MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. **Moving Forward Faster**

Document Classification: Proprietary Information

Document Conventions

↘	**Note:** Provides related information or information of special importance.
!	**Caution:** Indicates potential damage to hardware or software, or loss of data.
ϟ	**Warning:** Indicates a risk of personal injury.

Document Status

Doc Status: Preliminary	Technical Publication: 0.xx

For more information, visit our website at: www.marvell.com

Disclaimer

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document.

Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications.

With respect to the products described herein, the user or recipient, in the absence of appropriate U.S. government authorization, agrees:

1) Not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2;

2) Not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and,

3) In the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML").

At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.

Copyright © 2014. Marvell International Ltd. All rights reserved. Alaska, ARMADA, CarrierSpan, Kinoma, Link Street, LinkCrypt, Marvell logo, Marvell, Moving Forward Faster, PISC, Prestera, Qdeo (for chips), QDEO logo (for chips), QuietVideo, Virtual Cable Tester, Xelerated, and Yukon are registered trademarks of Marvell or its affiliates. Avanta, Avastar, DragonFly, HyperDuo, Kirkwood, Marvell Smart, Qdeo, QDEO logo, The World as YOU See It, Vmeta and Wirespeed by Design are trademarks of Marvell or its affiliates.

Patent(s) Pending—Products identified in this document may be covered by one or more Marvell patents and/or patent applications.

Revision History

Revision	Date	Comments
A	29-May-14	Unrestricted Release

Table of Contents

Copyright © 2014 Marvell

Copyright © 2014 Marvell

Copyright © 2014 Marvell

Copyright © 2014 Marvell

List of Tables

Copyright © 2014 Marvell

Copyright © 2014 Marvell

Copyright © 2014 Marvell

List of Figures

Copyright © 2014 Marvell

Preface

About this Document

This document describes the architectural aspects and the features implemented in the Marvell® ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors.
It also provides full register definitions for these devices.

This document is intended to be the basic source of information for designers of new systems. All feature descriptions and specifications described in this document do not necessarily apply to all of the devices. Refer to Section 1, Product Overview, on page 35 for a description of the features and interfaces relevant to each of these devices.

In this document, the MV78230, MV78260, and MV78460 are often referred to as “ARMADA® XP” or “the device/s”.

Copyright © 2014 Marvell

This Functional Specification is organized into the following parts:

- Part 1: Overview, on page 33

  Provides a general and functional overview and includes the address map and a description of t the internal architecture, system considerations, and bootROM firmware.

- Part 2: CPU Subsystem, on page 119

  Provides a brief description of the CPU, and a description of the Level-2 (L2) cache add-on, the cache coherency fabric, the Multiprocessor Interrupt Controller (MPIC) add-on, the general and watchdog timers/counters, and the debug modes.

- Part 3: External Memory Interfaces, on page 171

  Provides a description of the following external memory units:

  - DDR3 DRAM controller
  - NAND Flash controller
  - Device Bus controller.

- Part 4: Ethernet Networking Subsystem, on page 251

  Provides a description of the components of the Ethernet Networking subsystem:

  - Ethernet Networking controller
  - Hardware Buffer Management controller
  - 1G/2.5G Ethernet MAC
  - Precise Timing Protocol core

- Part 5: External Interfaces, on page 331

  Provides a description of each of the external interfaces:

  - PCI Express
  - UBS
  - SATA
  - Serial Peripheral Interface
  - Time Division Multiplexing
  - SDIO
  - I^{2}C
  - UART
  - Real-Time Clock
  - General Purpose Input Output

- Part 6: System Functions and Engines, on page 447

  Provides a description of the following system functions and engines:

  - Power management capabilities
  - Integrated multiple Marvell® high-speed SERDES interface
  - Cryptographic engines and security accelerators (CESA)
  - Two dual-channel XOR engines
  - IDMA controller with its four independent IDMA engines
  - One-time programmable eFuse module

- Appendix A: Register Set, on page 539

  Details the MV78230/78x60 registers.

Copyright © 2014 Marvell

Relevant Devices

- MV78230
- MV78260
- MV78460

Related Documentation

The following documents contain additional information related to the MV78230/78x60. For the latest revision, contact a Marvell representative.

Table 1: Related Documents

Ref # and Title		Document Number
[1]	*MV78230 Hardware Specifications*	MV-S106687-00
[2]	*MV78260 Hardware Specifications*	MV-S106688-00
[3]	*MV78460 Hardware Specifications*	MV-S106689-00
[4]	*ARMADA®* XP *MP Core Highly Integrated Marvell® ARMv7 SoC Processors Datasheet*	MV-S108492-00
[5]	*Serial ETM3 Common Physical (PHY) Layer Preliminary Specifications*	MV-S105867-00
External Documents:		
[6]	*SATA II phase 1.0a specification (Extensions to SATA I specification)*	
[7]	*USB-HS High-Speed Controller Core Reference*	

See the Marvell Extranet website for the latest product documentation.

Document Conventions

The following conventions are used in this document:

Signal Range	A signal name followed by a range enclosed in brackets represents a range of logically related signals. The first number in the range indicates the most significant bit (MSb) and the last number indicates the least significant bit (LSb). Example: DB_Addr[12:0]
Active Low Signals #	An n letter at the end of a signal name indicates that the signal's active state occurs when voltage is low. Example: INTn
State Names	State names are indicated in *italic* font. Example: *linkfail*
Register Naming Conventions	Register field names are indicated by angle brackets. Example: <RegInit> Register field bits are enclosed in brackets. Example: Field [1:0] Register addresses are represented in hexadecimal format. Example: 0x0 Reserved: The contents of the register are reserved for internal use only or for future use. A lowercase <n> in angle brackets in a register indicates that there are multiple registers with this name. Example: Multicast Configuration Register<n>
Reset Values	Reset values have the following meanings: 0 = Bit clear 1 = Bit set
Abbreviations	B: Byte Kb: kilobit KB: kilobyte Mb: megabit MB: megabyte Gb: gigabit GB: gigabyte
Numbering Conventions	Unless otherwise indicated, all numbers in this document are decimal (base 10). An 0x prefix indicates a hexadecimal number. An 0b prefix indicates a binary number.

Terms and Abbreviations

The following terms and abbreviations are used throughout this datasheet:

Table 2: Terms and Abbreviations

Acronym / Term	Definition
3DES	Triple Data Encryption Standard
AER	Advanced Error Reporting
AES	Advanced Encryption Standard
AES128/128	128 data bits AES with 128-bit key width
AES128/192	128 data bits AES with 192-bit key width
AES128/256	128 data bits AES with 256-bit key width
AMBA®[1]	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ARM	Advanced RISC Machine
ARP	Address Resolution Protocol
ATB	Advanced Trace Bus
AVB	Audio Video Bridging
BAR	Base Address Register
BIST	Built-In Self Test
Block/data	Block of 512 bits in the authentication engine
BMD	Bandwidth Map Decode
BMU	Buffer Management Unit
BP	Buffer Pointer
BPDU	Bridge Protocol Data Unit
BPU	Branch Prediction Unit
CBC	Cipher Block Chain
CESA	Cryptographic Engines Security Accelerators
CFB	Cipher Feedback
CFU	Coherency Fabric Unit
CIB	Coherency IO Bridge
CRC	Cyclic Redundancy Check
CTI	Cross Trigger Interface
CTM	Cross Trigger Module
DA	Destination Address
DAP	Debug Access Port
DES	Data Encryption Standard
3DES	Triple Data Encryption Standard
DLB	DRAM Line Buffer
DMA	Direct Memory Access

Table 2: Terms and Abbreviations (Continued)

Acronym / Term	Definition
DMB	Data Memory Barrier
DPRAM	Dual-Port Random Access Memory
DSA	Distributed Switching Architecture
DSB	Data Synchronization Barrier
DSCP	Differentiated Service (Diff-Serv) Code Point
DSP	Digital Signal Processor
ECB	Electronic Code Book
EDE	Encryption Decryption Encryption
EDMA	Enhanced-DMA
EEE	Encryption Encryption Encryption (Cryptographic Engine and Security Accelerator (CESA))
	Energy Efficient Ethernet
EHCI	Enhanced Host Controller Interface
EJP	Egress Jitter Pacing
ETB	Embedded Trace Buffer
ETM	Embedded Trace Macrocell
FCS	Frame Check Sequence
FPDMA	First Party DMA
FPGA	Field Programmable Gate Array
GbE	Gigabit Ethernet
GPIO	General-Purpose I/O Port
HDLC	High-Level Data Link Control
I^{2}C	Inter-Integrated Circuit Interface
ICE	In-Circuit Emulation
ISB	Instruction Synchronization Barrier
iSCSI	Internet Small Computer System Interface
IV	Initial Vector/Initial Value
JTAG	Joint Test Action Group
LLID	Logical Link Identification
LPAE	Large Physical Address Extension
MAC	Media Access Controller
MCDMA	Multi Channel Direct Memory Access
MCSC	Multi-Channel Serial Controller
MD5	Message Digest 5
MMC	Multi-Media Card
MMU	Memory Management Unit
MRCRx	MCSC Channel-x Receive Configuration Register

Copyright © 2014 Marvell

Table 2: Terms and Abbreviations (Continued)

Acronym / Term	Definition
MSI	Message Signaled Interrupts
NAPT	Network Address Port Translation
NFP	Network Fast Processing
ODT	On-Die Termination
OFB	Output Feedback
PCIe	PCI Express
PCM	Pulse Code Modulation
PCS	Physical Coding Sublayer
PLS	Power Levelling Sequence
PM	Power Management
PMU	Power Management Unit
PPPoE	Point-to-Point over Ethernet
PRBS	Pseudo-Random Bit Sequence
PTM	Program Trace Macrocell
PTP	Precise Timing Protocol
PSI	Physical Synchronization and Alignment
PTM	Program Trace Macrocell
QoS	Quality of Service
QSGMII	Quad Serial Gigabit Media Independent Interface
RGMII	Reduced Gigabit Media Independent Interface
RISC	Reduced Instruction Set Computer
RMON	Remote Monitoring
ROM	Read Only Memory
RTC	Real-Time Clock
SA	Source Address
SAR	Sample At Reset
SATA	Serial Advanced Technology Attachment
SATAHC	Serial-ATA II Host Controller
SDIO	Secure Digital Input/Output
SERDES	Serializer/Deserializer
sETM	Serial Embedded Trace Macrocell
SGL	SERDES Glue Logic
SGMII	Serial Gigabit Media Independent Interface
SHA-1	Secure Hash Algorithm 1
SLAC	Subscriber Line Audio-processing Circuit
SLIC	Subscriber Line Interface Circuit

Table 2: Terms and Abbreviations (Continued)

Acronym / Term	Definition
SMI	Serial Management Interface
SoC	System on Chip
SPI	Serial Peripheral Interface
TAI	Timing Applications Interface
TBI	10-Bit Interface
TC	Traffic Class
TCP	Transmission Control Protocol
TDM	Time Division Multiplexing
TPIU	Trace Port Interface Unit
TTL	Time to Live
UART	Universal Asynchronous Receiver/Transmitter
UDP	User Datagram Protocol
USB	Universal Serial Bus
UTM	Upstream Traffic Module
VID	VLAN Identifier
VLAN	Virtual Local Area Network
W0…W15	Designates the 16 words in an authentication input data block; W0 is the first word and W15 is the last word.
WOL	Wake On LAN
WORD	32-bit

[1] AMBA are registered trademark of the ARM Corporation.

Copyright © 2014 Marvell

MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Part 1: Overview

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. Moving Forward Faster

Document Classification: Proprietary Information

THIS PAGE IS INTENTIONALLY LEFT BLANK

1 Product Overview

The ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors are a complete system-on-chip solution based on the Marvell® Core Processor embedded CPU technology. By leveraging the successful Marvell system controllers and extensive expertise in ARM instruction-set-compliant CPUs, the ARMADA® XP SoCs present a new level of performance, integration, and efficiency to raise the performance/power and performance/cost bar.

These devices integrate:

- Dual/quad Superscalar, ARMv7-compliant Marvell® Core Processors with the latest Marvell micro-architecture enhancements, with a double precision IEEE-754-compliant Floating Point Unit (FPU) per core.
- Shared Level-2 (L2) cache in a size of 1 MB or 2 MB.
- Low-latency, high-bandwidth tightly coupled DDR3 memory controller.

The advance I/O peripherals include PCI Express (PCIe) Gen2.0, USB2.0 with integrated PHYs, SATA ports, Ethernet, and TDM interfaces.

To allow an enhanced handshake and data flow, a full hardware I/O cache coherency scheme is implemented between the I/Os and the CPU.

Optimized for low-power operation and providing advanced power management capabilities, the 40 nm process-based ARMADA® XP is ideally suited for a wide range of applications that require both high performance and minimal power consumption. The rich and diversified interface mix of the series facilitates being the perfect solution for different types of applications and systems in various fields, such as:

- ARM-based servers and workstations
- High density high performance clusters and computational farms
- Networking control plane applications
- Wireless infrastructure:
  - Cellular
  - WiMax
  - Wifi
- High-end consumer and enterprise appliances like laser printers
- Surveillance Video Recorders (DVR/NVR/Hybrid)
- Enterprise Network Storage (NAS, RAID, iSCSI) products

The innovative Coherency Fabric architecture provides a coherent interconnect between the CPUs themselves and between the CPUs and the I/O masters. This enables the system to work either in Symmetrical Multi Processing (SMP) mode or Asymmetric Multi Processing (AMP) mode, with I/O cache coherency. In addition, the efficiency of the bus enables a high-frequency, high- bandwidth, and low-latency access time throughout the CPU memory subsystem.

The on-chip Mbus architecture, a Marvell® proprietary crossbar interconnect for non-blocking any-to-any connectivity, enables concurrent transactions among multiple units. This design results in high system throughput, allowing system designers to create high-performance products.

With full pin and software compatibility between the different device flavors, the ARMADA® XP enables full performance scalability to best fit the requirements of any specific application.

Table 3 lists the feature differences and similarities between the ARMADA® XP devices:

- MV78230
- MV78260
- MV78460

Table 3: ARMADA® XP Device Differences and Similarities

Feature	MV78460	MV78260	MV78230
		Differences	
Marvell® Core Processor ARMv7- compliant CPU, with a Floating Point Unit (FPU) per CPU core	Quad CPU Cores @ up to 1.6 GHz	Dual CPU Cores @ up to 1.6 GHz	Dual CPU Cores @ up to 1.6 GHz
Shared L2 cache	2 MB	1 MB	1 MB
DDR3 SDRAM Interface includes 8-bit support for Error Checking and Correction (ECC) checking and generation.	40/72-bit Width DDR3-1600 MHz	40/72-bit Width DDR3-1600 MHz	40-bit Width DDR3-1600 MHz
Gigabit Ethernet Interface GMII/MII/RGMII/SGMII/QSGMII **NOTE:** The GbE interface options are multiplexed with other functionality. For more information about the SERDES options, see the device's *Hardware Specifications*.	4 Ports	4 Ports	3 Ports
SERDES Lanes Multiplex of: • SGMII • PCIe • SATA • sETM • QSGMII	16 Lanes	12 Lanes	7 Lanes
PCI Express (PCIe) Gen2.0 Interface **NOTE:** The SERDES lanes are multiplexed with other functionality. For more information about the SERDES options, see the device's *Hardware Specifications*.	4 PCIe Interfaces: • 2 units x4 or quad x1 • 2 units x4/x1	3 PCIe Interfaces: • 2 units x4 or quad x1 • 1 unit x4/x1	2 PCIe Interfaces: • 1 unit x4 or quad x1 • 1 units x1
Device Bus	8/16/32-bit Interface	8/16/32-bit Interface	8/16-bit Interface
		Similarities	
Serial ATA II (SATA II) Interface with Integrated PHY(s) **NOTE:** The SERDES lanes are multiplexed with other functionality. For more information about the SERDES options, see the device's *Hardware Specifications*.		2 Ports	
USB Interface		3 Ports USB2.0 w/ integrated PHYs	
Cryptographic Engine and Security Accelerator (CESA)		2 Engines	
XOR DMA Engine		4 Channels	
Serial Peripheral Interface (SPI)		2 Ports	

Copyright © 2014 Marvell

Table 3: ARMADA® XP Device Differences and Similarities (Continued)

Feature	MV78460	MV78260	MV78230
SDIO/MMC Interface	Yes		
Secured Boot	Yes		
16750-Compatible UART Interface	4 Ports		
I^{2}C Interface	2 Ports		
Time Division Multiplexing (SLIC/codec) Interface	32 Channels		
Advanced Power Management Unit	Yes		
Advanced Interrupt Controller	Yes		

Block Diagrams

The following figures illustrate the main functional blocks within the MV78230/78x60 device.

Figure 1: MV78230 Block Diagram

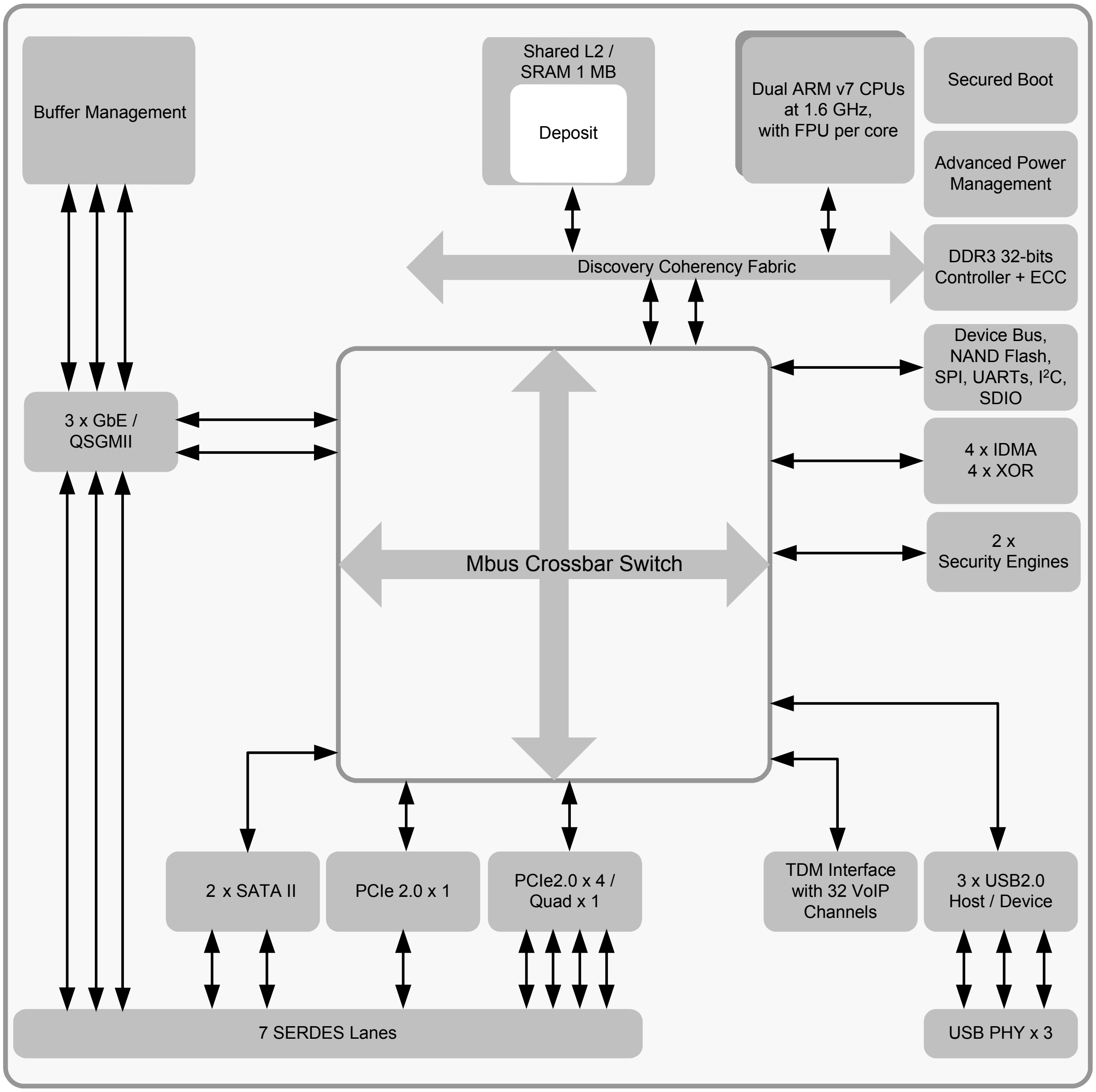


Figure 2: MV78260 Block Diagram

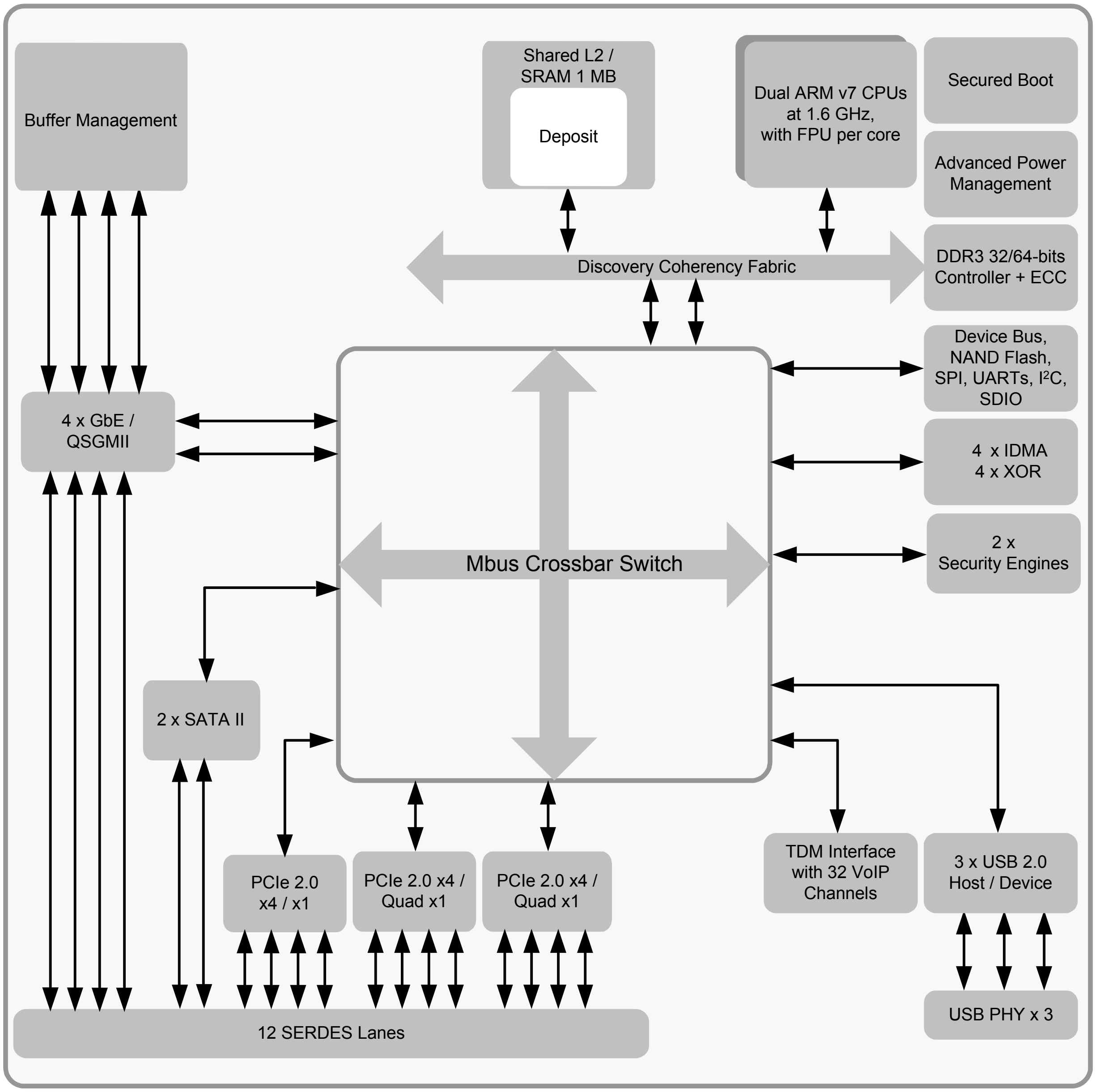
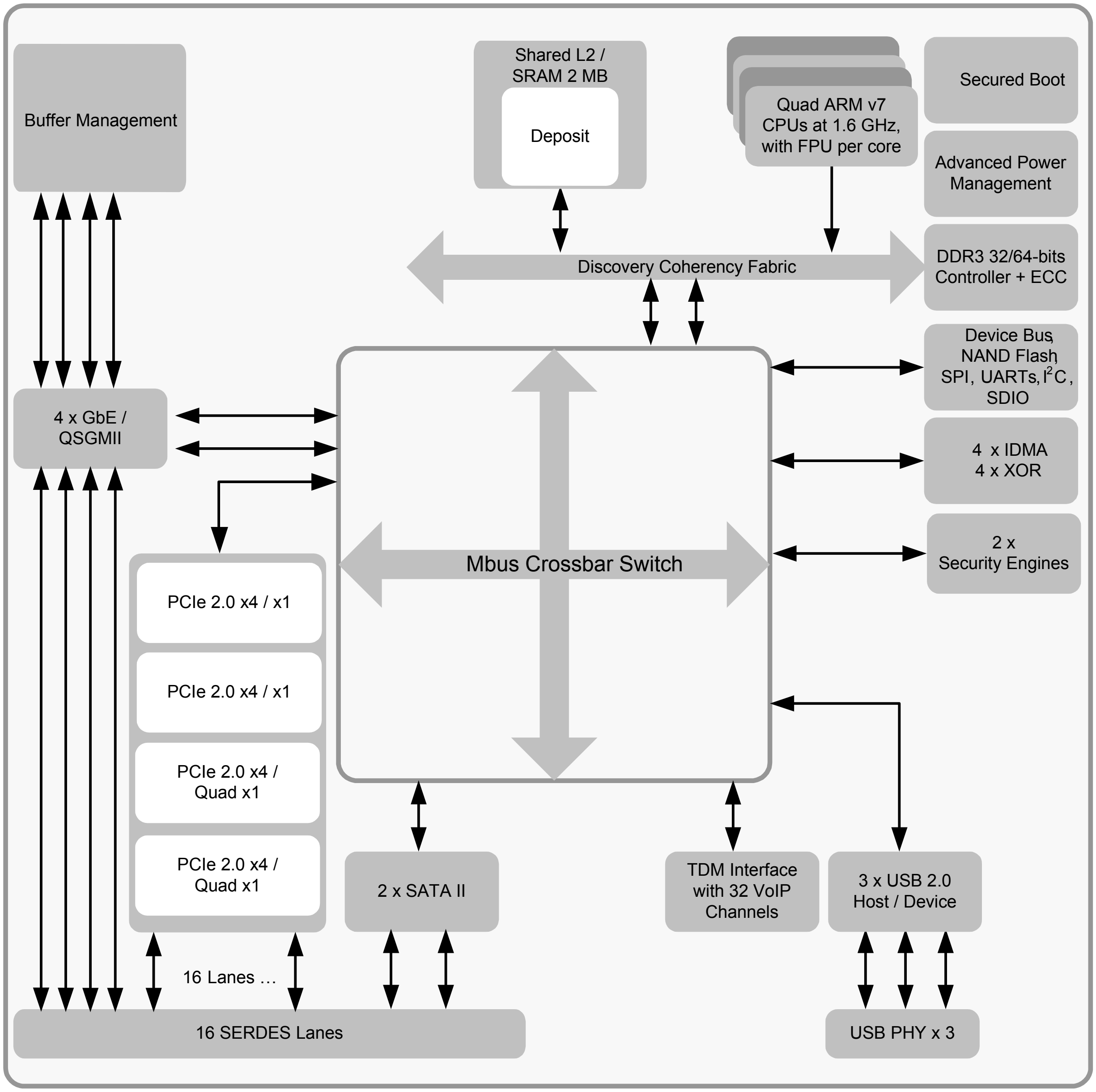


Figure 3: MV78460 Block Diagram



Copyright © 2014 Marvell

2 Functional Overview

The following sections describe the functional blocks and features of the ARMADA® XP.

2.1 Marvell® Core Processor

The ARMADA® XP devices are based on the dual-issue, Marvell® Core Processor CPU.
This ARMv7-MP-compliant core supports the following features:

- 2.6 DMIPS/MHz per core
- Superscalar RISC CPU with Harvard architecture that issues two instructions per cycle
- Single/double precision IEEE-754-compliant Floating Point Unit (VFP3-D32)
- Compliant with ARMv7-A architecture, published in the *ARM Architecture Reference Manual*, Second Edition
- Supports 32-bit instruction set for performance and flexibility
- Thumb-2 and Thumb-EE instruction set for code density
- Supports Large Physical Address Extension —up to 40-bit address space
- Supports DSP instructions to boost performance for signal processing applications
- MMU-ARMv7 compliant VMSA MMU
- 4-KB L0 Instruction and data cache, direct mapping
- 32-KB L1 Instruction cache 4-way set-associative, physically indexed, physically tagged, parity protected
- 32-KB L1 Data cache, 8-way set-associative, physically indexed, physically tagged, parity protected
- MESI Cache Coherency scheme
- Hit-under-miss and multiple outstanding requests
- Advanced write coalescing support
- Variable stages pipeline—6 to 10 stages
- Out-of-order execution for increased performance
- In-order retire via a Reordering Buffer (ROB)
- Advanced branch prediction—32 Branch Target Buffer (BTB) and 1K entries Branch Prediction Unit (BPU) with GShare algorithm
- Branch Return Stack Point for subroutine call
- 64-bit internal data bus with 64-bit load/store instructions
- JTAG/ARM-compatible ICE, and Serial Embedded Trace Module (sETM) for enhanced real-time debug capabilities
- Endianess options: Little, Big, and Mixed Endianess

For further information about the Marvell® Core Processor interface, see Section 7, Marvell® Core Processor, on page 121.

2.2 Level-2 Cache

The devices integrate a shared, unified Level-2 cache.

MV78460	2 MB, 32-way set-associative cache
MV78230 **MV78260**	1 MB, 16-way set-associative cache

L2 cache features include:

- Physically addressed
- Non-blocking pipeline supports multiple outstanding requests and Hit Under Miss (HUM) operation
- Per-way configured byte addressable SRAM or L2
- Direct I/O access to/from L2/SRAM for all Mbus masters, allowing data storing directly into the L2/SRAM
- Data/Tag ECC/Parity protected

For further information about the L2-Cache, see Section 8, Level-2 Cache, on page 122.

2.3 System Considerations

For proper and effective system operation, the device provides specific system level tools that include:

- Data integrity mechanisms
- Secured boot process

The MV78230/78x60 offers multiple tools to ensure data integrity in data transfers between various interfaces of the device.

It supports ECC (Error Checking and Correction) on SDRAM and on the L2 cache, with single error correction and two errors detection (SEC/DED). It supports read-modify-write access for partial writes. If a non-correctable ECC error occurs, an interrupt is set and the transaction address and data are registered.

The PCI Express link is LCRC protected, as defined by PCI Express specification.

The device also supports parity protection on its internal data path and internal memory arrays, providing end-to-end data integrity.

ECC/parity errors are optionally propagated between the interfaces, and can be reported to the host processor via a maskable interrupt.

For further information about the Data integrity capabilities, see Section 5, System Considerations, on page 82.

2.4 BootROM Firmware

The bootROM firmware is executed according to the sample at reset configuration bits in the Sample at Reset Register (Table 1286 p. 1434).

The bootROM firmware performs basic and fundamental initialization of the device. It also loads and executes code from any of the following boot devices:

- Serial (SPI) flash
- NAND flash
- NOR flash
- SATA interface
- PCI Express interface
- UART interface (using the Xmodem protocol)

Boot from the UART interface (using the Xmodem protocol) is attempted before a boot from any of the other interfaces. The device attempts to detect the various UART interfaces before proceeding with boot from the selected interface.

The firmware can also perform a flexible multi-core initialization, and support the device's advanced power save capabilities.

For further information about the Boot ROM, see Section 6, BootROM Firmware, on page 93.

Secured Boot

The trusted boot framework uses the embedded secured boot firmware and the embedded One Time Programmable (OTP) memory to perform secured and authenticated boot sequences of one or more CPUs. This framework is based on a flexible chain of trust using an industry standard public/private key cryptographic scheme. Using the authenticated boot sequence, the boot loader can be loaded from several resources, including SPI Flash, Flash memory (both NAND and NOR), and SATA. The operating system software can be loaded from additional resources, including the network (using TFTP).

The secured boot flow and boot device is enforced by an OTP memory configuration that also disables the JTAG debug interface.

For further information about the secured boot process, see Section 6, BootROM Firmware, on page 93.

2.5 Coherency Fabric

The Coherency Fabric provides a flexible, high speed, low latency switching structure for routing the CPUs and I/O initiated transactions to the target modules. The device manages cache-to-cache coherency between the different coherent processors and forwards I/O initiated transactions that are tagged with a shared attribute onto the processors snoop engine, allowing on-chip caches to snoop these transactions (I/O cache coherency). Any access to the DRAM either by an IO master or by a CPU may result in a snoop transaction driven through the peer CPUs' caches. In the case of a hit in a modified line in the CPU cache, the most updated line is delivered directly to the requester.

The Coherency Fabric allows the system architecture to use the CPUs in the most efficient way that fits the given application. The CPUs may be used to run in a Symmetrical Multi-Processing (SMP) mode, in which all of the CPUs are running the same OS. Or, in an Asymmetrical Multi Processing (AMP) mode in which all of the CPUs can run different OS images or bare metal code.

The high performance of the Marvell® CPU cores and the flexibility of the Coherency Fabric enable system applications to combine between the two modes. For instance, some CPUs can run in SMP mode, and the remaining cores can run in AMP mode. Or, it is possible to create two SMP groups, with two CPUs per group. An example of such a system can be a networking application in which two CPUs are running the networking control plane stack in SMP mode. This provides intense computing power for the demanding requirements of the control plane. Meanwhile, the other two CPUs are running asymmetrically different images of light bare-metal code performing data plane tasks.

Using a snoop affinity scheme, the logic ensures that a memory access from an I/O master only snoops the CPUs that were assigned to handle this I/O master, preventing unnecessary disturbance to other CPU group's work.

2.6 Multiprocessor Interrupt Controller (MPIC)

The Multiprocessor Interrupt Controller (MPIC) is a single functional unit placed in the device Coherency Fabric. It provides multiprocessor interrupts management, and is responsible for receiving interrupts from different sources, prioritizing them, and distributing them to the target CPUs.

The interrupt controller can generate two CPU interrupts:

- Interrupt Request (IRQ)
- Fast Interrupt Request (FIQ)

Each interrupt has its own masking option that enables for the presentation of different events on different CPU interrupt pins.

The interrupt controller provides an advanced priority mechanism that assigns a priority to an interrupt to create an interrupt hierarchy. Running at coherency fabric clock frequency enables low-access latency for the processors to the interrupt registered information.

The interrupt controller also supports generation and broadcasting of interrupts between all of the multi-processing processors.The separated masking options provide the capability to assign a specific system event to a specific CPU core, while not affecting the other CPUs' operation.

The device's GPIO lines can also be configured to act as interrupt inputs that enables registration of external interrupts toward the Marvell® Core Processors.

The PCI Express port supports receiving interrupt messages (MSIx) from PCI Express endpoints and route them to the local hosts. When configured to act as a PCI Express endpoint, it is also capable of generating MSI messages.

For further information about the Multiprocessor Interrupt Controller, see Section 9, Multiprocessor Interrupt Controller (MPIC), on page 139.

2.7 Timers, Counters, and Watchdog

The device integrates four 32-bit wide general purpose global timers/counters and one global watchdog that can reset the device. Each can be selected to operate as a timer or as a counter.

In addition, the device integrates three 32-bit wide timers/counters—two general purpose and one watchdog, that are dedicated per CPU core. These are private timers/counters and are not visible to other CPU cores or external hosts.

The device provides the option of an interrupt assertion upon timer expiration, and the capability to enable a global reset in case of watchdog expiration.

For further information about the timers and counters, see Section 10, Timers, Counters, and Watchdog, on page 157.

2.8 CPU Debug Capabilities

The Serial Embedded Trace Module (sETM) is a CoreSight™ component designed for use with the CoreSight Design Kit. CoreSight is an ARM extensible, system-wide debug and trace architecture.

The sETM generates a real-time trace that can be configured to include instruction tracing controlled by programmable filtering and triggering. The trace generated by the sETM is in a highly compressed form, and it is traced out to the on-chip embedded buffers.

The device also incorporates a Cross Trigger Interface (CTI) per CPU Core that allows for debug event synchronization between the cores.

For further information about the debug features, see Section 11, Debug Capabilities, on page 161.

2.9 DDR3 SDRAM Controller

The SDRAM interface for each device is as follows:

MV78260 **MV78460**	40-bit or 72-bit DDR2 and DDR3 SDRAM interface (32/64-bit data + 8-bit ECC)
MV78230	40-bit DDR2 and DDR3 SDRAM interface (32-bit data + 8-bit ECC)

The DRAM controller supports up to four DRAM ranks and all device densities up to 8 Gb DDR3, up to a total of 16 GB memory space. Flexible configuration of different DIMM and on board DRAM components may be implemented.High frequencies are supported, even with a heavy load configuration, by the supported 2T and 3T modes. The controller offers support for the DDR3 fly-by topology architecture including:

- Read and write leveling
- Address mirroring
- Other Marvell® proprietary training mechanisms that allow fluent work with a fast interface as high-speed DDR3.

The MV78230/78x60 device architecture is oriented to reduce CPU-to-DRAM access time. The Marvell® Core Processor, the Coherency Fabric, and the DRAM interface all run synchronously. The CPU-Coherency Fabric-DRAM supported clock ratios provide maximum flexibility, by using 1:N and 2:N divisible options between the different clock domains.

The DRAM Controller have interfaces to the Coherent Fabric and to the Mbus interconnects. Mbus masters can access the DRAM Controller directly over the Mbus for non coherent transactions or through the Coherency Fabric for coherent transactions.

The DRAM controller is designed to best utilize DRAM bandwidth and latency through the following features:

- Up to 128B memory access in a single transaction from MBus masters
- Support for up to 32 simultaneous open pages
- Support for DDR3 BL8
- Issue an effective bank interleave

With the low power architecture of the MV78230/78x60, the devices provide options for putting the DRAM into power save modes, such as self refresh and power down, to minimize the overall system power consumption in low power states.

For further information about the DRAM controller, see Section 12, DRAM Controller, on page 173.

2.10 NAND Flash Controller (NFC)

Only the NFC version 1.0 can be used in parallel with the Device Bus. This option is not supported in the enhanced NFC version 2.0/

The NFC version 2.0 provides an 8/16 bits interface to NAND flash components over, and up to, four physical ranks. Each rank can host NAND Flash devices supporting 32/64/128/256 page block sizes and page sizes of either 512B, 2 KB, 4 KB, or 8 KB.

The controller has an integrated Error Correction Code (ECC) computation mechanism that corrects single-bit errors and detects 2-bit errors per page using Hamming ECC. Alternatively, it can compute ECC and correct up to 16 errors per 512B (including spare, if enabled and parity bits themselves) while using BCH ECC.

For further information about the NFC, see Section 13, NAND Flash Controller (NFC), on page 203.

2.11 Device Bus Controller

The supported device bus controller width for each device is as follows:

MV78260 **MV78460**	8/16/32-bit multiplexed address/data interface, and provides up to 8 data beats per access.
MV78230	8/16-bit multiplexed address/data interface, and provides up to 8 data beats per access.

The device bus controller supports different types of memory and I/O devices such as Flash, ROM, or an application-specific Field Programmable Gate Array (FPGA). It supports flexible timing parameters that enable accesses to slow asynchronous devices.

The device bus controller also supports a glueless interface to NOR Flash devices on any of its 5 physical ranks. Each rank can be address mapped up to 512 MB.

For further information about the Device bus interface, see Section 14, Device Bus Controller, on page 237.

2.12 Ethernet Networking Controller

The Network Ethernet ports for each device are:

MV78260 **MV78460**	Four 10/100/1000 Mbps, full duplex network Ethernet ports, with a configured selection of RGMII/SGMII/GMII/MII interfaces. For the configuration options, see Table 4.
MV78230	Three 10/100/1000 Mbps, full duplex network Ethernet ports, with a configured selection of RGMII/SGMII/GMII/MII interfaces. For the configuration options, see Table 5.

Table 4: Four 10/100/1000 Mbps Network Ethernet Ports—Configuration Options

Ethernet Port	Option 1	Option 2	Option 3
Port 0	RGMII / SGMII	GMII	MII
Port 1	RGMII / SGMII	SGMII	SGMII
Port 2	SGMII	SGMII	SGMII
Port 3	SGMII	SGMII	SGMII

Table 5: Three 10/100/1000 Mbps Network Ethernet Ports—Configuration Options

Ethernet Port	Option 1	Option 2	Option 3
Port 0	RGMII / SGMII	GMII	MII
Port 1	RGMII / SGMII	SGMII	SGMII
Port 2	SGMII	SGMII	SGMII

Note

Each port can be set as Not Connected (NC) if not used.

While working in SGMII mode, all of the ports can support a standard SGMII link with a net data rate of 1 Gbps (running at a baud rate 1.25 GHz) or a faster SGMII link with a net data rate of 2.5 Gbps (running at a baud rate of 3.125 GHz).

Receive and transmit buffer management is based on a buffer-descriptor list. Descriptors and data transfers are performed by the port-dedicated SDMA.

The Ethernet port includes advanced DA address filtering on received packets. It also detects packet type/encapsulations that can be used by the Marvell® Core Processor core for packet routing:

- Layer 2: BPDU, programmable VLAN-Ethertype (VLAN), Ethernet v2, LLC/SNAP
- Layer 3: IPv4, IPv6 (according to Ethertype), other (no MPLS detection)
- Layer 4: TCP, UDP, other

Each port has 8 receive and transmit priority queues. Queuing is performed based on DA, VLAN-Tag, and IP-TOS. The ports also support Marvell® proprietary Marvell Header or DSA Tag, resulting in significant CPU off-load when interfacing with Marvell switches.

Each port supports long frames, up to 10 KB. The port also supports TCP and UDP over IPv4/v6 checksum calculation on receive and generation on transmit, for off loading CPU's work.

The port also supports precise time stamping for packets, as defined in IEEE 1588 Precise Time Protocol (PTP) v1 and v2 and IEEE 802.1AS standards.

The Ethernet ports also support extended Wake On LAN (WOL) options for power save modes, and Energy Efficient Ethernet (EEE) LPI mode, for overall minimal system power consumption in power save modes.

For further information about the Network Ethernet Ports, see Section 15, Ethernet Networking Controller, on page 253.

2.13 Hardware Buffer Management (BM) Controller

The BM block may off load the software from the intensive task of memory buffer allocation for incoming or outgoing packets. On the ingress flow, the Ethernet port can pull a buffer from the BM with no software intervention. On the egress path, the Ethernet port may release the buffer back to its pool immediately after the packet is transmitted. In addition, the BM block provides the Marvell® core processor with the option to pull or release buffers during run time.

The BM block consists of four buffer pools that can be characterized by the size and memory location of the buffer. To improve performance, the BM locally stores in its FIFOs a configurable amount of buffers per pool for fast assignment. Watermarks are used to avoid overflow conditions. When the watermark is crossed, the BM empties some of the locally stored buffers back to the main buffer pools in memory. A similar watermark is used to fetch more buffers from the main pool into the local BM FIFOs.

2.14 PCI Express Interface

The PCIe interface for each device is as follows:

MV78460	4 PCIe units Gen2.0 Two of those units can be configured as x4 or quad x1 lanes. The other 2 units are x4 or x1.
MV78260	3 PCIe units Gen2.0 Two units can be configured as x4 or quad x1 lanes. One unit is x4/x1.
MV78230	2 PCIe units Gen2.0. One unit can be configured as x4 or quad x1 lanes. The other unit is x1.

The ports are PCI Express Base 2.0 compliant (according to the list in the table above) and may run at 5 Gbps in each direction per lane. Each of the lanes support lane polarity inversion and link reversal.

The PCIe port supports a single Virtual Channel (VC).

Upon CPU or DMA access to a PCIe address space, the PCIe port acts as a master. In master mode, it supports memory, I/O, and configuration cycles. It supports a maximum of 128B read and write requests, and up to four outstanding master read requests.

When it receives memory read/write requests from an external device, the PCIe port acts as a target. It supports up to four read requests, with a maximum read request size of 4 KB. As a target, it supports PCI Express accesses to all of the device’s internal registers, as well as all other chip resources.

The PCIe port uses 64-bit addressing, as a master or target and supports:

- Extended PCI Express configuration space
- Advanced error reporting, power management
- L0s and L1 ASPM active power state support

- Software L1 / L2 support for interrupt emulation message and error messages (MSIx)
- P2P bridging (non-transparent bridge) between each of the lanes

For further information about the PCIe controller, see Section 19, PCI Express Interface (PCIe) 2.0, on page 333.

2.15 Universal Serial Bus (USB 2.0) Interface

The MV78230/78x60 supports three USB 2.0 compliant ports, including integrated PHYs. Each port can be configured as a USB host or USB peripheral.

Each port integrates a USB controller (including a DMA and protocol engines) and an embedded USB 2.0 compatible PHY.

A common USB bridge connects all ports to the Mbus interconnect via an arbiter.

The USB ports are EHCI-compatible as hosts. When configured as a host, the port supports a direct connection to all peripheral types (LS, FS and HS).

As a peripheral, it supports up to six independent endpoints supporting control, interrupt, bulk, and isochronous data transfers and connects to all host types (FS or HS) and hubs.

For further information about the USB interface, see Section 20, Universal Serial Bus (USB 2.0) Interface, on page 352.

2.16 Serial-SATA (SATA) II Interface

The device includes two SATA II compliant ports, with integrated SERDES.

There is full support for the SATA II Phase 1.0 specification. The following advanced SATA II Phase 2.0 specification features are also supported:

- SATA II 3-Gbps speed
- Advanced SATA PHY characteristics for SATA backplane support
- SATA II Port Multiplier advanced support
- SATA II Port Selector control
  (Generates the protocol-based Out of Band (OOB) sequence to select the active host of the SATA II Port Selector.)
- Compliant with SATA II Phase 1 specifications
  - Supports SATA II Native Command Queuing (NCQ), up to 128 outstanding commands per port
  - First party DMA (FPDMA) full support
  - Backwards compatible with SATA I devices
- Supports SATA II Phase 2 advanced features
  - 3 Gbps (Gen2i) SATA II speed
  - Port Multiplier (PM)
    Performs FIS-based switching as defined in SATA working group PM definition
  - Port Selector (PS)
    Issues the protocol-based Out-Of-Band (OOB) sequence to select the active host port
- Supports device 48-bit addressing
- Supports ATA Tag Command Queuing
- Enhanced-DMA [EDMA] for the SATA port
  - Automatic command execution without host intervention
  - Command queuing support up to 128 outstanding commands

  - Separate SATA request/response queues
  - 64-bit addressing support for descriptors and data buffers in system memory
- Read ahead
- Advanced interrupt coalescing
- Advanced drive diagnostics via the ATA SMART command

For further information about the SATA interface, see Section 21, Serial-ATA (SATA) II Interface, on page 354.

2.17 Serial Peripheral Interface (SPI)

The device integrates two general-purpose Serial Peripheral Interface (SPI) ports. The SPI is a synchronous serial data protocol used for transferring data simply and quickly from one device to another.

The device supports a boot sequence from an SPI flash. A different usage for the SPI interface can be managing the SLIC/SLACs used for voice sampling. The device provides 8 physical ranks (chip selects).

For further information about the SPI interface, see Section 22, Serial Peripheral Interface (SPI), on page 378.

2.18 Time-Division Multiplexing (TDM) Controller

The device includes a Time-Division Multiplexing (TDM) interface (a.k.a Multi Channel TDM—TDMMC).

The TDM is a generic interface for standard SLIC/SLAC/Codec devices. It is compatible with standard PCM highway formats, supports 32 channels, and up to 128 time slots. The unit’s flexible configuration allows each of the 128 slots to be assigned to any of the 32 channels or be configured as inactive. It also has dedicated Rx and Tx DMAs per channel that are used to transfer the voice samples from/to the interface, or to/from memory, while minimizing CPU intervention.

The unit supports compound (A-law, U-law), linear or wideband voice samples.

It can also be configured to generate or receive the clock and frame sync signals independently. It can also use different clock polarities and frame sync while sampling/driving the data.

SLIC/SLAC/codec registers read/write access is done through the device’s SPI interface which offers up to eight physical ranks to lower access and search time during interrupt or access events.

Each of the TDM channels can be also used in High-Level Data Link Control (HDLC) Bit Oriented protocol mode. In the HDLC mode, the unit offloads the CPU from HDLC link management by performing flag generation and stripping, bit stuffing and stripping, address recognition (8/16-bits addresses), CRC generation and checking, and line condition monitoring and so on.

For further information about the TDM interface see Section 23, Time-Division Multiplexing (TDM) Controller, on page 386.

2.19 Secure Digital Input/Output (SDIO) Controller

The device integrates an SD/SDIO/MMC host controller.

This controller functions as a host for the SD/MMC bus to transfer data through between SDMem, SDIO, and MMC cards on one side and the internal system buffers on the other side. Dedicated DMA is used to do the data transfers.

The controller supports:

- 1-bit/4-bit SDMem, SDIO, and MMC cards
- Hardware generate/check CRC on all command and data transactions on the card bus

- SD Host 1.0
- SD PHY 1.1 up to 50 MHz

For further information about the SDIO interface, see Section 24, Secure Digital Input/Output (SDIO) Controller, on page 413.

2.20 Inter-Integrated Circuit Interface (I^2C)

The device integrates two generic I^2C ports that can be configured as a master or a slave interface. One of the ports can also be used for serial ROM initialization.

The I^2C interface fully supports multiple I^2C master environments (clock synchronization, bus arbitration). The I^2C interface can be used for multiple applications such as:

- A master to control other I^2C based on-board devices
- To read the DIMM SPD ROM
- To auto-load values from an external serial ROM device.

The I^2C interface can also be used as a slave for communication with other on-board I^2C masters.

For further information about the I^2C interface, see Section 25, Inter-Integrated Circuit Interface (I^2C), on page 420.

2.21 UART Interface

The device integrates four 16750-compatible Universal Asynchronous Receiver/Transmitter (UART) ports. Each port has 2 pins for transmit and receive operations and 2 pins for modem control functions.

One port also supports an integrated transmit DMA, capable of up to 64-KB transfer.

For further information about the UART interface, see Section 26, Universal Asynchronous Receiver/Transmitter (UART) Interface, on page 432.

2.22 Real Time Clock (RTC)

The device integrates an RTC unit that records the second, minute, hour, date, day, month, and year.

When the system power is off, a backup battery (3V) can keep the RTC unit operational.

The RTC unit operates with an external 32.768 kHz crystal input.

It also supports driving an alarm signal to the external power regulator for restoring power to the system at a specific time even when device power is down.

For further information about the RTC interface, see Section 27, Real-Time Clock (RTC), on page 439.

2.23 General Purpose I/Os (GPIO) Ports

The device includes general purpose input/output pins, as follows, multiplexed on Multiple Purpose Pins (MPP):

MV78260 **MV78460**	67 pins
MV78230	49 pins

Each MPP can be assigned to act as a general purpose input or output pin and can be used to register external edge or level sensitive interrupts, when assigned as an input pin.

The reset strap of the device is configured by a subset of the MPPs. To reduce design complexity, MPPs which are part of the reset strap subset should not be configured as general purpose input unless the state of the pin is guaranteed to be compatible with the reset strap at reset. They can be configured as general purpose output.

For further information about the general purpose I/Os, see Section 28, General Purpose Input/Output Ports, on page 444.

2.24 Power Management

With the increasing requirement for greener environment and the world wide governmental energy save regulations, the MV78230/78x60 includes extensive power management and power reduction options for controlling and limiting the device and system power, assisting overall power reduction for meeting these tight low power requirements.

Instead of running in a full operational Run mode, the CPUs and their sub-system in the device may operate in a power save mode. For example, the throttle power save mode means that the CPU speed is reduced to match the DRAM interface speed, to save some dynamic power. In the retained state Idle mode, the CPU clocks are gated to save all of the dynamic power. In Deep Idle mode, the CPU cores and their sub-systems are completely powered down to save the dynamic and static power.

The ARMADA® XP provides the infrastructure to maintain system coherency while entering into one of these power save modes, and allows for a very fast recovery time when maintaining full operational mode is needed. As the device implements internal power switches to control the CPU power domain, power restore and recovery time from Deep Idle state is very fast as it does not involve switching the on board power regulator on (it remains on at all times).

The key power save options provided by the devices include:

- CPU Idle (dynamic clock gating)
- L2 Idle (dynamic clock gating)
- CPU Deep Idle (CPU and L2 are powered down, context is stored in the DRAM and coherent I/O transactions are routed directly to the DRAM)
- DRAM self refresh and power down modes
- EEE (Energy Efficient Ethernet) and LPI (Low Power Idle) support
- SERDES power down options
- USB interface placed in suspend mode
- Shutdown or clock gating of non-used units and interfaces
- PCIe power save states

The device also provides many wake-up options from the power save modes. These include:

- An extended Wake On LAN (WOL) definition
- Wake on USB
- Wake by a System Timer / RTC Alarm
- Wake on GPIO interrupt

The internal power-sensitive structure of the device, the advanced 40 nm process technology, and the power save options yields a very power-aware device with ultra-low power consumption for its class.

For further information about the Power Management capabilities, see Section 34, Power Management, on page 524.

Copyright © 2014 Marvell

2.25 High-Speed Integrated SERDES Interface

MV78460	Integrates 16 high-speed SERDES lanes
MV78260	Integrates 12 high-speed SERDES lanes.
MV78230	Integrates 7 high-speed SERDES lanes.

The SERDES lanes provide a physical SERDES link to the following interfaces:

- PCI Express Gen2.0 up to 5 Gbps, and Gen1.1 up to 2.5 Gbps operational modes
- SGMII (both 1.25 Gbps and 3.125 Gbps)
- QSGMII (up to 5 Gbps)
- SATA in both Gen1 (1.5 Gbps) and Gen2 (3 Gbps)
- sETM

The SERDES lanes multiplex options are listed in the Hardware Specifications of this device. Each lane can be configured independently for the required link type, according to the specified application. If a lane is unused, it can be turned off.

For more information about the SERDES configuration flows, see Section 29, High-Speed Integrated SERDES Interface, on page 449.

2.26 Cryptographic Engines and Security Accelerators (CESA)

To support data encryption and authentication, the device integrates 2 Cryptographic Engines and Security Accelerators (CESA) that are useful for offloading networking security protocols, such as IPSec, and for storage encryption and de-duplication.

In Enhanced Software Flow mode, the CESA reduces the CPU overhead of IPSec intensive cryptographic algorithms by performing complete, descriptor based, IPSec packet processing (encryption and authentication) on the packet buffer without software intervention.

The Cryptographic Engine supports:

- AES, DES, and 3DES encryption algorithms
- SHA1, SHA2, and MD5 authentication algorithms

Dedicated DMAs are used to feed the hardware engines.

For further information about the Cryptographic engines, see Section 30, Cryptographic Engines and Security Accelerators (CESA), on page 457.

2.27 XOR Engines

The device incorporates 4 XOR DMA engines, that are useful for Redundant Array of Independent Disks (RAID) applications. The XOR DMA engines can also be used for normal DMA operation.

Each XOR DMA runs on a linked list of descriptors. It can read from up to eight sources, perform bitwise XOR between the eight sources, and write the result to a destination.

Each DMA can also be configured to:

- Initialize memory (Efficiently zero a section in memory)
- Clean single bit ECC errors in DRAM (memory ECC errors cleanup operation—memory scrubbing)
- Calculate iCSCI CRC32 on a data buffer
- Act as a high-performance general purpose DMA engine

For further information about the XOR engines, see Section 31, XOR Engines, on page 492.

2.28 Independent DMA (IDMA) Controller

The device incorporates an Independent DMA (IDMA) controller with four IDMA engines. Each IDMA engine can transfer data between any interface.

The DMA can run in Chain mode, via a linked list of descriptors, and transfer up to 16 MB per descriptor. The DMA independently supports increment/hold on source and destination addresses.

For further information about the IDMA engines, see Section 32, Independent DMA (IDMA) Controller, on page 511.

3 Address Map

The ARMADA® XP has a fully programmable address map. There is separate address map for each of the master units (units that can initiate transactions over the device Mbus). Each of the Mbus masters described below includes programmable address windows that enables it to access the different device resources.

The Mbus masters are:

- Marvell® Core Processor(s)
- PCI Express port x
- GbEx MAC
- USBx MAC
- SATA MAC
- Security accelerator
- IDMA channels 0–3
- XOR DMA channels 0-3
- Buffer Management unit
- SDIO
- TDM
- I^{2}C Serial Initialization

Note

Throughout this section, the term BAR means Base Address Register.

3.1 Marvell® Core Processor Address Decoding

The Marvell® Core Processor address decoding map consists of 29 configurable address windows that include:

- 4 windows dedicated for accessing the SRAM in case part of the L2 is configured as such.
- 4 windows dedicated for core access to the 4 DRAM chip selects (CS)
- 1 window for core access to the device internal registers space
- 1 window dedicated for core accesses to the device resources.[1]
- 20 windows for core access to the rest of the device's resources

Note

The address decoding from the original address to the final destination uses the following order:

1. Internal register
2. SRAM (in case part of the L2 was configured as SRAM)
3. Device resources (Mbus Bridge Window)
4. DRAM
5. Other device resources (I/O mapping)

[1] The MBus Bridge Window overrides the window that is defined by the DRAM CS window.

Each of the four SRAM windows have a minimum of 64 KB of address space and up to 512 KB, in 64 KB resolution. The SRAM windows may override any of the other CPU address windows. These windows are only relevant if part/all the L2 is defined as SRAM. In case there is no SRAM assignment, these windows need to be disabled.

The device supports up to 16 GB of DRAM memory space.

Each DRAM window can have an address space of between 16 MB and 4 GB. Each window is defined by a Base and Size registers. The base and size are 8-bits wide, corresponding to address bits[31:24]. The size must be programmed as a set of 1s (starting from the LSB) followed by a set of 0s. The set of 1s defines the size. For example, if Size[7:0] is set to 0x0F, it defines a size of 256 MB (number of 1s is 4, 2^4 x 16 MB = 256 MB). In addition, each window has additional bits on its base register for supporting Large Physical Address Extension (LPAE). These bits are compared to the high bits of the address (bits[35:32] of the address) for exact match with no masking option.

Note

- Different LPAE extensions to the same CS are not allowed.
- An SRAM address above 4 GB must hit an SDRAM CS window.
- An SRAM address below 4 GB may miss all the SDRAM CS window.

By default, each of the DRAM windows corresponds to a different DRAM chip select (M_CSn[3:0]). However, each window can be set to support any of the DRAM chip selects. This feature provides more flexibility in mapping DRAM space.

Note

- The CPU can also be configured to access the DRAM through the Mbus as any other interface in the device, instead of through the four dedicated DRAM windows.
- Do *not* overlap the CPU’s DRAM dedicated windows with CPU windows that access DRAM through the Mbus.

Apart from the DRAM and SRAM windows, each of the configurable address windows can have a minimum of 64 KB of address space and up to 4 GB. Each window is defined by a Window Base register and by the Window Control register’s <Size> field. The Base and Size are 16 bits wide, corresponding to address bits[31:16]. The Size must be programmed as a set of 1s (starting from the LSB) followed by a set of 0s. The set of 1s defines the size. For example, if Size[15:0] is set to 0x03FF, it defines a size of 64 MB (number of 1s is 10, 2^{10} x 64 KB = 64 MB)

The device’s internal registers window has a 1 MB fixed size (has Base register only, no Size register).

Address decoding starts with the address being compared with the values in the various Base Address registers. The Size sets the address bits that are significant for the comparison. In the previous example of a 64 MB size, the CPU address bits[31:26] are compared with the Base Address bits[15:10] (the Size masks address bits[25:0]). An address is considered as a window hit if it matches the Base Address register bits (the bits that are not masked by the Size).

Note

- Never program the Base and Size registers so that they result in an address windows overlap.
- The address decoding scheme restricts the address window to a size of 2^n and to a start address aligned to the window size.

Upon a hit in one of the 21 configurable windows, the transaction is forwarded to a specific target interface specified by the Window Control register’s <Target> bits[7:4], and with specific transaction attributes specified by the Window Control register’s configurable <Attr> bits [15:8].

Table 6 is a summary of target units IDs and attributes. It is relevant for access from the CPU core only.

Table 6: CPU Interface Mbus Decoding Units IDs and Attributes

Field	Description
Target Unit ID	0x0 = DRAM 0x1 = Device bus, BootROM, SPI, and DMA-based UART 0x2 = Reserved 0x3 = Ethernet 1 (Ports 2 and 3) 0x4 = PCI Express port 0 and port 2 0x5–0x6 = Reserved 0x7 = Ethernet 0 (Ports 0 and 1) 0x8 = PCI Express port 1 and port 3 0x9 = Security accelerator SRAM 0xA–0xB = Reserved 0xC = Buffer Management unit 0xD = NAND Flash Controller v2.0 0xE–0xF = Reserved

Table 6: CPU Interface Mbus Decoding Units IDs and Attributes (Continued)

Field	Description
Attributes[7:0]	**If the target is the DRAM interface:** Bits[7:0] = Memory chip select: 0x0E = M_CS[0] request in DRAM controller 0x0D = M_CS[1] request in DRAM controller 0x0B = M_CS[2] request in DRAM controller 0x07 = M_CS[3] request in DRAM controller All other values are reserved. **If the target is the Device bus interface:** Bits[7:0] = chip select: 0x1 = DMA based UART request 0x1E = SPI0 CS0 request 0x5E = SPI0 CS1 request 0x9E = SPI0 CS2 request 0xDE = SPI0 CS3 request 0x1F = SPI0 CS4 request 0x5F = SPI0 CS5 request 0x9F = SPI0 CS6 request 0xDF = SPI0 CS7 request 0x1A = SPI1 CS0 request 0x5A = SPI1 CS1 request 0x9A = SPI1 CS2 request 0xDA = SPI1 CS3 request 0x1B = SPI1 CS4 request 0x5B = SPI1 CS5 request 0x9B = SPI1 CS6 request 0xDB = SPI1 CS7 request 0x3E =- DevCS[0] request 0x3D = DevCS[1] request 0x3B = DevCS[2] request 0x37 = DevCS[3] request 0x2F = BootCS request 0x1D = BootROM request All other values are reserved. **If the target is the NAND Flash Controller v2.0 interface:** Bits[7:0] = Must be 0x20 **NOTE:** To enable NAND external access, set to 1 the <NfAddressMaskEn> **field in the SoC Device Multiplex Register (Table 1433 p. 1513)**. **NOTE:** When addressing SPI and working in Single Window mode, configure these bits to (Section 22.5.3, SPI Timing, on page 384): 0x1E for addressing SPI0 CS 0-7 0x1A for addressing SPI1 CS 0-7 When addressing SPI and working in Double Window mode configure these bits to: 0x1E for addressing SPI0 CS0-3 0x1F for addressing SPI0 CS4-7 0x1A for addressing SPI1 CS0-3 0x1B for addressing SPI1 CS4-7

Copyright © 2014 Marvell

Table 6: CPU Interface Mbus Decoding Units IDs and Attributes (Continued)

Field	Description
Attributes[7:0] (Continued)	**If the target is the PCI Express interface:** Bits[2:0] = Reserved. Must be 0x0. Bit[3] = Memory/IO select: 0x0 = I/O 0x1 = Memory Bits[7:4] = PCIE lane select (Any other value is reserved): 0xF = PCIe portX, Lane0 (X=2,3) 0xE = PCIe portX, Lane0 (X=0,1) 0xD = PCIe portX, Lane1 (X=0,1) 0xB = PCIe portX, Lane2 (X=0,1) 0x7 = PCIe portX, Lane3 (X=0,1) While accessing a port that is configured as x4, Lane0 must be accessed. **If the target is the Security Accelerator SRAM:** Bits[1:0] = Data swapping 0x0 = Byte swap 0x1 = No swap 0x2= Byte and word swap 0x3 = Word swap Bit[3:2] = Engine select: 0x2 = Engine0 0x1 = Engine1 Bits[7:4] = Reserved. Must be 0. **If accessing any other target unit set bits[7:0] to 0x00.**

Each of the 4 DRAM windows and the 21 configurable windows has an Enable bit (Size register's <En> bit[0]). When set to 1, the window is enabled. When cleared to 0, the window is disabled and does not take part in the address decoding process. A CPU address that is going through the address decoding scheme is 34-bits wide.

Table 7 shows the default Marvell® Core Processor memory map following reset de-assertion.

Table 7: Marvell® Core Processor Default Address Map

Decoder	Address Range	Target Unit ID	Target Attribute
SRAM Window 0	**NOTE:** Disabled in default.	NA	NA
SRAM Window 1	**NOTE:** Disabled in default.	NA	NA
SRAM Window 2	**NOTE:** Disabled in default.	NA	NA
SRAM Window 3	**NOTE:** Disabled in default.	NA	NA
DRAM M_CSn[0]	0x0 to 0x0.0FFF.FFFF 256 MB **NOTE:** Disabled in default.	NA	NA
DRAM M_CSn[1]	0x0.1000.0000 to 0x0.1FFF.FFFF 256 MB **NOTE:** Disabled in default.	NA	NA

Table 7: Marvell® Core Processor Default Address Map (Continued)

Decoder	Address Range	Target Unit ID	Target Attribute
DRAM M_CSn[2]	0x0.2000.0000 to 0x0.2FFF.FFFF 256 MB **NOTE:** Disabled in default.	NA	NA
DRAM M_CSn[3]	0x0.3000.0000 to 0x0.3FFF.FFFF 256 MB **NOTE:** Disabled in default.	NA	NA
Reserved	0x4000.0000 to 0x7FFF.FFFF	NA	NA
Configurable Window 0	0x8000.0000 to 0x87FF.FFFF 128 MB	0x4 (PCIe0)	0xE8 (Memory, port 0.0)
Configurable Window 1	0x8800.0000 to 0x8FFF.FFFF 128 MB	0x4 (PCIe0)	0xE8 (Memory, port 0.0)
Configurable Window 2	0x9000.0000 to 0x97FF.FFFF 128 MB	0x4 (PCIe0)	0xE8 (Memory, port 0.0)
Configurable Window 3	0x9800.0000 to 0x9FFF.FFFF 128 MB	0x4 (PCIe0)	0xE8 (Memory, port 0.0)
Configurable Window 4	0xA000.0000 to 0xA7FF.FFFF 128 MB	0x8 (PCIe1)	0xE8 (Memory, port 1.0)
Configurable Window 5	**NOTE:** Disabled in default.	NA	NA
Configurable Window 6	**NOTE:** Disabled in default.	NA	NA
Configurable Window 7	**NOTE:** Disabled in default.	NA	NA
Reserved	0xC000.0000 to 0xC800.FFFF	NA	NA
Configurable Window 8	0xC801.0000 to 0xC801.FFFF 64KB	0x9 (Security Accelerator SRAM)	0x09 (Engine0, no data swap)
Reserved	0xC802.0000 to 0xCFFF.FFFF	NA	NA
Internal Registers	0xD000.0000 to 0xD00F.FFFF 1 MB	NA	NA
Reserved	0xD030.0000 to 0xD37F.FFFF	NA	NA
Configurable Window 9	0xD800.0000 to 0xDFFF.FFFF 128 MB	0x1 (Device bus)	0x2F (BOOT_CS)
Configurable Window 10	0xE000.0000 to 0xE7FF.FFFF 128 MB	0x1 (Device bus)	0x3E (DEV_CS[0])
Configurable Window 11	0xE800.0000 to 0xEFFF.FFFF 128 MB	0x1 (Device bus)	0x3D (DEV_CS[1])
Configurable Window 12	0xF000.0000 to 0xF7FF.FFFF 128 MB	0x1 (Device bus)	0x3B (DEV_CS[2])

Table 7: Marvell® Core Processor Default Address Map (Continued)

Decoder	Address Range	Target Unit ID	Target Attribute
Configurable Window 13	0xF800.0000 to 0xFFFF.FFFF 128 MB	0x1 (Device bus)	According to boot device type: 0x2F (= DevBOOTCS) or 0x1E (= SPI0 ROM) or 0x1D (= BootROM)
Configurable Window 14	0xD400.0000 to 0xD7FF.FFFF 64MB	0x1 (Device bus)	0x1E (SPI0 CS[0])
Configurable Window 15	0xD030.0000 to 0xD030.FFFF 64KB **NOTE:** Disabled in default	NA	NA
Configurable Window 16	0xD031.0000 to 0xD031.FFFF 64KB **NOTE:** Disabled in default	NA	NA
Configurable Window 17	0xD032.0000 to 0xD032.FFFF 64KB **NOTE:** Disabled in default	NA	NA
Configurable Window 18	0xD380.0000 to 0xD3FF.FFFF 8MB	0xC (Buffer Management unit)	0x00
Configurable Window 19	0x0000.0000 to 0x07FF.FFFF 128MB	0x0 (DRAM)	0x0E (M_CS[0])

The device’s internal registers space has a fixed size of 1 MB, even though only part of this space is actually populated by the device’s internal registers. Upon a write to a non-implemented register, data is discarded. A read to a non-implemented register returns undefined data.

The registers are spread around the device’s different units (distributed register file). Therefore, ordering is not guaranteed upon a CPU back-to-back write to different registers. If ordering is required, perform a read after each write.

Note

Access to internal registers is limited to WORD boundary (no support of burst access to reg files). This means that register file space must never be cacheable.

3.1.1 Marvell® Core Processor-to-PCI Express Address Remapping

The device supports address remapping on Marvell® Core Processor accesses to the PCI Express interface. This enables relocating a CPU-to-PCI Express address window to a new location in the PCI Express address space, de-coupling the core and the PCI Express memory allocation.

Configurable windows 7–0 have associated Remap registers. Upon a hit in one of these windows, the upper bits of the CPU address are replaced by the corresponding bits of the Remap Low register, before being transferred to the PCI Express interface unit. The number of bits to be replaced is determined by the Size register. For example, with a 64 MB window, the CPU address bits[31:26] are replaced with bits[31:26] of the Remap Low register.

Each of these windows also has a 32-bit Remap High register, which can be used for 64-bit addressing on the PCI Express interface. When this register is not cleared to 0, a CPU address hit in this window results in the device's PCI Express master generating a 64-bit addressing transaction.

Note

Address window 13 has remap registers as well. They can be useful to allow boot from a 32b address SPI flash.

3.1.2 Accessing Buffer Management Controller from the Marvell® Core Processor

One of the CPU address windows may be used to access the buffer management controller. While hitting this window, an additional sub-decoding is performed on the incoming address:

- ADDR[17] = 0: Buffer management controller is accessed.

3.1.3 CPU Address Decoding Errors

In the case of an address decoding error (for example, no hit in any of the address windows):

1. An interrupt is set.
2. If it is a write transaction, it is discarded. If it is a read, dummy data is driven back to the Marvell® Core Processor with an erroneous data indication (resulting in an CPU external abort exception).

3.2 PCI Express Address Decoding

Each of the device's PCI Express ports has its own address map. The PCI Express interface address map consists of three Base Address Registers (BARs) that map the device address space. One BAR is dedicated to the device's internal registers space. The other two BARs are further sub-decoded by six programmable address windows to the different interfaces of the device.

The three BARs are 64-bit wide BARs. The internal registers space has a fixed size of 1 MB. The other two BARs have corresponding size registers. Each window can have a minimum of 64 KB of address space and up to 4 GB of space.

PCI Express address decoding is similar to the CPU address decoding scheme. An address is considered as a window hit if it matches the Base Address register bits. These are the bits not masked by the Size register.

Copyright © 2014 Marvell

Note

- Do not program the Base and Size registers so that they result in an address window overlap.
- The PCI Express address decoding scheme restricts the address window to a size of 2^n, and to a start address that is aligned to the window size.

Each of the two BARs has an Enable bit. If a BAR is disabled, no address decoding is performed against it.

Upon an address hit, the address is further sub-decoded against six address windows. Each of these windows also consists of Base and Size registers. Based on this address decoding, the transaction is forwarded to a specific target interface (e.g. DRAM controller), with specific transaction attributes (e.g., M_CS[0]). Table 8 is a summary of target units IDs and attributes.

Table 8: Units IDs and Attributes

Field	Description
Unit ID	0x0 = DRAM 0x1 = Device bus, BootROM, SPI, and DMA-based UART 0x2 = Reserved 0x3 = Ethernet 1 (Ports 2 and 3)–packet modification tables 0x4 = PCI Express port 0 and port 2 0x5–0x6 = Reserved 0x7 = Ethernet 0 (Ports 0 and 1)–packet modification tables 0x8 = PCI Express port 1 and port 3 0x9 = Security accelerator SRAM 0xA–0xB = Reserved 0xC = Buffer Management unit 0xD = NAND Flash Controller v2.0 0xE–0xF = Reserved

Table 8: Units IDs and Attributes (Continued)

Field	Description
Attributes[7:0]	**If the target is the DRAM bus interface:** Bits[3:0] = Memory chip select: 0xE = M_CS[0] request in DRAM controller 0xD = M_CS[1] request in DRAM controller 0xB = M_CS[2] request in DRAM controller 0x7 = M_CS[3] request in DRAM controller All other values are reserved. Bits[5:4] = Shared Memory Area With CPU (Coherency Status). Shared area forces sending the transaction through the coherency block. A Non-Shared transaction is driven directly to DRAM. 0x0 = Non-Shared transaction 0x1 = Shared transaction and no L2 read/write allocate (no L2 Deposit) 0x2 = Reserved 0x3 = Shared transaction + L2 write allocate (L2 Deposit) Bits[7:6] = IO Snoop Affinity group (meaningful only if bit[4] is 0x1): 0x0 = Originator belongs to IO snoop group 0. 0x1 = Originator belongs to IO snoop group 1. 0x2 = Originator belongs to IO snoop group 2. 0x3 = Originator belongs to IO snoop group 3. For further details about IO snoop affinity refer to Section 9.6.3, I/O Snoop Affinity , on page 154. **If the target is the Device bus interface:** Bits[7:0] = chip select: 0x1 = DMA based UART request 0x1E = SPI0 CS0 request 0x5E = SPI0 CS1 request 0x9E = SPI0 CS2 request 0xDE = SPI0 CS3 request 0x1F = SPI0 CS4 request 0x5F = SPI0 CS5 request 0x9F = SPI0 CS6 request 0xDF = SPI0 CS7 request 0x1A = SPI1 CS0 request 0x5A = SPI1 CS1 request 0x9A = SPI1 CS2 request 0xDA = SPI1 CS3 request 0x1B = SPI1 CS4 request 0x5B = SPI1 CS5 request 0x9B = SPI1 CS6 request 0xDB = SPI1 CS7 request 0x3E = DevCS[0] request 0x3D = DevCS[1] request 0x3B = DevCS[2] request 0x37 = DevCS[3] request 0x2F = BootCS request 0x1D = BootROM request All other values are reserved.

Table 8: Units IDs and Attributes (Continued)

Field	Description
Attributes[7:0] (Continued)	**If the target is PCI Express interface:** Bits[2:0] = Reserved. Must be 0x0. Bit[3] = Memory/IO select: 0x0 = I/O 0x1 = Memory Bits[7:4] = PCIe lane select (Any other value is reserved): 0xF = PCIe portX, Lane0 (X=2,3) 0xE= PCIe portX, Lane0 (X=0,1) 0xD = PCIe portX, Lane1 (X=0,1) 0xB = PCIe portX, Lane2 (X=0,1) 0x7 = PCIe portX, Lane3 (X=0,1) While accessing a port that is configured as x4, Lane0 must be accessed. **If the target is the Security Accelerator SRAM:** Bits[1:0] = Data swapping 0x0 = Byte swap 0x1 = No swap 0x2 = Byte and word swap 0x3 = Word swap Bit[3:2] Engine select: 0x2 = Engine0 0x1 = Engine1 Bits[7:4] = Reserved. Must be 0. **If accessing any other target unit, set bits[7:0] to 0x00.**

3.2.1 PCI Express to Memory Address Remapping

The device supports PCI Express address remapping. Each of the 6 PCI Express address windows has a Remap Register associated with it. The upper bits of the address, that are found to be a hit in one of the PCI windows, are replaced by the corresponding bits of the Remap register before being transferred to the target interface. The number of bits to be replaced is determined according to the Size register.

Windows 4 and 5 also have Remap High registers. This is useful for 64-bit addressing if the target interface supports 64-bit addressing (for example, PCI Express to PCI Express bridging).

Each Remap register has an Enable bit. If it is disabled, no remap action takes place and the original address is transferred to the destination.

3.2.2 PCI Express Address Decoding Errors

If the device's PCI Express port receives a transaction that does not match any of the BARs:

1. An interrupt is set and the error is registered.
2. The transaction is terminated as an unsupported request.

If the device's PCI Express port receives a transaction that does hit one of the BARs, but does not match any of the sub-decoding windows:

1. An interrupt is set and the error is registered.
2. The transaction is forwarded to a default target, as defined in the PCI Express Default Window Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 862 p. 1156).

Note

Set the PCI Express Default Window Control Register (x=0–0, x=0–3, y=2–3, y=0–1) to point to a dummy target device, so that no destructive operation is performed due to the address decoding error.

3.2.3 Accessing Buffer Management Controller from the PCI Express Interface

One of the PCIe address windows may be used to access the buffer management controller. While hitting this window, an additional sub-decoding is performed on the incoming address:

To access the buffer management controller, address bit[17] has to be 0.

3.3 IDMA Address Decoding

The four IDMA channels share a single address decoding logic, which consists of eight address windows, each defined by a Base and Size register. Each window can map a space of 64 KB, up to 4 GB. Each of the eight windows can be configured to a specific target interface and to specific transaction attributes, as shown in Table 8.

The IDMA channels also support an address override capability. This feature is useful for decoupling the memory and PCI Express address space. When address override is enabled, the IDMA address is not compared against the eight windows. Instead, the transaction is forwarded to a pre-defined target interface. This feature makes it possible to set the same Source and Destination Address, and still make the source be the DRAM and the destination be the PCI Express.

Four out of the eight windows also have a Remap High register. Use these registers to generate an address beyond the standard 4 GB space. This is useful for 64-bit PCI Express addressing or for a memory space which is larger than 4 GB.

When an IDMA channel attempts to access an unmapped address, an interrupt is set, the error status is registered, and the channel halts.

3.4 XOR DMA Address Decoding

The two XOR DMAs share a single address decoding logic consisting of eight address windows. Whenever one of the ports generates a read or a write transaction (e.g., fetch descriptor), the address is compared against these address windows, to determine which interface must be accessed.

The address decoding scheme is shown in Table 8.

3.5 Ethernet Networking Controllers Address Decoding

The Ethernet networking controllers address decoding logic consists of 6 address windows. Whenever the port’s SDMA generates a read or a write transaction (for example, fetch descriptor), the address is compared against these address windows to determine which interface must be accessed.

The address decoding scheme is the same as the IDMA logic with one exception. In the case of an address miss match, the SDMA transaction is retargeted to a fixed address and the target interface, as defined in the Ethernet Unit Default Address (EUDA) Register (i=0–3) (Table 577 p. 923).

Note

The GbE port is restricted to access only the DRAM interface (M_CS[3:0]), or the Buffer Management unit. Setting Ethernet port address decoding windows differently results in unpredictable behavior.

The address decoding scheme is shown in Table 8.

3.6 USB Address Decoding

Each USB port uses an address decoding logic consisting of four address windows. Whenever one of the ports generates a read or a write transaction (for example, fetch descriptor), the address is compared against these address windows, to determine which interface must be accessed.

Note

The USB port is restricted to access only the DRAM interface (M_CS[3:0]). Setting USB port address decoding windows differently results in unpredictable behavior.

The address decoding scheme is shown in Table 8.

3.7 SATA Address Decoding

The SATA port DMA uses an address decoding logic consisting of four address windows. The address decoding scheme is shown in Table 8.

Note

The SATA port is restricted to access only the DRAM interface (M_CS[3:0]). Setting SATA port address decoding windows differently results in unpredictable behavior.

3.8 Security Accelerator Address Decoding

The Security Accelerator DMA uses an address decoding logic consisting of four address windows. The address decoding scheme is shown in Table 8.

Note

The Security accelerator DMA is restricted to access only the DRAM interface (M_CS[3:0]). Setting security accelerator DMA address decoding windows differently results in unpredictable behavior.

3.9 Buffer Management Address Decoding

Each pool in the buffer management unit can be configured to work with a different area in memory, through configurable target destination and attributes. For example, pool0 may be configured to work with M_CS[0] by configuring to 0x0 and to 0xE. Similarly, pool2 may be configured to with M_CS[1] by setting to 0x0 and to 0xD.
The address decoding scheme is shown in Table 8.

Note

- The Buffer management Unit is restricted to access only the DRAM interface (M_CS[3:0]). Setting Buffer Management pools attributes differently results in unpredictable behavior.
- A single buffer pointers pool cannot not reside in two different DRAM chip selects.
- It is forbidden to configure the DRAM memory accesses of the buffer management as “Shared”, i.e bit[4] of the Mbus transaction attribute must be cleared to 0x0, as shown in Table 8.

3.10 SDIO Address Decoding

The SDIO unit uses an address decoding logic consisting of 4 address windows. The address decoding scheme is shown in Table 8.

Note

The SDIO unit is restricted to access only the DRAM interface (M_CS[3:0]). Setting SDIO unit’s address decoding windows differently results in unpredictable behavior.

3.11 TDM Address Decoding

The TDM unit uses an address decoding logic consisting of 12 address windows. The address decoding scheme is shown in Table 8.

Note

The TDM unit is restricted to access only the DRAM interface (M_CS[3:0]). Setting TDM unit’s address decoding windows differently results in unpredictable behavior.

3.12 I^{2}C Address Decoding

The device’s I^{2}C serial ROM initialization allows for access to the entire device’s internal resources.

The serial ROM initialization access to the device resources consists of a 32-bit address followed by 32-bit data. Bit[0] of the transaction address must be cleared to 0 for register access.

Copyright © 2014 Marvell

3.13 Accessing the L2/SRAM from Mbus Masters

The device provides the capability for any of the Mbus IO masters to store/load data directly from the L2/SRAM. This is useful in order to bring data closer to the CPU and reduce the latency for CPU access to this data (for example, storing Ethernet descriptor headers).

L2/SRAM access from an IO master is configured through setting attribute bits[5:4] of the relevant IO master address decoding window to 0x3 as listed in Table 8. This setting will force an IO transaction that hits this address window to access the L2 cache as follows:

- Full cache line writes will be stored in the L2 cache while evicting existing line in case of L2-miss, or override existing data in case of L2-hit.
- Partial write accesses will only be written into the L2/SRAM hitting an existing valid line. If line is not valid in the L2 cache, it will be forwarded to DRAM.
- An IO read access will get its data back directly from the L2/SRAM in case of hitting a valid line, or from the DRAM in case of a miss.

While accessing the SRAM from one of the IO masters, the user may choose to map the SRAM as part of the DRAM address space. In this case, no need to define a special SRAM address window in the IO master, but instead use the DRAM address window where the SRAM address range is mapped, and set for this window the window attribute bits[5:4] to 0x3 (Shared transaction + L2 write allocate (L2 Deposit)) as listed in Table 8. The transaction will be directed from the IO master to the Coherency Fabric where it will be assigned internally to the SRAM according to the CPU dedicated SRAM address windows. See Section 3.1, Marvell® Core Processor Address Decoding for more details on how the SRAM windows are mapped by the CPU core.

Note

In case of hitting an SRAM window, write data will be written to the SRAM regardless of its size.

Transactions that hit this IO master DRAM window, but are not mapped to SRAM, are forwarded to the DRAM by the Coherency Fabric.

For example, the SRAM is mapped in the system as base address 0x0 with a size of 128 KB, and DRAM CS[0] is mapped by one of the IO master address decoding windows as 256 MB starting at address 0x0.

Setting the attribute bits [5:4] of this window to 0x3 will send every transaction in the address range between 0x0 to 0x0FFF.FFFF to the Coherency Fabric, while only transactions with an address range of 0x0 to 0x0001.FFFF will access the SRAM. Other transactions are forwarded to DRAM.

In cases where the user chooses not to map the SRAM as part of the DRAM address map, rather as a separate area, a dedicated address window has to be assigned for accessing the SRAM in the IO master address window registers. The register setting of the target unit should be “DRAM” (0x0) and attribute bits[5:4] should be set to 0x3 (attribute bits [3:0] may be set to any of the allowed values according to Table 8).

4 Internal Architecture

The ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors implements a high-bandwidth, high-speed, and low-latency coherent interconnect between the CPU cores, shared L2 cache, and memory interface. Dedicated ports allow the IO masters to issue coherent requests and maintain coherency with the CPU on shared memory areas.

The devices' I/O sub-system internal architecture is based on a 64-bit full duplex data path connecting the different units in the Mbus, a Marvell® proprietary crossbar interconnect, optimized for high performance and high bandwidth traffic.

The advanced power management enables the device to power down any CPU, L2, and Coherency Fabric in the Deep Idle low power state. It supports flexible clock domains with each domain using its own optimal clock configuration (see Table 9).

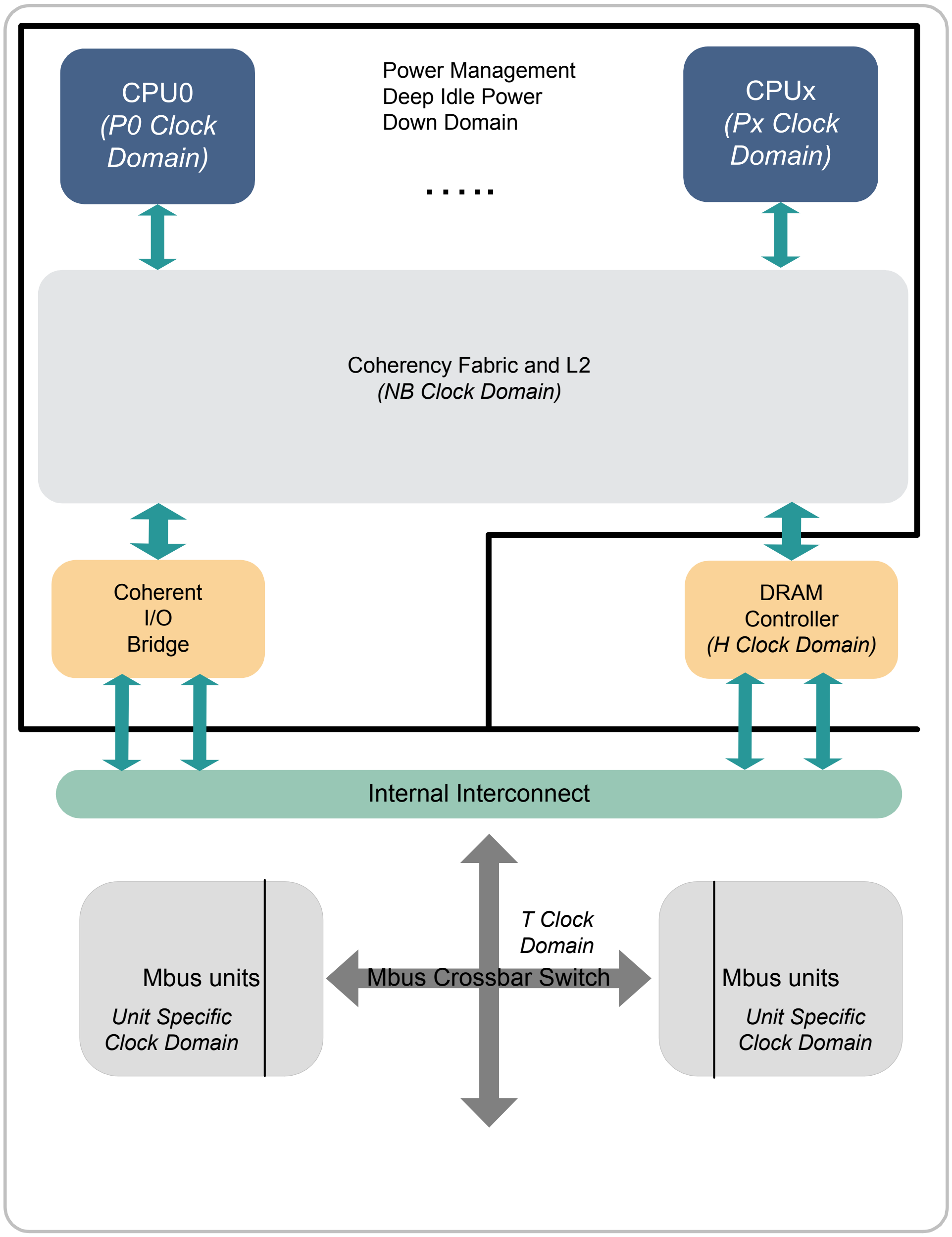
Table 9 lists the primary clock domains. The CPU(s), Coherency Fabric and DRAM clock domains are synchronous to each other (all generated from the CPU PLL). The core clock domain is generated from the MISC PLL and is asynchronous to the former domains.

Table 9: Primary Clock Domains

Clock Domain	Description
Px	CPUx clock domain This clock can be gated off in the Idle low power state.
NB	L2 and Coherency Fabric clock domain
H	DRAM controller clock domain The DRAM interface can run at 1:1 or 2:1 ratio to the H clock domain, refer to the device *Hardware Specifications*.
T	Core (Mbus crossbar switch and Mbus units) clock domain
Unit Specific	Each Mbus unit supports its own specific clock or multiple clocks (for example, PHY or I/O interfaces)

Copyright © 2014 Marvell

Figure 4: ARMADA® XP Clock and Power Domains



The following sections outline the device's internal bus architecture.

4.1 Coherency Fabric

The Coherency Fabric uses the AXI protocol (see the *AMBA AXI Protocol v1.0 Specification*) as the internal interface to the rest of the device. The Coherency Fabric maintains cache coherency between the IO masters and the CPU caches, and coherency between the CPUs caches in a multi-CPU environment. An IO transaction that addresses a shared memory area goes through the Coherency Fabric and snoops the CPU caches before accessing the DRAM. An IO transaction that accesses a non-shared memory area is routed directly to DRAM, without snooping the CPU caches.

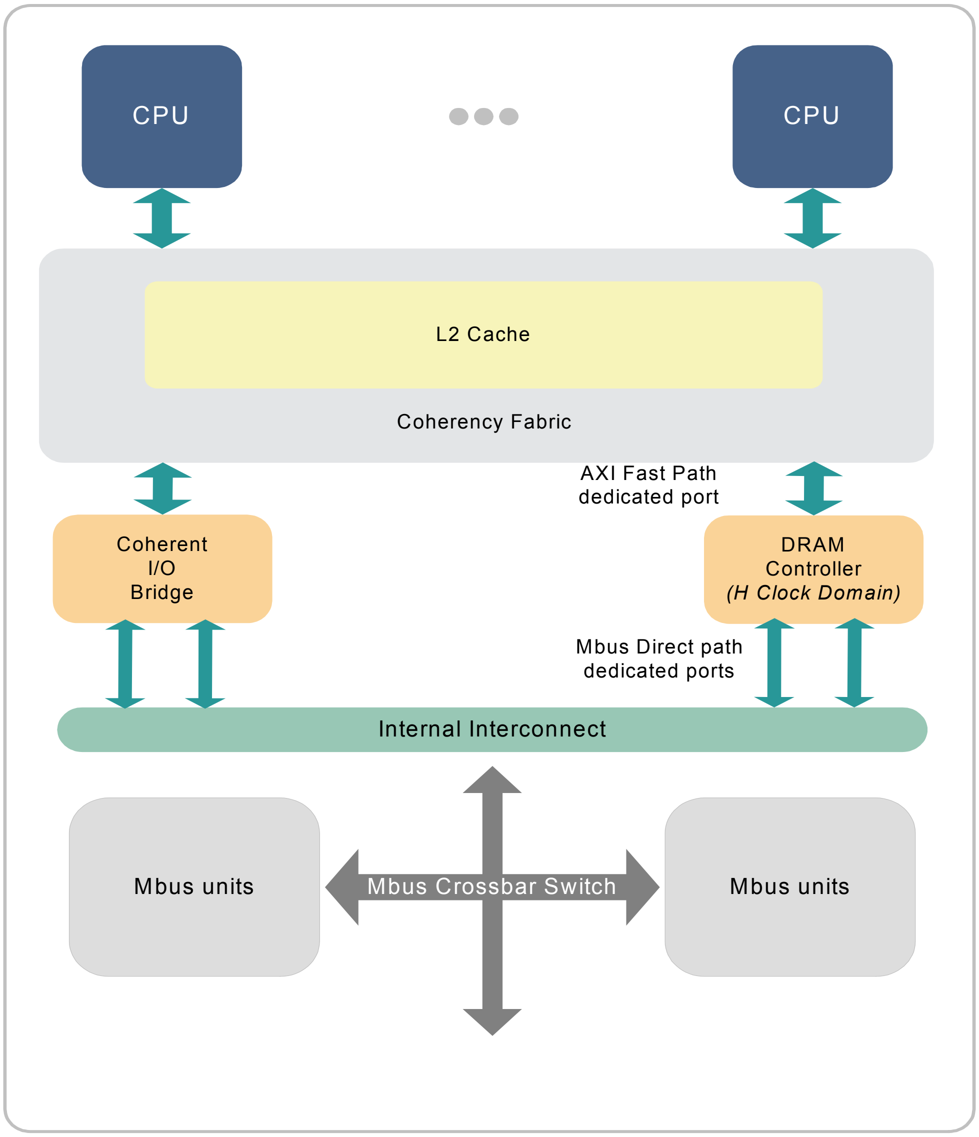
In addition, the device implements a dedicated AXI-based fast path between the Marvell® Core Processors and L2 interface to the DRAM controller, in order to reduce CPU to DRAM latency. This is a full duplex,256-bit wide interface running at H clock domain. The H clock domain can be a 1:N or 2:N fraction of the CPU clock domain. Since the Marvell® Core Processors and the DRAM clocks are edge-aligned, no synchronization is required on CPU access to DRAM, thereby resulting in minimum latency.

The AXI protocol enables high-performance, high-frequency interconnect to the L2 cache, the device peripheral, and to the DRAM controller. It is intended to maximize the device DRAM throughput, and to minimize the bus read latency. The AXI interconnect provides the following features:

- Separate address/control and data channels—burst-based transactions with only start addresses:
  - Enables multiple outstanding read transactions.
  - Enables new read/write requests during the read response data phase.
  - Enables new read transfer requests during the write data phase.
  - Supports unaligned data transfer using byte strobes.
- Separate data interfaces for read and write.
  - Enables write data transfer and read response data transfer to occur simultaneously.
  - Guarantees streaming read response transfer, with no acknowledgment needed.
- Separate read and write response channels
  - Enables out-of-order transaction completion.
  - Enables point-to-point read/write ordering.
- Supports the implementation of atomic access primitives.
  - Lock access to enable the atomic processor write and read pair.
  - Exclusive access mechanism enables the implementation of semaphore type operations without locking the bus.
- Supplies outer cache hits, controlling the shared Level-2 cache policy.
- Coherency extension.
  - Supplies different snoop hit on the regular AXI transactions.
  - Independent snoop probes and snoop response channels.
  - Enables snoop pipelining and out-of-order snoop responses.
- Separate interfaces to Level-2 cache, SDRAM controller and to Mbus bridge:
  - Enables simultaneous transactions to Level-2 cache, SDRAM controller and to Mbus bridge.
  - Guarantees streaming read response transfer, no acknowledgment needed.
- Transaction ID is attached to the transaction command:
  - Enables out-of-order transaction completion.

Figure 5 is a block diagram illustrating the Coherency Fabric interconnect to the Mbus peripherals and to the DRAM controller.

Figure 5: Coherency Fabric Interconnect to Mbus and DRAM Controller



4.2 DRAM Controller Interface Interconnect

The DRAM controller interface works with the AXI interface and the Mbus interface.

AXI Interface

- The DRAM controller provides three read/write transaction buffers for CPU based transactions:
  - Supports multiple outstanding transactions
  - Enables back-to-back transactions over the DRAM memory
- The fast path interface between the Coherency Fabric and the DRAM controller has two levels of arbitration:
  - A round-robin arbitration scheme between the CPUs and Coherent I/O Bridge (CIB) while accessing DRAM. The Coherency Fabric resolves address collision cases as read after write, and write after write, and guarantees correct order of execution.
  - Arbitration between the outcome of the arbiter mentioned above with the L2 outgoing transactions (evictions).

For additional information about the DRAM controller AXI interconnect see Section 12, DRAM Controller, on page 173.

Mbus Interface

In software Cache Coherency mode of operation, enables high throughput of the DRAM, the device's DRAM controller interfaces the Mbus fabric with two 64-bit Mbus ports.

The DRAM controller resolves any address collisions between CPU and Mbus transactions, and guarantees correct ordering on the DRAM bus. A controllable arbitration scheme between mbus traffic and CPU traffic applies.

For further details, see Section 12.4, Arbitration and Ordering, on page 179.

4.3 Mbus Interconnect

The Mbus fabric is a Marvell proprietary, high-bandwidth, full duplex any-to-any interconnect. It supports transaction pipelines, split transactions, and out-of-order completion. Moreover, it allows concurrent data transfers between different interfaces. For example, it is possible for the Ethernet Network Controller to read from the DRAM, while simultaneously the Marvell® Core Processors writes to PCI Express port0, and the DMA transfers data from DRAM to PCI Express Port1.

4.3.1 Mbus Traffic

The device units can act as Mbus masters that generate read/write transactions over the Mbus, or as an Mbus slave unit that responds to read/write transactions.

The Mbus master units include the following:

- Marvell® Core Processor(s) (via the Coherency Fabric)
- IDMA
- XOR DMA
- PCI Express
- Security Acceleration Engine
- I^{2}C serial EEPROM
- SATA DMA
- USB DMA
- Ethernet Networking Controller DMA
- TDM DMA
- Audio DMA

- SDIO DMA
- Buffer Management unit

The Mbus slave units include the following:

- DRAM controller
- Device bus controller
- UART controller
- SPI controllers
- NAND Flash Controller
- RTC
- Security Acceleration Engine local SRAM
- PCI Express
- Integrated bootROM
- All registers files (which are distributed in all units)

The Mbus supports the following traffic routes between the master and slave units:

- CPU read/write from/to all of the device interfaces and registers
- PCI Express read/write from/to all of the device interfaces and registers (including itself—see note below)
- I^{2}C Serial ROM initialization access to all the device interfaces and registers
- Network ports, read/write access to DRAM, and Buffer Management Unit
- IDMA, read/write access to DRAM, Device bus, and PCI Express. In addition it supports write access to the UART interface for DMA based UART
- XOR read/write access to DRAM, Device bus, SPI[1] and PCI Express
- USB, SATA, TDM, SDIO, Buffer Management unit, and Cryptographic Engine read/write access to DRAM.

Note

- When the PCI Express port is configured as quad x1 lanes, peer-to-peer traffic between the x1 lanes is possible.
- The device supports peer-to-peer traffic between different PCIe ports (for example, traffic from PCIe0.x lane—that is port 0 lane x—to PCIe1.y lane and vice versa).
- Access to internal registers is restricted not to exceed a 32-bit word boundary (no burst access to the device registers).

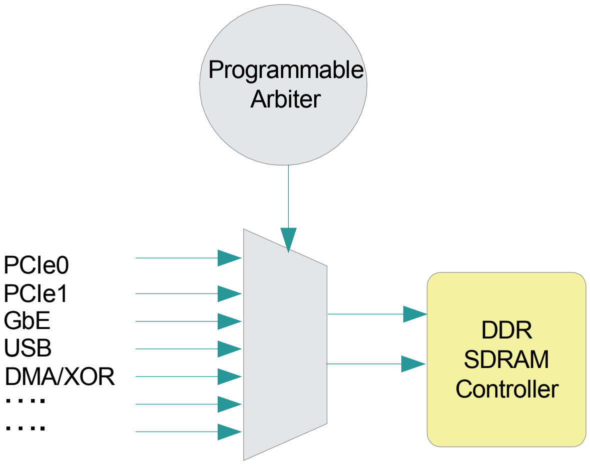
4.3.2 Mbus Arbitration

Since multiple Mbus master units may attempt to simultaneously access the same target unit, there is an arbiter per each target resource.

Most of the device units act as targets for register access (for example, USB and SATA), and are typically managed only by the Marvell® Core Processor. These unit arbiters are fixed round-robin arbiters. Other target units, like the DRAM controller, use programmable arbitration mechanisms to optimize device performance, according to system requirements, as shown in Figure 6.

[1] Read only from SPI Flash

Figure 6: SDRAM Interface Arbitration

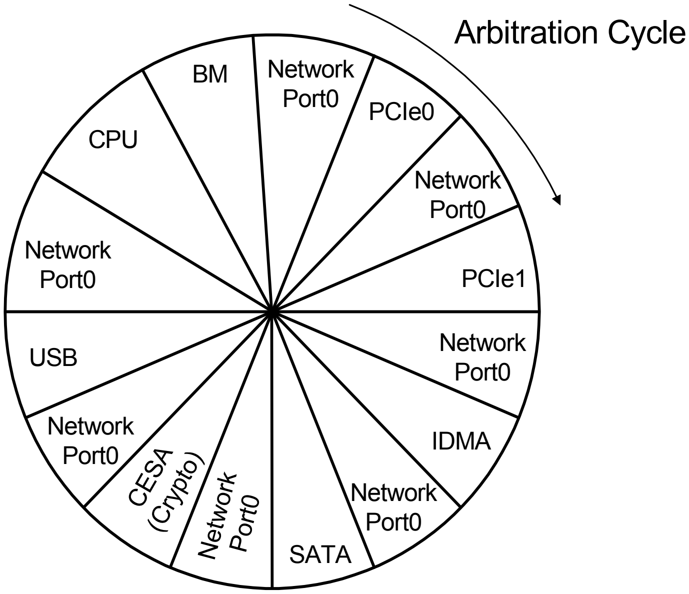


The programmable arbiter is a user-defined round-robin arbiter (called a “Pizza Arbiter”). Each slice of the arbiter can be assigned to a specific master unit ID. The following list defines the unit ID code on the Mbus, and may be used to create the Pizza Arbiter priority and scheduling scheme:

- 0x1 – DMA based UART and I^{2}C serial init
- 0x2 – ARMv7-A Cortex™-A9 processor
- 0x3 – Network ports 2,3
- 0x4 – PCIe ports 0.x and 2.x
- 0x5 – USB
- 0x6 – 4 IDMA engines and XOR engines 0 and 1
- 0x7 – Network ports 0,1
- 0x8 – PCIe ports 1.x and 3.x
- 0x9 – Security Engines 0 and 1
- 0xA – SATA0 and SATA1
- 0xB – TDM
- 0xC – Buffer Management unit
- 0xD – SDIO and NAND Flash controller
- 0xF – XOR engines 2 and 3

Figure 7 illustrates the DRAM controller arbiter setting.

Figure 7: Configurable Weights Arbiter



By default, there are 32 slices in the pizza arbiter. It is possible to define each of the slices of this pizza arbiter. This arbiter works on a per transaction basis. A transaction can vary from one up to 16 Qwords (Qword = 64-bit data transfer). In the above example, if all units are constantly requesting access to the device DRAM, almost 50% of N transactions running on the DRAM interface will be Network Port0 transactions.

The pizza arbiter configuration also allows the user to guarantee minimum latency. Even if the Network Port0 does not require its 50% transaction allocation, the above configuration guarantees that, in the worst case, the Network Port0 request needs to wait for one access from another unit before being served.

The pizza arbiter is based on Mbus transactions. This means that it does not always reflect DRAM bandwidth allocation. For example, UnitA can have a typical Mbus transaction size of 128B and UnitB can have a typical Mbus transaction size of 32B. By setting the pizza arbiter to the same number of transactions for UnitA and UnitB, UnitA will get 4x the DRAM bandwidth as UnitB.

The different Mbus master units have the following burst capabilities as Mbus masters:

- The IDMA can be configured to generate bursts of 8, 16, 32, 64, or 128B per single Mbus transaction.
- The XOR DMA can be configured to generate bursts of 32, 64, or 128B per single Mbus transaction.
- The network port DMAs can be configured to generate bursts of 8, 16, 32, 64, or 128B per single Mbus transaction.
- The USB DMA maximum burst on a single Mbus transaction is 64B. The DMA can be configured to generate bursts of 16, 32, or 64B for a single Mbus transaction.
- The SATA DMA maximum burst on a single Mbus transaction is 128B. Writes can be limited (by configuration) to 32B.

- PCIe maximum burst on a single Mbus transaction is 128B. Writes can be limited (by configuration) to 32B.
- The Security engine DMAs can be configured to generate bursts of 32 or 128B per single Mbus transaction.
- The SDIO DMA maximum burst of a single Mbus transaction is 32B.
- The Buffer Management maximum burst of a single Mbus transaction is 128B.

At each clock cycle, the Mbus arbiter samples all of the requests and gives the bus to the next agent, according to the pizza arbitration scheme. It is parked on the last access.

An arbiter slice can also be marked as NULL. If marked as NULL, the arbiter works as if the NULL slice does not exist.

Once a unit is removed from an interface's pizza arbiter control register, the arbiter has no access to this interface. If, for example, the USB unit is removed from the DRAM interface pizza arbiter, it can no longer access the DRAM. If the USB unit attempts to access the DRAM, the unit hangs.

A CPU access to the DRAM controller registers uses the Mbus. It does not pass through the CPU-to-DRAM fast path bus.

Note

Always define a slice in the DRAM controller arbiter for the Marvell® Core Processors (unit ID = 0x2).

Multiple masters can share the same Mbus port and unit ID:

- The Ethernet Mbus port integrates 2 Ethernet Controllers
- The USB Mbus port integrates 3 USB Controllers
- The TDM unit integrates 32 voice channels
- The 4 IDMA engines and 2 of the 4 XOR DMA engines are integrated into a joined unit
- The XOR DMA Mbus port integrates two engines (out of the 4 XOR DMA channels)
- PCIe unit0 integrates 5 PCI Express MACs
- PCIe unit1 integrates 5 PCI Express MACs

In each of these cases, the unit incorporates an internal fixed round-robin arbiter to arbitrate between these master sub-units.

