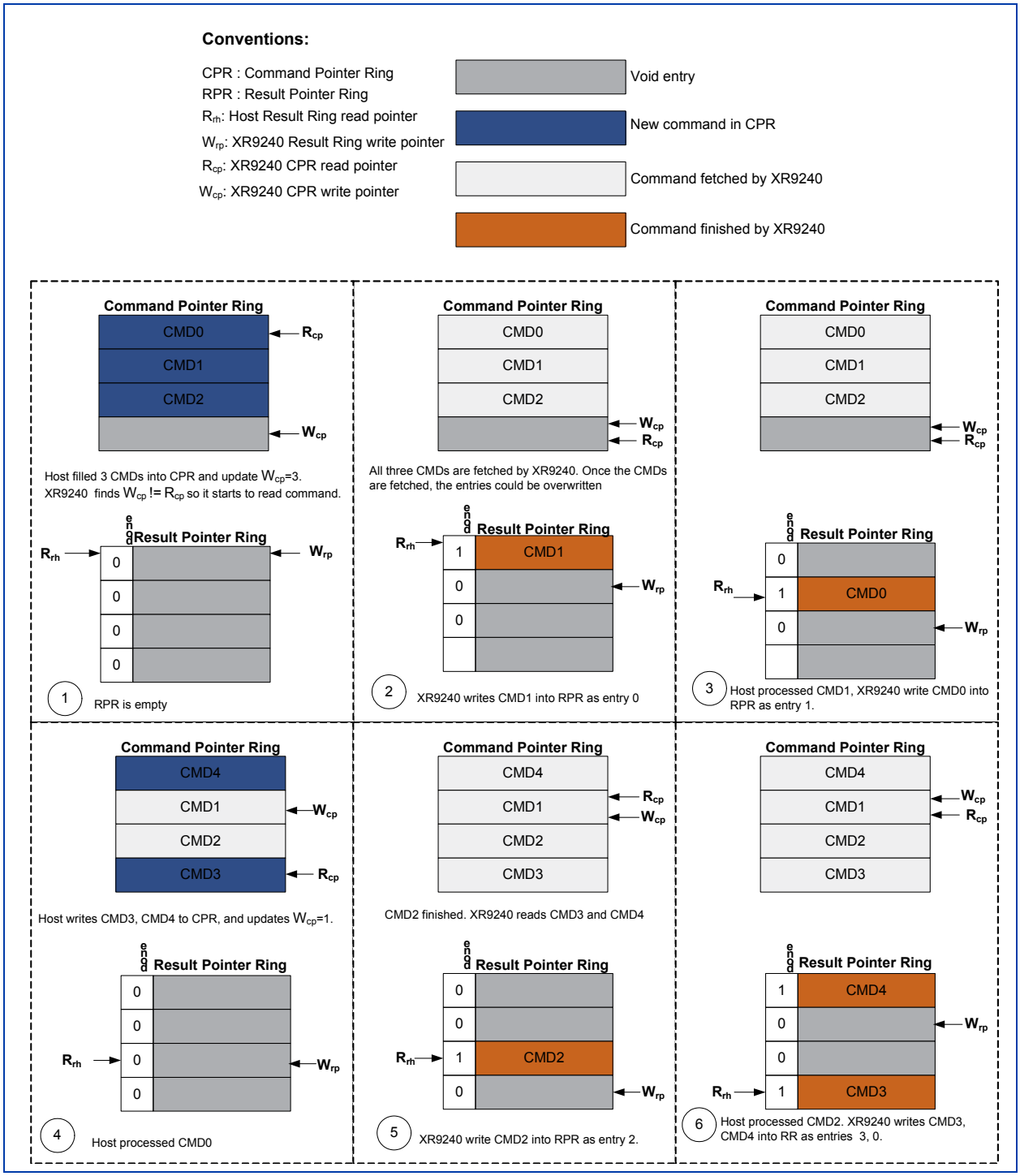


Figure 3-12. Result Pointer Ring Example

3.4.1 Destination Buffer Organization

The destination data to be output to the destination data buffer is specified in the TE session descriptors. The user may select which portion of the result may be sent to the destination buffer in the "Transform Engine Session Descriptor".

For storage applications, the predata may be dropped to save PCIe bandwidth, however some network applications may require the predata.

For applications that only require the sideband data, such as the hash result or compression length, the inband result could be dropped.

Table 3-7 lists the destination buffer output organization options.

Table 3-7. Destination Buffer Output Organization Options

DROP_INB	DROP_PRE	CMP_HIST	CMP_ST	IHV	IV	So_IV	AAD	AUTH	AUTH_LEN	Inband Result	Sideband Result
0	0	√	√	√	√	√	√	√	√	√	Optional
0	1									√	Optional
1	0	√	√	√	√	√	√	√	√		Optional
1	1										√

Notes.

1. The session structure including the encryption key and MAC key is always discarded from the output.

3.4.2 Sideband Data

Both the Transform Engines and FPGA may generate sideband data. As shown in the example in Figure 3-13, a sideband data buffer may consist of multiple records. Each record has a 4-byte header that specifies the data type and its length. The last record is identified by a header with zero length.

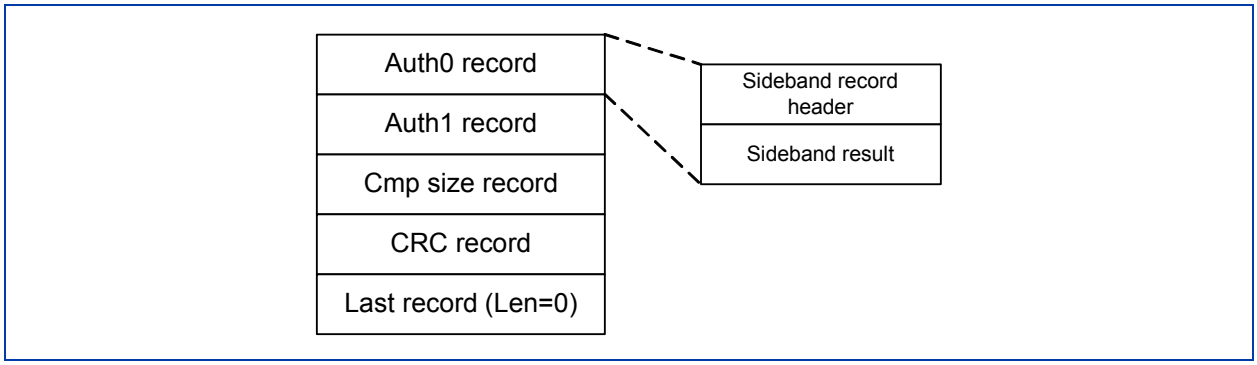
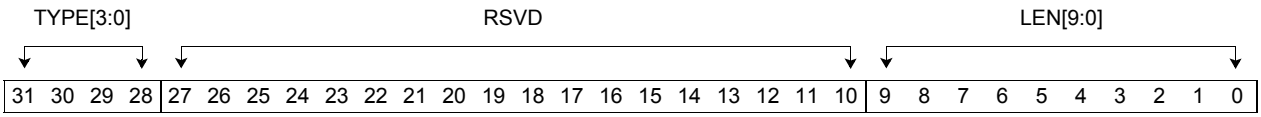


Figure 3-13. Sideband Buffer Example

The format of the header is shown below.



Field Name	Bits	Description
TYPE[3:0]	31:28	Sideband Data Type. See Table 3-8 for additional information. 0000Auth Eng 0 data when result is configured to output to sideband buffer 0001Auth Eng 1 data when result is configured to output to sideband buffer. If real time verification also enabled, only Auth Eng 0 result is output 0010Compression size 0011CRC sideband data 0100RC4 SBOX+i+j 0101Compression alignment 0110gzip CRC/zlib Adler32 0111Pad output All other values reserved.
RSVD	27:10	Reserved.
LEN[9:0]	9:0	Length of Sideband Data. This field is the length of the sideband data in bytes, not including the 4 byte header. Refer to Table 3-8 for details. A record with LEN=0 indicates the end of the sideband buffer.

Table 3-8. Sideband Result Length based on TYPE[3:0]

TYPE[3:0]	Description	Algorithm	Sideband Result Length	Sideband Record Length[1]
4'b0000 or 4'b0001	Auth 0 or Auth 1	MD5	16B	20B
		SHA1	20B	24B
		SHA224	32B for first or middle; 28B for last or complete	36B for first or middle; 32B for last or complete
		SHA256	32B	36B
		SHA384	64B for first or middle; 48B for last or complete	68B for first or middle; 52B for last or complete
		SHA512	64B	68B

Table 3-8. Sideband Result Length based on TYPE[3:0]

TYPE[3:0]	Description	Algorithm	Sideband Result Length	Sideband Record Length[1]
4'b0010	Compression size. For compression, TYPE[3:0] represents the compression size. If the command is a stateful operation, the size includes the length of the previous fragments of the file. For decompression, TYPE[3:0] represents the byte length of invalid output (dst_last_blk_pos) from the incomplete block at the end of the fragment. Based on this information, the host will discard this invalid output.	Cmp/Decmp	Cmp 6B, Decmp 4B	Cmp 12B Decmp 8B
4'b0011	CRC result. If the CRC_SB bit is set in the command session, the CRC of the input payload data (not including the predata) will be output.		4B	8B
4'b0100	RC4. sbox + i +j is output to sideband buffer.		258B	264B
4'b0101	Compression Alignment. This option only applies to stateful compression/ decompression. For compression, TYPE[3:0] represents the number of padding bits in the last byte before z_flush_sync. For decompression, the sideband output record contains the consumed source data length. A deflate block is not aligned on a byte boundary, therefore its decomp length is measured in bits.	Cmp/Decmp	Cmp 1B, Decmp 4B	Cmp 5B, Decmp 8B

Table 3-8. Sideband Result Length based on TYPE[3:0]

TYPE[3:0]	Description	Algorithm	Sideband Result Length	Sideband Record Length[1]
4'b0110	Compression CRC. This option only applies to stateful compression/ decompression. For compression, TYPE[3:0] represents the CRC/ ADLER32 intermediate result. For decompression, TYPE[3:0] represents the intermediate CRC/ADLER32 of the valid output. The output from the incomplete deflate block at the end of the fragment is not included in this value.	Cmp/Decmp	Cmp gzip 1B, Cmp zlib 4B, Decmp 4B	8B
4'b0111	Pad If ESP=0, pad length is output. If ESP=1, both pad length and next_hdr are output.		1B when ESP = 0, 2B when ESP = 1	8B
Notes: 1. Sideband record length includes the sideband record header.				

3.5 Free Buffer Pointer Ring

The host software may use the XR9240 Free Buffer to store destination data if all the command's destination buffers are consumed to prevent an overflow condition. Some applications may elect to use only free buffers instead of destination buffers. Figure 3-14 illustrates the Free Buffer Pointer Ring concept. Internal to the XR9240, each virtual function has its own free buffer pointer ring.

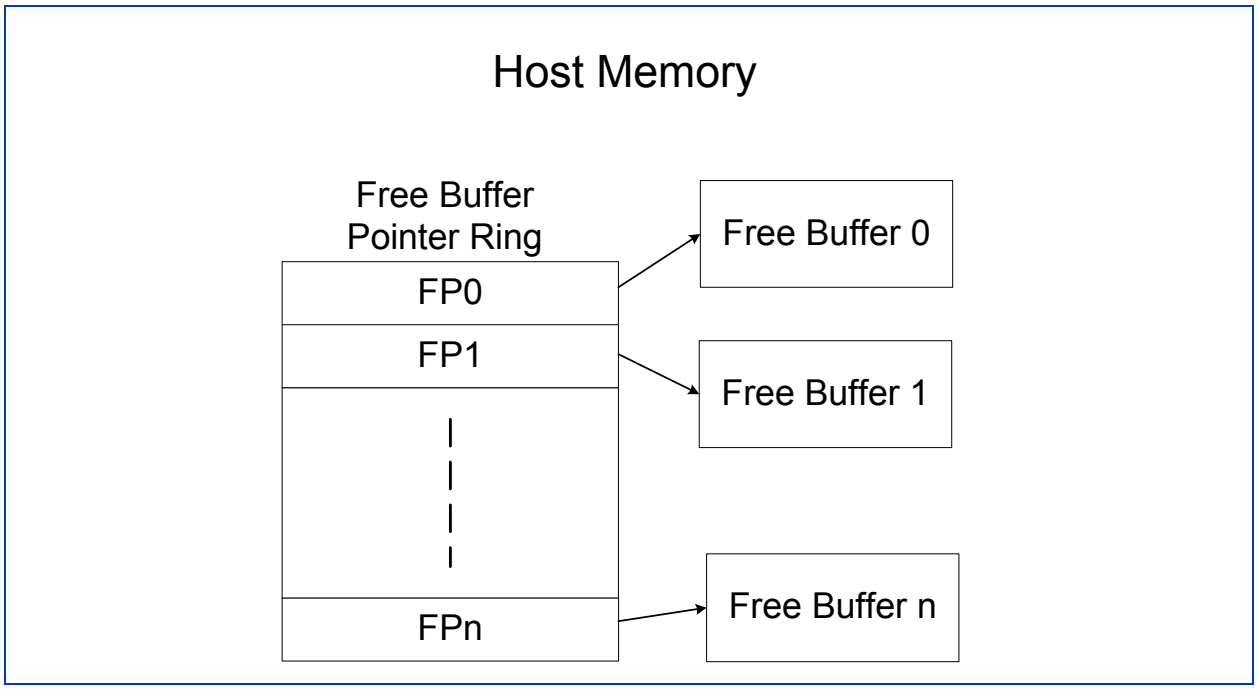


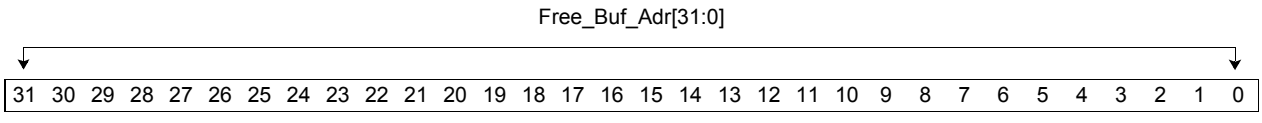
Figure 3-14. Free Buffer Pointer Ring

The host software must maintain the free buffer pointer ring and ensure that the free buffer pointer ring base address is 8-byte aligned. The entries in the free buffer pointer ring should be returned in consecutive order.

The minimum number of entries in the free buffer pointer ring is 32 entries. The length of the free buffer pointer ring must be an exponential power of 2 that is greater than or equal to 32.

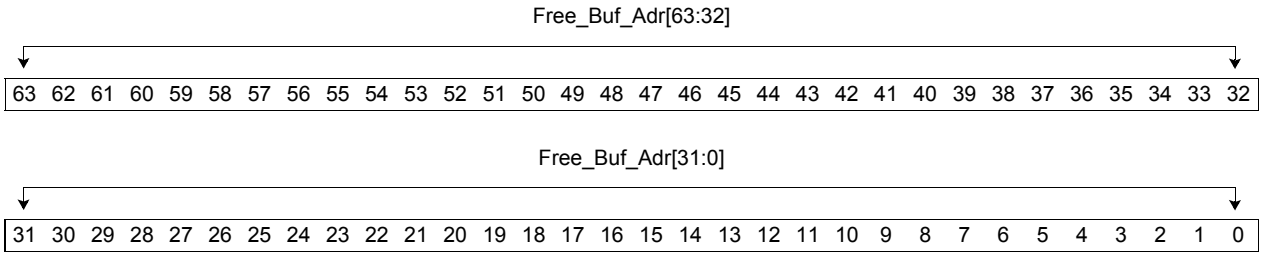
The XR9240 does not maintain a "full" bit for the free buffer pointer ring; it is the responsibility of the host not to overflow the free buffer pointer ring.

32-bit address Mode Format



Field Name	Bits	Description
Free_Buf_Adr[31:0]	31:0	Free Buffer Pointer Starting Address. The starting physical address of the descriptor free buffer. The XR9240 free buffer physical addresses must be located in the first 4GB (0 - 0xFFFFFFFF). The Free Buffer Pointer starting address must be aligned on an 8 byte boundary. However, the free buffer itself must only be byte aligned.

64-bit address Mode Format



Field Name	Bits	Description
Free_Buf_Adr[63:0]	63:0	Free Buffer Pointer Starting Address. The starting physical address of the descriptor free buffer. The XR9240 free buffer physical addresses must be located in the first 4GB (0 - 0xFFFFFFFF). This field must be aligned on an 8 byte boundary.

The host must set the FB_EN bit in the "Ring Configuration Register" if using the free buffers. If the host clears the DST_BUF bit, the XR9240 will send all destination data to the free buffers instead of the destination buffers.

The host writes to the free buffer pointer ring at initialization, and the XR9240 reads from the ring when needed. After the XR9240 uses a free buffer entry, it will write the buffer address into the command structure.

If a command had several destination buffers, the first free buffer address will be written after the last destination buffer. In this case, both the last destination buffer and the last free buffer address will have the LAST flag set. The XR9240 will not clear the LAST flag for the last destination buffer. After a command completes, the host should check the FB bit in the result pointer descriptor. If the FB_EN bit is set, the two LAST flags will also be set.

If the command has no destination buffer specified (DST_BUF bit in the "Ring Configuration Register" is cleared), the free buffer address will start from the first destination buffer descriptor. In this case, the host will only see one LAST flag set.

The host should configure the free buffer pointer ring size using the FBR_SZ[3:0] in the "Ring Configuration Register" during initialization. When a free buffer is used, an interrupt may be sent to the host if enabled.

A command will return with a free buffer overflow error if the command structure has no more space to hold free buffer addresses, or all free buffers were consumed.

4 XR9240Command Operation

4.1 Basic Command Operation Sequence

The basic sequence for the host to process a command is:

1. The host configures the "XR9240 Configuration Registers" for the physical function according to the application environment.

2. The host writes to the XR9240 "Ring Allocation Register" to assign command/result pointer ring pairs to the physical and virtual functions.

3. The host writes to the "Ring Registers" to configure the command, result, free buffer pointer ring addresses and read/write pointers.

4. The host builds the source buffer's headers and data according to the "Transform Channel Source Buffer Format" or the "Public Key Source Buffer Format".

5. The host creates a command structure in host memory that follows the "Command Structure Format" and submits the address pointer of the command to the "Command Pointer Ring".

6. The host prepares the destination buffer and/or free buffer space in host memory.

7. The host then writes the index of the command in the command pointer ring to the XR9240 "Command Pointer Ring Write Pointer Register" and updates its own write pointer. The XR9240 also maintains a "Command Pointer Ring Read Pointer Register" which may be read by the host at any time. It is the responsibility of the host not to overflow the command pointer ring by wrapping the write pointer back to the read pointer. Note that if several commands are available for submission at one time, the host may build all of the command structures and then issue only a single write to the write pointer register, if desired.

8. As long as the read and the write pointer registers of a physical/virtual function's command pointer ring are not equal, the ring is not empty. The XR9240 will select which ring to serve according to Class of Service (COS) bandwidth allocation configuration set in the "Transform Channel Class of Service Register" or the "Public Key Class of Service Register" as appropriate.

9. The command that wins the COS arbitration is fetched from the ring and written to command manager.

10. The command manager dispatches the command to the appropriate data channels.

11. The XR9240 DMA fetches the source data from host memory.

12. The XR9240 DMA organizes the source buffer into a logical packet and sends it to the Transform Engine or Public Key Engine.

13. The Transform Engine or Public Key Engine processes the source data.

14. The Transform Engine or Public Key Engine writes the destination data to XR9240 DMA.

15. The XR9240 DMA writes the destination data to host memory.

16. The XR9240 DMA writes the result descriptor to the result pointer ring.

17. If enabled, an interrupt is triggered to inform the host that the command has completed.

18. Host processes the completed command(s).

4.2 Transform Processing Sequence

This section describes the processing that is unique to transform commands.

The XR9240 has eight transform channels that provide superior parallel processing capabilities. Each transform channel consists of a Transform Engine (TE) and two Interlaken interfaces.

Each Transform Engine may be powered on or off using the "Power Enable Register" and "Module Enable Register". The power to the Encryption, Compression, Authentication, and Padding engines within the Transform Engine module cannot be independently enabled or disabled, however, the clocks to these modules may be independently enabled or disabled using the "TE Control Register".

The order of processing depends on whether the operation is in the encode or decode direction. The positions of Compression, Encryption and Pad engines are fixed, but the position of the Authentication engine is configurable per command using the Authentication engine position configuration bits AUTH_POS[1:0] in the "Session Control Word 2 - Authentication". If the TE engine pipeline is processing multiple commands from different sessions and those sessions have different Authentication engine position configurations, there will be a slight performance reduction.

The Interlaken interface may be used to expand the XR9240 processing to an external FPGA. FPGA commands may be configured to come before or after the XR9240 commands, allowing customers to refine the transform data processing to their application. Refer to the *XR9240 FPGA Design Application Note*, APN-0001, for a description of the ILK command processing sequence.

4.2.1 Class of Service Processing

Each dynamic Transform Channel (TC) queue may be configured with a unique class of service. By default, the XR9240 disables all transform channel queues to participate in strict priority class of service arbitration.

1. If the default settings do not apply to the application, the host should reconfigure the TC_COS_EN and BEST_EFFORT bits in the "Transform Channel Class of Service Register" for each queue.

2. For each command ring, the host configures the "Ring Registers". The host may assign each ring to a TC queue using the COS_Q_ID[3:0] field in the "Ring Configuration Register" if desired. The command rings will be assigned to the TC queues in a round-robin fashion.

3. Host sets the cost and peak bandwidth fields in the "Transform Channel Class of Service Register". These fields are used by the COS arbiter to determine which command in the queue will be selected as the next command to process.

4. Over time, the cost and peak bandwidth settings may be adjusted to improve performance, however, the command ring must be empty before modifying these settings.

4.3 Using Tokens to Process Transform Commands

4.3.1 General Token Rules

1. The Data token must be the last source token.

2. No other tokens may be placed between the compression toggle-on token and compression toggle-off token.

3. For AES-GCM operations, the AAD source token must be the last token before the payload data.

4. For GMAC operations, the AAD is wrapped in the data token.

5. The active region of action tokens is limited by the payload boundary. If the XR9240 sees a status token that signifies the end of the payload, all the operations will be complete.

4.3.2 Token Insertion Example

The following diagram shows a typical example of how tokens are used to identify fields within the command body.

In this GCM example, the pre-data contains IV and AAD fields before the data stream. The IV field and token must appear before the AAD token and field, and both must be before the data. Source tokens identify the IV and AAD fields so that the appropriate engines extract the IV and AAD values.

Action tokens in the data stream identify the active data region for the Encryption, Padding, and Authentication engines.

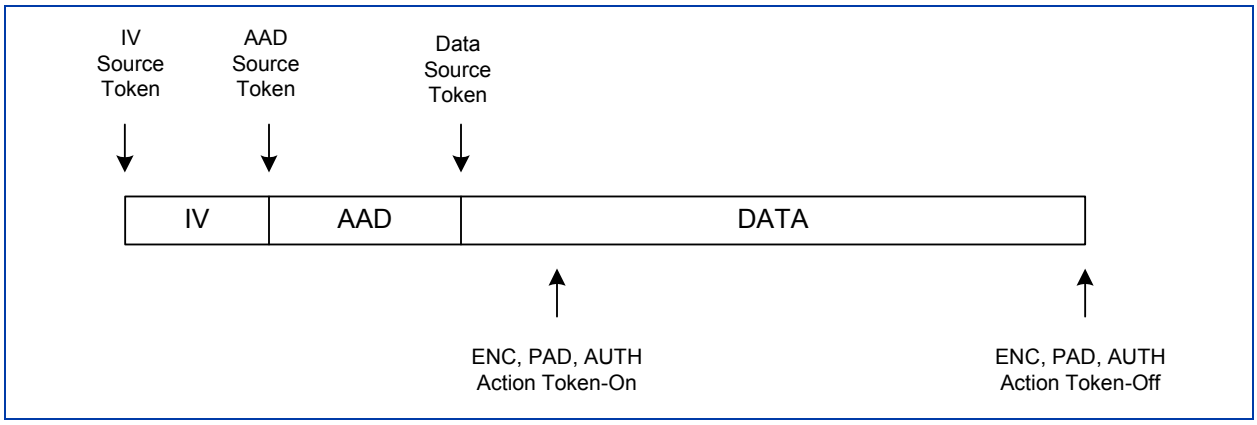


Figure 4-1. Token Insertion Example

4.4 Public Key Command Processing

Public key commands follow the "Basic Command Operation Sequence" and are executed in the PK engines.

This section gives an example of a data structure being sent to the PK module to execute a 128-bit modular exponentiation. Figure 4-2 shows an example of PK source data that would be sent to the XR9240 from the host. Note that the values in a PK descriptor are defined in words.

The first descriptor in the header, SRC_HDR_CTL, gives the total length of the PK header. In the user descriptors that follow, the address defines the access type (BER, MMR, LIR, CSR) and the length defines the length of each user descriptor (see Table 6-8).

The first four user descriptors defines the base, exponent, modulus, and the pre-compute data that will be written into the master core of the PK engine when the command executes. The fifth user descriptor gives the detailed instructions that will be programmed into the LIR. The sixth user descriptor sets the control and status field that will start the execution of the PK command.

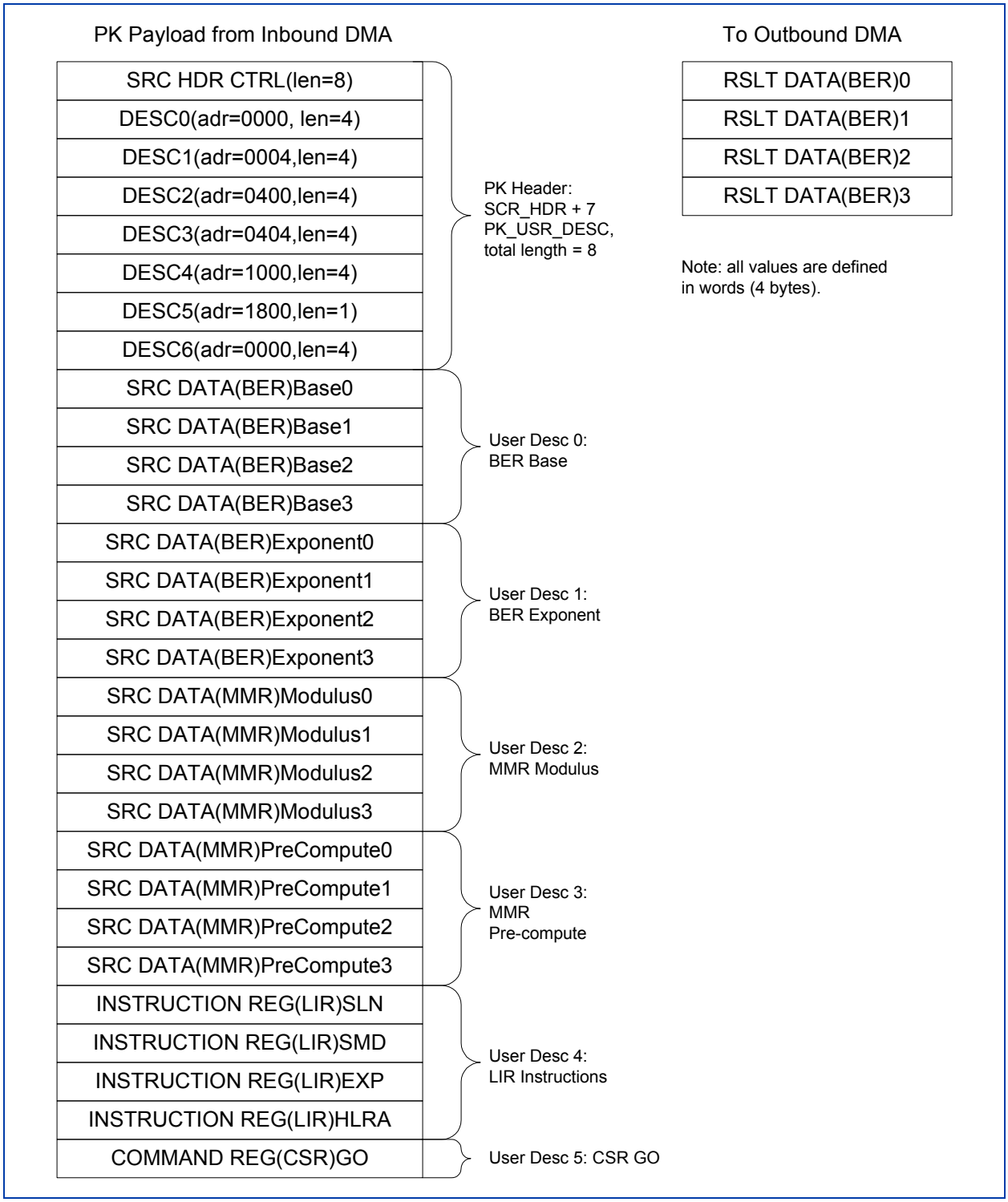


Figure 4-2. Example of a PK Command

4.5 Accessing an External Non-Volatile Device

PCIe accesses to offsets starting from the address set in the PCIe "Expansion ROM Base Address Register" will reference the external non-volatile device, provided that the expansion ROM base address register contains a valid (non-zero) base memory address.

Figure 4-3 illustrates the defined regions allocated within the non-volatile device memory.

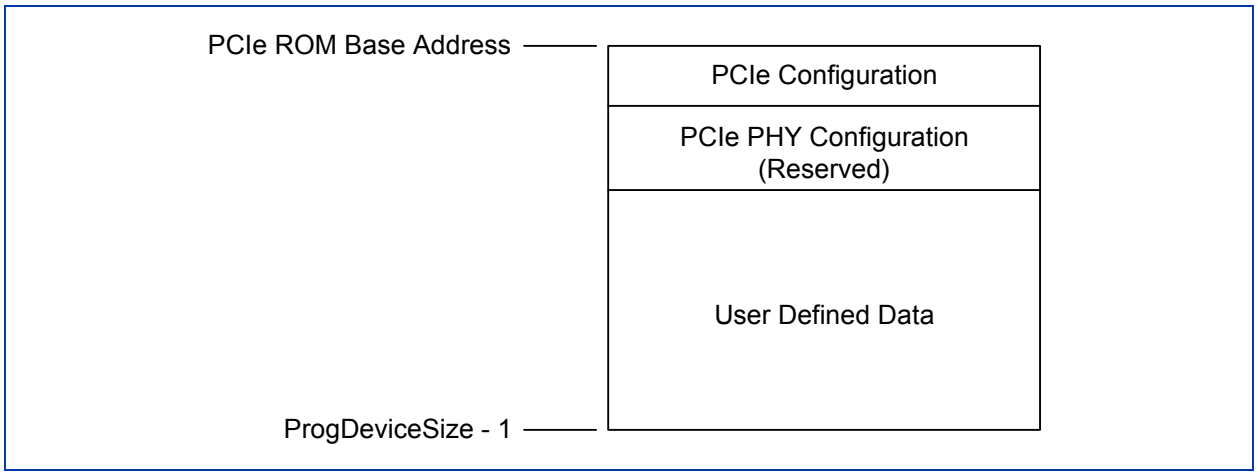
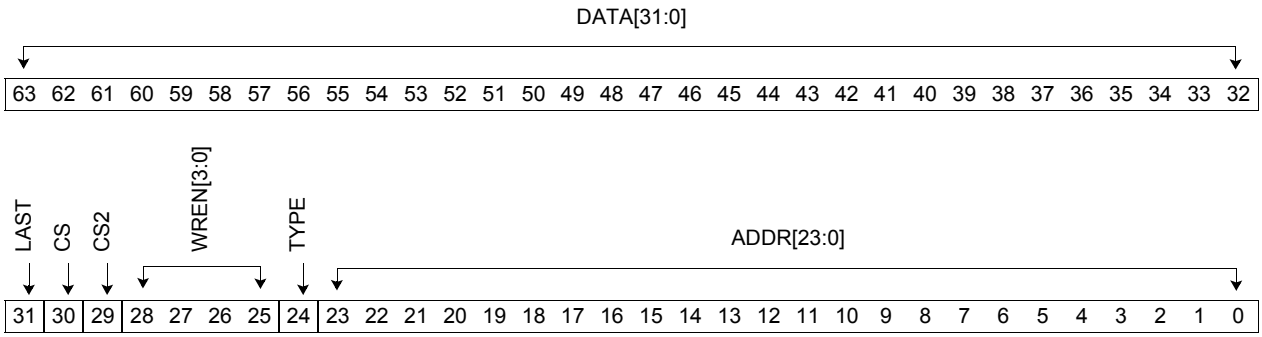


Figure 4-3. Non-volatile Device Memory Allocation

The first two regions are loaded into the XR9240 device if the XR9240 external signal LD_FLASH_EN is enabled. The first region stores PCIe configuration information that may be used to override the default PCIe configuration values initialized by the XR9240 device. The second region is reserved.

The third region is for storing user specific data. The starting and ending addresses for the software data region is managed by the host driver. The XR9240 device does not store this information in its internal registers.

The non-volatile device regions are a list of 2-word records with the formats shown in the tables below.



Field Name	Bits	Description
DATA[31:0]	63:32	Data. This field holds the data to be written into the PCIe configuration space.
LAST	31	Last Flag. 0 Indicates this is not the last record in the configuration space 1 Indicates this is the last record in the configuration space
CS	30	Chip Select. This bit is used to assert the chip select for the PCIe configuration space. 0 Disabled 1 Enabled
CS2	29	Secondary Chip Select. This bit is used to assert the chip select for the PCIe PHY configuration space. 0 Disabled 1 Enabled
WREN[3:0]	28:25	Write Enable. WREN[0] = Byte 0 WREN[1] = Byte 1 WREN[2] = Byte 2 WREN[3] = Byte 3
TYPE	24	Access Type. 0 PCIe configuration space 1 PCIe PHY configuration space (reserved)
ADDR[23:0]	23:0	Address.

If the XR9240 default PCIe configuration values are used, the external signal LD_FLASH_EN should disabled, or an empty record should be placed in the PCIe configuration space with the LAST flag set and all other fields cleared.

Refer to the *XR9240 Hardware Design Guide*, USR-0032, for the recommended non-volatile devices and for more detailed information about using a non-volatile device with the XR9240.

5 Modules

This chapter gives a more detailed description of some of the key XR9240 internal modules.

5.1 PCIe Interface

5.1.1 SR-IOV Capability

The XR9240 is designed to support Single Root I/O Virtualization (SR-IOV) in the PCIe configuration space. Multi-Root I/O Virtualization (MR-IOV) is not supported.

SR-IOV allows for a single PCIe physical function to scale to a large number of virtual functions, thus creating additional PCIe functions at a reduced hardware cost.

A physical function (PF) has all of the traditional features of a PCIe function, including a full configuration space, a completely independent set of BARs, etc. It also has a supervisory role for the virtual functions it is associated with. Virtual functions (VFs) do not have permission to access the complete set of XR9240 registers (see Figure 6-1).

The XR9240 supports a single physical function and 128 virtual functions. The physical function device ID is different from the virtual function device ID. The driver is able to distinguish between the PF and VF by interpreting the device ID.

5.1.1.1 Function Level Reset (FLR)

Before initiating a reset of a virtual function, the host must first ensure that there are no active operations for that virtual function. The virtual machine or hypervisor may reset a particular virtual function by setting the Function Level Reset (FLR) flag for that virtual function. The XR9240 is able to handle multiple virtual functions receiving a FLR at the same time.

After a FLR, the XR9240 will stop sending read requests to the VF's command pointer, command structure and source data. In addition, the output from the VF will be discarded, all commands for the reset VF will be deleted, and all the internal registers for that VF will be set to their reset value.

5.1.1.2 Alternative Requestor ID Interpretation (ARI)

Alternative Requestor ID Interpretation (ARI) allows more than eight functions per EndPoint without requesting an internal switch, as usually needed in virtualization scenarios.

Requestor ID (RID) allocation of the VF is done using the First VF Offset field in the IOV structure. This pointer is a relative offset to the Bus/Device/Function (BDF) of the PF. The First VF Offset field is added to PF's requester ID to determine the requester ID of the first VF. An additional field VF Stride in the IOV capability structure describes the distance between two consecutive VF's requester IDs.

Table 5-1 gives the Requestor ID format in ARI Mode.

Table 5-1. Requestor ID Format in ARI Mode

Function	Bus, Device, Function (Decimal)	Bus, Device, Function (Binary)
PF	B, 0, 0	B, 00000, 000
VF0 (see Notes)	B, 16, 0	B, 10000, 000
VF1	B, 16, 1	B, 10000, 001
|		
VF127	B, 31, 7	B, 11111, 111
Note: The offset from the PF to the first VF is 128 (8'b10000000).		

When ARI mode is disabled, non-zero devices in the first bus cannot be used, thus a second bus is needed to provide enough RIDs. Table 5-2 gives the Requestor ID format in non-ARI Mode. These values are hard coded into the XR9240 device.

Table 5-2. Requestor ID Format in non-ARI Mode

Function	Bus, Device, Function (Decimal)	Bus, Device, Function (Binary)
PF	B, 0, 0	B, 00000, 000
VF0 (see Notes)	B+1, 0, 0	B+1, 00000, 000
VF1	B+1, 0, 1	B+1, 00000, 001
|		
VF127	B+1, 15, 7	B+1, 01111, 111

5.1.1.3 ID Based Ordering (IDO)

ID based ordering (IDO) improves performance by avoiding stalls caused by ordering rules. Posted writes are never normally allowed to pass each other in a queue, but if they are requested by different functions, the requests are likely not dependent on each other. The Attribute bit [2] and the RO bit indicate the ID ordering with or without relaxed ordering for memory requests, and is reserved for Configuration or IO requests.

IDO may be enabled or disabled via the PCIe "Device Control 2 Register".

5.1.1.4 Virtual Function to Physical Function Communication

In order to facilitate IOV compliance, the XR9240 provides a mailbox mechanism for communication between the VF and PF drivers. The mailbox may be used by the PF driver to send status updates to the VFs (such as link change, memory parity error, etc.), or for a VF to send requests to the PF (add to VLAN).

A set of "IOV Communication Registers" are used to control the messaging to the mailbox. The physical function has access to 128 mailboxes, one mailbox for each of the 128 virtual functions. Each of the 128 virtual functions has its own mailbox, thus allowing each VF to uniquely communicate with the PF. Although the data in the mailbox is physically shared between the PF and VF, logically the PF and VFs access the mailbox data using distinct addresses.

The VF and PF drivers will receive indication that a function wrote a message to the mailbox by means of interrupt or by polling.

The procedure for a physical function (PF) to write a message to a virtual function (VF) is shown in Table 5-3 and described in more detail below the table.

Table 5-3. Procedure for the Physical Function to Write to the Virtual Function Mailbox

Step	PF Driver	XR9240	VF #N Driver
1	Send request to XR9240 to set the PF_MUTEX bit in the "Physical Function Mailbox Control Register".		
2		If the VF_MUTEX is not already set, the XR9240 will set the PF_MUTEX bit in the "Physical Function Mailbox Control Register".	
3	Read the PF_MUTEX bit in the "Physical Function Mailbox Control Register". If not set, go to step 1, otherwise continue.		
4	Write the message to the "Mailbox Data Register".		
5	Set the PF_REQ_VF bit in the "Physical Function Mailbox Control Register". Poll the VF_ACK_PF bit in the "Virtual Function Mailbox Control Register".		
6		Copy the PF_REQ_VF bit to the "Virtual Function Mailbox Control Register" and set the appropriate bit in the REQ field of the "Mailbox Interrupt Source Register".	
7			Poll the PF_REQ_VF bit in the "Virtual Function Mailbox Control Register". When set, read the message from the "Mailbox Data Register".
8			Set the VF_ACK_PF bit in the "Virtual Function Mailbox Control Register".
9		Set the appropriate bit in the ACK field of the "Mailbox Interrupt Source Register" to interrupt the PF.	
10	Clear the PF_MUTEX bit in the "Physical Function Mailbox Control Register".		

1. The PF driver requests access to the mailbox by attempting to set the PF_MUTEX bit in the "Physical Function Mailbox Control Register". If the VF_MUTEX bit is set to one, the buffer is in use by the VF.

2. If the VF_MUTEX bit is zero, the mailbox is free, and the XR9240 responds by setting the PF_MUTEX bit in the "Physical Function Mailbox Control Register".

3. The PF driver polls the "Physical Function Mailbox Control Register" to determine if the PF_MUTEX bit is set. If not set, go to step 1, otherwise, continue.

4. The PF driver writes the message to the "Mailbox Data Register" using the appropriate address for the VF that is attempting to communicate with (each VF contains its own mailbox data register).

5. The PF driver sets the PF_REQ_VF bit in the "Physical Function Mailbox Control Register" to signify that the PF message was written to the VF mailbox.

   The XR9240 will automatically copy the PF_REQ_VF bit to the "Virtual Function Mailbox Control Register" and set the appropriate bit in the REQ field of the "Mailbox Interrupt Source Register" to inform the VF that a message is waiting in its mailbox.

6. The VF driver reads the message from the "Mailbox Data Register" and then sets the VF_ACK_PF bit in the "Virtual Function Mailbox Control Register".

   The XR9240 will automatically set the appropriate bit in the ACK field of the "Mailbox Interrupt Source Register" to inform the PF that the VF has read its message.

7. The PF driver waits for the VF acknowledge by polling the VF_ACK_PF bit in the "Virtual Function Mailbox Control Register".

8. The message transaction is complete, so the PF driver clears the PF_MUTEX bit in the "Physical Function Mailbox Control Register" to free the mailbox for another use.

The procedure for a virtual function (VF) to write a message to a physical function (PF) is shown in Table 5-4 and described in more detail below the table.

Table 5-4. Procedure for the Virtual Function to Write to the Physical Function Mailbox

Step	PF Driver	XR9240	VF #N Driver
1			Send request to XR9240 to set the VF_MUTEX bit in the "Virtual Function Mailbox Control Register".
2		If the PF_MUTEX is not set, the XR9240 will set the VF_MUTEX bit in the "Virtual Function Mailbox Control Register".	
3			Read the VF_MUTEX bit in the "Virtual Function Mailbox Control Register". If not set, go to step 1, otherwise continue.

Table 5-4. Procedure for the Virtual Function to Write to the Physical Function Mailbox

Step	PF Driver	XR9240	VF #N Driver
4			Write message to the "Mailbox Data Register".
5			Set the VF_REQ_PF bit in the "Virtual Function Mailbox Control Register". Poll the PF_ACK_VF bit in the "Virtual Function Mailbox Control Register".
6		Copy the VF_REQ_PF bit to the "Physical Function Mailbox Control Register" and set the appropriate bit in the REQ field of the "Mailbox Interrupt Source Register".	
7	Poll the VF_REQ_PF bit in the "Physical Function Mailbox Control Register". When set, read the message from the "Mailbox Data Register". Set the PF_ACK_VF bit in the "Physical Function Mailbox Control Register".		
8		Copy the PF_ACK_VF bit to the "Virtual Function Mailbox Control Register". Set the appropriate bit in the ACK field of the "Mailbox Interrupt Source Register" to interrupt the VF.	
9			Poll the PF_ACK_VF bit in the "Virtual Function Mailbox Control Register".
10			Clear the VF_MUTEX bit in the "Virtual Function Mailbox Control Register".

1. The VF driver requests access to the mailbox by attempting to set the VF_MUTEX bit in the "Virtual Function Mailbox Control Register". If the PF_MUTEX bit is set to one, the buffer is in use by the PF.

2. If the PF_MUTEX bit is zero, the mailbox is free, and the XR9240 responds by setting the VF_MUTEX bit in the "Virtual Function Mailbox Control Register".

3. The VF driver polls the "Virtual Function Mailbox Control Register" to determine if the VF_MUTEX bit is set. If not set, go to step 1, otherwise, continue.

4. The VF driver writes the message to the "Mailbox Data Register" using the appropriate address (each VF contains its own mailbox data register).

5. The VF driver sets the VF_REQ_PF bit in the "Virtual Function Mailbox Control Register" to signify that the VF message was written to the mailbox.

6. The XR9240 will automatically copy the VF_REQ_PF bit to the "Physical Function Mailbox Control Register" and set the appropriate bit in the REQ field of the "Mailbox Interrupt Source Register" to inform the PF that a message is waiting in its mailbox.

7. The PF driver reads the message from the "Mailbox Data Register" and then sets the PF_ACK_VF bit in the "Physical Function Mailbox Control Register".

8. The XR9240 will automatically copy the PF_ACK_VF bit to the "Virtual Function Mailbox Control Register" and set the appropriate bit in the ACK field of the "Mailbox Interrupt Source Register" to inform the VF that the PF has read its message.

9. The VF driver waits for the PF acknowledge by polling the PF_ACK_VF bit in the "Virtual Function Mailbox Control Register".

10. The message transaction is complete, so the VF driver clears the VF_MUTEX bit in the "Virtual Function Mailbox Control Register" to free the mailbox for another use.

5.1.2 Power Management

The XR9240 supports the D0 and D3 power states defined in the PCI Power Management and PCIe specifications.

5.1.2.1 D0 State

D0 is divided into two distinct sub-states as required by the PCIe base specification 3.0: the "uninitialized" sub-state, and the "active" sub-state. When the XR9240 device comes out of reset or FLR, it defaults to the D0 uninitialized state.

After the host enables the memory space, the XR9240 enters the active sub-state where it can transmit and receive PCIe packets if properly configured by the software device driver.

Entry into D0 state

The D0 state is reached from either the D3cold state or the D3hot state (on de-assertion of PERST_N) or the D3hot state (by configuration software writing a value of 00b to the Power State field of the PCI PM registers).

When the XR9240 is in the D0 un-initialized state, it will be enumerated and configured by the hierarchy enumeration process. Following the completion of the enumeration and configuration process the XR9240 enters the D0 active state, the fully operational state for a PCI Express function. A function enters the D0active state whenever any single or combination of the function's Memory Space Enable, I/O Space Enable, or Bus Master Enable bits have been enabled by system software.

Setting PERST_N low will reset the entire XR9240, including the sticky bits in the PCIe configuration header.

If the No_Soft_Reset field in the Power Management Control/Status Register (PMCSR) is set, the XR9240 PCI configuration space will not be reset on a transition from the D3hot state to the D0 state.

5.1.2.2 D3hot State

The XR9240 transitions to the D3hot state when the host writes 11b to the Power State field of the Power Management Control/Status Register (PMCSR).

Configuration and message requests are the only TLPs accepted by a function in the D3hot state. All other received requests must be handled as unsupported requests, and all received completions can optionally be handled as unexpected completions. If an error caused by a received TLP (such as an unsupported request) is detected while in D3hot, and reporting is enabled, the link must be returned to L0 if it is not already in L0 and an error message must be sent.

Entry into D3hot State

Transition to D3hot state is through a configuration write to the Power State field of the PCIe "Power Management Control/Status Register".

Prior to transition from D0 to the D3hot state, the software device driver disables scheduling further tasks to the XR9240 and masks all interrupts.

In this state, no wake up capability is supported.

As a response to being programmed into the D3hot state, the XR9240 transitions its PCIe link state into the L1 link state. As part of the transition into L1 state, the XR9240 suspends scheduling of new TLPs and waits for the completion of all previous TLPs it has sent.

Exit from D3hot State

A D3hot state is followed by either a D0 state or by a transition to D3cold state. To transition back to D0, the system writes a 00b to the Power State field of the Power Management Control/Status Register (PMCSR). Transition to D3cold state is through a pme_turn_off message.

5.1.2.3 D3cold

Transition to the D3cold state is initiated by a PME_Turn_Off/Pme_To_Ack handshake completion at any of the D0, D1, D2, D3 states.

To exit from the D3cold state, the host should assert the XR9240 reset after the power is recovered as the XR9240 does not support a wake-up beacon. A full initialization of each function is required to establish all function context. The function will then transition to the D0active state.

5.1.2.4 PCIe Link Power Management

The PCIe link state follows the power management state of the XR9240. Since the XR9240 incorporates multiple PCI functions, its power management state is defined as the power management state of the most awake function. The XR9240 follows the link power management states shown in Figure 5-1.

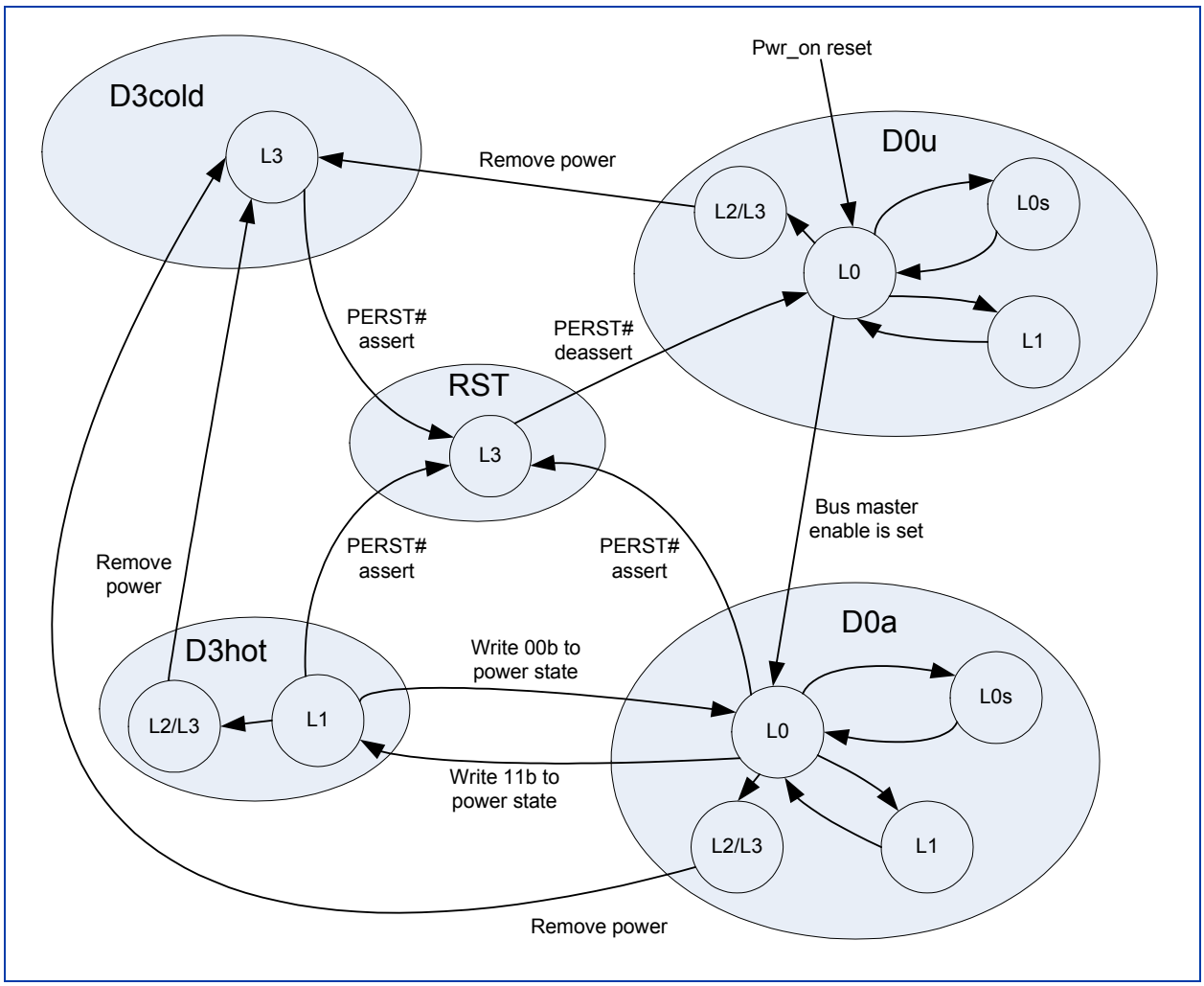


Figure 5-1. PCIe Link Power Management States

5.2 PCIe DMA

5.2.1 Endian Data Format

Internally, the XR9240 operates on data using little endian format. The XR9240 automatically converts the data endianness from/to host, both in terms of swapping bytes within a 32-bit word and swapping 32-bit words within a 64-bit word.

The HOST_EF field in the "XR9240 Control Register" specifies the endian format of the host command pointer ring, command structure, result pointer ring and free buffer pointer ring. The BYTE_ORDER[1:0] field in the "Command Descriptor" specifies the endian format of the source and destination data.

The mapping of byte order to bit fields and byte-selects is shown in Table 5-5 and in Figure 3-5.

Table 5-5. Endian Byte Mapping

Byte Address	Field	32-bit Equivalent
0	[7:0]	LSB, 1st word
1	[15:8]	
2	[23:16]	
3	[31:24]	MSB, 1st word
4	[39:32]	LSB, 2nd word
5	[47:40]	
6	[55:48]	
7	[63:56]	MSB, 2nd word

5.2.2 Messages

The PCIe interface supports legacy interrupt messages, power signaling messages and error signaling messages.

Legacy interrupt messages include Assert_INTA and Deassert_INTA.

Power signaling messages include PME_TO_Ack, PM_Active_State_Nak, PME_Turn_Off, Set_Slot_Power_Limit and error messages. The PME_TO_Ack signal is generated by the PMC after the PME_TURN_OFF is acknowledged by the application. When a PM_Active_State_Nak message is received, the L1 entry request is denied. The PME_Turn_Off signal indicates that the application is ready as soon as this message is received. The Set_Slot_Power_Limit message copies the contents of the Slot Power Limit Value and Scale fields in the Slot Capabilities register of the downstream port to the corresponding Captured Slot Power Limit Value and Scale fields in the "Device Capabilities Register".

The following PCIe standard error messages are generated automatically by the PCIe Interface when PCIe errors are detected: COR_ERR, ERR_NONFATAL, and ERR_FATAL.

5.2.3 INTx and MSI Interrupts

The XR9240 supports the following PCIe interrupt modes.

- PCI legacy interrupts and Message Signaled Interrupts (MSI) or MSI-X with only a single vector allocated
- MSI-X with multiple MSI-X vectors in non-IOV mode

- MSI-X in IOV mode

The mapping of the interrupts depends on the interrupt mode. The following sections describe the interrupt registers and device functionality at all operation modes.

The "PCIe Interrupt Source Register" details the defined XR9240 interrupts.

5.2.3.1 PCI Legacy and MSI Interrupt Modes

In PCI legacy and MSI modes, the source of the interrupt is reflected in the "PCIe Interrupt Source Register", where each bit reflects the interrupt source.

5.2.3.2 MSI-X Mode in Non-IOV Mode

MSI-X expands the basic MSI functionality. The XR9240 supports 17 MSI-X vectors and therefore sets the Table Size field of the MSI-X capability structure in the "MSI-X Capability Registers" to 17.

The interrupt causes in the "PCIe Interrupt Source Register" may be statically mapped to the MSI-X vectors, as in bit 0 mapped to MSI-X vector 0, etc. When the MSI-X vectors are used in this way, the host avoids having to read the Interrupt Source register. MSI-X mode also allows load balancing of the MSI-X vectors to different CPUs.

Each MSI-X vector may also be associated to a bit in the "PCIe Interrupt Throttle Register" using the index number (MSI-X vector 0 to bit[0], MSI-X vector 1 to bit[1],...) to optimized interrupt moderation schemes per MSI-X vector. In this mode, 16 vectors are assigned to the Command Done interrupt, leaving one vector assigned for all other interrupts. If more than 16 queues are activated, some queues will have to share a MSI-X vector.

5.2.3.3 MSI-X Interrupts In IOV Mode

In IOV mode, interrupts must be implemented as MSI-X vectors.

The physical function may have up a maximum of 17 MSI-X vectors The first 16 PF rings are statically assigned to MSI-X vectors. The host must use polling to read the status of the remaining PF rings.

Each of the 128 virtual functions VF(0...127) supported by the XR9240 can generate up to two MSI-X vectors. For VF rings, the command done interrupts may only be mapped to MSI-X vector 0.

The number of requested MSI-X vectors also is reflected in the Table Size field in the PCIe MSI-X capability structure for each VF and PF.

5.2.3.4 Interrupt Moderation

Interrupt rates can be tuned to reduce CPU utilization and minimize latency by limiting the interrupt delivery rate to the host CPU. Using the XR9240 "PCIe Interrupt Throttle Register", a guaranteed delay between interrupts asserted by the XR9240 can be achieved. In MSI or legacy interrupt modes, only a single Interrupt Throttle register is supported. However, in MSI-X and non-IOV modes, the XR9240 supports 16 Interrupt Throttle

registers that are mapped to MSI-X vectors 0...15, respectively. In IOV mode, there are an additional 128 Interrupt Throttle registers that are mapped to the MSI-X vectors of the virtual functions.

The following formula converts the interrupt interval value to the common "interrupts/ second" performance metric:

$$\text{Interrupts/microsecond} = (2 * 10^{-6} \text{ seconds} * \text{Interval}) - 1$$

For example, if the interval is programmed to 125d, the network controller guarantees the host will not be interrupted by the network controller for at least 250 microseconds since the last interrupt. In this case, the maximum observable interrupt rate from the adapter should not exceed 4000 interrupts/sec.

Inversely, interrupt interval value can be calculated as:

$$\text{Interrupt interval} = (2 * 10^{-6} \text{ sec} * \text{interrupt/sec}) - 1$$

The optimal performance setting for the throttle value is system and configuration specific.

Whenever an interrupt occurs, the corresponding bit in the "PCIe Interrupt Source Register" is set. However, an interrupt message is not sent out on the PCIe interface until the Interrupt Throttle counter assigned to that Interrupt Source bit has counted down to zero. As soon as the interrupt is issued, the Interrupt Throttle counter is reloaded with its initial value and the process repeats.

If the Interrupt Throttle counter counts down to zero and no interrupt events have occurred, then the Interrupt Throttle counter value will remain at zero so that the next interrupt event triggers an interrupt immediately.

Some low latency events such as a command completion error or the free buffer being used require the host to respond immediately. For these or other interrupt causes, the host may disable the corresponding bit in the Interrupt Throttle register.

5.2.4 TLP Processing Hint (TPH)

The XR9240 supports Transaction Layer Packet (TLP) Processing Hints (TPHs) at the data structure level. TPH may be enabled or disabled for the following data structure types: command ring, command structure, free buffer pointer ring, result pointer ring, destination data, and MSIX interrupt vector writes.

A memory read or write request header contains three fields that apply to TLP:

- A 2-bit Processing Hints (PH) field
- An 8-bit Steering Tag (ST) that may be used to steer the TLP to a particular CPU cache
- A TLP processing hint enable bit that indicates the PH and ST fields are valid

These fields are configured using the "TLP Processing Hint (TPH) Registers".

5.2.4.1 General assumptions

The following general assumptions can be made regarding TPH:

1. The XR9240 will typically use the Target type option for the 2-bit PH field, which indicates either that the XR9240 wrote data that the host is expected to read soon, or that the XR9240 read data that the host is believed to have written recently.

2. The value of the 8-bit Steering Tag for each type is set by the host. The XR9240 uses Device Specific Mode, and as such there is no ST table, nor are the ST values stored in the MSI-X table.

3. The optional TLP TPH prefixes are not needed.

4. TPH will be most beneficial for writes of control structures to the host, specifically to the command pointer ring and result pointer ring.

5.2.4.2 Configuring the XR9240 for TPH

Before using TPH, the host must configure the XR9240 as described in the steps below.

1. Set the TPH capability in the PCIe "TPH Requester Capability Register" and the Steering Tag table size in the "TPH Requester Capability Register".

2. In the PCIe TPH Requester Control Register, set the per function TPH enable and ST mode select values.

3. In the XR9240 "TLP Processing Hint (TPH) Registers", enable TPH for each data structure type.

4. In the XR9240 "TLP Processing Hint (TPH) Registers", set the 2-bit Processing Hints (PH) field per data structure type

5. In the XR9240 "TLP Processing Hint (TPH) Registers", set the 8-bit Steering Tag (ST) field per data structure type.

5.3 Command Manager

The PCIe DMA and Command Manager work together to provide Class of Service (COS) arbitration that manages how the commands from the 128 command rings are channeled to the command manager queues and from the queues to the data processing engines. Figure 5-2 illustrates the modules within the XR9240 DMA that interact with the Command Manager module. The subsequent sections discuss each function in more detail.

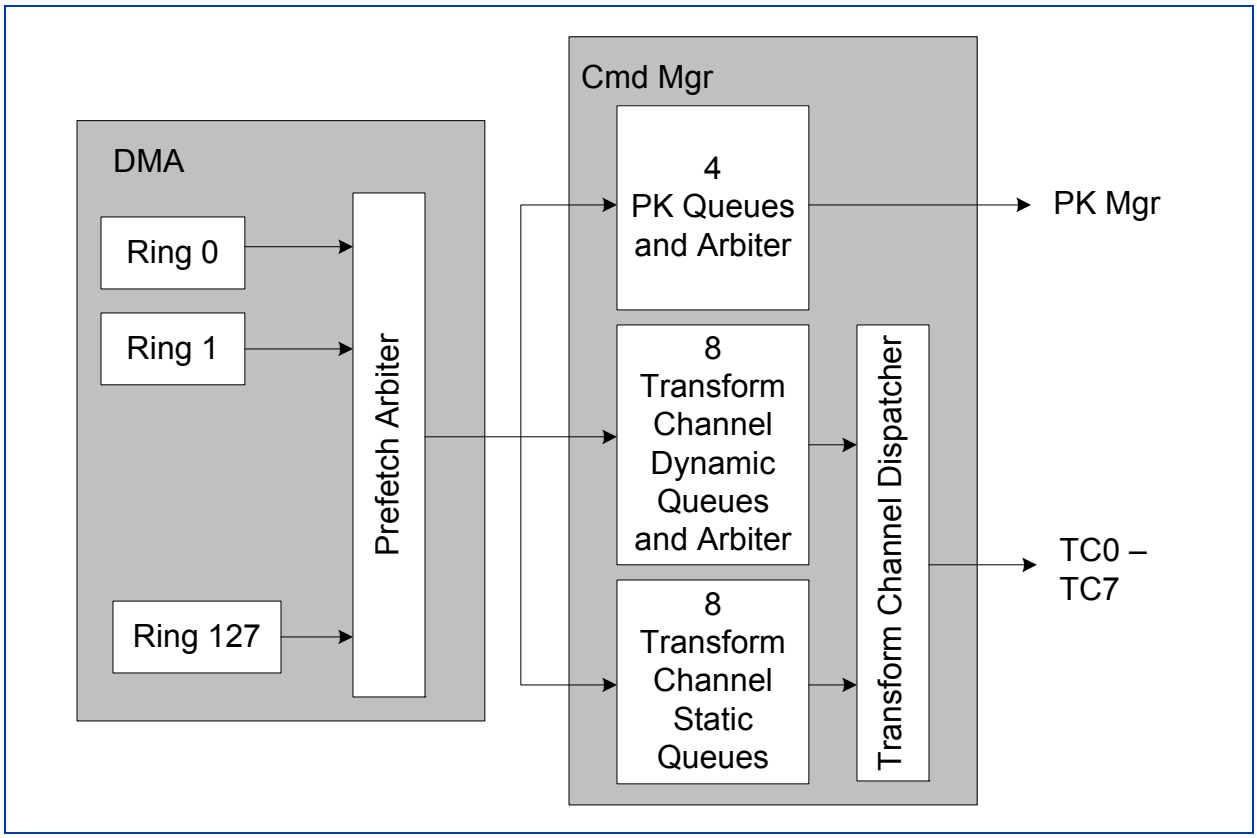


Figure 5-2. Class Of Service Arbitration Logic

COS arbitration is disabled by default. During configuration, the host will configure the XR9240 queue's priority and arbitration scheme using the "Transform Channel Class of Service Register" and the "Public Key Class of Service Register". During runtime, the COS property of a command ring must only be changed when the ring is empty.

The host may disable COS arbitration. When COS is disabled, the commands from all Public Key queues or Transform Channel dynamic queues are dispatched in round-robin order.

5.3.1 Prefetch Arbiter

The prefetch arbitration is a simple round-robin among the queues; there is no priority at this level. The Prefetch Arbiter attempts to ensure that all active queues are filled with commands. Because many rings may be mapped to a single queue, there is also simple round-robin arbitration among multiple rings that are mapped to a single queue. Both levels of arbitration (among the different queues and among rings for a single queue) are simple round-robin at this stage. The host must set a fixed ring to queue assignment using the COS_Q_ID[2:0] field in the "Ring Configuration Register"). A ring may not be assigned to multiple COS queues simultaneously.

By default, each TC dynamic queue is assigned a prefetch buffer with 32 entries. However, the buffer depth is configurable using the PREF_NUM[4:0] field in the "Transform Channel Class of Service Register" or "Public Key Class of Service Register" if a COS queue will be used for commands with smaller bandwidth. The number of commands to allocate in the prefetch buffer can be calculated as:

PN = 32*(COS queue bandwidth/total bandwidth)

For example, if a queue is configured for 1/8 of the total bandwidth, the prefetch buffer depth could be set to 32/8 ≈ 4 commands.

5.3.2 Transform Channel Dynamic Queues

Each transform channel command ring can be assigned to one of the eight dynamic transform channel COS queues. Transform channel dynamic queues may be configured as strict priority queues or best effort queues. Strict priority queues have higher priority than best effort priority queues.

5.3.2.1 Strict Priority Arbitration

Strict arbitration applies only to the queues that are configured as strict priority queues. The strict arbiter has the highest priority among the COS arbiters.

Strict queues are assigned a peak bandwidth limit that is measured in units of bits/sec. If the limit is set to the maximum value, the user must control the number of commands that are sent to that strict queue to prevent that queue from monopolizing the command processing. The peak bandwidth limit value, PK_BS[7:0] in the "Transform Channel Class of Service Register", must be a power of 2. The maximum allowed bandwidth is 64 Gbps; the minimum is 1 Mbps.

The priority of the strict priority queues is the same as the queue ID. Queue ID 7 has the highest priority, while queue ID 0 has the lowest. If more than one strict queue has waiting commands, the higher priority queue will gets serviced exclusively until its bandwidth limit is reached or the queue is empty. The queue ID is configured by the host via the COS_Q_ID[3:0] in the "Ring Configuration Register".

If all strict priority queues have reached their bandwidth limit, the strict arbiter will not dispatch any commands until the bandwidths have expired. Each of the (up to 8) strict priority queues will set a timer for each dispatched command, and no commands will be dispatched from that strict priority queue until the timer has expired. For instance, if a command with a 1KB length is dispatched, and the "tick" timer for the strict priority bandwidth limiter is 250 Mhz and the bandwidth limit is set to 512 Mb/s from this queue, then the counter would be loaded with:

((1KB * 8 b/Byte) / 512 Mb/s Limit) * 250Mhz Clk Freq = 4000 clock periods before the next command is eligible to be dispatched from this queue

Note that the arbitration only occurs when there is available bandwidth and the datapath is ready. Strict arbitration occurs on the command boundary.

5.3.2.2 Best Effort Arbitration

The best effort COS arbiter is logically second in line after the strict COS arbiter. If the strict arbiter does not dispatch a command, the best effort arbiter will dispatch a command from one of the best effort COS queues.

Best effort arbitration uses a Weighted Deficit Round Robin (WDRR) algorithm that distributes the bandwidth according to a cost value (see "Transform Channel Class of Service Register"). A queue that is configured for best effort arbitration must also be configured with a bandwidth cost. The best effort arbitration selects among the queues that have commands available when the TC datapath is ready to process the next command. If a queue is empty, its bandwidth is cleared.

The equation below gives the bandwidth percentage for each queue, based on the cost values assigned to all the queues:

$$\text{BW\% for Q0} = (1/\text{Cost}_{Q0}) / \Sigma\ (1/\text{Cost}_{Q0} + 1/\text{Cost}_{Q1} + \ldots\ 1/\text{Cost}_{Q7})$$

The cost values are set by the host using the COST[7:0] field in the "Transform Channel Class of Service Register".

The DWRR algorithm used by the XR9240 automatically allocates commands to a datapath whenever there is a datapath available. If some queues have no traffic, their bandwidth is automatically assumed by queues with traffic. This assures maximum utilization of the XR9240 resources as long as there are commands available to execute.

5.3.2.3 Command Length Considerations

The length used in arbitration is taken from the SRC_LEN[47:0] field in the command descriptor. The value includes the entire command body, including the payload and the pre-data. To prevent overflow within the XR9240, it is recommended that large files be split into several smaller stateful commands (<= 32MB). This guideline also ensures optimum COS performance in an environment with a mix of small and large commands.

5.3.3 Public Key Queues

Each PK ring could be assigned to a PK queue. Each PK queue has 2 entries. Public Key queues may be assigned as strict or best effort arbitration. The discussion of these arbitration schemes in "Transform Channel Dynamic Queues" applies to the PK queues as well. The bandwidth limit of a PK queue is configured in units of operations/second (op/sec). The minimum value is 256 op/sec; the maximum value is 256*256 op/sec.

5.3.4 Transform Channel Static Queues

There are eight transform channel static queues. Each queue has 4 entries. A queue entry includes the command tag, the ring ID and the length (in 4 bytes) of the predata and payload for the command.

Transform channel static queues accept commands from rings in round-robin order. Peak bandwidth and cost are not configured for static queues.

Static queues are mapped directly to their corresponding transform channels. If a transform channel has load balancing disabled and is available, it will request a command from the corresponding static queue. Load balancing for the transform channels may be enabled or disabled via the "Transform Channel Load Balance Enable Register".

5.4 Transform Engine

The Transform Engine module is a high performance algorithm acceleration engine that supports the flexible algorithm combinations listed in Table 5-6.

Table 5-6. Supported TE Algorithm Combinations

Operation	Allowed Combinations	Notes
comp	LZS / eLZS	Compression history may be input only for stateless operations.
	gzip / zlib / deflate	Compression history may be input for stateful and stateless operations. Support for z_full_flush and z_sync_flush.
enc	3DES-CBC	IV supported for stateful operations.
	AES-CBC-128, AES-CBC-192, AES-CBC-256 AES-GCM-128, AES-GCM-192, AES-GCM-256 AES-CTR-128, AES-CTR-192, AES-CTR-256 AES-ECB-128, AES-ECB-192, AES-ECB-256 AES-F8-128, AES-F8-192, AES-F8-256 AES-XTS-256, AES-XTS-512	IV supported for stateful operations.
	RC4	SBOX supported for stateful operations.
auth	MD5	IHV supported for stateful / LB operations.
	SHA1, SHA224, SHA256, SHA384, SHA512	
	HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-SHA384, HMAC-SHA512, HMAC-MD5, P_hash	
	GMAC	
	CMAC	
	XCBC-MAC	
	SSLv3-MAC	
comp/enc/auth	comp + (padding) + enc + auth	Note: authentication operation order is flexible
	comp + auth + (padding) + enc	
	comp + auth	
	comp + (padding) + enc	
	enc + auth	

Table 5-6. Supported TE Algorithm Combinations

Operation	Allowed Combinations	Notes
dec/dcomp/auth	auth + dec + (depadding) + dcomp	Note: authentication operation order is flexible
	dec + (depadding) + auth + dcomp	
	auth + dec + (depadding)	
	auth + dcomp	
wireless	SNOW 3G-EEA1/UEA2	
	AES128-EEA2	
	ZUC-EEA3	
	KASUMI-UEA1	
	EIA1, EIA2, EIA3, UIA1, UIA2	IHV supported for stateful operations.

As shown in Table 5-7, most storage and network protocols complete in a single pass through the TE and mixed commands are allowed. In comparison to a full pass, a mini pass through the Transform engine does not touch all the data and is used to obtain information for the full pass processing. A mini pass could be executed in hardware or software.

Table 5-7. Supported Network Protocol

Protocol	Completion Throughput
AH	1 full pass
ESP outbound	1 full pass
ESP inbound	1 full pass if IPCOMP is not enabled, 1 mini pass +1 full pass if IPCOMP is enabled (See Note 1)
AH-ESP outbound	1 full pass for single MAC, 2 full passes for dual MAC
AH-ESP inbound	1 full pass + 1 mini pass for single MAC, 2 full passes + 1 mini pass for dual MAC
UDP-ESP outbound	1 full pass
UDP-ESP inbound	1 full pass if IPCOMP is not enabled 1 mini pass + 1 full pass if IPCOMP enabled (See Note 1)
AH-UDP-ESP outbound	1 full pass for single MAC, 2 full passes for dual MAC
AH-UDP-ESP inbound	1 full pass + 1 mini pass for single MAC, 2 full passes + 1 mini pass for dual MAC
SRTP	1 full pass
SRTCP	1 full pass

Table 5-7. Supported Network Protocol

Protocol	Completion Throughput
SSL outbound	1 full pass without compression enabled, 2 full passes with compression enabled (See Note 2)
SSL inbound	1 full pass when using RC4 and GCM 1 mini pass + 1 full pass when using CBC encryption with no compression (see Note 3) 2 full passes when compression is enabled (see Note 4)
Notes: 1. If IPCOMP is enabled, the nxt_hdr in the trailer should be checked before the next full pass. 2. Length must be updated for a compression operation that comes before an authentication operation. 3. Mini-pass retrieves the pad length. TCipher.length = TLS compressed.length + MAC length + pad length. TLSCompressed.length can then be deduced for MAC calculation in full pass. 4. First pass performs decryption. Second pass performs authentication and decompression.	

5.4.1 Compression/Decompression Engine

The XR9240 Compression engine dynamically supports any combination of the following compression/decompression algorithms: LZS, eLZS, gzip, zlib and Deflate.

In order to obtain a larger compression ratio, the XR9240 implements a 1-byte "lazy match" feature, for LZS and gzip algorithms. This feature is enabled by default; to disable this feature, set the LAZY_DIS bit in the "gzip Control Register".

5.4.1.1 LZS/eLZS

The XR9240 LZS engine performs stateless compression/decompression using the industry-standard LZS algorithm as well as the enhanced LZS (eLZS) algorithm with anti-expansion. One compression command produces a single LZS/eLZS record. The LZS engine cannot process two commands simultaneously, but the output buffer can store more than one command output.

5.4.1.2 gzip/zlib/Deflate

The XR9240 gzip engine has a 4 KB compression history window and the decompression engine has a 32 KB history window. The types of gzip compression supported are auto-dynamic mode (stored, static Huffman and dynamic Huffman). For stateful gzip compression, the gzip compression engine supports zlib z_full_flush and z_sync_flush modes by padding a zero length "uncompressed" deflate block so that every compressed deflate block ends on a byte boundary.

The XR9240 gzip/zlib/deflate engine supports both stateless and stateful compression and decompression operations. The stateful feature allows larger files to be split into smaller fragments that can then be sent as separate commands.

Stateful gzip/zlib/Deflate Compression

For the first fragment of a stateful gzip/zlib/Deflate compression operation, the compression history and compression state are optional. If included, the XR9240 will use the history as a dictionary. The XR9240 will add padding bits so that the output is aligned to a byte boundary. The intermediate CRC/ADLER32 result and length will also be output to the sideband buffer. For debugging purposes, the number of bits in the last byte before adding the stateful compression padding may be output to the sideband buffer.

For the middle fragments of a stateful gzip/zlib/Deflate compression operation, the compression history, the previous CRC/ADLER32 value, and the length are required. The number of bits in the last byte, the intermediate CRC/ADLER32 result and the intermediate length (including the length of the previous fragments) will be output to the sideband buffer.

For the last fragment, the XR9240 will output only the file length and the CRC/ADLER32 to the sideband buffer.

Real time verification may be enabled for stateful gzip/zlib/Deflate compression operations.

For stateless GZIP/zlib operations, the XR9240 will add a header and a trailer to form the correct GZIP/zlib format. For stateful GZIP/zlib operations, the XR9240 will not add a header or trailer, therefore the software must add both.For stateful GZIP/zlib operations, the software must append the CRC and file length to the output in the destination buffer. For zlib operations, the software must append the ADLER32 to the output in the destination buffer.

Stateful gzip/zlib/Deflate Decompression

A stateful gzip/zlib/Deflate decompression operation is best described with an example. Suppose a large deflate file contains 4 deflate blocks as illustrated in Figure 5-3. The position of the block boundaries are not known to the XR9240. In this example, fragment0 ends in the middle of deflate_blk1.

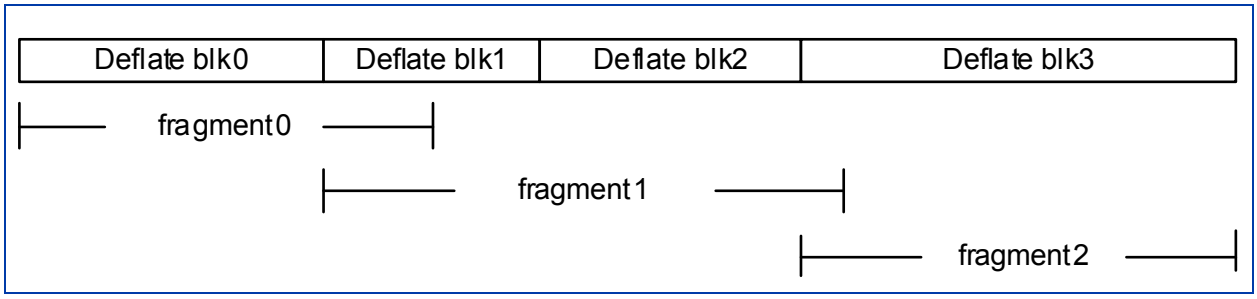


Figure 5-3. Stateful gzip/zlib/Deflate Decompression Example

Only after the fragment0 result is returned will the XR9240 know the starting position of fragment1. The entire file is decompressed using 3 commands, where each command corresponds to one fragment. The XR9240 guarantees that all fragments start exactly at the block boundary using the method described below.

For the first fragment of a stateful gzip/zlib/Deflate decompression operation, the XR9240 recognizes that since the deflate_blk1 has no EOB, the last block of fragement0 is incomplete. The number of bits after the EOB in the deflate_blk0 block is defined as the source buffer last block position, src_last_blk_pos. Using this information, the software is

able to align fragment1 to the start of deflate_blk1. The XR9240 may output some invalid bytes from deflate_blk1 before getting to the end of fragment0. The number of invalid bytes from fragment0 that need to be discarded is defined as dst_last_blk_pos. The intermediate CRC/ADLER32 result is then calculated on the remaining valid output from deflate_blk0. The XR9240 will output the src_last_blk_pos, dst_last_blk_pos and intermediate CRC/adler32 result to the sideband buffer.

For the middle and last fragments of a stateful gzip/zlib/Deflate decompression operation, the compression history, the previous CRC/ADLER32 and start_bit are required. The start_bit is taken from src_last_blk_pos of the previous fragment. The src_last_blk_pos and dst_last_blk_pos are relative to the start of the fragment. The XR9240 outputs the src_last_blk_pos, dst_last_blk_pos and intermediate CRC/adler32 result to the sideband buffer.

If a fragment length is less than the length of the deflate block, all of the output will be discarded. In this case, the software would need to increase the fragment size.

For gzip/zlib operations, the file header in the first fragment and the trailer from the last fragment need to be stripped by software before the source data is sent to the XR9240.

5.4.2 Authentication Engine

The XR9240 Authentication engine can perform either single hash or hash/MAC authentication operations. The following stateful hash/MAC operations are supported:

- MD5, SHA1, SHA224, SHA256, SHA384, SHA512
- HMAC-MD5, HMAC-SHA1, HMAC-SHA224, HMAC-SHA256, HMAC-384, HMAC-512, SSL3.0-MAC
- GMAC
- X-CBC MAC, CMAC
- P_hash
- UIA1, UIA2, EIA1, EIA2, EIA3 (described in Section 5.4.3.5)

The Authentication engine contains dual engines (Auth Eng 0, and Auth Eng 1) that allow the XR9240 to perform two separate hash transforms in one pass on the input data, or perform a hash transform with real time verification in one pass. For both hash and MAC operations, the position of Authentication engine in the pipeline is programmable.

5.4.2.1 MD5, SHA1, SHA224, SHA256, SHA384, SHA512 Operation

Authentication calculations are block based. If the source data is not a multiple of the block size, the XR9240 Authentication engine will automatically add padding bits to the last block. For a stateful hash operation, the length of the first and middle blocks must be a multiple of the hash block size. A stateful command with a compression-hash combination will require the Padding engine to insert padding before the hash operation. In the other words, the user must submit a CMP>PAD>HASH command to ensure that the hash input is block properly aligned.

The XR9240 Authentication engine supports a session-based fixed slice hash operation that splits a large command into smaller fixed slices. The XR9240 Authentication engine calculates the hash result for each slice and outputs the results to the destination result buffer. The hash MD5, SHA1, SHA256, SHA384 and SHA512 algorithms support fixed slice hash. Slice hash is enabled by the SLICE bit in the "Session Control Word 0". The slice size is set by the field SLICE_SZ[4:0] in "Session Control Word 2 - Authentication". Fixed slice hash operations may not be combined with other operations such as compression or encryption, but two different hash algorithms may be enabled in the same command. The output length for a fixed slice hash is fixed and algorithm dependant: MD5 (128bits), SHA1 (160 bits), SHA224 (224 bits), SHA256 (256 bits), SHA384 (384 bits), and SHA512 (512 bits).

The XR9240 may be configured to verify and remove the MAC for decode operations. For decode operations, the bytes preceding the authentication toggle "Off" token will be treated as the MAC; the number of bytes to be removed is defined by the MAC_SZ[4:0] field in the "Session Control Word 2 - Authentication". For encode operations, a MAC of programmable length may be inserted into the data stream preceding the hash toggle off token (see "Action Token Flags").

For authentication operations, the XR9240 allows skipping on a byte boundary anywhere in the source data being processed by a authentication command. This feature is command based (one skip per command only) and is enabled using the AUTH action flag token in the user descriptor. Compression and encryption operations that are combined with the authentication operation may not skip the source data. In addition, a MAC cannot be inserted when skipping.

The Authentication engine supports a mute feature that is granular to the nibble (see "Transform Engine Header" Control Word 2). The 16B mute region may start from byte 0-127 from the authentication toggle "On" token.

The MAC/hash result may be output to a sideband buffer (see "Sideband Data").

5.4.2.2 HMAC, SSL 3.0-MAC Operation

HMAC

The XR9240 performs the HMAC calculation:

HMAC MAC = H(K XOR Pad2 || H(K XOR Pad1 || M))

The expression may be abbreviated as:

HMAC MAC(K,M) = H(OPAD, H(IPAD,M))

where:

IPAD = H(K XOR Pad1)

OPAD = H(K XOR Pad2)

The XR9240 Software Development Kit (SDK) will calculate the IPAD and OPAD, input the message data M, and then initialize the IPAD and OPAD fields for the session. If not using Exar's SDK, this calculation must be done in software.

SSL 3.0 MAC

The calculation for SSL 3.0 MAC is:

SSL 3.0 MAC MAC = H(K || Pad2 || H(K || Pad1 || Seq. No. || Type || Length || Application Data))

The expression may be abbreviated as:

SSL 3.0 MAC MAC(K,M) = H(OPAD, H(IPAD,M))

where:

IPAD = H(K || Pad1)

OPAD = H(K || Pad2)

For a SSL3.0-MAC MD5 or SHA-256 operation, the SDK or software must pre-compute the IPAD and OPAD using H(K || Pad1) and H(K || Pad2) then input the data "Seq. No. || Type || Length || Data".

For a SSL3.0-MAC SHA-1 operation, the SDK or software does not need to pre-compute the IPAD and OPAD may directly send K to the XR9240 device with the format "Seq. No. || Type || Length || Data".

5.4.2.3 GMAC Operation

A GMAC operation is implemented as a special case of an "AES-GCM and GMAC Operation", where the cipher text length is zero. The GMAC calculation is performed by the Authentication engine, using the AAD as the input data.

For a GMAC operation, the AAD source token must be ahead of the data source token, and the AUTH_LEN source token that identifies the AAD length must be ahead of the AAD source token.

5.4.2.4 XCBC-MAC-96, CMAC Operation

XCBC-MAC-96

The XCBC-MAC-96 algorithm is used to secure messages of arbitrary length and is a variant of the basic CBC-MAC with the obligatory 10 times padding. The XCBC-MAC-96 calculations require numerous encryption operations that must be accomplished using an AES 128-bit key.

Given a 128-bit secret key K, XCBC-MAC-96 is calculated as follows for a message M that consists of n blocks, M[1]...M[n], in which the blocksize of blocks M[1]...M[n-1] is 128 bits and the blocksize of block M[n] is between 1 and 128 bits:

1. Derive three 128-bit keys (K1, K2 and K3) from the 128-bit secret key K, as follows:

   K1 = 0x01010101010101010101010101010101 encrypted with Key K

   K2 = 0x02020202020202020202020202020202 encrypted with Key K

   K3 = 0x03030303030303030303030303030303 encrypted with Key K

where:

K1 = Constant1A encrypted with Key K | Constant1B encrypted with Key K. However, the second encryption operation is only needed for XCBC-MAC with keys greater than 128 bits; thus, it is not included in the definition of XCBC-MAC-96.

2. Define E[0] = 0x00000000000000000000000000000000

3. For each block M[i], where i = 1 ... n-1:

   XOR M[i] with E[i-1], then encrypt the result with Key K1, yielding E[i].

4. For block M[n]:

   a. If the blocksize of M[n] is 128 bits:

   XOR M[n] with E[n-1] and Key K2, then encrypt the result with Key K1, yielding E[n].

   b. If the blocksize of M[n] is less than 128 bits:

      i. Pad M[n] with a single "1" bit, followed by the number of "0" bits (possibly none) required to increase M[n]'s blocksize to 128 bits.

      ii. XOR M[n] with E[n-1] and Key K3, then encrypt the result with Key K1, yielding E[n].

5. The authentication value is the leftmost 96 bits of the 128-bit E[n].

If M is the empty string, pad and encrypt as in (4b) to create M[1] and E[1] (this will never be the case for ESP or AH, but is included for completeness sake).

Given a a key K and a message M consisting of 128-bit blocks M_1, M_2, M_3, ... M_n, Figure 5-9 illustrates the XCBC-MAC algorithm.

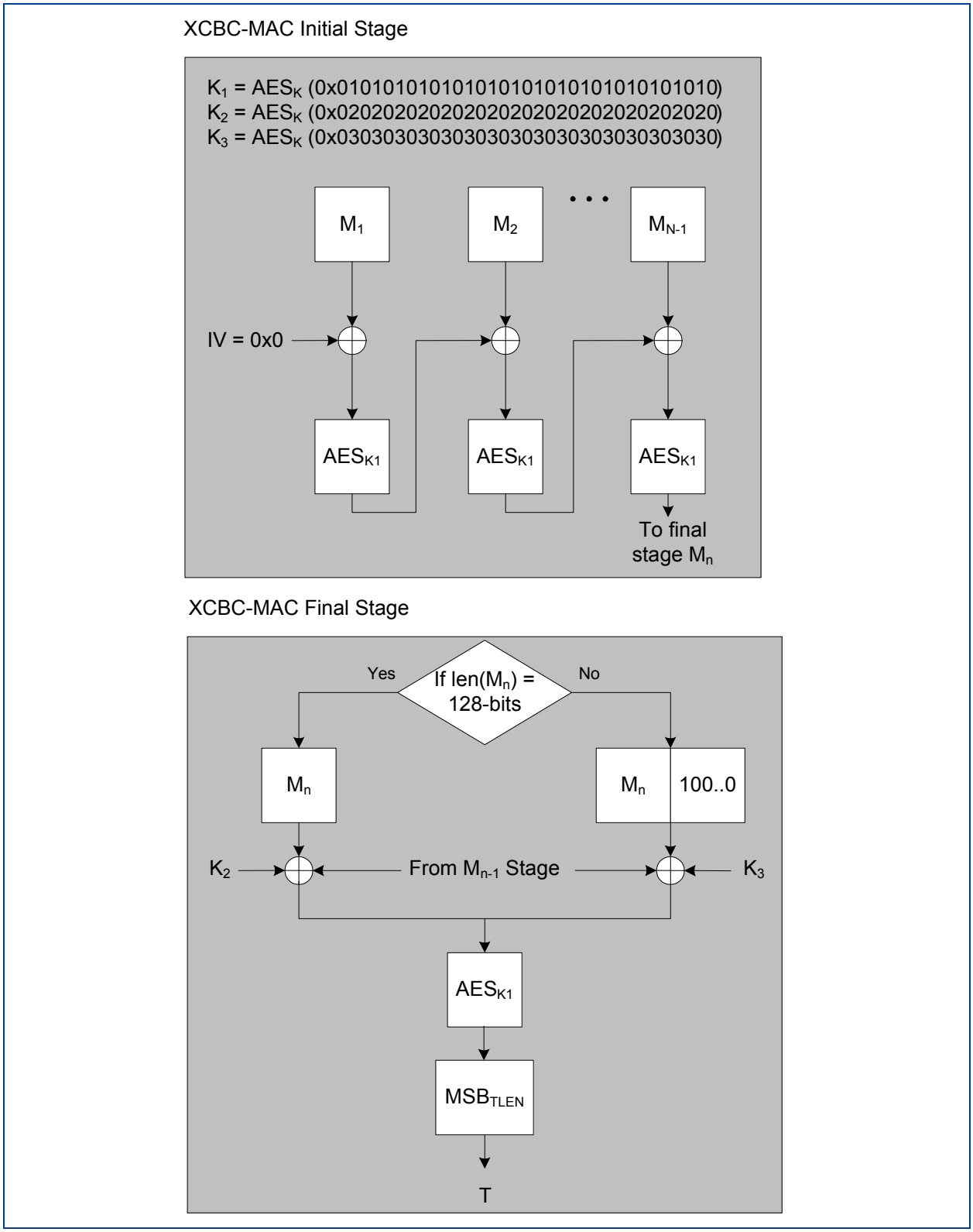


Figure 5-4. XCBC-MAC-96 Implementation

CMAC

The CMAC calculation is similar to the XCBC-MAC-96 calculation. Given a a key K and a message M consisting of 128-bit blocks M_1, M_2, M_3, ... M_n, Figure 5-5 illustrates the CMAC algorithm.

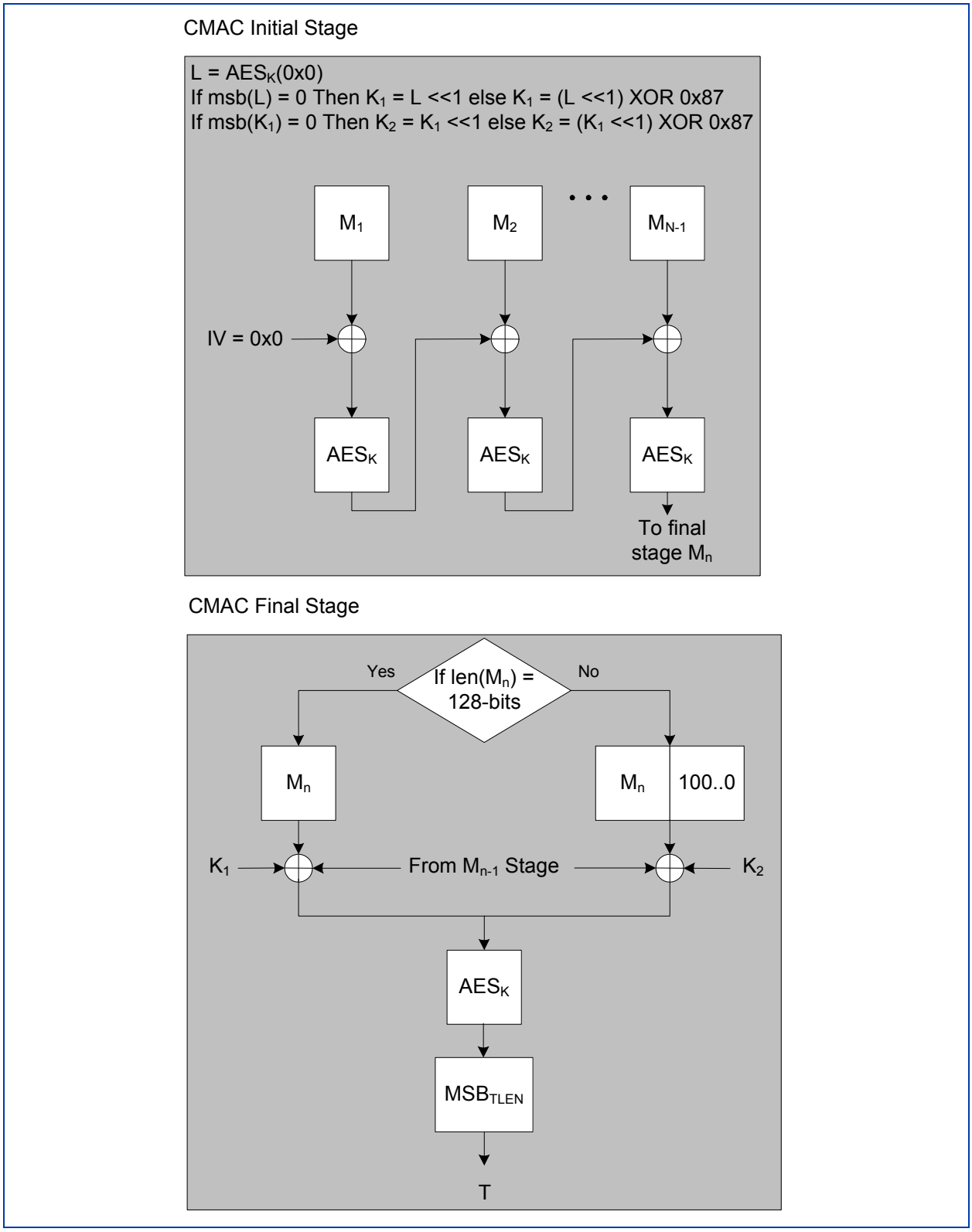


Figure 5-5. CMAC Implementation

5.4.2.5 P_hash Operation

P_hash is used to accelerate SSL/TLS/SRTCP, in particular, the pseudo-random function (PRF) calculation that is used widely in key exchange protocols. To be clear, the P_hash is defined as (secret, seed), whereas the PRF is defined as (secret, label, seed).

P_hash uses a single hash function to expand a secret and seed into an arbitrary quantity of output. The general expression is:

```
P_hash(secret, seed) = HMAC_hash(secret, A(1) + seed) +
    HMAC_hash(secret, A(2) + seed) +
    HMAC_hash(secret, A(3) + seed) + ...
```

where:

+ indicates concatenation,

A() is defined as:

A(0) = seed

A(i) = HMAC_hash(secret, A(i-1))

In the XR9240 device, the P_hash algorithm may be iterated up to 32 times to produce the required quantity of data. The iteration number is encapsulated in the auth_len token, which is 2 bytes in length in P_hash mode. The XR9240 device supports seeds up to 80 bytes in length.

When performing a P_hash operation, the secret should be input as the MACKEY and the seed should be input as the source data. The P_hash result is output to the sideband buffer.

5.4.3 Encryption/Decryption Engine

The XR9240 Encryption engine supports multiple encryption/decryption algorithms including AES-GCM, AES-CBC, AES-CTR, AES-F8, AES-ECB, AES-XTS, 3DES-CBC, RC4 and wireless algorithms UEA1, UEA2, EEA1, EEA2 and EEA3. The XR9240 accepts an IV input for block ciphers and substitution-box (SBOX) input for RC4 operations.

The XR9240 design for the AES-CBC, AES-CTR, AES-ECB, 3DES-CBC algorithms are not described in detail in this section because industry standard implementations were followed.

The encryption algorithm parameters are defined in the "Transform Engine Header" Control Word 1. The ENC_ALG[3:0] field is used to set the encryption algorithm for the session. The encryption key length is set using the ENC_KEY_LEN[6:0] field.

5.4.3.1 AES-GCM and GMAC Operation

An AES-GCM operation is processed differently from other AES operations. The output of an AES-GCM operation has two parts: a cipher text whose length is identical to the plain text, and an authentication tag.

As shown in Figure 5-6, the cipher text is an output of the Encryption engine, while the authentication tag is an output of the Authentication engine. Both engines must be enabled by the host software when performing an AES-GCM operation.

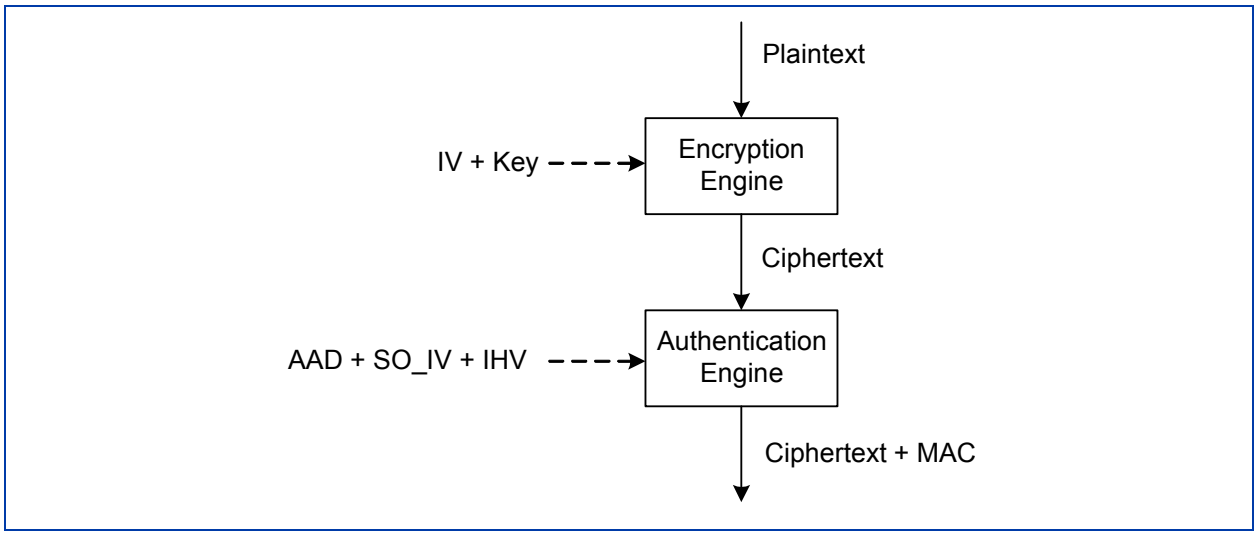


Figure 5-6. AES-GCM Operation

For both AES-GCM and GMAC operations, the AAD source token must be ahead of the data source token, and the AAD_len source token must be ahead of AAD source token.

Figure 5-7 shows the detailed XR9240 implementation of the AES-GCM algorithm.

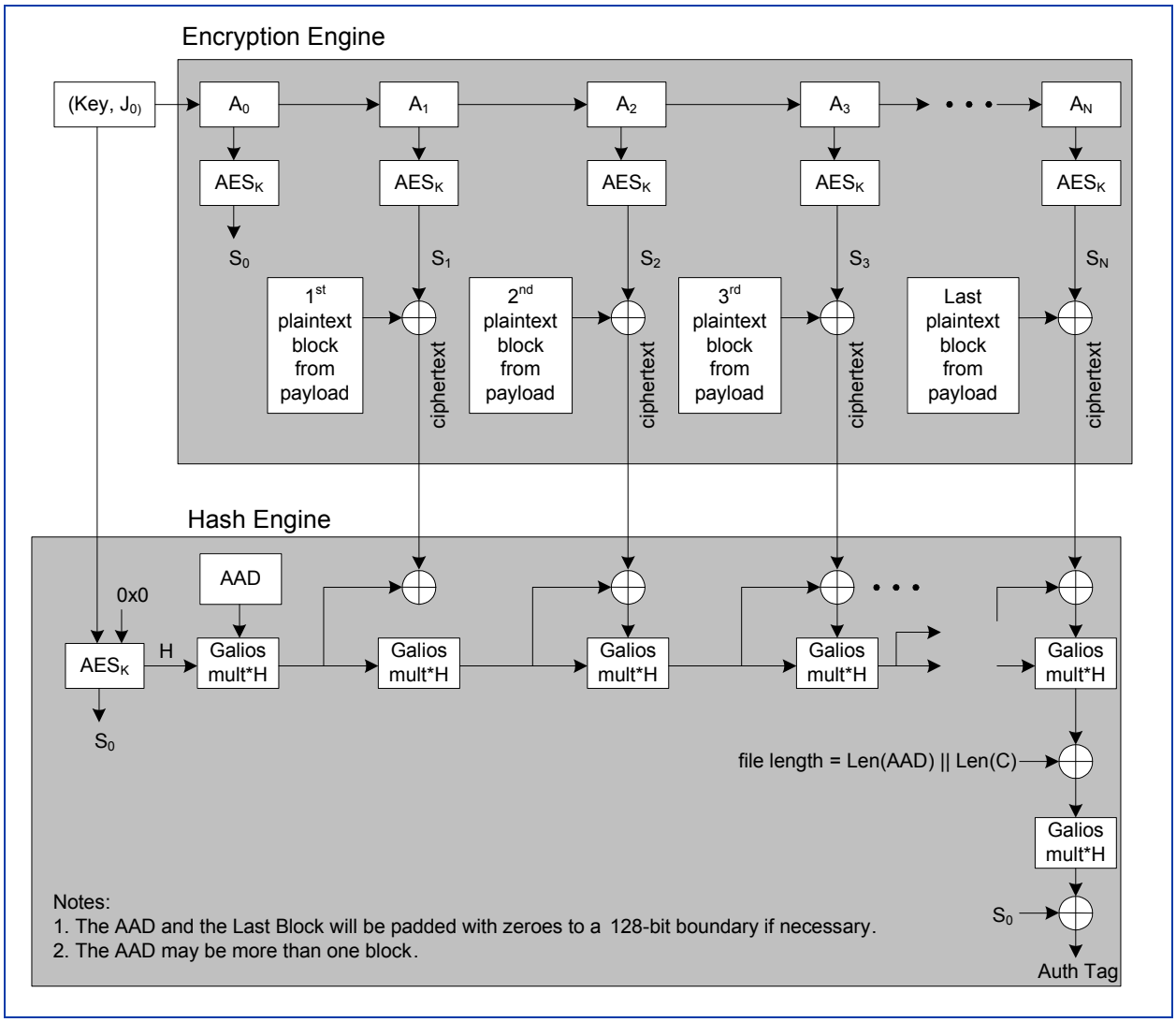


Figure 5-7. Detailed AES-GCM Implementation

The inputs to the encryption engine are the key, the J_0, and the 16 byte plaintext data blocks from the payload, and the file length. The encryption engine ciphertext output is fed into the hash engine, along with the J_0, and the AAD.

The input parameter J_0 is defined as:

If len(IV) = 96

$J_0 = IV \,||\, 0_{31} \,||1.$

If len(IV) ≠ 96

$s = 128[len(IV)/128]\text{-}len(IV),$

$J_0 = GHASH_H(IV||0_{s+64}||[len(IV)]64)$

where $H = CIPH_K(0^{128})$

The encrypted J_0, also called S_0, is then calculated using:

$$S_0 = CIPH_K(J_0)$$

GMAC is a special case of AES-GCM mode in which the cipher text size equals zero. As shown in Figure 5-8, a GMAC operation is calculated in the hash engine, where the input data is the AAD from the previous engine.

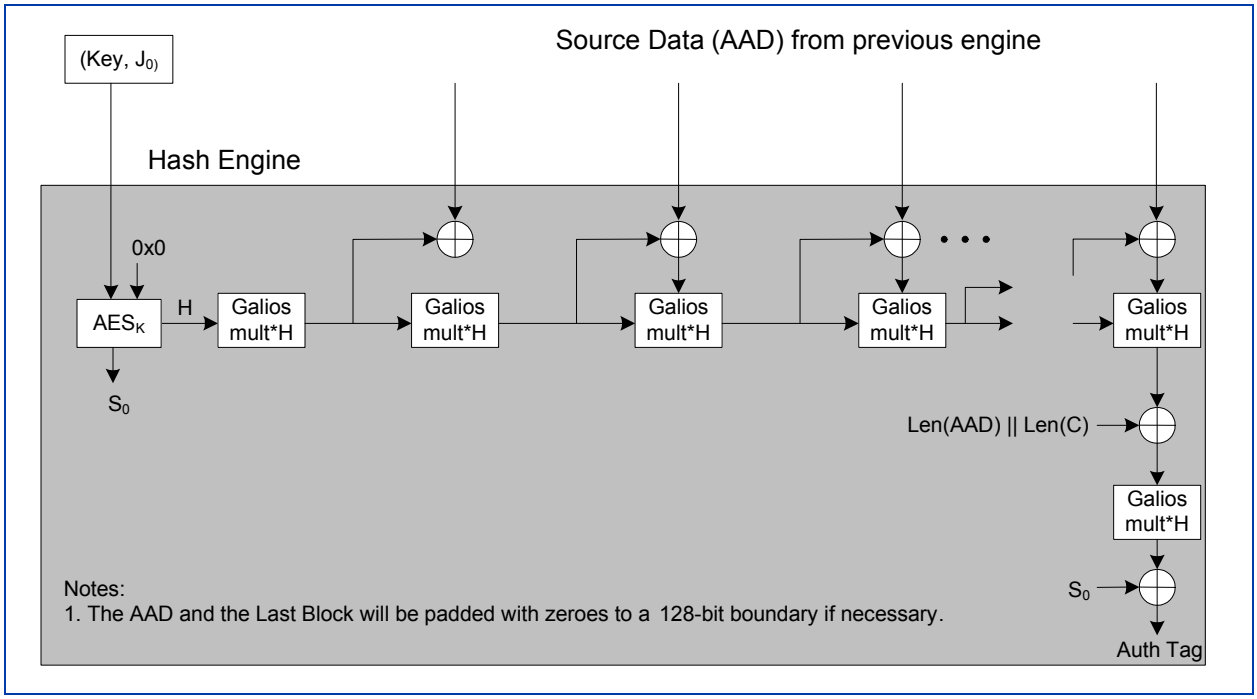


Figure 5-8. Detailed GMAC Implementation

5.4.3.2 AES-F8 Operation

The AES-F8 calculation is computed using the procedure shown in Figure 5-9, as defined by the Secure Real-time Transport Protocol (SRTP) RFC3711.

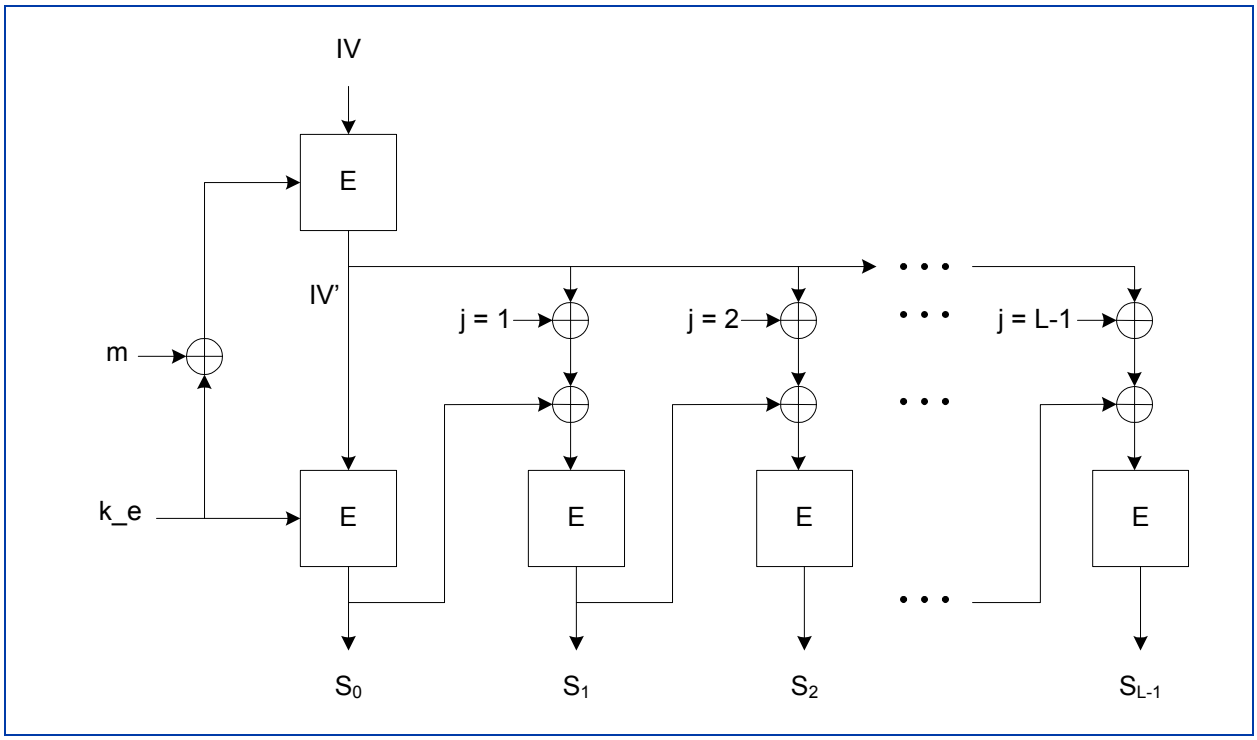


Figure 5-9. AES-F8 Implementation

5.4.3.3 AES-XTS Operation

AES-XTS is used to encrypt storage data. The XR9240 device supports both AES-XTS-256 and AES-XTS-512. AES-XTS requires two key inputs.

AES-XTS commands may contain multiple sectors that may optionally be separated by a Data Integrity Field (DIF). The XR9240 supports sector sizes that are a power of 2 from 64B to 64MB (the sector size does not include the DIF field). AES-XTS commands that share the same configuration may be concatenated. Although there is no limit on the number of sectors, the total size of the command must be less than 2^{48} bytes.

The XR9240 supports two AES-XTS DIF protection modes when DIF is enabled: (1) protect the sector without the DIF, and (2) protect the sector with the DIF. In the first mode, the XR9240 verifies the DIF CRC, performs the encryption/decryption, and then recalculates the CRC. In the second mode, the XR9240 verifies the DIF CRC and performs the encryption for encode commands, and performs the decryption and then verifies the CRC for decode commands.

The XTS_CTL[7:0] field defined in "Session Control Word 1 - Encryption" is used to enable/disable DIF, as well as set the DIF mode, format and AES-XTS sector size. If DIF is disabled, the output will naturally not contain the DIF.

The XR9240 supports two DIF formats. Figure 5-10 gives an example AES-XTS command with 3 sectors and DIF enabled for DIF format T10/03-310r0(T10/03-224r0).

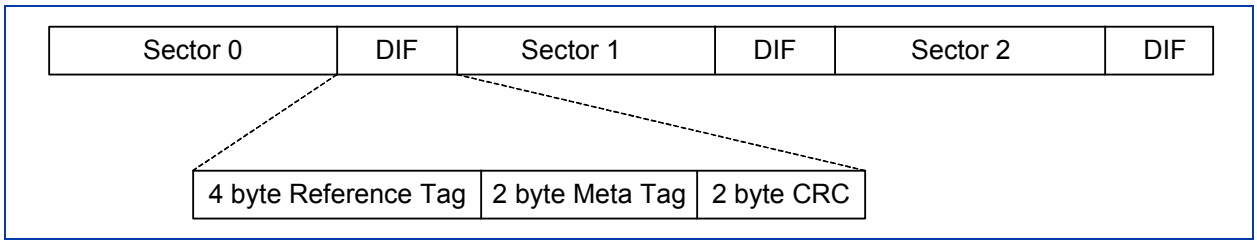


Figure 5-10. Source Data Organization T10/03-310r0 (T10/03-224r0)

Figure 5-11 gives an example AES-XTS command with 3 sectors and DIF enabled for DIF format T10/08-044r1.

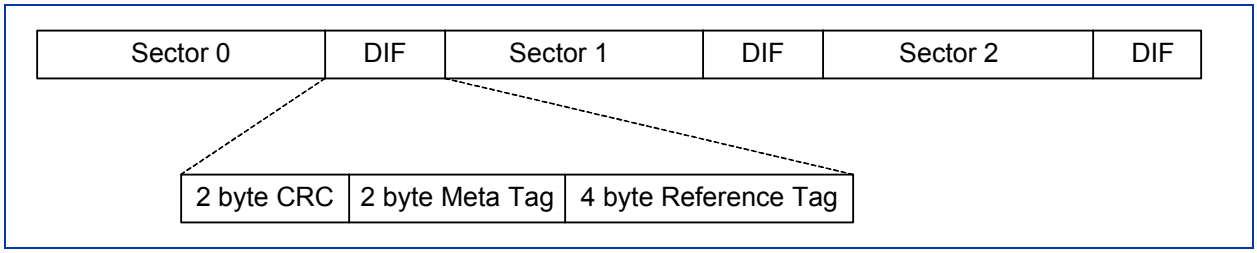


Figure 5-11. Source Data Organization T10/08-044r1

The XR9240 supports tweakable AES-XTS. If used, the 128-bit tweak value should be input as the IV token.

Real time verification may be enabled for AES-XTS encode commands. When real time verification is enabled for encode operations, the key is organized as:

EKEY1 || FKEY1 || EKEY2

Where:

EKEY1 = Encryption key 1

FKEY1 = Decryption key 1

EKEY2 = Encryption key 2

5.4.3.4 RC4 Operation

The RC4 algorithm generates a pseudo-random stream of bits, called a keystream. The keystream is bit-wise XOR'd with the plaintext for encryption or decryption. The XR9240 device generates the keystream using the standard RC4 Key Scheduling Algorithm that is then passed through a pseudo-random generation algorithm (PRGA). These standard algorithm are described in the following sections.

The XR9240 supports stateful RC4 operation. The key is a required input for the first block of a stateful RC4 operation. For the intermediate blocks and last block, the required input are the algorithm variables “SBOX”, “i” and “j” defined below. For each stateful or stateless RC4 command, the keystream is output to the host so that it may be used as the keystream input for subsequent blocks.

Key scheduling algorithm (KSA)

The XR9240 device generates the keystream using a permutation of 256 possible bytes (denoted by the array "SBOX" in the code example below), and two 8-bit index-pointers (denoted "i" and "j").

```
for i from 0 to 255
    SBOX[i] := i
endfor
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap values of SBOX[i] and SBOX[j]
endfor
```

The array "SBOX" is first initialized to the identity permutation. The permutation requires an input key and key length. The key length is defined as the number of bytes in the key and must be 256 bytes for RC4. However, the XR9240 device allows the software to enter a key length that is less than 256 bytes as long as the entered key length is a common multiple of the input key length and 4. For example, if the software key length is 5 bytes, the XR9240 requires that the input key length is 20 bytes. The XR9240 hardware will then expand the 20 bytes to 256 bytes.

Pseudo-random Generation Algorithm (PRGA)

The XR9240 implements the well-known Pseudo-random Generation Algorithm (PRGA) below to generate the RC4 keystream.

```
i := 0
j := 0
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + SBOX[i]) mod 256
    swap values of SBOX[i] and SBOX[j]
    K := SBOX[(SBOX[i] + SBOX[j]) mod 256]
    output K
endwhile
```

5.4.3.5 UEA1, UIA1 Operation

The UEA1 and UIA1 algorithms are widely used in Universal Mobile Telecommunications System (UMTS) and are based on the KASUMI block cipher. Note that real time verification is not supported for these algorithms.

UEA1 Implementation

Figure 5-12 illustrates the UEA1 implementation. The inputs to the UEA1 algorithm are listed in Table 5-8.

Table 5-8. UEA1 Algorithm Input Parameters

Input Parameter	Size (bits)	Description
COUNT	32	Frame dependent input COUNT[0]…COUNT[31]]
BEARER	5	Bearer identity BEARER[0]…BEARER[4]
DIRECTION	1	Direction of transmission DIRECTION[0]
CK	128	Confidentiality key CK[0]….CK[127]
IBS	1-20000	Input bit stream IBS[0]….IBS[LENGTH-1]
KS		Key stream

When a command is organized,

DIRECTION is from the session field

Predata IV = COUNT || BEARER || 0…0

Predata Key = CK

OBS is the output stream

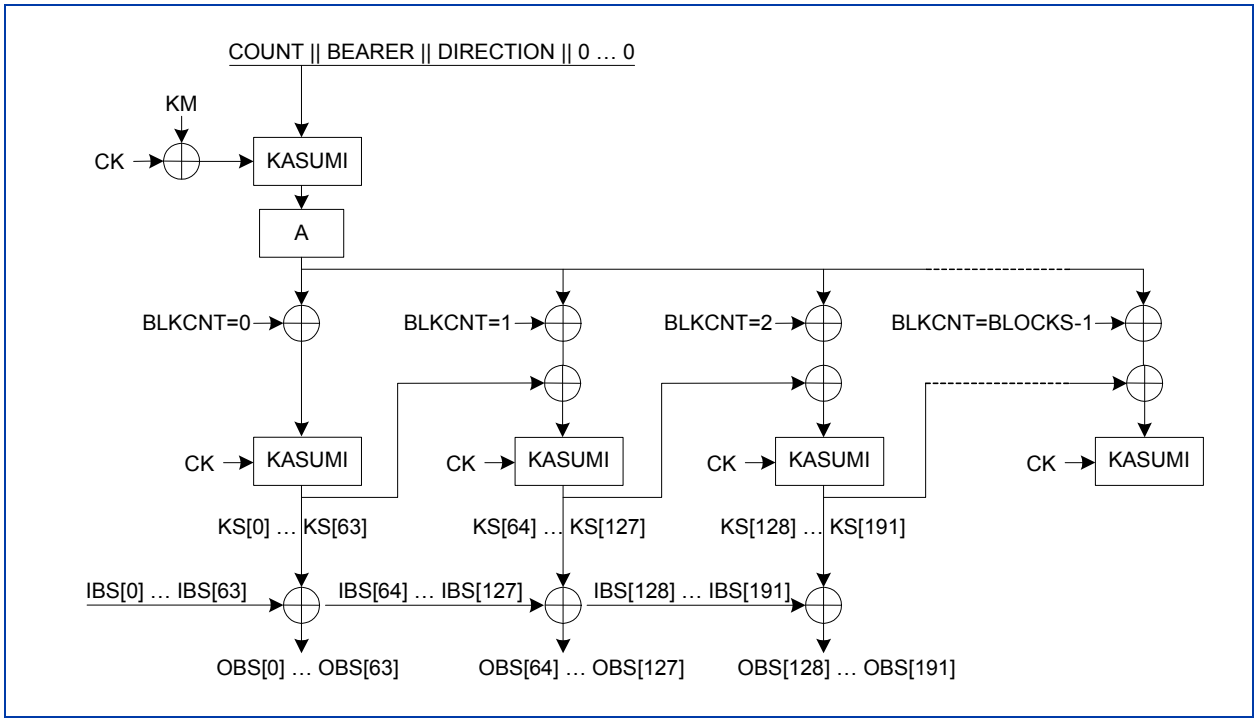


Figure 5-12. UEA1 Implementation

UIA1 Implementation

Figure 5-13 illustrates the UIA1 implementation.

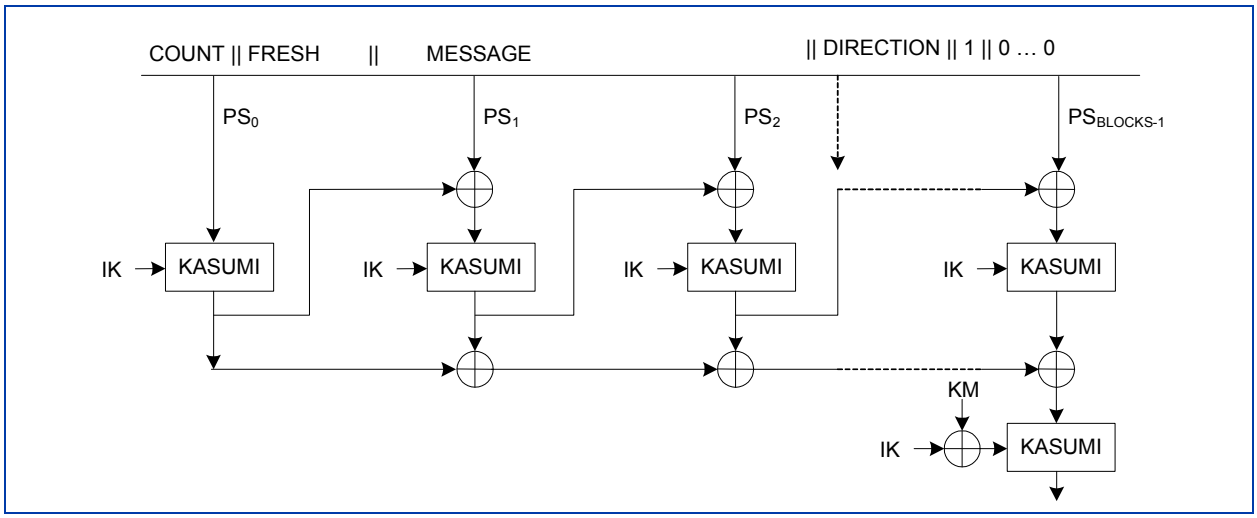


Figure 5-13. UIA1 Implementation

The inputs to the UIA1 algorithm are listed in Table 5-9.

Table 5-9. UIA1 Algorithm Input Parameters

Input Parameter	Size (bits)	Description
COUNT-I	32	Frame dependent input COUNT-I[0]…COUNT-I[31]
FRESH	32	Random number FRESH[0]…FRESH[31]
DIRECTION	1	Direction of transmission DIRECTION[0]
IK	128	Integrity key IK[0]…IK[127]
KM		Key Mask
MESSAGE	LENGTH	Input bit stream

When a command is organized,

Predata IHV = COUNT-I || FRESH

Predata Key = IK

If UIA1 is called with UEA1 in the same command,

UIA1 IK = UEA1 CK

5.4.3.6 EEA1, EIA1 Operation

The EEA1 and EIA1 algorithms are also based on the KASUMI block cipher. EEA1 is the same as UEA2; EIA1 is the same as UIA2. Note that real time verification is not supported for these algorithms.

EEA1/UEA2 Implementation

Figure 5-14 illustrates the EEA1 implementation. The inputs to the EEA1 algorithm are listed in Table 5-10.

Table 5-10. EEA1 Algorithm Input Parameters

Input Parameter	Size (bits)	Description
COUNT-C	32	Frame dependent input COUNT[0]…COUNT[31]]
BEARER	5	Bearer identity BEARER[0]…BEARER[4]
DIRECTION	1	Direction of transmission DIRECTION[0]
CK	128	Confidentiality key CK[0]….CK[127]
IBS	LENGTH	Input bit stream IBS[0]….IBS[LENGTH-1]

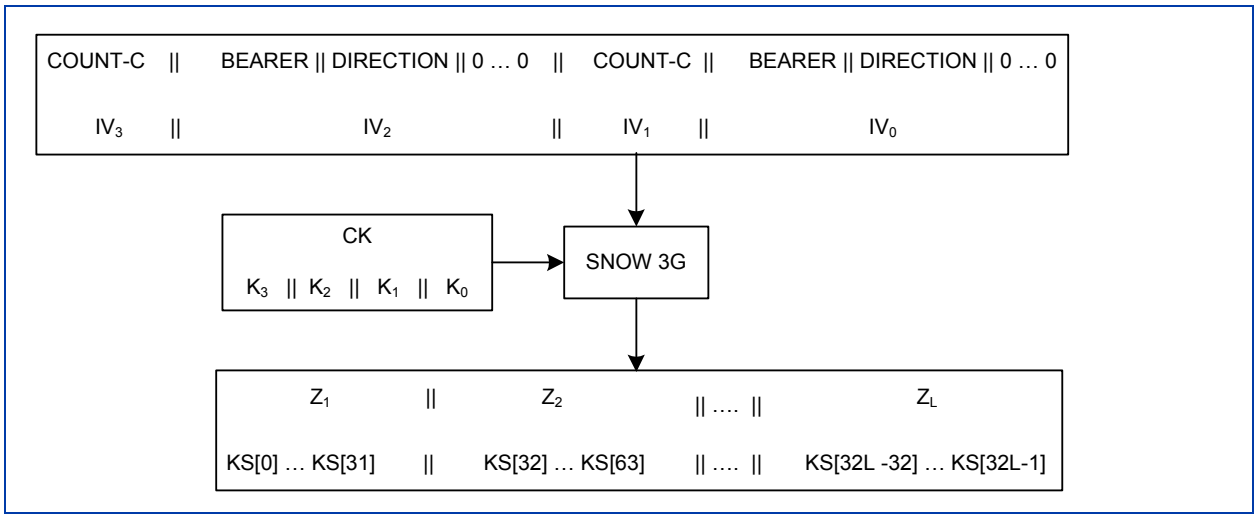


Figure 5-14. EEA1 Implementation

In Figure 5-14,

K3 = CK[0] || CK[1] || CK[2] || … || CK[31]

K2 = CK[32] || CK[33] || CK[34] || … || CK[63]

K1 = CK[64] || CK[65] || CK[66] || … || CK[95]

K0 = CK[96] || CK[97] || CK[98] || … || CK[127]

IV3 = COUNT-C[0] || COUNT-C[1] || COUNT-C[2] || … || COUNT-C[31]

IV2 = BEARER[0] || BEARER[1] || … || BEARER[4] || DIRECTION[0] || 0 || … || 0

IV1 = COUNT-C[0] || COUNT-C[1] || COUNT-C[2] || … || COUNT-C[31]

IV0 = BEARER[0] || BEARER[1] || … || BEARER[4] || DIRECTION[0] || 0 || … || 0

SNOW 3G is run to produce the keystream consisting of the 32-bit words Z1 … ZL. The word produced first is Z1, the next word Z2 and so on.

The sequence of keystream bits is KS[0] … KS[LENGTH-1], where KS[0] is the most significant bit and KS[31] is the least significant bit of Z1, KS[32] is the most significant bit of Z2 and so on.

The output (not shown in the diagram) is:

OBS[i] = IBS[i] XOR KS[i]

When a command is created,

IV = COUNT || BEARER || DIRECTION || 0…0

Key = CK

EIA1/UIA2 Implementation

Figure 5-14 illustrates the EIA1 implementation. The inputs to the EIA1 algorithm are listed in Table 5-10.

Table 5-11. EIA1 Algorithm Input Parameters

Input Parameter	Size (bits)	Description
COUNT-I	32	Frame dependent input COUNT-I[0]…COUNT-I[31]
FRESH	32	Random number FRESH[0]…FRESH[31]
DIRECTION	1	Direction of transmission DIRECTION[0]
IK	128	Integrity key IK[0]…IK[127]
MESSAGE	LENGTH	Input bit stream

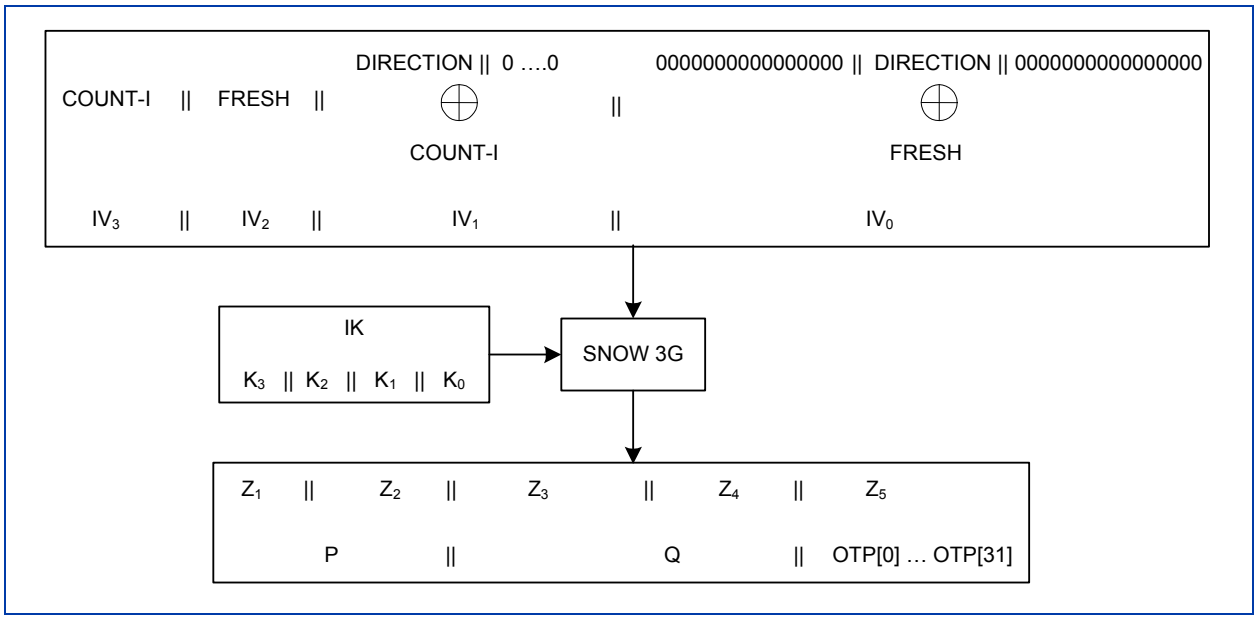


Figure 5-15. EIA1 Implementation

In Figure 5-15,

K3 = IK[0] || IK[1] || IK[2] || … || IK[31]

K2 = IK[32] || IK[33] || IK[34] || … || IK[63]

K1 = IK[64] || IK[65] || IK[66] || … || IK[95]

K0 = IK[96] || IK[97] || IK[98] || … || IK[127]

IV3 = COUNT-I[0] || COUNT-I[1] || COUNT-I[2] || … || COUNT-I[31]

IV2 = FRESH[0] || FRESH[1] || FRESH[2] || … || FRESH[31]

IV1 = DIRECTION[0] XOR COUNT-I[0] || COUNT-I[1] || COUNT-I[2] || … || COUNT-I[31]

IV0 = FRESH[0] || FRESH[1] || … || FRESH[15] || FRESH[16] DIRECTION[0] || FRESH[17] || … || FRESH[31]

Let OTP[0], OTP[1], OTP[2], …, OTP[31] be bit-variables such that

Z5 = OTP[0] || OTP[1] ||… || OTP[31],

i.e. OTP[0] is the most and OTP[31] the least significant bit of Z5.

The EIA1 calculation is shown in Figure 5-16. The input message is evaluated using the values for Z1 through Z5 from the initialization.

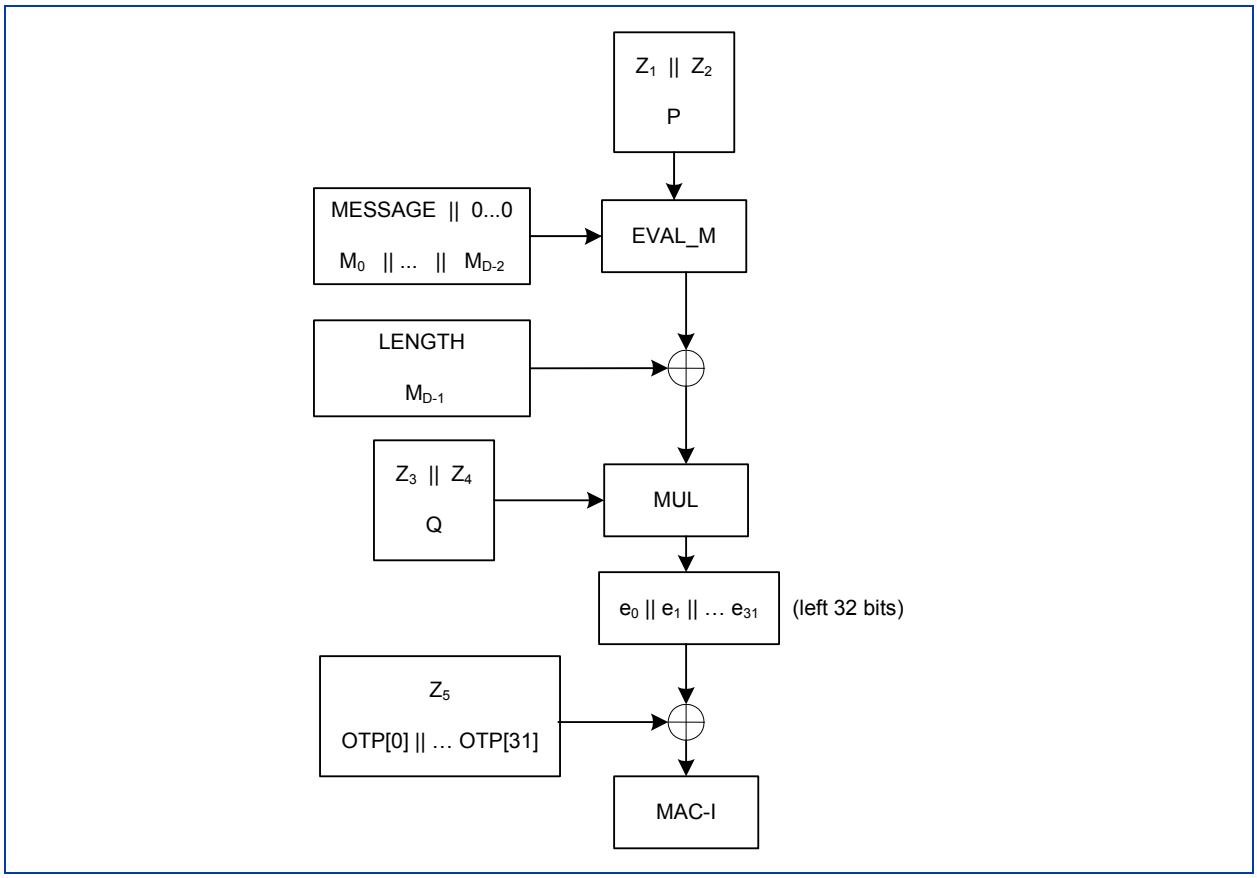


Figure 5-16. EIA1 Calculation

5.4.3.7 EEA3, EIA3 Operation

The EEA3 and EIA3 wireless algorithms are based on the ZUC algorithm. The command format for these algorithms is similar to "UEA1, UIA1 Operation" and "EEA1, EIA1 Operation".

5.4.4 Pad/Depad Engine

The Pad engine may be used to add or remove spare bytes to/from the data stream to maintain byte alignment rules based on the padding algorithm defined in the PAD_ALG[3:0] field of the "Transform Engine Header" Control Word 1. The padding length may be rounded to modulo 4/8/16/32/64/128/256, based on the PAD_MOD[2:0] field in the TE Session Control Word 1. The maximum padding/depadding length is 256 bytes.

The XR9240 uses tokens to inform the Pad engine where to start/stop padding the data. The Padding/depadding engine inserts/removes the padding bytes according to the relative position of the padding toggle off token. The padding bytes are always located before the padding toggle off token. For decode operations, the padding value may be verified by enabling the PV_EN bit. Note that the padding value will be removed after it is verified.

The Pad engine also supports IPsec ESP trailer generation. If the ESP bit in the "Transform Engine Header" Control Word 1 is set, the Pad engine will add the ESP trailer to the payload and append the IPv4/IPv6 protocol number defined in the NXT_HDR[7:0] field to the data stream.

A depadding operation requires the pad length in data stream. The Depadding engine uses the padding toggle off token and the pad length to determine the padding region in the data stream. The padding length offset from padding toggle off token defines the padding region. The padding token is not required for LZS and gunzip decompression operations because the Decompression engine discards all bytes following the EOR.

5.5 Public Key Module

The Public Key module offloads public key operations from the host. The supported PK algorithms include RSA, DSA, DH, ECDSA and ECDH.

As shown in Figure 5-17, the PK module consists of eight PK engines and a PK manager. The PK module interfaces to the XR9240 DMA Inbound Data Controller (IDC) and Outbound Data Controller (ODC).

The PK manager parses the PK command header from the DMA IDC and dispatches commands to the PK engines using a round arbitration to the first available PK engine. The host may disable a PK engine from the arbitration process via the "Public Key Load Balance Enable Register". When a command completes, the PK manager sends the result to the DMA ODC.

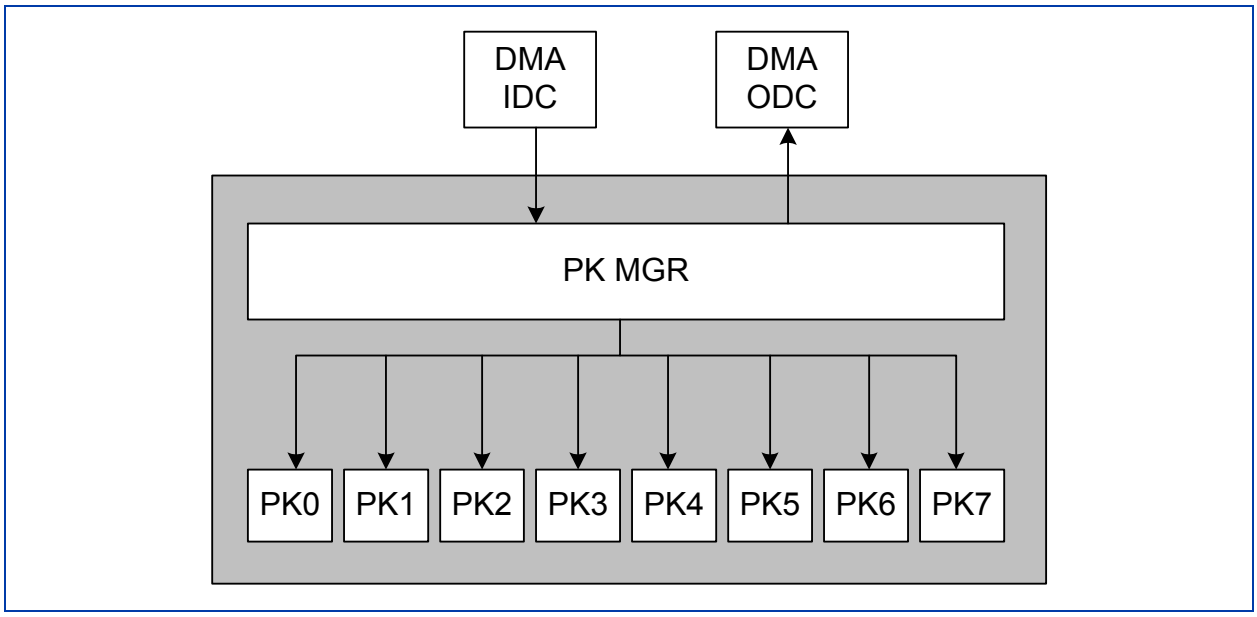


Figure 5-17. Public Key Block Diagram

Each PK engine contains two exponentiation (EXP) modules that operate as master/slave for commands such as RSA CRT and modExp. For these types of commands, the master and slave EXP modules are loaded with different microcode programs using the MSTR_SLV bit in the Section 3.3.1.2. If all the operations are coded with the same set of microcode images, there is no need to download new microcode with each operation.

Before disabling the power or clock to a PK engine power domain, the host must first disable the PK engine in that power domain from arbitration. The PK engines are grouped into the power domains defined in the "Power Enable Register".

Errors generated from the PK module are listed in Table 10-2.

5.5.1 EXP Module Programming Model

The EXP module is a combination of logic and firmware. The EXP programming model is shown in Figure 5-18.

During runtime, the host may send a PK command that instructs the EXP module to load a new PK algorithm to the control path. If the PK algorithm will be used for subsequent PK commands, the host will only need to send PK commands to set the datapath registers.

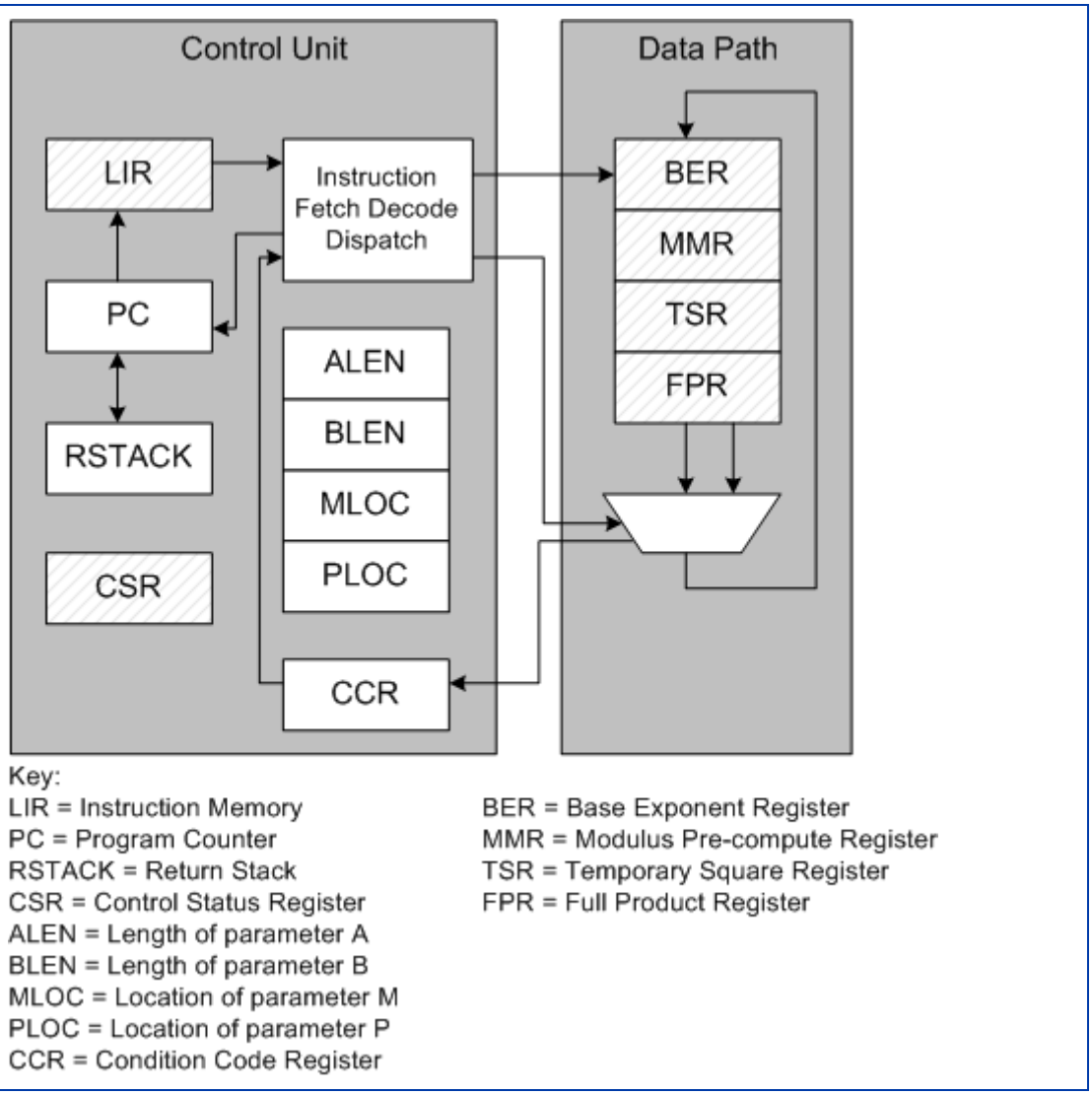


Figure 5-18. Public Key Module Programming Model

The EXP module controls the translation of the PK command source buffer into the appropriate memory location within the PC. The shaded boxes represent the locations that are directly linked to the PK user descriptors. The following sections describes the shaded memory mapped registers in more detail.

5.5.1.1 LIR

The Linear Instruction (LIR) register comprises 1024 32-bit words used to store a sequence of operations for the PK engine.

The initial value of the LIR is technology dependent and should generally be assumed to be undefined until explicitly defined by the user.

The LIR is memory mapped and may be accessed when the PK engine is idle. The LIR is addressed and accessed in units of 32-bit words.

5.5.1.2 BER

The Base and Exponent (BER) register is used to store the base and exponent for exponentiation operations, or the data operands for other operations such as multiplication and addition. Results are also stored in the BER. The BER is memory mapped and may be accessed whenever the PK engine is idle, as indicated by the BUSY bit of the control and status register.

5.5.1.3 MMR

The Modulus and Pre-Compute (MMR) register is used to store the modulus and pre-compute values used for modular reduction. The MMR may also used to provide addressing space for ROM constant values. For PK engine configuration the upper quarter may be used to provide storage for NIST P-curve values. The MMR is memory mapped and may be accessed whenever the PK engine is idle, as indicated by the BUSY bit of the control and status register.

5.5.1.4 TSR

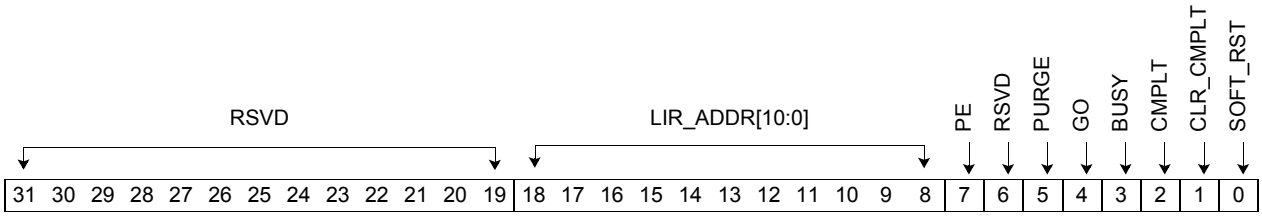
The Temporary and Square (TSR) register is used to store temporary values during the execution of some instructions. It is possible to use it as temporary storage in programs, however, it is only addressable by format C instructions (MOV, DXI, and DXO). The TSR is memory mapped and may be accessed whenever the PK engine is idle, as indicated by the BUSY bit of the control and status register.

5.5.1.5 FPR

The Full Product (FPR) register is used to store full-length products produced in the course of a computation or the result of a non-modular multiplication operation. Like the TSR, the FPR may be used for temporary storage with the same limitations. The FPR is memory mapped and may be accessed whenever the PK engine is idle, as indicated by the BUSY bit of the control and status register.

5.5.1.6 EXP Control and Status

The Control and Status (CSR) register is used to initiate a soft reset of the EXP, control the execution of the PK command, and reflect the computation completion status for each EXP. The CSR is memory mapped and may be accessed whenever the PK engine is idle, as indicated by the BUSY bit of the control and status register. The values stored in this register are not impacted by resets. The format of the PK control and status field is shown below.



Field Name	Bits	Default	Description
RSVD	31:19	0	Reserved.
LIR_ADDR[10:0]	18:8	0	Linear Instruction Register Starting Address. This field is used by the host to write the starting address of the PK algorithm that is stored in the PK engine LIR for the PK command. Each PK command may point to a different PK algorithm.
PE	7	0	Parity Error. If set, this bit indicates whether a single or multiple bit parity error has been detected. The "PK ECC Error Register" should be consulted to diagnose the error type. 0 No parity errors detected 1 Parity error occurred
RSVD	6	0	Reserved.
PURGE	5	0	Purge. Setting the PURGE bit will clear all the EXP registers to ensure that all the keys/operands are flushed. 0 Do not execute the Purge command 1 Execute Purge command
GO	4	0	Start Computation Command Bit. This write only bit controls the execution of the PK command. 0 Do not start computation 1 Start computation
BUSY	3	0	Busy Status. This read only bit reflects the status of the PK engine. A PK engine must be in a not busy state before reading or writing to the data and LIR registers. 0 The PK engine is not busy 1 The PK engine is busy
CMPLT	2	0	Computation Complete Status Bit. This read only bit reflects the status of the PK engine's computation. 0 Computation not complete 1 Computation complete

Field Name	Bits	Default	Description
CLR_CMPLT	1	0	Clear Computation Complete Status Bit. This write only bit allows the host to the clear the status of the PK engine's computation bit CMPLT. 0 Do not clear CMPLT bit 1 Clear CMPLT bit
SOFT_RST	0	0	Soft Reset. This bit can be used to initiate a soft reset of a PK engine. 0 Do not initiate soft reset of PK engine 1 Initiate soft reset of PK engine

5.6 Interlaken Interface Module

The Interlaken interface provides chip-to-chip communication between the XR9240 and an external device such as a FPGA.

Figure 5-19 illustrates the XR9240 DMA and Transform Engine interface to the ILK module. Within the DMA module, eight groups of Inbound/Outbound Data Controller (IDC/ODC) modules target eight groups of ILK interfaces, which connect to eight Transform engines. Within the ILK module, eight groups of data channels interface to the ILK DMA. Interlaken channels 0 and 1 are assigned to transform channel 0, while Interlaken channels 14 and 15 are assigned to transform channel 7.

In addition to the eight groups of Interlaken channel modules, the ILK module also contains a DMA module, ILK MAC and PHY. The ILK DMA includes an arbiter that accepts requests from the XR9240 ILK interface to the transmit user interface, and a receive buffer for each Interlaken channel that reorganizes the ILK burst into XR9240 packet objects. The XR9240 supports a data burst size of 64 bytes. The ILK PHY supports up to eight Serdes lanes. These blocks are described in more detail in "Packet Transmission".

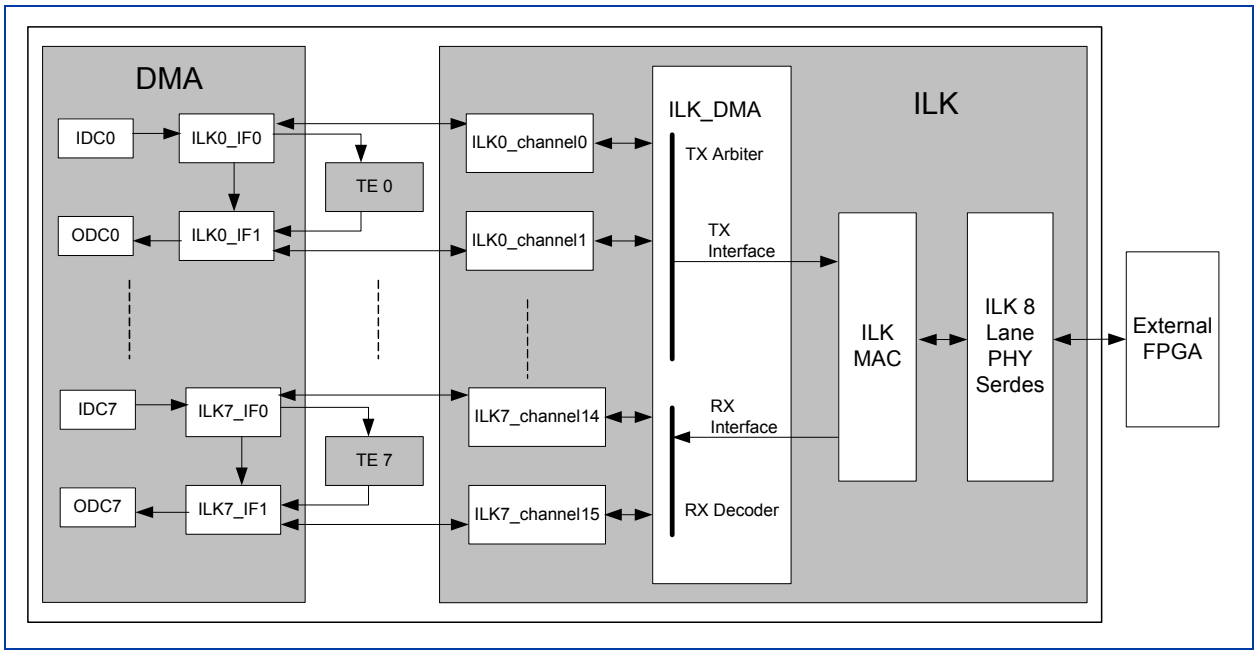


Figure 5-19. Interlaken Module Block Diagram

Refer to "Transform Processing Sequence" for a discussion of how the FPGA and transform engine commands may be interlaced within the XR9240.

The FPGA that is connected to the XR9240 must implement a compatible Interlaken interface. The *XR9240 FPGA Design Guidelines Application Note* contains more detailed information.

Refer to the *XR9240 FPGA Design Application Note*, APN-0001, for a description of how to program the FPGA image through the ILK interface.

Exar plans to offer an ILK interface design for the FPGA that will simplify the FPGA design for customers considerably.

5.6.1 Packet Transmission

The ILK DMA controls the ILK packet transmission between the XR9240 DMA and the external FPGA over the Interlaken interface. Customers using Exar's ILK DMA logic will not need to understand the concepts in this section in detail.

5.6.1.1 Arbitration

An ILK packet from the XR9240 DMA is sent in one or multiple 64B Interlaken data bursts. Transmit arbitration is made at the burst boundary on a round robin basis. The transmit channel buffer stores at least one data burst before it requests for arbitration.

5.6.1.2 Credit System

The XR9240 uses a proprietary credit system for data transmission over the ILK interface that is similar to the PCIe credit system. This protocol must be duplicated in the FPGA by the user.

Channel 16 on the Interlaken interface is used solely for transferring credit availability updates between the XR9240 and FPGA. Each transmit channel maintains a table that records the available credits for that channel. This table is updated by the XR9240 through an Interlaken receive burst packet on receiver channel 16. Each transmit channel maintains a consumed credit counter, which increases by one when one buffer entry of data is sent to the ILK module. Before initiating a burst transmit request to the transmit request arbiter, each transmit channel must ensure there are enough transmit credits available.

Each ILK receive channel maintains a 512B receive buffer. The receive buffer is partitioned into 32 blocks, where each 16B block is allocated as one credit. Therefore, the XR9240 Interlaken receive channel has up to 32 credits. After reset, the transmit credit counter of the sender is initialized to zero and the receive credit counter of the XR9240 channel is initialized to 32. The receive buffer of the FPGA may have more than 32 credits if its receive buffer has more buffer blocks; the XR9240 supports up to 1024 receive credits.

Each receive channel maintains a Credit Available Counter (CAC) that decrements for each 16 bytes of data received. The receive CAC value is advertised to the device on the other end of the link over channel 16. Before the transmitter sends out a burst, it will check that the credit value is large enough to hold the burst. After a burst is sent, the channel transmitter increases its transmit credit counter by the buffer block number of the burst. On the receive side, the receiver of the connection will increase its receive credit counter by one after the block of data is read by the higher layer application. The latest receive credit counter will then be sent to the sender through channel 16.

5.6.1.3 Retry Buffer

Data transfer over the ILK interface is lossless. Each transmit packet is stored in a retry buffer until the ACK is sent through link control channel and fed back by the receiver. Each burst is assigned a 6 bit sequence number, SEQ[5:0]. The sequence number is assigned based on the order of the data bursts and is embedded into the multi-use field of the data packet.

The retry buffer and sequence number are shared between all the ILK channels. Packets sent through link control channel are not stored in the retry buffer.

To save bandwidth, multiple bursts could be acknowledged with the same link control packet. The sequence number in the link control packet is the latest burst received.

If the receiver detects an error on an incoming burst, the receiver will send a NAK and the SEQ number to the transmitter. The transmitter must then resend the burst.

The retry buffer also has a timeout counter. If the data burst or the NAK packet is lost, the transmitter will resend the burst after the timeout event.

5.6.2 ILK Packet Format

This section describes the data and link channel packet format that are based on the Interlaken Protocol Definition.

5.6.2.1 Data Channel Format

Channels 0-15 are dedicated as data channels. A data packet between the XR9240 and FPGA must follow the following rules:

1. Each data burst is sent as a single ILK packet. The Start of Packet (SOP) and End of Packet (EOP) bits must be set during the burst.

2. The end of packet information is sent in the multi-use field (extension channel number) using the format: {End of Frame, 1'b0, SEQ[5:0]}.

3. Inband data and sideband data must be sent in separate frames.

5.6.2.2 Link Control Channel Format

Channel 16 is used to transfer link layer control information. The link control packet, which is similar to DLLP in PCIE, has the format shown in Figure 5-20 and must follow the following rules:

1. A link control packet is always 32B in length. Each channel contains the 8-bit receive credit counter value, CAC[15:0] (see "Credit System").

2. The control word of the link control packet must have the SOP and EOP bits set.

3. The multi-use field is used to transfer the control information and sequence number and is defined as {NAK, ACK, SEQ[5:0]}.

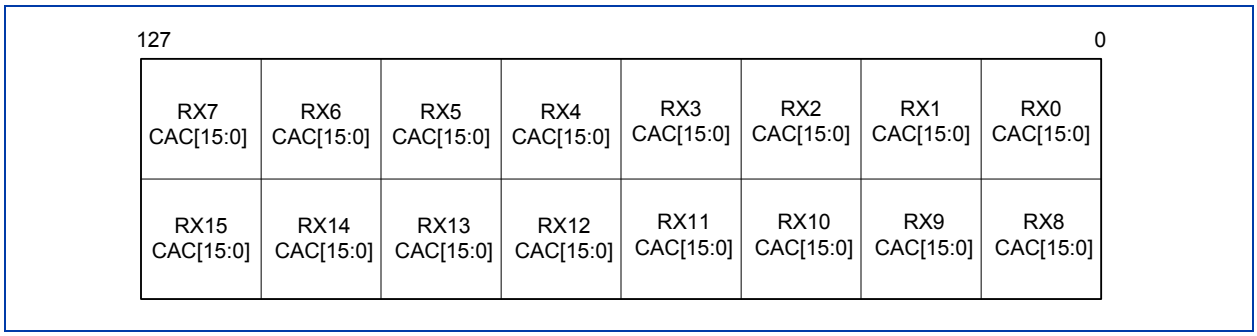


Figure 5-20. ILK Link Control Channel Payload Format

A link control packet is sent periodically, based on a configurable time period. However, sending too many link control packets may cause performance degradation. The parameter LK_UPD_MIN[8:0] in the "ILK DMA Configuration Register" sets the time interval allowed between two link control packets and is set to 0x50 (80 cycles or 200 ns) by default.

5.6.3 Programming the ILK DMA and MAC

The Interlaken DMA and MAC registers are mapped to the XR9240 BAR0 register space. The ILK interface registers should not be written to unless the TX_CFG_RST and RX_CFG_RST bits in the "ILK DMA Reset Register" are set. The address is specified at the byte boundary.

5.6.4 Programming the ILK PHY

The ILK PHY registers must only be programmed when the interface is in reset state and ILK module power is enabled. The ILK PHY may be put into a reset state using the "ILK PHY Control Registers". The Interlaken PHY is in hard reset mode after power on reset.

The Interlaken PHYs have a separate 16-bit address space that is indirectly mapped to the XR9240 register space 0x4900-49FF.

Because the ILK PHY registers are indirectly mapped to the PCIe host space, accessing an ILK PHY register is a two stepprocess.

Step 1 Configure the indirect address information

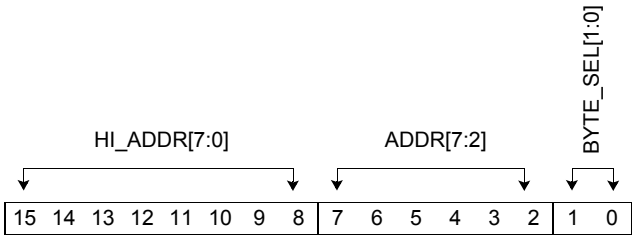
Before accessing an ILK PHY register, program the upper 8 bits of the address, byte enable and PHY select fields in the "ILK DMA PHY Access Register". The description for the "ILK PHY Control Registers" lists the specific values for these fields for each lane.

a. The upper word of the ILK PHY register address is placed into the HI_ADDR[7:0] field.

b. The last two bits of the ILK PHY register address identify the value to be programmed into the BYTE_EN[3:0] field. For example, if the last two bits are 2'b01, then BYTE_EN[3:0] would be set to 4'b0010 to represent byte one.

c. Internally, the XR9240 contains two ILK PHY modules to accommodate eight Interlaken lanes. Each ILK PHY module supports four ILK lanes. The PHY_SEL bit should be set according to which ILK PHY module is to be accessed. ILK PHY 0 controls lanes 0-3 and ILK PHY 1 controls lanes 3-7.

The ILK PHY register address mapping is shown graphically below.



Field Name	Bits	Reset	Description
HI_ADDR[7:0]	15:8	0x09	This field should be programmed into the "ILK DMA PHY Access Register" HI_ADDR[7:0] field.
ADDR[5:0]	7:2	0	This field will be copied to the final address as bits [7:2]. See Step 2 below.
BYTE_SEL[1:0]	1:0	0	This field should be used to program the BYTE_EN[3:0] field in the "ILK DMA PHY Access Register".

Step 2 Access the ILK PHY register from PCIe host

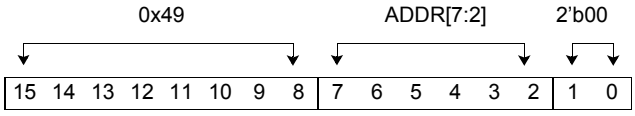
1. Generate the XR9240 PCIe BAR0 address using the mapping defined below.

   a. The upper address bits [15:8] are fixed at 0x49.

   b. Address bits [7:2] are taken from bits [7:2] of the original ILK PHY register address.

   c. Address bits [1:0] are fixed at 2'b00.

   The final XR9240 register format is shown graphically below.



For example, to reset ILK PHY lane 2 using the "ILK PHY Lane Reset Register":

1. Host configures the "ILK DMA PHY Access Register" with BYTE_EN = 4'b0100, PHY_SEL = 0, and HI_ADDR = 8'h40.

2. Host writes 0x000B to the "ILK PHY Lane Reset Register" at address 0x4900.

5.6.5 Receiver Detection

The XR9240 provides the ability for the host to detect whether a FPGA is available on the card. The procedure to do so follows, where all referenced fields are from the "ILK DMA Receiver Detection Register".

1. The host requests a receiver detection test by setting the REQ bit.
2. The ILK PHY clears any previous results and initiates the test.
3. The ILK PHY indicates the test is complete and the results are ready for sampling by setting ACK bit.
4. The host polls the ACK bit until it is set.
5. Once the ACK bit is set, the host can read the DET bit. If DET is set, a FPGA is available on the card.
6. The host clears the REQ bit, which will cause the XR9240 to clear the ACK bit. The transaction is complete.

5.7 RNG

The XR9240 coprocessor contains an internal Random Number Generator (RNG) entropy source that is used to generate a raw random number and may be used to seed FIPS-compliant Deterministic Random Bit Generator (DRBG). The XR9240 SDK will provide API functions that provide a NIST SP 800-90 certified software PRNG.

The RNG is implemented as 16 shielded, free running ring oscillators, with each ring containing a different prime number of gate inversions such that all ring oscillators generate an independent frequency higher than the 125 MHz clock frequency that is used to sample the oscillator state. Exact frequencies are dependent on individual component process factors, operating die temperature, and operating voltage (PVT), but the ratios should be similar over these process variations.

"RNG Control Registers" provide the interface for software to start the random number hardware engine, retrieve sixteen 32-bit random number values, and configure the random number hardware engine.

5.8 FPGA Field Programming Interface

The programming interface allows the host to download or upgrade the FPGA/Flash image. See the *XR9240 FPGA Design Application Note*, APN-0001, for details.

5.9 Clock and Reset Generator

The clock and reset module is responsible for the generation and management of all clocks and resets throughout the device.

5.10 Device Power Management

The power and clocks on specific XR9240 modules may be gated to conserve power.

5.10.1 Power Gating

The sequence to disable power to a XR9240 module is:

1. Disable load balancing to the desired module by clearing the appropriate bit in the "Transform Channel Load Balance Enable Register" or "Public Key Load Balance Enable Register".

2. Ensure that there are no outstanding commands assigned to that module by reading the "Outstanding Command Counter 0 Register", "Outstanding Command Counter 1 Register", or "Outstanding Command Counter 3 Register".

3. Disable the module by clearing the appropriate module enable bit in the "Module Enable Register".

4. Disable the power to that module by clearing the appropriate module power enable bit in the "Power Enable Register".

The sequence to enable power to a XR9240 module is:

1. Enable the power by setting the appropriate module power enable bit in the "Power Enable Register".

2. Enable the module by setting the appropriate module enable bit in the "Module Enable Register".

3. Enable load balancing to the desired module by setting the appropriate bit in the "Transform Channel Load Balance Enable Register" or "Public Key Load Balance Enable Register".

5.10.2 Clock Gating

The sequence to disable a clock to a XR9240 module is:

1. Disable load balancing to the desired module by clearing the appropriate bit in the "Transform Channel Load Balance Enable Register" or "Public Key Load Balance Enable Register".

2. Ensure that there are no outstanding commands assigned to that module by reading the "Outstanding Command Counter 0 Register", "Outstanding Command Counter 1 Register", or "Outstanding Command Counter 3 Register".

3. Disable the module by clearing the appropriate module enable bit in the "Module Enable Register".

4. Disable the clock by clearing the appropriate module clock enable bit in the "Clock Enable Register".

The sequence to enable a clock to a XR9240 module is:

1. Enable the clock by setting the appropriate module clock enable bit in the "Clock Enable Register".

2. Enable the module by setting the appropriate module enable bit in the "Module Enable Register".

3. Enable load balancing to the desired module by setting the appropriate bit in the "Transform Channel Load Balance Enable Register" or "Public Key Load Balance Enable Register".

5.11 Serial Peripheral Interface

This section describes the XR9240 Serial Peripheral Interface (SPI) used to connect the XR9240 to a non-volatile device such as a Flash or EEPROM to store information such as the card version and serial numbers.

The host may access the devices connected to the SPI through the XR9240 SPI registers mapped to PCIe memory or as mapped memory through the PCIe Expansion ROM (see "Accessing an External Non-Volatile Device"). This section only describes the XR9240 Serial Peripheral Interface.

Figure 5-21 illustrates how the XR9240 may be connected to a non-volatile device.

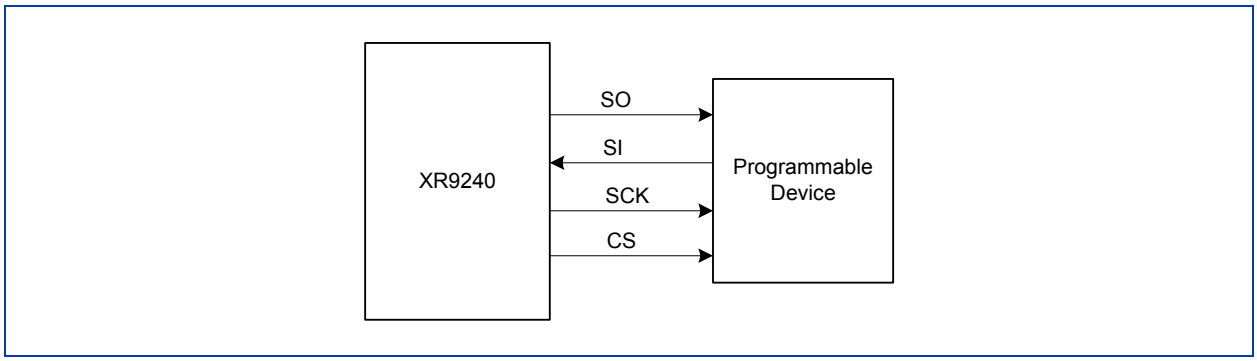


Figure 5-21. SPI Example Usage

The non-volatile device memory regions are illustrated in Figure 5-22.

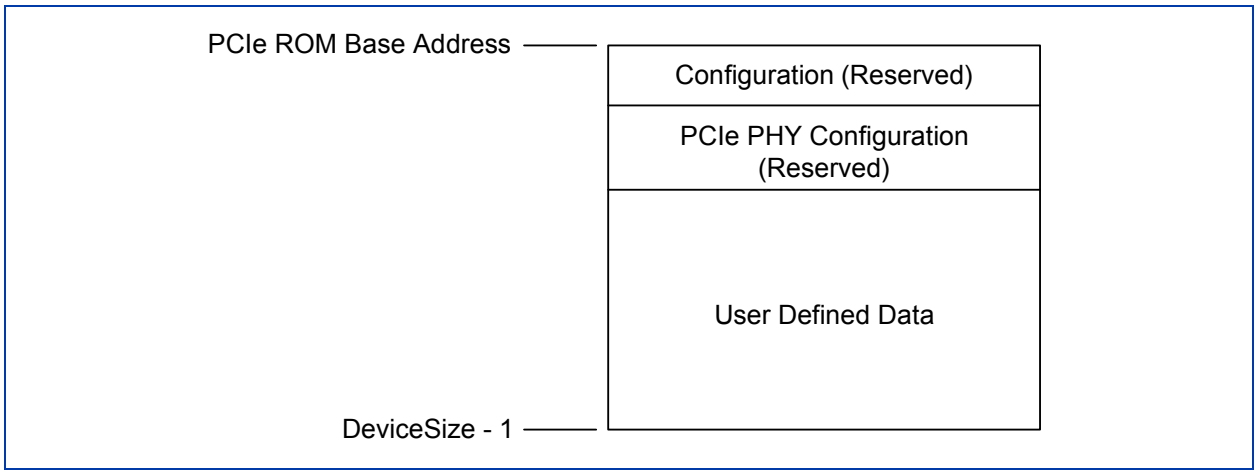


Figure 5-22. Non-volatile Device Memory Regions

The procedure below describes the steps to access a non-volatile device using the SPI.

1. Initialize the "SPI Command Configuration 0 Register", "SPI Command Configuration 1 Register" and "SPI User Defined Register".

2. For a write operation, write to the "SPI Data Register". Poll the "SPI Status Register" to confirm that the operation is complete.

3. For read operation, write to the "SPI Command Address Register". Poll the "SPI Status Register" to confirm that the operation is complete and then read the "SPI Data Register", or use the SPI interrupt event.

5.12 Temperature Sensor Controller

The XR9240 contains an internal temperature sensor that measures the die temperature using measured voltages from internal analog circuitry. These voltages are converted to a signal value through an ADC, allowing the values to be read by the host.

The XR9240 also contains a controller for the temperature sensor. The host may read/write directly to the XR9240 temperature sensor controller (TSC) registers. In contrast, the temperature sensor registers may be accessed by the host via the temperature sensor controller registers. Refer to "XR9240 Configuration Registers" for a complete description of the TSC registers.

To control the XR9240 temperature, the XR9240 supports maximum and minimum temperature thresholds. The maximum temperature threshold represents the temperature which, if exceeded, may cause the XR9240 device not to operate. The minimum temperature threshold represents the temperature at which the XR9240 device has returned to a safe operating temperature. Together, these thresholds provide hysteresis for responding to an overheated temperature condition. The host may configure the "Temperature Sensor Threshold Register" to specify the maximum and minimum temperature thresholds. The default maximum temperature threshold is +110°C. The default minimum temperature threshold is +90°C.

An interrupt will be asserted if the temperature rises higher than the maximum temperature threshold. Commands should not be sent to a XR9240 that is in an overheated state. After the XR9240 temperature returns to lower than the minimum temperature threshold, another interrupt will alert the host that the device is ready to accept commands. The host must read the "Temperature Sensor Data Register" in order to distinguish the temperature sensor event interrupts.

5.12.1 Calculating the Temperature Reading

If enabled, the temperature sensor will continuously sample the XR9240 die temperature. The value placed in the TS_DATA[11:0] field of the "Temperature Sensor Data Register" must be manipulated to determine the temperature in °C, as follows.

1. If bit[11] is 1, the temperature is below zero, otherwise, the temperature is above zero.
2. If the temperature is negative, calculate the two's complement of the value in the TS_DATA[10:0] field.
3. Divide the result by 4.

Example 1: TS_DATA[11:0] = 0x1B8

1. Bit 11 = 0, so temperature is positive
2. N/A
3. 0x1B8 = 440 decimal. 440/4 = 110. Final temperature is +110°C.

Example 2: TS_DATA[11:0] = 0xFD8

1. Bit 11 = 1, so temperature is negative
2. Two's complement of 0x7D8 = 0x028
3. 0x028 = 40 decimal. 40/4 = 10. Final temperature is -10°C.

5.12.2 Calculating the Temperature Thresholds

The maximum temperature threshold default value of 110°C and the minimum temperature threshold default value of 90°C will be suitable for most applications. The procedure to change the temperature threshold values is the reverse of what is described in "Calculating the Temperature Reading".

1. Multiply the temperature by 4.
2. If the temperature is below zero, set bit[11] to 1, otherwise, set bit[11] to 0.
3. If the temperature is negative, calculate the two's complement of the value in the TS_DATA[10:0] field.

Example 1: Set temperature high threshold to +110°C

1. 110*4 = 'd440 = 'h1B8
2. Temperature is positive, so set bit[11] to 0.
3. Write 0x1B8h to the TS_HI_THRESH[13:0] field in the "Temperature Sensor Threshold Register".

Example 2: Set temperature low threshold to -10°C (Note that a typical low threshold value would be +90°C. This example demonstrates the calculation for a negative temperature value.)

1. 10*4 = 'd40 = 'h28
2. Temperature is negative, so set bit[11] to 1.
3. Two's complement of 0x028 = 0xFD8
4. Write 0xFD8 to the TS_LO_THRESH[13:0] field in the "Temperature Sensor Threshold Register".

5.13 Fan Monitor

The XR9240 contains circuitry to monitor an external fan mounted on the PCB. Once enabled using the "Fan Monitor Enable Register" and the default settings are programmed into the XR9240 registers, the fan monitor circuit will send an interrupt to the host if the fan stops rotating. The fan speed may be read from the host through the fan monitoring registers.

5.13.1 Calculating the Fan Thresholds

The fan monitoring module allows for two thresholds to be set that control the number of fan error interrupts that are sent to the host.

The first threshold determines the fan speed at which the XR9240 will assert a fan error interrupt. If the fan speed drops below this threshold, the XR9240 will send an interrupt to the host. The fan speed threshold is configured in the "Fan Threshold Register" using the FAN_SPD_THRESH[15:0] field.

The equation used to calculate the fan speed thresholds is:

$$P = (60 * 250 * 10^6) / (2 * Q * 2^{16})$$

where:

P: is the register value to be written by the host for the fan speed threshold

Q: is the desired lowest fan speed threshold in rpm

For example, to configure a fan speed threshold of 5000 rpm, the equation would be:

$$P = (60*250*10^6)/(65536*5000*2) = 0x16$$

There will be some variance in the result due to the fixed point calculation, however the variance is less than 5%.

The second threshold determines the number of fan errors that can occur before the XR9240 will send a fan error interrupt to the host. The fan error threshold is configured in the "Fan Threshold Register" using the FAN_ERR_THRESH[15:0] field.

5.13.2 Calculating the Fan Speed

The "Fan Speed Register" reflects the real time fan speed. The "Fan Speed After Error Register" captures the fan speed when the fan error occurred.

The formula below must be used to calculate the fan speed in units of rpm for both fan speed register values.

$R = (60 * 250 * 10^6) / (C*2)$

Where:

R: Fan speed in rpm

C: Value from the "Fan Speed Register" or "Fan Speed After Error Register", converted to decimal

For example, if the register has a value of 0x160000, the decimal value is 1441792. The fan speed would be:

$R = (60*250*10^6)/C = (60*250*10^6)/(1441792*2) = 5202$ rpm

6 PCIe BAR0 Register Definition

This section describes the XRXR9240 ring control and sub-module registers. These registers are mapped into 64K bytes of PCIe BAR0 memory space that can be accessed through the PCIe bus. The registers are accessed using a 32-bit word. The absolute register address can be calculated using:

PCIe Address = PCIe BAR 0 Address + Offset

Figure 6-1 illustrates the PCIe BAR 0 memory space and lists the offset values. The first 16K is used for device configuration and control. The following 16K is used to access the internal XRXR9240 sub-module registers and is therefore only accessible by the physical function. The last 32K of the memory space is used for defining the ring parameters and is accessible by both physical and virtual functions.