4.4 Transaction Ordering

The device implements a set of rules and methods to enforce ordering on the system level. This section outlines these ordering aspects and the way they are handled by the device:

- Semaphore management between the processors in multi-CPU applications, through CPU Lock and exclusive instructions
- Read after write ordering and write after write ordering
- PCI Express bridge ordering rules
- Producer-consumer model ordering (where the producer can be an I/O device and consumer is the Marvell® Core Processors, or the other way around) and sync barrier operation.

4.4.1 CPU Lock and Exclusive Access

The Coherency Fabric supports the synchronization primitives that are a critical part of a multi-core operation. The primitives are swap (SWP) instruction, load exclusive instruction (LDREX), and store exclusive instruction (STREX).

The swap instruction triggers an automatic access, and is associated with a locking request on the interconnect. When the core processor initiates a swap instruction, the Coherency Fabric ensures that only when this processor has initiated an unlock transaction, other CPUs can access the interconnect. Throughout the time between the lock and unlock transactions, CPUs other than the lock initiator are stalled by the interconnect.

The exclusive access mechanism enables a non-blockage implementation of automatic operation, using a Load-Exclusive/Store-Exclusive instruction pair. When a Marvell® Core Processors generates a load exclusive command (LDREX) to address A, the address is recorded by the Coherency Fabric. The corresponding store exclusive instruction (STREX) succeeds in writing back to memory only if no peer CPU has performed a more recent store of address A. The store-exclusive instruction returns a status bit to the CPU, indicating if the memory write has been completed successfully.

4.4.2 Read after Write and Write after Write Ordering

The implementation of the transaction queues for each of the device’s master and slave units guarantees read-after-write, and write-after-write ordering of transactions from the same originator to the same target.

However, the basic implementation of the transaction queues cannot guarantee ordering of transactions between different sources and destinations. For example, if the ARMv7-A Cortex™-A9 generates 2 consecutive write transactions (the first to the Device bus and the second to the PCIe interface), the hardware cannot guarantee that the write on the Device bus will be executed first on the interface lines.

The Coherency Fabric resolves all read-after-write and write-after-write hazards between peer CPUs and also between CPU and IO masters that are accessing the same shared resources. Although CPU transactions to DRAM go through a different path than transactions to other interfaces, transaction ordering is still maintained by the Coherency Fabric. For example, if a write to DRAM is followed by a write to one of the Network port registers, the Coherency Fabric implementation guarantees that the write to the Network port register is not performed before the write to DRAM is committed. Similarly, when a memory region is defined as shared in both the CPU MMU tables and the IO master address window decoding attributes, any read access by the IO master from a pending CPU write address is resolved with the most updated data from memory, including the CPU write update.

For more details about the Coherency Fabric ordering maintenance implementation, see Section 9, Coherency Fabric, on page 147.

4.4.3 PCI Express Host Hardware Enforced Ordering

The device supports PCI Express bridge transaction ordering rules. Particularly in cases of a CPU read from the PCI Express endpoint, the device drives the read response on the CPU bus only after flushing all upstream write data previously posted from the PCI Express into memory.

To enable PCI Express host hardware enforced ordering, clear both the <RxNpPushDis> field and the <RxCmplPushDis> field in the PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 889 p. 1183) to 0 (the default settings).

4.4.4 PCI Express Endpoint Hardware Enforced Ordering

The device also supports the PCIe transactions ordering rules when it acts as an endpoint. Particularly, when an external PCI Express host performs a read from the device local memory. The device only drives downstream read completion after all pending downstream posted write data is flushed.

To enable PCI Express endpoint hardware enforced ordering, set the both the <TxNpPushDis> field and the <TxCmplPushDis> field in the PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 889 p. 1182) to 0 (the default settings).

4.4.5 Producer-Consumer Ordering

The basic concept of a producer-consumer model works as follows:

1. The Producer places some data in memory for the consumer to process.
2. The Producer notifies the consumer (via interrupt or by any other means) that there is data pending in memory for the consumer to process.
3. The Consumer reads the data from memory and processes.

The device implementation guarantees that this model works properly, meaning it is guaranteed that when the consumer reads data from memory, it reads the valid data.

Note The following description deals with producer-consumer operation between Marvell® Core Processors and a Network port DMA. However, it also applies to the rest of the device DMA engines (e.g IDMA, XOR, USB, SATA, Cryptographic Engine etc.).

The Network port transmit operation consists of:

1. The Marvell® Core Processors prepares Tx descriptors and buffers in memory.
2. If the memory area is defined as non-shared between the CPU and the DMA master, the software flushes buffer descriptors from L1 and L2 caches to the DRAM, using cache operations. The software also performs drain write buffer operation, to guarantee that data is flushed all the way to memory.
   If the memory area is defined as shared between the CPU and the DMA master, there is no need for the software to flush the buffer descriptors from the L1 and L2 caches. Coherency with the IO is guaranteed by the Coherency Fabric. Refer to Section 9.6, I/O Coherency, on page 153 for more details about IO cache coherency.
3. Triggering the Network port's DMA (a write to a Network port register).
4. The Network port DMA starts to fetch descriptors and buffers from the DRAM.

   If the target memory address was defined as shared, the Coherency Fabric takes care of snooping the L1 and L2 caches and returns the data, either directly from the caches in case of cache hit, or from the DRAM in case of cache miss.

   If the target memory address was defined as non-shared, no cache snooping is needed, as caches were flushed by the software. The DRAM controller implementation guarantees that the Network port DMA reads the latest data from DRAM, rather than old invalid data (see Section 12, DRAM Controller, on page 173 for full details).

Network port receive operation consists of:

1. The Network port DMA writes received packets to buffers in memory.
2. The Network port DMA writes the Rx descriptor status to memory (either to DRAM, L2 cache, or SRAM). It may then interrupt the Marvell® Core Processors.
3. The Marvell® Core Processors reads the interrupt cause register (identify interrupt cause).
4. If the memory area is shared, the software initiates a sync barrier operation to flush the Coherency Fabric buffers to memory (for details about Sync Barrier operation refer to

Section 9.6.4, I/O Synchronization Barrier Mechanism, on page 155). If the memory area is not shared, no sync barrier operation is needed.

5. The Marvell® Core Processors reads the Rx descriptor and data from memory.

   The DRAM controller implementation guarantees that the Marvell® Core Processors reads the latest data from DRAM, rather than old invalid data (see Section 12, DRAM Controller, on page 173 for full details).

5 System Considerations

This section describes the ARMADA® XP system considerations, including data integrity and Big and Little Endian byte ordering.

5.1 Data Integrity

The ARMADA® XP support data integrity on most of its external interfaces, as listed below:

- LCRC checking and generation on the PCI Express interface
- CRC checking and generation on the Ethernet, SATA, and USB ports
- ECC checking and generation on the DRAM interface

The device supports parity protection on CPU core L1 and L2 caches and ECC on its L2 cache. In addition, the device supports parity protection on the internal data paths.

5.1.1 Cache Protection

The CPU L0 and L1 cache supports Data and Tag RAM parity protection for both Instruction and Data caches. For further details, see the Marvell® Core Processor specifications (see Related Documentation on page 27).

Level 2 cache supports Data RAM ECC protection and Tag RAM parity protection. For further details, see Section 8, Level-2 Cache, on page 122.

5.1.2 SDRAM ECC

The device implements Error Checking and Correction (ECC) on accesses to the SDRAM. It supports detection and correction of one data bit errors, and detection of 2 errors (SEC/DED).

5.1.2.1 ECC Calculation

ECC is calculated on a 64-bit vector. When using 32-bit DRAM, the calculation is performed assuming that the higher 32 data bits [63:32] unused bits are all zeros. Each of the 64 data bits and 8 check bits have a unique 8-bit ECC check code, as shown in Table 10. For example, data bit 12 has the check value of 01100001, and check bit 5 has the check value of 00100000.

Table 10: ECC Code Matrix

Check Bit	Data Bit	ECC Code Bits								Number of 1s in syndrome
		7	6	5	4	3	2	1	0	
	63	1	1	0	0	1	0	0	0	3
	62	1	1	0	0	0	1	0	0	3
	61	1	1	0	0	0	0	1	0	3
	60	1	1	0	0	0	0	0	1	3
	59	1	1	1	1	0	1	0	0	5
	58	1	0	0	0	1	1	1	1	5

Table 10: ECC Code Matrix (Continued)

Check Bit	Data Bit	ECC Code Bits								Number of 1s in syndrome
		7	6	5	4	3	2	1	0	
4		0	0	0	1	0	0	0	0	1
3		0	0	0	0	1	0	0	0	1
	57	1	1	1	0	0	0	0	0	3
	56	1	0	1	1	0	0	0	0	3
	55	0	0	0	0	1	1	1	0	3
	54	0	0	0	0	1	0	1	1	3
	53	1	1	1	1	0	0	1	0	5
	52	0	0	0	1	1	1	1	1	5
5		0	0	1	0	0	0	0	0	1
2		0	0	0	0	0	1	0	0	1
	51	1	0	0	0	0	1	1	0	3
	50	0	1	0	0	0	1	1	0	3
	49	0	0	1	0	0	1	1	0	3
	48	0	0	0	1	0	1	1	0	3
	47	0	0	1	1	1	0	0	0	3
	46	0	0	1	1	0	1	0	0	3
	45	0	0	1	1	0	0	1	0	3
	44	0	0	1	1	0	0	0	1	3
	43	1	0	1	0	1	0	0	0	3
	42	1	0	1	0	0	1	0	0	3
	41	1	0	1	0	0	0	1	0	3
	40	1	0	1	0	0	0	0	1	3
	39	1	0	0	1	1	0	0	0	3
	38	1	0	0	1	0	1	0	0	3
	37	1	0	0	1	0	0	1	0	3
	36	1	0	0	1	0	0	0	1	3
	35	0	1	0	1	1	0	0	0	3
	34	0	1	0	1	0	1	0	0	3

Table 10: ECC Code Matrix (Continued)

Check Bit	Data Bit	ECC Code Bits								Number of 1s in syndrome
		7	6	5	4	3	2	1	0	
	33	0	1	0	1	0	0	1	0	3
	32	0	1	0	1	0	0	0	1	3
	31	1	0	0	0	1	0	1	0	3
	30	0	1	0	0	1	0	1	0	3
	29	0	0	1	0	1	0	1	0	3
	28	0	0	0	1	1	0	1	0	3
	27	1	0	0	0	1	0	0	1	3
	26	0	1	0	0	1	0	0	1	3
	25	0	0	1	0	1	0	0	1	3
	24	0	0	0	1	1	0	0	1	3
	23	1	0	0	0	0	1	0	1	3
	22	0	1	0	0	0	1	0	1	3
	21	0	0	1	0	0	1	0	1	3
	20	0	0	0	1	0	1	0	1	3
	19	1	0	0	0	1	1	0	0	3
	18	0	1	0	0	1	1	0	0	3
	17	0	0	1	0	1	1	0	0	3
	16	0	0	0	1	1	1	0	0	3
	15	0	1	1	0	1	0	0	0	3
	14	0	1	1	0	0	1	0	0	3
	13	0	1	1	0	0	0	1	0	3
	12	0	1	1	0	0	0	0	1	3
	11	1	1	1	1	1	0	0	0	5
	10	0	1	0	0	1	1	1	1	5
7		1	0	0	0	0	0	0	0	1
0		0	0	0	0	0	0	0	1	1
	9	0	1	1	1	0	0	0	0	3
	8	1	1	0	1	0	0	0	0	3

Table 10: ECC Code Matrix (Continued)

Check Bit	Data Bit	ECC Code Bits								Number of 1s in syndrome
		7	6	5	4	3	2	1	0	
	7	0	0	0	0	0	1	1	1	3
	6	0	0	0	0	1	1	0	1	3
	5	1	1	1	1	0	0	0	1	5
	4	0	0	1	0	1	1	1	1	5
6		0	1	0	0	0	0	0	0	1
1		0	0	0	0	0	0	1	0	1
	3	1	0	0	0	0	0	1	1	3
	2	0	1	0	0	0	0	1	1	3
	1	0	0	1	0	0	0	1	1	3
	0	0	0	0	1	0	0	1	1	3

The device calculates the ECC by taking the EVEN parity of ECC check codes of all logic data bits. For example, if the 64 bit data is 0x45, the binary equivalent is 01000101. From Table 10, the required check codes are 00001101 (bit [6]), 01000011 (bit [2]) and 00010011 (bit [0]). Bitwise XOR of the check codes (even parity) results in an ECC value of 01011101.

If the device is configured to 32-bit mode, it reads 32 bits of data, pads bits [63:32] with zeros to create a 64-bit bus, and adds 8-bits of ECC data. When the device is configured to 64-bit mode, the device reads 64-bits of data, and adds 8 bits of ECC data. The device calculates the ECC based on the 64-bit data and then compares that against the received ECC. The result of this comparison (bitwise XOR between received ECC and calculated ECC) is called the syndrome.

If the syndrome is 00000000, both the received data and ECC are correct.

- If the syndrome is any other value, the device assumes that there is an error either in the received data or the received ECC.
- If the syndrome contains a single 1, there is a single bit error in the ECC byte. For example, if the received data is 0x45, the calculated ECC is 01011101. If the received ECC is 01010101, the resulting syndrome is 00001000. Table 10, ECC Code Matrix, on page 82 shows that this syndrome corresponds to check bit 3. Since there is no error in the data itself (just in the ECC byte), data correction is not necessary.
- If the result syndrome contains 2 or more 1’s, this indicates that there is a double-bit error.

Note

These types of errors cannot be corrected. The device reports an error but will not change the data.

5.1.2.2 DRAM Interface Operation

On DRAM reads, the device reads the ECC byte with the data, calculates the ECC on the 64-bit read data (while using 32-bits DRAM, upper 32 bits are zero padding, lower 32 bits were read from SDRAM), and compares it against the ECC byte being read from DRAM. In the case of a single data bit error, it corrects the error and drives the correct data to the initiating interface. In the case of 2-error detection (or 3 or 4 errors that reside in the same nibble), it only reports an error (see Section 5.1.2.3, ECC Error Report, on page 86).

On a write transaction, the device calculates the new ECC and writes it to the ECC bank, with the data that is written to the data bank. Since the ECC calculation is based on 64-bit data width, if the write transaction is smaller than 64 bits, the device performs a read modify write (RMW) sequence. It reads the full 32/64-bit data, merges the incoming data with the read data (while zero padding the upper 32 bits for a 32-bit DRAM, for the ECC calculation only), and writes the new data back to the SDRAM bank with new ECC byte.

Note

If the device identifies a non-correctable error during the read portion of the RMW sequence, the device writes the data back to DRAM with a non-correctable ECC byte. This behavior guarantees that the error is still visible if there is a future read from this DRAM location.

In the case of a burst write to DRAM, the device executes a RMW access of the entire burst, even if only part of the data requires RMW. Performing a RMW access on only part of the burst is not efficient, due to the overhead of bus turnaround cycles.

The device also supports forcing a bad ECC written to the ECC bank for debug purposes. If this mode is enabled, it drives a fixed ECC byte configured in the SDRAM Error Control Register (Table 444 p. 768), rather than calculating the ECC to be written to the ECC bank.

5.1.2.3 ECC Error Report

In the case of ECC error detection, the device asserts an interrupt (if not masked), and latches the following:

- Address in the SDRAM Error Address Register (Table 443 p. 767)
- 64-bit read data in the SDRAM Error Data (High) Register (Table 439 p. 766) and SDRAM Error Data (Low) Register (Table 440 p. 766).

  For 32-bit DRAM, 32-bit read data in the SDRAM Error Data (Low) Register (Table 440 p. 766).

Note

For further information about these registers, see Appendix A.3.1, SDRAM Data Protection and Error Report Registers, on page 766.

In the case of multiple errors detection, only the address and data for the first error are latched in the corresponding registers. Latching of new data into these registers is enabled only after reading the ECC Error Address register, and clearing the interrupt. The interrupt handler must read this register last.

The device reports an ECC error whenever it detects but cannot correct an error (2, 3, or 4 bits errors).

The device also reports on single bit errors (correctable errors), based on the setting of the ECC threshold, bits [23:16], in the ECC Control register.

- If the threshold is cleared to 0, there is no report on single bit errors.

- If set to 1, the device reports each single bit error.
- If set to ***n***, the device reports each ***n*** single bit error.

The SDRAM controller also contains two 32-bit ECC error counters. One counter is for single bit ECC errors and the other one handles double bit errors. Counters can be used by the system software to monitor the soft errors rate. Writing 0x0 to a counter resets its value (see Section 12.9, Error Checking and Correction (ECC) and Read Modify Write, on page 188).

5.1.2.4 Memory Scrubbing

When using high-density memory arrays, the probability for soft errors must be noted. These error types typically flip one bit at a time, causing single bit ECC errors. However, if the single bit error is not corrected, there is a probability of an additional flip in the same data row. The result is 2 bits of uncorrectable error.

To avoid single bit ECC errors, the device contains the required logic for the automatic cleaning of single bit ECC errors, without interfering with normal operation of the system. This logic is activated via one of the XOR DMA engines.

When the XOR DMA is configured for ECC errors cleanup, it is no longer used as a XOR engine. However, the device contains two such DMAs, therefore one of them can be used for errors cleanup, while the other can continue working as a XOR machine.

The memory space to be cleaned of ECC errors is defined by system software in the XOR DMA descriptors list. When configured to ECC errors cleanup, the XOR DMA generates consecutive write transactions to DRAM, with all bytes masked (dummy writes with all byte enables de-asserted). This action causes the SDRAM controller to perform Read-Modify-Write (RMW) (as is the case for any write of less than 64-bits). It reads 32/64-bits of data from the DRAM and, if it detects a single bit data or ECC code error (or not detected error at all), it writes the corrected data back to memory, with proper ECC. Since all byte enables are inactive, the data received from the XOR DMA is ignored.

When the XOR DMA is configured to ECC errors cleanup, it can be set to Timer mode. Rather than generating consecutive dummy writes to DRAM, it transfers a small chunk of data every time the timer expires. This method avoids interfering with the normal operation of the SDRAM controller.

5.1.3 PCI Express Data Integrity

The device supports the different PCI Express layers data protection mechanisms as defined in PCI Express specification.

When the MAC layer detects a receiver error (e.g. 8b/10b error), it drops the symbol and reports an error. When the link layer detects an error (e.g. LCRC or sequence ID error), it drops the packet and reports an error. It also responds with acknowledge.

MAC and link layer errors are considered correctable errors; the initiator should re-transmit the packets. Still, If it is a repeated error, the link eventually enters a recovery state.

Until this point, when the transaction layer detects a poisoned TLP (EP bit set), it reports an error. The transaction is forwarded to the target interface with an erroneous data indication, except for the case of write to the device’s internal registers, in which data is discarded.

Note

The device does not support ECRC.

On the Tx side, the device always drives correct LCRC (and, in case of LCRC error in the receive side, retransmits).

5.1.4 Errors Forwarding

Although each interface includes the required logic to detect and report data errors, this is sometimes inadequate due to the latency of interrupt routines. For example, poisoned TLP is detected upon a PCI Express write to SDRAM. If the Marvell® core processor reads this data from memory (without knowing that this is erroneous data), it can cause undesired results.

To guarantee that this scenario does not occur, the device supports an error forwarding mechanism.

When a transaction received on one interface is detected as erroneous:

- An error interrupt is asserted, and transaction information is registered.
- The transaction is forwarded to the target interface with an erroneous data indication.

The device supports an error-forwarding mechanism. When a transaction received on a unit or interface is detected as erroneous[1] (for example, poisoned TLP is detected upon a PCI Express write to SDRAM):

- An error interrupt is asserted, and transaction information is registered.
- The transaction is forwarded to the target interface with an erroneous data indication.

The device internal DMAs (IDMA, XOR DMA, Ethernet SDMA, SATA DMA, SDIO DMA, USB DMA, TDM DMA) never generate erroneous data indication. However, they are sensitive to erroneous data indication received from other units:

- IDMA, if it detects an erroneous data indication on read data, also forwards this indication during the write to the destination.
- Ethernet SDMA, if it detects an erroneous data indication on data buffer read, transmit this buffer with bad CRC.
- Any of the DMAs stop if they encounter an erroneous data indication during descriptor read.

Note

A write is discarded (register is not updated) when the device detects an error during a write to one of the chip internal registers.

- <PerrProp> = 1 and <ECC> = 1: Forward error to DRAM; Meaning, perform the RMW as usual, but flip ECC bits before the write, resulting in a non-correctable error in DRAM.
- <PerrProp> = 1 and <ECC> = 0: Data is discarded (write with all byte enables in-active).

5.1.4.1 Device Bus

The MV78230/78x60 does not support parity on the local device bus. In case of a write access to the Device bus with bad parity indication, the Device controller discards the data.

5.1.4.2 Marvell® Core Processor

If a CPU read from the SDRAM results in an uncorrectable error, the error is propagated to the Marvell® Core Processor pipeline (transaction abort), and the Marvell® Core Processors enters an exception.

Similarly, if the <MBUS Err Prop Enable> field in the Coherency Fabric Control Register (Table 155 p. 612) is set to 1, upon a CPU read from the I/O device that results in erroneous read data, the error is propagated to the CPU core pipeline (transaction abort), and the Marvell® Core Processor enters an exception.

[1] For interfaces and units that support error forwarding

Copyright © 2014 Marvell

5.1.4.3 PCI Express

If the <RxDPPropEn> field in the PCI Express Mbus Adapter Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 865 p. 1158) is set to 1, when receive poisoned TLPs (TLP with EP bit set), the PCI Express port forward the transaction to the target interface with erroneous data indication.

If the <TxDPPropEn> field in the PCI Express Mbus Adapter Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 865 p. 1158) is set to 1, the PCI Express port will force poisoned TLP (set EP bit) in case of:

- A PCI Express read from DRAM that results in none correctable ECC error
- Any case of a CPU or DMA write to PCI Express with erroneous data indication

5.1.5 Internal Data Path Parity Support

Each of the device internal buffers implemented as an SRAM array is parity protected. Each entry of SRAM is coupled with a parity bit (bit per buffer entry). Even parity is generated with every write to the SRAM and checked with every read from SRAM. If a parity error is detected, an interrupt is asserted (if not masked).

Note

A list of parity protected exceptions will be included in a future revision of this document.

Note

The internal data path error cannot be related to a specific transaction. This means that it is very hard for the device to recover from this error. Typically, a data path error requires a reset.

For software development and debug purposes, parity generation can be set to ODD parity. This results in parity error detection (since checking is always for EVEN parity).

5.2 PCI Express Reference Clock Output

The device supports dynamic gating of the output reference clock for additional power saving via CLKREQn signaling as defined in the PCI Express specification.

This functionality is available only when the port is configured as Root Complex.

5.3 Big and Little Endian Byte Ordering

The ARMADA® XP device supports both Big and Little Endian byte ordering, as defined in the ARMv7 architectures. It also supports hardware hooks to perform data conversion on part of its interfaces.

5.3.1 CPU Core Byte Ordering

The processor treats words in memory as:

1. Big Endian format
2. Little Endian format
3. Mixed Endian and unaligned data accesses

Details of Unaligned Data and Mixed Endian Data Structure can be found in ARM® Architecture Reference Manual (Document Number DDI 0100I).

5.3.2 PCI Express Space

The PCI Express specification defines that a TLP data payload is presented as Big Endian, and the packet header (address, command) is presented as Little Endian. The MV78230/78x60 PCI Express interface meets these requirements.

The least significant byte is always the one to be transmitted first. For example, if a CPU is configured to Big Endian and there is a CPU write of 4B to address 0x0 (appears on the CPU bus as 64'hAABBCCDD.XXXXXXXX), the MV78230/78x60 drives 0xAA as the first byte on the PCI Express link.

Note The PCI Express interface does not support different byte swapping per master and slave operation, nor different byte swapping on per address window basis.

5.3.3 DMA Data Swapping

All of the MV78230/78x60 DMAs (for example IDMA, XOR DMA, GbE SDMA, USB DMA, SATAHC DMA, and Encryption DMA) support the required mechanisms for proper data transfer both in Big and Little Endian environments.

5.3.3.1 IDMA Data Swapping

The IDMA transfers data from source to destination over the MV78230/78x60 Mbus. Since the Mbus byte orientation is the same in both source and destination accesses, there is no need for the IDMA to perform any byte swapping.

When configured to chain mode, the IDMA fetches the descriptors from memory and loads them into its local internal registers. Since the MV78230/78x60 internal registers are set to Little Endian mode, it is important that the descriptors are saved in the memory in Little Endian byte orientation. Moreover, when fetching a descriptor, the IDMA performs a read burst of 16B as follows:

- The first set of 4B (lower address) stands for the Byte Count.
- The second set of 4B stands for the Source Address.
- The third set of 4B stands for the Destination Address
- The last 4B stands for the Next Descriptor Pointer. The descriptor should be prepared in memory accordingly.

If the <DescBS> field in the Channeln Control (High) Register (n=0–3) (Table 1404 p. 1495) registers is set to 1, the IDMA performs byte swap on a 64-bit qword basis when fetching (and closing) descriptors.

5.3.3.2 XOR DMA Data Swapping

Similar to the IDMA, the XOR DMA does not need to perform any data byte swapping when transferring data buffers over the Mbus. However, the XOR DMA does support byte swapping on a 64-bit qword basis upon a read from source buffers or a write to a destination buffer via the <DrdResSwp> field in the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1479) and the <DwrReqSwp> field. This can be useful for endianess conversion.

The XOR DMA descriptor is being fetched from memory as a burst of 32B or 64B, depending on the XOR mode of operation. The descriptor is loaded into the XOR DMA reg file, which maintains Little Endian convention. Thus, the descriptor is expected to be prepared in memory accordingly.

If the <DesSwp> field in the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1479) is set to 1, the DMA performs a byte swap on a 64-bit qword basis when fetching (and closing) descriptors.

5.3.3.3 GbE SDMA Data Swapping

When transferring data between memory and MAC, the GbE SDMA can swap the bytes on 64-bit qword basis. Set the <BLMR> field in the SDMA Configuration (SDC) Register (i=0–3) (Table 585 p. 929) to 0, for byte swap of Rx buffers when written to memory, and set <BLMT> to 0, for byte swap of Tx buffers when being read from memory.

When reading Tx data from memory, the GbE SDMA always first transmits byte[7:0], followed by byte[15:8], and so on. In Big Endian convention, byte[63:56] stands for the least significant byte. This means that it is expected to be transmitted first. To achieve this behavior, set <BLMR> and <BLMT> fields to 0.

The GbE SDMA descriptor is being fetched from memory as a burst of 16B. The descriptor is loaded into the SDMA register file, which maintains Little Endian convention. Thus, the descriptor is expected to be prepared in memory accordingly.

If the <SwapMode> field in the SDMA Configuration (SDC) Register (i=0–3) (Table 585 p. 929) is set to 1, the SDMA performs byte swap on a 64-bit qword basis when fetching (and closing) descriptors.

5.3.3.4 USB Data Swapping

The MV78230/78x60 supports byte swap (within 8B qword) upon read/write access to memory. To enable byte swap, clear the <BS> field in the USB 2.0 Bridge Control Register (n=0–2) (Table 927 p. 1210).

Note

The data swapping logic cannot distinguish between descriptors to raw data. It is a static setting.

5.3.3.5 SATAHC Data Swapping

The MV78230/78x60 SATA controller supports the following byte swap mechanisms through the SATAHC Configuration Register (Table 939 p. 1219):

- Byte swap (within a 8B qword) upon Basic DMA read/write access to memory. Clear the <DmaBS> to enable byte swapping.
- Byte swap (within a 8B qword) upon EDMA read/write access to memory. Clear the <EDmaBS> to enable byte swapping.
- Byte swap (within a 8B qword) upon PRDP read/write access to memory. Clear the <PrdpBS> to enable byte swapping.

Note

The data swapping logic cannot distinguish between descriptors to raw data. It is a static setting.

5.3.3.6 Cryptographic Engine Data Swapping

The MV78230/78x60 Cryptographic hardware accelerator supports the following byte swap mechanisms:

- Byte swap and word swap (within a 8B qword) upon read/write access to the Cryptographic hardware accelerator integrated SRAM. Set the Target Attributes field in the CPU address decoding registers to achieve the required data swapping (see Section 3.1, Marvell® Core Processor Address Decoding, on page 55).
- Byte swap (within a 8B qword) upon TDMA read/write access to memory. Clear the <BS> field in the Control Register (p=0–1) (Table 1339 p. 1457) to enable byte swapping.

Note

The data swapping logic cannot distinguish between descriptors to packet data. It is a static setting.

6 BootROM Firmware

This section describes the boot sequence of the device. The boot method is determined according to Sample At Reset (SAR) configuration.

For multi-CPU support, it is possible to load and execute codes of one or more CPUs.

Figure 8 illustrates the bootROM firmware flow.

Figure 8: BootROM Firmware Flow

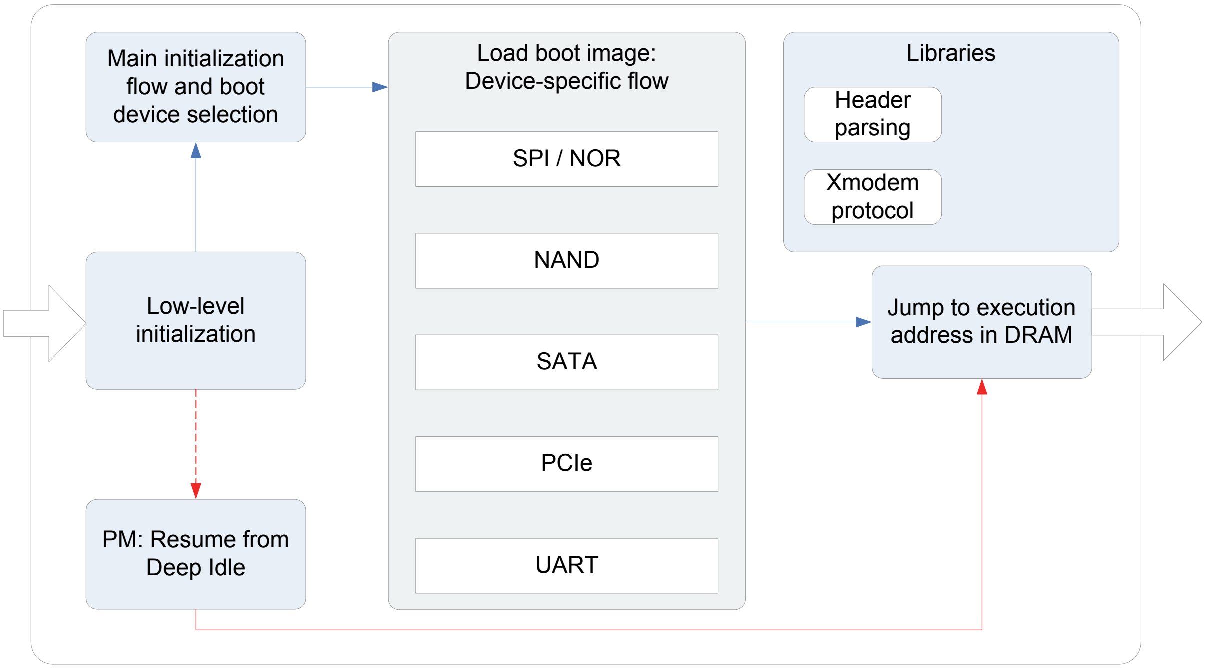
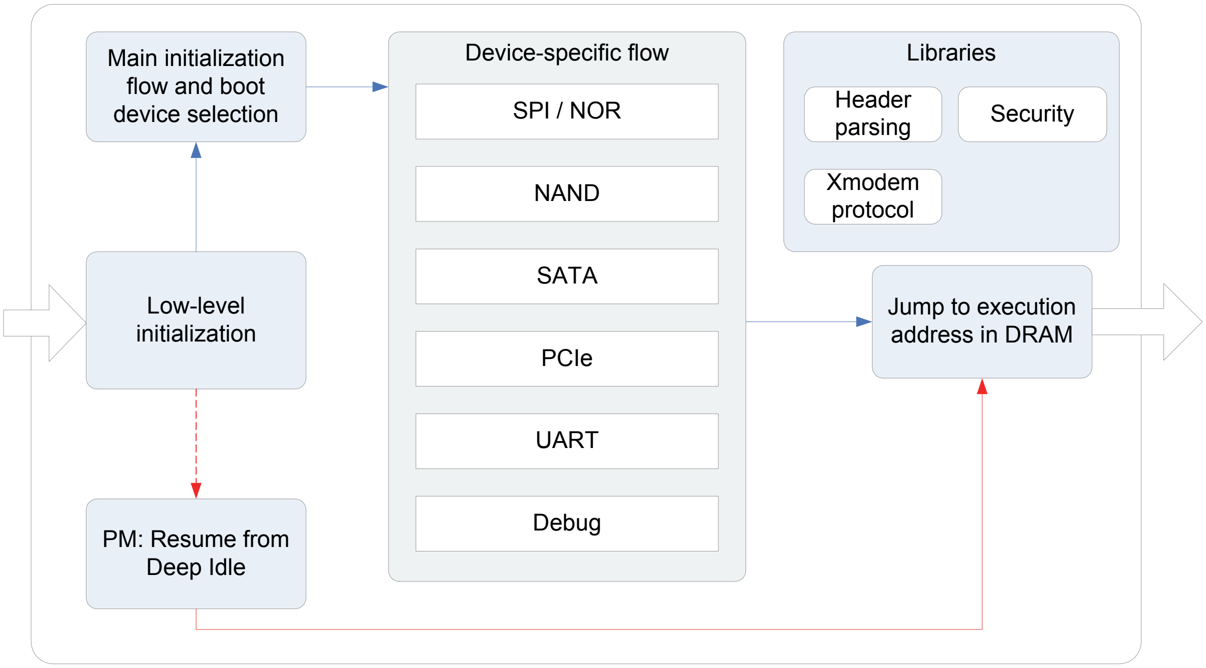

Main initialization flow and boot device selection
Load boot image: Device-specific flow
SPI / NOR
NAND
SATA
PCIe
UART
Libraries
Header parsing
Xmodem protocol
Low-level initialization
Jump to execution address in DRAM
PM: Resume from Deep Idle

Figure 9: BootROM Firmware Block Diagram



6.1 Features

The bootROM firmware provides the following features:

- Basic and fundamental initialization of the device
- Load and execute boot image from the boot device
- Recovery from a backup flash image
- Secured and Non-Secured Boot mode
- Multiple boot devices
- Error Handling
- Power Management
- The main features of the Secured Boot mode include:
- Secured and authenticated boot sequence (optional)
- Public key based cryptographic scheme
- Supported software upgrades and trusted debug
- Flexible chain of trust
- 256 bit embedded authentication key digest, programmed by software on the system, supporting 2048 bit key size.

Copyright © 2014 Marvell

6.2 Functional Description

The bootROM firmware—on the device's integrated bootROM—is executed according to the Sample At Reset (SAR) configuration bits in the Sample at Reset Register (Table 1286 p. 1434).

The bootROM firmware performs basic initialization of the device and loads and executes code from one or more CPUs, that is from one of the following boot devices:

- Serial (SPI) flash
- NAND flash
- NOR Flash
- SATA interface
- PCI Express interface
- UART interface (using the Xmodem protocol)

The device supports Secured Boot mode. In Secured Boot mode the bootROM firmware performs a secured and authenticated boot sequence, based on an industry standard public/private key cryptographic scheme.

6.3 General Considerations

BootROM firmware code was written and compiled to take into account the following:

Endianness BootROM firmware always executes in Little Endian mode. However there are no restrictions on the Endianness of the image booted from the device's interfaces. If the image was compiled to Big Endian mode, it is the responsibility of the image to switch back to Big Endian mode.

ARM/Thumb Mode Most of the bootROM firmware code is compiled to run in Thumb mode, due to code size considerations. However, bootROM firmware switches back to ARM mode before booting an image from one of the device's interfaces.

6.4 Address Decoding and Memory Management Unit (MMU) Operations

For performance enhancement purposes, the bootROM performs the following:

- Enables the L2 Cache as SRAM (L2-SRAM)
- Enables the L1 Instruction Cache
- Enables the MMU: Page table is located in the SRAM
- Enables the L1 Data Cache

Once the MMU is enabled, the 4 GB memory space is accessed using virtual addresses. Table 11 is a virtual-to-physical address translation table. The bootROM uses the first level descriptors (PTEs) with 1 MB resolution. All regions unspecified in the table below are inaccessible. Any access to these regions will cause an exception.

Note

The first 16KB of L2-SRAM are occupied by the MMU translation table. As result, the address used for the headers block copy begins at the L2-SRAM base address + 0x4000. This offset must be taken into account when calculating the Binary Header code location.

Table 11: MMU Virtual-to-Physical Address Translation Table

Device	Physical Address	Virtual Address	Size	Caching
SDRAM	0x0	0x0	1 GB	Non-Cacheable
L2-SRAM	0x40000000	0x40000000	1 MB	Cacheable
PCI Express	0x80000000	0x80000000	640 MB	Non-Cacheable
CESA SRAM	0xC8000000	0xC8000000	1 MB	Non-Cacheable
Internal register	0xD0000000	0xD0000000	1 MB	Non-Cacheable
NOR Flash	0xD8000000	0xD8000000	128 MB	Non-Cacheable
SPI Flash	0xD4000000	0xD4000000	64 MB	Cacheable
Device Bus	0xD3800000	0xD3800000	711 MB	Non-Cacheable
BootROM	0xFFF00000	0xFFF00000	1 MB	Cacheable

6.5 Boot Image Format

The boot image is the binary image residing on the specific boot device. It must contain a valid main image header at offset 0x0 and may include a header extension, according to the header extension flag in the Main header. In addition, it includes the binary image to be copied and executed in the SDRAM (excluding boot from SPI/NOR Flash when the image is executed directly from the external memory space).

The following Extension headers may be present between the Main header and the binary image:

Secured header: If present, immediately follows the Main header and contains the RSA public key and the RSA signatures for all of the header blocks and the binary image. The header cannot be omitted if the device is configured for Secured Boot mode operation by eFuse.

Register set header: Includes address-value pairs for device registers configuration prior to execution of the binary image. This header usually includes DRAM registers configuration and can be omitted if the binary image is decompressed and executed directly from the external flash device.

Binary header: Contains a list of parameters and ARM machine code to be executed before the binary image. This header can be omitted.

NOTE: When booting from UART the device can use the binary header in the image.

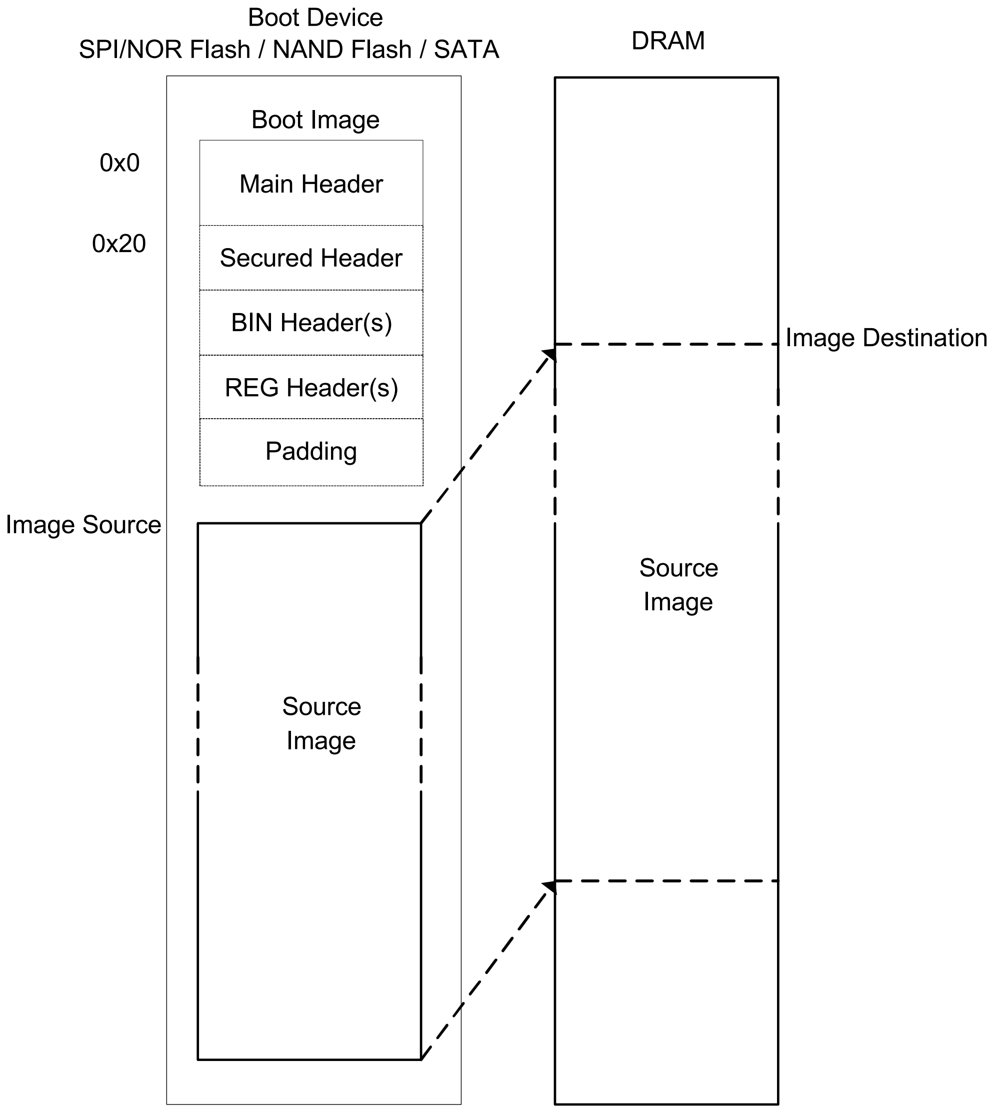
The headers block (Main header with all Extension headers) is padded to the size of the flash page (for NAND flash boot source).

The source image can immediately follow the headers block, or may reside at non-zero offset from the header block (see Figure 10). This option is necessary in boot from SATA or for NAND flash page alignment.

Note

- The order of headers in the header block is application-specific and flexible.
- The image header block can contain several BIN and REG headers.
- The headers are executed in the same order as they encountered in the boot image.
- These facts should be taken into account by the image preparation applications. For example, if the DDR3 memory setup requires setting up specific SDRAM Controller registers (REG) prior to running memory training procedure (BIN), the REG header in such boot image should be placed ahead of the BIN headers.

Figure 10: Binary Image Layout in the Boot Device



6.5.1 Main Header Format

The Main header is 32B long and is in Little Endian mode. Its content differs according to the desired boot method and the device where the header is located. Table 12 outlines the Main Header format.

Table 12: Main Header Format

Byte	Field	Description / Usage
0x0	Identifier	0x5A = Boot from SPI/NOR flash 0x69 = Boot from UART0 0x78 = Boot from SATA device 0x8B = Boot from NAND flash 0x9C = Boot from PCIe interface
0x1–0x3	Reserved	Must be 0x0.
0x4–0x7	Block size	Image size in bytes to be downloaded to DRAM.
0x8	Version	Boot image format version—must be 0x1.
0x9	Header Size MSB	Header block size MSB (in bytes)
0xA–0xB	Header Size LSB	Header block size LSB (in bytes)—the total header size including the Main header and all Extension headers supplied with the image.
0xC–0xF	Source address	• Boot from SPI/NOR flash and NAND flash: Image offset in bytes from the beginning of the flash device used for boot. For NAND flash devices, this address must be aligned to the boundaries of 512B. Necessary for the ECC calculation. • Boot from SATA: Image LBA location offset (in sectors) in the hard drive used for boot. • Boot from UART0: Image location offset in bytes from the file transferred by the Xmodem. • Boot from PCI Express: This parameter is not used, since the Root Complex device initiates the transfer of the image to the DRAM. Set to 0xFFFFFFFF.
0x10–0x13	Destination address	Destination address in DRAM where the image will be copied. • For boot from SPI/NOR flash, If this address equals 0xFFFFFFFF, the image is not downloaded to DRAM. • For Boot from PCI Express, this field indicates the start address of the image (Starting from this address, the 32-bit checksum is calculated and verified).
0x14–0x17	Execution address	Address (point of entry) from which to start executing the image. This address is usually on the DRAM. However, if the destination address is set to 0XFFFFFFFF, then the execution address is located on the boot device. The only valid boot devices are the SPI/NOR flashes.
0x18	Reserved	Must be 0x0.
0x19	NAND flash block size	The size of NAND flash block in 64-KB units (that is, for 128-KB blocks, this field should be 0x2). When this field is set to zero, the default block size is used. This default size is defined by the NAND flash page size (16 KB for a 512B page or small page NAND and 64 KB for a large page NAND flash).
0x1A	Bad block location	Bad block indicator location: 0x0 = Page 0 or page 1 0x1 = Last page in the block

Copyright © 2014 Marvell

Table 12: Main Header Format (Continued)

Byte	Field	Description / Usage
0x1B–0x1D	Reserved	Must be 0x0.
0x1E	Header extension	0x1 = Additional header follows the Main header. 0x0 = No extra header exists (relevant for SPI/NOR flash boot device only).
0x1F	Checksum	8-bit checksum of the headers block (Main header + all additional headers included with the boot image) The checksum is calculated by adding together every 8 bits of data. Each time the sum exceeds 0xFF, it restarts at zero (for example, 0xF0 + 0x12 = 0x02).

When byte 0x1E of the Main header does not equal 0x0, the Extension header begins immediately after the Main header.

The following types of Extension headers are supported:

- Secured Header Format
- Binary Header Format
- Register Set Header Format

Secured header is mandatory for Secured Boot mode and optional for Unsecured Boot mode.

Note

The boot image in Secured Boot mode may be either encrypted or not encrypted. The encryption flag is located in Secured header.

The same image that was prepared for Secured Boot mode can be used for Non-Secured Boot mode if the image itself is not encrypted, since in the Non-Secured Boot mode flow the Secured header is bypassed (not parsed).

Register Set Header and Binary Header are optional (although typically used).

6.5.2 Secured Header Format

The Secured header is only present in images destined to boot from a secure-enabled SoC. The header includes all the information required for image and headers block verification, and cannot be omitted if secured boot is required by the eFuse. Table 13, Secured Header Format, on page 99 outlines the Secured Header format.

Table 13: Secured Header Format

Byte	Field	Description / Usage
0x0	Header Type	Must be 0x1.
0x1–0x3	Header Length	Must be 0x424.
0x4	Encryption	Indicates if the binary image is encrypted or not using AES-128-CBC algorithm and AES key stored in hidden eFuse. 0x1 = Image is encrypted. 0x0 = Image is NOT encrypted.
0x5–0x7	Reserved	Must be 0x0.
0x8–0x213	RSA-2048 Public Key	The RSA Public Key used for RSA signatures creation in BER encoding (524B = 256 (N) + 256 (Exp) + 12 (BER overhead)).

Table 13: Secured Header Format (Continued)

Byte	Field	Description / Usage
0x214	JTAG Enable	A value different than 0x0 enables JTAG (for debugging). The value itself indicates the delay time for JTAG synchronization in seconds.
0x215–0x217	Reserved	Must be 0x0.
0x218–0x21B	Box ID	Must be the same value programmed in the Box ID eFuse field. This value is ignored if the eFuse Box ID is programmed to 0x0.
0x21C–0x21D	Flash ID	Must be the same value programmed in the FlashID eFuse field. This value is ignored if eFuse Flash ID programmed to 0x0.
0x21E–0x21F	Reserved	Must be 0x0.
0x220–0x31F	Header Signature	Header block RSA signature.
0x320–0x41F	Boot Image Signature	Boot image signature.
0x420	Next Header	Indicates if more headers follow this header. 0x1 = Next header exists. 0x0 = This is the last header.
0x421–0x423	Reserved	Must be 0x0.

6.5.3 Binary Header Format

The Binary (BIN) header can be used for running an arbitrary set of commands prior to passing control to the binary image that follows the headers block. The code included in this header must be implemented using ARM instructions and linked accordingly, to support running directly from an image header loaded into RAM without additional relocation. Table 14, Binary (BIN) Header, on page 101 outlines the BIN Header format.

The BIN header's purpose is more than just DRAM controller initialization. It may be used, for example, to change memory windows or Device Bus settings.

Using multiple BIN headers inside the single headers block is allowed. Typically, the binary header is used for DDR3 configuration, but one may use this header for other purposes as well.

The Binary Header code is executed “in place”, without being copied into a pre-defined DRAM address. The DRAM cannot be initialized during the Binary Header execution.

Therefore, the header preparation software must take into account the actual location of the code in the device's L2 SRAM. This location depends on the Main Header size and start address, the amount and size of all headers located between the Main Header and the Binary Header, as well as the number of the Binary Header parameters.

Additionally, the ARM code inside the BIN header must always be aligned with the 128-bit boundary. The alignment requirement can be met by inserting some dummy parameters into the Binary Header, if such is needed.

Table 14: Binary (BIN) Header

Byte	Field	Description / Usage
0x0	Header Type	Must be 0x2.
0x1–0x3	Header Length	The header length in bytes. The value must be aligned to 4B.
0x4	Number of Arguments	The number of binary function arguments included in the header.
0x5–0x7	Reserved	Must be 0x0.
0x8–(0x7 + 0x4 * Number of Arguments)	Arguments list	• If number of arguments is 0, then this field does not exist in the header. • If the number of arguments is not 0, then each argument is 4B long and the field occupies offsets from 0x8 to (0x7 + 0x4 * N) where N - is the number of arguments. For instance: - If N=1, then this field occupies bytes 0x8–0xB - If N=2, then this field occupies bytes 0x8–0xF, and so on.
Follows Arguments List	Binary	Raw executable without any additional headers (like ELF). The binary must be 4B aligned.
Follows Binary	Next Header	Indicates if more headers follow this header. 0x1 = Next header exists. 0x0 = This is the last header.
Last 3 Bytes of the Header	Reserved	Must be 0x0.

6.5.4 Register Set Header Format

The main purpose of the Register Set (REG) header is to provide initialization values to the DRAM memory controller. Table 15, Register Set (REG) Header Format, on page 101 outlines the Register Set header format.

The DRAM must be accessed by the executable BIN header or boot loader image (the image is always loaded into DRAM, except for direct boot from SPI/NOR Flash) after the DRAM have been initialized by the appropriate REG header and optional BIN header for DDR3 initialization sequence flow.

When the Image Destination Address field of the Main header is set to 0xFFFFFFFF, the boot image may or may not contain a REG header, and the system is instructed to boot directly from the SPI/NOR Flash. The REG header's purpose is more than just DRAM controller initialization. It may be used, for example, to change memory windows or Device Bus settings.

Using multiple REG headers inside the single headers block is allowed.

Table 15: Register Set (REG) Header Format

Byte	Field	Description / Usage
0x0	Header Type	Must be 0x3.
0x1–0x3	Header Length	The header length in bytes. The value must be 4B aligned.
0x4–0x7	Address 0	Register address.
0x8–0xB	Value 0	Value to be written to the addressed register.

Copyright © 2014 Marvell

Table 15: Register Set (REG) Header Format (Continued)

Byte	Field	Description / Usage
0xC–0xF	Address 1	
0x10–0x13	Value 1	
0x4 + 8*N - 0x7 + 8*N	Address N	
0x8 + 8*N - 0xB + 8*N	Value N	
0xC + 8*N	Next Header	Indicates if more headers are following this header. 0x1 = The next header exists. 0x0 = This is the last header.
0xD + 8*N	Delay	The delay, in milliseconds, to be kept after this header execution completion, prior to continuing the processing of the remainder of the headers. If the delay field is cleared to 0x0, the bootROM firmware assumes that the header was used for SDRAM controller setup, and at the end of header execution, instead of a delay, it polls the SDRAM Initialization Control Register (Table 467 p. 791) for clearance of Initialization Enable bit [0]. Since the DRAM controller needs to be initialized only once, make sure that in the entire header block, only one REG header is set to delay=0. That header should also set the above mentioned DRAM Initialization Enable bit as a part of its Address-Value settings.
0xE + 8*N - 0xF + 8*N	Reserved	Must be 0x0.

6.5.5 Source Image Considerations

BootROM firmware assumes the following for the images that can be booted from the device's interfaces:

- Image base address at the specific device interface is aligned to 32 bits. (For NAND flash boot, the image must be aligned to the boundaries of the NAND page size.)
- Image size including the checksum is aligned to 32 bits.
- If the image will be downloaded to the DRAM, the destination offset on the DRAM is aligned to 32 bits.
- The image includes a simple 32-bit checksum in the last 4B.
- The first instruction of the image is in Little Endian mode. If the image will be executed in Big Endian mode, the image must include the appropriate code to switch back to Big Endian mode.
- The first instruction is in ARM mode.

6.6 BootROM Firmware Boot Sequence

The bootROM firmware boot sequence of the first CPU can be divided into the following phases, as described in the sub-sections below:

1. Initialization
2. Checking serial ports for a boot sequence
3. Boot mode selection
4. Load, check, and update the boot image header block.
5. In Secured Boot mode, verify the public key and the headers block signature.

6. If Extension headers exist, for each header:
   a) Check the type of Extension header.
   b) If a REG header is detected, initialize the registers using the values read from the header.
   c) If a BIN header is detected, execute the header binary with its parameters.
7. Load and check the device image (unless it is to be executed from the SPI/NOR Flash).
8. In Secured Boot mode, verify the image signature and decrypt the image, as needed.
9. Execute the device image code.

In addition, bootROM firmware contains the following functionality:

- Boot stages monitoring using the terminal on UART0
- Error reporting and handling
- Exception handling

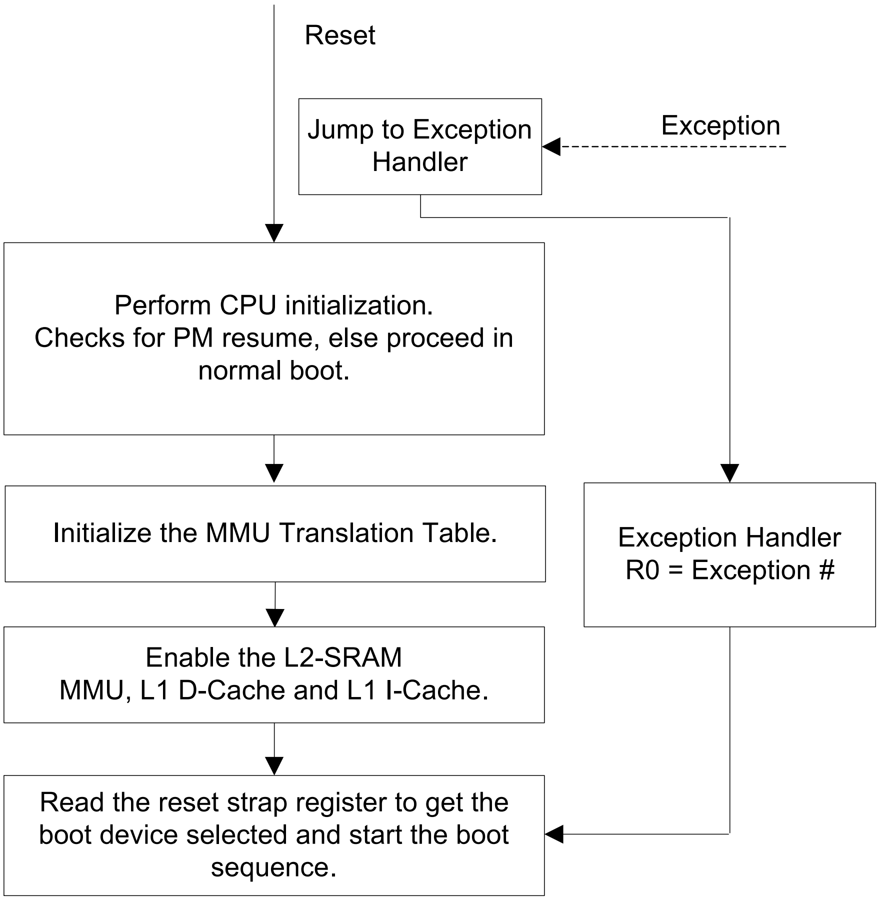
6.6.1 Initialization

After reset, the CPU starts running at 0xFFFF0000, where the bootROM firmware code is located. It performs the following operations (in Assembly language):

1. Performs the initial CPU configurations.
2. Checks for Power Management Resume state.
3. Configures L2 in SRAM mode.
4. Initializes the MMU Translation table.
5. Enables MMU, L1 D-Cache, and L1 I-Cache.
6. Starts the boot sequence, by detecting the selected boot device.

This process is illustrated in Figure 11.

Figure 11: Initialization and Boot Method Selection Flow


Reset
Jump to Exception Handler
Exception
Perform CPU initialization. Checks for PM resume, else proceed in normal boot.
Initialize the MMU Translation Table.
Exception Handler R0 = Exception #
Enable the L2-SRAM MMU, L1 D-Cache and L1 I-Cache.
Read the reset strap register to get the boot device selected and start the boot sequence.

6.6.2 Boot Device Selection

The Boot device is selected in the main routine, executed directly after initialization of the MMU and caches and the initial scan of serial ports for a boot command.

The following sequence of operations is executed:

1. Calls the UART Initialization routine, initializing the UART0 to the 115,200-8N1 baud rate (8N1 means 8-bits of Data, No parity, 1 stop bit) according to the detected value of the TCLK.
2. If the main routine was executed as the result of an exception, calls the Error Handler.
3. Calls the Execution Handler.

Execution Handler

This routine reads the Sample at Reset Register (Table 1286 p. 1434) and calls the appropriate routine, according to the value of the SAR [Boot Device] field. Also refer to *Boot Device* in the Reset Configuration table of the *Hardware Specifications* for more information about the SAR boot settings.

If the Execution Handler returns an error code, the error is registered in the BootROM Routine and Error Code Register (Table 1464 p. 1537).

Copyright © 2014 Marvell

6.6.3 UART Booting

There are two modes of operating the UART interfaces that can be either:

- Based on the Sample at Reset Register (Table 1286 p. 1434) and Sample at Reset High Register (Table 1287 p. 1434))
  or
- Triggered by detection of a special command sequence received from one of the serial lines.

Note

Neither of these two modes is available when Secured Boot mode is enabled in eFuse.

When the UART mode is selected by SAR, the system always uses the default UART port (UART0) for loading the boot image.

The default serial port assignment, however, can be changed. This change occurs if a special command sequence is received over one of UART ports when the bootROM monitors the serial ports for activity. There are two cases when all available UART interfaces are monitored for reception of the command sequence:

- **Short check:** Before accessing the dedicated boot device that is configured by SAR
- **Fail recovery:** Endless port scans followed by failure to boot from the configured boot device

Practically the first case can be used for fast system recovery if the default boot device failed for some reason or even for testing a new boot image prior to modification of the boot device content.

To take control of the system, send a long sequence of the same commands over a serial line when the system exits reset state. The first valid command that is detected by the bootROM will define its future behavior.

Once a valid command sequence is detected, the port that received the sequence switches to Xmodem Protocol mode and becomes ready for loading the boot image.

Since a serial port input can also be connected to a noisy source, the bootROM will stop specific UART monitoring if more than 24 invalid (non-command) symbols are received from it.

When the bootROM enters fail recovery UART monitoring, the message “Trying UART” is sent to the default serial port (UART0).

Note

- All UART settings during port monitoring stage are the same and similar to the default port (UART0) settings—8N1, 115200 baud.
- RX lanes of all ports that are activated during the scan are de-activated for ports that were not selected for UART Boot mode.
- TX lanes are not active during the port scan, and they are only activated for a specific port if the input command instructed it to activate the port.
- TX lanes of the default port remains active all the time for the boot messages. The boot monitoring messages are always sent to the default serial port. This assignment cannot be changed by the input command.

6.6.3.1 Boot from UART

The Boot from UART is activated when the following 8B command sequence is detected on one of available serial ports:

0xBB, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77.

The monitored port that receives this command first is switched to the Boot mode, instead of the default serial port and starts the Xmodem protocol for loading the boot image.

If SAR requests boot from UART, the bootROM configures UART0 and starts the Xmodem protocol for loading the boot image from the default serial port (UART0) to the system memory (DRAM).

Note

Boot from the UART interface is not available if Secured Boot mode is selected by eFuse.

6.6.4 eFuse Structure

The device contains a 310-bit one time programmable module (eFuse) that utilizes the boot firmware to enable Secured Boot mode.

This eFuse consists of the following fields:

- **Public Key Digest:** A 256-bit digest of the RSA public key that is used during the boot firmware authentication phase. It is based on FIPS 180-2 (SHA-256).
- **Box ID:** A 32-bit unique Box ID that may be optionally used to associate an image to a specific product. It is Typically used to prevent leakage of debug image.
- **Flash ID:** A 16-bit field that can be used to differentiate products by their type of service using different images.
- **Secured Boot Enable:** A 1-bit field, indicating whether the boot sequence is Secured Boot mode or Non-Secured Boot mode.

  1 = Secured Boot mode

  0 = Non-Secured.Boot mode
- **JTAG Enable:** A 1-bit field, indicating whether JTAG is enabled or disabled, during the boot sequence (Debug mode).

  1 = JTAG Enabled

  0 = JTAG Disabled
- **Boot Source Device:** This 4-bit field overrides the reset strap, if the secured boot is enabled, and it determines the boot source device.

6.6.5 Non-Secured Boot Flow Description

Each of the boot methods is described in detail in Section 6.7, BootROM Firmware Boot Options. However, all boot methods use the same logic to read or decode the header, and to execute the loaded image.

Each time an error report appears in the bootROM execution flow for SPI, NOR, or NAND flash devices, the recovery procedure is executed. The recovery procedure increments the byte offset from the boot device start, and restarts the entire boot sequence, trying to read the boot headers and image from the new position defined by this offset value. The process is stopped when the maximum offset value is reached.

This mechanism allows, for example, recovery from interrupted boot image upgrades due to power failure or other interruptions. Prior to overriding the working image, the application must copy the old working boot loader image from the boot device start to any flash offset from 4 MB to 64 MB for NAND (in 4-MB increments), and 2 MB to 16 MB for SPI/NOR flash (in 2-MB increments). If an error

Copyright © 2014 Marvell

is unrecoverable, the message is logged on the UART0 interface and the CPU is halted. An example of an unrecoverable error is a wrong bootstrap configuration.

The boot flow is as follows (see Figure 12, "Boot Sequence):

1. **Main Header decoding:**

   a) Read the entire header block from the boot device into the stack residing in the L2-SRAM (for NAND flash boot, at least 512B of data are loaded, since ECC is calculated over chunks of 512B).

   b) Checksum is calculated and verified on the header block.

   c) If the checksum is valid, verify the header ID. If either is invalid, report an error.

2. **Header Extension decoding:**

   a) If byte 0x1E of the Main header is 0x0, no Extension header exists, and no register configuration occurs. Proceed to step 4.
   If byte 0x1E of the Main header is not 0x0, an Extension header exists.

   b) Check the type of Extension header.

   c) If the type is REG or BIN, execute the Extension header.

   d) If the header type is unknown, report the error.

   e) If the Extension header is not the last one (its Next Header field is not 0), start the next header processing from step 2b.

3. **Image Loading and verification:**

   For boot from SATA, NAND flash, NOR flash or SPI flash (with image copy to DRAM), the image must be loaded to the DRAM and executed from there.

   The following sequence is performed:

   a) Copy the image from the Source Address indicated in bytes 0xC–0xF of the Main header to the Destination Address indicated in bytes 0x10–0x13. The size of the image to be copied is indicated in bytes 0x4–0x7. (This size includes the extra 4B of the checksum-32.)
   NOTE: For NAND flash and SPI/NOR flash, the Source Address is the offset in bytes from the start of the device (where the Main header resides).
   For SATA, the Source Address is the LBA address (the number of the first sector of the partition containing the image).

   b) Calculate the checksum-32 on the entire image (in the DRAM), without the last 4B, and compare the result to the last 4B of the image.

   c) If the calculated checksum-32 and the recorded checksum-32 (following the image) match, continue to the next step. Otherwise, report the error.

   For boot from SPI flash or NOR flash, the image can optionally be executed from the same location, without copying it to the DRAM. The following sequence is performed:

   a) The destination must be 0xFFFFFFFF, as an indication that no copy is necessary and the image must be executed in place (i.e., executed directly from the flash, using the directly mapped memory space).

   b) The Source Address indicates the offset in bytes (from the beginning of the flash, where the Main header resides) at which the image starts.

   c) Calculate the checksum-32 on the entire image (on the flash) without the last 4B, and compare the result to the last 4B of the image.

   d) If the calculated checksum-32 and the recorded checksum-32 (following the image) are a match, continue to the next step. Otherwise, report the error.

For boot from PCI Express, the image is loaded by the Root Complex and no copying is necessary. The following sequence is performed:

NOTE: Only PCIe0 is initialized for loading the boot image, regardless of the actual number of PCIe interfaces available.
Boot from PCIe is not available in Secured Boot mode.

a) The bootROM starts by initializing the with 0xFFFFFFFF.
It then performs PCI Express interface initialization, setting it to Endpoint mode.
Next, it enters an infinite loop, waiting for the Root Complex to change the and update it with the address holding the image header. This is a handshake, indicating that the Root Complex has completed the necessary DRAM configuration and downloaded the image to the execution location.

b) When the has been updated with the location of the Main header address, the bootROM verifies the Main header.
NOTE: The Source Address in the header is **not used** if the boot device is PCI Express.

c) The Destination Address specifies the location of the image.

d) The bootROM calculates the checksum-32 on the image in the location specified in the Destination Address.

e) If the calculated checksum-32 and the recorded checksum-32 (following the image) are a match, continue to the next step. Otherwise, report the error.

4. **Image execution:**

a) Disable MMU, I-Cache, and D-Cache.

b) Disable L2 cache.

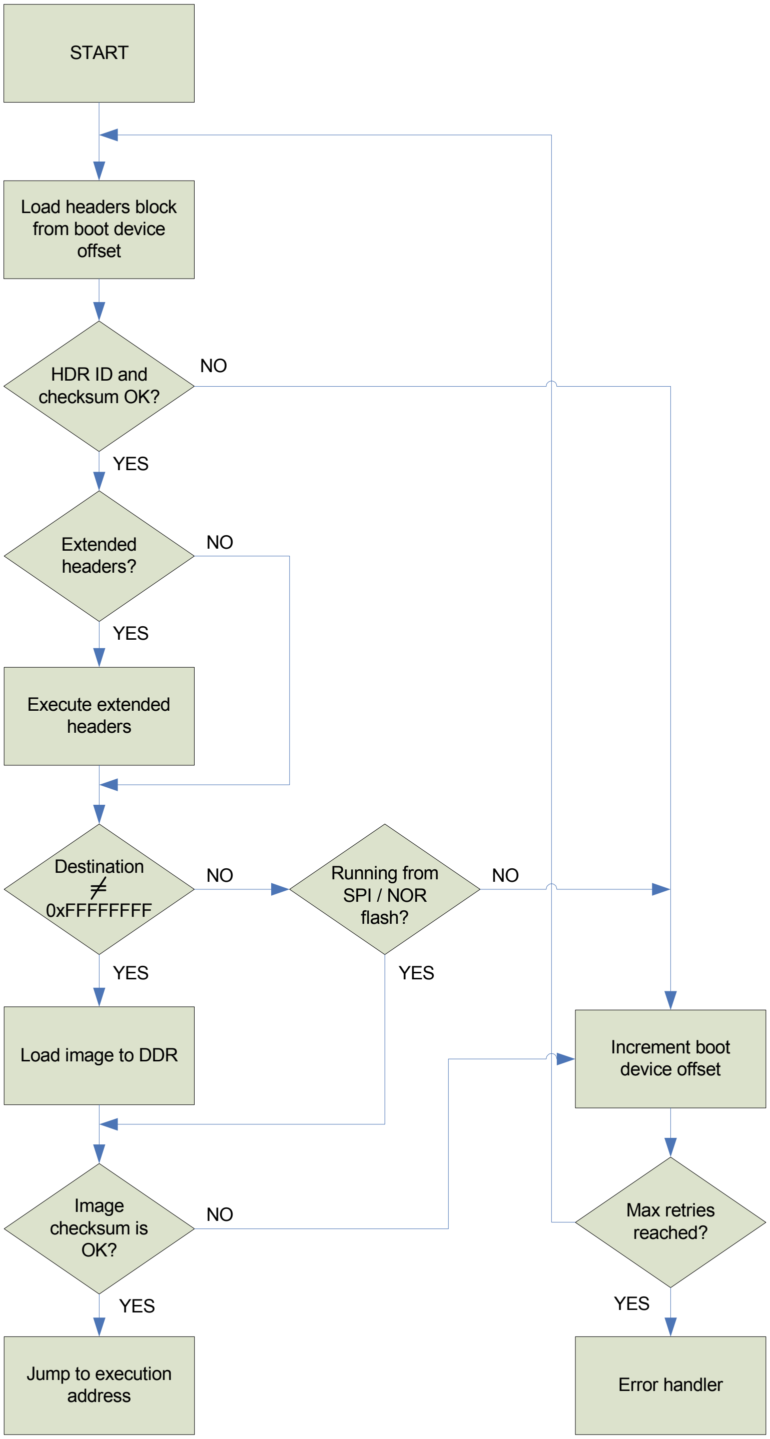
c) Jump to the execution address that was extracted from the Main header.

Note

If any error occurs during the entire process, it is registered in the <Error Code> field in the BootROM Routine and Error Code Register (Table 1464 p. 1538), and the entire process is restarted from the early boot stages, where the sample at reset configuration is read from the bootstrap register. The Error Code is updated on the first error reported. Successive errors will not affect the reported error code.

Copyright © 2014 Marvell

Figure 12: Boot Sequence



6.6.6 Secured Boot Flow Description

Each of the boot device is described in detail in Section 6.7, BootROM Firmware Boot Options. However, all boot devices use the same logic to read or decode the header, and to execute the loaded image.

The first procedure performed, in the bootROM, is reading the value of the eFuse Boot mode field. If Secured Boot mode is requested by eFuse, the boot flow is executed as described below.

Each time an error report appears in the bootROM execution flow for SPI/NOR or NAND flash devices, the recovery procedure is executed. The recovery procedure increases the byte offset from the boot device start, and restarts the entire boot sequence, trying to read the boot headers and image from the new position defined by this offset value. The process is stopped when the maximum offset value is reached.

This mechanism allows recovery from interrupted boot image upgrades due to power failure or other interruptions. If an error is unrecoverable, the message is logged on the UART0 interface and the CPU is stopped. An example of an unrecoverable error is a incorrect bootstrap configuration. Prior to overriding the working image, the application must copy the old working boot loader image from the boot device start to any flash offset from:

- 4 MB to 64 MB for NAND (in 4-MB increments)
  and
- 2 MB to 16 MB for SPI/NOR flash (in 2-MB increments).

The Secured boot flow, after reading the boot mode configuration from eFuse, (see Figure 13, Secured Boot Sequence, on page 112) is as follows.

1. **Main Header decoding:**

   a) Read the header block from the boot device into the stack (for NAND flash boot, at least 512B of data are loaded, since ECC is calculated over chunks of 512B).

   b) Checksum is calculated and verified on the header block.

   c) If the checksum is valid, verify the header ID. If either is invalid, report an error.

   NOTE: The Header Extension field in the Main header must not be zero.

2. **Secured Header decoding:**

   a) Calculate the SHA-256 for the RSA public key included with the header and compare it to the SHA-256 digest programmed to eFuse. If the digest comparison fails, report an error.

   b) Calculate the RSA signature for the entire header block and compare it with the header signature supplied in the security header. If the signature comparison fails, report an error.

   c) If the header Flash ID and Box ID are not zero, compare them to the appropriate values programmed in eFuse. If the ID comparison fails, report an error.

   d) If the Box ID and JTAG fields in the Secured header are both not zero, delay execution by the JTAG field value in seconds, and enable the JTAG interface.

3. **Header Extension decoding:**

   a) If the next header field of the Secured header is 0x0, no Extension header exists, and no register configuration occurs. Proceed to step 4.
   If the next header field of the Secured header is not 0x0, an Extension header exists.

   b) Check the type of Extension header.

   c) If the Extension header type is REG or BIN, execute the Extension header.

   d) If the Extension header type is unknown, increment the boot device offset, and if the maximum number of retries is not reached, start from step 1. Otherwise, stop and report the error.

   e) If the Extension header is not the last one (its Next Header field is not 0), start the next header processing from step 3b because the image may have several Extension headers.

4. **Image Loading and verification:**

> **Note**
>
> Boot from SPI or NOR flash can be accomplished either by image coping to DRAM or *without* copying the image coping to DRAM, as explained in the two lists of steps below.

For boot from SATA, NAND flash, SPI, or NOR flash (with image copy to DRAM), the image must be loaded to the DRAM and executed from there. The following sequence is performed:

a) Copy the image from the Source Address indicated in bytes 0xC–0xF of the Main header to the Destination Address indicated in bytes 0x10–0x13. The size of the image to be copied is indicated in bytes 0x4–0x7. (This size includes the extra 4B of the checksum-32.)

> **Note**
>
> - **For NOR flash and SPI flash**, the Source Address is the offset in bytes from the start of the device (where the Main header resides).
> - **For SATA**, the Source Address is the LBA address (the number of the first sector of the partition containing the image).

b) Calculate the checksum-32 on the entire image (in the DRAM) without the last 4B, and compare the result to the last 4B of the image.

c) If the calculated checksum-32 and the recorded checksum-32 (following the image) are a match, continue to the next step. Otherwise, report an error.

d) Create an RSA signature of the binary image located in the DRAM, and compare it to the signature supplied in the Secured header. If both signatures are equal, continue to the next step. Otherwise, report an error.

e) If the Encryption field of the Secured header is not zero, decrypt the binary image in the DRAM using the AES key from the second eFuse.

For boot from SPI or NOR flash, the image can optionally be executed from the same location, without copying it to the DRAM. The following sequence is performed:

a) The destination must be 0xFFFFFFFF, as an indication that no copy is necessary, and the image must be executed in place (executed directly from the flash, using the mapped memory space).
The Source Address indicates the offset in bytes (from the beginning of the flash, where the Main header resides) at which the image starts.

b) Calculate the checksum-32 on the entire image (on the flash) without the last 4B, and compare the result to the last 4B of the image.

c) If the calculated checksum-32 and the recorded checksum-32 (following the image) are a match, continue to the next step. Otherwise, report an error.

d) Create an RSA signature of the binary image located in the flash and compare it to the signature supplied in the Secured header. If both signatures are equal, continue to the next step. Otherwise, report an error.

e) If the Encryption field of the Secured header is not zero, report an error.

5. **Image Execution:**

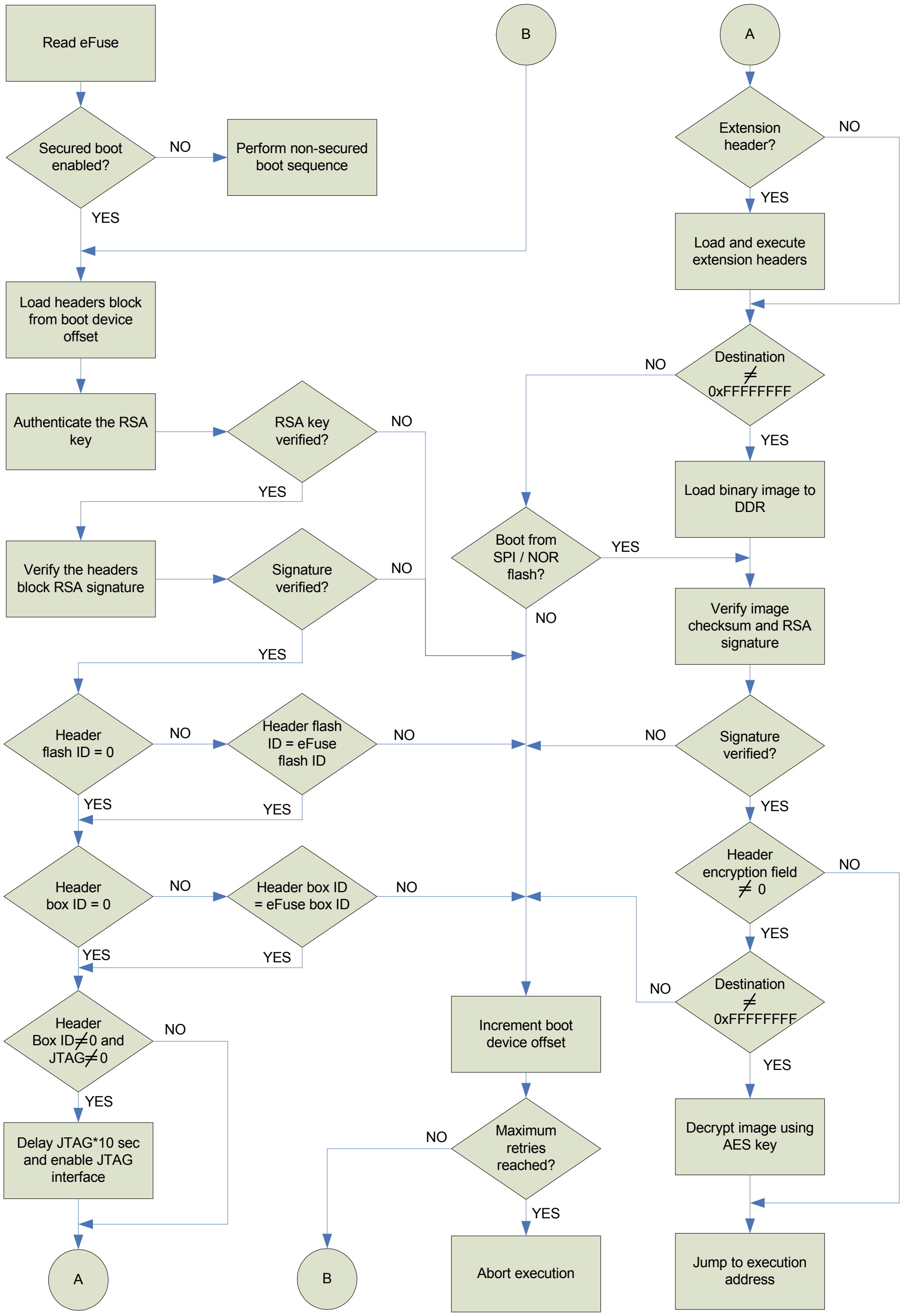
a) Disable MMU, I-Cache, and D-Cache.

a) Disable L2 cache.

b) Flush L2-Cache, I-Cache, and D-Cache.

c) Jump to the execution address that was extracted from the Main header.

Figure 13: Secured Boot Sequence


Read eFuse
Secured boot enabled?
NO
Perform non-secured boot sequence
YES
B
A
Extension header?
NO
YES
Load and execute extension headers
Load headers block from boot device offset
Authenticate the RSA key
RSA key verified?
NO
YES
Destination ≠ 0xFFFFFFFF
NO
YES
Load binary image to DDR
Verify the headers block RSA signature
Signature verified?
NO
YES
Boot from SPI / NOR flash?
YES
NO
Verify image checksum and RSA signature
Header flash ID = 0
NO
Header flash ID = eFuse flash ID
NO
YES
YES
Signature verified?
NO
YES
Header box ID = 0
NO
Header box ID = eFuse box ID
NO
YES
YES
Header encryption field ≠ 0
NO
YES
Destination ≠ 0xFFFFFFFF
NO
YES
Header Box ID≠0 and JTAG≠0
NO
YES
Delay JTAG*10 sec and enable JTAG interface
Increment boot device offset
Maximum retries reached?
NO
YES
Decrypt image using AES key
A
B
Abort execution
Jump to execution address

Copyright © 2014 Marvell

6.6.7 Error Handling

6.6.7.1 BootROM Firmware Error Registers

All boot methods use the same Error Handling mechanism. This mechanism takes advantage of the BootROM Routine and Error Code Register (Table 1464 p. 1537).

6.6.7.2 BootROM Firmware Error Handling Description

Error Handling is performed as follows:

1. On error, the specific boot method calls the Error Handler with the appropriate <Error Code> and <Error Location> representing the current execution method.
2. The Error Handler automatically increments the <Retry_Count> field.
3. If this is the first call to the Error Handler (checked by examining the <Retry_Count> field), it updates the <Error Code> and <Error Location> fields.
4. The bootROM tries to perform the boot method again by jumping to the main routine.

6.6.7.3 BootROM Monitoring

In Secured Boot mode, every step of the boot procedure is reported by Error/Success messages that are sent to the UART0 interface. Boot from UART is disabled in Secured Boot mode. Therefore, sending messages toward the serial console does not disturb the boot operation.

6.6.7.4 BootROM Messages

Information messages

The boot procedure provides the following information messages:

Booting from <boot source>

Possible values for “boot source” field:

- SPI flash
- NOR flash
- NAND flash
- SATA
- PCIe

Trying UART

The message is printed when bootROM returns to Boot from UART mode and starts looking for a boot pattern on the available serial ports.

Pattern detected on UARTx

The message is printed when valid boot pattern is detected on UARTx (where x is the UART port number) before switching to the Boot mode.

Error messages

The bootROM error messages are listed in Table 16.

Table 16: BootROM Error Messages

Message Type	Description	Error Code
CPU cache RW test FAILED	The CPU cache is used instead of DRAM until the DRAM controller gets initialized. This error is severe and further execution cannot continue.	ERROR_MEMORY_INIT
Boot image decryption FAILED	The encrypted image length is not 16-bit aligned.	ERROR_INVALID_IMG_LEN
Image checksum verification FAILED	Boot image checksum verification failed after image copy from the boot device to DRAM. The may occur due to improperly initialized or unstable DRAM or boot device failure.	ERROR_INVALID_IMG_CHKSUM
Boot image signature verification FAILED	The boot image RSA signature computed for the image copied to DRAM differs from the one supplied in the Secured header (Secured Boot mode only). This may occur due to improperly initialized or unstable DRAM, boot device failure, or due to unauthorized image modification (a hacker attack).	ERROR_INVALID_RSA_IMG_SIGN
In-flash image checksum verification FAILED	The boot image checksum computed for an image located in the SPI/NOR flash differs from the checksum supplied with image.	ERROR_INVALID_IMG_CHKSUM
In-flash image signature verification FAILED	The boot image RSA signature computed for the image located in the SPI/NOR flash differs from the one supplied in the Secured header (Secured Boot mode only). This may occur due to boot device failure or due to unauthorized image modification (a hacker attack).	ERROR_INVALID_RSA_IMG_SIGN
Invalid header checksum	The headers block checksum differs from the one supplied as part of the main header. This may occur as a result of improperly initialized, an unstable DRAM, or a boot device failure.	ERROR_INVALID_HDR_CHKSUM
Invalid header version	The header version is not equal to 1.	ERROR_INVALID_HDR_VERSION
Invalid header ID	The header ID is not in sync with selected boot device.	ERROR_INVALID_HEADER_ID
Invalid header alignment	The header block size field value is not 8-bit aligned.	ERROR_ALIGN_SIZE
Invalid HDR destination address	The main header destination address field value is not 8-bit aligned.	ERROR_ALIGN_DEST
Invalid HDR source address	The main header source address field value is not 8-bit aligned and the boot device is not SATA.	ERROR_ALIGN_SRC
Invalid security header size/type	The eFuse requires Secured boot, but the header following the main header is not secured. Its size differs from the defined security header size.	ERROR_INVALID_SECURITY_HDR
Invalid RSA key format	The RSA key is not properly DER-formatted (The first 2B are used for length).	ERROR_INVALID_RSA_PUB_KEY_FMT
Invalid RSA key length	The RSA key length verification failed.	ERROR_INVALID_RSA_PUB_KEY

Table 16: BootROM Error Messages (Continued)

Message Type	Description	Error Code
RSA Public key verification FAILED	The RSA key digest differs from the one saved in eFuse.	ERROR_INVALID_RSA_PUB_KEY
Invalid RSA key modulo length OR Invalid RSA key exponent length	The RSA key modulo/exponent size is not 256B	ERROR_INVALID_RSA_PUB_KEY
RSA error	RSA library initialization error.	ERROR_RSA_LIB_ERROR
Boot header signature verification FAILED	The header block signature differs from the one supplied in the security header.	ERROR_INVALID_RSA_HDR_SIGN
Flash ID verification FAILED	The Flash ID in the security header differs from the one saved in eFuse.	ERROR_INVALID_FLASH_ID
Box ID verification FAILED	The Box ID in the security header differs from the one saved in eFuse.	ERROR_INVALID_FLASH_ID
Bad header <offset value>]	The main header sanity check failed for data obtained from the boot device. If the boot device is a flash memory, the current offset is indicated, and the execution continues trying to find a valid header on the next flash sector.	The error code depends on the reason for the failure: • ERROR_INVALID_HEADER_ID or • ERROR_INVALID_HDR_SIZE
Wrong configuration: encrypted image and direct SPI/NOR boot	The SPI/NOR header indicates that the image is AES encrypted, but direct boot from flash is required (no copy to RAM). Such a configuration is not supported.	ERROR_INVALID_BINARY

6.7 BootROM Firmware Boot Options

6.7.1 Boot from UART0

As previously indicated, boot from UART0 is defined as a sample at reset configuration option. Since the UART0 interface is used for boot image transfer, the bootROM monitoring though the serial console is not active in this mode.

In this case, the boot image must be a continuous image including a Main header and an Extension header. (REG Extension header is mandatory in this case since the DRAM must be initialized prior to loading it with the boot image. When using DDR3, the existence of the BIN header in the boot image becomes mandatory.

The Main header and source image must contain valid checksum values. The Main header checksum includes the checksum for the entire header block.

According to the boot sequence described above, this boot method performs the following sequence:

1. Executes the REG and/or BIN Extension header(s).
2. Downloads the source image to DRAM.
3. Executes it in the DRAM.

6.7.2 Boot from Serial or Parallel Flash

In this boot method, a boot image must be located on the external flash. The Main header must exist at offset 0 (or at any 2 MB offset up to a 16 MB boundary) of the external flash. Extension headers may exist, if the Extension Header bit is set in the Main header.

The headers block and source image must contain valid checksum values. In Secured Boot mode, the headers block and source image are signed by RSA signatures that, in turn, are included in the attached Secured header.

Boot from flash can be performed in two ways, based on the value of the Destination Address specified in the Main header.

- If the Destination Address field holds a value other than 0xFFFFFFFF, the source image is copied to the DRAM address specified in the destination field and executed from the address indicated in the execution address field. In this case, a REG Extension header is mandatory to perform the necessary DRAM initialization prior to loading it. When using DDR3, the BIN header is mandatory.
- If the Destination Address field equals 0xFFFFFFFF, the source image is executed directly from the flash using directly mapped memory space. This option is not available in Secured Boot mode when the image is encrypted using the AES algorithm field in the Secured header is set (see the <Symmetric Key N> field in the Fuse Secured Boot Symmetric Key N Register (N=0–3) (Table 1419 p. 1503)).

The user’s flash burning application (firmware update) should:

1. Burn a backup image at offset 2MB/4MB/6MB/8MB/10MB/12MB/14MB/16MB and verify it (phase 1).
2. Burn the main boot image at offset 0 (phase 2).

   If a power down event occurs during one of these firmware update phases, the system remains bootable, even through the image burn process interrupted in the middle.

6.7.3 Boot from NAND Flash

In this boot method, a boot image must be located on the first page of the NAND flash. The Main header must exist at offset 0 (or at any 4 MB offset up to a 64 MB boundary) and REG and/or BIN Extension headers is mandatory for this boot device to perform DRAM initialization. (The image must be copied to the DRAM prior to execution and cannot be invoked directly from NAND flash.)

- The source image must be located at the offset specified by the Main header.
- The Main header and source image must contain valid checksum values.
- The source image is downloaded to the DRAM byte-by-byte, using a NAND flash software protocol.

Flash type (Large or Small page, number of read cycles) and ECC algorithm (Hamming or BCH) for bootROM are defined by reset strap (see Sample at Reset Register (Table 1286 p. 1434).

6.7.3.1 Bad Block Management

The bootROM supports bad block skipping. Before reading from a block, it is verified to be a good block by checking the appropriate Out of Band (OOB) byte or bytes in the Spare area are 0xFF.

- For 512B page devices, the number of pages per block is fixed at 32 (block size is 16 KB) and the bad block indicator is located in the sixth byte of the OOB area.
- For Large page devices (2 KB and above) the block size and the bad block indicator location are specified at runtime from the header.

Copyright © 2014 Marvell

The bad block indicator location in the OOB is checked by the bootROM based on the type of NAND, block size and bad block indicator location specified in the header. The three possible options are listed in Table 17.

Table 17: Bad Block Indicators per NAND Flash Cell Type

	NAND Flash Type	Read Command Sequence
1	Large page and bad block in the last page	Byte[0] of the spare area in the last page of the block (for a good block, the byte should be equal to 0xFF).
2	Large page and bad block in the first page	Byte[0] and Byte[5] of the spare area in the first and second pages of the block (for a good block, both bytes should be equal to 0xFF).
3	512B page SLC devices	Byte[5] of the spare area in the first and second pages of the block (for a good block, the bytes should be equal to 0xFF).

6.7.4 Boot from a SATA Device

The presence of the REG and/or BIN Extension headers is mandatory for performing DRAM initialization (since the image must be copied to the DRAM prior to execution and cannot be invoked directly).

In this boot device, the Main header must be located in sector number 1 of the hard disk (since sector 0 holds the partition table).

The source image must be located at the beginning of the sector specified by the Main header. (Usually the first partition starts in sector 63.)

This boot method downloads the source image to the DRAM and executes it from there. The source image is downloaded to DRAM using the DMA mechanism.

Note

The image destination address should not point to the first 2 KB of the system DRAM, if the SATA device is used for boot.

6.7.5 Boot from PCI Express Interface

This boot device differs from the other four boot devices. The device functions as a PCI Express Endpoint, and the Root Complex is responsible for performing the basic steps of the boot process.

In this boot mode, the first task the bootROM performs is to set the PCI Express Boot Address Register (Table 1441 p. 1516) with the value 0xFFFFFFFF.

Then the bootROM initializes the PCI Express interface and configures the PCI Express controller to function as an Endpoint. By default, the device configures its BARs to allow access to its internal registers. This allows the Root Complex to use BAR 0 to configure the device, initialize the DRAM (if it exists), and set up the DRAM BAR 1 and windows, if necessary.

After performing the interface initialization, the bootROM enters an infinite loop, waiting for the Root Complex to change the PCI Express Boot Address Register and update it with the address (locally on the DRAM) holding the image header. This is a handshake mechanism, indicating that the Root Complex is ready, having performed the necessary configurations and downloaded the image to the execution location on the DRAM.

After detecting the change in the PCI Express Boot Address Register, the bootROM uses the value passed as the location of the Main header and verifies the header checksum to make sure that it is valid.

The bootROM uses the Destination Address as the absolute location of the image to check the image checksum-32.

Finally, if the checksum is valid, the bootROM jumps to the Execution Address specified in the Main header and starts running from there.

6.8 BootROM Behavior When Recovering from Deep Idle Mode

During the normal boot process, the bootROM only activates CPU0. Any additional CPUs must be handled by the second stage boot loader. However, when recovering from Deep Idle mode, each CPU is activated separately, since each CPU has its own stack and execution pointer.

MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Part 2: CPU Subsystem

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. **Moving Forward Faster**

Document Classification: Proprietary Information

THIS PAGE IS INTENTIONALLY LEFT BLANK

7 Marvell® Core Processor

The device uses the Marvell® Core Processor ARMv7 implementation in ARMv7, compliant with the ARMv7 architecture.

For full details and specifications about the Marvell® Core Processor, refer to the *ARMADA® XP MP Core Highly Integrated Marvell® ARMv7 SoC Processors Datasheet*, Doc. No. MV-S108492-00.

The CPU registers are located in Appendix A.2, CPU Sub-System Registers, on page 600.

8 Level-2 Cache

The Level 2 (L2) cache is a high-performance memory subsystem add-on, for the MV78230/78x60.

The addition of a secondary cache, also referred to as a L2, is a recognized method of improving the performance of processors-based systems, when significant memory traffic is generated by the processors and I/O.

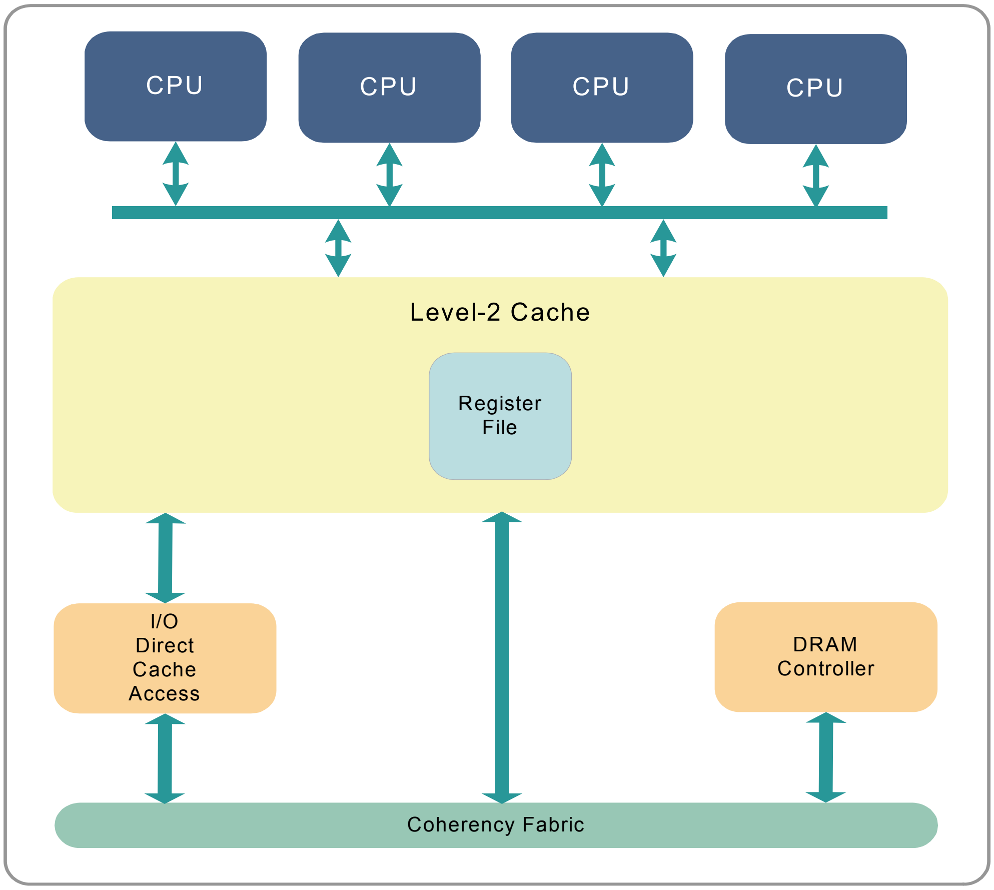
The device's L2 cache is an on-chip, unified instruction and data Level 2 cache, shared by all CPUs on the device. This L2 cache is designed to serve as a large last level cache, while keeping the L1 caches small and tightly coupled to the core.

The L2 cache architecture maintains coherency between all the CPU cores, as well as maintaining the ability of serving multiple CPU accesses in parallel.

The shared L2 cache is architecturally visible to each processor core. It appears as Level 2 to the processor. Thus, in addition to the memory mapped register programming specified in this section, the L2 cache is also discoverable, maintainable, and enabled through CP15 registers and operations. This is described in the CP15 chapter of the CPU datasheet. Only legacy and firmware control of the L2 cache need to be performed by the memory mapped register model. All cache maintenance, discovery, and software-visible enabling of the L2 cache can be performed by the same architectural features that are used to manage the internal caches inside the processor core.

Additional functionality of the unit is SRAM, where part of the memory used for L2 cache is allocated for direct CPU and IO accesses and functions as memory space. These memories have no cache characteristics, and instead, are pure SRAM memories.

Figure 14: Quad Core Level-2 Cache High-Level Block Diagram



The L2 cache registers are located in Appendix A.2, CPU Sub-System Registers, on page 600.

8.1 Features

The MV78230/78x60 supports the following L2 cache configurations:

MV78460	■ Unified, shared, 2 MB cache ■ 32-way set-associative cache
MV78230 **MV78260**	■ Unified, shared, 1 MB cache ■ 16-way set-associative cache

The L2 key features are:

- Physically indexed and physically tagged
- Single Tag per Line cache structure, with a fixed line length of 32B; Data RAM is byte writable
- Configurable SRAM or cache mode per way
- Write-back and Write-through mode selection, according to the page table setting, as determined by the processor memory page attributes

- Only Write-through mode, regardless of the page table setting—on cache addresses only (configurable)
- Only Write-back mode, regardless of the page table setting—on cache addresses only (configurable)
- Pipelined cache structure, supporting multiple, outstanding, Read and Write requests, improved non-blocking conditions, delivering a data rate of 32B per cycle upon multiple masters simultaneous access.
- Multiple, outstanding misses, enabling miss-under-miss and multiple outstanding line fills from memory
- Hit-under-miss, the new request is served while the L2 cache performs the line-fill operation from memory
- Configurable Pseudo-Random or Pseudo-LRU replacement algorithm
- Support for multiple masters simultaneously access and data delivering
- Way Lockdown format, for data and instructions based on requesting processor
- Way Lockdown format, for I/O data write accesses
- Access to SRAM Way is performed by the regular L2 cache pipeline, supplying full pipelining between different accesses types
- SRAM maps all processors accesses types and masters (instruction, data, and the Memory Management Unit (MMU)), including non-cacheable accesses
- L1 Data and L2 coherency snooping for I/O according to request attribute
- L2 allocate for I/O according to request attribute
- ECC generation, checking, correction, detection, and reporting for data arrays
- Parity generation, checking, and reporting for tag arrays
- ECC and parity error injection modes for testing
- Different static and dynamic Power Management modes
- Supports interrupt generation on cache controller events
- Memory mapped L2 cache register file accessible by all processors
- Cache maintenance operations, performed by programming the operation in the L2 cache registers
- Clean, Invalidate, and Clean-and-Invalidate atomic range operation
- L2 cache performance events monitoring

8.2 L2 Cache Requests Types

There are two types of requests to the L2 cache that may be initiated from the attached processors and the I/O masters that share it.

- General Memory Level-2 Cache Requests
- L2 Cache Maintenance Operations

8.2.1 General Memory Requests to L2 Cache

The L2 controller receives various types of requests from the Coherency Fabric. The request can be marked as allocate or non-allocate while targeting the L2 cache. Whenever a read request misses the L2 cache, the associated data is retrieved from the lower memory level and replicated to the L2 cache, if allocation is required by the request. If not, the data is routed directly to the requester and the L2 cache is not updated. Whenever a write request misses the L2 cache, the associated data is stored to the L2 cache on full cache-line access, if allocation is being required. If not, the data is forwarded directly to the lower memory level.

The allocation modes are utilized for coherency maintenance between the processors cached data in the L2 cache and any I/O master requiring the most updated version of it. By setting an I/O read and write request as coherent and non-allocate, the L2 cache is snooped but is not updated on a miss. This mode of operation ensures data consistency between the I/O masters and the processors. It also prevents trashing critical processor cache lines and minimizes cache pollution.

The L2 controller also supplies caching capabilities for non-cacheable agents, or for address spaces that are defined as non-cacheable in the processors' Level-1 caches as programmed in the processors memory page attributes. For example, the MMU descriptors that are non-cacheable in the processor can be programmed as outer cacheable. They are replicated in the L2 cache to speed up the TLB Lookup process, and are shared among all the processors.

The global write-allocate modes is controlled by programing the <Force Write Allocate> field in the L2 Auxiliary Control Register (Table 295 p. 691).

Table 18 details the L2 controller supported transactions.

Table 18: General Memory Level-2 Cache Requests

Request Type	Description
Instruction line fill	L2 line-fill request from L1 Instruction Cache (I-cache) of the requester processor. On a miss, the L2 issues a line-fill request through the external memory.
Data line fill	L2 line-fill request from L1 Data Cache (D-cache) of the requester processor. On a miss, the L2 issues a line-fill request through the external memory.
MMU line fill	L2 read request from MMU of the requester processor. On a miss, the L2 issues a line-fill request through the external memory. On a hit, data is returned directly from the L2 cache. For the MMU transaction type, only the required double-word is returned to the requester while the L2 is filled with the entire cache-line containing it.
L1 stores in Write-through (WT) page attribute[1]	On an L2 cache hit, the data is written to L2. On a miss, data is not written to L2, unless a write-allocate on a full cache line is required and the global write-allocate mechanism is enabled. On both a hit and a miss, data is also forwarded to the main memory through the write buffer.
L1 stores in Write-back (WB) page attribute	On an L2 cache hit, data is written to L2 and is not forwarded to the main memory. On a miss, data is forwarded to the main memory, unless write-allocate on a full cache line is required, and the global write-allocate mechanism is enabled.
L1 dirty cache-line victimization	Request to store victim data when the L1 Data Cache victimizes a dirty line in the WB page attribute.
Register file access	Access to the L2 private register file on Read and Write for configuration or control operations.
IO Master Read access	On a hit, data is returned directly from the L2. On a miss, data is returned from the main memory but is not allocated to the L2.

Table 18: General Memory Level-2 Cache Requests (Continued)

Request Type	Description
IO Master Write access	On a hit, data is written directly to L2. On a miss, data is written to the main memory, unless a full cache-line Write is required with a Write-allocate indication.

1. <L2 WBWT Mode> field in the L2 Auxiliary Control Register (Table 295 p. 694) is configured as a *PageAttibute* mode.

8.2.2 L2 Cache Maintenance Operations

L2 cache maintenance is performed through a set of memory-mapped registers and is shared by all the processors in the coherency fabric. The L2 controller supplies the following groups of maintenance operations:

- Invalidate, Clean and Flush by Way
- Invalidate, Clean and Flush by Physical Address (PA)
- Invalidate, Clean and Flush by Index/Way
- Invalidate, Clean and Flush by Physical Address range

All cache maintenance operations by the physical address are non-blocking, consecutive accesses to the cache. These operations are served, and memory ordering with respect to these operations is handled by the cache controller.

The cache maintenance operation is completed once the line is drained to the cache eviction buffer. The cache memory barrier operation is required to synchronize these operations with the lower level memory. This mode of operation delivers better performance than draining it directly to lower memory level, when a range of addresses should be synchronized to the point of coherency with the I/O agents. To further improve it, the L2 controller supports range cache maintenance by single command operations.

Triggering a range operation is performed in two steps. In the first step, the range base address is programmed. The second step sets the range top address and triggers the command. To enable atomic range operation in a multiprocessor environment, the L2 supplies a dedicated base address register for each requester, virtually accessed by all processors through Table 272, L2 Range Base Address Virtual Register, on page 679 and a shared command triggering register. Any access to the command triggering register results in a cache range operation with the base address associated with the initiating requester. Additional requests from other processors are queued for service in the cache controller, thus releasing the bus for other memory operations.

Way operations run as background tasks. The L2 controller serves any consecutive memory access while processing the maintenance operation. Further maintenance operations are blocked until the completion of the current one.

The cache controller returns the status of the current maintenance operation by reading the <CPU0 Op Pending> field in the L2 Maintenance Status Register (Table 270 p. 678).

The cache controller synchronizes all pending and active operations to point-of-coherency by triggering the L2 Sync Barrier operation.

Table 19 details the supported cache maintenance operations.

Table 19: Core Special Maintenance Instruction L2 Requests

Request Type	Description
L2 Sync Barrier	The L2 controller completes all active and pending operations, drains intermediate and eviction buffers to memory. **NOTE:** Completion is returned to the initiator only on completion. Meanwhile the processor is stalled. This operation is used after cache maintenance operations. Trigger the operation by writing to the L2 Sync Barrier Register (Table 269 p. 677).
Invalidate L2 Way	Invalidate all lines in the specified Ways, including any dirty lines. Trigger the operation by writing to the L2 Invalidate by Way Register (Table 278 p. 681).
Invalidate L2 Index/Way	Invalidate an L2 cache line as specified by the Index and Way. Trigger the operation by writing to the L2 Invalidate by Index Way Register (Table 277 p. 680).
Invalidate L2 line by PA	Invalidate a specific L2 cache line, if the supplied physical address matches an address entry in the array. Trigger the operation by writing to the L2 Invalidate by Physical Address Register (Table 275 p. 680).
Invalidate range by PA	Invalidate a memory range in the L2 cache, presented by the base and top PA boundaries. Trigger the operation by writing to the L2 Invalidate Range Register (Table 276 p. 680). This uses the range base address from the pre-programmed corresponding processor in the L2 Range Base Address Virtual Register (Table 272 p. 679)
Clean L2 Way	Clean all dirty lines in the specified Ways. Trigger the operation by writing to the L2 Clean by Way Register (Table 283 p. 684). When a Way bit is set to 1, it is cleared to 0 when the corresponding Way is totally cleaned.
Clean L2 line by Index/Way	Clean a L2 cache line as specified by the Index and Way. If that line is dirty, it is written back to memory and marked as non-dirty. The valid bit remains unchanged. No operation is performed in L2 WT mode. Trigger the operation by writing to the L2 Clean by Index Way Register (Table 282 p. 683).
Clean L2 line by PA	Clean a specific L2 cache line, if the supplied physical address matches an address entry in the array. No operation is performed in L2 WT mode. Trigger the operation by writing to the L2 Flush by Physical Address Register (Table 284 p. 684).
Clean range by PA	Clean a memory range in the L2 cache, presented by base and top PA boundaries. No operation is performed in L2 WT mode. Trigger the operation by writing to the L2 Clean by Range Register (Table 281 p. 683). It uses the range base address from the pre-programmed corresponding processor.

Table 19: Core Special Maintenance Instruction L2 Requests (Continued)

Request Type	Description
Clean and Invalidate L2 line by Index/Way	Clean and invalidate a L2 cache line as specified by the Index and Way. In L2 WT mode, the operation only invalidates the line (since lines are never dirty). Trigger the operation by writing to the L2 Flush by Index Way Register (Table 286 p. 685).
Clean and Invalidate L2 line by PA	Clean and invalidate a specific L2 cache line, if the supplied physical address matches an address entry in the array. In L2 WT mode, the operation only invalidates the line (since lines are never dirty). Trigger the operation by writing to the L2 Flush by Physical Address Register (Table 284 p. 684).
Clean and Invalidate range by PA	Clean and invalidate a memory range in the L2 cache, presented by base and top PA boundaries. In L2 WT mode, the operation only invalidates the line (since lines are never dirty). Trigger the operation by writing to the L2 Flush by Range Register (Table 285 p. 685). It uses the range base address from the pre-programmed corresponding processor L2 Range Base Address Virtual Register (Table 272 p. 679).