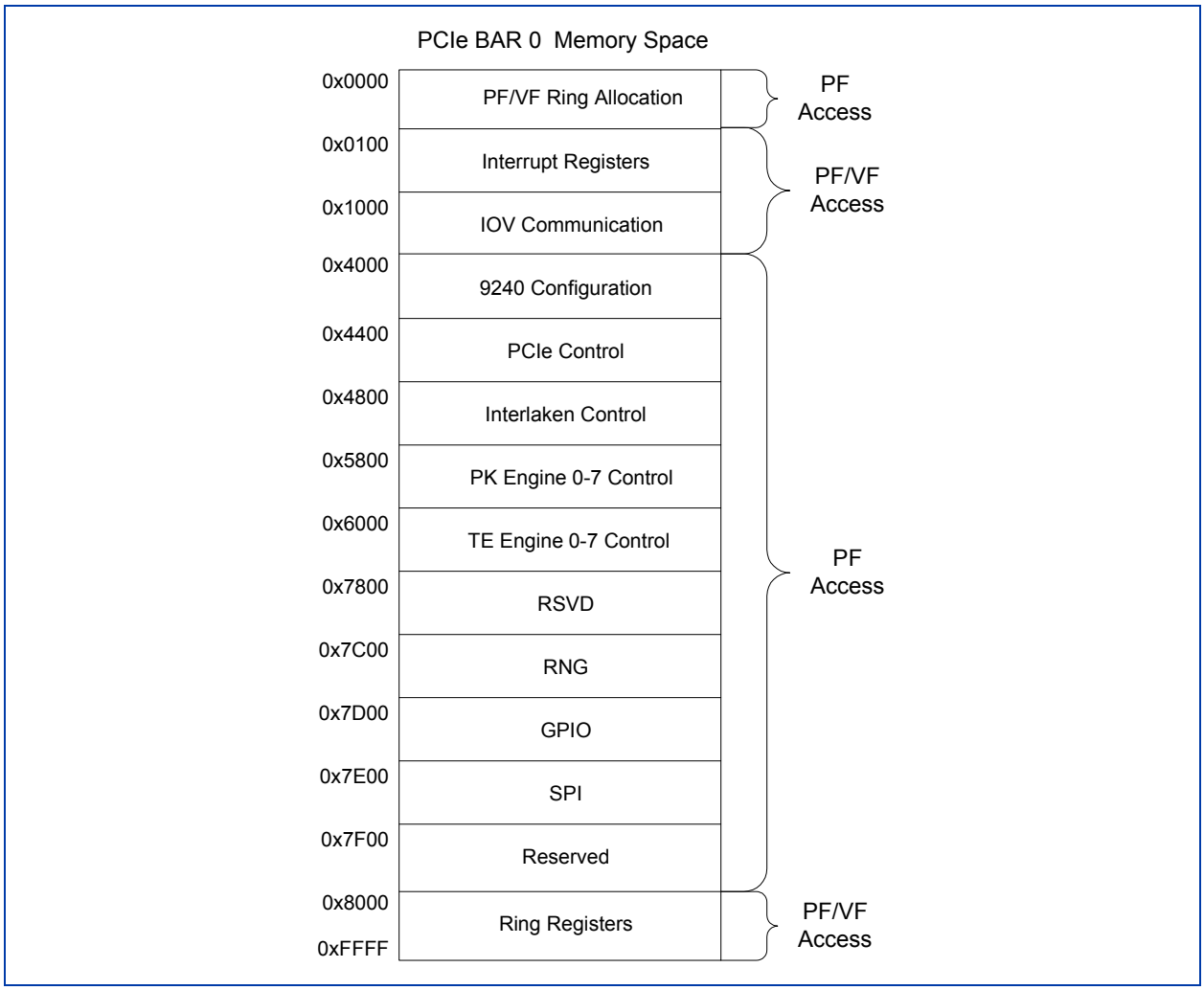


Figure 6-1. PCIe BAR 0 Memory Map

The physical function and each virtual function have their own unique BAR address.

The host should not read/write to/from reserved registers. If the host writes to a reserved register, the write will be ignored by the XRXR9240. Likewise, if the host reads a reserved register, the return value will be all zeros.

For the PCIe BAR3 and Expansion BAR memory space, please refer to Chapter 7.

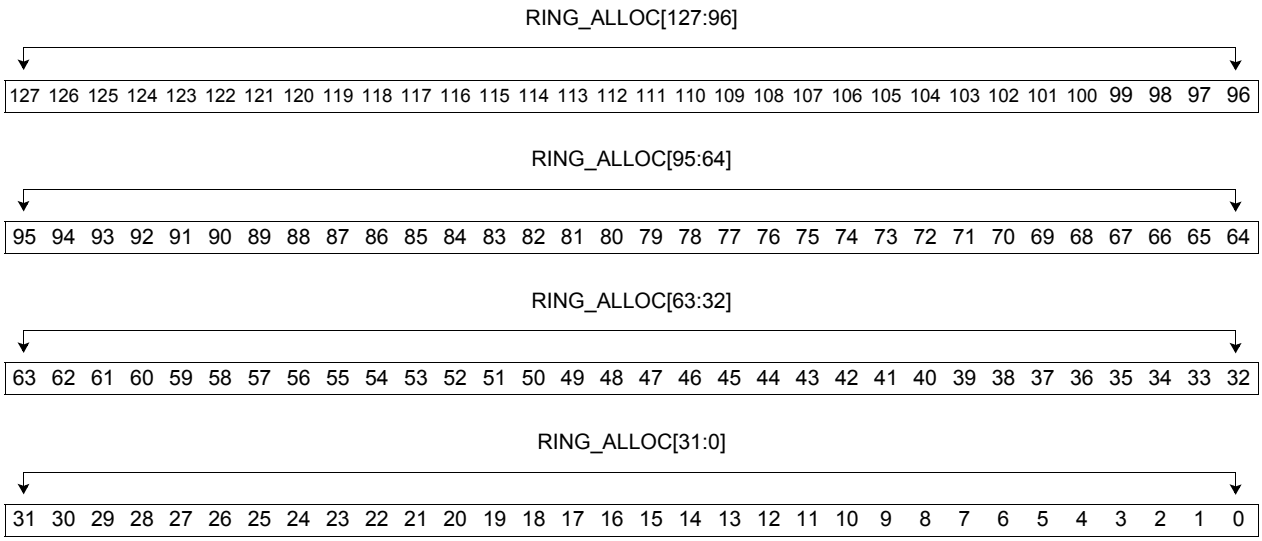
6.1 Ring Allocation Register

The Ring Allocation register allows the host to assign each of the XRXR9240's command/ result ring pairs to a physical or virtual function. The XR9240 supports one physical function that may have up to 128 rings and 128 virtual functions that each may have a single ring structure.

This register is only accessible by the physical function. Unused rings should be left unconfigured and will be allocated to the physical function by default.

Type: Read/Write

Offset 0x0000
0x0004
0x0008
0x000C



Field Name	Bits	Reset	Description
RING_ALLOC[127:0]	127:0	128'h0000 0000 0000 0000	Ring Allocation. This field represents the bitwise allocation of the rings. Note that bit 127 corresponds to ring 127 while bit 0 corresponds to ring 0. 0 Ring assigned to physical function (PF) 1 Ring assigned to virtual function (VF)

6.2 Interrupt Registers

The physical and virtual function interrupt registers share the same memory space address but are mapped into physically separate registers within the XR9240.

Although the physical function supports up to 128 rings, note that only 16 of those rings may assert an interrupt to the host, as identified by their RingID. The host may poll the remaining physical function rings.

6.2.1 PCIe Interrupt Source Register

The PCIe Interrupt Source register may be read by the interrupt handler software to determine the source of the interrupt.

The bits in this register may be set either by the XR9240 hardware or by the software via the "PCIe Software Interrupt Enable Register".

There are multiple ways to clear the bits in this register.

1. Writing a one to any bit in this register will clear the setting for that bit. Writing a zero has no effect.

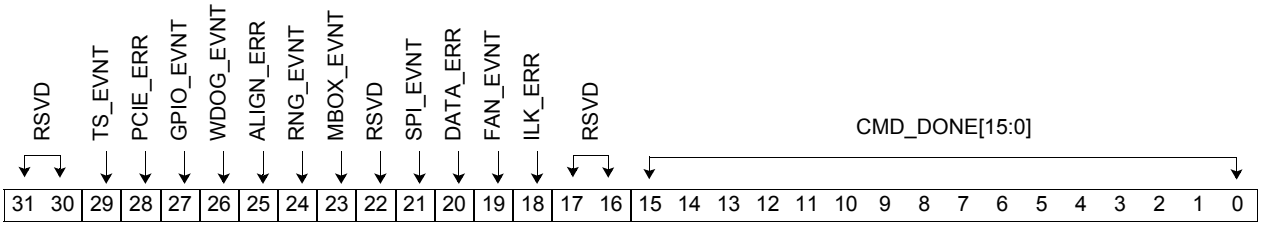
2. All bits in this register may be cleared on read if the CLR_ON_RD bit in the "PCIe Interrupt Configuration Register" is cleared. If CLR_ON_RD is set, then only bits [29:16] are cleared on read, which may be useful in MSI-X mode where the command complete interrupts are independent of the other interrupts in this register.

3. Bits in this register may be automatically cleared by the XR9240 after the interrupt has been sent to the host if the auto-clear feature is enabled in the "PCIe Interrupt Auto-Clear Register".

Type: Read/Write

Offset for Physical Function: 0x0100

Offset for Virtual Function: 0x0100



Field Name	Bits	Reset	Description
RSVD	31:30	1'b0	Reserved.

Field Name	Bits	Reset	Description
TS_EVNT	29	1'b0	Temperature Sensor Event. This bit will be set for a temperature sensor event, such as a XR9240 overheat error or to indicate that the XR9240 temperature has returned to within the allowable range. The host must read the "Temperature Sensor Data Register" to determine the cause of the event. 0 Interrupt not due to temperature sensor event 1 Interrupt due to temperature sensor event
PCIE_ERR	28	1'b0	PCIe Error. 0 Interrupt not due to PCIe error 1 Interrupt due to PCIe error
GPIO_EVNT	27	1'b0	GPIO Event. 0 Interrupt not due to GPIO 1 Interrupt due to GPIO
WDOG_EVNT	26	1'b0	Watchdog Timer Event. 0 Interrupt not due to watchdog timer expiration 1 Interrupt due to watchdog timer expiration
ALIGN_ERR	25	1'b0	Alignment Error. 0 Interrupt not due to alignment violation on a ring structure 1 Interrupt due to alignment violation on a ring structure
RNG_EVNT	24	1'b0	Random Number Generator Event. 0 Interrupt not due to RNG event 1 Interrupt due to RNG event
MBOX_EVNT	23	1'b0	PF/VF Mailbox Event. 0 Interrupt not due to PF/VF Mailbox event 1 Interrupt due to PF/VF Mailbox event
RSVD	22	1'b0	Reserved.
SPI_EVNT	21	1'b0	Serial Peripheral Interface Event. 0 Interrupt not due to SPI event 1 Interrupt due to SPI event
DATA_ERR	20	1'b0	Data Integrity Error. 0 Interrupt not due to data integrity error 1 Interrupt due to data integrity error
FAN_EVNT	19	1'b0	Fan Monitor Event. 0 Interrupt not due to fan event 1 Interrupt due to fan event
ILK_ERR	18	1'b0	Interlaken Interface Error. 0 Interrupt not due to ILK error 1 Interrupt due to ILK error
RSVD	17:16	2'b00	Reserved.

Field Name	Bits	Reset	Description
CMD_DONE[15:0]	15:0	16'h00	Command Done. This field is a bitwise static mapping of the command done bit per ring or group of rings. Only the first 16 rings of the physical function may interrupt the host. Each virtual function may only be assigned to one ring. 0x00Ring 0 : 0xFFRing 15

6.2.2 PCIe Software Interrupt Enable Register

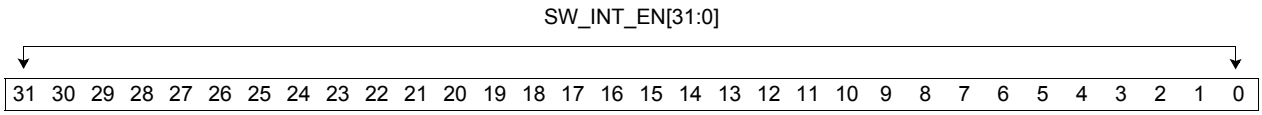
The PCIe Software Interrupt Enable Register allows software to initiate interrupts to the host. Setting any of the bits in this register to one will also set the corresponding bit in the "PCIe Interrupt Source Register" and cause an interrupt if enabled in the "PCIe Interrupt Mask Set Register".

Setting a bit in this register will generate either a fast interrupt or a throttled interrupt depending on the FI_EN bit setting in the "PCIe Interrupt Configuration Register". If the FI_EN bit is set, then setting a bit in this register will cause a fast interrupt. (Note that the FI_EN bit can only be set high when in auto-mask mode). If the FI_EN bit is cleared, then setting a bit in this register will cause an interrupt after the corresponding interrupt throttling timer expires.

Type: Read/Write

Offset for Physical Function: 0x0104

Offset for Virtual Function: 0x0104



Field Name	Bits	Reset	Description
SW_INT_EN[31:0]	31:0	32'h0000 0000	Software Interrupt Enable. This field represents the bitwise mechanism to set the corresponding bit in the "PCIe Interrupt Source Register" and generates an interrupt if enabled in the "PCIe Interrupt Mask Set Register". 0 No change to corresponding bit in "PCIe Interrupt Source Register" 1 Set corresponding bit in "PCIe Interrupt Source Register"

6.2.3 PCIe Interrupt Mask Set Register

The PCIe Interrupt Mask Set register may be used by the host to either allow or prevent interrupts identified in the "PCIe Interrupt Source Register" from being sent to the host. Note that the "PCIe Interrupt Source Register" will store and reflect the interrupt events regardless of the state of this register.

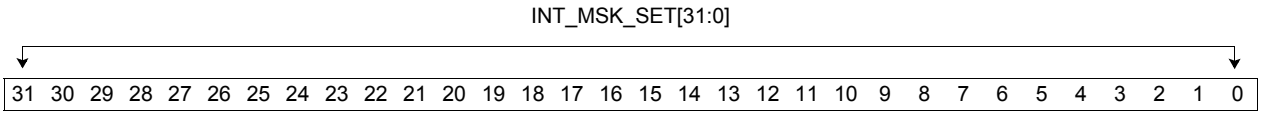
The bits in this register correspond one-to-one to the bits in the "PCIe Interrupt Source Register". Writing a value of one will allow the interrupt to be sent to the host; writing a value of zero has no effect. The host may read this register to determine which interrupts are masked.

The bits in this register may be cleared by writing to the "PCIe Interrupt Mask Clear Register". This mechanism for independently setting and clearing bits allows for simple programming in multi-threaded, multi-CPU core systems by preventing the need for a read/ modify/write operation.

Type: Read/Write

Offset for Physical Function: 0x0108

Offset for Virtual Function: 0x0108



Field Name	Bits	Reset	Description
INT_MSK_SET[31:0]	31:0	32'h0000 0000	Interrupt Mask Set. This field represents the mechanism to mask the corresponding bit in the "PCIe Interrupt Source Register". The setting for each bit are: 0 No change 1 Allow corresponding bit in "PCIe Interrupt Source Register" to be send to the host

6.2.4 PCIe Interrupt Mask Clear Register

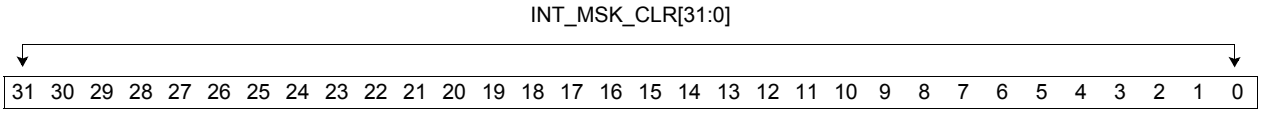
The PCIe Interrupt Mask Clear register may be used by the host to clear the interrupt masks defined in the "PCIe Interrupt Mask Set Register".

The bits in this register correspond one-to-one to the bits in the "PCIe Interrupt Mask Set Register". Writing a value of one will clear the interrupt mask; writing a value of zero has no effect. Reading this register does not return any meaningful information.

Type: Read/Write

Offset for Physical Function: 0x010C

Offset for Virtual Function: 0x010C



Field Name	Bits	Reset	Description
INT_MSK_CLR[31:0]	31:0	32'h0000 0000	Interrupt Mask Clear. This field is used to clear the interrupt mask for the corresponding bit in the "PCIe Interrupt Source Register". The setting for each bit are: 0 No change to corresponding bit in "PCIe Interrupt Mask Set Register" 1 Clear corresponding bit in "PCIe Interrupt Mask Set Register"

6.2.5 PCIe Interrupt Auto-Clear Register

The PCIe Interrupt Auto-Clear register may be used by the host to automatically clear the interrupts identified in the "PCIe Interrupt Source Register". When a bit is set in this register, the corresponding interrupt bit in the "PCIe Interrupt Source Register" will be automatically cleared after the interrupt is sent to the host.

Auto-clear may be useful for command done interrupts that have dedicated MSI-X vectors. When the cmd_dnX interrupt is shared with other interrupt sources, the relevant bits in this register should not be set.

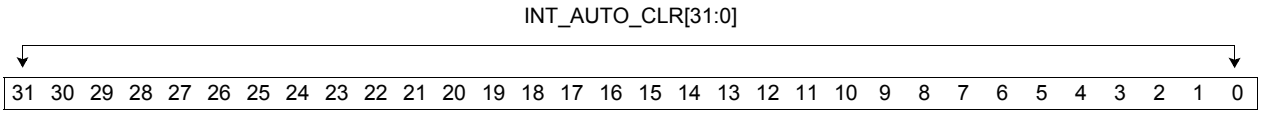
Caution

Bits [29:16] of this register should never be set because they share the same MSI-X vector.

Type: Read/Write

Offset for Physical Function: 0x0110

Offset for Virtual Function: 0x0110



Field Name	Bits	Reset	Description
INT_AUTO_CLR [31:0]	31:0	32'h0000 0000	Interrupt Auto-Clear. This field represents the mechanism to automatically clear the interrupts defined in the "PCIe Interrupt Source Register". The setting for each bit are: 0 No change to corresponding bit in "PCIe Interrupt Source Register" 1 Auto-clear interrupt bit in "PCIe Interrupt Source Register"

6.2.6 PCIe Interrupt Auto-Mask Register

The PCIe Interrupt Auto-Mask register may be used by the host to automatically mask the interrupts defined in the "PCIe Interrupt Source Register". If any of the Auto-Mask enable bits are set, the IAME bit in the "PCIe Interrupt Configuration Register" must be set as well.

Each bit in this register enables auto clearing and auto setting of the corresponding bit in the "PCIe Interrupt Mask Set Register" as follows:

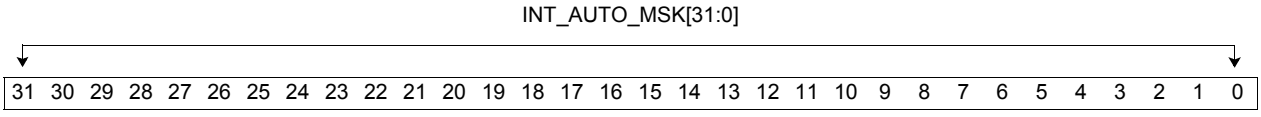
- Following a write of 1'b1 to any bit in the "PCIe Software Interrupt Enable Register", the corresponding bit in the "PCIe Interrupt Mask Set Register" will be automatically set to enable that interrupt.
- A write of 1'b1 to any bit in the "PCIe Interrupt Source Register" will clear the corresponding bit in the "PCIe Interrupt Mask Set Register", thus masking further interrupts for that source.
- A read to clear the "PCIe Interrupt Source Register" will clear the "PCIe Interrupt Mask Set Register" bits (enabled by the "PCIe Interrupt Auto-Mask Register") thus masking further interrupts. Note that if the CLR_ON_RD bit in the "PCIe Interrupt Configuration Register" is set, transmit and receive interrupt sources (bits [15:0]) will not be cleared on read. In this case, bits [15:0] in the "PCIe Interrupt Mask Set Register" are not cleared.
- In MSI-X mode the, autoclear feature can be driven by MSI-X vector assertion if the IAME bit in the "PCIe Interrupt Configuration Register" is set.

Bits [29:16] of this register should never be set because they share the same MSI-X vector.

Type: Read/Write

Offset for Physical Function: 0x0114

Offset for Virtual Function: 0x0114



Field Name	Bits	Reset	Description
INT_AUTO_MSK [31:0]	31:0	32'h0000 0000	Interrupt Auto-Mask. This field represents the mechanism to automatically mask the interrupts defined in the "PCIe Interrupt Source Register". The setting for each bit are: 0 Unmask the interrupt bit in "PCIe Interrupt Source Register" 1 Mask the interrupt bit in "PCIe Interrupt Source Register"

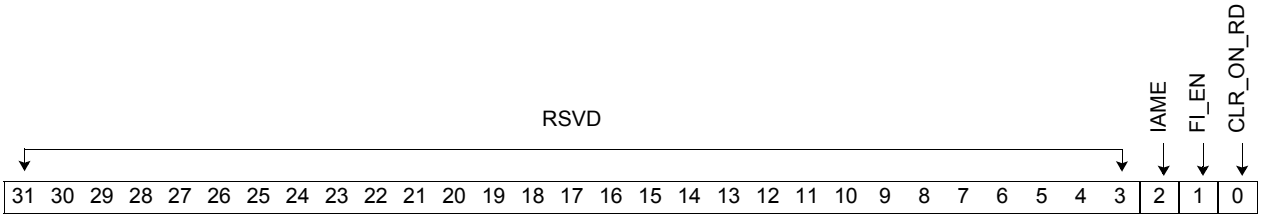
6.2.7 PCIe Interrupt Configuration Register

The PCIe Interrupt Configuration register provides the host the ability to control extended and fast interrupt generation, and the clearing of the "PCIe Interrupt Source Register".

Type: Read/Write

Offset for Physical Function: 0x0118

Offset for Virtual Function: 0x0118



Field Name	Bits	Reset	Description
RSVD	31:3	1'h0000 0000	Reserved.
IAME	2	1'b0	Interrupt Auto Mask Enable. This bit must only be set when using interrupts in MSI-X mode. 0 Bits in the "PCIe Interrupt Mask Set Register" may not be auto-cleared (depending on "PCIe Interrupt Auto-Mask Register" setting) upon interrupt assertion 1 Bits in the "PCIe Interrupt Mask Set Register" may be auto-cleared (depending on "PCIe Interrupt Auto-Mask Register" setting) upon interrupt assertion
FI_EN	1	1'b0	Fast Interrupt Enable. 0 When a bit is set in the "PCIe Software Interrupt Enable Register", an interrupt will be generated after the throttle timer expires (see "PCIe Interrupt Throttle Register") 1 When a bit is set in the "PCIe Software Interrupt Enable Register", a fast interrupt will be generated
CLR_ON_RD	0	1'b0	Clear on Read. 0 Only bits [29:16] of the "PCIe Interrupt Source Register" are cleared when read 1 Entire "PCIe Interrupt Source Register" is cleared when read

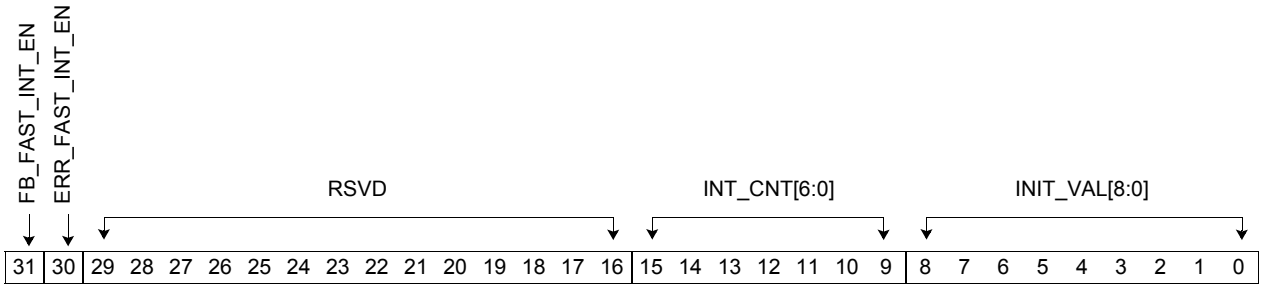
6.2.8 PCIe Interrupt Throttle Register

The PCIe Interrupt Throttle Register is used to set the interrupt throttle timer initial value.

Type: Read/Write

Offset for Physical Function: 0x011C - 0x0158 Each VF entry is 4 bytes and can be located at 0x0400 + 4*N, where N = VF number (0..127)

Offset for Virtual Function: 0x011C



Field Name	Bits	Reset	Description
FB_FAST_INT_EN	31	1'b0	Free Buffer Command Completion Fast Interrupt Enable. 0 Disable fast interrupt for command completion when this bit is set 1 Enable fast interrupt for command completion when this bit is set
ERR_FAST_INT_EN	30	1'b0	Command Completion with Error Fast Interrupt Enable. 0 Disable fast interrupt for command completion when error bit is set 1 Enable fast interrupt for command completion when error bit is set
RSVD	29:16	1'b0	Reserved.
INT_CNT[6:0]	15:9	7'b000 0000	Interrupt Counter. This field represents the seven most significant bits of the 9-bit interrupt counter. The interrupt counter counts down in 1 ms intervals from the initial value INIT_VAL[8:0] loaded each time an associated interrupt is asserted. When the interrupt counter reaches zero, the XR9240 triggers an interrupt.
INIT_VAL[8:0]	8:0	9'h000	Throttle Timer Initial Value. If set to zero, the interrupt counter is disabled. The range for this field is 2 µs to 1024 µs in increments of 2 µs.

6.3 IOV Communication Registers

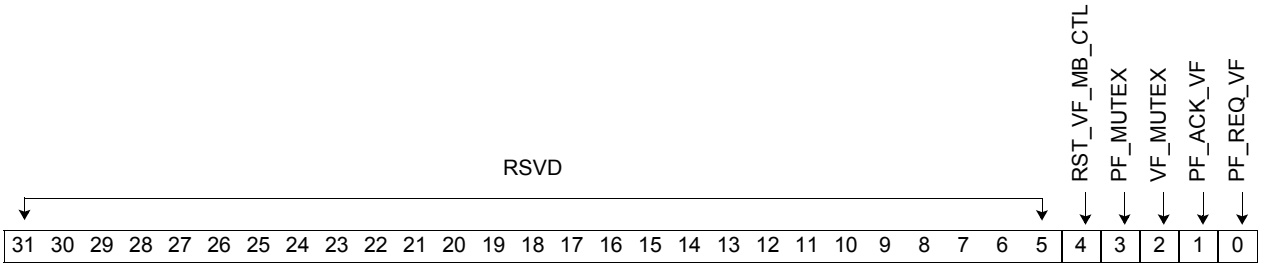
The Input/Output Virtualization registers allow the physical and virtual function drivers to communicate. See "Virtual Function to Physical Function Communication" for more information.

6.3.1 Physical Function Mailbox Control Register

The Physical Function Mailbox Control register provides the XR9240 physical function control of the mailbox communication between the host and the XR9240. Virtual functions do not have direct access to this register.

The VF_MUTEX bit in this register is a copy of the VF_MUTEX bit in the "Virtual Function Mailbox Control Register".

Type:	Read/Write	PF_MUTEX
	Read only	VF_MUTEX
	Write only	RST_VF_MB_CTL, PF_ACK_VF, PF_REQ_VF
Offset for Physical Function:	0x1000 - 0x11FF	4 bytes are allocated for each VF at address 0x0800 + 4*N, where N = 0...127.



Field Name	Bits	Reset	Description
RSVD	31:5	27'h0000000	Reserved.
RST_VF_MB_CTL	4	1'b0	Reset Virtual Function Mailbox Control. (Write only) NOTE: This bit should only be set if the VF driver is not operational. Setting this bit clears the VF_MUTEX bit in the appropriate "Virtual Function Mailbox Control Register" and also resets the appropriate bits in the REQ[15:0] and ACK[15:0] fields of the "Mailbox Interrupt Source Register". After being set, this bit will self clear and will always return zero when read. The XR9240 will be able to determine the VF that should be reset based on the address used by the PF driver to access this register.

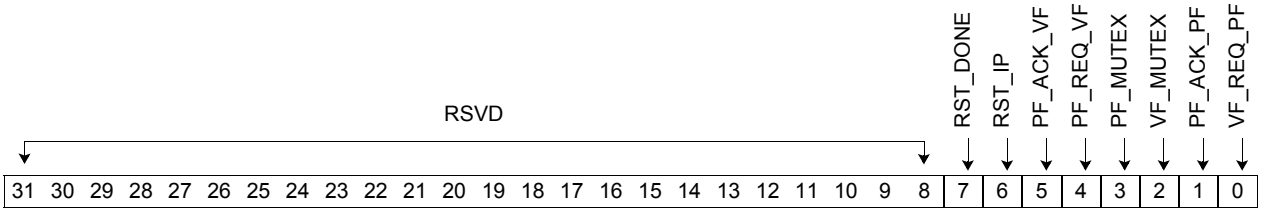
Field Name	Bits	Reset	Description
PF_MUTEX	3	1'b0	Physical Function Mutex. This bit is set by the PF driver to indicate that the PF is using the mailbox. This bit can be set only if the VF_MUTEX bit in this register is cleared. The XR9240 will automatically copy this bit to the PF_MUTEX bit in the "Virtual Function Mailbox Control Register". This bit is cleared by the PF driver writing a zero to this bit. 0 Mailbox is not being used by PF 1 Mailbox is being used by PF
VF_MUTEX	2	1'b0	Virtual Function Mutex. (Read only) This bit is a read only copy of the by the VF_MUTEX bit in the "Virtual Function Mailbox Control Register". 0 Mailbox is not being used by VF 1 Mailbox is being used by VF
PF_ACK_VF	1	1'b0	Physical Function Acknowledge Virtual Function Message. (Write only) The PF driver will set this bit to inform the VF that the message from the VF was read. The XR9240 will automatically copy this bit to the PF_ACK_VF bit in the "Virtual Function Mailbox Control Register". After being set, this bit will self clear and will always return zero when read.
PF_REQ_VF	0	1'b0	Physical Function Request to Virtual Function. (Write only) The PF driver will set this bit to inform the VF that a message has been placed in the VF mailbox. The XR9240 will automatically copy this bit to the PF_REQ_VF bit in the "Virtual Function Mailbox Control Register". After being set, this bit will self clear and will always return zero when read.

6.3.2 Virtual Function Mailbox Control Register

The Virtual Function Mailbox Control register provides the XR9240 virtual function control of the mailbox communication between the host and the XR9240. This register may only be accessed by the virtual functions.

There are 128 Virtual Function Mailbox Control registers; one for each VF. The offset for this register is the same for all VFs, however, the BAR address for each VF is unique, therefore each VF has its own Virtual Function Mailbox Control register.

Type:	Read/Write	VF_MUTEX
	Write only	VF_REQ_PF, VF_ACK_PF
	Read Only	RST_IP, PF_MUTEX
	Clear on Read	RST_DONE, PF_ACK_VF, PF_REQ_VF
Offset for Virtual Function:	0x1000	



Field Name	Bits	Reset	Description
RSVD	31:8	24'h0000 0	Reserved.
RST_DONE	7	1'b0	VF Mailbox Control Reset Done. This bit is set in response to the PF setting the RST_VF_MB_CTL bit for a particular VF and is only set when all bits in the VF Mailbox control and interrupt source register are cleared. This bit is cleared when read by the VF driver. 0 Reset of VF Mailbox Control not complete 1 Reset of VF Mailbox Control complete
RST_IP	6	1'b0	VF Mailbox Control Reset In Progress. (Read only) This read-only bit is in response to the PF setting the RST_VF_MB_CTL bit for a particular VF. 0 Reset of VF Mailbox Control not in progress 1 Reset of VF Mailbox Control in progress

Field Name	Bits	Reset	Description
PF_ACK_VF	5	1'b0	Physical Function Acknowledge Virtual Function Message. This bit is a copy of the PF_ACK_VF bit in the "Physical Function Mailbox Control Register". This bit is cleared after being read. 0 PF did not read message from VF 1 PF read message from VF; interrupt acknowledge will be set in the relevant "Mailbox Interrupt Source Register"
PF_REQ_VF	4	1'b0	Physical Function Request to Virtual Function. This bit is a read only copy of the PF_REQ_VF bit in the "Physical Function Mailbox Control Register". This bit is cleared after being read. 0 PF did not write message to VF mailbox 1 PF wrote message to VF mailbox; interrupt request will be set in the relevant "Mailbox Interrupt Source Register"
PF_MUTEX	3	1'b0	Physical Function Mutex. (Read only) This bit is a read only copy of the PF_MUTEX bit in the "Physical Function Mailbox Control Register". 0 Mailbox is not being used by PF 1 Mailbox is being used by PF
VF_MUTEX	2	1'b0	Virtual Function Mutex. This bit is set by the VF driver to indicate that the VF is using the mailbox. This bit can be set only if the PF_MUTEX bit of this register is cleared. The XR9240 will automatically copy this bit to the VF_MUTEX bit in the "Physical Function Mailbox Control Register". This bit may be cleared either by the VF driver writing a one to this bit, or after the PF resets the VF Mailbox control by asserting the RST_VF_MB_CTL bit in the "Physical Function Mailbox Control Register". 0 Mailbox is not being used by VF 1 Mailbox is being used by VF
VF_ACK_PF	1	1'b0	Virtual Function Acknowledge Physical Function Message. (Write only) The VF driver will set this bit to inform the PF that the message from the PF was read. Setting this bit will automatically set the appropriate bit in the ACK[15:0] field of the "Mailbox Interrupt Source Register". After being set, this bit will be cleared after one clock cycle and will always return zero when read. 0 VF did not read message from PF 1 VF read message from PF; interrupt acknowledge will be sent to the PF

Field Name	Bits	Reset	Description
VF_REQ_PF	0	1'b0	Virtual Function Request to Physical Function. (Write only) The VF driver will set this bit to inform the PF that a message has been placed in its mailbox. Setting this bit will automatically set the appropriate bit in the REQ[15:0] field of the "Mailbox Interrupt Source Register". After being set, this bit will be cleared after one clock cycle and will always return zero when read. 0 VF did not write message to PF mailbox 1 VF wrote message to PF mailbox; interrupt request will be sent to PF

6.3.3 Mailbox Interrupt Source Register

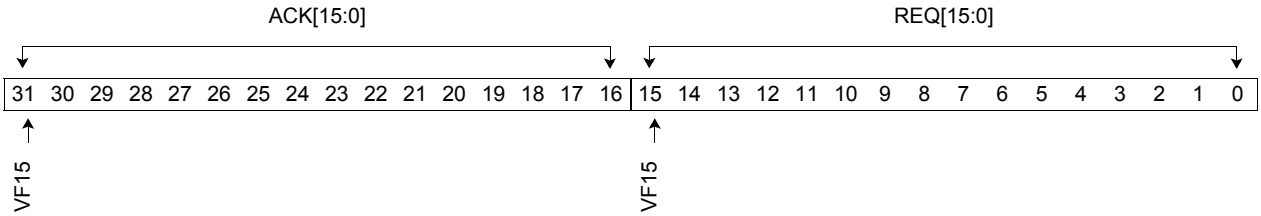
The Mailbox Interrupt Source register is accessible only by the physical function. A virtual function will indirectly set bits in this register according to the mechanisms described in "Virtual Function to Physical Function Communication". Functions may interrupt each other to request mailbox access, or to acknowledge that a mailbox message was read.

The fields in this register may be cleared by the physical function when it asserts the RST_VF_MB_CTL bit for a particular virtual function. The status of the reset may be monitored by the host by reading the "Reset In Progress" bit (RST_IP) or the "Reset Done" bit (RST_DONE) in the "Virtual Function Mailbox Control Register".

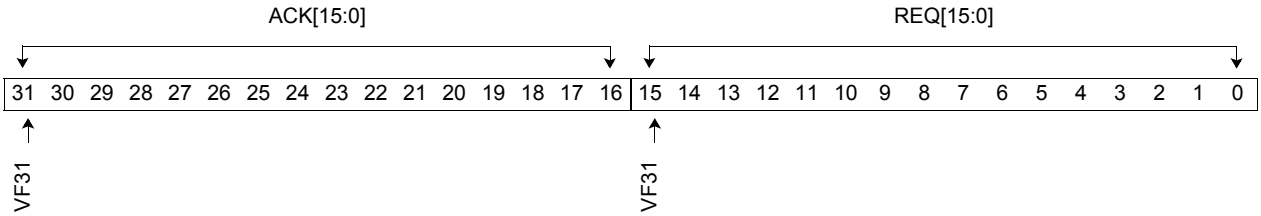
Type: RW1C

Offset for Physical Function: 0x1400 - 0x141F

Address Offset = 0x1400, N = 0

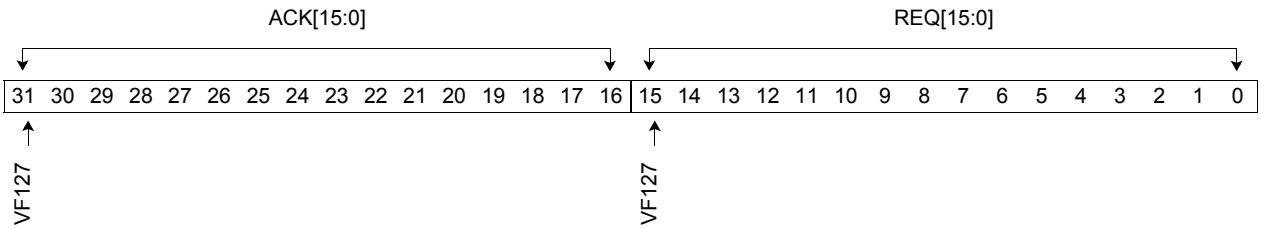


Address Offset = 0x1404, N = 1



|

Address Offset = 0x141F, N = 7



Field Name	Bits	Reset	Description
ACK[15:0]	15:0	16'h0000	Acknowledge. A bit in the ACK[15:0] field will be set when function number (16*N+J) acknowledges a message from the mailbox. Where: 'N' is the register index, N=0...7 'J' is the bit position with the field, J=0...15 '16+J' is the bit index in the field ACK[15:0].
REQ[15:0]	15:0	016'h0000	Request. A bit in the REQ[15:0] field will be set when function number (16*N+J) writes a message to its mailbox. Where: 'N' is the register index, N=0...7 'J' is the bit position with the field, REQ[15:0].

6.3.4 Mailbox Data Register

This register contains the data entries for 128 mailboxes. Each entry is 16B and may be accessed by the physical function or the virtual function using logically unique address offsets.

Type: Read/Write

Offset for Physical Function: 0x2000 - 0x27FF

Offset for Virtual Function: 0x2000 - 0x200C

Field Name	Bits	Reset	Description
MB_DATA[127:0]	127:0	128'h0000 0000	Mailbox Data.

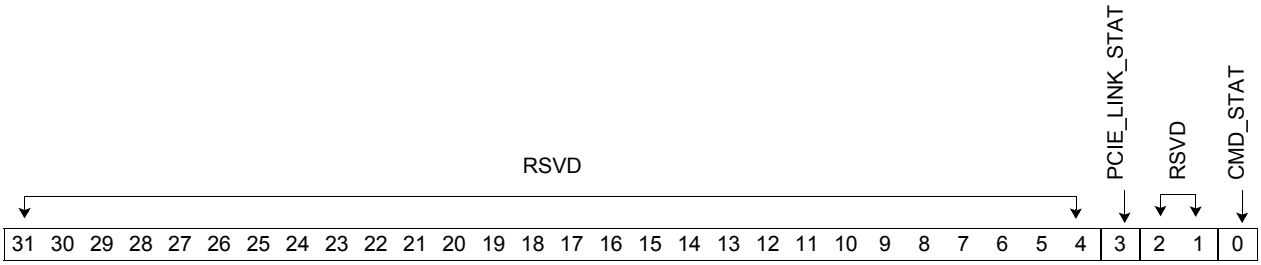
6.4 XR9240 Configuration Registers

6.4.1 Status Register

The Status register provides the XR9240 internal status to host. These signals may be connected to LEDs as status indicators.

Type: Read only

Offset 0x4000



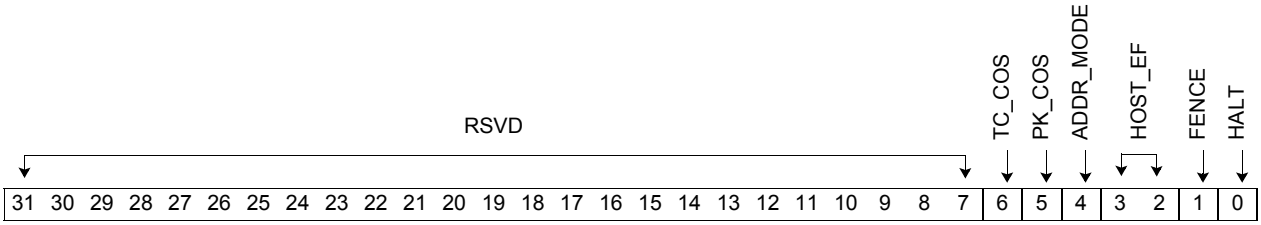
Field Name	Bits	Reset	Description
RSVD	31:4	28'h00 00000	Reserved.
PCIE_LINK_STAT	3	1'b0	PCIe Link Status. 0 PCIe link is down and not operational 1 PCIe link is up and operational
RSVD	2:1	2'b00	Reserved.
CMD_STAT	0	1'b0	Command Status. 0 XR9240 has no prefetched commands 1 XR9240 busy

6.4.2 XR9240 Control Register

The XR9240 Control register is used to set some fundamental features of the device.

Type: Read/Write

Offset 0x4004



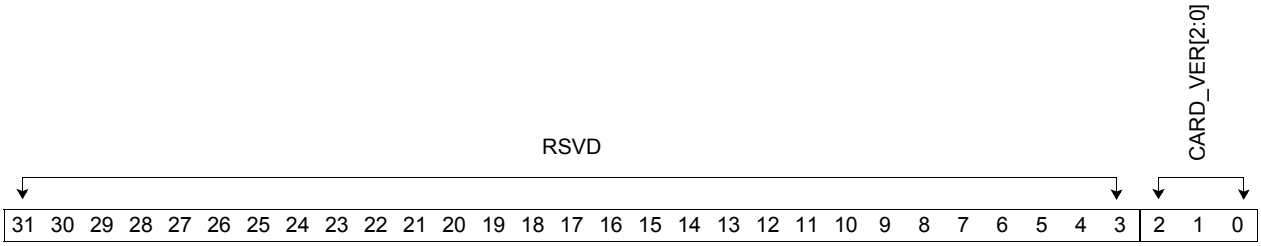
Field Name	Bits	Reset	Description
RSVD	31:7	25'h0000 00	Reserved.
TC_COS	6	1'b0	Transform Channel Class of Service Control. 0 Disable transform channel COS 1 Enable transform channel COS
PK_COS	5	1'b0	Public Key Class of Service Control. 0 Disable public key COS 1 Enable public key COS
ADDR_MODE	4	1'b1	DMA Addressing Mode. 0 32-bit addressing mode 1 64-bit addressing mode
HOST_EF	3:2	2'b00	Host Endian Format. Sets the endian format of the host command pointer ring, command structure, result ring and free pool ring. 00 No swap 01 Byte swap in word 10 Word swap, no byte swap in word 11 Word swap and byte swap in word
FENCE	1	1'b0	Fence Enable. 0 XR9240 will continue processing if an integrity error is detected in the PCIe core 1 XR9240 will reset itself if an integrity error is detected in the PCIe core
HALT	0	1'b0	Halt on Integrity Error Enable. 0 XR9240 will continue processing if an integrity error is detected 1 XR9240 will stop processing if an integrity error is detected

6.4.3 Card Version Register

The Card Version register allows the host to read the version number of the card. The card version is read from pins BOARD_VER[2:0] that may be pulled high or low on the card.

Type: Read only

Offset 0x4008



Field Name	Bits	Reset	Description
RSVD	31:3	29'h0000 0000	Reserved.
CARD_VER[2:0]	2:0	value from BOARD_ VER[2:0] pins	Card Version.

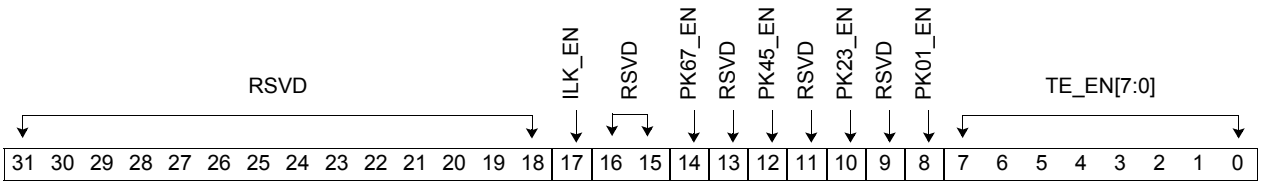
6.4.4 Module Enable Register

The Module Enable register provides the host the ability to enable or disable individual modules within the XR9240. The host must disable a module before that module is powered down using the "Power Enable Register", or clock gated using the "Clock Enable Register". A disabled module will have all its interface signals in an idle state. By default, all modules will be disabled after reset.

Note that the PK engines are enabled/disabled in power domain pairs rather than individually.

Type: Read/Write

Offset 0x403C



Field Name	Bits	Reset	Description
RSVD	31:18	14'h00	Reserved.
ILK_EN	17	1'b0	Interlaken Interface Module Enable. 0 Disable the Interlaken module 1 Enable the Interlaken module
RSVD	16:15	2'b00	Reserved.
PK67_EN	14	1'b0	Public Key 7/6 Power Domain Enable. 0 Disable the PK 7/6 Power Domain 1 Enable the PK 7/6 Power Domain
RSVD	13	1'b0	Reserved.
PK45_EN	12	1'b0	Public Key 5/4 Power Domain Enable. 0 Disable the PK5/4 Power Domain 1 Enable thePK5/4 Power Domain
RSVD	11	1'b0	Reserved.
PK23_EN	10	1'b0	Public Key 3/2 Power Domain Enable. 0 Disable the PK3/2 Power Domain 1 Enable the PK5/4 Power Domain
RSVD	9	1'b0	Reserved.
PK01_EN	8	1'b0	Public Key 1/0 Power Domain Enable. 0 Disable the PK1/0 Power Domain 1 Enable the PK1/0 Power Domain

Field Name	Bits	Reset	Description
TE_EN[7:0]	7:0	8'h00	Transform Engine Module Enable. Each bit of this field represents one of the eight Transform Engines. bit 0TE0 ---- bit 7TE7 The behavior for each bit is: 0 Disable TEn 1 Enable TEn

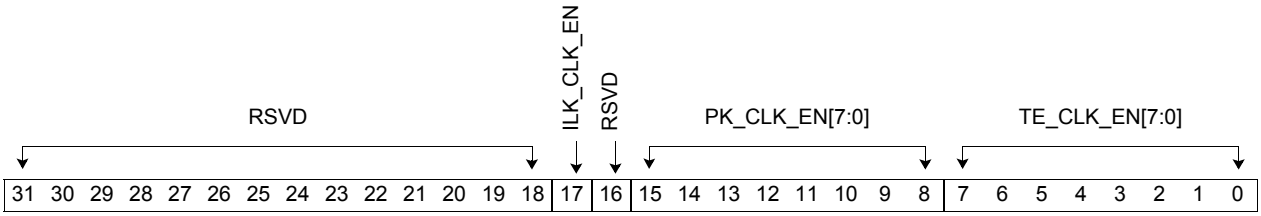
6.4.5 Clock Enable Register

The Clock Enable register provides the host software the ability to disable the clocks to unused engines to conserve power. The clocks to all engines are enabled by default. Commands must not be submitted to an engine whose clock is disabled. A module must be disabled using the "Module Enable Register" before its clock is disabled.

Note that the clocks to the PK engines are enabled/disabled in pairs rather than individually. By default, the clocks to the PK engines are enabled. Applications that do not use the PK engines may request XR9240 devices that have the PK clocks permanently disabled in order to conserve power.

Type: Read/Write

Offset 0x4040



Field Name	Bits	Reset	Description
RSVD	31:18	14'h00	Reserved.
ILK_CLK_EN	17	1'b0	Interlaken Interface Clock Enable. Note: this bit has no effect on the ILK PHY. 0 Disable clock for the Interlaken interface 1 Enable clock for the Interlaken interface
RSVD	16	1'b0	Reserved.
PK_CLK_EN[7:0]	15:8	8'hFF	Public Key Clock Enable. The clocks to the PK engines are grouped into four power domains. Each bit in this field represents one of the four PK power domains. Bit 0PK 0/1 power domain Bit 2PK 2/3 power domain Bit 4PK 4/5 power domain Bit 8PK 6/7 power domain All other values are reserved. The behavior for each bit is: 0 Disable clock for PKn 1 Enable clock for PKn

Field Name	Bits	Reset	Description
TE_CLK_EN[7:0]	7:0	8'hFF	Transform Engine Clock Enable. Each bit of this field represents one of the eight Transform Engines. bit 0TE0 ---- bit 7TE7 The behavior for each bit is: 0 Disable clock for TEn 1 Enable clock for TEn

6.4.6 Soft Reset Register

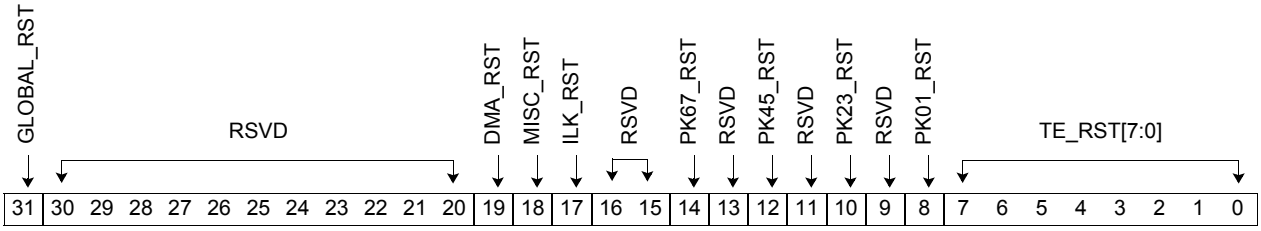
The Soft Reset register provides the host the capability to reset the XR9240 during error recovery. The Soft Reset register will be cleared by the XR9240 after the reset is complete. After a soft reset, the XR9240 will wait 64 clock cycles (256 ns) to ensure that all modules are reset.

Note: All soft resets may only be asserted when the CMD_STAT bit in the "Status Register" is set to zero.

The PK engines are reset in pairs rather than individually.

Type: Write 1 to clear

Offset 0x4044



Field Name	Bits	Reset	Description
GLOBAL_RST	31	1'b0	Global Software Reset. 0 Do not reset the entire XR9240 1 Reset the entire XR9240 except PCIe Interface and PHY
RSVD	30:20	11'h00	Reserved.
DMA_RST	19	1'b0	PCIe DMA Soft Reset. 0 Do not reset the PCIe DMA 1 Reset the PCIe DMA
MISC_RST	18	1'b0	Miscellaneous Soft Reset. 0 Do not reset the miscellaneous modules 1 Reset the GPIO, SPI, and RNG modules
ILK_RST	17	1'b0	Interlaken Interface Soft Reset. 0 Do not reset the Interlaken interface 1 Reset the Interlaken interface
RSVD	16:15	2'b00	Reserved.
PK67_RST	14	1'b0	Public Key 7/6 Soft Reset. 0 Do not reset the PK 7/6 1 Reset the PK 7/6
RSVD	13	1'b0	Reserved.

Field Name	Bits	Reset	Description
PK45_RST	12	1'b0	Public Key 5/4 Soft Reset. 0 Do not reset the PK 5/4 1 Reset the PK 5/4
RSVD	11	1'b0	Reserved.
PK23_RST	10	1'b0	Public Key 3/2 Soft Reset. 0 Do not reset the PK 3/2 1 Reset the PK 3/2
RSVD	9	1'b0	Reserved.
PK01_RST	8	1'b0	Public Key 1/0 Soft Reset. 0 Do not reset the PK 1/0 1 Reset the P K1/0
TE_RST[7:0]	7:0	8'h00	Transform Engine Soft Reset. Each bit of this field represents one of the eight Transform Engines. bit 0TE0 ---- bit 7TE7 The behavior for each bit is: 0 Do not reset the TEn 1 Reset the TEn

6.4.7 Power Enable Register

The Power Enable register is used to selectively enable or disable the power to specific XR9240 modules. A module must be disabled using the "Module Enable Register" before its power is disabled.

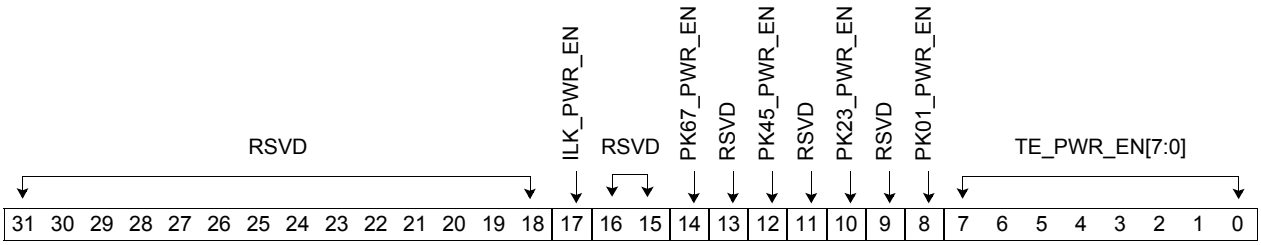
Enabling or disabling power to a module will take approximately TBD microseconds. The host should poll "Power Status Register" to confirm that the module is in the expected power state.

The default values of this register will depend of the XR9240 device type.

Note that the power to the PK engines is enabled/disabled in pairs rather than individually.

Type: Read/Write

Offset 0x4048



Field Name	Bits	Reset	Description
RSVD	31:18	Set by OTP	Reserved.
ILK_PWR_EN	17	Set by OTP	Interlaken Power Enable. 0 Disable power to the Interlaken module 1 Enable power to the Interlaken module
RSVD	16:15	Set by OTP	Reserved.
PK67_PWR_EN	14	Set by OTP	Public Key 7/6 Power Enable. 0 Disable power to the PK 7/6 engines 1 Enable power to the PK 7/6 engines
RSVD	13	Set by OTP	Reserved.
PK45_PWR_EN	12	Set by OTP	Public Key 5/4 Power Enable. 0 Disable power to the PK 5/4 engines 1 Enable power to the PK 5/4 engines
RSVD	11	Set by OTP	Reserved.

Field Name	Bits	Reset	Description
PK23_PWR_EN	10	Set by OTP	Public Key 3/2 Power Enable. 0 Disable power to the PK 3/2 engines 1 Enable power to the PK 3/2 engines
RSVD	9	Set by OTP	Reserved.
PK01_PWR_EN	8	Set by OTP	Public Key 1/0 Power Enable. 0 Disable power to the PK 1/0 engines 1 Enable power to the PK 1/0 engines
TE_PWR_EN[7:0]	7:0	Set by OTP	Transform Engine Power Enable. Each bit of this field represents one of the eight Transform Engines. bit 0TE0 ---- bit 7TE7 The behavior for each bit is: 0 Disable power to the TEn 1 Enable power to the TEn

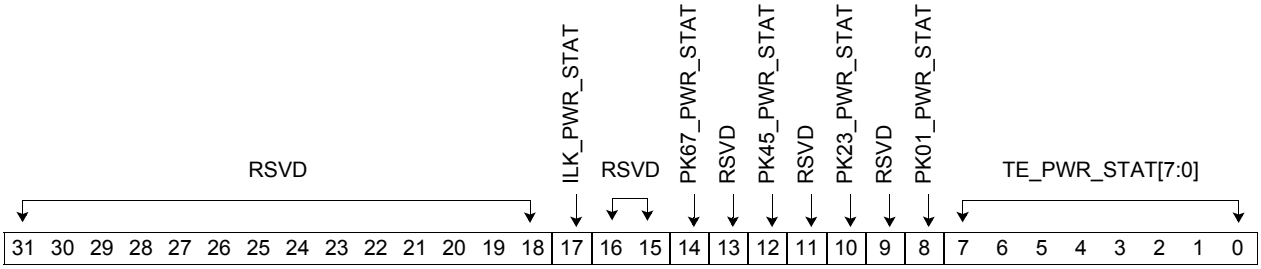
6.4.8 Power Status Register

The Power Status register is used to read whether the modules listed in this register have their power enabled or disabled.

Note that the power status for the PK engines is given in pairs rather than individually.

Type: Read only

Offset 0x404C



Field Name	Bits	Reset	Description
RSVD	31:18	X	Reserved.
ILK_PWR_STAT	17	X	Interlaken Power Status. 0 Power to the Interlaken module is enabled 1 Power to the Interlaken module is disabled
RSVD	16:15	0	Reserved.
PK67_PWR_STAT	14	X	Public key 7, 6 Power Status. 0 Power to the PK7 and PK6 engines is enabled 1 Power to the PK7 and PK6 engines is disabled
RSVD	13	0	Reserved.
PK45_PWR_STAT	12	X	Public key 4, 5 Power Status. 0 Power to the PK4 and PK5 engines is enabled 1 Power to the PK4 and PK5 engines is disabled
RSVD	11	0	Reserved.
PK23_PWR_STAT	10	0	Public key 2, 3 Power Status. 0 Power to the PK2 and PK3 engines is enabled 1 Power to the PK2 and PK3 engines is disabled
RSVD	9	0	Reserved.
PK01_PWR_STAT	8	X	Public key 0, 1 Power Status. 0 Power to the PK0 and PK1 engines is enabled 1 Power to the PK0 and PK1 engines is disabled

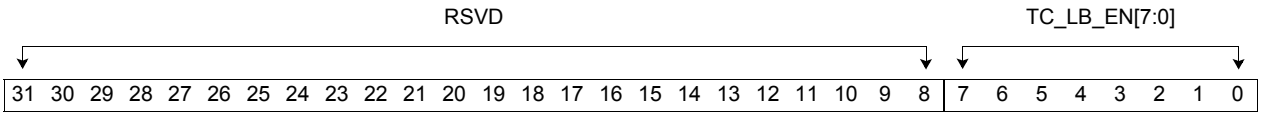
Field Name	Bits	Reset	Description
TE_PWR_STAT[7:0]	7:0	X	Transform Engine Power Status. Each bit of this field represents one of the eight Transform Engines. bit 0TE0 ---- bit 7TE7 The behavior for each bit is: 0 Power to the TEn is enabled 1 Power to the TEn is disabled

6.4.9 Transform Channel Load Balance Enable Register

The Transform Channel Load Balance Enable register is used to selectively enable the load balance control for each of the eight transform channels. Each transform channel consists of an Interlaken interface and a Transform Engine. If enabled, the load balancing will apply to the entire transform channel.

Type: Read/Write

Offset 0x4050



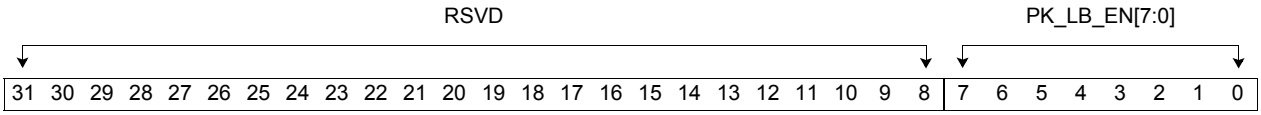
Field Name	Bits	Reset	Description
RSVD	31:8	24'h000000	Reserved.
TC_LB_EN[7:0]	7:0	8'hFF	Transform Channel Load Balance Enable. Each bit of this field represents one of the eight transform channels. bit 0TC0 ---- bit 7TC7 The behavior for each bit is: 0 Disable TCn to participate in load balance arbitration 1 Enable TCn to participate in load balance arbitration

6.4.10 Public Key Load Balance Enable Register

The Public Key Load Balance Enable register is used to selectively enable the load balance control for each of the eight Public Key processor engines.

Type: Read/Write

Offset 0x4054



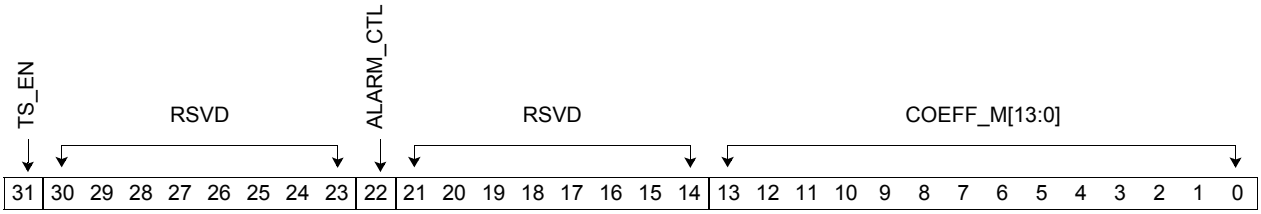
Field Name	Bits	Reset	Description
RSVD	31:8	24'h000000	Reserved.
PK_LB_EN[7:0]	7:0	8'hFF	Public Key Load Balance Enable. Each bit of this field enables one of the Public Key engines to participate in load balance arbitration. bit 0PK engine 0 bit 1PK engine 1 | bit 7PK engine 7 The behavior for each bit is: 0 Disable PK engine to participate in load balance arbitration 1 Enable PK engine to participate in load balance arbitration

6.4.11 Temperature Sensor Control 0 Register

The Temperature Sensor Control 0 register allows the host to enable/disable the temperature sensor and set the parameters used in calculating the temperature sensitivity.

Type: Read/Write

Offset 0x4060



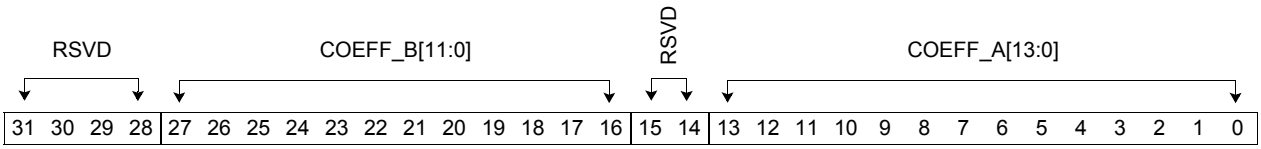
Field Name	Bits	Reset	Description
TS_EN	31	1'b0	Temperature Sensor Enable. 0 Disable temperature sensor 1 Enable temperature sensor
RSVD	30:23	8'h00	Reserved. Must be set to zero.
ALARM_CTL	22	1'b1	Alarm Control. This bit enables the TS_ERR_SRC bit in the "Temperature Sensor Data Register" to prevent false alarms from occurring, particularly during initialization. 0 Enable temperature sensor alarm 1 Disable temperature sensor alarm
RSVD	21:14	8'h00	Reserved.
COEFF_M[13:0]	13:0	14'h1E66	Temperature Sensor Coefficient M. This field is used by the XR9240 to calculate the device temperature. See "Temperature Sensor Controller" for more information. The default value should be suitable for most applications.

6.4.12 Temperature Sensor Control 1 Register

The Temperature Sensor Control 1 register provides the host the capability to set the coefficients A and B that are used to calculate the temperature sensitivity.

Type: Read/Write

Offset 0x4064



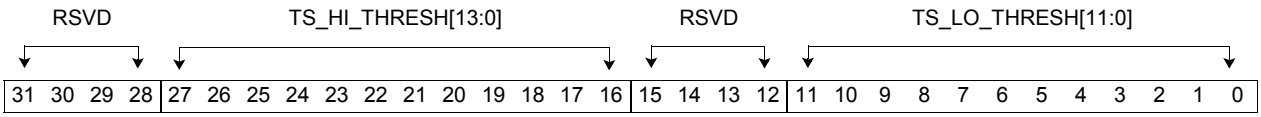
Field Name	Bits	Reset	Description
RSVD	31:28	4’b0000	Reserved.
COEFF_B[11:0]	27:16	12'h8A8	Temperature Sensor Coefficient B. This field is used by the XR9240 to calculate the device temperature. See “Temperature Sensor Controller” for more information. The default value should be suitable for most applications.
RSVD	15:14	2’b00	Reserved.
COEFF_A[13:0]	13:0	14'h172C	Temperature Sensor Coefficient A. This field is used by the XR9240 to calculate the device temperature. See “Temperature Sensor Controller” for more information. The default value should be suitable for most applications.

6.4.13 Temperature Sensor Threshold Register

The Temperature Sensor Threshold register provides the host the capability to set the high and low temperature thresholds.

Type: Read/Write

Offset 0x4068



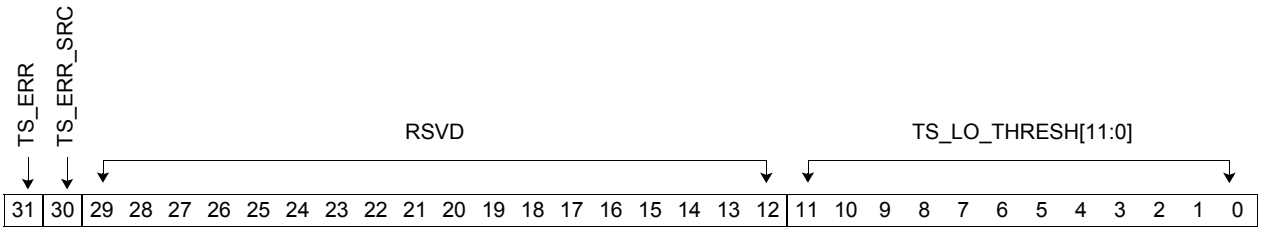
Field Name	Bits	Reset	Description
RSVD	31:28	4'b0000	Reserved.
TS_HI_THRESH [13:0]	27:16	14'h1B8	Temperature Sensor High Threshold. If the temperature is greater than this threshold, a temperature sensor event, TS_EVNT, interrupt will be asserted. To determine the reason for the interrupt, the host must read the value of the TS_ERR_SRC bit or the temperature value in the "Temperature Sensor Data Register". Refer to "Calculating the Temperature Thresholds" for a description on how to set this field. The default value corresponds to +110°C.
RSVD	15:12	4'b0000	Reserved.
TS_LO_THRESH [11:0]	11:0	12'h168	Temperature Sensor Low Threshold. If the XR9240 temperature returns to less than this threshold *after* an overhead condition, a temperature sensor event interrupt will be asserted. To determine the reason for the interrupt, the host must read the value of the TS_ERR_SRC bit or the temperature value in the "Temperature Sensor Data Register". Refer to "Calculating the Temperature Thresholds" for a description on how to set this field. The default value corresponds to +90°C.

6.4.14 Temperature Sensor Data Register

The Temperature Sensor Data register provides the host the capability to determine if a temperature error has occurred and read the conditions that caused the error.

Type: Read only

Offset 0x406C



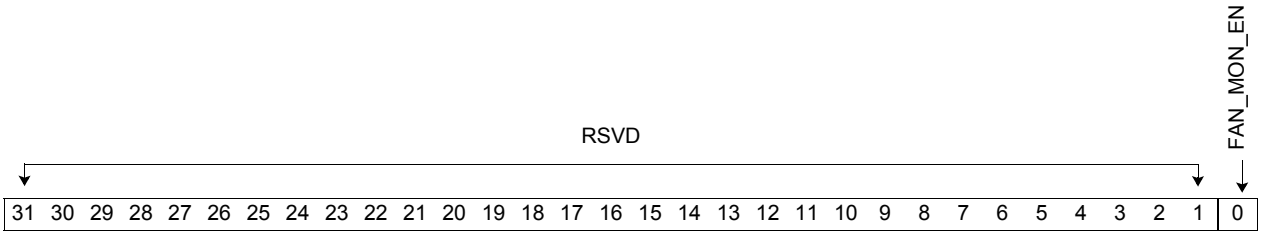
Field Name	Bits	Reset	Description
TS_ERR	31	X	Temperature Sensor Error. 0 Temperature is below high threshold 1 Temperature is above high threshold, or lower than the low threshold after an overheat condition
TS_ERR_SRC	30	1'b0	Temperature Sensor Error Source. This bit may be read by the host to infer the cause for the temperature sensor interrupt. 0 XR9240 temperature is less than the low threshold 1 XR9240 temperature is greater than the high threshold
RSVD	29:12	18'h0 0000	Reserved.
TS_DATA[11:0]	11:0	12'h000	Temperature Sensor Data. This field contains the real-timeXR9240 die temperature. Follow the instructions in "Calculating the Temperature Reading" to determine the temperature in °C.

6.4.15 Fan Monitor Enable Register

The Fan Monitor Enable register is used to enable or disable the fan monitoring capability in the XR9240. The fan monitor should be disabled while the host sets the threshold values in the "Fan Threshold Register" in order to mask spurious errors.

Type: Read/Write

Offset 0x4070



Field Name	Bits	Reset	Description
RSVD	31:1	31'h0000 0000	Reserved.
FAN_MON_EN	0	1'b0	Fan Monitor Enable. 0 Disable fan monitor 1 Enable fan monitor

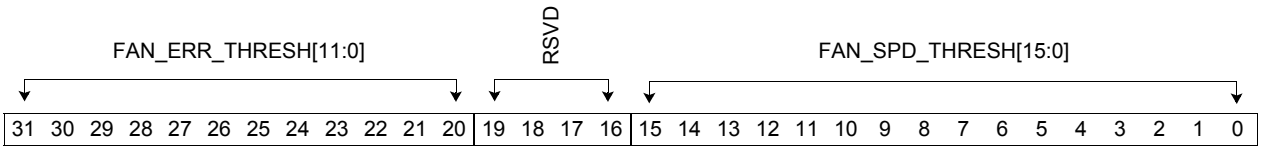
6.4.16 Fan Threshold Register

The Fan Threshold register is used to control the number of fan error interrupts sent to the host. Before writing to this register, the fan monitoring feature must be disabled by setting the FAN_EN bit in the "Fan Monitor Enable Register" to zero.

Refer to "Calculating the Fan Thresholds" for the formulas used to calculate the values to enter into this register.

Type: Read/Write

Offset 0x4074



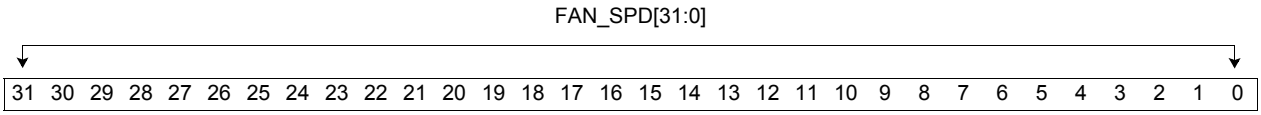
Field Name	Bits	Reset	Description
FAN_ERR_THRESH [11:0]	31:20	12'h7D0	Fan Error Threshold. A counter records the number of fan errors. An interrupt will be asserted if the number of fan errors is greater than the value of this field. The fan error counter is cleared after a fan interrupt is sent. The default value is 2000.
RSVD	19:16	4'b0000	Reserved.
FAN_SPD_THRESH [15:0]	15:0	16'h0016	Fan Speed Threshold. This field indicates the lowest fan speed threshold, in units of rpm (revolutions per minute) before the XR9240 will assert a fan error interrupt. The default value represents a fan speed threshold of 5000 rpm.

6.4.17 Fan Speed Register

The Fan Speed register may be read by the host to calculate the current fan speed. Refer to "Calculating the Fan Speed" for the formula to convert the register value to fan speed in rpm.

Type: Read only

Offset 0x4078



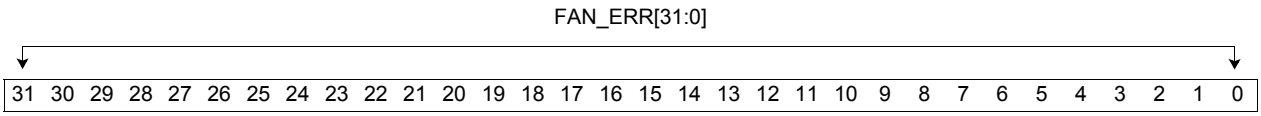
Field Name	Bits	Reset	Description
FAN_SPD[31:0]	31:0	32'h0000 0000	Fan Speed. An intermediate value that is used to calculate the fan speed. This value is the un-latched "live" fan speed that is updated every measurement period (half fan cycle). Refer to "Fan Monitor" for the formula used to calculate the actual fan speed.

6.4.18 Fan Speed After Error Register

The Fan Error register is the value of the fan speed that is latched when a fan error has occurred. The value will be updated for each fan error. The value will be cleared when the FAN_MON_EN bit in the "Fan Monitor Enable Register" is de-asserted. Refer to "Calculating the Fan Speed" for the formula to convert the register value to fan speed in rpm.

Type: Read only

Offset 0x407C

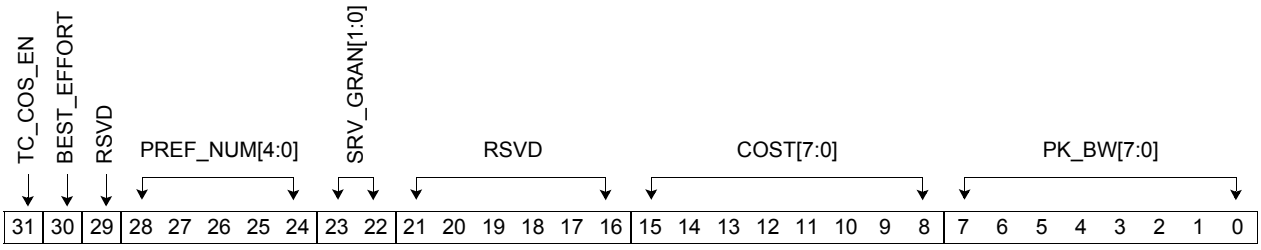


Field Name	Bits	Reset	Description
FAN_ERR[31:0]	31:0	32'h0000 0000	Fan Speed Latched During a Fan Error. This field represents the previous fan speed from the last fan error interrupt. If the fan has not rotated since the fan monitoring has been enabled, the value in this register will be all zeroes. Use the formulas in "Fan Monitor" to calculate the actual fan speed based on the value in this register.

6.4.19 Transform Channel Class of Service Register

The Transform Channel Class of Service register provides the host the capability to configure the class of service arbitration for each of the eight transform channel dynamic queues.

Type:	Read/Write	
Offset	0x4080	Dynamic Transform Channel Queue 0
	0x4084	Dynamic Transform Channel Queue 1
	0x4088	Dynamic Transform Channel Queue 2
	0x408C	Dynamic Transform Channel Queue 3
	0x408F	Dynamic Transform Channel Queue 4
	0x4094	Dynamic Transform Channel Queue 5
	0x4098	Dynamic Transform Channel Queue 6
	0x409C	Dynamic Transform Channel Queue 7



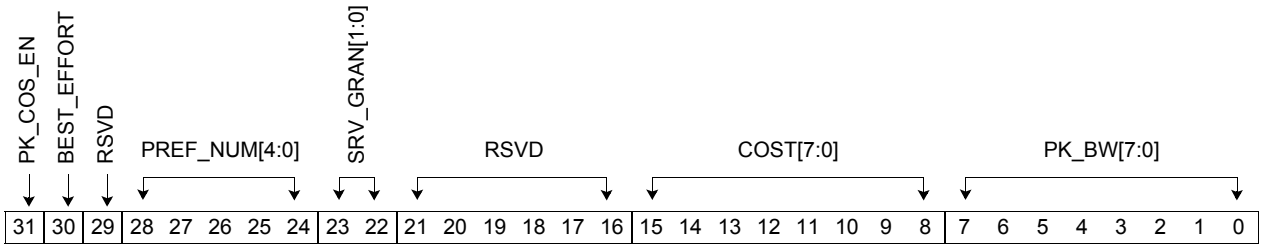
Field Name	Bits	Reset	Description
TC_COS_EN	31	1'b1	Transform Channel Class of Service Enable. If disabled, the TC queue will not request commands to prefetch. 0 Disable the TC queue to participate in COS arbitration 1 Enable the TC queue to participate in COS arbitration
BEST_EFFORT	30	1'b0	Best Effort Priority. This bit sets the priority scheme for the COS arbitration. This bit is only valid if TC_COS_EN is set to one. 0 Strict priority 1 Best effort priority
RSVD	29	1'b0	Reserved.

Field Name	Bits	Reset	Description
PREF_NUM[4:0]	28:24	5'b00000	Prefetch Number. This field determines the number of entries that the Transform Channel queue should prefetch. 0000032 entries 000011 entry : 1111131 entries
SRV_GRAN[1:0]	23:22	2'b00	Service Granularity. This field represents the number of command pointers that should be fetched per PCIe access. Increasing the number of entries may appear to save PCIe bandwidth, however the performance will vary based on the application. For most applications, it is recommended to use the default value. 00 1 entry 01 2 entries 10 4 entries 11 Reserved
RSVD	21:16	6'b000000	Reserved.
COST[7:0]	15:8	8'h00	Cost of the Queue. This field represents the cost of the queue in WDRR arbitration. This field is only valid for best effort priority. See "Best Effort Arbitration" for more information.
PK_BW[7:0]	7:0	8'h00	Peak Bandwidth of the Queue. This field is only valid fora strict priority queue. The value is a power of 2. The minimum bandwidth is 1 Mbps; the maximum bandwidth is 64 Gbps. 0x001 Mbps * 2^0 : 0xFF1 Mbps * 2^{16}

6.4.20 Public Key Class of Service Register

The Public Key Class of Service register provides the host the capability to configure the class of service arbitration for each of the four public key queues.

Type:	Read/Write	
Offset	0x40A0	Public Key module 0/1
	0x40A4	Public Key module 2/3
	0x40A8	Public Key module 4/5
	0x40AC	Public Key module 6/7



Field Name	Bits	Reset	Description
PK_COS_EN	31	1'b1	Public Key Class of Service Enable. If disabled, the PK queue will not request commands to prefetch. 0 Disable the PK queue to participate in COS arbitration 1 Enable the PK queue to participate in COS arbitration
BEST_EFFORT	30	1'b0	Best Effort Priority. This bit sets the priority scheme for the COS arbitration. This bit is only valid if PK_COS_EN is set to one. 0 Strict priority 1 Best effort priority
RSVD	29	1'b0	Reserved.
PREF_NUM[4:0]	28:24	5'b00001	Prefetch Number. This field determines the number of entries that the PK queue should prefetch. For the PK queue, the value cannot be changed from the default.

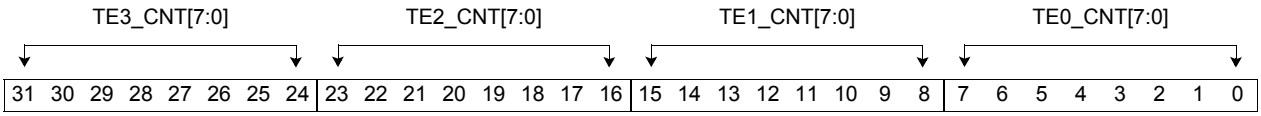
Field Name	Bits	Reset	Description
SRV_GRAN[1:0]	23:22	2'b00	Service Granularity. This field represents the number of command pointers that should be fetched per PCIe access. Increasing the value saves PCIe bandwidth, however the XR9240 may not be able fetch enough command pointers for processing. 00 1 entry 01 2 entries 10 4 entries 11 Reserved
RSVD	21:16	6'b0000 00	Reserved.
COST[7:0]	15:8	8'h00	Cost of the Queue. This field represents the cost of the queue in WDRR arbitration. This field is only valid for best effort priority. See "Best Effort Arbitration" for more information.
PK_BW[7:0]	7:0	8'h00	Peak Bandwidth of the Queue. This field is only valid for a strict priority queue. The minimum bandwidth (default value) is 256 op/s; the maximum bandwidth is 65,536 op/s. Refer to "Strict Priority Arbitration" for more information.

6.4.21 Outstanding Command Counter 0 Register

The Outstanding Command Counter 0 register provides the host the capability to read the number of outstanding command assigned to Transform Engines 0-3. This register should be read before gating the power or clocks to these modules (see "Device Power Management").

Type: Read only

Offset 0x40B0



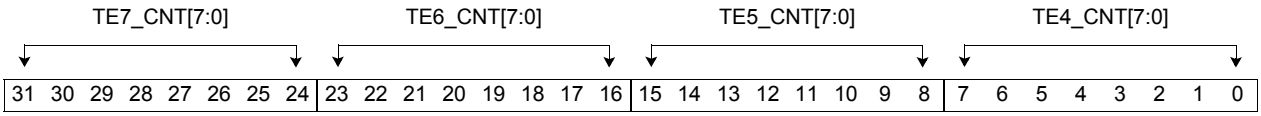
Field Name	Bits	Reset	Description
TE3_CNT[7:0]	31:24	8'h00	Transform Engine 3 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.
TE2_CNT[7:0]	23:16	8'h00	Transform Engine 2 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.
TE1_CNT[7:0]	15:8	8'h00	Transform Engine 1 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.
TE0_CNT[7:0]	7:0	8'h00	Transform Engine 0 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.

6.4.22 Outstanding Command Counter 1 Register

The Outstanding Command Counter 1 register provides the host the capability to read the number of outstanding command assigned to Transform Engines 7-4. This register should be read before gating the power or clocks to these modules (see "Device Power Management").

Type: Read only

Offset 0x40B4



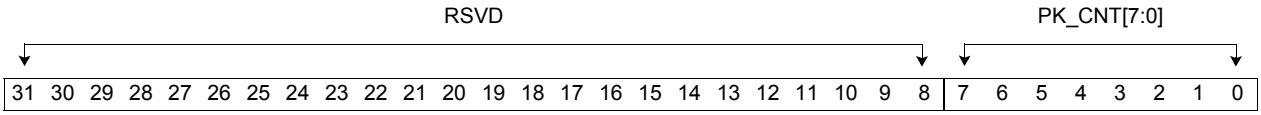
Field Name	Bits	Reset	Description
TE7_CNT[7:0]	31:24	8'h00	Transform Engine 7 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.
TE6_CNT[7:0]	23:16	8'h00	Transform Engine 6 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.
TE5_CNT[7:0]	15:8	8'h00	Transform Engine 5 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.
TE4_CNT[7:0]	7:0	8'h00	Transform Engine 4 Counter. Reading this counter returns the number of outstanding commands assigned to this Transform Engine.

6.4.23 Outstanding Command Counter 3 Register

The Outstanding Command Counter 3 register provides the host the capability to read the number of outstanding command assigned to Public Key Engine. This register should be read before gating the power or clocks to this module (see "Device Power Management").

Type: Read only

Offset 0x40B8



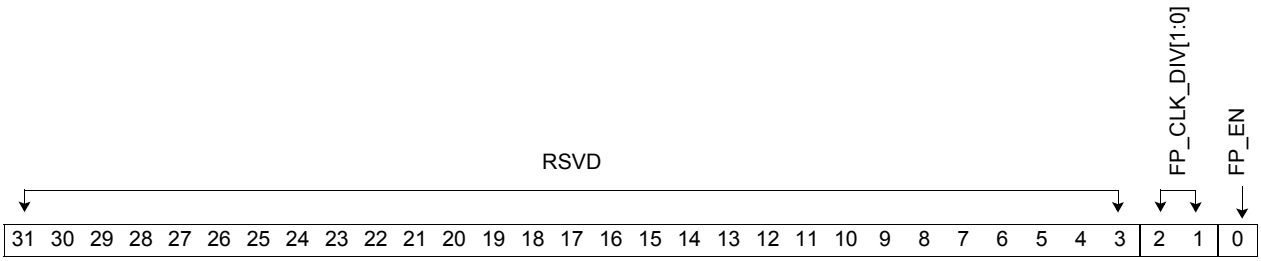
Field Name	Bits	Reset	Description
TE7_CNT[7:0]	31:8	24'h000000	Reserved.
PK_CNT[7:0]	7:0	8'h00	Public Key Engine Counter. Reading this counter returns the number of outstanding commands assigned to the Public Key Engine.

6.4.24 Field Programming Interface Control Register

The Field Programming Interface Control register provides the host the capability to enable/ disable this interface and set the clock divider.

Type: Read/Write

Offset 0x40D0



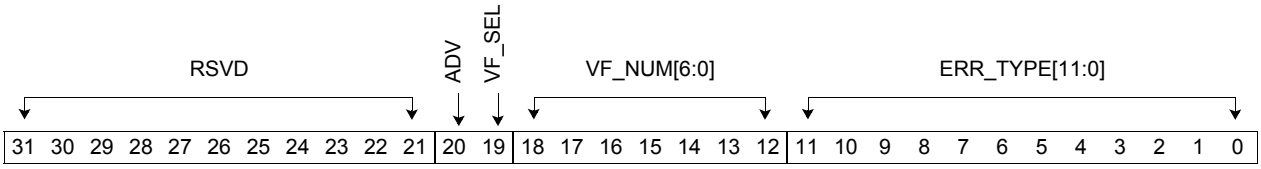
Field Name	Bits	Reset	Description
RSVD	31:3	29'h0000 0000	Reserved.
FP_CLK_DIV[1:0]	2:1	2'b01	Field Programming Clock Divider. This field is used to adjust the clock frequency of the output signal PROG_CLK_OUT (see "FPGA Field Programming Interface"). 00 250M 01 125M 10 83.3M 11 62.5M
FP_EN	0	1'b0	Field Programming Enable. 0 Disable the field programming interface 1 Enable the field programming interface

6.4.25 PCIe Error Injection Register

The PCIe Error Injection register may be used to test the PCIe interface error handling capabilities of the XR9240 and its software development kit (SDK).

Type: Read/Write

Offset 0x40E0



Field Name	Bits	Reset	Description
RSVD	31:21	11'h000	Reserved.
ADV	20	1'b0	Advisory Error. 0 Injected error is not an advisory error 1 Injected error is an advisory error
VF_SEL	19	1'b0	Virtual Function Selection. This bit determines whether the error is injected into the physical or virtual function. 0 PF selected 1 VF selected
VF_NUM[6:0]	18:12	7'b000 0000	Virtual Function Number. This field is only valid if VF_SEL is set to one. When the error is send to a VF, this field represents the particular VF affected.
ERR_TYPE[11:0]	11:0	12'h000	PCIe Error Type. Only one error type should be set at a time. Bit 0Malformed TLP Bit 1Receiver overflow Bit 2Unexpected CPL Bit 3Completer abort Bit 4CPL timeout Bit 5Unsupported request Bit 6ECRC verification failed Bit 7Poisoned TLP received Bits 8-11 Reserved

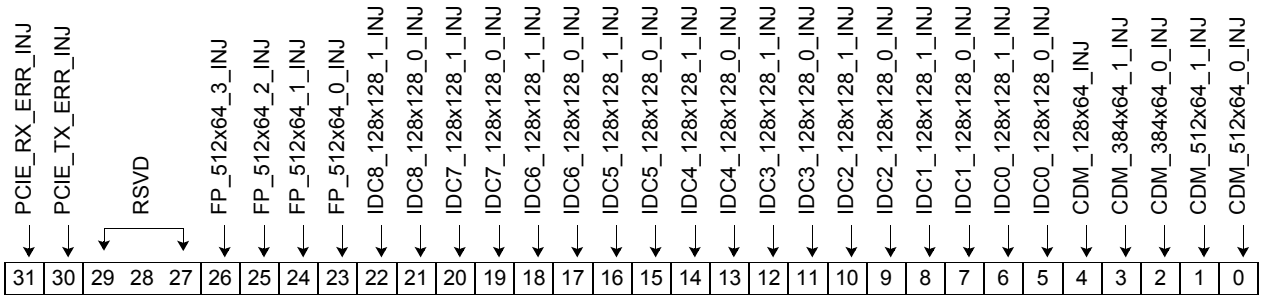
6.5 PCIe DMA Control Registers

6.5.1 DMA 2-bit ECC Error Injection Register

This register is used for testing the 15 DMA memories by allowing 2-bit ECC errors to be injected into the DMA Control module memories. The results of the test may be read using the "DMA 2-bit ECC Error Log Register".