8.2.2.1 Broadcast of Unified Cache Maintenance

ARMv7 requires MVA-based data and unified cache maintenance operations affect through the Point of Coherency. With the addition of the architecture-shared L2 cache, the point of coherence is defined as beyond the shared cache. Therefore, all the MVA-based maintenance instructions are applied on the shared cache as well.

The extensions also include data/unified cache maintenance operations that broadcast the following:

- DCIMVAC: Invalidate cache line to the point of coherency
- DCCMVAC: Clean cache line to the point of coherency
- DCCIMVAC: Clean and invalidate cache line to the point of coherency
- DCCMVAU: Clean cache line to the point of unification
- Hierarchal invalidate, clean, and clean-and-invalidate data cache by Set/Way is defined by the CPU CP15 coprocessor.

The cache controller supports acceptance and pipelining of multiple maintenance instructions to the point of coherency by the processors. It services a CPU Data Memory Barrier instruction that is globally broadcast to the L2 cache.

8.3 L2 Cache Functional Description

This section provides a functional description of the following L2 Cache mechanisms:

- L2 Cache Register File
- L2 Cache Initialization
- Way Disabling Mechanism
- L2 Global Write-Through and Write-Back Modes
- Way Lockdown
- Replacement Policy
- SRAM Allocation

8.3.1 L2 Cache Register File

The L2 cache includes a single memory-mapped register file for system configuration and control operations that is shared among all the processors. The registers are divided into the following categories:

- L2 Configuration registers
- L2 Cache Maintenance registers
- L2 Cache Performance Monitor registers
- L2 Error and Interrupt Control registers
- Miscellaneous L2 registers

The register file can be accessed in both L2 Enable and L2 Disable modes.
On a write to one of the L2 configuration registers, the L2 performs cache synchronization. The write occurs only after the L2 buffers are empty. There are no outstanding requests whatsoever. All expected line-fill commands are completed, as well. The L2 is in Idle state. New commands for both the register file and L2 regular operation are not served until the configuration write takes place.

8.3.2 L2 Cache Initialization

The L2 cache is initially disabled following a power-on, hard reset or exit from low power management mode. Following reset, the L2 controller performs self-invalidation, and all of the cache lines in all of the ways are invalidated. After the global hardware or software invalidation has been performed, and the other L2 configurations have been set, the L2 controller can be enabled for normal operation by setting the <L2Enable> field in the L2 Control Register (Table 294 p. 690).

8.3.3 Way Disabling Mechanism

The L2 controller is a 16/32-way, set associative cache, and it supports a Way Disabling mechanism for power reduction by changing the cache <Associativity> field in the L2 Auxiliary Control Register (Table 295 p. 692).

Once a Way is disabled, it is never used for L2 cache operations and is considered as locked-down for all the requesters and access types.

This mode enables flexible tuning between performance and power.

8.3.4 L2 Global Write-Through and Write-Back Modes

The L2 controller supports the following cache WT and WB policies:

- Request Attribute: Policy whether the write is forwarded to the memory in addition to the cache updating is determined by the request attribute. If the requester is the processor, these attributes are programmed for each page in the processor MMU. An I/O request is always considered as write-back.
- Always Write-Back mode: In this mode, the write is never forwarded to the memory on cache hit regardless of the request attribute.
- Always Write-Through mode: In this mode, the write is always forwarded to the memory on cache hit regardless of the command attribute. The line never becomes dirty, so there is no need for the data entry to be cleaned back to memory, after it has been evicted on replacement.

All of the operation modes are controlled by the <L2 WBWT Mode> field in the L2 Auxiliary Control Register (Table 295 p. 694).

8.3.5 Way Lockdown

When using the Way Lockdown mechanism, the unified L2 can be dynamically divided between different processors and between Instruction, Data and the IO requests. In addition, a critical code or data can easily be directed to a specific cache Way, with the assurance that it will not be polluted or evicted.

In the Way Lockdown for I/O Write access, with Write-Allocation configuration, the L2 evicts a line only for a Way that is not locked down to the I/O master.

On cache read miss, a new line fill is required from a processor L1 cache for instruction or data. Potentially the line can be located in any Way in the L2. The Way Lockdown mechanism restricts the cache, so that it only locates new lines on Ways that are not locked.

The L2 controller supports a separate locking mechanism for the instruction and data per processor. Way lockdown for the specified processor data caching is controlled by the L2 CPUn Data Lockdown Register (n=0–3) (Table 256 p. 664) and Way Lockdown for the specified processor instruction caching is controlled by the L2 CPUn Instruction Lockdown Register (n=0–3) (Table 257 p. 667).

The L2 controller supports the option for locking down a specified Way for I/O data accesses by programing the IO Bridge Lockdown Register (Table 258 p. 669).

If all the Ways are data locked and there is a data miss that also misses the L2, the L2 controller issues a data Read request to the external memory. The data bypasses the L2 cache, going directly to the L1 D-cache, without updating the L2 data arrays. The same process occurs for instruction.

8.3.6 Replacement Policy

The MV78230/78x60 L2 controller uses a pseudo-random replacement algorithm, that fills empty, unlocked Ways first. The victim is chosen as the next unlocked Way relative to the last accessed Way.

Any combination of Ways can be locked. The locking mechanism affects the replacement algorithm so that only the unlocked Ways are candidates for replacement. If only one Way is unlocked, this Way is always replaced. If all Ways are locked, no replacement takes place.

Pseudo-Random (LFSR)

A free-running LFSR defines the victim candidate. If the victim candidate is not available for allocation, the first capable Way over a pre-defined cycle order.

Semi-pLRU

Pseudo LRU replacement is a variant of LRU where the “ages” of the lines in the cache are not linearly ordered, but are arranged in a tree as shown in Figure 15. The advantage over a pure LRU system is that it needs fewer state bits and hence needs a less complex update logic.

pLRU maintains a tree of cache Ways for every index. Every inner tree node has a bit pointing to the sub-tree that was used most recently. The nodes are updated upon a hit and upon every replacement. A victim is chosen by walking down the tree through the arcs that are not pointed.

Copyright © 2014 Marvell

Figure 15: pLRU 8-Way Tree Example

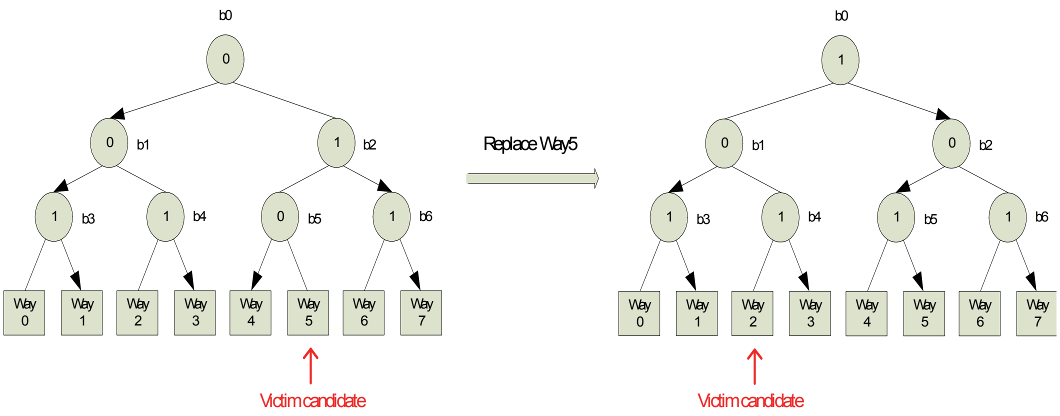
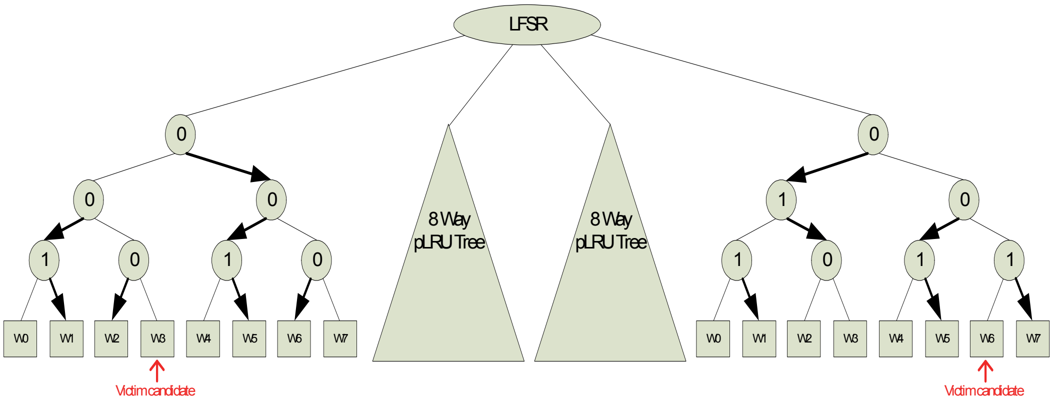


Figure 15 is an example of a change in the 8-way pLRU tree. The tree has 7 variable {b0, b1, b2, b3, b4, b5, b6}. The emphasized arrows provide the meaning of the variables. In the left tree, Way3 is the victim candidate. Following its replacement, the tree is updated so that the nodes point to the replaced Way, therefore pointing to the last Way to be used. As a result of this transformation, Way1 becomes the victim candidate.

For implementations that contain more than 8 ways, a semi pLRU victim decision is used as shown in Figure 16. In this scheme, the hardware tree decision is localized per groups of 8 ways (quads). The victim group is pseudo randomly chosen using an LFSR. Each pLRU tree independently maintains a tree for its ways.

Figure 16: Semi-pLRU for 32 Ways



A replacement mode is determined by the <Replacement Strategy> field in the L2 Auxiliary Control Register (Table 295 p. 691).

8.3.7 Write Allocate Modes

New cache lines are allocated for full-cache-line writes (32B).

Partial write requests (less than 32B) update the cache only if the line is already valid in the cache. In this case, the write data is merged into the valid data that is already placed in the cache. The allocation request is controlled by an attribute from the initiator and it is enabled for the processors and I/O accesses.

Write allocate feature can be forced by <Force Write Allocate> field in the L2 Auxiliary Control Register (Table 295 p. 691).

8.3.8 SRAM Allocation

Each Way of the MV78230/78x60 L2 cache can be configured as a sequential memory-mapped SRAM with ECC protection on the data RAMs, supporting both read and write, cacheable and non-cacheable accesses for the processors and I/O masters.

The SRAM is implemented as part of the L2 logic via an allocation algorithm that converts the L2 cache Way to SRAM, utilizing the L2 controller pipeline capabilities, serving cache and SRAM accesses simultaneously.

Configuring a specific Way (notated way i) to function as an SRAM that is based at a 64KB aligned address (notated saddr[31:0]) is done by performing the following sequence:

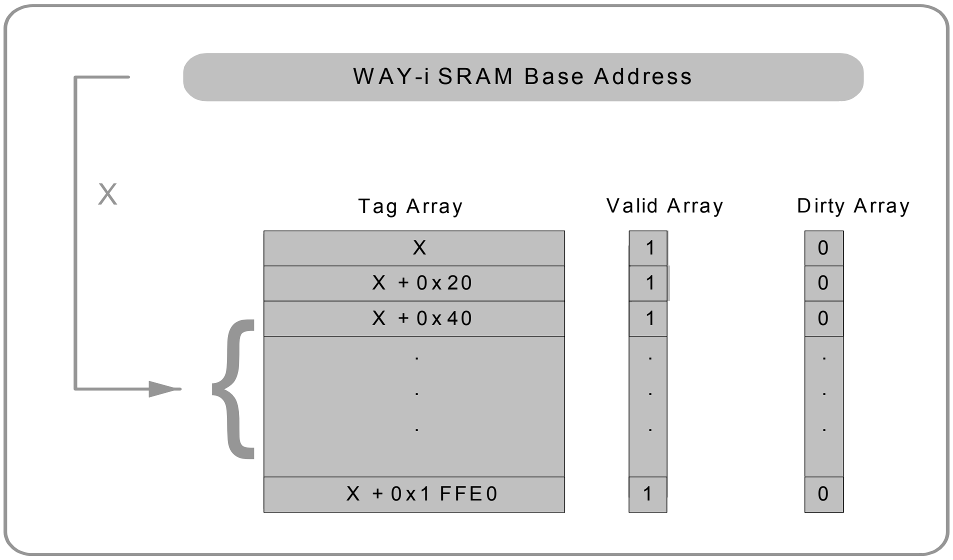
1. Lockdown the specific way for all masters. This is done by setting bit[i] in:
   - L2 CPUn Data Lockdown Register (n=0–3) (Table 256 p. 664)
   - L2 CPUn Instruction Lockdown Register (n=0–3) (Table 257 p. 667),
   - IO Bridge Lockdown Register (Table 258 p. 669).
2. Trigger an Allocation Block command by writing to L2 Block Allocation Register (Table 279 p. 681) the following data:
   - Set <Allocation Way ID> to be i
   - Set <Allocation Data> to Disable (0x0)
   - Set <Allocation Atomicity> to Foreground (0x0)
   - Set <Allocation Base Addr> to be saddr[31:10]
3. Configure one the SRAM windows SRAM Window n Control Register (n=0–3) (Table 161 p. 617) to direct the required range to be an SRAM:
   - Set Base to be saddr[31:16]
   - Set Size to 64KB (0x0)
   - Set WinEn to True

In case an SRAM region is larger than 64 KB and thus requires 2 or 4 ways, step 2 can be repeated for all ways and a single SRAM Window that covers more than 64 KB can be enabled. For example, to configure a 128 KB SRAM, step 2 can be performed on two ways and then the <Size> field in the SRAM Window n Control Register (n=0–3) (Table 161 p. 617) is set to 128 KB (0x1).

Software does not need to handle the status of the SRAM allocated way prior to this procedure. The cache controller evicts and invalidates any previous content in the allocated Way.

Figure 17 illustrates the Per-Way SRAM initialization flow.

Figure 17: SRAM Initialization per Way



8.3.9 DRAM Aware Eviction

When activating a 64-bit DRAM interface with DDR3, the DRAM native burst length is 64 Bytes. To avoid the utilization degradation that exists, when every evicted 32-Byte cache line performs a 64-Byte access over the DDR, the DRAM controller contains a write coalescing mechanism that packs memory writes. The efficiency of the DRAM controller write coalescing depends on the gap that exists between the eviction of consecutive cache lines. The L2 can be configured such that when a 32 byte cache line is evicted due to its replacement, the 32 byte cache line that completes the 64 Bytes native access is looked up and if it is modified, its data is written to the DRAM controller. Following this action, the completing cache line remains allocated and clean in the L2 cache. This action is referred as a DRAM aware eviction. To enable DRAM aware eviction, set the <L2DualEvictionEnable> field in the L2 Auxiliary Control Register (Table 295 p. 693).

8.4 Error Handling

This section describes the following L2 error handling functions:

- Error Protection Support
- Error Protection Flows
- Error Reporting
- Error Injection

8.4.1 Error Protection Support

The L2 controller supports error protection for both data and tag arrays.

- Data arrays are ECC protected.
- Tag and valid arrays are parity protected.

The Error Protection mechanism can be enabled and disabled.

8.4.2 Error Protection Flows

8.4.2.1 Data ECC Protection

The L2 SRAM unit delivers data memory protection, using the ECC Extended Hamming algorithm for both SRAM and L2 data memories. The L2 ECC protection is guaranteed to detect a double-bit error, and to correct a single-bit error in the L2 data arrays, dubbed uncorrectable and correctable errors respectively. The ECC is calculated on double-word (64-bits) data resolution, resulting in a read-modified-write (RMW) operation on a sub-cache-line write to L2, when not all bytes within a double word of the write operation are valid.

The unit behaves differently when an uncorrectable error is detected on the following transactions types:

L2 Read hit with an uncorrectable ECC error	The unit returns an abort indication with the L2 trashed data. An error is reported.
L2 Read miss	No change occurs.
L2 Eviction with an uncorrectable ECC error	If the evicted line is not dirty, no error is reported. If the evicted line is dirty, an error is reported and writing-back the erroneous line through the lower memory level is determined by the setting of the <L2InvalEvicLineUCErr> field in the L2 Auxiliary Control Register (Table 295 p. 691).
L2 Clean, Flush, and Invalidate	If the line is not dirty, no error is reported. If the line is dirty, an error is reported and writing-back the erroneous line result of the clean operation to the lower memory level is determined by the setting of the <L2InvalEvicLineUCErr> field in the L2 Auxiliary Control Register (Table 295 p. 691).
L2 Write hit with double word resolution (Nx64-bit) access	ECC is calculated. No error is checked or reported.
L2 sub cache-line Write hit, when RMW is required, and uncorrectable ECC error	The unit writes the inversion of two bits, at the calculated ECC, to the ECC arrays. This ensures an ECC error on the next read or eviction from this entry. An error is reported on the write operation.

8.4.2.2 Tag Parity Protection

The parity protection mechanism is implemented in the L2 and SRAM, for the tag and valid arrays. Each tag input is stored in memory, with an additional bit that holds the result of the XORed value. This bit is compared when the tag is read and XORed to check for correctness. The appropriate interrupt bit that indicates the error is allocated in the L2 Interrupt Cause Register (Table 292 p. 689).

On each tag lookup, the entry is considered a *hit* only when there is a hit indication and no parity error on the tag (a true hit). If a parity error is detected, the entry is considered a *false hit,* regardless of the comparator output.

The unit behaves differently when tag parity error is detected on the following transactions types.

Read

Read with “true hit”, no error is reported.

Read miss and no parity error is detected on any Way. It is a “true miss” indication, no error is reported.

Read miss and at least one parity error is detected on any Way, the L2 controller treats it as a “false hit”. Erroneous data associated with the error indication is returned to the requester read operation. An error is reported.

Write

Write with “true hit”, no error is reported.

Write miss and no parity error is detected on any Way. It is a “true miss” indication. No error is reported.

Write miss and at least one parity error is detected on any Way, the L2 controller treats it as a “false miss”. The write is not allocated to L2, and the write data is written to the lower memory level. An error is reported.

L2 Clean, Flush and Invalidate by PA/Range

Operation with “true hit”, no error is reported.

Operation miss and no parity error is detected on any Way. It is a “true miss” indication. No error is reported.

Operation miss and at least one parity error is detected on any Way, no cache operation is performed. Error is reported.

L2 Clean, Flush, and Invalidate by Index/Way

Operation results with no parity error, no error is reported.

Operation results with parity error on the requested Index/Way. No error is reported for invalidate only or clean or flush of not dirty line. Otherwise, an error is reported and the line is not written back to memory.

8.4.3 Error Reporting

To deliver maximum assistance to the software, special error monitoring registers were added for capturing the erroneous event information.

L2 ECC Error Count Register (Table 259 p. 672)
For counting the correctable and uncorrectable ECC errors detected.

L2 ECC Error Threshold Register (Table 260 p. 673)
For setting the threshold value of the counters for interrupt generation.

L2 Error Address Capture Register (Table 262 p. 674)
For holding the address that caused the error.

L2 Error Way Set Capture Register (Table 263 p. 675)
For holding the Index/Way containing the erroneous data or tag.

L2 Error Attr Capture Register (Table 261 p. 673)
For holding the validity information on the pended error and its related attribute. Includes the source of the transaction, the transaction type, and the captured error type.

The L2 controller counts the correctable and uncorrectable ECC errors, when the counter reaches the configured threshold, the appropriate interrupt is set in the L2 Interrupt Cause Register (Table 292 p. 689).

8.4.4 Error Injection

The L2 controller includes support for injecting an error into the L2 ECC and Parity arrays. This may be used for testing the error recovery software. The error is deterministically injected on a specified address during the line fill operation. When the allocated physical address is equal to the <ErrAddrInject> field in the L2 Error Injection Control Register (Table 264 p. 675), and the <EccErrInjectEn> or/and <TagParErrInjectEn> are enabled, the corresponding error will be injected to this address. In addition, the ECC Error location and type is configured through the <Error Injection Mask>.

8.5 Events Monitor Counters

Event counters monitor both L2 performance and the Shared L2.

8.5.1 L2 Performance Monitor

The L2 cache performance monitor provides the ability to monitor and count the following pre-defined events among others:

- Cache activity and idle clocks
- Total instruction
- Data accesses to the cache
- Misses for instruction
- Internal stalling conditions

The L2 controller includes two sets of 64-bits performance counters for monitoring different L2 cache events. Each event can be configured to be monitored only for the processors or I/O,.

The events monitor includes the following registers:

L2 Counter <X>_ Configuration Register (X=0–1) (Table 289 p. 687)	Per counter programming register for controlling the monitor operation: enabling the counter, selecting the monitored event, and the monitored masters (CPUs or I/O).
L2 Counter X Value Low Register (X=0–1) (Table 290 p. 689)	The counter low value (bits[31:0])
L2 Counter X Value High Register (X=0–1) (Table 291 p. 689)	The counter high value (bits[63:32])

The performance monitor enables the generation of an interrupt triggered by a counter overflow condition.

8.5.2 Shared L2 Events Monitoring

The event monitor can be configured to gather statistics on the operations of the MV78230/78x60 Shared L2. Event monitor control is performed directly through each CPU local event monitor interface (that is, CP15 Performance Monitor Unit).

Table 20 lists the events that are monitored per CPU in the MV78230/78x60 Shared L2.

Table 20: Shared L2 Events

Event Monitored	Counter Description	Mapping in CP15 Event Select Register
Per CPU L2 cache event.		
L2 Cache Eviction	Counts total number of write evictions from the L2 cache	0xD0
Data Read Hit	Counts total number of data read hits	0xD1
Data Read Request	Counts total number of data read requests	0xD2
Data Write Hit	Counts total number of data write hits	0xD3
Data Write Request	Counts total number of data write requests	0xD4
Instruction Read Hit	Counts total number of instruction read hits	0xD5
Instruction Read Requests	Counts total number of instruction read requests	0xD6
Request Causes and Eviction	Counts total number of requests that caused an eviction from this CPU	0xD7
Reserved	Reserved	0xD8–0xD15

8.6 Power Management

The L2 Cache supports the following Power Management features:

- L2 Idle state: dynamic clock gating
- L2 Retention state: dynamic memory state retention
- L2 Deep Idle state: cache controller and its memories are powered down, state is stored in lower memory level.

8.6.1 Cache Controller Power Management Features

The L2 contains a mechanism that reduces power consumption upon a configurable period of IDLE cycles. By doing so, the L2 offloads from software the need to actively do the same actions upon an IDLE period that it enters (e.g., when the CPU enters to WFI). In addition, it eliminates the power management coordination between the cores in the multiprocessor subsystem on trying to place the L2 in low power mode while not all the cores are IDLE simultaneously.

Enabling the <DCPDEnable> field in the L2 Dynamic Clock Gating Register (Table 267 p. 676) results in L2 clocks shut-down after the <DCPDTreshold> IDLE period. Setting the <FCPD> field in the L2 Force Power Down Register (Table 268 p. 677) forces immediate L2 clock gating.

Enabling the <DMPDEnable> field in the L2 Dynamic Memory Power Down Register (Table 266 p. 676) results in placing the L2 arrays to low power state retention state after the <DMPDTreshold> IDLE period. Setting the <FMPD> field in the L2 Force Power Down Register (Table 268 p. 677) forces an immediate L2 Memory powering down.

Waking-up from these two modes is automatically performed by the cache controller upon the detection of any cache access.

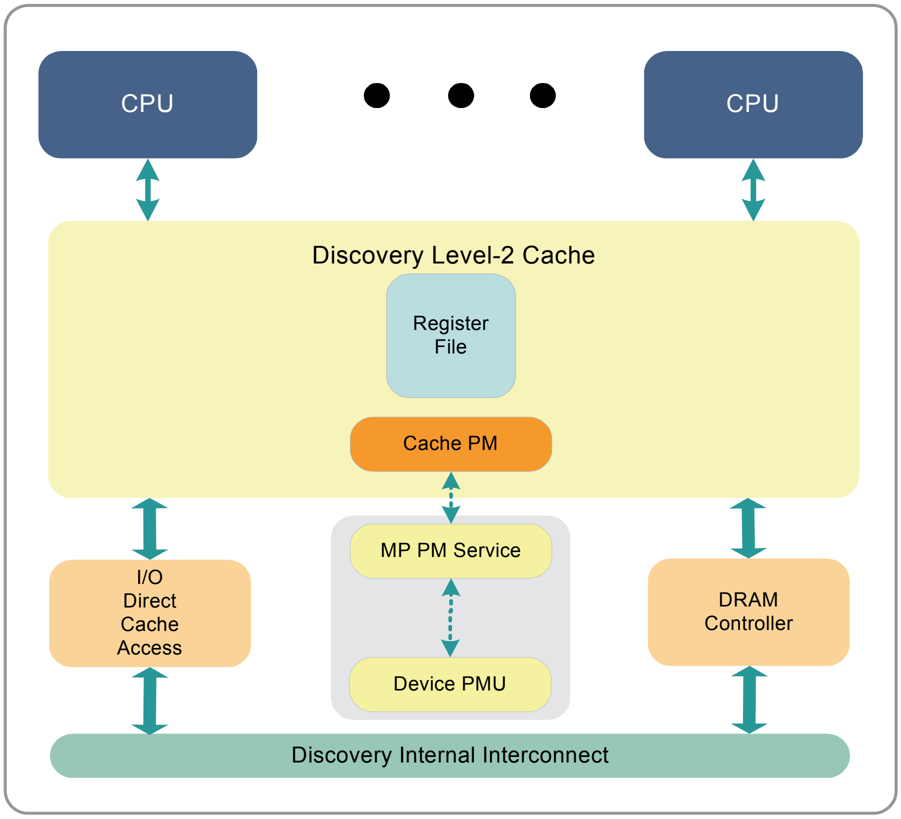
8.6.2 Multiprocessing Power Management Features

The MV78230, MV78260, and MV78460 support powering down of the entire multiprocessor subsystem, including all the cores and the shared L2 cache.

For shutting down the core and its L2 cache, each core's powering down might be independent of the other cores. Therefore, it must not request flushing the shared L2 cache. However, on the time point when all the cores have been placed in the power-down state, the shared L2 cache can be powered down as well for further power saving. This is fully achieved by the MP_PMU service unit and the L2 contoller.

The PM_PMU service unit includes hardware coordination logic to detect when all cores have reached their power down conditions. It then instructs the shared L2 cache to flush its content to the lower-level memory, places it in a reset state, and communicate with the DEV_PMU to power down the entire multiprocessor subsystem. Immediately following the power-down exit, the L2 controller performs a self invalidation.

Figure 18: L2 Power Down



Copyright © 2014 Marvell

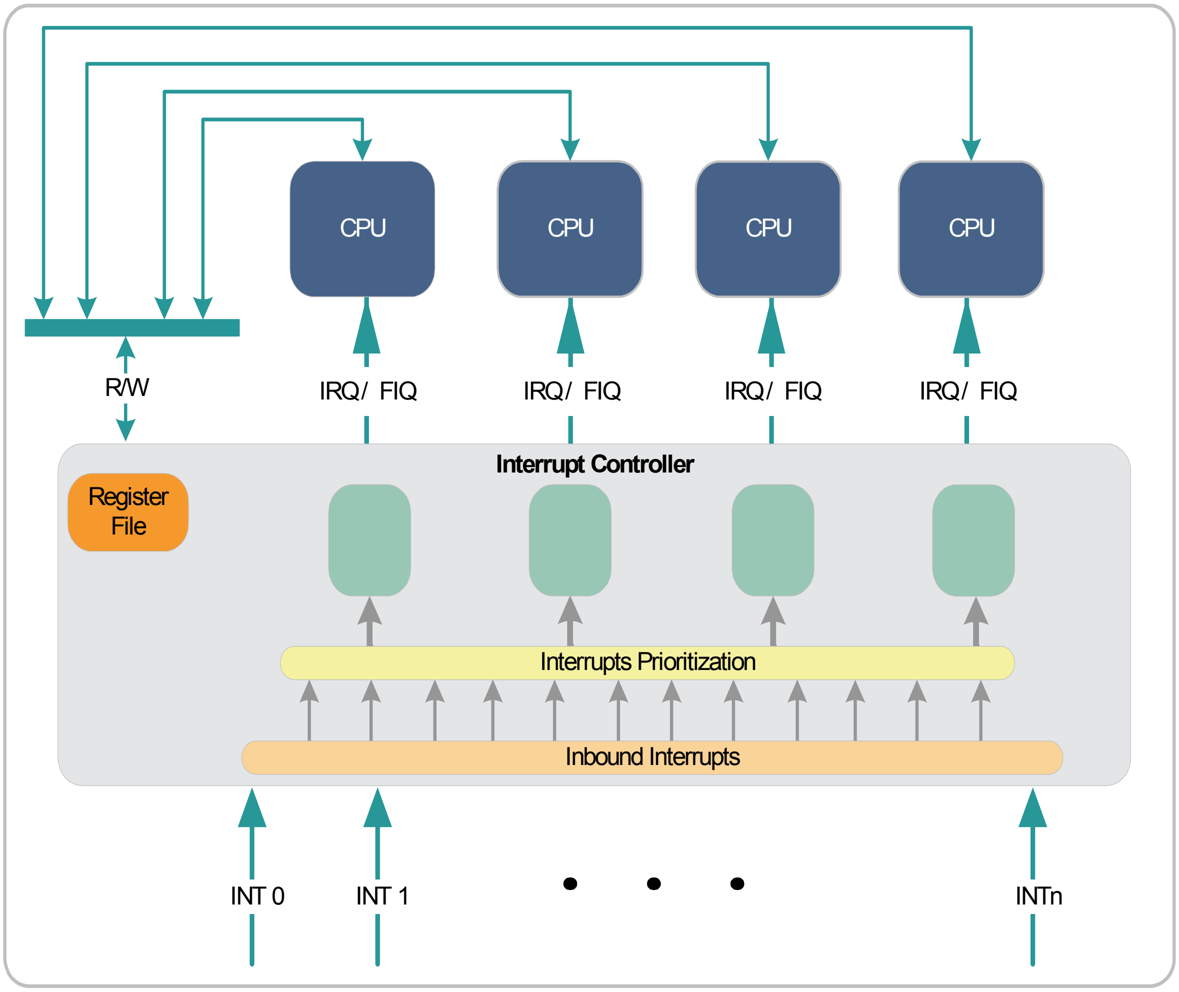
9 Multiprocessor Interrupt Controller (MPIC)

This section describes the operation of the Multiprocessor Interrupt Controller (MPIC) add-on for the MV78230/78x60 device.

The Interrupt Controller is a single functional unit placed in the device Coherency Fabric, alongside the CPUs. It provides multiprocessor interrupt management, and is responsible for receiving interrupts from different sources, prioritizing them, and distributing them to the target CPUs.

Figure 19 describes the Interrupt Controller unit.

Figure 19: Main Interrupt Controller High-Level Block Diagram



Copyright © 2014 Marvell

The MPIC registers are located in Appendix A.2, CPU Sub-System Registers, on page 600.

9.1 Features

The following Multiprocessor Interrupt Controller features are supported:

- Configurable model for multiprocessor interrupts distribution and prioritization
- Centralized engine for all the processors
- Programmable Critical (FIQ) and Non-Critical (IRQ) interrupt mapping per interrupt source and per processor
- Programmable interrupt prioritization levels, with fairness and interrupt preemption mechanism
- Programmable Task Priority per processor
- Supports interrupt selection by hardware (return the highest priority ID) or by software (*Select Cause*: legacy)
- Generation and broadcasting of software interrupts between all the MP processors and between I/O masters (for example, external hosts) and the processors
- Global SoC Error Cause with per-processor masking
- PCIe Root Complex MSI capturing and delivering to the processors
- Supplies mapped memory registers interface, accessed by all the CPUs
- Running at Coherency Fabric clock frequency, enables low access latency for the processors

9.2 Functional Description

9.2.1 Interrupt Sources IDs

The MV78230/78x60 Interrupt Controller supports a setting for the maximum number of main interrupt sources through the <NumINT> field in the MPIC Control Register (Table 296 p. 694). Each interrupt source is identified by an interrupt ID.

The interrupt's sources and their mapping are listed in Table 21, Internal Interrupts Mapping, on page 151.

The <NumINT> interrupts are divided to the following 2 groups:

CPU Private Interrupts: Interrupt sources ID0–ID28 are private events per CPU. Thus, each processor has a different set of events map interrupts ID0–ID28.

The following events are allocated by hardware per each CPU:

- 32-bit maskable Inbound Doorbell for IPI and message passing
- Two Local Timers
- One Local Watchdog Timer
- Summary of SoC Error events
- Summary of Coherent Fabric events
- GPIO Interrupt Control
- Network ports events

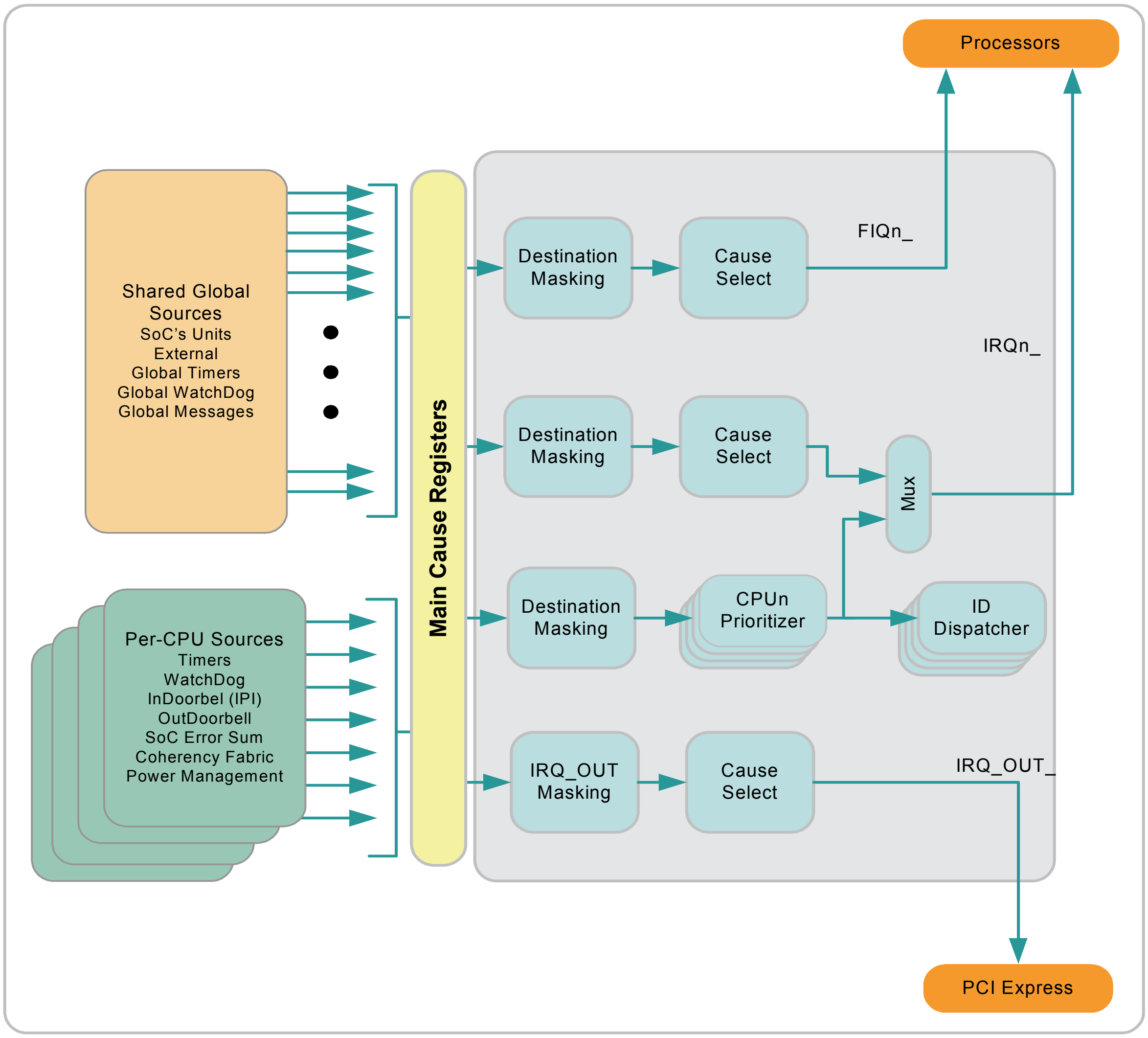
Shared Global Interrupts: Interrupt sources ID29–ID115 are events shared to all CPUs. Thus, for all CPUs the event mapped to interrupts ID29–ID115 are mutual.

9.2.2 Interrupt Sources Control

Each interrupt source has a programmable interface that defines its distribution to the processors.

Figure 20 illustrates the main interrupt controller sources and distribution blocks.

Figure 20: Interrupt Sources Block Diagram



Each source (identified by an ID) has a 32-bit control register for programming its distribution attributes.

- The fields controlling CPU private interrupts are in the Interrupt Source i Control Register (i=0–28) (Table 301 p. 697) (i = interrupt source ID).
- The fields controlling the Shared Global Interrupts are in the Interrupt Source i Control Register (i=29–115) (Table 302 p. 699).

The following control options exist for each interrupt source:

CPU IRQ Mask field: For each CPU, a single bit field that defines whether the interrupt assertion is sensed as an IRQ event.

CPU FIQ Mask field: For each CPU, a single bit field that defines rather the interrupt assertion is sensed as an FIQ event.

EP Interrupt Mask field: This field only exists for the global shared interrupts (ID29-ID115). It is used when the MV78230/78x60 functions as a PCIe Endpoint. This is a single bit field that defines the interrupt assertion sent over the PCIe as an interrupt message.

CPU EP Interrupt Mask field: This field exists only for the CPU Private Interrupts (ID0–ID28). It is used when the MV78230/78x60 functions as a PCIe Endpoint (EP). For each CPU, a single bit field defines the private interrupt assertion sent over the PCIe as an interrupt message, or is not sent to the PCIe Host.

Priority field: Defines 16 priority levels of the interrupt source. For further details on the priority usage see Section 9.2.4.2, Hardware Interrupts Prioritization Selection Mode, on page 144.

Interrupt Enable field: This field exists only for the global shared interrupts (ID29–ID115). When this field is not set, the interrupt is not delivered to any CPU regardless of the IRQ and FIQ Mask bits.

NOTE: The associated interrupt cause register is set regardless of the value in this field.

To minimize the locking procedures, as required for read-modified-write atomic access to the interrupt source control registers, the MV78230/78x60 Interrupt Controller provides an indirect mechanism that makes it possible to perform a single write for masking, or unmasking, the requester’s mask field, and for enabling or disabling the general interrupt enable bit:

- To mask an interrupt with a specific ID, processor writes the interrupt ID to the Interrupt Set Mask (ISM) Register (N=0–3) (Table 315 p. 706).
- To unmask an interrupt with a specific ID, processor writes the interrupt ID to the Interrupt Clear Mask (ICM) Register (N=0–3) (Table 316 p. 706).
- To set the interrupt enable bit of a specific ID, processor writes the interrupt ID to the Interrupt Set Enable (ISE) Register (Table 299 p. 696)
- To clear the interrupt enable bit of a specific ID, processor writes the interrupt ID to the Interrupt Clear Enable (ICE) Register (Table 300 p. 696)

The ISM and ICM are banked registers and the ISE and ICE affect a field that is not processor specific. Therefore, the interrupt handler is able to perform these accesses without knowing which processor it is executed on.

Further details on register banking can be seen in the following section.

9.2.3 Per CPU Register Banking

Register banking refers to providing multiple copies of a register at the same address. The identity of the requesting processor determines which copy of the register is addressed. Registers at offsets 0x00021000–0x000210FF are banked per CPU.

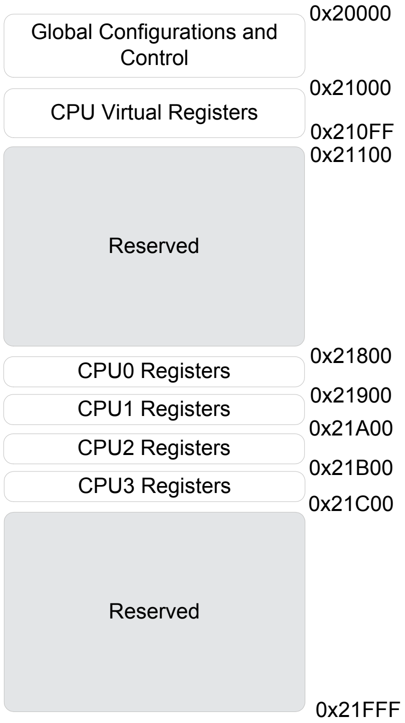
The MV78230/78x60 banked registers can also be accessed with a direct access mechanism, using an explicit *physical offset* of each processor. This mechanism enables processors to access other processors’ private space and for peripherals accesses to the registers as well. The offsets of the direct access of the banked registers are at 0x00021800–0x00021BFF, such that a direct access to the banked registers of CPU[i] is accomplished by accessing 0x21800 + i*0x100.

Copyright © 2014 Marvell

To enable interrupt controllers' execution in a multiprocessor environment, without the need to know which core it is being executed on, interrupt controller's status and control registers are banked.

Figure 21 illustrates the banked register file mapping and the mapped direct access.

Figure 21: MP Subsystem Register File Mapping



9.2.4 Interrupts Delivering Modes of Operation

The Main Interrupt Controller provides 2 modes of delivering the source interrupt:

Software Interrupt Selection: Priority field is ignored, the interrupt pin is asserted to the processor for each valid interrupt (valid is a non-masked asserted interrupt).
Software is responsible of selecting the next interrupt to process.

Hardware Interrupt Selection with Prioritization: Each interrupt source has a configured priority of 16 levels.
Each processor maintains a configurable task priority of 16 levels.
Hardware delivers to each processor the highest priority interrupt among all the pending interrupts that exceed the task priority of the target processor.

IRQ interrupt assertion delivering mode can use either of these 2 modes.

FIQ and PCIe Endpoint interrupts are enabled only by the Software Interrupt Selection mode.

9.2.4.1 Software Interrupt Selection Mode

The interrupted CPU accesses the Main Interrupt Cause register that consists of four 32-bit registers. These read-only registers store all the pre-masking version of the interrupts.

The CPU Interrupt Main Cause registers are:

- Main Interrupt Cause (Vec 0) Register (Table 319 p. 707)
- Main Interrupt Cause (Vec 1) Register (Table 320 p. 707)
- Main Interrupt Cause (Vec 2) Register (Table 321 p. 708)
- Main Interrupt Cause (Vec 3) Register (Table 322 p. 708)

Once an interrupt is asserted, an interrupt controller can identify the exact interrupt to be served by reading all the Main Interrupt Cause vectors and the masks of the asserted interrupt sources.

To minimize this procedure to a single read, the MV78230/78x60 contains a Selected Cause register for both IRQ and FIQ:

- A read from the IRQ Select Cause Register (N=0–3) (Table 311 p. 703) returns the following information:
  - <Stat> is a single bit field that indicates that there is a valid interrupt (meaning, there exists an interrupt with a cause and mask bit set)
  - <VecSel> is 2 bit field that defines the Main interrupt Case Vector which contains the valid interrupt. At the event of several Vectors with a valid interrupt, the Vector with smallest index is chosen.
  - <Cause> is a 29 bit field that contains a masked version of the content of Main interrupt cause Vector that is pointed by the <VecSel>.
- Similarly, a read from the FIQ Select Cause Register (N=0–3) (Table 312 p. 704), returns a main interrupt cause vector of an asserted interrupt that is not masked for FIQ.

Note

The Main interrupt cause vectors as well as the Select Cause registers are banked registers per CPU, and as such, an interrupt controller that approaches them receives the information associated with the processor on which it is running.

9.2.4.2 Hardware Interrupts Prioritization Selection Mode

Hardware Interrupts Prioritization Selection mode can be applied to IRQ interrupt selection by setting the <IRQPrioEnable> field in the MPIC Control Register (Table 296 p. 694). It can not be applied to FIQ and PCIe EP interrupts.

As described in Section 9.2.2, Interrupt Sources Control, on page 141, each interrupt source has a control register that contains a 4-bit field for its priority. In addition, each processor has a <Task Priority> field in the Current Task Priority (CTP) Register (N=0–3) (Table 313 p. 705).

The MV78230/78x60 Multiprocessor Interrupt Controller asserts IRQn of a specific processor when it has a pending unmasked interrupt event with a priority that is greater than its specific <Task Priority>. For example, if the CPU <Task Priority> value is 7, and the IDMA completion interrupt bit is set with a priority level of 5, IRQn is not asserted. However, under the same Task Priority configuration, if the USB interrupt is set with a priority of 9, IRQn is asserted.

On assertion of IRQn, any read of the IRQ Interrupt Acknowledge (IIACK) Register (N=0–3) (Table 314 p. 705) returns an ID of an unmasked pending interrupt with a priority that exceeds the <Task Priority>. If there are several pending interrupts with priorities that exceed the <Task Priority>, the pending interrupt with the highest priority will be served.

When the MV78230/78x60 Multiprocessor Interrupt Controller detects several interrupt events with the same highest priority that exceeded the <Task Priority>, a pseudo-random arbitration is performed between consecutive reads of the IRQ Interrupt Acknowledge (IIACK) Register (N=0–3)

Copyright © 2014 Marvell

(Table 314 p. 705), guaranteeing fairness among all pending interrupts.

The MV78230/78x60 Multiprocessor Interrupt Controller supplies a preemption mechanism, configured so that if, during the period lying between the IRQn assertion and the event of reading of the IRQ Interrupt Acknowledge (IIACK) Register (N=0–3), a new interrupt event with higher priority occurs, the new interrupt (that has a higher priority) is the one to be received.

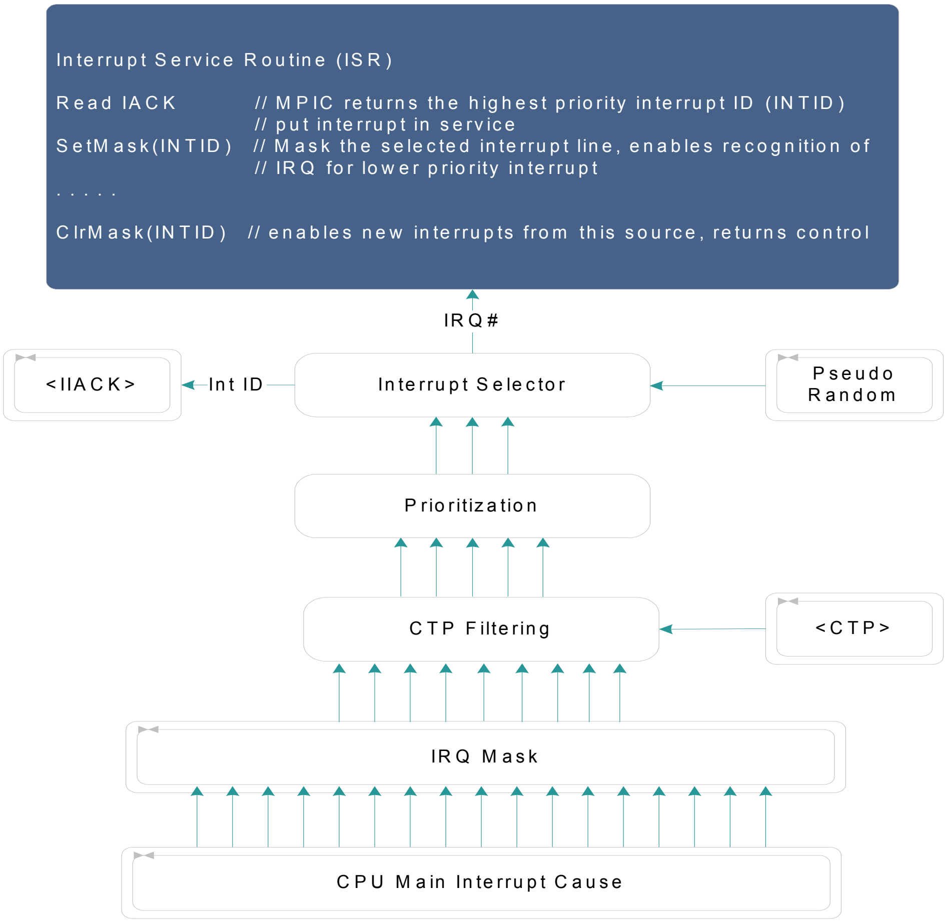
A spurious interrupt ID (ID = 1023) is returned to the CPU on reading the IRQ Interrupt Acknowledge (IIACK) Register (N=0–3) while no pending winning interrupt is listed.

The software procedure to utilize this mode is as follows:

1. Upon initialization, set the <Task Priority> field to 0. The CPU processes all interrupts. Upon any MV78230/78x60 unmasked interrupt events, IRQn is asserted.
2. When an interrupt occurs and the CPU determines the nature of the interrupt, the <Task Priority> is updated with a new value. For example, if an IDMA completion interrupt (priority 5) is received, the <Task Priority> field is set to 6. This automatically clears IRQn.
3. During processing of the interrupt, if IRQn is asserted again (which only occurs if a higher priority interrupt is asserted), the CPU toggles to process the new interrupt and updates the value of the <Task Priority> field accordingly.
4. When the CPU finishes processing an interrupt and toggles back to the lower priority interrupt, it updates the <Task Priority> field.
5. When there are no more interrupts to process, the CPU sets the <Task Priority> field to 0.

Figure 22 illustrates the IRQ prioritization flow.

Figure 22: Interrupt IRQ Processing Flow



```
Interrupt Service Routine (ISR)

Read IACK          // MPIC returns the highest priority interrupt ID (INTID)
                   // put interrupt in service
SetMask(INTID)     // Mask the selected interrupt line, enables recognition of
                   // IRQ for lower priority interrupt
. . . . .

ClrMask(INTID)   // enables new interrupts from this source, returns control
```

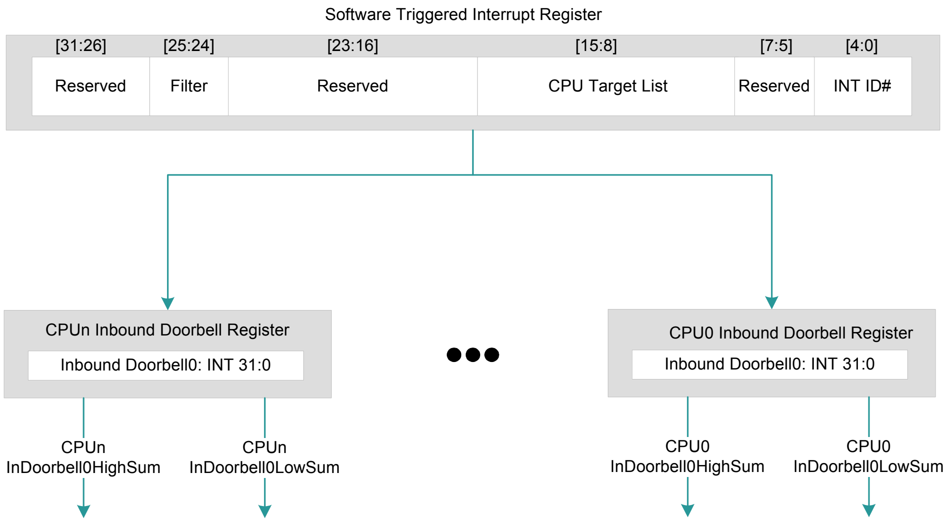
9.2.5 Inter-Processor Interrupts and Self Interrupts

The Main Interrupt Controller supplies an integrated shared mechanism for Inter-Processor Interrupts (IPIs) dispatched in the MV78230/78x60. A single write to the global Software Triggered Interrupt Register (Table 297 p. 694) delivers the required Interrupt ID <IntID> to groups of CPUs, as specified by <CPU0 in Target List>. The requesting CPU may also determine whether to perform a self-interrupt, in addition to the target list or to inclusively target itself by <Target List Filter>.

Each CPU supports up to 32 independent inbound doorbell maskable interrupts that can be simultaneously set through the global Software Triggered Interrupt Register by different requesters. Once a non-masked bit is set in the CPU's Inbound Doorbell Register (N=0–3) (Table 305 p. 701) the target CPU is interrupted to serve the pending IPIs as captured in the Inbound Doorbell Register (N=0–3) (Table 305 p. 701).

Copyright © 2014 Marvell

Figure 23: IPI Dispatching Scheme Using the Software Trigger Interrupt



9.2.6 PCIe MSI/MSI-X Capturing as Root Complex

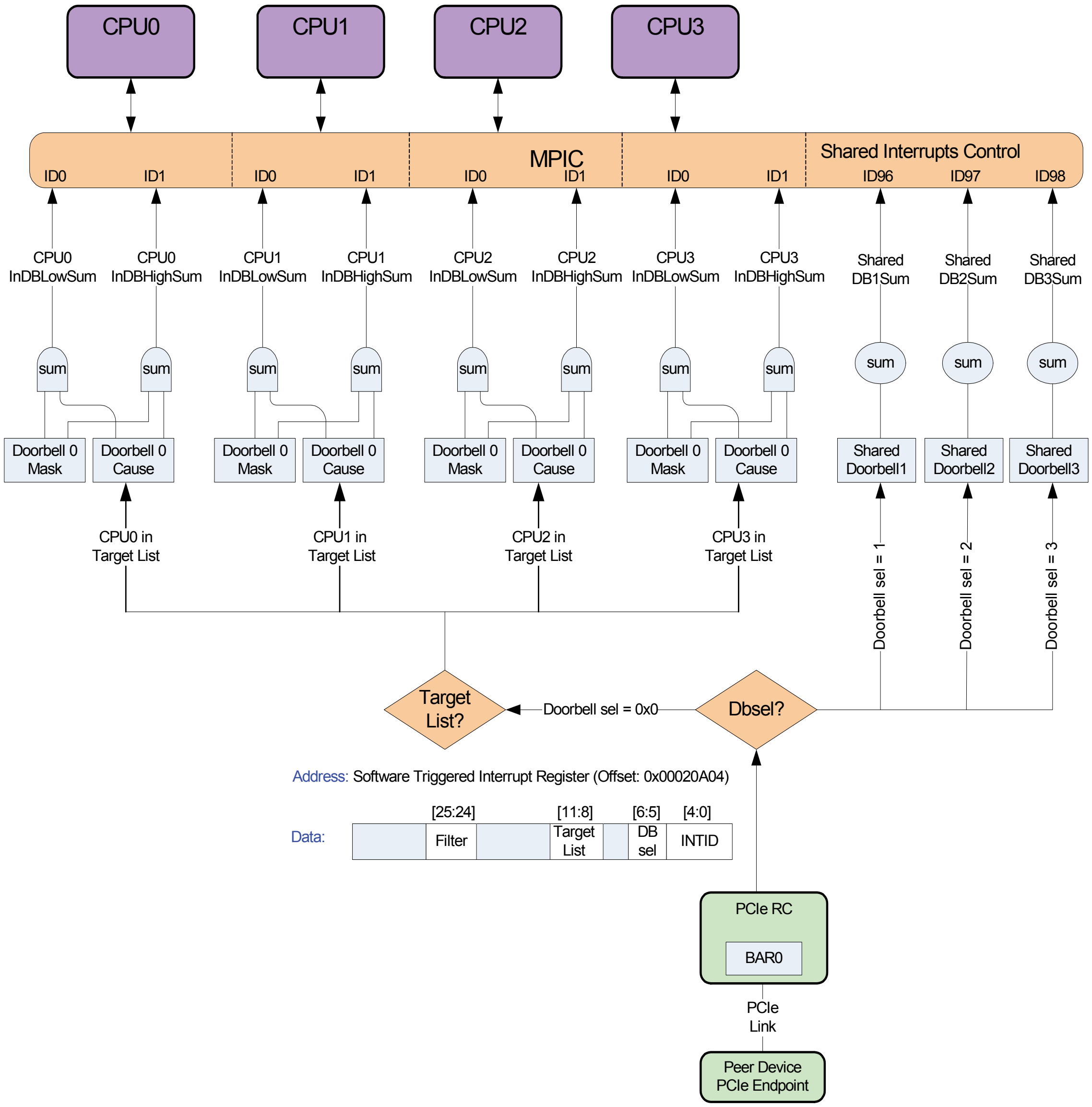
The MV78230/78x60 Interrupt Controller supplies an advanced PCIe Root Complex interrupts mechanism for mapping and capturing different pre-programmed PCIe's MSI and MSI-X messages. These messages are dispatched by a PCIe device once internal interrupt events occur, and are routed to the Root Complex's Main Interrupt Controller, as MSI or MSI-X messages.

MSI-X provides multiple interrupt vectors for each device, which allows multiple interrupts to be handled simultaneously, and load-balanced across multiple cores in multiprocessors environment. Fine-grained MSI and MSI-X interrupts capturing and delivering across the Root Complex's multiple CPUs helps to improve the CPU utilization, reduces the CPUs' interference and lowers the interrupt handling latency.

By utilizing the Software Triggered Interrupt Register (Table 297 p. 694), the MV78230/78x60 supplies a mechanism that can capture up to 128 different interrupts in the Main Interrupt Controller Inbound Doorbell registers.

Figure 24 shows the software triggered interrupt delivery scheme for serving inbound MSI/MSI-X messages.

Figure 24: MSI-X Capturing Scheme Using the Software Trigger Interrupt, Quad-CPU Configuration



9.2.6.1 Private Inbound Doorbell Messages

The CPU Private Inbound Doorbell Register (N=0–3) (Table 305 p. 701) captures message interrupts 0–31, and is banked per CPU. Every event has a mask bit in the Inbound Doorbell Mask Register (N=0–3) (Table 306 p. 702) that is also banked per CPU. This means that a mask bit blocks the event from being propagated to the specific CPU's interrupt controller.

Doorbell messages 0-15 are held in a banked private interrupt (ID0), referred to as the Inbound Doorbell Low Summary. Similarly, Doorbell messages 16-31 held in an additional banked private interrupt (ID1), referred to as Inbound Doorbell High Summary.

A peer PCIe endpoint device sets a doorbell message in the target CPU's Private Inbound Doorbell Register (N=0–3), by sending an MSI-X message with the following attributes:

- The address is set to the Software Triggered Interrupt Register (Table 297 p. 694) offset.
- Bits[4:0] are set so that the <IntID> field points to the target doorbell message.
- Bits[6:5] are set so that the <Inbound Doorbell Sel> field points to Doorbell0.
- Bits[11:8] are set so that the <CPU0 in Target List> field is high for all of the CPUs that must receive the message.
- Bits[25:24] are set to 0 (AllAndSelf) so that the <Target List Filter> field sends the interrupt to all of the CPUs that are defined in the Target List.

9.2.6.2 Shared Inbound Doorbell Messages

The Shared Inbound Doorbells capture the message interrupts that are received when <Inbound Doorbell Sel> is not set to 0. There are 96 Shared Inbound Doorbell interrupts that are partitioned into the following three groups:

Doorbell1 Shared doorbell messages 0-31

Messages are captured at Shared Inbound Doorbell0 Register (Table 326 p. 710). Summary of these messages set interrupt ID96.

Doorbell2 Shared doorbell messages 32-63

Messages are captured at Shared Inbound Doorbell1 Register (Table 327 p. 710). Summary of these messages set interrupt ID97

Doorbell3 Shared doorbell messages 64-95

Messages are captured at Shared Inbound Doorbell2 Register (Table 328 p. 710). Summary of these messages set interrupt ID98

Note

As with any shared interrupt, the group of CPUs that are interrupted, as a result of a shared doorbell summary, are defined by setting <IRQ Mask> and the <FIQ Mask> in the Interrupt Source i Control Register (i=29–115) (Table 302 p. 699).

A peer PCIe endpoint device sets a shared doorbell message by sending an MSI/MSI-X message with the following attributes:

- The address is set to the Software Triggered Interrupt Register (Table 297 p. 694) offset.
- Bits[6:5] are set so that the <Inbound Doorbell Sel> field points to the doorbell group that contains the message.
- Bits[4:0] are set so that the <IntID> field is the target doorbell message within the group defined by <Inbound Doorbell Sel>
- Bits[25:24] are set to 0 (AllAndSelf) so that the <Target List Filter> field sends the interrupt to all of the CPUs that are defined in the Target List.

Note

When triggering a shared doorbell interrupt, bits[11:8] that correspond to <CPU%N in Target List> are ignored by the hardware.

9.2.7 SoC and CPU Sub-system Error Reporting

The MV78230/78x60 includes SoC and a CPU sub-system error reporting capabilities.

9.2.7.1 SoC Error Reporting

The MV78230/78x60 Interrupt Controller uses the SOC Main Interrupt Error Cause Register (Table 298 p. 696) to latch all SoC error events. The SoC errors are also observable through this register.

These events are compiled into a single interrupt that is banked for each of the CPUs, and for the PCIe Host, if the device is configured as a PCIe endpoint.

For each of these processing elements (CPU0, CPU1, CPU2, CPU3, and PCIe Root Complex) there is a set of masks. These masks enable a static configuration to define the processing elements that are interrupted as a result of each error event. To configure the settings for these masks:

- For the CPUs, use the SOC Main Interrupt Error Mask Register (N=0–3) (Table 317 p. 707).
- For the PCIe Host, use the EP SOC Main Interrupt Error Mask Register (Table 325 p. 709).

For each of these processing elements, the interrupt source ID4 is asserted when a non-masked error event occurs.

The device errors' sources and mapping are detailed in Table 22, SoC Errors Mapping, on page 155.

9.2.7.2 CPU Sub-system Error Reporting

To report CPU sub-system errors, the MV78230/78x60 Interrupt Controller implements the following registers:

Coherency Fabric Local Cause Register (Table 164 p. 620)

This register latches all asynchronous events reported by any of the CPUs, the coherent fabric, the IO-coherent Bridge, or the shared L2. These events are compiled into a single private interrupt (ID3) that is banked for each of the CPUs, and for the PCIe Host, if the device is configured as a PCIe endpoint.

For each of these processing elements (CPU0, CPU1, CPU2, CPU3, and PCIe Root Complex) there is a set of masks to enable a static configuration that defines each event. These are the processing elements that are interrupted as a result of the event occurring. For the CPUs, the Coherency Fabric Local Interrupt Mask Register (N=0–3) (Table 318 p. 707) is the banked register that defines these settings. For the PCIe Host, EP Coherency Fabric Local Interrupt Mask Register (Table 324 p. 709) defines these settings.

Coherency Fabric Error Status Register (Table 162 p. 617)

This register latches all error events reported by the CPU and Fabric hardware components. For each of these error events, a configurable mask bit exists in the Coherency Fabric Error Mask Register (Table 163 p. 620).

The sum of all non-masked events is a single event that is referred to as the FabricErrorSum, which is then treated as an interrupt event by the <CoherencyFabricErrorSum> field in the Coherency Fabric Local Cause Register (Table 164 p. 621).

Copyright © 2014 Marvell

L2 Interrupt Cause Register (Table 292 p. 689)
This register latches all soft errors reported by L2 memories. For each of these error events, a configurable mask bit exists in the L2 Interrupt Mask Register (Table 293 p. 690).

The sum of all non-masked events is an event that is referred to as L2CacheInterrupt. This is then treated as an interrupt event by the <L2 Cache Interrupt> field in the Coherency Fabric Local Cause Register (Table 164 p. 621). For further information regarding this error event, see Section 8.4, Error Handling, on page 133.

CIB Parity Error Status Register (Table 171 p. 625)
This read-only register provides information about the memory in the CIB that was found to be a source of soft error. The sum of these events is an event that is referred to as a CIB Parity Error, which is then treated as an interrupt event by the <CPU0 DC Parity Error> field in the Coherency Fabric Error Status Register (Table 162 p. 619).

9.2.8 MV78230/78x60 Interrupts Sources

Table 21 lists the MV78230/78x60 interrupt sources and their associated ID mapping.

Table 21: Internal Interrupts Mapping

Interrupt Number	Interrupt Source	Interrupt Description
Per-CPU		
0	CPUx InDBLowSum	Inbound Doorbell Low Summary (0-15)
1	CPUx InDBHighSum	Inbound Doorbell High Summary (16-31)
2	Outbound Doorbell Summary	CPU Local Outbound Doorbell are based on 0x21070 (CPU virtual offset)
3	Coherent Fabric Local Summary	Sum of Coherency Fabric Local Cause (offset 0x20260)
4	SoC Error Summary	Sum of Coherency Fabric Error Cause (offset 0x20258)
5	Local Timer0	CPU Local timer0 interrupt comes from 0x21068[0] (CPU virtual offset)
6	Local Timer1	CPU Local timer0 interrupt comes from 0x21068[8] (CPU virtual offset)
7	Local Watchdog (WD) Timer	Each CPU has a local WD timer interrupt that exists in 0x21068[24] (CPU virtual offset). This interrupt is a sum of all local WD with the masks that are configured through the ARM WD Mask (offset 0x20708)
8	GbE TH_RXTX_Int	Gigabit Ethernet Controller 0 Rx and Tx Queue Threshold cross interrupt
9	GbE 0_RXTX_Int	Gigabit Ethernet Controller 0 Rx and Tx Interrupt
10	GbE 1_TH_RXTX_Int	Gigabit Ethernet Controller 1Rx and Tx Queue Threshold cross interrupt
11	GbE 1_RXTX_Int	Gigabit Ethernet Controller 1 Rx and Tx Interrupt
12	GbE 2_TH_RXTX_Int	Gigabit Ethernet Controller 2 Rx and Tx Queue Threshold cross interrupt

Table 21: Internal Interrupts Mapping (Continued)

Interrupt Number	Interrupt Source	Interrupt Description
13	GbE 2_RXTX_Int	Gigabit Ethernet Controller 2 Rx and Tx Interrupt
14	GbE 3_TH_RXTX_Int	Gigabit Ethernet Controller 3 Rx and Tx Queue Threshold cross interrupt
15	GbE 3_RXTX_Int	Gigabit Ethernet Controller 3 Rx and Tx Interrupt
16	GPIO per CPU [7:0] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[7:0]
17	GPIO per CPU [15:8] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[15:8]
18	GPIO per CPU [23:16] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[23:16]
19	GPIO per CPU [31:24] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[31:24]
20	GPIO per CPU [39:32] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[39:32]
21	GPIO per CPU [47:40] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[47:40]
22	GPIO per CPU [55:48] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[55:48]
23	GPIO per CPU [63:56] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[63:56]
24	GPIO per CPU [66:64] Interrupt [i]	Each CPU has a dedicated set of mask. level and interrupt cause registers that sum GPIO[66:64]
25	CPU[i] SCNT_Int	Secured Timer Interrupt For details, see Generic Timer section in the *ARMADA*® XP *MultiProcessor Core Highly Integrated Marvell*® *ARMv7 SoC Processors Datasheet*.
26	CPU[i] PCNT_Int	Physical Timer Interrupt For details, see Generic Timer section in the *ARMADA*® XP *MultiProcessor Core Highly Integrated Marvell*® *ARMv7 SoC Processors Datasheet*.
27	Reserved	-
28	CPU[i] VCNT_Int	Virtual Timer Interrupt For details, see Generic Timer section in the *ARMADA*® XP *MultiProcessor Core Highly Integrated Marvell*® *ARMv7 SoC Processors Datasheet*.
Global shared interrupts		
30	SPI0_Int	SPI0 interrupt
31	I2C0_Int	I2C0 interrupt
32	I2C1_Int	I2C1 interrupt
33	IDMA0_Int	IDMA Channel 0 interrupt
34	IDMA1_Int	IDMA Channel 1 interrupt
35	IDMA2_Int	IDMA Channel 2 interrupt
36	IDMA3_Int	IDMA Channel 3 interrupt

Table 21: Internal Interrupts Mapping (Continued)

Interrupt Number	Interrupt Source	Interrupt Description
37	Global_Timer0_int	Timer 0 interrupt
38	Global_Timer1_int	Timer 1 interrupt
39	Global_Timer2_int	Timer 2 interrupt
40	Global_Timer3_int	Timer 3 interrupt
41	UART0_Int	UART 0 interrupt
42	UART1_Int	UART 1 interrupt
43	UART2_Int	UART 2 interrupt
44	UART3_Int	UART 3 interrupt
45	USB0_int	USB 0 core interrupt
46	USB1_int	USB 1 core interrupt
47	USB2_int	USB 2 core interrupt
48	CESA0_Int	Cryptographic Engine and Security Accelerator 0 completion interrupt
49	CESA1_Int	Cryptographic Engine and Security Accelerator 1 completion interrupt
50	RTC_Int	RTC interrupt
51	XOR0_Ch0_Int	XOR 0 Channel 0 interrupt
52	XOR0_Ch1_Int	XOR 0 Channel 1 interrupt
53	BM_Int	Buffer Management interrupt
54	SDIO_int	SDIO interrupt
55	SATA0_int	SATA port 0 interrupt
56	TDM_int	TDM unit interrupt
57	SATA1_int	SATA port 1 interrupt
58	PCIe00_Int	PCI Express Port0.0 INTA/B/C/D assert message interrupt.
59	PCIe01_Int	PCI Express Port0.1 INTA/B/C/D assert message interrupt.
60	PCIe02_Int	PCI Express Port0.2 INTA/B/C/D assert message interrupt.
61	PCIe03_Int	PCI Express Port0.3 INTA/B/C/D assert message interrupt.
62	PCIe10_Int	PCI Express Port1.0 INTA/B/C/D assert message interrupt.
63	PCIe11_Int	PCI Express Port1.1 INTA/B/C/D assert message interrupt.
64	PCIe12_Int	PCI Express Port1.2 INTA/B/C/D assert message interrupt.
65	PCIe13_Int	PCI Express Port1.3 INTA/B/C/D assert message interrupt.

Table 21: Internal Interrupts Mapping (Continued)

Interrupt Number	Interrupt Source	Interrupt Description
66	GbE 0_Legacy_Sum_Int	Legacy interrupt pin Represents the <EtherIntSum> field in the Port Interrupt Cause register.
67	GbE 0_Legacy_RX_Int	Legacy interrupt pin. It represents the following interrupt cause register fields: In the Port Interrupt Cause register, the fields: <CRCErr>, <RxErrorQueue[x]>, <RxError>, and <RxBufferQueue[x]> (coaliasced). In the Port Interrupt Cause Extended register, the field: <RxOVR>.
68	GbE 0_Legacy_TX_Int	Legacy interrupt pin. It represents the following interrupt cause register fields: In the Port Interrupt Cause register, the fields: TxEndx In the Port Interrupt Cause Extended register, the fields: <TxErrorx>, <TxUdr>, and <TxBuffer[x]> (coalesced).
69	GbE 0_Misc_Int	Works both for Legacy and Enhanced mode. It represents the following Port Interrupt Cause Extended register fields: <PhySTC>, <PTP>, <LinkChange>, <InternalAddrError>, <PRBSError>.
70	GbE 1_Legacy_Sum_Int	Same as port 0
71	GbE 1_Legacy_RX_Int	Same as port 0
72	GbE 1_Legacy_TX_Int	Same as port 0
73	GbE 1_Misc_Int	Same as port 0
74	GbE 2_Legacy_Sum_Int	Same as port 0
75	GbE 2_Legacy_RX_Int	Same as port 0
76	GbE 2_Legacy_TX_Int	Same as port 0
77	GbE 2_Misc_Int	Same as port 0
78	GbE 3_Legacy_Sum_Int	Same as port 0
79	GbE 3_Legacy_RX_Int	Same as port 0
80	GbE 3_Legacy_TX_Int	Same as port 0
81	GbE 3_Misc_Int	Same as port 0
82	GPIO_0_7_int	GPIO pins [7:0] interrupt
83	GPIO_8_15_int	GPIO pins [15:8] interrupt
84	GPIO_16_23_int	GPIO pins [23:16] interrupt
85	GPIO_24_31_int	GPIO pins [31:24] interrupt
86	Reserved	Reserved
87	GPIO_32_39_int	GPIO pins [39:32] interrupt
88	GPIO_40_47_int	GPIO pins [47:40] interrupt
89	GPIO_48_55_int	GPIO pins [55:48] interrupt

Table 21: Internal Interrupts Mapping (Continued)

Interrupt Number	Interrupt Source	Interrupt Description
90	GPIO_56_63_int	GPIO pins [63:56] interrupt
91	GPIO_64_66_int	GPIO pins [66:64] interrupt
92	SPI1 Intr	SPI port 1 interrupt
93	WD Intr	Global Watchdog interrupt
94	XOR1_Ch2_Int	XOR 1 Channel 2 interrupt
95	XOR1_Ch3_Int	XOR 1 Channel 3 interrupt
96	Shared DB1sum	Shared Inbound Doorbell1 Summary. MSI/MSIx shared interrupt 0
97	Shared DB2sum	Shared Inbound Doorbell2 Summary. MSI/MSIx shared interrupt 1
98	Shared DB3sum	Shared Inbound Doorbell3 Summary. MSI/MSIx shared interrupt 2
99	PCIe20_Int	PCI Express Port2.0 INTA/B/C/D assert message interrupt
100	Reserved	Reserved
101	Reserved	Reserved
102	Reserved	Reserved
103	PCIe30_Int	PCI Express Port3.0 INTA/B/C/D assert message interrupt.
104	Reserved	Reserved
105	Reserved	Reserved
106	Reserved	Reserved
107	PMU_Int	Power Management Unit interrupt
108	DRAM_Int	DRAM power modes interrupt
109	GbE0_WakeUp_Int	GigE port 0 WOL interrupt
110	GbE1_WakeUp_Int	GigE port 1 WOL interrupt
111	GbE2_WakeUp_Int	GigE port 2 WOL interrupt
112	GbE3_WakeUp_Int	GigE port 3 WOL interrupt
113	NAND_Int	NAND Flash Controller interrupt
114	Reserved	Reserved
115	Reserved	Reserved

Table 22 details the MV78230/78x60 SoC Errors and their mapping to the SOC Main Interrupt Error Cause Register (Table 298 p. 696).

Table 22: SoC Errors Mapping

Interrupt Number	Error Interrupt Source	Error Interrupt Description
0	CESA0_Err_Int	Crypto engine 0 error interrupt
1	DevBus_Err_Int	Device bus error interrupt (DEVICE_READYn Timer)

Table 22: SoC Errors Mapping (Continued)

Interrupt Number	Error Interrupt Source	Error Interrupt Description
2	IDMA_Err_Int	DMA error (address decoding and protection)
3	XOR1_Err_Int	XOR engine 1 error
4	PCIe0_Err_Int	PCI Express port 0 Error (summary of PCI Express Cause register)
5	PCIe1_Err_Int	PCI Express port 1 Error (summary of PCI Express Cause register)
6	GbE_Err_Int	Gigabit Ethernet error (address decoding and protection) Summary of Ethernet Unit Interrupt Cause (EUIC) registers of all GbE ports
7	CESA1_Err_Int	Crypto engine 1 error interrupt
8	USB_Err_Int	USB error (address decoding and RAM parity) Summary of errors from all USB ports
9	DRAM_Err_Int	DRAM ECC error interrupt
10	XOR0Err_Int	XOR engine 0 error interrupt
11	Reserved	Reserved
12	BM_Err_Int	Buffer Management error interrupt
13	CIB_Err_Int	Coherent IO Bridge error interrupt
15	PCIe2_Err_Int	PCI Express port 2 Error (summary of PCI Express Cause register)
16	PCIe3_Err_Int	PCI Express port 3 Error (summary of PCI Express Cause register)
17	SATA0_Err_Int	SATA unit 0 error interrupt
18	SATA1_Err_Int	SATA unit 1 error interrupt
19	Reserved	Reserved
20	TDM_Err_Int	TDM error interrupt
21	NAND_Err_Int	NAND controller error interrupt
22	Reserved	Reserved
23–31	Reserved	Reserved

10 Timers, Counters, and Watchdog

The device incorporates two levels of timers/counters sets, a global set and a multiprocessor topology private set.

Global The global timers/counters include four 32-bit wide general purpose timers/counters and one watchdog timer/counter. Each can be selected to operate as a timer or as a counter. Each timer/counter can be configured to work with a 25 MHz frequency or at a specific ratio of the coherency fabric clock (NBCLK/2).

- To select the 25 MHz frequency, set the relevant 25 MHz frequency enable field (bits[14:10]) in Timers Control Register (Table 329 p. 710).
- To set a specific ratio of the coherency fabric clock for the watchdog or specific CPU timer, set the relevant field (bits[30:16]) in the Timers Control Register.

CPU Private The private timers/counter include two 32-bit wide general purpose timers/counters and one watchdog timer/counter, that are assigned per CPU core. These are private timers/counters that are not visible to other CPU cores. Each timer/counter can be configured to work with a 25 MHz frequency or at a specific ratio of the coherency fabric clock (NBCLK/2).

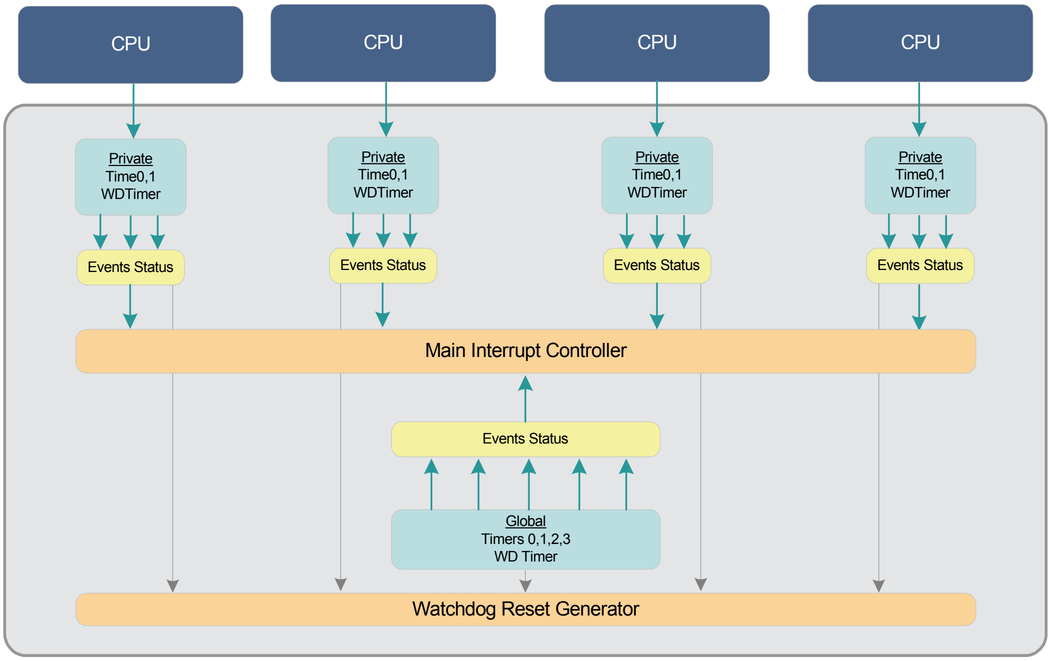
- To select the 25 MHz frequency, set the relevant 25 MHz frequency enable field (bits[12:10]) in the Timers Control Register (N=0–3) (Table 248 p. 660)**.**
- To set a specific ratio of the coherency fabric clock for the watchdog or specific CPU timer, set the relevant field (bits[24:16]) in the Timers Control Register (N=0–3).

Note The watchdog timer functionality is the same as the other two CPU private timers, with the exception that it can also generate a reset event (see the Reset section in the device's Hardware Specifications).

Copyright © 2014 Marvell

Figure 25 details the multiprocessor timers topology, including the private and global partitioning and events connectivity to the main interrupt controller.

Figure 25: Multiprocessor Timers Topology



10.1 Features

The timers, counters and watchdog have the following features:

- A 32-bit counter that generates an interrupt when it reaches zero
- An 8-bit prescaler to enable better control of the period
- Configurable single-shot or auto-reload modes
- Configurable starting values for the counter
- Configurable reset generation on Watchdog Timer expiration

10.2 Functional Description

This section describes the timer and counter modes and the functions of the global, private and watchdog timer and counter.

10.2.1 Timer/Counter Modes

Each Timer/Counter delivers the following functionality.

Counter Mode In the Counter mode, the counter counts down, stops when it reaches 0, and issues an interrupt

Setting the <CPUTimer*Auto> bit in each CPU Timer Control Register to 0 programs it to Counter mode

Timer Mode In the Timer mode, the timer counts down, issues an interrupt when it reaches 0, reloads itself to the Reload register's value, and continues counting.

Setting the <CPUTimer*Auto> bit in each CPU Timer Control Register to 1 programs it to Timer mode

Each counter/timer has a Reload register and a Timer register. The Reload register represents the initial value to be loaded to the timer/counter. The Timer register represents the timer/counter itself. The software polls this register to determine the actual timer value.

Each CPU has the option to mask the interrupt asserted by a timer/counter expiration. This applies for both the global timers/counters and the private CPU timers/counters.

For example, it is possible to assign global timer 0 to CPU0 by masking the interrupt of this counter for CPU1. Similarly, it is possible to assign timer1 to CPU1 by masking its interrupt to CPU0.

10.2.2 Global Timers/Counters

The four timers/counters and one watchdog timer:

- All of the timers are accessible and shared among all the CPUs.
- Each global timer can be programmed to interrupt a specified CPU, or a group of CPUs, through the main interrupt controller with appropriate priority. The main interrupt controller delivers the interrupt to the attached CPUs, with respect to running priority.

The global set is controlled through the following registers:

Timers Control Register Includes all the timers/counters enable and control functions. This includes a per counter enable, mode of operation, and an actual counter updating period relative to its source clock.

Timers Events Status register Register Summarizes the expiration status for all of the timers. This is an RW0C type register. A corresponding interrupt is asserted to the CPUs, if enabled in the main interrupt controller. The interrupt handler de-asserts the active global timer interrupt line by clearing the corresponding timer expiration event in this register.

Timer0 Reload Register and Timer 0 Register The timer 0 register couple. The same set exists for the other global timers and watchdog timer.

10.2.3 Private Timers/Counters

The private set includes two timers/counters and one watchdog timer per CPU.

- Each CPU timer set is placed in different register banks. It is virtually accessed with the same address from the CPU it is tied to.
- All the CPUs private timers sets are independent, with respect to each other, and expiration event interrupts only the CPU it belongs to through the main interrupt controller private interrupt lines. Timer expiration events are virtually mapped to same interrupt line, for example. CPU0 timer0 expiration asserts CPU0 private interrupt line ID and only its interrupt, while CPU1 timer1 expiration asserts same interrupt line ID but to CPU1 only.

Each CPU private set is controlled through the following registers:

Timers Control Register (N=0–3) Includes all the timers/counters enable and control functions. This includes a per counter enable, mode of operation, and an actual counter updating period relative to its source clock.

Timers Events Status register Register (N=0–3) Summarizes each CPU timers expiration status. This is an RW0C type register. A corresponding interrupt is asserted to the specified CPU, if enabled in the main interrupt controller. The interrupt handler de-asserts the active private timer interrupt line by clearing the corresponding timer expiration event in this register.

Timer0 Reload Register (N=0–3) and Timer0 Reload Register The CPU Timer 0 register couple. The same set also exists for Timer 1 and the Watchdog timer.

10.2.4 Watchdog Reset

For each CPU, the MV78230/78x60 includes a private watchdog timer and a global watchdog timer that can be independently activated, reloaded, and expired. The device delivers flexibly programmable options for mapping the different watchdog timers expiration events to single system watchdog reset event.

The following registers controls the generation of watchdog expiration reset.

The <CPU WD Group> field in the WD RSTOUTn Mask Register (Table 347 p. 719) defines the CPUs that participate in the watchdog group. A watchdog reset is generated on the timers expiration of all the group members.

In addition to this mechanism, every CPU and the global watchdog has an independent mask bit. Once set, a reset is generated upon expiration, independent to the group value. This is controlled through programming of <CPU0 WD mask> and <Global WD mask> field in the WD RSTOUTn Mask Register (Table 347 p. 719).

For example, when a watchdog reset is required on expiration of the global watchdog timer or both CPU0 and CPU1 private watchdog timers, enable the <Global WD mask> and program the <CPU WD Group> to include both CPU0 and CPU1.

10.2.5 Watchdog Interrupt

Watchdog timers expiration can be configured to generate an interrupt.

The <CPUn Local WD Interrupt Mask> field in the WD Interrupt Mask Register (Table 348 p. 719) defines for each CPU which of the Private Watchdog timers expiration generates an interrupt to the corresponding CPU.

For example, when the expiration of the Private Watchdog of CPU0 requires an interrupt to CPU1 and the expiration of the Private Watchdog of CPU1 requires an interrupt to CPU0, set bit[1] in the <CPU0 Local WD Interrupt Mask> and set bit[0] in the <CPU1 Local WD Interrupt Mask>.

11 Debug Capabilities

This section describes the debug modes of MV78230/78x60 device.

See Table 2, Terms and Abbreviations, on page 29 for the definitions of terms used in this section.

The MV78230/78x60 implements the CoreSight™ debug subsystem. CoreSight[1] is an extensible, system-wide debug and trace architecture of ARM. The architecture consists of a user-visible portion specifying a programmer's model, and an architecture portion specifying standard interfaces for transmission of configuration commands, trace information, and cross-trigger signaling between different components.

A CoreSight system may contain differing numbers and types of components, connected in different topologies. CoreSight defines the AMBA® Trace Bus (ATB), a standard interface on which to generate and receive trace information. CoreSight components, which can be configured, each have an Advanced Peripheral Bus (APB) programming interface for register state.

In the MV78230/78x60 CPU Subsystem, one ATB / ETB (Embedded Trace Buffer) per core configuration and a single Trace Port Interface Unit (TPIU) with attached Serial Embedded Trace Macrocell (sETM) has been used as shown in Figure 26.

[1] CoreSight and AMBA are trademark/registered trademark of the ARM Corporation.

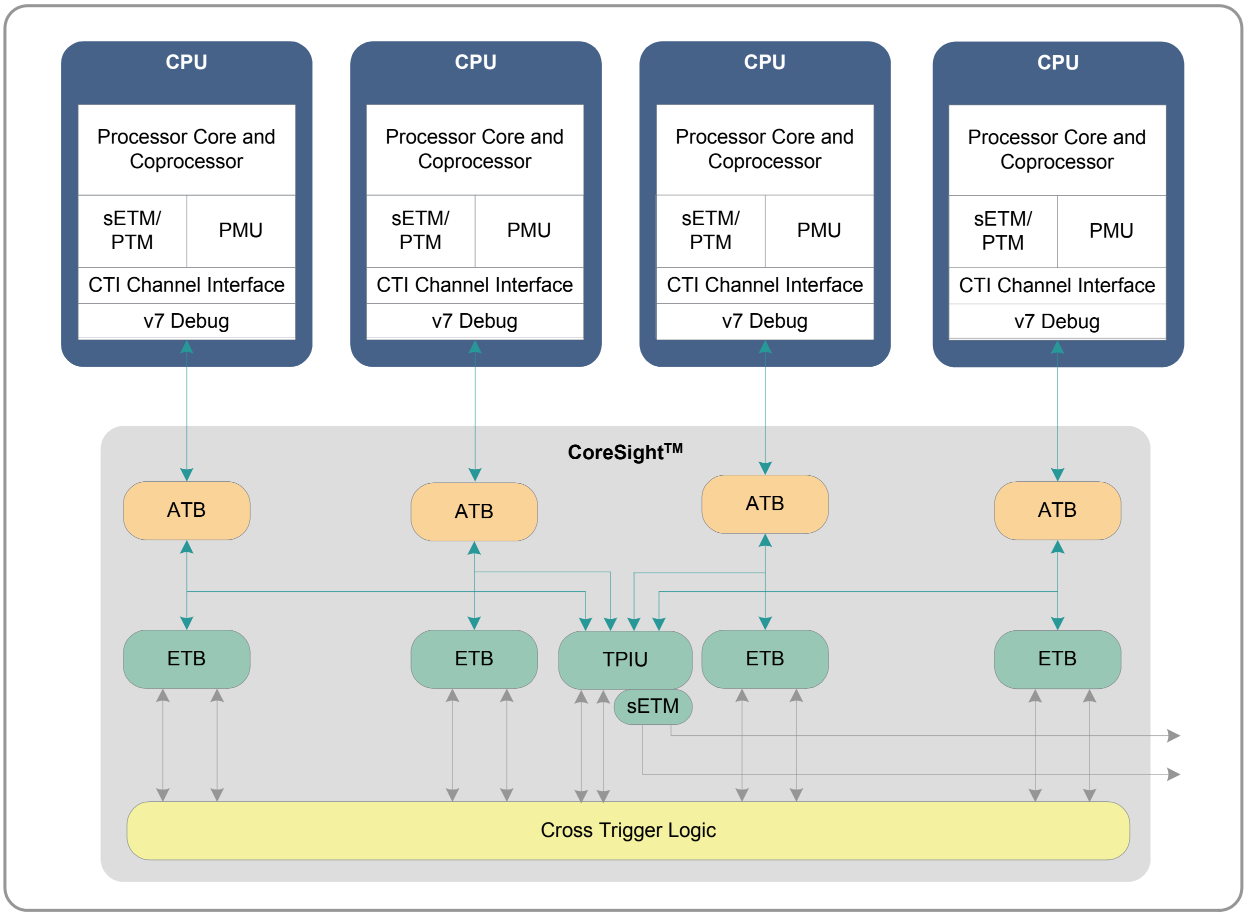


Figure 26: MV78230/78x60 CoreSight Configuration

A directory of CoreSight blocks in the system is also provided in the form of a ROM within one of the CoreSight components called the CoreSight Debug Access Port (DAP). This ROM is accessible over the APB programming interface and contains pointers to the standard register space for each component. Figure 27 shows the CoreSight DAP and APB interface. Debug tools can read the ROM and the standard component ID registers to determine what the MV78230/78x60 debug system contains.

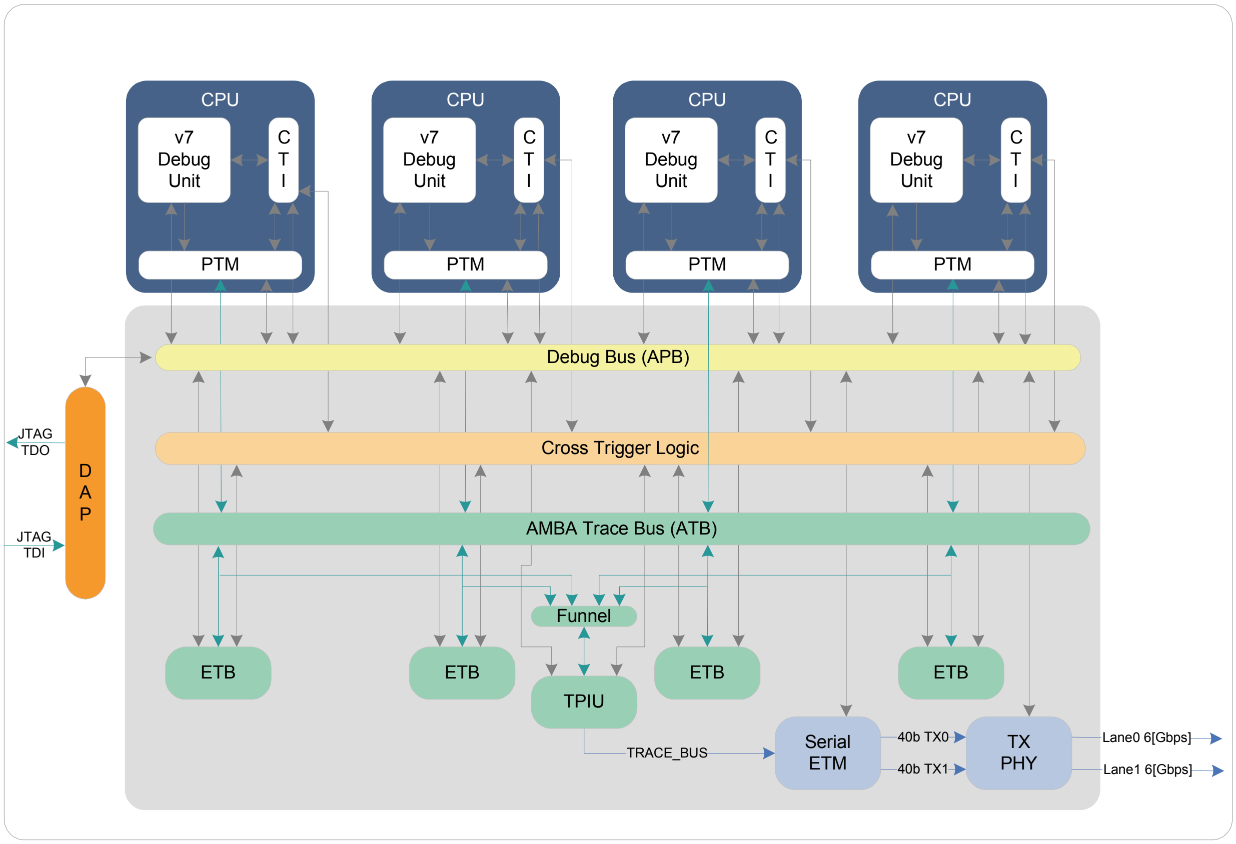
MV78230/78x60 is CoreSight compliant[1], representing itself to the debug system as 4 CoreSight components. In v7 mode, it implements four 4-KB CoreSight register programming regions per core on its debug APB interface. One region for each of the following functions:

- Core registers with invasive debug functionality
- On-board PTM (Program Trace Macrocell)
- On-board Cross Trigger Interface (CTI) that facilitates cross-triggering between core logic, the PTM, and other CoreSight components in the system
- On-board trace sinks, including an ETB per CPU and a central TPIU and sETM for extracting traces serially to an external debugger

The Core Subsystem with these components is shown in more detail in Figure 27.

[1] For details about CoreSight functionality, see the CoreSight Architecture Specification.

Figure 27: CoreSight Subsystem v7 Mode Details



MV78230/78x60 trace subsystem incorporates four PTMs as the trace source—one for each of the cores—and two CTIs that facilitate cross-triggering between the debug units and the PTM trace sources and trace sinks. The PTMs ATB interface is connected directly to the four ETBs and is also replicated to a single TPIU through the Funnel[1]. This interface transmits data using the sETM in two transmission lanes at 6 Gbps each. The DAP's APB programming interface output port is connected to the APB programming interfaces in the four cores, the four ETBs, the Funnel, the TPIU, the sETM and the two external CTIs.

Figure 28 illustrates details of CTI / CTM (Cross Trigger Module) connections, using the CTI Channel interfaces. The two CTIs external to the four cores are connected to the channel interface extending from the four core's CTI via three CTMs. This allows the input and output trigger signals of the ETB and TPIU to be mapped to trigger inputs and outputs of the CTI.

Figure 28: CTI / CTM Connection Details

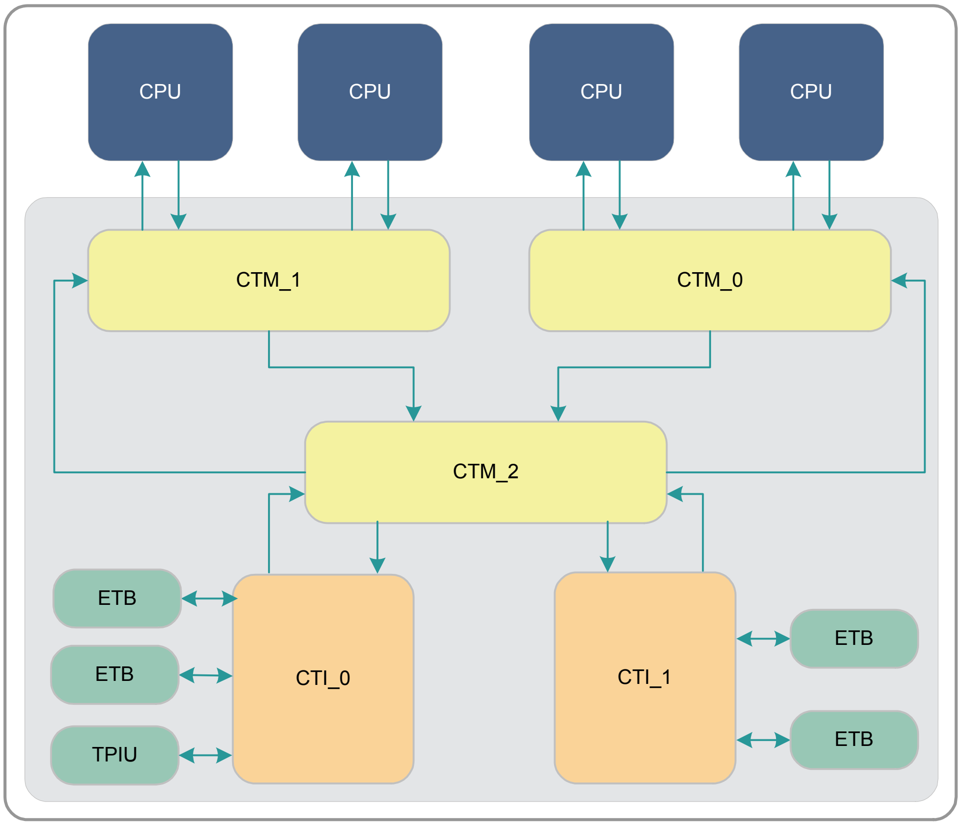


Table 23: Trigger In Trigger Out of CTI0

Index	Trigger In	Trigger Out
0	**ETB0_FULL** Indicates that the ETB RAM has overflowed or wrapped around to address zero.	**ETB0_FLUSHIN** Drains any historical FIFO information on the bus.
1	**ETB0_ACQCOMP** Indicates that trace acquisition is complete.	**ETB0_TRIGIN** Enables the ETB to initiate the collection of ATB trace data.
2	**ETB1_FULL** Same as ETB0	**ETB1_FLUSHIN** Same as ETB0
3	**ETB1_ACQCOMP** Same as ETB0	**ETB1_TRIGIN** Same as ETB0
4	Reserved	**TPIU_FLUSHIN** Invokes an ATB signal and drain any old historical information on the bus.

Table 23: Trigger In Trigger Out of CTI0 (Continued)

Index	Trigger In	Trigger Out
5	Reserved	**TPIU_TRIGIN** Enable the trigger to affect the output trace stream.
6	Reserved	Reserved
7	Reserved	Reserved

Table 24: Trigger In Trigger Out of CTI1

Index	Trigger In	Trigger Out
0	**ETB2_FULL** Same as ETB0	**ETB2_FLUSHIN** Same as ETB0
1	**ETB2_ACQCOMP** Same as ETB0	**ETB2_TRIGIN** Same as ETB0
2	**ETB3_FULL** Same as ETB0	**ETB3_FLUSHIN** Same as ETB0
3	**ETB3_ACQCOMP** Same as ETB0	**ETB3_TRIGIN** Same as ETB0
4	Reserved	Reserved
5	Reserved	Reserved
6	Reserved	Reserved
7	Reserved	Reserved

Note

For additional information about CTI and about trigger inputs and outputs, refer to the *Cross Trigger Interface* section of the *ARMADA*® XP *MultiProcessor Core Highly Integrated Marvell*® *ARMv7 SoC Processors Datasheet*.

11.1 Debug and Power Saving Modes

MV78230/78x60 CoreSight™ debug subsystem is capable of powering up SYSTEM and DEBUG domains through the DAP component that is always on.

- The SYSTEM domain is defined in respect to DAP to all CPU and Fabric power domains that are membered within the wake group (see the L2C NFabric PM Status and Mask Register (Table 430 p. 759)).
- The DEBUG domain is defined in respect to DAP to Fabric power domain that is membered within the wake group (see the L2C NFabric PM Status and Mask Register).

The MV78230/78x60 allows continuing debug over powered up CPUs as usual in case one or more other CPUs are powered down.

Power down and Power up sequence emulation is possible per CPU by setting it to Debug No Power Down mode (configured by APB-to-CPU debug unit Device Power-down and Reset Control register[1] at offset 0x310). In this mode, when the CPU is instructed to power down the Fabric's power controller does not disconnect the power, but allows the debug system to emulate the sequence that powers it down and power it up afterwards.

APB accesses to powered down components are replied to by the APB slave error response.

Note

When a dedicated CPU is powered down, its internal CoreSight components are also powered down, so the APB sequence from DAP are replied to with the APB slave error.

11.2 Serial Tracing

In addition to the tracing through embedded the Trace Buffers, the MV78230/78x60 enables high-speed serial tracing.

The Serial Tracing is applied by the sETM unit that is attached to the CoreSight™ TPIU that takes the incoming ATB packets and extracts them as a Trace Bus formation. The incoming Trace Bus information is then serialized by the sETM device that transmits the data over two SERDES transmission lanes at a rate of 6 Gbps. The sETM and attached PHYs per lane are managed by the CoreSight™ DAP APB interface. For their specific registers, refer to the *Serial ETM3 Common Physical (PHY) Layer Preliminary Specifications*.

11.3 System Debug Management

Within the DAP CoreSight™ component, there is an internal APB Multiplexor that enables external tools and system access to the CoreSight Debug APB. In the MV78230/78x60 CoreSight™ Debug subsystem any CPU is capable of accessing to any pre-detected CoreSight component as an external debugger does through APB. Before accessing a specific CoreSight component register, the CPU first need to set the CoreSight Components Base Address. Then, for read or write to specific component register, it should access the MultiProcessor CoreSight Components and with the exact address offset gain access to the specific component register.

For example, to read-modify-write the PTM register ETMCR (offset 0x000) that exists within CPU3 and enable its Timestamp feature, using APB accesses initiated by CPU0 and then reading the CPU3 ETMIDR register:

```
ldr r0, =Coresight Components Base Address Register (Table 359 p. 726).

ldr r1, =0x42313000 ; CPU3 PTM trace source base address as detected previously by ROM table accesses.

str r1, [r0] ; All future APB accesses are directed to CPU3 PTM.

ldr r0, =0xd0023000 ; <0xd0023>-APB master base address, <xxx> register offset of Coresight component.

ldr r1, [r0] ; reading CPU3 PTM ETMCR.

orr r1, r1, #0x10000000

str r1, [r0] ; Turning on CPU3 PTM <Timestamp Enable> bit at ETMCR register.

;; In case wish to read CPU3 PTM ETMIDR register at offset 0x1E4 all need to to do is:

ldr r0, =0xd00231e4

ldr r1, [r0] ;  CPU3 PTM ETMIDR content is existing within CPU0 <r1> register.
```

[1] Refer to the Debug section of the *ARMADA*® XP *MultiProcessor Core Highly Integrated Marvell*® *ARMv7 SoC Processors Datasheet*.

11.4 Timestamp Mechanism

Timestamping is a mechanism that periodically inserts a time value into the trace stream. The MV78230/78x60 implements timestamp 48-bit counter to provide the source of the timestamp values It must broadcast the same value to all compatible trace sources in the system.

Each trace source samples the timestamp value and inserts it as an absolute value in the trace stream.

Each timestamp value inserted in the trace is a gray-coded number. Each Program (Flow) Trace Macrocell (PTM) outputs the 48-bit gray code and the debugger generates the 48-bit binary number from the traced 48-bit gray code.

This timestamping mechanism provides the following features:

- Correlation of multiple independent trace sources in a system, for example, multiple PTM's in a multiprocessor environment
- Simple analysis of code performance, with a coarse granularity
- Faster searching of large trace buffers when looking for points in multiple-trace streams that were executed in close proximity

The timestamp indicates the time the packet is generated, not the time the request was made. A timestamp is only a time indicator inserted into the trace stream near the event that requested the timestamp.

Timestamp Register Summary

A list of the timestamp registers appears Table 25. For a full description of the timestamp registers, see Appendix A.2, CPU Sub-System Registers, on page 600.

Table 25: Timestamp Register Summary

Address Offset	Register Name	Type	Reset	Description
Configuration				
0x000	Timestamp Counter Enable Register	RW	0x0	Enables the 48-bit timestamp counter
0x004	Timestamp Counter Preload 1 Register	RW	0x0	Preloads the [31:0] bits of timestamp counter
0x008	Timestamp Counter Preload 2 Register	RW	0x0	Preloads the [47:32] bits of the timestamp counter
0xFA0	Timestamp Claim Tag Set Register	RW	0x0	See *CoreSight Design Kit TRM*
0xFA4	Timestamp Claim Tag Clear Register	RW	0x0	
0xFB0	Timestamp Lock Access Register	WO	-	
0xFB4	Timestamp Lock Status Register	RO	0x3	
0xFB8	Timestamp Authentication Status Register	RO	0x0	
0xFC0–0xFC4	Reserved	-	-	
Device				
0xFC8	Timestamp Device ID Register	RO	0x0	

Table 25: Timestamp Register Summary (Continued)

Address Offset	Register Name	Type	Reset	Description
0xFCC	Timestamp Device Type Identifier Register	RO	0x4	
Peripheral Identification				
0xFD0	Timestamp Peripheral ID4 Register	RO	0x05	
0xFD4	Timestamp Peripheral ID5 Register	RO	0x9A	
0xFD8	Timestamp Peripheral ID6 Register	RO	0x0E	
0xFDC	Timestamp Peripheral ID7 Register	RO	0x00	
0xFE0	Timestamp Peripheral ID0 Register	RO	0x03	
0xFE4	Timestamp Peripheral ID1 Register	RO	0x00	
0xFE8	Timestamp Peripheral ID2 Register	RO	0x00	
0xFEC	Timestamp Peripheral ID3 Register	RO	0x00	
Component Identification				
0xFF0	Timestamp Component ID0 Register	RO	0x0D	
0xFF4	Timestamp Component ID1 Register	RO	0x90	
0xFF8	Timestamp Component ID2 Register	RO	0x05	
0xFFC	Timestamp Component ID3 Register	RO	0xB1	

11.5 Memory Map for CoreSight Components

Internally, the CoreSight components (four ETBs, two CTIs and a single TPIU) and the MultiProcessor Core's CoreSight components (debug registers, Cross Trigger Interface registers, and PTM registers) can be configured through an APB programming interface. The programming model assigns 4 KB of contiguous memory-mapped space for each of these components. Some standard registers are defined for the blocks, including ID registers for the purpose of allowing tools to determine the attributes of each component without requiring prior knowledge of the component’s address. The CoreSight DAP has a ROM table containing a pointer to the base address of each of these components relative to the location of the ROM table. Debugger software should use the memory map of Table 26 to access each of the above registers[1].