Type: Read/Write

Offset 0x4400



Field Name	Bits	Reset	Description
PCIE_RX_ERR_INJ	31	1'b0	PCIe Controller Receiver ECC Error Injection. 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
PCIE_TX_ERR_INJ	30	1'b0	PCIe Controller Transmitter ECC Error Injection. 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
RSVD	29:27	3'b000	Reserved.
FP_512x64_3_INJ	26	1'b0	Free Pool 512x64 ECC Error Injection. (part 3) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
FP_512x64_2_INJ	25	1'b0	Free Pool 512x64 ECC Error Injection. (part 2) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
FP_512x64_1_INJ	24	1'b0	Free Pool 512x64 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
FP_512x64_0_INJ	23	1'b0	Free Pool 512x64 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error

Field Name	Bits	Reset	Description
IDC8_128x128_1_INJ	22	1'b0	Inbound Data Controller 8 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC8_128x128_0_INJ	21	1'b0	Inbound Data Controller 8 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC7_128x128_1_INJ	20	1'b0	Inbound Data Controller 7 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC7_128x128_0_INJ	19	1'b0	Inbound Data Controller 7 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC6_128x128_1_INJ	18	1'b0	Inbound Data Controller 6 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC6_128x128_0_INJ	17	1'b0	Inbound Data Controller 6 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC5_128x128_1_INJ	16	1'b0	Inbound Data Controller 5 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC5_128x128_0_INJ	15	1'b0	Inbound Data Controller 5 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC4_128x128_1_INJ	14	1'b0	Inbound Data Controller 4 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC4_128x128_0_INJ	13	1'b0	Inbound Data Controller 4 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC3_128x128_1_INJ	12	1'b0	Inbound Data Controller 3 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error

Field Name	Bits	Reset	Description
IDC3_128x128_0_INJ	11	1'b0	Inbound Data Controller 3 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC2_128x128_1_INJ	10	1'b0	Inbound Data Controller 2 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC2_128x128_0_INJ	9	2'b00	Inbound Data Controller 2 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC1_128x128_1_INJ	8	1'b0	Inbound Data Controller 1 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC1_128x128_0_INJ	7	1'b0	Inbound Data Controller 1 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC0_128x128_1_INJ	6	1'b0	Inbound Data Controller 0 128x128 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
IDC0_128x128_0_INJ	5	1'b0	Inbound Data Controller 0 128x128 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
CDM_128x64_INJ	4	1'b0	CDM 128x64 ECC Error Injection. 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
CDM_384x64_1_INJ	3	1'b0	CDM 384x64 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
CDM_384x64_0_INJ	2	1'b0	CDM 384x64 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
CDM_512x64_1_INJ	1	1'b0	CDM 512x64 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
CDM_512x64_0_INJ	0	1'b0	CDM 512x64 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error

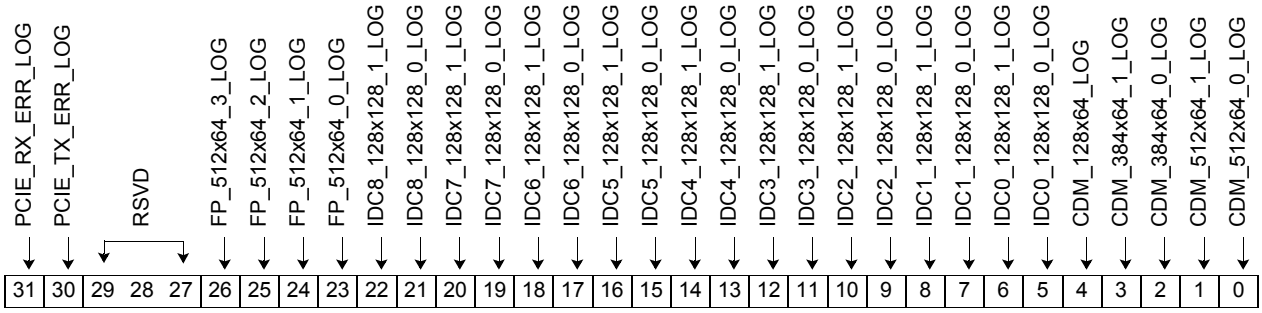
6.5.2 DMA 2-bit ECC Error Log Register

This register contains the 2-bit ECC error log for real-time or injected errors.

This register may be used for testing in conjunction with the "DMA 2-bit ECC Error Injection Register". It allows the injected 2-bit ECC error log to be read.

Type: Read/Write

Offset 0x4404



Field Name	Bits	Reset	Description
PCIE_RX_ERR_LOG	31	1'b0	PCIe Controller Receiver ECC Error Log. 0 No 2-bit error occurred 1 2-bit ECC error occurred
PCIE_TX_ERR_LOG	30	1'b0	PCIe Controller Transmitter ECC Error Log. 0 No 2-bit error occurred 1 2-bit ECC error occurred
RSVD	29:27	3'b000	Reserved.
FP_512x64_3_LOG	26	1'b0	Free Pool 512x64 ECC Error Log. (part 3) 0 No 2-bit error occurred 1 2-bit ECC error occurred
FP_512x64_2_LOG	25	1'b0	Free Pool 512x64 ECC Error Log. (part 2) 0 No 2-bit error occurred 1 2-bit ECC error occurred
FP_512x64_1_LOG	24	1'b0	Free Pool 512x64 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
FP_512x64_0_LOG	23	1'b0	Free Pool 512x64 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred

Field Name	Bits	Reset	Description
IDC8_128x128_1_LOG	22	1'b0	Inbound Data Controller 8 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC8_128x128_0_LOG	21	1'b0	Inbound Data Controller 8 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC7_128x128_1_LOG	20	1'b0	Inbound Data Controller 7 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC7_128x128_0_LOG	19	1'b0	Inbound Data Controller 7 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC6_128x128_1_LOG	18	1'b0	Inbound Data Controller 6 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC6_128x128_0_LOG	17	1'b0	Inbound Data Controller 6 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC5_128x128_1_LOG	16	1'b0	Inbound Data Controller 5 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC5_128x128_0_LOG	15	1'b0	Inbound Data Controller 5 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC4_128x128_1_LOG	14	1'b0	Inbound Data Controller 4 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC4_128x128_0_LOG	13	1'b0	Inbound Data Controller 4 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC3_128x128_1_LOG	12	1'b0	Inbound Data Controller 3 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred

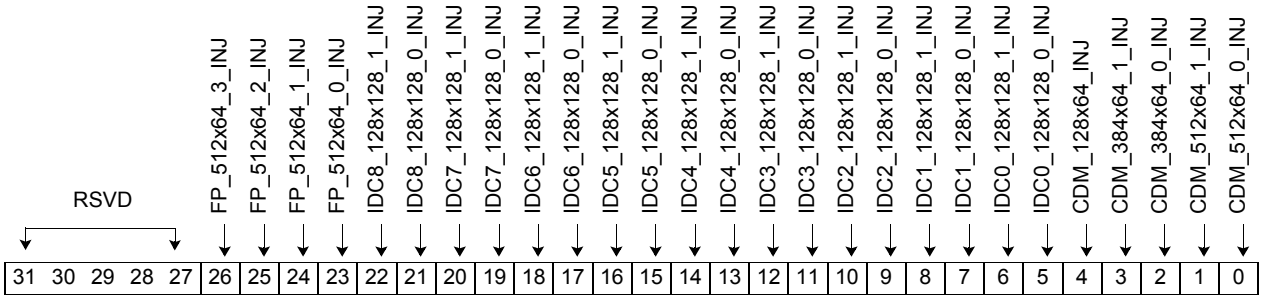
Field Name	Bits	Reset	Description
IDC3_128x128_0_LOG	11	1'b0	Inbound Data Controller 3 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC2_128x128_1_LOG	10	1'b0	Inbound Data Controller 2 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC2_128x128_0_LOG	9	2'b00	Inbound Data Controller 2 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC1_128x128_1_LOG	8	1'b0	Inbound Data Controller 1 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC1_128x128_0_LOG	7	1'b0	Inbound Data Controller 1 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC0_128x128_1_LOG	6	1'b0	Inbound Data Controller 0 128x128 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
IDC0_128x128_0_LOG	5	1'b0	Inbound Data Controller 0 128x128 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
CDM_128x64_LOG	4	1'b0	CDM 128x64 ECC Error Log. 0 No 2-bit error occurred 1 2-bit ECC error occurred
CDM_384x64_1_LOG	3	1'b0	CDM 384x64 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
CDM_384x64_0_LOG	2	1'b0	CDM 384x64 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred
CDM_512x64_1_LOG	1	1'b0	CDM 512x64 ECC Error Log. (part 1) 0 No 2-bit error occurred 1 2-bit ECC error occurred
CDM_512x64_0_LOG	0	1'b0	CDM 512x64 ECC Error Log. (part 0) 0 No 2-bit error occurred 1 2-bit ECC error occurred

6.5.3 DMA 1-bit ECC Error Injection Register

This register is used for testing. It allows non-fatal correctable 1-bit ECC errors to be injected into the DMA Control module. The results of the test may be read using the "DMA 1-bit ECC Error Counter Register".

Type: Read/Write

Offset 0x4408



Field Name	Bits	Reset	Description
RSVD	31:27	5'b00000	Reserved.
FP_512x64_3_INJ	26	1'b0	Free Pool 512x64 ECC Error Injection. (part 3) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
FP_512x64_2_INJ	25	1'b0	Free Pool 512x64 ECC Error Injection. (part 2) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
FP_512x64_1_INJ	24	1'b0	Free Pool 512x64 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
FP_512x64_0_INJ	23	1'b0	Free Pool 512x64 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC8_128x128_1_INJ	22	1'b0	Inbound Data Controller 8 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC8_128x128_0_INJ	21	1'b0	Inbound Data Controller 8 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error

Field Name	Bits	Reset	Description
IDC7_128x128_1_INJ	20	1'b0	Inbound Data Controller 7 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC7_128x128_0_INJ	19	1'b0	Inbound Data Controller 7 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC6_128x128_1_INJ	18	1'b0	Inbound Data Controller 6 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC6_128x128_0_INJ	17	1'b0	Inbound Data Controller 6 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC5_128x128_1_INJ	16	1'b0	Inbound Data Controller 5 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC5_128x128_0_INJ	15	1'b0	Inbound Data Controller 5 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC4_128x128_1_INJ	14	1'b0	Inbound Data Controller 4 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC4_128x128_0_INJ	13	1'b0	Inbound Data Controller 4 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC3_128x128_1_INJ	12	1'b0	Inbound Data Controller 3 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC3_128x128_0_INJ	11	1'b0	Inbound Data Controller 3 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC2_128x128_1_INJ	10	1'b0	Inbound Data Controller 2 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error

Field Name	Bits	Reset	Description
IDC2_128x128_0_INJ	9	2'b00	Inbound Data Controller 2 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC1_128x128_1_INJ	8	1'b0	Inbound Data Controller 1 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC1_128x128_0_INJ	7	1'b0	Inbound Data Controller 1 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC0_128x128_1_INJ	6	1'b0	Inbound Data Controller 0 128x128 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
IDC0_128x128_0_INJ	5	1'b0	Inbound Data Controller 0 128x128 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
CDM_128x64_INJ	4	1'b0	CDM 128x64 ECC Error Injection. 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
CDM_384x64_1_INJ	3	1'b0	CDM 384x64 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
CDM_384x64_0_INJ	2	1'b0	CDM 384x64 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
CDM_512x64_1_INJ	1	1'b0	CDM 512x64 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
CDM_512x64_0_INJ	0	1'b0	CDM 512x64 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error

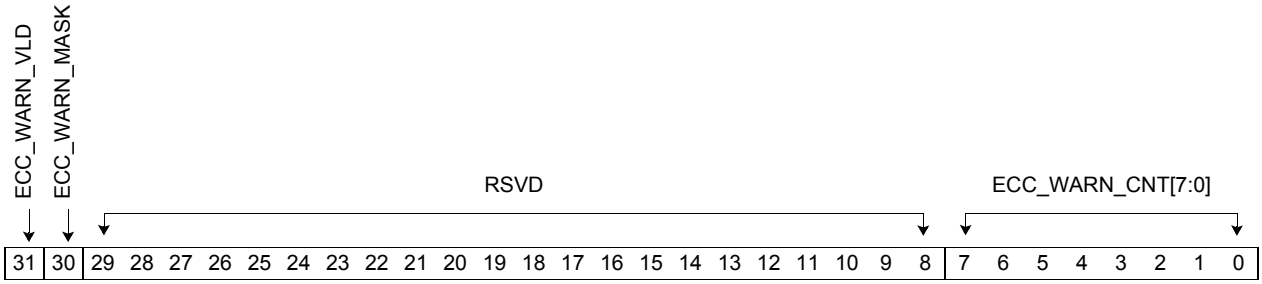
6.5.4 DMA 1-bit ECC Error Counter Register

This register contains the 1-bit ECC error log for real-time or injected errors.

This register may be used for testing in conjunction with the "DMA 1-bit ECC Error Injection Register". See "SRAM Protection" for more information.

Type: Read/Write

Offset 0x440C



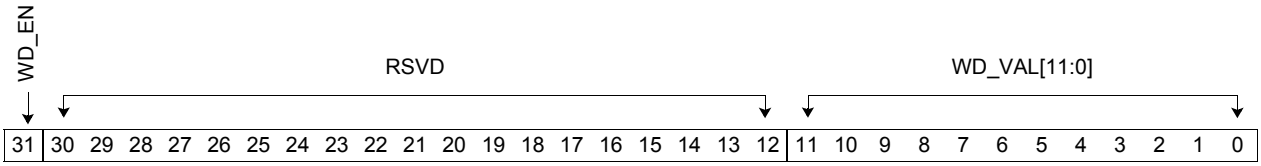
Field Name	Bits	Reset	Description
ECC_WARN_VLD	31	1'b0	ECC Warning Valid. This bit will be set if at least one 1-bit ECC warning has been detected. 0 ECC warning not detected 1 ECC warning detected
ECC_WARN_MASK	30	1'b0	ECC Warning Overflow Mask. This bit is used to indicate whether to mask or unmask the DATA_ERR interrupt in the "PCIe Interrupt Source Register" if a 1-bit ECC warning was detected. 0 Do not mask the ECC warning interrupt 1 Mask the ECC warning interrupt
RSVD	29:8	22'h00000	Reserved.
ECC_WARN_CNT[7:0]	7:0	8'h00	ECC Warning Counter. This field contains the 1-bit ECC warnings from all memories. If multiple 1-bit ECC warnings are reported within the same clock cycle, only one warning will be recognized. The host may write to this field at any time to clear the counter.

6.5.5 PCIe DMA WatchDog Timer Control Register

The PCIe DMA WatchDog Timer Control register is used to set the initial value for the PCIe DMA watchdog timer. Once enabled, the timer will start counting when a new command is dispatched to the PCIe DMA. If the watchdog timer expires, an error will be logged in the "PCIe DMA WatchDog Timer Status Register".

Type: Read/Write

Offset 0x4420



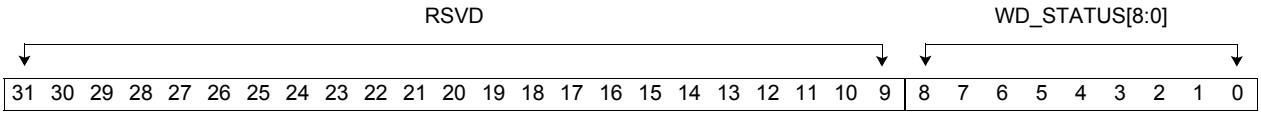
Field Name	Bits	Reset	Description
WD_EN	31	1'b0	PCIe DMA WatchDog Enable. 0 Disable PCIe DMA watchdog timer 1 Enable PCIe DMA watchdog timer
RSVD	30:12	19'h0000 00	Reserved.
WD_VAL[11:0]	11:0	12'h000	WatchDog Preload Value. This field is the initial value for the watchdog timer. Each tick represents a time unit of 1ms. The watchdog timer must be configured before being enabled because the reset null value is undefined.

6.5.6 PCIe DMA WatchDog Timer Status Register

The PK WatchDog Timer Status register allows the host to monitor the expiration of the PCIe watchdog timers in the XR9240 device.

Type: Read/Write

Offset 0x4424

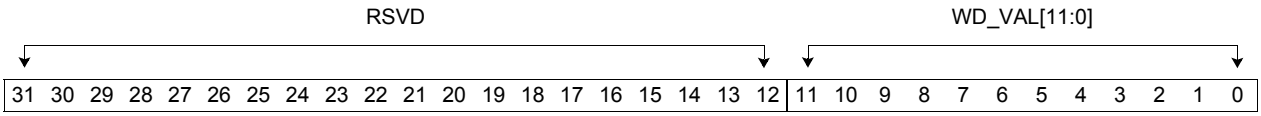


Field Name	Bits	Reset	Description
RSVD	31:9	23'h000000	Reserved.
WD_STATUS[8:0]	8:0	9'h000	WatchDog Status. Each bit of this field corresponds to a DMA channel. bit 8: PK DMA channel bit 7: TC DMA channel 7 : bit 0: TC DMA channel 0 The encoding for each bit is: 0 Watchdog timer not expired 1 Watchdog timer expired

6.5.7 PCIe DMA WatchDog Timer Value Register

The PCIe WatchDog Timer Value register is used to read the value for the PCIe watchdog timers.

Type:	Read only	
Offset	0x4440	TC DMA channel 0
	0x4444	TC DMA channel 1
	0x4448	TC DMA channel 2
	0x444C	TC DMA channel 3
	0x4450	TC DMA channel 4
	0x4454	TC DMA channel 5
	0x4458	TC DMA channel 6
	0x445C	TC DMA channel 7
	0x4460	PK DMA channel



Field Name	Bits	Reset	Description
RSVD	31:12	20'h00000	Reserved.
WD_VAL[11:0]	11:0	12'h000	WatchDog Timer Value.

6.6 ILK Control Registers

There are three types of ILK registers defined within the XR9240 PCIe BAR0 ILK address space:

- "ILK DMA Control" (0x4800 - 0x48FF)
- "ILK PHY Control Registers" (0x4900 - 0x49FF)
- "ILK MAC Control Registers" (0x4C00 - 0x4CFF)

All offsets within the ILK register space that are not described in this section are reserved and should not be written to by the host.

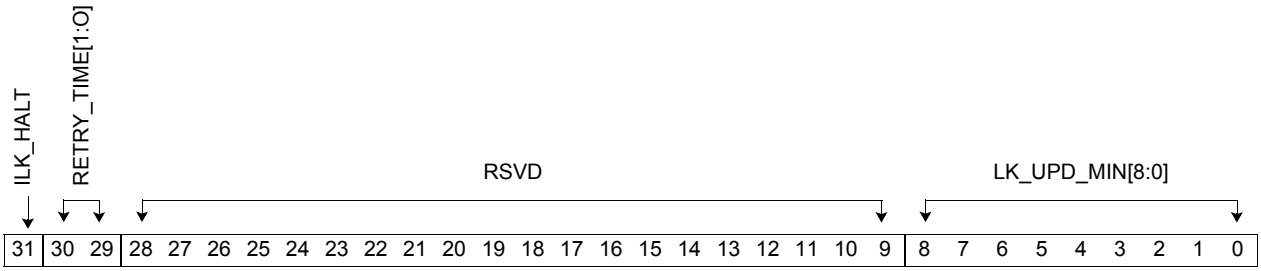
6.6.1 ILK DMA Control

6.6.1.1 ILK DMA Configuration Register

The ILK DMA Configuration register provides the host the ability to configure the ILK DMA interface.

Type: Read/Write

Offset 0x4828



Field Name	Bits	Reset	Description
ILK_HALT	31	1'b0	Halt ILK. This bit is used to select the ILK behavior if a serious error occurs. 0 Continue working 1 Halt
RETRY_TIME[1:0]	30:29	2'b00	ILK Retry Time Delay. This field is used to select the number of clock cycles to wait before resending the data burst. 00 512 cycles 01 1028 cycles 10 2048 cycles 11 4096 cycles
RSVD	28:9	20'h00000 0	Reserved.
LK_UPD_MIN[8:0]	8:0	9'h050	Link Update Minimum Value. This field sets the time interval allowed between two link control packets. The default value represents the fixed value for this field of 200 ns.

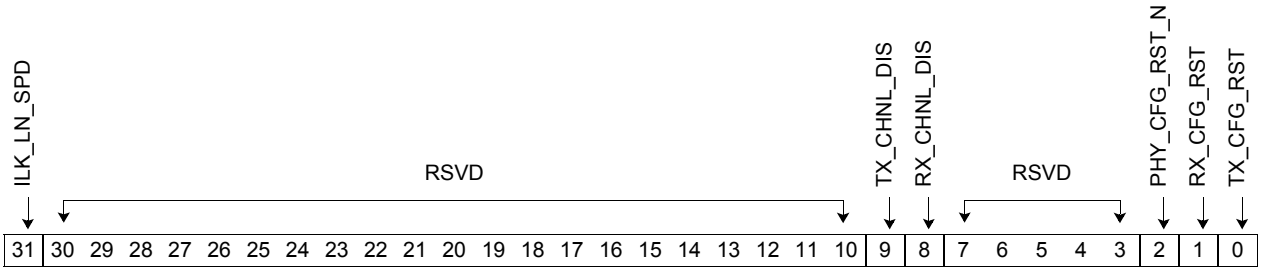
6.6.1.2 ILK DMA Reset Register

The ILK DMA Reset register provides the host the ability to reset the Interlaken PHY, receiver and transmitter so that their registers may be accessed. Once set, a reset bit must be cleared by the host. The ILK function will be immediately reset.

This register is also used to set the Interlaken lane speed, and enable or disable the transmit and receive channels.

Type: Read/Write

Offset 0x482C



Field Name	Bits	Reset	Description
ILK_LN_SPD	31	1'b0	Interlaken Lane Speed. This bit is used to configure the ILK PHY Serdes speed. 0 6.25Gbps 1 3.125Gbps
RSVD	30:10	21'h000000	Reserved.
TX_CHNL_DIS	9	1'b0	Interlaken Transmit Channel Disable. This bit is used to enable or disable the Interlaken TX channels. The host must disable the TX channels before configuring the ILK PHY to a low power state. 0 Enable ILK TX channel 1 Disable ILK TX channel
RX_CHNL_DIS	8	1'b0	Interlaken Receive Channel Disable. This bit is used to enable or disable the Interlaken RX channels. The host must disable the RX channels before configuring the ILK PHY to a low power state. 0 Enable ILK RX channel 1 Disable ILK RX channel
RSVD	7:3	5'b00000	Reserved.

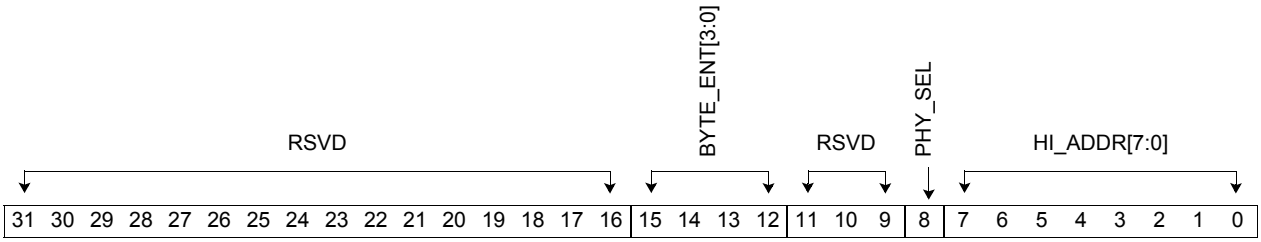
Field Name	Bits	Reset	Description
PHY_CFG_RST_N	2	1'b0	ILK PHY Reset. This active low bit must be cleared before programming the Interlaken PHY registers and then set after the programming is complete. 0 Reset ILK PHY 1 Do not reset ILK PHY
RX_CFG_RST	1	1'b1	ILK Receiver Reset. This bit must be set before programming the receiver registers of the ILK controller and then cleared after the programming is complete. 0 Do not reset ILK RX 1 Reset ILK RX
TX_CFG_RST	0	1'b1	ILK Transmitter Reset. This bit must be set before programming the transmitter registers of the ILK controller. 0 Do not reset ILK TX 1 Reset ILK TX

6.6.1.3 ILK DMA PHY Access Register

The ILK DMA PHY Access register provides the host the ability to access the ILK PHY registers using indirect addressing. Refer to "Programming the ILK PHY" for detailed usage information.

Type: Read/Write

Offset 0x4830

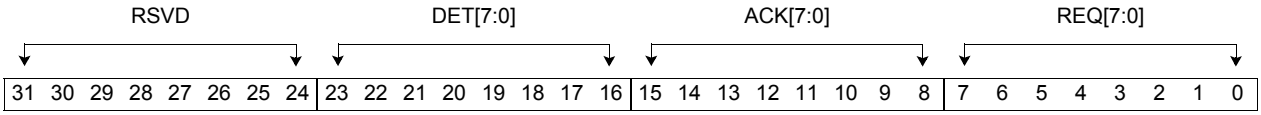


Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
BYTE_EN[3:0]	15:12	4'b1111	Byte Enable. Software uses this field to determine which byte of the read word address data is returned. If set, the corresponding byte is read back; if not set, the byte will be read back as 8'h00. The value for this field is determine by the least two significant bits of the ILK PHY register address. 0000Reserved xxx1Byte 0 is valid xx1xByte 1 is valid x1xxByte 2 is valid 1xxxByte 3 is valid
RSVD	11:9	3'b000	Reserved.
PHY_SEL	8	1'b0	ILK PHY Select. This bit is used to select between the two ILK PHYs. 0 ILK PHY 0 (corresponds to lanes 0-3) 1 ILK PHY 1 (corresponds to lanes 4-7)
HI_ADDR[7:0]	7:0	8'h00	Upper ILK PHY Address. This field represents the upper byte of the ILK PHY address. Refer to "Programming the ILK PHY" for more information.

6.6.1.4 ILK DMA Receiver Detection Register

The ILK DMA Receiver Detection register provides the host the ability to detect an available ILK receiver on the FPGA. Refer to Section 5.6.3 for a description of its usage.

Type:	Read only	Bits [23:8]
	Read/Write	Bits [7:0]
Offset	0x4834	



Field Name	Bits	Reset	Description
RSVD[7:0]	31:24	8'h00	Reserved.
DET[7:0]	23:16	8'h00	Receiver Detect Flag. Each bit in this field represents one of the ILK receiver lanes. 0 No receiver detected 1 Receiver detected
ACK[7:0]	15:8	8'h00	Receiver Acknowledge. Each bit in this field represents one of the ILK receiver lanes. The bit is auto-cleared after the corresponding REQ bit has been cleared. 0 Receiver acknowledge not present 1 Receiver acknowledge present
REQ[7:0]	7:0	8'h00	Request to Detect Receiver. Each bit in this field represents one of the ILK receiver lanes. 0 Do not request receiver detection 1 Request receiver detection

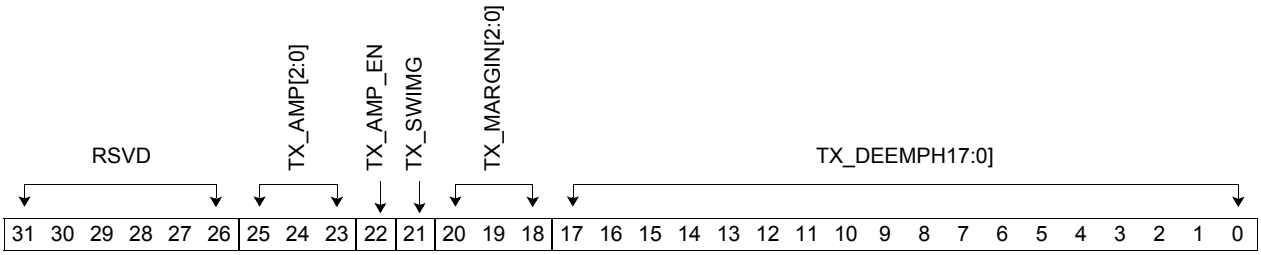
6.6.1.5 ILK DMA Transmitter Equalization Register

The ILK DMA Transmitter Equalization register provides the host the ability to adjust the ILK transmitter amplifier equalization settings in the ILK interface logic.

The recommended values for the fields in this register will be provided for the production part after calibration measurements have been taken.

Type: Read/Write

Offset 0x4838



Field Name	Bits	Reset	Description
RSVD	31:26	6'b00 0000	Reserved.
TX_AMP[2:0]	25:23	3'b000	Transmitter Amplifier. This field is only valid when TX_AMP_EN is set to one.
TX_AMP_EN	22	1'b0	Transmitter Amplifier Enable. 0 Disable the TX amplifier 1 Enable the TX amplifier
TX_SWING	21	1'b0	Transmitter Swing. This bit must be set to zero.
TX_MARGIN[2:0]	20:18	3'b000	Transmitter Margin.
TX_DEEMPH[17:0]	17:0	18'h0 0000	Transmitter De-emphasis.

6.6.2 ILK PHY Control Registers

The ILK PHY registers are accessed indirectly using the procedure defined in Section 5.6.4. The ILK PHY registers listed in Table 6-1 must be programmed after every reset with the values shown.

In Table 6-1, the columns labeled HI_ADDR[7:0] and BYTE[3:0] must be programmed into the "ILK DMA PHY Access Register" before the Value is programmed into the PCIe Address Offset. Note that to simplify the table, the "ILK DMA PHY Access Register" bit PHY_SEL is not shown. In order to configure all lanes, each entry in the table must be programmed twice: once with PHY_SEL = 0, and then again with PHY_SEL = 1.

Table 6-1. ILK PHY Fixed Register Values

ILK PHY Word Address (Hex)	HI_ADDR[7:0]	BYTE_EN[3:0]	PCIe Addr Offset (Hex)	Value (Hex)
0x0065 0x2065 0x4065 0x6065	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0010	0x4964	BF
0x0066 0x2066 0x4066 0x6066	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0100	0x4964	BF
0x0067 0x2067 0x4067 0x6067	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b1000	0x4964	08
0x0068 0x2068 0x4065 0x6068	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0001	0x4968	08
0x0069 0x2069 0x4069 0x6069	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0010	0x4968	13
0x006A 0x206A 0x406A 0x606A	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0100	0x4968	13
0x006B 0x206B 0x406B 0x606B	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b1000	0x4968	01

Table 6-1. ILK PHY Fixed Register Values

ILK PHY Word Address (Hex)	HI_ADDR[7:0]	BYTE_EN[3:0]	PCIe Addr Offset (Hex)	Value (Hex)
0x006C 0x206C 0x406C 0x606C	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0001	0x496C	01
0x006D 0x206D 0x406D 0x606D	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0010	0x496C	01
0x006E 0x206E 0x406E 0x606E	8'b00000000 8'b00100000 8'b01000000 8'b01100000	4'b0100	0x496C	01
0x8065	8'b10000000	4'b0010	0x4964	AA
0x8066	8'b10000000	4'b0100	0x4964	00
0x8067	8'b10000000	4'b1000	0x4964	62
0x8068	8'b10000000	4'b0001	0x4968	BF
0x8069	8'b10000000	4'b0010	0x4968	BF
0x806A	8'b10000000	4'b0100	0x4968	08
0x806B	8'b10000000	4'b1000	0x4968	08
0x806C	8'b10000000	4'b0001	0x496C	13
0x806D	8'b10000000	4'b0010	0x496C	13
0x806E	8'b10000000	4'b0100	0x496C	07
0x806F	8'b10000000	4'b1000	0x496C	02
0x8070	8'b10000000	4'b0001	0x4970	10
0x8071	8'b10000000	4'b0010	0x4970	00
0x8072	8'b10000000	4'b0100	0x4970	10
0x8073	8'b10000000	4'b1000	0x4970	00
0x8074	8'b10000000	4'b0001	0x4974	FF
0x8075	8'b10000000	4'b0010	0x4974	5F
0x8076	8'b10000000	4'b0100	0x4974	F5
0x8077	8'b10000000	4'b1000	0x4974	D9
0x8078	8'b10000000	4'b0001	0x4978	F2
0x8079	8'b10000000	4'b0010	0x4978	FC
0x807A	8'b10000000	4'b0100	0x4978	FE
0x807B	8'b10000000	4'b1000	0x4978	FF
0x807C	8'b10000000	4'b0001	0x497C	FF
0x807D	8'b10000000	4'b0010	0x497C	FF
0x807E	8'b10000000	4'b0100	0x497C	FF
0x807F	8'b10000000	4'b1000	0x497C	DB

Table 6-1. ILK PHY Fixed Register Values

ILK PHY Word Address (Hex)	HI_ADDR[7:0]	BYTE_EN[3:0]	PCIe Addr Offset (Hex)	Value (Hex)
0x8080	8'b10000000	4'b0001	0x4980	E9
0x8081	8'b10000000	4'b0010	0x4980	C2
0x8082	8'b10000000	4'b0100	0x4980	F2
0x8083	8'b10000000	4'b1000	0x4980	FC
0x8084	8'b10000000	4'b0001	0x4984	FE
0x8085	8'b10000000	4'b0010	0x4984	F2
0x8086	8'b10000000	4'b0100	0x4984	F2
0x8087	8'b10000000	4'b1000	0x4984	FF
0x8088	8'b10000000	4'b0001	0x4988	FF
0x8089	8'b10000000	4'b0010	0x4988	DB
0x808A	8'b10000000	4'b0100	0x4988	E9
0x808B	8'b10000000	4'b1000	0x4988	C2
0x808C	8'b10000000	4'b0001	0x498C	F2
0x808D	8'b10000000	4'b0010	0x498C	FC
0x808E	8'b10000000	4'b0100	0x498C	FE
0x808F	8'b10000000	4'b1000	0x498C	F2
0x8090	8'b10000000	4'b0001	0x4990	F2
0x8091	8'b10000000	4'b0010	0x4990	FF
0x8092	8'b10000000	4'b0100	0x4990	FF
0x8093	8'b10000000	4'b1000	0x4990	FF
0x8094	8'b10000000	4'b0001	0x4994	F2
0x8095	8'b10000000	4'b0010	0x4994	63
0x8096	8'b10000000	4'b0100	0x4994	00
0x8097	8'b10000000	4'b1000	0x4994	50
0x8098	8'b10000000	4'b0001	0x4998	00
0x8099	8'b10000000	4'b0010	0x4998	02
0x809A	8'b10000000	4'b0100	0x4998	01
0x809B	8'b10000000	4'b1000	0x4998	05
0x809C	8'b10000000	4'b0001	0x499C	05
0x809D	8'b10000000	4'b0010	0x499C	04
0x809E	8'b10000000	4'b0100	0x499C	00
0x809F	8'b10000000	4'b1000	0x499C	00
0x80A0	8'b10000000	4'b0001	0x49A0	08
0x80A1	8'b10000000	4'b0010	0x49A0	04
0x80A2	8'b10000000	4'b0100	0x49A0	00
0x80A3	8'b10000000	4'b1000	0x49A0	00

6.6.2.1 ILK PHY Lane Reset Register

The ILK PHY Lane Reset register is used to perform a soft or hard reset of the Interlaken RX or TX PHY lanes. The reset may be controlled by the ILK signals or from this register. The resets in this register will only take place if the LANE_RST_CTL bit is set to one.

The table below gives the values needed to configure the "ILK DMA PHY Access Register".

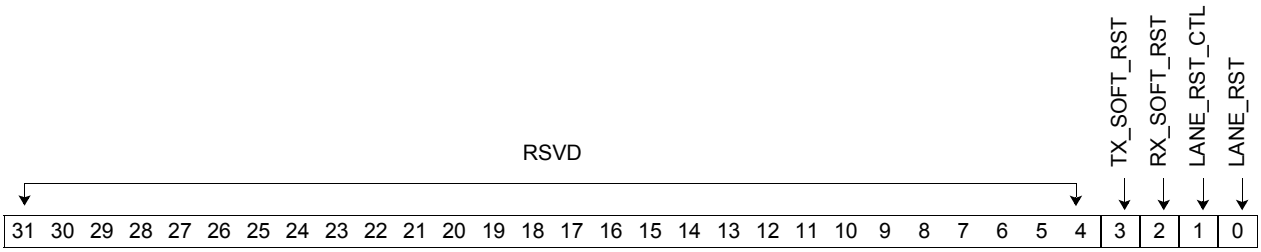
Table 6-2. Values for ILK DMA PHY Access Register

Lane #	ILK PHY Addr	PHY_SEL	HI_ADDR[7:0]	BYTE_EN[3:0]
0	0x0002	1'b0	8'b00000000	4'b0100
1	0x2002	1'b0	8'b00100000	4'b0100
2	0x4002	1'b0	8'b01000000	4'b0100
3	0x6002	1'b0	8'b01100000	4'b0100
4	0x0002	1'b1	8'b00000000	4'b0100
5	0x2002	1'b1	8'b00100000	4'b0100
6	0x4002	1'b1	8'b01000000	4'b0100
7	0x6002	1'b1	8'b01100000	4'b0100

After the indirect addressing values have been configured, the host may access this ILK PHY register using the PCIe address listed below.

Type: Read/Write

PCIe Offset 0x4900 Applies to all lanes



Field Name	Bits	Reset	Description
RSVD	31:4	28'h0000 0000	Reserved.
TX_SOFT_RST_N	3	1'b0	TX Lane Soft Reset. This bit is used to perform a soft reset of the TX PHY. The TX PHY must be in a reset state before writing to its configuration registers. 0 Assert soft reset of TX lane 1 De-assert soft reset of TX lane

Field Name	Bits	Reset	Description
RX_SOFT_RST_N	2	1'b0	RX Lane Soft Reset. This bit is used to perform a soft reset of the RX PHY. The RX PHY must be in a reset state before writing to its configuration registers. 0 Assert soft reset of RX lane 1 De-assert soft reset of RX lane
LANE_RST_CTL	1	1'b0	TX/RX Lane Reset Control. This bit controls whether the reset is initiated by the register or the interface pins. 0 Reset is taken from interface pins 1 Reset is taken from registers
LANE_RST_N	0	1'b0	TX/RX Lane Reset. This active low bit can be used to reset all RX and TX lanes. This bit is only active if LANE_RST_CTL is set to one. 0 Reset all TX/RX lanes 1 Do not reset all RX/RX lanes

6.6.2.2 ILK PHY RX Lane Power Register

The ILK PHY RX Lane Power register is used to control the receiver power state for each ILK lane.

The table below gives the values needed to configure the "ILK DMA PHY Access Register".

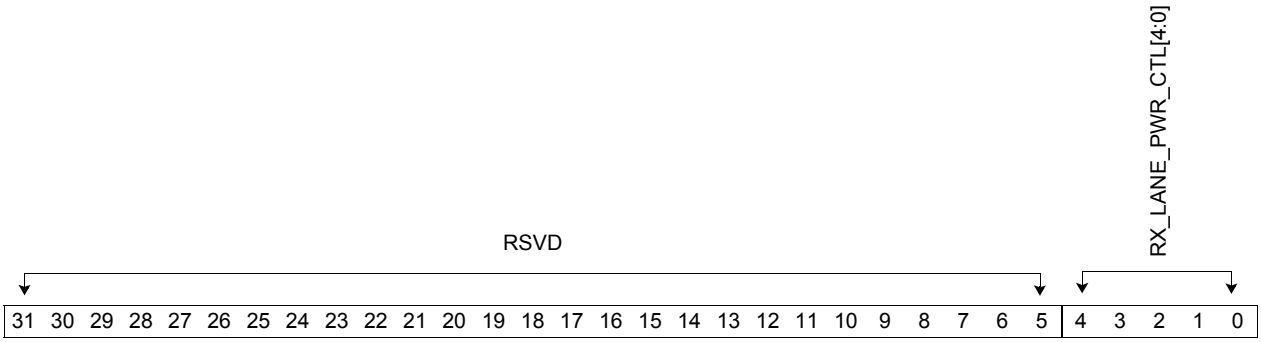
Table 6-3. Values for ILK DMA PHY Access Register

Lane #	ILK PHY Addr	PHY_SEL	HI_ADDR[7:0]	BYTE_EN[3:0]
0	0x0003	1'b0	8'b00000000	4'b1000
1	0x2003	1'b0	8'b00100000	4'b1000
2	0x4003	1'b0	8'b01000000	4'b1000
3	0x6003	1'b0	8'b01100000	4'b1000
4	0x0003	1'b1	8'b00000000	4'b1000
5	0x2003	1'b1	8'b00100000	4'b1000
6	0x4003	1'b1	8'b01000000	4'b1000
7	0x6003	1'b1	8'b01100000	4'b1000

After the indirect addressing values have been configured, the host may access this ILK PHY register using the PCIe address listed below.

Type: Read/Write

PCIe Offset 0x4900 Applies to all lanes



Field Name	Bits	Reset	Description
RSVD	31:5	27'h0000 0000	Reserved.
RX_LANE_PWR_CTL [4:0]	4:0	5'b00001	RX Lane Power State Control. 00001Power Down 10000Power active All other values are reserved.

6.6.2.3 ILK PHY TX Lane Power Register

The ILK PHY TX Lane Power register is used to control the transmitter power state for each ILK lane.

The table below gives the values needed to configure the "ILK DMA PHY Access Register".

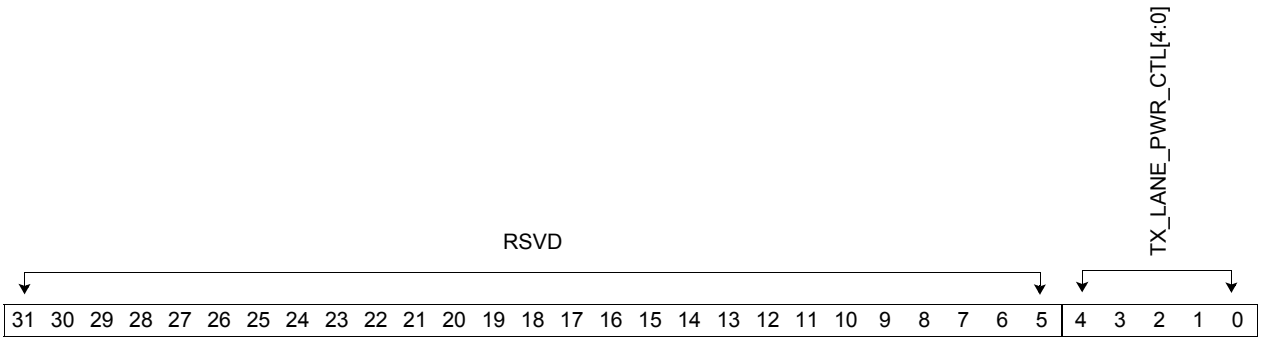
Table 6-4. Values for ILK DMA PHY Access Register

Lane #	ILK PHY Addr	PHY_SEL	HI_ADDR[7:0]	BYTE_EN[3:0]
0	0x0004	1'b0	8'b00000000	4'b0001
1	0x2004	1'b0	8'b00100000	4'b0001
2	0x4004	1'b0	8'b01000000	4'b0001
3	0x6004	1'b0	8'b01100000	4'b0001
4	0x0004	1'b1	8'b00000000	4'b0001
5	0x2004	1'b1	8'b00100000	4'b0001
6	0x4004	1'b1	8'b01000000	4'b0001
7	0x6004	1'b1	8'b01100000	4'b0001

After the indirect addressing values have been configured, the host may access this ILK PHY register using the PCIe address listed below.

Type: Read/Write

PCIe Offset 0x4904 Applies to all lanes



Field Name	Bits	Reset	Description
RSVD	31:5	27'h0000 0000	Reserved.
TX_LANE_PWR_CTL [4:0]	4:0	5'b00001	TX Lane Power State Control. 00001Power Down 10000Power active All other values are reserved.

6.6.2.4 ILK PHY Lane Width Register

The ILK PHY Lane Width register is used to configure the lane width for both the receiver and transmitter for each ILK lane.

The table below gives the values needed to configure the "ILK DMA PHY Access Register".

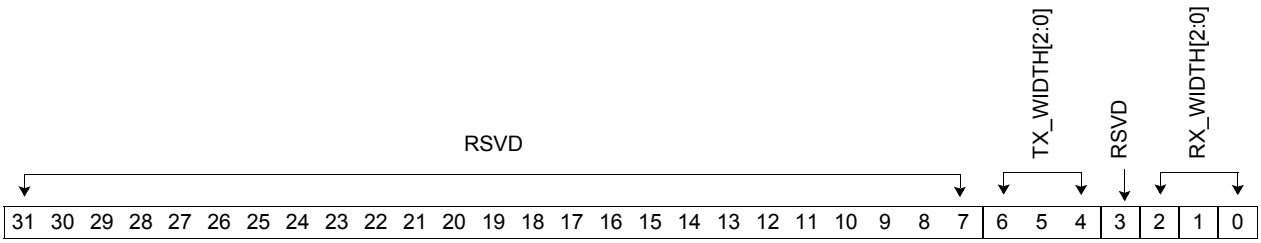
Table 6-5. Values for ILK DMA PHY Access Register

Lane #	ILK PHY Addr	PHY_SEL	HI_ADDR[7:0]	BYTE_EN[3:0]
0	0x0005	1'b0	8'b00000000	4'b0010
1	0x2005	1'b0	8'b00100000	4'b0010
2	0x4005	1'b0	8'b01000000	4'b0010
3	0x6005	1'b0	8'b01100000	4'b0010
4	0x0005	1'b1	8'b00000000	4'b0010
5	0x2005	1'b1	8'b00100000	4'b0010
6	0x4005	1'b1	8'b01000000	4'b0010
7	0x6005	1'b1	8'b01100000	4'b0010

After the indirect addressing values have been configured, the host may access this ILK PHY register using the PCIe address listed below.

Type: Read/Write

PCIe Offset 0x4904 Applies to all lanes



Field Name	Bits	Reset	Description
RSVD	31:7	25'h000 0000	Reserved.
TX_WIDTH[2:0]	6:4	3'b000	TX Lane Width. 0008-bit 001 10-bit 01016-bit 011 20-bit 10032-bit 101 40-bit All other values are reserved.

Field Name	Bits	Reset	Description
RSVD	3	1'b0	Reserved.
RX_WIDTH[2:0]	2:0	3'b000	RX Lane Width. 0008-bit 001 10-bit 01016-bit 011 20-bit 10032-bit 101 40-bit All other values are reserved.

6.6.2.5 ILK PHY Lane Divisor Rate Register

The ILK PHY Lane Divisor Rate register is used to configure the lane speed for both the receiver and transmitter for each ILK lane.

The table below gives the values needed to configure the "ILK DMA PHY Access Register".

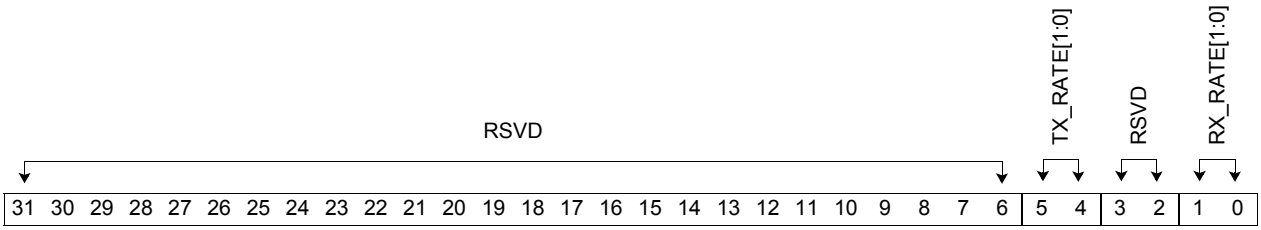
Table 6-6. Values for ILK DMA PHY Access Register

Lane #	ILK PHY Addr	PHY_SEL	HI_ADDR[7:0]	BYTE_EN[3:0]
0	0x0006	1'b0	8'b00000000	4'b0100
1	0x2006	1'b0	8'b00100000	4'b0100
2	0x4006	1'b0	8'b01000000	4'b0100
3	0x6006	1'b0	8'b01100000	4'b0100
4	0x0006	1'b1	8'b00000000	4'b0100
5	0x2006	1'b1	8'b00100000	4'b0100
6	0x4006	1'b1	8'b01000000	4'b0100
7	0x6006	1'b1	8'b01100000	4'b0100

After the indirect addressing values have been configured, the host may access this ILK PHY register using the PCIe address listed below.

Type: Read/Write

PCIe Offset 0x4904 Applies to all lanes



Field Name	Bits	Reset	Description
RSVD	31:6	26'h0000 0000	Reserved.
TX_RATE[1:0]	5:4	2'b01	TX Lane Divisor Rate Select. 00 Divide by 8 01 Divide by 4 10 Divide by 2 11 Divide by 1
RSVD	3:2	2'b00	Reserved.

Field Name	Bits	Reset	Description
RX_RATE[1:0]	1:0	2'b01	RX Lane Divisor Rate Select. 00 Divide by 8 01 Divide by 4 10 Divide by 2 11 Divide by 1

6.6.2.6 ILK PHY Control Register

The ILK PHY Control register is used to configure the ILK PHY data locking signal.

The table below gives the values needed to configure the "ILK DMA PHY Access Register".

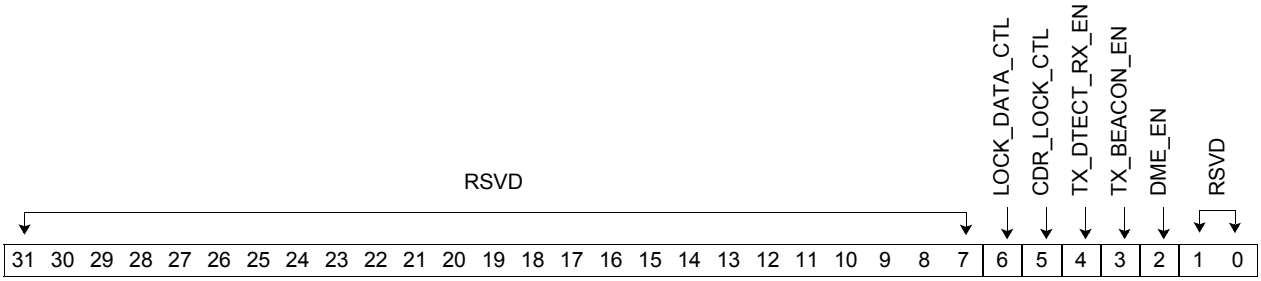
Table 6-7. Values for ILK DMA PHY Access Register

Lane #	ILK PHY Addr	PHY_SEL	HI_ADDR[7:0]	BYTE_EN[3:0]
0	0x0028	1'b0	8'b00000000	4'b0001
1	0x2028	1'b0	8'b00100000	4'b0001
2	0x4028	1'b0	8'b01000000	4'b0001
3	0x6028	1'b0	8'b01100000	4'b0001
4	0x0028	1'b1	8'b00000000	4'b0001
5	0x2028	1'b1	8'b00100000	4'b0001
6	0x4028	1'b1	8'b01000000	4'b0001
7	0x6028	1'b1	8'b01100000	4'b0001

After the indirect addressing values have been configured, the host may access this ILK PHY register using the PCIe address listed below.

Type: Read/Write

PCIe Offset 0x4928 Applies to all lanes



Field Name	Bits	Reset	Description
RSVD	31:7	25'h0000 0000	Reserved.
LOCK_DATA_CTL	6	1'b0	Lock to Data Control. 0 CDR uses its internal locked to data signal 1 CDR uses bit 5 CDR_LOCK_CTL
CDR_LOCK_CTL	5	1'b0	CDR Lock Control. This bit controls whether the CDR should lock to data or lock to reference. 0 CDR is locked to reference 1 CDR is locked to data

Field Name	Bits	Reset	Description
TX_DTECT_RX_EN	4	1'b0	TX Lane Transmitter Detect RX Enable. 0 Disable TX lane transmitter to detect RX 1 Enable TX lane transmitter to detect RX
TX_BEACON_EN	3	1'b0	TX Lane Beacon Enable. 0 Disable TX lane beacon 1 Enable TX lane beacon
DME_EN	2	1'b0	Receiver DME mode enable. 0 Disable DME mode 1 Enable DME mode
RSVD	1:0	2'b00	Reserved

6.6.3 ILK MAC Control Registers

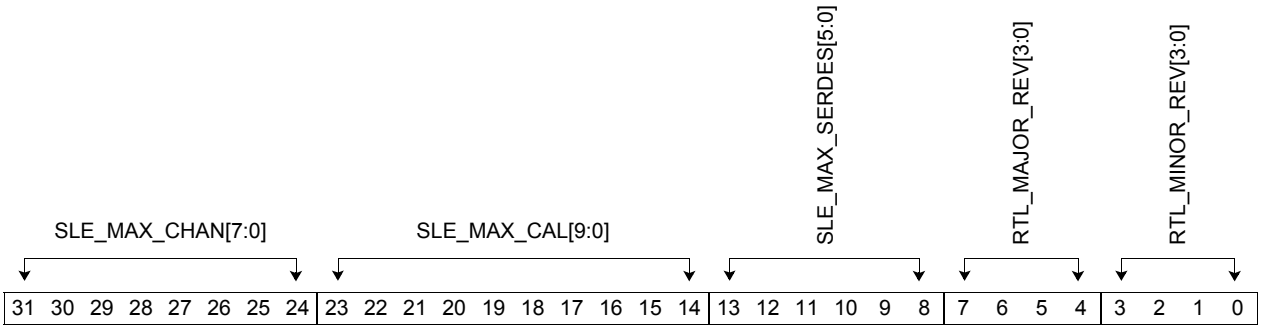
6.6.3.1 ILK MAC Information 1 Register

The ILK MAC Information 1 register contains general information about the ILK interface.

Type: Read only

Offset 0x4C00 TX channel

0x4E00 RX channel



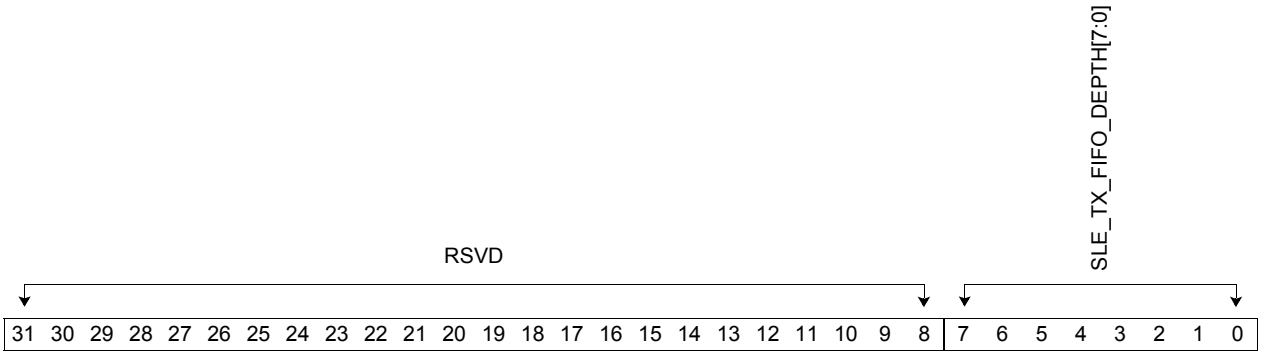
Field Name	Bits	Reset	Description
SLE_MAX_CHAN [7:0]	31:24	8'h12	Maximum Number of Channels. This field reports the number of per channel statistics counters that are kept.
SLE_MAX_CAL[9:0]	23:14	10'h002	Maximum Calculator. This field sets the maximum depth of the non-volatile calculator.
SLE_MAX_SERDES [5:0]	13:8	6'b001000	Maximum Serdes. This field sets the maximum number of Serdes lanes.
RTL_MAJOR_REV [3:0]	7:4	4'b1010	Major Version Number.
RTL_MINOR_REV [3:0]	3:0	4'b0001	Minor Version Number.

6.6.3.2 ILK MAC Information 2 Register

The ILK MAC Information 2 register contains more general information about the ILK interface.

Type: Read only

Offset 0x4C04 TX channel



Field Name	Bits	Reset	Description
RSVD	31:8	24'h00 0000	Reserved.
SLE_TX_FIFO_ DEPTH[7:0]	7:0	8'h04	Transmit FIFO Depth. The TX FIFO depth is fixed at 8'h04.

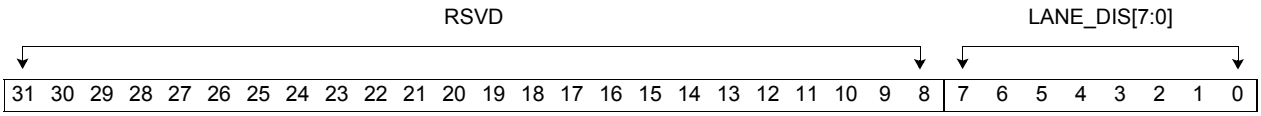
6.6.3.3 ILK MAC Lane Disable Register

This register sets which, if any, ILK Serdes lanes are disabled. The value programmed into this register should be set before releasing the transmitter from reset and then remain unchanged. The reset value enables all available Serdes lanes.

Type: Read/Write

Offset 0x4C08 TX Channel

0x4E08 RX Channel



Field Name	Bits	Reset	Description
RSVD	31:8	24'h00 0000	Reserved.
LANE_DIS[7:0]	7:0	8'h00	Lane Disable. This field is a bitwise control to enable/disable each of the 8 ILK Serdes lanes. 0 Enable 1 Disable

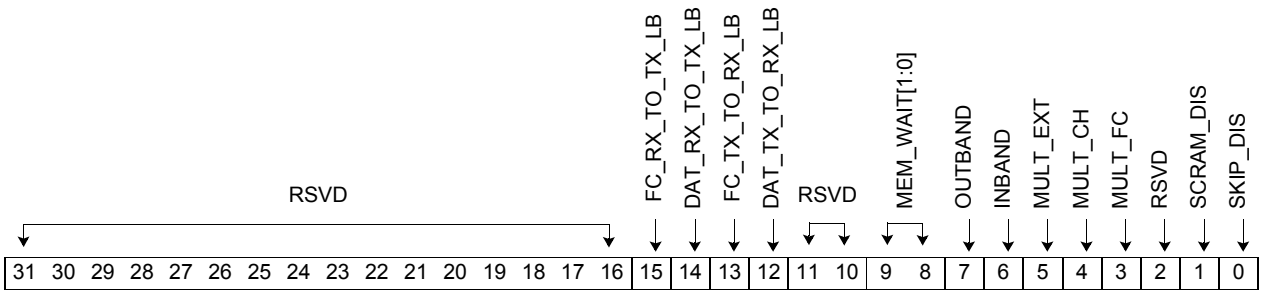
6.6.3.4 ILK MAC Configuration Register

This register configures the basic functions of the ILK transmitter. The values in this register should be set before releasing the transmitter from reset and then remain unchanged.

Type: Read/Write to clear

Offset 0x4C10 TX Channel

0x4E10 RX Channel



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
FC_RX_TO_TX_LB	15	1'b0	Enable Internal Flow Control Loopback. 0 RX-to-TX loopback on the out-of-band flow control path disabled 1 RX-to-TX loopback on the out-of-band flow control path enabled
DAT_RX_TO_TX_LB	14	1'b0	Enable External Data Loopback. 0 RX-to-TX loopback data path disabled 1 RX-to-TX loopback data path enabled
FC_TX_TO_RX_LB	13	1'b0	Enable External Flow Control Loopback. 0 TX-to-RX loopback on the out-of-band flow control path disabled 1 TX-to-RX loopback on the out-of-band flow control path enabled
DAT_TX_TO_RX_LB	12	1'b0	Enable Internal Data Loopback. 0 TX-to-RX loopback data path disabled 1 TX-to-RX loopback data path enabled
RSVD	11:10	2'b00	Reserved.
MEM_WAIT[1:0]	9:8	2'b01	Memory Wait. This field is used to set the number of wait cycles when accessing the statistics counters in memory. 00 Read data on the cycle following read enable/ address 01 Read data two cycles after read enable/address All other values are reserved.

Field Name	Bits	Reset	Description
OUTBAND	7	1'b0	Outbound Flow Control Enable. INBAND and OUTBAND should not be set to 1 at the same time. 0 Outbound flow control is disabled 1 Outbound flow control is enabled
INBAND	6	1'b1	Inbound Flow Control Enable. INBAND and OUTBAND should not be set to 1 at the same time. 0 Inband flow control is disabled 1 Inband flow control is enabled
MULT_EXT	5	1'b0	Multi-Field External Select. This bit must be cleared when the ILK interface is enabled on the XR9240. 0 The multiple-use field is not for XR9240 to FPGA use 1 The multiple-use field is for XR9240 to FPGA use
MULT_CH	4	1'b0	Multi-Field Channel Select. This bit must be set when the ILK interface is enabled on the XR9240. The host must set this bit to zero during initialization. 0 The multiple-use field is not for channel number extension 1 The multiple-use field is for channel number extension
MULT_FC	3	1'b0	Multi-Field Flow Control Select. The XR9240 requires that this bit be clear for correct ILK communication to the FPGA. The host must set this bit to zero during initialization. 0 The multiple-use field is not used for flow control extension 1 The multiple-use field is for flow control extension
RSVD	2	1'b0	Reserved.
SCRAM_DIS	1	1'b0	Scrambler Disable. 0 Scrambler of the 64/67 code is enabled 1 Scrambler of the 64/67 code is disabled
SKIP_DIS	0	1'b0	Skip Disable. This bit is only defined in the transmit direction. 0 Generation of skip words in the metaframe is enabled 1 Generation of skip words in the metaframe is disabled

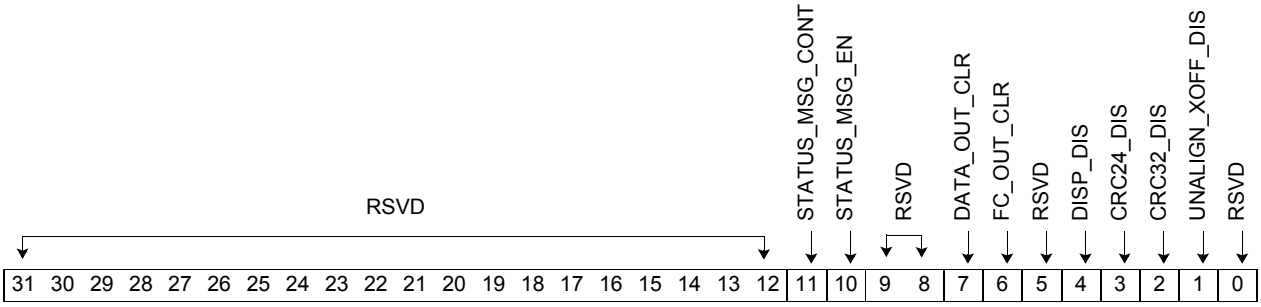
6.6.3.5 ILK MAC Control Register

This register configures the basic functions of the ILK transmitter. The values in this register should be set before releasing the transmitter from reset and then remain unchanged.

Type: Read/Write

Offset 0x4C18 TX Channel

0x4E18 RX Channel



Field Name	Bits	Reset	Description
RSVD	31:12	20'h00 0000	Reserved.
STATUS_MSG_CONT	11	1'b0	Status Message Control. This bit is only valid if STATUS_MSG_EN is set to a one and is only defined for the RX Channel. 0 RX will send the status messaging on the out of band flow control only when one of the enabled lanes identifies a problem 1 RX will send the status messaging on the out of band flow control on every calendar
STATUS_MSG_EN	10	1'b0	Status Message Enable. 0 Optional status messaging on the out of band flow control disabled 1 Optional status messaging on the out of band flow control enabled
RSVD	9:8	2'b00	Reserved.
DATA_OUT_CLR	7	1'b0	Data Output Clear. This field is used to clear the Serdes output data and is typically used in conjunction with TX to RX loopback on the data path which is enabled in the "PK Control Registers". 0 Data output not cleared 1 Serdes output data is forced to all zeros

Field Name	Bits	Reset	Description
FC_OUT_CLR	6	1'b0	Flow Control Output Clear. 0 No change to flow control output 1 TX flow control output is forced to all zeros, regardless of the out-of-band flow control received from the Interlaken Interface
RSVD	5	1'b0	Reserved.
DISP_DIS	4	1'b0	Disparity Disable. 0 The disparity portion of the 64/67 code is enabled 1 The disparity portion of the 64/67 code is disabled
CRC24_DIS	3	1'b0	CRC24 Enable. 0 Control word CRC24 is enabled 1 Control word CRC24 is disabled
CRC32_DIS	2	1'b0	CRC32 Enable. 0 Control word CRC32 is enabled 1 Control word CRC32 is disabled
UNALIGN_XOFF_DIS	1	1'b0	Receiver XOFF Disable. 0 If the receiver is not in the aligned state, XOFF will be sent on the flow control if enabled 1 Receiver alignment state does not affect flow control
RSVD	0	1'b0	Reserved.

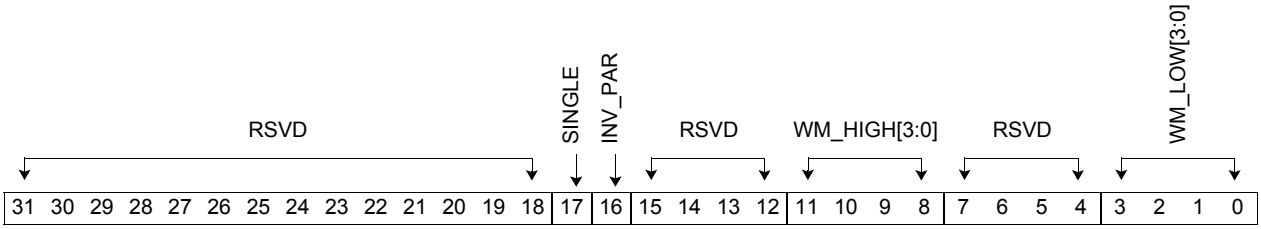
6.6.3.6 ILK MAC Transmit FIFO Configuration Register

This register provides a mechanism for injecting parity errors into the portion of the FIFO that is kept in memory, if any.

The values in this register should be set before releasing the transmitter from reset and then remain unchanged.

Type: Read/Write

Offset 0x4C20 TX Channel



Field Name	Bits	Reset	Description
RSVD	31:18	14'b0000	Reserved.
SINGLE	17	1'b0	Single. This bit is only valid when INV_PAR is set to one. This bit is used for testing the parity function of the transmitter's input FIFO. 0 Continue to invert the parity bits 1 Inject the parity error once then quit
INV_PAR	16	1'b0	Inverse Parity. This bit is used for testing the parity function of the transmitter's input FIFO. After being set, a parity error should be reported when that location is read. 0 No change to parity bits 1 The next data written into the FIFO will have all its parity bits inverted
RSVD	15:12	4'b0000	Reserved.
WM_HIGH[3:0]	11:8	4'b0001	High Water Mark.
RSVD	7:4	4'b0000	Reserved.
WM_LOW[3:0]	3:0	4'b0002	Low Water Mark.

6.6.3.7 ILK MAC Burst Configuration Register

This register is used to set the Interlaken BURSTMAX, BURSTMIN, and BURSTSHORT parameters and enables the functions that use these parameters.

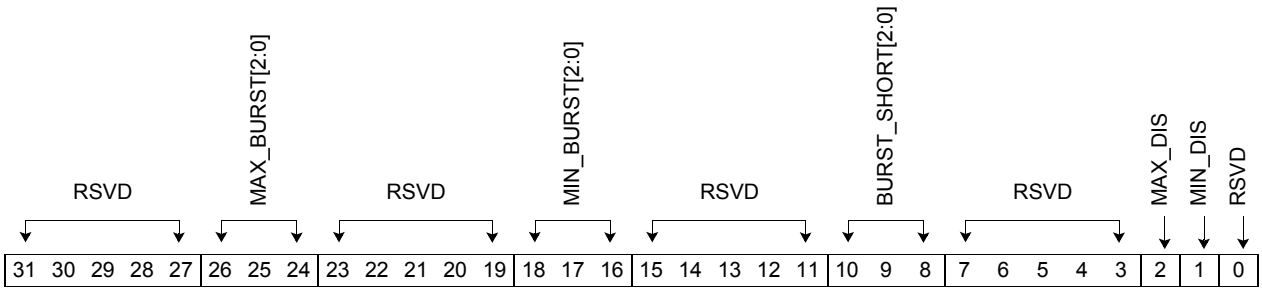
Note that short packets that are smaller than BURSTMIN but are transferred entirely within one burst are not considered to fail the BurstMin check.

The values in this register should be set before releasing the transmitter from reset and then remain unchanged.

Type: Read/Write

Offset 0x4C30 TX Channel

0x4E30 RX Channel



Field Name	Bits	Reset	Description
RSVD	31:27	5'b00000	Reserved.
MAX_BURST[2:0]	26:24	3'b000	Maximum Data Burst Size. This field sets the RX and TX maximum data burst rate. The RX and TX will only verify the burst size against this value if MAX_DIS is set to enabled. 000 = 64 bytes 001 = 128 bytes 010 = 192 bytes 011 = 256 bytes 100 = 320 bytes 101 = 384 bytes 110 = 448 bytes 111 = 512 bytes
RSVD	23:19	5'b00000	Reserved.

Field Name	Bits	Reset	Description
MIN_BURST[2:0]	18:16	3'b000	Minimum Data Burst Size. This field sets the RX and TX minimum data burst rate. The RX and TX will only verify the burst size against this value if MIN_DIS is set to enabled. The value of MIN_BURST[2:0] in bytes must be greater than or equal to BURST_SHORT[2:0] and less than or equal to half of MAX_BURST[2:0]. 000 = 32 bytes 001 = 64 bytes 010 = 96 bytes 011 = 128 bytes 100 = 160 bytes 101 = 192 bytes 110 = 224 bytes 111 = 256 bytes
RSVD	15:11	5'b00000	Reserved.
BURST_SHORT[2:0]	10:8	3'b000	Transmitter Short Data Burst Size. This field sets the short data burst rate and only applies to the TX. The transmit data is padded with idles to ensure that this minimum burst size is met. This field is always enabled. 000 = 32 bytes 001 = 64 bytes 010 = 16 bytes 100 = 24 bytes 111 = 8 bytes All other values are reserved.
RSVD	7:3	5'b00000	Reserved.
MAX_DIS	2	1'b0	Burst Maximum Disable. 0 Burst maximum verification enabled 1 Burst maximum verification disabled
MIN_DIS	1	1'b0	Burst Minimum Disable. If enabled, bursts smaller than MIN_BURST will set a flag in the debug output. 0 Burst minimum verification enabled 1 Burst minimum verification disabled
RSVD	0	1'b0	Reserved.

6.6.3.8 ILK MAC MetaFrame Synchronization Register

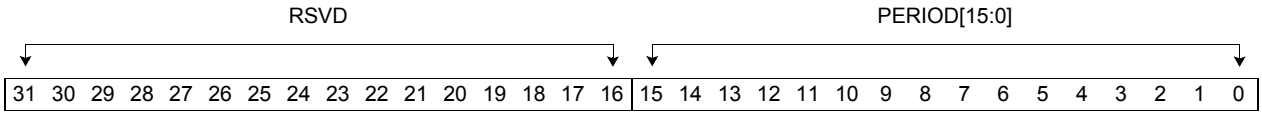
This register contains the MetaFramePeriod timer used to transmit Synchronization Words. One unit of the timer corresponds to 67 bits on the serial lane (64 bits data plus 3 bits framing). The default value corresponds to an interval of 16 Kbytes (not including the 3 bits framing).

The values in this register should be set before releasing the transmitter from reset and then remain unchanged.

Type: Read/Write

Offset 0x4C34 TX Channel

0x4E34 RX Channel



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
PERIOD[15:0]	15:0	16'h0800	Timer Period. This value must be set the same in both the RX and TX. The minimum value is 64 (decimal). Setting the timer to zero in the TX disables the generation of alignment frames.

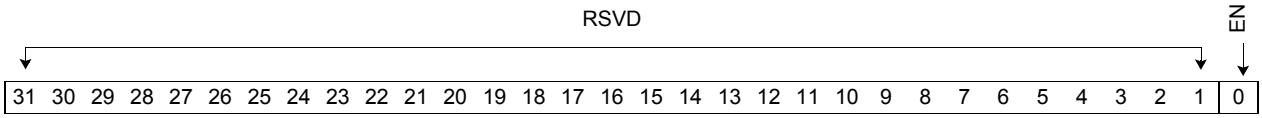
6.6.3.9 ILK MAC Transmit Rate Limiter 1 Register

The XR9240 ILK module implements a proprietary credit system that controls the ILK transmit rate. This register is only included in this document because this feature must be disabled.

The values in this register should be set before releasing the transmitter from reset and then remain unchanged.

Type: Read/Write

Offset 0x4C38 TX Channel



Field Name	Bits	Reset	Description
RSVD	31:1	31'h0000 0000	Reserved.
EN	0	1'b0	Enable. 0 Rate limit function disabled 1 Rate limit function enabled

6.6.3.10 ILK MAC Transmitter Interrupt Register

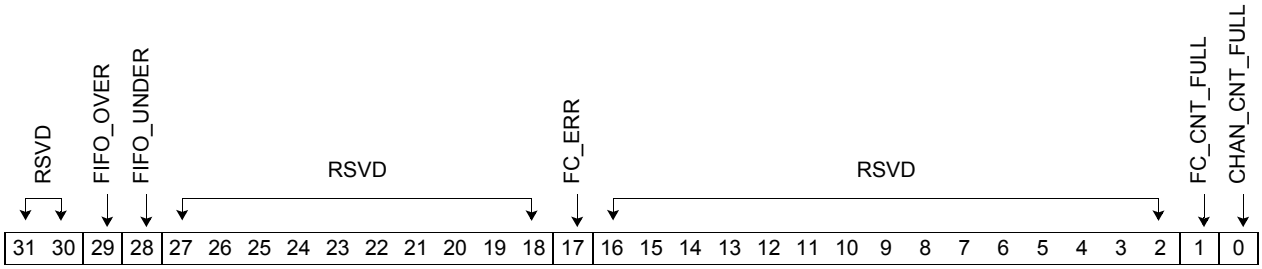
This register stores the ILK MAC Transmitter interrupts. Each bit has an associated mask bit in the "ILK MAC Transmit Interrupt Mask Register", however the bits in this register will be set regardless of the value of their associated mask.

Two interrupt signals are provided by the ILK MAC in the FPGA design: TX_INT1 and TX_INT2. The second output may be used to differentiate between high priority and low priority interrupts. Each of the interrupt outputs has its own interrupt and interrupt mask registers. When a hardware event occurs, the appropriate bit will be set in both interrupt registers.

Type: Read/Write, 1 Clear

Offset 0x4D80 TX Interrupt 1

0x4D88 TX Interrupt 2



Field Name	Bits	Reset	Description
RSVD	31:30	2'b00	Reserved.
FIFO_OVER	29	1'b0	FIFO Overflow Error. This error will occur if the high watermark field in the "ILK MAC Transmit FIFO Configuration Register" is not set properly given the latency required for the user to respond to a stall. 0 No error 1 Transmitter input FIFO overflow error
FIFO_UNDER	28	1'b0	FIFO Underflow Error. This error will not occur if the user sends data in complete bursts and the low watermark in the "ILK MAC Transmit FIFO Configuration Register" is set properly given the latency required for the user to respond to a stall. 0 No error 1 Transmitter input FIFO underflow error
RSVD	27:18	10'h000	Reserved.

Field Name	Bits	Reset	Description
FC_ERR	17	1'b0	Flow Control Error. This error indicates there was a mismatch between the calendar length and reset calendar pulse or CRC4 error. 0 No error 1 Out-of-band flow control error
RSVD	16:2	15'h000	Reserved.
FC_CNT_FULL	1	1'b0	Flow Control Counter Full Error. 0 No error 1 Out-of-band flow control error counter is half full or overflowed
CHAN_CNT_FULL	0	1'b0	Channel Counter Full Error. 0 No error 1 One or more per channel statistics counters are half full or overflowed

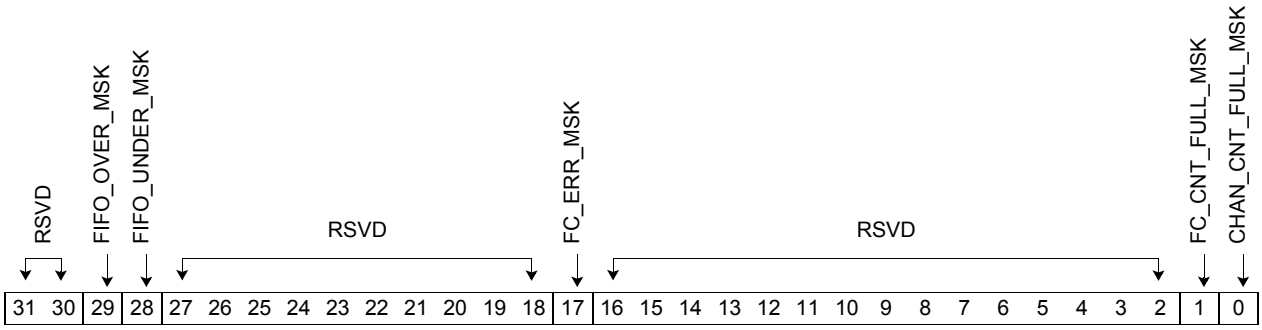
6.6.3.11 ILK MAC Transmit Interrupt Mask Register

This register is used to mask or unmask the ILK interrupts defined in the "ILK MAC Transmitter Interrupt Register". This register determines which interrupts are OR'ed together to create the interrupt outputs.

Type: Read/Write

Offset 0x4D84 Interrupt output 1

0x4D8C Interrupt output 2



Field Name	Bits	Reset	Description
RSVD	31:30	2'b00	Reserved.
FIFO_OVER_MSK	29	1'b1	FIFO Overflow Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
FIFO_UNDER_MSK	28	1'b1	FIFO Underflow Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
RSVD	27:18	10'h000	Reserved.
FC_ERR_MSK	17	1'b1	Flow Control Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
RSVD	16:2	15'h000	Reserved.
FC_CNT_FULL_MSK	1	1'b1	Flow Control Counter Full Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
CHAN_CNT_FULL_MSK	0	1'b1	Channel Counter Full Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked

6.6.3.12 ILK MAC Receiver Interrupt Register

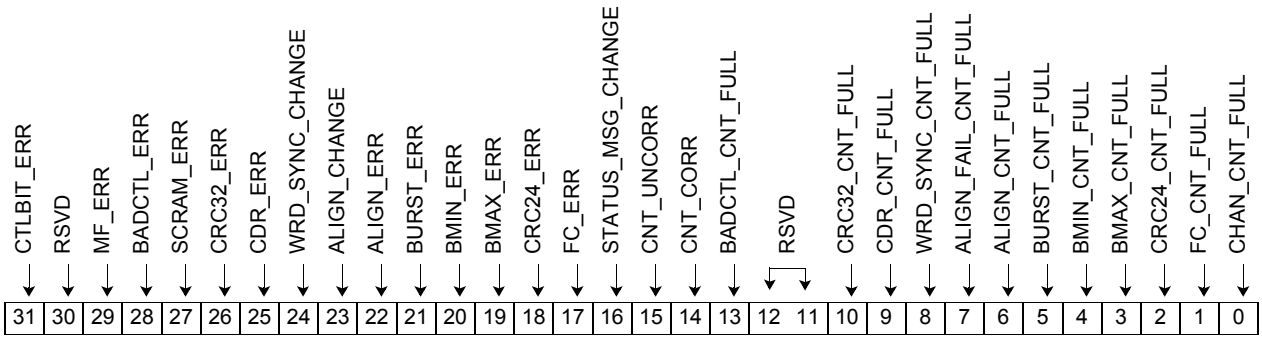
This register stores the ILK Receiver interrupts. Each bit has an associated mask bit in the "ILK MAC Receiver Interrupt Mask Register", however the bits in this register will be set regardless of the value of their associated mask. The bits in this register are cleared by writing a one to the desired bit.

Two interrupt signals are provided by the ILK MAC in the FPGA design: RX_INT1 and RX_INT2. The second output may be used to differentiate between high priority and low priority interrupts. Each of the interrupt outputs has its own interrupt and interrupt mask registers. When a hardware event occurs, the appropriate bit will be set in both interrupt registers. However, the firmware will clear the interrupt only in the register that was written.

Type: Read/Write, 1 Clear

Offset 0x4F80 RX Interrupt 1

0x4F88 RX Interrupt 2



Field Name	Bits	Reset	Description
CTLBIT_ERR	31	1'b0	Control Word Bit Error. The Control word format is defined by the Interlaken protocol. 0 No error 1 A control word appeared to have a valid synchronization or skip word, but bits [55:0] of the word did not match the expected value
RSVD	30	1'b0	Reserved.
MF_ERR	29	1'b0	Metaframe Error. To clear this bit, first write 0x00000000 to address 0x3AC. 0 No error 1 For one or more of the enabled lanes, a valid synchronization word was not received during the metaframe period

Field Name	Bits	Reset	Description
BADCTL_ERR	28	1'b0	Control Word Error. To clear this bit, first write 0x00000000 to address 0x3B0. 0 No error 1 A control word was received that does not match one of the defined control word types
SCRAM_ERR	27	1'b0	Scrambler Error. To clear this bit, first write 0x00000000 to address 0x3B4. 0 No error 1 For one or more enabled lanes, a scrambler state word did not match its expected value
CRC32_ERR	26	1'b0	CRC32 Error. To clear this bit, first write 0x00000000 to address 0x3B8. 0 No error 1 For one or more enabled lanes, a CRC32 error was reported
CDR_ERR	25	1'b0	Clock and Data Recovery Error. To clear this bit, first write 0x00000000 to address 0x3BC. 0 No error 1 For one or more enabled lanes, the i_rx_serdes_valid input is low
WRD_SYNC_CHANGE	24	1'b0	Word Synchronization Change. To clear this bit, first write 0x00000000 to address 0x3C0. 0 No error 1 For one or more enabled lanes, the 64b/67b synchronization has changed, either from out of sync to in sync or from in sync to out of sync
ALIGN_CHANGE	23	1'b0	Alignment State Error. 0 No error 1 The alignment state has changed, either from out of alignment to aligned or from aligned to out of alignment
ALIGN_ERR	22	1'b0	Alignment Error. 0 No error 1 An alignment error occurred (synchronization word not seen on all lanes at the same time)
BURST_ERR	21	1'b0	Burst Error. 0 No error 1 A protocol error occurred in the datapath related to the burst delineation logic
BMIN_ERR	20	1'b0	BurstMin Error. 0 No error 1 BurstMin violation was detected

Field Name	Bits	Reset	Description
BMAX_ERR	19	1'b0	BurstMax Error. 0 No error 1 BurstMax violation was detected
CRC24_ERR	18	1'b0	CRC24 Error. 0 No error 1 CRC24 error was detected
FC_ERR	17	1'b0	Flow Control Error. This error indicates there was a mismatch between the calendar length and reset calendar pulse or CRC4 error. 0 No error 1 Out-of-band flow control error
STATUS_MSG_ CHANGE	16	1'b0	Status Message Change. This field is only used for debugging purposes. 0 No change 1 Diagnostics word's status message changed
CNT_UNCORR	15	1'b0	Uncorrectable Error in Statistic Counter. Each channel has a statistics counter. This bit will be set if any channel reports this error. 0 No error 1 A statistics counter had an uncorrectable parity or ECC error
CNT_CORR	14	1'b0	Correctable Error in Statistic Counter. Each channel has a statistics counter. This bit will be set if any channel reports this error. 0 No error 1 A statistics counter had an correctable parity or ECC error
BADCTL_CNT_FULL	13	1'b0	Receiver Bad Control Counter Warning. 0 No error 1 RX_Bad_Control_Error counter is over half full
RSVD	12:11	2'b00	Reserved.
CRC32_CNT_FULL	10	1'b0	Receiver Lane CRC Counter Warning. 0 No error 1 One or more of the per lane RX_Lane_CRC_Error counters is over half full
CDR_CNT_FULL	9	1'b0	Receiver Lane CDR Counter Warning. 0 No error 1 One or more of the per lane RX_CDR_Error counters is over half full
WRD_SYNC_CNT_ FULL	8	1'b0	Receiver Word Synchronization Counter Warning. 0 No error 1 One or more of the per lane RX_Word_Sync_Error counters is over half full

Field Name	Bits	Reset	Description
ALIGN_FAIL_CNT _FULL	7	1'b0	Receiver Alignment Failure Counter Warning. 0 No error 1 The RX_Alignment_Failure counter is over half full
ALIGN_CNT_FULL	6	1'b0	Receiver Alignment Counter Warning. 0 No error 1 RX_Alignment_Error counter is over half full
BURST_CNT_FULL	5	1'b0	Receiver Burst Counter Warning. 0 No error 1 Burst error counter is over half full
BMIN_CNT_FULL	4	1'b0	Receiver BurstMin Counter Warning. 0 No error 1 BurstMin error counter is over half full
BMAX_CNT_FULL	3	1'b0	Receiver BurstMax Counter Warning. 0 No error 1 BurstMax error counter is over half full
CRC24_CNT_FULL	2	1'b0	In-band Flow Control Counter Warning. 0 No error 1 In-band flow control error counter is over half full
FC_CNT_FULL	1	1'b0	Out-band Flow Control Counter Warning. 0 No error 1 Out-of-band flow control error counter is half full or overflowed
CHAN_CNT_FULL	0	1'b0	Channel Counter Warning. 0 No error 1 One or more per channel statistics counters are half full or overflowed

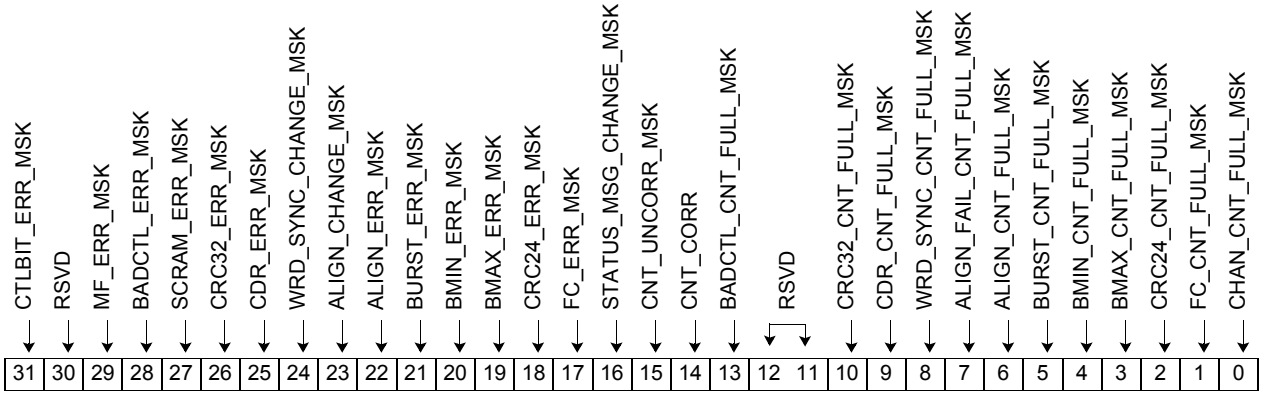
6.6.3.13 ILK MAC Receiver Interrupt Mask Register

This register is used to mask or enable the ILK Receiver interrupts that are listed in the "ILK MAC Receiver Interrupt Register". Upon reset, all interrupts are masked by default.