Table 26: Summary of CoreSight Memory Mapped Registers—v7 Mode

Physical Address	Register Name
0x4230_0000	CoreSight DAP ROM table location
0x4230_1000	Core-0 DBG Registers
0x4230_2000	Core-0 PTM Registers
0x4230_3000	Core-0 CTI Registers

[1] For more details on the registers for CoreSight components refer to the CoreSight Architecture Specification

Copyright © 2014 Marvell

Table 26: Summary of CoreSight Memory Mapped Registers—v7 Mode (Continued)

Physical Address	Register Name
0x4230_4000	Core-0 Performance Monitor Unit Registers
0x4230_5000	CoreSight ETB-0 Registers
0x4230_6000	CoreSight CTI-0 Registers
0x4230_7000	Core-1 DBG Registers
0x4230_8000	Core-1 PTM Registers
0x4230_9000	Core-1 CTI Registers
0x4230_A000	Core-1 Performance Monitor Unit Registers
0x4230_B000	CoreSight ETB-1 Registers
0x4230_C000	Core-2 DBG Registers
0x4230_D000	Core-2 PTM Registers
0x4230_E000	Core-2 CTI Registers
0x4230_F000	Core-2 Performance Monitor Unit Registers
0x4231_0000	CoreSight ETB-2 Registers
0x4321_1000	CoreSight CTI-1 Registers
0x4231_2000	Core-3 DBG Registers
0x4231_3000	Core-3 PTM Registers
0x4231_4000	Core-3 CTI Registers
0x4231_5000	Core-3 Performance Monitor Unit Registers
0x4231_6000	CoreSight ETB-3 Registers
0x4231_7000	Timestamp
0x4231_8000	TPIU Funnel
0x4231_9000	TPIU
0x4231_A000	sETM
0x4231_B000	sETM Lane0 PHY (see Appendix A.9.9, PCI Express Communication PHY, on page 1183) **NOTE:** The APB slave not part of CoreSight component list.
0x4231_C000	sETM Lane1 PHY (see Appendix A.9.9, PCI Express Communication PHY, on page 1183) **NOTE:** The APB slave not part of CoreSight component list.

Note

For additional information, refer to the Debug section of the *ARMADA*® XP *MultiProcessor Core Highly Integrated Marvell*® *ARMv7 SoC Processors Datasheet*.

THIS PAGE IS INTENTIONALLY LEFT BLANK

MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Part 3: External Memory Interfaces

- Section 12, DRAM Controller
- Section 13, NAND Flash Controller (NFC)
- Section 14, Device Bus Controller

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. Moving Forward Faster

Document Classification: Proprietary Information

THIS PAGE IS INTENTIONALLY LEFT BLANK

12 DRAM Controller

This section describes the device's DRAM Controller. This JEDEC compatible controller:

- Supports DDR3, DDR3L modes of operation.
- Provides control over multiple physical ranks (Chip Selects), and interfaces various types of x8 and x16 DRAM components.
- Full support for all major DIMM types by the DRAM controller including unbuffered and registered DIMMs.

The Marvell proprietary DDR link training algorithm enables the device to reach the highest possible DRAM speeds.

Special attention and features were integrated to enable heavy-load topologies (that is, multiple DRAM device topologies) while running at high speeds to allow maximum memory space for memory intensive applications.

The diverse power management options offered by the DRAM controller, significantly minimize both chip level and system level power consumption.

The registers relating to the SDRAM controller are located in Appendix A.3, DRAM Controller Registers, on page 763.

12.1 Feature List

The DRAM controller supports the following features:

- DDR3/DDR3L Controller
- 64/32-bit interface with ECC option
- SSTL 1.5V/1.35V I/O for DDR3
- 4 SDRAM ranks (CSs)
- Support all DDR3 components densities up to 8 Gb
- Support all DDR3/DDR3L components densities
- Supports DRAM bank interleaving
- Up to 32 simultaneous open pages in DDR3
- Auto calibration of I/Os output impedance
- Support DIMM configurations (Registered and Unbuffered, x8 or x16 devices)
- DDR3 address mirroring support
- DDR3 BL=8
- 2T and 3T modes to enable high-frequency operation even under heavy-load configurations
- DDR Write and Read Leveling support
- Multiple addressing options for translating master originated address into DRAM Row, Column, and BA enable efficient memory access
- Power save options for chip level and system level efficiency: Self Refresh, Pre-Charged Power Down, Active Power Down. All options with automatic entry/exit option.
- Smart memory access scheduler for maximum DRAM utilization with configurable priority scheme

The device does not support the following optional DDR3 features:

- Fixed BL4 and BC4 (Only BL8 is supported)
- Additive latency

- DLL Off mode
- TDQS
- DLL Off mode

Note

The device can be configured to either a 32-bit or 64-bit DRAM interface, with an additional 8-bit ECC option—total width 40/72-bits.

For additional information about ECC, see Section 12.9, Error Checking and Correction (ECC) and Read Modify Write, on page 188.

The DDR3 SDRAM (Double Data Rate-Synchronous DRAM) controller supports up to 4 DRAM banks (4 DRAM chip selects). It has a 19-bit address bus (M_A[15:0] and M_BA[2:0]) and a 40/72-bit data bus (M_DQ[31:0]/M_DQ[63:0], M_CB[7:0]).

The DRAM controller supports 32/40-bit or 64/72-bit interfaces. It supports the following DRAM DDR3 devices:

DDR3: 512 Mb, 1 Gb, 2 Gb, 4 Gb, and 8 Gb

The total memory space supported by the DRAM interface is up to 16 GB on 4 physical ranks. The DRAM controller supports DRAM DIMMs (both registered and unbuffered), as well as DRAM devices on board. It also supports the 2T and 3T clocking mechanism, enabling heavy load without using registered DIMM.

4 DRAM banks (4 DRAM chip selects)The DRAM can be accessed from any of the device's interfaces. The DRAM controller supports up to a 128B burst per a single transaction from any of the Mbus masters. It supports DRAM bank interleaving, as well as open pages (up to eight pages per chip select). This is typically useful for long DMA bursts to/from the DRAM.

The DRAM controller also supports ECC that allows it to detect and correct single-bit errors and detect double-bit ECC errors (SEC/DED—Single Error Correction/Double Error Detection).

12.2 Functional Description

The DRAM controller receives read and write requests from any of the device units through the device's Mbus fabric or from the Marvell® CPU via the dedicated, tightly coupled, low-latency Fast AXI path, and translates these requests into DRAM transactions.

For transactions coming from the Mbus fabric, the DRAM controller contains a transaction queue, with read and write buffers. These buffers can absorb up to 8 transactions of 128B each. Any Mbus originated new incoming transaction is going through a synchronization stage from the Mbus clock domain to the DRAM Controller clock domain before being served.

For CPU originated transactions, the DRAM controller has a dedicated queue to collect the incoming requests from the AXI path. This queue may absorb 6 additional Read/Write transactions. As the DRAM controller clock is synchronized to the CPU and the Coherency Fabric clock, no synchronization is needed and the transaction can be immediately served, reducing overall access latency.

The system can be configured to support non-synchronous clocks between the DRAM controller and the CPU and the coherency fabric. This mode should be used to reach maximum DRAM bandwidth on the expense of CPU to DRAM latency.

Once accepted, the transaction's original address is being decoded according to the addressing table mode, the used DRAM component type and the interface width into Row address, Column address and Bank address. More details about the address translation can be found in Section 12.4.1, DRAM Addressing, on page 180.

The logic will now pick the next transaction to be served to best utilize the DRAM bus on the one hand and maintain system ordering and system level transaction priority on the other hand. For

Copyright © 2014 Marvell

example, latency sensitive transactions may get higher priority and are served first if no ordering hazards apply. In the same way, the logic may also reorder the transaction execution to maximize the DRAM bus utilization through improved bank interleaving, minimization of read-write shifts and so on, resulting in much higher DRAM utilization percentage and overall lower average latency to memory. More details about the arbitration and smart scheduling scheme can be found in Section 12.4, Arbitration and Ordering, on page 179.

When ECC is used, the DRAM controller calculates ECC for a write data before forward it to the DRAM. It does the same for returned data from DRAM before returning it to the requesting master. In case the calculated ECC does not match the DRAM registered ECC, the controller reports an error to the CPU. More details about the ECC support may be found in Section 12.9, Error Checking and Correction (ECC) and Read Modify Write, on page 188.

The DRAM controller interacts with the Marvell DRAM PHY to drive the transaction over the DRAM interface. It drives a portion of the selected transaction’s address bits on M_A[15:0] and M_BA[2:0], during the activate cycle (M_RASn).The remaining bits are driven during the command cycle (M_CASn). The data path flow of both reads and writes are described in Section 12.3, Transaction Data Path, on page 175.

12.3 Transaction Data Path

12.3.1 Write Data Path

For a write transaction originating from the Mbus, write data coming from the requesting unit is placed in the write buffer. The write buffer is necessary to compensate for the data rate differences between the received write data rate (running at core clock domain) and the rate that data is driven to the DRAM interface (DRAM clock domain, double data rate).

When using a 64-bit DRAM interface, the DRAM interface write data path is 128 bits wide. Write data received from the 64-bit wide Mbus is packed to 128 bits, if the 1:1 ratio is used, or packed to 256 bits, if the 2:1 ratio is used, before being driven to DRAM. Write data received from the CPU over the AXI interface does not need to be packed. The AXI width and speed is aligned with the DRAM bandwidth. As a function of the selected clock ratio of 1:1 or 2:1 between the DRAM interface and the controller, the device’s AXI bus data rate may be programmed to match the DRAM’s data rate (of 128/256 bits of data per each DRAM cycle).

Using a 32-bit DRAM interface, the DRAM interface write data path is 64 bits wide. Write data received from the 64-bit wide Mbus is forwarded to DRAM (no need for packing). Write data from the CPU is received over the AXI bus that should be configured to 128 bits wide.As a function of the selected clock ratio of 1:1 or 2:1 between the DRAM interface and the controller, the device’s AXI bus data rate is automatically programmed to match the DRAM’s data rate (of 64 bits of data per each DRAM cycle). The DRAM controller completes the necessary unpacking of data from the AXI bus, and drives the data properly on the DRAM bus.

Note

According to the DRAM width and the clock ratio between the DRAM clock and the DRAM controller clock (either 1:1 or 2:1, respectively), set the AXI interface to be 128b or 256b as follows.

For a 64-bit DRAM interface and a clock ratio of 1:1:

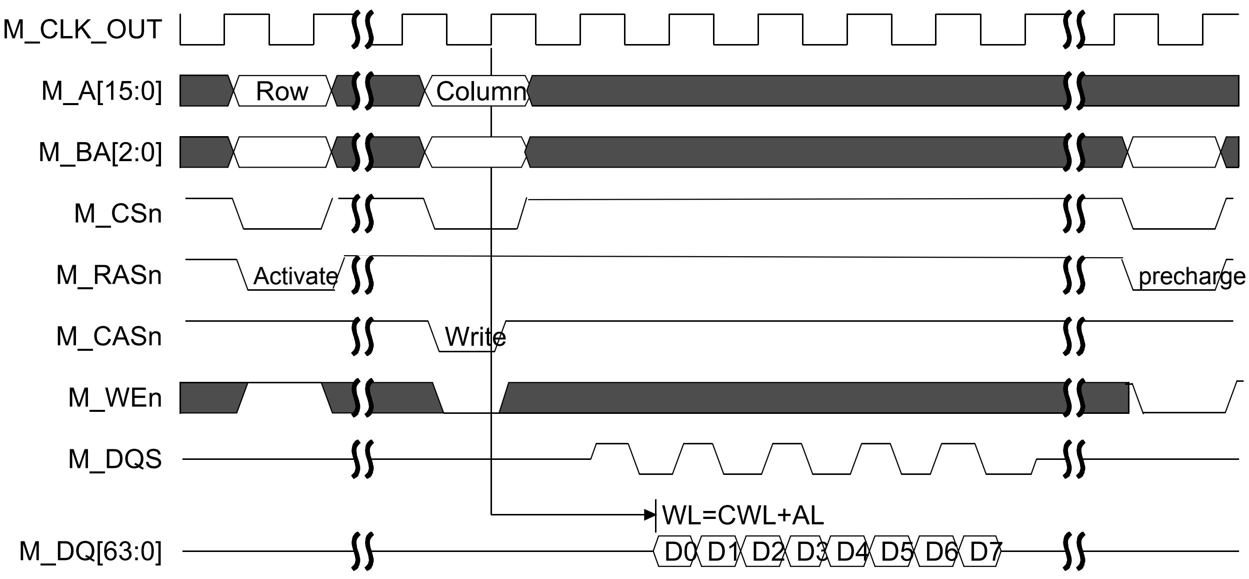
- Reset the <AxiDataBusWidth> field in the AXI Control Register (Table 473 p. 801).
- Set <DDRBusInUse> field in the SDRAM Configuration Register (Table 451 p. 774).
- Software can override the default value of <Phy2UnitClkRatio> field in the DDR IO Register (Table 482 p. 810) to force a different interface width.

For 64bits DRAM interface and a clock ratio of 2:1:

- Set the <AxiDataBusWidth> field in the AXI Control Register (Table 473 p. 801).
- Reset <DDRBusInUse>.
- Software can override the default value of <Phy2UnitClkRatio> field in the DDR IO Register (Table 482 p. 810) to force a different interface width.

An example of write transactions is shown in Figure 29. The DRAM controller access to DRAM consists of an activate cycle (row address), a command (column address), and a precharge at the end of transaction. Write data is driven with each of the clock's rising and falling edges, along with DQS. The DRAM controller also inserts the required preamble and post-amble phases.

Figure 29: DDR3 Burst Write Example (BL=8)



Signals timing and delays presented in the figures not necessarily represent the actual timing of the device.

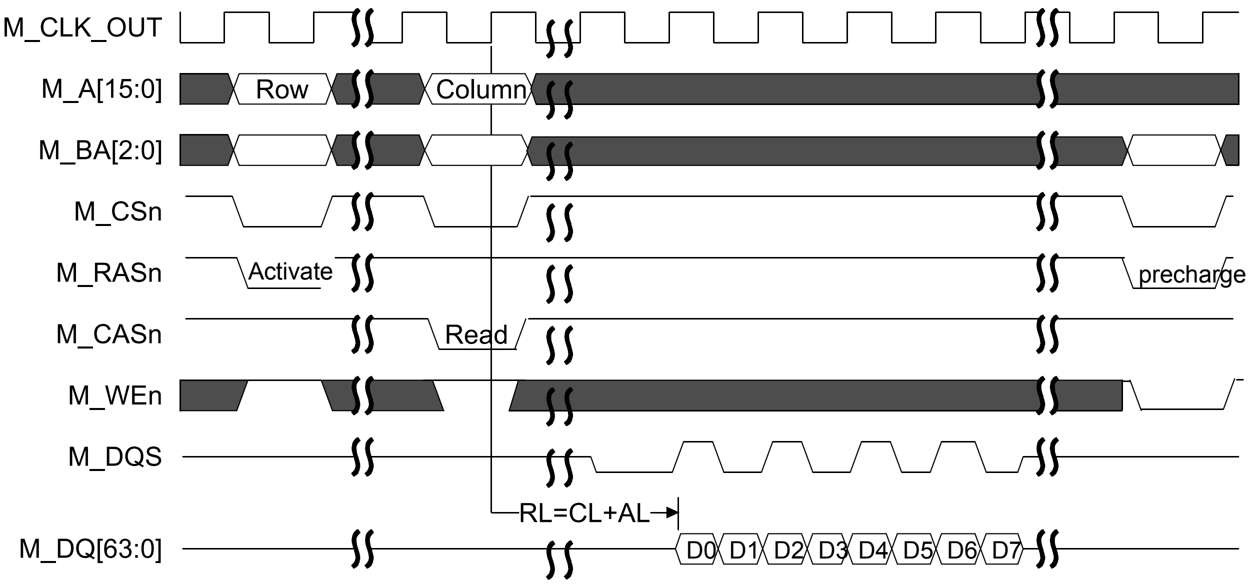
12.3.2 Read Data Path

For a read transaction, after the command cycle (M_CASn), the DRAM controller samples read data driven by the DRAM (the sample window depends on the CL parameter, the board topology and the

Copyright © 2014 Marvell

link training results). An example of a read transaction is shown in Figure 30. The DRAM controller latches the incoming data with each rising and falling of DQS input.

Figure 30: DDR3 Burst Read Example



Signals timing and delays presented in the figures not necessarily represent the actual timing of the device.

Once received, the DRAM controller pushes the data into the read buffer, and drives it back to the requesting unit.

The buffer is required to compensate for the data rate differences between the received read data from the DRAM (running at DRAM clock domain, double data rate) and the data rate of the requesting unit (running at core clock domain for Mbus masters). It is also used for temporary storage of read data, when the originator unit cannot absorb this data.

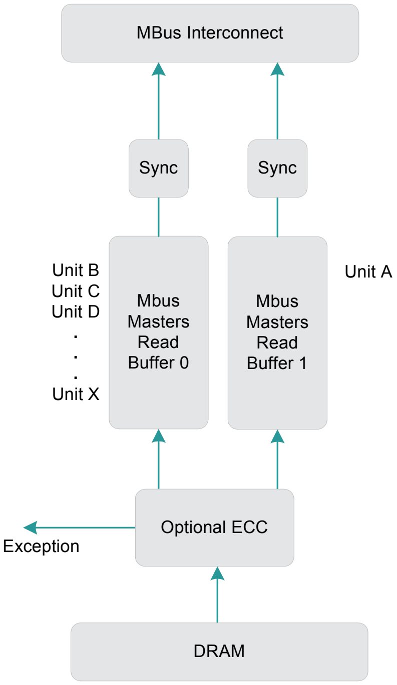
To minimize the read latency for CPU reads from the DRAM, the read data is not pushed to the read buffer. The read data is sent directly to the CPU by the Coherency Fabric, via the dedicated, low latency AXI path.

When using a 64-bit DRAM interface, the DRAM interface read data path is 128-bits wide. Read data received from the DRAM interface is unpacked before being driven over the 64-bit wide Mbus. Read data targeted to the CPU is driven over the AXI interface and does not need to be unpacked. The AXI width and speed are aligned with the DRAM bandwidth. As a function of the selected clock ratio between the DRAM interface and the controller, either a 1:1 ratio or a 2:1 ratio, respectively, the device's AXI bus data rate can be programmed to match the DRAM's data rate (of 128 bits of data per each DRAM cycle) (see Section 12.3.1, Write Data Path, for details on how to align the AXI and DRAM interfaces).

Using a 32-bit DRAM interface, the DRAM interface read data path is 64 bits wide. Read data received from the DRAM interface is forwarded to the 64-bit Mbus (no need for unpacking). Read data targeted to the CPU is driven over the AXI bus. As a function of the selected clock ratio between the DRAM interface and the controller, either a 1:1 ratio or a 2:1 ratio, respectively, the DRAM controller completes the necessary packing of data before properly driving it back to the CPU. The DRAM controller completes the necessary packing of data before driving it back over the AXI bus.

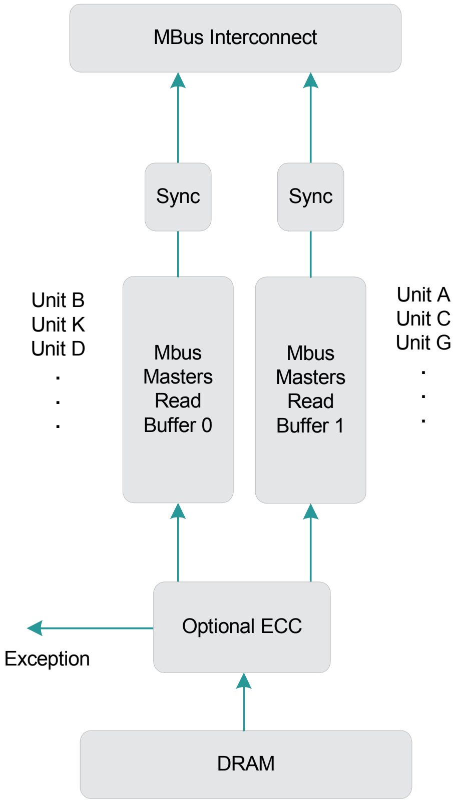
The DRAM controller has two Mbus read buffers, one per each Mbus physical port. Data can return in parallel from these two buffers over different Mbus ports and be sent to the requesting unit. Other than buffering data for rate synchronization and compensation of serving latencies, the 2 DRAM controller Mbus read buffers are also used for decoupling reads to different resources. Set the Read Buffer Select Register (Table 472 p. 799) to assign a read buffer per each of the Mbus master units. This is especially useful for resources that are latency sensitive. As all transactions in a given buffer return in the order of arrival, latency sensitive transactions can be stuck after former, but lower priority, transactions before being served on the Mbus. Setting one read buffer for a higher priority “Unit A” and the other buffer for all other units guarantees that “Unit A” read completions are not delayed after a read completion to some other, lower priority, unit. An illustration of such configuration is in Figure 31.

Figure 31: DRAM Controller Mbus Read Buffers—Latency Sensitive Setting



If there are no units that require special latency prioritization, it is recommended to split the units between the 2 buffers in a balanced way that will balance the amount of data returned on each Mbus port according to the master bandwidth capabilities in the given application. An illustration of such configuration is in Figure 32.

Figure 32: DRAM Controller Mbus Read Buffers—Even Bandwidth Allocation



12.4 Arbitration and Ordering

Transactions coming from the device's Mbus fabric are pushed into a transaction queue. The DRAM controller arbitrates between the transaction at the top of the queue and transactions coming from the CPU AXI path.

The DRAM controller can re-order transactions over the DRAM interface to optimize the DRAM utilization, while still maintaining system level ordering (Read after Write, Write after Write hazards, and so on).

When receiving a CPU-to-DRAM transaction over the AXI path, the DRAM controller performs an address lookup, against pending Mbus write transactions to DRAM. If an address match occurs, the CPU transaction is postponed until the matched Mbus transaction is forwarded to the DRAM. This lookup mechanism guarantees proper producer-consumer operation.

Note

The DRAM controller guarantees not to issue the CPU transaction before the preceding Mbus transaction, if a lookup hit occurs. However, there is no guarantee that the CPU transaction will be the next one served after the Mbus transaction.

12.4.1 DRAM Addressing

Table 27: DDR3 DRAM Addressing

DRAM Type		Bank Address	Row Address	Column Address	Auto Precharge	BC Switch On the Fly
512Mb	64Mx8	M_BA[2:0]	M_A[12:0]	M_A[9:0]	M_A[10]	M_A[12]
	32Mx16	M_BA[2:0]	M_A[11:0]	M_A[9:0]	M_A[10]	M_A[12]
1 Gb	128Mx8	M_BA[2:0]	M_A[13:0]	M_A[9:0]	M_A[10]	M_A[12]
	64Mx16	M_BA[2:0]	M_A[12:0]	M_A[9:0]	M_A[10]	M_A[12]
2 Gb	256Mx8	M_BA[2:0]	M_A[14:0]	M_A[9:0]	M_A[10]	M_A[12]
	128Mx16	M_BA[2:0]	M_A[13:0]	M_A[9:0]	M_A[10]	M_A[12]
4 Gb	512Mx8	M_BA[2:0]	M_A[15:0]	M_A[9:0]	M_A[10]	M_A[12]
	256Mx16	M_BA[2:0]	M_A[14:0]	M_A[9:0]	M_A[10]	M_A[12]
8 Gb	512Mx16	M_BA[2:0]	M_A[15:0]	M_A[9:0]	M_A[10]	M_A[12]

The DRAM controller supports up to 4 DRAM physical banks, DRAM chip selects. The total DRAM bank address space is determined by the nature of the DRAM devices. For example, while using 512 Mb x 8 devices (64Mx8), the physical bank, built up of 8 such devices (for 64-bit DRAM interface), has 512 MB of address space.

The maximum overall supported DRAM space is up to 16 GB, while populating four DRAM physical banks.

12.4.2 DRAM Address Multiplexing

The <CSxAddrSel> fields in the SDRAM Address Control Register (Table 455 p. 779) define how the address bits driven by the requesting master to the DRAM controller are translated to row and column address bits on M_A[15:0] and M_BA[2:0]. Setting the <CSxAddrSel> field to 0, provides page interleaving between different banks for sequential accesses. A new page is accessed only after the same page was accessed in all available banks. Setting a <CSxAddrSel> field to 1 provides bank interleaving for sequential accesses. A new bank is accessed only after all pages from the current bank were accessed.

The row and column address translation is different for different DRAM interface widths, DRAM densities, DRAM organization x8/x16, and DDR3 DRAM types.

Table 28 through Table 31 list the multiplexing combinations for DDR3 based on the <CSxAddrSel> setting.

Table 28: DDR3 Address Multiplex for 64-bit Interface, <CSxAddrSel> = 0

DRAM Configuration		M_BA[2:0]	Row M_A[15:0]	Column M_A[15:0]
512 Mb	64Mx8	15:13	28:16	12:3
	32Mx16	15:13	27:16	12:3
1 Gb	128Mx8	15:13	29:16	12:3
	64Mx16	15:13	28:16	12:3
2 Gb	256Mx8	15:13	30:16	12:3
	128Mx16	15:13	29:16	12:3
4 Gb	512Mx8	15:13	31:16	12:3
	256Mx16	15:13	30:16	12:3
8 Gb	512Mx16	15:13	31:16	12:3

Table 29: DDR3 Address Multiplex for 64-bit Interface, <CSxAddrSel> = 1

DRAM Configuration		M_BA[2:0]	Row M_A[15:0]	Column M_A[15:0]
512 Mb	64Mx8	18:16	28:19, 15:13	12:3
	32Mx16	18:16	27:19, 15:13	12:3
1 Gb	128Mx8	18:16	29:19, 15:13	12:3
	64Mx16	18:16	28:19, 15:13	12:3
2 Gb	256Mx8	18:16	30:19, 15:13	12:3
	128Mx16	18:16	29:19, 15:13	12:3
4 Gb	512Mx8	18:16	31:19, 15:13	12:3
	256Mx16	18:16	30:19, 15:13	12:3
8 Gb	512Mx16	18:16	31:19, 15:13	12:3

Table 30: DDR3 Address Multiplex for 32-bit Interface, <CSxAddrSel> = 0

DRAM Configuration		M_BA[2:0]	Row M_A[14:0]	Column M_A[14:0]
512 Mb	64Mx8	14:12	27:15	11:2
	32Mx16	14:12	26:15	11:2
1 Gb	128Mx8	14:12	28:15	11:2
	64Mx16	14:12	27:15	11:2

Copyright © 2014 Marvell

Table 30: DDR3 Address Multiplex for 32-bit Interface, <CSxAddrSel> = 0 (Continued)

DRAM Configuration		M_BA[2:0]	Row M_A[14:0]	Column M_A[14:0]
2 Gb	256Mx8	14:12	29:15	11:2
	128Mx16	14:12	28:15	11:2
4 Gb	512Mx8	14:12	30:15	11:2
	256Mx16	14:12	29:15	11:2
8 Gb	512Mx16	14:12	30:15	11:2

Table 31: DDR3 Address Multiplex for 32-bit Interface, <CSxAddrSel> = 1

DRAM Configuration		M_BA[2:0]	Row M_A[14:0]	Column M_A[14:0]
512 Mb	64Mx8	18:16	27:19, 15:12	11:2
	32Mx16	18:16	26:19, 15:12	11:2
1 Gb	128Mx8	18:16	28:19, 15:12	11:2
	64Mx16	18:16	27:19, 15:12	11:2
2 Gb	256Mx8	18:16	29:19, 15:12	11:2
	128Mx16	18:16	28:19, 15:12	11:2
4 Gb	512Mx8	18:16	30:19, 15:12	11:2
	256Mx16	18:16	29:19, 15:12	11:2
8 Gb	512Mx16	18:16	30:19, 15:12	11:2

Table 32: DDR3 Address Multiplex for 64-bit Interface, <CSxAddrSel> = 0

DRAM Configuration		M_BA[2:0]	Row M_A[15:0]	Column M_A[15:0]
512 Mb	64Mx8	15:13	28:16	12:3
	32Mx16	15:13	27:16	12:3
1 Gb	128Mx8	15:13	29:16	12:3
	64Mx16	15:13	28:16	12:3
2 Gb	256Mx8	15:13	30:16	12:3
	128Mx16	15:13	29:16	12:3
4 Gb	512Mx8	15:13	31:16	12:3
	256Mx16	15:13	30:16	12:3

Table 32: DDR3 Address Multiplex for 64-bit Interface, <CSxAddrSel> = 0 (Continued)

DRAM Configuration		M_BA[2:0]	Row M_A[15:0]	Column M_A[15:0]
8 Gb	512Mx16	15:13	31:16	12:3

Table 33: DDR3 Address Multiplex for 64-bit Interface, <CSxAddrSel> = 1

DRAM Configuration		M_BA[2:0]	Row M_A[15:0]	Column M_A[15:0]
512 Mb	64Mx8	18:16	28:19, 15:13	12:3
	32Mx16	18:16	27:19, 15:13	12:3
1 Gb	128Mx8	18:16	29:19, 15:13	12:3
	64Mx16	18:16	28:19, 15:13	12:3
2 Gb	256Mx8	18:16	30:19, 15:13	12:3
	128Mx16	18:16	29:19, 15:13	12:3
4 Gb	512Mx8	18:16	31:19, 15:13	12:3
	256Mx16	18:16	30:19, 15:13	12:3
8 Gb	512Mx16	18:16	31:19, 15:13	12:3

Table 34: DDR3 Address Multiplex for 64-bit Interface, <CSxAddrSel> = 0

DRAM Configuration		M_BA[2:0]	Row M_A[15:0]	Column M_A[15:0]
512 Mb	64Mx8	15:13	28:16	12:3
	32Mx16	15:13	27:16	12:3
1 Gb	128Mx8	15:13	29:16	12:3
	64Mx16	15:13	28:16	12:3
2 Gb	256Mx8	15:13	30:16	12:3
	128Mx16	15:13	29:16	12:3
4 Gb	512Mx8	15:13	31:16	12:3
	256Mx16	15:13	30:16	12:3
8 Gb	512Mx16	15:13	31:16	12:3

Table 35: DDR3 Address Multiplex for 64-bit Interface, <CSxAddrSel> = 1

DRAM Configuration		M_BA[2:0]	Row M_A[15:0]	Column M_A[15:0]
512 Mb	64Mx8	18:16	28:19, 15:13	12:3
	32Mx16	18:16	27:19, 15:13	12:3
1 Gb	128Mx8	18:16	29:19, 15:13	12:3
	64Mx16	18:16	28:19, 15:13	12:3
2 Gb	256Mx8	18:16	30:19, 15:13	12:3
	128Mx16	18:16	29:19, 15:13	12:3
4 Gb	512Mx8	18:16	31:19, 15:13	12:3
	256Mx16	18:16	30:19, 15:13	12:3
8 Gb	512Mx16	18:16	31:19, 15:13	12:3

By default, all four physical banks (M_CS[3:0]) are populated with the same DRAM device density and configuration. However, the device can also support a different DRAM configuration for each physical bank, thereby enabling population of each physical DRAM bank with a different DRAM configuration. For example, M_CS[0] bank can be populated with a 256-MB DRAM consisting of four 64-MB devices (64Mx8), while M_CS[1] is populated with a 512-MB DRAM consisting of four 128-MB devices (128Mx8). This option is especially useful for systems supporting memory expansion.

12.5 DRAM Timing Parameters

The DRAM controller supports a wide range of DRAM timing parameters. These parameters can be configured through the:

- SDRAM Timing (Low) Register (Table 453 p. 776)
- SDRAM Timing (High) Register (Table 454 p. 777)
- SDRAM Address Control Register (Table 455 p. 779)
- DDR3 Timing Register (Table 460 p. 786)
- Read Data Sample Delays Register (Table 484 p. 812)
- Read Data Ready Delays Register (Table 485 p. 813)
- DDR3 MR0 Register (Table 492 p. 819)
- DDR3 MR2 Register (Table 494 p. 822)
- DDR3 Registered DRAM Timing (Table 505 p. 834)

Note

The DRAM controller does not support different timing parameters per each physical bank.

Table 36 lists the supported DRAM timing parameters.

Table 36: DRAM Timing Parameters

DRAM Timing Parameters	Description
CAS Latency (CL)	The number of cycles from a READ command until the DRAM drives the first read data. The DRAM controller supports a CL of 6 to 14 cycles.
RAS Precharge (Trp)	The minimum number of cycles from precharge to a new activate cycle, to the same DRAM bank.
M_RASn to M_CASn (Trcd)	The minimum number of cycles between activate cycle to command cycle, to the same DRAM bank.
Row Active Time (Tras)	The minimum number of cycles between activate cycle to precharge cycle, to the same DRAM bank.
Write to Precharge (Twr)	The minimum number of cycles between write command and precharge, to the same DRAM bank.
Write to Read (Twtr)	The minimum number of cycles between write command and read command to the same DRAM device.
Active to Active (Trrd)	The minimum number of cycles between activate bank A to activate bank B to the same DRAM device.
Refresh Command (Trfc)	The minimum number of cycles between refresh command and new activate or refresh command.
Average Refresh Rate (Trefi)	The number of cycles of the periodic refresh interval.
Read Sample Delay (RdSmplDel)	The number of cycles from M_CASn assertion for a READ command until the sampling of the first read data. May be equal to or larger than CL, as determined by the Read Leveling training procedure.
Read Ready Delay (RdReadyDel)	The number of cycles from M_CASn assertion for a READ command until all data is ready to be popped from the PHY read FIFO. This allows the DRAM controller to align data from different data octets.
Read to Read (Tr2r)	The minimum number of cycles between consecutive read commands to different devices. It is not part of the JEDEC specification. It is used for preventing contention between consecutive reads to different DRAM devices (different chip selects). The DRAM controller supports a Tr2r of 1–32 cycles.
Read to Write and Write to Read (Tr2w_w2r)	The minimum number of cycles between read command to write command. It is not part of the JEDEC specification. It is used for preventing contention between consecutive Read-after-Write or Write-after-Read commands. The DRAM controller supports a Tr2w_w2r of 0–31 cycles.
Write and Write to Two Different Ranks (Tw2w)	The minimum number of cycles between two consecutive write commands to different DRAM ranks. It is not part of the JEDEC specification. The DRAM controller supports a Tw2w of 0–31 cycles.
CAS Write Latency (Tcwl)	The number of cycles from a WRITE command until the DRAM controller drives the first write data. The DRAM controller supports a CWL of 5 to 10 cycles.

Table 36: DRAM Timing Parameters (Continued)

DRAM Timing Parameters	Description
Read to Precharge Delay (Trtp)	The number of cycles between a Read command and a following Precharge command.
Initialization Calibration Time on Power Up and Reset (Tzqinit)	The number of cycles needed to complete the full calibration process upon power up or reset. Hard wired to 512 cycles.
Normal Operation Full Calibration Sequence Time (Tzqoper)	The number of cycles needed to complete the full calibration process during normal DRAM operation time. Hard wired to 256 cycles.
Normal Operation Short Calibration Sequence Time (Tzqcs)	The number of cycles needed to complete the partial fast-calibration process during normal DRAM operation time. Hard wired to 64 cycles
Exit Active Power Down Time (Txp)	The number of cycles to wait after exiting the Active Power Down mode before issuing a new command. The DRAM controller always uses the value of Txpdll as Txp.
Exit Precharge Power Down Time (Txpdll, Txard, Txards)	The number of cycles to wait after exiting the Precharge Power Down mode before issuing a new command.
Exit Self Refresh Time (Txsdll)	The number of cycles after exiting self refresh. **NOTE:** The DRAM controller uses for DDR3: Txsdll = Tdllk = 512 cycles.
Power Down Entry to Exit Time (Tpd)	The maximum number of cycles from Power-down entry to Power-down exit. The resolution of the DRAM controller implemented counter is of Trefi.
Mode Register to Non Mode Register Time (Tmod)	Minimum number of cycles from an Mode Register Set (MRS) command to a non-MRS command.
Write Leveling DQS to DQ Time (Twlo+Twloe)	Minimum number of cycles before the controller can sample the DQ after DQS assertion, during the write leveling sequence.

12.6 DRAM Burst

A DRAM can be configured to different Burst Lengths (BL) and burst ordering. The device's DRAM controller supports the following burst length:

- BL setting of 8

A single DRAM access request by one of the device's masters can vary from a 1B up to a 128B burst. The DRAM controller drives the DRAM address and control signals accordingly, concatenating multiple BL accesses as one long burst.

Even when the required DRAM access is not a full multiple of the DRAM BL, the DRAM controller always completes the burst to the next BL boundary. For a write transaction, the controller masks the redundant beats using a data mask (M_DM).

Since the DRAM controller effectively accesses 128 bits on each cycle (in 64-bit mode), it always accesses the DRAM to a 128-bit aligned address, and uses an M_DM to mask non-desired writes.

Using a 32-bit DRAM interface, the DRAM accesses are always done to a 64-bit aligned address. The DRAM controller uses the M_DM bus to mask non-desired writes.

All accesses to the DRAM require straight linear ordering. If the required access crosses the BL boundary, the DRAM controller drives a new column address (asserting a new CAS) when crossing the BL, and prevents the DRAM from wrapping around the address.

Copyright © 2014 Marvell

A CPU cache line read (32B) is always aligned to a 32B boundary When running with a burst length of 4 , the DRAM controller always issues a read from DRAM offset 0x0, and drives data back to the CPU interface in the order of arrival.

When running in DDR3 BL8 mode with a 64-bits DRAM interface width, the DRAM controller issues a 64B read either from DRAM offset 0x0 or from DRAM offset 0x4, according to the cache line address. The requested 32B is driven back immediately to the CPU interface.

Burst Interject Support

The Marvell® maximum read transaction size is 32B (cache line) while an Mbus request can be as long as 128B. This design offers a fair arbitration scheme between the CPU and the Mbus. Since CPU read access is latency-sensitive, waiting for an entire 128B transaction to complete before serving a CPU read request can result in a CPU performance penalty.

This performance penalty is especially relevant for a 32-bit DRAM interface, in which a 128B access takes 16 DRAM clock cycles to complete.

To resolve this conflict, the device's DRAM controller supports a burst interject feature.

With this feature, the DRAM controller splits every Mbus transaction on 32B or 64B boundaries. (A 128B transaction is split into four 32B transactions or into two 64B transactions) If the DRAM controller receives a CPU request while in the middle of serving an Mbus transaction, it forwards the CPU transaction in between the 32/64B segments of the Mbus transaction.

This feature is selectable via the <CPU_Interjection> field in the DDR Controller Control (High) Register (Table 458 p. 785). When burst interject is enabled:

If interfacing DDR3 32-bit DRAM, the DRAM controller splits every Mbus transaction on 32B boundaries. (A 128B transaction is split into four 32B transactions).

In DDR3-64-bit mode, the DRAM controller splits every Mbus transaction on 64B boundaries. (A 128B transaction is split into two 64B transactions.)

Note

If there is no CPU transaction interference, the burst interject feature does not introduce any performance penalty on Mbus transactions.

12.7 DRAM Bank Interleaving

The device supports both physical bank (M_CS[3:0]) interleaving and virtual bank (M_BA[2:0] for DDR3 or BG[1:0], BA[1:0] for DDR4) interleaving.

Interleaving provides higher system performance by hiding a new transaction's activation cycles within a previous transaction's data cycles. This technique gains maximum utilization of the DRAM bus bandwidth.

The DRAM controller has a pipeline to perform bank interleaving between the current active transaction and the next two transactions, if they are targeted to different banks (either virtual or physical).

Proper selection of DRAM address multiplexing, via the DRAM Address Control register, can sometimes increase the probability of virtual bank interleaving. See the DRAM Address Multiplexing (Table 12.4.2 p. 180) for further details about the address multiplexing options.

12.8 DRAM Open Pages

It is possible to configure the device's DRAM controller to keep DRAM pages open, via the SDRAM Open Pages Control Register (Table 456 p. 781). It supports up to 32 open pages (one page per each virtual bank) for DDR3 and 64 open pages (one page per each virtual bank) for DDR4.

When a page is kept open at the end of a burst (no precharge cycle), and if the next cycle to the same virtual bank hits the same page (same row address), there is no need for a new activate cycle. This is typically useful for large DMA transfers to/from the DRAM.

Once a page is open, it is kept open until any one of the following events occur:

- There is an access to the same bank but to a different row address. The DRAM controller performs a precharge (closes the page) and opens a new one (the new row address).
- The refresh counter expires. The DRAM controller closes all open pages and performs a refresh to all banks.
- Before entering Self Refresh mode
- Before entering Precharge Power Down mode

12.9 Error Checking and Correction (ECC) and Read Modify Write

The device supports Error Checking and Correction (ECC) for both 16b and 32b DRAM.

ECC is enabled via the <ECC> field in the SDRAM Configuration Register (Table 451 p. 773). ECC checking and generation requires additional 8-bit data (M_CB pins) for ECC code, resulting in a 72-bit wide DRAM interface (or 40-bit in case of 32-bit mode).

During reads, the DRAM controller can identify and correct single-bit errors or detect (but not correct) double-bit ECC errors (SEC/DED).

Where the DRAM controller detected a double ECC error, a maskable interrupt is reported to the local CPU, and the error indication is logged in the SDRAM Error Data (High) Register (Table 439 p. 766) or SDRAM Error Data (Low) Register (Table 440 p. 766).

In addition to the report to the local CPU, the device provides the capability of driving the double ECC error indication to an external entity via the M_DECC_ERR pin (multiplexed over MPP pin). In case a double ECC error was detected, and the external indication was enabled via the <ECC> field in the SDRAM Configuration Register (Table 451 p. 773), the DRAM controller will assert the ECC error bit as long as the error indication was not cleared from the SDRAM Error Data (High) Register/SDRAM Error Data (Low) Register**.**

During writes, the DRAM controller calculates the ECC based on the write data, and drives it on M_CB[7:0] lines with the write data driven on the M_DQ pins. Partial access to DRAM is defined by the device as an access smaller than 64-bits. To generate the ECC on partial writes (writes that are less than 64-bits), the DRAM controller must perform a Read-Modify-Write (RMW) access as follows:

1. Reads the existing 64-bit data from the DRAM, or 32-bits for 32-bit DRAM. In the latter case, creates a 64-bit data vector by padding the higher 32 bits with zeros, as if 64 bits were read from the DRAM (Bus_for_ECC_calculation [63:0] = {0x00000000,M_DQ[31:0]}).
2. Reads the existing 32-bit data from the DRAM and creates a 64-bit data vector by padding the higher 32 bits with zeros, as if 64 bits were read from the DRAM (Bus_for_ECC_calculation [63:0] = {0x00000000,M_DQ[31:0]}).
3. Calculates ECC on the data, and compares it to the ECC being read.
4. If no ECC error is found, merges the new incoming data with the data read from the DRAM.
5. Calculates new ECC byte based on the data that is to be written.
6. If a single-bit error was detected during the read, fixes the corrupted bit, before merging the data with the new incoming data.

If double-bit ECC errors are detected (non correctable) during the read, corrupts the new ECC byte after the merge (maintains the 2-bit error in the DRAM).

7. Writes the new data and new ECC byte back to the DRAM bank. During this write, all M_DM lines are de-asserted (write of a full 72-bits, or 40-bits for 32-bit DRAM).
8. Writes the new data and new ECC byte back to the DRAM bank. During this write, all M_DM lines are de-asserted (write of a full 40-bits).

When the DRAM controller is accessed with a write request, while all byte enables are not active, it still performs RMW. It reads the data out of the DRAM, calculates ECC, corrects any single-bit ECC errors, and writes the data back to DRAM (no data to merge with, since all byte enables are not active). This behavior is useful for “cleaning” single-bit ECC errors in DRAM. For more information, see Section 31.3.5, Memory ECC Errors Cleanup (Scrubbing), on page 496.

In case of a burst write to the DRAM with a start or end address that is not 8B aligned, the device executes a RMW access of the whole burst, even though only the first and/or last data requires RMW.

When using DRAM ECC protection, it is necessary to first initialize the DRAM. However, if using partial writes (writes smaller than 64-bit) for this initialization procedure, while the DRAM controller performs RMW, the controller is likely to detect non-correctable errors, and preserve the errors. Setting the <IErr> field in the SDRAM Configuration Register (Table 451 p. 773) to 1 disables ECC checking. If ECC is enabled while <IErr> is set to 1, the DRAM controller performs RMW on partial writes, ignores ECC errors that are found on the read data, and generates proper ECC per the merged data. When the <IErr> bit is set to 1, no ECC errors are reported.

Note

<IErr> is useful for DRAM initialization or for debug. It must be cleared to 0 for proper DRAM ECC operation.

12.10 DRAM Refresh

The device implements a counter to maintain the DRAM refresh rate requirements.

The refresh rate for all banks is determined according to the <Refresh> field in the SDRAM Configuration Register (Table 451 p. 774). For example, the default value of Refresh is 0x400. If the M_CLK_OUT frequency is 400 MHz (2.5 ns cycle), a refresh sequence occurs every 2.56 us.

Every time the refresh counter reaches its terminal count, a refresh request is sent to the DRAM Controller. The refresh request has the highest priority over any other DRAM access request. As soon as the current outstanding DRAM transactions complete, the DRAM controller precharges all banks (both the ones that are opened, and the ones that are not), and performs an auto-refresh command to all DRAM banks.

12.11 DRAM Initialization

The DRAM controller starts the DRAM initialization sequence as soon as the <InitEn> field in the SDRAM Initialization Control Register (Table 467 p. 792) is set to 1. The software must initialize the DRAM Control registers prior to setting the <InitEn> bit.

After the first initialization, it is possible to change the DRAM mode registers or to repeat the full initialization sequence. To change the DRAM mode and DRAM extended mode values after initialization has been completed, see Section 12.12, DRAM Operation Mode Register, on page 191.

The DRAM controller postpones any attempt to access the DRAM before the initialization sequence completes. It is recommended that the Marvell® software poll the <InitEn> bit.

After setting to 1 to start the initialization process, reading a value of 0 means that the DRAM interface is ready for normal operation.

Note

After being set, the <InitEn> field in the SDRAM Initialization Control Register (Table 467 p. 792) is only cleared once both the reset and initialization sequence and the interface training sequence have completed.

The initialization sequence DDR3 modes are described below:

12.11.1 DDR3 Initialization Sequence

The DDR3 DRAM specification requires at least 200 us of M_RESET maintained low after DRAM power up, before starting the initialization, as well as M_CKE driven low at for least 10 ns, before de-asserting M_RESET. The DRAM controller starts driving M_RESET and M_CKE to low once the device is powered up. To maintain the DRAM requirement make sure the that the initialization sequence does not start earlier than 200 us after the device's power rails are stable.

The DDR3 DRAM initialization sequence consists of the following steps:

1. M_RESET de-assertion.
2. After 500 us, de-assert M_CKE.
3. Wait tXPR.
4. Issue the MRS(2) command based on the DRAM MR2 register value.
5. Issue the MRS(3) command based on the Extended DRAM MR3 register value.
6. Issue the MRS(1) command based on the DDR3 MR 1 register value, to enable the DRAM DLL and configure the ODT values.
7. Issue the MRS(0) command based on the DDR3 MR 0 register value, and with the reset DLL (bit[8]) activated.
8. Issue the ZQCL command to start the ZQ calibration phase, for calibrating Rtt and Ron values for PVT.
9. For high frequencies, further tuning steps may be required. Refer to the DRAM Operation Mode Register (Table 12.12 p. 191) for more details.

Note

For the registered DIMM initialization process, see Section 12.14, DIMM Support, on page 194.

The user must not attempt to change any of the following register settings after the <InitEn> field in the SDRAM Initialization Control Register (Table 467 p. 792) has been set, and before this field is auto-cleared by the DRAM controller.

Note

Upon writing to these registers, the actual written value will be duplicated to DDR3 MR0 CSn Register (n=0–3) through DDR3 MR3 CSn Register (n=0–3).

- DDR3 MR0 Register (Table 492 p. 819)
- DDR3 MR1 Register (Table 493 p. 820)

- DDR3 MR2 Register (Table 494 p. 822)
- DDR3 MR3 Register (Table 495 p. 823)

To guarantee this restriction, it is recommended that the Marvell® software poll the <InitEn> field in the SDRAM Initialization Control Register (Table 467 p. 792) until this bit is cleared, indicating that the initialization process has completed.

12.12 DRAM Operation Mode Register

In addition to the normal DRAM operation mode, the DRAM controller also supports special DRAM commands through the <Cmd> field in the SDRAM Operation Register (Table 457 p. 784). These operations include:

- Normal DRAM mode (default mode)
- Precharge all banks
- Force a Refresh Cycle on all banks
- DRAM Mode Register Setting (DDR3 MR0)
- DDR3 Mode Register 1 (MR1) access
- DDR3 MR2 access
- DDR3 MR3 access
- NOP commands
- Enter Self Refresh mode
- Enter Active Power Down mode
- Enter Precharged Power Down mode
- Force long calibration command (ZQCL)
- Force short calibration command (ZQCS)
- Control Word Access (CWA) command—DDR3 registered DRAM only

The register contains 4 bits of command type. Once the Marvell® CPU changes the register default to one of the command types, and after executing the required command, the DRAM controller resets the register back to the default value, and returns to normal operation. The Marvell® CPU must poll on this register to identify when the DRAM controller is back in normal operation mode.

When using DIMMs, the DRAM parameters are recorded in the DIMM Serial Presence Detect (SPD) serial ROM. The Marvell® can read the SPD via the device's I^{2}C interface and program the DRAM parameters accordingly, using the Load Mode register command.

12.13 Power Save Options

The device includes several options to reduce power consumption of the DRAM interface:

- DRAM Power Down mode:
  - Active Power Down
  - Precharged Power Down
- DRAM Self Refresh mode

The following sections describe ways of taking advantage of these modes and saving system power.

12.13.1 DRAM Power Down

The device's DRAM controller supports two methods for placing the DRAM into Power Down mode:

- **Precharged power down:** Enter power down while all pages are closed.
  Going into Precharged Power Down mode allows lower power consumption at the expense of a longer exit time.
- **Active power down:** Enter power down while one or more pages are left open (active).
  Going into Active Power Down mode provides faster exit time from the Power Down state at the expense of higher power consumption than Precharged Power Down mode.

The Marvell® places the DRAM into either one of these power down states, by setting the appropriate value in the SDRAM Operation Register (Table 457 p. 782).

- Setting this register to a value of 0x11 triggers a precharged power down entry sequence on the DRAM interface. The DRAM controller first closes all open pages by executing a precharge all command on the DRAM bus, and then issues a power down command on the DRAM bus. Software can poll this register to determine when the DRAM was placed into Power Down mode.
- Setting this register to a value of 0x10 triggers an active power down entry sequence on the DRAM interface. When using this method of power down, there is no need to close all open pages before entering Power Down mode. The DRAM controller issues a power down command on the DRAM bus, even if some pages remain open. The software can poll this register to determine when the DRAM was placed into Power Down mode.

During Power Down mode, all of the DRAM signals (excluding M_CLK_OUT, M_CLK_OUTn, M_CKE, M_RESET, and M_ODT) are floated configurable via the <SRFloatEn> field in the DDR3 Registered DRAM Control (Table 504 p. 833). This significantly reduces the power consumption of both the DRAM and the device.

Note

There are 2 CKE signals (M_CKE[1:0]) that behave the same. The device does not support the separate placement of each physical bank into Power Down state. If the board topology does not require the use of all of these signals, it is possible to only use 1 of the signals.

The maximum number of cycles to stay in Power Down state is configurable through <tPD> field in the DDR3 Timing Register (Table 460 p. 787) that sets the power down counter. This counter granularity is in refresh periods. The counter counts down every time a refresh period has expired. A value of 0x8 sets 8 refresh periods, the maximum allowed number of periods to maintain the DRAM contents.

The DRAM controller exits Power Down mode only as a result of one of these events:

- The power down counter expires
- An incoming transaction from one of the device's masters to DRAM
- An NOP command setting in the SDRAM Operation Register.

Note Any attempt to access the DRAM with one of the operation commands other than the NOP command (for example an MRS command), while in Power Down mode, may result in a system hang.

Upon exiting the Power Down mode, the DRAM controller issues the appropriate number of refresh commands, required to compensate for the lack of refresh commands during the Power Down state, before proceeding with any other activity.

12.13.2 DRAM Self Refresh Mode

Note If the counter expires and there is no pending transaction in the DRAM controller buffers, the DRAM controller exits the Power Down state, issues the appropriate number of refresh commands to compensate for the lack of refresh cycles during the Power Down state, and then re-enters the Power Down state automatically.

The DRAM controller also supports a DRAM Self Refresh mode. This feature is mainly useful for:

- Power saving (and for the frequency change procedure)
- Battery backup (in case of power failure)

The DRAM controller puts the DRAM in Self Refresh mode by generating a refresh cycle with M_CKE driven low. When exiting the Self Refresh mode, the controller drives M_CKE high, and tDLLK cycles (in DDR3 mode), before generating any new transaction to DRAM.

The DRAM controller supports the power saving or battery backup applications differently. For battery backup, the <SRMode> field in the SDRAM Configuration Register (Table 451 p. 773) is cleared to 0. Once the DRAM has entered the Self Refresh mode, it can no longer be accessed, and only returns to normal operation after power on reset. For power saving, set the <SRMode> field to 1, and normal operation resumes after any new DRAM access request.

Note There are 4 CKE signals (M_CKE[3:0]) that behave the same. The device does not support the separate placement of each physical bank into Power Down state. If the board topology does not require the use of all of these signals, it is possible to only use some of the signals.

During self refresh, all of the DRAM signals (excluding M_CLK_OUT, M_RESET, and M_CKE) are floated. This significantly reduces power consumption. If the <SRClk> field in the DDR Controller Control (Low) Register (Table 452 p. 776) is set to 1, the device also stops driving M_CLK_OUT and M_CLK_OUTn when the DRAM is in Self Refresh mode.

The device's DRAM controller places the DRAM into Self Refresh mode when the <Cmd> field in the SDRAM Operation Register (Table 457 p. 784) is set to 0x7. Once the field is set, the DRAM controller waits for 256 cycles, generates a Self Refresh command to DRAM, and clears the DRAM Operation register.

The device's DRAM controller puts the DRAM into Self Refresh mode in one of the following situations:

- The <Cmd> field in the SDRAM Operation Register (Table 457 p. 784) was set to 0x7. Once the field is set, the DRAM controller waits for 256 cycles and generates a Self Refresh command to DRAM, and clears the DRAM Operation register.
- During DRAM frequency change (see Section 34.5, Dynamic Frequency Scaling (DFS), on page 530).
- If there are new pending transactions to DRAM, or a NOP command was registered in the SDRAM Operation Register, the DRAM controller sets M_CKE[3:0] to 1 to take the DRAM out of the Self Refresh mode. The controller then waits tDLLK cycles in DDR3 before generating the transaction to DRAM.

For DDR3, the DRAM controller provides the option to automatically attach a ZQCS calibration command to the self refresh exit flow. Calibration commands are periodically needed to compensate for voltage and temperature variations. The DRAM controller may interleave a ZQCS calibration command during the time the DLL is being locked, while exiting Self Refresh mode. To enable this automatic calibration command upon exiting Self Refresh, set bit <SR Exit ZQCS> field in the ZQC Configuration Register (Table 497 p. 825) to 1. For full details about the calibration process and considerations refer to Section 12.20, DRAM Calibration, on page 201.

Note

The DRAM controller exits Self Refresh mode, only as a result of a DRAM read/write access or the setting of a NOP command in the SDRAM Operation Register. Any attempt to access the DRAM with one of the operation commands other than the NOP command (for example an MRS command), while in Self Refresh mode, may result in a system hang.

12.14 DIMM Support

The JEDEC standard defines many types DDR3 DRAM DIMMs:

- Registered or unbuffered DIMMs
- 32-bit, 64-bit or 40/72-bit wide (for ECC)
- 1, 2, or 4 physical banks (chip selects)
- Different bank organization— x8 devices, x16 devices
- Mirrored DIMMs for DDR3
- Different densities

The device's DRAM controller supports all of these DIMMs.

Note

When working with DRAM DIMM, note the following configuration parameters:

- **Do not** use unbuffered DIMM and registered DIMM in the same system.
- **Do not** use ECC with non-ECC DIMM in the same system.
- The DIMMs can be produced by different vendors, as long as the combined timing configuration of the different vendors is available.
- Each DIMM can be configured with a different capacity and organization. For example, one DIMM can be 2 GB, x8 devices, and the other DIMM can be 1 GB, x16 devices.

12.15 DRAM Topologies

The device provides several tools to enable high-speed, heavy-load DRAM topologies (multiple DRAM device topologies), as described in the following sub-sections.

- Registered Address / Command Bus
- 2T and 3T Modes
- DDR3 Address Mirroring

12.15.1 Registered Address / Command Bus

When using multiple physical banks, the address and control signals are heavily loaded, and may be unable to meet the DRAM AC timing requirements.

Using registered DIMMs solves this issue by buffering the address/control signals.

When interfacing registered DRAM DIMMs or on-board registration logic, all address and control signals (M_A[15:0], M_BA[2:0], M_RASn, M_CASn, M_WEn, M_CSn, and M_CKE) are registered on the DIMM/Board. This means that the signals arrive at the DRAM device one cycle after they are driven by the DRAM controller. It also means that Read data arrives back at the DRAM controller one cycle later (in comparison to non-registered topologies).

When the DRAM controller is configured to registered DRAM via the <RegDIMM> field in the SDRAM Configuration Register (Table 451 p. 774), it drives Write data one cycle later and samples the Read data one cycle later (in comparison to non-registered topologies).

Note

The device can work either with registered DIMMs or with unbuffered DIMMs.The device cannot operate with both DIMM types at the same time.

The device supports SSTE32882 devices as defined by the JEDEC standard.

SSTE32882 3T Mode

The optional 3T mode (RC2 bit DA4) is supported for all commands, under these limitations:

- DRAM controller operates in 2-to-1 mode towards the PHY. (The <Phy2UnitClkRatio> field in the DDR IO Register (Table 482 p. 810) is configured to 0x1.)
- DRAM Controller is configured to 3T mode. (The <2T> field in the DDR Controller Control (Low) Register (Table 452 p. 776)<2T> 0x1404[4:3]) is configured to 0x2.)

This mode is enabled by setting the <Register 3T mode> field in the DDR3 Registered DRAM Control (Table 504 p. 834), and issuing a control word access (CWA) to enable bit DA4 in RC2 (see Control Word Access (CWA).

Note

The DRAM Controller's 3T mode may be enabled regardless of the register's 3T mode (but not vice versa).

DDR3 Initialization

Prior to starting the regular DDR3 initialization (see Section 12.11.1, DDR3 Initialization Sequence, on page 190), it may be necessary to configure the control words of the SSTE32882. In this case, prior to setting the <InitEn> field in the SDRAM Initialization Control Register (Table 467 p. 792) as part of the initialization flow, software must perform the following steps:

1. Set either the <Init RESET Deassert> field or the <Init CKE Assert> field in the SDRAM Initialization Control Register (Table 467 p. 791).
   The DRAM controller will perform the memory initialization steps until the step that is defined by these registers and will stop there, rather than completing the full initialization flow.
2. Poll the above fields for 0x0. This indicates the DRAM controller has completed this stage.
3. Issue a control word access (CWA) command via the SDRAM Operation Register (Table 457 p. 782) (see Control Word Access (CWA)) to configure the SSTE32882.
4. Poll the <Cmd> field in the SDRAM Operation Register (Table 457 p. 784) for 0x0.
   This indicates the DRAM controller has completed the CWA (including any required delays after the CWA).
5. Repeat steps 3 and 4 as required.
6. Set the <InitEn> field in the SDRAM Initialization Control Register (Table 467 p. 792) to complete the initialization sequence.

Control Word Access (CWA)

Control Word Access (CWA) is provided through the DRAM controller's SDRAM Operation Register (Table 457 p. 782).

1. Set the <Cmd> field to 0xE.
2. Enable the appropriate group of chip selects using fields <Cmd_CS0>, <Cmd_CS1>, <Cmd_CS2>, and <Cmd_CS3>.
   NOTE: At least two of these fields must be enabled to perform CWA, per the SSTE32882 specification.
3. Select the requested Register Control via the <CWA RC> field.
4. Place the control word data in the <CWA Data>field.
5. Set the <CWA Delay Sel> field to specify whether the CWA affects the SSTE32882 clock timing and therefore should impose tSTAB delay after the access (setting to 0x1). Otherwise, tMRD delay would be used (clear to 0x0).
   NOTE: The tSTAB and tMRD delay values may be reconfigured via the DDR3 Registered DRAM Timing (Table 505 p. 834). tSTAB may have to be reconfigured if frequency band 2 is used.

Note

Issuing CWA while the DRAM is in Power Down mode would bring the DRAM out of Power Down (CKE will be asserted). Issuing CWA while the DRAM is in Self Refresh mode will not bring the DRAM out of Self Refresh (CKE is kept de-asserted).

Parity

The device does not drive the PAR_IN input of the SSTE32882, and does not support its ERROUT# output.

The PAR_IN pin must be pulled either high or low on the board.

During control word access (CWA) commands, the DRAM controller avoids corruption of the command by toggling any of the unused M_RASn, M_CASn, or M_WEn pins, so that the parity of the driven command matches the PAR_IN constant value.

1. To select the value of the corresponding pins during CWA commands, configure the <CWA RAS> field, the <CWA CAS> field, and the <CWA WE> field in the DDR3 Registered DRAM Control (Table 504 p. 834)
2. Also configure the <CWA Parity>, according to the PAR_IN constant value.

CKE Grouping

The device allows flexible connectivity of its M_CKE[3:0] outputs to the SSTE32882 DCKE[1:0] inputs, to simplify board design and layout. To ensure correct protocol behavior, particularly during Self Refresh Entry and Power Down Entry commands, software should configure the <CKE0 CS Grp> field, the <CKE1 CS Grp> field, the <CKE2 CS Grp>, and the <CKE3 CS Grp> field in the DDR3 Registered DRAM Control (Table 504 p. 833), which associate each M_CKE[3:0] pin with the CSs connected to it.

Similarly, the M_ODT[3:0] outputs may be flexibly connected to the SSTE32882 DODT[1:0] inputs. This feature is available regardless of the Registered DRAM configuration, via the registers:

- SDRAM ODT Control (Low) Register (Table 469 p. 793)
- SDRAM ODT Control (High) Register (Table 470 p. 797)

12.15.2 2T and 3T Modes

An alternative solution to registered DIMM/board logic for the heavy-load configuration on the command and address lines, is using the 2T or 3T mode. The device supports both these modes. While configured to one of these modes, all address/control signals, except for M_CS[3:0] and M_ODT[3:0], are asserted for 2/3 cycles respectively, instead of one cycle. The DRAM protocol is still maintained, since all of the signals are qualified with M_CS[3:0]. These signals are still asserted for only one cycle. This operation results in an easier timing requirement on the address/control signals.

To enable this mode, configure the <2T> field in the DDR Controller Control (Low) Register (Table 452 p. 776) to the desired delay value.

12.15.3 DDR3 Address Mirroring

The device also supports DDR3 DIMMs that implement address mirroring. DDR3 devices have the option to flip over (“mirror”) their address bus. Putting 2 such mirrored devices back to back on a dual ranked DIMM or on board results in minimum tracing on the DIMM/board. Still, some of the address pins are cross-lined between the 2 ranks.

Table 37 lists the address signals relationship between the DIMM connector, un-mirrored DRAM rank, and a mirrored DRAM rank.

Table 37: DDR3 Address Mirroring Mapping Table

DIMM Connector	Un-mirrored Rank	Mirrored Rank
BA[0]	BA[0]	BA[1]
BA[1]	BA[1]	BA[0]
BA[2]	BA[2]	BA[2]
A[0]	A[0]	A[0]
A[1]	A[1]	A[1]

Table 37: DDR3 Address Mirroring Mapping Table (Continued)

DIMM Connector	Un-mirrored Rank	Mirrored Rank
A[2]	A[2]	A[2]
A[3]	A[3]	A[4]
A[4]	A[4]	A[3]
A[5]	A[5]	A[6]
A[6]	A[6]	A[5]
A[7]	A[7]	A[8]
A[8]	A[8]	A[7]
A[9]	A[9]	A[9]
A[10]	A[10]	A[10]
A[11]	A[11]	A[11]
A[12]	A[12]	A[12]
A[13]	A[13]	A[13]
A[14]	A[14]	A[14]
A[15]	A[15]	A[15]

Regular DRAM accesses to each of the DIMM's ranks is transparent and the DRAM controller drives the address in the same way to both of the ranks. The one exception is while issuing MRS commands. In these commands, the address and bank address buses are used as data signals carrying values to write to the MRS register. While accessing MRS registers of a mirrored rank, the DRAM controller flips the data between relevant pins to make the mirror transparent to software. Software writes to the DRAM controller's MRS registers as if the rank is not mirrored, and if the rank is mirrored, the DRAM controller flips the relevant pins during DRAM access.

For example, for software to configure the DRAM MR0 register of a mirrored rank that resides on M_CS[1] to a CAS Latency value of 7 (A[4]=1) and to Nibble sequential Read burst type (A[3]=0), it must write 0x6 to the <CL> field DDR3 MR0 Register (Table 492 p. 819), and 0x0 to <RBT> field in the same register. Upon triggering the MR0 command in the SDRAM Operation Register (Table 457 p. 782), the DRAM controller samples the CS1_Mirror field in DDR3 Rank Control Register (Table 496 p. 824) to see if rank1 was defined as populated by a mirrored DIMM.

If accessing a mirrored DIMM, the DRAM controller flips the values of A[3] and A[4] to 1 and 0, respectively, before driving the signals to the DIMM connector. On the mirrored DIMM itself, A[3] from the DIMM connector is routed to A[4] on the DRAM and A[4] on the connector is routed to pin A[3] on the DRAM, resulting in sending the right values to the appropriate location.

12.16 DRAM Clocking

The Coherency Fabric, DRAM controller and the DRAM clocks are all derived from the same PLL that generates the CPU core clock (PCLK). The clocks are edge aligned, and the entire Marvell®-to-DRAM path runs synchronously, enabling very low latency.

The DRAM controller drives the DRAM M_CLK_OUT[3:0] and M_CLK_OUTn[3:0] differential pairs, satisfying the DRAM clock specifications. However, the driving strength of these 4 clock pairs is limited.

When interfacing with registered DIMMs, this limitation is not a problem. The DIMM contains a zero delay clock buffer (DLL). When interfacing with multiple DRAM devices on a board, some topologies may require a buffer for the device's clock outputs, using an on board PLL.

The device, besides the CPU and DRAM interfaces, runs at a different clock domain (TCLK). Any request for DRAM access from other masters than the CPUs over the device's Mbus passes through a synchronization logic on the way to and from the DRAM.

12.17 DRAM Address/Data Drive

The DRAM clock is driven by the device's M_CLK_OUT/M_CLK_OUTn differential pairs. All DRAM address and control signals driven by the device (single data rate signals) are coupled to the falling edge.

Note Typically, address and control signals should be driven with the rising edge of M_CLK_OUT. However, under certain board topology and DRAM load, there may be a hold time problem on these signals. In this case, use the falling edge configuration by setting <AddrCntrlToClkSkew> field in the DRAM PHY Configuration Register (Table 498 p. 825), or the 90 degree setting in the same field.

The front-end logic of the DRAM controller is responsible for correct drive of the double data rate data with the M_DQS signals, as well as unpack of the data from DRAM.

During a write transaction, 128-bit wide data is pulled out of the write buffer and driven as 64-bit DRAM on the bus. The first 64-bits are driven with the rising edge of M_CLK_OUT, and the second 64-bits are driven with falling edge of M_CLK_OUT.

The DRAM controller drives DQS (data strobe) along with the data. The DRAM specification requires very accurate DQS timing in respect to the DRAM clock. The DRAM controller uses an analog DLL (ADLL) block to achieve correct timing (shift DQ by 1/4 cycle).

For a 32-bit DRAM interface, 64-bit wide data is pulled out of the write buffer and driven as two 32-bit data bits on the DRAM bus.

12.18 DRAM Read Data Sample

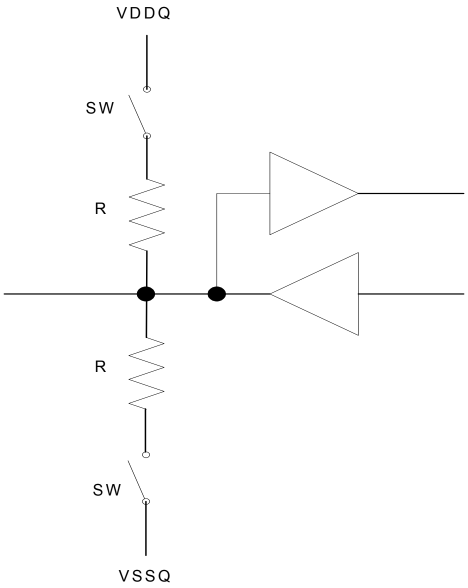
The front-end logic of the DRAM controller is responsible for correct sampling of the double data rate data with M_DQS signals, as well as the pack of data from double data rate to single data rate.

According to the DRAM interface width, DRAM read data is latched by the received DQS (shifted by approximately 1/4 cycle). The data bits that arrived on the first DRAM data beat, are sampled with the rising of DQS, and the data that arrived on the following clock beat is sampled with the falling edge of DQS.

12.19 On-Die Termination (ODT)

The DDR mode support dynamic ON and OFF termination resistors within the DRAM I/O buffers, as well the DRAM controller I/O buffers. Figure 33 shows a schematic of a DRAM I/O buffer.

Figure 33: DRAM I/O Buffer



The DDR technology offers dynamic ODT to further enhance the signal integrity on the data bus. The device fully supports ODT and dynamic ODT as described in the following sections.

The DRAM controller has 4 ODT signals (M_ODT[3:0]), shared among the 4 DRAM ranks. There is another, internal ODT signal that controls the termination inside the device’s I/O buffers. The ODT signals can dynamically turn the DRAM termination ON and OFF. This is useful for maintaining proper signal integrity with minimum reflections on the lines, without needing any external termination resistors.

The DRAM controller can also be configured to static termination configuration, rather than dynamic termination per transaction.

12.19.1 DDR3 ODT

While configured to DDR3 mode, the R nominal (RZQ) value is fixed at 240 ohm. This results in Rtt_NOM ODT values of 20, 30, 40, 60, or 120 ohm that the device fully supports both on the DRAM I/O buffers and the device’s I/O buffers.

For further improved write signaling, greater termination impedance of 60 or 120 ohm is preferred during the write operation, while the preference during idle states is typically for 30 or 40 ohm termination values. The DDR3 dynamic ODT (Rtt_WR) feature enables the DRAM to switch between HIGH or LOW termination impedance without issuing a mode register set (MRS) command.

If enabled through an MRS command, dynamic ODT causes the DRAM to switch termination from Rtt_NOM to Rtt_WR, while a write command is issued. The DRAM changes the termination value back to Rtt_NOM as soon as the write has completed. Rtt_WR may be used during write operations, even if Rtt_NOM is disabled.

Note

As defined in the JEDEC specification, ODT applies only to DM, DQ, and DQS signals.

The <Rtt_NOM>, <Rtt_NOM_6>, and <Rtt_NOM9> fields in the DDR3 MR1 Register control the termination values of Rtt_NOM during ODT operation or disable termination.

The <Rtt_WR> field in the DDR3 MR2 Register controls the termination values of Rtt_WR during dynamic ODT operation or disables dynamic termination.

Typically, when driving a signal and eliminating reflections, place a termination resistor at the end of the line. When the device drives data on the DQ lines (write transactions), it is desired to turn ON the termination on the DRAM. On the other hand, when the DRAM drives DQ signals (read transactions), it is required to turn ON the termination inside the device's I/O buffer.

In a multiple DRAM bank environment, termination topology is more complex, and requires some board simulation. The device's DRAM controller provides the full flexibility to select which of the 4 DRAM banks terminations to turn ON or OFF, per any read or write transaction to any of the 4 banks (refer to the device Design Guide).

Note

As DDR3 DRAM cannot terminate and drive at the same time, do not operate ODT during reads on the DRAM chip select that is driving the data. For example, do not set the <ODT0_READ_CS3> field in register SDRAM ODT Control (Low) Register to a value of 1, meaning M_ODT[0] is driven high while reading from M_CS[3].

12.20 DRAM Calibration

The DDR DRAM supports a sequence to calibrate the DRAM's output buffers and ODT values through the ZQCL and ZQCS commands.

ZQCL is used to perform the initial calibration phase as part of the DRAM initialization process. This command executes a thorough calibration process on the DRAM using the internal DRAM's calibration engine before setting the correct driving strength and ODT values on the DRAM I/O buffers. ZQCL may be used at any time and not only as part of the initialization sequence.

ZQCS command is used as a periodic calibration update to compensate for voltage and temperature variations on the DRAM. This command is able to perform relatively small updates to the current calibrated values of the DRAM I/O buffers, and takes less time to complete than ZQCL.

The device's DRAM controller supports both calibration commands. The DRAM controller issues a ZQCL command on the DRAM if one of the following apply:

- As part of the initialization sequence.
- The software sets <Cmd> field in the SDRAM Operation Register (Table 457 p. 784) to 0x12.
- The type of command defined for a periodic calibration sequence was set to ZQCL through field <Auto ZQC select> in ZQC Configuration Register.
- Periodically, when the <Auto ZQC Timing> counter in ZQC Configuration Register expires and the <Auto ZQC Select> field in the same register is configured to the ZQCL type of calibration. For more details about periodic calibration, see Section , Periodic Calibration.

Before issuing the ZQCL command on the bus, the DRAM controller automatically precharges all open pages, unless the calibration is done upon exiting Self Refresh mode.

The DRAM controller issues a ZQCS command on the DRAM if the following apply:

- The DRAM controller issues a ZQCS command on the DRAM if the software sets the <Cmd> field in the SDRAM Operation Register (Table 457 p. 784) to 0x13. The type of command defined for a periodic calibration sequence was set to ZQCS through field <Auto ZQC Select> in ZQC Configuration Register.
- Upon exiting Self Refresh mode, if the option was enabled through the <SR Exit ZQCS> field in ZQC Configuration Register. In this situation, the calibration process is completed in parallel to the DRAM DLL lock time.
- Periodically, when the <Auto ZQC Timing> counter in ZQC Configuration Register expires and the <Auto ZQC Select> field in the same register is configured to the ZQCS type of calibration. For more details about periodic calibration see Section , Periodic Calibration.

Before issuing the ZQCS command on the bus, the DRAM controller automatically precharges all open pages.

Periodic Calibration

It is recommended that the system allow periodic calibration updates, to maintain the correct behavior of the electrical interface and to minimize VT effects on signal integrity. The DRAM controller supports automatic periodic calibration tuning on the DRAM interface. It utilizes a programmable counter that, once it expires, issues a process of periodic calibrations. As part of this sequence, the DRAM controller precharges all open pages, and issues either a ZQCL or ZQCS command, according to the value of <Auto ZQC select> field in the ZQC Configuration Register (Table 497 p. 825). The time period between two consecutive calibration periods is counted in Trefi resolution (counts down in the refresh periods) and is configurable through the <Auto ZQC timing> field in the same register. On each expiration of the counter, the DRAM controller issues a calibration command on a populated DRAM rank (starting from the lowest number M_CSx that is populated). Once calibration completes, the counter is reloaded and starts to count down again. On the next expiration, the DRAM controller issues a calibration command on the next populated DRAM rank and so on. Once all populated DRAM ranks are calibrated, the process starts again.

13 NAND Flash Controller (NFC)

This section describes the device's NAND Flash Controller (NFC).

Note

- MV78230/78x60 supports NAND Flash Controller (NFC) with hardware ECC, as described in this section.
- The reference to 16b NAND Flash controller or 16B flag refers to the 16-bit bus width mode (configured in the <DWIDTH_C> field or the <DWIDTH_M> field in the Data Flash Control Register (NDCR) (Table 534 p. 858). These fields can be configured either to:
  - 0x0 for an 8-bit bus
  - 0x1 for a 16-bit bus

The NFC is used to interface with an external NAND data flash memory. The usage model for the NAND data flash memory is that of a Solid State Drive for the device. A flash file system (FFS) performs tasks such as wear leveling, garbage collection, and bad-block replacement for the flash media. The NFC accepts commands from the file system, controls the flash device, according to these commands, and performs error control.

The NFC provides a glue-less interface to external data flash memory (SLC or MLC). The main NFC clock is ND_CLK. ECC is calculated according to the ECC_CLK (2 x ND_CLK). These clocks are set by configuring the <NandEccClkDivRatioFull> field in the Core Divider Clock Ratio Full0 Register (Table 1477 p. 1545).

Data-flash devices use a multi-cycle addressing scheme, so the number of interface pins remains the same for all memory densities.

- NF_IO[7:0] are used for sending command and addresses to the flash device, in both 8-bit and 16-bit NAND flash.
- NF_IO[7:0] are also used for data transfer with 8-bit devices. NF_IO[15:0] are used for data transfer with 16-bit devices.

The NFC controls these interface pins for the specific command placed in its command buffer.

The NAND Flash controller registers are located in Appendix A.4, NAND Flash Registers, on page 856.

13.1 Features

Following are the major features integrated in this interface:

- Supports 4 chips selects and 8-/16-bit interface to the data-flash device.
- Supports 32/64/128/256-KB page block sizes.
- Supports 512B, 2-KB, 4-KB, and 8-KB page sizes.
- Computes Error Correction Code (ECC) and corrects single-bit errors and detects 2-bit errors per page using Hamming ECC.
- Computes ECC and corrects up to 16 errors per page (including spare, if enabled and parity bits themselves) using BCH[1] ECC.
- Supports programmable interface timing.

[1] BCH = Bose-Chaudhuri-Hocquengham.

Copyright © 2014 Marvell

- Supports enabling Interrupts to indicate page and command completion, bad blocks, bit errors, flash-ready status and command, and data-write/read requests.
- Supports boot from NAND Flash device on CS0.

Note The total storage that the NFC can access is limited by the number of chip select pins. The addressing is sufficiently flexible to enable any size NAND device, but any configuration requiring more than four chip enable signals is not directly supported by this controller.

13.1.1 NAND Single-Level Cell (SLC) and Multi-Level Cell (MLC)

The key high-level attributes for NAND devices are listed below.

Table 38: NAND Attribute Definitions

Attribute	Definition
Density	The total storage capacity of the NAND device, typically in units of Mbits or Gbits, also denoted Mb or Gb. The true usable density may vary depending on file system overhead, garbage collection policy, and bad blocks. This controller supports a wide range of densities by virtue of its support for both 4-cycle and 5-cycle addresses, and by stacking NAND devices using multiple CS.
Interface Width	Number of bits for the data bus coming out of the NAND device. “Bare NAND” is available in two widths, 8 and 16 bit. This controller supports native widths.
Interface Speed	This is specified differently depending on the nature of the specific interface. For bare NAND devices, the interface is asynchronous and the interface speed is the read/write pulse cycle time, which is specified in nanoseconds. This controller supports varying interface speeds by enabling programmability into the timing generation.
NAND Cell Type	Single Level Cell (SLC) or Multi Level Cell (MLC).
ECC	Error Correction Coding. This indicates the recommended number of bits that should be corrected by the ECC solution on the controller. Typically the controller would require the capability to detect more errors than the minimum ECC requirement. Typically ECC requirements are specified in the number of bits per 512B section, or alternatively as the number of bits per page and this requirement can vary from vendor to vendor. The ECC algorithm can be simple for single bit correction (e.g. Hamming code) or complex for multiple bit correction (with manufacturers suggesting Reed-Solomon or Bose-Chaudhuri-Hochquenghem).
Page Size	This is the number of bytes in a single chunk of the array that can be read or written with a single command. The page size can be considered to be just the data area (and therefore be a power of two in size), or can be the data area plus the “spare” area (therefore, cannot be a power of 2). This controller supports a range of page sizes.
“Spare” Area	This is additional bits at the end of the page typically used for overhead functions. ECC syndrome bits may be stored here or file system “meta data” may be here. The spare area is considered out-of-band storage. Generally, there are 16B of spare per 512B of page size. MLC NAND increases the needed overhead dedicated to the spare area because of the potentially extreme ECC levels needed.

Copyright © 2014 Marvell

Table 38: NAND Attribute Definitions (Continued)

Attribute	Definition
Garbage Collection	NAND is written in pages (for example, 2048B) and pages are combined into erase blocks (for example, 64 pages in a block). New data can obsolete already written data where the old data has not yet been erased. As the NAND fills up, the old obsolete data needs to be garbage collected to make room. This typically involves moving the remaining “good” data in a block being garbage collected to an erased block to make all blocks in an old erase block obsolete. The old block is then erased (that is, garbage collected). The newly written block contains only valid and current data.
Endurance	**NOTE:** This is the number erase/program cycles that can reliably be supported by the device over the complete lifetime of the device. Currently, for SLC devices this is ~100,000 and for MLC devices this is ~10,000.

13.2 NAND Flash Interface

The NFC supports both 8- and 16-bit-wide data buses. Up to 4 chip selects (NF_CSn[3:0]) interface to the flash devices. Up to five cycles of addresses can be sent on the NF_IO bus to address flash devices interfaced, using these chip selects. The chip select to activate can be specified in the command for the NFC.

The NAND I/O Pins can be shared with the Device Bus I/O pins. This can be controlled through the MPP configuration (see Appendix A.19, Multi-Purpose Ports (MPP) Registers, on page 1432) that is controlled by the software, or using an enhanced arbiter. By default, the Hardware Arbiter mode is enabled in the device.

13.3 NAND Flash Connectivity

To select which CS is used, set the <ADDR5> field in the NAND Controller Command Buffer 2 (NDCB2) Register (Table 545 p. 887) and the <CSEL> field in the NAND Controller Command Buffer 0 (NDCB0) Register (Table 543 p. 883) as listed in Table 39.

To activate CS2 and CS3, write 1 to <NfCsExpansionEn> field in the SoC Device Multiplex Register (Table 1433 p. 1513).

Table 39: Chip Select Assignment

Selected CS	NDCB2 <ADDR5> bit[7]	NDCB0 <CSEL> bit[24]
CS0	0	0
CS1	0	1
CS2	1	0
CS3	1	1

A selection within the pair is made by decoding address bit[39]. This additional interface ability is intended to support stacked NAND to increase the memory density.

Copyright © 2014 Marvell

Different types of configuration are illustrated in the following figures.

Figure 34: Stacked Data Flash Memory System Example Using NF_CSn[0] and NF_CSn[1]

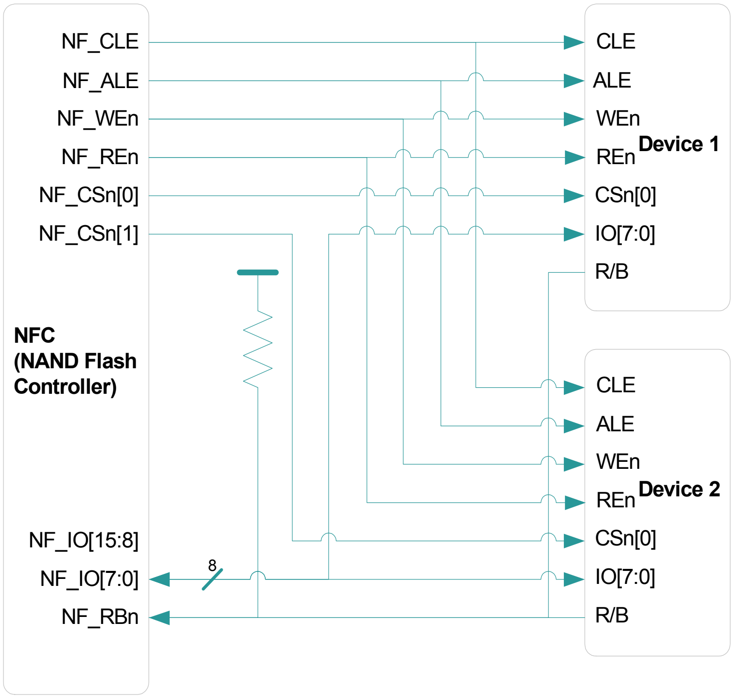
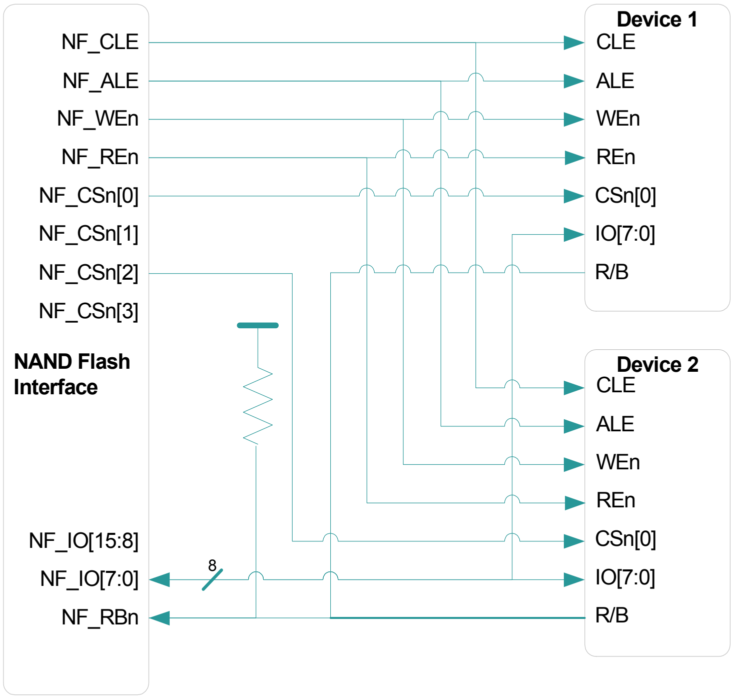


Figure 35 shows dual chip select using NF_CSn[0] and NF_CSn[2].

Figure 35: Stacked Data Flash Memory System Example Using NF_CSn[0] and NF_CSn[2]



13.4 Operation

NAND flash devices accept a variety of commands from the NFC to perform functions such as program, erase, and different types of reads. This section provides information for configuring the NFC to perform successful program, erase, and reads.

- Error checking and correction (ECC)
- Bad-block management support

The <CMD_XTYPE> field in the NAND Controller Command Buffer 0 (NDCB0) Register (Table 543 p. 881) field defines the type of command. The double-byte command <DBC> field in the register indicates that the present command is a 2B command, where CMD1 and CMD2 are the commands sent to the flash device in the CMD1-ADDR-CMD2 (or CMD1-ADDR-DATA-CMD2 in case of program commands) sequence. If the current command is a 1B command, only CMD1 and address are sent, and the sequence would be CMD1-ADDR (or CMD1-ADDR-DATA in case of program commands). The addressing (ADDR phase) is performed in multiple cycles, using pins NF_IO[7:0]. ADDR1 through ADDR5 are the addresses sent in address cycles 1 through 5, respectively. The CMD_CTRL field contains information about the number of address cycles, single/double CMD and other information, as illustrated in Table 40. One command descriptor always corresponds to a single- or double-byte NAND flash command.

Table 40: Command Format

Command Format								
Byte	7	6	5	4	3	2	1	0
	ADDR4	ADDR3	ADDR2	ADDR1	CMD_CTRL		CMD2	CMD1
	ADDR7	ADDR6	NDLENCNT		Status Mask	Status Cmd.	Page Count	ADDR5

Having <CMD1> field and <CMD2> field in the NAND Controller Command Buffer 0 (NDCB0) Register (Table 543 p. 885) in the command gives users the flexibility to choose the appropriate commands because the commands can vary widely between flash vendors and for different product families for the same vendor. This format also supports future additions to the command set as long as the command, address, and data sequence remains the same. The type of command is defined using the <CMD_XTYPE> field in the NAND Controller Command Buffer 0 (NDCB0) Register (Table 543 p. 881). The chip select to be asserted for the NAND flash access is specified by CSEL bit. The <AUTO_RS> specifies whether an automatic status check must be performed by the NFC after the command completion.

The following interrupts should be enabled by clearing the appropriate mask bits in the Data Flash Control Register (NDCR) (Table 534 p. 857).

- Write-data request interrupt
- Read-data request interrupt
- Write-command request interrupt

In this mode, the <ND_RUN> field in the Data Flash Control Register (NDCR) (Table 534 p. 858) must be set after configuring the data-flash registers, and software responds to a write-command request interrupt by writing the command to the command buffer (NDCBx). NDCBx should be written only after receiving a write-command request. If the command is a write (single page or multi page), the write-data request interrupt is activated, and software writes data corresponding to a page to the data controller data buffer (NDDB). For multi-page writes, the write-data request interrupt gets activated multiple times, requesting software attention for a page of data each time. Similarly for read operations, the read-data request interrupt is activated after reading a page of data from the flash device, and the device responds to this interrupt by emptying the data buffer.

For read ID and read status commands, software reads 8B from NDDB, because the NFC allocates one buffer entry (8B) to hold the read data for these commands. Valid data is aligned to the LSB; users should discard non-valid bytes, for example, for a 5B read-ID command, bytes 0 through 4 are valid while bytes 5 through 7 should be ignored.

Note

- Even if interrupts are not enabled, software can poll bits in the NAND Controller Status Register (NDSR) (Table 537 p. 871) corresponding to the above-mentioned interrupts to perform write/read data and write commands.
- Make sure to clear the WRDREQ/RDDREQ status bits by writing to NDSR before accessing the NDDB. This will prevent missing WRDREQ/RDDREQ interrupts for subsequent pages during multi-page PROGRAM and multi-page READ cycles.
- If you are issuing a command with the next-command (NC) bit set, the next command should not be read ID or read status because of difficulties with servicing interrupts. Read ID and read status commands are too short for the interrupt service routine to finish servicing the first command completed interrupt in time to acknowledge the second command completed interrupt, and therefore, the second interrupt will go undetected.

13.4.1 Error Checking and Correction (ECC)

Error-detection code/error-correction code (EDC/ECC) is required in the NFC to detect and correct errors occurring in the flash device due to bit flipping (Bit flipping occurs when a bit is either reversed, or is reported reversed). ECC is computed when write data is transferred from data buffer to the NAND flash. After completing the write, the computed ECC bytes are written to the spare area of the flash. Table 41 shows the available spare area and spare-area bytes used for storing the ECC for different page sizes. When ECC is enabled, the number of spare area bytes required for use by ECC are shown in the middle column of Table 41. These bytes are not available for use by the system software and cannot be written to or read from.

In addition, the number of bytes read from and written to the NAND that pass through the data FIFO must be a multiple of eight.

Note

Enable ECC only for Program page or Read page operations. Therefore, for Read-ID and Read-status operations disable ECC mode.

To interleave Read-ID with other operations, the control and the command registers have to be reconfigured for the Read-ID, and therefore, ECC setting can be disabled.

For example, if the page size is 512B with ECC and the spare area enabled, bytes 0 through 511 are written with data and the first 8B of the spare area (bytes 512 through 519) are written with spare data (file-system-dependent information). Next, 6B of spare area (bytes 520 through 525) are used by the NFC to write the ECC data. The remaining 2B cannot be written and any data read must be ignored.

When the read operation is completed, ECC bytes are read from the spare area, and any bit errors are computed and corrected. Depending on the ECC mode, the NFC uses Hamming code to correct 1-bit random error in a page and detects 2-bit errors or a BCH algorithm to correct multiple bit errors. Interrupts can be enabled to get information about single-bit and 2-bit errors.

When the 16-bit NFC flag is set to ON, there is an interaction between the ganging configuration and the ECC option available:

- Ganging of 512B page size devices is only supported with the Hamming ECC engine
- Ganging of 2048B (or 4096B) page size devices is only supported with the BCH ECC engine.

Table 41: Spare Area Used for ECC

NAND Flash Page Size (Bytes)	Typical Spare Area in NAND Flash (Bytes)	Number of Spare Area Bytes Used for Hamming ECC	Number of Spare Area Bytes Used for BCH ECC	Number of Inaccessible Bytes
512	16	0	N/A	0
512	16	6	N/A	2
2048	64	0	0	0
2048	64	24	0	0
2048	64	0	30	2
4096	128	48		0
4096	128	0	60	4

13.4.2 Hamming Code for ECC

During a page write, data is split into even and odd streams before computing the ECC. This data split lessens the error-correcting requirement on the ECC algorithm to 1 bit. If D_{ij} represents the j^{th} bit of the i^{th} byte, Table 42 and Table 43 show the data streams used for ECC computation for an 8-bit wide data bus. When the 16-bit NFC flag is set to ON, for NAND flash device that are 16-bits wide, the lower and upper bytes are treated as successive bytes of read data. However, where two 8-bit devices are interfaced through a single chip select, the ECC computations are performed independently on the upper and lower bytes of the data bus. Figure 36 illustrates the computation of partial parities (indicated in Figure 36 by Px for odd numbered bits and Px' for even numbered bits) for a 256B data stream. ECC is computed separately for each data stream and is written to the spare area of the flash device. The ECC algorithm can correct 1 bit of random error in a chunk of 256B, de-interleaved from 512B. Each of the odd and even data streams are 256B long, and the computed ECC for this data stream occupies 3B. Therefore, 6B of ECC data are written for a NAND flash with page size of 512B. These bytes are stored into the spare byte location in the NAND flash memory, as indicated in Table 44, using the same notation for partial parity bits as in Figure 36. The same ECC computation process is employed four times for a page size of 2048B, resulting in 24B of ECC. Hamming ECC is not supported in the 4-KB logical page size. Therefore, for a 4-KB logical page use BCH ECC.

Table 42: Even Data Stream

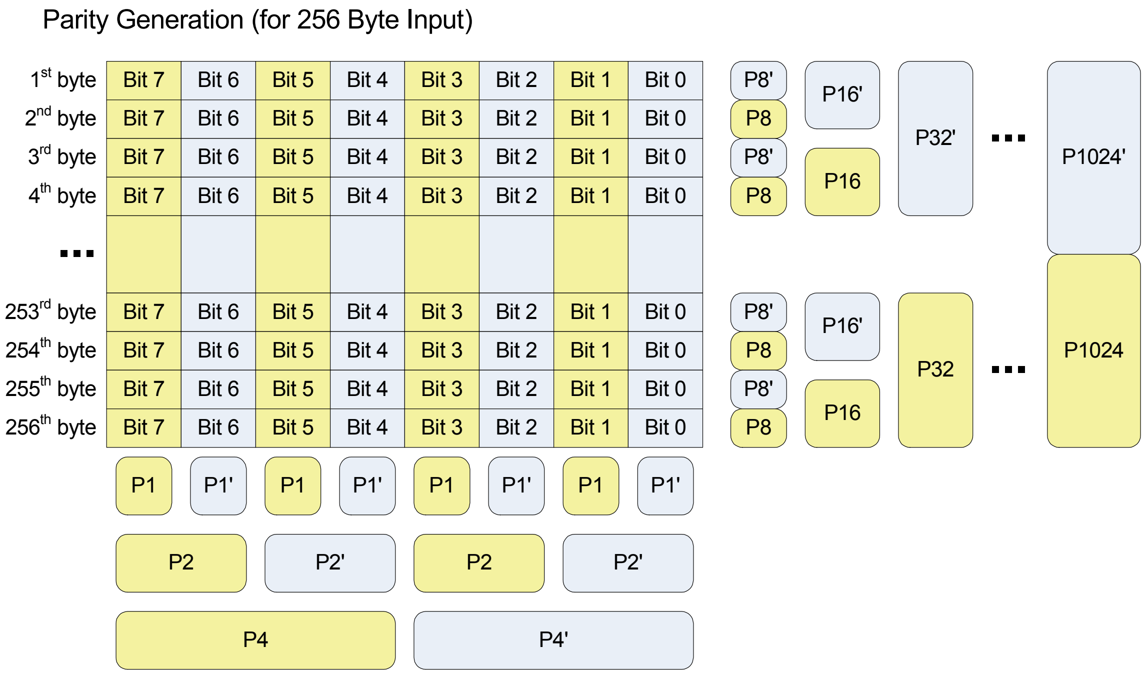
Bit	7	6	5	4	3	2	1	0
	$D_{i+1\ 6}$	$D_{i+1\ 4}$	$D_{i+1\ 2}$	$D_{i+1\ 0}$	$D_{i\ 6}$	$D_{i\ 4}$	$D_{i\ 2}$	$D_{i\ 0}$

Table 43: Odd Data Stream

Bit	7	6	5	4	3	2	1	0
	$D_{i+1\ 7}$	$D_{i+1\ 5}$	$D_{i+1\ 3}$	$D_{i+1\ 1}$	$D_{i\ 7}$	$D_{i\ 5}$	$D_{i\ 3}$	$D_{i\ 1}$

Copyright © 2014 Marvell

Figure 36: ECC Code Generation



In Table 44 *p* refers to the partial parity bit for the first 256B and *P* is the partial parity for the second 256B. The "first" and "second" are not consecutive by address, but interleaved by bit:

- The "first" 256 being composed of the even bits.
- The "second" 256 being composed of the odd bits.

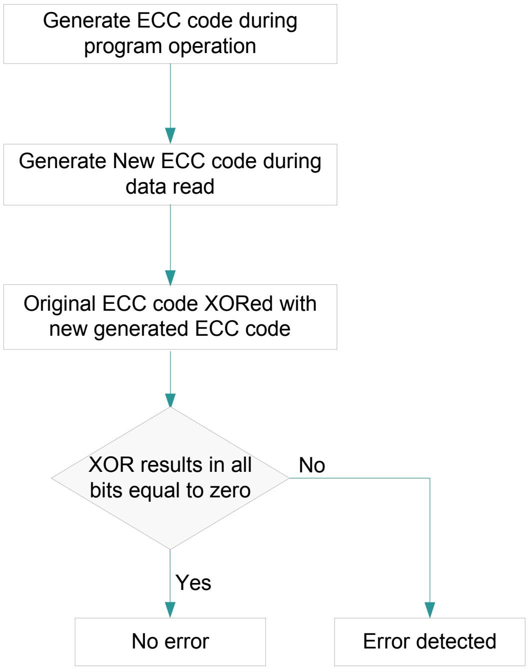
Table 44: ECC Byte Placement

Bit	7	6	5	4	3	2	1	0
Byte 0	P8'	p8'	p4'	p4'	p2'	p2'	p1'	p1'
Byte 1	p128'	p128'	p64'	p64'	p32'	p32'	p16'	p16'
Byte 2	P1	P1	p1024'	p1024'	p512'	p512'	p256'	p256'
Byte 3	P16	P16	P8	P8	P4	P4	P2	P2
Byte 4	P256	P256	P128	P128	P64	P64	P32	P32
Byte 5	0	0	0	0	P1024	P1024	P512	P512

During a page read, the read data is again split into odd and even streams. The associated ECC data is read and compared with the computed ECC from the received data stream. If a 1-bit error is detected in a data stream, it is corrected. For two errors in any or both of the data streams, the ECC algorithm detects this scenario and flags a 2-bit error condition.

The ECC logic is initialized at the end of every 512B (256B per data stream) of transfer.

Figure 37: Hamming Error Detection Process


Generate ECC code during program operation
Generate New ECC code during data read
Original ECC code XORed with new generated ECC code
XOR results in all bits equal to zero
No
Yes
No error
Error detected

13.4.3 BCH Error Correction

BCH is an acronym for Bose, Ray-Chaudhuri, Hocquengham, three individuals who collectively developed the mathematical background for multilevel, cyclic, error-correcting, variable-length digital codes. There are three parameters which define BCH codes: symbol size, field length and correction power.

For this purpose, the symbol is, of course, a binary digit. While this may seem obvious it is somewhat unfortunate because the NAND Flash interface is byte parallel whereas the BCH algorithm is inherently bit serial.

The mathematics of BCH is based on operations on Galois fields where the size of the field is (2^m -1) symbols (bits). Since BCH error correction is not intended for small block NAND flash devices, the field is chosen to cover all 2048B in a large block page, plus any spare bytes that are defined. There are 16,384 bits in 2048 bytes, but when you add in the ECC bits and the spare bits (which are also protected), the next larger power of two (minus one) is required that results in 32,767 bits. Therefore, the BCH parameter for the field size “m” is 15. The inherent capability of a BCH algorithm means that the ECC corrects any combination of n errors in 2^m -1 bits.

The NAND flash manufacturers, for reasons dating back to small block NAND flash, specified the ECC requirements for MLC as the number-of-bits-of-correction per 512B sector. The statistics for the bit error rate (BER)[1] for each NAND flash cell for MLC is uniform and the division of those statistics into 512B sectors is arbitrary. For BCH, the number of correctable bits is the parameter ‘t’.

[1] The term BER is defined as the probability of error in any one bit.

For example, if the requirement is for 8 bits of correction per 512B sector, the implication is that there is a requirement of 32-bits per 2-KB page. However, there is no way to prevent 9 of those bits from being in the first sector, or 10, or 11, or any combination up to the maximum of 32, it is solely a function of the probabilities and possible combinations.

After a massive combinational analysis where all possibilities of 32 bits of errors distributed over four 512B sectors, it was discovered that fewer total bits of correction, when designed to cover the entire page, give better results than dividing the page into arbitrary chunks. It turns out that 16-bits of correction per page gives better coding gain (that is, an increase in effective BER) than four 8-bit, 512B sector correction.

The trade-off is that the amount of hardware is approximately proportional to (m*t) so increasing t and m simultaneously (more bits of correction over a larger field) significantly increases the number of gates. However, the BCH algorithm is highly sensitive to the particular values of m and t, and all of the polynomials used are chosen specifically for each choice. This means that a generic or programmable BCH engine is not a practical alternative so a choice has to be made as to how far into the future one wishes to be able to intercept MLC BER rates.

13.4.4 Page Allocation

There are three potential types of data in a page, the main data, the so-called spare data, and the ECC. In addition, in certain common cases, some bytes are simply inaccessible. The layout of this data depends on the programming of <PAGE_SZ>, <SPARE_EN>, and <ECC_EN> in the Data Flash Control Register (NDCR) (Table 534 p. 857), and <BCH_EN> in the NAND ECC Control (NDECCCTRL) Register (Table 540 p. 878). The following table shows how bytes are allocated in the page, according to the various page sizes. Because of the 64-bit data buffer, the size of any area must be evenly divisible by 8B, so certain combinations result in unusable spare bytes. Unusable bytes must be programmed to zero in data blocks.

For small block devices, there can be a single Hamming 6B ECC, and its location depends on the spare usage. For small block devices, there is a maximum of 16B of spare area. Since the BCH ECC produces a 30B syndrome, BCH cannot be used for small block NAND flash.

Table 45: Small Block Page Allocation

PAGE_SZ == 0b00 (512 bytes)	SPARE_EN == FALSE	ECC_EN == FALSE	BCH_EN == don't care
0	511	512	527
512B Data		Inaccessible	
PAGE_SZ == 0b00 (512 bytes)	**SPARE_EN == TRUE**	**ECC_EN == FALSE**	**BCH_EN == don't care**
0	511	512	527
512B Data		16B Spare	
PAGE_SZ == 0b00 (512 bytes)	**SPARE_EN == FALSE**	**ECC_EN == TRUE**	**BCH_EN == FALSE**
0	511	512 517	518 527
512B Data		Hamming ECC	Inaccessible

Table 45: Small Block Page Allocation (Continued)

<table>
<tr><th colspan="2">PAGE_SZ == 0b00 (512 bytes)</th><th colspan="1">SPARE_EN == TRUE</th><th colspan="4">ECC_EN == TRUE</th><th colspan="3">BCH_EN == FALSE</th></tr>
<tr><td>0</td><td></td><td>511</td><td>512</td><td></td><td>519</td><td>520</td><td></td><td>525</td><td>526,
527</td></tr>
<tr><td colspan="3">512B Data</td><td colspan="3">8B Unprotected Spare</td><td colspan="3">Hamming ECC</td><td>Inaccessible</td></tr>
</table>

For large block devices, there can be four concatenated 6B Hamming ECC syndromes or a single 30B BCH syndrome. The location of the ECC depends on the spare usage.

Table 46: Large Block Page Allocation

<table>
<tr><th colspan="2">PAGE_SZ == 0b01 (2048B)</th><th>SPARE_EN == FALSE</th><th colspan="3">ECC_EN == FALSE</th><th colspan="3">BCH_EN == don't care</th></tr>
<tr><td>0</td><td></td><td>2047</td><td>2048</td><td></td><td></td><td></td><td></td><td>2111</td></tr>
<tr><td colspan="3">2048B Data</td><td colspan="6">Inaccessible</td></tr>
<tr><th colspan="2">PAGE_SZ == 0b01 (2048B)</th><th>SPARE_EN == TRUE</th><th colspan="3">ECC_EN == FALSE</th><th colspan="3">BCH_EN == don't care</th></tr>
<tr><td>0</td><td></td><td>2047</td><td>2048</td><td></td><td></td><td></td><td></td><td>2111</td></tr>
<tr><td colspan="3">2048B Data</td><td colspan="6">64B Spare Data</td></tr>
<tr><th colspan="2">PAGE_SZ == 0b01 (2048B)</th><th>SPARE_EN == FALSE</th><th colspan="3">ECC_EN == TRUE</th><th colspan="3">BCH_EN == FALSE</th></tr>
<tr><td>0</td><td></td><td>2047</td><td>2048</td><td></td><td>2071</td><td>2072</td><td></td><td>2111</td></tr>
<tr><td colspan="3">2048B Data</td><td colspan="3">(4) Hamming ECC</td><td colspan="3">Inaccessible</td></tr>
<tr><th colspan="2">PAGE_SZ == 0b01 (2048B)</th><th>SPARE_EN == TRUE</th><th colspan="3">ECC_EN == TRUE</th><th colspan="3">BCH_EN == FALSE</th></tr>
<tr><td>0</td><td></td><td>2047</td><td>2048</td><td></td><td>2087</td><td>2088</td><td></td><td>2111</td></tr>
<tr><td colspan="3">2048B Data</td><td colspan="3">40B Spare Data</td><td colspan="3">(4) Hamming ECC</td></tr>
<tr><th colspan="2">PAGE_SZ == 0b01 (2048B)</th><th>SPARE_EN == FALSE</th><th colspan="3">ECC_EN == TRUE</th><th colspan="3">BCH_EN == TRUE</th></tr>
<tr><td>0</td><td></td><td>2047</td><td>2048</td><td></td><td>2077</td><td>2078</td><td></td><td>2111</td></tr>
<tr><td colspan="3">2048B Data</td><td colspan="3">BCH ECC</td><td colspan="3">Inaccessible</td></tr>
</table>

Copyright © 2014 Marvell

Table 46: Large Block Page Allocation (Continued)

PAGE_SZ == 0b01 (2048B)	SPARE_EN == TRUE		ECC_EN == TRUE			BCH_EN == TRUE			
0		2047	2048		2079	2080		2109	2110, 2111
2048B Data			32B Spare Data			BCH ECC			Inaccessible

It is expected that all NAND Flash devices require ECC, so all examples include some type of ECC. The total number of bytes in a 4-KB page is unspecified as the size of the spare area has not been standardized.

Note

For 4-KB pages of type SLC or MLC with 4-bit BCH, always use 2-KB chunks packed together in some manner. The same applies to 8-KB chunks.

For the BCH ECC examples, the BCH ECC must be padded with two zero bytes to maintain the 8B alignment.

If 4-KB pages scale, there would be 4096B of main area and a spare area of 128B. It is expected that there will be more than 128B of spare area (due to MLC requirements), but these additional bytes will not be accessible by the NFC.

Table 47: 4-KB Physical or Logical Page Allocation Using 2-KB Chunks

PAGE_SZ == 0b01 (2048B)			SPARE_EN == FALSE				ECC_EN == TRUE			BCH_EN == FALSE		
0		2047	2048		2071	2072		4119	4120		4143	4144...
2KB Data			(4) 6B Hamming ECC			2KB Data			(4) 6B Hamming ECC			Inaccessible

PAGE_SZ == 0b01 (2048B)			SPARE_EN == FALSE				ECC_EN == TRUE			BCH_EN == TRUE		
0		2047	2048		2077	2078		4125	4126		4155	4156...
2KB Data			30B BCH ECC			2KB Data			30B BCH ECC			Inaccessible

PAGE_SZ == 0b01 (2048B)			SPARE_EN == TRUE			ECC_EN == TRUE		BCH_EN == FALSE		
0		2047	2048, 2087	2088, 2111	2112		4159	4160, 4199	4200, 4223	(4224...)
2KB Data			40B Spare	(4) 6B Hamming ECC	2KB Data			40B Spare	(4) 6B Hamming ECC	Inaccessible

Table 47: 4-KB Physical or Logical Page Allocation Using 2-KB Chunks (Continued)

PAGE_SZ == 0b01 (2048B)			SPARE_EN == TRUE			ECC_EN == TRUE		BCH_EN == TRUE		
0		2047	2048 - 2079	2080 - 2109	2110		4157	4158 - 4189	4190 - 4219	(4220...)
2KB Data			32B Spare	30B BCH ECC	2KB Data			32B Spare	30B BCH ECC	Inaccessible

The drawback in this case is that essentially the entire currently defined spare area would be consumed with ECC bytes, and for each 2-KB page, there would be four “wasted” bytes.

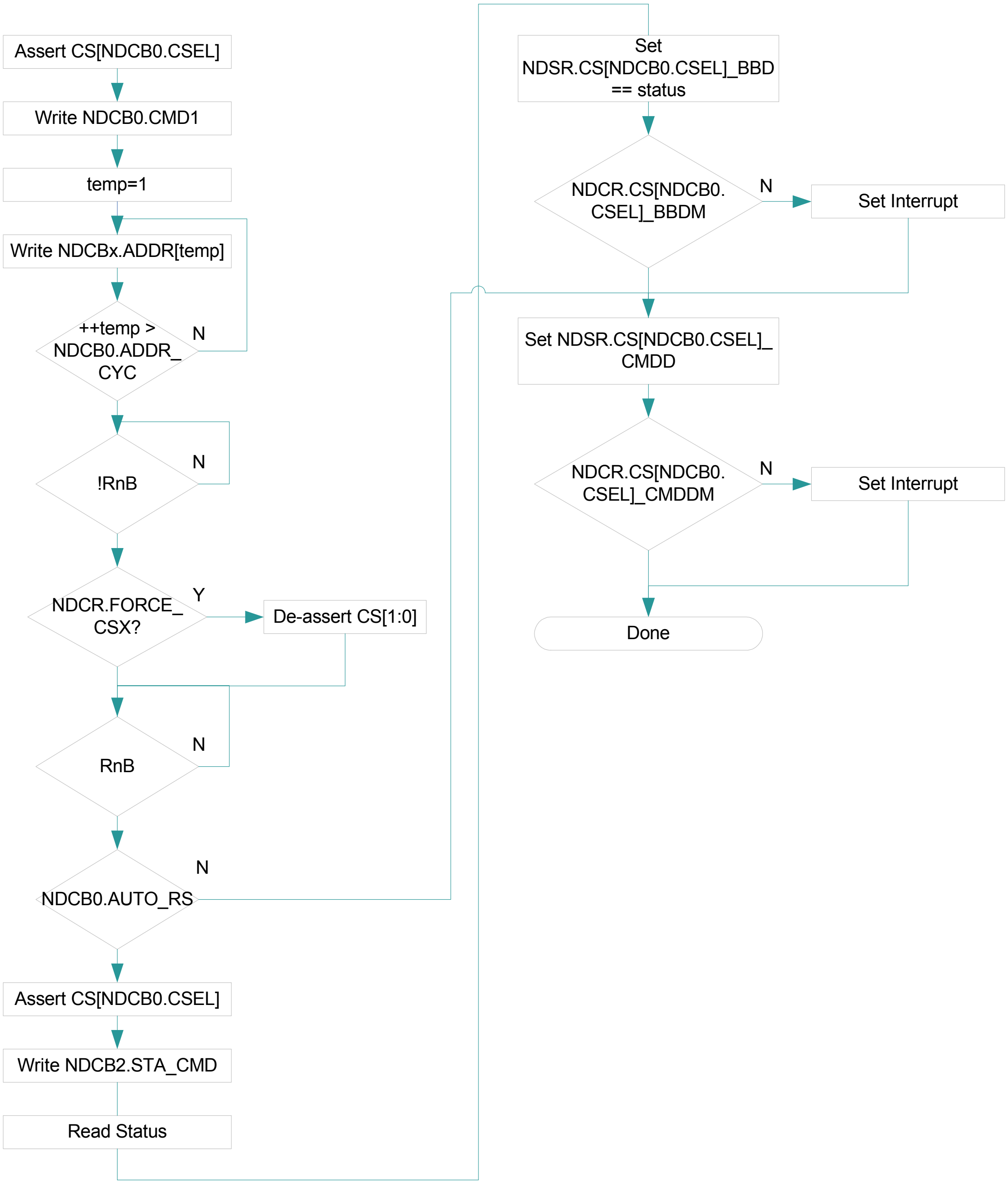
This can be extended yet again by using BCH on a 512B sector, again wasting 4B per syndrome. For this to operate, there would need to be a doubling of the spare area from 64B per 2-KB page to at least 120B per 2-KB page.

The same logic applies to 4-KB pages.

13.4.5 Command Semantics

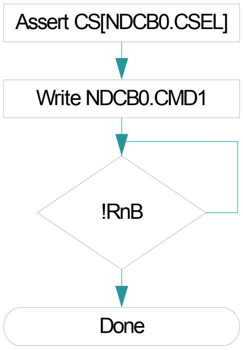
The command semantics are defined in the NAND Controller Command Buffer 0 (NDCB0) Register (Table 543 p. 881). This register defines how each NAND command type is to proceed, and how or if it may be overlapped. There are eight basic semantics: Read, Program, Erase, Read ID, Read Status and Reset, Naked Command and Naked Address. The semantics differ in the sequence of operations and in the count of operations. However, each semantic runs as a continuous operation. Read and Program semantics can be decomposed into their “naked” segments, and Program and Erase semantics allow overlap of commands to a different chip select.

Figure 38: Erase Semantic Flowchart



- **Read ID Semantic** The Read ID semantic is very similar to the Read semantic, only the length is fixed by a a different field to be either 2 or 4B. There is no concept of spare or ECC, but the flowchart is very similar, and therefore, is not replicated.
- **Read Status Semantic** The Read Status semantic is very similar to the Read semantic, only the length is fixed to 1B. There is no concept of spare or ECC, but the flowchart is very similar and therefore, is not replicated.
- **Reset Semantic** The Reset semantic issues a single command (CMD1) to the NAND, regardless of the programming of other command block parameters (such as NDCB0.DBC). Address bytes have no meaning, regardless of the address count. There is no data transfer regardless of any length settings.

Figure 39: Reset Semantic Flowchart



13.4.6 Bad Block Management Support

The NFC performs a status check after each program/erase operation to determine if the transaction was successful. This status check can be disabled by clearing the NAND Controller Command Buffer 0 (NDCB0) Register (Table 543 p. 881). If a status check returns an error, as indicated by a 1 in the LSB of the status-read data, a bad-block-detect interrupt (if enabled) is sent out. The Bad Block registers (see Table 533, Summary Map Table for the NAND Flash Registers, on page 856) can be read to determine the address of the bad block. The OS file system software marks the detected bad block as non-valid.

- If a bad-block error is encountered at the end of a page program, additional transactions to that block are aborted.
- If the error occurs during a page program in block A, software transfers other valid pages in block A to another unassigned block B and marks block A as non-valid.
- If an error occurs during a block erase, software is expected to mark the block as non-valid.

Command Execution When Bad Blocks are Detected

When a bad block is detected, the behavior of the NFC remains the same, regardless of whether commands are executed sequentially or in parallel. The command without a bad-block detection completes, but the command that resulted in bad-block detection does not execute. (If it is a program command, it must be executed again after the command descriptors have been reprogrammed to process the bad-block scenario.) As a result of a bad-block detection, the

ND_RUN bit in NDCR is cleared after the command buffer is emptied. Resume the operation by setting ND_RUN after modifying the command-descriptor.

13.4.7 Usage Examples

The following tables show how the application processor interacts with the NFC to schedule operations. These tables only document the use model. Each table shows resources in columns and steps in rows and the tables are intended to show how the hardware resources are used in sequence and in time. The right most column is the time equation (that is, how to compute the time taken for this step).

For each of the tables Table 48 through Table 51, time increases as one moves down the rows, although the time taken for each row is not constant. In most instances, the time equation column defines the time taken by each step. Each column describes the activities executed by the various hardware segments that make up the system.

Table 48 shows the sequence for reading a 2-KB page from a MLC device with BCH enabled.

Table 48: Read 2 KB MLC Page

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Decoder	Time Equation
Read command control block from memory					
Read commands from memory	Write “NAND Read” command sequence to NDCB0				
Read Data Move control block from memory		Read cmd 1			tWC
Stall		5-cycle address			5 * tWC
		Read cmd 2			tWC
	Busy				tR
				Reset	
		Read data	Write NAND flash data into Buffer	Compute syndrome	2048*tRC
		Read spare	Write NAND flash Spare into Buffer		(64-ECC)*tRC
		Read ECC			ECC*tRC
				Load BMA	
				Compute BMA	
				Load Chien search	

Table 48: Read 2 KB MLC Page (Continued)

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Decoder	Time Equation
Read data block from memory	Read data block from NDDB		Read buffer data to system bus	Compute ECC bits with Chien search	
	Interrupt				

Table 49 shows the sequence for reading a 4-KB page from a MLC device with BCH enabled. Because the buffer is not large enough to contain the entire page, the data is read from the NAND flash in 2-KB chunks.

Table 49: Read 4 KB MLC Page

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Decoder	Time Equation
Read Command control block from memory					
Read commands from memory	Write “NAND Read” command sequence to NDCB0				
Read Data Move control block from memory		Read cmd 1			tWC
Stall		5-cycle address			5 * tWC
		Read cmd 2			tWC
	Busy				tR
				Reset	
		Read data	Write NAND flash data into Buffer	Compute syndrome	2048*tRC
		Read spare	Write NAND flash Spare into Buffer		(64-ECC)*tRC
		Read ECC			ECC*tRC
				Load BMA	
				Compute BMA	
				Load Chien search	
Read data block from memory	Read data block from NDDB		Read buffer data to system bus	Compute ECC bits with Chien search	

Copyright © 2014 Marvell

Table 49: Read 4 KB MLC Page (Continued)

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Decoder	Time Equation
Read Command control block from memory					
Read commands from memory	Write “Naked Read” command sequence to NDCB0				
Read Data Move control block from memory					
Stall				Reset	
		Read data	Write NAND flash data into Buffer	Compute syndrome	2048*tRC
		Read spare	Write NAND flash Spare into Buffer		(64-ECC)*tRC
		Read ECC			ECC*tRC
				Load BMA	
				Compute BMA	
				Load Chien search	
Read data block from memory	Read data block from NDDB		Read data block	Compute ECC bits with Chien search	
	Interrupt				

Table 50 shows the sequence for writing a 2 KB page to a MLC device with BCH enabled.

Table 50: Program (Write) 2 KB MLC Page

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Encoder	Time Equation
Read Command control block from memory					
Read commands from memory	Write “NAND Write” command sequence to NDCB0				

Table 50: Program (Write) 2 KB MLC Page (Continued)

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Encoder	Time Equation
Read Data Move control block from memory					
Read data block from memory	Write data block to NDDB		Write system bus data to Buffer		
		Write cmd 1			tWC
		5-cycle address		Reset	5 * tWC
		Write data	Read data from buffer to NAND flash	Compute	2048*tWC
		Write spare	Read Spare from buffer to NAND flash	Compute	(64-ECC)*tWC
		Write ECC		Read Stall	ECC*tWC
		Write cmd 2			tWC
	Busy				tPROG
	Interrupt				

Table 51 shows the sequence for writing a 4 KB page to a MLC device with BCH enabled.

Table 51: Program (Write) 4 KB MLC Page

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Encoder	Time Equation
Read Command control block from memory					
Read commands from memory	Write “NAND Write dispatch” command sequence to NDCB0				
		Write cmd 1			tWC
		5-cycle address			5 * tWC
Read Command control block from memory					
Read commands from memory	Write “Naked Write” command sequence to NDCB0				

Table 51: Program (Write) 4 KB MLC Page (Continued)

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Encoder	Time Equation
Read Data Move control block from memory					
Read data block from memory	Write 2KB data block to NDDB		Write system bus data to Buffer		
				Reset	
		Write data	Read data from buffer to NAND flash	Compute	2048*tWC
		Write spare	Read spare data from buffer to NAND flash	Compute	(64-ECC)*tWC
		Write ECC		Read Stall	ECC*tWC
Read Command control block from memory					
Read commands from memory	Write “Naked Write” command sequence to NDCB0				
Read Data Move control block from memory					
Read data block from memory	Write 2KB data block to NDDB		Write system bus data to Buffer		
				Reset	
		Write data	Read data from buffer to NAND flash	Compute	2048*tWC
		Write spare	Read spare data from buffer to NAND flash	Compute	(64-ECC)*tWC
		Write ECC		Read Stall	ECC*tWC
Read Command control block from memory					
Read commands from memory	Write “NAND Write dispatch” command sequence to NDCB0				

Table 51: Program (Write) 4 KB MLC Page (Continued)

Application Processor and System Bus	NAND Flash Registers and Control	NAND Flash I/F Pins	Data Buffer	BCH Encoder	Time Equation
		Write cmd 2			tWC
	Busy				tPROG
	Interrupt				

13.5 Usage Models

With the introduction of the new semantics and the so-called “naked” semantics, the use of these command sequences to achieve any given goal may not be obvious. This section presents each basic command in sequence and shows how it may be used to construct the range of command sequences each command type can support.

13.5.1 Programming the Timing Registers

The two timing registers allow for considerable flexibility in control over interface timing. NAND flash device datasheets contain a large number of timing parameters, but not all of them are equally important.

The contents of the fields for each parameter are the number of NFC clocks (plus one). The granularity of the NFC clock is in ND_CLK periods (denoted as t(ND_CLK)). This means that all timing parameters are in increments of t(ND_CLK).

13.5.1.1 tWP, tWH (and tWC)

NAND flash datasheets have 3 components to any write cycle:

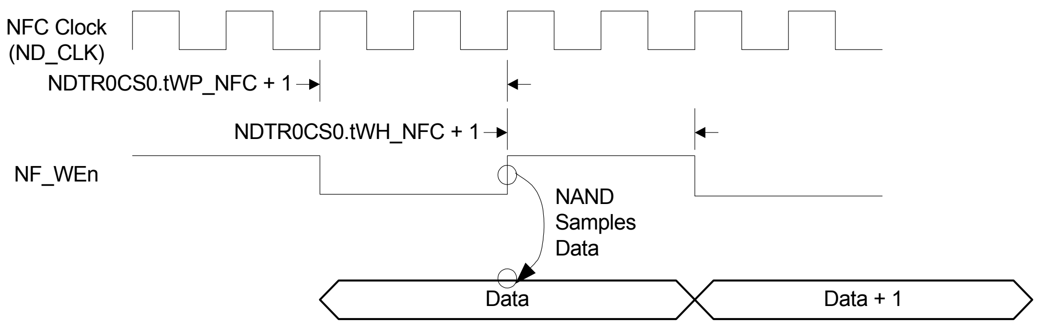
- The total cycle time (tWC)
- The active portion of the write (tWP, when NF_WEn is asserted)
- The hold portion of the write (tWH, when NF_WEn is not asserted and holding for the next tWP)

The value of tWC is not necessarily the sum of tWP and tWH as it is possible that tWC > tWP+tWH. When programming these two parameters make sure that (NDTR0CS0.tWP_NFC+1) * t(ND_CLK) is greater than or equal to the tWP parameter in the NAND flash datasheet. The same holds true for tWH. But in addition, it must to be verified that (NDTR0CS0.tWP_NFC+NDTR0CS0.tWH_NFC+2) * t(ND_CLK) is greater than the NAND flash datasheet value for tWC. It is entirely possible that NDTR0CS0.tWP_NFC and/or NDTR0CS0.tWH_NFC will need to be artificially increased to meet tWC, and if this is the case, refer to the AC timing in the device Hardware Specifications to determine the preferred parameter. If the output data valid time is critically aligned to the rising edge of NF_WEn, increase tWP_NFC, and if the data hold time to the NAND flash is critical, increase tWH_NFC.

It should be noted that these parameters effect the timing between NF_WEn edges of all write cycles of the same class. For example, a write cycle is composed of a command cycle, 5 address cycles, {page_size} data cycles and a command cycle. These parameters effect the relative write timings of the address cycles and data cycles, but the transition between command, address, and data may be effected by other timing parameters.

Figure 40 provides a timing example for these parameters. For the purpose of illustration, both tWP_NFC and tWH_NFC are set to 1, meaning that each portion of the pulse is two NFC clocks in length.

Figure 40: tWC Timing Example



The NFC clock (ND_CLK) is the internal clock for the NFC controller. NF_WEn and Data are also internally generated relative to the rising edge of NFC clock. There is I/O delay between the internal representations of NF_WEn and Data due to propagation through the I/O cell and capacitive loading on the wires. However, since all signals are unidirectional (that is, being sent to the NAND flash device only), the only AC timing considerations that require analysis is the potential for a differential skew between NF_WEn and the data (which may occur because of the possibility of different capacitive loads on the two signals). This differential skew is only important if tWP and/or tWH are critically timed, according to the AC parameters in the device Hardware Specifications and the NAND flash device datasheet.

Note

When the BCH ECC is used, NDTR0CS0.tWP_NFC + NDTR0CS0.tWH_NFC + 2 must be greater than or equal to 4 NFC clock (ND_CLK) cycles.

13.5.1.2 tRP, tRH (and tRC)

NAND flash datasheets have three components to any read cycle:

- The total cycle time (tRC)
- The active portion of the read (tRP, when NF_REn is asserted)
- The hold portion of the read (tRH, when NF_REn is not asserted and holding for the next tRP).

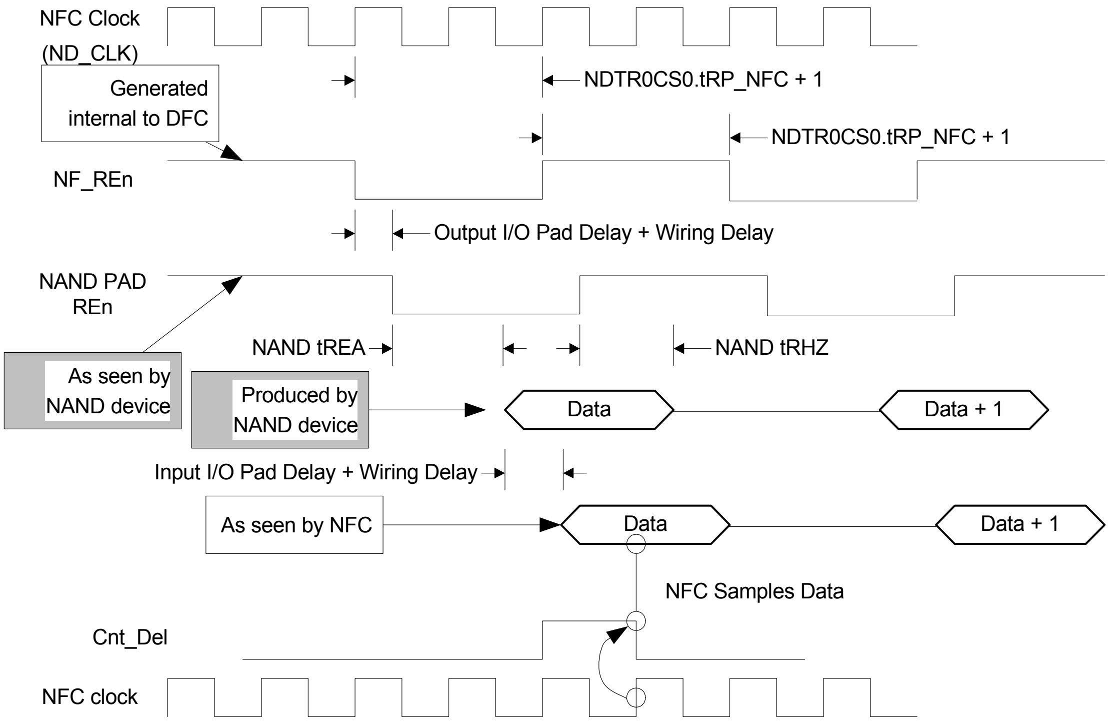
The value of tRC is not necessarily the sum of tRP and tRH, as it is possible that tRC > tRP+tRH. When programming these two parameters make sure that (NDTR0CS0.tRP_NFC+1) * t(ND_CLK) is greater than or equal to the tRP parameter in the NAND flash datasheet. The same holds true for tRH. But in addition, it must be verified that (NDTR0CS0.tRP_NFC+NDTR0CS0.tRH_NFC+2) * t(ND_CLK) is greater than the NAND flash datasheet value for tRC. It is entirely possible that NDTR0CS0.tRP_NFC or NDTR0CS0.tRH_NFC will need to be artificially increased to meet tRC. If this is the case, refer to the AC timing in the device Hardware Specifications to determine the preferred parameter. If the access time from the NAND flash is critical, increase the NDTR0CS0.tRP_NFC value, and if the hold time to the NFC is critical, increase the NDTR0CS0.tRH_NFC value. In addition, there is skew between the internal NFC clock and I/O pad that drives NF_REn. This skew effectively adds to tRP because NF_REn is “seen” by the NAND flash chip “late”. In addition, data presented to the NFC by the NAND chip has to go through I/O buffers, and this increases both the access and the hold time.

If NDTR0CS0.Rd_Cnt_Del is selected by NDTR0CS0.SelCntr, the sample clock is a fixed integer count from the rising edge of the NFC clock (ND_CLK) that generates the falling edge of NF_REn.

Figure 41 shows this alternative sample mechanism. In this figure, it is assumed that the I/O delays from the NFC to and from the NAND flash device place the Data window one full NFC clock cycle after the rising edge of NF_REn. This requires that NDTR0CS0.Rd_Cnt_Del have the value of 2.

In Figure 41, the NFC clock is shown twice only to simplify the figure.

Figure 41: tRC Timing Example Using Cnt_Del



Note

- When the BCH ECC is used, NDTR0CS0.tRP_NFC + NDTR0CS0.tRH_NFC + 2 must be greater than or equal to 4 NFC clock (ND_CLK) cycles.
- Since the NOR/NAND hardware arbiter is always enabled by default, the software must add a value of 3 to the <Rd_Cnt_Del> field in the NAND Interface Timing Parameter 0 Register (NDTR0CS0) (Table 535 p. 865) to compensate on internal sampling stages.

13.5.1.3 tCS

The NDTR0CS0.tCS_NFC parameter is used in four distinct contexts, only one of which is to address the NAND flash tCS parameter. The tCS parameter on NAND flash datasheets specifies the time between the assertion of chip select and the rising edge of NF_WEn. For the NFC, the specification of tWP specifies a portion of the tCS, because NF_WEn is asserted at a different time than NF_CSn. For this reason, the NDTR0CS0.tCS_NFC parameter actually specifies the positive difference between the value specified by NDTR0CS0.tWP and the value needed for tCS.

For example, where the NAND device tWP specification is 15 ns and the tCS specification is 30 ns, and the NFC clock period (t(ND_CLK)) is 5 ns. NDTR0CS0.tWP must be set to the value 2 so that (2+1) * 5 = 15 ns, which meets the tWP NAND flash specification. NDTR0CS0.tCS must be set to a

Copyright © 2014 Marvell

value that makes up the difference between 15 and 30. Since this means an additional 15 ns is required, and this value is obtained by setting tCS to the value of 2 so that (2 + 1) * 5 = 15 ns. The NAND flash requirement (15) is met in this case, since 15 + 15 = 30 ns.

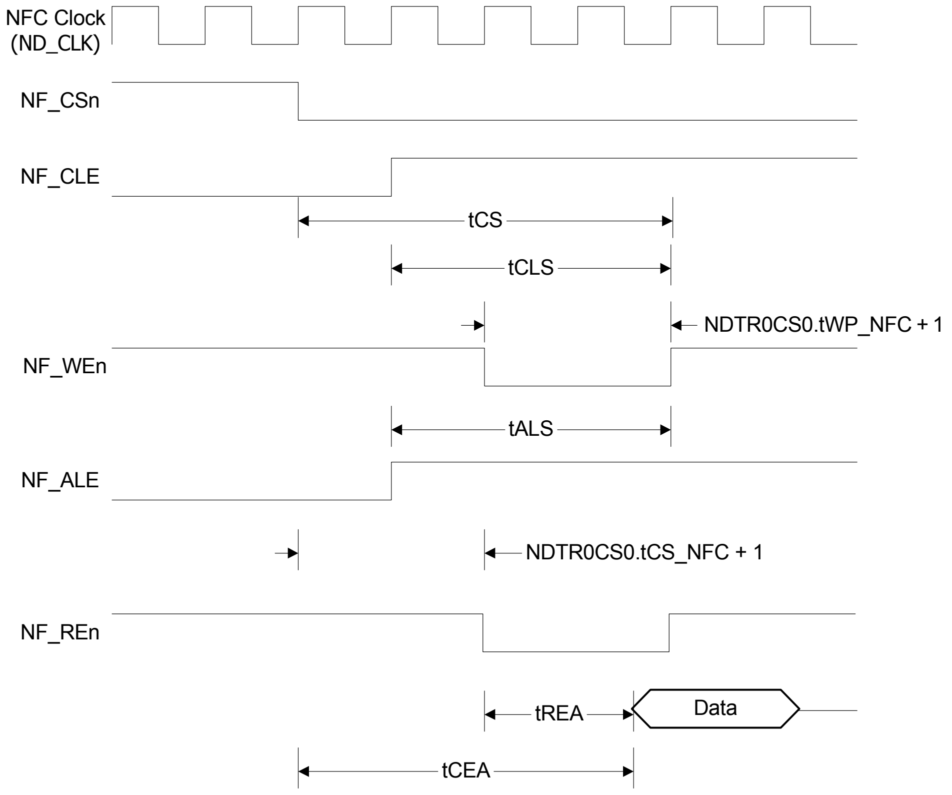
In addition there are two other parameters tCLS and tALS that are the CLE and ALE setup times, respectively, to the rising edge of NF_WEn. The NDTR0CS0.tCS parameter is also used to specify these values, again as the positive difference between tWP and tCLS and/or tALS, whichever is larger.

The last usage of this parameter is to meet the data valid time from CS when NDCR.FORCE_CSX is true. The NAND flash devices have a parameter tCEA that is the data access time from the activating edge of NF_CSn. The read data is valid tCEA, after falling edge of NF_CSn and tREA after the falling edge of NF_REn. The NDTR0CS0.tCS parameter is used to specify tCEA again as the positive difference between tREA (which is related to tRP) and tCEA.

The actual value needed for the NDTR0CS0.tCS parameter is the maximum of whichever values are needed to satisfy tCS, tCLS, tALS, and tCEA (when NDCR.FORCE_CSX is true).

Figure 42 is representative of all of the timing possibilities but not accurate logically as address, command, and data cycles occur at different times and of course it cannot be the case that both NF_REn and NF_WEn are simultaneously asserted.

Figure 42: tCS Timing Example



13.5.1.4 tCH

NAND flash devices have timing parameters that specify the hold time of the control signals (NF_CS, NF_CLE, NF_ALE) relative to the rising edge of NF_WEn. They are tCH, tCLH and tALH. There is no specific usage of the NDTR0CS0.tCH value in isolation as the actual constraint is the maximum of tCH, tCLH, tALH, and tWH.

The NDTR0CS0.tCH value is set to be the larger of the NAND flash tCH, tCLH, and tALH. The actual hold time implemented by the NFC is the maximum of NDTR0CS0.tCH or NDTR0CS0.tWH.

13.5.1.5 tADL

For program operations, the tADL parameter specifies the time between the rising edge of NF_WEn of the last address cycle and the rising edge of NF_WEn of the first data cycle. There are internal delays between the last address cycle and the first write cycle. In addition, a maximum value of NDTR0CS0.tWH_NFC or NDTR0CS0.tCH_NFC is part of the data cycle. The NDTR0CS0.tADL_NFC field in the NAND Interface Timing Parameter 0 Register (NDTR0CS0) (Table 535 p. 865) must be set to the difference between the NAND's device tADL requirement and the Max(NDTR0CS0.tWH_NFC,NDTR0CS0.tCH_NFC) * t(ND_CLK) ns.

13.5.1.6 tAR and tWHR

The tAR parameter is the setup time between the falling edge of NF_ALE and the falling edge of NF_REn for a read cycle. The only time the NFC encounters this case is for a Read ID where this timing parameter must be met between the Read ID address and the first ID data value.

The tWHR parameter is the write hold time between the rising edge of NF_WEn and the falling edge of NF_REn specifically for status reads.

To the NFC, a Read ID is similar to a status read and different from a page read, because neither a Read ID nor a status read cycle wait for RBn.

The parameters must be set to one less than the total required time (in NFC clock units).

13.5.1.7 tR

There are multiple ways in which the tR parameter is used. The most obvious is of course to match the NAND flash device tR parameter. If NDTR1CS0.WAIT_MODE is false, a read will wait the time specified in NDTR1CS0.tR_NFC and then proceed to generate read data cycles. In this case, it is important that tR be programmed to meet or exceed the actual NAND flash device tR parameter. If NDTR1CS0.WAIT_MODE is true, a read will wait at least the time specified in NDTR1CS0.tR, and then may continue to wait until NDSR.RDYx actually asserts. In this case, NDTR1CS0.tR_NFC must be programmed to meet or exceed the NAND flash device tWB parameter that is the delay between the rising edge of NF_WEn and the falling edge of NF_RBn.

There is a case that is described in more detail in overlapped program operations where one can use the tR value to simply act as a delay. A “naked read” waits for tR and then perform zero or more data cycles. The key is the zero, because if NDCB0.LEN_OVRD is true and NDCB3.NDLENCNT is set to zero a “zero length naked read” will act as a simple timing delay. This solves a particular problem with overlapped multi-plane, multi-device writes, as described in Section 13.5.5, Program (Write) Usage, on page 233.

13.5.2 Entering a First Command

There are two operations to be performed to enter a first command:

- The NFC must be started (that is, NDCR.ND_RUN must be set)
- A three or four word command must be written to the command FIFO at NDCB0.

If the NFC is idle, these operations can be performed in either sequence. If the ND_RUN is set first, the software should wait until NDSR.WRCMDREQ is TRUE. If the NFC is idle, the command buffer will be ready for a command immediately, and the first test should return TRUE. However, it should never be assumed that (in ND_RUN mode) the NFC is ready for a command unless NDSR.WRCMDREQ is TRUE. Once WRCMDREQ is TRUE, it must be cleared in NDSR.

If the NFC is idle, the NDCB FIFO can be first written with the command and then NDCR.ND_RUN set. In this case the software does *not* wait for NDSR.WRCMDREQ before writing the NDCB FIFO. If the command FIFO contains a command and NDCR.ND_RUN is set, the command immediately begins to sequence its execution. For a write type command, NDSR.WRDREQ will set indicating that the NFC is ready for and awaiting data to be written to NDDB. For any other command, the command begins to sequence on the DFI, and for read type commands, NDSR.RDREQ will be set later, after data has been written into NDDB (see Figure 43).

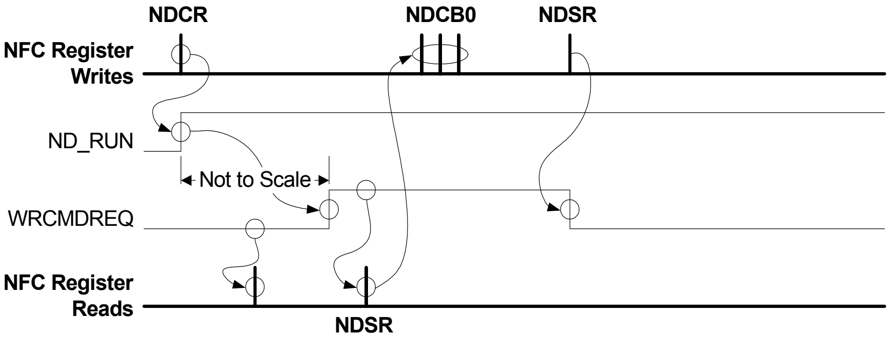
Note

In Figure 43 through Figure 46:

- A dark line signifies an event caused by a processor operation
- A narrow line indicates an internal NFC state
- A circle on a vertical edge indicates an event that causes a state change (where the arrow indicates the cause-effect relation)
- A circle on a horizontal line indicates a sampling operation (where the arrow indicates the reason for the sample).

These diagrams indicate the sequence of operations and events, and are not intended to convey absolute time.

Figure 43: First Command Event Causality Diagram



What happens next depends on the actual command. A reset or erase command will wait for NDSR.CMDD, because there is no data transfer. A read (Read ID, Status) command will wait for NDSR.DTREQ to indicate that data is ready to be read. After reading all data, it should be true that NDSR.CMDD is also set. A program command will wait for NDSR.WRDREQ to indicate that data is ready to be written, and then will wait for NDSR.CMDD.

It is not necessary for software to poll the status register to observe status state changes, a software program may be interrupt driven. What follows is representative of a polled software driven usage model presented for the purpose of example and is not intended to be canonical.

13.5.3 Reset Usage

A reset semantic is a single command cycle followed by NDSR.RDYx sensing. For this semantic, the value of the following commands are Don't Care:

- NDCB0.CMD2
- NDCB0.ST_ROW_EN
- NDCB1
- NDCB0.ADDR_CYC
- NDCB0.LEN_OVRD
- NDCB2
- NDCB0.DBC,
- NDCB0.CMD_XTYPE
- or optionally NDCB3

However, it would be preferable to leave all of these fields as zero. For a reset semantic the only fields that matter are:

- NDCB0.CMD1
- NDTR1CS0.tR_NFC
- NDTR1CS0.WAIT_MODE

NDCB0.AUTO_RS should be false.

As is true of any semantic, it is not required that a reset semantic actually issue a reset command. So long as the desired operation is a single command cycle followed by NDSR.RDYx sensing, the semantic can be useful.

An example of a non-reset use of this reset semantic would be:

1. Issue a Naked Command cycle of 0x00 (that is, NDBC0.CMD_TYPE == 0b110).
2. Issue a 5-beat Naked Address cycle (that is, NDBC0.CMD_TYPE == 0b111 and NDCB{1,2} contains Address).
3. Issue a Reset command of 0x30.
4. Issue a Naked Read (that is, NDBC0.CMD_TYPE == 0b000 and CMD_XTYPE == 0b001).

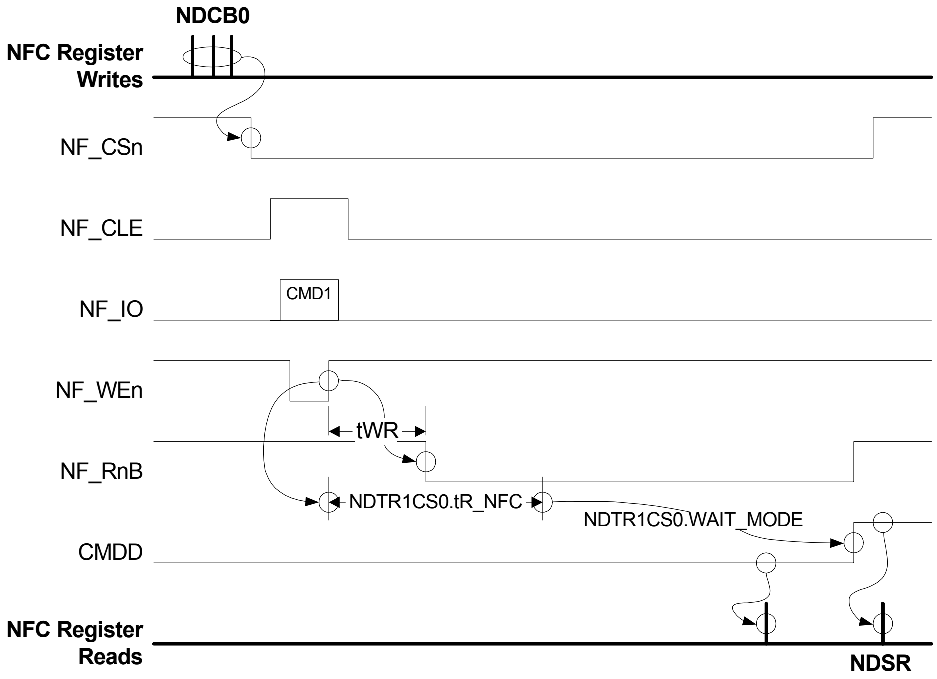
This sequence of four commands has the same effect as a "Monolithic Read". The Naked Command issues the first read command cycle (0x00), the Naked Address issues the 5-cycles of read address, the Reset command issues the Read Commit command cycle (0x30) and waits for NDSR.RDYx. The last command actually reads the data from the NAND and requests that NDDB be emptied. While this is a contrived example, it should illustrate the point that semantics can be strung together in novel ways, as they are needed.

The implementation uses the NDTR1CS0.tR_NFC timing value and NDTR1CS0.WAIT_MODE to wait for NDSR.RDYx.

The busy time for a Reset command can vary considerably, from microseconds to milliseconds, depending on the exact circumstances and individual NAND flash devices. The NDTR1CS0.tR_NFC register has a limited dynamic range since it is only 16 bits, and can only measure 420 µs, at the maximum value. To implement the reset in the initial platform case, set the NDTR1CS0.tR_NFC register to a value greater than the value listed in the NAND flash datasheet for tWB (the time from the rising edge of NF_WEn to the falling edge of NF_RBn) and set NDTR1CS0.WAIT_MODE. The software should wait for CMDD to become true. When CMDD becomes true, if NDSR.RDYx has been set, there is a NAND flash present, because it cycled NF_RBn, and if NDSR.RDYx has not been set, a NAND flash is not present in that chip enable position. In this way, the NFC is guaranteed to execute the Reset command to completion, in the NAND present case and is guaranteed to correctly sense the absence of a NAND flash device.

It is possible that the NF_RBn is not available. In this case, the state of NF_RBn is not reliable and cannot be used to determine the end of the Reset command. In this case, if NDTR1CS0.WAIT_MODE is reset, tR does not have enough dynamic range, and if NDTR1CS0.WAIT_MODE is set, the NFC may lock while waiting for a NDSR.RDYx indication that it will never receive. For this reason, a NDTR1CS0.PRESCALE has been added that extends the dynamic range of the NDTR1CS0.tR_NFC value by 16. When NDTR1CS0.PRESCALE is used, the effective range of the NDTR1CS0.tR_NFC register is increased to 6.72 ms. If a Reset instruction based discovery is required on platforms where NF_RBnx is connected to something other than a NAND flash device, it will be necessary to set NDTR1CS0.tR_NFC to its maximum value and to also set NDTR1CS0.PRESCALE. This causes the Reset command to always take 6.7 ms, but it also guarantees that there will not be a lockup. In this case, the state of NDSR.RDYx may or may not be a reliable indication of the presence of the NAND flash, as it depends on exactly what the NF_RBn pin is connected to and whether or not it is possible for that pin to assert a positive edge during the Reset command (see Figure 44).

Figure 44: Reset Command Event Causality Diagram



13.5.4 Read ID Usage

A Read ID semantic is a CMD1 command cycle, 0–7 address cycles, a max(NDTR1CS0.tAR,NDTR1CS0.tWHR) timing wait and 0–7 data cycles. There is no RBn wait. Required fields are:

- NDCB0.CMD1
- NDCB0.ADDR_CYC and
- Either NDCR.RD_ID_CNT or NDCB0.LEN_OVRD and NDCB3.NDLENCNT

The following fields are Don’t Care but should be set to zero:

- NDCB0.CMD_XTYPE
- NDCB0.ST_ROW_EN
- NDCB0.DBC
- NDCB0.CMD2

NDCB0.AUTO_RS should be false. The number of required ADDR fields in NDCB{1,2} are a function of the ADDR_CYC value.

The most obvious use for a Read ID is to read the NAND ID. In addition, for NAND flash devices that are ONFI compliant, the Read ID can be used to determine this compliance. In the normal ID case, there is a single 0x00 address cycle and either 2, 4, or 5 data cycles. For the ONFI ID case, the address cycle is set to 0x20, and there will be 4 data cycles: O, N, F, and I.

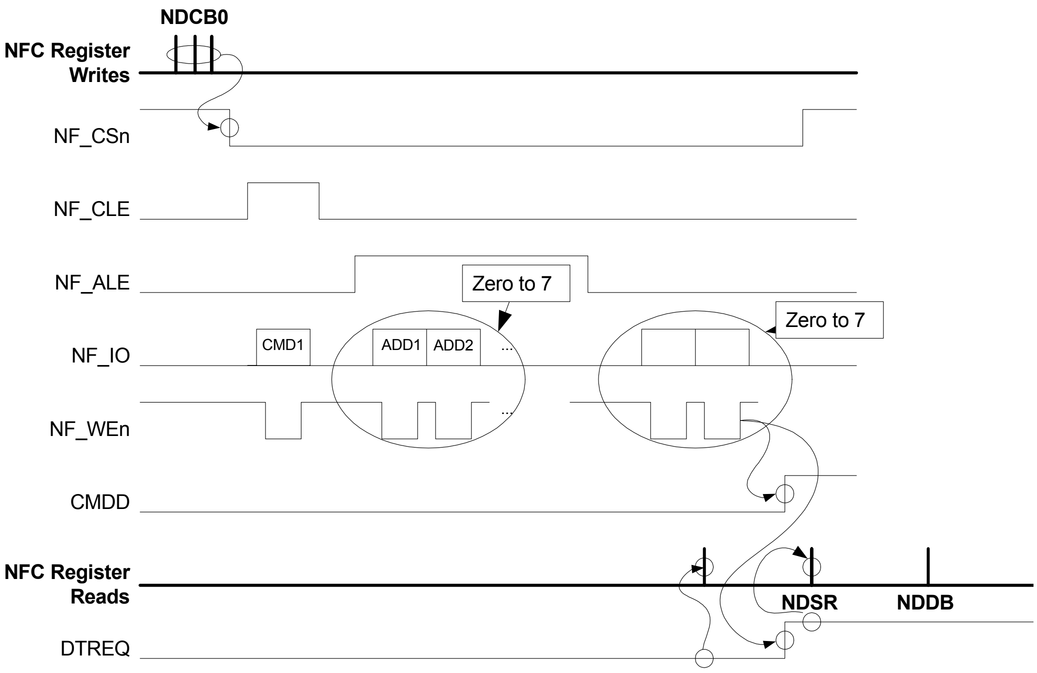
Another possible usage model for the Read ID is for block unlock commands on NAND flash devices that support the lock, lock-tight, and unlock region commands. The unlock command is typically a command cycle followed by three block address cycles to specify the start of the unlocked region, followed by another command cycle with three block address cycles to specify the end of the unlocked region. There should be no data cycles; so to use this semantic as a unlock region command would require that the NDCR.RD_ID_CNT field be set to zero.

There are certain NAND flash devices (or NAND flash device specifications) that have command-address sequences where the single command is a function, and the single address is like an argument for the function. These sequences can be generated as a Read ID with NDCR.RD_ID_CNT set to zero.

Note

- The address cycle count and data cycle counts are now allowed to be zero. If there are zero address cycles specified, no address cycles are generated, and NF_ALE does not assert. If there are zero data cycles specified, no data cycles are generated. Previously the minimum value for either of these fields was defined to be 1, but if these either or both of these fields were set to zero, a minimum of 1 cycle of each type would have been generated.
- For Read ID commands, the software must disable ECC. Clear the <ECC_EN> field in the Data Flash Control Register (NDCR) (Table 534 p. 857) and the <BCH_EN> field in the NAND ECC Control (NDECCCTRL) Register (Table 540 p. 879).

Figure 45: Read ID Command Event Causality Diagram



13.5.5 Program (Write) Usage

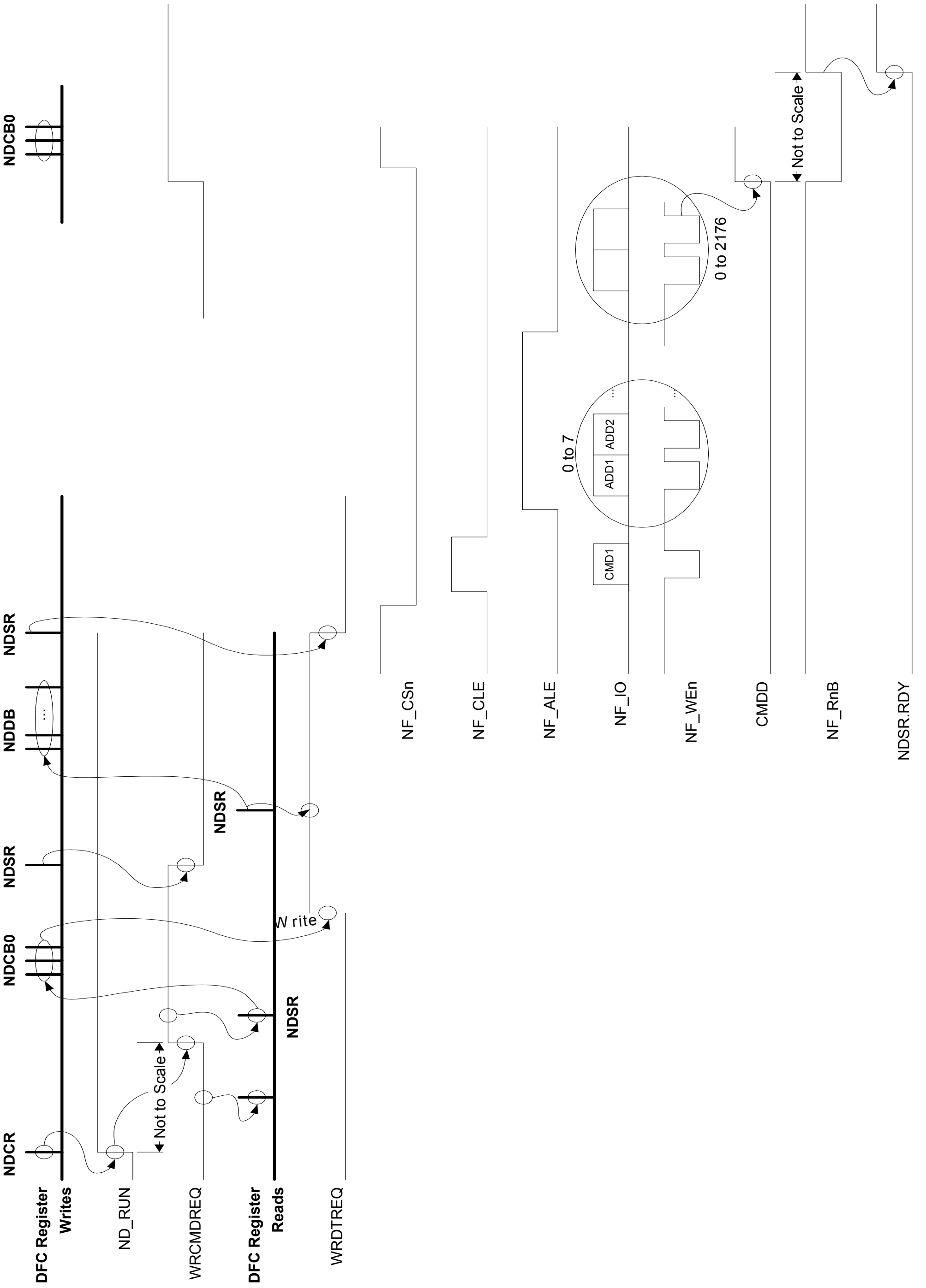
The program command can be implemented in a number of ways, there are rules for overlapping operations. Now there is support for larger pages and ganging, and, in general, the issues with program operations have become more complex.

A program is a program command, address, write data, and program commit sequence. A write completes its operation in the NFC once the program commit command is issued (that is, NDSR.CMDD sets). It is possible to overlap another program or erase command with a previous program or erase command if NDCB0.NC is true, NDCB0.RDY_BYP is false, and the second program or erase is to a different chip select. A first program or erase command starts by setting NDCR.RUN, waiting for NDSR.WRTCMDREQ, and writing NDCBx with the program command values. Since this is a program, data is required, so next wait for NDSR.WRDREQ. Once the data has been written, the program cycle is sequenced to the NAND flash device. After the command has been sequenced to the NAND flash device, NDSR.CMDDx is set to indicate that the NFC is ready for another command. If NDCB0.NC is true, the NFC sets NDSR.WRCMDREQ and waits for the processor to write in a second command. If the chip selects of the 2 operations specify the same NAND flash device and NDCB0.RDY_BYP is false, the second command stalls while waiting for NDSR.RDYx. If the chip selects of the 2 operations are different and the second command is a write or erase, the NFC continues to process the command, assuming that the RDY indication of this other chip is not also indicating busy.

After the NAND flash has finished the program cycle, it will assert NF_RBn to indicate ready. NDSR.RDYx sets on the rising edge of NF_RBn.

Figure 46 shows an event causality diagram where a program command is being entered, and optionally if NDCB0.NC is set, a second command is entered for overlap.

Figure 46: Write Command Event Causality Diagram



Copyright © 2014 Marvell

Programming Larger Pages

To support larger page sizes, the program command needs to be broken into two commands, using Naked semantics or otherwise known as <CMD_XTYPE> field in the NAND Controller Command Buffer 0 (NDCB0) Register (Table 543 p. 881).

The data transfer portion of the program command is at most 2112 bytes (or 2110 bytes with BCH).

To program a 4 KB page device, the software can do one of the following:

Option 1

1. Issue the initial command or command dispatch: <CMD_XTYPE> = 110.
2. Issue Naked write command: <CMD_XTYPE> = 101.
   and
   Transfer up to 2112 bytes.
3. Issue a second Naked write command
   and
   Transfer another 2112 bytes.
4. Issue the final command: <CMD_XTYPE> = b011.

Option 2

1. Issue the initial command or command dispatch with write: <CMD_XTYPE> = 100.
   and
   Transfer the command, address, 2112 bytes data.
2. Issue Naked write command <CMD_XTYPE> = 101.
   and
   Transfer another 2112 bytes.
3. Issue the final command <CMD_XTYPE> = 011.

Option 3

1. Issue the initial command or command dispatch with a write: <CMD_XTYPE> = 100.
   and
   Transfer the command, address, 2112 bytes of data.
2. Issue Naked write with final command: <CMD_XTYPE> = 001.
   and
   Transfer another 2112 bytes, as well as the final confirm command.

The data transfer portion of the program command is at most 2112 (or 2110 with BCH) bytes. To support larger page sizes, the program command must be broken into two commands using “naked” semantics:

- A first program command can issue the program command, address and 2112B of write data.
- A second program command can issue 2112B of write data and the commit command.

The program is composed of two commands, but the total data transfer will support 4-KB page sizes. In the event that 4-KB page NAND flash devices are ganged to create a logical 8-KB page, this may be supported by:

- A first program command that issues the program command, the address and 2112B of write data.
- A second program command can issue 2112B of write data.
- A third program command can issue 2112B of write data.
- A fourth program command can issue 2112B of write data and the commit command.

Naked data writes can be performed to enable any page size. The only caveat is that the data format always has the ECC associated with each data chunk (up to 2 KB).

Read Usage

The Read command can be implemented in a number of ways and it has supports for larger pages and ganging, and in general the issues with read operations have become more complex.

A Read semantic is an initial command, address, confirm, or commit command (only for devices that support double byte reads), NAND device pulling RBn Low followed by data transfer. A Read command is considered complete in the NFC only after the read data in the NAND Controller Data Buffer (NDDB) Register (Table 542 p. 880) is read by the CPU.

Reading Larger Pages

To support larger page sizes, the read command needs to be broken down using Extended semantics or Naked semantics.

To read a 4 KB page, software can preform the following steps:

1. Issue command dispatch: <CMD_XTYPE> = 110.
2. Issue a naked read command: <CMD_XTYPE> = 101).
   and
   Read 2 KB bytes.
3. Finally issue a second naked read.
   and
   Read the second 2 KB chunk.

Note

- After every 2 KB read from the device the NAND Controller Data Buffer (NDDB) Register needs to be drained.
- To support even larger page sizes, software must perform multiple 2 KB reads.

Naked Command and Naked Address

There are cases where the command and address sequences are not standard or represent new features. In this case, the NFC may not have built-in commands to perform the function(s).

- The ability to issue a naked command means that any NFC operation that depends on CLE being true can be issued.
- The naked address semantic means any sequence of ALE cycles (from 1 to 7) can be issued.
- A naked read cycle can read between zero and 2176B as can a naked write.

In this way, sequences of CLE cycles, ALE cycles, and data cycles can strung together in an almost arbitrary sequence. The only limitation is that the chip select for the NAND flash device will cycle between each operation. The primary purpose of these new semantics is to support new features, particularly the command sequences associated with logical block accessed NAND flash device.

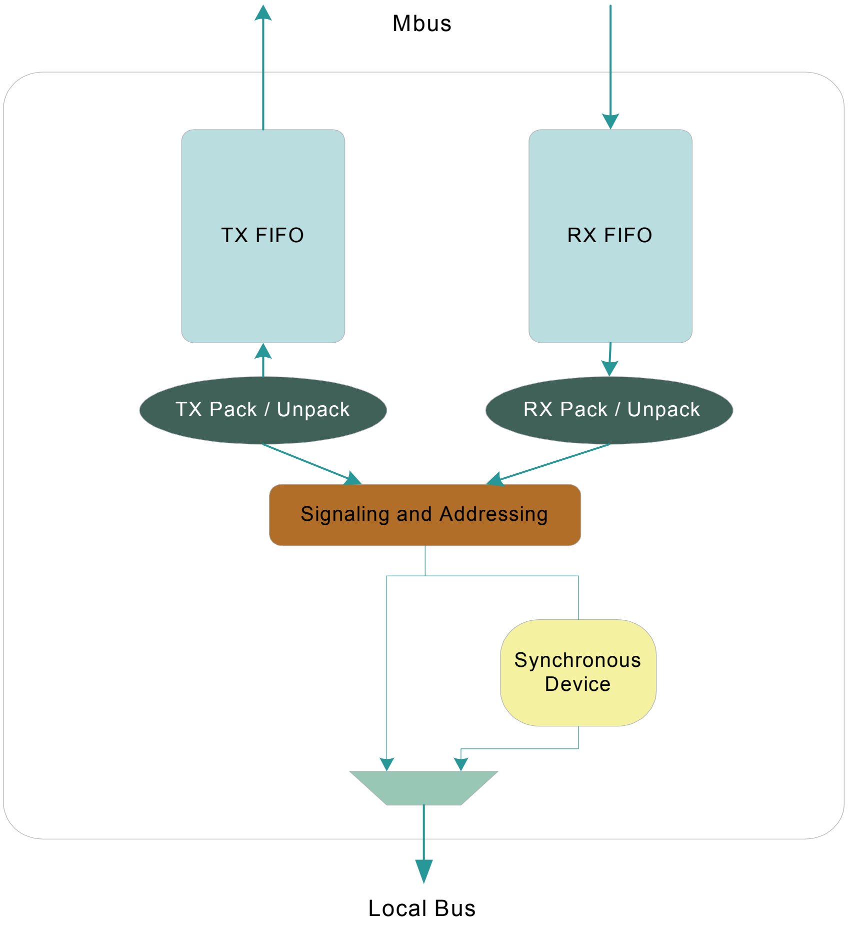
14 Device Bus Controller

This section describes the Device Bus controller. The Device Bus controller is a generic local bus controller that enables the device to interface with off-the-shelf NOR type flashes, SRAMs and custom FPGAs. It describes the different connectivity options of the interface, the various programmable timing parameters and the interface signaling.

The Device Bus registers are located in Appendix A.5, Device Bus Registers, on page 889.

Figure 47 presents the Device Bus Controller block diagram.

Figure 47: Device Bus Controller Block Diagram



14.1 Features

The Device Bus includes the following features:

- 32-bit multiplexed address/data bus
- Supports different types of standard memory devices such as Flash and ROM
- 5 chip selects with programmable timing
- Optional external wait-state support
- 8/16/32-bit width device support
- Up to 128B burst per a single device bus access
- Synchronous device bus support

14.2 Functional Description

The Device Bus controller supports up to 5 banks of devices. Each bank supports up to 512 MB of address space, and has its own timing parameter register. The bank width can be programmed to 8, 16, or 32 bits. The bank timing parameters can be programmed to support different device types (for example, Flash, ROM, I/O Controllers).

The 5 chip selects are typically separated into 4 individual chip selects (DEV_CSn[3:0]) and 1 chip select for a boot device (DEV_BOOTCSn). The only difference between the boot device bank and any of the other banks is that, by default, the CPU core boots from the boot device and its default width is sampled at reset.

The Device Bus controller supports a multiplexed address/data bus. External latches should be used to latch the address (up to 512 MB of address space). The interface supports up to 128B transfer per a single device access.

Note

- When used as 16-bit interface, DEV_D[23:16] and DEV_WEn[3:2] can be used as Multi Purpose Pins (see the “Pin Multiplexing” section in the device *Hardware Specifications*).
- When used as 8-bit interface, DEV_AD[15:9] and DEV_WEn[1] can also be used for pins multiplexing.

Figure 48 provides the connectivity of devices using 2 external latches over the Device Bus.

Copyright © 2014 Marvell

Figure 48: Device Bus Connectivity (Using 2 External Latches)

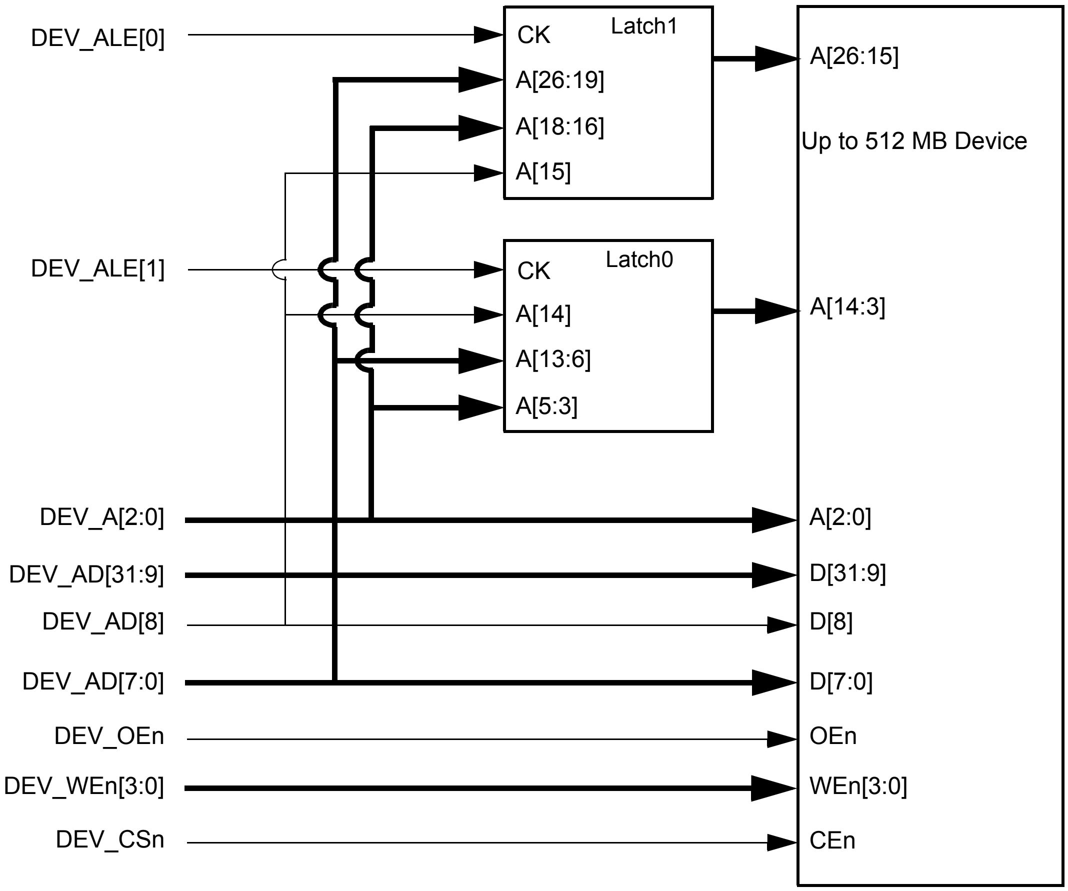
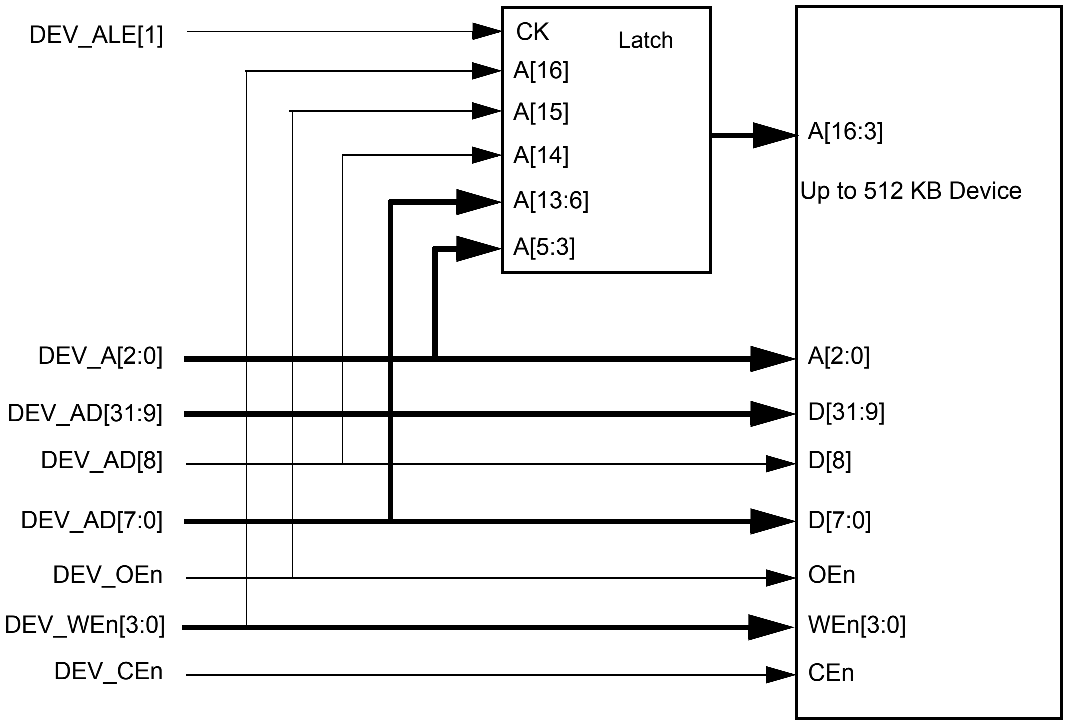


Figure 49 provides the connectivity over the Device bus for devices using a single external latch with address space of up to 512 KB.

Figure 49: Device Bus Connectivity for up to 512 KB Address Space (Using a Single External Latch)



14.3 Address Multiplexing

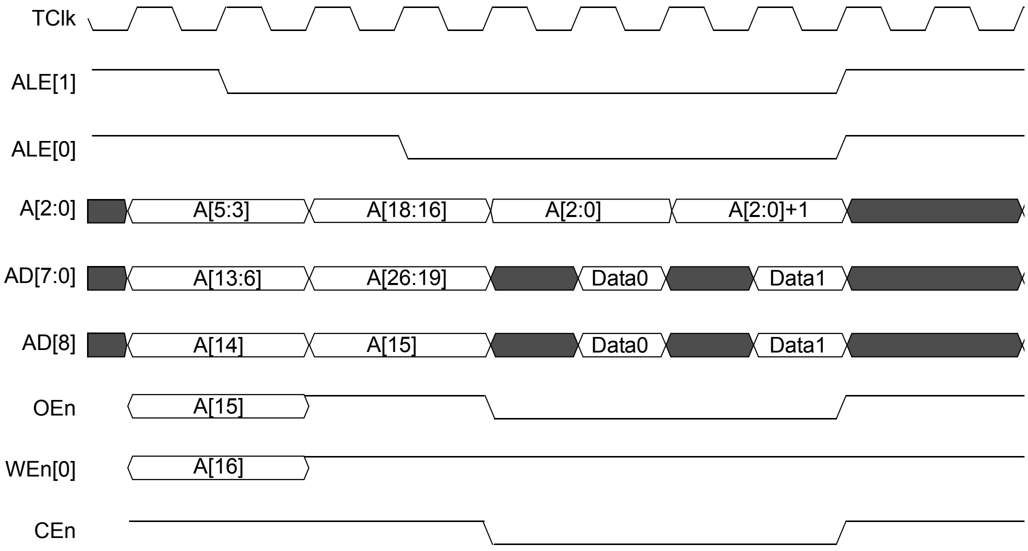
Figure 50 provides a diagram of the address multiplexing.

- There are 2 address phases (ALE[1:0]) followed by data phase.
- This figure is for 32-bit devices. For 16-bit or an 8-bit devices. the address is shifted right.

Note

In some of the figures that follow, the signal names appear without the DEV_ prefix.

Figure 50: Address Multiplexing



Note

- DEV_OEn and DEV_WEn are inactive (high) when the Device bus in idle (that is, from DEV_CSn de-assertion at the end of a transaction until the next transaction DEV_ALE[1] assertion).
- If the <OEWE_shared> field in the Device Bus Interface Control Register (Table 552 p. 893) is cleared to 0, DEV_OEn and DEV_WEn are kept inactive from DEV_CSn de-assertion at the end of a transaction until next transaction's DEV_CSn assertion. This means that for a two latch scheme (see Figure 50), <OEWE_shared> must be 0x0. To support boot from Device in the desired configuration, <OEWE_shared> is sampled at reset.

14.4 Device Bus Timing

14.4.1 Device Bus Address Phase Timing

As described, a transaction address phase consists of two parts—latching of low address bits followed by latching of high address bits.

In an optimal utilization configuration, each of these parts is 2 TCLK cycles long, with DEV_ALE asserted after 1 cycle. This timing results in 1 TCLK cycle setup time and 1 TCLK cycle hold time of address in respect to ALE fall. Since DEV_ALE is loaded with a single latch device, while DEV_AD might be heavily loaded, the actual setup time can become shorter and not sufficient. To solve this issue, the device implements the <Addr2ALE> field in the Device Bus Interface Control Register (Table 552 p. 893) to define the number of cycles in which the address is steady before any ALE assertions.

Note

Since the NOR/NAND hardware arbiter is always enabled by default, the actual setup time is 8 clock cycles longer than configured in the <Add2ALE>.

14.4.2 Device Bus Read Timing Parameters

To allow flexible interfacing to slow and fast devices, the interface can be programmed with different timing parameters for each Device bus chip select.

TurnOff

The <TurnOff> parameter defines the number of TCLK cycles that the device does not drive the AD bus after the completion of a Device read. This prevents contentions on the Device bus after a read cycle from a slow device. The actual number of TCLK cycles from DEV_CSn de-assertion to next DEV_ALE[1] negation is <TurnOff> - <RdHold>.

The minimum setting for this parameter is <RdHold>+2.

Acc2First

The <Acc2First> parameter defines the number of TCLK cycles from the negation of ALE[0] to the cycle that the first read data is sampled by the device. When the maximum configurable value does not meet the requirements, DEV_READYn can be used to further delay the sample (see Section 14.6, READYn Support, on page 248).

The number of TCLK cycles is <Acc2First>-2*<Addr2Ale>-1.
The minimum setting of this parameter is 2 * <Addr2Ale>+2+<RdSetup>+2.

NOTE: When interfacing a synchronous device, set to a value greater than <Acc2Next> + 4.

Copyright © 2014 Marvell

Acc2Next

The <Acc2Next> parameter defines the number of TCLK cycles between the cycle that samples data N to the cycle that samples data N+1 (in burst accesses). When the maximum configurable value does not meet the requirements, DEV_READYn can be used to further delay the sample (see Section 14.6, READYn Support, on page 248).

The minimum setting for this parameter is 0x2

NOTE: When interfacing a synchronous device, set to the same value as
<Tclk Divide Value>-2 (see the in the Device Bus Sync Control Register (Table 554 p. 894)).

CS<n>Setup (n = [3:0])

The <CS<n>Setup> parameter defines the number of TCLK cycles between address stable point and CS assertion (see the General Purpose 1 Register (Table 1443 p. 1517)). If cleared (default), DEV_CSn assertion will be done on the same TCLK cycle as setting of DEV_A[2:0]. If set, DEV_A[2:0] will be stable 4 TCLK cycles before DEV_CSn assertion.

NOTE: When enabling the <CS<n>Setup> parameter and running with a small value for ALE2Wr, DEV_WEn asserts before the CS. Therefore, add the restriction:
(ALE2Wr >= 2*(Addr2ALE+1) + 4).

RdSetup

The <RdSetup> parameter defines the number of TCLK cycles between the end of the address phase and DEV_OEn assertion. If cleared to 0 (default), DEV_CSn assertion, DEV_OEn assertion, and DEV_A[2:0] setting will be occur on the same cycle.

To assure compliancy with other timing configurations:

1. Set <RdSetup> to a value smaller than <Acc2First>-2*<ALE2Addr>-2.
2. If <CsSetup> is enabled, set <RdSetup> to a value which is greater or equal to 4.

NOTE: RdSetup < ALE2Wr.

RdHold

The <RdHold> parameter defines the number of TCLK cycles between the last data sample to the de-assertion of DEV_CSn. If cleared to 0 (default), DEV_OEn and DEV_CSn are de-asserted at the same cycle (the cycle of the last data sample).

The <RdHold> parameter has no affect on DEV_OEn de-assertion. DEV_OEn is always de-asserted the next cycle after last data sampled.

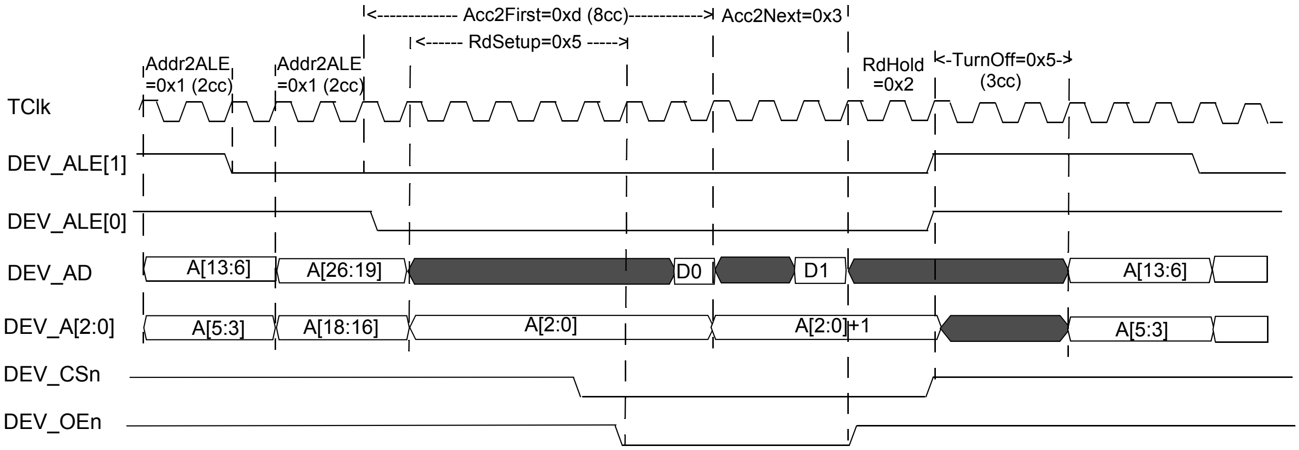
Also, the <RdHold> parameter has no affect on <TurnOff> parameter. Set <RdHold> to a value smaller than <TurnOff>.

RdHoldADEn

Setting the <RdHoldADEn> field in the General Purpose 1 Register (Table 1443 p. 1517) forces DEV_A[2:0] to remain stable until the de-assertion of DEV_CSn.

Figure 51 shows a device read timing parameters example.

Figure 51: Read Parameters Example



14.4.3 Device Bus Write Timing Parameters

To allow flexible interfacing to slow and fast devices, the interface can be programmed with different timing parameters.

ALE2Wr

The <ALE2Wr> field in the DEV_CSn[n] WriteParameters Register (n=0–3) (Table 551 p. 892) defines a guaranteed gap between address phase and DEV_WEn assertion. The number of cycles from DEV_ALE[0] negation to DEV_WEn assertion will be at least<ALE2Wr>-2*<Addr2ALE>-3.

The minimum setting of this parameter is 2*(<Addr2ALE>+1)+2.

NOTE: RdSetup < ALE2Wr.

WrLow

The <WrLow> field in the DEV_CSn[n] WriteParameters Register (n=0–3) (Table 551 p. 892) defines the number of TCLK cycles that DEV_WEn is active (low). Extend this parameter by extending the READYn pin (see Section 14.6, READYn Support, on page 248). DEV_A[2:0] and Data are kept valid as long as DEV_WEn is active. This parameter defines the setup time of address and data to DEV_WEn rise. The minimum setting of this parameter is 0x1.

WrHigh

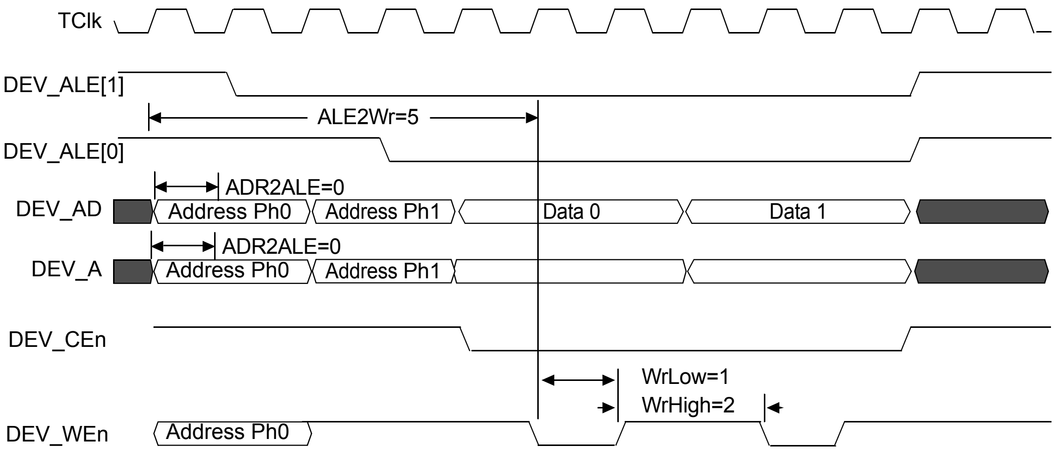
The <WrHigh> field in the DEV_CSn[n] WriteParameters Register (n=0–3) (Table 551 p. 892) defines the number of TCLK cycles that DEV_WEn is kept inactive (high) between data beats of a burst write. DEV_A[2:0] and Data are kept valid (do not toggle) for <WrHigh>-1 periods. This parameter defines the hold time of address and data after DEV_WEn rise. The minimum setting of this parameter is 0x1.

If setting <WrHigh> to 1, DEV_A[2:0] and write data are toggled on the same cycle DEV_WEn toggles from High to Low.

Copyright © 2014 Marvell

Figure 52 shows a device write timing parameters example.

Figure 52: Write Parameters Example



The Device bus is typically used for interfacing asynchronous devices that do not require clock input. However, it can also be used to interface synchronous devices. The MV78230/78x60 can drive this clock on the DEV_CLK_OUT pin.

DEV_CLK_OUT reflects the internal core clock (TCLK). It can be configured to drive a divided TCLK via the <Tclk Divide Value> field in the Device Bus Sync Control Register (Table 554 p. 895). This can be useful for interfacing synchronous devices that can not run as fast as TCLK frequency.

Note

Device bus signals are driven and sampled on the rising edge of the core clock (TCLK), even when the chip is configured to drive a divided TCLK on DEV_CLK_OUT pin.

On odd divider values, the duty cycle of the driven clock will not be 50%.

Interacting with a Synchronous Device

When DEV_CLK_OUT is used for interacting with a synchronous device the timing parameters configurations must correlate to the required clock frequency. In particular, the following restrictions must hold:

- <Acc2First> must be greater than <Acc2Next> + 4.
- In case of a burst read, DEV_READYn can be used to delay the sampling point of the first data (that is, it can extend <Acc2First>) but it cannot be used to delay the sampling point of the following phases (that is, it cannot extend <Acc2Next>).
- <WrLow> + <WrHigh> must equal <Tclk Divide Value>.
- <ALE2Wr> must be greater than <Tclk Divide Value>.

14.4.4 Device Bus Transaction Timing Summary

The following section summarizes the Device Bus configuration options presented in Section 14.4.1, Device Bus Address Phase Timing, on page 242 through Section 14.4.3, Device Bus Write Timing Parameters, on page 244. It provides additional clarification due to inherited dependencies between the defined timing configurations.

The section contains diagrams and tables that provide a method for defining the required minimal durations of non-connected gaps based on system requirements and then applies the configurations to meet these requirements.

14.4.4.1 Read Transaction Timing

Read Timing Gaps

Figure 53 provides a diagram of the Read timing for the NOR interface.

Figure 53: NOR Interface Read Timing Diagram

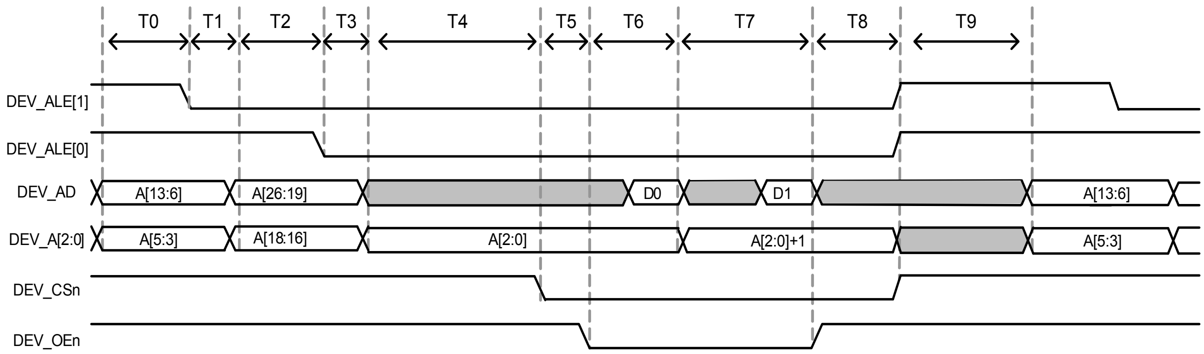


Table 52 provides a description of the timing segments and the possible range of values that can be configured to achieve a guaranteed minimal gap.

Table 52: Read Timing Parameters

Label	Description	Minimum
T0	Number of TCLK cycles from first address phase stable cycle to DEV_ALE[1] negation	1,2,3
T1	Number of TCLK cycles from DEV_ALE[1] negation in which the address remains stable	1
T2	Number of TCLK cycles from the second address phase stable cycle to DEV_ALE[0] negation	1,2,3
T3	Number of TCLK cycles from DEV_ALE[0] negation in which the address remains stable	1
T4	Number of TCLK cycles from the end of the address phase to assertion of DEV_CSn	0,4
T5	Number of TCLK cycles from DEV_CSn assertion to DEV_OEn assertion	0,1..30
T6	Number of TCLK cycles from DEV_OEn assertion to the cycle in which the first data is sampled	1,2..30
T7	Number of TCLK cycles from DEV_A[2:0] toggle to the cycle in which the corresponding data is sampled	1,2..63
T8	Number of TCLK cycles from DEV_OEn de-assertion to DEV_CSn de-assertion	0,1..31
T9	Number of TCLK cycles from DEV_ALE[1:0] release until the first cycle of the address phase of the following transaction	1,2..30

Read Timing Configurations

Given a set of values for T0-T9 with in the configuration as defined in Table 52, the following expressions provide the required configuration:

- <Addr2ALE> = T0–1 = T2–1
- <CsDelay> = If (T4==0), <CsDelay> = 0x0, otherwise <CsDelay> = 0x1
- <RdSetup> = T4 + T5
- <Acc2First> = T0 + 1 + T2 + 1 + T4 + T5 + T6
  <Acc2Next> = T7
- <RdHold> = T8
- <TurnOff> = T9 + T8

Copyright © 2014 Marvell

Ready Support

An inactive Ready signal delays Acc2First (gap T6) or Acc2Next (gap T7).

14.4.4.2 Write Transaction Timing

Write Timing Gaps

Figure 54 provides a diagram of the Write timing for the NOR interface.

Figure 54: NOR Interface Write Timing Diagram

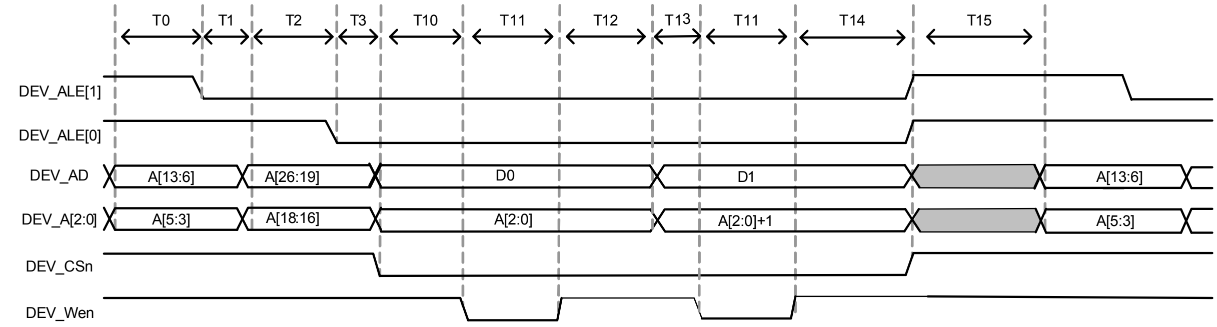


Table 53 provides description of the timing segments and the possible range of values that can be configured to achieve a guaranteed minimal gap.

Table 53: Write Timing Parameters

Label	Description	Minimum
T0	Number of TCLK cycles from the first address phase stable cycle to DEV_ALE[1] negation	1,2,3
T1	Number of TCLK cycles from DEV_ALE[1] negation in which the address remains stable	1
T2	Number of TCLK cycles from the second address phase stable cycle to DEV_ALE[0] negation	1,2,3
T3	Number of TCLK cycles from DEV_ALE[0] negation in which the address remains stable and DEV_CSn is not asserted	1
T10	Number of TCLK cycles from the point that DEV_CSn is asserted and the first address is stable until DEV_WEn assertion	1,2,3..55
T11	Number of TCLK cycles in which DEV_WEn is kept low	1,2,3..63
T12	Number of TCLK cycles from DEV_WEn rise in which the address and data remains stable	
T13	Number of TCLK cycles from the address and data setting until negation of DEV_WEn	1
T14	Number of TCLK cycles from DEV_WEn last rise until DEV_ALE[1:0] release	1,2,3..63
T15	Number of TCLK cycles from DEV_ALE[1:0] release of a write transaction until the first cycle of the address phase of the following transaction	5

Write Timing Configurations

Given a set of values for T0–T2,T10–T15 with in the configuration as defined in Table 53, the following expressions provide the required configuration:

- <Addr2ALE> = T0–1 = T2–1
- <ALE2Wr> = T0 + 1 + T2 + 1 + T10

- <WrLo> = T11
- <WrHi> = T12+1 = T14 + 1

Ready Support

An inactive Ready signal delays WrLo (gap T11).

14.5 Data Pack/Unpack and Burst Support

The Device interface is an 8-,16-, 32-bit wide interface and supports up to a 128B burst. The Device controller contains a single 128B buffer for RX and 128B buffer for TX. When a 8b or 16b device is used, Multiple 8B bursts are performed towards the device until the data transfer is completed. When a 32b device is used, Multiple 32B bursts are performed towards the device until the data transfer is completed.

The burst address on the external interface is supported by a dedicated 3-bit A[2:0] signals.

A[2:0] must be connected directly to the device address bus (as opposed to the latched address on the multiplexed AD bus—see Figure 48). The Device bus controller supports packing/unpacking of data between the external 8-/16-/32bit device to the internal 64-bit data path.

As previously described, the Device controller needs to pack read data from the 8-, 16-, 32-bit wide Device to the internal 64-bit data path. The Device controller drives the read data to the initiator only when the transaction on the Device bus completes and all data resides on its internal buffer.

14.6 READYn Support

READYn input is used to extend the programmable device timing parameters. This is useful for two cases:

- Interfacing a very slow device that has access time greater than the maximum programmable values
- Interfacing a device with a non deterministic access time (access time depends on other system events and activity)

READYn can extend these timing parameters <Acc2First>, <Acc2Next>, and <WrLow> in the following registers:

- DEV_BOOTCSn Read Parameters Register and DEV_BOOTCSn Write Parameters Register
- DEV_CSn[n] Read Parameters Register (n=0–3) and DEV_CSn[n] WriteParameters Register (n=0–3)

During a read access, the Device controller first counts TCLK cycles based on <Acc2First> programmable parameters. If at the time <Acc2First> is expired, READYn input is not asserted, it keeps waiting until READYn is sampled asserted, and only then samples first read data. Similarly, if at the time <Acc2Next> is expired, READYn is not asserted, it keeps waiting until READYn is sampled asserted, and only then samples next read data. On a write access, if at the time WrLow is expired, READYn input is not asserted, it keeps driving write data until READYn is sampled asserted. Figure 55, Figure 56, and Figure 57 show examples of the READYn operation.

Note

- When interfacing a device with a non-deterministic access time, set the timing parameters to their minimum values, and the actual access time is controlled via the READYn pin.
- When working in synchronous mode, READY can be used to extend <Acc2First> only (see Section, Interacting with a Synchronous Device, on page 245).

Figure 55: READYn Extending Acc2First Example

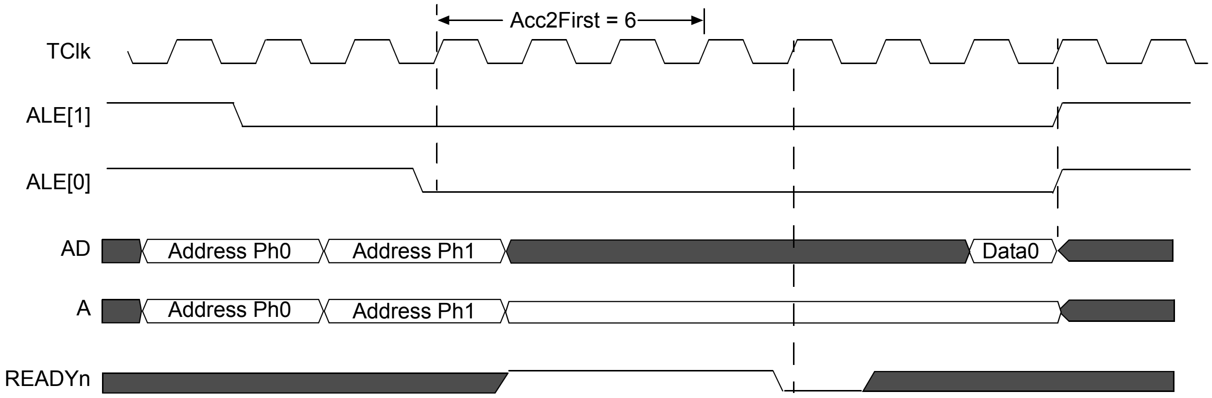


Figure 56: READYn Extending Acc2Next Example

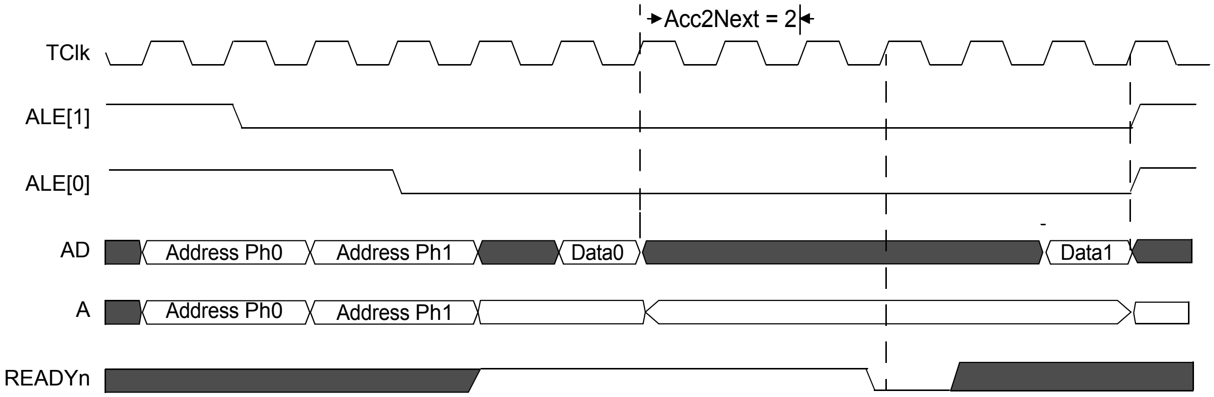
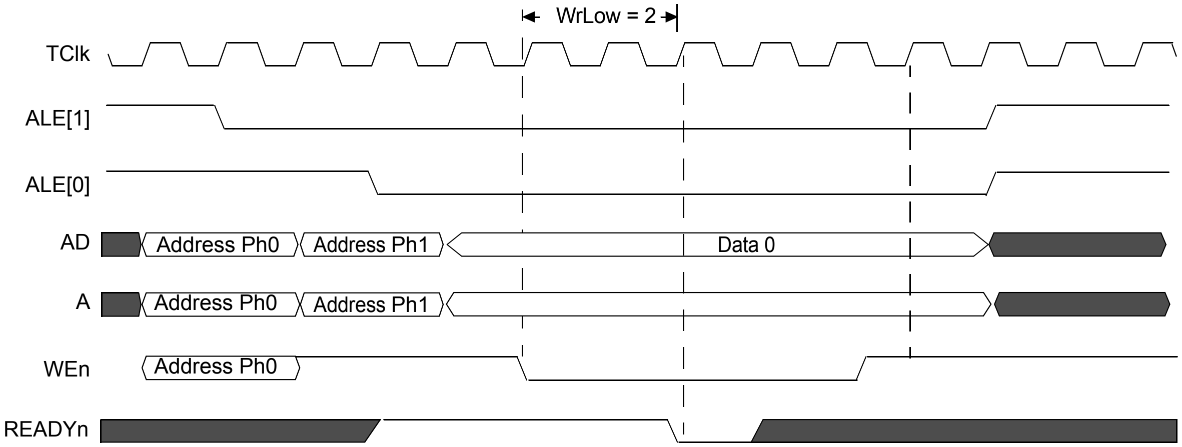


Figure 57: READYn Extending WrLow Example



By default, the Device bus controller samples read data after DEV_READYn assertion on a read access.

Each Device bus chip select can be configured to support DEV_READYn signal or not in the CSx Ready ignore fields in the Device Bus Sync Control Register (Table 554 p. 894). If configured not to support DEV_READYn, access time is controlled by the timing parameters, If configured to support DEV_READYn, read and write timing is controlled by DEV_READYn assertion as described above.

If interfacing with multiple devices on the Device bus which require DEV_READYn support, external logic is required to qualify the DEV_READYn input with the appropriate device chip selects.

To prevent system hang due to a lack of READYn assertion, the device implements a <Timeout> field in the Device Bus Interface Control Register (Table 552 p. 893) programmable timer. This field allows for a termination of a device access even without READYn assertion.

If during a device access the time-out timer expires, the Device controller completes the transaction as if READYn was asserted, generates an interrupt, and latches the accessed address in <Addr> field in the Device Bus Error Address Register (Table 553 p. 893). This might result in bad data read/write from/to the device. Setting the timer to 0x0 disables the timer, and the Device controller waits for READYn forever.

Note

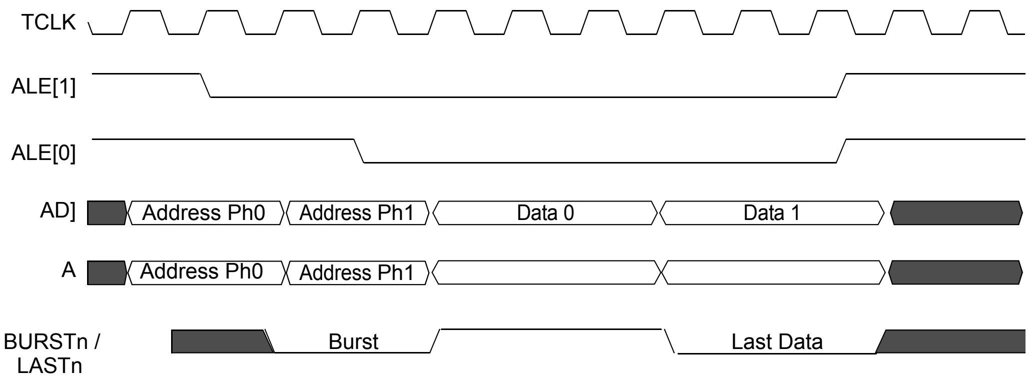
The timer must be programmed to a number that must never be exceeded in normal operation.

14.7 Additional Device Interface Signaling

To make it easy to glue external logic on the Device bus, the device supports a burst and last indication. BURSTn/LASTn is driven low on the address phase to indicate a burst access and is driven low on the last data phase to indicate the last data transfer. It needs to be latched via ALE[0].

Figure 58 shows an example of these additional signals.

Figure 58: BURSTn/DEV_LASTn Example



For full details, see Section 6, BootROM Firmware, on page 93.

Copyright © 2014 Marvell

MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Part 4: Ethernet Networking Subsystem

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. **Moving Forward Faster**

Document Classification: Proprietary Information

THIS PAGE IS INTENTIONALLY LEFT BLANK

15 Ethernet Networking Controller

This section describes the Ethernet Networking Controller.

In addition to this section, the Networking Subsystem includes the elements described in the following sections:

- Section 16, Hardware Buffer Management Controller, on page 301
- Section 17, 1G/2.5G Ethernet MAC, on page 308
- Section 18, Precise Time Protocol (PTP), on page 320

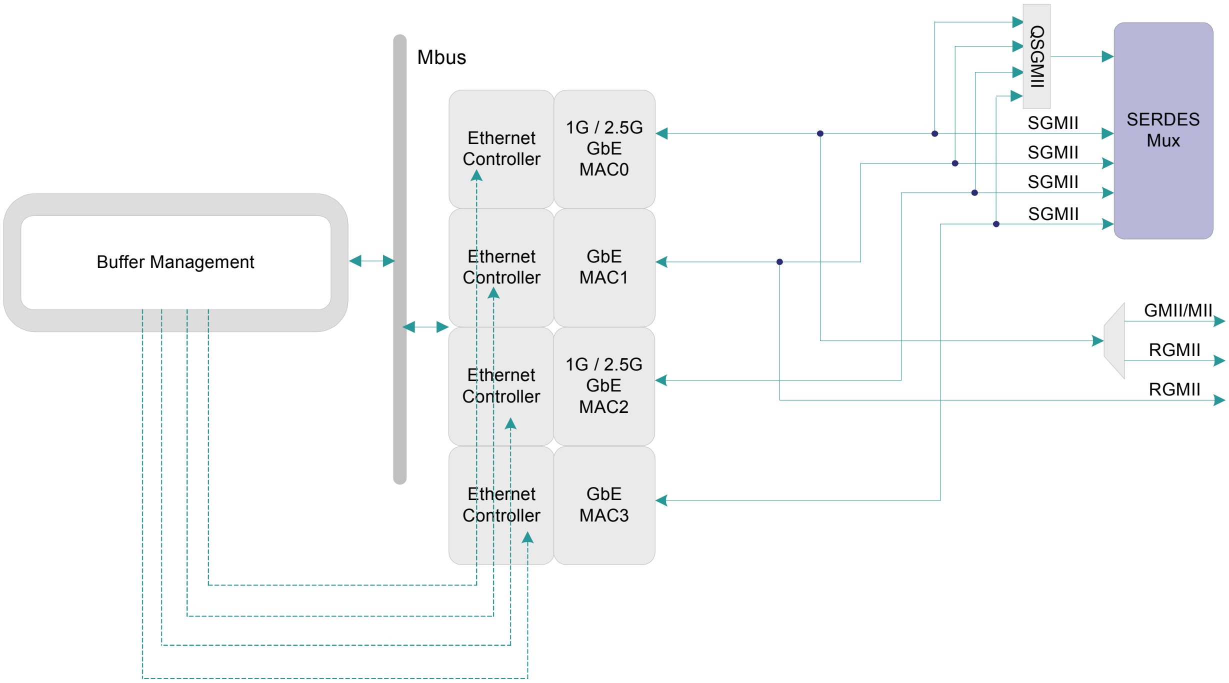
Note Not all the features described in this section are supported by all the of the devices. Refer to Section 1, Product Overview, on page 35 and Section 2.12, Ethernet Networking Controller, on page 47 for a description of the features supported by each of the devices.

To configure the ports to support SGMII mode, set the <PcsEn> field in the Port MAC Control Register2 (i=0–3) (Table 704 p. 1010) to 1x0. The SGMII mode can support 2.5G Ethernet MAC speed, which is configured by the SERDES speed.

The registers for the Ethernet controller are located in Appendix A.6, Ethernet Networking Controller and MAC Registers, on page 897.

Figure 59 provides a block diagram of the Ethernet Networking controller.

Figure 59: Ethernet Networking Controller Block Diagram



15.1 Features

The networking controller provides the following features:

- Transmit functions:
  - 8 Tx queues.
  - Rate limitation for each Tx queue and for the aggregated port traffic
  - Configurable strict priority/WRR arbitration between the queues
  - Short frame (less than 64B) zero padding
  - Long frames transmission (including jumbo frames), with IPv4/v6/TCP/UDP checksum generation
  - Cycle Redundancy Check (CRC) generation
  - Error reporting
- Receive functions:
  - Advanced unicast DA address filtering.
  - 256 Multicast DA filtering
  - Hardware parsing
    - Layer 2: BPDU, programmable VLAN-EtherType (VLAN), Ethernet v2, LLC/SNAP
    - Layer 3: IPv4, IPv6
    - Layer 4: TCP, UDP
  - Detects packet type/encapsulation that can be used by the host processor for packet routing.
  - IPv4 and TCP/UDP over IPv4/IPv6 checksum check
  - 8 Rx priority queues
  - Queuing is performed based on DA, VLAN-Tag, and IP-TOS.
  - Automatic discard of error frames (CRC error smaller than the programmable minimum frame size)
  - Reception of long frames (programmable legal frame size is up to 9700B)
    NOTE: Frames larger than the limit are actually received, *clipped to the programmed maximum legal length*, and marked in the descriptor as oversize errors.
  - Error report
- Wake On LAN (WOL) options for power save modes
- Support Marvell® proprietary Marvell Header or DSA Tag, resulting in significant CPU off-load when interfacing with Marvell switches.
- Energy Efficient Ethernet (EEE) LPI mode, for overall minimal system power consumption in power save modes.
- Precise Timing Protocol (PTP) with:
  - Precise time stamp for packets, as defined in IEEE 1588 PTP v1 and v2 and IEEE 802.1AS draft standards
  - Flexible Time Application interface to distribute PTP clock and time to other devices in the system
  - Optionally accepts an external clock input for time stamp

- IEEE 802.1Qav Audio Video Bridging networks:
  - Egress Jitter Pacer (EJP): Time- and priority-aware egress pacing algorithm for AVB-Class A and AVB-Class B traffic, and strict priority for legacy traffic queues.
  - Prevent bunching and bursting effects—suitable for audio/video applications

- The hardware buffer management supports buffer allocation and release without software intervention.

  For ingress packets: The networking controller gets the buffer(s) directly from the Buffer Management (BM) controller (see Section 16, Hardware Buffer Management Controller, on page 301).
  For egress packets: The networking controller releases the buffer(s) directly to the BM controller.
  NOTE: The buffer allocation and release by the software is also supported.
- Simple hardware-to-software interface for Rx and Tx queuing management.

  This reduces software overhead and improves performance.
- Receive and transmit queuing is based on a buffer-descriptor list. Descriptors and data transfers are performed by the port-dedicated DMA.
- IPv6 support: TCP/UDP checksum check (RX) and generation (TX), IPv6 based QoS.
- The networking controller leaves configured space in the header of the buffer for software use (for example, to add the VLAN header) by writing the packet to the buffer with an offset.
- Interrupt coalescing for received packets (time and packet based).

15.2 Networking Controller Modes Overview

During initialization, the networking controller mode is configured for the Receive (Rx) and Transmit (Tx) paths by the <AccelerationMode> field in the Port Acceleration Mode (PACC) Register (i=0–3) (Table 586 p. 930).

Buffer Management mode options

- Legacy buffer management in which buffers allocation and release is accomplished by the software only.
- Hardware buffer management in which buffers allocation and release is accomplished by the BM controller or the software.

Table 54 lists the coding for the <AccelerationMode> field.

Table 54: Networking Controller Modes and Features

<AccelerationMode> value:	1	2
<AccelerationMode> Feature	Queue: Counter Parsing: Hardware BM: Software	Queue: Counter Parsing: Hardware BM: Hardware/ Software
Queues handling method	By counters	By counters
RX DMA leaves configured space in the header of the buffer for software use	Yes	Yes
Buffers management method	Software only	Hardware or software: Supporting virtual address pointer for hardware allocated buffers

Table 54: Networking Controller Modes and Features (Continued)

<AccelerationMode> value:	1	2
<AccelerationMode> Feature	Queue: Counter Parsing: Hardware BM: Software	Queue: Counter Parsing: Hardware BM: Hardware/ Software
L2 CRC generation (Tx) and checking (Rx)	Yes	Yes
L3/L4 checksum checking (Rx)	IPv4 TCP/UDP over IPv4/IPv6	IPv4 TCP/UDP over IPv4/IPv6
L3/L4 checksum generation (Tx)	IPv4 TCP/UDP over IPv4/IPv6	IPv4 TCP/UDP over IPv4/IPv6
Wake On LAN	Yes	Yes
PTP	Yes	Yes
Interrupt coalescing for Rx queues	Time based and descriptors based	Time based and descriptors based
Interrupt coalescing for Tx queues	Descriptors based	Descriptors based
Descriptors format	32B Rx descriptors 16B Tx descriptors	32B Rx descriptors 16B Tx descriptors
Rx snoop support	Yes	Yes
L2 deposit support	Yes	Yes
Prefetch support	Yes	Yes
Multi-CPU support	Yes	Yes
Port Flow Control	Yes	Yes
Marvell Header/DSA tag	Yes	Yes
Transmit arbitration (strict priority, WRR)	Yes	Yes
Tx Bandwidth limitation resolution	10 Kbps	10 Kbps
Configurable Inter packet gap	Yes	Yes

15.3 Functional Description

The networking controller provides Ethernet ports functionality, with the port(s) capable of running at either 10 or 100 Mbps (half or full duplex) and 1000 Mbps (full duplex only). In addition, 2500 Mbps is supported. The networking controller manages packet data transfer between memory and PHY. The data is stored in memory buffers. A packet may be stored in single buffer or multiple buffers (and multiple descriptors, as each descriptor is pointing to a single buffer) if necessary.

IPv4 checksum, TCP over IPv4/v6 checksum, and UDP over IPv4/v6 checksum are always checked on received packets, and may be generated for transmit.

This capability increases performance significantly by off-loading these operations from the software. Checksum is checked for Jumbo frames on received packets. Checksum generation for transmitted Jumbo frames is supported.

Each networking controller includes eight dedicated receive DMA queues and eight dedicated transmit DMA queues, plus two dedicated DMA engines (one for receive and one for transmit) that operate concurrently.

The networking controller provides programmable zero padding of short transmitted frames (frames that are less than 64B).

Detailed status is given for each receive frame in the packet descriptors, while statistics are accumulated for received and transmitted traffic in the MIB counters, on a per port basis.

The Ethernet controller handles all functionality associated with moving packet data between local memory and the networking interface.

Received frames smaller than the programmable minimum frame size are automatically discarded. Reception and transmission of long frames, up to 9700B, are supported. The frame type, encapsulation method, errors, and checksums are reported in the buffer descriptor. Automatic IP header 32-bit alignment is done in memory by adding 2B at the beginning of each frame. The TCP and UDP checksum calculations are put into the receive descriptor (and are compared with the frame checksum for non-IP fragmented frames), even for frames over 9 KB.

Byte based bandwidth distribution among transmit queues by a weighted-round-robin arbitration mechanism is programmable, to provide QoS among the queues. This includes programming of hybrid strict and round-robin priorities among the transmit queues. The maximum byte based bandwidth allocation per transmit queue is also programmable. An Egress Jitter Pacing (EJP) arbitration mechanism is supported, to comply with IEEE 802.1Qav pre-draft, for AVB (Audio Video Bridging).

Zero Padding of Short Frames

The Ethernet networking controller offers a per frame padding enable bit in the transmit descriptor. This causes the networking controller to enlarge frames shorter than 64B by appending zero bytes. When this feature is used, only frames equal or larger than 64B are transmitted as is. Frames smaller than 64B are zero padded and transmitted as 64B packets.