Type: Read/Write

Offset 0x4F84 Interrupt output 1

0x4F8C Interrupt output 2



Field Name	Bits	Reset	Description
CTLBIT_ERR_MSK	31	1'b1	Control Word Bit Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
RSVD	30	1'b0	Reserved.
MF_ERR_MSK	29	1'b1	Metaframe Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
BADCTL_ERR_MSK	28	1'b1	Control Word Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
SCRAM_ERR_MSK	27	1'b1	Scrambler Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
CRC32_ERR_MSK	26	1'b1	CRC32 Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
CDR_ERR_MSK	25	1'b1	Clock and Data Recovery Error Interrupt Mask 0 Interrupt unmasked 1 Interrupt masked

Field Name	Bits	Reset	Description
WRD_SYNC_CHANGE_MSK	24	1'b1	Word Synchronization Change Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
ALIGN_CHANGE_MSK	23	1'b1	Alignment State Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
ALIGN_ERR_MSK	22	1'b1	Alignment Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
BURST_ERR_MSK	21	1'b1	Burst Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
BMIN_ERR_MSK	20	1'b1	BurstMin Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
BMAX_ERR_MSK	19	1'b1	BurstMax Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
CRC24_ERR_MSK	18	1'b1	CRC24 Error Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
FC_ERR_MSK	17	1'b1	Flow Control Error Interrupt Mask. This error indicates there was a mismatch between the calendar length and reset calendar pulse or CRC4 error. 0 Interrupt unmasked 1 Interrupt masked
STATUS_MSG_CHANGE_MSK	16	1'b1	Status Message Change Interrupt Mask. This field is only used for debug. 0 Interrupt unmasked 1 Interrupt masked
CNT_UNCORR_MSK	15	1'b1	Uncorrectable Error in Statistic Counter Interrupt Mask. Each channel has a statistics counter. This bit will be set if any channel reports this error. 0 Interrupt unmasked 1 Interrupt masked
CNT_CORR_MSK	14	1'b1	Correctable Error in Statistic Counter Interrupt Mask. Each channel has a statistics counter. This bit will be set if any channel reports this error. 0 Interrupt unmasked 1 Interrupt masked

Field Name	Bits	Reset	Description
BADCTL_CNT_FULL_MSK	13	1'b1	Receiver Bad Control Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
RSVD	12:11	2'b00	Reserved.
CRC32_CNT_FULL_MSK	10	1'b1	Receiver Lane CRC Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
CDR_CNT_FULL_MSK	9	1'b1	Receiver Lane CDR Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
WRD_SYNC_CNT_FULL_MSK	8	1'b1	Receiver Word Synchronization Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
ALIGN_FAIL_CNT_FULL_MSK	7	1'b1	Receiver Alignment Failure Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
ALIGN_CNT_FULL_MSK	6	1'b1	Receiver Alignment Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
BURST_CNT_FULL_MSK	5	1'b1	Receiver Burst Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
BMIN_CNT_FULL_MSK	4	1'b1	Receiver BurstMin Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
BMAX_CNT_FULL_MSK	3	1'b1	Receiver BurstMax Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
CRC24_CNT_FULL_MSK	2	1'b1	In-band Flow Control Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked
FC_CNT_FULL_MSK	1	1'b1	Out-band Flow Control Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked

Field Name	Bits	Reset	Description
CHAN_CNT_FULL_MSK	0	1'b1	Channel Counter Warning Interrupt Mask. 0 Interrupt unmasked 1 Interrupt masked

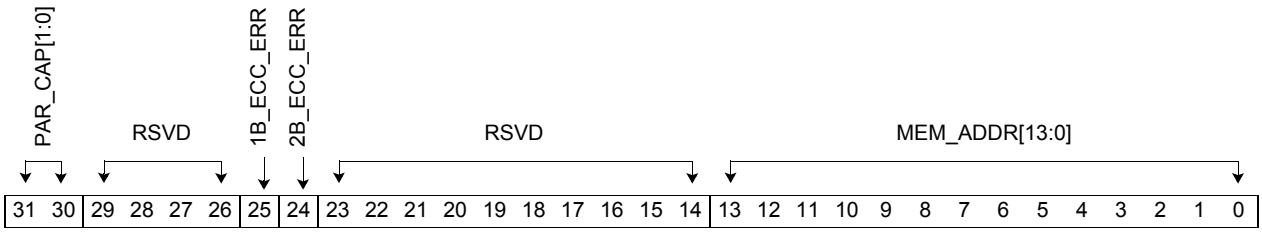
6.7 PK Control Registers

The PK Control registers are used to indirectly address the PK EXP programming registers. There is a set of PK Address, Data and WatchDog Timer registers for each PK module.

6.7.1 PK ECC Error Register

The PK Memory Error register may be used to determine whether single or multi-bit ECC memory errors were detected and corrected. The fields in this register represent the current ECC error state. Refer to the "PK 1-Bit ECC Error Counter Register" and "PK 2-Bit ECC Error Counter Register" for the PK memory ECC error history.

Type:	Read/Write	
Offset	0x5800	PK module 0
	0x5900	PK module 1
	0x5A00	PK module 2
	0x5B00	PK module 3
	0x5C00	PK module 4
	0x5D00	PK module 5
	0x5E00	PK module 6
	0x5F00	PK module 7



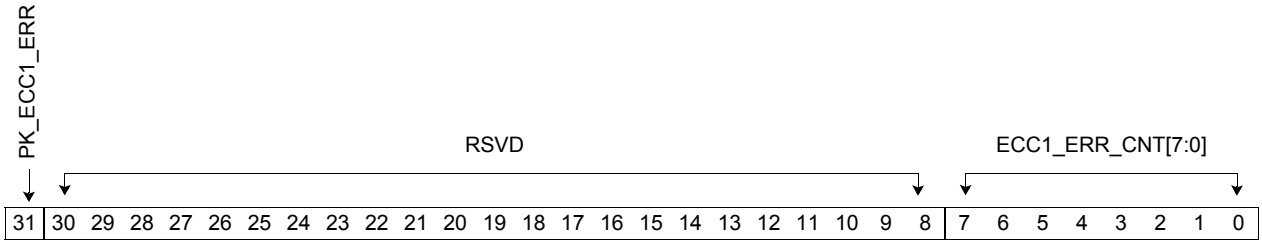
Field Name	Bits	Default	Description
PAR_CAP[1:0]	31:30	2'b10	Parity Capability. This field may be used in conjunction with the 1B_ECC_ERR and 2B_ECC_ERR bits to determine whether a single bit or multi-bit ECC memory error occurred. 00 No parity errors detected 01 Byte parity capable 10 ECC error detected but not corrected 11 Reserved
RSVD	29:26	4'b0000	Reserved.
1B_ECC_ERR	25	1'b0	One Bit ECC Error Detected. 0 No 1-bit ECC errors detected 1 1-bit ECC error detected

Field Name	Bits	Default	Description
2B_ECC_ERR	24	1'b0	Two-Bit ECC Error Detected. 0 No 2-bit ECC errors detected 1 2-bit ECC error detected
RSVD	23:14	10'h000	Reserved.
MEM_ADDR[13:0]	13:0	14'h0000	Memory Address. This field holds the memory word address where the ECC error was detected.

6.7.2 PK 1-Bit ECC Error Counter Register

The PK 1-Bit ECC Error Counter register records the number of 1-bit ECC errors that have occurred in the PK memories.

Type:	Read/Write	
Offset	0x5804	PK module 0
	0x5904	PK module 1
	0x5A04	PK module 2
	0x5B04	PK module 3
	0x5C04	PK module 4
	0x5D04	PK module 5
	0x5E04	PK module 6
	0x5F04	PK module 7

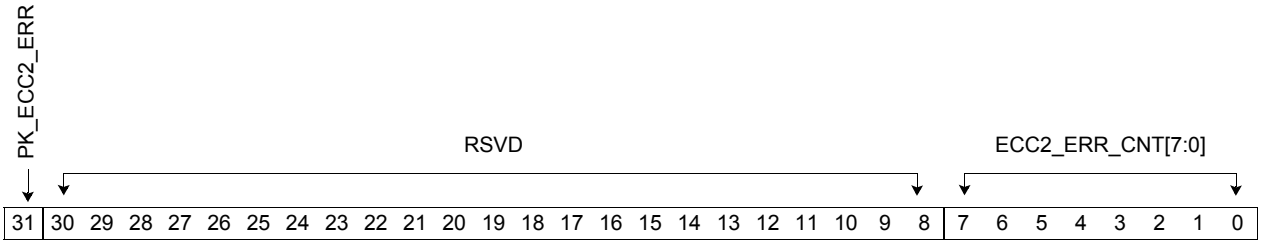


Field Name	Bits	Default	Description
PK_ECC1_ERR	31	1'b0	PK 1-bit ECC Error. This bit is sticky. Once set, it will remain set until cleared by host software. 0 No 1-bit ECC errors detected 1 1-bit ECC error detected
RSVD	30:8	23'h00 0000	Reserved.
ECC1_ERR_CNT[7:0]	7:0	8'h00	PK 1-bit ECC Error Counter. This field reflects the number of 1-bit ECC errors in the PK memory. If the counter overflows, the value read will be 8'hFF. This field may be overwritten.

6.7.3 PK 2-Bit ECC Error Counter Register

The PK 2-Bit ECC Error Counter register records the number of 2-bit ECC errors that have occurred in the PK memories.

Type:	Read/Write	
Offset	0x5808	PK module 0
	0x5908	PK module 1
	0x5A08	PK module 2
	0x5B08	PK module 3
	0x5C08	PK module 4
	0x5D08	PK module 5
	0x5E08	PK module 6
	0x5F08	PK module 7

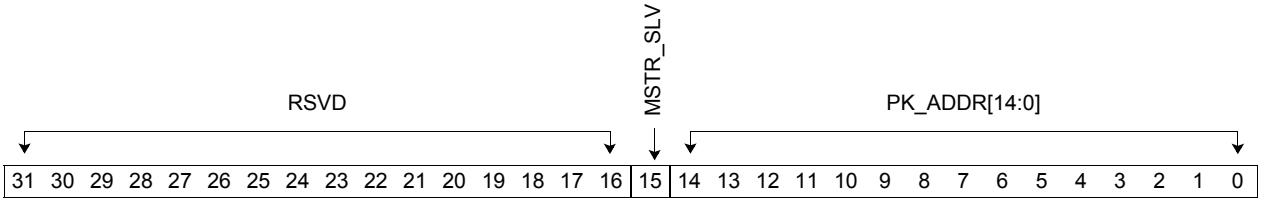


Field Name	Bits	Default	Description
PK_ECC2_ERR	31	1'b0	PK 2-bit ECC Error. This bit is sticky. Once set, it will remain set until cleared by host software. 0 No 2-bit ECC errors detected 1 2-bit ECC error detected
RSVD	30:8	23'h00 0000	Reserved.
ECC2_ERR_CNT[7:0]	7:0	8'h00	PK 2-bit ECC Error Counter. This field reflects the number of 2-bit ECC errors in the PK memory. If the counter overflows, the value read will be 8'hFF. This field may be overwritten.

6.7.4 PK Address Register

Type: Read/Write

Offset		
	0x5820	PK module 0
	0x5920	PK module 1
	0x5A20	PK module 2
	0x5B20	PK module 3
	0x5C20	PK module 4
	0x5D20	PK module 5
	0x5E20	PK module 6
	0x5F20	PK module 7



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
MSTR_SLV	15	1'b0	Master/Slave Select. This bit selects the master/slave EXP module within the PK module. 0 Slave 1 Master
PK_ADDR[14:0]	14:0	15'h0000	PK Indirect Address. This field represents the address of the user descriptor in words. The address defines the access type, as listed in Table 6-8.

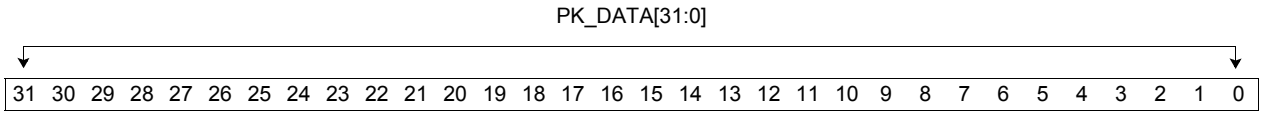
Table 6-8. PK User Descriptor Address Map

Word Address (Hex)	Identifier	Description
0000-03FF	BER	Base and exponent value register
0400-07FF	MMR	Modulus and pre-compute value register
0800-0BFF	TSR	Temporary and square register
0C00-0FFF	FPR	Full product register
1000-17FF	LIR	Linear instruction register
1800	CSR	Control and status register

6.7.5 PK Data Register

The PK Data register holds the read or write data from the PK engine. The "PK Address Register" must be set before accessing this register. A write to this register initiates a write operation to the PK module selected in the "PK Address Register". Likewise, reading this register initiates a read operation to the PK module selected in the "PK Address Register".

Type:	Read/Write	
Offset	0x5824	PK module 0
	0x5924	PK module 1
	0x5A24	PK module 2
	0x5B24	PK module 3
	0x5C24	PK module 4
	0x`D24	PK module 5
	0x5E24	PK module 6
	0x5F24	PK module 7

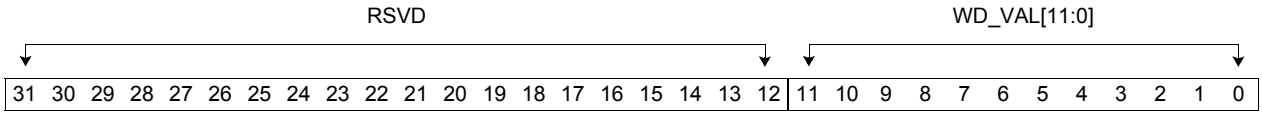


Field Name	Bits	Reset	Description
PK_DATA[31:0]	31:0	32'h0000 0000	PK Indirect Data. This field represents either the read or write data from the PK engine selected in the "PK Address Register".

6.7.6 PK WatchDog Timer Register

The PK WatchDog Timer register is used to set the initial value for the watchdog timer for PK engines. Once enabled, the timer will start counting when a new command is dispatched to a PK module. If the watchdog timer expires, an error will be logged in the destination descriptor. Refer to "Public Key Error ID Codes" for the exact error code.

Type:	Read/Write	
Offset	0x5880	PK module 0
	0x5980	PK module 1
	0x5A80	PK module 2
	0x5B80	PK module 3
	0x5C80	PK module 4
	0x5D80	PK module 5
	0x5E80	PK module 6
	0x5F80	PK module 7



Field Name	Bits	Reset	Description
RSVD	31:24	20'h00 0000	Reserved.
WD_VAL[11:0]	23:0	24'h00 0100	WatchDog Preload Value. This field is the initial value for the watchdog timer. The watchdog preload value unit is 1ms. The reset value of 24'h000100 corresponds to 256ms. A value of 24'h000000 disables the watchdog timer.

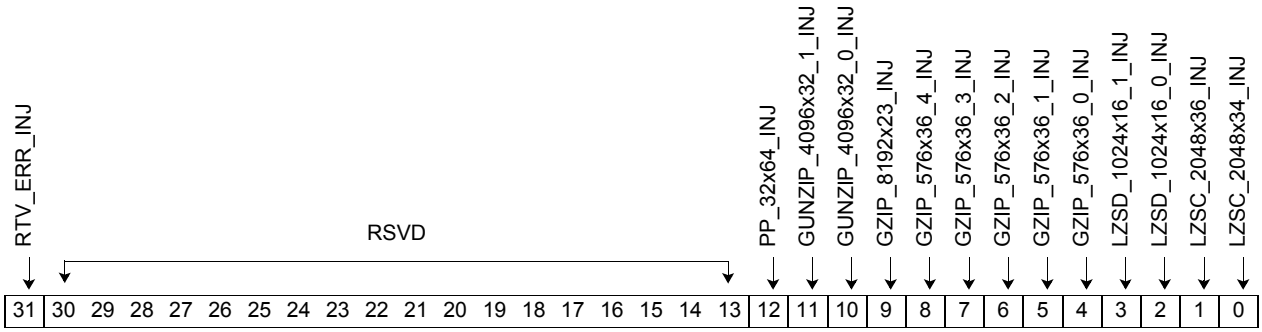
6.8 Transform Engine Control Registers

The XR9240 devices includes eight Transform Engines. Each engine has it own set of control registers.

6.8.1 TE 2-bit ECC Error Injection Register

This register is used for testing. It allows 2-bit ECC errors to be injected into the Transform Engine. The results of the test may be read using the "TE 2-bit ECC Error Injection Log Register".

Type:	Read/Write	
Offset	0x6000	Transform Engine 0
	0x6100	Transform Engine 1
	0x6200	Transform Engine 2
	0x6300	Transform Engine 3
	0x6400	Transform Engine 4
	0x6500	Transform Engine 5
	0x6600	Transform Engine 6
	0x6700	Transform Engine 7



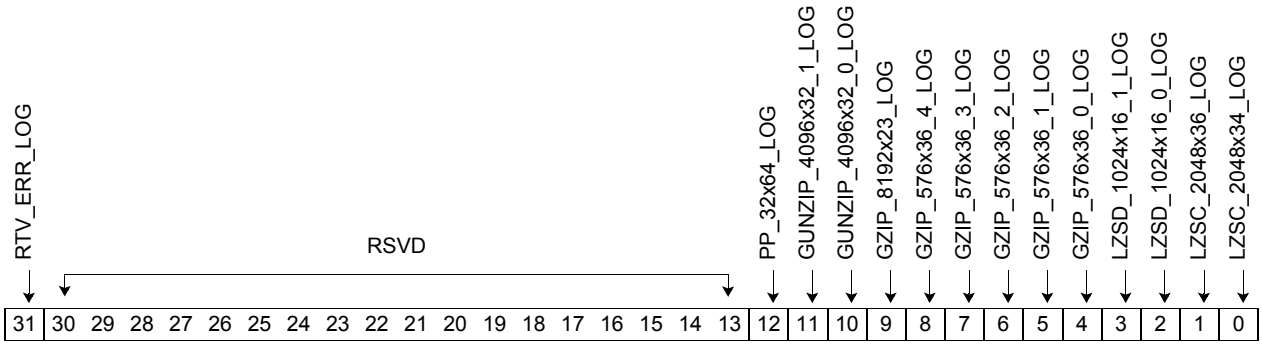
Field Name	Bits	Reset	Description
RTV_ERR_INJ	31	1'b0	Real Time Verification ECC Error Injection. 0 Do not inject RTV ECC error 1 Inject RTV ECC error
RSVD	30:13	18'h00000	Reserved.
PP_32x64_INJ	12	1'b0	Packet Processor 32x64 ECC Error Injection. 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GUNZIP_4096x32_1_INJ	11	1'b0	GunZip 4096x32 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error

Field Name	Bits	Reset	Description
GUNZIP_4096x32_0_INJ	10	1'b0	GunZip 4096x32 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GZIP_8192x23_INJ	9	2'b00	gzip 8192x23 ECC Error Injection. 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GZIP_576x36_4_INJ	8	1'b0	gzip 576x36 ECC Error Injection. (part 4) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GZIP_576x36_3_INJ	7	1'b0	gzip 576x36 ECC Error Injection. (part 3) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GZIP_576x36_2_INJ	6	1'b0	gzip 576x36 ECC Error Injection. (part 2) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GZIP_576x36_1_INJ	5	1'b0	gzip 576x36 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GZIP_576x36_0_INJ	4	1'b0	gzip 576x36 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
LZSD_1024x16_1_INJ	3	1'b0	LZS Decoder 1024x16 ECC Error Injection. (part 1) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
LZSD_1024x16_0_INJ	2	1'b0	LZS Decoder 1024x16 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
LZSC_2048x36_INJ	1	1'b0	LZS Coder 2048x36 ECC Error Injection. 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
LZSC_2048x34_INJ	0	1'b0	LZS Coder 2048x34 ECC Error Injection. 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error

6.8.2 TE 2-bit ECC Error Injection Log Register

This register is used for testing in conjunction with the "TE 2-bit ECC Error Injection Register". It allows the injected 2-bit ECC error log to be read.

Type:	Read/Write	
Offset	0x6004	Transform Engine 0
	0x6104	Transform Engine 1
	0x6204	Transform Engine 2
	0x6304	Transform Engine 3
	0x6404	Transform Engine 4
	0x6504	Transform Engine 5
	0x6604	Transform Engine 6
	0x6704	Transform Engine 7



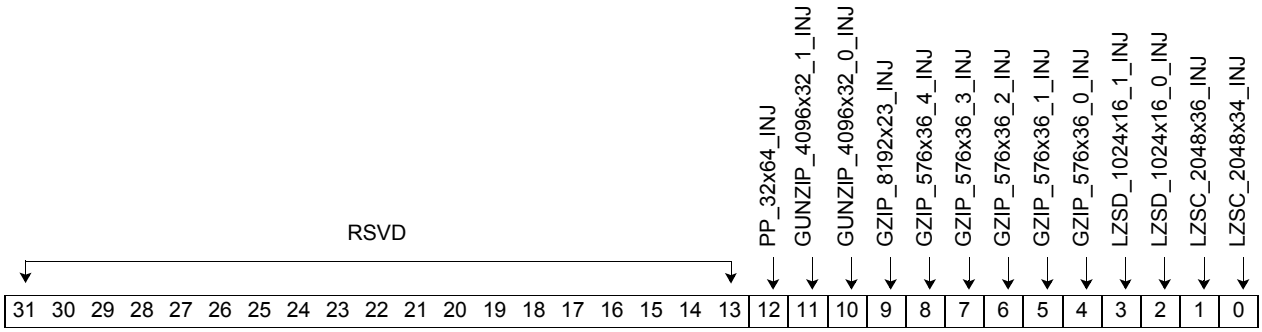
Field Name	Bits	Reset	Description
RTV_ERR_LOG	31	1'b0	Real Time Verification ECC Error Log. 0 No 2-bit ECC error 1 2-bit ECC error set
RSVD	30:13	18'h00000	Reserved.
PP_32x64_LOG	12	1'b0	Packet Processor 32x64 ECC Error Log. 0 No 2-bit ECC error 1 2-bit ECC error set
GUNZIP_4096x32_1_LOG	11	1'b0	GunZip 4096x32 ECC Error Log. (part 1) 0 No 2-bit ECC error 1 2-bit ECC error set
GUNZIP_4096x32_0_LOG	10	1'b0	GunZip 4096x32 ECC Error Log. (part 0) 0 No 2-bit ECC error 1 2-bit ECC error set

Field Name	Bits	Reset	Description
GZIP_8192x23_LOG	9	2'b00	gzip 8192x23 ECC Error Log. 0 No 2-bit ECC error 1 2-bit ECC error set
GZIP_576x36_4_LOG	8	1'b0	gzip 576x36 ECC Error Log. (part 4) 0 No 2-bit ECC error 1 2-bit ECC error set
GZIP_576x36_3_LOG	7	1'b0	gzip 576x36 ECC Error Log. (part 3) 0 No 2-bit ECC error 1 2-bit ECC error set
GZIP_576x36_2_LOG	6	1'b0	gzip 576x36 ECC Error Log. (part 2) 0 No 2-bit ECC error 1 2-bit ECC error set
GZIP_576x36_1_LOG	5	1'b0	gzip 576x36 ECC Error Log. (part 1) 0 No 2-bit ECC error 1 2-bit ECC error set
GZIP_576x36_0_LOG	4	1'b0	gzip 576x36 ECC Error Log. (part 0) 0 No 2-bit ECC error 1 2-bit ECC error set
LZSD_1024x16_1_ LOG	3	1'b0	LZS Decoder 1024x16 ECC Error Log. (part 1) 0 No 2-bit ECC error 1 2-bit ECC error set
LZSD_1024x16_0_ LOG	2	1'b0	LZS Decoder 1024x16 ECC Error Log. (part 0) 0 No 2-bit ECC error 1 2-bit ECC error set
LZSC_2048x36_LOG	1	1'b0	LZS Coder 2048x36 ECC Error Log. 0 No 2-bit ECC error 1 2-bit ECC error set
LZSC_2048x34_LOG	0	1'b0	LZS Coder 2048x34 ECC Error Log. 0 No 2-bit ECC error 1 2-bit ECC error set

6.8.3 TE 1-bit ECC Error Injection Register

This register is used for testing. It allows 1-bit ECC errors to be injected into the Transform Engine. The results of the test may be read using the "TE 1-bit ECC Error Injection Log Register".

Type:	Read/Write	
Offset	0x6008	Transform Engine 0
	0x6108	Transform Engine 1
	0x6208	Transform Engine 2
	0x6308	Transform Engine 3
	0x6408	Transform Engine 4
	0x6508	Transform Engine 5
	0x6608	Transform Engine 6
	0x6708	Transform Engine 7



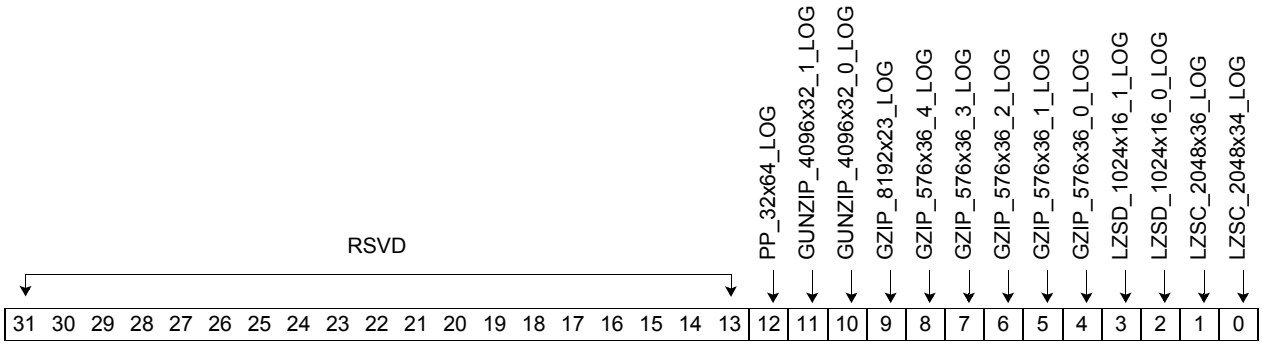
Field Name	Bits	Reset	Description
RSVD	31:13	19'h00000	Reserved.
PP_32x64_INJ	12	1'b0	Packet Processor 32x64 ECC Error Injection. 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
GUNZIP_4096x32_1_INJ	11	1'b0	GunZip 4096x32 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
GUNZIP_4096x32_0_INJ	10	1'b0	GunZip 4096x32 ECC Error Injection. (part 0) 0 Do not inject 2-bit ECC error 1 Inject 2-bit ECC error
GZIP_8192x23_INJ	9	2'b00	gzip 8192x23 ECC Error Injection. 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error

Field Name	Bits	Reset	Description
GZIP_576x36_4_INJ	8	1'b0	gzip 576x36 ECC Error Injection. (part 4) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
GZIP_576x36_3_INJ	7	1'b0	gzip 576x36 ECC Error Injection. (part 3) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
GZIP_576x36_2_INJ	6	1'b0	gzip 576x36 ECC Error Injection. (part 2) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
GZIP_576x36_1_INJ	5	1'b0	gzip 576x36 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
GZIP_576x36_0_INJ	4	1'b0	gzip 576x36 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
LZSD_1024x16_1_INJ	3	1'b0	LZS Decoder 1024x16 ECC Error Injection. (part 1) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
LZSD_1024x16_0_INJ	2	1'b0	LZS Decoder 1024x16 ECC Error Injection. (part 0) 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
LZSC_2048x36_INJ	1	1'b0	LZS Coder 2048x36 ECC Error Injection. 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error
LZSC_2048x34_INJ	0	1'b0	LZS Coder 2048x34 ECC Error Injection. 0 Do not inject 1-bit ECC error 1 Inject 1-bit ECC error

6.8.4 TE 1-bit ECC Error Injection Log Register

This register is used for testing in conjunction with the "TE 1-bit ECC Error Injection Register". It allows the injected 1-bit ECC error log to be read.

Type:	Read/Write	
Offset	0x600C	Transform Engine 0
	0x610C	Transform Engine 1
	0x620C	Transform Engine 2
	0x630C	Transform Engine 3
	0x640C	Transform Engine 4
	0x650C	Transform Engine 5
	0x660C	Transform Engine 6
	0x670C	Transform Engine 7



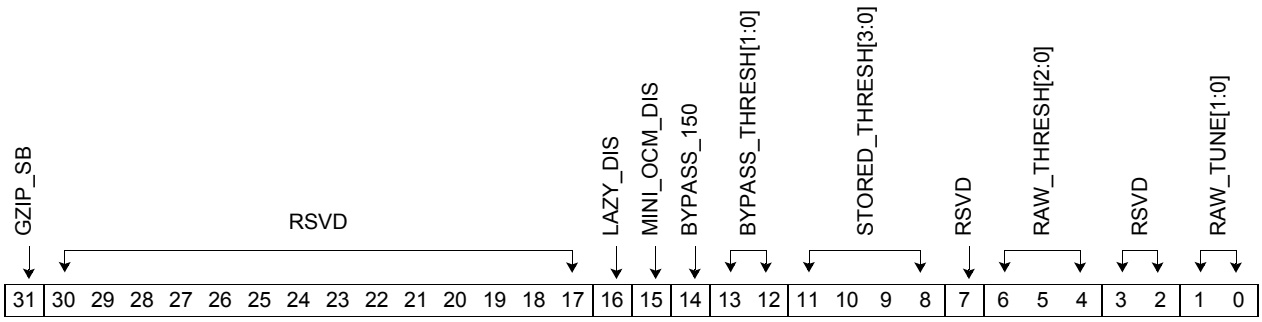
Field Name	Bits	Reset	Description
RSVD	31:13	19'h00000	Reserved.
PP_32x64_LOG	12	1'b0	Packet Processor 32x64 ECC Error Log. 0 No 1-bit ECC error 1 1-bit ECC error set
GUNZIP_4096x32_1_LOG	11	1'b0	GunZip 4096x32 ECC Error Log. (part 1) 0 No 1-bit ECC error 1 1-bit ECC error set
GUNZIP_4096x32_0_LOG	10	1'b0	GunZip 4096x32 ECC Error Log. (part 0) 0 No 1-bit ECC error 1 1-bit ECC error set
GZIP_8192x23_LOG	9	2'b00	gzip 8192x23 ECC Error Log. 0 No 1-bit ECC error 1 1-bit ECC error set

Field Name	Bits	Reset	Description
GZIP_576x36_4_LOG	8	1'b0	gzip 576x36 ECC Error Log. (part 4) 0 No 1-bit ECC error 1 1-bit ECC error set
GZIP_576x36_3_LOG	7	1'b0	gzip 576x36 ECC Error Log. (part 3) 0 No 1-bit ECC error 1 1-bit ECC error set
GZIP_576x36_2_LOG	6	1'b0	gzip 576x36 ECC Error Log. (part 2) 0 No 1-bit ECC error 1 1-bit ECC error set
GZIP_576x36_1_LOG	5	1'b0	gzip 576x36 ECC Error Log. (part 1) 0 No 1-bit ECC error 1 1-bit ECC error set
GZIP_576x36_0_LOG	4	1'b0	gzip 576x36 ECC Error Log. (part 0) 0 No 1-bit ECC error 1 1-bit ECC error set
LZSD_1024x16_1_ LOG	3	1'b0	LZS Decoder 1024x16 ECC Error Log. (part 1) 0 No 1-bit ECC error 1 1-bit ECC error set
LZSD_1024x16_0_ LOG	2	1'b0	LZS Decoder 1024x16 ECC Error Log. (part 0) 0 No 1-bit ECC error 1 1-bit ECC error set
LZSC_2048x36_LOG	1	1'b0	LZS Coder 2048x36 ECC Error Log. 0 No 1-bit ECC error 1 1-bit ECC error set
LZSC_2048x34_LOG	0	1'b0	LZS Coder 2048x34 ECC Error Log. 0 No 1-bit ECC error 1 1-bit ECC error set

6.8.5 gzip Control Register

This register contains control fields that are unique to the gzip algorithm for each Transform Engine.

Type:	Read/Write	
Offset	0x6018	Transform Engine 0
	0x6118	Transform Engine 1
	0x6218	Transform Engine 2
	0x6318	Transform Engine 3
	0x6418	Transform Engine 4
	0x6518	Transform Engine 5
	0x6618	Transform Engine 6
	0x6718	Transform Engine 7



Field Name	Bits	Reset	Description
GZIP_SB	31	1'b0	gzip Sideband Enable. For gzip stateful compression, the flush bits of the last byte are calculated. 0 Disable sideband output 1 Enable sideband output
RSVD	30:17	14'h0000	Reserved.
LAZY_DIS	16	1'b0	Lazy Match Disable. 0 Enable lazy match 1 Disable lazy match
MINI_OCM_DIS	15	1'b0	Mini OCM Disable. 0 Enable Mini OCM 1 Disable Mini OCM

Field Name	Bits	Reset	Description
BYPASS_150	14	1'b0	Bypass 150. This bit enables fallback operations (any operation that allows valid output to be constructed) in the event of a heap ECC error. For the Huffman encoder, the allowed fallback is to use either static Huffman or stored mode for the block. If set, the Bypass Threshold will only be in effect for a block with more than 150 LLD (literal and length/ distance) entries. 0 Fallback operations are not allowed 1 Allow fallback operations to proceed, if available, in the event of a heap ECC error
BYPASS_THRESH[1:0]	13:12	2'b00	Bypass Threshold. This field is used to enable/disable stored block bypass mode, and to determine the threshold of the "static/stored" ratio below which stored blocks will be selected. 00 Disable stored mode bypass 01 Select stored blocks below ratio of 1.125 10 Select stored blocks below ratio of 1.0625 11 Select stored blocks below ratio of 1.03125
STORED_THRESH [3:0]	11:8	4'b0000	Stored Threshold. This field determines the running compression ratio beyond which stored mode will be discounted for the current block and input data will be released. 00002 : 1 0001 1.5 : 1 00101.25 : 1 0011 1.125 : 1 01001.0625 : 1 0101 1.03125 : 1 01101.015625 : 1 0111 1.0078125 : 1 All other values reserved.
RSVD	7	1'b0	Reserved.
RAW_THRESH[2:0]	6:4	3'b010	Raw Threshold. This field enables/disables bias preference for stored blocks, and determines the ratio above which stored blocks will be selected. 000 Disable stored mode biasing 001 - 111 Each unit represents a bias of 0.78125%
RSVD	3:2	2'b00	Reserved.

Field Name	Bits	Reset	Description
RAW_TUNE[1:0]	1:0	2'b11	Raw Tuning of the Compression Ratio. This field determines the page boundary upon which decisions about data compressibility are made. 00 Disable early buffer release 01 512 byte boundary 10 1KB boundary 11 2KB boundary

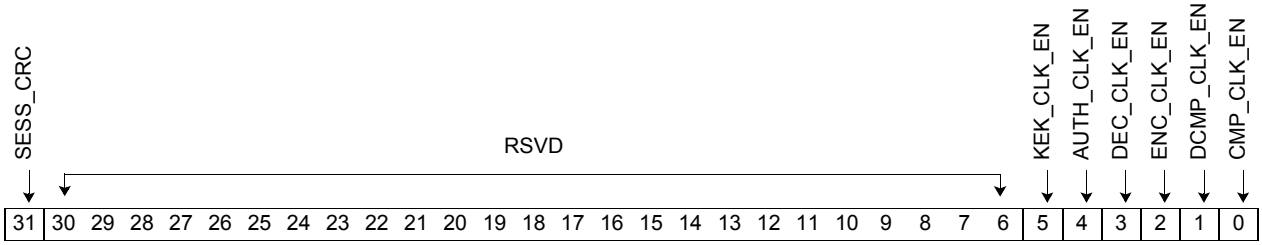
6.8.6 TE Control Register

This register is used to enable session CRC and disable the clock to the KEK, authentication, decryption, and encryption engines in each Transform Engine.

Caution

Sending a command to an engine whose clock is disabled may cause the device to not function properly.

Type:	Read/Write	
Offset	0x601C	Transform Engine 0
	0x611C	Transform Engine 1
	0x621C	Transform Engine 2
	0x631C	Transform Engine 3
	0x641C	Transform Engine 4
	0x651C	Transform Engine 5
	0x661C	Transform Engine 6
	0x671C	Transform Engine 7



Field Name	Bits	Reset	Description
SESS_CRC	31	1'b0	Session CRC Enable. When enabled, the Transform Engine will verify the CRC for each command session. 0 Disable session CRC 1 Enable session CRC
RSVD	30:6	25'h0000	Reserved. The reserved bits in this register must be set to zero.
KEK_CLK_EN	5	1'b1	Key Encryption Key Clock Enable. 0 Disable clock to KEK engine 1 Enable clock to KEK engine

Field Name	Bits	Reset	Description
AUTH_CLK_EN	4	1'b1	Authentication Engine Clock Enable. 0 Disable clock to authentication engine 1 Enable clock to authentication engine
DEC_CLK_EN	3	1'b1	Decryption Engine Clock Enable. 0 Disable clock to decryption engine 1 Enable clock to decryption engine
ENC_CLK_EN	2	1'b1	Encryption Engine Clock Enable. 0 Disable clock to encryption engine 1 Enable clock to encryption engine
DCMP_CLK_EN	1	1'b1	Decompression Engine Clock Enable. 0 Disable clock to decompression engine 1 Enable clock to decompression engine
CMP_CLK_EN	0	1'b1	Compression Engine Clock Enable. 0 Disable clock to compression engine 1 Enable clock to compression engine

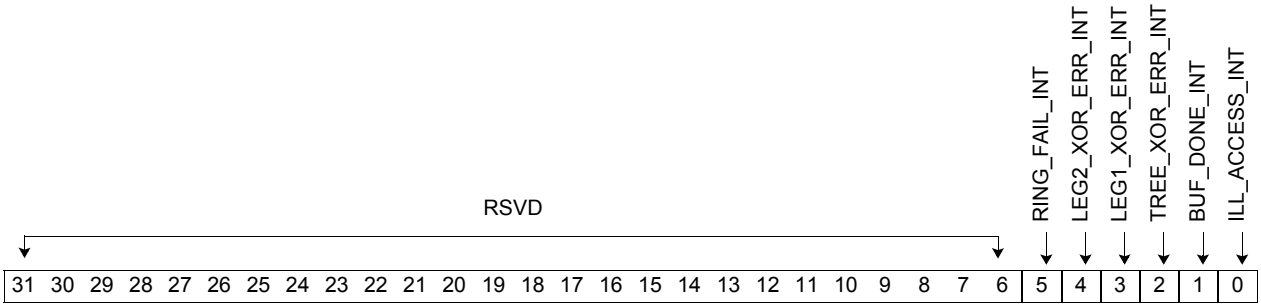
6.9 RNG Control Registers

6.9.1 RNG Interrupt Enable Register

The RNG Interrupt Enable register should be used by the host to enable the RNG interrupts. If the interrupt enable bit is set to 1, the interrupt will be enabled for that event. If the enable bit is cleared to 0, the interrupt will be disabled and will not interrupt the host, but the interrupt status flag in the "RNG Status Register" will function as expected.

Type: Read/Write

Offset: 0x7C04



Field Name	Bits	Reset	Description
RSVD	31:6	26'h0000000	Reserved.
RING_FAIL_INT	5	1'b0	Ring Failure Interrupt Enable. If set, an interrupt will be asserted if any of the ring oscillators are not oscillating. 0 Disable ring failure interrupt 1 Enable ring failure interrupt
LEG2_XOR_ERR_INT	4	1'b0	Leg2 XOR Error Interrupt Enable. If set, an interrupt will be asserted if a Leg2 XOR error occurs. 0 Disable Leg2 XOR error interrupt 1 Enable Leg2 XOR error interrupt
LEG1_XOR_ERR_INT	3	1'b0	Leg1 XOR Error Interrupt Enable. If set, an interrupt will be asserted if a Leg1 XOR error occurs. 0 Disable Leg1 XOR error interrupt 1 Enable Leg1 XOR error interrupt
TREE_XOR_ERR_INT	2	1'b0	Tree XOR Error Interrupt Enable. If set, an interrupt will be asserted if a Tree XOR error occurs. 0 Disable tree XOR error interrupt 1 Enable tree XOR error interrupt

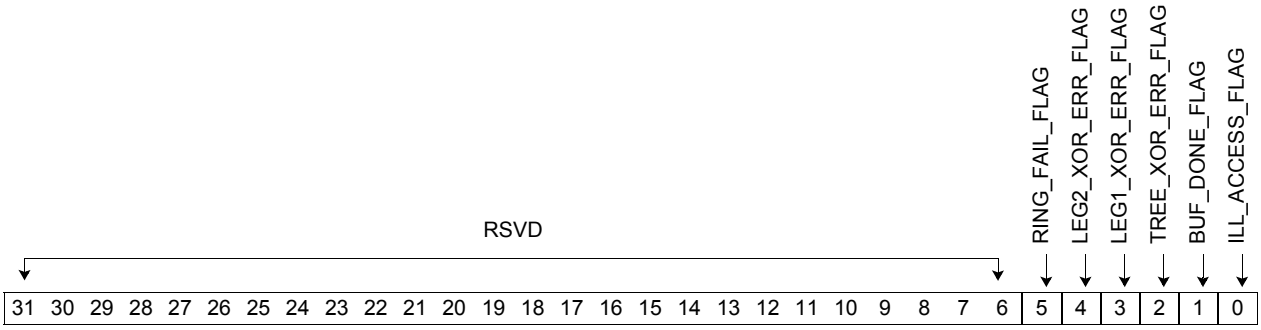
Field Name	Bits	Reset	Description
BUF_DONE_INT	1	1'b0	Buffer Done Interrupt Enable. If set, an interrupt will be asserted if the BUF_DONE_FLAG in the "RNG Status Register" is set. 0 Disable buffer done interrupt 1 Enable buffer done interrupt
ILL_ACCESS_INT	0	1'b0	Illegal Access Interrupt Enable. If set, an interrupt will be asserted if the ILL_ACCESS_FLAG in the "RNG Status Register"is set. 0 Disable illegal access interrupt 1 Enable illegal access interrupt

6.9.2 RNG Status Register

The RNG Status register reflects the status of the RNG module. Writing a one to a flag bit in this register clears the value, whereas writing a zero has no effect. If a flag's corresponding enable bit is set in the "GPIO Interrupt Enable Register", an interrupt will also be generated.

Type: Read/Write 1 to clear

Offset: 0x7C08



Field Name	Bits	Reset	Description
RSVD	31:6	26'h0000 000	Reserved.
RING_FAIL_FLAG	5	1'b0	Ring Failure Flag. 0 No ring failure 1 One for more ring oscillators failed to oscillate
LEG2_XOR_ERR_ FLAG	4	1'b0	Leg2 XOR Error Flag. 0 No Leg2 XOR error 1 XOR has detected an error in the RNG Leg2
LEG1_XOR_ERR_ FLAG	3	1'b0	Leg1 XOR Error Flag. 0 No Leg1 XOR error 1 XOR has detected an error in the RNG Leg2
TREE_XOR_ERR_ FLAG	2	1'b0	Tree XOR Error Flag. 0 No tree XOR error 1 Error detected in the Tree XOR
BUF_DONE_FLAG	1	1'b0	Buffer Done Flag. 0 Buffer not filled 1 Buffer has been filled with random seed
ILL_ACCESS_FLAG	0	1'b0	Illegal Access Flag. 0 Illegal access not detected 1 Access to reserved memory space was detected

6.9.3 RNG Buffer Control Register

This register controls the RNG seed buffer control and provides status information.

This register may be read at any time; writes are allowed only when BUF_BSY is zero, otherwise the access will be flagged as an illegal access.

The sample interval should be set such that at least 32 bits will be sampled between sample intervals for the currently specified bit rate (see "RNG Configuration Register" BIT_RATE[2:0]). Setting the sample interval to a larger value increases the entropy of the random numbers, at the expense of the random number generation rate. The following guideline should be followed for normal operation:

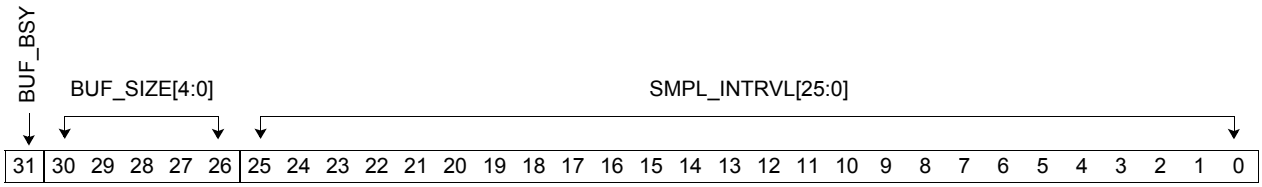
Sample Interval ≥ 32 * (Bit Rate + 1)

A sample interval ≈ 128 * (Bit Rate + 1) is recommended for reasonable entropy versus performance.

Type: Read/Write

Read Only BUF_SIZE[4:0]

Offset 0x7C0C



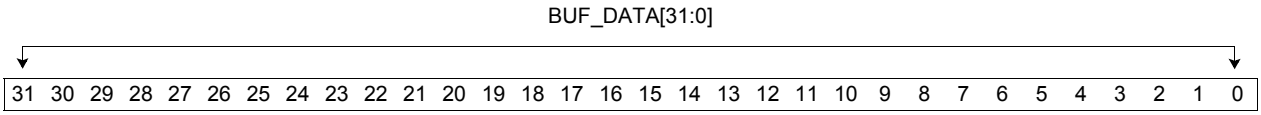
Field Name	Bits	Reset	Description
BUF_BSY	31	1'b0	Buffer Busy. Once set, this bit will remain set until the operation is complete, at which time it will be cleared. 0 Buffer fill complete 1 Fill buffer with 16 new random seeds, regardless of the number of unread seeds that are still in the buffer
BUF_SIZE[4:0]	30:26	5'b00000	Buffer Size. The number of random seeds available in the buffer. This field will be decremented with each valid access to the Buffer Data Register.
SMPL_INTRVL[25:0]	25:0	26'h000 0000	Sample Interval. This field is the initial value for the watchdog timer. Each bit represents a time unit of 1ms.

6.9.4 RNG Buffer Data Register

This register allows the host to read the RNG buffer, one word at a time.

This register is read only and may be read only when the fields in the "RNG Buffer Control Register" BUF_BSY is zero and BUF_SIZE is nonzero, otherwise the read will be flagged as an illegal access. Any writes to this register will be flagged as an illegal access.

Type:	Read only	When BUF_BSY is zero AND BUF_SIZE is nonzero
Offset:	0x7C10	



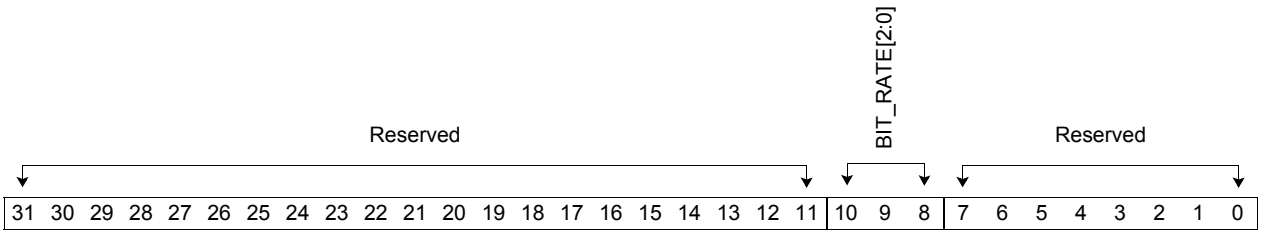
Field Name	Bits	Reset	Description
BUF_DATA[31:0]	31:0	32'h0000 0000	Data from Seed Buffer.

6.9.5 RNG Configuration Register

The RNG Configuration register is used to set the RNG bit rate. The bit rate is used to determine the RNG sample rate, as described in "RNG Buffer Control Register".

Type: Read/write

Offset 0x7C20



Field Name	Bits	Reset	Description
RSVD	31:11	21'h000000	Reserved. All reserved bits in this field must be set to zero.
BIT_RATE[2:0]	10:8	3'b000	Bit Rate. The number of 200 MHz clock cycles between bit samples in the serial-to-parallel conversion / whitening LFSR. 000 200 Mbps (continuous) 001 100 Mbps 010 66.6 Mbps 011 50 Mbps 100 40 Mbps 101 33.3 Mbps 110 28.5 Mbps 111 25 Mbps
RSVD	7:0	8'h00	Reserved. All reserved bits in this field must be set to zero.

6.10 GPIO Control Registers

The GPIO registers are used to configure the XR9240 GPIO interface. The operation sequence for configuring the GPIO registers is:

1. Set the GPIO port data direction to either input or output in the "GPIO Data Direction Register".

2. Configure the interrupt modes, de-bounce, and level sensitive synchronization as needed.

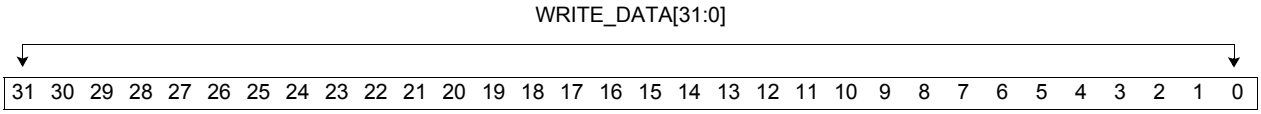
3. Read data from the "GPIO Read Data Register" if the GPIO are configured as input; write data to the "GPIO Write Data Register" if the GPIO are configured as output.

6.10.1 GPIO Write Data Register

The GPIO Write Data register allows the host to write to the XR9240 General Purpose I/O data port. The value written to this register will be output on the GPIO pins if the corresponding data direction bits DATA_DIR[31:0] in the "GPIO Data Direction Register" are set to Output. Reading this register returns the last value written to this register.

Type: Read/Write

Offset 0x7D00



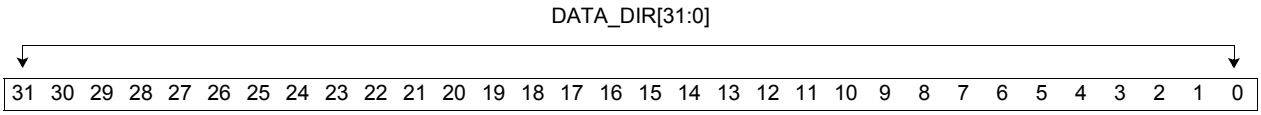
Field Name	Bits	Reset	Description
WRITE_DATA[31:0]	31:0	32'h0000 0000	GPIO Write Data.

6.10.2 GPIO Data Direction Register

The GPIO Data Direction register is used to control the direction of the GPIO data. Each bit may be independently set as either input or output. The default direction is input.

Type: Read/Write

Offset 0x7D04



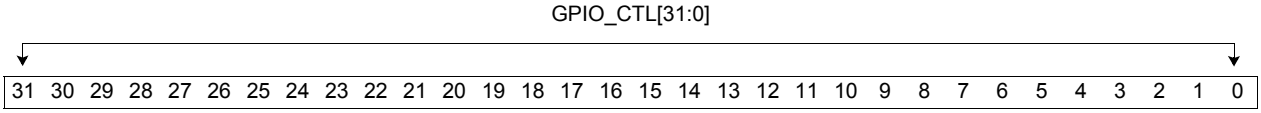
Field Name	Bits	Reset	Description
DATA_DIR [31:0]	31:0	32'h0000 0000	GPIO Data Direction. The direction for each bit can be set as: 0 Input 1 Output

6.10.3 GPIO Control Register

This register is used internally and must be written as all zeroes.

Type: Read/Write

Offset 0x7D08



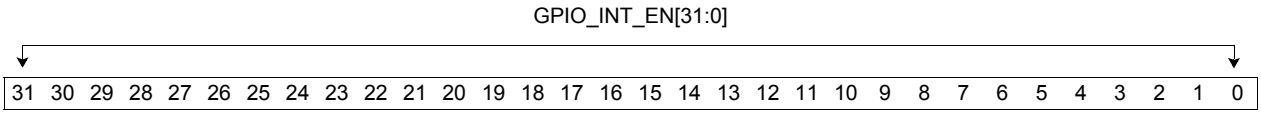
Field Name	Bits	Reset	Description
GPIO_CTL[31:0]	31:0	32'h0000 0000	GPIO Control. This field is used internally and must be written as all zeroes.

6.10.4 GPIO Interrupt Enable Register

The GPIO Interrupt Enable register allows each GPIO bit to be independently configured as an interrupt. By default, using the GPIO ports as interrupts is disabled. If a one is written to a bit of this register, that GPIO bit will become an interrupt. Otherwise, it operates as a normal GPIO port. Interrupts will be disabled if the corresponding "GPIO Data Direction Register" DATA_DIR[31:0] field is set to Output.

Type: Read/Write

Offset 0x7D30



Field Name	Bits	Reset	Description
GPIO_INT_EN [31:0]	31:0	32'h0000 0000	GPIO Interrupt Enable. Each GPIO bit can be configured as: 0 Normal GPIO port (default) 1 Interrupt

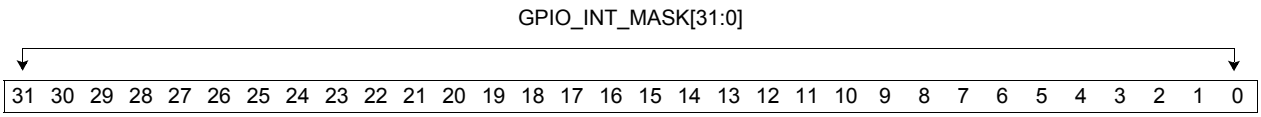
6.10.5 GPIO Interrupt Mask Register

The GPIO Interrupt Mask register is used to mask the GPIO interrupts. By default, all interrupts bits are unmasked. The mask setting for each GPIO interrupt is ignored unless it has been enabled via the "GPIO Interrupt Enable Register".

If a one is written to a bit in this register, that bit will be masked from generating an interrupt. The host may read this register to determine the masked/unmasked status of each bit.

Type: Read/Write

Offset 0x7D34



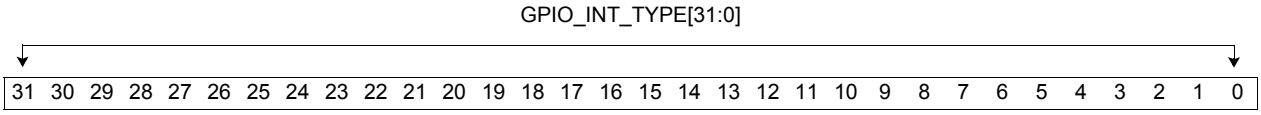
Field Name	Bits	Reset	Description
GPIO_INT_MASK [31:0]	31:0	32'h0000 0000	GPIO Interrupt Mask. Each GPIO bit can be masked as: 0 Unmasked (default) 1 Masked

6.10.6 GPIO Interrupt Type Register

The GPIO Interrupt Type register controls whether the interrupt is level-sensitive or edge-sensitive. Each bit may be independently set.

Type: Read/Write

Offset 0x7D38



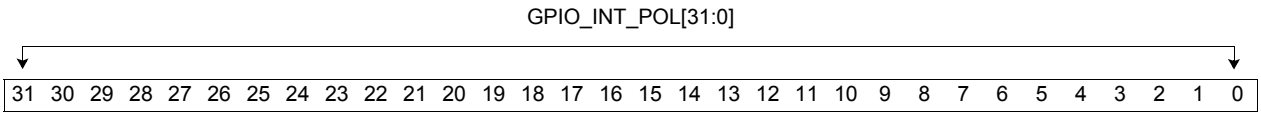
Field Name	Bits	Reset	Description
GPIO_INT_TYPE [31:0]	31:0	32'h0000 0000	GPIO Interrupt Type. Each GPIO bit can be set as: 0 Level-sensitive (default) 1 Edge-sensitive

6.10.7 GPIO Interrupt Polarity Register

The GPIO Interrupt Polarity register controls the polarity of the edge or level sensitivity for each GPIO bit. If a zero is written to a bit of this register, the polarity for that GPIO interrupt bit will be configured as falling-edge or active-low; otherwise, if a one is written, the bit will be configured as rising-edge or active-high. Each bit may be independently set.

Type: Read/Write

Offset 0x7D3C



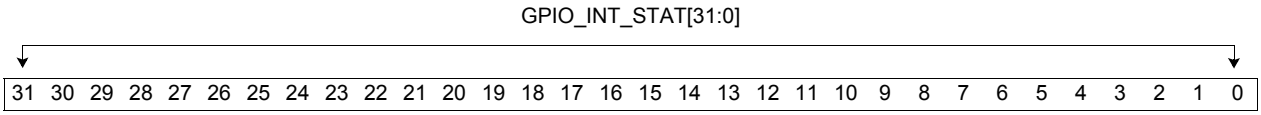
Field Name	Bits	Reset	Description
GPIO_INT_POL [31:0]	31:0	32'h0000 0000	GPIO Interrupt Polarity. Each GPIO bit can be set as: 0 Active-low (default) 1 Active-high

6.10.8 GPIO Interrupt Status Register

The GPIO Interrupt Status register may be read by the host to determine the GPIO interrupt status. This register will reflect any masking that has been set in the "GPIO Interrupt Mask Register".

Type: Read only

Offset 0x7D40



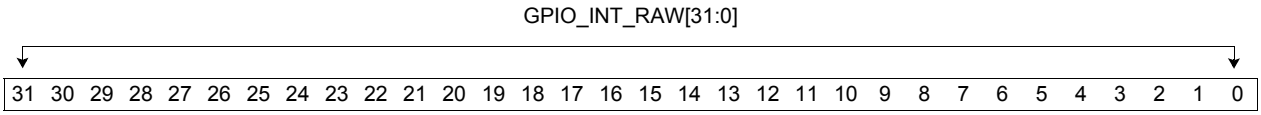
Field Name	Bits	Reset	Description
GPIO_INT_STAT [31:0]	31:0	32'h0000 0000	GPIO Interrupt Status.

6.10.9 GPIO Interrupt Raw Status Register

The GPIO Interrupt Raw Status register may be read by the host to determine the GPIO interrupt status. This register does not reflect any masking that has been set in the "GPIO Interrupt Mask Register".

Type: Read only

Offset 0x7D44



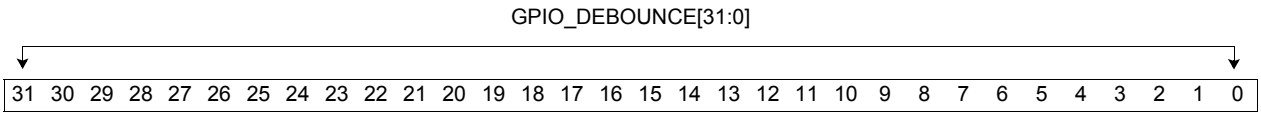
Field Name	Bits	Reset	Description
GPIO_INT_RAW STAT [31:0]	31:0	32'h0000 0000	GPIO Interrupt Raw Status.

6.10.10 GPIO De-Bounce Register

The GPIO De-Bounce register controls whether an external signal that is the source of an interrupt needs to be debounced to remove any spurious glitches. Writing a one to a bit in this register enables the debouncing circuitry. Once enabled, a signal must be valid for two periods of an external clock before it is internally processed.

Type: Read/Write

Offset 0x7D48



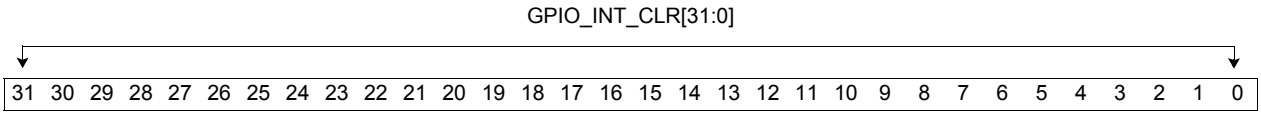
Field Name	Bits	Reset	Description
GPIO_DEBOUNCE [31:0]	31:0	32'h0000 0000	GPIO De-bounce. 0 Disable debounce (default) 1 Enable debounce

6.10.11 GPIO Interrupt Clear Register

The GPIO Interrupt Clear register controls the clearing of edge type interrupts. If a one is written into a bit of this register, the interrupt corresponding to that bit will be cleared.

Type: Read/Write

Offset 0x7D4C



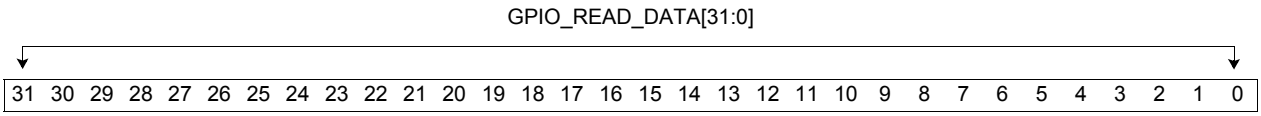
Field Name	Bits	Reset	Description
GPIO_INT_CLR [31:0]	31:0	32'h0000 0000	GPIO Interrupt Clear. 0 Do not clear interrupt (default) 1 Clear interrupt

6.10.12 GPIO Read Data Register

If the data direction of the GPIO port in the "GPIO Interrupt Enable Register" is configured as Input, then this register can be used to read the values on the GPIO signals. When the data direction of GPIO Port is set as Output, reading this location reads the "GPIO Write Data Register".

Type: Read only

Offset 0x7D50



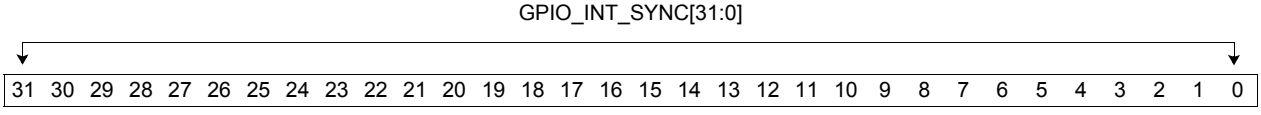
Field Name	Bits	Reset	Description
GPIO_READ_DATA [31:0]	31:0	32'h0000 0000	GPIO Read Data. If Data_DIR = input Read GPIO signals If Data_DIR = output Read data register

6.10.13 GPIO Interrupt Sync Register

The GPIO Interrupt Sync register controls whether level sensitive interrupts are synchronized to an XR9240 internal 250 MHz system clock. Applications that wish to use the GPIO signals as interrupts should enable synchronization for those GPIO signals.

Type: Read/Write

Offset 0x7D60



Field Name	Bits	Reset	Description
GPIO_INT_SYNC [31:0]	31:0	32'h0000 0000	GPIO Interrupt Synchronization. 0 Do not synchronize 1 Synchronize

6.11 SPI Control Registers

Refer to "Serial Peripheral Interface" for an overview of the SPI and for the sequence to access these registers.

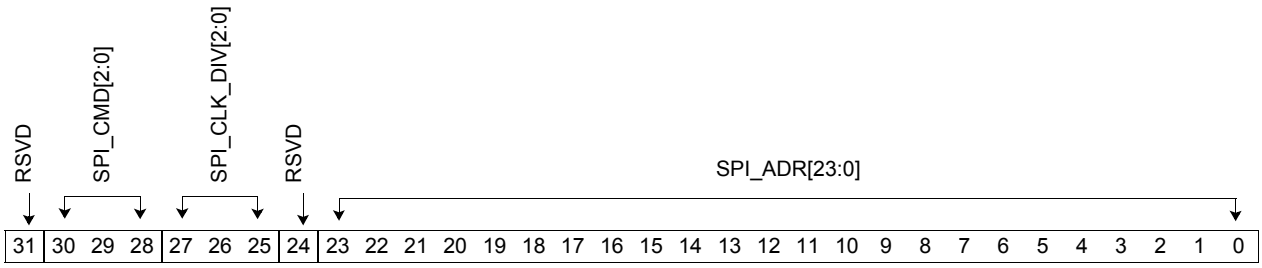
6.11.1 SPI Command Address Register

The SPI Enable register is used by the host to initiate the SPI command. Once the host writes to this register, the XR9240 will issue a SPI write/read operation to the external non-volatile device. For a write operation, the host should write the data to the "SPI Data Register" before writing to this register. For a read operation, the read data will be stored in the "SPI Data Register" after the SPI_OP_Done bit in the "SPI Status Register" is set.

The non-volatile device command definitions in the "SPI Command Configuration 0 Register" and "SPI Command Configuration 1 Register" should have already been confirmed/set before writing to this register.

Type: Read/Write

Offset 0x7E04



Field Name	Bits	Reset	Description
RSVD	31	1'b0	Reserved.

Field Name	Bits	Reset	Description
SPI_CMD[2:0]	30:28	3'b001	Serial Peripheral Interface Command. This field is used to set the type of SPI command. Writing to this register will execute the command. 000 Write 4 bytes of data to the non-volatile device 001 Read 4 bytes of data from the non-volatile device 010 Read Identification (one byte manufacture ID and one byte Device ID) 011 Sector erase 100 Erase device 101 User defined command (the user command is written to the "SPI User Defined Register" and the XR9240 writes the user command to the non-volatile device) 110 WRSR All other values are reserved.
SPI_CLK_DIV[2:0]	27:25	3'b011	SPI Clock Divisor. The SPI clock is equal to 250 MHZ/SPI_CLK_DIV. 000 Reserved 001 Divide by 8 010 Divide by 16 011 Divide by 32 100 Divide by 64 101 Divide by 128 110 Divide by 256 111 Divide by 512
Reserved	24	1'b0	Reserved.
SPI_ADR[23:0]	23:0	24'h00 0000	Non-volatile device Physical Address. This is the sector address if the SPI operation is set to "Sector erase".

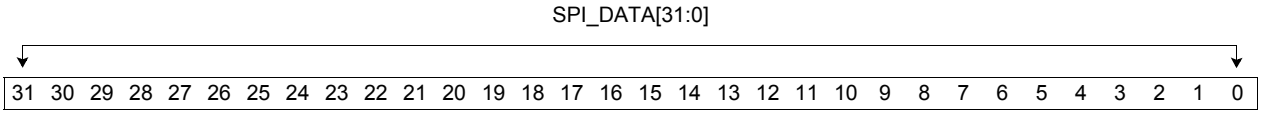
6.11.2 SPI Data Register

The SPI Data register may be used by the host to read and write data through the XR9240 Serial Peripheral Interface.

The MSB must be written to bit [31] and the LSB written to bit [0]. The MSB will be transmitted first on the SPI and LSB will be transmitted last.

Type: Read/Write

Offset 0x7E08



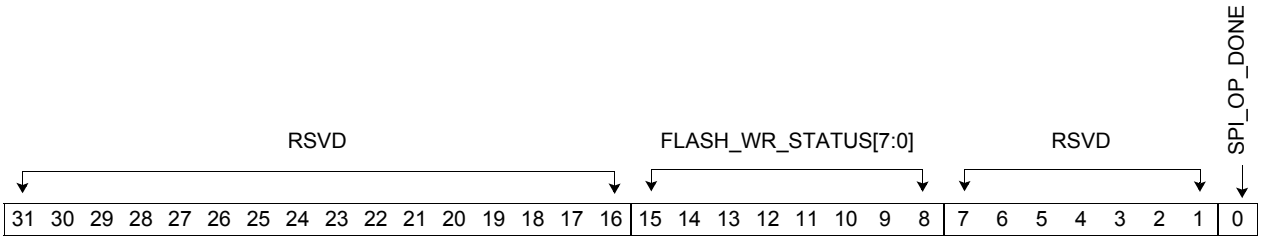
Field Name	Bits	Reset	Description
SPI_DATA[31:0]	31:0	32'h0000 0000	Serial Peripheral Interface Data. For SPI write operations, the host should write the data to this register before writing to the "SPI Command Address Register". For SPI read operations, the XR9240 will write the data to this register when the operation is done.

6.11.3 SPI Status Register

The SPI Status register may be read by the host to determine when a SPI read or write operation on the Serial Peripheral Interface has completed. This register may also be used to read the non-volatile device's write status.

Type: Read/Write

Offset 0x7E0C



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
FLASH_WR_STATUS [7:0]	15:8	8'h00	Non-volatile device Write Status. This field is only valid for 4 byte *write* operations. This field is a copy of the non-volatile device's status register. Please refer to the non-volatile device's specification for details regarding this field. The XR9240 will read this field to determine if the write operation has completed.
RSVD	7:1	7'b000 0000	Reserved.
SPI_OP_DONE	0	1'b0	SPI Operation Done. This bit is read only from the host. When a SPI read/write operation is done, the XR9240 will assert this signal. When the host writes to the "SPI Command Address Register", the XR9240 will automatically clear this bit. 0 SPI read/write operation not done 1 SPI read/write operation done

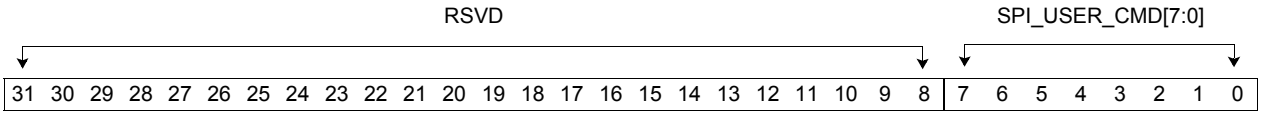
6.11.4 SPI User Defined Register

The SPI User Defined register may be used by the host to write custom commands to the device(s) attached to the Serial Peripheral Interface. Please refer to the specifications for the attached devices for a list of predefined commands.

For example, if a Spansion S25FL064A device is attached to the SPI interface, the user may issue a deep power down mode command (B9h). The host would write 'B9h' to this field, and then set SPI_CMD to '100' in the "SPI Command Address Register". The XR9240 will then write the power down mode command to the non-volatile device.

Type: Read/Write

Offset 0x7E10



Field Name	Bits	Reset	Description
RSVD	31:8	24'h00 0000	Reserved.
SPI_USER_CMD [7:0]	7:0	8'h00	SPI User Defined Command. This field may be used by the host to write any valid command defined by the device attached to the SPI interface.

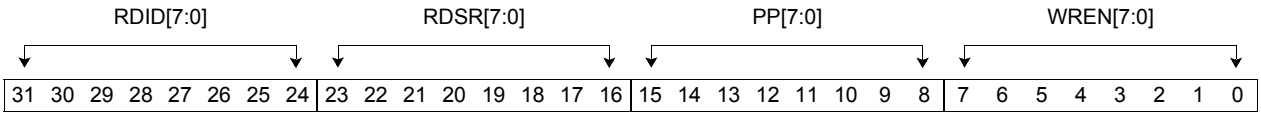
6.11.5 SPI Command Configuration 0 Register

The SPI Command Configuration Register 0 may be used by the host to store the command definitions that the XR9240 will use to access the external non-volatile device. The default values of this register apply to the recommended non-volatile devices (see the *XR9240 Hardware Design User Guide*, USR-0032). The user should confirm that the default values are compatible with the non-volatile device and initialize this register accordingly.

If not using one of the recommended non-volatile devices, the host must set the command definitions for the non-volatile device in this register. This register simply holds the non-volatile device command definitions; the command will not be executed until writing to the "SPI Command Address Register".

Type: Read/Write

Offset 0x7E14



Field Name	Bits	Reset	Description
RDID[7:0]	31:24	8'h9F	Read Identification. The command definition to read the non-volatile device's vendor ID and device ID. The default value is the command definition for the recommended devices.
RDSR[7:0]	23:16	8'h05	Read Status Command. The command definition to read the non-volatile device's status register. The default value is the command definition for the recommended devices.
PP[7:0]	15:8	8'h02	Page Program command. The command definition for a page program command on the non-volatile device. The default value is the command definition for the recommended devices.
WREN[7:0]	7:0	8'h06	Write Enable. The command definition to set the write enable for the non-volatile device. The default value is the command definition for the recommended devices.

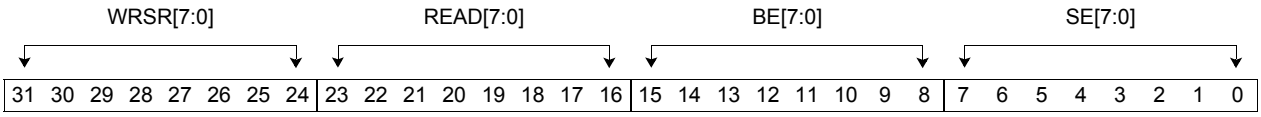
6.11.6 SPI Command Configuration 1 Register

The SPI Command Configuration Register 1 can be used by the host to store the command definitions that the XR9240 will use to access the external non-volatile device. The default values of this register apply to the recommended non-volatile devices. The user should confirm that the default values are compatible with the non-volatile device and initialize this register accordingly.

If not using one of the recommended non-volatile devices, the host must set the command definitions for the non-volatile device in this register. This register simply holds the non-volatile device command definitions; the command will not be executed until writing to the "SPI Command Address Register".

Type: Read/Write

Offset 0x7E18



Field Name	Bits	Reset	Description
WRSR[7:0]	31:24	8'h01	Write Status Register. The command definition to write to the non-volatile device's status register. The default value is the command definition for the recommended devices.
READ[7:0]	23:16	8'h03	Read Command. The command definition to read from the non-volatile device. The default value is the command definition for the recommended devices.
BE[7:0]	15:8	8'hC7	Bulk Erase. The command definition to erase the entire non-volatile device. The default value is the command definition for the recommended devices.
SE[7:0]	7:0	8'h20	Sector Erase. The command definition to erase the a sector from the non-volatile device. The default value is the command definition for the recommended devices.

6.12 Ring Registers

Each of the 128 rings has an associated set of ring registers. The ring registers should be programmed by the host during initialization to set the command pointer ring, result ring, free pool ring address and pointers, as well as configure the ring settings.

Address byte alignment restrictions must be adhered to. Rather than truncating bits to achieve the alignment, the XR9240 required the full address so that the alignment requirements may be verified. The XR9240 will report a ALIGN_ERR interrupt (see "PCIe Interrupt Source Register") to the host if an alignment error is detected.

The physical and virtual function ring registers share the same memory space address but are mapped into physically separate registers within the XR9240 based on the function ID. The physical function may have up to 128 rings and each of the 128 virtual functions may have one ring. For all ring registers, the address for physical functions contains a variable R that represents the ring number.

The behavior of a VF BAR is the same as the PCI Local Bus Specification 3.0 normal PCI memory space BARs, except that a VF BAR describes the aperture for each VF, whereas a PCI BAR describes the aperture for a single function. The following excerpt from the PCIe SR-IOV standard, rev 0.9 specification is included here for reference.

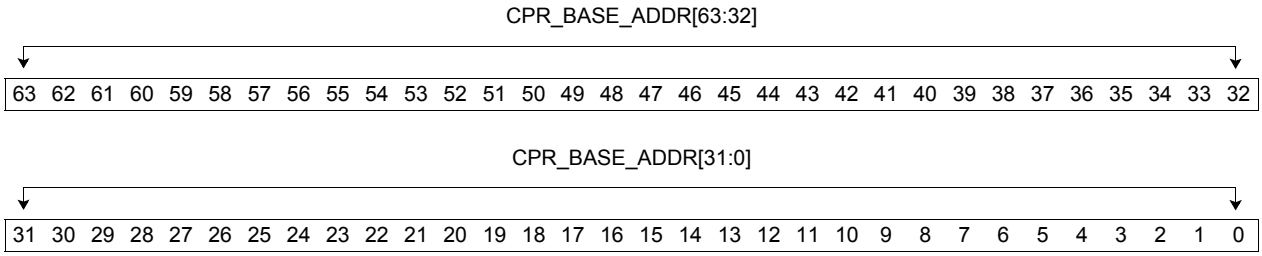
> The behavior described in the PCI Local Bus Specification 3.0 for determining the memory aperture of a Function's BAR applies to each VF BAR. That is, the size of the memory aperture required for each VF BAR can be determined by writing all "1"s and then reading the VF BAR. The results read back must be interpreted as described in the PCI Local Bus Specification.
>
> The behavior for assigning the starting memory space address of each BAR associated with the first VF is also as described in the PCI Local Bus Specification 3.0. That is, the address written into each VF BAR is used by the Device for the starting address of the first VF.
>
> The difference between VF BARs and PCI Local Bus Specification 3.0 BARs is that for each VF BAR, the memory space associated with the second and higher VFs is derived from the starting address of the first VF and the memory space aperture. That is, BAR1 VF N SA = BAR1 VF1 SA + (N – 1) x (BAR1 VF MA); where BAR1 VF 1 SA is the address written into VF BAR1 and BAR1 VF MA is the memory aperture of VF BAR1. Additionally, VF memory space is not enabled until both VF Enable and VF MSE have been Set. Note that reading the BARs before software has configured the PF's System Page Size may not return the final aperture, because the PF may change its aperture based on the System Page Size value.

6.12.1 Command Pointer Ring Registers

6.12.1.1 Command Pointer Ring Base Address Register

This register is the base address of the command pointer ring (CPR) in host memory. The XR9240 uses this base address plus an offset to fetch a command pointer, and then uses the command pointer to fetch the corresponding command structure. The CPR base address must be 8-byte aligned, and the command pointer must be 128-byte aligned.

Type:	Read/Write	
Offset for Physical Function:	0x8000 + 0x40*R	where R = Ring number (0-127)
Offset for Virtual Function:	0x8000	



Field Name	Bits	Reset	Description
CPR_BASE_ADDR [63:0]	63:0	64'h0000 0000	Command Pointer Ring Base Address.

6.12.1.2 Command Pointer Ring Write Pointer Register

This register is a copy of the command pointer ring write pointer in host memory. The host must update this register whenever new commands are added into the command pointer ring.

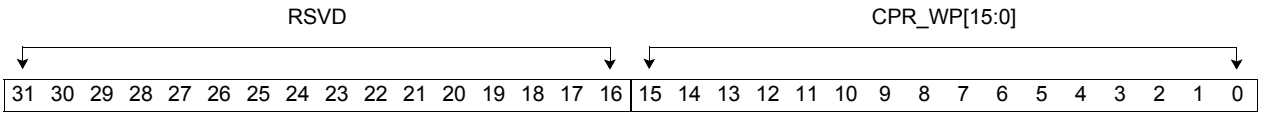
The XR9240 Command Ring Write Pointer register stores the index number (i.e., 1, 2, 3…) of the corresponding command ring write pointer in host memory. The host updates this register whenever a new command is submitted to Command Ring. The XR9240 uses the stored index number (i.e., 1, 2, 3) to refer to the "Command Pointer Ring Base Address Register", and calculate the actual address of the newly submitted command structure in Command Ring.

Please refer to "Basic Command Operation Sequence" for more information about this register.

Type: Read/Write

Offset for Physical Function: 0x8008 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8008



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
CPR_WP[15:0]	15:0	16'h0000	Command Pointer Ring Write Pointer.

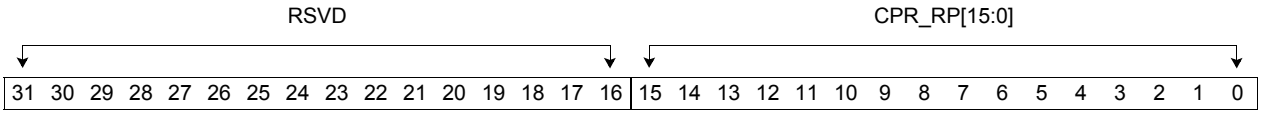
6.12.1.3 Command Pointer Ring Read Pointer Register

This register is the XR9240 command pointer ring read pointer. The host may read this register to determine how many entries in the command pointer ring have been read by the XR9240.

Type: Read/Write

Offset for Physical Function: 0x800C + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x800C



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
CPR_RP[15:0]	15:0	16'h0000	Command Pointer Ring Read Pointer.

6.12.2 Result Pointer Ring Registers

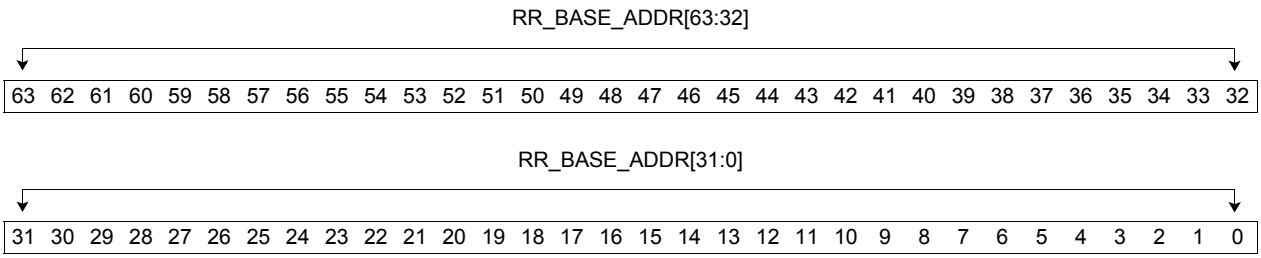
6.12.2.1 Result Pointer Ring Base Address Register

This register is the result pointer ring base address in host memory. After the XR9240 finishes a command, it will write the command status to the result pointer ring. The Result Pointer Ring base address must be 16-byte aligned.

Type: Read/Write

Offset for Physical Function: 0x8010 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8010



Field Name	Bits	Reset	Description
RR_BASE_ADDR [63:0]	63:0	64'h0000 0000	Result Pointer Ring Base Address.

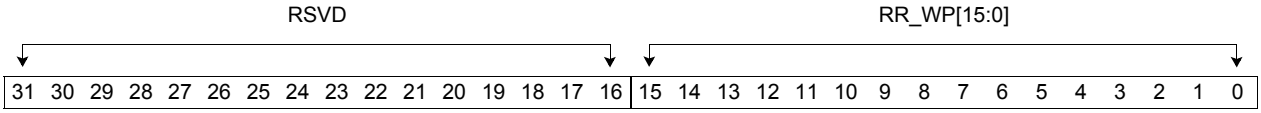
6.12.2.2 Result Pointer Ring Write Pointer Register

This register is the XR9240 result ring write pointer. The XR9240 automatically increments this register after successfully writing to the Result Pointer Ring. The host does not need to write to this register once the XR9240 is initialized. The host may read this register to determine how many entries the XR9240 has written to the Result Pointer Ring. There is no corresponding Result Ring Read Pointer register.

Type: Read/Write

Offset for Physical Function: 0x8018 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8018



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
RR_WP[15:0]	15:0	16'h0000	Result Pointer Ring Write Pointer.

6.12.3 Free Buffer Pointer Ring Registers

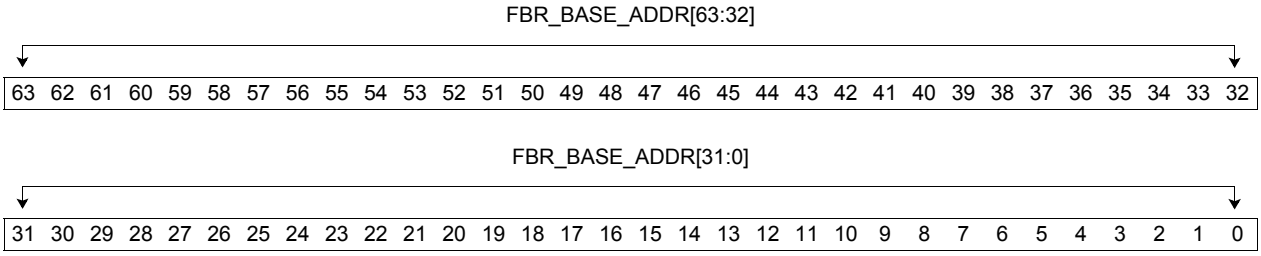
6.12.3.1 Free Buffer Pointer Ring Base Address Register

This register is the base address of the Free Buffer Pointer Ring. The XR9240 uses this base address plus an offset to fetch a free buffer entry. The base address must be 8-byte aligned.

Type: Read/Write

Offset for Physical Function: 0x801C + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x801C



Field Name	Bits	Reset	Description
FBR_BASE_ADDR [63:0]	63:0	64'h0000 0000	Free Buffer Pointer Ring Base Address.

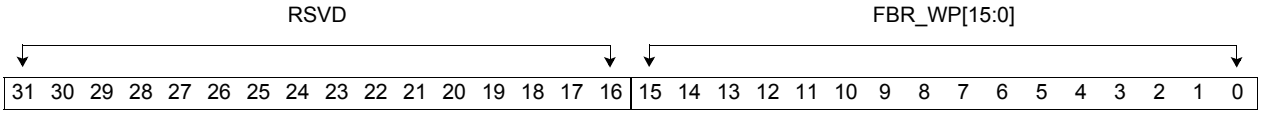
6.12.3.2 Free Buffer Pointer Ring Write Pointer Register

This register is a copy of the Free Buffer Pointer Ring write pointer in host memory. The host must update this register when a new entry is added to the free buffer pointer ring.

Type: Read/Write

Offset for Physical Function: 0x8024 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8024



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
FBR_WP[15:0]	15:0	16'h0000	Free Buffer Pointer Ring Write Pointer.

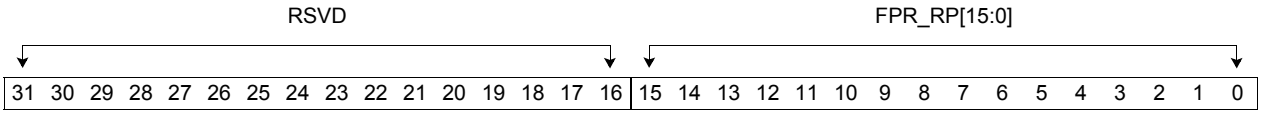
6.12.3.3 Free Buffer Pointer Ring Read Pointer Register

This register is the XR9240 Buffer Pool Pointer Ring read pointer. The host may read this register to determine how many free buffer pointer ring entries have been read by the XR9240.

Type: Read/Write

Offset for Physical Function: 0x8028 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8028



Field Name	Bits	Reset	Description
RSVD	31:16	16'h0000	Reserved.
FBR_RP[15:0]	15:0	16'h0000	Free Buffer Pointer Ring Read Pointer.

6.12.4 Ring Configuration Register

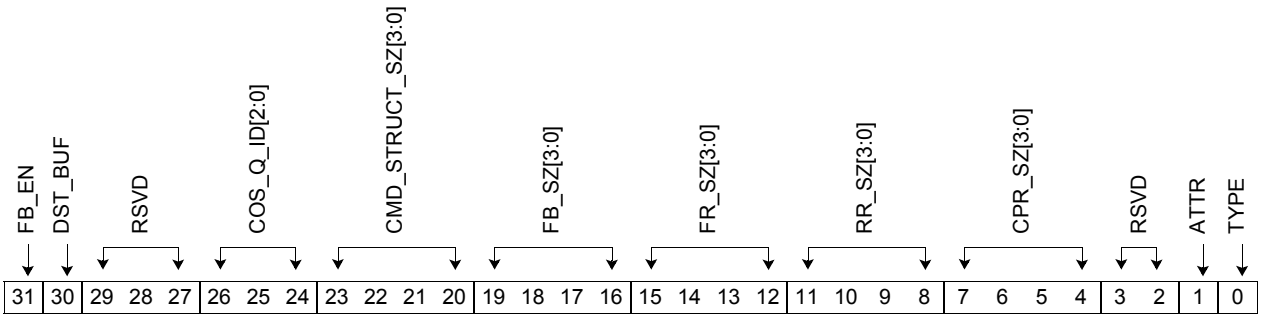
This register is used by the host to configure the 128 rings during initialization and to read the status during runtime.

For each ring type (command, result, free), this register specifies the maximum allowable size of the ring, and the maximum allowable size of the structure that the ring points to.

Type: Read/Write

Offset for Physical Function: 0x802C + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x802C



Field Name	Bits	Reset	Description
FB_EN	31	1'b0	Free Buffer Enable. This bit is used to disable or enable the XR9240 use of the free buffers to store destination data if the destination buffers are not large enough to accommodate the data. If this condition occurs, the FB bit in "Result Pointer Ring" descriptor will be set. 0 Disable use of free buffer 1 Enable use of free buffer
DST_BUF	30	1'b0	Destination Buffer Present. Applications that use destination buffers must set this bit during configuration. 0 Not all commands in this ring will have a destination buffer 1 All commands in this ring will have at least one destination buffer in the command structure
RSVD	29:27	3'b000	Reserved.

Field Name	Bits	Reset	Description
COS_Q_ID[2:0]	26:24	3'b000	Class of Service Queue Identifier. This field is set by the host to identify the queue that this ring will be sent to. For TC dynamic commands: 000Ring ID for TC dynamic queue 0 : 111Ring ID for TC dynamic queue 7 For TC static commands: 000Ring ID for TC static queue 0 : 111Ring ID for TC static queue 7 For PK commands: 000Ring ID for PK COS queue 0 001Ring ID for PK COS queue 1 010Ring ID for PK COS queue 2 011Ring ID for PK COS queue 3 All other PK values are not allowed.
CMD_STRUCT_SZ [3:0]	23:20	4'b0000	Command Structure Size. This field indicates the maximum size of each command structure. 000032 000164 0010128 0011256 0100512 01011024 01102048 01114096 10008192 100116,384 All other values reserved.

Field Name	Bits	Reset	Description
FB_SZ[3:0]	19:16	4'b0000	Free Buffer Size. This field indicates the size of the free buffer in bytes. 000032 000164 0010128 0011256 0100512 01011024 01102048 01114096 10008192 100116,384 All other values reserved.
FBR_SZ[3:0]	15:12	4'b0000	Free Buffer Pointer Ring Size. This field indicates the maximum number of entries in the free buffer pointer ring. 000032 000164 0010128 0011256 0100512 01011024 01102048 01114096 10008192 100116,384 All other values reserved.

Field Name	Bits	Reset	Description
RR_SZ[3:0]	11:8	4'b0000	Result Ring Size. This field indicates the maximum number of results in the results ring. Note that the result ring structure size is fixed at 16 bytes. 000032 000164 0010128 0011256 0100512 01011,024 01102,048 01114,096 10008,192 100116,384 101032,768 101165,536 All other values reserved.
CPR_SZ[3:0]	7:4	4'b0000	Command Pointer Ring Size. This field indicates the maximum number of commands in the command pointer ring. 000032 000164 0010128 0011256 0100512 01011,024 01102,048 01114,096 10008,192 100116,384 10103,2768 101165,536 All other values reserved.
RSVD	3:2	2'b00	Reserved.
ATTR	1	1'b0	Ring Attribute. 0 Dynamic Ring 1 Static Ring
TYPE	0	1'b0	Ring Type. 0 Transform Channel Ring 1 Public Key Ring

6.12.5 TLP Processing Hint (TPH) Registers

The TPH registers may be used by the host to individually enable or disable TPH for the command pointer ring, result pointer ring, free buffer pointer ring, command structure, and destination data structures. TLP Processing Hints are defined by the PCIe specification and in the PCIe "TPH Requestor Capability Registers" in this document.

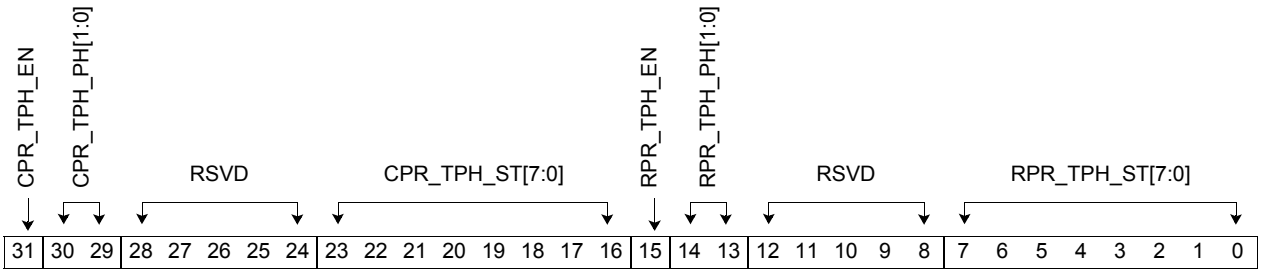
6.12.5.1 TPH Command Pointer Ring and Result Pointer Ring Configuration Register

The TPH Configuration register is used to enable TPH, set the processing hint and set the steering tag for the command or result pointer rings.

Type: Read/Write

Offset for Physical Function: 0x8030 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8030



Field Name	Bits	Reset	Description
CPR_TPH_EN	31	1'b0	Command Pointer Ring TPH Enable. 0 Disable TPH access to command pointer ring 1 Enable TPH access to command pointer ring
CPR_TPH_PH[1:0]	30:29	2'b00	Command Pointer Ring Processing Hint.
RSVD	28:24	5'b00000	Reserved.
CPR_TPH_ST[7:0]	23:16	8'h00	Command Pointer Ring TPH Steering Tag.
RPR_TPH_EN	15	1'b0	Result Pointer Ring TPH Enable. 0 Disable TPH access to result pointer ring 1 Enable TPH access to result pointer ring
RPR_TPH_PH[1:0]	14:13	2'b00	Result Pointer Ring TPH Processing Hint.
RSVD	12:8	5'b00000	Reserved.
RPR_TPH_ST[7:0]	7:0	8'h00	Result Pointer Ring TPH Steering Tag.

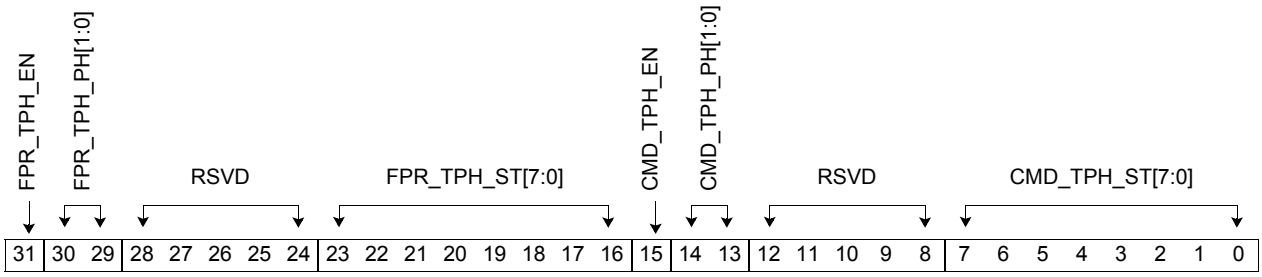
6.12.5.2 TPH Free Buffer Pointer Ring and Command Structure Configuration Register

The TPH Configuration register is used to enable TPH, set the processing hint and set the steering tag for the free buffer pointer ring and command structure.

Type: Read/Write

Offset for Physical Function: 0x8034 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8034



Field Name	Bits	Reset	Description
FPR_TPH_EN	31	1'b0	Free Buffer Pointer Ring TPH Enable. 0 Disable TPH access to free buffer pointer ring 1 Enable TPH access to free buffer pointer ring
FPR_TPH_PH[1:0]	30:29	2'b00	Free Buffer Pointer Ring Processing Hint.
RSVD	28:24	5'b00000	Reserved.
FPR_TPH_ST[7:0]	23:16	8'h00	Free Buffer Pointer Ring TPH Steering Tag.
CMD_TPH_EN	15	1'b0	Command Structure TPH Enable. 0 Disable TPH access to command structure 1 Enable TPH access to command structure
CMD_TPH_PH[1:0]	14:13	2'b00	Command Structure TPH Processing Hint.
RSVD	12:8	5'b00000	Reserved.
CMD_TPH_ST[7:0]	7:0	8'h00	Command Structure TPH Steering Tag.

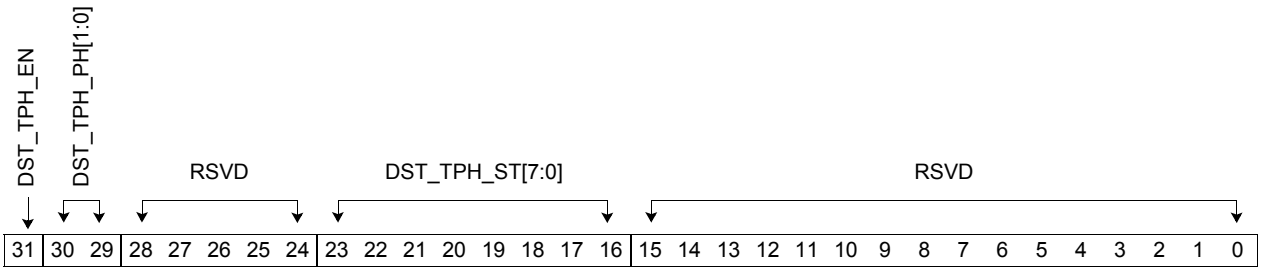
6.12.5.3 TPH Destination Data and MSIX Configuration Register

The TPH Configuration register is used to enable TPH, set the processing hint and set the steering tag for the destination data.

Type: Read/Write

Offset for Physical Function: 0x8038 + 0x40*R where R = Ring number (0-127)

Offset for Virtual Function: 0x8038



Field Name	Bits	Reset	Description
DST_TPH_EN	31	1'b0	Destination Data TPH Enable. 0 Disable TPH access to the destination data 1 Enable TPH access to the destination data
DST_TPH_PH[1:0]	30:29	2'b00	Destination Data TPH Processing Hint.
RSVD	28:24	5'b00000	Reserved.
DST_TPH_ST[7:0]	23:16	8'h00	Destination Data TPH Steering Tag.
RSVD	15:0	16'h0000	Reserved.

7 PCIe BAR3 Register Definition

This section describes the registers that are mapped into the PCIe BAR3 memory space that may be accessed through the PCIe bus. The registers are accessed using a 32-bit word. The absolute register address may be calculated using:

PCIe Address = PCIe BAR3 Address + Offset

The host should not read/write to/from reserved registers. If the host writes to a reserved register, the write will be ignored by the XR9240. Likewise, if the host reads a reserved register, the return value will be all zeros.

For the PCIe BAR0 memory space, please refer to Chapter 6.

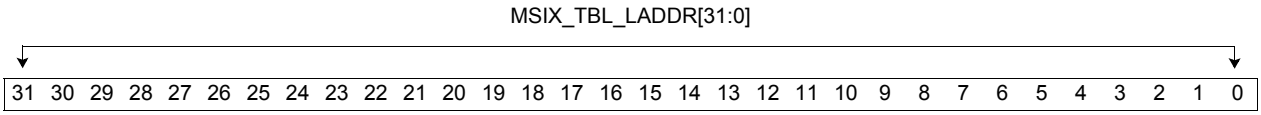
7.1 MSI-X Registers

The MSI-X registers may be accessed by both the physical and virtual functions.

7.1.1 MSI-X Table Lower Address Register

The MSI-X Table Lower Address register should be programmed by the host during initialization.

Type:	Read/Write	
Offset for Physical Function:	0x0000 + 0x10*R	where R = Ring number (0-127)
Offset for Virtual Function:	0x0000 + 0x10*R	where R = Ring number (0-1)

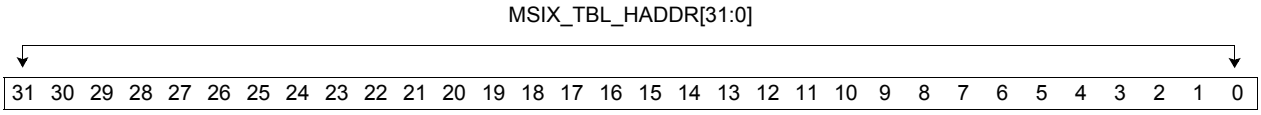


Field Name	Bits	Reset	Description
MSIX_TBL_LADDR [31:0]	31:0	32'h00000000	MSI-X Table Lower Address.

7.1.2 MSI-X Table Upper Address Register

The MSI-X Table Upper Address register should be programmed by the host during initialization.

Type:	Read/Write	
Offset for Physical Function:	0x0004 + 0x10*R	where R = Ring number (0-127)
Offset for Virtual Function:	0x0004 + 0x10*R	where R = Ring number (0-1)

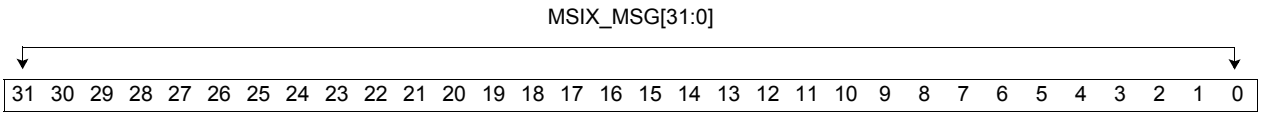


Field Name	Bits	Reset	Description
MSIX_TBL_HADDR [31:0]	31:0	32'h00000000	MSI-X Table Upper Address.

7.1.3 MSI-X Table Message Data Register

The MSI-X Table Message Data register is used to store system message data. Typically, the host will write the message to this register and the XR9240 will read the message.

Type:	Read/Write	
Offset for Physical Function:	0x0008 + 0x10*R	where R = Ring number (0-127)
Offset for Virtual Function:	0x0008 + 0x10*R	where R = Ring number (0-1)

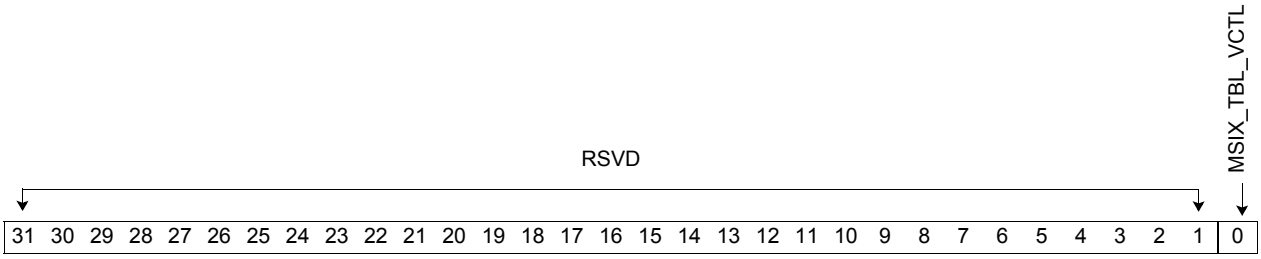


Field Name	Bits	Reset	Description
MSIX_MSG[31:0]	31:0	32'h00000000	MSI-X Table Message Data. This field may be read by the XR9240 to receive the host's message.

7.1.4 MSI-X Table Vector Control Register

The MSI-X Table Vector Control register is used to control whether the XR9240 physical function may send messages using MSI-X table entries. Typically, the host will write to this register and the XR9240 will read the control information.

Type:	Read/Write	
Offset for Physical Function:	0x000C + 0x10*R	where R = Ring number (0-127)
Offset for Virtual Function:	0x000C + 0x10*R	where R = Ring number (0-1)



Field Name	Bits	Reset	Description
RSVD	31:1	31'h00000000	Reserved.

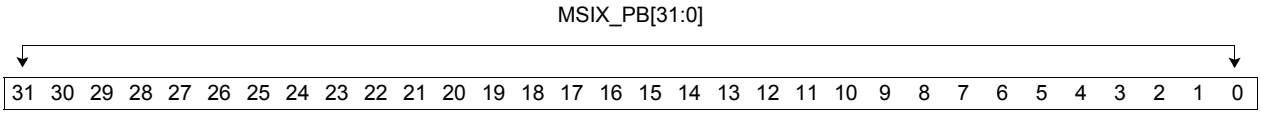
Field Name	Bits	Reset	Description
MSIX_TBL_VCTL	0	1'b0	MSI-X Table Vector Control. 0 Physical function may send a message using this MSI-X table entry 1 Physical function is prohibited from sending a message using this MSI-X table entry

7.1.5 MSI-X Pending Register

The MSI-X Pending register is used by the XR9240 to determine the host's MSIX interrupt pending status. The XR9240 will set the interrupt request using this register and the host will read the interrupt pending status.

Type: Read only

Offset for both Physical and Virtual Functions: 0x2000



Field Name	Bits	Reset	Description
MSIX_PB[31:0]	31:0	32'h00000000	MSI-X Pending Bits. This bit-wise field is used by the XR9240 to determine when the host has set or cleared a MSI-X interrupt. 0 Appropriate interrupt request is cleared 1 Appropriate interrupt request is set

8 PCIe Configuration Register Definition

This section describes the XRXR9240 PCIe configuration registers. The registers are mapped into 4K bytes of PCIe configuration space that can be accessed through the PCIe bus. The offset values in this section are given in hex.

The host should not read/write to/from reserved registers. If the host writes to a reserved register, the write will be ignored by the XR9240. Likewise, if the host reads a reserved register, the return value will be all zeros.

Note that the value read from the PCIe registers may not be the XR9240 default value because the PCIe controller may write to the PCIe configuration registers before the application software can access these registers.

Table 8-1 lists the register types definitions used to describe the PCIe configuration registers in this chapter. Fields marked as 'Read Only' usually indicate the XR9240 capability and may not be altered by the host; fields marked as 'Read/Write' may be altered by the host (usually BIOS or OS) for special purposes.

Table 8-1. Register Type Definitions

Register Type	Description
RO	Read only. Register bits are read-only and cannot be altered by software.
ROS	Sticky - Read only. Registers are read-only and cannot be altered by software. Registers are not initialized or modified by hot reset.
RW	Read/Write. Register bits are read-write and may be either set or cleared by software to the desired state.
RW1C	Read-only status, Write-1-to-clear status. Register bits indicate status when read, a set bit indicating a status event may be cleared by writing a 1. Writing a 0 to RW1C bits has no effect.
RWS	Sticky - Read-Write. Registers are read-write and may be either set or cleared by software to the desired state. Bits are not initialized or modified by hot reset.
HWINIT	Hardware Initialized. Register bits are initialized by firmware or hardware mechanisms such as pin strapping or serial EEPROM. Bits are read-only after initialization and can only be reset with a power on reset.

Table 8-2 illustrates the XR9240 PCIe Configuration registers.

Table 8-2. PCIe Configuration Space

Memory Space	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
Type 0 PCIe 3.0 Compatible Configuration Space Header	0x0	Device ID		Vendor ID	
	0x4	Status		Command	
	0x8	Class Code			Rev ID
	0xC	BIST	Header Type	Latency Timer	Cache Line Size
	0x10	BAR 0			
	0x14	BAR 1			
	0x18	BAR 2			
	0x1C	BAR 3			
	0x20	BAR 4			
	0x24	BAR 5			
	0x28	CardBus CIS Pointer			
	0x2C	SubSystem Device ID		SubSystem Vendor ID	
	0x30	Expansion ROM Base Address			
	0x34	Reserved			Cap Pointer
	0x38	Reserved			
	0x3C	Max Latency	Min Grant	Intr Pin	Intr Line
Power Management Capabilities Structure	0x40	Power Management Cap		Next Cap Ptr	Cap ID
	0x44	Data	Bridge	Pwr Mgt Control and Status	
MSI Capabilities Structure	0x50	Message Control		Next Cap Ptr	Cap ID
	0x54	Message Address			
	0x58	Message Upper Address			
	0x5C	Reserved		Message Data	
	0x60	Mask Bits			
	0x64	Pending Bits			

Table 8-2. PCIe Configuration Space

Memory Space	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
PCIe Capability Structure	0x70	PCIe Cap		Next Cap Ptr	Cap ID
	0x74	Device Capability			
	0x78	Device Status		Device Control	
	0x7C	Link Capability			
	0x80	Link Status		Link Control	
	0x84	Reserved			
	0x88	Reserved		Reserved	
	0x8C	Reserved			
	0x90	Reserved		Reserved	
	0x94	Device Capability 2			
	0x98	Reserved		Device Control 2	
	0x9C	Link Capability 2			
	0xA0	Link Status 2		Link Control 2	
	0xA4	Reserved			
	0xA8	Reserved		Reserved	
MSI-X Capabilities Structure	0xB0	Message Control		Next Cap Ptr	Cap ID
	0xB4	Table Offset			
	0xB8	PBA Offset			
Advanced Error Reporting Capability Structure	0x100	Next Cap Ptr		Version	AER Cap ID
	0x104	Uncorrectable Error Status			
	0x108	Uncorrectable Error Mask			
	0x10C	Uncorrectable Error Severity			
	0x110	Correctable Error Status			
	0x114	Correctable Error Mask			
	0x118	Advanced Error Capabilities and Control Register			
	0x11C - 0x128	Header Log			
ARI Capability Structure	0x148	Next Cap Ptr		Version	ARI Cap ID
	0x14C	ARI Control		ARI Capabilities	
Secondary PCIe Capability Structure (Gen 3)	0x158	Secondary PCIe Extended Capability Header			
	0x15C	Link Control 3 Register			
	0x160	Lane Error Status Register			
	0x164	Equalization Control Register			

Table 8-2. PCIe Configuration Space

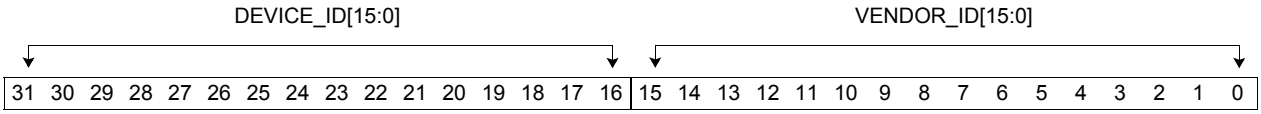
Memory Space	Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
SR-IOV Capability Structure	0x178	Next Cap Ptr		Version	IOV Cap ID
	0x17C	SR-IOV Capabilities			
	0x180	SR-IOV Status		SR-IOV Control	
	0x184	TotalVFs		InitialVFs	
	0x188	Reserved	Func Dep Link	NumVFs	
	0x18C	VF Stride		First VF Offset	
	0x190	VF Device ID		Reserved	
	0x194	Supported Page Size			
	0x198	System Page Size			
	0x19C	VF BAR0 (low)			
	0x1A0	VF BAR0 (High)			
	0x1A4	VF BAR 2			
	0x1A8	VF BAR3 (low)			
	0x1AC	VF BAR3 (High)			
	0x1B0	VF BAR 5			
	0x1B4	VF Migration State Array Offset			
TPH Capability Structure	0x1B8	Extended Capability Header			
	0x1BC	TPH Requester Capability			
	0x1C0	TPH Requester Control			

8.1 Type 0 PCIe Compatible Configuration Space

8.1.1 Vendor and Device ID Register

The Vendor and Device ID register identifies Exar as the manufacturer of the XR9240 device.

Offset 0x0000



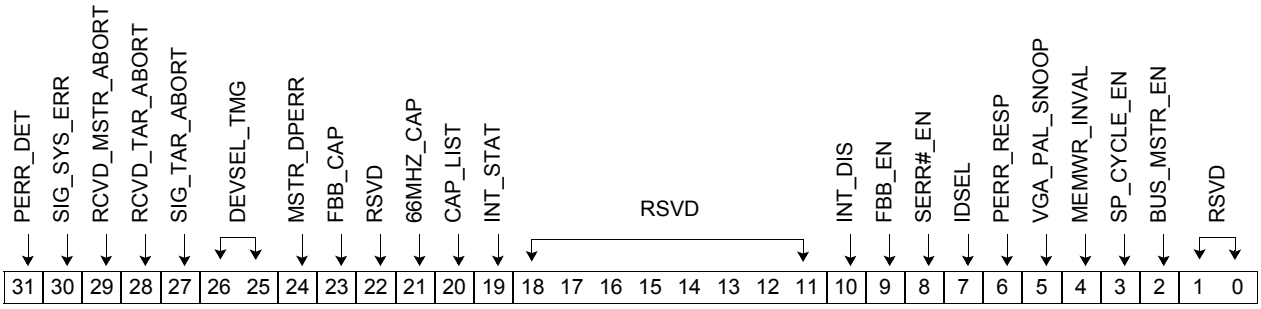
Field Name	Bits	Type	XR9240 Value	Description
DEVICE_ID[15:0]	31:16	RO	Set from OTP	This 16-bit value is assigned by Exar and identifies the XR9240. The values for this field will be defined for the production release of the XR9240. 0x9240 = XR9240 device
VENDOR_ID[15:0]	15:0	RO	0x13A3	This 16-bit value identifies Exar as the manufacturer of the XR9240. The value hardwired in this read-only register is assigned by a central authority (the PCI SIG) that controls issuance of the numbers.

8.1.2 Command and Status Register

The Command fields are used to enable or disable the I/O space, Memory space, Bus Master, Parity Error Response, System Errors, and Interrupts.

The Status fields are used to report errors and interrupts from the XR9240 to the host. The read only fields of this register are automatically updated by the XR9240 to reflect their internal status.

Offset 0x0004



Field Name	Bits	Type	XR9240 Value	Description
PERR_DET	31	RW1C	1'b0	Parity Error Detected. Regardless of the state the Parity Error Enable bit in the XR9240's Command register, this bit is set if the XR9240 receives a Poisoned TLP. 0 XR9240 has not received a Poisoned TLP. 1 XR9240 received a Poisoned TLP.
SIG_SYS_ERR	30	RW1C	1'b0	Signaled System Error. 0 XR9240 has not sent a fatal or non-fatal message 1 The XR9240 sent an ERR_FATAL or ERR_NONFATAL message, and the SERR Enable bit in the Command register is set to one
RCVD_MSTR_ABORT	29	RW1C	1'b0	Received Master Abort. 0 XR9240 has not sent a master abort message 1 XR9240 received a Completion with Unsupported Request Completion Status
RCVD_TAR_ABORT	28	RW1C	1'b0	Received Target Abort. 0 XR9240 has not sent a received target abort message 1 XR9240 received a Completion with Completer Abort Completion Status

Field Name	Bits	Type	XR9240 Value	Description
SIG_TAR_ABORT	27	RW1C	1'b0	Signaled Target Abort. 0 XR9240 has not sent a signal target abort message 1 The XR9240, acting as a Completer, terminated a request by issuing Completer Abort Completion Status to the Requester
DEVSEL_TIM[1:0]	26:25	RO	2'b00	DEVSEL Timing. This field must be hardwired to zero.
MSTR_DPERR	24	RW1C	1'b0	Master Data Parity Error. The Master Data Parity Error bit is set by a XR9240 if the PERR_RESP bit is set and either of the following two conditions occurs: -the XR9240 receives a poisoned completion, or -the XR9240 poisons a write request. If the PERR_RESP bit is cleared, the MSTR_DPERR status bit will never be set. 0 No Master Data Parity Error occurred 1 Master Data Parity Error occurred
FBB_CAP	23	RO	1'b0	Fast Back-to-Back Transactions Capable. This bit must be hardwired to zero.
RSVD	22	RO	1'b0	Reserved.
66MHZ_CAP	21	RO	1'b0	66 MHz Capable. This bit must be hardwired to zero.
CAP_LIST	20	RO	1'b1	Capabilities List. Indicates the presence of one or more extended capability register sets in the lower 48 dwords of the XR9240's PCI-compatible configuration space. All PCIe functions must implement a capability structure. 0 Capabilities List not present 1 Capabilities List present

Field Name	Bits	Type	XR9240 Value	Description
INT_STAT	19	RO	1'b0	Interrupt Status. Indicates that the XR9240 has an interrupt request outstanding (that is, the XR9240 transmitted an interrupt message that is waiting to be serviced). Note that INTx emulation interrupts forwarded by Root and Switch Ports from devices downstream of the Root or Switch Port are not reflected in this bit. Note: this bit is only associated with INTx messages, and has no meaning if the device is using Message Signaled Interrupts. 0 XR9240 has no interrupt request outstanding 1 XR9240 has an interrupt request outstanding
RSVD	18:11	RO	8'h00	Reserved.
INT_DIS	10	RW	1'b0	Interrupt Disable. Controls whether the XR9240 can generate INTx interrupt messages. 0 XR9240 enabled to generate INTx interrupt messages 1 XR9240 disabled to generate INTx interrupt messages If the XR9240 had already transmitted an Assert_INTx emulation interrupt messages and this bit is then set, the XR9240 must transmit a corresponding Deassert_INTx message for each previously transmitted assert message. Note that INTx emulation interrupt messages forwarded by Root and Switch Ports from devices downstream of the Root or Switch Port are not affected by this bit.
FBB_EN	9	RO	1'b0	Fast Back-to-Back Transactions Enable. This bit must be hardwired to zero.
SERR_EN_N	8	RW	1'b0	System Error Enable. This active low bit enables or disables the reporting of fatal and non-fatal errors detected by the XR9240 to the Root Complex. 0 XR9240 may send detected errors to the Root Complex 1 XR9240 may not send detected errors to the Root Complex
IDSEL	7	RO	1'b0	IDSEL Stepping/Wait Cycle Control. This bit must be hardwired to zero.

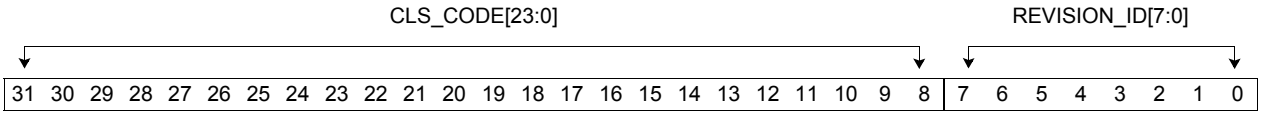
Field Name	Bits	Type	XR9240 Value	Description
PERR_RESP	6	RW	1'b0	Parity Error Response. This bit controls the logging of poisoned TLPs in the MSTR_DPERR bit. 0 Disable MSTR_DPERR 1 Enable MSTR_DPERR
VGA_PAL_SNOOP	5	RO	1'b0	VGA Palette Snoop. This bit must be hardwired to zero.
MEMWR_INV	4	RO	1'b0	Memory Write and Invalidate. This bit must be hardwired to zero.
SP_CYCLE_EN	3	RO	1'b0	Special Cycle Enable. This bit must be hardwired to zero.
BUS_MSTR_EN	2	RW	1'b0	Bus Master Enable. 0 Disables the XR9240 from issuing memory or IO requests, and from generating MSI/MSI-X messages. 1 Enables the XR9240 to issue memory or IO requests, including MSI/MSI-X messages. Requests other than memory or IO requests are not controlled by this bit.
RSVD	1:0	RO	2'b00	Reserved.

8.1.3 Revision ID and Class Code Register

The Revision ID register identifies the major revision number assigned to each XR9240 device. Please refer to Section 6.4.7 for the minor revision number of the XR9240 device.

The Class Code field defines the XR9240 as an encryption/decryption controller.

Offset 0x0008



Field Name	Bits	Type	XR9240 Value	Description
CLS_CODE[23:0]	31:8	RO	24'h120000	Encryption/Decryption controller.
REVISION_ID[7:0]	7:0	RO	8'h00	This 8-bit value is assigned by Exar to identify the revision number of the XR9240. 0 = First major revision of the XR9240

8.1.4 Cache Line Size, Header Type, Latency Timer, BIST Register

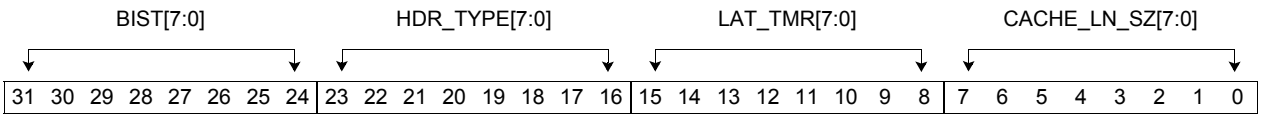
The Cache Line Size is implemented as a read-write field for legacy compatibility purposes but has no impact on the XR9240's functionality. The typical read value is 0x10.

The Master Latency Timer is implemented for legacy compatibility purposes but has no impact on the XR9240's functionality. This field is hard-wired to 0x00.

The Header Type is a read-only optional field whose value is hard-wired to 0x00.

The BIST is used for control and status of the BIST function. This field is hard-wired to 0x00.

Offset 0x000C

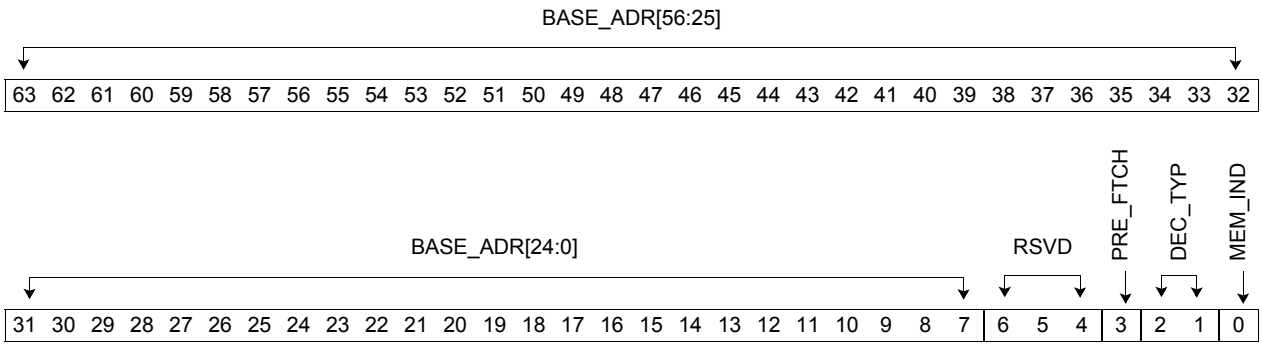


Field Name	Bits	Type	XR9240 Value	Description
BIST[7:0]	31:24	RO	8'h00	Built in Self Test.
HDR_TYPE[7:0]	23:16	RO	8'h00	Header Type.
LAT_TMR[7:0]	15:8	RO	8'h00	Latency Timer.
CACHE_LN_SZ[7:0]	7:0	RW	8'h00	Cache Line Size.

8.1.5 Base Address 0, 1 Register

The Base Address 0 register provides the XR9240 physical function base address on a 4KB boundary and some memory configuration parameters. This address in memory space is where the standard XR9240 register set will reside and be accessed. For 32-bit systems, the base address is set using only BAR 0. For 64-bit systems, the base address is set using both BAR 0 and BAR 1.

Offset 0x0010

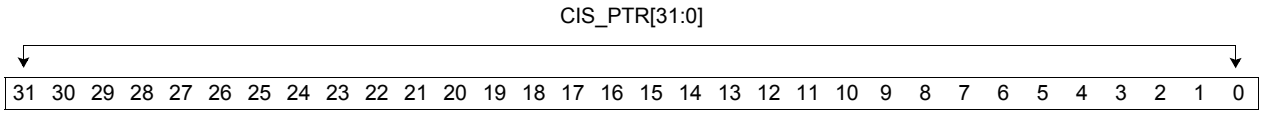


Field Name	Bits	Type	XR9240 Value	Description
BASE_ADR[56:0]	63:7	RW	57'h0000 00000000 00	Base Address. [63:32] Only used for 64-bit 4KB memory base address [31:7] Indicates 32-bit 4KB memory base address
RSVD	6:4	RO	3'b000	Reserved.
PRE_FTCH	3	RO	1'b0	Prefetchable memory. 0 Non-Prefetchable memory 1 Prefetchable memory
DEC_TYP	2:1	RO	2'b00 for a 32-bit system; 2'b10 for a 64-bit system	Decoder Type. 00 32 bit decoder; locate memory anywhere in lower 4GB; use only BAR0 01 Reserved 10 64-bit decoder; locate memory anywhere in 264 memory space; uses BAR0 and BAR1 11 Reserved
MEM_IND	0	RO	1'b0	Memory Space Indicator. x0 Base Address Register0 is a memory address decoder x1 Base Address Register0 is an IO address decoder

8.1.6 Cardbus CIS Pointer Register

This optional read-only register is used by devices that contain a CardBus and PCIe interface. The XR9240 does not support Cardbus so this register is hard-wired to 0x00000000.

Offset0 0x0028



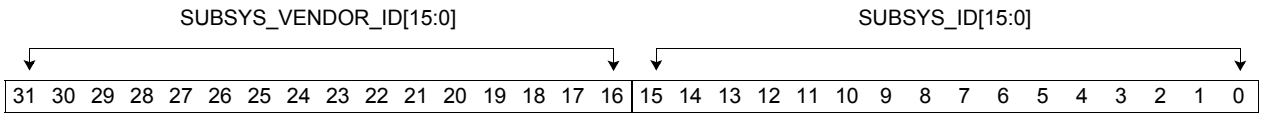
Field Name	Bits	Type	XR9240 Value	Description
CIS_PTR[31:0]	31:0	RO	32'h0000 0000	Cardbus CIS Pointer

8.1.7 Sub-System Vendor and Device ID Register

The Sub-System Vendor ID register identifies Exar as the manufacturer of the XR9240 device. This value is hard coded in the XR9240 device.

The Sub-System ID register identifies the XR9240. This value is hard coded in the XR9240 device.

Offset 0x002C

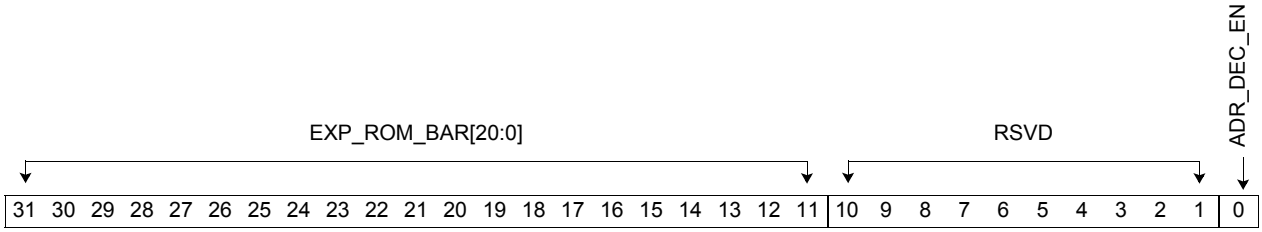


Field Name	Bits	Type	XR9240 Value	Description
SUBSYS_ID[15:0]	31:16	RO	16'h9200	This 16-bit value is assigned by the subsystem manufacturer and identifies the type of subsystem.
SUBSYS_VENDOR_ ID[15:0]	15:0	RO	16'h13A3	This 16-bit register identifies the manufacturer of the subsystem. The value hardwired in this read-only register is assigned by a central authority (the PCI SIG) that controls issuance of the numbers.

8.1.8 Expansion ROM Base Address Register

The ROM Base address register (ROM BAR) provides the XR9240 expansion ROM Base address and address decode enable. This register sets the address space in memory for the optionally attached non-volatile device.

Offset 0x0030 If the Expansion ROM is enabled, bits [23:11] are RO to reserve 16M space.

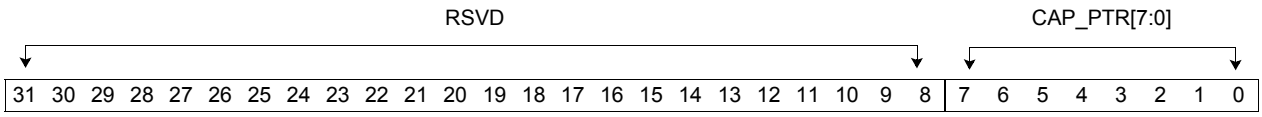


Field Name	Bits	Type	XR9240 Value	Description
EXP_ROM_BAR [20:0]	31:11	RW	21'h00000	Expansion ROM Base Address. [31:24] The base address of the expansion ROM If the Expansion ROM is enabled, bits [23:11] are RO to reserve 16M space.
RSVD	10:1	RO	10'b00000 00000	Reserved.
ADR_DEC_EN	0	RW	1'b0	Address Decode Enable. 0 Disable expansion ROM 1 Enable expansion ROM

8.1.9 Capabilities Pointer Register

The Capabilities Pointer register is used to point to a linked list of capabilities implemented by the XR9240. This read-only register value is hard-wired to 0x40, the Power Management Capabilities structure.

Offset 0x0034



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:8	RO	24'h000000	Reserved.
CAP_PTR[7:0]	7:0	RO	8'h40	Capabilities Pointer. This field contains the offset to the next PCI Express capability structure.

8.1.10 Interrupt Line and Pin, Min_Gnt and Max_Lat Register

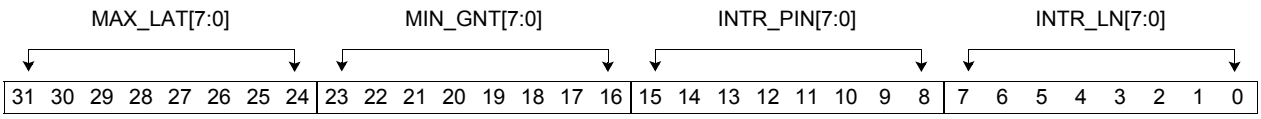
The Interrupt Line field communicates interrupt line routing information to device drivers and operating systems. Values in this field are programmed by system software and are system architecture specific. The typical read value is 0x0B.

The Interrupt Pin field identifies the legacy interrupt Message(s) used by the XR9240. This read-only field will be read as 0x01, indicating the XR9240 uses INTA.

The Min_Gnt register is a read-only field whose value is hard-wired to 0x00.

The Max_Lat register is a read-only field whose value is hard-wired to 0x00.

Offset 0x003C



Field Name	Bits	Type	XR9240 Value	Description
MAX_LAT[7:0]	31:24	RO	8'h00	Maximum Latency.
MIN_GNT[7:0]	23:16	RO	8'h00	Minimum Grant.
INTR_PIN[7:0]	15:8	RO	8'h01	Interrupt Pin.
INTR_LN[7:0]	7:0	RW	8'hFF	Interrupt Line.

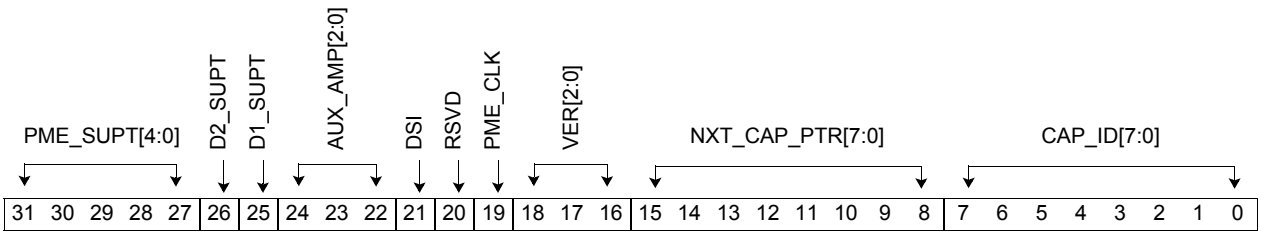
8.2 Power Management Capabilities Registers

The Power Management registers indicate the XR9240 power management capabilities.

8.2.1 Capability ID Register

The Power Management Capabilities read-only fields provide information on the capabilities of the function related to power management. The information in this register is generally static.

Offset 0x0040



Field Name	Bits	Type	XR9240 Value	Description
PME_SUPT[4:0]	31:27	RO	5'b00000	PME Support. This field is not supported by the XR9240.
D2_SUPT	26	RO	1'b1	D2 Support. 1 = D2 PM state supported
D1_SUPT	25	RO	1'b1	D1 Support. 1 = D1 PM state supported
AUX_AMP[2:0]	24:22	RO	3'b000	Aux Current. The XR9240 does not have an auxiliary power supply.
DSI	21	RO	1'b0	Device-Specific Initialization. A one in this bit indicates that immediately after entry into the D0 Uninitialized state, the XR9240 requires additional configuration above and beyond setup of its PCI configuration Header registers before the Class driver can use the XR9240. Microsoft OSs do not use this bit. Rather, the determination and initialization is made by the Class driver.
RSVD	20	RO	1'b0	Reserved.
PME_CLK	19	RO	1'b0	PME Clock. This bit must be hardwired to zero.

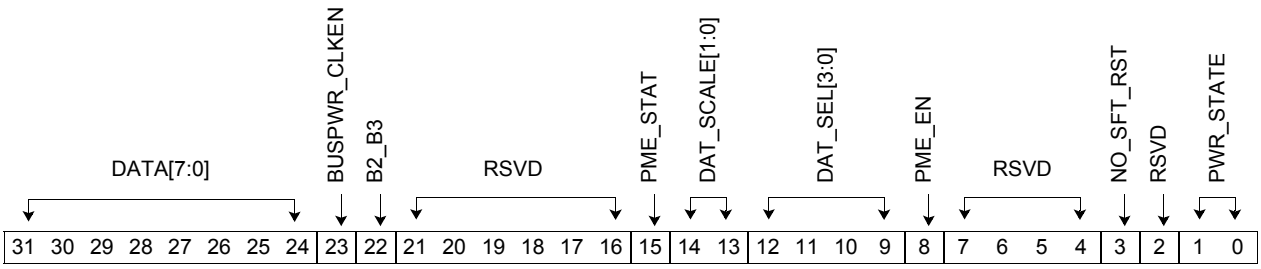
Field Name	Bits	Type	XR9240 Value	Description
VER[2:0]	18:16	RO	3'b011	Version Field. This field indicates the version of the PCI Bus PM Interface spec that the XR9240 complies with.
NXT_CAP_PTR	15:8	RO	8'h50	Power Management Next Capabilities Pointer. This field contains the offset to the next PCI Express capability structure.
CAP_ID	7:0	RO	8'h01	Power Management Capability ID.

8.2.2 Power Management Control/Status Register

The Power Management Control/Status (PMCSR) register is used to manage the PCI function's power management state.

The Data fields are not used by the XR9240 but may be written to by system software. When read, these fields will typically return all zeroes.

Offset 0x0044



Field Name	Bits	Type	XR9240 Value	Description
DATA[7:0]	31:24	RO	8'h00	Data. This field is not supported by the 920x.
BUSPWR_CLKEN	23	RO	1'b0	Bus Power/Clock Control Enable.
B2_B3	22	RO	1'b0	B2/B3 Support.
RSVD	21:16	RO	6'b000000	Reserved.
PME_STAT	15	RW1C	1'b0	PME Status. This bit is set to "0" because the XR9240 does not support PME# generation from D3cold.
DAT_SCALE[1:0]	14:13	RO	2'b00	Data Scale. This field is set to "00b" in the XR9240 because the PM Data register is not implemented.
DAT_SEL[3:0]	12:9	RO	4'b0000	Data Select. This field is set to "0000b" in the XR9240 because the PM Data register is not implemented.
PME_EN	8	RWS	1'b0	PME Enable. This bit is set to "0" because the XR9240 does not support PME# generation from D3cold.
RSVD	7:4	RO	4'b0000	Reserved.
NO_SFT_RST	3	RO	1'b1	No Soft Reset. The XR9240 will not perform an internal reset when transitioning from D3hot to D0 due to a PowerState command.

Field Name	Bits	Type	XR9240 Value	Description
RSVD	2	RO	1'b0	Reserved.
PWR_STATE	1:0	RW	2'b00	Power State. This 2-bit field is used both to determine the current power state of the XR9240 and to set the XR9240 into a new power state. The definition of the field values is given below. 00b - D0 01b - D1 (unsupported) 10b - D2 (unsupported) 11b - D3hot If software attempts to write an unsupported state to this field, the write operation must complete normally on the bus; however, the data is discarded and no state change will occur.

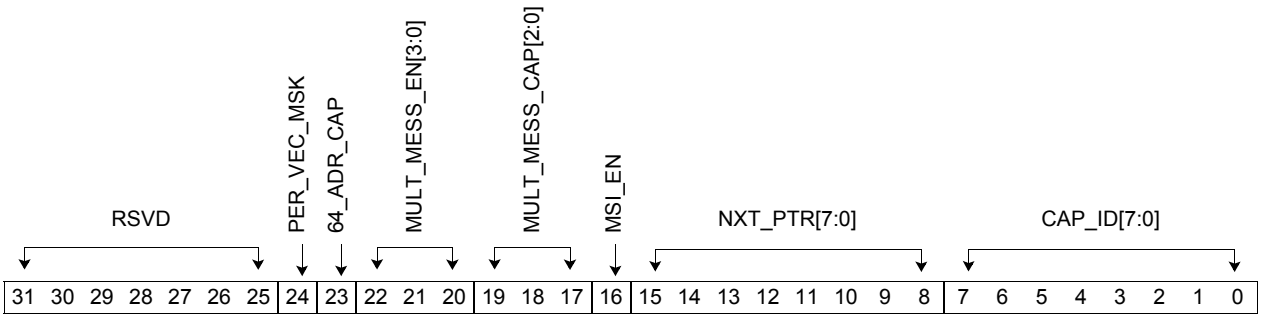
8.3 MSI Capability Registers

The MSI Capability registers indicate the XR9240 Message Signaled Interrupt capabilities. The XR9240 uses a 64-bit Message Address.

8.3.1 MSI Capability Register

The MSI Capability register, when read by system software as 05h, indicates that the XR9240 is MSI capable.

Offset 0x0050

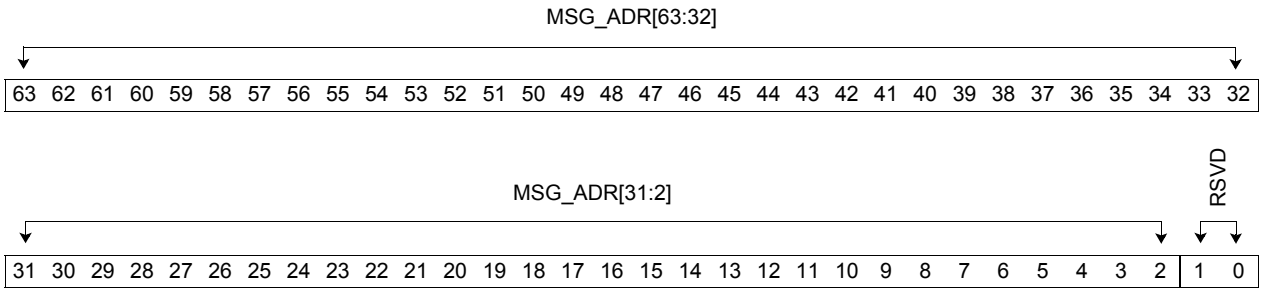


Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:25	RO	7'b0000000	Reserved.
PER_VEC_MSK	24	RO	1'b1	Per-vector masking. The XR9240 supports MSI per-vector masking. 1 = Per-vector masking supported
64_ADR_CAP	23	RO	1'b1	64-bit Address. The XR9240 is capable of generating a 64-bit message address. 1 = 64-bit address capable
MULT_MESS_EN [3:0]	22:20	RW	3'b000	Multiple Message Enable. System software writes to this field to indicate the number of allocated vectors (equal to or less than the number of requested vectors).
MULT_MESS_CAP [2:0]	19:17	RO	3'b000	Multiple Message Capable. System software reads this field to determine the number of requested vectors. 000 = 1 vector requested

Field Name	Bits	Type	XR9240 Value	Description
MSI_EN	16	RW	1'b0	MSI Enable. System configuration software sets this bit to enable MSI. 0 = XR9240 is prohibited from using MSI to request service 1 = If the MSI-X Enable bit in the MSI-X Message Control register is 0, the XR9240 is permitted to use MSI to request service.
NXT_CAP_PTR[7:0]	15:8	RO	8'h70	Next Capability Pointer.
CAP_ID[7:0]	7:0	RO	8'h05	MSI Capability. 05h = MSI capable

8.3.2 MSI Message Address Register

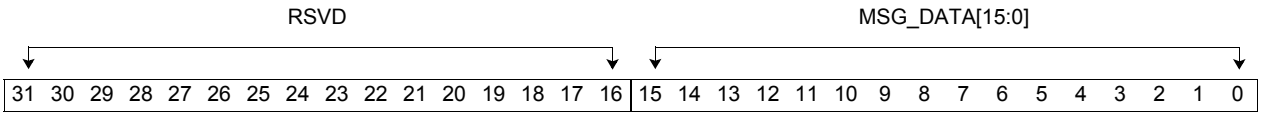
Offset 0x0054 - 0x0058



Field Name	Bits	Type	XR9240 Value	Description
MSG_ADR[63:2]	63:2	RW	configured by host	MSI Message Address. If the Message Enable bit (bit 0 of the "MSI Message Address Register") is set, the contents of this register specifies the DWORD aligned address (AD[63:02]) for the MSI memory write transaction. AD[1:0] are driven to zero during the address phase.
Reserved	1:0	RO	2'b00	Reserved. Always returns 0 on read. Write operations have no effect.

8.3.3 MSI Message Data Register

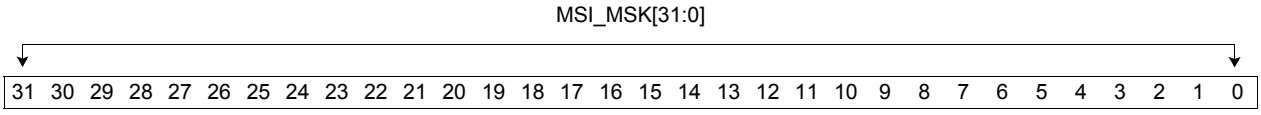
Offset 0x005C



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:16	RO	16'h00000	Reserved.
MSG_DATA	15:0	RW	16'h00000	MSI Message Data. If the Message Enable bit (bit 0 of the "MSI Message Address Register") is set, the message data is driven onto the lower word (MSG_ADR[15:0]) of the memory write transaction's data phase. MSG_ADR[31:16] are driven to zero during the memory write transaction's data phase. C/BE[3:0]# are asserted during the data phase of the memory write transaction.

8.3.4 MSI Mask Bits Register

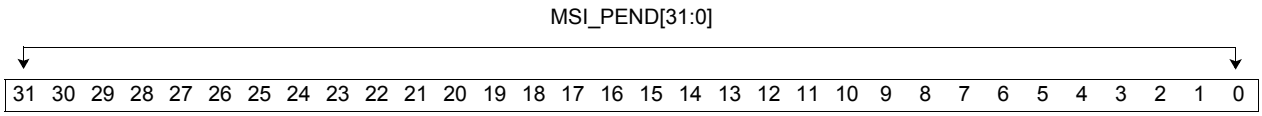
Offset 0x0060



Field Name	Bits	Type	XR9240 Value	Description
MSI_MSK[31:0]	31:0	RW	32'h0000 0000	MSI Mask Bits. MSI vectors are numbered 0 through N-1, where N is the number of vectors allocated by software. Each vector is associated to the corresponding numbered bit in the Mask Bits and Pending Bits registers.

8.3.5 MSI Pending Bits Register

Offset 0x0064



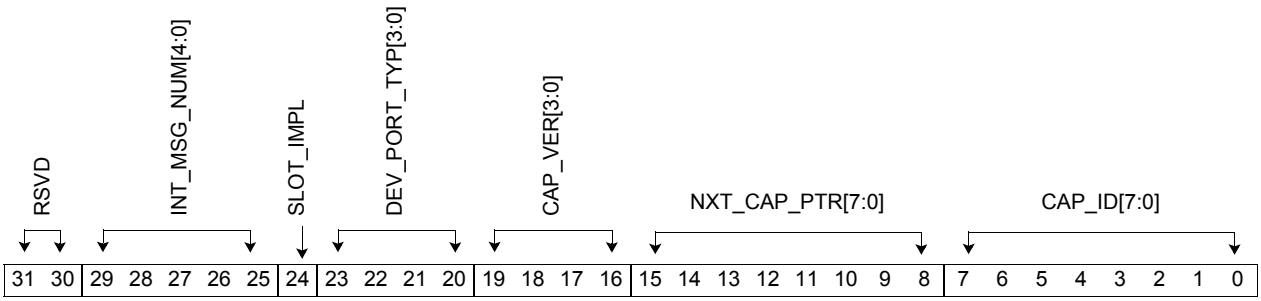
Field Name	Bits	Type	XR9240 Value	Description
MSI_PEND[31:0]	31:0	R0	32'h0000 0000	MSI Pending Bits. When a bit is set in this field, the XR9240 has a pending message for that vector.

8.4 PCI Express Capability Registers

The PCI Express Capability registers are required for PCI Express devices. The capability structure is a mechanism for enabling PCI software transparent features requiring support on legacy operating systems. In addition to identifying a PCI Express device, the PCI Express Capability structure is used to provide access to PCI Express specific Control/Status registers and related Power Management enhancements.

8.4.1 PCIe Capability Register

Offset 0x0070

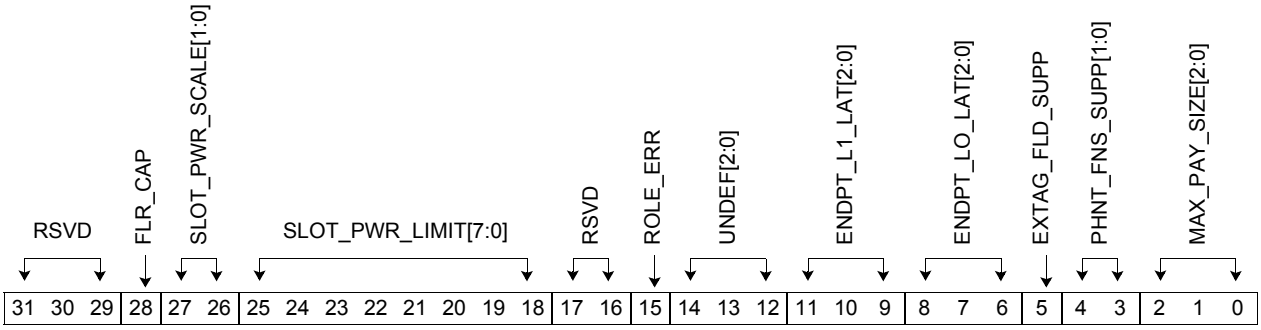


Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:30	RO	2'b00	Reserved.
INT_MSG_NUM[4:0]	29:25	RO	5'b00000	Interrupt Message Number. This field is not used by the XR9240.
SLOT_IMPL	24	HWINIT	1'b0	Slot Implemented. This bit is not used by the XR9240.
DEV_PORT_TYP[3:0]	23:20	RO	4'b0000	Device/Port Type. Indicates the type of PCI Express logical device. 0000 = XR9240 is PCI Express Endpoint device
CAP_VER[3:0]	19:16	RO	4'b0010	Capability Version. Indicates the PCI Express capability structure version number.
NXT_CAP_PTR[7:0]	15:8	RO	8'hB0	Next Capability Pointer.
CAP_ID[7:0]	7:0	RO	8'h10	PCIe Capable. 10h = XR9240 is PCIe capable

8.4.2 Device Capabilities Register

The Device Capabilities register identifies the PCI Express device specific capabilities.

Offset 0x0074



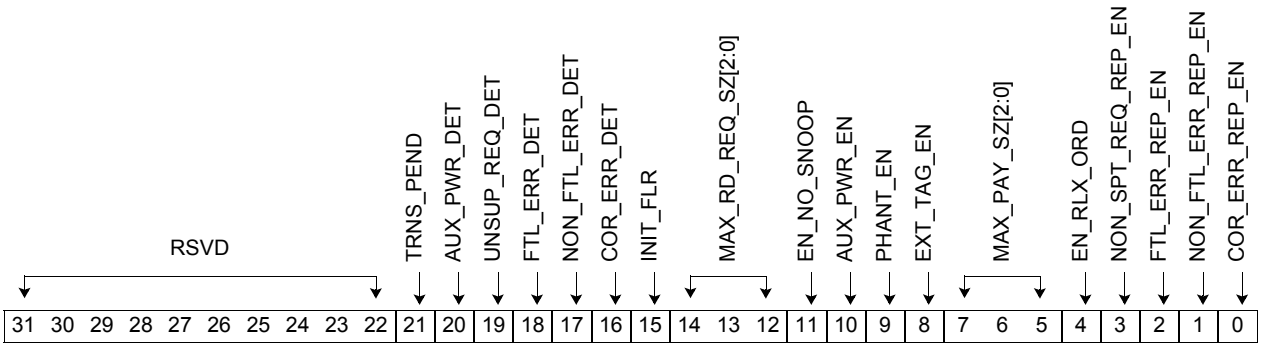
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:29	RO	3'b000	Reserved.
FLR_CAP	28	RO	1'b1	Function Level Reset Capability. The XR9240 supports FLR.
SLOT_PWR_SCALE [1:0]	27:26	RO	2'b00	Captured Slot Power Limit Scale. This field specifies the scale used for the Slot Power Limit Value, which is set by the Set_Slot_Power_Limit Message for the XR9240. Most systems will read zero, which means 1.0x.
SLOT_PWR_LIMIT [7:0]	25:18	RO	8'h00	Captured Slot Power Limit Value. This field is used in combination with the Slot Power Limit Scale value to specify the upper limit on the power supplied by the slot. The power limit (in Watts) is calculated by multiplying the value in this field by the value in the Slot Power Limit Scale field. This value is set by Set_Slot_Power_Limit Message for the XR9240.
RSVD	17:16	RO	2'b00	Reserved.
ROLE_ERR	15	RO	1'b1	Role-Based Error Reporting. This must be set by XR9240 to indicate that the XR9240 implements standard PCIe Error Reporting.
UNDEF[2:0]	14:12	RO	3'b000	Undefined.
ENDPT_L1_LAT[2:0]	11:9	RO	3'b111	Endpoint L1s Acceptable Latency. The XR9240 requires an endpoint L1 acceptable latency of 8µs maximum.

Field Name	Bits	Type	XR9240 Value	Description
ENDPT_LO_LAT[2:0]	8:6	RO	3'b111	Endpoint L0s Acceptable Latency. The XR9240 requires an endpoint L0 acceptable latency of 1µs maximum.
EXTAG_FLD_SUPP	5	RO	1'b1	Extended Tag Field Supported. Max supported size of the Tag field when the XR9240 acts as a Requester. 0 = 5-bit Tag field supported (max of 32 outstanding request per Requester) 1 = 8-bit Tag field supported (max of 256 outstanding request per Requester) If 8-bit Tags are supported and will be used, this feature is enabled by setting the Extended Tag Field Enable bit in the Device Control register to one.
PHNT_FNS_SUPP [1:0]	4:3	RO	2'b00	Phantom Functions Supported. 00 = The Phantom Function feature is not available on the XR9240 device.
MAX_PAY_SIZE[2:0]	2:0	RO	3'b011	Max Payload Size Supported. Defines the Max data payload size that the XR9240 supports for TLPs. The XR9240 supports 1k byte max payload size.

8.4.3 Device Control and Status Register

The Device Control register controls PCI Express device specific parameters.

Offset 0x0078



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:22	RO	10'b00000 00000	Reserved.
TRNS_PEND	21	RO	1'b0	Transactions Pending. This bit when set indicates that the XR9240 has issued non-posted requests which have not been completed. The XR9240 reports this bit cleared only when all outstanding non-posted requests have completed or have been terminated by the completion timeout mechanism.
AUX_PWR_DET	20	RO	1'b0	AUX Power Detected. This bit is set to 0 in the XR9240, indicating the XR9240 does not use AUX power.
UNSUP_REQ_DET	19	RW1C	1'b0	Unsupported Request Detected. This bit indicates that the XR9240 received an unsupported request. Errors are logged in this register regardless of whether error reporting is enabled or disabled.
FTL_ERR_DET	18	RW1C	1'b0	Fatal Error Detected. This bit indicates the status of the fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or disabled. Errors are logged in this register regardless of the settings of the correctable error mask register in the "Advanced Error Reporting Extended Capability Header Register".

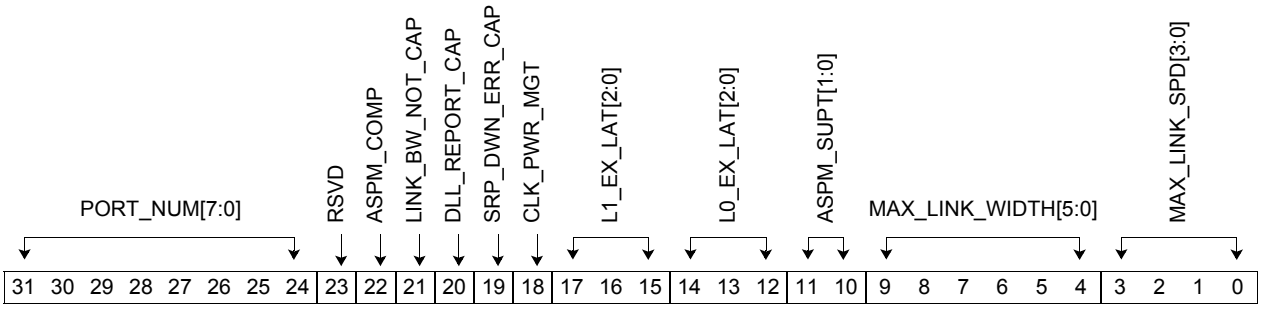
Field Name	Bits	Type	XR9240 Value	Description
NON_FTL_ERR_DET	17	RW1C	1'b0	Non-Fatal Error Detected. This bit indicates the status of the non-fatal errors detected. Errors are logged in this register regardless of whether error reporting is enabled or disabled. Errors are logged in this register regardless of the settings of the correctable error mask register in the "Advanced Error Reporting Extended Capability Header Register".
COR_ERR_DET	16	RW1C	1'b0	Correctable Error Detected. This bit indicates the status of the correctable errors detected. Errors are logged in this register regardless of whether error reporting is enabled or disabled. Errors are logged in this register regardless of the settings of the correctable error mask register in the "Advanced Error Reporting Extended Capability Header Register".
INIT_FLR	15	RW	1'b0	Initiate FLR. 0 FLR not initiated for the function 1 Initiate FLR for the function
MAX_RD_REQ_SZ [2:0]	14:12	RW	3'b010	Maximum Read Request Size. This field sets the maximum Read Request size for the XR9240. Supported XR9240 read request sizes: 000 = 128 bytes 001 = 256 bytes 010 = 512 bytes
EN_NO_SNOOP	11	RW	1'b1	Enable No Snoop. This bit is configurable in the XR9240. A one indicates the XR9240 is permitted to set the No Snoop bit in the Requester Attributes for transactions it initiates that do not require hardware enforced cache coherency.
AUX_PWR_EN	10	RWS	1'b0	Auxiliary (AUX) Power PM Enable. This bit is set to 0 by the XR9240, disabling the XR9240 to draw AUX power independent of PME AUX power.
PHANT_EN	9	RW	1'b0	Phantom Functions Enable. This bit is hardwired to 0 as the XR9240 device does not implement this capability.
EXT_TAG_EN	8	RW	1'b1	Extended Tag Field Enable. This bit is set to 1 in the XR9240 to enable the XR9240 to use an 8-bit Tag field as a requester.

Field Name	Bits	Type	XR9240 Value	Description
MAX_PAY_SZ[2:0]	7:5	RW	3'b000	Maximum Payload Size. This field is set by the host and signifies the maximum TLP payload size for the device/function. As a receiver, the device must handle TLPs as large as the set value; as transmitter, the device must not generate TLPs exceeding the set value. Supported XR9240 payload sizes: 000 = 128 bytes max payload size 001 = 256 bytes max payload size 010 = 512 bytes max payload size
EN_RLX_ORD	4	RW	1'b1	Enable Relaxed Ordering. This bit is not used by the XR9240 device .
NON_SPT_REQ_REP_EN	3	RW	1'b0	Unsupported Request Reporting Enable. This bit, in conjunction with other bits, controls the signaling of Unsupported Requests by sending Error Messages.
FTL_ERR_REP_EN	2	RW	1'b0	Fatal Error Reporting Enable. This bit, in conjunction with other bits, controls sending ERR_FATAL Messages.
NON_FTL_ERR_REP_EN	1	RW	1'b0	Non-Fatal Error Reporting Enable. This bit, in conjunction with other bits, controls sending ERR_NONFATAL Messages.
COR_ERR_REP_EN	0	RW	1'b0	Correctable Error Reporting Enable. This bit, in conjunction with other bits, controls sending ERR_COR Messages.

8.4.4 Link Capabilities Register

The Link Capabilities register identifies PCI Express Link specific capabilities.

Offset 0x007C



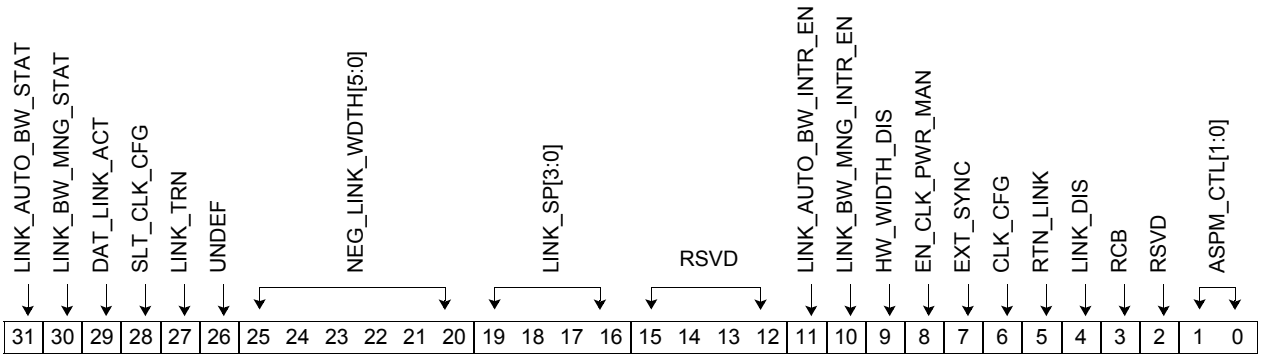
Field Name	Bits	Type	XR9240 Value	Description
PORT_NUM[7:0]	31:24	HWInit	8'h00	Port Number. The XR9240 has a single PCIe port; the port number is zero.
RSVD	23	RO	1'b0	Reserved.
ASPM_COMP	22	HWInit	1'b1	ASPM Optionally Compliance. This bit must be set to 1b in all functions.
LINK_BW_NOT_CAP	21	RO	1'b0	Link Bandwidth Notification Capability This bit is reserved for the XR9240.
DLL_REPORT_CAP	20	RO	1'b0	Data Link Layer Link Active Reporting Capable. This bit is reserved for the XR9240.
SRP_DWN_ERR_CAP	19	RO	1'b0	Surprise Down Error Reporting Capable.
CLK_PWR_MTG	18	RO	1'b0	Clock Power Management.
L1_EX_LAT[2:0]	17:15	RO	3'b110	L1 Exit Latency. Indicates the L1 exit latency for the Link (i.e., the length of time this Port requires to complete a transition from L1 to L0). The XR9240 L1 Exit latency required is 8µs to 16µs.
L0_EX_LAT[2:0]	14:12	RO	3'b100	L0 Exit Latency. Indicates the L0s exit latency for the Link (i.e., the length of time this Port requires to complete a transition from L0s to L0). The XR9240 L0 Exit latency required is 256ns to less than 512ns.

Field Name	Bits	Type	XR9240 Value	Description
ASPM_SUPT[1:0]	11:10	RO	2'b01	Active State Power Management (ASPM) Support. Indicates the level of ASPM supported on this Link. The XR9240 supports L0 and L1.
MAX_LINK_WIDTH [5:0]	9:4	RO	6'b001000	Max Link Width. The XR9240 max link width is x4.
MAX_LINK_SPD[3:0]	3:0	RO	4'b0011	Max Link Speed. 0011 = (Gen3 8.0 GT/s)

8.4.5 Link Status and Control Register

The Link Status and Control register controls PCI Express Link required parameters.

Offset 0x0080



Field Name	Bits	Type	XR9240 Value	Description
LINK_AUTO_BW_ STAT	31	RO	1'b0	Link Autonomous Bandwidth Status. This field is not applicable to the XR9240 device and is hardwired to zero.
LINK_BW_MNG_ STAT	30	RO	1'b0	Link Bandwidth Management Status. This field is not applicable to the XR9240 device and is hardwired to zero.
DAT_LINK_ACT	29	RO	1'b0	Data Link Layer Link Active. This bit is not applicable to the XR9240 device and is hardwared to zero.
SLT_CLK_CFG	28	HWINIT	1'b1	Slot Clock Configuration. This bit indicates that the XR9240 uses the same physical reference clock that the platform provides on the connector.
LINK_TRN	27	RO	1'b0	Link Training. This field is not applicable to the XR9240 device and is hardwared to zero.
UNDEF	26	RO	1'b0	Undefined. This legacy bit is no longer used.

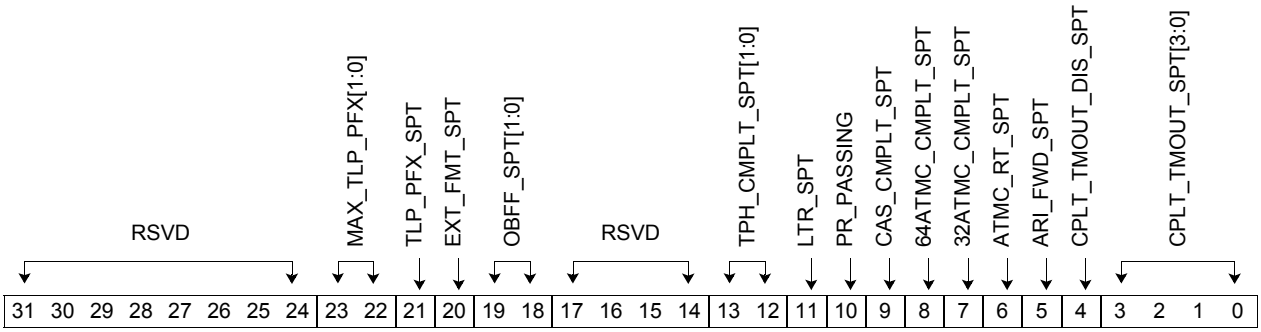
Field Name	Bits	Type	XR9240 Value	Description
NEG_LINK_WDTH [5:0]	25:20	RO	6'b00 1000	Negotiated Link Width. This field indicates the negotiated width of the given PCI Express Link. Defined encodings are: 000001 = x1 000010 = x2 000100 = x4 (typical value) 001000 = x8 001100 = x12 010000 = x16 100000 = x32 All other encodings are reserved. The value in this field is undefined when the Link is down.
LINK_SP[3:0]	19:16	RO	4'b0001	Link Speed. This field indicates the negotiated Link speed of the given PCI Express Link. Defined encodings are: 0001b (Gen1 2.5 GT/s) 0010b (Gen2 5.0 GT/s) 0100b (Gen3 8.0 GT/s)
RSVD	15:12	RO	4'b0000	Reserved.
LINK_AUTO_BW_ INTR_EN	11	RW	1'b0	Link Autonomous Bandwidth Interrupt Enable. This field is not applicable and is reserved for the XR9240 device.
LINK_BW_MNG_ INTR_EN	10	RW	1'b0	Link Bandwidth Management Interrupt Enable. This field is not applicable and is reserved for the XR9240 device.
HW_WIDTH_DIS	9	RO	1'b0	Hardware Autonomous Width Disable. When set, this bit disables hardware from changing the Link width for reasons other than attempting to correct unreliable Link operation by reducing Link width. This bit only applies to function 0. For all other functions, this bit is reserved.
EN_CLK_PWR_MAN	8	RW	1'b0	Enable Clock Power Management. This bit is hardwired to 0 as the XR9240 device does not support Clock Power Management.
EXT_SYNC	7	RW	1'b0	Extended Synch. This bit is not used by the XR9240.

Field Name	Bits	Type	XR9240 Value	Description
CLK_CFG	6	RW	1'b0	Common Clock Configuration. This bit when set indicates that the XR9240 and the component at the opposite end of this Link are operating with a distributed common reference clock.
RTN_LINK	5	RO	1'b0	Retrain Link. This field is not applicable and is reserved for the XR9240 device.
LINK_DIS	4	RO	1'b0	Link Disable. This field is not applicable and is reserved for the XR9240 device.
RCB	3	RO	1'b0	Read Completion Boundary. This bit is hardwired to 0 as the XR9240 device does not support RCB.
RSVD	2	RO	1'b0	Reserved.
ASPM_CTL[1:0]	1:0	RW	2'b00	Active State Power Management (ASPM) Control. This field controls the level of ASPM supported on the given PCI Express Link. XR9240 supported values are: 00 = Disabled

8.4.6 Device Capabilities 2 Register

The Device Capabilities 2 register identifies the PCI Express device optional capabilities.

Offset 0x0094



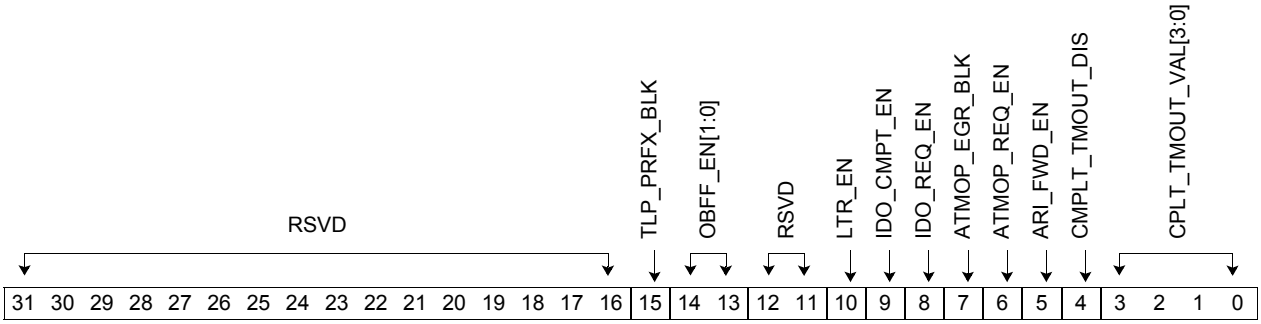
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:24	RO	8'b00000000	Reserved.
MAX_TLP_PFX[1:0]	23:22	RO	2'b00	Max End-End TLP Prefixes.
TLP_PFX_SPT	21	RO	1'b0	End-End TLP Prefix Supported. The XR9240 does not support end-end TLP prefixes.
EXT_FMT_SPT	20	RO	1'b0	Extended Fmt Field Supported. The XR9240 does not support this feature.
OBFF_SPT[1:0]	19:18	RO	2'b00	OBFF Supported. The XR9240 does not support this feature, therefore this bit must be set to zero.
RSVD	17:14	RO	4'b0000	Reserved.
TPH_CMPLT_SPT [1:0]	13:12	RO	2'b01	TPH Completer Supported. The XR9240 supports TPH Completer, however Extended TPH Completer is not supported.
LTR_SPT	11	RO	1'b0	Latency Tolerance Reporting (LTR) Mechanism Supported. The XR9240 does not supports LTR, therefore this bit must be set to zero.
PR_PASSING	10	HwInit	1'b0	No RO-enabled PR-PR Passing. The XR9240 does not support this feature, therefore this bit must be set to zero.

Field Name	Bits	Type	XR9240 Value	Description
CAS_CMPLT_SPT	9	RO	1'b0	128-bit CAS Completer Supported. The XR9240 does not support 128-bit CAS completer, therefore this bit must be set to zero.
64ATMC_CMPLT_SPT	8	RO	1'b0	64-bit AtomicOp Completer Supported. The XR9240 does not support 64-bit atomicOp completer, therefore this bit must be set to zero.
32ATMC_CMPLT_SPT	7	RO	1'b0	32-bit AtomicOp Completer Supported. The XR9240 does not support 32-bit atomicOp completer, therefore this bit must be set to zero.
ATMC_RT_SPT	6	RO	1'b0	AtomicOp Routing Supported. The XR9240 does not support atomicOp routing, therefore this bit must be set to zero.
ARI_FWD_SPT	5	RO	1'b0	ARI Forwarding Supported. The XR9240 does not support ARI forwarding, therefore this bit must be set to zero.
CPLT_TMOUT_DIS_SPT	4	RO	1'b1	Completion Timeout Disable Supported. This feature is supported by the XR9240 device.
CPLT_TMOUT_SPT [3:0]	3:0	HwInit	4'b1111	Completion Timeout Ranges Supported. The XR9240 supports all valid completion timeout values.

8.4.7 Device Control 2 Register

The Device Control 2 register controls the optional PCI Express device specific parameters.

Offset 0x0098



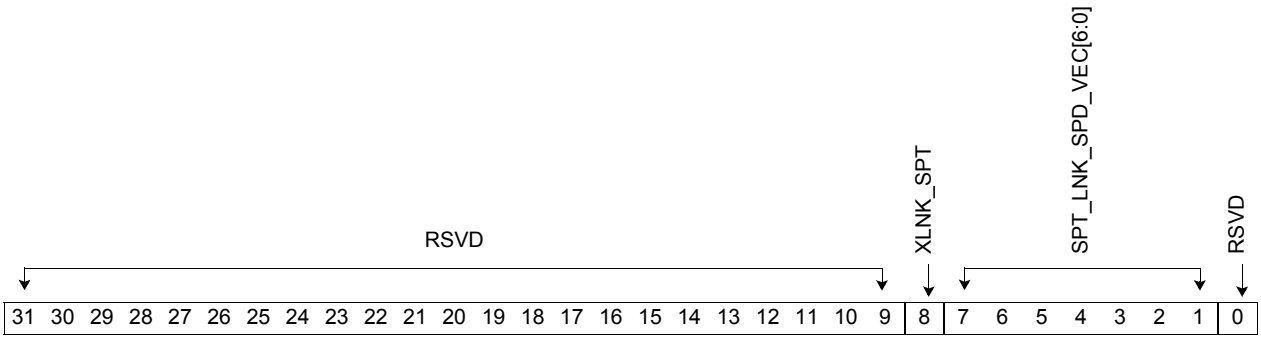
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:16	RO	16'h00000	Reserved.
TLP_PRFX_BLK	15	RW	1'b0	End-End TLP Prefix Blocking. This bit may be set to enable end-to-end TLP prefix blocking.
OBFF_EN[1:0]	14:13	RW	2'b00	OBFF Enable. This field must be set to zero for the XR9240.
RSVD	12:11	RO	2'b00	Reserved.
LTR_EN	10	RW	1'b0	LTR Mechanism Enable. This bit may be set at any time to enable LTR.
IDO_CMPT_EN	9	RW	1'b0	IDO Completion Enable. This bit may be set at any time to enable IDO completions.
IDO_REQ_EN	8	RW	1'b0	IDO Request Enable. This bit may be set at any time to enable IDO requests.
ATMOP_EGR_BLK	7	RW	1'b0	AtomicOp Egress Blocking. This bit must be set to zero for the XR9240.
ATMOP_REQ_EN	6	RW	1'b0	AtomicOp Requester Enable. This bit must be set to zero for the XR9240.
ARI_FWD_EN	5	RW	1'b0	ARI Forwarding Enable. This bit may be set at any time to enable ARI forwarding.

Field Name	Bits	Type	XR9240 Value	Description
CMPLT_TMOUT_DIS	4	RW	1'b0	Completion Timeout Disable. This bit is always set to zero for the XR9240.
CPLT_TMOUT_VAL [3:0]	3:0	RW	4'b0000	Completion Timeout Value. 0000 = Default range: 50 µs to 50 ms 0001 = 50 µs to 100 µs 0010 = 1 ms to 10 ms 0101 = 16 ms to 55 ms 0110 = 65 ms to 210 ms 1001 = 260 ms to 900 ms 1010 = 1 s to 3.5 s 1101 = 4 s to 13 s 1110 = 17 s to 64 s

8.4.8 Link Capabilities 2 Register

The Link Capabilities 2 register controls the optional PCI Express link parameters.

Offset 0x009C

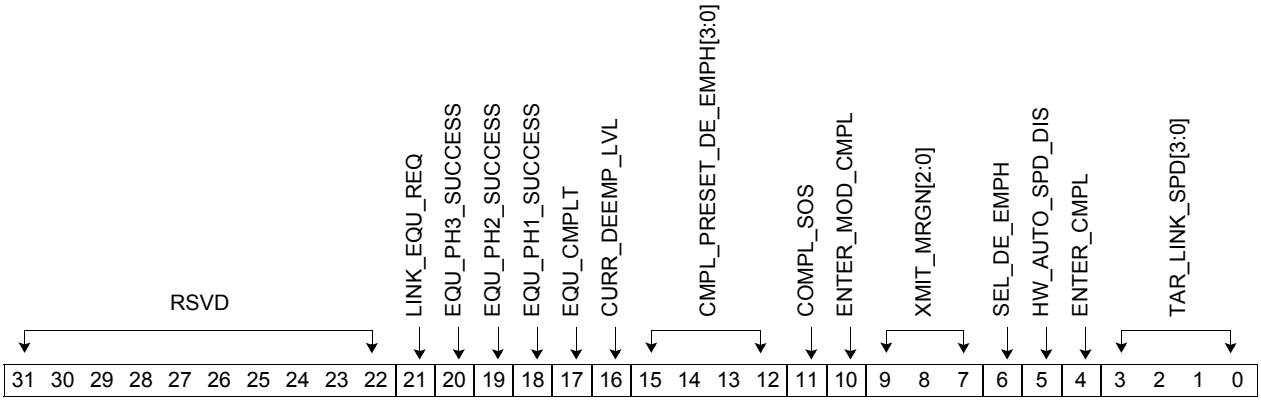


Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:9	RO	23'h00000	Reserved.
XLNK_SPT	8	RO	1'b0	Crosslink Supported. This feature is not supported by the XR9240.
SPT_LNK_SPD_VEC [6:0]	7:1	RW	7'b000 0111	Supported Link Speeds Vector. 0000001 = Gen1 2.5 GT/s 0000011 = Gen2 5.0 GT/s 0000111 = Gen3 8.0 GT/s
RSVD	0	RO	1'b0	Reserved.

8.4.9 Link Status and Control 2 Register

The Link Control and Status 2 register controls the optional PCI Express Link parameters.

Offset 0x00A0



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:22	RO	10'b0000000000	Reserved.
LINK_EQU_REQ	21	RW1C	1'b0	Link Equalization Request Status.
EQU_PH3_SUCCESS	20	ROS	1'b0	Equalization Phase 3 Successful. This bit is used for testing compliance.
EQU_PH2_SUCCESS	19	ROS	1'b0	Equalization Phase 2 Successful. This bit is used for testing compliance.
EQU_PH1_SUCCESS	18	ROS	1'b0	Equalization Phase 1 Successful. This bit is used for testing compliance.
EQU_CMPLT	17	ROS	1'b0	Equalization Complete Status.
CURR_DEEMP_LVL	16	ROS	1'b1	Current De-emphasis Level Status.
CMPL_PRESET_DE_EMPH[3:0]	15:12	RO	4'b0000	Compliance Preset/De-emphasis. This bit is used for testing compliance.
COMPL_SOS	11	RW	1'b0	Compliance SOS. This bit is used for testing compliance.
ENTER_MOD_CMPL	10	RWS	1'b0	Enter Modified Compliance. This bit is used for testing compliance.

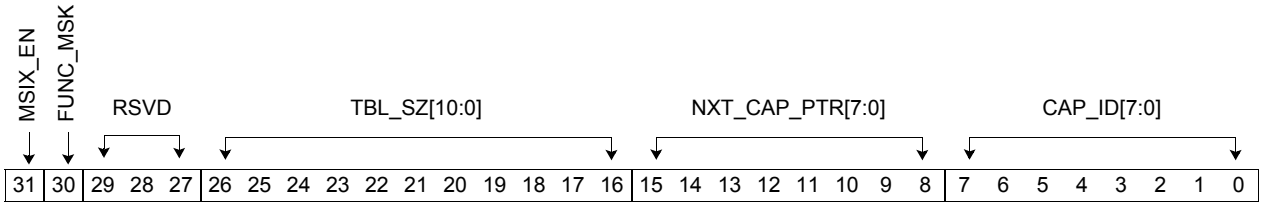
Field Name	Bits	Type	XR9240 Value	Description
XMIT_MRGN[2:0]	9:7	RWS	3'b000	Transmit Margin. 000: 800-1200 mV for full swing 400-600 mV for half-swing 001-010: values must be monotonic with a non-zero slope 011: 200-400 mV for full-swing and 100-200 mV for half-swing 100-111: Reserved
SEL_DE_EMPH	6	HWINIT	1'b0	Selectable De-emphasis. This bit is not applicable to the XR9240.
HW_AUTO_SPD_DIS	5	RO	1'b0	Hardware Autonomous Speed Disable.
ENTER_CMPL	4	RWS	1'b0	Enter Compliance. This bit is used for testing compliance.
TAR_LINK_SPD[3:0]	3:0	RWS	4'b0011	Target Link Speed. The XR9240 supports Gen3 speeds.

8.5 MSI-X Capability Registers

The MSI-X Capability registers indicate the XR9240 Message Signaled Interrupt Extended capabilities.

8.5.1 MSI-X Capability Register

Offset 0x00B0

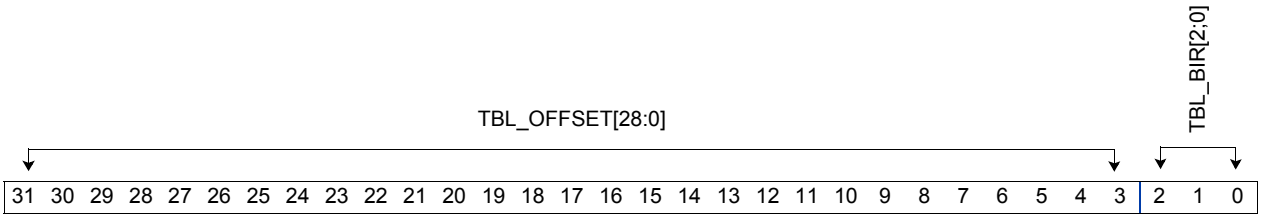


Field Name	Bits	Type	XR9240 Value	Description
MSIX_EN	31	RW	1'b0	MSI-X Enable. System configuration software sets this bit to enable MSI-X. 0 = XR9240 is prohibited from using MSI-x to request service 1 = If the MSI Enable bit in the "MSI Message Address Register" is 0, the XR9240 is permitted to use MSI-x to request service
FUNC_MSK	30	RW	1'b0	Function Mask. 0 = Each vector's mask bit determines whether the vector is masked or not 1 = All vectors associated with the function are masked, regardless of their per-vector mask bit
RSVD	29:27	RO	3'b000	Reserved.
TBL_SZ[10:0]	26:16	RO	11'b00000 010000	Table Size. System software reads this field to determine the MSI-X Table Size N. The maximum value for the XR9240 is 17.
NXT_CAP_PTR[7:0]	15:8	RO	8'h01	Next Pointer.
CAP_ID[7:0]	7:0	RO	8'h11	MSI-X Capability Identifier. 11h = MSI-X capable

8.5.2 MSI-X Table Offset Register

The MSI-X Table Offset register indicates which BAR is associated to the function and indicates the QWORD-aligned Table Offset where the structure begins relative to the base address associated with the BAR.