Note

When using the Marvell® Header mode, DSA tag, or Extended DSA Tag, the networking controller performs zero padding to packet sizes of 66, 68, or 72B, respectively. This guarantees that the packet is still legal after the destination MAC strips the extra bytes of Marvell Header or DSA tag.

15.3.1 Address Decoding

The Mbus port that serves the networking controller has six address windows. Each address window can be individually configured.

With the networking controller DMA transaction (buffer read/write, descriptor read/write), the address is compared against the address decoding registers. Address comparison is done to select the correct target interface and attributes.

Four of the address windows have an upper 32-bit address register, for 64-bit addressing support. These are used for accessing interfaces that support more than 4-GB address space. The address generated on the interface is composed of the 32-bit address issued by the DMA (if it hits the relevant address window) concatenated with the High Address Remap register content.

To ensure that the networking controller DMA avoids accessing a forbidden address space (due to a programming bug), the Mbus port uses access protection logic that prevents it from read/write accesses to specific address windows.

If the address does not match any of the address windows, or if it violates the access protection settings, an interrupt is generated. The transaction is executed but not to the original address. Instead, the transaction is executed to a default address and target as specified in the Default Address (see Ethernet Unit Default Address (EUDA) Register (i=0–3) (Table 577 p. 923)) and ID registers (see Ethernet Unit Default ID (EUDID) Register (i=0–3) (Table 578 p. 923)).

15.3.2 Endianess and Swap Modes

Each DMA channel has configurable behavior on Little or Big Endian support, per DMA channel data receive (including virtual address read from the buffer) and data transmission. (see the <BLMR> field and the <BLMT> field in the SDMA Configuration (SDC) Register (i=0–3) (Table 585 p. 929).

For every DMA channel, descriptor accesses may be swapped (see the <SwapMode> field in the SDMA Configuration (SDC) Register (i=0–3) (Table 585 p. 929).

15.3.3 Hardware Parser Overview

The basic parser mode is defined by setting the <AccelerationMode> field in the Port Acceleration Mode (PACC) Register (i=0–3) (Table 586 p. 930).

Each networking controller includes in its receive path, a Layer-2 Destination Address (MAC-DA) recognition and filtering mechanism of up to 16-Unicast MAC addresses, 256 IP Multicast addresses, and 256 Multicast/Broadcast address. The networking controller may also detect Layer-2 frame-type encapsulation, as well as common Layer 3 and Layer 4 protocols.

Queue classification on received traffic is assigned to the DMA queue based upon a configurable analysis, which evaluates the DA-MAC, IP TOS (Type of Service), IEEE 802.1p priority tag, and protocol (ARP, TCP, or UDP). An example for use of this feature is the implementation of differentiated services or real-time, jitter-sensitive voice/video traffic intermixed with data traffic. As each queue has its own buffering, blocking is avoided and latency is reduced for CPU service.

Frame type/encapsulation detection is available on:

Layer 2 for:

- Bridge Protocol Data Unit (BPDU)
- VLAN (programmable VLAN-EtherType)
- Ethernet v2, LLC/SNAP

Layer 3 for:

- IPv4
- IPv6

Layer 4 (over IPv4/IPv6) for:

- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

Frame enqueueing is in accordance with DA, VLAN-802.1p, IP-TOS using the *highest* priority counts. Frame enqueueing is captured according to protocol type for TCP, UDP, ARP, or BPDU. The networking controller also supports enqueuing based on the proprietary Marvell® Header or the DSA Tag. These are useful when interfacing Marvell FE or GbE switches that utilize these features, resulting in improved routing performance, and support for cascade and stack of switches.

15.3.4 IP Checksum Generation

IPv4 checksum may be calculated during the packet DMA from memory, and it is replaced in the checksum field for IPv4 packets. One bit in the transmit descriptor is used for specifying if the IP checksum generation is required for a specific packet.

15.3.5 TCP Checksum Generation

The TCP checksum may be enabled per frame. When TCP checksum is enabled, it is calculated during the packet read from memory by the DMA and the calculated value is written in the checksum field before transmission begins.

Copyright © 2014 Marvell

Since a TCP segment may be transmitted over several Ethernet packets, and since the checksum in the next packets continue the checksum calculation of previous packets, there are two types of checksum generation commands (depending on the <GL4chk> field in the Tx descriptor):

- Calculate the checksum on the *first* packet in the segment:
  In this case, the 16-bit checksum field in the descriptor must be zero. The checksum is done fully by the device, and includes parsing the header according to the descriptor fields and calculating the checksum on pseudo-header. The checksum continues with full checksum calculation on the TCP data, and finally it is placed in the packet before transmission.
- Calculating checksum on *non-first* packets in the segment:
  The CPU is required to calculate the initial checksum, including the pseudo-header checksum in the <L4iChk> field in the Tx descriptor. The DMA uses this initial checksum value in calculating the TCP checksum over the TCP payload in the packet and places it in the TCP checksum field of the packet before transmission.

The CPU may choose to always calculate the checksum over the pseudo-header, and let the hardware calculate of the payload checksum.

15.3.6 UDP Checksum Generation

The UDP checksum generation is the same as the TCP checksum generation support, with both first and non-first modes (see Section 15.3.5, TCP Checksum Generation, on page 258).

15.3.7 Frame Type Indications

The receive processing of the frame (see Section 15.4, Queues Management, on page 262) allows passing various useful indications about each packet in the Rx descriptor. These indicators include MAC level errors (like Ethernet CRC check fail), to facilitate CPU processing overhead in packet header processing, and in Layer 3 and Layer 4 checksum calculations.

See the descriptor description for details. For a definition of the indications, see Section 15.4.2.4, Receive Descriptor Structure in Hardware Parsing Mode, on page 270.

15.3.8 TCP Checksum Checking

TCP frames include a 16-bit checksum that protects the entire segment payload (that usually spans over a number of packets) as well as the TCP header and some of the IP fields.

Frames may be received in an interleaved fashion from different TCP connections, and also out of order, within any TCP connection.

The Rx frame parsing allows offloading most of the overhead from the software. The Rx descriptor (see Section 15.4.2.4, Receive Descriptor Structure in Hardware Parsing Mode, on page 270) provides frame type indications such as:

- IPv4
- IPv6
- Validity of IP header with correct IPHL, IPTL, and IPv4 checksum checked OK
- Layer 2 encapsulation information (VLAN, Ethernetv2 or LLC/SNAP)
- TCP or UDP type detection.

TCP over IPv6 checksum check results are generated in the Rx descriptor. The checksum check is supported for IPv6 packets without extension headers.

TCP over IPv4 checksum check results are generated in the Rx descriptor in the following way, where two cases are identified—fragmented or non-fragmented IP packets:

1. If it is a non-fragmented IPv4 packet (the packet’s IP Header Flag <MF> = 0 and <Offset> = 0x0), meaning, the packet includes both TCP header and the full payload, (and therefore TCP checksum can be calculated), the descriptor will be closed with an indication that the frame is not fragmented. The TCP checksum calculation also takes into account the pseudo-header if

the <RxCS> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 994) is set to 1. If the <RxCS> field is set, the L4 checksum compare result will be valid (descriptor's <L4ChkOK> bit is valid).

Note

The length field for the pseudo-header is taken from the following operation:
`IPTL - IPHL * 4.`

For the checksum calculation, the value 0x0 is used instead of the checksum field in the received frame as required by the standard. In addition, the checksum calculation for each frame always starts with the initial value of 0x0.

2. If it is a fragmented IPv4 packet (either <MF> != 0 or <Offset> != 0) or if the <RxCS> field is cleared to 0 (calculation without pseudo-header), the pseudo-header is not calculated in the checksum. The checksum is calculated only on the TCP segment (header + data) and places the result in the first descriptor of each frame. Therefore, the checksum compare bit (L4ChkOK bit) is not valid.

Note

For fragmented IPv4 packets, the checksum calculation does not put a zero in the checksum field, and therefore, in a frame that is the first fragment of IP packet (<Offset> =0x0 and <MF>!= 0), the checksum result can be wrong (since it includes the checksum field of the TCP header). This should be corrected by the software.

The networking controller calculates the checksum per each packet. The software should sum all the checksum calculations for the complete IP packet and compare it to the checksum field in the TCP header.

15.3.9 UDP Checksum Checking

UDP frames include a 16-bit checksum that protects the entire segment payload (that usually spans over a number of packets) as well as the UDP header and some of the IP fields.

Frames may be received in an interleaved fashion from different UDP streams, and also out of order within any UDP stream.

The Rx frame parsing allows offloading most of the overhead from the software. The Rx descriptor provides frame type indications such as:

- IPv4 validity of IP header with correct IPHL, IPTL, and IP checksum checked OK
- IPv6
- Layer 2 encapsulation info (VLAN, Ethernetv2 or LLC/SNAP)
- TCP or UDP type detection

UDP over IPv6 checksum check results are written to the Rx descriptor. The checksum check is supported for IPv6 packets without extension headers.

UDP over IPv4 checksum results are written to the Rx descriptor, when two cases are identified:

- Non-fragmented IPv4 packet
- Fragmented IPv4 packet

Non-Fragmented IPv4 Packet

If it is a non-fragmented IPv4 packet (the packet IPv4 Header Flag <MF> = 0 and <Offset> = 0x0):

- The packet includes both the UDP header and the full payload, and therefore UDP checksum can be calculated.

- The descriptor is closed with an indication that the frame is not fragmented. (The descriptor's <IPv4_Frg> bit cleared to 0).
- The L4 checksum compare results are valid.

The UDP checksum calculation also takes into account the pseudo-header, if the <RxCS> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 994) is set to 1.

Note

The length field for the pseudo-header is taken from the following operation: `IPTL - IPHL * 4.`

For the checksum calculation, the value 0x0 is used instead of the checksum field in the received frame, as required by the standard. In addition, the checksum calculation for each frame always starts with the initial value of 0x0.

If the frame checksum is 0x0, the checksum function does not compare any values, and closes the descriptor as checksum OK. According to the standards, this is the indication that checksum function was disabled.

Fragmented IPv4 Packet

If it is a fragmented IPv4 packet (either <MF> != 0 or <Offset> != 0) or if the <RxCS> field is cleared to 0 (calculation without pseudo-header), the pseudo-header is not calculated in the checksum. The checksum is calculated only on the UDP segment (header + data), and places the result in the first descriptor of each frame. The checksum compare bit (L4ChkOK bit) is not valid.

Note

For fragmented IP packets, the checksum calculation does *not* put zero in the checksum field. In a frame that is the first fragment of IP packet (<Offset> =0x0 and <MF>!= 0), the checksum result can be wrong because it includes the checksum field of the UDP header. This issue should be corrected by the software.

The networking controller calculates the checksum per packet. The software should sum all the checksum calculations for the complete IP packet, and compare this value to the checksum field in the UDP header.

The device supports IPv4 and TCP/UDP over IPv4 checksum generation for maximum packet size of 9700B.

15.4 Queues Management

15.4.1 Transmit DMA

15.4.1.1 Descriptors Management by Counters

The descriptors of each queue are located in successive locations in a preassigned external memory space (for example, DRAM).

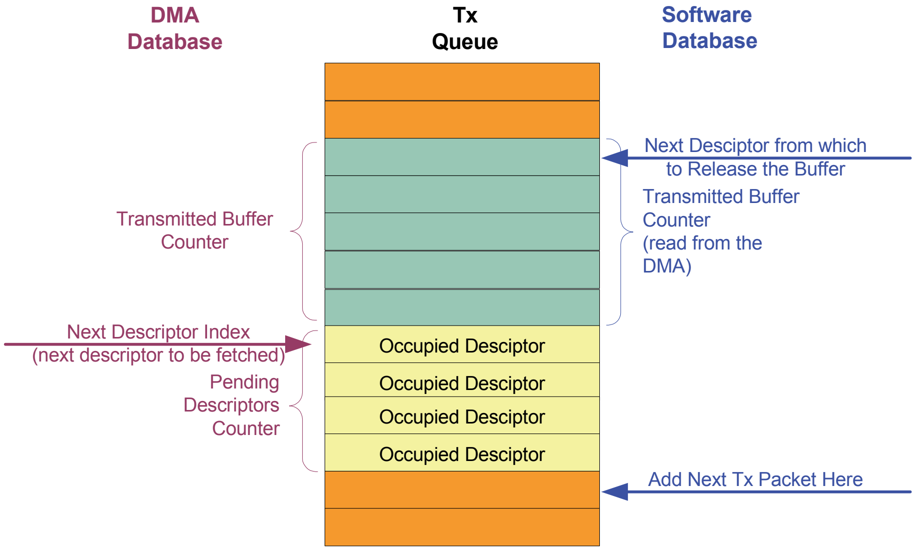
The Tx descriptors are 32B aligned.

Figure 60 provides a block diagram of the transmit queue management counters.

The following parameters are maintained for each queue:

- Queue starting address, configured by the <DescriptorsQueueStartingAddress> field in the Port TX Queues Descriptors Queue Address (PTXDQA)<q> Register (i=0–3, q=0–7) (Table 643 p. 968).
- Queue size, configured by the Port TX Queues Descriptors Queue Size (PTXDQS)<q> Register (i=0–3, q=0–7) (Table 644 p. 968).

Figure 60: Descriptors Management by Counters Transmit Queue



The networking controller maintains the following counters.

Pending Descriptors Counter

Number of descriptors pointing to the buffers that contain Tx packets yet to be processed by the networking controller. This counter can be read in the <PendingDescriptorsCounter> field in the Port TX Queues Status (PTXS)<q> Register (i=0–3, q=0–7) (Table 645 p. 970).

Following the preparation of new descriptors, the sender (for example, the CPU) increments this counter by accessing the <NoOfWrittenDescriptors> field in the Port Tx Queues Status Update (PTXSU)<q> Register (i=0–3, q=0–7) (Table 646 p. 970).

Copyright © 2014 Marvell

The networking controller decrements this counter after closing the last descriptor in packet. The decrement is by the number of descriptors occupying the packet.

Transmitted Buffers Counter

Number of buffers that were transmitted, but not yet released by the software. This counter can be read in the <TransmittedBuffersCounter> field in the TX Transmitted Buffers Counter (TXTBC)<q> Register (i=0–3, q=0–7) (Table 648 p. 971).

The networking controller increments this counter by the number of descriptors used for specific packet (one or more), after writing the last descriptor associated with the packet.

The CPU decrements this counter by writing to the <NoOfReleasedBuffers> field in the Port Tx Queues Status Update (PTXSU)<q> Register (i=0–3, q=0–7) (Table 646 p. 970).

Next Descriptor Index

Index of the next descriptor to be fetched. This counter can be read in the <NextDescriptorIndex> field in the Port TX Queues Descriptor Index (PTXDI)<q> Register (i=0–3, q=0–7) (Table 647 p. 971). This field is implemented as a cyclical counter; it counts from 0 to <DescriptorsQueueSize> minus 1.

If there are no descriptors remaining in the queue, as indicated by the <PendingDescriptorsCounter> field in the Port TX Queues Status (PTXS)<q> Register (i=0–3, q=0–7) (Table 645 p. 970), further packet transmission from this queue is disabled. When addition of new descriptors to the queue is indicated by the CPU, by writing to the <NoOfWrittenDescriptors> field in the Port Tx Queues Status Update (PTXSU)<q> Register (i=0–3, q=0–7) (Table 646 p. 970), the packets transmission is resumed.

Note

If it is required to start transmission from the first descriptor in the queue, in all queues, the CPU must set the <TxDMAInit> field in the Port TX Initialization (PTXINIT) Register (i=0–3) (Table 649 p. 972).

15.4.1.2 Tx Data Read from Buffers

The packet data is read from the buffers starting from the offset defined in the descriptor's <PacketOffset> field.

This is done for all of the buffers, not just the first buffer.

15.4.1.3 Transmit Descriptor Structure

Figure 61 shows the Transmit Descriptor structure.

Figure 61: Transmit Descriptor Description

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th><th>Offset</th></tr>
<tr><th colspan="8">Byte3</th><th colspan="8">Byte2</th><th colspan="8">Byte1</th><th colspan="8">Byte0</th><th></th></tr>
<tr><td colspan="2">GL4chk [1:0]</td><td colspan="7">Packet_offset[6:0]</td><td>buf_mode</td><td>F</td><td>L</td><td>Padding</td><td>GIPchk</td><td>L3Type</td><td>L4Type</td><td></td><td colspan="2">Pool_ind [1:0]</td><td colspan="5">IP_HdLen [6:2]</td><td>Reserved</td><td colspan="7">L3_offset[6:0]</td><td>+0</td></tr>
<tr><td colspan="16">Byte Count[15:0]</td><td colspan="16">L4iChk[15:0]</td><td>+4</td></tr>
<tr><td colspan="32">Buffer Pointer (physical address) [31:0]</td><td>+8</td></tr>
<tr><td colspan="2">DSA_Tag [1:0]</td><td colspan="10"></td><td colspan="12">Reserved</td><td colspan="4">MH_select [3:0]</td><td>Reserved</td><td colspan="2">EC</td><td>ES</td><td>+C</td></tr>
<tr><td colspan="32">Reserved</td><td>+10</td></tr>
<tr><td colspan="32">Reserved</td><td>+14</td></tr>
<tr><td colspan="32">Reserved</td><td>+18</td></tr>
<tr><td colspan="32">Reserved</td><td>+1C</td></tr>
</table>

Table 55 through Table 62 lists the transmit descriptor fields, based on the offset in use.

Table 55: Transmit Descriptor—Offset 0

Bits	Name	Description
6:0	L3_offset	Layer 3 Offset The offset of Layer 3 header beginning (in bytes), from the beginning of the packet. It is used for checksum calculation. **NOTE:** This field is valid only for the first descriptor of a packet (F==1).
7	Reserved	Reserved

Copyright © 2014 Marvell

Table 55: Transmit Descriptor—Offset 0 (Continued)

Bits	Name	Description
12:8	IP_HdLen	IP Header Length Provides the length in long words (4B) of the IP header. Supports up to 32 long words =128B. **NOTE:** This is only valid if the F bit [21] is set, and either L4 or IPV4 checksum generation are required by <GL4chk> field (bits [31:30]) and <GIPchk> field (bit 18).
14:13	Pool_ind	Pool Index This field is only valid if buf_mod field in the descriptor = 1 (buffer released by the BM unit), and the BM unit is used, as selected by the acc_mode of the PxACC register. It contains the pool index, to which the buffer should be released.
16	L4Type	Layer 4 Type 0 = TCP 1 = UDP When Layer 4 checksum generation is required, according to <GL4chk> field, then this bit indicates which Layer 4 protocol is carried in the frame. **NOTE:** This is only valid if the <F> bit[21] is set.
17	L3Type	Layer 3 Type 0 = IPv4 1 = IPv6 **NOTE:** This bit is only valid if either IPv4 header checksum or L4 checksum are to be generated (according to <GIPchk> and <GL4chk> fields.
18	GIPchk	Generate IPv4 checksum. 1 = Generate IPv4 header checksum 0 = Do not generate IPv4 header checksum **NOTE:** This is only valid if the <F> bit[21] is set and L3Type indicates IPV4 packet.
19	Padding	Padding When this bit is set and the packet is smaller than 60B, zero-value bytes are appended to the packet. Use this feature to prevent transmission of fragments. **NOTE:** This is only valid if <F> bit[21] is set. If disabling hardware CRC generation (i.e. CRC is generated by the software), zero padding must be disabled.
20	L	Last When equal to 0x1, indicates the last buffer used for the frame.
21	F	First When equal to 0x1, indicates the first buffer used for the frame.
22	Buf_mod	Buffer Release Mode 0 = The buffer should be release by software. 1 = The buffer should be released by the networking controller (to the Buffer Management unit). **NOTE:** This bit is valid for all descriptors of a packet (not only for the first).
29:23	Packet_offset	Packet Offset The offset of the packet (in bytes) from the beginning of the buffer. **NOTE:** This offset is valid for all buffers (not only for the first).

Table 55: Transmit Descriptor—Offset 0 (Continued)

Bits	Name	Description
31:30	GL4chk	Generate Layer 4 (TCP/UDP) Checksum 0 = Generate L4 checksum. Packet is fragmented and this is the first fragment. The CPU provides in the <L4iChk> field of the descriptor the checksum calculated over the pseudo header of the first fragment and over the TCP/UDP payload of the second to last fragments. The networking controller calculates the checksum over TCP/UDP header + L4 payload (starting from L3 offset + L3 header length * 4), adds to that the initial checksum value supplied by <L4iChk> field in Tx descriptor and inserts the calculated checksum to relevant place in UDP/TCP header. 1 = Generate L4 checksum. Packet is not fragmented. The networking controller Generates L4 checksum as required by UDP/TCP standard over IPv4/IPv6 pseudo header + TCP/UDP header + L4 payload, and inserts checksum to the UDP/TCP header. 2 = Do not generate L4 checksum 3 = Reserved The payload length over which the checksum is calculated is determined as: **For IPV4 packets:** “Total Length” field in the IPV4 header - 4*IP_HdLen(5b) field in the descriptor. **For IPV6 packets:** The “Payload Length” field in the IPV6 header. **NOTE:** This field is only valid if the <F> bit[21] is set.

Table 56: Transmit Descriptor—Offset 4

Bits	Name	Description
15:0	L4iChk	Layer 4 Initial Checksum In this field, the CPU provides the initial checksum value calculated on the pseudo header and the payload of the second to last fragments of a fragmented TCP/UDP segment. It is valid only for the first fragment of a fragmented TCP/UDP segment (indicated by <GL4ichk> == 0). **NOTE:** Only valid if the <F> bit[21] is set.
31:16	Byte Count	Byte Count Number of bytes to be transmitted from the associated buffer. This is the payload size in bytes.

Table 57: Transmit Descriptor—Offset 8

Bits	Name	Description
31:0	Buffer_physical_pointer	A 32-bit pointer to the beginning of the buffer associated with this descriptor.

Copyright © 2014 Marvell

Table 58: Transmit Descriptor—Offset C

Bits	Name	Description
0	ES	Error Summary This field is written by the networking controller. Error Summary of MAC level errors on frame transmission. 0 = No Error 1 = Error occurred (Late Collision, or Retransmit Limit, or Underrun Error) **NOTE:** This field is only valid if <L> bit[20] is set.
2:1	EC	MAC Error Coding This field is written by the networking controller. 00 = LC (Late Collision) 01 = UR (Underrun) 10 = RL (retransmit limit) reached (excessive collision) 11 = Reserved **NOTE:** Valid only if <L> bit[20] is set and <ES> bit[0] is set.
3	Reserved	Reserved
7:4	MH_ select	Marvell Header select. This field defines how the generated Tx packet header (the first 2B of the packet) is created: 0 = No change of the header. packet is transmitted with same header as in the buffer (may be used when the packet is received from CPU). 1 = Replace the header with the value of TX_MH_reg_1 configuration field. 2 = Replace the packet header with the value TX_MH_reg_2 configuration contents. 3 = Replace the packet header with the value TX_MH_reg_3 configuration contents. 4 = Replace the packet header with the value TX_MH_reg_4 configuration contents. 5 = Replace the packet header with the value TX_MH_reg_5 configuration contents. 6 = Replace the packet header with the value TX_MH_reg_6 configuration contents. 7 = Replace the packet header with the value TX_MH_reg_7 configuration contents. 8 = Replace the packet header with the value TX_MH_reg_8 configuration contents. 9 = Replace the packet header with the value TX_MH_reg_9 configuration contents. 10 = Replace the packet header with the value TX_MH_reg_10 configuration contents. 11 = Replace the packet header with the value TX_MH_reg_11 configuration contents. 12 = Replace the packet header with the value TX_MH_reg_12 configuration contents. 13 = Replace the packet header with the value TX_MH_reg_13 configuration contents. 14 = Replace the packet header with the value TX_MH_reg_14 configuration contents. 15 = Replace the packet header with the value TX_MH_reg_15 configuration contents. •The packet header change is performed only if Marvell Header is enabled, by the MHEn bit in the legacy Marvell Header register.
19:8	Reserved	Reserved
31:30	DSA_Tag	DSA Tag Indicates the type of DSA tag in the packet: 0 - no DSA tag 1 - 1 word DSA tag 2 - 2 words DSA tag 3 - Reserved

Copyright © 2014 Marvell

Table 59: Transmit Descriptor—Offset 10

Bits	Name	Description
31:0	Reserved	Reserved

Table 60: Transmit Descriptor—Offset 14

Bits	Name	Description
31:0	Reserved	Reserved

Table 61: Transmit Descriptor—Offset 18

Bits	Name	Description
31:0	Reserved	Reserved

Table 62: Transmit Descriptor—Offset 1C

Bits	Name	Description
31:0	Reserved	Reserved

15.4.2 Receive DMA

15.4.2.1 Queues Management by Counters

The descriptors for each queue are located in successive locations in a pre-assigned descriptor queue.

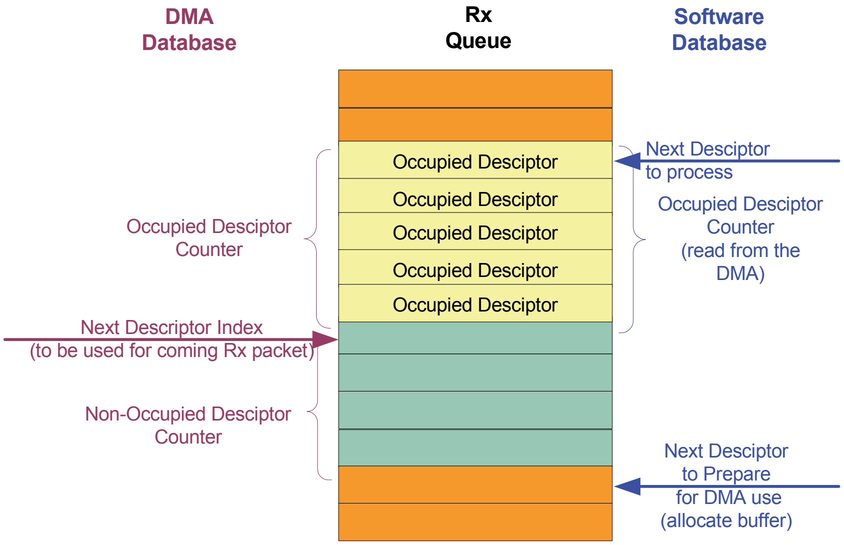
The queue starting address is configured by the <DescriptorsQueueStartingAddress> field in the Port RX Queues Descriptors Queue Address (PRXDQA) <q> Register (i=0–3, q=0–7) (Table 612 p. 948).

The queue size is configured by the <DescriptorsQueueSize> field in the Port RX Queues Descriptors Queue Size (PRXDQS)<q> Register (i=0–3, q=0–7) (Table 613 p. 948).

Figure 62 provides a block diagram of the receive queue management counters.

Copyright © 2014 Marvell

Figure 62: Descriptors Management by Counters Receive Queue



The networking controller maintains the following counters for each Rx queue.

Occupied Descriptors Counter

This counter indicates the number of descriptors pointing to the buffers that contain Rx packets, and are not yet processed by the CPU. It can be read in the <OccupiedDescriptorsCounter> field in the Port RX Queues Status (PRXS)<q> Register (i=0–3, q=0–7) (Table 615 p. 950).

The networking controller increments this counter following a packet data transfer to the buffer.

The CPU decrements this counter by writing to the <NoOfProcessedDescriptors> field in the Port RX Queues Status Update (PRXSU)<q> Register (i=0–3, q=0–7) (Table 616 p. 951).

Non-Occupied Descriptors Counter

This counter indicates the number of descriptors that are received from the CPU and are unused by the networking controller. It is read in the <NonOccupiedDescriptorsCounter> field in the Port RX Queues Status (PRXS)<q> Register (i=0–3, q=0–7) (Table 615 p. 950).

The CPU increments this counter by writing to the <NoOfNewDescriptors> field in the Port RX Queues Status Update (PRXSU)<q> Register (i=0–3, q=0–7) (Table 616 p. 951).

The networking controller decrements this counter after writing to the descriptor.

Next Descriptor Index

This counter indicates the index of the next descriptor to be fetched. It is read in the <NextDescriptorIndex> field in the Port RX Descriptor Index (PRXDI)<q> Register (i=0–3, q=0–7) (Table 617 p. 951).

This is a cyclical counter that counts from 0 to <DescriptorsQueueSize>DescriptorsQueueSize, minus1.

Note If it is required to start reception from the first descriptor in the queue, for all queues, the CPU must set the <RXDMAInit> field in the Port RX Initialization (PRXINIT) Register (i=0–3) (Table 623 p. 954).

15.4.2.2 Rx Data Transfer to Buffers

When the received packet is written to the Rx buffer the data is written as follows:

- For the first buffer:
  - The packet is written starting from the <PacketOffset> field in the Port RX Queues Configuration (PRXC)<q> Register (i=0–3, q=0–7) (Table 609 p. 945) plus 2B.
  - If a Marvell® Header (2B length) exists, it is written to the buffer starting from <PacketOffset>. Otherwise, 2B of zeros are written instead of the Marvell Header.
  - The offset configured is according to the <PacketOffset>.
- For the second to last buffer:
  - If the buffer is allocated by the software, the packet is written from the first byte of the packet.
  - If the buffer is allocated by the BM unit, the packet is written to the buffer, starting from an offset of 8B. This is done so the virtual buffer address is not overwritten. The virtual buffer address is written in the first 4B of the buffer.

15.4.2.3 Rx Buffer Virtual Pointer

Note Available when Buffer Management unit is enabled.

Besides the physical address, each buffer has a virtual address that is used by the CPU to access the buffer.

The CPU must store each buffer's virtual address in the first Dword (4B) of the buffer.

During the DMA, the networking controller writes the packet data with an offset, as described in Section 15.4.2.2, so that the virtual address is not overwritten.

When writing to the descriptor, the networking controller copies the virtual address from the buffer's first Dword to the buffer_virtual_pointer field of the descriptor (see Table 63, Receive Descriptor Description, on page 271).

15.4.2.4 Receive Descriptor Structure in Hardware Parsing Mode

This section describes the receive descriptor structure, in hardware parsing mode, as defined by the <AccelerationMode> field in the Port Acceleration Mode (PACC) Register (i=0–3) (Table 586 p. 930)—when it is set to 0x1 or 0x2.

- The descriptor length is 32B, and it must be 32B aligned (that is, Descriptor_Address[4:0] ==00000).
- Descriptors can reside anywhere in the address space.
- Receive buffers associated with receive descriptors are limited to 64-1 KB and must be 64-bit aligned (that is, Buffer_Address[2:0]==000).
- The minimum buffer size for the receive buffer is 64B.

Figure 63 shows the receive descriptor structure.

Figure 63: Receive Descriptor Description

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>09</th><th>08</th><th>07</th><th>06</th><th>05</th><th>04</th><th>03</th><th>02</th><th>01</th><th>00</th><th>Offset</th></tr>
<tr><th colspan="8">Byte3</th><th colspan="8">Byte2</th><th colspan="8">Byte1</th><th colspan="8">Byte0</th><th></th></tr>
<tr><td>IPv4_Frg</td><td>L4ChkOK</td><td>Reserved</td><td>U</td><td>F</td><td>L</td><td>IPHeadOK</td><td>L3IP</td><td>Layer2Ev2</td><td colspan="2">Layer4[1:0]</td><td>BPDU</td><td>Vlan</td><td colspan="2">EC</td><td>ES</td><td>Reserved</td><td colspan="2">Pool_ind[1:0]</td><td colspan="13">Reserved</td><td>+0</td></tr>
<tr><td colspan="16">Byte Count[15:0]</td><td colspan="16">Reserved</td><td>+4</td></tr>
<tr><td colspan="29">Buffer_physical_pointer[31:3]</td><td colspan="3">Reserved</td><td>+8</td></tr>
<tr><td colspan="32">Reserved</td><td>+C</td></tr>
<tr><td colspan="32">Buffer_virtual_pointer[31:0]</td><td>+10</td></tr>
<tr><td colspan="16">L4Chk[15:0]</td><td colspan="16">Prefetch_command[15:0]</td><td>+14</td></tr>
<tr><td colspan="31">Reserved</td><td></td><td>+18</td></tr>
<tr><td colspan="32">Reserved</td><td>+1C</td></tr>
</table>

Table 63 through Table 70 lists the receive descriptor fields, based on the offset in use.

Table 63: Receive Descriptor—Offset 0

Bits	Name	Description
12:0	Reserved	
14:13	Pool_ind	Pool index to which the buffer belongs. This field is valid only when the Buffer Management unit is enabled by the networking controller (as configured in the acc_mode field of the PxACC register), and is enabled for the queue (as configured in <HwBuffAloc> field in the PxRXyC register.
15	Reserved	Cleared to 0 by the controller.
16	ES	Error Summary 0 = No Error 1 = Error Occurred (CRC Error, Overrun Error, Maximum Frame Length Error, Resource Error) **NOTE:** This is only valid if <F> bit[27] is set.

Table 63: Receive Descriptor—Offset 0 (Continued)

Bits	Name	Description
18:17	EC	MAC Error Coding 00 = CE - CRC Error 01 = OR - Overrun Error 10 = MF - Maximum Frame Length Error. Frame is longer than the MAX_FRAME_SIZE. 11 = RE - Resource Error (No descriptors in the middle of the frame) **NOTE:** This is only valid if the <F> bit[27] and the <ES> bit[16] are set. If multiple errors occurred, then the reporting priority is: Resource Error, Maximum Frame Length Error, Overrun Error, and CRC Error.
19	VLAN	When equal to 0x1, VLAN Frame is VLAN tagged (according to programmed VLAN-EtherType). **NOTE:** This is only valid if the <F> bit[27] is set, and the <ES> bit[16] is cleared to 0.
20	BPDU	Bridge Protocol Data Unit Set to 0x1 when the frame is BPDU. **NOTE:** Only valid if the <F> bit[27] <ES> bit[16] is cleared to 0.
22:21	Layer4	Frame encapsulation and protocol. 00 = Frame is TCP over IPv4/IPv6 over Ethernet v2 or LLC/Snap (with or without VLAN tag). The Checksum result is provided in the <L4Chk> bits[18:3]. 01 = Frame is UDP over IPv4/IPv6 over Ethernet v2 or LLC/Snap (with or without VLAN tag). The Checksum result is provided in the <L4Chk> bits[18:3]. 10 = Other Frame type. 11 = Reserved **NOTE:** TCP/UDP parsing over IPv6 is supported for IPv6 header without the extension header. This is only valid if the <F> bit[27] is set, and the <ES> bit[0] is cleared to 0. This is only valid if the <IPHeadOK> bit[25] is set and <L3IP> bit[24] is set.
23	Layer2Ev2	Set if Layer 2 is Ethernet v2. **NOTE:** This is only valid if the <F> bit[27] is set, and the <ES> bit[16] is cleared to 0.
24	L3IP	• <L3IP> = 0, <IPHeadOK> = 0: Unknown L3 protocol (was not recognize as IPv4 or IPv6). • <L3IP> = 0, <IPHeadOK> = 1: L3 type is IPv6. (IP header <version> field = 6 and EtherType = 0x86DD over Ethernet v2, or over LLC/SNAP, with or without VLAN tag). • <L3IP> = 1, <IPHeadOK> = 0: L3 type is IPv4. (EtherType = 0x800 over Ethernet v2, or over LLC/SNAP, with or without VLAN tag). IP header is not OK (OK as defined below). • <L3IP> = 1, <IPHeadOK> = 1: L3 type is IPv4. (EtherType = 0x800 over Ethernet v2, or over LLC/SNAP, with or without VLAN tag). IP header is OK (OK as defined below). IPv4 header is OK if all the following occur: • IP header <version> field = 4) • IPHL >=5 • IPHL *4<= IPTL • IPv4 header Checksum is OK. **NOTE:** This is only valid if the <F> bit[27] is set, and the <ES> bit[0] is cleared to 0.
25	IPHeadOK	
26	L	Last When equal to 0x1, indicates the last buffer used for the frame.
27	F	First When equal to 0x1, indicates the first buffer used for the frame.

Table 63: Receive Descriptor—Offset 0 (Continued)

Bits	Name	Description
28	U	Unknown Destination Address The frame is Unicast and was not matched to the MAC Address Base (DA[47:4]). **NOTE:** This is only set if working in promiscuous mode. See <UPM> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 996). This is only valid if the <F> bit[27] is set, and the <ES> bit[16] is cleared to 0.
29	Reserved	Cleared to 0 by the controller.
30	L4ChkOK	Layer4 Checksum OK 1 = OK (passed). 0 = Check failed. If <Layer4> bits[22:21] is 01(UDP indication) and the received frame checksum field was 0x0, then the bit will indicate passed. This is only valid if the <IPHeadOK> bit[25] is set, and <L3IP> bit[24] is set. This is only valid if <Layer4> field is 00 or 01.(indicating TCP/UDP). This is only valid if the <F> bit[27] is set, and the <ES> bit[0] is cleared to 0. For an IPv4 packet, this is only valid if the Receive Descriptor <IPv4_Frg> field is cleared (That indicates a non-fragmented IPv4 packet). This is only valid if <RxCS> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 994) is set to 1 (indication for L4 calculation including pseudo header).
31	IPv4_Frg	IPv4 fragmented indication. 1 = Fragmented 0 = Not fragmented This is only valid if the <IPHeadOK> bit[25] and the <L3IP> bit[24] are set. This is only valid if the Receive Descriptor Command/Status Register <F> bit[27] is set, and <ES> bit[0] is cleared to 0.

Table 64: Receive Descriptor—Offset 4

Bits	Name	Description
15:0	Reserved	Reserved
31:16	Byte Count	When a descriptor is closed, this field is written by the DMA, indicating the number of bytes actually written by the DMA into the buffers. **NOTE:** This is only valid if the <F> bit[27] is set.

Copyright © 2014 Marvell

Table 65: Receive Descriptor—Offset 8

Bits	Name	Description
2:0	Reserved	Cleared to 0 by the controller.
31:3	Buffer_physical_pointer	Written by DMA or CPU The 29 MSB of a 32-bit physical pointer to the beginning of the buffer associated with this descriptor. • When the queue is configured as software buffer allocation (see <HwBuffAloc> field in the PxRXyC register), then this field is written by the CPU. • When the queue is configured as a hardware buffer allocation (see <HwBuffAloc> field in the PxRXyC register), then the Gigabit Ethernet unit writes the physical buffer pointer received from BM unit to this field. **NOTE:** The physical buffer pointer must be 64-bit aligned; therefore, bits[2:0] of the pointers are considered as zeros.

Table 66: Receive Descriptor—Offset C

Bits	Name	Description
31:0	Reserved	Reserved

Table 67: Receive Descriptor—Offset 10

Bits	Name	Description
31:0	Buffer_virtual_pointer	Buffer virtual address. This field is valid only when BM unit is used, as selected by the acc_mode field of the PxACC register and <HwBuffAloc> field in the PxRXyC register. The buffer virtual address is copied by the networking controller from the first Dword of the buffer to the descriptor.

Table 68: Receive Descriptor—Offset 14

Bits	Name	Description
15:0	Prefetch_command	Prefetch_command, to be used by the PFE (Level-2 cache prefetch engine). The PFE reads the prefetch_command from the descriptor, and according to it, executes L2 prefetch of the relevant parts of the packet. The networking controller copies one of the configurable prefetch_command profiles to this field (see Section 15.15, L2-Cache Prefetch Engine Support, on page 299). **NOTE:** This is only valid if the <F> bit is set. For the non first descriptors (<F> bit is not set), “0” will be written. (means no prefetch to be done for this buffer).

Table 68: Receive Descriptor—Offset 14 (Continued)

Bits	Name	Description
31:16	L4Chk	Calculated TCP/UDP over IPv4/IPv6 Checksum **NOTE:** This is only valid if <Layer4> bits[22:21] are set to 00 or 01 (Indicating TCP/UDP). This is only valid: - If the <F> bit[27] is set and no MAC errors occurred. - If the <IPHeadOK> bit[25] is set (indicating IPv6 header or legal IPv4 header). The calculation does not include the pseudo-header if the <RxCS> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 994) is cleared to 0 (calculation without the pseudo-header). The calculation does not include the pseudo-header if <L3IP> is set (the packet is IPv4) and the Receive Descriptor <IPv4_Frg> field is set (calculation without the pseudo-header). The field equals 0x0 if the calculated UDP value is 0xFFFF.

Table 70: Receive Descriptor—Offset 1C

Bits	Name	Description
31:0	Reserved	Reserved

15.5 Hardware Buffer Management

This section describes:

- Hardware Buffer Allocation for Ingress Packets
- Hardware Buffers Release for Egress Packets

15.5.1 Hardware Buffer Allocation for Ingress Packets

When the hardware buffer management is enabled by the <AccelerationMode> field in the Port Acceleration Mode (PACC) Register (i=0–3) (Table 586 p. 930), the networking controller supports hardware buffer allocation, for Rx queues that have the <HwBuffAlloc> field in the Port RX Queues Configuration (PRXC)<q> Register (i=0–3, q=0–7) (Table 609 p. 946) set to 1.

The size of the buffers of each pool is configured by the CPU via the configuration registers:

- Port Pool0 Buffer Size (PPL0BSZ) Register (i=0–3) (Table 618 p. 952)
- Port Pool1 Buffer Size (PPL1BSZ) Register (i=0–3) (Table 619 p. 952)
- Port Pool2 Buffer Size (PPL2BSZ) Register (i=0–3) (Table 620 p. 952)
- Port Pool3 Buffer Size (PPL3BSZ) Register (i=0–3) (Table 621 p. 953)

Each Rx queue utilizes only two of the four pools according to the <PoolIdShort> field and the <PoolIdLong> field in the Port RX Queues Configuration (PRXC)<q> Register (i=0–3, q=0–7) (Table 609 p. 945).

For each received packet, the pool to be used is selected according to the packet size as following:

- Buffers from pools indicated by the <PoolIdShort> field are used for packets that can be stored in a single buffer of this pool.
- Buffers from pools indicated by <PoolIdLong> field are used for packets that cannot be stored in a single buffer of the Pool ID Short pool.

Note

- The buffer size of the short pools must be at least (64B + packet offset), and no more than 4800B.
- The buffer size of the long pools should be at least the buffer size of the short pools.

Each buffer allocated by the BM unit contains in the first Dword the virtual address of the buffer.

Therefore, following reception of physical buffer pointer from the BM unit:

1. The virtual address must be read from the buffer and kept in temporary storage.
2. When the descriptor associated with this buffer is closed, the virtual address of the buffer is written by the networking controller to the Buffer_virtual_pointer field of the descriptor (see Figure 61, Transmit Descriptor Description, on page 264).
3. In addition, the networking controller writes the pool number from which the buffer was taken to the <PoolInd> field of the descriptor.

Note

The used buffers are released later by the CPU or the networking controller Tx DMA to the pool indicated by the <PoolInd> field of the descriptor.

15.5.2 Hardware Buffers Release for Egress Packets

When the hardware buffer management is enabled by the <AccelerationMode> field in the Port Acceleration Mode (PACC) Register (i=0–3) (Table 586 p. 930), the networking controller will support the hardware buffer release for buffers that are attached to descriptors that have the <buf_mode> bit set to 1.

For these buffers the release procedure is:

1. Wait until all the data is read from the buffer to the Tx FIFO.
2. Release the buffer (by proper access to the BM unit) to the pool indicated by the <pool_ind> field of the descriptor.

15.6 Receive Frame Processing—Hardware Parser Mode

This section describes the receive frame processing, when working in hardware parser mode.

The hardware parser mode is defined by setting the <AccelerationMode> field in the Port Acceleration Mode (PACC) Register (i=0–3) (Table 586 p. 930) to one of the following values: 0, 1 or 2.

Once a frame is received by the networking interface, it is parsed using the following process:

- MAC errors checking
- Accept or reject decision
- Select the Receive queue (0 through 7)
- MIB counter increments
- Extract Layer 2/3/4 protocols and perform IP and/or TCP/UDP checksum

Some MAC level errors, such as fragments, are normally filtered from the frame reception and are only counted in MIB counters. Other MAC level errors (such as CRC error frames and frames beyond the maximum allowed size) are reported in the first descriptor and in the MIB counters.

The frames maximum size is defined in the <FrameSizeLimit> field in the Port MAC Control Register0 (i=0–3) (Table 702 p. 1008).

The receiver accepts Jumbo frames when the <FrameSizeLimit> field is configured accordingly, where the IEEE 802.3 Type/Length field is set to 0x8870 (with or without IEEE 802.1Q VLAN tag).

Note

The MAC Destination address bit[40] is the Multicast/Unicast bit. When GMII interface is used, the first DA byte received on the GMII RXD[7:0] pins is DA[40:47]. The last byte GMII received on the RXD[7:0] pins is DA[0:7]).

15.6.1 Parsing the Frames

This section describes the filtering and enqueuing of frames.

15.6.1.1 Filtering

Note

The Marvell Header and DSA also control the filtering process (see Section 15.7, Marvell® Header Support, on page 279 and Section 15.8, Distributed Switching Architecture (DSA) Tag Support, on page 282).

The frame goes through the following stages to determine if it is accepted:

1. If the frame is a flow control pause packet, it is processed by the flow control mechanism and then discarded (see the Section 17.10.1, Pause Receive Operation, on page 314).
2. If the frame is in the Bridge Protocol Data Unit (BPDU) format (DA is equal to
01-80-C2-00-00-00 through
01-80-C2-00-00-FF, except for the Flow-Control Pause packets), frame is accepted/rejected according to the <Span> field in the Port Configuration Extend (PxCX) Register (i=0–3) (Table 691 p. 997).
3. If the frame is Unicast then the MAC DA bits[47:4] are compared with MAC[47:4] (see the MAC Address Low (MACAL) Register (i=0–3) (Table 590 p. 931) and MAC Address High (MACAH) Register (i=0–3) (Table 591 p. 932)).
   - If they do not match, the frame is accepted/rejected according to <UPM> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 996) (Unicast Promiscuous Mode).
   - If they match, then the MAC DA[3:0] bits are used as a pointer to the Unicast Table entries in the DA-Filter table. The frame is accepted/rejected according to the appropriate <Pass> bit in the Destination Address Filter Unicast Table (DFUT) Register (n=0–3).
   - If <ForceUnicstHit> is set, then any Unicast frame is handled as the frame DA bits[47:4] match the MACAL/MACAH registers.
4. If DA=0xFFFFFFFF and the protocol is 0x806 (an ARP broadcast) in Ethernet-v2 (tagged or not), frame is accepted/rejected according to the <RBArp> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 995).
5. If DA=0xFFFFFFFF and the protocol is 0x800 (an IP broadcast), in Ethernet-v2 or LLC/SNAP (tagged or not), the frame is accepted/rejected according to the <RBIP> in the same register.
6. If DA=0xFFFFFFFF and it is not an ARP nor an IP packet (another type of broadcast), the frame is accepted/rejected according to the <RB> in the same register.

7. If DA=0x01-00-5E-00-00-XX (an IP multicast; XX is between 0x00 and 0xFF) the MAC DA[7:0] bits are used as a pointer to the Special Multicast Table entries in the DA-Filter table. Frame is accepted/rejected according to the <Pass> bit.
8. If DA bit[0] is 1 and it is not a broadcast nor an IP multicast (the frame is a Multicast of another type), a 8-bit polynomial CRC is calculated (x^8+x^2+x^1+1) and the result is used as an index to the Multicast Broadcast Table entries in the DA-Filter table. The frame is accepted/rejected according to the <Pass> bit value.

15.6.1.2 Enqueuing

Note

The Marvell Header and DSA also control the enqueuing process (see Section 15.7, Marvell® Header Support, on page 279 and Section 15.8, Distributed Switching Architecture (DSA) Tag Support, on page 282).

If the frame is accepted, the receive queue is determined according to the following flow:

1. If it is a BPDU packet, the Rx queue is determined by <BPDUQ> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 994) (by default, it is the highest priority queue - 7).
2. If it is an ARP broadcast packet, the Rx queue is determined by the <RXQArp> field in the same register.
3. If it is a Unicast packet that failed Unicast address compare (but the packet is still accepted because of Promiscuous Mode), the Rx queue is determined by the <RXQ> field in the same register (that is the default receive queue).

If none of the above three conditions is met, Rx queue selection proceeds as follows:

1. If it is a TCP packet, the Rx queue is determined according to <TCPQ>.
2. If it is a UDP packet, the Rx queue is determined according to <UDPQ>.

If it is not either a TCP or a UDP packet, the Rx queue selection proceeds as follows:

1. Calculate the Rx queue as follows:
   - If it is a Unicast packet that passed address compare, MAC DA[3:0] bits are used as a pointer to the Unicast Table entries in the DA-Filter table. The table's <Queue> bits determines Rx queue number.
   - If it is an IP Multicast packet, MAC DA[7:0] bits are used as a pointer to the Special Multicast Table entries in the DA-Filter table. The table's <Queue> bits determine the Rx queue number.
   - If it is another type of Multicast packet, an 8-bit CRC value is used as an index to the Multicast Broadcast Table entries in the DA-Filter table. The table's <Queue> bits determine the Rx queue number.
   - If it is a Broadcast packet (IP or other), the Rx queue is determined by <RXQ> field in the Port Configuration (PxC) Register (i=0–3) (Table 690 p. 996).
2. If it is a VLAN tagged packet, the Rx queue is determined by VLAN Priority Tag to Priority (VPT2P) Register (i=0–3) (Table 599 p. 936).
3. If it is an IPv4 or IPv6 packet (according to EtherType, as defined in the RX descriptor <L3IP> field), the DSCP field (6 bits) is mapped to the Rx queue number.
   To configure the DSCP-to-RX queue, set the IP Differentiated Services CodePoint 0 to Priority (DSCP0) Register (i=0–3) (Table 592 p. 932) and the other <DSCP1> to <DSCP6> registers.

Copyright © 2014 Marvell

4. Since a packet can match any of the above three conditions (steps 1, 2, and 3 of this list) (for example, a Unicast VLAN tagged IPv4 packet), the Rx queue number is determined as the *highest* queue from the one determined by the above 3 conditions.

Note

- To not use VLAN or DSCP mapping. Program those registers as all zero values, which are their default values.
- If the packet has an unknown L3 type, it is still accepted and queued based on L2 MAC address (and VLAN priority, if it is also VLAN tagged).
- Before enabling the port, the DA-Filter tables must be programmed in full, as their initial values is undefined.
- The Receive queue can be configured to be according to the Marvell Header or the DSA tag. In that configuration, the receive queue defined by the Marvell Header/DSA tag overrides the above enqueuing decision.

5. If the frame EtherType field matches the <EtherTypePriVal> field in the Ethernet Type Priority Register (i=0–3) (Table 600 p. 936), then:
   - If the <EtherTypePriFrstEn> field is set, then the queuing decision is according to the <EtherTypePriQ> field.
   - If the <EtherTypePriFrstEn> field is cleared, and the <EtherTypePriEn> field is set, then the queuing decision is according to the highest priority of the <EtherTypePriQ> field and the step 4 queuing decision.

15.7 Marvell® Header Support

When interfacing an external Marvell switch, Marvell recommends enabling the Marvell Header mode in the switch and in the ethernet networking controller. The Marvell Header consists of 2 octets placed ahead of the DA. The Marvell Header contains information used by the software to determine from which of the switch ports the packet arrived and to maintain QoS.

To enable Marvell Header mode, set the <MHEn> field in the Marvell Header Register (i=0–3) (Table 692 p. 999) to 1.

Note

- See the Marvell SOHO switches’ specification for additional information about the Marvell Header mode.
- When the Marvell Header is enabled, the Ethernet networking controller does not add 2B when a packet is received. The packet received from the switch is padded with the 2-octet Marvell Header.

15.7.1 Receive Operation

When the Marvell Header mode enabled in the switch, it pads each packet with two octets just before the DA as shown in Figure 64.

Figure 64: Rx Packet Marvell Header Example

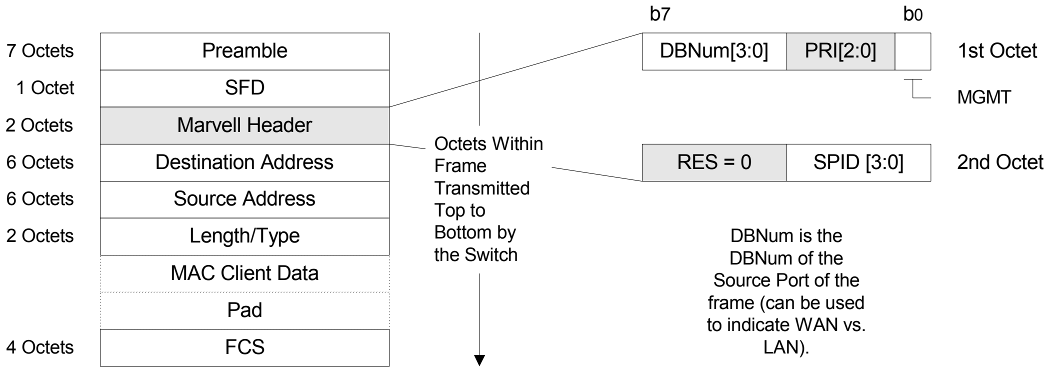


Table 71 details the Marvell header fields.

Table 71: Marvell Header Fields

Name	Description
DBNum[3:0]	Database Number Represents the address database assigned to this frame, when it ingresses into the switch (assigned by the switch source port). Used to indicate the VLAN number of the source port.
PRI[2:0]	Frame priority The priority assignment details are in the switch specification.
RES	Reserved for future use
SPID[3:0]	Source Port ID Indicates the physical port on the frame that the switch entered.
MGMT	Management For details on the MGMT bit assignment, see the switch specification.

Note

- The Marvell Header is transferred along with the rest of the packet to memory. Its fields are not updated in the Rx descriptor status.
- Packet CRC generation/checking includes the two Marvell Header octets.

The Ethernet networking controller also supports Rx queuing, based on Marvell Header fields. This is an alternative to the regular queuing described in Section 15.6.1, Parsing the Frames, on page 277.

The <DAPrefix> field in the Marvell Header Register (i=0–3) (Table 692 p. 999) selects the queuing policy:

- 0x0: The regular priority queuing is working.
- 0x1: The Rx queue is determined according to PRI[2:0] bits.
- 0x2: The Rx queue is determined according to DBNum[0] and PRI[2:1].
- 0x3: The Rx queue is determined according to SPID[3:0] and PRI[2:1].

For example, if the application requires two priority queues (Low and High) for Switch Port2, and two priority queues (Low and High) for the rest of the switch ports, set the following fields in the Marvell Header Register (i=0–3) (Table 692 p. 997):

- <DA_PREFIX> to 0x3
- <SPID> to 0x2
- <MH_Mask> to 0x1 (indicating to use queues 0–3)

With this configuration, packets received from Switch Port2 are placed in queues 2 (L) and 3 (H), and packets received from the other switch ports are placed in queues 0 (L) and 1 (H).

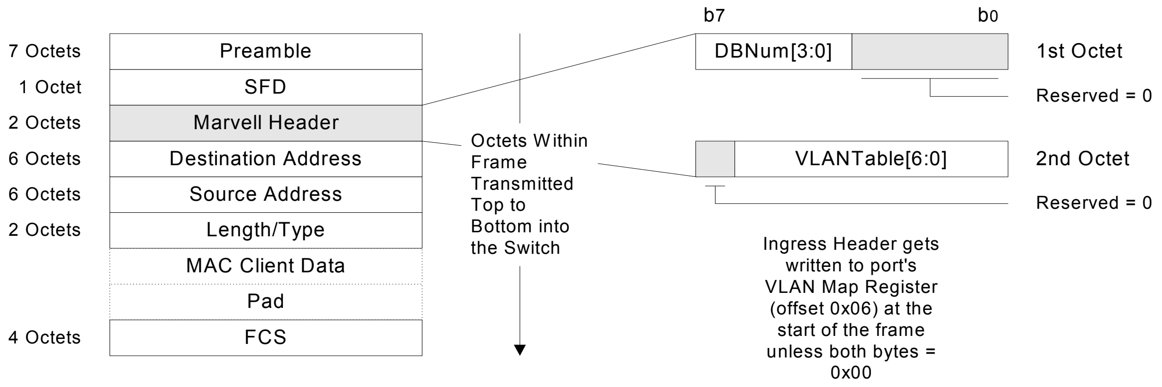
Note

- Packets accept/reject is handled the same way as in none Marvell Header mode as described in Section 15.6.1, Parsing the Frames, on page 277 with one exception. If the packet is marked as MGMT, it is always accepted.
- Receive MIB counters do not take the Marvell Header into account. For example, if receiving a packet that has a 128B size (including the Marvell Header), frames 128–255 Octets counter are incremented, instead of frames 64–127 Octets counter.

15.7.2 Transmit Operation

With the Marvell Header enabled, the software must add two octets of the Marvell Header to the top of the packet, as shown in Figure 65.

Figure 65: Tx Packet with Marvell Header Example



The Marvell Header gives the software the ability to control which VLAN and address database to use for the frame.

Note

See the specific SOHO switch specification for details on the Marvell Header format and usage.

The Ethernet networking controller generates CRC for the entire packet, including the Marvell Header. When the switch receives the packet, it strips the Marvell Header, and recalculates the new CRC before forwarding the frame to the network.

15.8 Distributed Switching Architecture (DSA) Tag Support

Most Marvell GbE switches support the Distributed Switching Architecture (DSA) feature. This feature is useful for cascading and stacking switch devices. Even when using a single GbE switch device, the DSA tag is useful for the software to determine from which of the switch ports the packet arrived and to maintain QoS. This is similar to using the Marvell Header mode, but DSA provides additional information on the received packet.

When interfacing one of Marvell Gigabit Ethernet switches, Marvell recommends enabling the Marvell DSA Tag mode, both in the switch and the Ethernet networking controller.

To enable Marvell the DSA feature, set the <DSAEn> field in the Marvell Header Register (i=0–3) (Table 692 p. 998) to 1 or 2.

Note

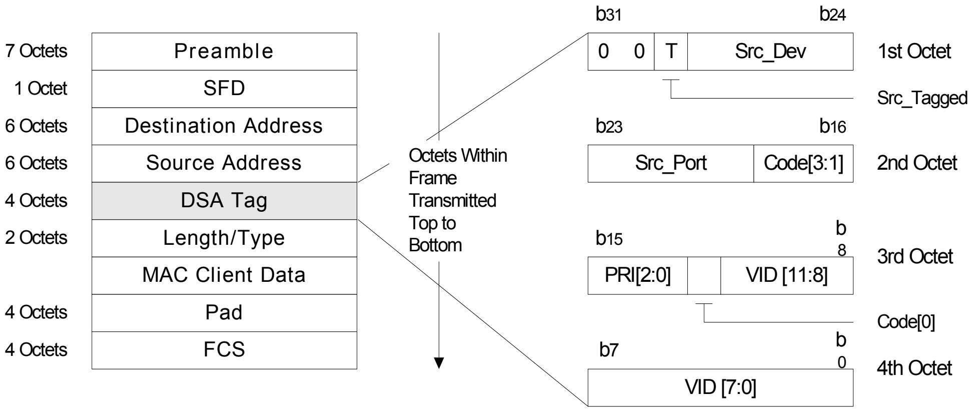
- See the Marvell Gigabit Ethernet switch specification for additional information on DSA tag.
- The Ethernet networking controller supports both extended (8B) and non-extended (4B) DSA tag formats.
- If using the DSA tag, disable the Marvell Header mode by setting the <MHEn> field in the Marvell Header Register (i=0–3) (Table 692 p. 999) to 0. The DSA feature cannot be used simultaneously with the Marvell Header.

Copyright © 2014 Marvell

15.8.1 Receive Operation

When the DSA tag is enabled in the switch, each packet is padded with four or eight octets after the SA (see Figure 66).

Figure 66: Rx Packet with DSA Tag Example (4B tag, TO_CPU Format)



The Ethernet networking controller can receive one of the two DSA tag formats: TO_CPU format or FORWARD format. Table 72 and Table 73 summarize the DSA tag fields (for full details, see the Gigabit Ethernet switch specification).

Table 72: DSA Tag Fields (TO_CPU Format)

Bits	Name	Description
Word0		
31:30	TagCommand	00 = TO_CPU
29	SrcTagged/ TrgTagged	0 = Packet was received/transmitted from/to a network port untagged. 1 = Packet was received/transmitted from/to a network port tagged.
28:24	SrcDev/ TrgDev	Source/Destination device number on which the packet was received/transmitted.
23:19	SrcPort[4:0]/ TrgPort[4:0]	Source/Destination port number on which the packet was received/transmitted.
18:16	CPU_Code[3:1]	CPU_Code[3:1] when using none extended mode. CPU_Code[3:0] must be set to 0xF to indicate an Extended DSA tag.
15:13	UP	The IEEE 802.1p User Priority field assigned to the packet.
12	CPU_Code[0]	CPU_Code[0] when using none extended mode. CPU_Code[3:0] must be set to 0xF to indicate an Extended DSA tag.
11:0	VID	The packet’s incoming/outgoing VID
Word1		
31	Extend	Word1 is the last extension. Must be cleared to 0.
30	CFI	When SrcTagged = 1, this is the VLAN Tag CFI bit with which the packet was received from the network port.
29:27	Reserved	

Copyright © 2014 Marvell

Table 72: DSA Tag Fields (TO_CPU Format) (Continued)

Bits	Name	Description
26	Truncated	Packet sent to CPU is truncated.Indicates that only the first 128B of the packet are sent to the CPU. The packet's original byte count is forwarded to the CPU in <PktOrigBC> field.
25:12	PktOrigBC	The packet's original byte count.
11	Reserved	
10	SrcPort[5]/ TrgPort[5]	Bit 5 (MSB) of the SrcPort[5:0]/TrgPort[5:0] field.
9	Reserved	
8	SrcTrg	SrcTrg indicates the type of data forwarded to the CPU. 0 = The packet was forwarded to the CPU by the Ingress pipe and this tag contains the packet's source information. 1 = The packet was forwarded to the CPU by the Egress pipe and this tag contains the packet's destination information.
7:0	LongCPUCode	8 bit of CPU code

Table 73: DSA Tag Fields (FORWARD Format)

Bits	Name	Description
Word0		
31:30	TagCommand	11 = FORWARD
29	SrcTagged	0 = Packet was received from a network port untagged. 1 = Packet was received from a network port tagged.
28:24	SrcDev	Source device number on which the packet was received.
23:19	SrcPort[4:0]/ SrcTrunk[4:0]	If SrcIsTrunk = 0, indicates source port number on which the packet was received. If SrcIsTrunk = 1, indicates source trunk number on which the packet was received.
18	SrcIsTrunk	If the packet was received from a network port that is part of a trunk, this bit is set to 1
17	Reserved	
16	CFI	When SrcTagged = 1, this is the VLAN Tag CFI bit with which the packet was received from the network port
15:13	UP	The IEEE 802.1p User Priority field assigned to the packet.
12	Extend	1 = There is one more DSA tag word.
11:0	VID	The packet's incoming/outgoing VID
Word1		
31	Extend	Word1 is the last extension. Must be cleared to 0.
30	SrcTrunk[6]	When SrcIsTrunk = 1, this is the VLAN Tag CFI bit with which the packet was received from the network port (if SrcIsTrunk = 0, this bit is reserved)
29	SrcPort[5]/ SrcTrunk[5]	If SrcIsTrunk = 0, indicates source port number on which the packet was received. If SrcIsTrunk = 1, indicates source trunk number on which the packet was received.

Copyright © 2014 Marvell

Table 73: DSA Tag Fields (FORWARD Format) (Continued)

Bits	Name	Description
28	EgressFilter Registered	If set to 1, indicates that the packet is Egress filtered as a Registered packet (when this field is 0, the type of the packet—Multicast or Unicast—is set according to the packet's MAC DA[40]).
27:26	Reserved	
25	Routed	If set to 1, indicates that the Packet has been Layer 3 routed.
24:20	SrcID	Packet's Source ID.
19:13	QoSProfile	Packet's QoS profile.
12	use_vidx	0 = Unicast packet forwarded to the Target port specified in this tag. 1 = Multicast packet forwarded to the Multicast group specified in this tag.
11	VIDX[11]	When use_vidx = 1, indicates the Multicast group to which the packet is transmitted (if use_vidx = 0, reserved)
10:5	VIDX[10:5]/ TrgPort	When use_vidx = 1, indicates the Multicast group to which the packet is transmitted. When use_vidx = 0, specifies the target port to which the packet is forwarded.
4:0	VIDX[4:0]/ TrgDev	When use_vidx = 1, indicates the Multicast group to which the packet is transmitted. When use_fivx = 0, specifies the target device to which the packet is forwarded.

Note

- The DSA tag is transferred along with the rest of the packet to memory. None of its fields are updated in the Rx descriptor status.
- Packet CRC generation/checking includes the four/eight octets of the Marvell Header.

The Ethernet networking controller also supports Rx queuing based on DSA tag fields. This is an alternative to the regular queuing, as described in Section 15.6.1, Parsing the Frames, on page 277.

The <DAPrefix> field in the Marvell Header Register (i=0–3) (Table 692 p. 999) selects the queuing policy:

- If cleared to 0x0, the regular priority queuing is working.
- If set to 0x1, Rx queue is determined according to UP field.
- If set to 0x2, needs to distinguish between two cases:
  - If it is a FORWARD format, the Rx queue is determined according to UP field (as if <DAPrefix> is set to 1).
  - If it is a TO_CPU format, the Rx queue is determined according to the 4-bit/8-bit <CPU_Code> field of the DSA tag. When using <DAPrefix> is set to 2, the Destination Address Filter Special Multicast Table (DFSMT) is no longer used for address filtering. Instead, it is used for mapping from <CPU_Code> to Rx queue.
- If set to 0x3, the Rx queue is determined according to SrcPort, SrcDev, and UP[2:1].

For example, if the application requires four priority queues (Low and High) for Switch Device Number3 Port2, and four priority queues (Low and High) for the rest of the switches ports, set the following fields in the Marvell Header Register (i=0–3) (Table 692 p. 997):

- <DAPrefix> to 0x3
- <SPID> field to 0x2

- <SDID> field to 0x3
- <MHMask> to 0x0 (indicating to use all eight queues)

This way, packets received from Switch Number 3 Port2 are placed in queues 4–7, and packets received from the other ports are placed in queues 0–3.

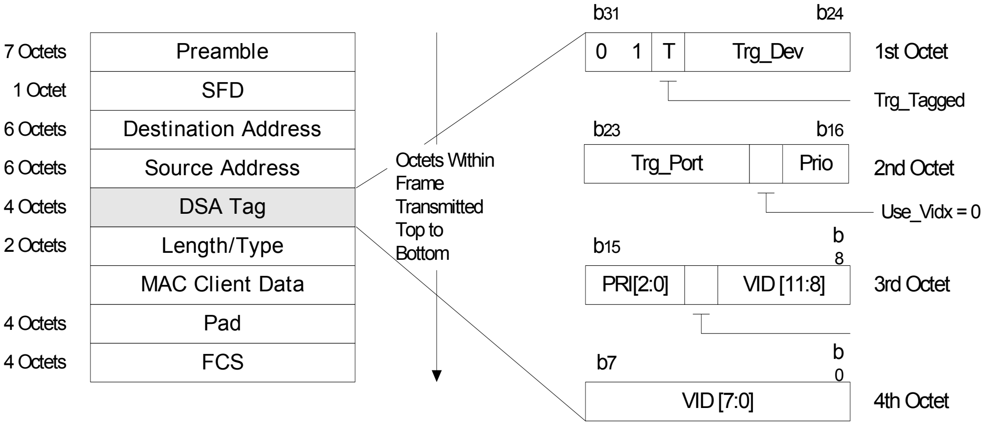
Note

- Packet accept/reject is handled in the same way as described for the Marvell Header mode, see Section 15.6.1, Parsing the Frames, on page 277.
- When <DAPrefix> is set to 0x3, and the <SDIDEn> bit is cleared to 0, the <SrcDev> field are not taken into account in the queuing decision. Only the SrcPort is used, as in the Marvell Header mode.
- When <DAPrefix> is set to 0x3, and there is no SrcPort (FORWARD format, with SrcIsTrunk bit set), the Rx queue is determined according to UP[2:0] field only.
- If the received packet is not set to TO_CPU or FORWARD format, the packet is queued as if <DAPrefix> is cleared to 0x0 (regular queuing).

15.8.2 Transmit Operation

When the DSA tag is enabled, the software must add four/eight octets of DSA tag after SA, as shown in Figure 67.

Figure 67: Tx Packet with a DSA Tag Example (FROM_CPU format, use_vidx = 0)



Note

See the specific GbE switch specification for exact details on the DSA tag format and usage.

The Ethernet networking controller generates CRC for the entire packet, including the DSA tag. When the switch receives the packet, it strips the DSA tag, and recalculates the new CRC before forwarding the frame to the network.

Copyright © 2014 Marvell

15.9 Wake On LAN (WOL) Modes and Events

The WOL feature supports IPv4.

The port supports 2 WOL modes:

- Active mode
- Sleep mode

The software can configure the port to Sleep mode, by setting the <SleepMode> field in the WOL Sleep (WOLSLP) Register (i=0–3) (Table 626 p. 955). In this mode, instead of the active mode's queuing decision, the port only queues the packets that pass the WOL filter. All other packets are discarded.

If a packet passes the WOL filtering, it will be queued. If a packet is queued, a WOL interrupt is asserted (if not masked), and the port moves (by hardware) back to Active mode, so the next packets will be queued according to the Active mode queuing decision.

The WOL event cause can be read through the WOL Sleep Interrupt Cause (WOLSIC) Register (i=0–3) (Table 628 p. 957).

Any WOL event interrupt can be masked through the WOL Sleep Interrupt Mask (WOLSIM) Register (i=0–3) (Table 629 p. 959).

The WOL mode filters all packets (but the waking packets), so the CPU can move to any power saving mode. When a WOL event occurs (and the event interrupt is not masked), a WOL interrupt is asserted and the power management mechanism wakes the CPU.

The WOL events are the following:

- Magic packet arrival (6B of 0xFF, followed by 16 times the device MAC address)
- ARP Packet arrival with a configurable IPv4 address
- Waking frame arrival (user-configured waking frame)
- A change in the link status
- A Unicast packet arrival, with MAC address that passed the hardware parsing
- A Multicast packet arrival, with MAC address that passed the hardware parsing
- Packet arrival with a specifically configured EtherType or IP-type packets (for example, wake up by a Ping packet (IP type = ICMP)) that passed the Rx parsing

In WOL mode, the port has the option to be woken up (returned to Active mode) by one of the WOL events listed in Section 15.9.2.

Each WOL event can be enabled/disabled via the corresponding bit in the WOL Wakeup Event Enable (WOLWEE) Register (i=0–3) (Table 627 p. 955). Once disabled, the port does not wake up due to this type of event.

15.9.1 WOL Wakeup Events Counter

The WOL Wakeup Events Counter (WOLWEC) Register (i=0–3) (Table 630 p. 960) counts the number of identified and enabled wake-up events while in Wake mode.

15.9.2 Magic Packet Event

The incoming RX packets are compared with a “magic pattern”.

The magic pattern consists of 6B of 0xFF (byte aligned), followed by 16 iterations of the device’s MAC address. The MAC address is defined by the MAC Address High (MACAH) Register (i=0–3) (Table 591 p. 932) and MAC Address Low (MACAL) Register (i=0–3) (Table 590 p. 931) registers.

A magic pattern event is identified if the received packet matches with the magic pattern.

15.9.3 ARP Packet Event

ARP wakeup event, is defined as the reception of an ARP packet, with IP address equal to one of two preconfigured IP addresses.

For the ARP wakeup event configuration, set the following registers:

- Arp IP0 Address (AIP0ADR) Register (i=0–3) (Table 624 p. 954)
- Arp IP1 Address (AIP1ADR) Register (i=0–3) (Table 625 p. 954)

15.9.4 Wakeup Frame Event

A wakeup frame is a group of packets that match one of four preconfigured patterns.

Each pattern is up to 128B long. Each byte of the pattern has a mask bit.

The received packet data, starting from a preconfigured location is compared in parallel with all 4 patterns.

For the wakeup frame filtering configuration, set the following registers:

- WOL Wakeup Frame Location (WOLWFL) Register (i=0–3) (Table 634 p. 962)
- WOL Wakeup Frame Size (WOLWFS) Register (i=0–3) (Table 635 p. 963)
- WOL Wakeup Frame Select (WOLWFSEL) Register (i=0–3) (Table 636 p. 963)
- WOL Wakeup Frame Data (WOLWFD)<n> Register (i=0–3, n=0–31) (Table 637 p. 964)
- WOL Wakeup Frame Mask (WOLWFM)<n> Register (i=0–3, n=0–31) (Table 638 p. 965)

15.9.5 Specific EtherType/IP Frame Event

An EtherType/IP Frame is the reception of a frame with preconfigured EtherType or IPV4 protocol field (assuming that the packet was identified as IPV4 by the hardware parser).

There are three sets of preconfigured types that can be set in the following registers:

- WOL Type1 (WOLT1) Register (i=0–3) (Table 631 p. 960)
- WOL Type2 (WOLT2) Register (i=0–3) (Table 632 p. 961)
- WOL Type3 (WOLT3) Register (i=0–3) (Table 633 p. 961)

Note

- For VLAN, SNAP-LLS etc., the last EtherType is compared.
- The packets, with types matching the preconfigured wakeup packet types, are considered as wakeup events only if the packet passed the hardware parser filtering.

15.10 Interrupts

The networking controller supports many interrupt events, registered and controlled in the following registers:

- Ethernet Unit Interrupt Cause (EUIC) Register (i=0–3) (Table 580 p. 924) and Ethernet Unit Interrupt Mask (EUIM) Register (i=0–3) (Table 581 p. 926)
- Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3) (Table 723 p. 1024) and Port RX_TX Threshold Interrupt Mask (PRXTXThIM) Register (i=0–3) (Table 724 p. 1026).
- Port RX_TX Interrupt Cause (PRXTXIC) Register (i=0–3) (Table 725 p. 1026) and Port RX_TX Interrupt Mask (PRXTXIM) Register (i=0–3) (Table 726 p. 1028).
- Port Misc Interrupt Cause (PMiscIC) Register (i=0–3) (Table 727 p. 1028) and Port Misc Interrupt Mask (PMiscIM) Register (i=0–3) (Table 728 p. 1031).
- Port Interrupt Cause Register (i=0–3) (Table 708 p. 1012) and Port Interrupt Mask Register (i=0–3) (Table 709 p. 1013)
- WOL Sleep Interrupt Cause (WOLSIC) Register (i=0–3) (Table 628 p. 957) and WOL Sleep Interrupt Mask (WOLSIM) Register (i=0–3) (Table 629 p. 959).

The Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3) (Table 723 p. 1024) and Port RX_TX Interrupt Cause (PRXTXIC) Register (i=0–3) (Table 725 p. 1026) contain various receive and transmit events indication related to the eight receive and eight transmit queues.

15.10.1 Interrupt Coalescing

Since the networking controller line rate provides a high packet rate, it is important to reduce the number of interrupts that the Ethernet DMA may generate.

For this purpose, the DMA receive and transmit operations include several modes that provide the option of selecting the type of events that generate interrupts.

The most intensive types of interrupts are the packet-level interrupts on receive and transmit. The device provides a programmable mechanism that allows coalescing these types of interrupts.

15.10.1.1 Tx Interrupt Coalescing

Transmit interrupt coalescing is accomplished by setting the TXBufferThresholdCrossQueue* bits in the Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3), only when the number of transmitted buffers that where not yet released by the CPU, is larger than the threshold programmed in the <transmittedBuffersThreshold> field in the Port TX Queues Descriptors Queue Size (PTXDQS)<q> Register (i=0–3, q=0–7) (Table 644 p. 969).

The number of transmitted buffers that where not yet released by the CPU is represented by the <TransmittedBuffersCounter> field in the Port TX Queues Status (PTXS)<q> Register (i=0–3, q=0–7) (Table 645 p. 969).

15.10.1.2 Rx Interrupt Coalescing

Receive interrupts coalescing is done by setting the RxBufferInterruptCoaliascingThresholdAlertPriorityQueue* bits in the Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3), only when the number of received buffers that were not yet processed by the CPU, is larger than the threshold programmed in the <OccupiedDescriptorsThreshold> field in the Port RX Queues Descriptors Queue Threshold (PRXDQTH)<q> Register (i=0–3, q=0–7) (Table 614 p. 949)r.

In any case, the maximum delay from reception of a packet to a buffer will be not be longer than the delay programmed in the <RxInterruptTimeThreshold> field in the Port RX Interrupt Threshold (PRXITTH)<q> Register (i=0–3, q=0–7) (Table 722 p. 1023). The number of received buffers that were not yet processed by the CPU is represented by the <OccupiedDescriptorsCounter> field in the Port RX Queues Status (PRXS)<q> Register (i=0–3, q=0–7) (Table 615 p. 950).

15.11 Transmit Queues Arbitration and Bandwidth Limitation

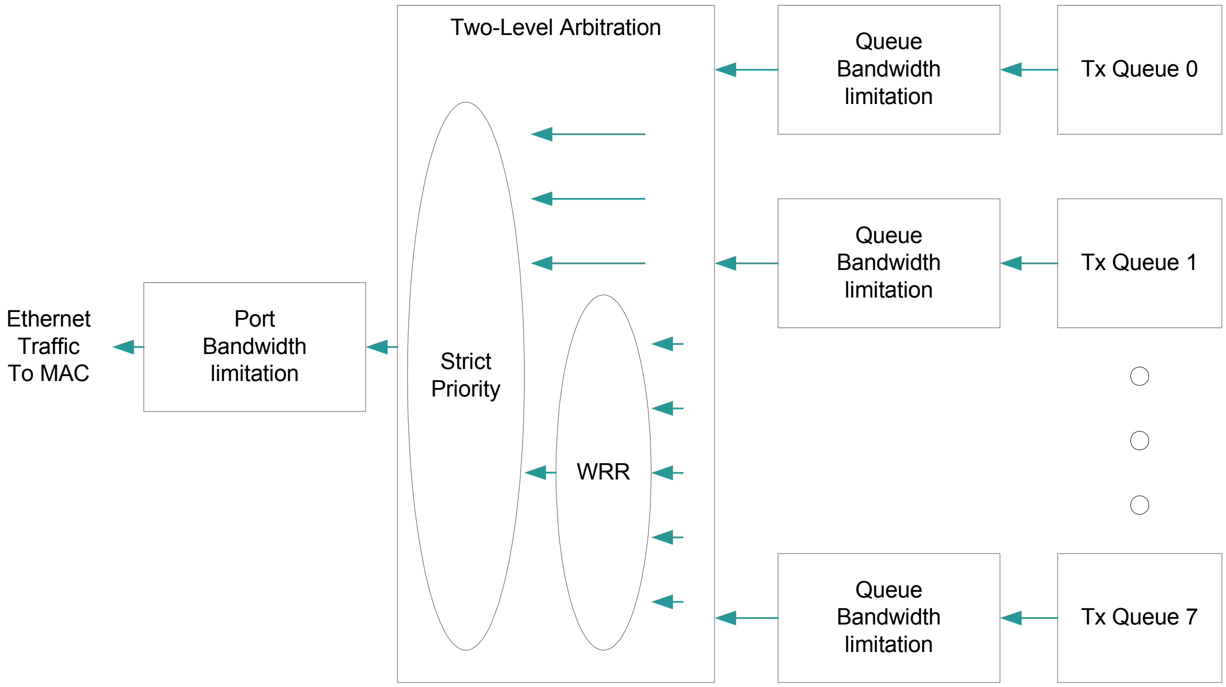
The transmit controller includes flexible bandwidth control distribution among eight transmit queues.

Each queue’s egress traffic is shaped by a queue bandwidth limitation mechanism. In addition, the overall port traffic is shaped by a port bandwidth limitation mechanism.

The arbitration between queues is implemented by the strict priority and WRR arbitration mechanism.

For a transmit queue to be selected to transmit the next frame, the queue must be enabled by setting the corresponding bit to 1 in the <ENQ> field in the Transmit Queue Command (TQC) Register (i=0–3) (Table 639 p. 967), and the queue should have a frame ready for transmit.

Figure 68: Transmit Queues Arbitration and Bandwidth Limitation


Two-Level Arbitration
Queue Bandwidth limitation
Tx Queue 0
Queue Bandwidth limitation
Tx Queue 1
Ethernet Traffic To MAC
Port Bandwidth limitation
Strict Priority
WRR
Queue Bandwidth limitation
Tx Queue 7

15.11.1 Priority Arbitration Modes

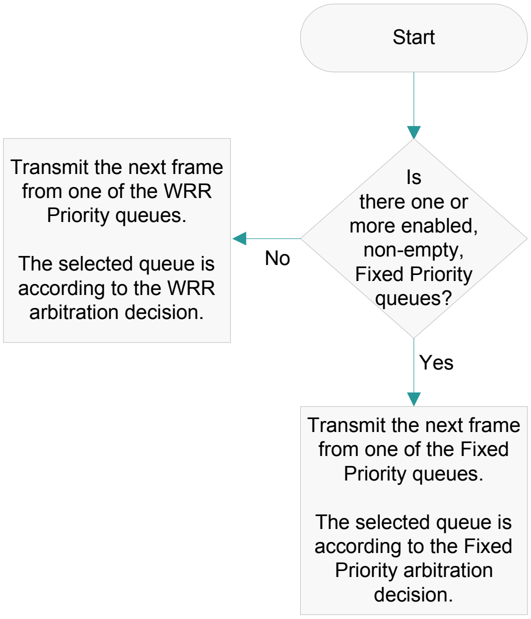
Each transmit queue can be configured to one of the two arbitration modes:

- Fixed Priority arbitration
- Weighted-Round-Robin (WRR) arbitration

The priority arbitration mode of each queue is configured by the corresponding bit of the <FIXPR> field in the Transmit Queue Fixed Priority Configuration - Arbiter ver1 (TQFPC_AV1) Register (i=0–3) (Table 666 p. 980) (when version 1 mechanism is used) or by the corresponding <FIXPR> field in the Transmit Queue Fixed Priority Configuration - Arbiter ver3 (TQFPC_AV3) Register (i=0–3) (Table 675 p. 985) (when version 3 mechanism is used).

The transmit controller arbitrates between the queues in two modes as indicated in Figure 69.

Figure 69: Transmit Controller Arbitration Flow


Start
Is there one or more enabled, non-empty, Fixed Priority queues?
No
Transmit the next frame from one of the WRR Priority queues.
The selected queue is according to the WRR arbitration decision.
Yes
Transmit the next frame from one of the Fixed Priority queues.
The selected queue is according to the Fixed Priority arbitration decision.

15.11.1.1 Fixed Priority Arbitration

Select the Fixed Priority mode to transmit the non-empty, enabled and non-bandwidth limited queue with the highest queue number first (for example, select queue 7, then 6, etc.). A queue can be excluded from the arbitration, if it passed per-queue programmable bandwidth-limitation based on the token-bucket mechanism (see Section 15.11.2, Transmit Bandwidth Limitation, on page 292).

15.11.1.2 Weighted Round-Robin Arbitration

The controller would service one of the Weighted Round Robin (WRR) queues only when the fixed priority queues have nothing to transmit, or they are limited by the bandwidth limitation.

The WRR priority mode is utilized to distribute bandwidth among transmit queues in a round-robin fashion when each WRR queue gets bandwidth portion relative to its configured weight. This is called below “WRR arbitration”.

A WRR queue may be excluded from the WRR arbitration, if it passed per-queue programmable bandwidth-limitation based on token-bucket mechanism (see Section 15.11.2, Transmit Bandwidth Limitation, on page 292).

Configure the queue weight (0–255) by writing to the corresponding Transmit Queue Fixed Priority Configuration - Arbiter ver1 (TQFPC_AV1) Register (i=0–3) (Table 666 p. 980) (when version 1 mechanism is used) or Transmit Queue Fixed Priority Configuration - Arbiter ver3 (TQFPC_AV3) Register (i=0–3) (Table 675 p. 985) (when version 3 mechanism is used) and setting the <WRRWGT> field to the desired weight value (measured in 256B units).

As used here, the *WRR bandwidth* is the available transmit bandwidth, that is not consumed by traffic of the fixed-priority queues.

The WRR arbitration end result is calculated by dividing the WRR bandwidth between the WRR-queues according to each queue's WRRWGT/ (Sum of all WRRGTs of the WRR-queues that are not bandwidth limited by the token bucket mechanism).

When several <WRRWGT> combinations yield the same bandwidth distribution, the user must use a combination with the smallest <WRRWGT> value closest to the Maximum Transmit Unit.

The WRR bandwidth distribution is determined by counting the transmitted bytes from each WRR queue and limiting the schedule of queues that used up their bandwidth portion (WRRWGT) until all other queues finish transmission.

15.11.2 Transmit Bandwidth Limitation

This device integrates 2 transmit bandwidth limitation mechanisms:

- Version 1 (Basic mechanism)
  - Low resolution bandwidth limitation (3.125 Mbps for TCLK=200 MHz)
  - Software compatible with legacy Marvell® controllers.
- Version 3 (Enhanced mechanism)
  - High resolution bandwidth limitation (less than 10 Kbps)
  - Transmit Queues Egress Jitter Pacing (EJP) Arbitration

15.11.2.1 Transmit Bandwidth Limitation Mechanism Version 1

Transmit Queue Bandwidth Limitation

To implement bandwidth limitation on transmit queues, specify the maximum available bandwidth for each queue.

The bandwidth limitation is implemented by a Token Bucket per queue. The tokens are added to the *Bucket* in a constant configurable rate and drained from the Bucket when the queue transmits. A queue is only allowed to transmit when the number of Tokens is larger then Maximum Transmit Unit (MTU). It is possible to configure the queue Token-Bucket size that gives control over the amount of credit (silent time) that the queue is allowed to accumulate.

The queue Token-Rate is programmed by setting the corresponding queues' <TKNRT> field in the Transmit Queue Token Bucket Configuration - Arbiter ver1 (TQxTBCFG_AV1)<q> Register (i=0–3, q=0–7) (Table 671 p. 983).

To disable a bandwidth limitation, set the <TKNRT> field to its maximum value.

The queue Bucket size is programmed by writing to the corresponding queue's <MTBS> field in the Transmit Queue Token Bucket Configuration - Arbiter ver1 (TQxTBCFG_AV1)<q> Register (i=0–3, q=0–7) (Table 671 p. 982), in 256B units. The tokens accumulate in the bucket until the amount equals the <MTBS> setting.

The user may examine or modify the current value of a queues token bucket (1/64B units) by read/write to the corresponding queue's <TKNBKT> field in the Transmit Queue Token-Bucket Counter - Arbiter ver1 (TQxTBC_AV1)<q> Register (i=0–3, q=0–7) (Table 670 p. 982).

Transmit Port Bandwidth Limitation

The transmit controller implements a bandwidth limitation on all outgoing traffic. This applies to Fixed priority and WRR priority queues.

It is possible to specify the maximum available port bandwidth. The bandwidth limitation is implemented by a port Token-Bucket. The tokens are added to the Bucket in a constant configurable rate and drained from the Bucket when the port transmits. The port is allowed to transmit only when the number of Tokens is bigger than the MTU. It is also possible to configure the port Token-Bucket size for the amount of credit (silent time) the queue is allowed to accumulate.

Copyright © 2014 Marvell

The port Token-Rate is programmed by setting the <PTKNRT> field in the Port Transmit Token-Bucket Rate Configuration - Arbiter ver1 (PTTBRC_AV1) Register (i=0–3) (Table 667 p. 981) to the desired value in [1/64B per 8 clock cycles] units.

To disable bandwidth limitation, set the <PTKNRT> field to its maximum value.

The port Bucket size is set by writing to the <PMTBS> field in the Port Maximum Token Bucket Size - Arbiter ver1 (PMTBS_AV1) Register (i=0–3) (Table 669 p. 982), in 256B units. The tokens accumulate in the port bucket until the setting for <PMTBS> is met.

The user may examine or modify the current value of port Token-Bucket by reading/writing to the <PTKNBKT> field in the Port Transmit Token-Bucket Counter - Arbiter ver1 (PTTBC_AV1) Register (i=0–3) (Table 673 p. 984), in [1/64B per 8 clock cycles] units.

Maximum Transmit Unit

The MTU is a configurable value common to the transmit port bandwidth limitation and to each of the eight Tx queues bandwidth limitations.

Transmission from a queue is enabled if the following 2 conditions occur:

- The number of tokens in the port token bucket is greater or equal to MTU value.
- The number of tokens in the queue token bucket is greater or equal to MTU value.

The MTU is programmed by setting the <MTU> field in the Maximum Transmit Unit - Arbiter ver1 (MTU_AV1) Register (i=0–3) (Table 668 p. 981), in 256B units.

Bandwidth Limitation Configuration

The bandwidth (BW) limitation is achieved by configuring the Token Rate value.

The bandwidth limitation is a function of the Token Rate value and the TCLK (core clock) frequency, as described in the following formula:

BW[Mb/sec] = TokenRate[1/64B per 8 cycles]*TCLK[MHz]/64

Table 74 displays some examples of the Token Rate bandwidth configuration values.

Table 74: Token Rate Configuration Examples

Bandwidth (Mbps)	TCLK (MHz)	Token Rate Decimal Value (1/64B/8 cycles)
3.125	200	1
1000	200	320
2506.172	200	629
3.906	250	1
1000	250	256

15.11.2.2 Transmit Bandwidth Limitation Mechanism Version 3

To enable the transmit bandwidth limitation mechanism version 3, set the <BW_Lim_Sel> field in the Transmit Queue Command1 - Arbiter ver3 (TQC1_AV3) Register (i=0–3) (Table 674 p. 984).

The transmit bandwidth limitation implements egress bandwidth limitation, using a separate token bucket mechanism per port and per each queue.

The token bucket is bit-based, with the following attributes:

Table 75: Token Bucket Attributes

<BasicRefillNoOfClocks>	16-bit	Basic Refill Number of Clocks The basic period used for the token bucket update, in units of core clock cycles. The <BasicRefillNoOfClocks> field in the Basic Refill No of Clocks - Arbiter ver3 (BRC_AV3) Register (i=0–3) (Table 676 p. 985) is configurable per Ethernet port.
<PortRefillPeriod> **<QueueRefillPeriod>**	10-bit	Port Refill Period and Queue Refill Period The interval at which the token bucket is incremented by the value of the <PortTokenRefillValue> or <QueueTokenRefillValue> field. This interval for the port/queue is equal to: Port refill interval [ns] = BasicRefillNoOfClocks * PortRefillPeriod * Core clock period [ns] Queue refill interval [ns] = BasicRefillNoOfClocks * QueueTokenRefillValue* Core clock period [ns] ■ The <PortRefillPeriod> field in the Port Bucket Refill - Arbiter ver3 (PRefill_AV3) Register (i=0–3) (Table 678 p. 986) is configurable per port. ■ The <QueueRefillPeriod> field in the Queue Bucket Refill - Arbiter ver3 (QRefill_AV3)<q> Register (i=0–3, q=0–7) (Table 681 p. 988) is configurable per transmit queue.
<PortTokenRefillValue> **<QueueTokenRefillValue>**	19-bit	Port Token Refill Value and Queue Token Refill Value Number of tokens to be added to the port/queue token bucket every port/queue refill interval. One token represents one packet bit. The token refill value is configured independently for the per-port token buckets and for the per-queue token buckets. The token refill value ranges from 1 token to 512K-1 tokens (0 value is not allowed). The token bucket counter cannot exceed the token bucket size.
<PortBucketSize> **<QueueBucketSize>**	32-bit 31-bit	Port Bucket Size and Queue Bucket Size The maximum number of tokens that can be accumulated by the token bucket. The token bucket size determines the maximum burst size permitted. The bucket size ranges from 512 bits to 4 Gb -1 (for port buckets) or from 512 bits to 2 Gb-1 (for queue buckets), in steps of 1 bit. A value lower than 512 bits is not allowed. The token bucket size is configured independently for the per-port token bucket and for the per-queue token buckets.

Table 75: Token Bucket Attributes (Continued)

<MTU>	19-bit	Maximum Transmit Unit The minimum number of tokens required to allow packet transmission. **NOTE:** One token represents 1 packet bit. This value defines the minimum number of tokens required to permit a packet transmission. Packet will be transmitted from a queue only if both the queue token bucket counter and the Ethernet port token bucket counters value is equal or higher than the MTU. The MTU is a per Ethernet port configuration.

Bandwidth limitation calculation:

Bandwidth limitation [Mbps] = Core Clock Frequency [MHz] * Port/QueueTokenRefillValue [bits] /(BasicRefillNoOfClocks * Port/QueueRefillPeriod)

The current value of the 32-bit port/queue token bucket can be read through the <PTKNBKT> field in the Port Transmit Token-Bucket Counter - Arbiter ver1 (PTTBC_AV1) Register (i=0–3) (Table 673 p. 983) / Port Transmit Token-Bucket Counter - Arbiter ver3 (PTTBC_AV3) Register (i=0–3) (Table 680 p. 988) and the <QTKNBKT> field in the Queue Transmit Token-Bucket Counter - Arbiter ver3 (QxTTBC_AV3)<q> Register (i=0–3, q=0–7) (Table 683 p. 990) (one per queue).

Table 76 provides a bandwidth limitation configuration example for a 200-MHz clock.

Table 76: Bandwidth Limitation Configuration Example for a 5-ns (200 MHz) TCLK

Basic Refill (Number of Clocks)	Port / Queue Refill Period	Port / Queue Refill Interval	Bandwidth Limit Resolution (Port / Queue Token Refill Value = 1)
20	1000	20 * 1000 * 5 [ns] = 100 µs	1/100 µs = 10 Kbps
20	250	25 µs	40 Kbps
20	125	12.5 µs	80 Kbps
20	50	5 µs	200 Kbps
20	25	2.5 µs	400 Kbps
20	10	1 µs	1000 Kbps
20	5	0.5 µs	2000 Kbps
20	1	100 ns	10,000 Kbps

15.12 Transmit Queues Egress Jitter Pacing (EJP) Arbitration

The EJP (Egress Jitter Pacing) mechanism is an enhanced arbitration mechanism, enabling low jitter. It complies with IEEE 802.1Qav pre-draft and enables implementing AVB technology.

The EJP mechanism is an arbitration mechanism for four queues.

15.12.1 EJP Mechanism

When EJP mode is enabled, only enable queues 0–3, disable queues 4–7.

The EJP mechanism combines the token bucket mechanism with an advanced algorithm, for optimized jitter performance.

Note

Utilize this mode of operation for non-half duplex and non-10-Mbps operations only, where Flow Control is not expected to enabled.

The EJP algorithm is based on a combination of:

- Sending packets from specific queue, when its bucket is full enough ("full enough" means that the number of tokens accumulated in the bucket is equal to or more than the Queue Maximum Transmit Unit).
- Forcing a configurable minimum of inter-packet gap (IPG) between packets from the same queue.
- According to the leaky bucket parameters, the transmission timing of packets from queue 3 (highest priority) must not be delayed, because a packet from lower priority queue is in transmission.
  Therefore, if there is a possibility that a transmission from a lower priority queue (0,1,or 2) may delay a transmission from queue 3, then transmit queue 3 first, even if its bucket is not full enough.
- According to the leaky bucket parameters, the transmission timing of packets from queue 2 (second highest priority) must not be delayed, because a packet from lower priority queue is in transmission.
  Therefore, if there is a possibility that a transmission from a lower priority queue (0 or 1) may delay a transmission from queue 2, then transmit queue 2 before transmitting queue 0 or 1, even if its bucket is not full enough.

15.12.2 Initialization Sequence

To initialize the EJP follow these steps:

1. Disable Tx queues 0–7 (write the value 0xFF to the <DISQ> field in the Transmit Queue Command (TQC) Register (i=0–3) (Table 639 p. 966)).
2. Configure the following fields in the Transmit Queue Command1 - Arbiter ver3 (TQC1_AV3) Register (i=0–3) (Table 674 p. 984):
   - Set the <EJP_ENB> field.
   - Clear Reserved bit[3].
   - Set the <WRR_EJP_INIT> field.
   - Configure the <PTP_SYNC_ENB>.
3. Configure the <FIXPR> field in the Transmit Queue Fixed Priority Configuration - Arbiter ver3 (TQFPC_AV3) Register (i=0–3) (Table 675 p. 985).

Copyright © 2014 Marvell

4. Clear the <PTKNBKT> field in the Port Transmit Token-Bucket Counter - Arbiter ver3 (PTTBC_AV3) Register (i=0–3) (Table 680 p. 988).
5. Configure the port token bucket parameters:
   - Configure the <BasicRefillNoOfClocks> field in the <BRC_AV3> register.
   - Configure the <PortTokenRefillValue> field and <PortRefillPeriod> field in the Port Bucket Refill - Arbiter ver3 (PRefill_AV3) Register (i=0–3) (Table 678 p. 986), for Port Token Rate.
   - Configure the <MTU> field in the Maximum Transmit Unit - Arbiter ver3 (MTU_AV3) Register (i=0–3) (Table 677 p. 986). This defines the minimum tokens to be filled in the port bucket, before sending the next packet from one of the queues of the port.
   - Configure the <PortBucketSize> field in the Port Maximum Token Bucket Size - Arbiter ver3 (PMTBS_AV3) Register (i=0–3) (Table 679 p. 987). This field defines the maximum accumulated transmit credit, in 1-bit units, used for the port Token-Bucket bandwidth limitation mechanism. The <PTKNBKT> field in the Port Transmit Token-Bucket Counter - Arbiter ver3 (PTTBC_AV3) Register (i=0–3) (Table 680 p. 988) is incremented up to PMTBS.
6. For each of queues 0–3, configure the following parameters:
   - Clear the <QTKNBKT> field in the Queue Transmit Token-Bucket Counter - Arbiter ver3 (QxTTBC_AV3)<q> Register (i=0–3, q=0–7) (Table 683 p. 990).
   - Configure the <PortTokenRefillValue> field and <PortRefillPeriod> field in the Port Bucket Refill - Arbiter ver3 (PRefill_AV3) Register (i=0–3) (Table 678 p. 986) and the <QueueBucketSize> field in the Queue Maximum Token Bucket Size - Arbiter ver3 (QMTBS_AV3)<q> Register (i=0–3, q=0–7) (Table 682 p. 989).
   - Configure the <WRRWGT> field in the Transmit Queue Arbiter Configuration - Arbiter ver3 (TQxAC_AV3)<q> Register (i=0–3, q=0–7) (Table 684 p. 990).
   - Configure the <IPG> field in the Transmission Queue IPG - Arbiter ver3 (TQxIPG_AV3)<q> Register (i=0–3, q=2–3) (Table 685 p. 991). This field defines the minimum inter-packet gap, to be used by the EJP mechanism.
7. Configure the <TS> field in the Transmission Speed - Arbiter ver3 (TS_AV3) Register (i=0–3) (Table 689 p. 993).
8. Configure the <HiTKNinLoPkt> field in the High Token in Low Packet - Arbiter ver3 (HITKNinLOPKT_AV3) Register (i=0–3) (Table 686 p. 992). This field defines the number of tokens that are accumulated in the IsoHi (queue 3) token bucket, during transmission of the longest IsoLo (queue 2) packet.
9. Configure the <HiTKNinAsyncPkt> field in the High Token in Asynchronous Packet - Arbiter ver3 (HITKNinASYNCPKT_AV3) Register (i=0–3) (Table 687 p. 992). This field defines the number of tokens that are accumulated in the IsoHi (queue 3) token bucket, during transmission of the longest asynchronous (queues 1 and 0) packet.
10. Configure the <LoTKNinAsyncPkt> field in the Low Token in Asynchronous Packet - Arbiter ver3 (LOTKNinASYNCPKT_AV3) Register (i=0–3) (Table 688 p. 993). This field defines the number of tokens that are accumulated in the IsoLo (queue 2) token bucket, during transmission of the longest asynchronous (queue 1 and 0) packet.
11. To enable the EJP operation, clear the <WRR_EJP_INIT> field in the Transmit Queue Command1 - Arbiter ver3 (TQC1_AV3) Register (i=0–3) (Table 674 p. 985).
12. Enable Tx queues 0-3.

15.12.3 EJP Algorithm

The following pseudo-code describes the egress jitter pacing algorithm.

Note

In the following pseudo code, the addition of #*n* at the end of a register field, means the algorithm use this field from queue number *n* (e.g., QTKNBKT#3 means the QTKNBKT field of queue number 3).

```
function egress_jitter_pacer
begin
if {(QTKNBKT#3 >= QMTU#3) //enough tokens have accumulated in the
bucket of Q#3
  and (Nr of sys clk cycle passed since last packet transmition from
  Q#3 > TQxIPG#3 )}
    send out Packet from Q#3
else
  if {(QTKNBKT#2 >= QMTU#2) //enough tokens have accumulated in the
  bucket of Q#2
    and (Nr of sys clk cycle passed since last packet transmition from
  Q#2 > TQxIPG#2 )}
  )}
    if {(Q3 is not empty and (QTKNBKT#3 +HiTKNinLoPkt>= QMTU#3)
// Will next Q#3 packet expected transmition timing would be delayed by
sending the Q#2 packet?
send out Packet from Q#3
else
send out Packet from Q#2
else
if (QTKNBKT#1 >= QMTU#1) or QTKNBKT#0 >= QMTU#0
if {(Q3 is not empty and QTKNBKT#3 +HiTKNinAsyncPkt>= QMTU#3)
// Will next Q#3 packet expected transmition timing would be delayed by
sending the Q#0 or Q#1packet??
send out Packet from Q#3
else if {(Q2 is not empty and ((QTKNBKT#2 +LoTKNinAsyncPkt>= QMTU#2)))
// Will next Q#2 packet expected transmition timing would be delayed by
sending the Q#0 or Q#1packet?
send out Packet from Q#2
else
send out Async Packet(Q nr 1 or 0 - according to strict priority or
WRR)
end
```

15.13 Illegal Received Frames

The networking controller discards all undersized receive frames (with or without good CRC). These frames are not passed to the CPU, regardless of address filtering, and the appropriate error MIB counters are incremented. An undersized frame is determined by the Minimum Frame Size.

Oversized frames (greater than the Maximal Receive Unit—MRU) with or without bad CRC (bad checksum) are forwarded to the DMA queue with an error summary report in the Rx descriptor.

Note

An oversized packet is chopped according to the MRU setting (see <FrameSizeLimit> field in the Port MAC Control Register0 (i=0–3) (Table 702 p. 1008)), and only MRU bytes are transferred to memory. If the Marvell Header mode, DSA Tag, or Extended DSA Tag is used, an oversized packet is calculated based on MRU+2, MRU+4, or MRU+8 bytes, respectively.

15.14 Multi-CPU Support

The Networking controller supports Multi-CPU management of the Rx and Tx queues.

Each Rx and each Tx queue is associated to one or more of the CPUs, by configuring Port CPU0 to Queue (PCP02Q) Register (i=0–3) (Table 718 p. 1020), Port CPU1 to Queue (PCP12Q) Register (i=0–3) (Table 719 p. 1021), Port CPU2 to Queue (PCP22Q) Register (i=0–3) (Table 720 p. 1022) and Port CPU3 to Queue (PCP32Q) Register (i=0–3) (Table 721 p. 1022) registers.

Four sets of queue interrupt signals are supported, one set per each CPU.

The following interrupt cause/mask registers support multi-CPU:

- Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3) (Table 723 p. 1024)
- 0Port RX_TX Threshold Interrupt Mask (PRXTXThIM) Register (i=0–3) (Table 724 p. 1026)
- Port RX_TX Interrupt Cause (PRXTXIC) Register (i=0–3) (Table 725 p. 1026)
- Port RX_TX Interrupt Mask (PRXTXIM) Register (i=0–3) (Table 726 p. 1028)
- Port Misc Interrupt Cause (PMiscIC) Register (i=0–3) (Table 727 p. 1028)
- Port Misc Interrupt Mask (PMiscIM) Register (i=0–3) (Table 728 p. 1031)

When a queue cause bit in the above cause registers is set due to an event, and the corresponding mask bit is configured to generate an interrupt, then only the interrupt to the CPU (or CPUs) that are associated to the queue will be asserted. This is useful for reducing the interrupt processing load of each CPU, so each CPU receives the interrupts for its associated queues only.

In addition, when a specific CPU access the queues interrupt registers CPU ID is identified according to the Mbus CBE[13:12] bits:

- Cause bits of queues which are *associated* with the CPU will be read according to their value in the register.
- Cause bits of queues which are *not associated* with the CPU will be read as 0 (not asserted) and will not be affected by the access (for example, clear on read bits will not be cleared).

15.15 L2-Cache Prefetch Engine Support

The networking controller supports the L2-cache prefetch engine.
It set the Prefetch_command field in the RX descriptor, to be used by the PFE (Level-2 cache prefetch engine).

The CPU triggers the Prefetch Engine, and then the Prefetch Engine:

- Fetches the descriptor into L2 cache.
- Reads the prefetch_command from the descriptor, and according to it, executes L2 prefetch of the relevant parts of the packet.

The networking controller copies the configurable prefetch_command0 profile to the Prefetch_command field in the RX descriptor.

16 Hardware Buffer Management Controller

This section describes the Hardware Buffer Management (BM) controller.

The BM performs allocation and buffer release to offload the CPU from software-intensive dynamic memory allocation and to obtain greater flexibility in allocating buffers to hardware units. The BM has four Buffer Pointer (BP) pools. Fetching or releasing a buffer from/to a pool is accomplished using read/write operations. The BM is a target on the Mbus and fetch/release operations are implemented as read/write transactions to the BM external address space. Units that access the BM controller have to allocate one of the address space windows to the BM controller’s base address and target ID. The BM external address space must be configured as non-bufferable and non-cacheable.

The BM and buffer queues are initialized by the host CPU when buffers are allocated. Multiple pools are useful for supporting multiple CPUs and variable buffer sizes.

Free BPs can be allocated by the BM to the CPU or hardware units (such as the networking controller), and similarly freed buffer pointers can be released to the BM by the CPU or hardware units.

For example, during a packet forwarding application, a free BP is allocated to the GbE controller receiver for storing an incoming packet. After the packet has been forwarded, the buffer is released by the GbE controller transmitter without software buffer management overhead. During packet termination, a free buffer can be allocated to the GbE controller receiver and released by software.

The BP pools reside in external memory buffers (called BPPE) and the BM controller maintains an internal buffer pointer memory (called BPPI), to achieve low latency in allocation and release.

The Buffer Management controller registers are located in Appendix A.7, Buffer Management Controller Registers, on page 1035.