Offset 0x00B4

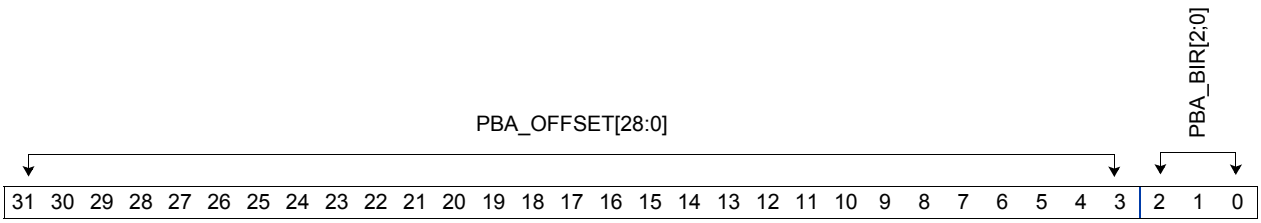


Field Name	Bits	Type	XR9240 Value	Description
TBL_OFFSET[28:0]	31:3	RO	28'h00 0000	Table Offset. Base address of the MSI-X Table, as an offset from the base address of the BAR indicated by the Table BIR bits.
TBL_BIR[2:0]	2:0	RO	3'b011	Table BAR Indicator Register (BIR). Indicates which BAR is used to map the MSI-X Table into memory space: 000 = BAR0 001 = BAR1 010 = BAR2 011 = BAR3 100 = BAR4 101 = BAR5 110 = Reserved 111 = Reserved

8.5.3 MSI-X PBA Offset Register

The MSI-X Pending Bit Array (PBA) Offset register points to a structure that contains the function's Pending Bits, one per table entry, organized as a packed array of bits within QWORDs. The last QWORD will not necessarily be fully populated.

Offset 0x00B8



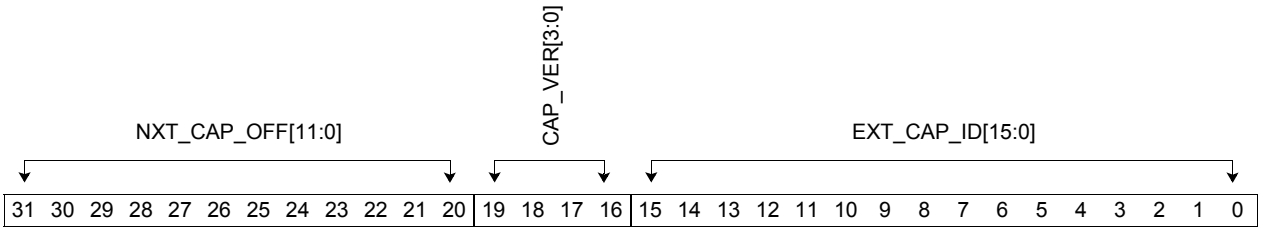
Field Name	Bits	Type	XR9240 Value	Description
PBA_OFFSET[28:0]	31:3	RO	29'h20	PBA Offset. Used as an offset from the address contained by one of the XR9240's Base Address registers to point to the base of the MSI-X PBA.
PBA_BIR[2:0]	2:0	RO	3'b011	PBA BAR Indicator Register (BIR). Indicates which BAR is used to map the MSI-X PBA into memory space: 000 = BAR0 001 = BAR1 010 = BAR2 011 = BAR3 100 = BAR4 101 = BAR5 110 = Reserved 111 = Reserved

8.6 Advanced Error Reporting Capability Registers

The Advanced Error Reporting Capability registers describe how the XR9240 device supports advanced error control and reporting.

8.6.1 Advanced Error Reporting Extended Capability Header Register

Offset 0x0100

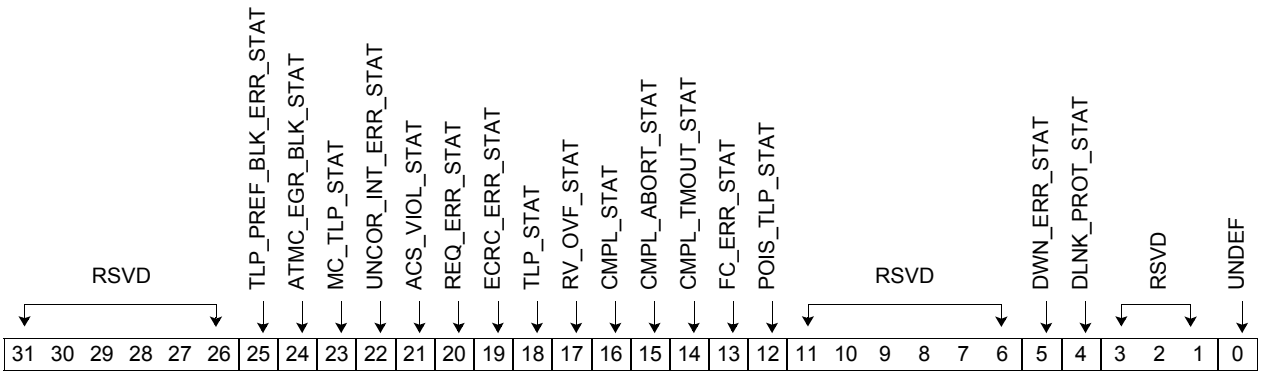


Field Name	Bits	Type	XR9240 Value	Description
NXT_CAP_OFF[11:0]	31:20	RO	12'h148	Next Capability Offset. This field contains the offset to the next PCI Express capability structure or 000h if no other items exist in the linked list of capabilities.
CAP_VER[3:0]	19:16	RO	4'b0010	Capability Version. This field is the version number of the capability structure present.
EXT_CAP_ID[15:0]	15:0	RO	16'h0001	Advanced Error Reporting Extended Capability ID. The Extended Capability ID for the Advanced Error Reporting Capability is 0001h.

8.6.2 Uncorrectable Error Status Register

The Uncorrectable Error Status register indicates the status of errors on a XR9240 device. An individual error status bit that is set indicates that a particular error was detected; software may clear an error status by writing a one to the respective bit.

Offset 0x0104



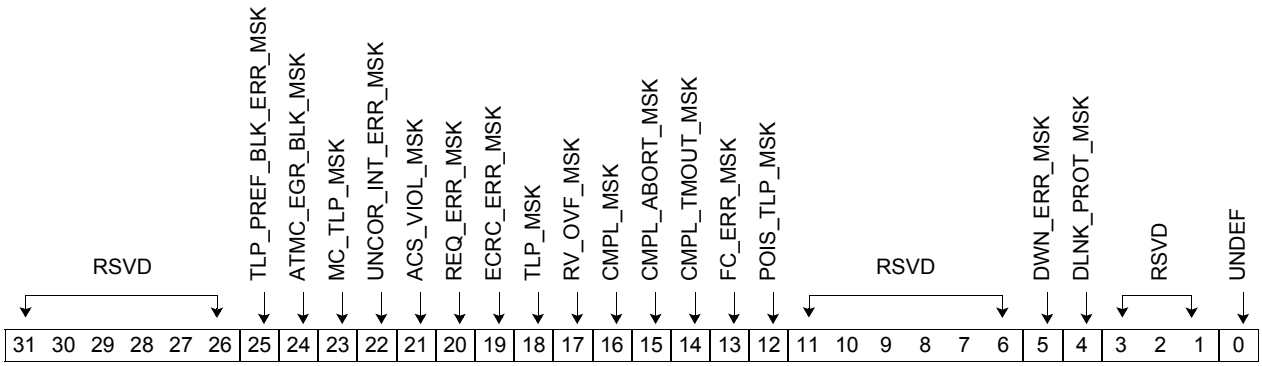
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:26	RO	6'b000000	Reserved.
TLP_PREF_BLK_ERR_STAT	25	RW1CS	1'b0	TLP Prefix Blocked Error Status.
ATMC_EGR_BLK_STAT	24	RW1CS	1'b0	AtomicOp Egress Blocked Status.
MC_TLP_STAT	23	RW1CS	1'b0	MC Blocked TLP Status.
UNCOR_INT_ERR_STAT	22	RW1CS	1'b0	Uncorrectable Internal Error Status.
ACS_VIOL_STAT	21	RW1CS	1'b0	ACS Violation Status.
REQ_ERR_STAT	20	RW1CS	1'b0	Unsupported Request Error Status.
ECRC_ERR_STAT	19	RW1CS	1'b0	ECRC Error Status.
TLP_STAT	18	RW1CS	1'b0	Malformed TLP Status.
RV_OVF_STAT	17	RW1CS	1'b0	Receiver Overflow Status.
CMPL_STAT	16	RW1CS	1'b0	Unexpected Completion Status.
CMPL_ABORT_STAT	15	RW1CS	1'b0	Completer Abort Status.
CMPL_TMOUT_STAT	14	RW1CS	1'b0	Completion Timeout Status.
FC_ERR_STAT	13	RW1CS	1'b0	Flow Control Protocol Error Status.
POIS_TLP_STAT	12	RW1CS	1'b0	Poisoned TLP Status.
RSVD	11:6	RO	6'b000000	Reserved.
DWN_ERR_STAT	5	RW1CS	1'b0	Surprise Down Error Status.

Field Name	Bits	Type	XR9240 Value	Description
DLNK_PROT_STAT	4	RW1CS	1'b0	Data Link Protocol Error Status.
RSVD	3:1	RO	3'b000	Reserved.
UNDEF	0	RO	1'b0	Undefined. This legacy bit is no longer used.

8.6.3 Uncorrectable Error Mask Register

The Uncorrectable Error Mask register controls reporting of errors by the XR9240 device to the PCI Express Root Complex via a PCI Express error Message. A masked error (respective bit set to 1b in the "Correctable Error Mask Register") is not logged in the "Header Log Register", does not update the First Error Pointer field in the "Advanced Error Capabilities and Control Register", and is not reported to the PCI Express Root Complex by a XR9240 device.

Offset 0x0108



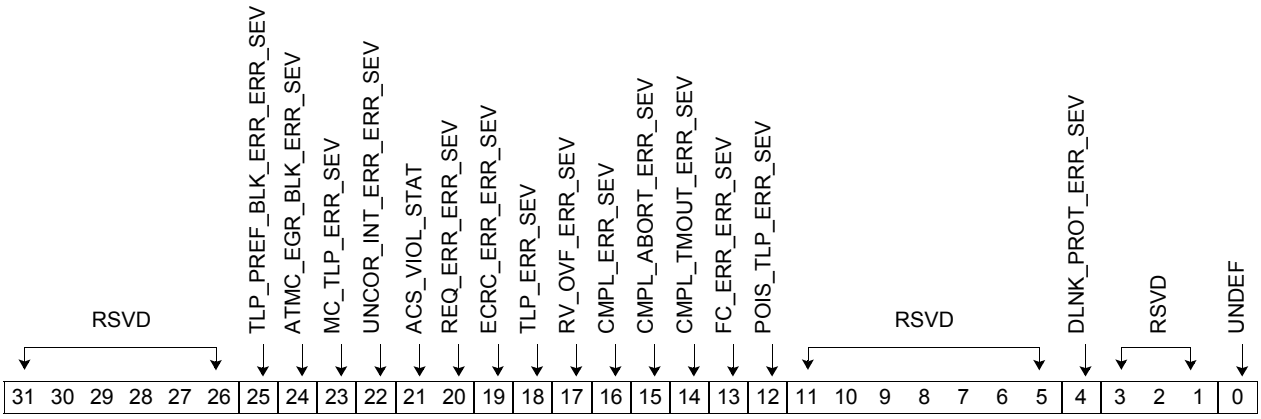
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:26	RO	6'b000000	Reserved.
TLP_PREF_BLK_ERR_MSK	25	RWS	1'b0	TLP Prefix Blocked Error Mask.
ATMC_EGR_BLK_MSK	24	RWS	1'b0	AtomicOp Egress Blocked Mask.
MC_TLP_MSK	23	RWS	1'b0	MC Blocked TLP Mask.
UNCOR_INT_ERR_MSK	22	RWS	1'b1	Uncorrectable Internal Error Mask.
ACS_VIOL_MSK	21	RWS	1'b0	ACS Violation Mask.
REQ_ERR_MSK	20	RWS	1'b0	Unsupported Request Error Mask.
ECRC_ERR_MSK	19	RWS	1'b0	ECRC Error Mask.
TLP_MSK	18	RWS	1'b0	Malformed TLP Mask.
RV_OVF_MSK	17	RWS	1'b0	Receiver Overflow Mask.
CMPL_MSK	16	RWS	1'b0	Unexpected Completion Mask.
CMPL_ABORTMSK	15	RWS	1'b0	Completer Abort Mask.
CMPL_TMOUT_MSK	14	RWS	1'b0	Completion Timeout Mask.
FC_ERR_MSK	13	RWS	1'b0	Flow Control Protocol Error Mask.
POIS_TLP_MSK	12	RWS	1'b0	Poisoned TLP Mask.

Field Name	Bits	Type	XR9240 Value	Description
RSVD	11:6	RO	6'b000000	Reserved.
DWN_ERR_MSK	5	RWS	1'b0	Surprise Down Error Mask (Optional).
DLNK_PROT_MSK	4	RWS	1'b0	Data Link Protocol Error Mask.
RSVD	3:1	RO	3'b000	Reserved.
UNDEF	0	RO	1'b0	Undefined. This legacy bit is no longer used.

8.6.4 Uncorrectable Error Severity Register

The Uncorrectable Error Severity register controls whether an error is reported as a nonfatal or fatal error. An error is reported as fatal when the corresponding error bit in this register is set. If the bit is cleared, the error is considered non-fatal.

Offset 0x010C



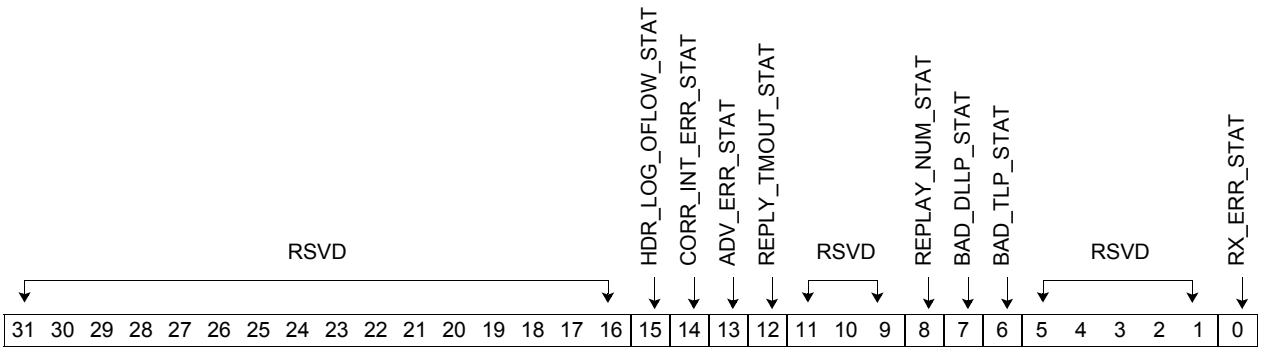
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:26	RO	6'b000000	Reserved.
TLP_PREF_BLK_ERR_SEV	25	RO	1'b0	TLP Prefix Blocked Error Severity.
ATMC_EGR_BLK_SEV	24	RO	1'b0	AtomicOp Egress Blocked Severity.
MC_TLP_SEV	23	RO	1'b0	MC Blocked TLP Severity.
UNCOR_INT_ERR_SEV	22	RWS	1'b1	Uncorrectable Internal Error Severity.
ACS_VIOL_SEV	21	RWS	1'b0	ACS Violation Severity.
REQ_ERR_SEV	20	RWS	1'b0	Unsupported Request Error Severity.
ECRC_ERR_SEV	19	RWS	1'b0	ECRC Error Severity.
TLP_SEV	18	RWS	1'b1	Malformed TLP Severity.
RV_OVF_SEV	17	RWS	1'b1	Receiver Overflow Severity.
CMPL_SEV	16	RWS	1'b0	Unexpected Completion Severity.
CMPL_ABORT_SEV	15	RWS	1'b0	Completer Abort Severity.
CMPL_TMOUT_SEV	14	RWS	1'b0	Completion Timeout Severity.
FC_ERR_SEV	13	RWS	1'b1	Flow Control Protocol Error Severity.
POIS_TLP_SEV	12	RWS	1'b0	Poisoned TLP Severity.
RSVD	11:5	RO	7'b000 0001	Reserved.

Field Name	Bits	Type	XR9240 Value	Description
DLNK_PROT_SEV	4	RWS	1'b1	Data Link Protocol Error Severity.
RSVD	3:1	RO	3'b000	Reserved.
UNDEF	0	RO	1'b0	Undefined.

8.6.5 Correctable Error Status Register

The Correctable Error Status register reports the status of correctable error sources on a XR9240 device. When an individual error status bit is set, it indicates that a particular error occurred. Software may clear an error status by writing a 1 to the respective bit.

Offset 0x0110

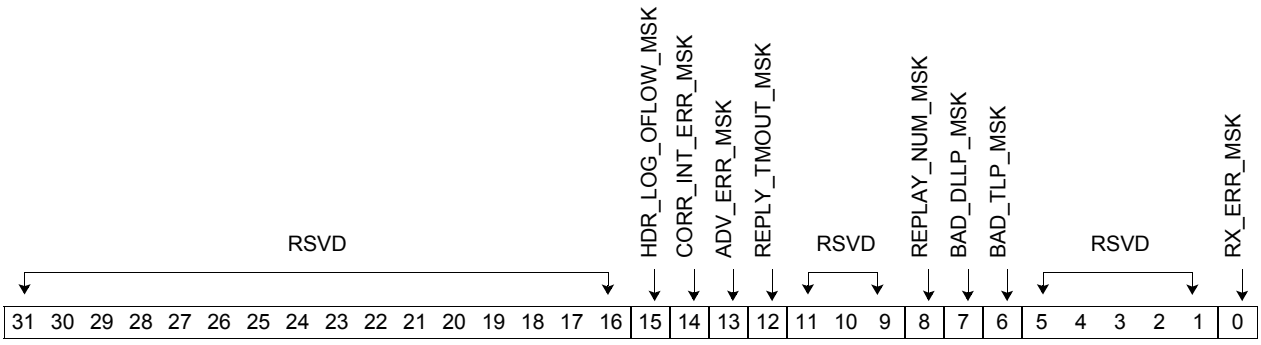


Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:16	RO	16'h0000	Reserved.
HDR_LOG_OFLOW_ STAT	15	RW1CS	1'b0	Header Log Overflow Status.
CORR_INT_ERR_ STAT	14	RW1CS	1'b0	Corrected Internal Error Status.
ADV_ERR_STAT	13	RW1CS	1'b0	Advisory Non-Fatal Error Status.
REPLY_TMOUT_STAT	12	RW1CS	1'b0	Replay Timer Timeout Status.
RSVD	11:9	RO	3'b000	Reserved.
REPLAY_NUM_STAT	8	RW1CS	1'b0	REPLAY_NUM Rollover Status.
BAD_DLLP_STAT	7	RW1CS	1'b0	Bad DLLP Status.
BAD_TLP_STAT	6	RW1CS	1'b0	Bad TLP Status.
RSVD	5:1	RO	5'b00000	Reserved.
RX_ERR_STAT	0	RW1CS	1'b0	Receiver Error Status.

8.6.6 Correctable Error Mask Register

The Correctable Error Mask register controls reporting of correctable errors by the XR9240 to the PCI Express Root Complex via a PCI Express error message. A masked error (respective bit set) is not reported to the PCI Express Root Complex by an individual device.

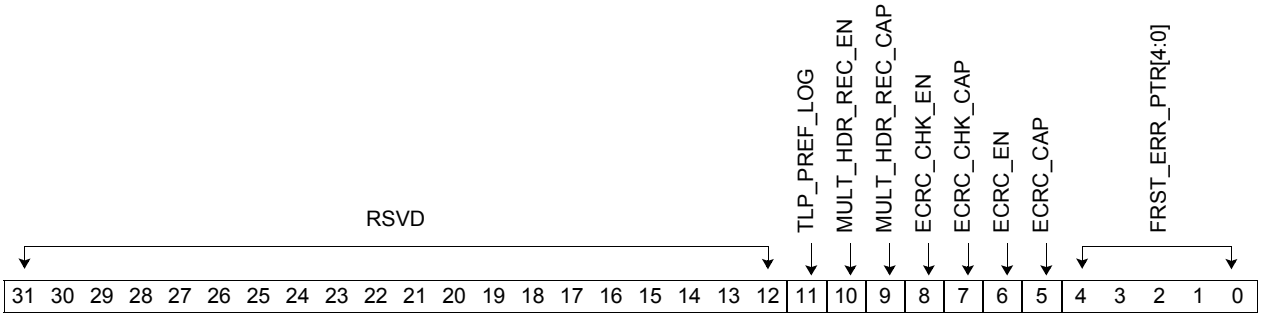
Offset 0x0114



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:16	RO	16'h0000	Reserved.
HDR_LOG_OFLOW_MSK	15	RO	1'b0	Header Log Overflow Mask.
CORR_INT_ERR_MSK	14	RWS	1'b1	Corrected Internal Error Mask.
ADV_ERR_MSK	13	RWS	1'b1	Advisory Non-Fatal Error Mask.
REPLY_TMOUT_MSK	12	RWS	1'b0	Replay Timer Timeout Mask.
RSVD	11:9	RO	3'b000	Reserved.
REPLAY_NUM_MSK	8	RWS	1'b0	REPLAY_NUM Rollover Mask.
BAD_DLLP_MSK	7	RWS	1'b0	Bad DLLP Mask.
BAD_TLP_MSK	6	RWS	1'b0	Bad TLP Mask.
RSVD	5:1	RO	5'b00000	Reserved.
RX_ERR_MSK	0	RWS	1'b0	Receiver Error Mask.

8.6.7 Advanced Error Capabilities and Control Register

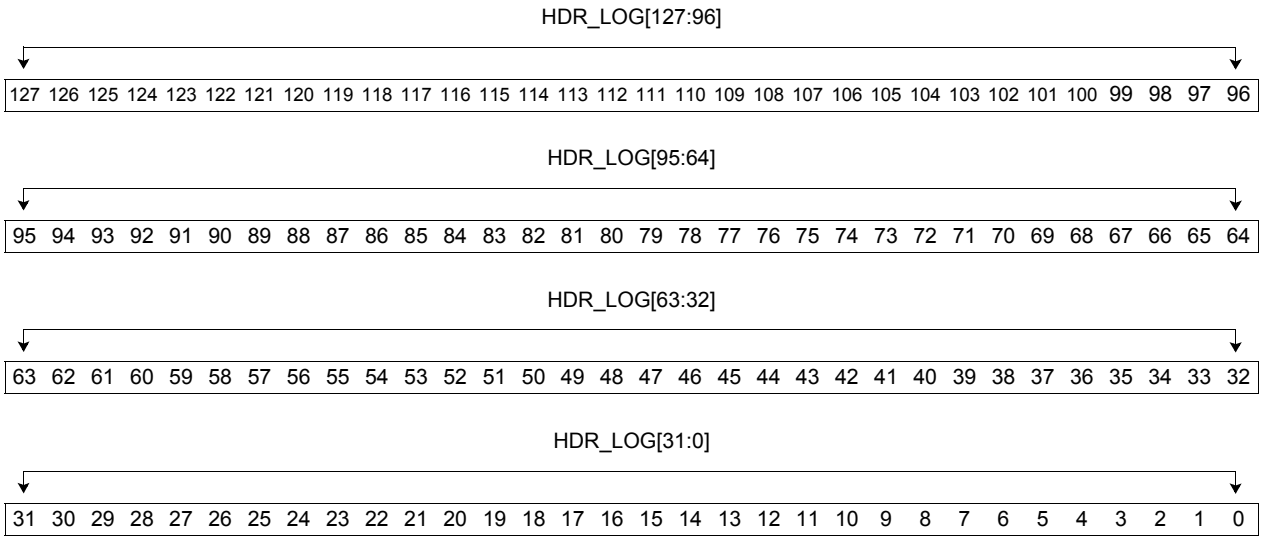
Offset 0x0118



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:12	RO	20'h00000	Reserved.
TLP_PREF_LOG	11	RO	1'b0	TLP Prefix Log Present. If set and the First Error Pointer is valid, indicates that the TLP Prefix Log register contains valid information. If clear or if First Error Pointer is invalid, the TLP Prefix Log register is undefined.
MULT_HDR_REC_EN	10	RO	1'b0	Multiple Header Recording Enable. When set, this bit enables the function to record more than one error header.
MULT_HDR_REC_ CAP	9	RO	1'b0	Multiple Header Recording Capable. If set, this bit indicates that the function is capable of recording more than one error header.
ECRC_CHK_EN	8	RWS	1'b0	ECRC Check Enable. This bit when set enables ECRC checking.
ECRC_CHK_CAP	7	RO	1'b1	ECRC Check Capable. This bit indicates that the XR9240 is capable of checking ECRC.
ECRC_EN	6	RWS	1'b0	ECRC Generation Enable. This bit when set enables ECRC generation.
ECRC_CAP	5	RO	1'b1	ECRC Generation Capable. This bit indicates that the XR9240 is capable of generating ECRC
FRST_ERR_PTR[4:0]	4:0	ROS	5'b00000	First Error Pointer. The First Error Pointer is a read-only register that identifies the bit position of the first error reported in the Uncorrectable Error Status register.

8.6.8 Header Log Register

Offset 0x011C - 0x0128



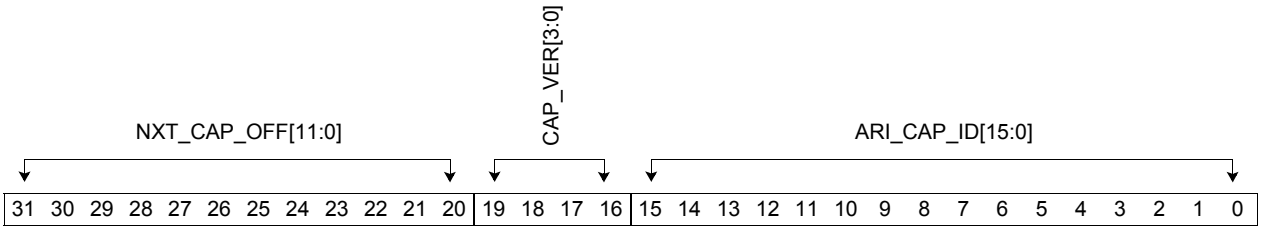
Field Name	Bits	Type	XR9240 Value	Description
HDR_LOG[127:0]	127:0	RO	All zeroes	Header Log Data.

8.7 ARI Capability Registers

The XR9240 device supports Alternative Routing-ID Interpretation.

8.7.1 ARI Capability Header Register

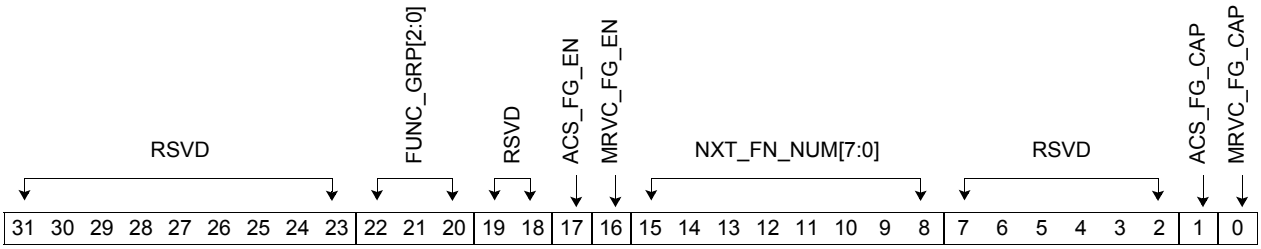
Offset 0x0148



Field Name	Bits	Type	XR9240 Value	Description
NXT_CAP_OFF[11:0]	31:20	RO	12'h158	Next Capability Offset. This field contains the offset to the next PCI Express capability structure.
CAP_VER[3:0]	19:16	RO	4'b0001	Capability Version. This field is the version number of the capability structure present.
ARI_CAP_ID[15:0]	15:0	RO	16'h000E	ARI Capability ID. The ARI Capability ID for the XR9240 device is 000Eh.

8.7.2 ARI Capability and Control Register

Offset 0x014C



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:23	RW	9'h000	Reserved.
FUNC_GRP[2:0]	22:20	RW	3'b000	Function Group. This field assigns a Function Group Number to this function.
RSVD	19:18	RW	2'b00	Reserved.
ACS_FG_EN	17	RW	1'b0	ACS Function Groups Enable. This bit is only applicable for function 0 and must be hardwired to 0b for all other functions. When set, each function in the ARI Device must associate bits within its Egress Control Vector with function group numbers rather than function numbers.
MRVC_FG_EN	16	RW	1'b0	MRVC Function Groups Enable. This bit is only applicable for Function 0 and must be hardwired to 0b for all other functions. When set, the ARI Device must interpret entries in its Function Arbitration Table as function group numbers rather than function numbers.
NXT_FN_NUM[7:0]	15:8	RO	8'h00	Next Function Number. This field indicates the function number of the next higher numbered Function in the Device, or 00h if there are no higher numbered functions. Function 0 starts this linked list of functions.
RSVD	7:2	RO	6'b000000	Reserved.

Field Name	Bits	Type	XR9240 Value	Description
ACS_FG_CAP	1	RO	1'b0	ACS Function Groups Capability. This bit is only applicable for function 0 and must be 0b for all other functions. If set, this bit indicates that the ARI Device supports Function Group level granularity for ACS P2P Egress Control via its ACS Capability structures.
MRVC_FG_CAP	0	RO	1'b0	MRVC Function Groups Capability. This bit is only applicable for function 0 and must be 0b for all other functions. If set, this bit indicates that the ARI Device supports Function Group level arbitration via its Multi-Function Virtual Channel (MFVC) Capability structure.

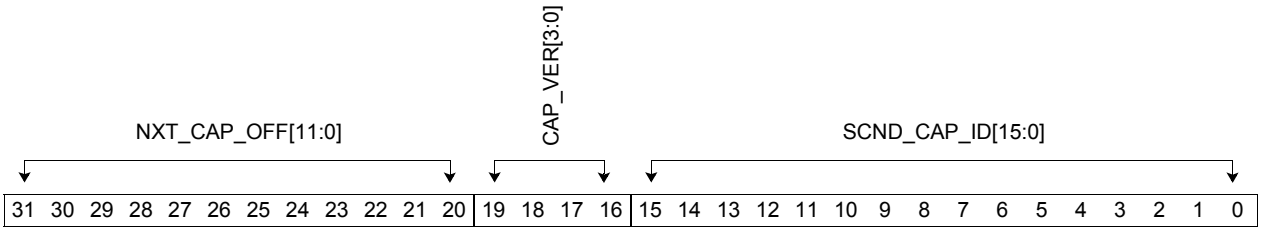
8.8 Secondary PCIe Capability Registers

As a port that supports link speeds up to 8 GT/s, the Secondary PCIe Capability registers are required for the XR9240 device. The Secondary PCIe Capability registers are only implemented in the XR9240 physical function (function 0). From the perspective of the host, the XR9240 device is always considered to be an upstream port.

The link speed is set by the Supported Link Speeds Vector field of the "Link Capabilities 2 Register".

8.8.1 Secondary PCIe Capability Header Register

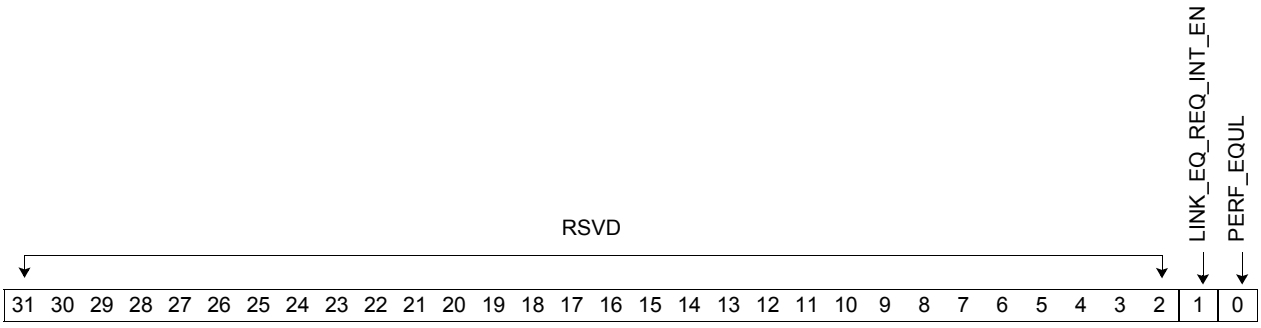
Offset 0x0158



Field Name	Bits	Type	XR9240 Value	Description
NXT_CAP_OFF[11:0]	31:20	RO	12'h178	Next Capability Offset. This field contains the offset to the next PCI Express capability structure.
CAP_VER[3:0]	19:16	RO	4'b0001	Capability Version. This field is the version number of the capability structure present.
SCND_CAP_ID[15:0]	15:0	RO	16'h0019	Secondary PCIe Capability ID.

8.8.2 Secondary PCIe Capability Link Control 3 Register

Offset 0x015C

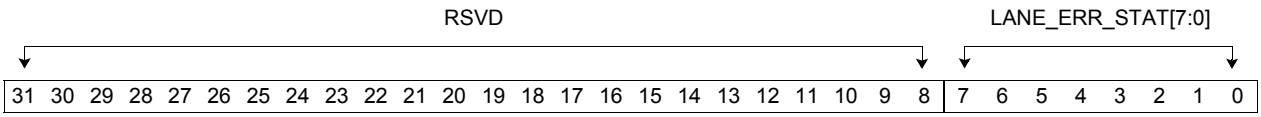


Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:2	RO	30'h0000 0000	Reserved.
LINK_EQ_REQ_INT_EN	1	RO	1'b0	Link Equalization Request Interrupt Enable. When Set, this bit enables the generation of an interrupt to indicate that the Link Equalization Request bit has been set.
PERF_EQUL	0	RO	1'b0	Perform Equalization. When this bit is set and the Retrain Link bit is set with the Target Link Speed field set to 8.0 GT/s in the Supported Link Speeds Vector field of the "Link Capabilities 2 Register", the down-stream port must perform link equalization.

8.8.3 Secondary PCIe Capability Lane Error Status Register

The Lane Error Status register consists of a 32-bit vector where each bit represents each lane's error status. The lane number is the default lane number which is invariant to link width and lane reversal negotiation that occurs during link training. The Maximum Link Width is set in the "Link Capabilities Register".

Offset 0x0160

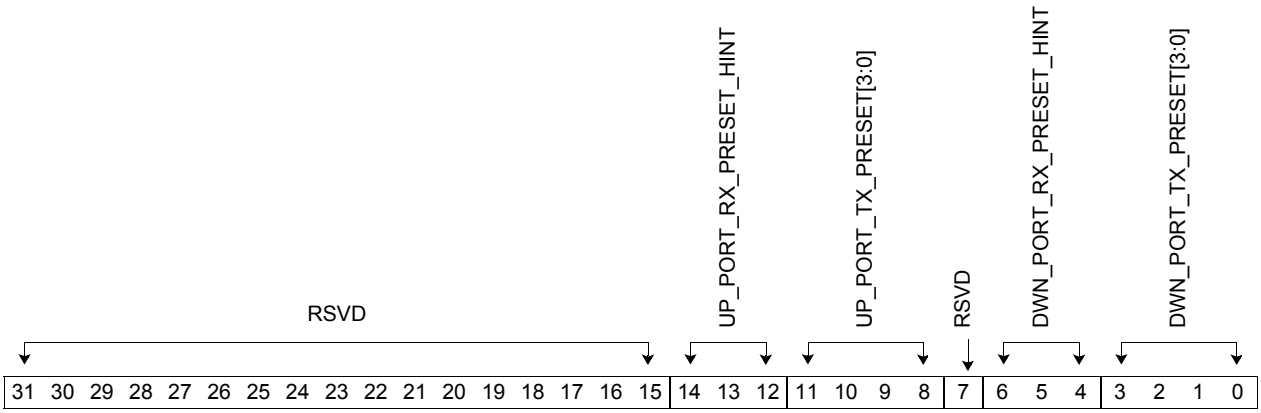


Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:8	RO	25'h0000 0000	Reserved.
LANE_ERR_STAT [7:0]	7:0	RW1CS	8'h00	Lane Error Status Bits. This field is a bit-wise representation of each lane's error status.

8.8.4 Secondary PCIe Capability Lane Equalization Control Register

The Lane Equalization Control register consists of control fields required for per lane equalization. The number of entries in this register are determined by Maximum Link Width field in the "Link Capabilities Register". The initial values in this register are applied to all lanes. During negotiation, each PHY will adjust to a unique value per lane.

Offset 0x0164



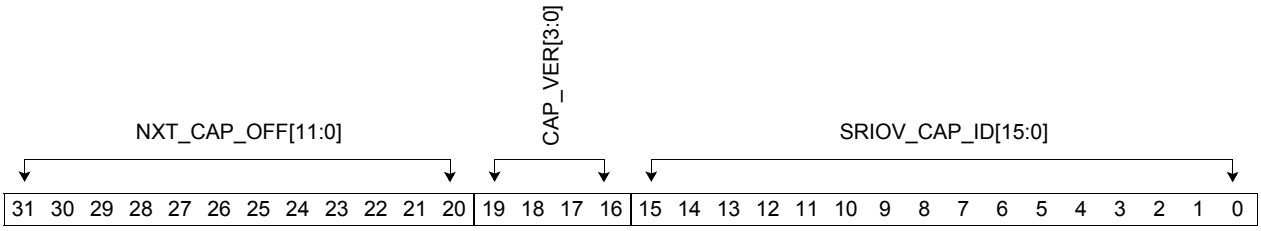
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:15	RO	17'h7F00	Reserved.
UP_PORT_RX_PRESET_HINT[2:0]	14:12	HwInit	3'b111	Upstream Port Receiver Preset Hint. This field does not apply to the XR9240.
UP_PORT_TX_PRESET[3:0]	11:8	HwInit	4'b111	Upstream Port Transmitter Preset. This field contains the transmit preset value sent or received during link equalization.
RSVD	7	RO	1'b0	Reserved.
DWN_PORT_RX_PRESET_HINT[2:0]	6:4	RsvdP	3'b000	Downstream Port Receiver Preset Hint. This field does not apply to the XR9240.
DWN_PORT_TX_PRESET[3:0]	3:0	RsvdP	4'b0000	Downstream Port Transmitter Preset. This field does not apply to the XR9240.

8.9 SR-IOV Extended Capability Registers

The XR9240 device supports Single Root I/O Virtualization (SR-IOV).

8.9.1 SR-IOV Extended Capability Header Register

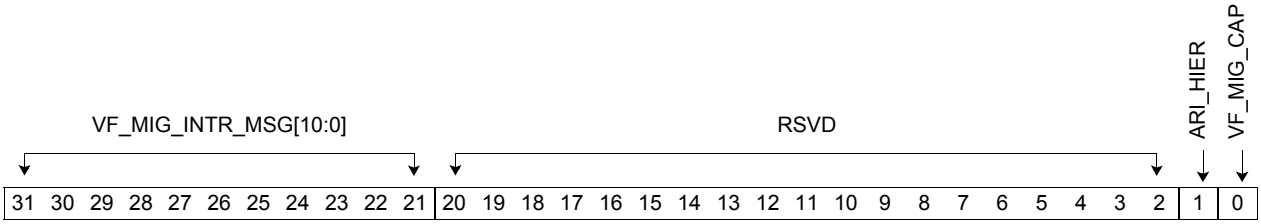
Offset 0x0178



Field Name	Bits	Type	XR9240 Value	Description
NXT_CAP_OFF[11:0]	31:20	RO	12'h1B8	Next Capability Offset. This field contains the offset to the next PCI Express capability structure.
CAP_VER[3:0]	19:16	RO	4'b0001	Capability Version. This field is the version number of the capability structure present.
SRIOV_CAP_ID [15:0]	15:0	RO	16'h0010	SR-IOV Capability ID. The SR-IOV Capability ID for the XR9240 device is 0010h.

8.9.2 SR-IOV Capabilities Register

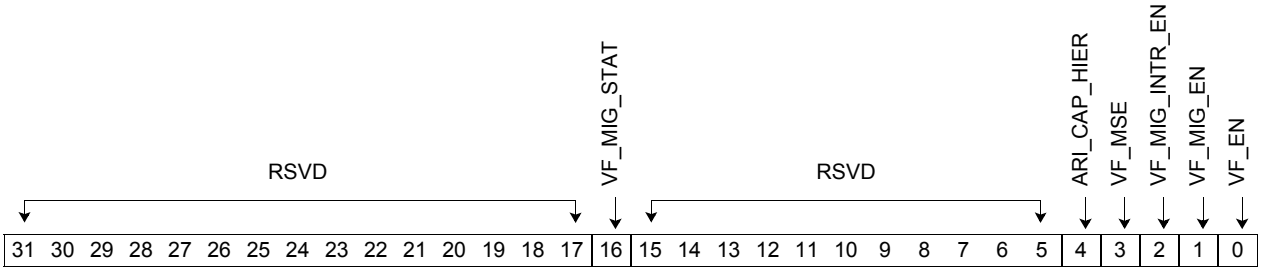
Offset 0x017C



Field Name	Bits	Type	XR9240 Value	Description
VF_MIG_INTR_MSG [10:0]	31:21	RO	11'h000	VF Migration Interrupt Message Number. MSI/MSI-X vector used for migration interrupts. The value in this field is undefined if bit 0 is set to zero.
RSVD	20:2	RO	19'h00000	Reserved.
ARI_HIER	1	RO	1'b1	ARI Capable Hierarchy Preserved. If set, the ARI Capable Hierarchy bit is preserved across certain power state transitions.
VF_MIG_CAP	0	RO	1'b0	VF Migration Capable. If set, the PF is Migration Capable and is operating under a Migration Capable MR-PCIM.

8.9.3 SR-IOV Control and Status Register

Offset 0x0180



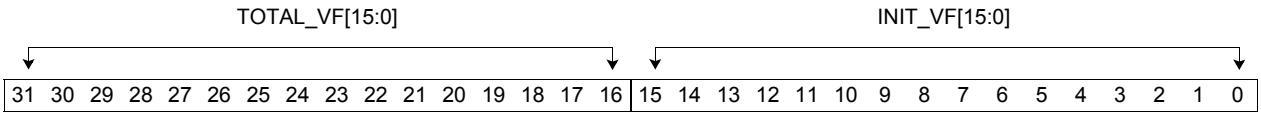
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:17	RsvdP	15'h0000	Reserved.
VF_MIG_STAT	16	RO	1'b0	VF Migration Status. When set, this bit indicates that a VF Migration In or Migration Out Request has been issued by MR-PCIM. To determine the cause of the event, software may scan the VF State Array.
RSVD	15:5	RsvdP	11'h000	Reserved.
ARI_CAP_HIER	4	RW	1'b0	ARI Capable Hierarchy. 0 = All PFs have non-ARI Capable Hierarchy 1 = All PFs have ARI Capable Hierarchy
VF_MSE	3	RW	1'b0	VF Memory Space Enable. When set, enables memory space for virtual functions.
VF_MIG_INTR_EN	2	RO	1'b0	VF Migration Interrupt Enable. When set, enables the VF migration state change interrupt.
VF_MIG_EN	1	RO	1'b0	VF Migration Enable. When set, enables VF migration support.
VF_EN	0	RW	1'b0	VF Enable. When set, enables VFs.

8.9.4 SR-IOV Total and Initial VFs Register

The Initial VFs register indicates the number of VFs that are initially associated with the PF. For the XR9240, the value for InitialVFs is the same as TotalVFs and is 128d.

The TotalVFs register indicates the maximum number of VFs that could be associated with the PF. For the XR9240, the value for Initial VFs is the same as TotalVFs and is 128d.

Offset 0x0184

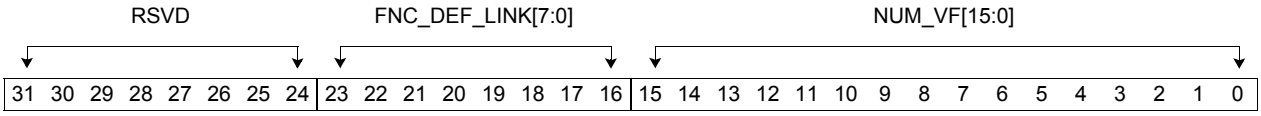


Field Name	Bits	Type	XR9240 Value	Description
TOTAL_VF[15:0]	31:16	RO	16'h0080	Total Number of Virtual Functions.
INIT_VF[15:0]	15:0	RO	16'h0080	Initial Number of Virtual Functions.

8.9.5 SR-IOV Function Dependency Link and Number VFs Register

The NumVFs register controls the number of visible VFs.

Offset 0x0188



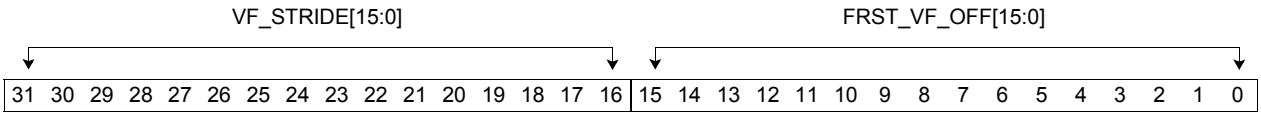
Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:24	RO	8'h00	Reserved.
FNC_DEF_LINK[7:0]	23:16	RO	8'h00	Function Dependency Link. This field is not used by the XR9240 device.
NUM_VF[15:0]	15:0	RW	16'h0000	Number of Virtual Functions that are visible.

8.9.6 SR-IOV VF Stride and First VF Offset Register

The First VF Offset register defines the Routing ID offset of the first VF that is associated with the XR9240 PF.

The VF Stride register defines the Routing ID offset from one VF to the next one for all VFs associated with the XR9240 PF.

Offset 0x018C

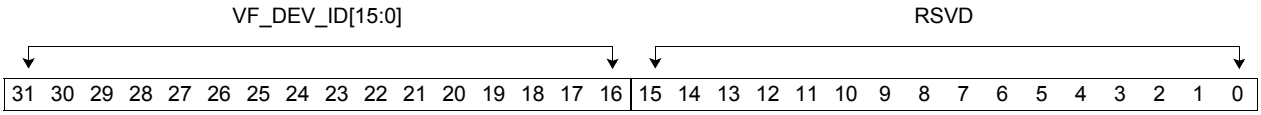


Field Name	Bits	Type	XR9240 Value	Description
VF_STRIDE[15:0]	31:16	RO	16'h0001	Virtual Function Stride.
FIRST_VF_OFF[15:0]	15:0	RO	16'h0100	First Virtual Function Offset.

8.9.7 SR-IOV VF Device ID Register

The VF Device ID register defines the device ID for each VF.

Offset 0x0190

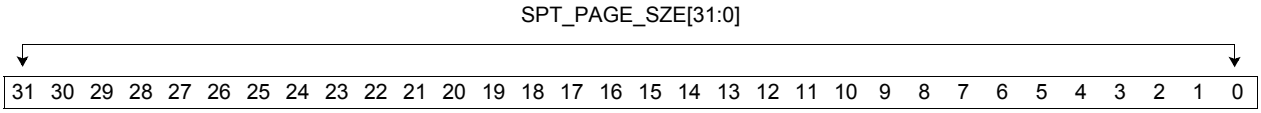


Field Name	Bits	Type	XR9240 Value	Description
VF_DEV_ID[15:0]	31:16	RO	16'h9200	Virtual Function Device Identifier.
RSVD	15:0	RO	16'h0000	Reserved.

8.9.8 SR-IOV Supported Page Sizes Register

The Supported Page Sizes register indicates the page sizes supported by the PF.

Offset 0x0194

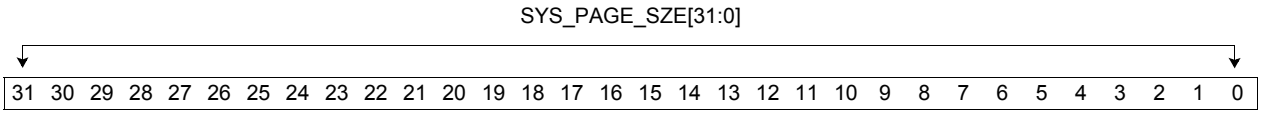


Field Name	Bits	Type	XR9240 Value	Description
SPT_PAGE_SZE [31:0]	31:0	RO	32'h000 00553	Physical Function Supported Page Size.

8.9.9 SR-IOV System Page Size Register

The System Page Sizes register defines the page size the system will use to map the VF's memory addresses.

Offset 0x0198

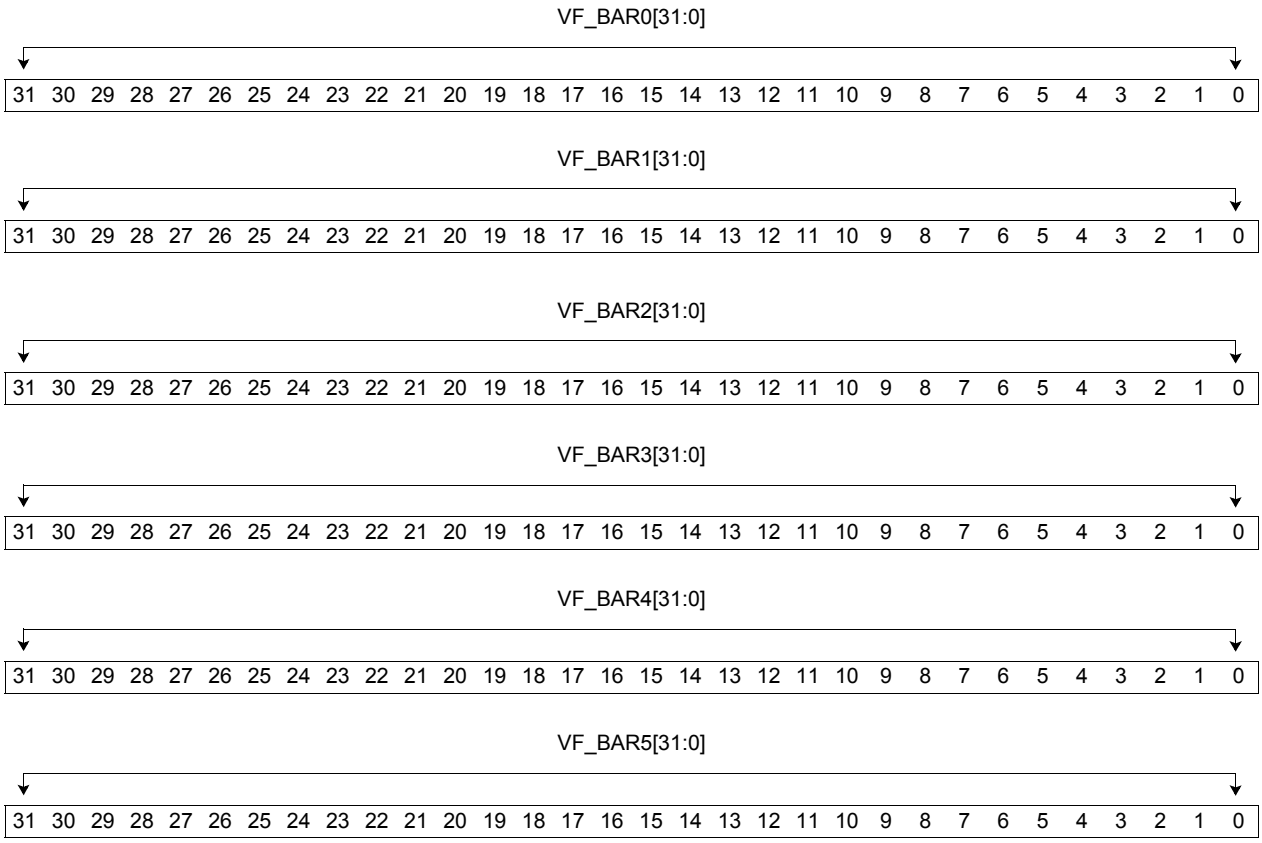


Field Name	Bits	Type	XR9240 Value	Description
SYS_PAGE_SZE [31:0]	31:0	RW	32'h000 00001	Physical Function System Page Size.

8.9.10 SR-IOV BAR Registers

The BAR registers define the VF's base addresses.

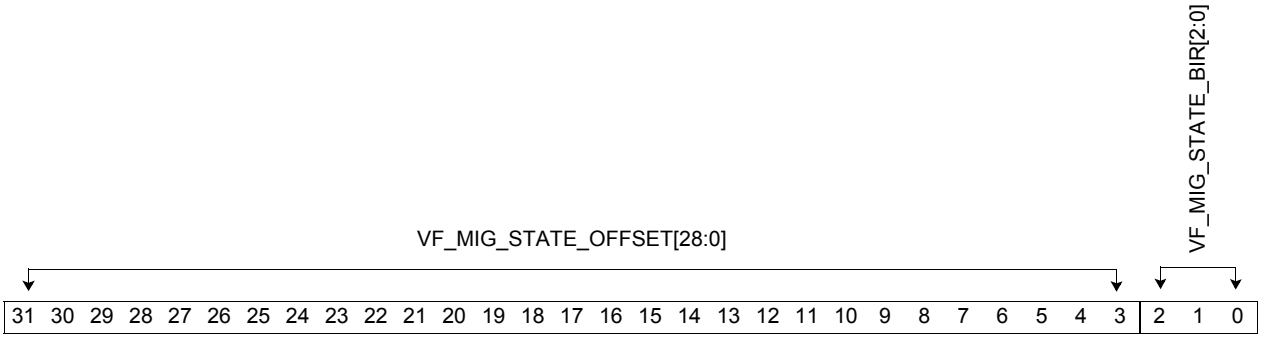
Offset	0x019C	BAR0
	0x01A0	BAR1
	0x01A4	BAR2
	0x01A8	BAR3
	0x01AC	BAR4
	0x01B0	BAR5



Field Name	Bits	Type	XR9240 Value	Description
VF_BARn[31:0]	31:0	RW	32'h000 00004 for BAR0, 32'h000 00000 for all others	Virtual Function BAR. BAR0 = BAR1 = BAR2 = BAR3 = BAR4 = BAR5 =

8.9.11 SR-IOV Migration State Array Offset Register

Offset 0x01B4

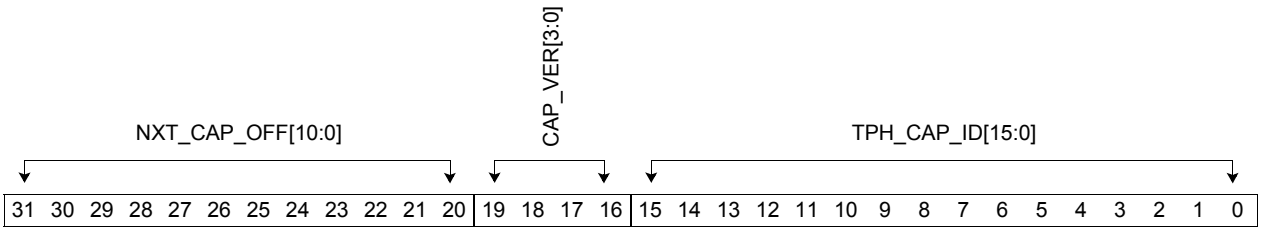


Field Name	Bits	Type	XR9240 Value	Description
VF_MIG_STATE_ OFFSET[28:0]	31:3	RO	29'h00000000	VF Migration State Offset. This field is used as an offset from the address contained by one of the function's BARs to point to the base of the VF migration state array. The lower three VF migration state BIR bits are masked off (set to zero) by software to form a 32-bit QWORD-aligned offset. This field is undefined if TotalVFs is 0.
VF_MIG_STATE_BIR [2:0]	2:0	RO	3'b000	VF Migration State BIR. This field indicates which one of a function's BARs is used to map the function's VF migration state array into memory space. For a 64-bit BAR, the VF migration state BIR indicates the lower DWORD. This field is undefined if TotalVFs is 0.

8.10 TPH Requestor Capability Registers

8.10.1 TPH Capability Header Register

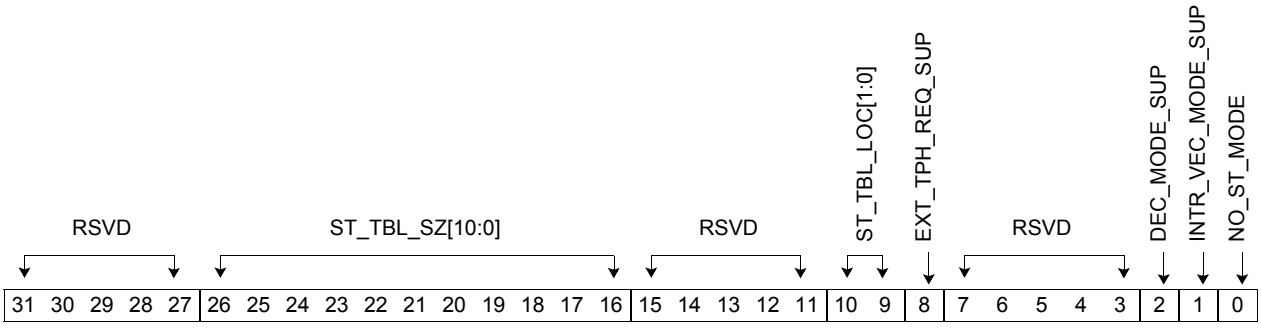
Offset 0x01B8



Field Name	Bits	Type	XR9240 Value	Description
NXT_CAP_OFF[11:0]	31:20	RO	12'h000	Next Capability Offset.
CAP_VER[3:0]	19:16	RO	4'b0001	Capability Version. Must be 1h for PCIe version 3.0.
TPH_CAP_ID [15:0]	15:0	RO	16'h0017	TPH Capability ID.

8.10.2 TPH Requester Capability Register

Offset 0x01BC

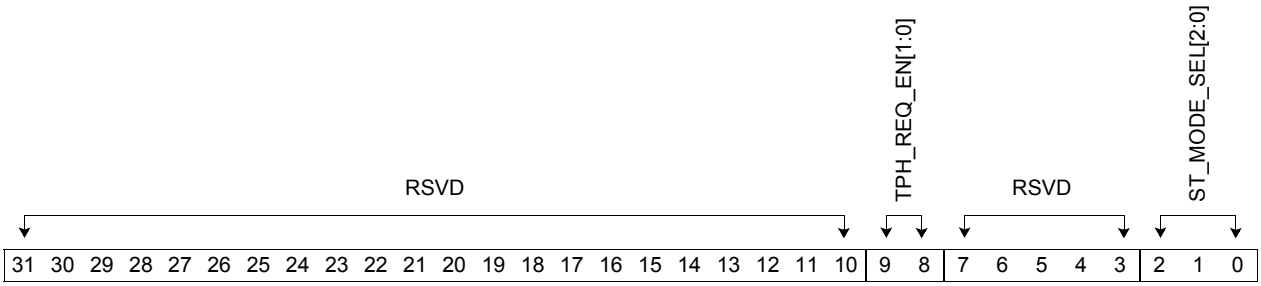


Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:27	RO	5'b00000	Reserved.
ST_TBL_SZ[10:0]	26:16	RO	11'b00000 000001	ST Table Size. This field indicates the maximum number of ST Table entries a function may use.
RSVD	15:11	RO	5'b00000	Reserved.
ST_TBL_LOC[1:0]	10:9	RO	2'b00	ST Table Location. 00b – ST Table is not present. 01b – ST Table is located in the TPH Requester Capability structure. 10b – ST Table is located in the MSI-X Table. 11b – Reserved
EXT_TPH_REQ_SUP	8	RO	1'b0	Extended TPH Requester Supported. If set, this bit indicates that the function is capable of generating Requests with a TPH TLP prefix.
RSVD	7:3	RO	5'b00000	Reserved.
DEC_MODE_SUP	2	RO	1'b1	Device Specific Mode Supported. If set, this bit indicates that the function supports the Device Specific Mode of operation.
INTR_VEC_MODE_ SUP	1	RO	1'b0	Interrupt Vector Mode Supported. If set, this bit indicates that the function supports the Interrupt Vector Mode of operation.

Field Name	Bits	Type	XR9240 Value	Description
NO_ST_MODE	0	RO	1'b1	No ST Mode Supported. If set, this bit indicates that the function supports the No ST Mode of operation. This mode is required by all functions that implement the TPH capability structure.

8.10.3 TPH Requester Control Register

Offset 0x01C0



Field Name	Bits	Type	XR9240 Value	Description
RSVD	31:10	RW	22'h00 0000	Reserved.
TPH_REQ_EN[1:0]	9:8	RW	2'b00	TPH Requester Enable. This field controls the ability to issue Request TLPs using either TPH or Extended TPH. 00b – Function operating as a Requester is not permitted to issue Requests with TPH or Extended TPH. 01b – Function operating as a Requester is permitted to issue Requests with TPH and is not permitted to issue Requests with Extended TPH. 10b – Reserved. 11b – Function operating as a Requester is permitted to issue Requests with TPH and Extended TPH.
RSVD	7:3	RW	5'b00000	Reserved.
ST_MODE_SEL[2:0]	2:0	RW	3'b000	ST Mode Select. 000b – No ST Mode 001b – Interrupt Vector Mode 010b – Device Specific Mode Al other encodings reserved.

9 Signal Definition

This chapter defines the XR9240 external interfaces and signals. All inputs without internal pull up resistors must be handled by the card design. All LVCMOS input signals are 1.8V.

9.1 PCI Express Interface

The XR9240 provides a PCIe x8 interface for communicating with the host.

Table 9-1. PCIe Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
PCIE_TX_P[7:0]	Output	CML		PCIe compliant differential positive transmit outputs for lanes 0 - 7. No termination is required for these pins.
PCIE_TX_N[7:0]	Output	CML		PCIe compliant differential negative transmit outputs for lanes 0 - 7. No termination is required for these pins.
PCIE_RX_P[7:0]	Input	CML		Differential positive receive inputs for lanes 0 - 7. No termination is required for these pins.
PCIE_RX_N[7:0]	Input	CML		Differential negative receive inputs for lanes 0 - 7. No termination is required for these pins.
PCIE_REFCLK_P	Input	LVDS		PCIe compliant 100 MHz differential positive reference clock.
PCIE_REFCLK_N	Input	LVDS		PCIe compliant 100 MHz differential negative reference clock.
PERST_N	Input	LVCMOS	Pull up	PCIe Reset. Assertion of this active low signal will reset the whole chip except the XR9240 sticky registers. This signal should be connected to PCIe slot reset. 0: Reset full chip except XR9240 sticky registers 1: No PCIe reset

9.2 Interlaken Interface

Table 9-2. Interlaken Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
ILK_TX_P[7:0]	Output	CML		ILK compliant differential positive transmit outputs for lanes 0 - 7. No termination is required for these pins.
ILK_TX_N[7:0]	Output	CML		ILK compliant differential negative transmit outputs for lanes 0 - 7. No termination is required for these pins.
ILK_RX_P[7:0]	Input	CML		Differential positive receive inputs for lanes 0 - 7. No termination is required for these pins.
ILK_RX_N[7:0]	Input	CML		Differential negative receive inputs for lanes 0 - 7. No termination is required for these pins.
ILK_REFCLK_P	Input	LVDS		ILK compliant 125 MHz differential positive reference clock, supplied by on card oscillator.
ILK_REFCLK_N	Input	LVDS		ILK compliant 125 MHz differential negative reference clock, supplied by on card oscillator.
ILK_BYPS_CLK	Input	LVCMOS		ILK Bypass Clock. Test signal that must be pulled down with a 1 ohm resistor.

9.3 GPIO Interface

Table 9-3. GPIO Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
GPIO[31:0]	Input/ Output	LVCMOS		General purpose IOs.
CONFIG_GPIO[7:0]	Input/ Output	LVCMOS		Alternate General purpose IO that may be used for configuration.

9.4 FPGA Field Programming Interface

This section describes the Programming Interface streaming bus. The host may download the FPGA image through this interface. Refer to the *XR9240 FPGA Design Application Note*, APN-0001 for more detailed information.

Table 9-4. Programming Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
FP_CLK_OUT	Output	LVCMOS		Field Programming Clock Output. This output may be used as the clock input for the programmable device on the card that is used to program the FPGA. The clock speed is configurable by the host using the PROG_CLK_DIV[1:0] field of the "Field Programming Interface Control Register".
FP_DATA[7:0]	Output	LVCMOS		Field Programming Data Output Bus. The data bus between the XR9240 and the external programmable device.
FP_VLD	Output	LVCMOS		Programming Output Valid. This bit is used by the FPGA to determine when PROG_DATA[7:0] is valid. 0: PROG_DATA[7:0] is not valid 1: PROG_DATA[7:0] is valid
FP_RDY	Input	LVCMOS	Pull Down	Field Programming Bus Output Ready. This bit is a handshake signal used by the XR9240 to determine when the programmable device output bus is ready. 0: Programmable device output bus is not ready 1: Programmable device output bus is ready
FP_DN	Input	LVCMOS	Pull Down	Field Programming Bus Done. This bit is used by the XR9240 to determine when the programmable device is done. 0: Programmable device not done 1: Programmable device done
FP_LAST	Output	LVCMOS		Field Programming Last Byte. This bit will be set to correspond with the last byte of the FPGA image. 0: Not the last byte of the FPGA image 1: Last byte of FPGA image

9.5 SPI Interface

This section describes the Serial Peripheral Interface (SPI) interface.

Table 9-5. SPI Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
SPI_CS_N	Output	LVCMOS		Chip Select. Active low output signal from the XR9240 to the non-volatile device. This output should be left unconnected if not using a non-volatile device.
SPI_SO	Output	LVCMOS		Serial Output. Output data from the XR9240 to the non-volatile device. This output should be left unconnected if not using a non-volatile device.
SPI_SI	Input	LVCMOS	Pull Down	Serial Input. Non-volatile device input data driven from the output of the non-volatile device.
SPI_SCK	Output	LVCMOS		Clock Output. Clock output to the non-volatile device. This output should be left unconnected if not using a non-volatile device.

9.6 Miscellaneous Interface

Table 9-6. Miscellaneous Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
POR_N	Input	LVCMOS	Pull Up	Power on reset. Active low power on reset. When asserted, this bit will reset the entire device. 0: Reset entire chip 1: No reset
CONFIG[0]	Input	LVCMOS	Pull Up	Configuration 0. This signal must be left unconnected.
CONFIG[1]	Input	LVCMOS	Pull Down	Configuration 1. This signal must be left unconnected.

Table 9-6. Miscellaneous Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
PCIE_LINKUP	Output	LVCMOS		PCIe Link Status. This signal may be connected to a LED. 0: PCIe link down 1: PCIe link up
92XX_BUSY	Output	LVCMOS		XR9240 Busy. This signal is asserted when the command manager has commands in its queue. This signal may be connected to a LED. 0: XR9240 not busy 1: XR9240 busy
PLL_LOCK	Output	LVCMOS		PLL Lock Status. Device internal PLL clock output frequency lock status. 0: One or more of the internal PLLs is not locked 1: All internal PLLs are successfully locked
FAN_SPD	Input	LVCMOS	Pull Down	Fan Speed. This signal is the serial fan speed input that is used to calculate the actual fan speed. See "Calculating the Fan Speed".
BOARD_VER[2:0]	Input	LVCMOS		Board Version. Users may drive this signal to annotate the version of the board containing the XR9240 device.
LD_FLASH_EN	Input	LVCMOS	Pull Up	Load From Flash Enable. 0: Load PCIe configuration information from PCIe registers 1: Load PCIe configuration information from Flash
CONFIG[3]	Input	LVCMOS	Pull Up	This signal must be left unconnected.
CONFIG[2]	Input	LVCMOS	Pull Down	This signal must be left unconnected.
CLK_TST_SEL [1:0]	Input	LVCMOS	Pull Down	Clock Test Select. CLK_TST_SEL[0] should be left unconnected. CLK_TST_SEL[1] should be pulled up during compliance testing.
PLL0_REF_CLK	Input	LVCMOS		PLL0 Reference Clock. See Note 1.
PLL1_REF_CLK	Input	LVCMOS		PLL1 Reference Clock. See Note 1.
PLL2_REF_CLK	Input	LVCMOS		PLL2 Reference Clock. See Note 1.

Table 9-6. Miscellaneous Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
PLL3_REF_CLK	Input	LVCMOS		PLL3 Reference Clock. See Note 1.
TEMP_SENSE [3:0]	Output	Analog		Temperature Sensor Output. Temperature data that may be sent to an external temperature sensor device.
Notes: 1. For test, each PLL has its own reference clock. The four PLL reference clock input signals may be driven by a single 33 MHz buffered clock source.				

9.7 JTAG Interface

There are two JTAG interfaces on XR9240 device.

The standard JTAG interface is for Exar manufacturing test purposes and may be used for PCB testing.

Table 9-7. JTAG Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
JTG_CLK	Input	LVCMOS	Pull Up	JTAG Clock.
JTG_TDI	Input	LVCMOS	Pull Up	JTAG Test Data Input.
JTG_TDO	Output	LVCMOS	Pull Up	JTAG Test Data Output.
JTG_TMS	Input	LVCMOS	Pull Up	JTAG Test Mode Select.
JTG_RST_N	Input	LVCMOS	Pull Down	JTAG Reset.

The PCIe PHY JTAG interface provides a means of performing PCIe loopback testing which may be useful for debug. If this feature is desired, the PCIe PHY JTAG interface must be populated on the card containing the XR9240 device.

Table 9-8. PCIe PHY JTAG Interface Signal Definition

Signal Name	I/O Type	Signal Type	Internal Pull Up/ Down	Description
PHY_JTG_CLK	Input	LVCMOS	Pull Up	Reserved.
PHY_JTG_TDI	Input	LVCMOS	Pull Up	Reserved.
PHY_JTG_TDO	Output	LVCMOS	Pull Up	Reserved.
PHY_JTG_TMS	Input	LVCMOS	Pull Up	Reserved.
PHY_JTG_RST_N	Input	LVCMOS	Pull Down	Reserved.

9.8 Power and Ground Interface

Table 9-9. Power and Ground Interface Description

Signal Name	Instances	Description	Voltage Level
VDD	58	Main digital power supply	0.9V
VSS	244	Main ground	GND
VDDIO1P8	40	I/O power supply	1.8V
AVDD1P8_ILK	14	Analog power supply for Interlaken Serdes	1.8V
AVDD0P9_ILK	8	Analog power supply for Interlaken Serdes	0.9V
AVDD1P8_PCIE	16	Analog power supply for PCIe Serdes	1.8V
AVDD0P9_PCIE	6	Analog power supply for PCIe Serdes	0.9V
AVDD1P8_PLL0	1	Analog power supply for PLL 0	1.8V
AVDD1P8_PLL1	1	Analog power supply for PLL 1	1.8V
AVDD1P8_PLL2	1	Analog power supply for PLL 2	1.8V
AVDD1P8_PLL3	1	Analog power supply for PLL 3	1.8V
VDD0P9_PLL0	1	Digital power supply for PLL 0	0.9V
VDD0P9_PLL1	1	Digital power supply for PLL 1	0.9V
VDD0P9_PLL2	1	Digital power supply for PLL 2	0.9V
VDD0P9_PLL3	1	Digital power supply for PLL 3	0.9V

10 Error Handling

10.1 Error Detection Methods

There are two methods the host software may use to detect if errors have occurred in the XR9240:

1. Read the ERR bit in the "Result Pointer Ring".
2. Some errors will cause the XR9240 to interrupt the host, as defined in the "PCIe Interrupt Source Register".

The ERR bit in the result descriptor reflects the current command's error status. The ER_ID[3:0] and ENG_ID[3:0] fields provide more detailed information on the error condition.

10.2 Error Categories

This section defines the error categories for the XR9240 and how they are reported to the host.

Category 1: Configuration Error

Description:

A configuration error will occur if there was a ring structure alignment violation. All errors in this category will not block the XR9240's operation even if the errors occur while processing a command, however the command result will be corrupted.

Error Reporting Mechanism:

The XR9240 will interrupt the host if a configuration error is detected, if that interrupt mask is enabled.

Category 2: PCIe Error

Description:

Errors in this category would include completion timeout, completion abort, and ECRC error. All errors in this category will block the XR9240's operation because the XR9240 cannot access the completion data.

Error Reporting Mechanism:

PCIe errors are logged to the PCIe configuration header. If Advanced Error Reporting (AER) is enabled, additional information will be logged and an interrupt will be sent to host. The host must reset the XR9240 after a PCIe error.

Category 3: Data Error

Description:

Data errors include errors in the session, source data, result verification or destination buffer overflow. Data errors are written to the command result descriptor for the host to

read. All errors in this category will not block the XR9240's operation even if the errors occur while processing a command, however the command result will be corrupted.

Error Reporting Mechanism:

Commands that complete with a data error will cause the error flag and error code to be set in the result descriptor. The error code reflects the first error encountered.

If a command completes with the error flag set, the command may cause a low latency interrupt if the corresponding LLI_en bit is set. A low latency interrupt can bypass the interrupt throttling, thus providing a fast response to the error. The host software may re-submit a command that returned with a data error.

Category 4: Integrity Error

Description:

Integrity errors include CRC errors, ECC memory errors, and real time verification errors. Because a memory failure would be due to a physical or electrical issue, the failure rate for memories is expected to be low.

The XR9240 device may be configured to stop processing or continue processing when an integrity error occurs using the fence or halt modes.

If fencing mode is enabled by setting the FENCE bit in the "XR9240 Control Register", an integrity error in the PCIe core will cause the XR9240 to reset itself.

Halt mode affects all the remaining modules besides the PCIe core in the XR9240 device. If halt mode is enabled by setting the HALT bit in the "XR9240 Control Register", the XR9240 will stop all processing if an integrity error is detected. If HALT is disabled, the XR9240 will continue to operate after an integrity error occurs.

Error Reporting Mechanism:

If an integrity error is detected during command processing, the ERR bit in the result descriptor will be set and the status of the corresponding command will be flagged with an error ID. If an integrity error is detected after an error flag has already been set, the error ID will not be updated, but the ERR bit will be set.

For DMA memories with ECC protection, both 2-bit and 1-bit errors are logged into the "DMA 2-bit ECC Error Log Register" and "DMA 1-bit ECC Error Counter Register". 2-bit ECC errors are not correctable and are reported to the host via a PCIe interrupt. 1-bit ECC errors are logged as warnings. An 8-bit counter records the number of warnings, and if the counter overflows, it will trigger a PCIe interrupt.

For the PK memories, 1-bit and 2-bit ECC errors are detected but not corrected. The number of detections are logged in the "PK 1-Bit ECC Error Counter Register" and in the "PK 2-Bit ECC Error Counter Register".

Category 5: Interlaken Error

Description:

If a corrupted packet is detected on the ILK interface, the XR9240 will automatically attempt to re-send the packet.

An Interlaken error indicates that the Interlaken link is down. Errors in this category will block the XR9240's operation.

Error Reporting Mechanism:

When an Interlaken error is detected, the XR9240 will interrupt the host if that interrupt mask is enabled. The host must reset the XR9240 and FPGA after an Interlaken error.

10.3 Error ID Encoding

10.3.1 Transform Engine Errors

Table 10-1. Transform Engine Error ID Codes

Engine ID	Error ID	Error name	Description
Pre-processing Error = 0000	0000	in_int _err	ECC error detected
	0001	pp_malform_err	Input source data packet length is too short
	0010	sess_crc_err	Session CRC verification error
	0011	FIPS_err	FIPS error detected
CRC0 engine = 0001	0000	crc0_chk_err	CRC 0 verification error
	0001	crc0_malform_err	Source data length is less than 4B
Compression engine = 0010	0000	cmp_int_err	ECC error detected in Compression engine.
Pad engine = 0011	0000	reserved	Reserved.
	0001	pad_cfg_err	The padding algorithm is configured to an unknown value
Encryption engine = 0100	0000	reserved	Reserved.
	0001	enc_malform_err	Incorrect length for one of the required input fields
	0010	enc_cfg_err	Encryption configuration error
Decryption engine = 0101	0000	dec_int_err(RSV)	ECC error in Decryption engine
	0001	dec_malform_err	Incorrect length for one of the required input fields
	0010	dec_cfg_err	Decryption configuration error
DePad engine = 0110	0000	dpad_int_err(RSV)	ECC error detected in Depadding engine
	0001	dpad_cfg_err	The depadding algorithm is configured to an unknown value
	0010	dpad_chk_err	Pad verification error detected
	0011	dpad_malform_err	Padding length is larger than packet length or MOD length, or input data doesn’t end at 4 byte boundary
Decompress engine = 0111	0000	dcmp_int_err	ECC error detected in Decompression engine.
	0001	dcmp_cfg_err	The decompression algorithm is configured to an unknown value
	0010	gzip_decode_err	gzip file can't be decompressed
	0100	lzs_corrupt_err	LZS data corruption
	0101	lzs_token_err	Error is detected on token of LZS record

Table 10-1. Transform Engine Error ID Codes

Engine ID	Error ID	Error name	Description
CRC1 engine = 1000	0000	crc1_chk_err	CRC decode verification error
	0001	rv_err	Real time verification error
	0010	crc1_malform_err	Decode source data length is less than 4B
Authentication engine = 1001	0000	auth_int_err	Integrity error detected in Authentication engine
	0001	auth_cfg_err	The authentication algorithm is configured to an unknown value
	0010	auth_rv_err	Real time verification error detected in the Authentication engine
	0011	auth_malform_err	Incorrect length for one of the required input fields
	0100	auth_chk_err	MAC/hash result verification error detected
KEK engine = 1010	1000	kek_dec_err	The IV unwrapped is not the same as the default IV
Post Transform Processing = 1011	0000	ptp_int_err	PTP integrity error
ILK_PRE = 1100	0000	ilkpre_link_err	Interlaken link error
	0001	ilkpre_hdr_err	Header format error asserted when ILK_PRE interface sees TE header
	0010 - 1111	(user defined)	User defined errors for the FPGA
ILK_POST = 1101	0000	ilkpost_link_err	Interlaken link error
	0001	ilkpost_hdr_err	Header format error asserted when ILK_POST interface TE header
	0010 - 1111	(user defined)	User defined errors for the FPGA
IDC = 1110	0000	idc_int _err	ECC error detected in Inbound Data Controller (IDC)
	0001	idc_pcie_err	PCIe error
ODC = 1111	0000	odc_int _err	ECC error detected in Outbound Data Controller (ODC)
	0001	odc_overflow	The destination data size exceeded the total destination buffer space

10.3.2 Public Key Errors

Table 10-2. Public Key Error ID Codes

Engine ID	Error ID	Error name	Description
PK Manager = 0000	0000	mgr_fmt_err	The length value in the descriptor is incorrect
For PK Engines 0-7 = 1000 - 1111	0000	pk_len_err	Length error
	0001	pk_range_err	An invalid address was accessed
	0010	pk_ecc2_err	2-bit ECC error
	0011	pk_ecc1_err	1-bit ECC error
	0100	pk_wdog_err	Watchdog error
IDC = 1110	0000	idc_int _err	ECC error detected in Inbound Data Controller (IDC)
	0001	idc_pcie_err	PCIe error
ODC = 1111	0000	odc_int _err	ECC error detected in Outbound Data Controller (ODC)
	0001	odc_overflow	The destination data size exceeded the total destination buffer space

11 DC Specifications

11.1 Absolute Maximum Ratings

Table 11-1. Absolute maximum ratings

Symbol	Description	Min	Max	Units
VDD	Main digital power 0.9V supply	VSS - 0.2	1.1	V
VDDIO1P8	Digital I/O power 1.8V supply	VSS - 0.2	2.2	V
AVDD1P8_ILK	Analog 1.8V power supply for ILK Serdes	VSS - 0.2	2.2	V
AVDD0P9_ILK	Analog 0.9V power supply for ILK Serdes	VSS - 0.2	1.1	V
AVDD1P8_PCIE	Analog 1.8V power supply for PCIe Serdes	VSS - 0.2	2.2	V
AVDD0P9_PCIE	Analog 0.9V power supply for PCIe Serdes	VSS - 0.2	1.1	V
AVDD1P8_PLL0, AVDD1P8_PLL1, AVDD1P8_PLL2, AVDD1P8_PLL3	Analog 1.8V power supplies for PLLs 0-3	VSS - 0.2	2.2	V
VDD0P9_PLL0, VDD0P9_PLL1, VDD0P9_PLL2, VDD0P9_PLL3	Digital 0.9V power supplies for PLLs 0-3	VSS - 0.2	1.1	V
T_{STG}	Storage Temperature	-40	130	°C

Caution

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions may affect device reliability.

11.2 Power Supplies

The following subsections provide details concerning the digital and analog power supply connections on the XR9240 device. Generally speaking, analog supplies can be derived from the same power source as the digital supplies, but are more noise-sensitive and require additional filtering and careful layout/routing.

11.2.1 Digital Power Supplies

Symbol	Description	Min	Typ	Max	Units
VDD	Main digital power 0.9V supply	0.87	0.9	0.93	V
VDDIO1P8	Digital I/O power 1.8V supply	1.71	1.8	1.89	V

AVDD1P8_ILK	Analog 1.8V power supply for ILK Serdes	1.71	1.8	1.89	V
AVDD0P9_ILK	Analog 0.9V power supply for ILK Serdes	0.85	0.9	0.95	V
AVDD1P8_PCIE	Analog 1.8V power supply for PCIe Serdes	1.71	1.8	1.89	V
AVDD0P9_PCIE	Analog 0.9V power supply for PCIe Serdes	0.85	0.9	0.95	V
AVDD1P8_PLL0, AVDD1P8_PLL1, AVDD1P8_PLL2, AVDD1P8_PLL3	Analog 1.8V power supplies for PLLs 0-3	1.71	1.8	1.89	V
VDD0P9_PLL0, VDD0P9_PLL1, VDD0P9_PLL2, VDD0P9_PLL3	Digital 0.9V power supplies for PLLs 0-3	0.85	0.9	0.95	V

11.2.2 Analog Power Supplies

11.2.2.1 PHY Analog Power AC Requirements

Table 11-2. PHY Analog Power AC Requirements

Symbol	Description	Min	Max	Units
VDD1AC-LOFREQ for AVDD0P9_PCIE/ AVDD0P9_ILK	0.9V analog core supply voltage maximum AC power supply noise. Total Integrated Peak-Peak noise for frequencies from 1KHz to 10MHz.		0.03	Vpkpk
VDD2AC-LOFREQ for AVDD1P8_PCIE/ AVDD1P8_ILK	1.8V analog core supply voltage maximum AC power supply noise. Total Integrated Peak-Peak noise for frequencies from 1KHz to 10MHz.		0.03	Vpkpk
VDD1AC-HIFREQ for AVDD0P9_PCIE/ AVDD0P9_ILK	0.9V analog core supply voltage maximum AC power supply noise. Total Integrated Peak-Peak noise for frequencies from 10MHz and higher.		0.05	Vpkpk
VDD2AC-HIFREQ for AVDD1P8_PCIE/ AVDD1P8_ILK	1.8V analog core supply voltage maximum AC power supply noise. Total Integrated Peak-Peak noise for frequencies from 10MHz and higher.		0.05	Vpkpk

11.2.2.2 VDDIO1P8 Analog Power AC Requirements

Table 11-3. VDDIO1P8 Analog Power AC Requirements

Symbol	Description	Min	Max	Units
TSVDD_REQ	Temperature sensor noise requirement		70	mV

11.3 Power Sequencing

The XR9240 power supplies require a particular power sequence. Please refer to the *XR9240 Hardware Design Reference Guide*, USR-0032, for detailed information.

11.4 Power Consumption

This section lists the power consumption for the XR9240 device when used in the following engine combinations:

- Transform, ILK and Public Key Engines enabled, Table 11-4
- Transform and Public Key Engines enabled, ILK Engine disabled, Table 11-5
- Transform Engine enabled, ILK and Public Key Engines disabled, Table 11-6
- Transform and ILK Engines enabled, Public Key Engine disabled, Table 11-7
- Public Key Engine enabled, ILK and Transform Engines disabled, Table 11-8

Table 11-4. XR9240 Power Consumption - Transform, ILK and Public Key Engines Enabled

Power Supply	Nominal Voltage (V)	Max Voltage (V)	TYP Current (mA)	Max Current (mA)	TYP Power (W)	Max Power[2] (W)
VDD[1]	0.9	0.927	14.320	24.374	12.888	22.595
AVDD0P9_ILK	0.9	0.944	0.404	0.602	0.364	0.568
AVDD0P9_PCIE	0.9	0.944	0.487	0.707	0.438	0.667
VDD0P9_PLLx	0.9	0.944	0.001	0.001	0.001	0.001
AVDD1P8_PLLx	1.8	1.889	0.003	0.004	0.005	0.008
AVDD1P8_PCIE	1.8	1.889	0.263	0.281	0.473	0.531
VDDIO1P8	1.8	1.889	0.057	0.057	0.103	0.108
AVDD1P8_ILK	1.8	1.889	0.197	0.224	0.355	0.423
Total					14.627	24.901

Note:

1. VDD current listed does not include any current delivered to external loads driven by the XR9240.
2. Max power is calculated as maximum voltage allowed in Section 11.2.1 at 125°C junction temperature.
3. Typical power is calculated using nominal voltage at 25°C room temperature.

Table 11-5. XR9240 Power Consumption - Transform and Public Key Engines Enabled, ILK Engine Disabled

Power Supply	Nominal Voltage (V)	Max Voltage (V)	TYP Current (mA)	Max Current (mA)	TYP Power (W)	Max Power[2] (W)
VDD[1]	0.9	0.927	13.880	23.336	12.492	21.632
AVDD0P9_ILK	0.9	0.944	0.035	0.149	0.032	0.141
AVDD0P9_PCIE	0.9	0.944	0.487	0.707	0.438	0.667
VDD0P9_PLLx	0.9	0.944	0.001	0.001	0.001	0.001
AVDD1P8_PLLx	1.8	1.889	0.003	0.004	0.005	0.008
AVDD1P8_PCIE	1.8	1.889	0.263	0.281	0.473	0.531
VDDIO1P8	1.8	1.889	0.057	0.057	0.103	0.108
AVDD1P8_ILK	1.8	1.889	0.001	0.001	0.002	0.002
Total					14.627	23.089

Note:

1. VDD current listed does not include any current delivered to external loads driven by the XR9240.
2. Max power is calculated as maximum voltage allowed in Section 11.2.1 at 125°C junction temperature.
3. Typical power is calculated using nominal voltage at 25°C room temperature.

Table 11-6. XR9240 Power Consumption - Transform Engine Enabled, ILK and Public Key Engines Disabled

Power Supply	Nominal Voltage (V)	Max Voltage (V)	TYP Current (mA)	Max Current (mA)	TYP Power (W)	Max Power[2] (W)
VDD[1]	0.9	0.927	8.888	16.728	7.999	15.507
AVDD0P9_ILK	0.9	0.944	0.035	0.149	0.032	0.141
AVDD0P9_PCIE	0.9	0.944	0.487	0.707	0.438	0.667
VDD0P9_PLLx	0.9	0.944	0.001	0.001	0.001	0.001
AVDD1P8_PLLx	1.8	1.889	0.003	0.004	0.005	0.008
AVDD1P8_PCIE	1.8	1.889	0.259	0.276	0.466	0.521
VDDIO1P8	1.8	1.889	0.057	0.057	0.103	0.108
AVDD1P8_ILK	1.8	1.889	0.001	0.001	0.002	0.002
Total					9.046	16.954

Note:

1. VDD current listed does not include any current delivered to external loads driven by the XR9240.
2. Max power is calculated as maximum voltage allowed in Section 11.2.1 at 125°C junction temperature.
3. Typical power is calculated using nominal voltage at 25°C room temperature.

Table 11-7. XR9240 Power Consumption - Transform and ILK Engines Enabled, Public Key Engine Disabled

Power Supply	Nominal Voltage (V)	Max Voltage (V)	TYP Current (mA)	Max Current (mA)	TYP Power (W)	Max Power[2] (W)
VDD[1]	0.9	0.927	9.357	17.645	8.421	16.357
AVDD0P9_ILK	0.9	0.944	0.413	0.602	0.372	0.568
AVDD0P9_PCIE	0.9	0.944	0.487	0.707	0.438	0.667
VDD0P9_PLLx	0.9	0.944	0.001	0.001	0.001	0.001
AVDD1P8_PLLx	1.8	1.889	0.003	0.004	0.005	0.008
AVDD1P8_PCIE	1.8	1.889	0.259	0.276	0.466	0.521
VDDIO1P8	1.8	1.889	0.057	0.057	0.103	0.108
AVDD1P8_ILK	1.8	1.889	0.197	0.224	0.355	0.423
Total					10.161	18.653

Note:

1. VDD current listed does not include any current delivered to external loads driven by the XR9240.
2. Max power is calculated as maximum voltage allowed in Section 11.2.1 at 125°C junction temperature.
3. Typical power is calculated using nominal voltage at 25°C room temperature.

Table 11-8. XR9240 Power Consumption - Public Key Engine Enabled, Transform and ILK Engines Disabled

Power Supply	Nominal Voltage (V)	Max Voltage (V)	TYP Current (mA)	Max Current (mA)	TYP Power (W)	Max Power[2] (W)
VDD[1]	0.9	0.927	6.464	13.364	5.818	12.388
AVDD0P9_ILK	0.9	0.944	0.035	0.149	0.032	0.141
AVDD0P9_PCIE	0.9	0.944	0.483	0.707	0.435	0.667
VDD0P9_PLLx	0.9	0.944	0.001	0.001	0.001	0.001
AVDD1P8_PLLx	1.8	1.889	0.003	0.004	0.005	0.008
AVDD1P8_PCIE	1.8	1.889	0.259	0.276	0.466	0.521
VDDIO1P8	1.8	1.889	0.052	0.061	0.094	0.115
AVDD1P8_ILK	1.8	1.889	0.001	0.001	0.002	0.002
Total					6.852	13.843

Note:

1. VDD current listed does not include any current delivered to external loads driven by the XR9240.
2. Max power is calculated as maximum voltage allowed in Section 11.2.1 at 125°C junction temperature.
3. Typical power is calculated using nominal voltage at 25°C room temperature.

11.5 I/O Characteristics

Table 11-9. IO Characteristics

Symbol	Description	Min	TYP	Max	Units
V_{IH}	DC Input High Voltage	1.26		1.8	V
V_{IL}	DC Input Low Voltage	VSS		0.54	V
V_{OH}	DC Output High Voltage	1.44		1.8	V
V_{OL}	DC Output Low Voltage	VSS		0.36	V
I_{RUP}	Input pull-up resistor current	-45			uA
I_{RDP}	Input pull-down resistor current	-45			uA
V_H	Input hysteresis	0.18			mV

12 AC Specifications

12.1 PLL Clock Input

Table 12-1. PLL Reference Clock Requirements

Symbol	Description	Min	Typ	Max	Units
F_{RefClk}	Input clock frequency	25		33.3	MHz
$D.C._{RefClk}$	Duty Cycle	45	50	55	%
$J_{CLK-REF}$	Input Jitter (peak-to-peak)			80	ps
T_{rdy}	Lock time		200		µs
Note: Spread spectrum clocks are not supported.					

12.2 SPI Interface Timing

This section describes the Serial Peripheral Interface timing.

Refer to the *XR9240 Hardware Design Guide*, USR-0032, for a table of supported devices and their capabilities.

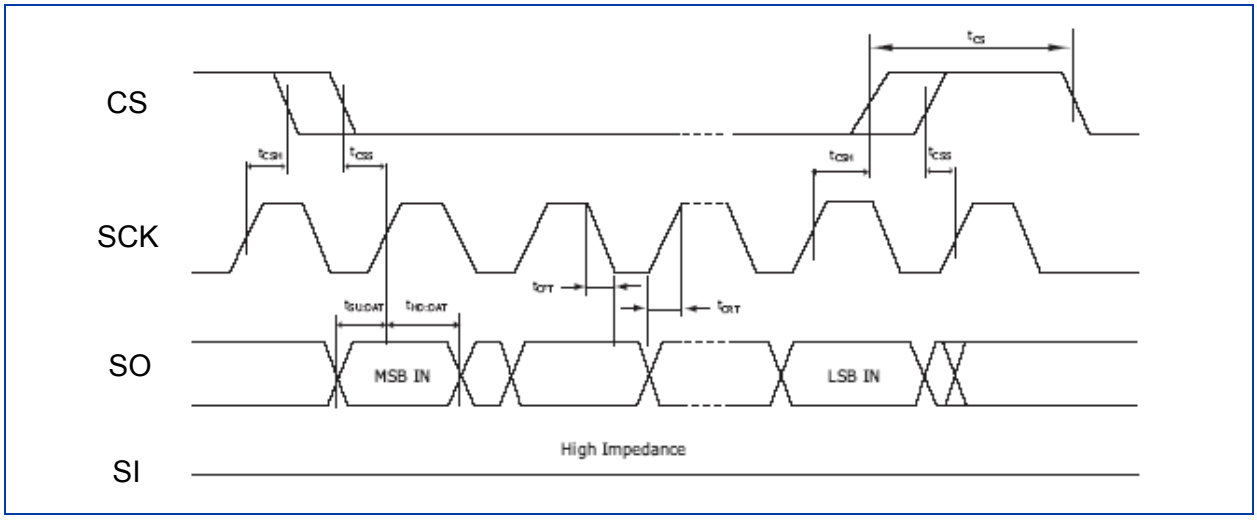


Figure 12-1. SPI Write Timing

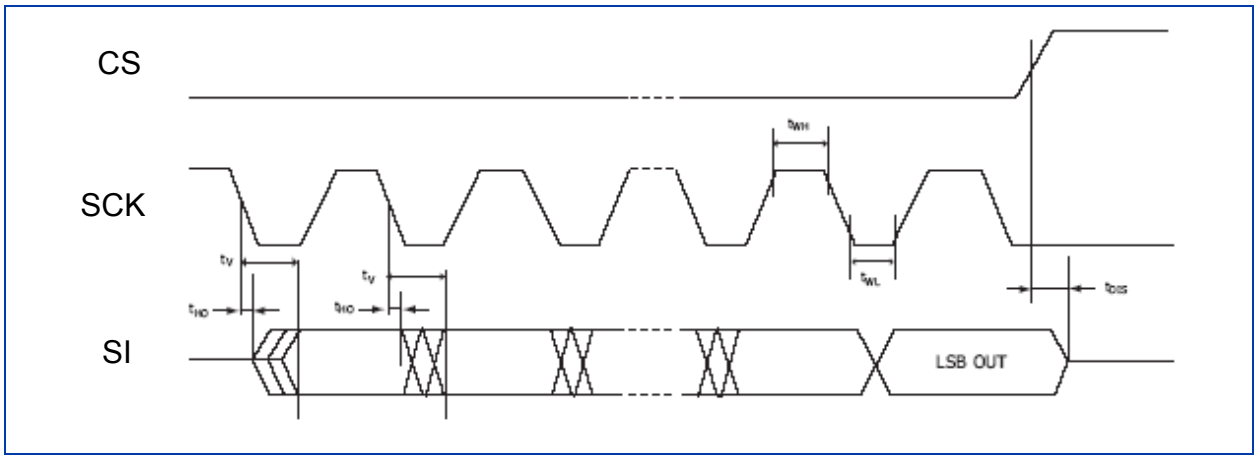


Figure 12-2. SPI Read Timing

Table 12-2. SPI Interface AC Characteristics

Symbol	Description	Min	Typ	Max	Units
t_{WH}	Programmable device SCK high time	9			ns
t_{WL}	Programmable device SCK low time	9			ns
t_{CSS}	Programmable device CS setup time	5			ns
t_{CSH}	Programmable device CS hold time	5			ns
t_V	Programmable device SI valid time	0		10	ns
t_{HO}	Programmable device SI hold time	0			ns
$t_{HD:DAT}$	Programmable device SO hold time	5			ns
$t_{SU:DAT}$	Programmable device SO setup time	5			ns
Note: The parameters in this table were measured with a 30pf load.					

12.3 JTAG Interface Timing

The XR9240 is designed to support the IEEE 1149.1 JTAG standard.

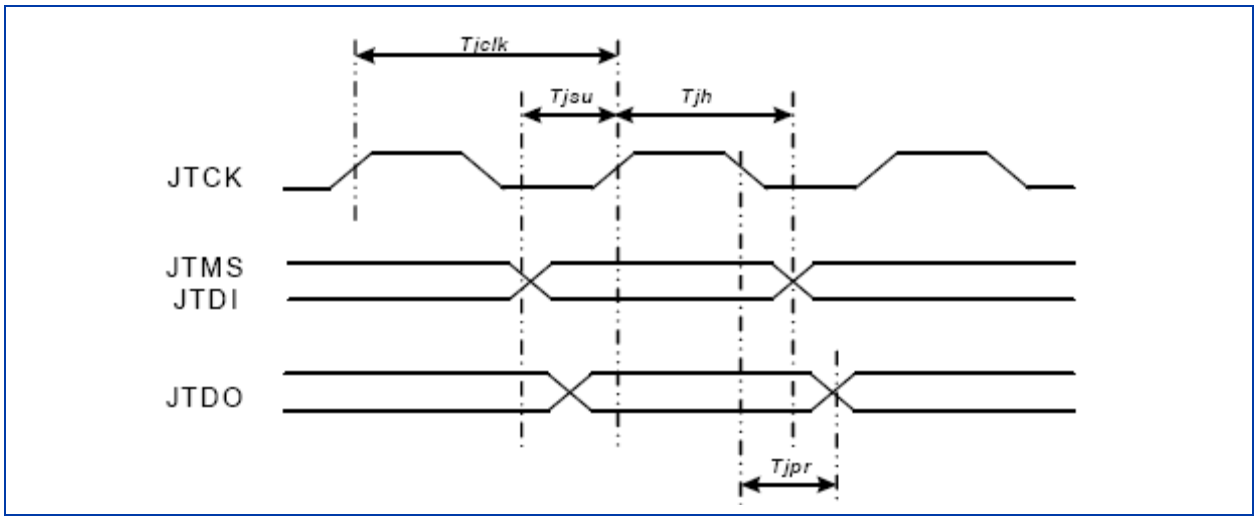


Figure 12-3. JTAG Timing

Table 12-3. JTAG Interface AC Characteristics

Symbol	Description	Min	Max	Units
T_{jclk}	JTAG clock frequency		20	ns
T_{th}	JTAG_TMS and JTAG_TDI hold time	10		ns
T_{jsu}	JTAG_TMS and JTAG_TDI setup time	10		ns
T_{jpr}	JTAG_TDO propagation delay		15	ns

12.4 PCIe Interface Timing

The XR9240 PCIe interface is fully compliant to the PCI Express Electromechanical Specification Revision 3.0 standard.

12.5 ILK Interface Timing

The XR9240 ILK interface is fully compliant to the Interlaken Protocol Definition revision 1.2.

13 Thermal Specifications

Each application will require thermal analysis based on its system environment.

Table 13-1. Thermal operating conditions

Symbol	Description	Min	Max	Units
Tj	Junction Temperature (applies to commercial parts)	-40	125	°C
Ta	Ambient Temperature	-40	55	°C

Notes:

1. For normal device operation, adhere to the limits in this table. Sustained operations of a device at conditions exceeding these values, even if they are within the absolute maximum rating limits, may result in permanent device damage or impaired device reliability. Device functionality to stated DC and AC limits is not guaranteed if conditions exceed recommended operating conditions.

2. Recommended operation conditions require accuracy of the power supplies as described in Section 11.2.

3. The ambient operating temperature for all devices must be managed with any combination of device speed rating, heatsink, or airflow to guarantee Tj is not exceeded.

Table 13-2. Thermal Specifications

Parameter	Max
Junction Temperature (Tj)	125 °C
Internal thermal resistance, (θ_{jc})	0.04 °C/W
Thermal resistance, (θ_{ja}) at 0 m/s airflow	8.9 °C/W
Thermal resistance, (θ_{ja}) at 1 m/s airflow	7.8 °C/W
Thermal resistance, (θ_{ja}) at 2 m/s airflow	7.3 °C/W
Temperature correlation (Ψ_{jt})	0.02 °C/W

13.1 Thermal Sensor Controller

The XR9240 contains a thermal sensor controller that can be used to read or monitor the XR9240 die temperature. Please refer to "Temperature Sensor Controller" and "XR9240 Configuration Registers" for detailed information.

14 Package Specifications

This chapter provides general and mechanical package information, as well as ball assignment drawings.

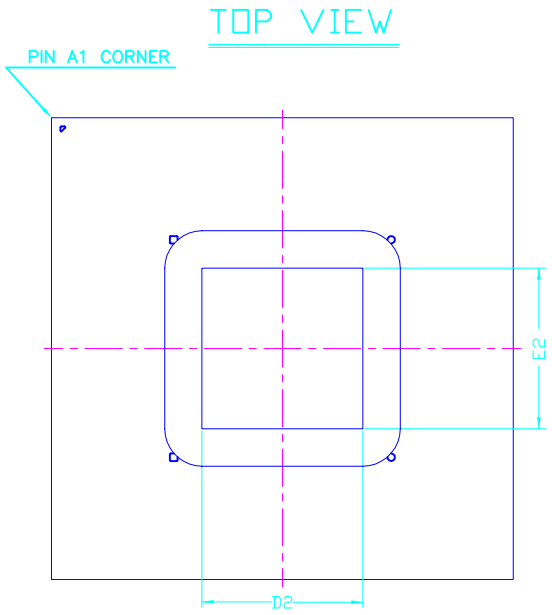
14.1 General Information

Table 14-1. General Package Information

Package Information	Description
Package Type	FCBGA
Ball Count	896
Die Size	9821 x 9910 µm
Package Size	31 mm x 31 mm
Pitch	1.0 mm

14.2 Mechanical Information

The package dimensions from the top view are given in Figure 14-2. The package dimensions from the bottom and side view are shown in Figure 14-2.



	SYMBOL	COMMON DIMENSIONS		
		MIN.	NOM.	MAX.
TOTAL THICKNESS	A	2.217	2.467	2.717
STAND OFF	A1	0.400	–	0.600
SUBSTRATE THICKNESS	A2		1.080	REF
THICKNESS FROM SUBSTRATE SURFACE TO DIE BACKSIDE	A3		0.887	REF
BODY SIZE	D		31.000	BSC
	E		31.000	BSC
BALL DIAMETER			0.600	
BALL WIDTH	b	0.500	–	0.700
BALL PITCH	e		1.000	BSC
BALL COUNT	n		896	
EDGE BALL CENTER TO CENTER	D1		29.000	BSC
	E1		29.000	BSC
EXPOSE DIE SIZE	D2		10.800	BSC
	E2		10.800	BSC
PACKAGE EDGE TOLERANCE	aaa		0.150	
SUBSTRATE PARALLELISM	bbb		0.100	
TOP PARALLELISM	ccc		0.200	
COPLANARITY	ddd		0.200	
BALL OFFSET(PACKAGE)	eee		0.250	
BALL OFFSET (BALL)	fff		0.100	

Figure 14-1. Top View Package Dimensions

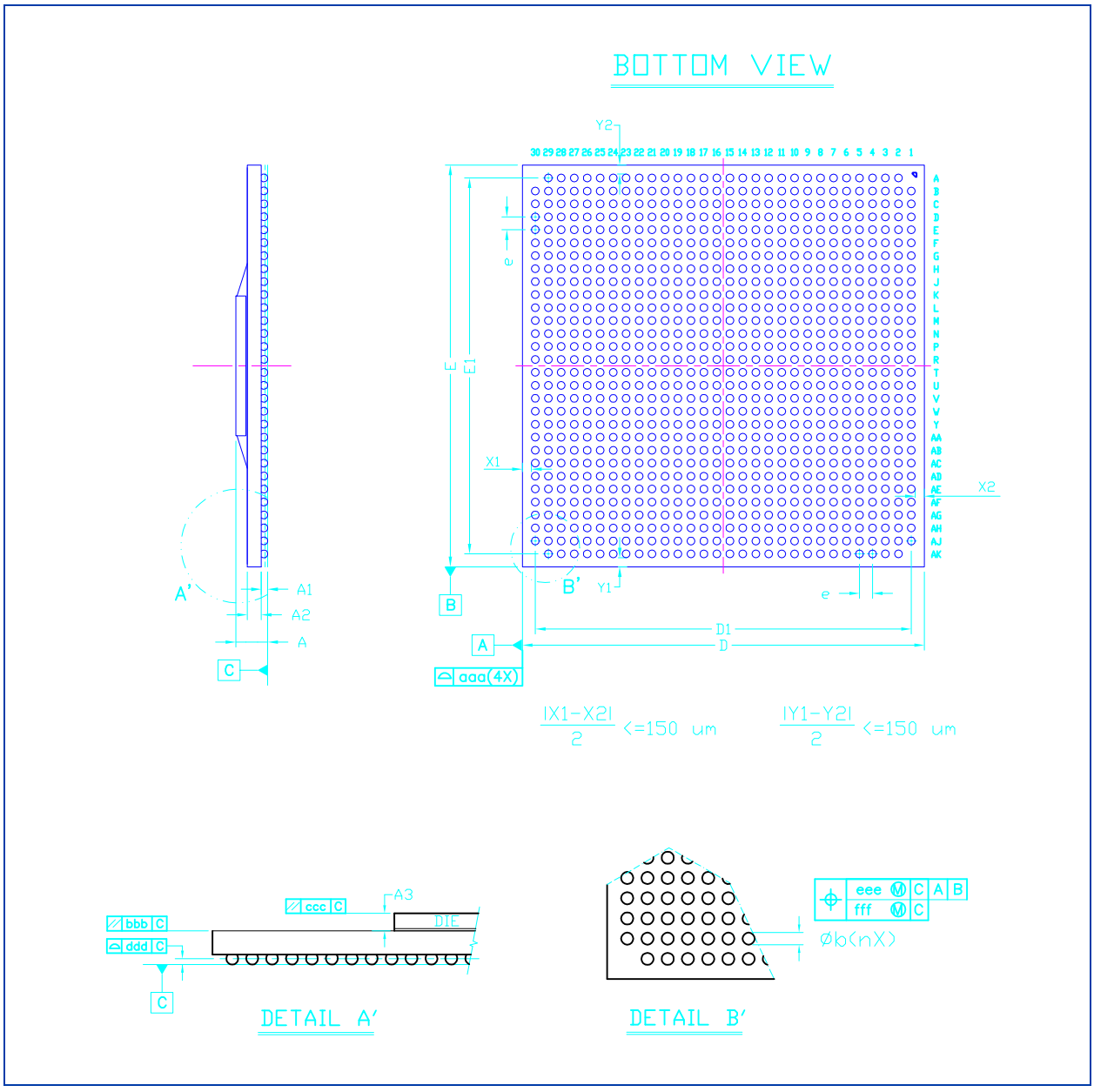
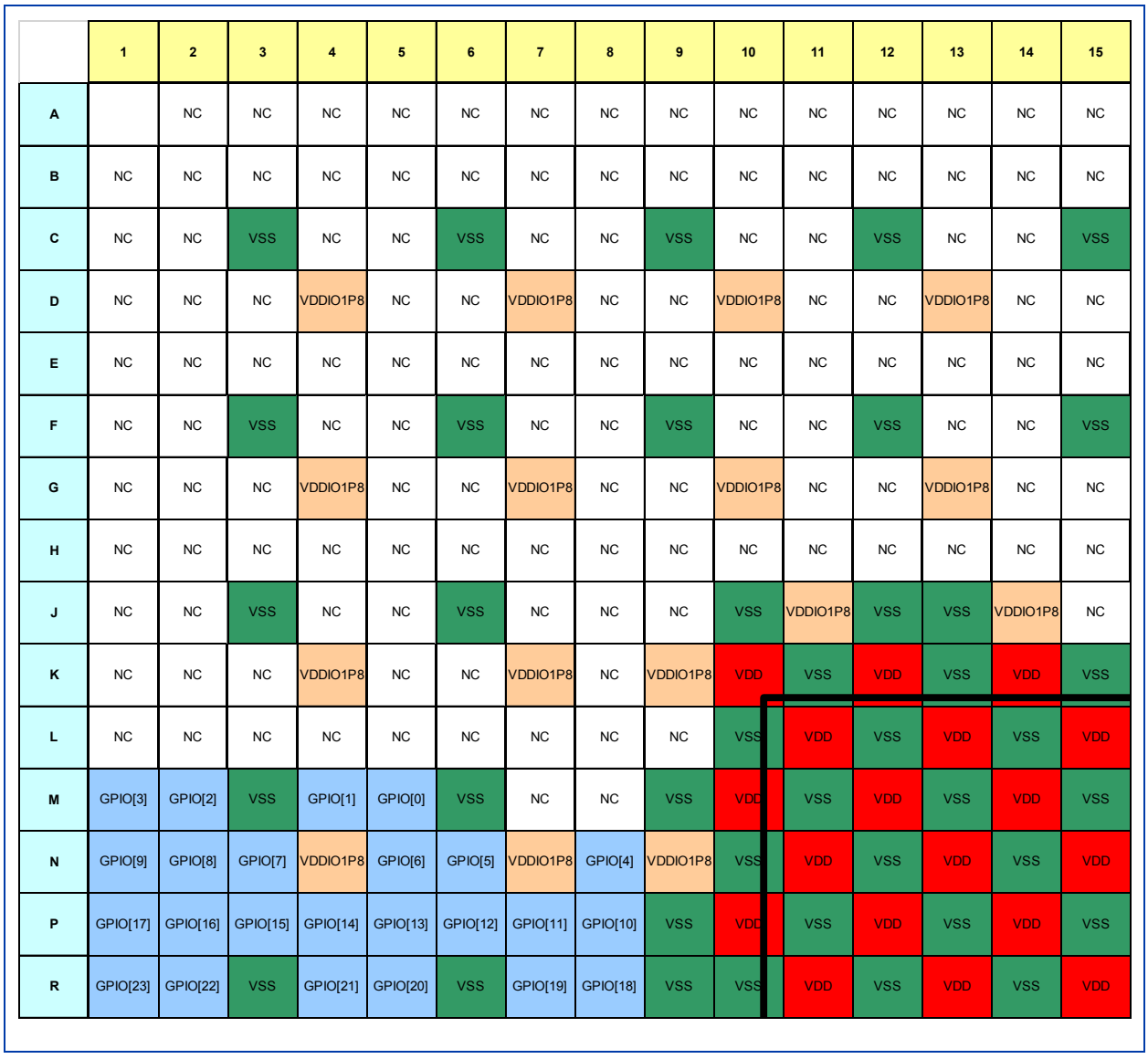


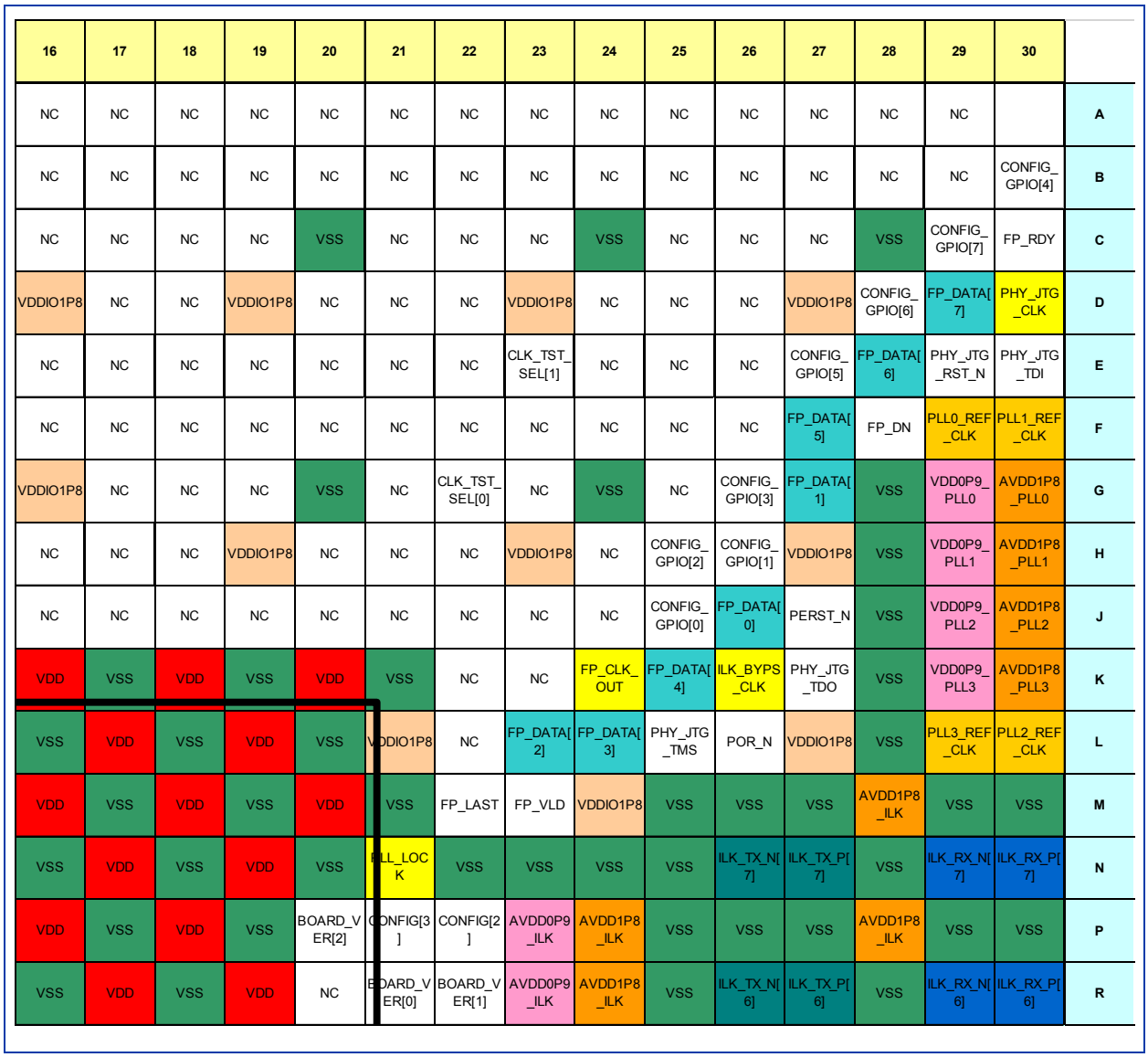
Figure 14-2. Bottom View and Side View Package Dimensions

14.3 Ball Map Drawings



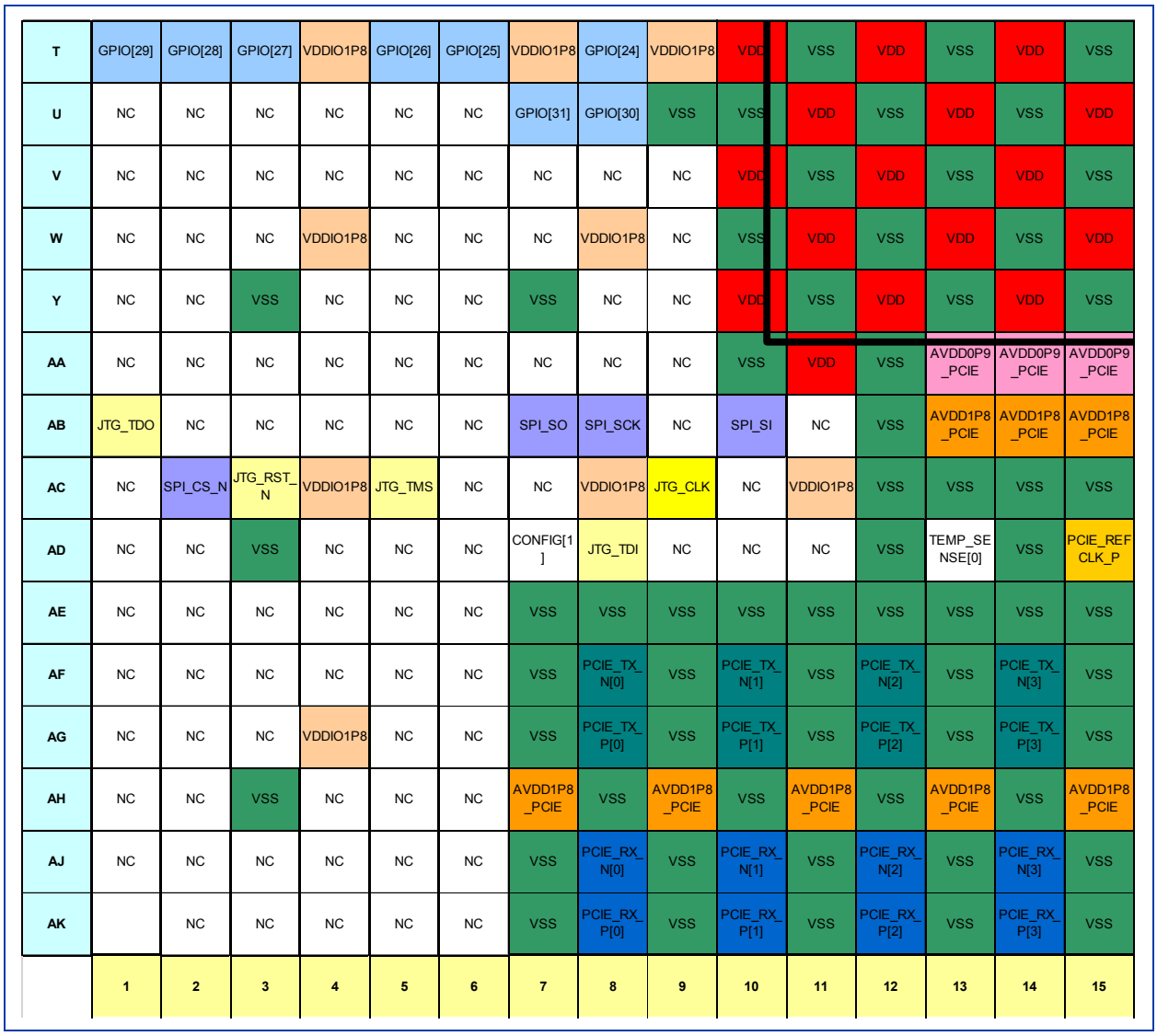
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A		NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
B	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
C	NC	NC	VSS	NC	NC	VSS	NC	NC	VSS	NC	NC	VSS	NC	NC	VSS
D	NC	NC	NC	VDDIO1P8	NC	NC	VDDIO1P8	NC	NC	VDDIO1P8	NC	NC	VDDIO1P8	NC	NC
E	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
F	NC	NC	VSS	NC	NC	VSS	NC	NC	VSS	NC	NC	VSS	NC	NC	VSS
G	NC	NC	NC	VDDIO1P8	NC	NC	VDDIO1P8	NC	NC	VDDIO1P8	NC	NC	VDDIO1P8	NC	NC
H	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC
J	NC	NC	VSS	NC	NC	VSS	NC	NC	NC	VSS	VDDIO1P8	VSS	VSS	VDDIO1P8	NC
K	NC	NC	NC	VDDIO1P8	NC	NC	VDDIO1P8	NC	VDDIO1P8	VDD	VSS	VDD	VSS	VDD	VSS
L	NC	NC	NC	NC	NC	NC	NC	NC	NC	VSS	VDD	VSS	VDD	VSS	VDD
M	GPIO[3]	GPIO[2]	VSS	GPIO[1]	GPIO[0]	VSS	NC	NC	VSS	VDD	VSS	VDD	VSS	VDD	VSS
N	GPIO[9]	GPIO[8]	GPIO[7]	VDDIO1P8	GPIO[6]	GPIO[5]	VDDIO1P8	GPIO[4]	VDDIO1P8	VSS	VDD	VSS	VDD	VSS	VDD
P	GPIO[17]	GPIO[16]	GPIO[15]	GPIO[14]	GPIO[13]	GPIO[12]	GPIO[11]	GPIO[10]	VSS	VDD	VSS	VDD	VSS	VDD	VSS
R	GPIO[23]	GPIO[22]	VSS	GPIO[21]	GPIO[20]	VSS	GPIO[19]	GPIO[18]	VSS	VSS	VDD	VSS	VDD	VSS	VDD

Figure 14-3. Ball Map Drawing - Top View - Upper Left Quadrant



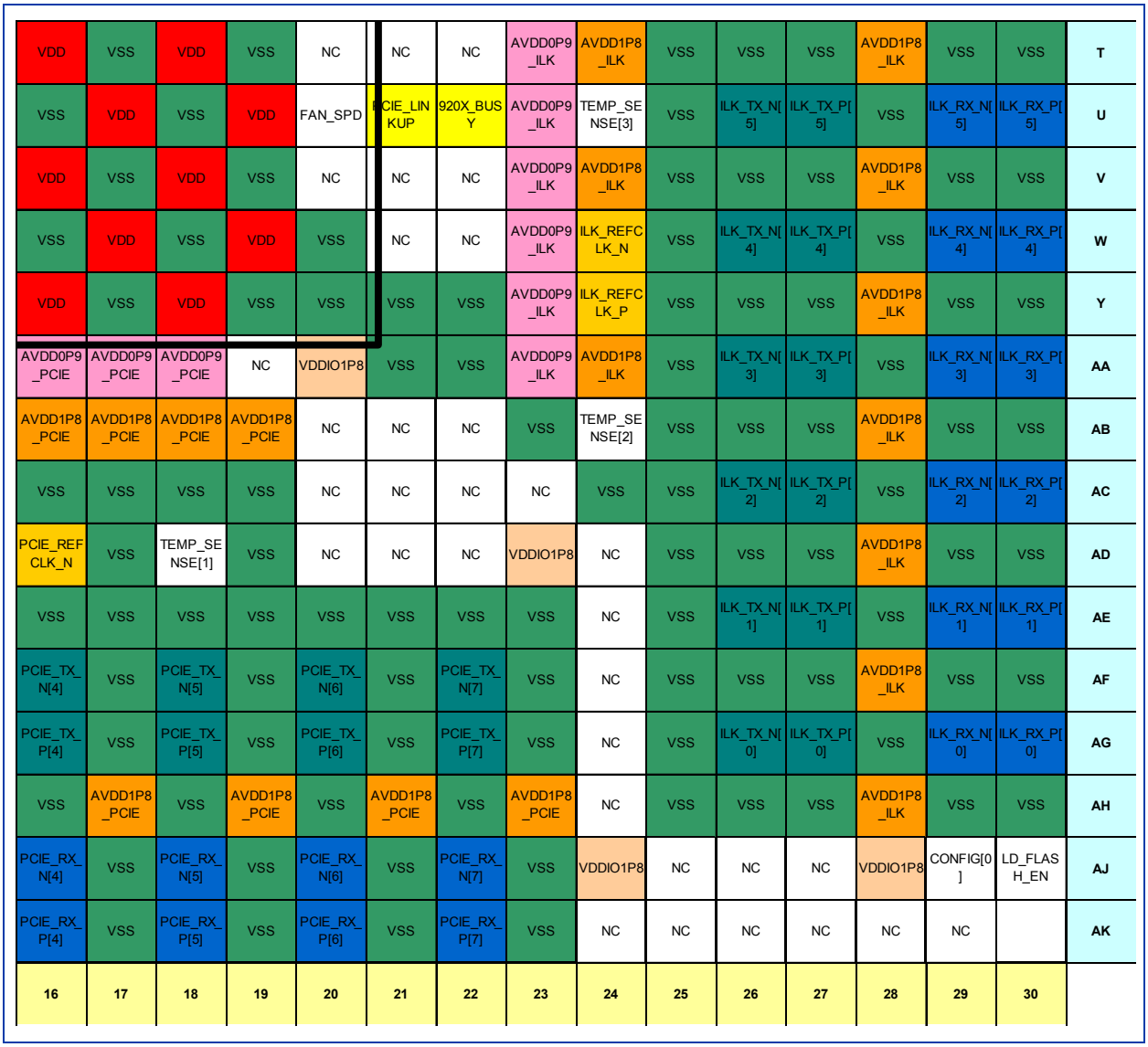
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC		A
NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	CONFIG_GPIO[4]	B
NC	NC	NC	NC	VSS	NC	NC	NC	VSS	NC	NC	NC	VSS	CONFIG_GPIO[7]	FP_RDY	C
VDDIO1P8	NC	NC	VDDIO1P8	NC	NC	NC	VDDIO1P8	NC	NC	NC	VDDIO1P8	CONFIG_GPIO[6]	FP_DATA[7]	PHY_JTG_CLK	D
NC	NC	NC	NC	NC	NC	NC	CLK_TST_SEL[1]	NC	NC	NC	CONFIG_GPIO[5]	FP_DATA[6]	PHY_JTG_RST_N	PHY_JTG_TDI	E
NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	NC	FP_DATA[5]	FP_DN	PLL0_REF_CLK	PLL1_REF_CLK	F
VDDIO1P8	NC	NC	NC	VSS	NC	CLK_TST_SEL[0]	NC	VSS	NC	CONFIG_GPIO[3]	FP_DATA[1]	VSS	VDD0P9_PLL0	AVDD1P8_PLL0	G
NC	NC	NC	VDDIO1P8	NC	NC	NC	VDDIO1P8	NC	CONFIG_GPIO[2]	CONFIG_GPIO[1]	VDDIO1P8	VSS	VDD0P9_PLL1	AVDD1P8_PLL1	H
NC	NC	NC	NC	NC	NC	NC	NC	NC	CONFIG_GPIO[0]	FP_DATA[0]	PERST_N	VSS	VDD0P9_PLL2	AVDD1P8_PLL2	J
VDD	VSS	VDD	VSS	VDD	VSS	NC	NC	FP_CLK_OUT	FP_DATA[4]	ILK_BYPS_CLK	PHY_JTG_TDO	VSS	VDD0P9_PLL3	AVDD1P8_PLL3	K
VSS	VDD	VSS	VDD	VSS	VDDIO1P8	NC	FP_DATA[2]	FP_DATA[3]	PHY_JTG_TMS	POR_N	VDDIO1P8	VSS	PLL3_REF_CLK	PLL2_REF_CLK	L
VDD	VSS	VDD	VSS	VDD	VSS	FP_LAST	FP_VLD	VDDIO1P8	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	M
VSS	VDD	VSS	VDD	VSS	PLL_LOCK	VSS	VSS	VSS	VSS	ILK_TX_N[7]	ILK_TX_P[7]	VSS	ILK_RX_N[7]	ILK_RX_P[7]	N
VDD	VSS	VDD	VSS	BOARD_VER[2]	CONFIG[3]	CONFIG[2]	AVDD0P9_ILK	AVDD1P8_ILK	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	P
VSS	VDD	VSS	VDD	NC	BOARD_VER[0]	BOARD_VER[1]	AVDD0P9_ILK	AVDD1P8_ILK	VSS	ILK_TX_N[6]	ILK_TX_P[6]	VSS	ILK_RX_N[6]	ILK_RX_P[6]	R

Figure 14-4. Ball Map Drawing - Top View - Upper Right Quadrant



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
T	GPIO[29]	GPIO[28]	GPIO[27]	VDDIO1P8	GPIO[26]	GPIO[25]	VDDIO1P8	GPIO[24]	VDDIO1P8	VDD	VSS	VDD	VSS	VDD	VSS
U	NC	NC	NC	NC	NC	NC	GPIO[31]	GPIO[30]	VSS	VSS	VDD	VSS	VDD	VSS	VDD
V	NC	NC	NC	NC	NC	NC	NC	NC	NC	VDD	VSS	VDD	VSS	VDD	VSS
W	NC	NC	NC	VDDIO1P8	NC	NC	NC	VDDIO1P8	NC	VSS	VDD	VSS	VDD	VSS	VDD
Y	NC	NC	VSS	NC	NC	NC	VSS	NC	NC	VDD	VSS	VDD	VSS	VDD	VSS
AA	NC	NC	NC	NC	NC	NC	NC	NC	NC	VSS	VDD	VSS	AVDD0P9_PCIE	AVDD0P9_PCIE	AVDD0P9_PCIE
AB	JTG_TDO	NC	NC	NC	NC	NC	SPI_SO	SPI_SCK	NC	SPI_SI	NC	VSS	AVDD1P8_PCIE	AVDD1P8_PCIE	AVDD1P8_PCIE
AC	NC	SPI_CS_N	JTG_RST_N	VDDIO1P8	JTG_TMS	NC	NC	VDDIO1P8	JTG_CLK	NC	VDDIO1P8	VSS	VSS	VSS	VSS
AD	NC	NC	VSS	NC	NC	NC	CONFIG[1]	JTG_TDI	NC	NC	NC	VSS	TEMP_SENSE[0]	VSS	PCIE_REFCLK_P
AE	NC	NC	NC	NC	NC	NC	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS
AF	NC	NC	NC	NC	NC	NC	VSS	PCIE_TX_N[0]	VSS	PCIE_TX_N[1]	VSS	PCIE_TX_N[2]	VSS	PCIE_TX_N[3]	VSS
AG	NC	NC	NC	VDDIO1P8	NC	NC	VSS	PCIE_TX_P[0]	VSS	PCIE_TX_P[1]	VSS	PCIE_TX_P[2]	VSS	PCIE_TX_P[3]	VSS
AH	NC	NC	VSS	NC	NC	NC	AVDD1P8_PCIE	VSS	AVDD1P8_PCIE	VSS	AVDD1P8_PCIE	VSS	AVDD1P8_PCIE	VSS	AVDD1P8_PCIE
AJ	NC	NC	NC	NC	NC	NC	VSS	PCIE_RX_N[0]	VSS	PCIE_RX_N[1]	VSS	PCIE_RX_N[2]	VSS	PCIE_RX_N[3]	VSS
AK		NC	NC	NC	NC	NC	VSS	PCIE_RX_P[0]	VSS	PCIE_RX_P[1]	VSS	PCIE_RX_P[2]	VSS	PCIE_RX_P[3]	VSS

Figure 14-5. Ball Map Drawing - Top View - Lower Left Quadrant



VDD	VSS	VDD	VSS	NC	NC	NC	AVDD0P9_ILK	AVDD1P8_ILK	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	T
VSS	VDD	VSS	VDD	FAN_SPD	PCIE_LINKUP	920X_BUSY	AVDD0P9_ILK	TEMP_SENSE[3]	VSS	ILK_TX_N[5]	ILK_TX_P[5]	VSS	ILK_RX_N[5]	ILK_RX_P[5]	U
VDD	VSS	VDD	VSS	NC	NC	NC	AVDD0P9_ILK	AVDD1P8_ILK	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	V
VSS	VDD	VSS	VDD	VSS	NC	NC	AVDD0P9_ILK	ILK_REFCLK_N	VSS	ILK_TX_N[4]	ILK_TX_P[4]	VSS	ILK_RX_N[4]	ILK_RX_P[4]	W
VDD	VSS	VDD	VSS	VSS	VSS	VSS	AVDD0P9_ILK	ILK_REFCLK_P	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	Y
AVDD0P9_PCIE	AVDD0P9_PCIE	AVDD0P9_PCIE	NC	VDDIO1P8	VSS	VSS	AVDD0P9_ILK	AVDD1P8_ILK	VSS	ILK_TX_N[3]	ILK_TX_P[3]	VSS	ILK_RX_N[3]	ILK_RX_P[3]	AA
AVDD1P8_PCIE	AVDD1P8_PCIE	AVDD1P8_PCIE	AVDD1P8_PCIE	NC	NC	NC	VSS	TEMP_SENSE[2]	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	AB
VSS	VSS	VSS	VSS	NC	NC	NC	NC	VSS	VSS	ILK_TX_N[2]	ILK_TX_P[2]	VSS	ILK_RX_N[2]	ILK_RX_P[2]	AC
PCIE_REFCLK_N	VSS	TEMP_SENSE[1]	VSS	NC	NC	NC	VDDIO1P8	NC	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	AD
VSS	VSS	VSS	VSS	VSS	VSS	VSS	VSS	NC	VSS	ILK_TX_N[1]	ILK_TX_P[1]	VSS	ILK_RX_N[1]	ILK_RX_P[1]	AE
PCIE_TX_N[4]	VSS	PCIE_TX_N[5]	VSS	PCIE_TX_N[6]	VSS	PCIE_TX_N[7]	VSS	NC	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	AF
PCIE_TX_P[4]	VSS	PCIE_TX_P[5]	VSS	PCIE_TX_P[6]	VSS	PCIE_TX_P[7]	VSS	NC	VSS	ILK_TX_N[0]	ILK_TX_P[0]	VSS	ILK_RX_N[0]	ILK_RX_P[0]	AG
VSS	AVDD1P8_PCIE	VSS	AVDD1P8_PCIE	VSS	AVDD1P8_PCIE	VSS	AVDD1P8_PCIE	NC	VSS	VSS	VSS	AVDD1P8_ILK	VSS	VSS	AH
PCIE_RX_N[4]	VSS	PCIE_RX_N[5]	VSS	PCIE_RX_N[6]	VSS	PCIE_RX_N[7]	VSS	VDDIO1P8	NC	NC	NC	VDDIO1P8	CONFIG[0]	LD_FLASH_EN	AJ
PCIE_RX_P[4]	VSS	PCIE_RX_P[5]	VSS	PCIE_RX_P[6]	VSS	PCIE_RX_P[7]	VSS	NC	NC	NC	NC	NC	NC		AK
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	

Figure 14-6. Ball Map Drawing - Top View - Lower Right Quadrant

14.4 Signal Names Lists

This section lists the alphabetic and numeric signal names. Please refer to the *XR9240 Hardware Design Reference Guide*, USR-0032, for detailed information about terminating the unused (no connect) signals and for special considerations for the power and ground signals.

Table 14-2. Alphabetical Ball List

Signal	Ball
92XX_BUSY	U22
AVDD0P9_ILK	P23
AVDD0P9_ILK	R23
AVDD0P9_ILK	T23
AVDD0P9_ILK	U23
AVDD0P9_ILK	V23
AVDD0P9_ILK	W23
AVDD0P9_ILK	Y23
AVDD0P9_ILK	AA23
AVDD0P9_PCIE	AA13
AVDD0P9_PCIE	AA14
AVDD0P9_PCIE	AA15
AVDD0P9_PCIE	AA16
AVDD0P9_PCIE	AA17
AVDD0P9_PCIE	AA18
AVDD1P8_ILK	M28
AVDD1P8_ILK	P24
AVDD1P8_ILK	P28
AVDD1P8_ILK	R24
AVDD1P8_ILK	T24
AVDD1P8_ILK	T28
AVDD1P8_ILK	V24
AVDD1P8_ILK	V28
AVDD1P8_ILK	Y28
AVDD1P8_ILK	AA24
AVDD1P8_ILK	AB28
AVDD1P8_ILK	AD28
AVDD1P8_ILK	AF28
AVDD1P8_ILK	AH28
AVDD1P8_PCIE	AB13
AVDD1P8_PCIE	AB14
AVDD1P8_PCIE	AB15
AVDD1P8_PCIE	AB16
AVDD1P8_PCIE	AB17
AVDD1P8_PCIE	AB18
AVDD1P8_PCIE	AB19

Signal	Ball
AVDD1P8_PCIE	AH7
AVDD1P8_PCIE	AH9
AVDD1P8_PCIE	AH11
AVDD1P8_PCIE	AH13
AVDD1P8_PCIE	AH15
AVDD1P8_PCIE	AH17
AVDD1P8_PCIE	AH19
AVDD1P8_PCIE	AH21
AVDD1P8_PCIE	AH23
AVDD1P8_PLL0	G30
AVDD1P8_PLL1	H30
AVDD1P8_PLL2	J30
AVDD1P8_PLL3	K30
BOARD_VER[0]	R21
BOARD_VER[1]	R22
BOARD_VER[2]	P20
CLK_TST_SEL[0]	G22
CLK_TST_SEL[1]	E23
CONFIG_GPIO[0]	J25
CONFIG_GPIO[1]	H26
CONFIG_GPIO[2]	H25
CONFIG_GPIO[3]	G26
CONFIG_GPIO[4]	B30
CONFIG_GPIO[5]	E27
CONFIG_GPIO[6]	D28
CONFIG_GPIO[7]	C29
CONFIG[0]	AJ29
CONFIG[1]	AD7
CONFIG[2]	P22
CONFIG[3]	P21
FAN_SPEED	U20
FP_CLK_OUT	K24
FP_DATA[0]	J26
FP_DATA[1]	G27
FP_DATA[2]	L23
FP_DATA[3]	L24

Signal	Ball
FP_DATA[4]	K25
FP_DATA[5]	F27
FP_DATA[6]	E28
FP_DATA[7]	D29
FP_DN	F28
FP_LAST	M22
FP_RDY	C30
FP_VLD	M23
GPIO[0]	M5
GPIO[1]	M4
GPIO[10]	P8
GPIO[11]	P7
GPIO[12]	P6
GPIO[13]	P5
GPIO[14]	P4
GPIO[15]	P3
GPIO[16]	P2
GPIO[17]	P1
GPIO[18]	R8
GPIO[19]	R7
GPIO[2]	M2
GPIO[20]	R5
GPIO[21]	R4
GPIO[22]	R2
GPIO[23]	R1
GPIO[24]	T8
GPIO[25]	T6
GPIO[26]	T5
GPIO[27]	T3
GPIO[28]	T2
GPIO[29]	T1
GPIO[3]	M1
GPIO[30]	U8
GPIO[31]	U7
GPIO[4]	N8
GPIO[5]	N6

Signal	Ball
GPIO[6]	N5
GPIO[7]	N3
GPIO[8]	N2
GPIO[9]	N1
ILK_BYPS_CLK	K26
ILK_REFCLK_N	W24
ILK_REFCLK_P	Y24
ILK_RX_N[0]	AG29
ILK_RX_N[1]	AE29
ILK_RX_N[2]	AC29
ILK_RX_N[3]	AA29
ILK_RX_N[4]	W29
ILK_RX_N[5]	U29
ILK_RX_N[6]	R29
ILK_RX_N[7]	N29
ILK_RX_P[0]	AG30
ILK_RX_P[1]	AE30
ILK_RX_P[2]	AC30
ILK_RX_P[3]	AA30
ILK_RX_P[4]	W30
ILK_RX_P[5]	U30
ILK_RX_P[6]	R30
ILK_RX_P[7]	N30
ILK_TX_N[0]	AG26
ILK_TX_N[1]	AE26
ILK_TX_N[2]	AC26
ILK_TX_N[3]	AA26
ILK_TX_N[4]	W26
ILK_TX_N[5]	U26
ILK_TX_N[6]	R26
ILK_TX_N[7]	N26
ILK_TX_P[0]	AG27
ILK_TX_P[1]	AE27
ILK_TX_P[2]	AC27
ILK_TX_P[3]	AA27
ILK_TX_P[4]	W27
ILK_TX_P[5]	U27

Signal	Ball
ILK_TX_P[6]	R27
ILK_TX_P[7]	N27
JTG_CLK	AC9
JTG_RST_N	AC3
JTG_TDI	AD8
JTG_TDO	AB1
JTG_TMS	AC5
LD_FLASH_EN	AJ30
PCIE_LINKUP	U21
PCIE_REFCLK_N	AD16
PCIE_REFCLK_P	AD15
PCIE_RX_N[0]	AJ8
PCIE_RX_N[1]	AJ10
PCIE_RX_N[2]	AJ12
PCIE_RX_N[3]	AJ14
PCIE_RX_N[4]	AJ16
PCIE_RX_N[5]	AJ18
PCIE_RX_N[6]	AJ20
PCIE_RX_N[7]	AJ22
PCIE_RX_P[0]	AK8
PCIE_RX_P[1]	AK10
PCIE_RX_P[2]	AK12
PCIE_RX_P[3]	AK14
PCIE_RX_P[4]	AK16
PCIE_RX_P[5]	AK18
PCIE_RX_P[6]	AK20
PCIE_RX_P[7]	AK22
PCIE_TX_N[0]	AF8
PCIE_TX_N[1]	AF10
PCIE_TX_N[2]	AF12
PCIE_TX_N[3]	AF14
PCIE_TX_N[4]	AF16
PCIE_TX_N[5]	AF18
PCIE_TX_N[6]	AF20
PCIE_TX_N[7]	AF22
PCIE_TX_P[0]	AG8
PCIE_TX_P[1]	AG10

Signal	Ball
PCIE_TX_P[2]	AG12
PCIE_TX_P[3]	AG14
PCIE_TX_P[4]	AG16
PCIE_TX_P[5]	AG18
PCIE_TX_P[6]	AG20
PCIE_TX_P[7]	AG22
PERST_N	J27
PHY_JTG_CLK	D30
PHY_JTG_RST_N	E29
PHY_JTG_TDI	E30
PHY_JTG_TDO	K27
PHY_JTG_TMS	L25
PLL_LOCK	N21
PLL0_REF_CLK	F29
PLL1_REF_CLK	F30
PLL2_REF_CLK	L30
PLL3_REF_CLK	L29
POR_N	L26
SPI_CS_N	AC2
SPI_SCK	AB8
SPI_SI	AB10
SPI_SO	AB7
TEMP_SENSE[0]	AD13
TEMP_SENSE[1]	AD18
TEMP_SENSE[2]	AB24
TEMP_SENSE[3]	U24
VDD	K10
VDD	K12
VDD	K14
VDD	K16
VDD	K18
VDD	K20
VDD	L11
VDD	L13
VDD	L15
VDD	L17
VDD	L19

Signal	Ball
VDD	M10
VDD	M12
VDD	M14
VDD	M16
VDD	M18
VDD	M20
VDD	N11
VDD	N13
VDD	N15
VDD	N17
VDD	N19
VDD	P10
VDD	P12
VDD	P14
VDD	P16
VDD	P18
VDD	R11
VDD	R13
VDD	R15
VDD	R17
VDD	R19
VDD	T10
VDD	T12
VDD	T14
VDD	T16
VDD	T18
VDD	U11
VDD	U13
VDD	U15
VDD	U17
VDD	U19
VDD	V10
VDD	V12
VDD	V14
VDD	V16
VDD	V18
VDD	W11

Signal	Ball
VDD	W13
VDD	W15
VDD	W17
VDD	W19
VDD	Y10
VDD	Y12
VDD	Y14
VDD	Y16
VDD	Y18
VDD	AA11
VDD0P9_PLL0	G29
VDD0P9_PLL1	H29
VDD0P9_PLL2	J29
VDD0P9_PLL3	K29
VDDIO1P8	D4
VDDIO1P8	D7
VDDIO1P8	D10
VDDIO1P8	D13
VDDIO1P8	D16
VDDIO1P8	D19
VDDIO1P8	D23
VDDIO1P8	D27
VDDIO1P8	G4
VDDIO1P8	G7
VDDIO1P8	G10
VDDIO1P8	G13
VDDIO1P8	G16
VDDIO1P8	H19
VDDIO1P8	H23
VDDIO1P8	H27
VDDIO1P8	J11
VDDIO1P8	J14
VDDIO1P8	K4
VDDIO1P8	K7
VDDIO1P8	K9
VDDIO1P8	L21
VDDIO1P8	L27

Signal	Ball
VDDIO1P8	M24
VDDIO1P8	N4
VDDIO1P8	N7
VDDIO1P8	N9
VDDIO1P8	T4
VDDIO1P8	T7
VDDIO1P8	T9
VDDIO1P8	W4
VDDIO1P8	W8
VDDIO1P8	AA20
VDDIO1P8	AC4
VDDIO1P8	AC8
VDDIO1P8	AC11
VDDIO1P8	AD23
VDDIO1P8	AG4
VDDIO1P8	AJ24
VDDIO1P8	AJ28
VSS	C3
VSS	C6
VSS	C9
VSS	C12
VSS	C15
VSS	C20
VSS	C24
VSS	C28
VSS	F3
VSS	F6
VSS	F9
VSS	F12
VSS	F15
VSS	G20
VSS	G24
VSS	G28
VSS	H28
VSS	J3
VSS	J6
VSS	J10

Signal	Ball	Signal	Ball	Signal	Ball
VSS	J12	VSS	N22	VSS	U10
VSS	J13	VSS	N23	VSS	U12
VSS	J28	VSS	N24	VSS	U14
VSS	K11	VSS	N25	VSS	U16
VSS	K13	VSS	N28	VSS	U18
VSS	K15	VSS	P9	VSS	U25
VSS	K17	VSS	P11	VSS	U28
VSS	K19	VSS	P13	VSS	V11
VSS	K21	VSS	P15	VSS	V13
VSS	K28	VSS	P17	VSS	V15
VSS	L10	VSS	P19	VSS	V17
VSS	L12	VSS	P25	VSS	V19
VSS	L14	VSS	P26	VSS	V25
VSS	L16	VSS	P27	VSS	V26
VSS	L18	VSS	P29	VSS	V27
VSS	L20	VSS	P30	VSS	V29
VSS	L28	VSS	R3	VSS	V30
VSS	M3	VSS	R6	VSS	W10
VSS	M6	VSS	R9	VSS	W12
VSS	M9	VSS	R10	VSS	W14
VSS	M11	VSS	R12	VSS	W16
VSS	M13	VSS	R14	VSS	W18
VSS	M15	VSS	R16	VSS	W20
VSS	M17	VSS	R18	VSS	W25
VSS	M19	VSS	R25	VSS	W28
VSS	M21	VSS	R28	VSS	Y3
VSS	M25	VSS	T11	VSS	Y7
VSS	M26	VSS	T13	VSS	Y11
VSS	M27	VSS	T15	VSS	Y13
VSS	M29	VSS	T17	VSS	Y15
VSS	M30	VSS	T19	VSS	Y17
VSS	N10	VSS	T25	VSS	Y19
VSS	N12	VSS	T26	VSS	Y20
VSS	N14	VSS	T27	VSS	Y21
VSS	N16	VSS	T29	VSS	Y22
VSS	N18	VSS	T30	VSS	Y25
VSS	N20	VSS	U9	VSS	Y26

Signal	Ball
VSS	Y27
VSS	Y29
VSS	Y30
VSS	AA10
VSS	AA12
VSS	AA21
VSS	AA22
VSS	AA25
VSS	AA28
VSS	AB12
VSS	AB23
VSS	AB25
VSS	AB26
VSS	AB27
VSS	AB29
VSS	AB30
VSS	AC12
VSS	AC13
VSS	AC14
VSS	AC15
VSS	AC16
VSS	AC17
VSS	AC18
VSS	AC19
VSS	AC24
VSS	AC25
VSS	AC28
VSS	AD3
VSS	AD12
VSS	AD14
VSS	AD17
VSS	AD19
VSS	AD25
VSS	AD26
VSS	AD27
VSS	AD29
VSS	AD30

Signal	Ball
VSS	AE7
VSS	AE8
VSS	AE9
VSS	AE10
VSS	AE11
VSS	AE12
VSS	AE13
VSS	AE14
VSS	AE15
VSS	AE16
VSS	AE17
VSS	AE18
VSS	AE19
VSS	AE20
VSS	AE21
VSS	AE22
VSS	AE23
VSS	AE25
VSS	AE28
VSS	AF7
VSS	AF9
VSS	AF11
VSS	AF13
VSS	AF15
VSS	AF17
VSS	AF19
VSS	AF21
VSS	AF23
VSS	AF25
VSS	AF26
VSS	AF27
VSS	AF29
VSS	AF30
VSS	AG7
VSS	AG9
VSS	AG11
VSS	AG13

Signal	Ball
VSS	AG15
VSS	AG17
VSS	AG19
VSS	AG21
VSS	AG23
VSS	AG25
VSS	AG28
VSS	AH3
VSS	AH8
VSS	AH10
VSS	AH12
VSS	AH14
VSS	AH16
VSS	AH18
VSS	AH20
VSS	AH22
VSS	AH25
VSS	AH26
VSS	AH27
VSS	AH29
VSS	AH30
VSS	AJ7
VSS	AJ9
VSS	AJ11
VSS	AJ13
VSS	AJ15
VSS	AJ17
VSS	AJ19
VSS	AJ21
VSS	AJ23
VSS	AK7
VSS	AK9
VSS	AK11
VSS	AK13
VSS	AK15
VSS	AK17

Signal	Ball
VSS	AK19
VSS	AK21
VSS	AK23

Table 14-3. Numeric Ball List

Signal	Ball
VSS	AA10
VDD	AA11
VSS	AA12
AVDD0P9_PCIE	AA13
AVDD0P9_PCIE	AA14
AVDD0P9_PCIE	AA15
AVDD0P9_PCIE	AA16
AVDD0P9_PCIE	AA17
AVDD0P9_PCIE	AA18
VDDIO1P8	AA20
VSS	AA21
VSS	AA22
AVDD0P9_ILK	AA23
AVDD1P8_ILK	AA24
VSS	AA25
ILK_TX_N[3]	AA26
ILK_TX_P[3]	AA27
VSS	AA28
ILK_RX_N[3]	AA29
ILK_RX_P[3]	AA30
JTG_TDO	AB1
SPI_SI	AB10
VSS	AB12
AVDD1P8_PCIE	AB13
AVDD1P8_PCIE	AB14
AVDD1P8_PCIE	AB15
AVDD1P8_PCIE	AB16
AVDD1P8_PCIE	AB17
AVDD1P8_PCIE	AB18
AVDD1P8_PCIE	AB19
VSS	AB23
TEMP_SENSE[2]	AB24
VSS	AB25
VSS	AB26
VSS	AB27
AVDD1P8_ILK	AB28

Signal	Ball
VSS	AB29
VSS	AB30
SPI_SO	AB7
SPI_SCK	AB8
VDDIO1P8	AC11
VSS	AC12
VSS	AC13
VSS	AC14
VSS	AC15
VSS	AC16
VSS	AC17
VSS	AC18
VSS	AC19
SPI_CS_N	AC2
VSS	AC24
VSS	AC25
ILK_TX_N[2]	AC26
ILK_TX_P[2]	AC27
VSS	AC28
ILK_RX_N[2]	AC29
JTG_RST_N	AC3
ILK_RX_P[2]	AC30
VDDIO1P8	AC4
JTG_TMS	AC5
VDDIO1P8	AC8
JTG_CLK	AC9
VSS	AD12
TEMP_SENSE[0]	AD13
VSS	AD14
PCIE_REFCLK_P	AD15
PCIE_REFCLK_N	AD16
VSS	AD17
TEMP_SENSE[1]	AD18
VSS	AD19
VDDIO1P8	AD23
VSS	AD25

Signal	Ball
VSS	AD26
VSS	AD27
AVDD1P8_ILK	AD28
VSS	AD29
VSS	AD3
VSS	AD30
CONFIG[1]	AD7
JTG_TDI	AD8
VSS	AE10
VSS	AE11
VSS	AE12
VSS	AE13
VSS	AE14
VSS	AE15
VSS	AE16
VSS	AE17
VSS	AE18
VSS	AE19
VSS	AE20
VSS	AE21
VSS	AE22
VSS	AE23
VSS	AE25
ILK_TX_N[1]	AE26
ILK_TX_P[1]	AE27
VSS	AE28
ILK_RX_N[1]	AE29
ILK_RX_P[1]	AE30
VSS	AE7
VSS	AE8
VSS	AE9
PCIE_TX_N[1]	AF10
VSS	AF11
PCIE_TX_N[2]	AF12
VSS	AF13
PCIE_TX_N[3]	AF14

Signal	Ball
VSS	AF15
PCIE_TX_N[4]	AF16
VSS	AF17
PCIE_TX_N[5]	AF18
VSS	AF19
PCIE_TX_N[6]	AF20
VSS	AF21
PCIE_TX_N[7]	AF22
VSS	AF23
VSS	AF25
VSS	AF26
VSS	AF27
AVDD1P8_ILK	AF28
VSS	AF29
VSS	AF30
VSS	AF7
PCIE_TX_N[0]	AF8
VSS	AF9
PCIE_TX_P[1]	AG10
VSS	AG11
PCIE_TX_P[2]	AG12
VSS	AG13
PCIE_TX_P[3]	AG14
VSS	AG15
PCIE_TX_P[4]	AG16
VSS	AG17
PCIE_TX_P[5]	AG18
VSS	AG19
PCIE_TX_P[6]	AG20
VSS	AG21
PCIE_TX_P[7]	AG22
VSS	AG23
VSS	AG25
ILK_TX_N[0]	AG26
ILK_TX_P[0]	AG27
VSS	AG28
ILK_RX_N[0]	AG29

Signal	Ball
ILK_RX_P[0]	AG30
VDDIO1P8	AG4
VSS	AG7
PCIE_TX_P[0]	AG8
VSS	AG9
VSS	AH10
AVDD1P8_PCIE	AH11
VSS	AH12
AVDD1P8_PCIE	AH13
VSS	AH14
AVDD1P8_PCIE	AH15
VSS	AH16
AVDD1P8_PCIE	AH17
VSS	AH18
AVDD1P8_PCIE	AH19
VSS	AH20
AVDD1P8_PCIE	AH21
VSS	AH22
AVDD1P8_PCIE	AH23
VSS	AH25
VSS	AH26
VSS	AH27
AVDD1P8_ILK	AH28
VSS	AH29
VSS	AH3
VSS	AH30
AVDD1P8_PCIE	AH7
VSS	AH8
AVDD1P8_PCIE	AH9
PCIE_RX_N[1]	AJ10
VSS	AJ11
PCIE_RX_N[2]	AJ12
VSS	AJ13
PCIE_RX_N[3]	AJ14
VSS	AJ15
PCIE_RX_N[4]	AJ16
VSS	AJ17

Signal	Ball
PCIE_RX_N[5]	AJ18
VSS	AJ19
PCIE_RX_N[6]	AJ20
VSS	AJ21
PCIE_RX_N[7]	AJ22
VSS	AJ23
VDDIO1P8	AJ24
VDDIO1P8	AJ28
CONFIG[0]	AJ29
LD_FLASH_EN	AJ30
VSS	AJ7
PCIE_RX_N[0]	AJ8
VSS	AJ9
PCIE_RX_P[1]	AK10
VSS	AK11
PCIE_RX_P[2]	AK12
VSS	AK13
PCIE_RX_P[3]	AK14
VSS	AK15
PCIE_RX_P[4]	AK16
VSS	AK17
PCIE_RX_P[5]	AK18
VSS	AK19
PCIE_RX_P[6]	AK20
VSS	AK21
PCIE_RX_P[7]	AK22
VSS	AK23
VSS	AK7
PCIE_RX_P[0]	AK8
VSS	AK9
CONFIG_GPIO[4]	B30
VSS	C12
VSS	C15
VSS	C20
VSS	C24
VSS	C28
CONFIG_GPIO[7]	C29

Signal	Ball
VSS	C3
FP_RDY	C30
VSS	C6
VSS	C9
VDDIO1P8	D10
VDDIO1P8	D13
VDDIO1P8	D16
VDDIO1P8	D19
VDDIO1P8	D23
VDDIO1P8	D27
CONFIG_GPIO[6]	D28
FP_DATA[7]	D29
PHY_JTG_CLK	D30
VDDIO1P8	D4
VDDIO1P8	D7
CLK_TST_SEL[1]	E23
CONFIG_GPIO[5]	E27
FP_DATA[6]	E28
PHY_JTG_RST_N	E29
PHY_JTG_TDI	E30
VSS	F12
VSS	F15
FP_DATA[5]	F27
FP_DN	F28
PLL0_REF_CLK	F29
VSS	F3
PLL1_REF_CLK	F30
VSS	F6
VSS	F9
VDDIO1P8	G10
VDDIO1P8	G13
VDDIO1P8	G16
VSS	G20
CLK_TST_SEL[0]	G22
VSS	G24
CONFIG_GPIO[3]	G26
FP_DATA[1]	G27

Signal	Ball
VSS	G28
VDD0P9_PLL0	G29
AVDD1P8_PLL0	G30
VDDIO1P8	G4
VDDIO1P8	G7
VDDIO1P8	H19
VDDIO1P8	H23
CONFIG_GPIO[2]	H25
CONFIG_GPIO[1]	H26
VDDIO1P8	H27
VSS	H28
VDD0P9_PLL1	H29
AVDD1P8_PLL1	H30
VSS	J10
VDDIO1P8	J11
VSS	J12
VSS	J13
VDDIO1P8	J14
CONFIG_GPIO[0]	J25
FP_DATA[0]	J26
PERST_N	J27
VSS	J28
VDD0P9_PLL2	J29
VSS	J3
AVDD1P8_PLL2	J30
VSS	J6
VDD	K10
VSS	K11
VDD	K12
VSS	K13
VDD	K14
VSS	K15
VDD	K16
VSS	K17
VDD	K18
VSS	K19
VDD	K20

Signal	Ball
VSS	K21
FP_CLK_OUT	K24
FP_DATA[4]	K25
ILK_BYPS_CLK	K26
PHY_JTG_TDO	K27
VSS	K28
VDD0P9_PLL3	K29
AVDD1P8_PLL3	K30
VDDIO1P8	K4
VDDIO1P8	K7
VDDIO1P8	K9
VSS	L10
VDD	L11
VSS	L12
VDD	L13
VSS	L14
VDD	L15
VSS	L16
VDD	L17
VSS	L18
VDD	L19
VSS	L20
VDDIO1P8	L21
FP_DATA[2]	L23
FP_DATA[3]	L24
PHY_JTG_TMS	L25
POR_N	L26
VDDIO1P8	L27
VSS	L28
PLL3_REF_CLK	L29
PLL2_REF_CLK	L30
GPIO[3]	M1
VDD	M10
VSS	M11
VDD	M12
VSS	M13
VDD	M14

Signal	Ball	Signal	Ball	Signal	Ball
VSS	M15	VSS	N23	GPIO[15]	P3
VDD	M16	VSS	N24	VSS	P30
VSS	M17	VSS	N25	GPIO[14]	P4
VDD	M18	ILK_TX_N[7]	N26	GPIO[13]	P5
VSS	M19	ILK_TX_P[7]	N27	GPIO[12]	P6
GPIO[2]	M2	VSS	N28	GPIO[11]	P7
VDD	M20	ILK_RX_N[7]	N29	GPIO[10]	P8
VSS	M21	GPIO[7]	N3	VSS	P9
FP_LAST	M22	ILK_RX_P[7]	N30	GPIO[23]	R1
FP_VLD	M23	VDDIO1P8	N4	VSS	R10
VDDIO1P8	M24	GPIO[6]	N5	VDD	R11
VSS	M25	GPIO[5]	N6	VSS	R12
VSS	M26	VDDIO1P8	N7	VDD	R13
VSS	M27	GPIO[4]	N8	VSS	R14
AVDD1P8_ILK	M28	VDDIO1P8	N9	VDD	R15
VSS	M29	GPIO[17]	P1	VSS	R16
VSS	M3	VDD	P10	VDD	R17
VSS	M30	VSS	P11	VSS	R18
GPIO[1]	M4	VDD	P12	VDD	R19
GPIO[0]	M5	VSS	P13	GPIO[22]	R2
VSS	M6	VDD	P14	BOARD_VER[0]	R21
VSS	M9	VSS	P15	BOARD_VER[1]	R22
GPIO[9]	N1	VDD	P16	AVDD0P9_ILK	R23
VSS	N10	VSS	P17	AVDD1P8_ILK	R24
VDD	N11	VDD	P18	VSS	R25
VSS	N12	VSS	P19	ILK_TX_N[6]	R26
VDD	N13	GPIO[16]	P2	ILK_TX_P[6]	R27
VSS	N14	BOARD_VER[2]	P20	VSS	R28
VDD	N15	CONFIG[3]	P21	ILK_RX_N[6]	R29
VSS	N16	CONFIG[2]	P22	VSS	R3
VDD	N17	AVDD0P9_ILK	P23	ILK_RX_P[6]	R30
VSS	N18	AVDD1P8_ILK	P24	GPIO[21]	R4
VDD	N19	VSS	P25	GPIO[20]	R5
GPIO[8]	N2	VSS	P26	VSS	R6
VSS	N20	VSS	P27	GPIO[19]	R7
PLL_LOCK	N21	AVDD1P8_ILK	P28	GPIO[18]	R8
VSS	N22	VSS	P29	VSS	R9

Signal	Ball
GPIO[29]	T1
VDD	T10
VSS	T11
VDD	T12
VSS	T13
VDD	T14
VSS	T15
VDD	T16
VSS	T17
VDD	T18
VSS	T19
GPIO[28]	T2
AVDD0P9_ILK	T23
AVDD1P8_ILK	T24
VSS	T25
VSS	T26
VSS	T27
AVDD1P8_ILK	T28
VSS	T29
GPIO[27]	T3
VSS	T30
VDDIO1P8	T4
GPIO[26]	T5
GPIO[25]	T6
VDDIO1P8	T7
GPIO[24]	T8
VDDIO1P8	T9
VSS	U10
VDD	U11
VSS	U12
VDD	U13
VSS	U14
VDD	U15
VSS	U16
VDD	U17
VSS	U18
VDD	U19

Signal	Ball
FAN_SPEED	U20
PCIE_LINKUP	U21
92XX_BUSY	U22
AVDD0P9_ILK	U23
TEMP_SENSE[3]	U24
VSS	U25
ILK_TX_N[5]	U26
ILK_TX_P[5]	U27
VSS	U28
ILK_RX_N[5]	U29
ILK_RX_P[5]	U30
GPIO[31]	U7
GPIO[30]	U8
VSS	U9
VDD	V10
VSS	V11
VDD	V12
VSS	V13
VDD	V14
VSS	V15
VDD	V16
VSS	V17
VDD	V18
VSS	V19
AVDD0P9_ILK	V23
AVDD1P8_ILK	V24
VSS	V25
VSS	V26
VSS	V27
AVDD1P8_ILK	V28
VSS	V29
VSS	V30
VSS	W10
VDD	W11
VSS	W12
VDD	W13
VSS	W14

Signal	Ball
VDD	W15
VSS	W16
VDD	W17
VSS	W18
VDD	W19
VSS	W20
AVDD0P9_ILK	W23
ILK_REFCLK_N	W24
VSS	W25
ILK_TX_N[4]	W26
ILK_TX_P[4]	W27
VSS	W28
ILK_RX_N[4]	W29
ILK_RX_P[4]	W30
VDDIO1P8	W4
VDDIO1P8	W8
VDD	Y10
VSS	Y11
VDD	Y12
VSS	Y13
VDD	Y14
VSS	Y15
VDD	Y16
VSS	Y17
VDD	Y18
VSS	Y19
VSS	Y20
VSS	Y21
VSS	Y22
AVDD0P9_ILK	Y23
ILK_REFCLK_P	Y24
VSS	Y25
VSS	Y26
VSS	Y27
AVDD1P8_ILK	Y28
VSS	Y29

Signal	Ball
VSS	Y3
VSS	Y30
VSS	Y7

Appendix A: Summary of 820x/XR9240 Differences

Table A-1. 820x/XR9240 Differences

Feature	820x	XR9240
PCIe ring structure	Command descriptor size limited only by host memory. Command descriptor used to deliver the command specific information. No software virtual address pointer is sent with the command. Result status field is in the command pointer ring. The result pointer ring entry size 8B. Supports dual command rings.	Command structure size is fixed at 16B. User desc is used to deliver the command specific information. Software virtual address pointer is sent with command. The pointer will be written back to result pointer ring. Result status field is in the result pointer ring. The result ring entry size 16B. Supports 128 rings with QOS.
Small packet commands	Special format.	No special treatment.
PCIe Interface	PCIe 2.0 Gen1	PCIe 3.0 Gen3 with support for SR-IOV, FLR, AER, ARI, IDO, TPH, and MSIX.
Interrupt throttle	Packet based solution	Time based solution. Both interrupt throttle and Fast interrupt are supported.
PK ring	8 commands per PK ring.	Share the DMA ring structure with Transform Channel commands. Unified the command submission mechanism. Remove the limitation of 8 commands per PK ring.
Hash calculation	Supports slice hash and file hash in the same pass.	Does not support file hash and slice hash in the same pass.
Integrity	ECC (3 bit detectable) + Parity	ECC (2 bit detectable) CRC is no longer correlated with real time verification. Parity is not calculated. Correctable errors are counted.
Real time verification	Enable in register. Each algorithm engine computed RTV independently	Enable in session descriptor. Comp/Pad/Enc engines share RTV, Auth engine has independent RTV When RTV is enabled for AES-CBC/ECB, both ENCKEY and DECKEY must be provided by host. If enabled, the CRC value may be output to the sideband buffer.

Table A-1. 820x/XR9240 Differences

Feature	820x	XR9240
Address alignment	The destination buffer address and length must be on an 8 byte boundary. AAD, IV address must be align on 8-byte boundary No error reported if value written to ring structure register violates the alignment restriction.	Removed destination data buffer address alignment restriction. Removed AAD, IV address alignment restriction. Reports error if value written to ring structure register violates alignment restrictions.
MAC removal	Not supported	Supported
Hash mute	Only first 16 bytes can be muted. 8 mute entries.	Nibble mute control in an 16B region. The start of the region is 0-127B from the start of the hash toggle on token. The mute value for the session is part of the command.
AES-XTS	Sector size could be contiguous from 128B-16MB.	Sector size must be a power of two.
New algorithms		SHA224, 384, 512 KASUMI (UEA1, UIA1) SNOW3G (EEA1, EIA1) AES-128 (EEA2,EIA2) ZUC (EEA3,EIA3) RC4 Zlib
Discard inband data from output	Not supported	Supported
Discard predata fields from output	Output IV/IHV/AAD to host	Predata fields may be discarded from the output
Sideband buffer	Not supported	Supported
FPGA interface	Not supported	40G Interlaken interface
Skip by byte	Skip by 4 bytes	Skip by one byte
Power island	Not supported	Supported
Error handling	Potential for chip to hang after system error such as CT, CA, or ECRC	All errors reported through command status word. Some PCIe errors will also be reported via AER and related interrupts.
Free buffer	Command index is written back to free buffer ring. Free buffer use is enabled per ring.	Free buffer pointers are written back to the command structure Free buffer use is enabled per command.
Done bit in Result Pointer Ring	820x sets and clears Done bit	XR9240 sets and software clears Done bit
Temperature Sensor		Adds overheat threshold and interrupt

Table A-1. 820x/XR9240 Differences

Feature	820x	XR9240
SSL Acceleration	Does not support p_hash.	Supports p_hash.
Enhanced Compression		Supports both stateful gzip and gunzip. 1B lazy match for improved compression ratio. Automatic store/static Huffman/dynamic Huffman block selection.
Self test	The key is not stored locally.	Stores master key locally.

Document Revision History

This section lists the additions, deletions, and modifications made to this document for each release of this document.

Document Revision A

Update 1. Initial release.

Document Revision B

Update 1. Changed "Result Ring" to "Result Pointer Ring" and changed "Free Pool Ring" to "Free Buffer Pointer Ring" throughout. Changed "Four PK pairs" to "Eight PK Engines" throughout.

Update 2. Section 1.1.2 Engine Features: added supported for AES-XTS.

Update 3. Section 1.1.6 Other features: updated GPIO to 32 bits and added package information.

Update 4. Section 1.4 Ordering Information: added this new section.

Update 5. Section 2.2 Data Integrity: created separate drawings for the TE and PK data paths. Removed parity from the ingress and egress buffers.

Update 6. Section 2.2.2 SRAM Protection: Removed parity test from all RAMs. Changed 1-bit ECC errors as logged as warnings and 2-bit ECC errors logged as errors.

Update 7. Section 2.2.4 Real Time Verification: Added P_hash to the list of algos that do not support RTV.

Update 8. Section 2.2.4.2 Authentication Real Time Verification: removed sentence on combined file hash + slice hash as this is no longer supported.

Update 9. Section 3.2.2.2 Transform Engine Session Descriptor: added PL_SB to bit 18 and KEK_EN to bit 24 of Session Control Word 0. Added AES-XTS and EEA2 to ENC_ALG[3:0], KEK_IDX[2:0], KEK_ALG, and XTS_CTL[7:0] to Session Control Word 1.

Update 10. Section 3.2.2.3 Transform Engine User Descriptor: added CMP_HIST to the source token flag field.

Update 11. Section 3.2.3.2 Data Input Requirements: added comp and decomp requirements. Added AES-XTS enc requirements. Added SSLv3-MAC-SHA1 auth requirements.

Update 12. Section 3.4 Result Pointer Ring: added Note 1 about Done bit.

Update 13. Section 3.4.1 Destination Buffer Organization: added options table.

Update 14. Section 3.4.2 Sideband Data: added Pad output to TYPE[3:0]. Expanded descriptions in Table 3-8.

Update 15. Section 4.2 Crypto Processing Sequence: moved section on processing over the ILK interface to the FPGA Design Application Note.

Update 16. Section 4.3.1 General Token Rules: added item 4.

Update 17. Section 4.5 Accessing an External Programmable Device: added this new section.

Update 18. Section 5.1.2 ARI: added requestor ID format tables.

Update 19. Section 5.1.1.4 Virtual Function to Physical Function Communication: added communication procedures in table and written form.

Update 20. Section 5.2.2 Messages: added sentence about Set_Slot_Power.

Update 21. Section 5.2.3.3 MSI-X Interrupts In IOV Mode: removed statement that vector assignments may be stored in Flash.

Update 22. Section 5.3.2.2 Best Effort Arbitration: corrected typo in %BW formula.

Update 23. Section 5.4 TE: added single operations to the table of supported TE algorithms.

Update 24. Section 5.4.1 Compression/Decompression Engine: created new sections for LZS/eLZS and gzip/zlib/Deflate.

Update 25. Section 5.4.2 Authentication Engine: Added SSL3.0-MAC, XCBC-MAC, P_hash, UIA1, EIA1, EIA2, EIA3 algos. Created new sections for describing each algo in detail.

Update 26. Section 5.4.2 Encryption/Decryption Engine: Added AES-XTS, UIA1, EEA1, EEA2, EEA3 algos. Created new sections for describing each algo in detail.

Update 27. Section 5.7 RNG: removed the detailed implementation drawing and text.

Update 28. Section 5.10 Device Power Management: added this new section.

Update 29. Section 5.11 SPI: added reference to Exp ROM access.

Update 30. Section 6.2.1 PCIe Interrupt Source Register: added more description of how to clear bits in this register.

Update 31. Section 6.2.2 PCIe Software Interrupt Enable Register: added this new section.

Update 32. Section 6.2.3 - Section 6.2.8: PCIe Interrupt Mask Set Register, PCIe Interrupt Mask Clear Register, PCIe Interrupt Mask Clear Register, PCIe Interrupt Auto-Clear Register, PCIe Interrupt Auto-Mask Register, PCIe Interrupt Configuration Register, PCIe Interrupt Throttle Register: added these new sections.

Update 33. Section 6.3 IOV Communication Registers: added this new section.

Update 34. Section 6.4.5 TLP Processing Hint (TPH) Registers: added this new section.

Update 35. Section 6.5.1 Status Register: removed ERR_STAT, INIT_STAT.

Update 36. Section 6.5.4 Module Enable Register: added this new section.

Update 37. Section 6.5.5 Clock Enable Register: added this new section.

Update 38. Section 6.5.9 Temperature Sensor Control 0 Register: updated reset value for coefficient M.

Update 39. Section 6.6 PCIe DMA Control Registers: Added the ECC Error Injection registers.

Update 40. Section 6.7 ILK Control Registers: Added the DMA, PHY and MAC registers to this section.

Update 41. Section 6.9 TE Control Registers: Added this new section.

Update 42. Section 6.10 RNG Control Registers: Added this new section.

Update 43. Section 6.11 GPIO Control Registers: Added this new section.

Update 44. Section 6.12 SPI Control Registers: Added this new section.

Update 45. Section 6.8 PK Control Registers: Added the ECC Error Injection registers. Added a unique address for each of the PK engines for each register.

Update 46. Section 6.8.1 PK Address Register: corrected ADDR field to [14:0].

Update 47. Section 6.9.1 gzip Control Register: corrected fields descriptions.

Update 48. Chapter 8 PCIe Configuration Register Definition: added this new chapter.

Update 49. Chapter 9 Signal Definition: renamed some signals to match the layout.

Update 50. Section 10.4.1 Transform Engine Errors: added FIPS error and gzip_decode table. Added kek_dec_err. Removed parity errors.

Update 51. Section 10.4.2 Public Key Errors: changed pk_run_err to pk_len_err.

Update 52. Chapter 14 Package Information: added this new chapter.

Update 53. Appendix A Table A-1: Hash calculation for 920x does not support file and slice hash in the same pass. Removed parity from 920x Integrity. Added 920x support for AES-XTS. Added last 3 entries in table.

Document Revision A02

Update 1. Section 1.1.3 PCI Express 3.0 Compatible Bus Interface: removed support for UEFI.

Update 2. Section 3.2.3.2 Data Input Requirements, Table 3.5 Authentication Source Buffer Data Requirements: changed IV token for GCM to don't care when LB = 1. Table 3.6 AEAD Source Buffer Data Requirements: changed IV values.

Update 3. Section 4.5 Accessing an External Non-Volatile Device: removed the UEFI region.

Update 4. Section 5.1.2.4 PCIe Link Power Management: updated power management state machine drawing.

Update 5. Section 5.6.4 Programming the ILK PHY: updated the instructions in this section.

Update 6. Section 6.4.21 Outstanding Command Counter 0 Register: added this new register.

Update 7. Section 6.4.22 Outstanding Command Counter 1 Register: added this new register.

Update 8. Section 6.4.23 Outstanding Command Counter 2 Register: added this new register.

Update 9. Section 6.4.25 PCIe Error Injection Register: added this new register.

Update 10. Section 6.6.2 ILK PHY Control Registers: added the required fields for the ILK DMA PHY Access register to Table 6-1 and to all the registers in this section.

Update 11. Section 6.6.2.7 ILK PHY Synthesis Lane Status Register: added this new register.

Update 12. Section 6.6.2.8 ILK PHY Synthesis Lane Status Register - Section 6.6.2.13 ILK PHY Loopback Pattern Register: added these new registers for testing the ILK PHY.

Update 13. Section 9.2 Interlaken Interface: added ILK_BYPS_CLK signal.

Update 14. Section 9.3 GPIO Interface: added FIPS_GPIO[7:0].

Update 15. Section 9.6 Miscellaneous Interface: made signal PCIE_LINKUP active high instead of active low. Added signals TEMP_SENSE[3:0].

Update 16. Section 9.7 JTAG Interface: added PCIe PHY JTAG interface pins. Removed the "A" from the JTAG_ signal names.

Update 17. Section 10.3.1 Transform Engine Errors: corrected error codes for ILK.

Update 18. Section 14.2 Mechanical Information: added the package drawings.

Update 19. Section 14.3 Ball Map Drawings: Updated to show FIPS_GPIO[7:0] and PCIe PHY JTAG interface pins. Added signals TEMP_SENSE[3:0] and ILK_BYPS_CLK.

Update 20. Section 14.4 Signal Names Lists: Updated pin lists to include FIPS_GPIO[7:0] and PCIe PHY JTAG interface pins. Removed the "A" from the JTAG_ signal names. Added signals TEMP_SENSE[3:0] and ILK_BYPS_CLK.

Update 21. Chapter 11 DC Specifications: updated all tables in this chapter.

Update 22. Chapter 12 AC Specifications: updated all tables in this chapter.

Document Revision A03

Update 1. Modified document to discuss XR9240 part only.

Update 2. Section 5.1.1.1 Function Level Reset: added note that host must ensure that VF has no outstanding commands before reset.

Update 3. Section 9.3 GPIO Interface: renamed alternate GPIO to CONFIG_GPIO[7:0].

Update 4. Section 9.6 Miscellaneous Interface: added new signals CLK_TST_SEL[1:0], and renamed some signals to CONFIG[3:0].

Update 5. Section 9.8 Power and Ground Interface: changed all 0.93V power supplies to 0.9V.

Update 6. Chapter 11 DC Specifications: changed all 0.93V power supplies to 0.9V.

Update 7. Section 11.2.1 Digital Power Supplies: updated all min and max values.

Update 8. Section 11.4 Power Consumption: updated this entire section.

Update 9. Section 14.3 Ball Map Drawings: updated all drawings with new pin names.

Update 10. Section 14.4 Signal Names Lists: updated all with new pin names.

48720 Kato Road
Fremont, CA 94538
p: 510.668.7000
www.exar.com