16.1 Features

The Hardware Buffer Management controller provides these features:

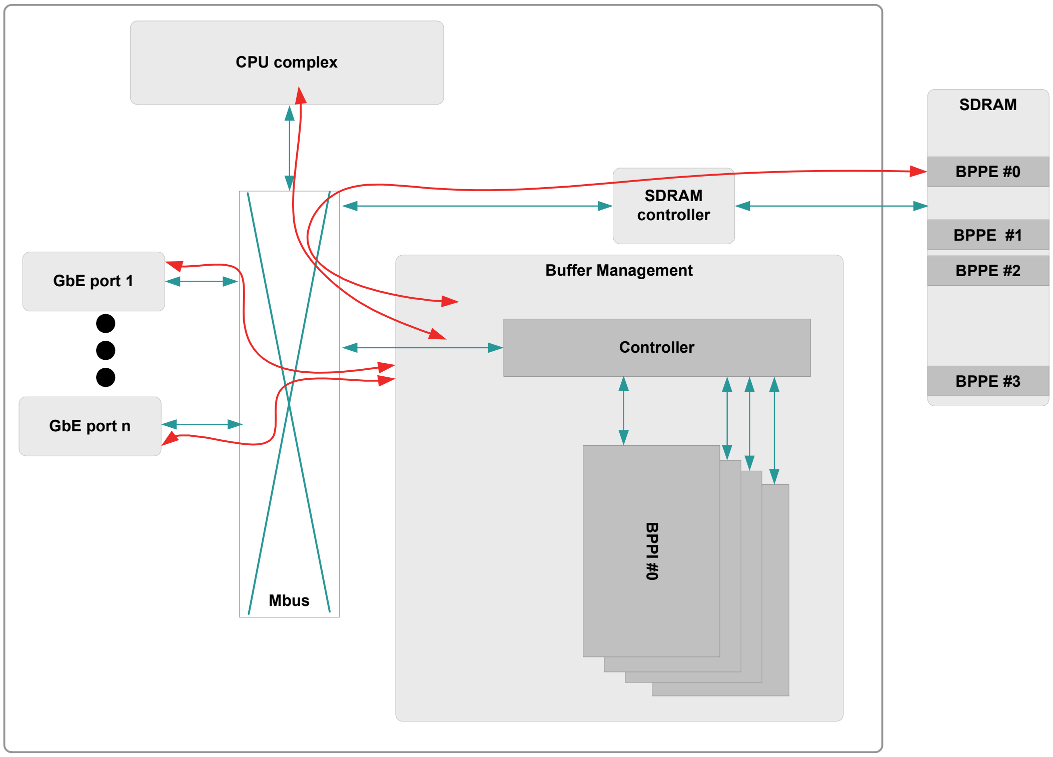
- Four separated BP pools that are managed independently. This allows for 4 different buffer and pool types or sizes.
- Up to 64-KB (16K BP) pool size (per pool).
- Dedicated internal memory of 256 BPs (per pool).
- Allocating and releasing BPs upon demand, via read and write Mbus transactions (The BM is an Mbus target).
- Up to 32 BPs can be allocated or released in a single transaction.
- Error reporting (interrupt) on a buffer pool overflow or underflow event.
- Dedicated DMA to copy BPs from external memory to internal memory and vise versa.
- 32 or 8 BPs are copied to/from external memory in a single DMA transaction.

16.2 Functional Description and Data Flow

16.2.1 System Integration

Figure 70 depicts the buffer management system integration and data flows. The arrows in this figure indicate Mbus transactions (BM is target) and the direction of the arrow indicates read (BP allocation) or write (BP release or BP initialization). The CPU accesses the BM to initialize the BPPE and to release freed BPs. The Gigabit Ethernet unit accesses the BM to fetch (allocate) or release BPs. The BM DMA accesses the external memory (shown in Figure 70 as the SDRAM) to initialize the external pool (BPPE) and to copy BPs to/from the internal cache (BPPI).

Figure 70: System Block Diagram and Data Flows



16.2.2 Functional Description

This section describes the operation of the Buffer Management controller.

Software allocates a set of buffers placed in memory (or any other memory mapped space). It allocates a memory area for the buffer pool where the buffer addresses (physical address pointers called buffer pointer) of these buffers are placed sequentially. After the BM controller is enabled, it exclusively controls the buffer pool memory area (the BPPE) as a cyclic array, where read to and write from the internal memory (BPPI) are performed by the BM DMA in batch access (8 or 32 BPs per operation).

Software can initialize BPs either by directly loading the BPPE in external memory before BM is enabled for that pool or indirectly by loading BPs into BPPE via the same mechanism that is used to release BPs, after enabling the BM. BPs cannot be added to a pool by direct loading to BPPE once the pool is enabled.

Copyright © 2014 Marvell

When a unit allocates a BP, it is fetched from the BPPI to achieve low latency. When the number of free BPs in the BPPI drops below the low threshold, the BM pre-fetches (pulls) an additional batch of BPs from BPPE. When a units release BPs to the BM, it stores them in its internal memory (BPPI). When the number of free BPs in the BPPI exceeds the high threshold, the BM controller moves (pushes) a batch (8 or 32)of the BPs into the BPPE.

The BM manages the external memory buffer pool (BPPE) as a cyclic array with read and write pointers. The read pointer points to next free BP, and the write pointer points to where the next released BP should be placed.

During initialization software configures the start and end location of each BPPE and the initial location of the read and write pointers, to indicate the location of available BP sequences. This allows activation with an initial number of BPs, including empty and nearly full BPPEs. The maximal number of BPs that can be initialized in a pool is 32 less then the pool size in BPs.

The BM can manage up to four separate buffer pools. Each pool may be used for a different purpose or application. The BM controller allocates BPs from a specific pool, according to the pool index it receives in the BP fetch request. Each pool can absorb 256 BPs in internal memory

A fetch of free BPs from the BM is done by an Mbus read transaction. The requestor indicates the pool index and the number of BPs in the request, and the BM retrieves the BPs from the BPPI. Release of BPs back to the pool is done by an Mbus write transaction. The requestor indicates the pool index and the number of BPs to be released, and the BM write the BPs back to the BPPI. The BPPI is managed as a cyclic buffer by the BM.

16.2.3 Data Flows

There are three main flows in the BM controller:

- Moving BPs (push, pull) between the external pool (BPPE) and the internal memory (BPPI) by the BM DMA.
- Accesses by requestors that request BP allocation or release from the BM controller.
- Control and configuration accesses towards register file.

Up to 32 BPs can be fetched or released in a single Mbus transaction. The pool index is indicated through address decoding, and the number of BPs are determined by the data size of the transaction.

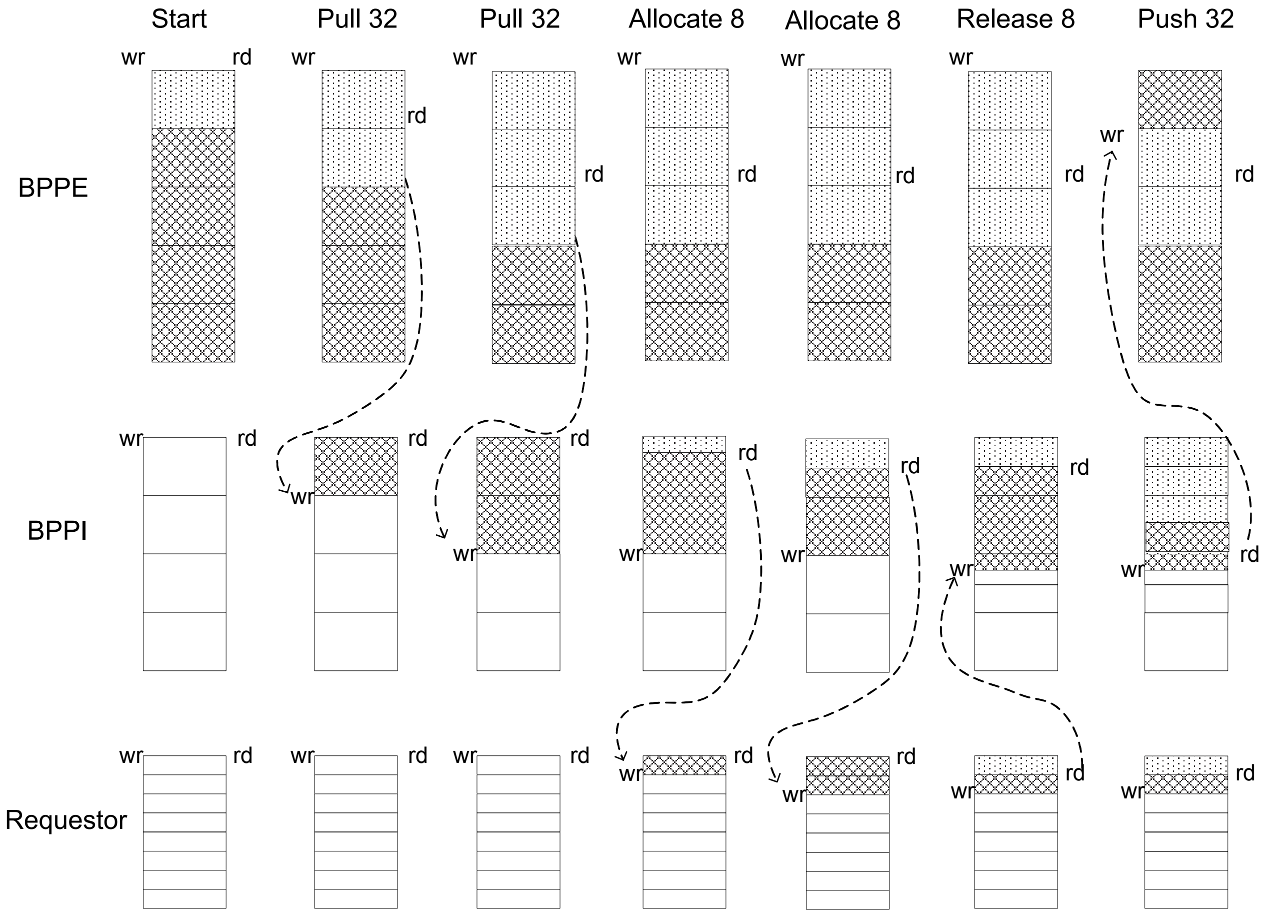
The BM controller is configured by software, and generates interrupts upon hardware events.

BPs are moved between the BM controller internal memory (BPPI) and the BPPE in the Memory by the dedicated DMA. The BM monitors the BPPI current status and contents and, accordingly, moves a set of 8 or 32 BPs in a single Mbus transaction.

Allocate/release transactions are handled in order, and therefore, allocation responses via the Mbus are in the same order that they were received over the Mbus interface. The only exception is that allocate/release requests that were accepted by the BM as Mbus target and not yet processed can be bypassed by DMA transactions that go directly into the BPPI memory.

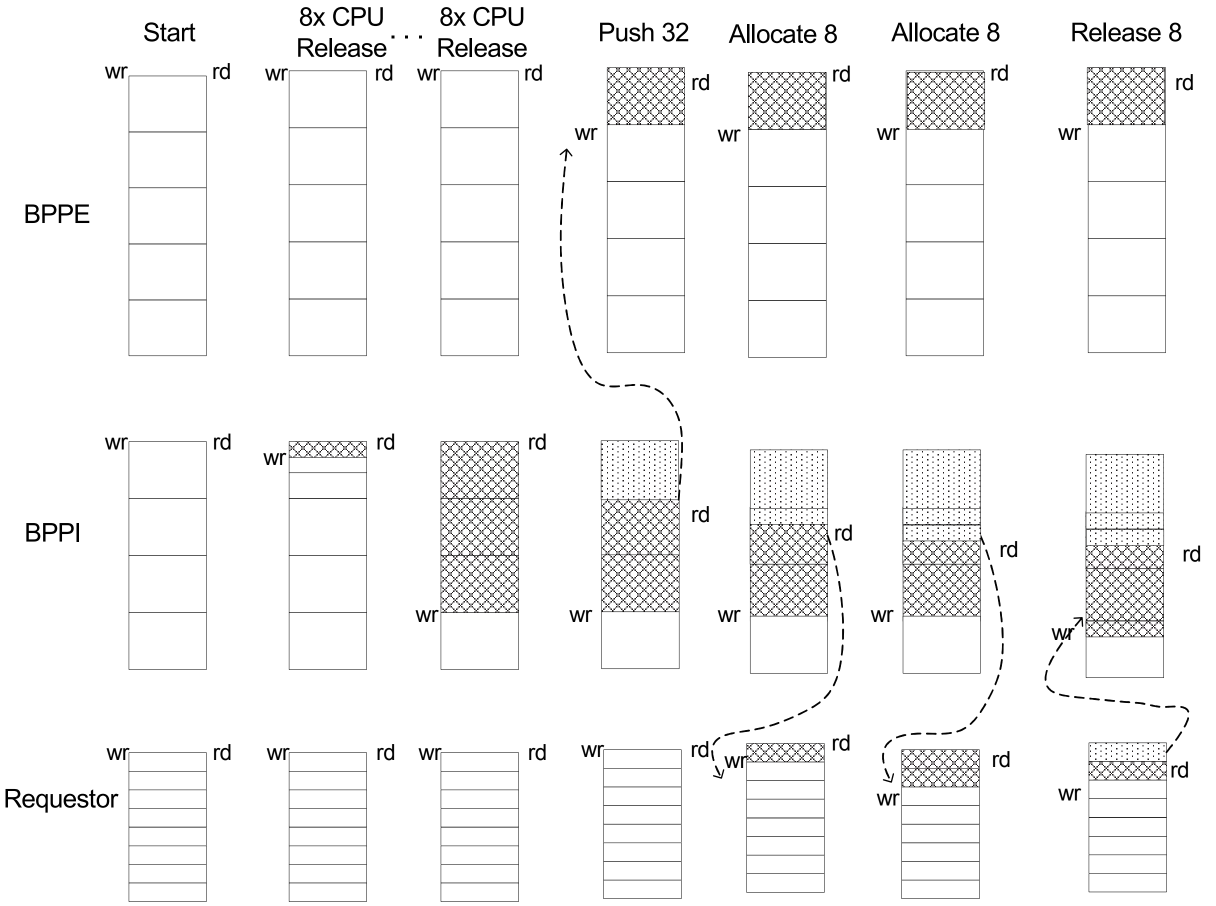
Figure 71 and Figure 72 provide examples of BM pool states under Buffer Management request flows.

Figure 71: Data Flow with Initial Nearly Full BPP



Copyright © 2014 Marvell

Figure 72: Data Flow with Initial Empty BPPE



16.3 Activation and Usage

16.3.1 Initialization of the Buffer Management Controller

The BM controller is configured by the BMCR Register (Table 733 p. 1036).

The BM controller is activated by the <BmStart> field in the BMACTR Register (Table 734 p. 1038). To activate a specific pool, set the <BPP_Enable_x> field in the BMBPPAR_x Registers (see Table 737, Table 741, Table 745, and Table 749). A specific BP pool becomes active when both <BmStart> and <BPP_Enable_x> are set.

Prior to activating the BP controller, program its parameters. Those parameters can**not** be modified while the BM controller is active. The parameters that should be programmed per BP pool are:

- The pool base address in the BMBPPAR_x Registers.
- The read pointer in the BMBPPRP_x Registers (see Table 738, Table 742, Table 746, and Table 750) should be initialized to the pool base address.
- The write pointer in the BMBPPWP_x Registers (see Table 739, Table 743, Table 747, and Table 751) should be initialized to the first empty entry in the pool.
- Set the size of the pool using the <BppSize_x> field in the BMBPPSR_x Registers (see Table 740, Table 744, Table 748, and Table 752).

- Low threshold definition, when the BM DMA copies BPs to the internal memory (BPPI) from the external memory (BPPE), using the <BppiLowThreshold> field in the BMCR Register (Table 733 p. 1036).
- High threshold definition when the BM DMA copies BPs from the internal memory (BPPI) to the external memory (BPPE), using the <BppiHighThreshold> field in the BMCR Register (Table 733 p. 1036).
- Mbus Target ID and Target attributes per pool in the BMXR_Px Registers (see Table 735 and Table 736).

16.3.2 Status and Interrupts

The BM controller reflects its active condition via the <BmStatus> field in the BMACTR Register (Table 734 p. 1037).

Certain events are reflected as interrupts via the BMICR Register (Table 753 p. 1045). Each interrupt can be masked via BMIMR.

Events that cause interrupts:

- Release fail (per pool)
- Allocation fail (per pool)
- BPPE empty (per pool)
- BPPE full (per pool)
- BM stop
- BM pause
- Internal parity error (read from BPPI)
- External parity error (read from BPPE)
- Low number of available BPs (per pool)

The following additional information about the status of the BM controller can be observed:

- Current BPPE read pointer (BMBPPRP register), per pool (see Table 738)
- Current BPPE write pointer (BMBPPWP register), per pool (see Table 739)
- Number of allocated BPs (BMABPS register), per pool (see Table 755 through Table 758)
- Number of released BPs (BMRBPS register), per pool (see Table 759 through Table 761)

16.3.3 Mbus Interface—Initiator and Target

A request for allocating BPs is called a fetch request. It is performed by a read transaction over the Mbus with the BM as the target.

Releasing BPs to the BM is called a release request. It is performed by a write transaction over the Mbus with the BM as the target.

The BM external address is specified in the memory map section. A 512 KB address space is required. The address space should be configured as non-cacheable and non-bufferable in the MMU. The address bits in the address phase are:

- Bit [31:19]: According to BM controller window configuration.
- Bit [18]: Must be cleared.
- Bits [17:10]: Ignored.
- Bits [9:8]: Defines the BP pool index.
- Bits [7:0]: Ignored.

The number of BPs in a transaction is determined by the transaction size (1,2,4,6,8,...,32 BPs supported).

When the BM is an initiator (DMA transactions to BPPE), the Mbus Target ID for outgoing requests towards external memory is defined by the <TrgtID_Px> fields in the BMXR_Px registers.

Similarly, the value of the Mbus attributes is defined by the <XbarAttr_Px> fields in the BMXR_Px registers. Each pool can be given a different attribute and Target ID.

16.3.4 Statistics Counters

- Allocated BPs counter (*clear-on-read*, per pool, saturated register)—number of BPs that were allocated since the last read from this register (BMABPS_<n> register—see Table 755 through Table 758)).
- Released BPs counter (*clear-on-read*, per pool, saturated register)—number of BPs that were released since the last read from this register (BMRBPS_<n> register—see Table 759 through Table 761).

16.3.5 Assumptions and Restrictions

- BPPE base address must be 128 or 32-byte aligned, corresponding to the Mbus burst size configuration. All address configuration registers: BMBPPAR, BMBPPRPR, BMBPPWPR, and BMBPPSR must be configured aligned.
- The size of the BPPE (configured by the BMBPPSR_x Register) must be a multiple of the BM controller burst size, (32B or 128B).
- The read pointer (in BMBPPRP Register) should be initialized to the BPPE base address.
- The write pointer (in BMBPPWP Register) must be initialized to be lower than the size of the BPPE by at least 32 BPs.
- Threshold limitations are: The value in the <BppiHighThreshold> must be greater than the value in the <BppiLowThreshold> by at least 3. However, the <BppiLowThreshold> value must be greater than 2, and the <BppiHighThreshold> value must be less than 14.
- BM controller should be activated (BM start) only after all configurations are set.

16.3.6 Special Cases, Errors, and Error Handling

- BPPI Underflow: The BPPI is **empty** during an allocation request:
  - Regardless of the BPPE state the BM controller returns a response with a *fixed null address (zero address)*.
  - An interrupt is asserted.
- BPPI Overflow: The BPPI is **full** at a release request:
  - The BM does not acknowledge the Mbus request until there is sufficient space in the BPPI to absorb it.
  - An interrupt is generated (This is not an error since the release is eventually completed).
- An allocation/release request to a pool that is not enabled:
  - The BM responds to the allocate request with a null address (as in underflow) and does not acknowledge a release request as in overflow.
  - An interrupt is generated.
- BPPE Full:
  - An interrupt is asserted.
  - BM controller operation continues.
- BPPE Empty: An interrupt is generated.
- Configurable “Low BP Count” - per pool alarm:
  The sum of available BPs in both the BPPE and the BPPIs is compared to a configured value in the <BMFBPAT> register (bits 14:0). When the sum is less then the configured value, a maskable interrupt is asserted.

17 1G/2.5G Ethernet MAC

This section describes the 1G/2.5G Ethernet MAC.

The registers for the 1G/2.5G Ethernet MAC Interface are located in Appendix A.6, Ethernet Networking Controller and MAC Registers, on page 897.

Features

The Ethernet MAC provides the following features:

- IEEE 802.3 compliant MAC layer function
- IEEE 802.3 compliant MII interface
- Marvell® proprietary 200 Mbps MII (MMII) interface
- 1000/2500 Mbps operation—full duplex
- 10/100 Mbps operation—half and full duplex
- EEE—Energy Efficient Ethernet - only supported in RGMII and SGMII modes
- MII symmetric Flow Control: Backpressure for half-duplex operation mode
- RGMII mode
- SGMII mode
- Transmit functions:
  - Programmable values for Inter Packet Gap
  - CRC generation
  - Backoff algorithm execution
  - Error reporting
- Receive functions:
  - CRC check
  - Error report

17.1 Functional Description

The port speed, duplex, and IEEE 802.3 Flow Control can be auto-negotiated, according to IEEE standard 802.3. Backpressure is supported for half-duplex mode when operating at 10/100 Mbps speeds. Each port supports MIB counters.

Layer-2 CRC is always checked on received packets, and can be generated for transmit. This capability increases performance significantly by off-loading these operations from the software.

The port's speed (10, 100, or 1000 Mbps) is auto-negotiated through the PHY and does not require user intervention. Auto-negotiation is according to IEEE standard 802.3. The 1000 Mbps speed operates only in full-duplex mode. The 100 Mbps and 10 Mbps speeds operate either in half- or full-duplex mode, with the selection of the duplex mode auto-negotiated through the PHY without user intervention. With the RGMII/GMII interface, the port only supports symmetric Flow Control.

Note

Auto-Negotiation also can be disabled. When Auto-Negotiation is disabled, the link parameters (link speed and Duplex mode) are forced by the software. The link must be forced down when changing the port speed.

17.2 Retransmission (Collision)

Collision support is integrated into the Ethernet port for half-duplex operation mode. Half-duplex mode is supported in 10 and 100 Mbps speeds only.

A collision event is indicated each time receive and transmit are active simultaneously. When that happens, active transmission is stopped, the jam pattern is transmitted and the collision count for the packet increments. The packet is retransmitted after a waiting period that conforms to the binary backoff algorithm specified in the IEEE 802.3 standard. The retransmit process continues for multiple collision events as long as a limit is not reached. This retransmit limit, which sets the maximum number of transmit retries for a single packet, is defined by the IEEE 802.3 standard as 16.

As long as a packet is being retransmitted, its last descriptor is kept under port ownership. When a successful transmission takes place (i.e. no collision), a status word containing collision information is written to the last descriptor and ownership is returned to the CPU.

If a retransmit limit is reached with no successful transmission, a status word with error indication is written to the packet's last descriptor, and the transmit process continues with the next packet.

It is important to note that collision is considered legal only if it happens before transmitting the 65th byte of a packet. Any collision event that happens after sending the first 64 byte window is known as a *late collision*, and is considered a fatal network error. Late collision is reported to the CPU through the packet status, and no retransmission is done.

Note

Any collision occurring during the transmission of the transmit packet's last four bytes is not detected.

17.3 CRC Generation

Ethernet CRC denotes four bytes of Frame-Check-Sequence appended to each packet.

The port can generate and append CRC to a transmitted packet. CRC generation can be disabled via the <TxCRCDis> field in the Port Configuration Extend (PxCX) Register (i=0–3) (Table 691 p. 996). If disabled, it is the software's responsibility to place the 32-bit CRC at the end of the packet.

If CRC generation is enabled, and the data fetched from memory is known to be erroneous, The packet is sent, and CRC is not generated (causing bad CRC detection at the receiving side).

17.4 Network Interface (10/100/1000 Mbps)

The supported Ethernet MAC interfaces include:

- RGMII
- MII
- If Auto-Negotiation is enabled for Speed only by clearing the <AnDuplexEn> field and the <AnFcEn> field in the Port Auto-Negotiation Configuration Register (i=0–3) (Table 701 p. 1005), the MDC/MDIO Auto-Negotiation takes place using MDC/MDIO pins, as defined in IEEE 802.3 standard.

To support all speeds, the device includes several MAC blocks suited for 10, 100, and 1000 Mbps with the following considerations:

- Support for half-duplex (for 10 and 100 Mbps only) and full duplex (in all speeds)
- Backpressure option in half duplex (for 10 and 100 Mbps only)
- Flow-control option in full-duplex

17.4.1 MII Interface

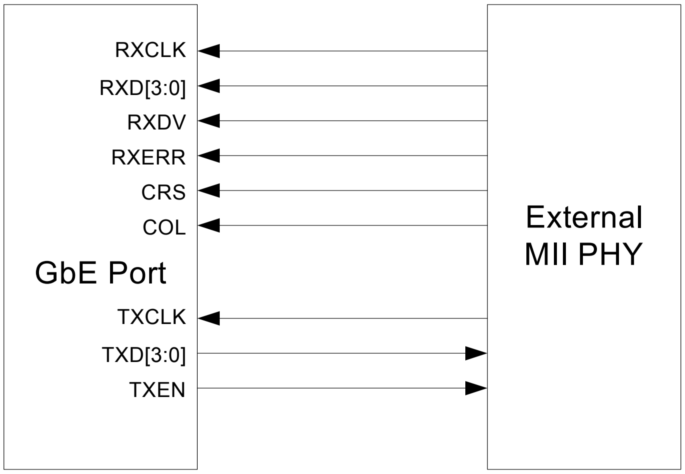
The device MAC allows it to be connected to a 10-Mbps or 100-Mbps network. The device interfaces with an IEEE 802.3 10/100 Mbps MII compatible PHY device. The data path consists of a separate nibble-wide stream for both transmit and receive activities.

The device also support the Marvell® proprietary 200 Mbps MII (MMII) interface.

Depending on the speed of the network, the device can automatically switch between 10- or 100-Mbps operation. Data transfers are clocked by the 25-MHz transmit and receive clocks in 100-Mbps operation, or by 2.5-MHz transmit and receive clocks in 10-Mbps operation. The clock inputs are driven by the PHY. The PHY controls the clock rate based on its configuration, or on the Auto-Negotiation function.

In MII mode, the GbE port receives both RXCLK and TXCLK from the external PHY, as shown in Figure 73.

Figure 73: MII Connection



In MII mode, the port operates at 10/100 Mbps (clock frequencies of 2.5/25 MHz respectively). In MMII mode, the port operates at 200 Mbps (clock frequency of 50 MHz).

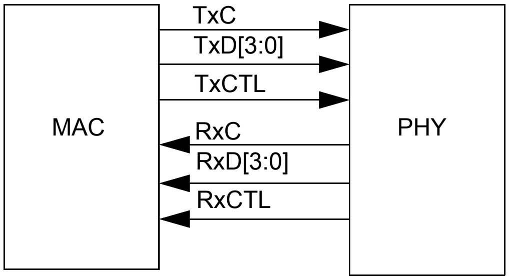
Note

- Only four data bits (RXD[3:0], TXD[3:0]) are used.
- The port TXCLK_OUT output is not used (left NC).

17.4.2 RGMII Interface

The RGMII specification reduces the number of pins required to interconnect the MAC and the PHY to 12 pins, in a cost effective and technology-independent manner. To accomplish this objective, the data paths and all associated control signals are reduced, control signals are multiplexed together, and both edges of the clock are used (see Figure 74). For Gigabit operation, the clocks operate at 125 MHz. For 10/100 operation, the clocks operate at 2.5 MHz or 25 MHz, respectively. The transmit and receive operations are done in full duplex and implement the standard.

Figure 74: RGMII Pin Interconnection Between MAC and PHY



The RGMII interface uses a 125 MHz DRAM clock with 4-bits wide data path. All signals shall be conveyed with positive logic, except where explicitly defined differently. For descriptive purposes, a signal shall be at a logic “high” when it is at a valid voltage level greater than VOH_MIN, and logic “low” when it is at a valid voltage level less than VOL_MAX.

17.4.2.1 RGMII 10/100 Mbps Functionality—Modified MII

This interface can be used to implement 10/100 Mbps Ethernet MII by reducing the clock rate to 25 MHz for 100 Mbps operation, and 2.5 MHz for 10 Mbps. The MAC always generates the SYS_CLK signal and the PHY always generates the Px_RXCLK signal.

During packet reception, Px_RXCLK can be stretched on either the positive or negative pulse to accommodate the transition from the free running clock to a data synchronous clock domain. When the speed of the PHY changes, a similar stretching of the positive or negative pulse is allowed. No glitching of the clocks is allowed during speed transitions.

The MAC must hold Px_TXCTL (TX_CTL) low until the MAC has ensured that Px_TXCTL (TX_CTL) is operating at the same speed as the PHY.

17.4.2.2 Signals Encoding

The RGMII interface is basically a GMII interface running at double data rate:

- GMII_TXD[3:0] is driven on RGMII_TXD[3:0] on the rising edge of RGMII_TXCLKOUT; GMII_TXD[7:4] is driven on RGMII_TXD[3:0] on the falling edge of RGMII_TXCLKOUT.
- GMII_TXEN is driven on RGMII_TXCTL on the rising edge of RGMII_TXCLKOUT
- A logical value of GMII_TXEN XOR GMII_TXERR is driven on RGMII_TXCTL on the falling edge of RGMII_TXCLKOUT.
- GMII_RXD[3:0] is sampled on RGMII_RXD[3:0] on the rising edge of RGMII_RXCLK; GMII_RXD[7:4] is sampled on RGMII_RXD[3:0] on the falling edge of RGMII_RXCLK.
- GMII_RXDV is sampled on RGMII_RXCTL on the rising edge of RGMII_RXCLK.
- A logical value of GMII_RXDV XOR GMII_RXERR is sampled on RGMII_RXCTL on the falling edge of RGMII_RXCLK.

17.4.2.3 In-Band Status

To ease detection of the link status, speed, and duplex mode of the PHY, inter-frame signals are placed onto the Px_RXD[3:0] signals. CRS is indicated when, simultaneously, `RX_DV = True` or `RX_DV = False`, `RX_ER = True`, and a value of FF binary exists on the Px_RXD[3:0] bits.

Collision is determined at the MAC when `TX_EN = True`, while either Px_CRS or P0_RXDV are true. The PHY does not assert Px_CRS as a result of Px_TXEN being true.

17.5 Input Signal Forcing

The device implements an option to force the de-assertion value for the following input signals:

- RXERR
- CRS
- COL

Use this option when connecting to a device that does not support these signals (for example, a switch). Forcing the de-assertion value eliminates the on-board pull up/down on these signals, and saves the input pins that may be used for other purpose, such as GPIO.

To enable this feature set the Port Serial Control0 (PSC0) Register (i=0–3) (Table 693 p. 1000)

17.6 Auto-Negotiation in Different MII/GMII/RGMII Modes

The device implements the standard IEEE Auto-Negotiation, using the Serial Management Interface (SMI), for the following:

- Detect Link status
- Duplex: half- and full-duplex operation
- Flow-control for full-duplex
- Speed

To implement speed Auto-Negotiation, set the <AnSpeedEn> field in the Port Auto-Negotiation Configuration Register (i=0–3) (Table 701 p. 1006) to 0.

To switch between GMII and MII modes, see the <SetMIISpeed> field and the <SetGMIISpeed> field in the Port Auto-Negotiation Configuration Register (i=0–3) (Table 701 p. 1006).

Note

The registers and bits referred to in this section (for example, registers 4, 5, and 15 and bit 1.8) are PHY device registers.

The device continuously reads the PHY register 1 to determine the link status, and also to determine whether or not bit 1.8 in PHY register 1 is set.

When exiting from reset, or when the link changes from up to down, the device advertises its Flow Control ability (if Auto-Negotiation for Flow Control is enabled by the <AnFcEn> field in the Port Auto-Negotiation Configuration Register (i=0–3) (Table 701 p. 1005).

The device reads register bit 1.8. If this bit is reset, then the PHY does not support 1000 Mbps.

If bit 1.8 is set, the PHY supports 1000 Mbps (but the speed may still resolve to 10 or 100 Mbps at the end). The device then continues to read register 15 to determine whether the PHY is 1000baseX-capable or 1000baseT capable. If it is 1000baseX, then the device regards the multiplexed speed as 1000 Mbps only, and follows the IEEE 802.3 rules for duplex and flow-control Auto-Negotiation. If it is 1000baseT capable, then the device follows the IEEE 802.3 rules to resolve the speed, the duplex mode and the Flow Control.

After Auto-Negotiation is complete, the device resolves negotiated modes of operation. These values update the Port Status register fields and affect the Network port operation.

17.7 Data Blinder

The port data blinder is the time period in which the port does not look at the wire to determine if it is necessary to defer a pending transmission, due to receive activity.

The port data blinder is 32 bit time.

17.8 Inter-Packet Gap

The Inter-packet Gap (IPG) is the idle time between two successive packets from the same port. The default (from the standard) is 96 ns.

Note Marvell does not recommend reducing the IPG setting in violation of the IEEE standards. Reducing the IPG can improve test scores but can create Ethernet compatibility problems.

Use the <IPG> field in the Port MAC Control Register3 (i=0–3) (Table 705 p. 1011) to set the IPG size.

17.9 Backpressure Mode

Backpressure is supported only when working in 10/100 Mbps speed and in half duplex mode.

The Backpressure algorithm is enabled by setting the <ForceBPMode> field in the Port Serial Control0 (PSC0) Register (i=0–3) (Table 693 p. 1000).

For a port in Backpressure mode, the port waits until the medium is idle and then transmits a JAM pattern of 1536 bytes. The IPG between two consecutive JAM patterns is 4 bytes, and between last transmitted packet to first JAM is 12 bytes.

When a port in Backpressure mode has a pending packet for transmission, it halts the transmission of the JAM pattern. After an IPG is completed, the port transmits the packet. If the port remains in Backpressure mode, it resumes the JAM pattern transmission, following the packet transmission.

17.10 Flow Control

The device implements the IEEE 802.3 Flow Control in full-duplex mode, including full Auto-Negotiation.

Auto-Negotiation for Flow Control is enabled for:

- The multiplexed interface
- PHYs that have an SMI interface (MII or GMII PHYs)

The behavior of the device is determined by the <RxFcEn> field and the <TxFcEn> field in the Port Status Register0 (i=0–3) (Table 717 p. 1019) and the <ForceFCMode> field in the Port Serial Control0 (PSC0) Register (i=0–3) (Table 693 p. 1001).

The CPU may write to the PSCR <ForceFCMode> field when Flow Control operation is enabled in the <RxFcEn> and <TxFcEn> fields (which may be result either of Auto-Negotiation resolution for Flow Control, or manual setting by the CPU to enable flow-control operation, which is then reflected in the <RxFcEn> and <TxFcEn> fields).

The CPU must trigger the initiation of pause disable transmission when detecting that it cannot keep up with the received traffic (this is typically done by monitoring the queue filling process).

When the CPU suspects that it may not be able to provide enough resources to the port, it must trigger the beginning of Flow Control packets by writing 01 value to <ForceFCMode>. When resources are made available, the CPU must again write a 00 value to <ForceFCMode> to trigger transmission of pause enable packet. The CPU response time to congestion cases would determine if and how many packets may be lost on receive.

The value in the <RxFcEn> and <TxFcEn> fields can be set from the following:

- CPU programming
- Result of Auto-Negotiation for Flow Control according to IEEE 802.3 standard in all modes

When in MII or GMII modes, and Auto-Negotiation for Flow Control is enabled, the device writes to the relevant advertisement register in the PHY on the following events:

- Exiting from reset
- Upon link fail detection (link changed from up to down)

Auto-Negotiation for Flow Control for 1000BASE-X PHY advertises that the device supports Symmetric Flow Control only according to the IEEE 802.3 standard.

The advertised ability of Pause support depends on the setting of the <Pause_Adv> field in the Port Auto-Negotiation Configuration Register (i=0–3) (Table 701 p. 1005) as follows:

- When set, the device advertises symmetric capability for Pause.
- When reset, the device advertises No Pause capability.

17.10.1 Pause Receive Operation

When the device receives a Pause packet, it avoids transmitting a new packet for the period of time specified in the received Pause packet.

The pause quantum is 512 bits time according to the port speed.

A received packet is recognized as Flow Control if it was received without errors and it has the following characteristics:

DA = 01-80-C2-00-00-01 and type=88-08 and MAC_Control_Opcode=00-01.

A packet received by the device from the network port that is identified as a Pause packet is always discarded, even if the Pause function is disabled.

17.10.2 Pause Transmit Operation

For enabling a Pause Transmit operation, or either enabling or disabling, the <TxFcEn> field in the Port Status Register0 (i=0–3) (Table 717 p. 1019) must be in the active state.

It is the CPU’s responsibility to determine that packets are in danger of being dropped by the receive port, according to the dynamic availability of resources. One way of doing it is monitoring how much of the descriptor chain is filled up by the port and how much is left. Another aspect is memory bandwidth should be allocated to the port via the Mbus “pizza arbiter” to avoid bandwidth shortage for the gigabit port.

Note

This mechanism does not provide hardware guarantee of zero frame-loss. This mechanism depends on CPU functionality in triggering it dynamically.

When the CPU suspects that it may not be able to provide enough resources to the port, it must trigger the beginning of Flow Control, pause-disable packets transmission by writing a 01 value to the <ForceFCMode> field in the Port Serial Control0 (PSC0) Register (i=0–3) (Table 693 p. 1001). The transmit port will schedule transmission of a pause-disable frame (timer=0xFFFF) at the next possible frame boundary and will automatically retransmit it according to the setting of the periodic Flow Control timer, as long as the value in the <ForceFCMode> field remains 01 (see the Periodic Flow Control Clock Divider Register (i=0–3) (Table 698 p. 1003)).

The other link partner is expected to stop packet transmission upon receiving the Flow Control disable packets, and the retransmission of them guarantees refreshing that indication continuously.

When resources are made available, the CPU must write a 00 value to <ForceFCMode>. This will trigger transmission of a single pause enable packet (timer = 0x0000) that would enable the other link partner to resume packet transmission.

When transmitting a pause packet, the port address is put into the source address field. The 48-bit port address is located in the MAC Address Low and the MAC Address High registers.

Note

When the link goes down, the <ForceFCMode> is always cleared to 00 (no pause disable frames are sent).

17.11 Serial Management Interface (SMI)—Ethernet MAC

The port MAC contains a Serial Management Interface (SMI) for working with an external networking device, such as GbE PHYs.

The SMI allows control and status parameters to be passed between the device and the PHY (parameters specified by the CPU) using one serial pin (MDIO) and a clocking pin (MDC), reducing the number of control pins required for PHY mode control. Typically, the device continuously queries the PHY device for the link status, without CPU intervention. The PHY addresses for the link query are programmable in the PHY Address Register (i=0–3) (Table 575 p. 922).

A CPU connected to the device can write/read to/from all PHY addresses/registers. The SMI allows the CPU to have direct control over compatible PHY device via the SMI Register (i=0–3) (Table 576 p. 922). This control allows the driver software to place the PHY in specific modes such as Full-Duplex, Power-Down, or 1000-speed selection. It also helps control the PHY device's Auto-Negotiation function, if it exists. The CPU writes commands to the SMI register and the device reads or writes control/status parameters to the PHY device via a serial, bi-directional data pin called MDIO. These serial data transfers are clocked by the device MDC clock output.

Link Detection and Link Detection Bypass (ForceLinkPass*)

Typically, the device continuously queries the PHY device for its link status, without CPU intervention. The PHY address used for the link query is determined by the PHY Addresses register, and it is programmable, where the default value is 8 for Port0, 9 for Port1. The device reads register 1 from PHY and updates the internal link bits according to the value of bit[2] of register 1. In the case of "link is down" (bit[2] is 0), that port enters link test fail state. In this state, all of the port's logic is reset. The port exit from link test fail state only when the "link is up", bit[2] of register 1 is read from the port's PHY as 1.

The device offers the option to disable the link detection mechanism by forcing the link state of the interface to the link test pass state. This is done by forcing the register bit, and then the link status of the port remains in the "link is up" state regardless of the Interface-PHY's link bit value. The link status of the Interface-PHY can be read through the SMI from the PHY devices (register 1, bit[2]).

17.12 Port MIB Counters

The device incorporates a set of management counters, the MAC MIB Counters. They provide the necessary counters that support MAU, IEEE 802.3, and EtherLike MIB. Each port has a set of counters that reside in consecutive address spaces. Some counters are 64-bits wide.

The counters are meant to provide management software to support:

- IEEE 802.3 DTE Management objects
- Ethernet-like interface MIB: RFC 2665
- Interface MIB: RFC 2863
- Remote Network Monitoring (RMON) groups 1-4: RFC 2819

Note

The MAC MIB counters are not intended to be used for Bridge MIB nor for SMON MIB.

Table 77 summarizes the terms used in the definition of the counters.

Table 77: Definitions for MAC MIB Counters

Term	Definition
Collision Event	A collision has been detected before 576-bit times into the transmitted packet after Px_TXEN is asserted. Relevant to 10 Mbps and 100 Mbps speeds in half-duplex mode only.
Late Collision Event	A collision has been detected after 576-bit times into the transmitted packet after Px_TXEN. Relevant to 10 Mbps and 100 Mbps speeds in half-duplex mode only.
Excessive Collision Event	When a packet to be transmitted suffers from 15 consecutive collision events, therefore, it ought to be dropped according to the IEEE 802.3 standard. Relevant to 10-Mbps and 100-Mbps speeds in half-duplex mode only.
MRU	Maximal Receive Unit: A programmable parameter that sets the maximal length of a valid received packet.
Rx Error Event	The Receive Error signal/symbol was asserted while a frame is received.
CRC Error Event	This event occurs whenever an Ethernet frame is received and the following conditions are satisfied: 1. Packet data length is between the Minimum Frame Size - and the MRU byte size inclusive (that is, it is a valid packet data length according to the IEEE standard). 2. Packet has an invalid CRC. 3. Collision Event has not been detected. 4. Late Collision Event has not been detected. 5. Rx Error Event has not been detected.
Undersize packet	An Ethernet frame satisfying *all* of the following conditions: 1. Packet length is less than Minimum Frame Size bytes. 2. Collision Event has not been detected. 3. Rx Error Event has not been detected. 4. Packet has a valid CRC.
Fragment	An Ethernet frame satisfying *all* of the following conditions: 1. Packet data length is less than 64 bytes, OR a packet without a Start Frame Delimiter (SFD) and the packet is less than 64 bytes in length. 2. Collision Event has not been detected. 3. Rx Error Event has not been detected. 4. Packet has an invalid CRC.
Oversize packet	An Ethernet frame satisfying *all* of the following conditions: 1. Packet length is more than the MRU byte size. 2. Collision Event has not been detected. 3. Late Collision Event has not been detected. 4. Rx Error Event has not been detected. 5. Packet has a valid CRC.

Table 77: Definitions for MAC MIB Counters (Continued)

Term	Definition
Jabber	An Ethernet frame satisfying *all* of the following conditions: 1. Packet data length is greater than the MRU. 2. Packet has an invalid CRC. 3. Rx Error Event has not been detected.
Tx Error Event	An internal error event in the transmit MAC. This is a very rare situation and when it happens, it means that there the system is misconfigured.
Bad frame	An Ethernet frame that has one of the following conditions met: CRC Error Event, Undersize, Oversize, Fragments, Jabber, Rx Error event and Tx Error Event.
MAC Control Frame	An Ethernet frame that is not a bad frame and has a value of 88-08 in the EtherType/Length field.
Flow Control Frame	A MAC Control Frame with an opcode equal to 00-01.
Good Flow Control Frame	A Flow Control frame with: 1. MAC Destination equal to 01-80-C2-00-00-01 2. 64-byte length
Bad Flow Control Frame	All Flow Control frames that are not good Flow Control frames
Good frame	An Ethernet frame that is not a bad frame NOR a MAC Control frame

Figure 75 and Figure 76 illustrate the terms defined above:

Figure 75: Ethernet Frame Classification

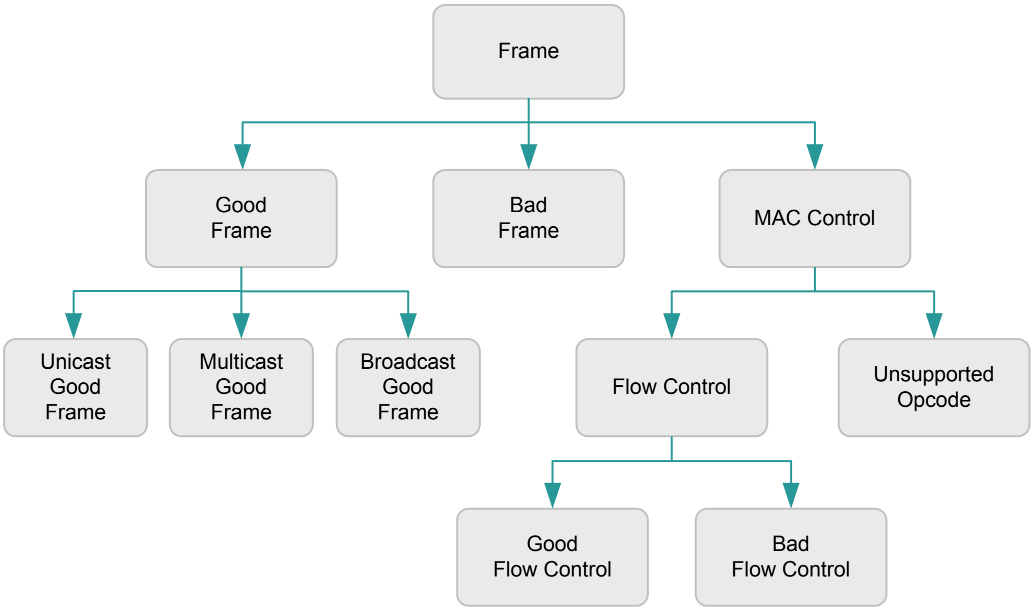
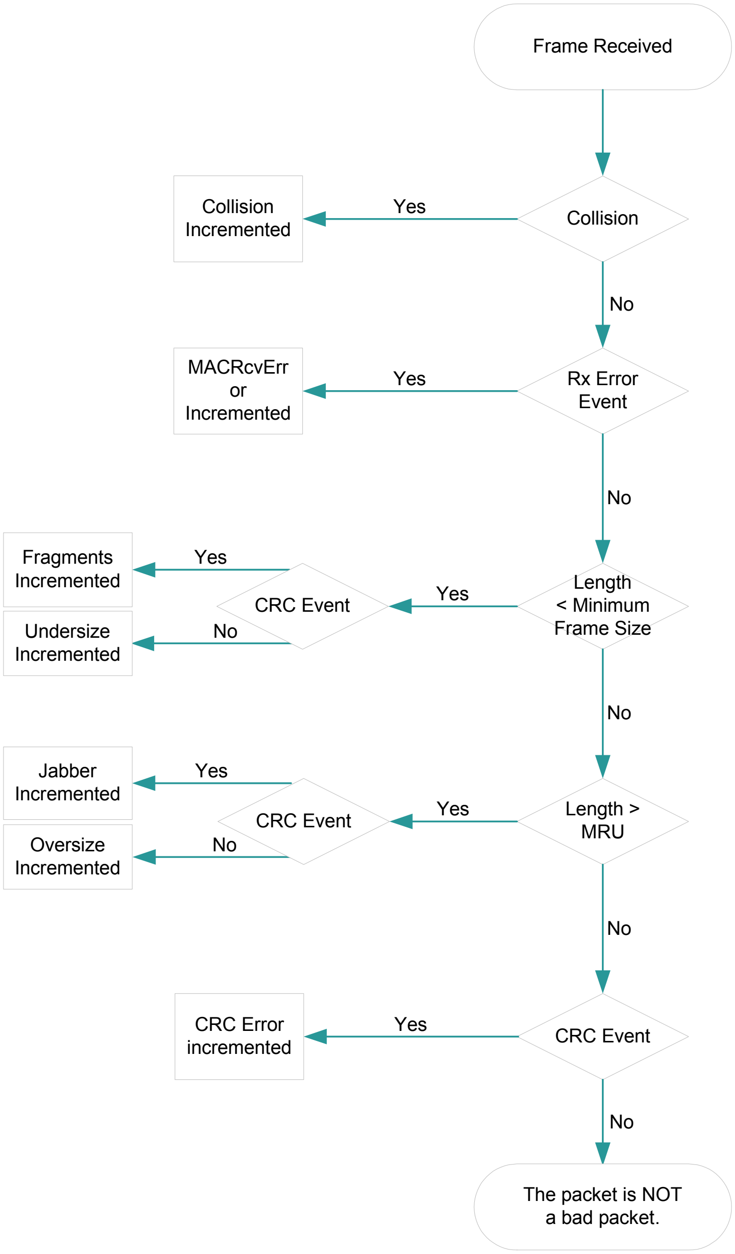


Figure 76: Bad Frame Procedure



The counters initialize to 0 immediately after reset. Most counters are 32-bits. However, the Good Bytes received and Bytes Sent are 64-bit counters that should be read in two separate cycles.

The 64-bit counters should be read from the low address and the next read from the MIB counters must be from the high address of the same counter. The MIB interface will always return the high counter data on a read from the MIBs following a read from the low address of a 64-bit counter.

In addition to the per port counters, there are additional counters that count filtered frames for example, for MAC address lookup results. These counters are in the Port Rx Discard Frame Counter

Copyright © 2014 Marvell

(PxDFC) Register (i=0–3) (Table 606 p. 942) and the Port Overrun Frame Counter (PxOFC) Register (i=0–3) (Table 607 p. 943). In conjunction with the counters block, they provide the information on total frames received. Also refer to Table 731, Gigabit Ethernet MAC MIB Counters, on page 1033.

18 Precise Time Protocol (PTP)

This section describes the functionality for Precise Time Protocol (PTP) Core hardware.

The Precise Time Protocol defines a method of transporting time-of-day information across a network of devices supporting this protocol. There are several industrial, consumer, and enterprise applications for PTP. For example, in consumer space, IEEE 802.1 AVB (Audio Video Bridging) defines the usage of PTP for transporting reliable audio video content across compliant equipment. PTP adds time information to nodes connected via Ethernet. This is in contrast to the IEEE 802 standard Ethernet that is an asynchronous interface where end stations do not operate using a common time base and neither do they have a concept of time.

The PTP allows multiple nodes within a given network to have a common notion of the time-of-day and also to be able to compute frequency offsets with respect to a master clock in the network.

There are 3 types of PTP frames that are supported in the device:

- IEEE 802.1AS frames on Layer 2 Ethernet
- IEEE 1588 frames on Layer 2 Ethernet
- IEEE 1588 frames on Layer 4 UDP

For IEEE 802.1AS and IEEE 1588 over Layer 2, the detection mechanism uses a special EtherType. For IEEE 1588 over Layer 4 UDP, the detection mechanism operates by checking the UDP destination port value. Once a PTP frame has been detected, event messages need to be time stamped in hardware. However, all PTP frames need to be forwarded to the CPU for further processing. Once the ingress PTP frames have been processed by the CPU, certain types of PTP frames get forwarded to other ports that again need to be detected and time stamped in hardware.

The VLAN tag of the incoming PTP frame is detected by hardware.

Using a clock selection algorithm, the PTP firmware computes the frequency and phase offset values, with respect to the best clock available in the network (labelled as the *grand master*). The computed offsets can optionally be used to control the hardware clock and time-of-day counter.

The types of frames (PTP events) that required time stamping by the hardware is configurable.

For every time-stamped frame, the sequence number from the PTP common header is captured along with the time stamp, in hardware for easier protocol software correlation.

Both IPv4 and IPv6 are supported by PTP, however, PTP does not support IPv4/IPv6 options.

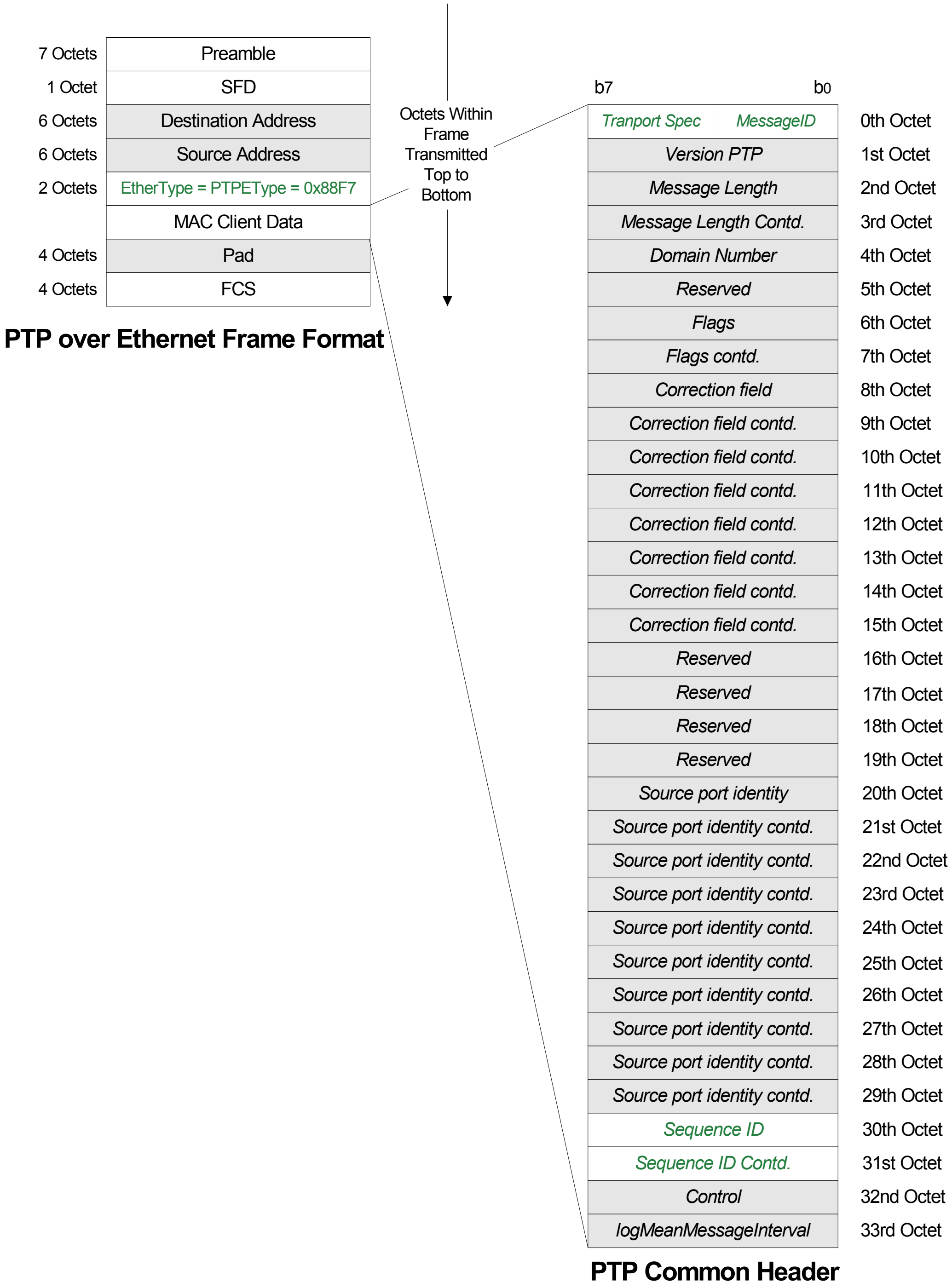
The Precise Time Protocol registers are located in Appendix A.8, Precise Time Protocol (PTP) Registers, on page 1052.

Copyright © 2014 Marvell

18.1 Frame Format

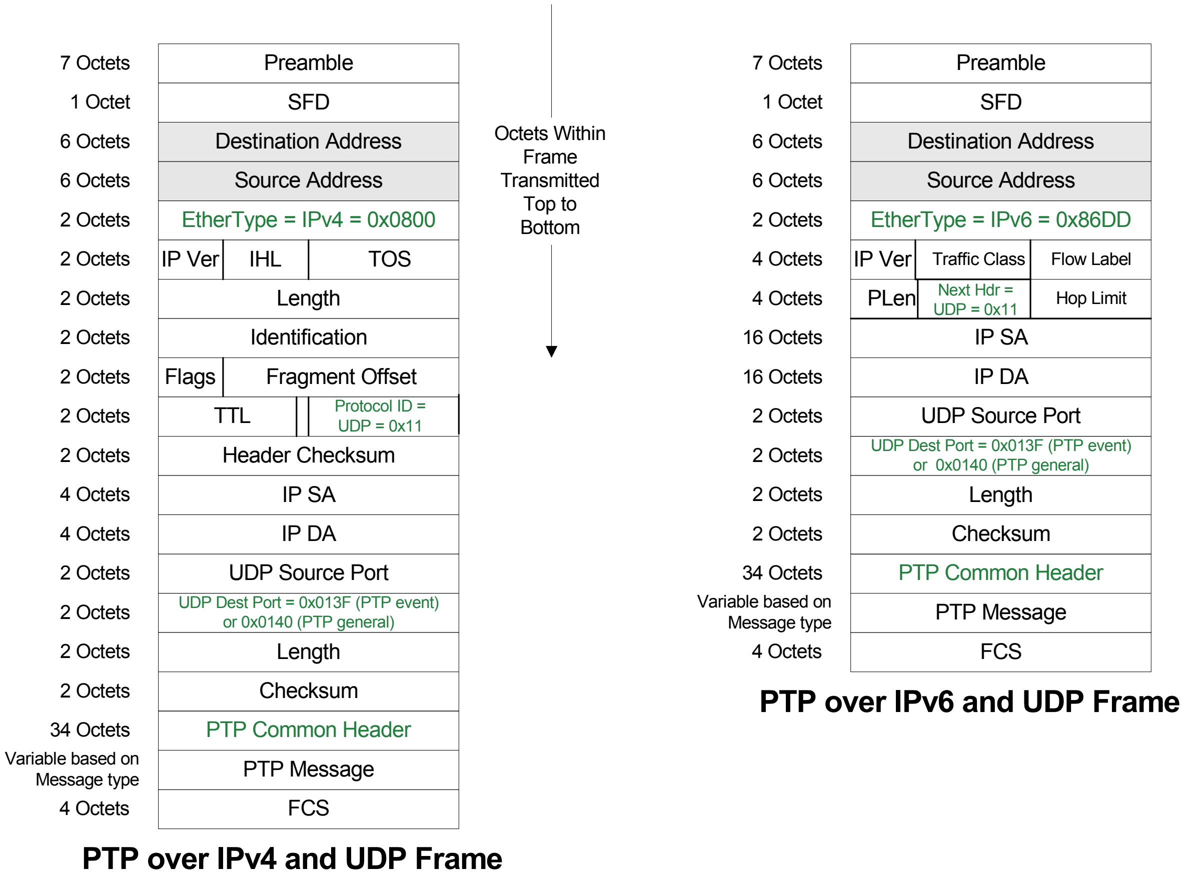
The PTP frame format is shown in Figure 77.

Figure 77: PTP Common Header Format



The PTP over UDP frame format, for both IPv4 and IPv6, is shown in Figure 78.

Figure 78: PTP over UDP Frame



7 Octets	Preamble
1 Octet	SFD
6 Octets	Destination Address
6 Octets	Source Address
2 Octets	EtherType = IPv4 = 0x0800
2 Octets	IP Ver | IHL | TOS
2 Octets	Length
2 Octets	Identification
2 Octets	Flags | Fragment Offset
2 Octets	TTL | Protocol ID = UDP = 0x11
2 Octets	Header Checksum
4 Octets	IP SA
4 Octets	IP DA
2 Octets	UDP Source Port
2 Octets	UDP Dest Port = 0x013F (PTP event) or 0x0140 (PTP general)
2 Octets	Length
2 Octets	Checksum
34 Octets	PTP Common Header
Variable based on Message type	PTP Message
4 Octets	FCS

PTP over IPv4 and UDP Frame

7 Octets	Preamble
1 Octet	SFD
6 Octets	Destination Address
6 Octets	Source Address
2 Octets	EtherType = IPv6 = 0x86DD
4 Octets	IP Ver | Traffic Class | Flow Label
4 Octets	PLen | Next Hdr = UDP = 0x11 | Hop Limit
16 Octets	IP SA
16 Octets	IP DA
2 Octets	UDP Source Port
2 Octets	UDP Dest Port = 0x013F (PTP event) or 0x0140 (PTP general)
2 Octets	Length
2 Octets	Checksum
34 Octets	PTP Common Header
Variable based on Message type	PTP Message
4 Octets	FCS

PTP over IPv6 and UDP Frame

18.2 PTP Frame Time Stamping Clock

The PTP core logic receives a reference clock input (from the PTP time stamp clock) for time stamping purposes. This clock can be a free 125 MHz running clock available in the chip or it can be a PTP synchronized <PTPClkSelect> field in the PTP Clock Select Register (Table 767 p. 1055) clock (PTP_CLK), that is driven into the device, as defined by the.

The benefits of synchronization are more prominent for endpoint devices than for standard AVB bridges. The definition of synchronized clock is when a free running clock's frequency is adjusted based on the frequency offset computed by the PTP software (this is the frequency difference between the PTP Grand Master node and the PTP slave node).

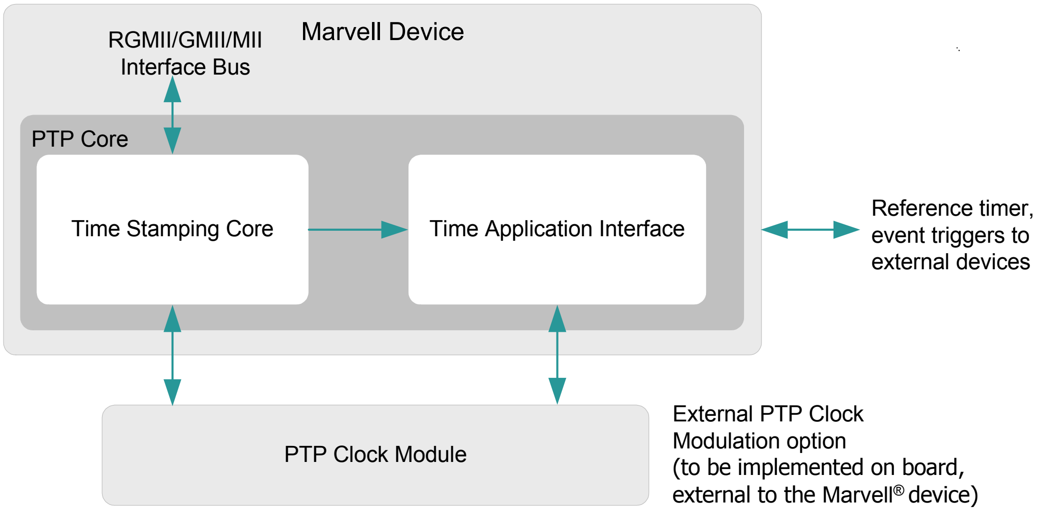
The clock frequency and/or phase adjustment is expected to be handled in an external device.

18.3 Pipeline Stages

The PTP core is divided into two main blocks (see Figure 79):

- Time Stamping Core (see Section 18.4)
- Time Application Interface (see Section 18.5)

Figure 79: PTP 2.1 Pipe Block Diagram



Note The PTP Clock Module, which is a DSP circuit used to adjust the frequency and phase offset of the clock with respect to the grand master clock, is not part of this Marvell® device.

18.4 Time Stamping Block

The time stamping block is a plug-in module snooping the MII/GMII/RGMII interface between the MAC and the PHY as shown in the Figure 80. All incoming frames go through the PTP Frame Detection flow to determine if the frame is a PTP frame or not. Once the frame is determined to be a PTP frame, fields from the PTP common header (shown in Figure 77, PTP Common Header Format, on page 321) are extracted to determine if the PTP frame needs time stamping or not.

The PTP logic captures the time stamp for every incoming frame. Once the frame has been determined to be a PTP event message (which requires time stamping), the time stamp, along with the SequenceID (from the PTP common header) is captured in the registers. The hardware optionally generates an interrupt.

- For incoming frames on Arrival0, this capture takes place in the following registers:
  - PTP Port Status0 Register
  - PTP Port Status1 Register
  - PTP Port Status2 Register

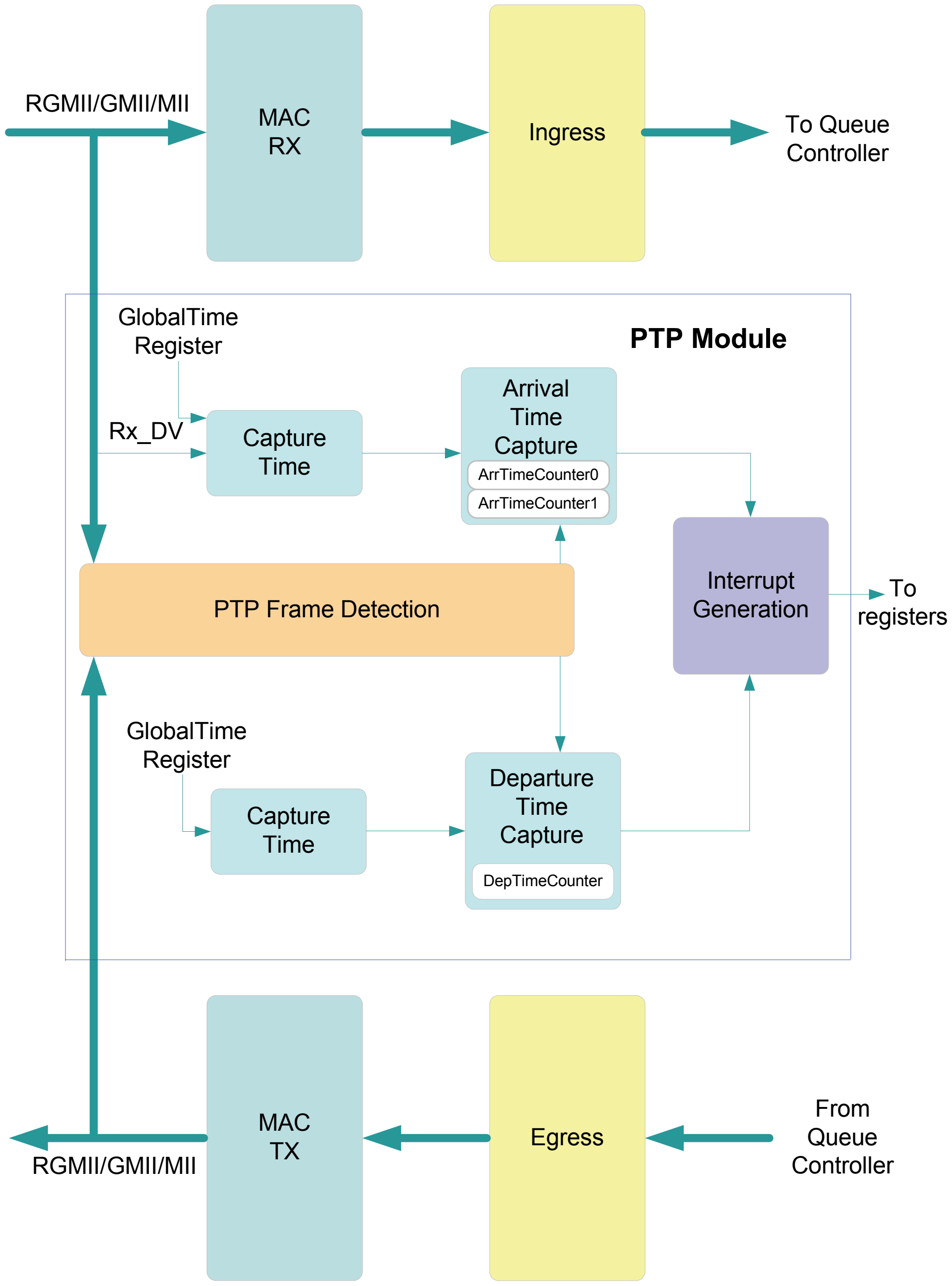
- For incoming frames on Arrival1, this capture takes place in the following registers:
  - PTP Port Status3 Register
  - PTP Port Status4 Register
  - PTP Port Status5 Register
- For all outgoing frames, this capture takes place in the following registers:
  - PTP Port Status6 Register
  - PTP Port Status7 Register
  - PTP Port Status8 Register

PTP frames go through normal MAC SA and DA processing. Since it is expected that a reserved MAC address is used in PTP frames, the MAC address lookup would result in a CPU destination port. There is no change in the logic for sending these frames to the CPU.

Rationale for Two Arrival Counters and One Departure Counter

The Sync/Follow-up messages flow directly from the Grand Master to all the PTP Slave nodes. PDelayReq/PDelayResponse messages are exchanged between link partners. The timing between these two sets of event messages is not controllable. Thus, the hardware logic must be designed for back-to-back PTP event messages that need time stamping. Each of these event message types can be configured to be captured in either Arrival Counter 0 or Arrival Counter 1, using the configuration parameter <TSArrPtr> field in the PTP Global Configuration2 Register (Table 770 p. 1057).

Figure 80: Time Stamping Pipeline Stages



18.5 Timing Application Interface (TAI) Block

The Precise Time Protocol provides both frequency and time-of-day with respect to the PTP Grand Master for the entire PTP network. In a given endpoint device (media talker or a media listener), once the PTP Grand Master-aware clock and time are available, this data must be transmitted over to the rest of the end-user applications, without loss of accuracy. For example, if a Digital Video Recorder (DVR) is the end device, the network clock and time need to be transported to the Video device and/or Storage device and the host processor seamlessly. This ensures that when the media is played out of the DVR, the presentation time of the content is required to be carried through the network between the media talker and the media listener.

18.5.1 Interfaces Supported by the TAI Block

There are three types of interfaces that this block supports:

- An event capture interface that captures the time of an event signaled by the requesting entity.
- A trigger-generate interface that causes an event to be signaled at a time specified by the requesting entity.
- A trigger-clock-generator interface that causes a periodic sequence of indications to be generated, with the rate specified by the requestor.

18.5.2 Precise Time Protocol—Time Application Interface (TAI)

The TAI block supports features required for the purposes described above. This block utilizes two signals to offer various services:

- PTP_EVENT_REQ input signal (event request)
- PTP_TRIG_GEN output signal (trigger generate)

Using these signals, there are several functions that this block supports:

- Event pulse capture
- Multiple-event counter
- Trigger pulse generate, with pulse width control
- Trigger clock generate, with digital clock compensation
- PTP global time increment and/or decrement

18.5.2.1 Event Pulse Capture Interface

In many IEEE 1588 applications, such as industrial automation, it is important to precisely capture the time at which a particular event occurred. The event is defined by a low-to-high transition on an external input signal called PTP_EVENT_REQ. The event time is captured in the <EventCap Register> field in the TAI Global Status2 Register (Table 795 p. 1076). This field is validated by <EventCapValid> field in the TAI Global Status1 Register (Table 794 p. 1076).

The captured event time register, TAI Global Status1 Register (Table 794 p. 1076), must be read by software and the <EventCapValid> field must be cleared before the hardware captures another event. If two back-to-back events occurred before the software read the results of the first event, an error indication is set in <EventCapErr>. To have the hardware overwrite the <EventCap Register> field rather than generate an interrupt, then set <EventCapOv> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1070).

Once an event has been captured, the software can generate an interrupt, if both the <EventCapIntEn> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1071) and the <EventInt> field in the TAI Global Status1 Register (Table 794 p. 1076) are set.

The maximum jitter associated with capturing the PTP_EVENT_REQ signal pulse is the value set in the <TSClkPer> field in the TAI Global Configuration Register, PTP Port = 0xE and 0xF (Table 787 p. 1072). The minimum pulse width of the PTP_EVENT_REQ signal must be 1.5 times the value set in the <TSClkPer>. For the hardware logic to detect distinct events on the PTP_EVENT_REQ signal, the minimum gap between two events needs to be 125 ns.

> **Note** If the hardware registers are being accessed by the CPU at the same time the hardware logic is trying to update an <EventCap Register> field in the TAI Global Status2 Register (Table 795 p. 1076), the <EventCapErr> field in the TAI Global Status1 Register (Table 794 p. 1076) is set to 0x1.

18.5.2.2 Multiple Event Counter Function

Similar to the Event Pulse capture interface described in Event Pulse Capture Interface, if multiple events need to be captured, for an application to detect how many times a particular event occurred on the PTP_EVENT_REQ input signal, the <EventCtrStart> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1071) must be set to a 0x1 and the <EventCapOv> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1070) must set to a 0x1.

The PTP core is capable of capturing up to 255 events in the <EventCapCtr> field in the TAI Global Status1 Register (Table 794 p. 1076).

> **Note** In the multiple event counter mode, the <EventCap Register> field in the TAI Global Status2 Register (Table 795 p. 1076) indicates the time stamp value for the last captured event register.

18.5.2.3 Trigger Pulse Generate Function

In many PTP applications, the time of day computed in PTP needs to be distributed in some form to the rest of the node. One commonly used method is generating a pulse whenever the PTP Global Time matches a certain configured value. The pulse is output on a PTP_TRIG_GEN output signal.

In the PTP core, configure the Trigger Pulse Generate function:

1. Set the <TrigGenReq> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1072) to 0x1.
2. Set the <TrigMode> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1072) to 0x1.
3. Configuring the <TrigGenAmt> field in the TAI Global Configuration0 Register (Table 788 p. 1073) with the length of time value after which the pulse needs to be generated.

The PTP Global Timer is compared to the value in the <TrigGenAmt> field, and upon a match, a pulse signal is generated on the PTP_TRIG_GEN output signal.

Optionally, after generating the PTP_TRIG_GEN pulse, the CPU can be notified by setting the <TrigGenIntEn> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1071), and along with the pulse output, the <TrigGenInt> field in the TAI Global Status0 Register (Table 793 p. 1075) is set. Upon receiving the interrupt, the CPU must clear the interrupt bit.

In Pulse mode The pulse width of the output signal can be controlled by the <PulseWidth> field in the TAI Global Configuration2 Register (Table 790 p. 1073).

> **Note** Do not set the <PulseWidth> to 0x0.

18.5.2.4 Trigger Clock Generate Function

Similar to the trigger pulse generation function described Trigger Pulse Generate Function, the same set of registers can be used to generate a periodic clock. The value specified in <TrigGenAmt> field is used to generate the base period of the clock output. For this functional mode, the <TrigMode> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1072) field must be set to a 0x0, and the <TrigGenReq> must be set to a 0x1.

The output clock can be compensated for by configuring the <TrigClkComp> field in the TAI Global Configuration1 Register (Table 789 p. 1073). This field specifies the remainder amount for the clock that is being generated with the period specified by the <TrigGenAmt> field. The <TrigClkComp> amount is constantly accumulated. When this accumulated amount exceeds the value specified in <TSClkPer> field in the TAI Global Configuration Register, PTP Port = 0xE and 0xF (Table 787 p. 1072), the <TSClkPer> value is added to the output clock momentarily, to compensate for the remainder accumulated overtime.

Note

The <TrigGenAmt> field should be set to no less than two times the <TSClkPer> value.

18.5.2.5 PTP Global Time Increment/Decrement Function

The 32-bit nanosecond segment of the Time-of-Day counter for the PTP is stored in hardware and the 64-bit seconds segment is stored in software. Since every PTP slave node constantly computes the time-of-day, it needs to adjust its hardware with the updated phase offset information. The PTP core assists by providing a PTP Global Timer adjustment functions.

For an increment operation:

- Set the <TimeIncDecEn> field in the TAI Global Configuration, PTP Port = 0xE (Table 786 p. 1071) to 0x1.
- Set <TimeIncDecOp> field in the TAI Global Configuration2 Register (Table 790 p. 1074) to 0x0.
- Configure the <TimeIncDecAmt> field to the value that needs to be added to the current PTP Global Time value.

For a decrement operation:

- Set the <TimeIncDecEn> field to 0x1.
- Set <TimeIncDecOp> field to 0x1.
- Configure the <TimeIncDecAmt> field to the value that needs to be subtracted from the current PTP Global Time value.

Since the <TimeIncDecAmt> field is only an 11-bit parameter, and the PTP Global Timer is 31-bits wide, multiple iterations of the operation need to be performed to get to the adjusted time-of-day value.

18.6 Software Initialization Procedure

The following are the steps required to initialize the PTP core:

1. Reset the PTP core by clearing the <PTPReset> field in the PTP Reset Register (Table 766 p. 1055).
2. Select the PTP core clock, using the <PTPClkSelect> field in the PTP Clock Select Register (Table 767 p. 1055).
3. De-assert reset of the PTP core by setting the <PTPReset> field in the PTP Reset Register (Table 766 p. 1055).

Copyright © 2014 Marvell

4. Set the corresponding per-port <DisPTP> field in the PTP Port Configuration0 Register (Table 773 p. 1059) to a 0x1 (or set all bits, if being done after the power cycle).
5. Configure the <PTPEType> field in the PTP Global Configuration0 Register (Table 768 p. 1056) to the EtherType value on which the PTP frames are expected to arrive.
6. Configure the <MsgIDTSEn> field in the PTP Global Configuration1 Register (Table 769 p. 1056), specifying the PTP message types that need to be time stamped by hardware.
7. Configure the <TSArrPtr> field in the PTP Global Configuration2 Register (Table 770 p. 1057), specifying which of the 16 message types identified by <MsgIDTSEn> field need to occupy Arrival counter 0 and/or 1.
8. Configure the <TransSpec> field in the PTP Port Configuration0 Register (Table 773 p. 1059) with the value defined in the IEEE standard. For IEEE 802.1AS, this needs to be configured to 0x1, and for IEEE1588, this needs to be configured to 0x0.
9. Configure the <DisTSOverwrite> field in the PTP Port Configuration0 Register (Table 773 p. 1059), the <PTPDepIntEn> field in the PTP Port Configuration2 Register (Table 775 p. 1060), the <PTPArrIntEn> field in the PTP Port Configuration2 Register (Table 775 p. 1061), and the <DisTSOverwrite> field in the PTP Port Configuration0 Register (Table 773 p. 1059), as required by the application.
10. Reset the per-port <DisPTP> field in the PTP Port Configuration0 Register (Table 773 p. 1059) to 0x0.

THIS PAGE IS INTENTIONALLY LEFT BLANK

MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Part 5: External Interfaces

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. Moving Forward Faster

Document Classification: Proprietary Information

THIS PAGE IS INTENTIONALLY LEFT BLANK

19 PCI Express Interface (PCIe) 2.0

The device integrates PCI Express ports as listed:

MV78260 3 PCIe units Gen2.0
2 units can be configured as x4 or quad x1 lanes, one unit is x4/x1 only.

MV78230 2 PCIe units Gen2.0
One unit can be configured as x4 or quad x1 lanes. The other unit can only be configured as x1.

MV78460 4 PCIe units Gen2.0
Two of those units can be configured as x4 or quad x1 lanes. Two units are x4/x1 only.

Note

Unless specifically noted, the interface information in this section refers to a single port. All ports are identical and have the same features.

Port annotation used in this section

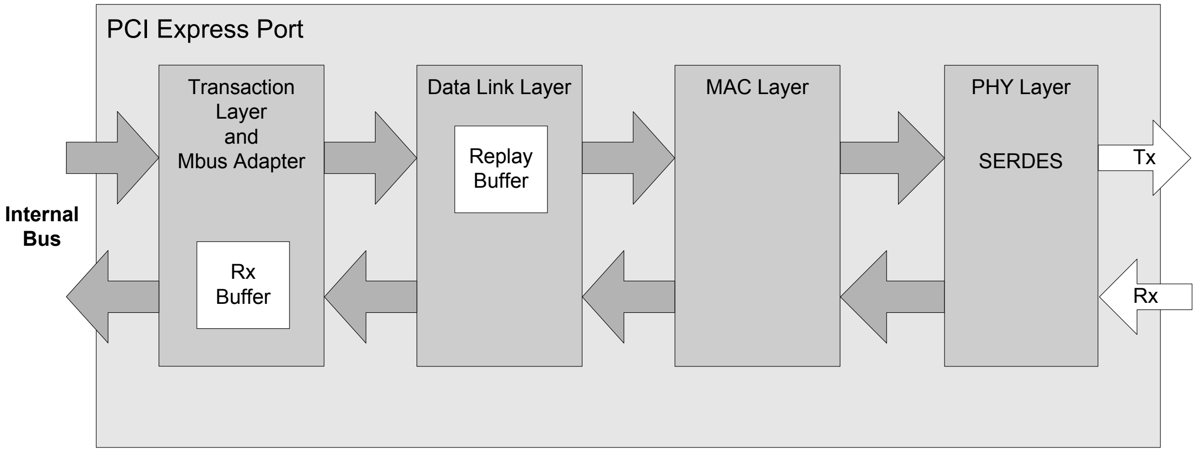
- Throughout this section, “x.0” refers to the port, regardless if the port is set to an x4 or x1 configuration. This means that if “Port x.0” is written it means Port 0.0 or Port 1.0. Similarly, Port x.1, x.2, and x.3 stands for ports 0.1, 0.2 and 0.3. Even if the port is set at reset as an x4 port, the 0.0 or 1.0 naming convention is still used.
- Port2 is referred to as Port 2.0. Ports 2.1, 2.2 and 2.3 do not exist in this device.
- Port3 is referred to as Port 3.0. Ports 3.1, 3.2 and 3.3 do not exist in this device.

Port configuration considerations

When port 0.0 or 1.0 is configured as x4 port then maintain ports 0.1–3 or 1.1–3 (x1 only ports) in power down mode.

Figure 81 is a high-level diagram of a single PCI Express port.

Figure 81: High-level Block Diagram



The PCI Express 2.0 registers are located in Appendix A.9, PCI Express 2.0 Registers, on page 1078.

19.1 Features

The PCI Express interface includes the following features:

- PCI Express Base 2.0 compatible
- Root Complex port
- Can be configured also as an Endpoint port
- Embedded PCI Express PHY based on proven Marvell® SERDES technology
- x4 link width, or x1 links when configured to quad mode
- Up to 5.0 GHz signalling
- Lane polarity inversion support
- Lane reversal support
- Replay buffer
- Maximum payload size of 128 bytes
- Single Virtual Channel (VC-0)
- Ingress and egress Flow Control
- Extended Tag support
- Interrupt emulation message support
- Power management support:
  - Software power management states D1, D2, D3 hot, and D3 cold
  - Active state power management L0s and L1.
- CLKREQn support as Root Complex
- Advanced Error Reporting (AER) capability support
- Single function device configuration header
- Message Signaled Interrupts (MSI) and MSIx support (MSIx support in Root Complex only)
- Support for Dynamic Link Width change
- Support for Dynamic Speed change

- Power Management (PM) capability support, as an Endpoint
- Programmable address map

Note

- Only port 0.0 can be configured as an Endpoint port, whether it is an x1 or x4 port. All other ports are always Root Complex ports.
- If a port is configured to be quad x1, port x.0 must not be powered down by the <PCIe10 Clock Gate> field or the <PCIe00 Clock Gate> field in the Power Management Clock Gating Control Register (Table 1459 p. 1534).

19.2 Functional Description

The PCI Express interface uses a layered architecture, according to the PCI Express specifications. The main layers are the PHY layer, MAC layer, Data Link layer, and Transaction layer. In addition, a Core Adapter layer handles the forwarding of the PCI Express Transaction Layer Packets (TLP) to the device Mbus.

Ports 0.0, 0.1, 0.2, 0.3 and 2.0 are grouped into one unit. The five ports arbitrate, using a fixed round- robin arbitration, on a single Mbus port. Similarly, ports 1.0, 1.1, 1.2, 1.3 and 3.0 are grouped into one unit. These five ports also arbitrate, using a fixed round-robin arbitration, on a single Mbus port.

19.2.1 PHY Layer

On the Tx path, the PHY layer receives symbols from the MAC layer, converts them into a serialized format, and transmits them on the PCIe port. On the Rx path, the PHY layer receives a serialized stream from the PCI-E port, and forwards them as parallel symbols to the MAC layer.

The PHY handles symbol-locking and 8b/10b encoding/decoding.

The PHY layer is responsible also for the clock tolerance compensation. The received symbol stream is adapted to the local clock, by adding or deleting skip OSs (Ordered Sets).

19.2.2 MAC Layer

The MAC layer is responsible for compensating for the skew between the different lanes. This is called lane-to-lane deskew. A skew between different lanes can be caused by factors such as board routing.

The MAC layer is also responsible for establishing and maintaining the PCI Express link, packet framing, and data packing according to the link width. The PCI Express Link Training and Status State Machine (LTSSM) is located in this layer. The LTSSM contains the functionality for the link configuration in terms of the link width, lane polarity, and lane numbering.

Additionally, the MAC layer performs the scrambling functions. The MAC layer controls the different link power-management modes, loopback mode, link disable mode, and link hot-reset function. In addition, the MAC layer handles the generation and detection of the various TSs (Training Sequences) and OSs.

On the Tx path, the MAC layer receives packets (TLPs and DLLPs—Data Link Layer Packets) from the Data Link layer. The packets are framed, scrambled, and packed into the relevant link width and forwarded to the PHY.

On the Rx path, the MAC layer receives aligned symbols from the PHY. The symbols are un-packed according to the relevant link width and un-framing is performed. The packets (DLLPs and TLPs) are then extracted from the frames and forwarded to the Data Link layer.

19.2.3 Data Link Layer

The Data Link layer provides a reliable TLP exchange between two components on the PCIe link. This layer performs most of the data integrity functions as specified by the PCI Express 2.0 specification.

The Data Link layer controls the sequence number generation and detection. It also controls the LCRC generation and detection. Outgoing TLPs are temporarily stored in a replay buffer until an acknowledge is received from the far-end component. When a corrupted or missing TLP is detected, the replay mechanism is used to recover and maintain reliable Transaction layer to Transaction layer connection. The replay buffer holds transmitted packets and re-transmits them when required.

The Data Link layer handles the generation and processing of DLLPs. DLLPs are used for conveying information such as flow control, TLP acknowledgment, and power management handshake.

19.2.4 Transaction Layer

The Transaction layer primary responsibility is handling of TLPs. Outgoing TLPs are assembled and scheduled for transmission. Incoming TLPs are parsed and checked for various errors. In addition, the Transaction layer is responsible for handling the split transaction protocol—both towards the PCIe port and the internal bus.

The Tx path accepts TLPs from the Mbus and schedules them for transmission, based on the flow-control credit availability and the relevant ordering rules. Non-Posted (NP) TLPs are assigned with a unique tag before they are scheduled for transmission. TLPs are then passed on to the Data Link layer for transmission.

The Rx path examines the incoming TLPs for a variety of packet formation errors. Incoming completions tags are checked for a valid NP request, which was sent by the device. TLPs are then passed to the Mbus.

PCIe x4

The MV78230/78x60 PCIe unit x4 ports (ports 0.0, 1.0, 2.0, and 3.0) support on Tx:

- 12 posted or completions, dynamically allocated
- 11 Non-Posted

The MV78230/78x60 PCIe unit x4 ports (ports 0.0, 1.0, 2.0 and 3.0) support on Rx:

- 11 Completions
- 5 Posted
- 11 Non-Posted

PCIe x1

The MV78230/78x60 PCIe unit x1 ports (ports 0.1, 0.2, 0.3, 1.1, 1.2 and 1.3) support on Tx:

- 6 posted or completions, dynamically allocated
- 8 Non-Posted

The MV78230/78x60 PCIe unit x1 ports (ports 0.1, 0.2, 0.3, 1.1, 1.2 and 1.3) support on Rx:

- 4 Completions
- 4 Posted
- 8 Non-Posted

19.3 Link Initialization

Enable the PCI Express interface by setting the <PCIe0En>, <PCIe1En>, <PCIe2En>, or <PCIe3En> field in the SoC Control Register (Table 1432 p. 1512). This allows programming of link parameters before the start of link initialization. The highest common link width is established according to the following order: x4 to x1.

Lane polarity inversion is supported. Each lane differential couple can be routed on the board regardless of its polarity. In addition, lane reversal is supported.

In case the initialization fails and no link is established, the PHY will keep on trying to initiate a link forever unless the port is disabled. As long as the port is enabled, the PHY will continue trying to establish a link; once the PHY identifies that a device is connected to it, a link will be established.

19.4 Master Memory Transactions

Master memory transactions are memory space read and write requests (MRd and MWr TLPs) that are generated and sent over the PCI Express link, and the respective completion TLPs that are received in return.

The following features are supported as a master memory requester:

- Port x.0: Eleven outstanding Non-posted (NP) requests (memory read requests) and up to eleven Posted (P) requests (memory write requests)
- Ports x.1/x.2/x.3: Four outstanding NP requests (memory read request) and up to five P requests (memory write request)
- Maximum memory read request of 128 bytes
- Maximum memory write request of 128 bytes
- 64-bit addressing

19.5 Master I/O Transactions

Master I/O transactions are I/O space read and write requests (IORd and IOWr TLPs) that are generated and sent over the PCI Express link, and the respective completion TLPs that are received in return.

The following features are supported as a master I/O requester:

- Port x.0: Eleven outstanding NP requests (I/O read or write)
- Ports x.1/x.2/x.3: Four outstanding NP request (I/O read or write)
- Maximum I/O read request of 4 bytes
- Maximum I/O write request of 4 bytes
- 32-bit addressing

Note

- Do not initiate I/O requests that are larger than 4 bytes and cross the 4 byte address boundary. These requests are illegal according to the PCI Express Base 2.0 specifications.
- In Endpoint mode, only memory request generation is allowed by the PCI Express Base 2.0 Specification. When working in Endpoint mode, do not initiate IO requests.

19.6 Master Configuration Transactions

Master Configuration transactions are configuration space read and write requests (CfgRd0, CfgWr0, CfgRd1 and CfgWr1 TLPs) that are generated and sent over the PCI Express link, and the respective completion TLPs that are received in return.

The following features are supported as a master configuration requester:

- Port x.0: Eleven outstanding NP requests (configuration read or write)
- Ports x.1/x.2/x.3: Four outstanding NP request (configuration read or write)
- Maximum Configuration read request of 4 bytes
- Maximum Configuration write request of 4 bytes
- Extended register number support (4 KB extended PCI Express configuration header space)

Note

In Endpoint mode, only memory request generation is allowed by the PCI Express Base 2.0 Specification. When working in Endpoint mode, do not initiate configuration requests.

Generation of Configuration Requests

As a Root Complex port, the Marvell® Core Processor may generate Type0 or Type1 configuration cycles to the PCI Express Endpoints, via indirect access using the PCI Express Configuration Address Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 798 p. 1097) and PCI Express Configuration Data Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 799 p. 1098). The following procedure is used for generating configuration cycles:

1. PCI Express Configuration Address Register (x=0–0, x=0–3, y=2–3, y=0–1)—Write the Target Bus, Device, Function, Register and Extended Register Numbers fields, using the <ConfigEn> field to enable this mechanism.
2. PCI Express Configuration Data Register (x=0–0, x=0–3, y=2–3, y=0–1)—Read or write to generate a respective read or write configuration request. The type of the request (type 0 or type 1) is set according to the following rules:
   - *Type1 request:* generated if Target Bus Number Is different from the internal Bus Number.
   - *Type0 request:* generated if Target Bus Number is same as the internal Bus Number, and the Target Device Number is different from the internal Device Number.

The transmitted Configuration TLP includes the Target Bus, Device, Function and Register Numbers as written to the PCI Express Configuration Address Register (x=0–0, x=0–3, y=2–3, y=0–1).

The Configuration request generation is only enabled when the <ConfigEn> bit is set.

19.7 Target Memory Transactions

Target Memory transactions are memory space read and write requests (MRd, MWr TLPs) that are received over the PCI Express link, and the respective completion TLPs that are generated and transmitted in return.

The following features are supported as a target memory completer:

- Port x.0: Reception of up to eleven Memory read requests
- Ports x.1, x.2 and x.3: Reception of up to eight Memory read requests
- Port x.0: Reception of up to five Memory write transactions
- Ports x.1, x.2, and x.3: Reception of up to four Memory write transactions
- Maximum received read request size of 4 KB
- Maximum received write request of 128 bytes
- Support PCI Express access to all of the device’s internal registers

- 64-bit addressing
- Three Memory BARs (64-bit), BAR0 is dedicated to internal register access
- In Endpoint mode: Expansion ROM support

19.8 Target I/O Transactions

Target I/O transactions are I/O space read and write requests (IORd, IOWr TLPs) that are received over the PCI Express link, and the respective completion TLPs that are generated and transmitted in return.

Target I/O transactions are not supported by the device and should not be generated by the downstream device.

19.9 Target Configuration Transactions

Target Configuration transactions are Configuration space read and write requests (CfgRd0, CfgWr0, CfgRd1 and CfgWr1 TLPs) that are received over the PCI Express link, and the respective completion TLPs that are generated and transmitted in return.

When configured as Root Complex port, target Configuration transactions are not supported and should not be generated by downstream device. In Endpoint mode, Target Type0 Configuration transactions are supported.

The following features are supported as a target Configuration completer:

- Reception of up to eleven NP requests (configuration read or write) for ports x.0 and up to eight for the rest.
- Maximum received configuration read request size of 4 bytes.
- Maximum received configuration write request of 4 bytes.

The device as an end point supports responding with Configuration Request Retry Status (CRS) to a configuration read/write access. This is useful for postponing the root complex access till the local CPU finish initialization of the end point.

To enable this feature, set the <CRS Enable> field in the PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 869 p. 1161) to 1 prior to enabling link training (meaning, prior to setting the <PCIe0En> field in the SoC Control Register (Table 1432 p. 1512). When the local CPU finishes end point initialization, clear the <CRS Enable> bit.

19.10 Target Special Cases

- Access attempts that fail address decoding (e.g., do no hit a memory BAR) are completed as Unsupported Requests.
- MemWr accesses to reserved, or not implemented registers, are completed normally on the PCI Express port, and the data is discarded.
- MemRd accesses to reserved, or not implemented registers, are completed normally on the PCI Express port, and a CplD TLP with data value of 0 and *SC* (*Successful Completion* status) is returned.

19.11 Messages

PCI Express defines a new message space. Messages are used to replace legacy PCI side-band signals such as interrupts, error signals, hot-plug signals etc. Messages are also used to enable new capabilities such as active power management, Slot Power Limit and others.

Table 78 lists the message groups supported as a Root Complex port, and if the group is supported by the device.

Copyright © 2014 Marvell

Table 78: Supported Message Groups—Root Complex Mode

Message Group	Supported	Action
Interrupt Signaling	Yes	A received *Assert_INTx* message is forwarded as an interrupt to the Marvell® Core Processor. Reception of a *Deassert_INTx* message clears the relevant interrupt. **NOTE:** Both INTA, INTB, INTC and INTD are supported. (x = A, B, C or D)
Power Management Event (PME)	Yes	A received PM_PME message is forwarded as an interrupt to the Marvell® Core Processor. The log of the message is registered in PCI Express Root Complex Power Management Event (PME) Register (x=0–0, x=0–3, y=2–3, y=0–1). While software is handling one PME message, a new PME message may be received and registered in this register.
Error Signaling	Yes	A received Error message is forwarded as an interrupt to the Marvell® Core Processor. Both Correctable, Non-fatal and Fatal error messages are supported.
Hot Plug Signaling	No	
Locked Transaction Support	No	
Slot Power Limit Support	Yes	To enable sending SSPL, set the <SsplMsgEnable> field in the PCI Express Root Complex Set Slot Power Limit (SSPL) Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 871 p. 1165) bit[16] to 1 before link is up. If enabled, upon reaching Link_up and whenever there is a change in the <SlotPowerLimitValue> bits[7:0] or <SlotPowerLimitScale> bits[9:8] an SSPL message is transmitted.
PME_TurnOff	Yes	To send a TurnOff message, set the <Send Turn Off Msg> field in the PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 874 p. 1167) bit[5] to 1. When the EP reach L2 state, <EPReady4TurnOff> interrupt is asserted, indicating to the host that it may turn off the primary power of the EP. To exit this state, software must initiate soft reset and clear <SndTurnOff> bit once link is up again.
Vendor Specific Messages	No	

Copyright © 2014 Marvell

Table 78 lists the message groups that are supported in Endpoint mode:

Table 79: Supported Message Groups—Endpoint Mode

Message Group	Supported	Action
Interrupt Signaling	Yes	Interrupt assertion on internal interface is forwarded as an Interrupt Assert message to the PCI Express port. Interrupt de-assertion on internal interface is forwarded as an Interrupt De-assert message to the PCI Express port. Both INTA, INTB, INTC and INTD are supported.
Power Management	Yes	Received PME_Turn_Off message triggers the relevant power management procedure. It is acknowledged by a PME_TO_Ack message that is generated and transmitted on the PCI Express port. PM_Active_State_Nak and PM_PME messages are not supported.
Error Signaling	Yes	Error in the PCI Express port is forwarded as an Error message to the PCI Express port. Correctable, Non-fatal, and Fatal error messages are supported.
Hot Plug Signaling	No	
Locked Transaction Support	No	
Slot Power Limit Support	Yes	When Set_Slot_Power_Limit message is received, the Slot Power Limit Configuration registers are updated accordingly.
Vendor Specific Messages	No	

19.12 Message Signaled Interrupt (MSI) Support

MSI interrupt support is required for PCI Express devices, and is driven by completing a memory write TLP.

MSI Root Complex (RC) Mode Support The MSI data must be monitored in the predefined address by the software.

Upon receiving an MSI, an interrupt is set in the interrupt main cause register, and interrupt data is saved in the PCI Express MSI Message Data Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 819 p. 1114).

MSI Endpoint (EP) Mode Support When enabled through the PCI Express MSI Message Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 816 p. 1112), the EP generates an MSI write for any edge interrupt assertion. The write address is set according to the PCI Express MSI Message Address Register.

For 64-bit addresses, the PCI Express MSI Message Address (High) Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 818 p. 1114) data content is the same as the PCI Express MSI Message Data Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 819 p. 1114).

19.13 Locked Transactions

Locked transaction semantics are not supported. This includes MRdLk, CplLk, and Unlock messages.

Copyright © 2014 Marvell

19.14 Arbitration and Ordering

The arbitration scheme on both Tx and Rx directions are following the PCI Express ordering rules. On each direction, there are separate queues for posted, non-posted, and completion TLPs that are used to adhere to these ordering rules.

19.14.1 Tx Ordering Rules

- All TLPs from same type (P, NP, C) are forwarded in-order.
- Non-Posted transactions push posted transactions (unless the <TxNpPushDis> field in the PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 889 p. 1182) is set to 1).
- Completions push posted transactions (unless the <TxCmplPushDis> field in the PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 889 p. 1182) is set to 1).
- For other transaction couples, re-ordering may occur.
- Relaxed-ordering is not used for ordering in internal queues.
- Simple round-robin arbitration is performed on all transactions which may be transmitted to the PCI Express fabric according to those rules.

19.14.2 Rx Ordering Rules

- All TLPs from same type (P, NP, C) are forwarded in-order.
- Non-Posted transactions push posted transactions (unless <RxNpPushDis> field in the PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 889 p. 1183) is set to 1).
- Completions push posted transactions (unless the <RxCmplPushDis> field in the PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 889 p. 1183) is set to 1).
- For other transaction couples, re-ordering may occur.
- Relaxed-ordering is not used for ordering in internal queues.
- Simple round-robin arbitration is performed on all transactions.

19.15 PCI Express Register Access

The PCI Express registers can be accessed from an external PCI Express device, or from the Marvell® Core Processor.

The Read Only (RO) registers have the following access permissions:

- An external PCI Express device can only have read access to these registers.
- If not hardwired or set/clear by the hardware, these registers are read/write (RW) from the Marvell® Core Processor.

If the device is configured as an endpoint, the configuration header registers can be accessed from an external Root Complex port via type0 configuration transactions.

The configuration header registers are also mapped to the chip internal address space as follows:

- All configuration header registers are mapped to the internal memory space.
- Direct memory access is enabled via the <CfgMapTo MemEn> field in the PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 869 p. 1163). When enabled, a direct memory access can be performed from the Marvell® Core Processor or from an external PCI Express device, even if the device is configured as Root Complex port.
- If not hardwired or set/clear by the hardware, all RO configuration header registers are RW, if accessed through the direct memory mapping.

Note

If Port0 is set to a quad x1 configuration, an external PCI Express device attached to Port 0.x has no access to the registers of the other three ports.

19.15.1 D3$_{hot}$ to D0 Transition—Endpoint Mode

Switching from *D3$_{hot}$* state to *D0* state is done by writing to <PMState> field in the PCI Express Power Management Control and Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 815 p. 1112).

When such transition occurs, the configuration headers are reset to default values, except the following registers:

- PCI Express Device and Vendor ID Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 800 p. 1099)
- PCI Express Class Code and Revision ID Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 802 p. 1102)
- PCI Express Subsystem Device and Vendor ID Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 810 p. 1107)

19.15.2 PHY Registers Access

The PCI Express PHY has its own register file. The software can access the PHY registers via the PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172).

Note

The PHY registers are for Marvell internal use (debug purposes). Do not access these registers unless explicitly directed to by a MV78230/78x60 related document.

To write to a PHY register, write to the following fields in the PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1):

- <PhyAddr> field to point to the required register offset.
  The register offset must be divided by 4. For example, writing to Power and PLL Control register (offset: 0x00000004) means that <PhyAddr> must be set to 0x1.
- <PhyData> field to set the desired data.
- <PhyAccssMd> field cleared to 0.

To read from a PHY register:

- Write to PCI Express PHY Indirect Access Register with the <PhyAddr> field pointing to the required register offset, and with the <PhyAccssMd> field set to 1.
- Read the PCI Express PHY Indirect Access Register; the read data is available in the <PhyData> field.

Each of the 4 lanes of the PHY has its own register. Register address bits[7:0] selects the register offset within a specific lane. Address bits[13:8] selects the lane as follows:

- Lane0 = 0x00
- Lane1 = 0x01
- Lane2 = 0x02
- Lane3 = 0x03
- Broadcast to all four lanes = 0x2X

19.16 Hot Reset

Hot Reset is an in-band reset indication that can be sent from the root-complex downstream and reset the PCI Express hierarchy. Use the following procedure to generate a hot reset:

1. Write to the <ConfMstrHot Reset> field in the PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 869 p. 1162).
2. To check that Hot Reset has been completed, poll the <DLDown> field in the PCI Express Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 870 p. 1164). When this bit is set, DL is down and Hot Reset has been completed.
3. Clear the <ConfMstrHot Reset> field.

Note

Root Complex registers are not reset by Hot Reset.

19.17 Error Handling

19.17.1 Physical Layer Errors

Table 80 list the conditions that can cause a PHY layer Receive error.

Table 80: Physical Layer Error List

Error Name	Conditions
Receiver Error	• PHY Overflow • PHY Underrun • PHY 8B/10B decode error • PHY Disparity error Severity: Correctable.

19.17.2 Data Link Layer Errors

Table 81 lists the Data Link layer errors.

Table 81: Data Link Layer Error List

Error Name	Conditions
Bad TLP	• LCRC Error detected in received TLP. • Sequence number error detected in received TLP. Severity: Correctable.
Bad DLLP	CRC Error detected in received DLLP. Severity: Correctable.
Replay Timeout Error	Replay timer expired. Severity: Correctable.
REPLAY_NUM Rollover Error	REPLAY_NUM rolled-over. Four consecutive replays were transmitted. Severity: Correctable.
Data Link Layer Protocol Error	Reception of an Ack with out-of-range ackNac_Seq_Num. Severity: Fatal.

19.17.3 Transaction Layer Errors

Table 82 lists the Transaction layer errors.

Table 82: Transaction Layer Error List

Error Name	Description
Flow Control Protocol Error	• DLLP receive timer expiration. • Received FC initial credit values, which are less than the minimum advertisement according to the spec. • Received update FC message for a credit type, which was advertised as infinite on initialization. Default severity: Fatal.
Malformed TLP	• Received TLP with data payload size larger than the Maximum Payload Size. • Received TLP with undefined *Type* and *Fmt* fields value. • Received TLP with length different than expected according to the length, type, and TD (TLP Digest) field. • Received request with Address/Length combination crossing the 4-KB boundary. • Received Power Management Set_Slot_Power, Unlock, INTx, and error message with TC field not equal to 0 (TC0). Default severity: Fatal.
Poisoned TLP Received.	Poisoned TLP received. Default severity: Non-Fatal.
Unsupported Request	• Received unsupported TLP type (CfgWr1, CfgRd1, MrdLk). • Received unsupported message codes. • Failed address decoding on received TLP. • Received CfgWr0 or CfgRd0 with function_number different than 0. • Received poisoned write request to internal register space. Default severity: Non-Fatal. **NOTE:** Reception of Vendor_Defined_Type_1 message is discarded silently. It is not an error state.
Received UR Completion	• Received Cpl TLP with UR completion status. • Received CplLk or CplD with UR completion status. Not a PCI Express error. Mapped to PCI status.
Completion Timeout	Outstanding Non Posted request to PCI Express timeout has expired. Default severity: Non-Fatal.
Completer Abort	Received read requests to the internal address space, with the Length field different than 1 DWORD. Default severity: Non-Fatal.
Received CA completion	Received a Cpl with CA completion status Not a PCI Express error. Mapped to PCI status.
Unexpected Completion	• Received unexpected completion TLP (Cpl or CplD). Completion does not correspond to one of the outstanding NP requests. • Received CplLk or CplDLk TLPs. Default severity: Non-Fatal.

19.17.4 Error Propagation

The PCI Express specification defines a mechanism for propagation of erroneous TLPs (erroneous data payload) via an EP (Error Poisoning) bit in the packet header.

Receive

Upon receiving a poisoned write TLP:

- If it is not an access to the chip internal registers and if the <RxDPPropEn> field in the PCI Express Mbus Adapter Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 865 p. 1158) is set, an erroneous data indication is forwarded along with the data. Regardless of <RxDPPropEn> field setting, Error status bits in the PCI Express configuration header registers are set (if enabled by the relevant configuration registers control bits) and Error messages are sent (if enabled).
- If it is an access to the chip internal registers (whether PCI Express register file or another register file), the transaction is discarded (not written to registers). An unsupported requests completion is sent if needed, and the relevant error status bits are set.

Upon receiving a poisoned completion TLP, if the <RxDPPropEn> field is set, an erroneous data indication is forwarded along with the data. Regardless of <RxDPPropEn> field setting, error status field are set, if enabled by the relevant control bits.

Transmit

Tx error forwarding is controlled by the <TxDPPropEn> field in the PCI Express Mbus Adapter Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 865 p. 1158). For either requests or responses, the corresponding TLP is poisoned (EP bit is set) if:

- Data received from the Mbus is marked as erroneous.
- Forwarding is enabled by the <TxDPPropEn> field.

19.17.5 Completion Timeout

The Completion timeout (Cpl TO) mechanism is defined in the PCI Express specification. When a device issues a NP request on the PCI Express port and does not receive all the related completions of the request after a pre-defined period of time, the device must indicate a Completion Timeout error status.

The Cpl TO period is set by the <Reserved> field in the PCI Express Completion Timeout Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 872 p. 1166). The Cpl TO mechanism can also be disabled through this register field.

If the Cpl TO expires before a Tx NP request is completed (not all completion fragments arrived):

- The relevant error status bits are set.
- An error message is transmitted, if not masked.
- The timed-out requests are completed on the Mbus with dummy read data, and with an erroneous data indication.

19.18 Power Management

The following sections describe the PCI Express interface power management features.

19.18.1 D1, D2, and D3 Software Power Management Options

The device supports the PCI Express software power management options D1, D2, and D3, as described in the PCI-E 2.0 Specification.

As an endpoint, it also supports the turnoff process. Upon receiving turnoff message from the Root Complex, a maskable interrupt is set. The device CPU acknowledges the turnoff request by setting the <Send Turn Off Ack Msg> field in the PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 874 p. 1167).

As a Root Complex, the MV78230/78x60 can set the device it works with to a turnoff state by sending a turnoff message. To send the turnoff message, set the <Send Turn Off Msg> field in the PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 874 p. 1167). When the turnoff process completes, a maskable interrupt is set in the main cause register.

As a Root complex, the MV78230/78x60 responds to a Power Management Event (PME) generated by the device. A maskable interrupt is set upon receiving a PME message. The PME data is saved in the PCI Express Root Complex Power Management Event (PME) Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 873 p. 1166).

19.18.2 Active State Power Management (ASPM)

An active power management event is a mechanism defined by the PCI Express specification. It allows the hardware to lower the link power state. This device supports all ASPM options as a root complex and endpoint.

The PCI Express driver should enable this feature, as described in the PCI Express specification.

L1 ASPM Endpoint Mode If this feature is enabled, the device starts the L1 ASPM event when the conditions for this event are met, as defined in the PCI Express Specification, and when the <L1 ASPM En> field in the PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 874 p. 1168) is set.

L1 ASPM Root Complex Mode As a root complex, the device acknowledges an L1 ASPM event when the <L1 ASPM Ack> field in the PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 874 p. 1168) is set. If this field is not set, an L1 ASPM non-acknowledge response is sent upon the L1 ASPM request.

L0 Support The device supports L0 events on receive and transmit, as defined in the PCI Express specification.

19.18.3 CLKREQn Support

The device supports dynamic gating of the output reference clock for additional power saving via CLKREQn signaling as defined in the PCI Express specification.

19.18.4 PHY Shutdown

Shutting down the PHY is done as part of the unit power down procedure.

To power down the unit, access the PCI Express Power Management Control and Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 815 p. 1111), and clear the relevant port's power up register.

Power down a port only after the software has confirmed that the link was not negotiated for at least 100 ms.

Before powering down the port, the software must clear the PCI Express enable bit for the relevant port in the SoC Control Register (Table 1432 p. 1511).

Note

When working in quad mode, always supply power to port x.0, unless it is desired to power down all of the PCIe ports.

19.19 Loopback Modes

The following DFT features are supported by the PCI Express port:

- Master Loopback
- Internal Loopback
- Slave Loopback
- PRBS (Pseudo-Random Bit Sequence) generation and checking

19.19.1 Master Loopback

Master Loopback mode forces the device on the other side of the PCI Express link to enter a Loopback mode and mirror all the received traffic back to its Tx side. This mode enables a self-test procedure for the PCI Express port and link.

Only the PCI Express TS1 Order Sets are sent during Master Loopback. To check the physical connectivity of the link, monitor the receiver errors.

Entering master loopback must be done before the PCI Express link is enabled. Use the following procedure:

1. Set the <ConfMstrLb> field in the PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 869 p. 1161).
2. Set the <PCIe0En>, <PCIe1En>, <PCIe2En>, or <PCIe3En> field in the SoC Control Register (Table 1432 p. 1512) to enable the specific PCI Express port.
3. Check that the PCI Express link is ready for a loopback test by polling bit[26] in the <TrnCntlBits> field in the PCI Express Debug MAC Status 1 Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 888 p. 1181). When this bit is cleared, the link is in master loopback mode and the loopback test can start.
4. Check for Receiver errors.

To exit Master Loopback mode, reset the entire chip.

19.19.2 Internal Loopback

When working in Internal Loopback mode, the PCI Express port mirrors all the traffic received from the Mbus back to its Rx side. This mode enables self-test procedures for the PCI Express port, even when no external device is attached.

The PCI Express PHY supports Shallow and Deep Loopback modes as shown in Figure 82 and Figure 83.

Figure 82: Shallow Internal Loopback

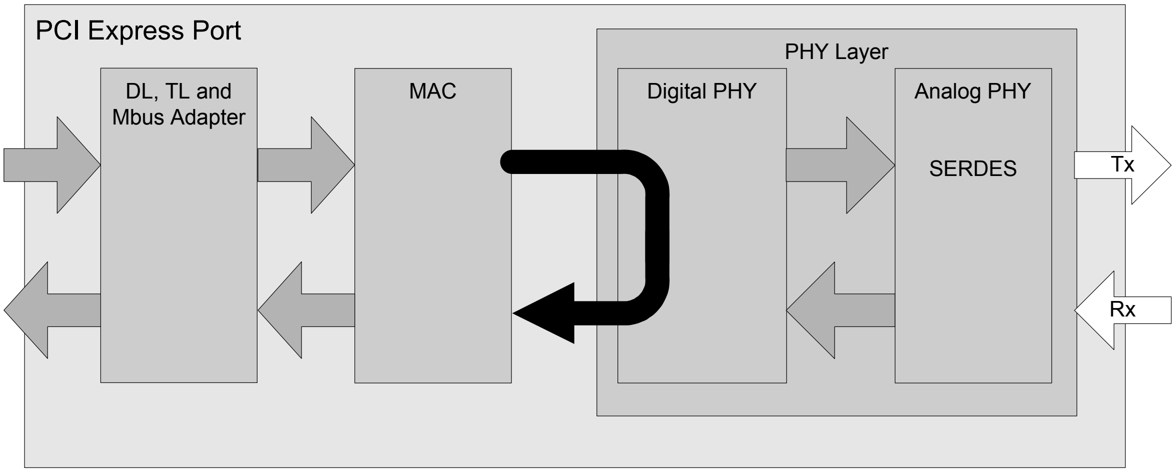
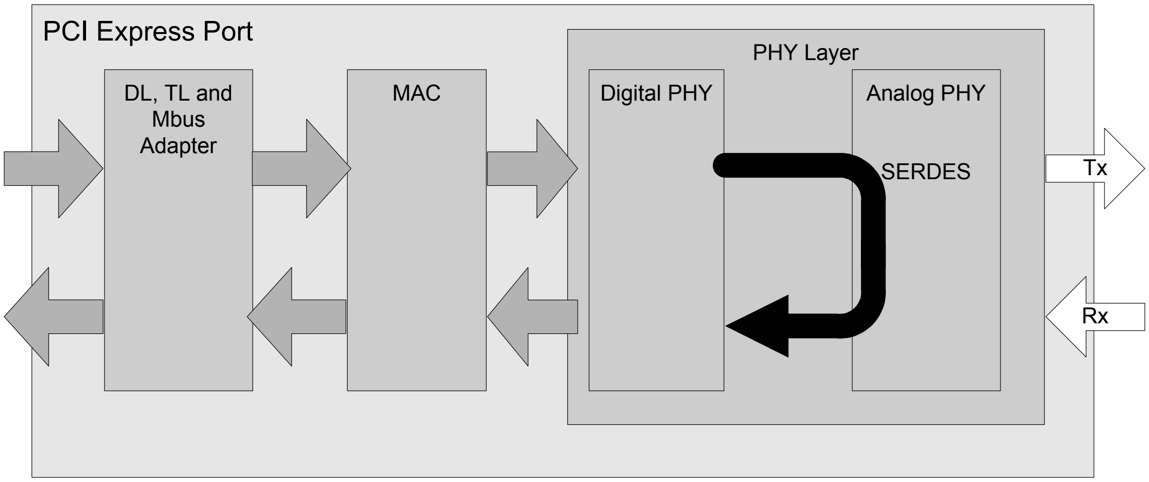


Figure 83: Deep Internal Loopback



Use the following procedure to enter Internal Loopback mode:

1. Set to 0x2 the <CfgForceRxPresent> field in the LANE_CFG0 Lane Configuration 0 Register (Table 900 p. 1189). This setting forces the receiver detection for this lane.
2. Set the following registers for Shallow, Deep, or SERDES loopback:
   - For Shallow loopback, set PCI Express PHY Register 0xC2 bit[10] to 1, using the PHY registers indirect access as explained in Section 19.15.2, PHY Registers Access, on page 343.
   - For Deep loopback set the PHY register 0xC2 bit[11] to 1.
   - For SERDES loopback, set the PHY register 0xC2 bit[12] to 1.
3. Set PCI Express PHY register 0x23 bits[13] to 0x1.

4. Set the <PCIe0En>, <PCIe1En>, <PCIe2En>, or <PCIe3En> field in the SoC Control Register (Table 1432 p. 1512) to enable the PCIe ports.
5. Check that the PCI Express Link is ready for the loopback test by polling the <DLDown> field in the PCI Express Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 870 p. 1164). When this bit is cleared, loopback test can start.
6. Set the relevant registers to control the inbound traffic (BAR, address space control and the address decoding windows).
7. Generate the loopback test traffic and check that it is received correctly.

To exit Internal Loopback mode, generate hot reset or reset the entire chip.

19.19.3 PRBS

When using an internal loopback, it is possible to use PHY PRBS generation and checking, rather than activate the entire chip to generate traffic.

To enable PRBS generation, choose the pattern by configuring PHY registers 0x15 bits[7:4], set PHY register 0x15 bit[15] to 1. PRBS errors are counted in PHY registers 0x20 and 0x1F. To reset the PRBS error counter, set PHY register 0x15 bit[14] to 1.

19.19.4 Slave Loopback

Slave Loopback is the opposite case of Master loopback. This procedure is initiated and controlled by the external PCI Express device.

19.20 Peer-to-Peer Traffic

The device supports the memory transactions between the PCI Express ports 0 and PCI Express ports 1.

The device does not comply with PCI transparent bridge specification:

- It does not implement a PCI bridge configuration header.
- It does not support type1 configuration cycles forwarding.
- It does not comply with the errors forwarding specification.

However, it is still very useful as a non-transparent bridge:

- It supports forwarding of memory and I/O transactions.
- It supports forwarding of PCI interrupts to PCI Express interrupt messages.
- It supports configuration cycle forwarding via indirect access, using the PCI Express Configuration Address Register (x=0–0, x=0–3, y=2–3, y=0–1) and the PCI Express Configuration Data Register (x=0–0, x=0–3, y=2–3, y=0–1).

19.21 Force Link Down

It is possible to close a link to one of the PCI Express ports, even if the interface is set to a quad x1 configuration, without affecting other PCI ports.

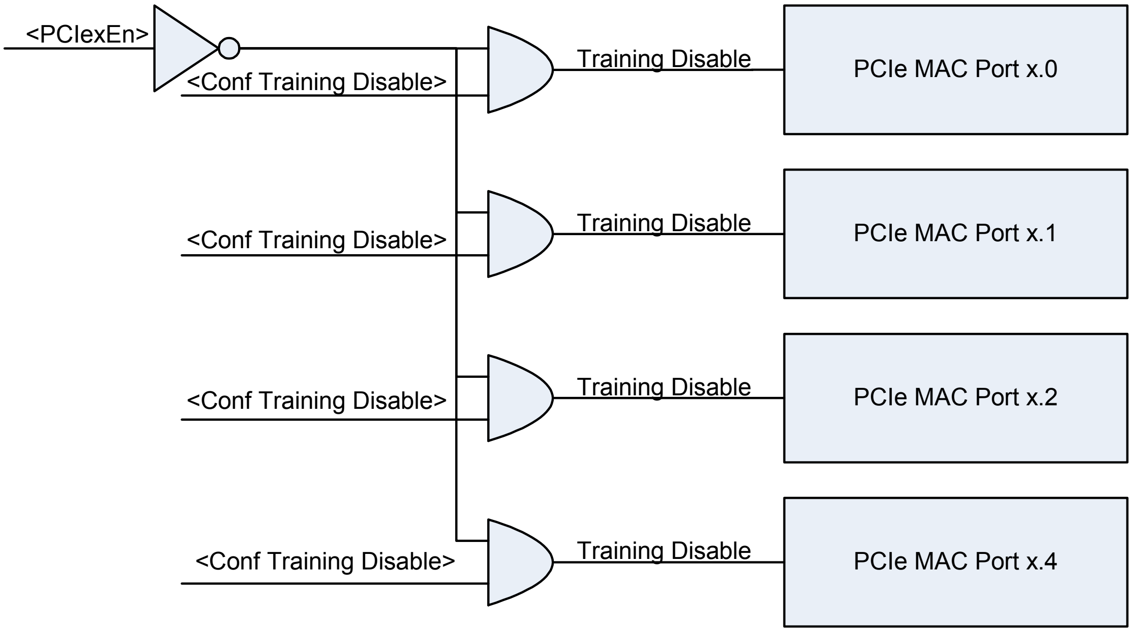
A sample procedure for forcing a link down for PCI Express port 0 (when configured to x4 mode) is as follows:

1. Trigger the link disable by setting <LnkDis> field in the PCI Express Link Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 824 p. 1126).
2. To ensure that the link is disabled, poll <DLDown> field in the PCI Express Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 870 p. 1164).
3. Clear <PCIe0En> field in the SoC Control Register (Table 1432 p. 1512).
4. Clear <LnkDis>.

When port 0 or 1 are configured as a quad x1 configuration:

1. For all of the ports, clear the <Conf Training Disable> field in the PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 869 p. 1161).
2. Clear the <PCIe0En> field in the SoC Control Register (Table 1432 p. 1512).
3. Trigger the link disable by setting <LnkDis> field in the PCI Express Link Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 824 p. 1126).
4. To ensure that the link is disabled, poll <DLDown> field in the PCI Express Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 870 p. 1164).
5. Clear <Conf Training Disable>.
6. Clear <LnkDis>.

Figure 84: PCI Training for Quad x1 Link Disable



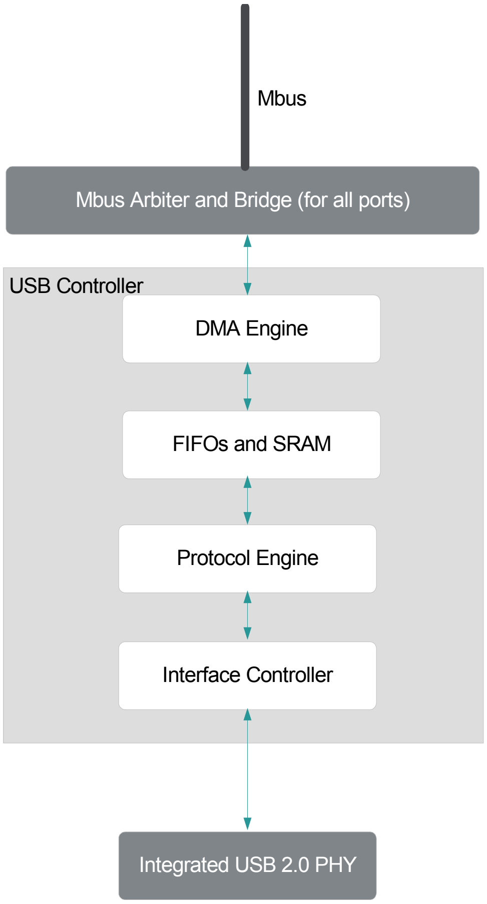
20 Universal Serial Bus (USB 2.0) Interface

The ARMADA® XP includes three USB 2.0 ports that can act as a USB host or a USB device.

Each port integrates a USB controller (including a DMA and protocol engines) and an embedded USB 2.0 compatible PHY.

A common USB bridge connects all ports to the Mbus interconnect via an arbiter.

Figure 85: USB Port Block Diagram



The USB 2.0 registers are located in Appendix A.10, USB 2.0 Registers, on page 1207.

Note

For more details about the USB controller implementation, refer to the controller specification document *USB-HS High-Speed Controller Core Reference.*

Copyright © 2014 Marvell

20.1 USB Controller Features

- USB 2.0 compatible
- Enhanced Host Controller interface (EHCI) compatible as a host
- As a host, supports direct connection to all peripheral types (LS, FS, HS)
- As a device, connects to all host types (HS, FS) and hubs
- 6 independent end points supporting control, interrupt, bulk, and isochronous data transfers
- Wake on USB options

20.2 USB PHY Features

- Embedded USB 2.0 compatible PHY
- 480 Mbps High Speed (HS), 12 Mbps Full Speed (FS) and 1.5 Mbps Low Speed (LS) serial data transmission rates
- Synchronization/End-of-Packet (SYNC/EOP) generation and checking
- Data and clock recovery from serial stream on the USB
- Non Return to Zero Invert (NRZI) encoding/decoding with bit stuffing/unstuffing
- Bit stuffing/unstuffing; bit stuff error detection
- Frame error detection
- Holding registers to stage transmit and receive data
- Low power suspend mode
- Resume signaling
- Wake up and Suspend detection
- Supports USB 2.0 Test modes
- Ability to switch between FS and HS terminations/signaling

20.3 Power Management

When the USB port is placed in suspend low power state, the PHY power consumption can be minimized by turning off blocks that are not required for the suspend/resume operation. To enable this, set PORTCS1[PHCD] to 1.

The port wakes up from the low power mode and resumes operation when configured to do so by the host software.

When working with 1 USB port, the unused port should be forced into a suspend low power state as follows:

1. Set <Enable force suspend> field in the USB 2.0 PHY Suspend/Resume Control Register (n=0–2) (Table 937 p. 1215) to 1 to enable forced suspend.
2. Clear <SUSPENDM> field in the USB 2.0 Power Control Register (n=0–2) (Table 936 p. 1214) to 0 to force PHY suspend.

When the ARMADA® XP SoC is in Idle or Deep Idle low power modes, the USB port can wake the SoC with one of the following maskable events:

- Resume signalling by a USB device
- USB device attachment
- USB device detachment

21 Serial-ATA (SATA) II Interface

The device integrates two SATA II compliant Serial-ATA (SATA) host controllers (SATAHC) with integrated SERDES interface. The SERDES interface is multiplexed with other functionality. Refer to the device's *Hardware Specifications* for details about the SERDES multiplex options.

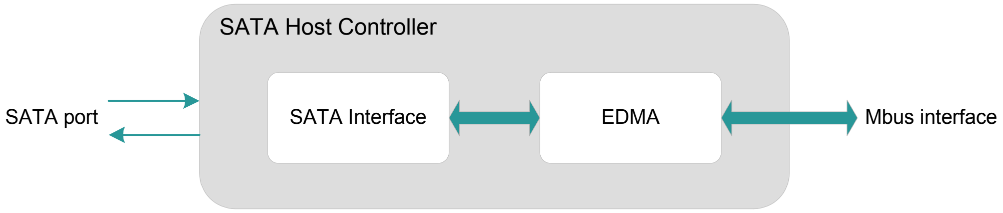
The SATA II port is based on the Marvell® SATA host controllers and SATA proven technology. The device is fully compatible with SATA II phase 1.0a specification (Extensions to SATA I specification).

This section describes a single SATA port, but the information applies to the second port as well.

In the device, the SATAHC consists of a SATA port and an Enhanced DMA (EDMA) that controls the port.

Figure 86 provides a SATAHC block diagram, showing the flow to/from the SATA port, SATA interface, EDMA, and the Mbus Interface.

Figure 86: SATAHC Block Diagram



SATAHC EDMA
- Controls the ATA transactions associated with its port.
- Contains a 0.5-KB buffer for posted write and prefetch read transactions.
- Contains the registers that control the EDMA operation.

SATA Interface The SATA interface is compliant with the Serial-ATA II phase 1.0 specification (Extension to SATA I specification).

The SATA Host Controller registers are located in Appendix A.11, Serial-ATA Host Controller (SATAHC) Registers, on page 1216.

21.1 Features

The device employs the latest SATA II PHY (SERDES) technology, with 3.0 Gbps (Gen2i) and backwards compatible with 1.5 Gbps (Gen1i) SATA I. The device SATA II PHY accommodates the following features:

- SATA II 3 Gb/s speed
- Backwards compatible with SATA I PHYs and devices[1]
- Support Spread Spectrum Clocking (SSC)
- Programmable PHY for industry leading backplane drive capability
- SATA II power management compliant
- SATA II Device Hot-Swap compliant

[1] AC coupling is still required while working with Gen1 devices.

- Low power consumption — Less then 300 mW per SATA II PHY
- PHY isolation Debug mode

The SATA II interface supports the following protocols:

- Non Data type command
- PIO read command
- PIO write command
- DMA read command
- DMA write command
- Queued DMA read command
- Queued DMA write command
- Read FPDMAQueued command
- Write FPDMAQueued command

The SATA II interface does not support the following protocols:

- ATAPI (Packet) command
- CFA commands

21.2 SATA Functional Description

21.2.1 SATAHC Initialization—Interrupt Coalescing

The command execution can be accomplished with or without using the coalescing mechanism:

- If the interrupt coalescing mechanism is used, initialize the following registers:
  - SATAHC Interrupt Coalescing Threshold Register (Table 942 p. 1221)
  - SATAHC Interrupt Time Threshold Register (Table 943 p. 1221)
- If Interrupt coalescing mechanism is not used, the <SataCoalDone> field in the SATAHC Main Interrupt Cause Register (Table 945 p. 1224) should be masked.

21.2.2 Host Direct Control Over the Hard Disk Drive

- When the EDMA is disabled, the <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) is cleared.
  - The host has direct control over the device through the ATA task registers (see Table 1020, Shadow Register Block Registers Map, on page 1277).
- When the EDMA is enabled, the <eEnEDMA> is set.
  - The EDMA has full control over the hard disk drive (HDD).
  - If any of the ATA task registers are written, a write transaction results in unpredictable behavior.

21.2.3 LED Indications

Each SATA port has two LED indications:

- Disk active indication
- Combined disk present and disk active indication—both indications appear on the same LED

These LED indications are selected through the MPP interface.

Optionally, by setting the GPIO Blink Enable registers of the corresponding GPP pins on which the SATA LED output signals are multiplexed, the LED indication for both the SATA and GPIO LEDs may blink.

The Disk Active or Disk Active/Presence (act_presence) LED indication is determined by the SATAHC LED Configuration Register (Table 947 p. 1226). Figure 87 shows the flow that sets the

Copyright © 2014 Marvell

LED state. Table 83, Disc Status LED State Settings, on page 356 explains the function of the bits shown in Figure 87.

Figure 87: Disc Status LED Indication Diagram

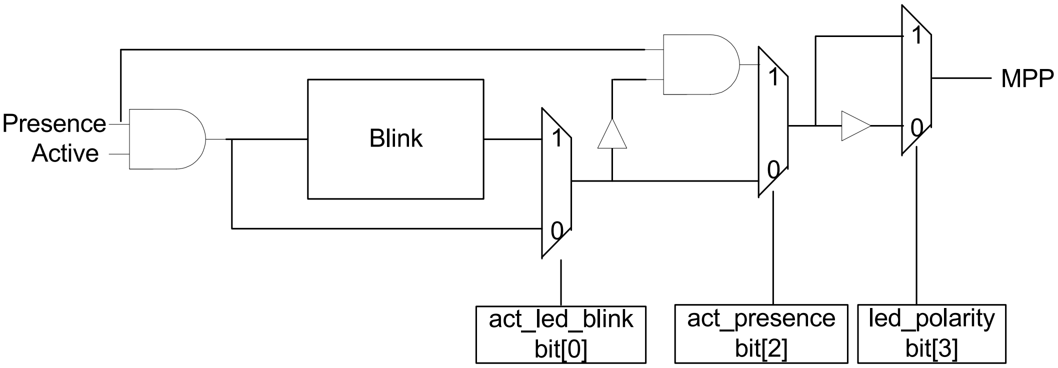


Table 83: Disc Status LED State Settings

Bit Number	Bit Name	Bit Function
0	<act_led_blink>	0 = Use indication as is. 1 = Set indication to blinking.
2	<act_presence>	0 = Use active indication only. 1 = Multiplex active and presence indication on the same LED.
3	<led_polarity>	0 = Invert the active indication. 1 = Do not change the active indication.

21.3 EDMA Functional Description

The interface between host CPU and the EDMA consists of two queues: the request queue and the response queue. The request queue is the interface that the host CPU uses to queue ATA DMA commands as a request between the system memory and the device. The response queue is the interface that the EDMA uses to notify the host CPU that a data transaction between the system memory and the device was completed. Each entry in the request queue consists of an ATA DMA command and the EDMA parameters and descriptors to initiate the device and to perform the data transaction.

The EDMA is further responsible for parsing the commands, initializing the device, controlling the data transactions, verifying the device status, and updating the response queue when the command is completed. This all occurs without CPU intervention. Direct access to the device is also supported for device initialization and error handling.

21.3.1 EDMA Request and Response Queues

The request queue and the response queue are each located in CPU memory and organized as a length of 32 entries, circular queues (FIFO) whose location is configured by the Queue In-Pointer and the Queue Out-Pointer entries. Since these pointers are implemented as indexes and each entry in the queue is a fixed length, the pointer can be converted to an address using the formula: Entry address = Queue Base address + (entry length * pointer value).

The request queue is the interface that the CPU software uses to queue ATA DMA commands as a request for a data transaction between the system memory and the device. Each entry in the request queue is 32 bytes in length, consisting of a command tag, the EDMA parameters, and the ATA device command to initiate the device and to perform the data transaction.

The response queue is the interface that the EDMA uses to notify the CPU software that a data transaction between the system memory and the device has completed. Each entry in the response queue is 8 bytes in length, consisting of the command tag and the response flags.

Figure 88: Command Request Queue—32 Entries

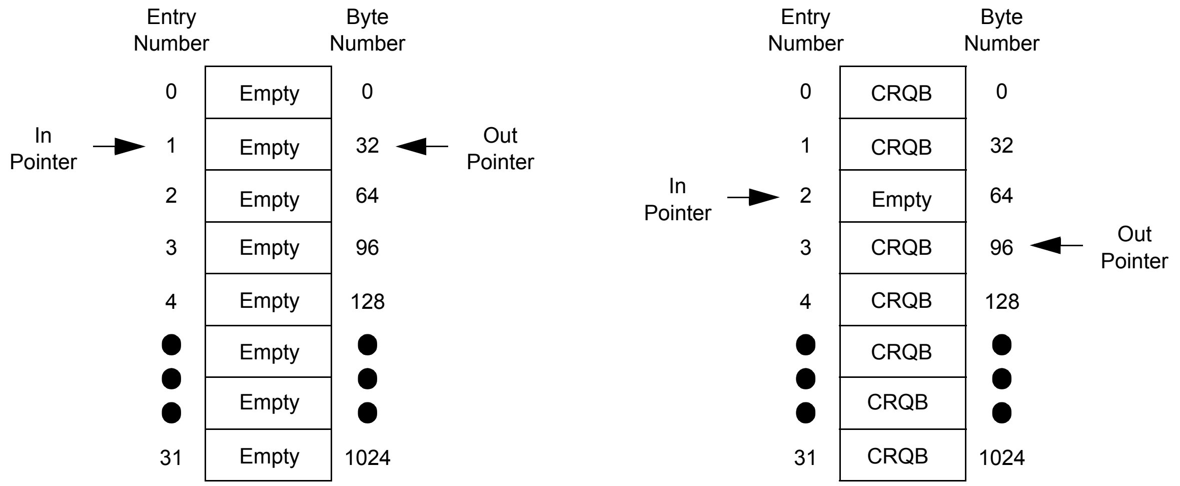
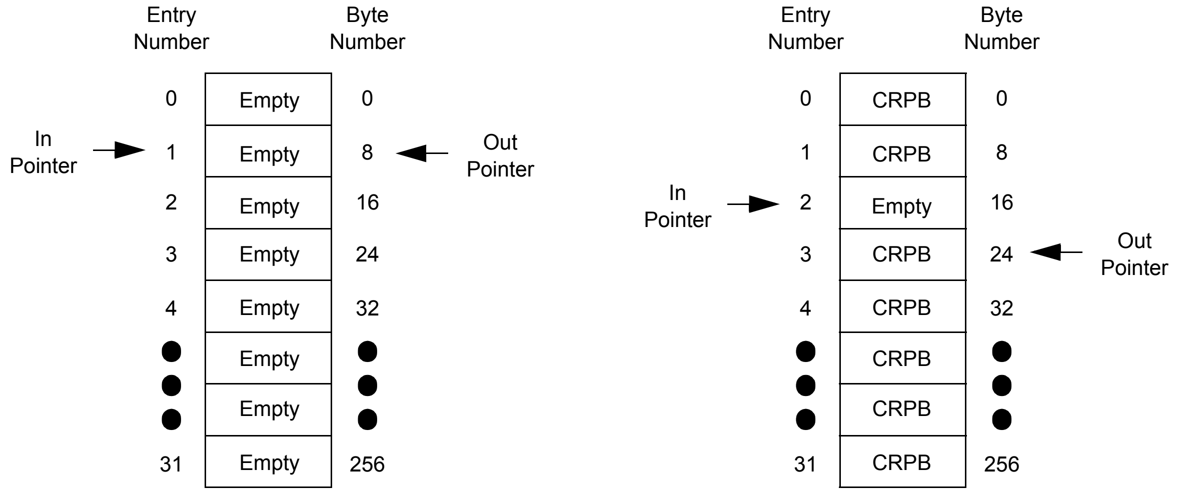


Figure 89: Command Response Queue—32 Entries



21.3.2 EDMA Configuration

The EDMA configuration is determined according by the EDMA Command Register (n=0–1) (Table 1005 p. 1267). The registers listed below may be changed only when <eEnEDMA> in that register, is cleared, and the EDMA is disabled. Do not change these registers when <eEnEDMA> is set:

- SATAHC Configuration Register (Table 939 p. 1219)

- EDMA Configuration Register (n=0–1) (Table 996 p. 1258)
- EDMA Command Delay Threshold Register (n=0–1) (Table 1008 p. 1270)
- All registers in Table 1020, Shadow Register Block Registers Map, on page 1277, except that the host is allowed to change the <HOB> field (bit [7]) in the ATA Device Control register (offset 0x82120) while the EDMA is active.
- All Basic DMA registers (page 1233)
- All Serial-ATA Interface registers (page 1238)
  - FIS Interrupt Cause Register (n=0–1) (Table 983 p. 1252)
  - FIS Interrupt Mask Register (n=0–1) (Table 984 p. 1254)

21.3.3 EDMA Mode of Operation

21.3.3.1 Basic DMA Operation

When the <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) is cleared to 0, the EDMA is disabled, therefore, the request queue and the response queue are not in use. The Basic DMA may be controlled directly using the following registers:

- Basic DMA Command Register (n=0–1) (Table 961 p. 1233)
- Basic DMA Status Register (n=0–1) (Table 962 p. 1234)
- Descriptor Table Low Base Address Register (n=0–1) (Table 963 p. 1236)
- Descriptor Table High Base Address Register (n=0–1) (Table 964 p. 1237)
- SATAHC Interrupt Cause Register (Table 944 p. 1222)

The DMA is used to perform only DMA data transactions. The hard drive must be programmed by writing to the ATA task registers before activating basic DMA.

When in Basic DMA Operation mode, the commands are processed one by one: the host configures the device, configures the DMA and starts it. The DMA indicates completion of the data transaction by setting <SaCrpb0Done/DMA0Done> field in the SATAHC Interrupt Cause Register (Table 944 p. 1224) and an interrupt is generated. If an error occurs during execution of the data transfer, the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262) is updated with the error cause, the Basic DMA Status Register (n=0–1) (Table 962 p. 1234) is updated with the completion error, and the host is further responsible for error handling.

Host Initialization of Basic DMA Operation

The host initializes the DMA Read/Write operation as follows:

1. Initializes the device with the data transfer command.
2. Initializes the Physical Region Descriptor [PRD] in memory.
3. Initializes the Descriptor Table Low Base Address Register (n=0–1) (Table 963 p. 1236).
4. Initializes the Descriptor Table High Base Address Register (n=0–1) (Table 964 p. 1237).
5. Confirms that the <eEarlyCompletionEn> field in the EDMA Configuration Register (n=0–1) (Table 996 p. 1260) is clear to 0.
6. If Port Multiplier is used, initializes the <PMportTx> field in the Serial-ATA Interface Control Register (n=0–1) (Table 978 p. 1247).
7. Activates the Basic DMA by setting the control bits in the Basic DMA Command Register (n=0–1) (Table 961 p. 1233).

Basic DMA Read/Write Operation

The Basic DMA performs only the data transaction.

1. The Basic DMA performs the data transaction.
2. It sets <SaCrpb0Done/DMA0Done> and a maskable interrupt is generated.
3. The host is further responsible for the device completion status.

Stop Basic DMA

The host may stop the Basic DMA operation before the commands are completed.

1. The host clears the <Start> field in the Basic DMA Command Register (n=0–1) (Table 961 p. 1234).
2. If the <Start> field is cleared while the Basic DMA is still active, as indicated by the active bit in the Basic DMA Status Register (n=0–1) (Table 962 p. 1234), the Basic DMA command is aborted, and the data transferred may be discarded before reaching its destination.

21.3.3.2 Target Mode Operation

When <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) is cleared to 0 and the <ComChannel> field in the Serial-ATA Interface Configuration Register (n=0–1) (Table 967 p. 1238) is set to 1, a communication channel is opened with the Serial-ATA port of another Marvell® device that supports the Target mode. The communication channel is not symmetric. One side should be configured as an initiator (<TargetMode> field in the Serial-ATA Interface Configuration Register (n=0–1) (Table 967 p. 1238) is cleared to 0), and the other side is configured as a target (<TargetMode> is set to 1).

Full Communication Support

To create a full handshake between two Marvell devices that support Target Mode, use the device's two channels as follows:

1. In channel A—One SATA port of this device is configured as the initiator, while the companion SATA port of the other Marvell device is configured as the target.
2. In channel B—The other SATA port of this device is configured as the target, while the companion SATA port of the other Marvell device is configured as the initiator.

Initiate Basic DMA Read Operation

1. Initiator Host—Activate the Initiator Basic DMA.
2. Initiator Host—Send the Register Device to the Host FIS (Frame Information Structure) using the Vendor Unique interface with 10-byte command (see Section 21.5, Vendor Unique Frames, on page 377).
3. Target Transport—Update ATA task registers, set the port's <SaDevInterrupt0> field in the SATAHC Interrupt Cause Register (Table 944 p. 1222), and generate the interrupt.
4. Target Host—Send the Register Device to the Host FIS using the Vendor Unique interface with acknowledge.
5. Initiator Transport—Update the ATA task registers, optionally set the port's <SaDevInterrupt0> field, and generate the interrupt if specified in the Register Device to the Host FIS.
6. Target Host—Activate Basic DMA, set <eDMAActivate> field in the Serial-ATA Interface Control Register (n=0–1) (Table 978 p. 1247).
7. Target Transport—Send the data as configured in the target Basic DMA.
8. Initiator Basic DMA—Set the port's <SaCrpb0Done/DMA0Done> field and generate the interrupt to the initiator host when data transfer completes.
9. Target Basic DMA—Set the port's <SaCrpb0Done/DMA0Done> field and generate the interrupt to the target host when data transfer completes.

Initiate Basic DMA Write Operation

1. Initiator Host—Activate Initiator Basic DMA.
2. Initiator Host—Send Register Device to Host FIS using the Vendor Unique interface with 10-byte command (see Section 21.5, Vendor Unique Frames, on page 377).
3. Target Transport—Update ATA task registers and set the port's <SaDevInterrupt0> field in the SATAHC Interrupt Cause Register (Table 944 p. 1222) and generate the interrupt.

4. Target Host—Send Register Device to Host FIS using the Vendor Unique interface with acknowledge.
5. Initiator Transport—Update the ATA task registers and optionally set the port's <SaDevInterrupt0> field and generate the interrupt if specified in the Register Device to Host FIS.
6. Target Host—Activate the Basic DMA, send DMA Activate frame using the Vendor Unique interface.
7. Initiator Transport—Set the <eDMAActivate> field in the Serial-ATA Interface Control Register (n=0–1) (Table 978 p. 1247).
8. Initiator Transport—Send the data as configured in the initiator Basic DMA.
9. Initiator Basic DMA—Set the port's <SaCrpb0Done/DMA0Done> field and generate the interrupt to initiator host when the data transfer completes.
10. Target Basic DMA—Set the port's <SaCrpb0Done/DMA0Done> field and generate the interrupt to target host when the data transfer completes.

Note

- In this mode, the ATA task registers are updated when Register Device to Host FIS is received regardless to the value of <BSY> bit in the ATA Status register (see Table 1020, Shadow Register Block Registers Map, on page 1277).
- Link Errors while transmitting Vendor Unique FIS are also reported in the <LinkCtlTxErr> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262).
- For testing, this mode can be used to generate an External Loopback between the Serial-ATA ports of the same device.

21.3.3.3 Non-Queued DMA Commands

When the <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) is set to 1, the <eSATANatvCmdQue> field in the EDMA Configuration Register (n=0–1) (Table 996 p. 1261) is cleared to 0, and the <eQue> is clear to 0, the EDMA is in Non-queued mode. In this mode, the EDMA supports only ATA DMA commands. It performs the commands that reside in the CRQB one by one. The next command is issued to the device only when the previous command has completed and the CRPB is updated. In this mode, the EDMA uses the following commands.

- Read DMA
- Read DMA EXT
- Write DMA
- Write DMA EXT
- Read STREAM DMA
- Write DMA FUA EXT
- Write STREAM DMA

21.3.3.4 Queued DMA Commands

When the <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) is set to 1, the <eSATANatvCmdQue> field in the EDMA Configuration Register (n=0–1) (Table 996 p. 1261) is clear to 0, and the <eQue> is set to 1, ATA QDMA commands are performed. These commands allows the CPU to issue concurrent commands to the same device. Along with the command, the EDMA provides the <cDeviceQueTag> to the device to uniquely identify the command. When the device restores register parameters during the execution of the SERVICE command, this tag is restored. The EDMA identify the command according to the <cDeviceQueTag> and the incoming PM port and restores the command parameters to execute the data transaction. The SATA devices support up to 32 concurrent queued commands, and these commands may perform out of order.

In this modes the EDMA uses the following commands:

- Read DMA Queued
- Read DMA Queued EXT
- Write DMA Queued
- Write DMA Queued EXT
- Write DMA Queued FUA EXT

21.3.3.5 SATA Native Command Queuing

When the <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) is set to 1 and the <eSATANatvCmdQue> field in the EDMA Configuration Register (n=0–1) (Table 996 p. 1261) is set to 1, a streamlined command queuing model for SATA (SATA native command queuing) is supported. This model minimizes the required number of protocol round trips and reduces the incurred overhead.

These commands allows the CPU to issue concurrent commands to the same device. Along with the command, the EDMA provides the <cDeviceQueTag> as the tag of the command to uniquely identify the command. When the device restores register parameters, this tag is restored, The EDMA identify the command according to the <cDeviceQueTag> and the incoming PM port and restores the command parameters to execute the data transaction. The SATA devices support up to 32 concurrent queued commands, and these commands may perform out of order.

In this mode the EDMA uses the following commands:

- Read FPDMA Queued
- Write FPDMA Queued

21.3.4 EDMA Activation

The CPU activates the EDMA according to the following flow:

1. Verifies that the device is ready to receive data commands, the <DET> field in the SStatus Register (n=0–1) (Table 970 p. 1240) equals 3, and the fields <Busy> and <DRQ> in the device Status register are cleared.
2. Clears the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262) and clears the appropriate <SaCrpb0Done/DMA0Done> field in the SATAHC Interrupt Cause Register (Table 944 p. 1224).
3. Initializes the EDMA Configuration Register (n=0–1) (Table 996 p. 1258).
4. Clears the FIS Interrupt Cause Register (n=0–1) (Table 983 p. 1252).
5. Initialized the FIS Configuration Register (n=0–1) (Table 982 p. 1251).
6. Initializes the EDMA Response Queue In-Pointer Register (n=0–1) (Table 1003 p. 1266).
7. Initializes the EDMA Response Queue Out-Pointer Register (n=0–1) (Table 1004 p. 1267).
8. Initializes the EDMA Response Queue In-Pointer Register (n=0–1) (Table 1003 p. 1266).
9. Initializes the EDMA Response Queue Out-Pointer Register (n=0–1) (Table 1004 p. 1267).
10. Activates the EDMA by writing 1 to <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268).

While the EDMA is enabled, the host should not access the registers listed in Section 21.3.2, EDMA Configuration, on page 357. The CPU accesses these registers for direct access to the device when the EDMA is disabled. Accessing the above registers while EDMA is enabled—see the <eEnEDMA> field—will result in unpredictable behavior.

21.3.5 Commands Hot Insertion to EDMA Queue

Hot insertion of commands into the EDMA queue follows these steps:

1. Sets the valid Physical Region Descriptors [PRD] for the new commands.
2. Initializes the new commands in the request queue.
3. Updates the EDMA Request Queue In-Pointer Register (n=0–1) (Table 1000 p. 1265), to enable EDMA access to the new CRQBs in the request queue.

21.3.6 Stop EDMA

To stop the EDMA operation, the CPU sets the <eDsEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to 1. The EDMA stops queue processing, aborts the current command, and clears <eEnEDMA>.

If EDMA is aborted during command processing, the host must set the <eAtaRst> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to recover.

21.3.7 Restart EDMA

To restart the queue, the CPU must follow the EDMA activation flow (see Section 21.3.4, EDMA Activation, on page 361).

21.3.8 Device Link Disconnect

See Section , Device Disconnect, on page 365 for the disconnect procedure.

Since loss of the link can occur at any time during EDMA programming, when the CPU receives a link-down interrupt, it must wait for re-establishment of the link (see Section 21.3.9, Device Link Connect, on page 362).

21.3.9 Device Link Connect

When the link to the device is renewed, the EDMA sets the <eDevCon> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264).

Device hard reset (setting the <eAtaRst> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) and device initialization are required before any attempt to access the device.

21.3.10 EDMA Read Burst Limit

The <eRdBSz> field in the EDMA Configuration Register (n=0–1) (Table 996 p. 1261) defines the maximum burst read transactions SATAHC initiates towards the Mbus. The EDMA supports a maximum read burst size of 128B.

21.3.11 EDMA Write Burst Limit

The EDMA support a maximum write burst size of 128B.

21.3.12 Port Multiplier Support

The device supports the Port Multiplier (PM) ingredient in the following modes:

21.3.12.1 Port Multiplier—Command Based Switching

When the <eEDMAFBS> field in the EDMA Configuration Register (n=0–1) (Table 996 p. 1260) is cleared to 0, the EDMA Performs Command Based Switching as defined in SATA working group PM definition. Field <cPMport> in each command in the request queue (CRQB) is used to define the specific port in the Port Multiplier (PM) ingredient that belongs to this command. The EDMA is further responsible for forwarding the commands to the correct target device. In this mode, Non

Copyright © 2014 Marvell

Queued DMA commands are supported (see Section 21.3.3.3, Non-Queued DMA Commands, on page 360).

21.3.12.2 Port Multiplier—FIS-Based Switching

The EDMA performs FIS-based switching, as defined in SATA working group PM definition. In this mode, the EDMA issues multiple outstanding commands across multiple devices at the same time. The overall system performance increases significantly with this type of switching.

The following commands are supported in this mode:

- Non Queued DMA commands (see Section 21.3.3.3, Non-Queued DMA Commands, on page 360)
- Tag Command Queuing (TCQ) commands (see Section 21.3.3.4, Queued DMA Commands, on page 360)
- Native Command Queuing commands (see Section 21.3.3.5, SATA Native Command Queuing, on page 361)

Note

The mode is selected before EDMA is enabled. It must not be changed when the <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) is set to 1.

21.3.13 Asynchronous Device Notification

When Set Device Bits (SDB) FIS is received with N bit set to 1, the following occurs:

In the FIS Configuration Register (n=0–1) (Table 982 p. 1251):

If <FISWait4RdyEn> is cleared to 0:

The device ignores the FIS.

If bit <FISWait4RdyEn>[1] is set to 1:

- Bit [1] of the <FISWait4HostRdy> field in the FIS Interrupt Cause Register (n=0–1) (Table 983 p. 1253) is set.
- The <eTransInt> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264) is also set if the corresponding bit in the FIS Interrupt Mask Register (n=0–1) (Table 984 p. 1254) is set to 1.

21.3.14 EDMA Interrupts

21.3.14.1 Error indication

The EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262), provides the various error indications that may occur during DMA operation. For more information (see Section 21.3.15, Error Handling, on page 364).

In addition, the DMA contains a EDMA Interrupt Error Mask Register (n=0–1) (Table 998 p. 1265). This register may be used to mask the error bits in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262). If one (or more) of the unmasked bits in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262) is set, an error indication is propagated to the SATAHC Main Interrupt Cause Register (Table 945 p. 1224).

21.3.14.2 Command Completion Indication

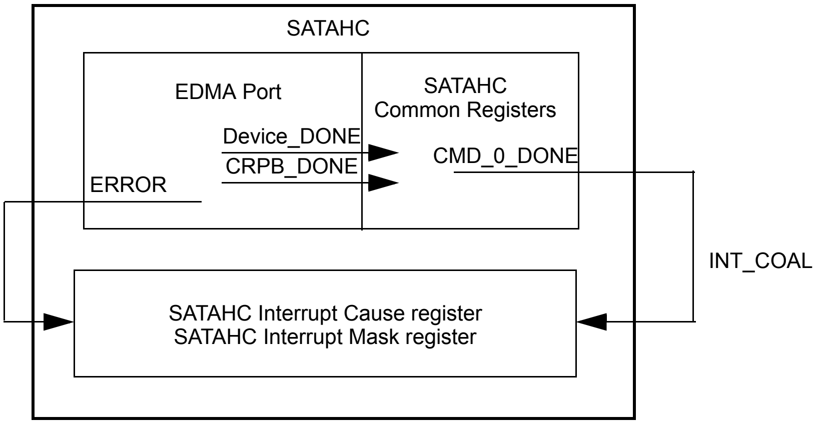
The SATAHC Interrupt Cause Register (Table 944 p. 1222), resides in the SATAHC arbiter. The command completion indications are propagated from the EDMAs to the appropriate bit in this register. The indications from the register are further propagated to the SATAHC Main Interrupt

Cause Register (Table 945 p. 1224).

When the EDMA completes an ATA transaction:

- The last data leaves the device.
- The CRPB is updated.
- The EDMA indicates the appropriate bit in the SATAHC Interrupt Cause Register (Table 944 p. 1222), and
- An interrupt indication is propagated to SATAHC Main Interrupt Cause Register (Table 945 p. 1224).

Figure 90: EDMA Interrupt Hierarchy[1]



21.3.14.3 Interrupt Coalescing

Since the SATA ports provide a high data rate, it is important to reduce the number of interrupts that the SATA EDMAs may generate. The device provides an interrupt coalescing mechanism that sets the interrupt coalescing bit in the SATAHC Interrupt Cause Register (Table 944 p. 1222), and propagates an interrupt indication if one of the following is true:

- The number of EDMA commands reached the SATAHC interrupt coalescing threshold value (see the SATAHC Interrupt Coalescing Threshold Register (Table 942 p. 1221).
- At least one EDMA commands has completed and the time that passed since its completion reached the SATAHC interrupt time threshold value (see the SATAHC Interrupt Time Threshold Register (Table 943 p. 1221)).

21.3.14.4 Device Interrupt

When the EDMA is active, the device interrupt request is masked. When the EDMA is disabled and the device interrupt request is active, a separate bit is set in the SATAHC Interrupt Cause Register (Table 944 p. 1222) and a command completion indication is propagated to the SATAHC Main Interrupt Cause Register (Table 945 p. 1224).

21.3.15 Error Handling

Error indications from all layers are gathered in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262).

[1] All interrupt indications from the SATAHC are propagated to the SATAHC Main Interrupt Cause Register (Table 945 p. 1224).

21.3.15.1 List Of Unrecoverable Errors

In the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262):

- <eDevDis>
- <eIORdyErr>
- <LinkCtlRxErr>
- <LinkDataRxErr>
- <LinkDataTxErr>
- <TransProtErr>

When an unrecoverable error indication is set from the list above, the EMDA is self disabled and the host must set bit <eAtaRst> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to recover.

21.3.15.2 PHY Layer Errors

SError Register Errors

For PHY layer errors, see the SError Register (n=0–1) (Table 971 p. 1240).

The <SerrInt> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264) is set when at least one bit in SError Register (n=0–1) is set to 1, and the corresponding bit in the SError Register (n=0–1) is enabled.

Device Disconnect

When the device is disconnected, the EDMA halts and:

- Sets the <eDevDis> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264).
- Disables the EDMA operation by clearing bit <eEnEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268).
- Sets the <eSelfDis> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264).

Host must set the <eAtaRst> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to recover.

The CPU is responsible for error recovery.

21.3.15.3 Link Layer Errors

Serial-ATA II Link Layer Error During Reception of a Control Frame

- **Transient Errors:** When the following errors occur during control FIS reception. The link layer responds with R_ERR to the received frame. The transport layer drop this frame and waits for re-transmission of the frame. This may be a transient error. The EDMA ignore these type of errors and proceeds with normal operation.
  - Serial-ATA CRC error occurs; the <LinkCtlRxErr> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1263) bit [0] is set.
  - Internal FIFO error occurs; the <LinkCtlRxErr> bit [1] is set to 1.
  - Link state errors, coding errors, or running disparity errors occur. Bit [3] of the <LinkCtlRxErr> field is set.
- **Non Transient Errors:** When the following error occurs during control FIS reception. The transport layer goes to protocol error state. The host must sets bit <eAtaRst> to recover.
  - The Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [2] in the <LinkCtlRxErr> field and the <TransProtErr> field are set.

Serial-ATA II Link Layer Error During Reception of a Data Frame

When the Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device, the transport layer goes to protocol error state, bit [2] in the <LinkDataRxErr>field and the <TransProtErr> field are set. The host must set bit <eAtaRst> to recover.

When the following errors occur during data FIS reception, the link layer responds with R_ERR to the received frame. The transport layer ignores the error but the EDMA is self disabled.

- Serial-ATA CRC error occurs. Bit [0] of the <LinkDataRxErr> field is set.
- Internal FIFO error occurs. Bit [1] in the <LinkDataRxErr> field is set.
- Link state errors, coding errors, or running disparity errors occur. Bit [3] in the <LinkDataRxErr> field is set.

Serial-ATA II Link Layer Error During Transmission of a Control Frame

When the following errors occur during control FIS transmission, the transport layer re-transmits the frame. This may be a transient error.

- Serial-ATA CRC error occurs. Bit [0] in the <LinkCtlRxErr> field is set.
- Internal FIFO error occurs. Bit [1] in the <LinkCtlRxErr> field is set.
- The Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [2] in the <LinkCtlTxErr> field is set.
- Link layer accepts DMAT primitive from the device. Bit [3] in the <LinkCtlTxErr> field is set.
- FIS transmission is aborted due to collision with received traffic. Bit [4] in the <LinkCtlTxErr> field is set.

Serial-ATA II Link Layer Error During Transmission of a Data Frame

When the following errors occur during data FIS transmission, the transport layer ignores the error, but the EDMA is self disabled.

- Serial-ATA CRC error occurs. Bit [0] in the <LinkCtlTxErr> field is set.
- Internal FIFO error occurs. Bit [1] in the <LinkCtlTxErr> field is set.
- The Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [2] in the <LinkCtlTxErr> field is set.
- Link layer accepts DMAT primitive from the device. Bit [3] in the <LinkCtlTxErr> field is set.
- FIS transmission is aborted due to collision with received traffic. Bit [4] in the <LinkCtlTxErr> field is set.

21.3.15.4 Transport Layer Errors

Serial-ATA II Transport Layer Protocol Non Transient Errors

When a violation of the Serial-ATA protocol was detected, the transport layer goes to protocol error state and sets the <TransProtErr> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262) and the EDMA is self disabled. The Host must set the <eAtaRst> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to recover. This error state can arise from invalid or poorly formed FISs being received, from invalid state transitions, or from other causes.

The host must set <eAtaRst> to recover.

The CPU is responsible for error recovery.

Device Error Indications

Device Errors in Non Queued or Queued DMA Commands—FIS-Based Switching Mode Disabled

FIS-Based Switching is disabled when the <eEDMAFBS> field in the EDMA Configuration Register (n=0–1) (Table 996 p. 1260) is cleared.

Note

See Section 21.3.3.3, Non-Queued DMA Commands, on page 360 and Section 21.3.3.4, Queued DMA Commands, on page 360.

Bit [2] in the <eHaltMask> field in the EDMA Halt Conditions Register (n=0–1) (Table 1009 p. 1271) should be set to 1:

When bit <Error> in the ATA status register is set to 1:

- The following registers are updated with the command information:
    - Shadow Register Block Registers Map (Table 1020 p. 1277)
    - Serial-ATA Interface Status Register (n=0–1) (Table 980 p. 1249)
    - EDMA Status Register (n=0–1) (Table 1006 p. 1269)
    - EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262)
- <eDevErr> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264) is set.
- The EDMA halts.

Device Error Indication in Serial-ATA Native Command Queuing

See Section 21.3.3.5, SATA Native Command Queuing, on page 361 and Section 21.3.12.2, Port Multiplier—FIS-Based Switching, on page 363.

Bit [2] in the <eHaltMask> field in the EDMA Halt Conditions Register (n=0–1) (Table 1009 p. 1271) should be set to 1:

When bit Error in the ATA status register is set to 1, the following registers are updated with the command information:

- Serial-ATA Interface Status Register (n=0–1) (Table 980 p. 1249)
- EDMA Status Register (n=0–1) (Table 1006 p. 1269)
- EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262)
- Host identify which drive causes the error via the <PortNumDevErr> field in the Serial-ATA Interface Test Control Register (n=0–1) (Table 979 p. 1248)

EDMA does NOT update CRPB with the error indication.

The host must:

1. Wait for completion of all outstanding commands associate to other devices (that did not experience a device error).
2. Check the <eCacheEmpty> field in the EDMA Status Register (n=0–1) (Table 1006 p. 1269). If cleared, then wait for another Device error interrupt.
3. Wait for the <EDMAIdle> field in the EDMA Status Register (n=0–1) (Table 1006 p. 1269) to clear. If set, then wait for another Device error interrupt.
   - Good CRPBs may be received.
4. Set the <eDsEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to disable EDMA operation.
5. Wait for clearing of the <eEnEDMA> field.
6. Issue a read log command to the drive and perform error handling accordingly.

Device Errors in Non Queued or Queued DMA Commands in Port Multiplier—FIS-Based Switching Mode Enabled

See Section 21.3.3.3, Non-Queued DMA Commands, on page 360, Section 21.3.3.4, Queued DMA Commands, on page 360 and Section 21.3.12.2, Port Multiplier—FIS-Based Switching, on page 363

Bit [2] in the <eHaltMask> field in the EDMA Halt Conditions Register (n=0–1) (Table 1009 p. 1271) should be cleared to 0:

Bit [0] of the <FISWait4RdyEn> field in the FIS Configuration Register (n=0–1) (Table 982 p. 1252) should be set to 1:

When the <Error> field in the ATA status register is set to 1 via the Register-Device to Host FIS, the following registers are updated with the command information:

- Bit [0] of the <FISWait4HostRdy> field in the FIS Interrupt Cause Register (n=0–1) (Table 983 p. 1253) is set to 1, and the transport layer blocks reception of any new FIS until the host clears this bit.
- EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262).
- The EDMA updates the CRPB with the error indication.

The host must:

1. Get detailed error information from the Shadow Register Block (see Shadow Register Block Registers Map (Table 1020 p. 1277)).
2. Detect the aborted commands for the disk that experience error according to the EDMA NCQ0 Done/TCQ0 Outstanding Status Register (n=0–1) (Table 1010 p. 1271).
3. Clear the <eDevErr> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264).
4. Clear bit [0] in the <FISWait4HostRdy> field in the FIS Interrupt Cause Register (n=0–1) (Table 983 p. 1253).
5. Wait for completion of all outstanding commands (that is, any commands to the EDMA that did not complete successfully and were not aborted or failed).
6. Set bit <eDsEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to disable the EDMA operation.
7. Wait for clearing of the <eEnEDMA> field.

21.3.15.5 DMA Errors

Internal Parity Error

The device's SATAHC is parity protected on internal memories.

The internal SRAMs contain a parity bit per entry (minimal transaction width). This bit is calculated and inserted on every write to the internal SRAMs. This bit is verified against the data when reading from the internal SRAMs.

The parity bit also indicates if errors occurred during the PCI transaction.

21.3.16 EDMA Data Structures

21.3.16.1 Command Request Queue

The request queue is the interface that the CPU software uses to request data transactions between the system memory and the device. The request queue has a length of 32 entries. The request queue is a circular queue (FIFO) whose location is configured by the EDMA Request Queue In-Pointer Register (n=0–1) (Table 1000 p. 1265), and the EDMA Request Queue Out-Pointer Register (n=0–1) (Table 1001 p. 1266).

- A queue is empty when Request Queue Out-pointer reaches to the Request Queue In-pointer.
- A queue is full when Request Queue In-pointer is written with the same value as the Request Queue Out-pointer. A full queue contains 32 entries.
- A queue contains N entries when the Request Queue Out-pointer is N less than the Request Queue In-pointer, taking into account the wraparound condition.

See Figure 88, Command Request Queue—32 Entries, on page 357.

Each 32-byte EDMA Command Request Block (CRQB) entry consists of EDMA parameters and commands for the ATA device. The CRQB data structure is written by the CPU. Table 84 provides a map of the CRQB data structure registers.

21.3.16.2 EDMA Command Request Block (CRQB) Data

Table 84: EDMA CRQB Data Structure Map

Register	Offset	Page
CRQB DW0—cPRD Descriptor Table Base Low Address	Offset: 0x00	Table 85, p. 369
CRQB DW1—cPRD Descriptor Table Base High Address	Offset: 0x04	Table 86, p. 369
CRQB DW2—Control Flags	Offset: 0x08	Table 87, p. 370
CRQB DW3—Data Region Byte Count	Offset: 0x0C	Table 88, p. 370
CRQB DW4—ATA Command	Offset: 0x10	Table 89, p. 371
CRQB DW5—ATA Command	Offset: 0x14	Table 90, p. 371
CRQB DW6—ATA Command	Offset: 0x18	Table 91, p. 371
CRQB DW7—ATA Command	Offset: 0x1C	Table 92, p. 372

Table 85: CRQB DW0—cPRD Descriptor Table Base Low Address
Offset: 0x00

Bits	Field	Description
31:0	cPRD[31:0]	CRQB ePRD. When <cPRDMode> is cleared to 0: The CPU at initialization should construct a ePRD table in memory. This table contains consecutive descriptors that describe the data buffers allocated in memory for this command. This DWORD contains bit [31:4] of the physical starting address of this table. Bits [3:0] must be 0x0. When <cPRDMode> is set to 1: This DWORD contains bits [31:1] of the physical starting address of a data region in system memory. Bit [0] must be 0.

Table 86: CRQB DW1—cPRD Descriptor Table Base High Address
Offset: 0x04

Bits	Field	Description
31:0	cPRD[63:32]	Reserved Must be cleared to 0x0.

Copyright © 2014 Marvell

Table 87: CRQB DW2—Control Flags
Offset: 0x08

Bits	Field	Description
0	cDIR	CRQB Direction of Data Transaction 0 = System memory to Device 1 = Device to system memory
5:1	cDeviceQueTag	CRQB Device Queue Tag This field contains the Queued commands used as tags attached to the command provided to the drive.
11:6	Reserved	Reserved Must be 0.
15:12	cPMport	PM Port Transmit This field specifies the Port Multiplier (PM) port (bits [11:8] in DW0 of the FIS header) inserted into the FISs transmission associate to this command.
16	cPRDMode	CRQB PRD Mode This bit defines how the physical data that resides in the system memory is described. 0 = PRD tables are being used. <cPRD[31:0]> and <cPRD[63:32]> provide the ePRD table starting address. 1 = Single data region, <cPRD[31:0]> and <cPRD[63:32]> provide its starting address. <cDataRegionByteCount> provides its length.
21:17	cHostQueTag	CRQB Host Queue Tag This 5-bit field contains the host identification of the command.
31:22	Reserved	Reserved

Table 88: CRQB DW3—Data Region Byte Count
Offset: 0x0C

Bits	Field	Description
15:0	cDataRegionByteCount	Data Region Byte Count When <cPRDMode> is cleared to 0: This field is reserved. When <cPRDMode> is set to 1: This field contains the count of the region in bytes. Bit [0] is force to 0. There is a 64 KB maximum. A value of 0 indicates 64 KB. The data in the buffer must not cross the boundary of the 32-bit address space; that is, the 32-bit high address of all data in the buffer must be identical.
31:16	Reserved	Reserved

The naming of the fields in the next four tables complies with the Serial-ATA convention. The corresponding name according to the ATA convention appears in parentheses.

Table 89: CRQB DW4—ATA Command
Offset: 0x10

Bits	Field	Description
15:0	Reserved	Reserved
23:16	Command	This field contains the contents of the Command register of the Shadow Register Block (see Table 1020 on page 1277).
31:24	Features	This field contains the contents of the Features (Features Current) register of the Shadow Register Block.

Table 90: CRQB DW5—ATA Command
Offset: 0x14

Bits	Field	Description
7:0	Sector Number	This field contains the contents of the Sector Number (LBA Low Current) register of the Shadow Register Block (see Table 1020 on page 1277).
15:8	Cylinder Low	This field contains the contents of the Cylinder Low (LBA Mid Current) register of the Shadow Register Block.
23:16	Cylinder High	This field contains the contents of the Cylinder High (LBA High Current) register of the Shadow Register Block.
31:24	Device/Head	This field contains the contents of the Device/Head (Device) register of the Shadow Register Block.

Table 91: CRQB DW6—ATA Command
Offset: 0x18

Bits	Field	Description
7:0	Sector Number (Exp)	This field contains the contents of the Sector Number (Exp) (LBA Low Previous) register of the Shadow Register Block (see Table 1020 on page 1277).
15:8	Cylinder Low (Exp)	This field contains the contents of the Cylinder Low (Exp) (LBA Mid Previous) register of the Shadow Register Block
23:16	Cylinder High (Exp)	This field contains the contents of the Cylinder High (Exp) (LBA High Previous) register of the Shadow Register Block.
31:24	Features (Exp)	This field contains the contents of the Features (Exp) (Features Previous) register of the Shadow Register Block.

Table 92: CRQB DW7—ATA Command
Offset: 0x1C

Bits	Field	Description
7:0	Sector Count	This field contains the contents of the Sector Count (Sector Count Current) register of the Shadow Register Block (see Table 1020 on page 1277).
15:8	Sector Count (Exp)	This field contains the contents of the Sector Count (exp) (Sector Count Previous) register of the Shadow Register Block
31:16	Reserved	Reserved

When the EDMA is in Non-Queued mode:

The following commands are supported.

- READ DMA
- READ DMA EXT
- READ STREAM DMA
- WRITE DMA
- WRITE DMA EXT
- WRITE DMA FUA EXT
- WRITE STREAM DMA

When the EDMA is in Queued mode:

The following commands are supported.

- READ DMA QUEUED
- READ DMA QUEUED EXT
- WRITE DMA QUEUED
- WRITE DMA QUEUED EXT
- WRITE DMA QUEUED FUA EXT

When the EDMA is in Native Command Queuing mode:

The following commands are supported.

- Read FPDMA Queued
- Write FPDMA Queued

Note

Other commands cause unpredictable results.

21.3.16.3 EDMA Physical Region Descriptors (ePRD) Table Data Structure

The physical memory region to be transferred is described by the EDMA Physical Region Descriptor [ePRD] for DWORDs 0–3. The data transfer proceeds until all regions described by the ePRDs in the table have been transferred. The starting address of this table must be 16B aligned, i.e., bits [3:0] of the table base address must be 0x0.

Note

The total number of bytes in the PRD table (total byte count in DMA command) must be 4-byte aligned.

Table 93: ePRD Table Data Structure Map

Descriptor	Table, Page
ePRD DWORD 0	Table 94, p. 373
ePRD DWORD 1	Table 95, p. 373
ePRD DWORD 2	Table 96, p. 373
ePRD DWORD 3	Table 97, p. 374

Table 94: ePRD DWORD 0

Bits	Field	Description
0	Reserved	Reserved
31:1	PRDBA[31:1]	The byte address of a physical memory region corresponds to address bits [31:1].

Table 95: ePRD DWORD 1

Bits	Field	Description
15:0	ByteCount	Byte Count The count of the region in bytes. Bit 0 is force to 0. There is a 64-KB maximum. A value of 0 indicates 64 KB. The data in the buffer must not cross the boundary of the 32-bit address space, that is the 32-bit high address of all data in the buffer must be identical.
30:16	Reserved	Reserved
31	EOT	End Of Table The data transfer operation terminates when the last descriptor has been retired. 0 = Not end of table 1 = End of table **NOTE:** The total number of bytes in the PRD table (total byte count in DMA command) must be 4-byte aligned.

Table 96: ePRD DWORD 2

Bits	Field	Description
31:0	PRDBA[63:32]	The byte address of a physical memory region corresponds to bits [64:32]. Must be cleared to 0x0.

Copyright © 2014 Marvell

Table 97: ePRD DWORD 3

Bits	Field	Description
31:0	Reserved	Reserved

21.3.16.4 Command Response Queue

The response queue is the interface that the EDMA uses to notify the CPU software that a data transaction between the system memory and the device was completed. The response queue is a 32 entry, circular queue (FIFO) whose location is configured by the EDMA Request Queue In-Pointer Register (n=0–1) (Table 1000 p. 1265) and the EDMA Request Queue Out-Pointer Register (n=0–1) (Table 1001 p. 1266).

The queue status is determined by comparing the two pointers:

- A queue is empty when the Response Queue Out-pointer reaches the Response Queue In-pointer.
- A queue is full when Response Queue In-pointer is written with same value as a Response Queue Out-pointer. A full queue contains 32 entries.
- A queue contains N entries when the Response Queue Out-pointer is N less than the Response Queue In-pointer, taking into account the wraparound condition.

Note

The EDMA may write over existing entries when the queue is full.

See Figure 89, Command Response Queue—32 Entries, on page 357.

Each 8-byte command response entry consists of command ID, response flags, and a timestamp (see Table 98, "EDMA CRPB Data Structure Map," on page 374). The CRPB data structure, described in Table 84, "EDMA CRQB Data Structure Map," on page 369, is written by the EDMA.

21.3.16.5 EDMA Command Response Block (CRPB) Data

Table 98 provides a map of the EDMA command response block data structure tables.

Table 98: EDMA CRPB Data Structure Map

Register	Offset	Table, Page
CRPB ID Register	Offset: 0x00	Table 99, p. 375
CRPB Response Flags Register	Offset: 0x02	Table 100, p. 375
CRPB Time Stamp Register	0Offset: x04	Table 101, p. 375

Table 99: CRPB ID Register
Offset: 0x00

Bits	Field	Description
4:0	cHostQueTag	CRPB ID In queued DMA commands, these bits are used as a tag. This field contains the host identification of the command. These bits are copied from field <cHostQueTag> of Table 87, CRQB DW2—Control Flags, on page 370.
15:5	Reserved	Reserved

Table 100: CRPB Response Flags Register
Offset: 0x02

Bits	Field	Description
6:0	cEdmaSts	CRPB EDMA Status This field contains a copy of the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1262) bits [6:0] accepted in the last command. **NOTE:** When the EDMA is in NCQ mode, ignore this field since the value of this field may reflect the status of other commands.
7	Reserved	Reserved This bit is always 0.
15:8	cDevSts	CRPB Device Status This field contains a copy of the device status register accepted in the last read of the register from the device.

Table 101: CRPB Time Stamp Register
0Offset: x04

Bits	Field	Description
31:0	Reserved	—

21.4 Built-In Self Test (BIST)

21.4.1 Far-End Loopback

This mode is performed according to *SATA 1.0 specification*, section 8.5.7. *BIST activate FIS*.

The supported BIST patterns are:

- L: Far-end Retimed Loopback
- TS: Transmit Only and Scrambling Bypass
- TSA: Transmit Only, Scrambling Bypass, and Align Bypass

21.4.2 BIST as the Initiator Side

The following flow should be performed by the host CPU:

- Send BIST Activate FIS using vendor unique interface to initiate BIST mode over the SATA link (see Section 21.5, Vendor Unique Frames, on page 377).
- Set the <BISTMode> field in the BIST Control Register (n=0–1) (Table 974 p. 1245) to 1 to determine FIS direction.
- Initiate <BISTPattern> according to the transmitted BIST Activate FIS.
- Initiate the <BistDw1> field in the BIST-Dword1 Register (n=0–1) (Table 975 p. 1245) according to the transmitted BIST Activate FIS.
- Initiate the <BistDw2> field in the BIST-Dword2 Register (n=0–1) (Table 976 p. 1246) according to the transmitted BIST Activate FIS.
- Set the <BISTEn> field in the BIST Control Register (n=0–1) (Table 974 p. 1245) to activate the pattern comparator operation.
- Read the <BISTResult> status to determine BIST test passes or not.
- Set the <eAtaRst> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268) to exit both sides of the link from BIST mode.

21.4.3 BIST as the Target Side

The following flow should be performed, when the Serial-ATA port receives BIST Activate FIS:

1. Host CPU must set bit [1] in the <FISWait4Rdy> field in the FIS Interrupt Cause Register (n=0–1) (Table 983 p. 1254) to 1.
2. The FIS content is updated in the FIS Dword0 Register (n=0–1) (Table 985 p. 1255) through FIS Dword6 Register (n=0–1) (Table 991 p. 1256):
3. Bit [3] in field <FISWait4HostRdy> is set.
4. Bit [8] <eTransInt> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264) is also set if the corresponding bit in the FIS Interrupt Mask Register (n=0–1) (Table 984 p. 1254) is set to 1.
5. Host CPU must clear bit [1] in the <FISWait4Rdy> field in the FIS Interrupt Cause Register (n=0–1) (Table 983 p. 1254).
6. Host CPU must clear bit [8] in the <eTransInt> field in the EDMA Interrupt Error Cause Register (n=0–1) (Table 997 p. 1264).
7. Host CPU sets the <BISTMode> field in the BIST Control Register (n=0–1) (Table 974 p. 1245) to 0 to determine FIS direction.
8. Host CPU initiates <BISTPattern> according to the received BIST Activate FIS.
9. Host CPU initiates the <BistDw1> field in the BIST-Dword1 Register (n=0–1) (Table 975 p. 1245) with the contents matching the received BIST Activate FIS.

Copyright © 2014 Marvell

10. Host CPU initiates the <BistDw2> field in the BIST-Dword2 Register (n=0–1) (Table 976 p. 1246) with the contents matching the received BIST Activate FIS.
11. Host CPU sets the <BISTEn> field in the BIST Control Register (n=0–1) (Table 974 p. 1245) to activate the internal pattern generators to send the data stream onto the Serial-ATA link.

21.5 Vendor Unique Frames

The following flow should be performed to activate transmission of Vendor Unique FIS.

1. Wait until all pending commands in the EDMA are completed.
2. Disable the EDMA, set the <eDsEDMA> field in the EDMA Command Register (n=0–1) (Table 1005 p. 1268).
3. Verify the EDMA is disabled, the <eEnEDMA> field is cleared.
4. Verify the Transport Layer is in idle, the <TransFsmSts> field in the Serial-ATA Interface Status Register (n=0–1) (Table 980 p. 1249) is cleared.
5. Set Vendor Unique Mode. Write 1 to the <VendorUqDn> field in the Serial-ATA Interface Status Register (n=0–1) (Table 980 p. 1250).
6. Insert data into the Vendor Unique Register (n=0–1) (Table 981 p. 1251).
7. Repeat step 6 until all data except the last DWORD in the vendor unique FIS is transferred. Note that according to the Serial-ATA protocol the FIS length is limited to 8 KB.
8. Write 1 to bit <VendorUqSend> field in the Serial-ATA Interface Control Register (n=0–1) (Table 978 p. 1247).
9. Write last DWORD in the FIS to Complete FIS transmission.
10. Wait for transmission completion. The <VendorUqDn> or the <VendorUqErr> fields are set to 1.
11. Verify successful transmission of the FIS. Bit <VendorUqErr> is cleared.
12. Clear Vendor Unique Mode. Write 0 to the <VendorUqMd> field in the Serial-ATA Interface Control Register (n=0–1) (Table 978 p. 1247).

22 Serial Peripheral Interface (SPI)

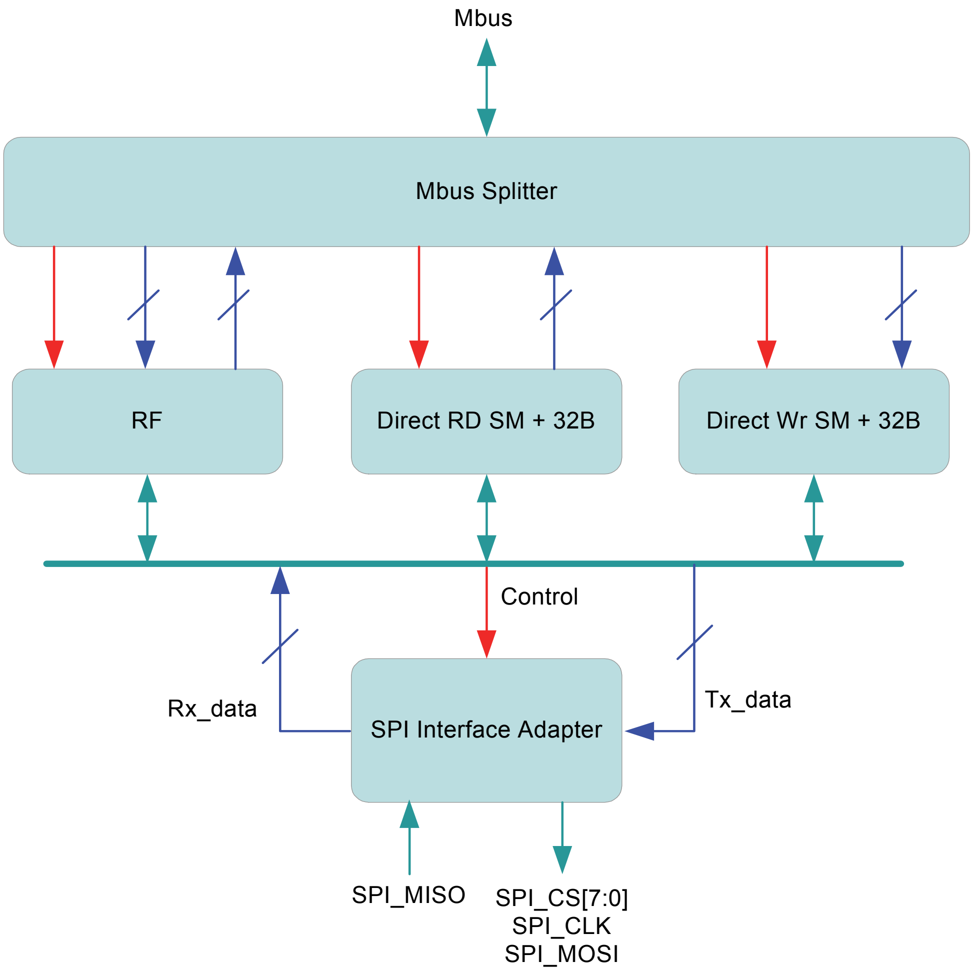
This section describes the general-purpose Serial Peripheral Interface (SPI) ports that are integrated into the device.

The SPI is a synchronous serial data protocol used for transferring data simply and quickly from one device to another. With an SPI connection, there is always one Master device that controls the peripheral devices (Slaves). Data over the SPI bus is sent and received in parallel and is related to the internally generated baud rate clock.

The SPI registers are located in Appendix A.12, Serial Peripheral Interface (SPI) Registers, on page 1278.

Figure 91 shows a block diagram of the SPI controller.

Figure 91: SPI Controller Block Diagram



Copyright © 2014 Marvell

22.1 Features

The main features of the SPI include:

- Master-Slave protocol:
  - Source synchronous bus that allows the Master device to toggle the clock during transactions.
  - As data is being clocked out, new data is clocked in
  - The slaves are controlled by the master clock, and may not manipulate the clock
- Configurable clock rate with extra flexible speed points (down to as low as Core clock/1920)
- Configurable clock inactive state polarity (CPOL)
- Configurable data driving/sampling phase (CPHA)
- Supports direct access to SPI Slave:
  - CPU direct access to the memory space of the SPI
  - Exchange of up to 32B burst in a single CPU transaction.
  - Configurable Read opcode
  - Configurable Write opcode
  - Configurable address length (1–4 bytes)
  - Configurable presence of a dummy byte between address phase and data phase
- Supports Indirect access to SPI Slave: Exchange of 1 or 2B (read or write) using register access
- Chip Selects:
  - Support of up to 8 slave devices with dedicated chip selects
  - Boot considerations:
  - Supports boot 24 or 32-bit address
  - Default read opcode is 0x3
  - Default timing mode is 0 (CPOL = 0, CPHA = 0)
  - Default clock rate is Core clock/26

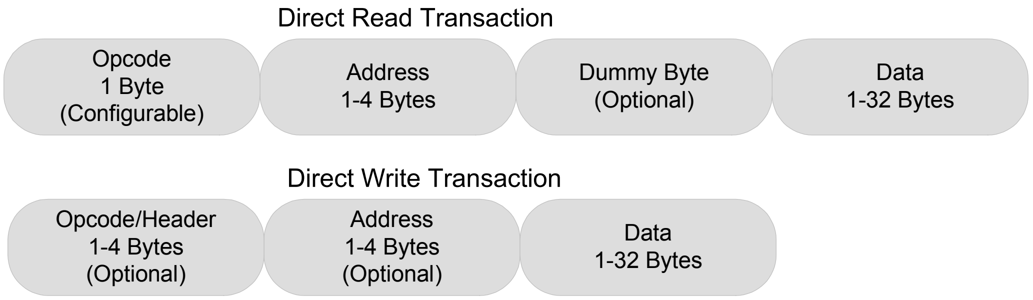
22.2 Functional Description

The SPI Slaves can be accessed directly or indirectly.

- Use the indirect access for most of the slave configurations
- Use the direct access to reduce software overhead for bulk read and write operations.

Figure 92 illustrates the direct access transaction level (read and write).

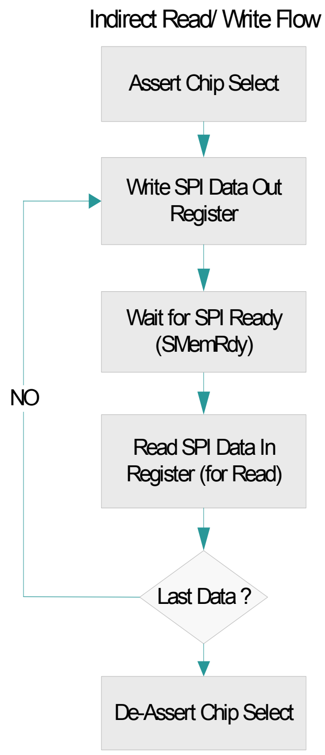
Figure 92: SPI Functional Diagram—Direct Access



Copyright © 2014 Marvell

Figure 93 illustrates indirect access transaction flow (read and write). It is possible for the software to control Chip-Select manually, or it can be asserted automatically for each transaction.

Figure 93: SPI Functional Diagram—Indirect Access


Indirect Read/ Write Flow
Assert Chip Select
Write SPI Data Out Register
Wait for SPI Ready (SMemRdy)
Read SPI Data In Register (for Read)
Last Data ?
NO
De-Assert Chip Select

22.3 SPI Modes

Bus Signals

SPI signals are multiplexed on the Multi-Purpose Pins (MPPs). For full details, refer to the *Hardware Specifications* of the device.

Modes

The SPI supports 4 clock modes configurable by <CPOL> and <CPHA> field in the SPI Interface Configuration Register (N=0–1) (Table 1023 p. 1280).

CPOL defines the clock inactive state (that is, if upon chip select (CS) assertion SPI_CLK is high or low).

CPHA selects between 2 options to transmit and sample data on SPI_CLK edges:

- CPHA = 0: Data is transmitted on even edges and sampled on odd edges, or
- CPHA = 1: Data is transmitted on odd edges and sampled on even edges.

Copyright © 2014 Marvell

Figure 94 shows the SPI timing in CPHA=0 for both CPOL=0 and CPOL=1

Figure 94: SPI Timing for CPHA=0

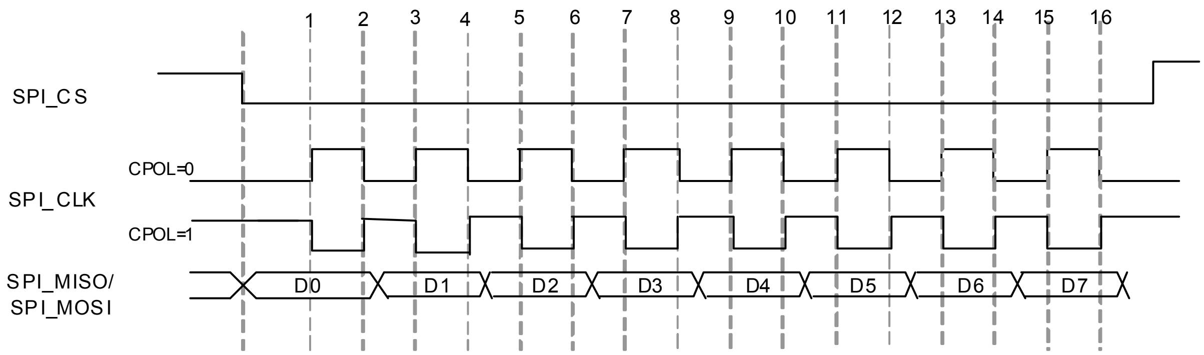
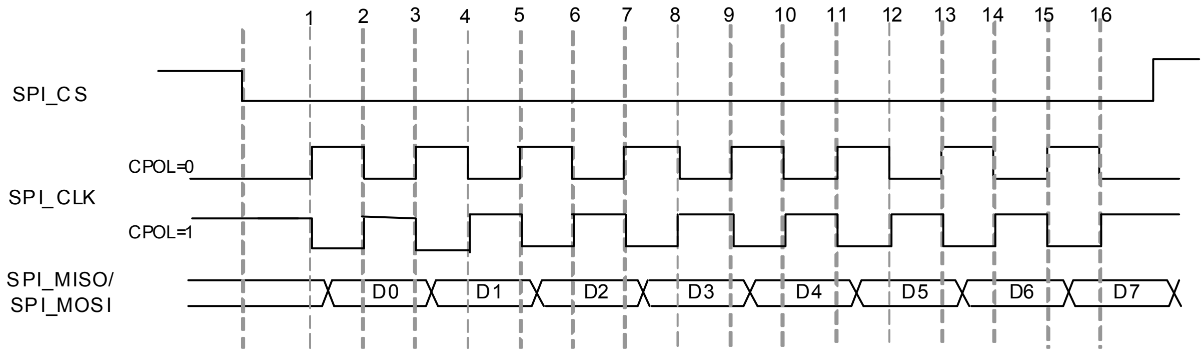


Figure 95 shows the SPI timing in CPHA=1 for both CPOL=0 and CPOL=1

Figure 95: SPI Timing for CPHA=1



22.4 Indirect Access

This section describes the SPI Indirect access operations:

- SPI Input/Output
- Output 1 Byte to SPI
- Input 1 Byte from SPI
- Output or Input 2 Bytes

22.4.1 SPI Input/Output

The serial data clock is generated out of the device core clock (TCLK) with a programmable baud generator that has a prescaler values of (2 ^ E) * D, where E<=7 and D = 4, 6, 8,…30. At power up, the prescaler value is 26. D (SPR) is configured by <SPI_SPR> field in the SPI Interface Configuration Register (N=0–1) (Table 1023 p. 1281). E (SPRR) is configured by the <SPI_SPPR0> and <SPI_SPPR_HI> field in the SPI Interface Configuration Register (N=0–1) (Table 1023 p. 1280).

The <SPI CS num> field in the SPI Control Register (N=0–1) (Table 1022 p. 1279) defines which SPI Chip-Select signal (SPI_CSn[N:0]) is asserted whenever the <CSnAct> field in the SPI Control

Register (N=0–1) (Table 1022 p. 1279) is set. The Serial data clock output SPI_SCK toggles only during transactions.

22.4.2 Output 1 Byte to SPI

To output one byte, the CPU writes to the SPI Data Out Register (N=0–1) (Table 1024 p. 1281).

This register's bits [7:0] are shifted out on SPI<n>_MOSI on the transmitting edge as defined by the <CPOL> field and the <CPHA> field in the SPI Interface Configuration Register (N=0–1) (Table 1023 p. 1280). There will be eight clock pulses output for eight data bits. The order of transmission is determined by <TxLSBF> in the same register. After all eight bits are shifted out, the <SMemRdy> field in the SPI Control Register (N=0–1) (Table 1022 p. 1279)is set. When the CPU again writes to the SPI Data Out Register (N=0–1), the Serial-Memory-Data-Ready status (<SMemRdy> field) is cleared.

22.4.3 Input 1 Byte from SPI

While SPI_SCK is toggling, the SPI<n>_MISO pin is sampled into the SPI Data In Register (N=0–1) (Table 1025 p. 1281).

Therefore, to read one byte from the external SPI device, the CPU can write a dummy data into the SPI Data Out Register (N=0–1). As dummy data is shifted out on SPI<n>_MOSI, data is shifted in from the SPI<n>_MISO pin. When the <SMemRdy> field is set, the CPU can read the SPI Data In Register (N=0–1) to retrieve the input data byte.

22.4.4 Output or Input 2 Bytes

To simplify firmware and improve the read/write throughput, the SPI interface logic can shift two bytes in and out for each access. When the <BYTE_LEN> field in the SPI Interface Configuration Register (N=0–1) (Table 1023 p. 1280)is set to 0x1, two byte I/O mode is enable. When the CPU writes to the SPI Data Out Register (N=0–1), bits [7:0] are shifted out on SPI<n>_MOSI first, followed by bits [15:8].

As for data input, SPI<n>_MISO is shifted into the SPI Data In Register (N=0–1) [7:0] bits first, followed by bits [15:8].

When both bytes are shifted out (and in), the <SMemRdy> field is set.

22.5 Direct Mode

The device SPI controller supports direct read and writes from/to external SPI devices, without software overhead of reading and writing from/to the SPI Control Register (N=0–1) (Table 1022 p. 1278), SPI Data Out Register (N=0–1) (Table 1024 p. 1281), and SPI Data In Register (N=0–1) (Table 1025 p. 1281).

This section describes the 2 SPI Direct mode operations:

- Direct Read from SPI
- Direct Write to SPI

22.5.1 Direct Read from SPI

Two read modes are supported, as defined by the <DirectRdHighSpeed> field in the SPI Interface Configuration Register (N=0–1) (Table 1023 p. 1280).

Read: Clear DirectRdHighSpeed:	There is no dummy byte insertion after the address phase.
High Speed Read: Set DirectRdHighSpeed: (also known as fast-read)	Dummy byte is inserted after the address phase.

The length of the address can be set to be between 1 and 4 bytes according to the <DirectAddrLen> field of that register.

If using SPI flash, set the <Attr> field in the CPU address decoding windows to match the SPI interface (port and CS number). Any CPU read to this address space is converted by the SPI controller to a SPI flash read transaction, composed of an address phase followed by a data phase.

The actual sequence that is driven on the SPI interface is:

1. Assert the SPI_CSn that corresponds to the <Attr> field.
2. Write command to the SPI.
   The 8 bit opcode is taken from the <DirectRdHeader>.
3. The Address is driven in 1–4 phases based on the configuration of the <DirectAddrLen> field. The order of the bytes is MSB to LSB.
4. In High Speed (HS) mode, as defined by <DirectRdHighSpeed>, add a 1B dummy write.
5. Read the data, according to the length given by the request.
6. De-assert the SPI_CSn signal.

Note

The maximum supported burst read is 32B.

22.5.2 Direct Write to SPI

The device SPI Controller supports direct writes that will be transmitted on the SPI. The action and data to be transmitted consists of the following:

1. Assert CS.
   This stage can be omitted by clearing the <Direct Wr Deassert Cs> field in the SPI Direct Write Configuration Register (N=0–1) (Table 1030 p. 1284).
2. CS hold Gap.
   All signals are steady. The length of the gap will be at least <Direct Wr Cs Hold> core clock cycles.This gap will not exist when <Direct Wr Deassert Cs> is cleared.
3. Constant Header (optional):
   This is a 1-4 byte field that is taken from the <SpiDirectHdr> field in the SPI Direct Write Header Register (N=0–1) (Table 1031 p. 1284).
   The size of the header is configured on <Direct Wr Hdr Size> field in the SPI Direct Write Configuration Register (N=0–1) (Table 1030 p. 1284). The header can be omitted by clearing <Direct Write Hdr Enable> of that same register. A possible usage of this header is setting a Write command, that will be served as a prefix to any command.
4. Address Phase.
   This is a 1–4 byte field that is taken from the address of the request.
   The size of the address is configured via the <DirectAddrLen> field in the SPI Interface Configuration Register (N=0–1) (Table 1023 p. 1280).

The entire Address phase is omitted when <Direct Wr Addr Enable> field in the SPI Direct Write Configuration Register (N=0–1) (Table 1030 p. 1284) is cleared.

5. Data Phase.
   A 1–32 byte transition of the data provided by the write request.
6. CS Preservation.
   The CS will be return to the state that it was prior to the Write command. If it was asserted before the action took place, then this stage is meaningless. If it was not asserted, then at this stage the CS will be de-asserted.
   This stage will be omitted if <Direct Wr Deassert Cs> is cleared.

Note The maximum supported burst write is 32B.

22.5.3 SPI Timing

Separated Attributes per CS mode

The <SPI DecodeMode> field is 0x0.
While using this mode the address resolution and assignment to the appropriate CS is accomplished, according to the attribute settings defined in the corresponding address windows of the originating unit. To address two different CSs, the originating unit should have two different address decoding windows each referenced to the address space covered by the specific CS. For details of the decoding scheme, refer to:

- Table 6, CPU Interface Mbus Decoding Units IDs and Attributes, on page 57
- Table 8, Units IDs and Attributes, on page 63

Single Window mode

The <SPI DecodeMode> field is 0x1.
In this mode, software defines a continuous address space that covers the entire physical space of the SPI devices. For example, if there are two SPI parts running over CS0 and CS1, each 128 MB in size, software will define the address space for these chip selects as address A to address A+256 MB. The software of the originating unit (for example, the CPU or PCIe) assigns a single address decoding window that covers address space A to A+256 MB (256 MB in size) without differentiating between the two SPI domains. This window can cover up to 8 CS.
Any transaction hits this SPI window (regardless of whether it originated in CS0 or CS1). Once it hits this window, it is routed internally to the SPI controller.
In the SPI controller, it is further decoded, and directed to the appropriate CS. This sub-decoding is accomplished according to 3 additional bits of the original address. For maximum flexibility, the location of these bits is configurable via the <SPI Window0IndexLSB> field in the SPI CS Address Decode Register (N=0–1) (Table 1033 p. 1285).

Copyright © 2014 Marvell

Double Window mode The <SPI DecodeMode> field is 0x2.
This mode uses the same method as the Single Window mode, but provides further address decoding flexibility by providing an additional decode window. This means that software now has the flexibility of assigning 2 different continuous SPI address spaces that use 2 different address decoding windows in the originating unit. Each of the 2 windows covers up to 4 CSs (of the total 8). The sub decoding, in this case, is accomplished via the <SPI Window0IndexLSB> field.

The SPI controller has a set of parameters to define guaranteed timing gaps, as listed below.

SPI CS Setup To ensure minimal gap between CS assertion and the first clock edge, set the <SPI_TCS_SETUP> field in the SPI Timing Parameters 1 Register (N=0–1) (Table 1028 p. 1282) to the required delay (in TCLK cycles).

SPI CS Hold To ensure a minimal gap between the last clock edge and the CS de-assertion, set the <SPI_TCS_HOLD> field in the SPI Timing Parameters 1 Register (N=0–1) (Table 1028 p. 1282) to the required delay (in TCLK cycles).

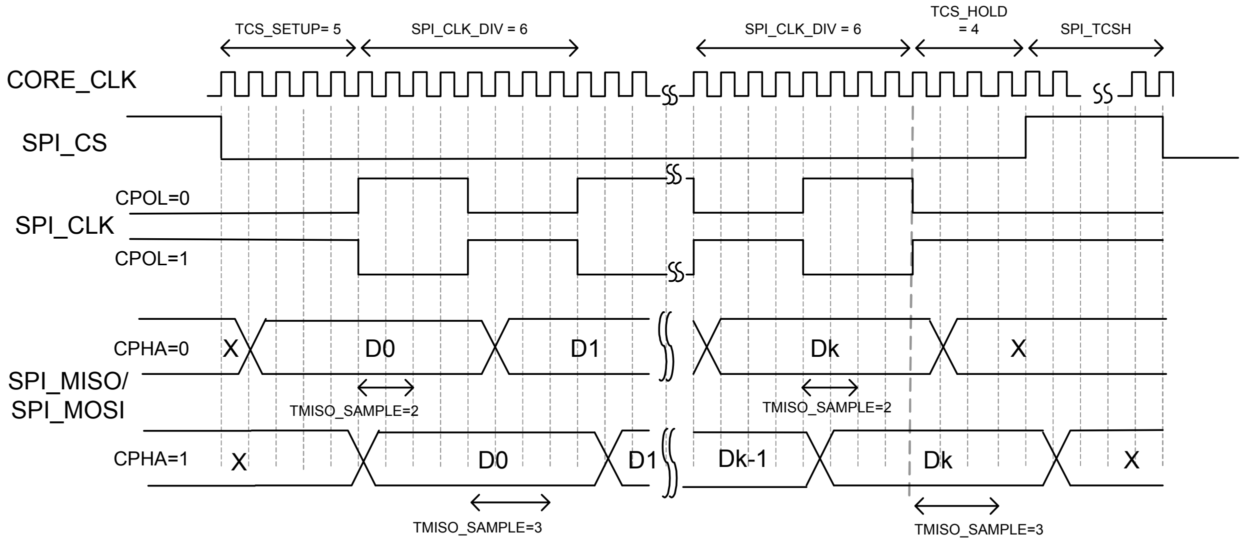
SPI CS De-select To ensure a minimal gap between the CS de-assertion and the next assertion, set the <SPI_TCSH_9_6> field and the <SP_TCSH> field in the SPI Timing Parameters 1 Register (N=0–1) (Table 1028 p. 1283) to the required delay (in TCLK cycles).

Typically, SPI<n>_MISO is sampled by the master in the sampling edge as defined by CPOL and CPHA modes. The internal sampling point of MISO can be delayed to relax the timing constraint for a high speed bus. To delay the sampling, configure the <SPI_TMISO_SAMPLE> field in the SPI Timing Parameters 1 Register (N=0–1) (Table 1028 p. 1283) to the required delay (in TCLK cycles), subject to the following constraints:

- The delay should be smaller then 1/2 SPI_CLK cycle.
- When CPHA=1, the delay should be smaller then SPI_TCS_HOLD.

Refer to Figure 96 for an example.

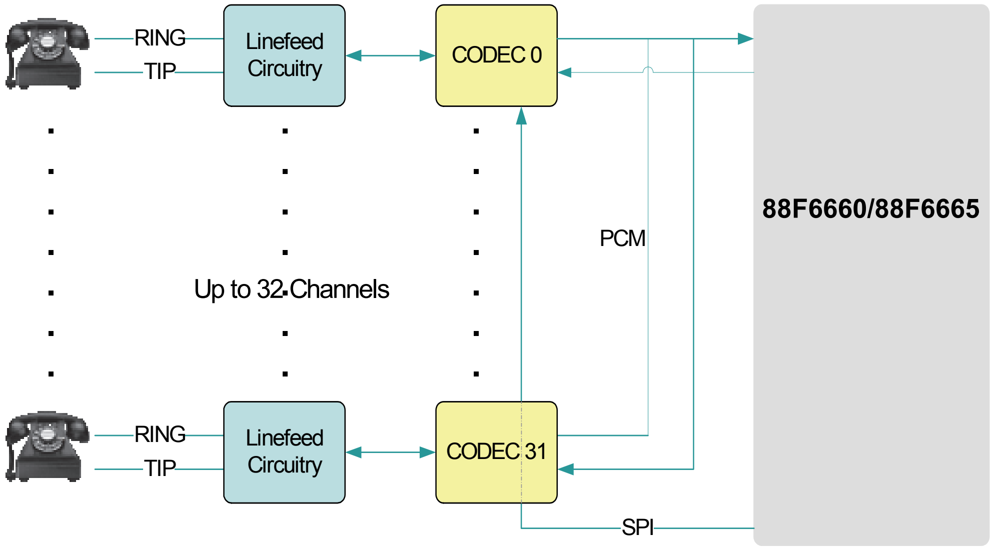
Figure 96: SPI Timing Example



23 Time-Division Multiplexing (TDM) Controller

This section describes the Multi-Channel Time-Division Multiplexing (TDMMC) controller (aka TDM controller). The TDM controller is used to interface with external Subscriber Line Interface Circuit (SLIC), Subscriber Line Audio-processing Circuit (SLAC), and codec devices such as VoIP applications (see Figure 97).

Figure 97: SLIC/Codec Connection Example



The TDM controller registers are located in Appendix A.13, Time Division Multiplexing (TDM) Controller Registers, on page 1288.

23.1 Features

The TDM controller supports the following features:

- Up to 32 independent channels.
- Dedicated DMA engine for each Rx/Tx channel (overall 64 DMA engines), with flexible buffer allocation and size per channel.
- Per-channel linked list descriptor chain for automatic data transfers from interface and memory.
- Fully flexible and configurable slot allocation on up to 128 full-duplex slots that can be assigned to the 32 channels.
- Each TDM channel can be used in High-Level Data Link Control (HDLC) Bit-oriented Protocol mode or in Transparent Protocol mode.
- When configured to HDLC mode, the channel supports:
  - Flag generation and stripping

Copyright © 2014 Marvell

  - Bit stuffing and stripping
  - Address recognition (8- or 16-bit addresses) and filtering
  - CRC generation and checking
  - Line condition monitoring
- Transparent protocol can be used for VoIP with connectivity to up to 32 SLICs / SLACs or an E1/T1 framer.
- Configurable to various bit clock rates (256 kHz to 8.192 MHz with increments in power of 2).
- Option to work with an internally generated Frame sync and PCM clock or with an external reference input clock and/or Frame sync.
- Configurable MSB / LSB bit order for receive and transmit.
- Compound (A-law/U-law) or linear voice samples.
- Narrow-band and wide-band voice channels.
- Supports various flavors of PCM (for example, short and long frame sync, inverted frame sync, posedge / negedge PCM data drive / sample).

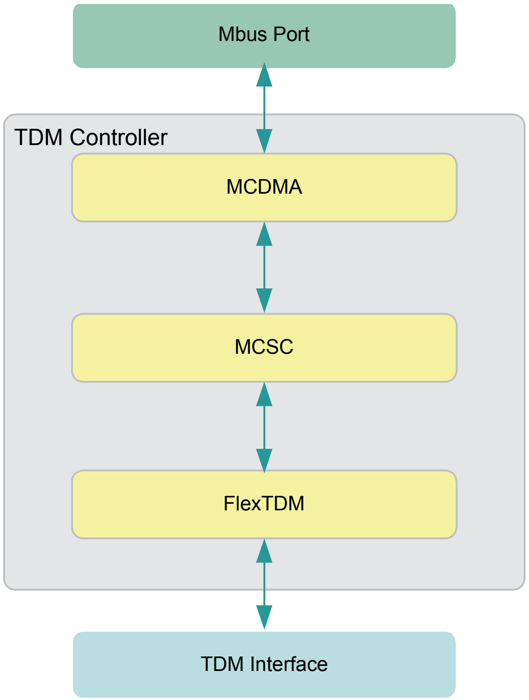
23.2 Functional Description

The TDM controller is composed of three major components:

- FlexTDM Controller (FTDM)
- Multi-Channel Serial Controller (MCSC)
- Multi-Channel Direct Memory Access (MCDMA)

Figure 98 illustrates the hierarchy of these blocks within the TDM controller.

Figure 98: TDM Controller Components



The following sections describe the architecture and configuration options of these components.

23.3 FlexTDM Controller (FTDM)

The FlexTDM controller is capable of interfacing a PCM highway and E1/ T1 lines, as well as other proprietary time slot assigned buses.

The time slot assignment is configured by programming an internal Dual Port RAM (DPRAM). This DPRAM supports static and dynamic configurations. The DPRAM-based design provides the flexibility to program the FlexTDM to any of the common time slot buses (that is PCM) or to a proprietary bus.

Note

The TDM pins are multiplexed on the Device bus and MPP pins. For further information, see the Pin Multiplexing section in the device hardware specifications.

The Frame Sync (FS) is generated (or sampled) indicating the start of the time slot. A time slot corresponds to one sample of 1 byte. The TDM controller provides the flexibility to assign any time slot to any MCSC channel to support multi-slot configurations. For example, in some cases a wide-band codec may be used for enhanced voice quality in VoIP networks, in which two to four PCM samples are transmitted and received in one frame sync (125 µs) from TDM to codec and vice-versa. The voice quality is enhanced because the effective sampling rate becomes 16–64 kHz (instead of the 8 kHz rate in the standard Narrowband mode) as more than one slot is sampled. The TDM DPRAM, provides full flexibility to tie any of the time slots within a frame to specific MCSC channel, which ensures that all the data that is collected from the various time slots reaches the allocated memory buffer of the channel in the right order. More details about the programming options of the DPRAM can be found in Section 23.3.2.1 “FlexTDM DPRAM” on page 389.

Transmit: In transmit operation, the data is driven out to the SLIC/SLAC/codec on the DTX line. The TDM can drive data on the positive edge or negative edge of PCLK. The order that the bits are driven on the PCM bus may be either MSB or LSB first via a programmable configuration option in the MCSC channel configuration registers. See Section 23.4.2.2 “MCSC Channel-x Transmit Configuration Register (MTCRx)” on page 392 for more details.

Receive: In receive operation, the SLIC/SLAC/codec can drive data on the DRX line, and the TDM interface can be programmed to capture the data on either the negative or positive edge of PCLK. The order that the bits are driven on the PCM bus may be either MSB or LSB first via a programmable configuration option in the MCSC channel configuration registers. See Section 23.4.2.1 “MCSC Channel-x Receive Configuration Register (MRCRx)” on page 392 for more details.

The TDM can be the master of FS and PCLK (it drives both of these) or it can be the slave (receiving these two inputs from an external master). In addition, it can generate and drive out the FS signal from a PCLK input. The frequency of the internally generated PCLK signal can be controlled via the TDM Clock Divider Control Register (Table 1135 p. 1351).

The FS can be short, long, inverted, and driven on the positive or negative edge of PCLK, depending upon programmed value via TDM Clock and Sync Control Register (Table 1134 p. 1349).

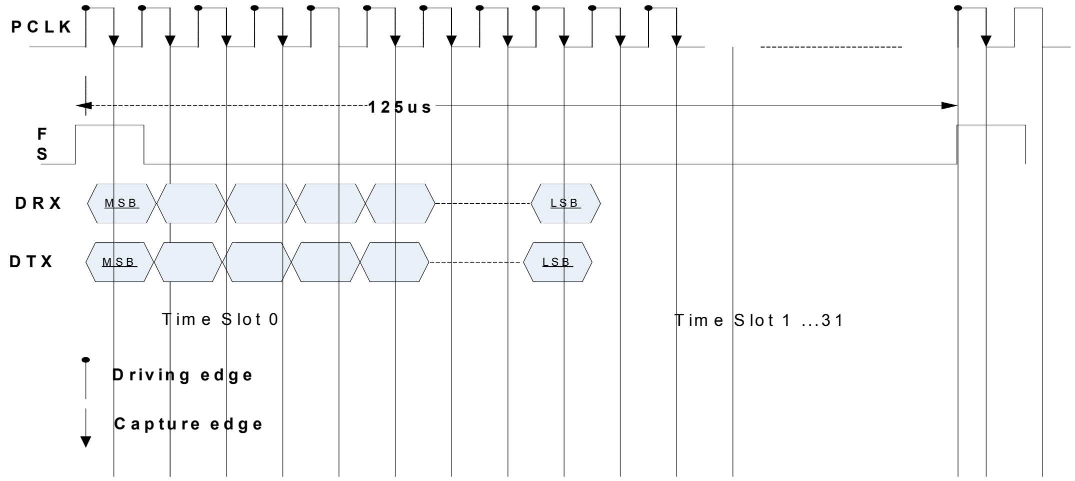
23.3.1 PCM Protocol Specification

The TDM interface defines time slots for Pulse Code Modulation (PCM) data transmit and receive. The start of the time slot is indicated by a Frame Sync (FS). The time slot consists of 8 PCM clocks (PCLK). The time interval of FS generation is programmable via the DPRAMs. A typical interval for FS assertion is at an interval of 125 µs that corresponds to an 8-kHz frequency. In each FS, one sample of PCM data is received and transmitted per slot.

The FS indicates the start of the Time Slot 0, as shown in the Figure 99. In this example:

- DRX—The external device drives DRX line on positive edge of PCLK, and TDM captures on negative edge of PCLK.
- DTX—The TDM drives data on the positive or negative edge of PCLK on the DTX line.

Figure 99: TDM Operation - Time Slot 0



23.3.2 FlexTDM Architecture

The FlexTDM controller consists of a transmit and a receive section. The architecture is based on two dual-port RAM (DPRAM) arrays. One RAM array is for receiving and one is for transmitting. The frame structure of a time slot assigned bus is configured by programming this DPRAM.

The MCSC that is connected to the FlexTDM gets its receive and transmit signals from the FlexTDM. The actual bit rate of the serial communications controller is defined by the FlexTDM programming and the time slots that are assigned to it.

The following section describes the DPRAM functionality and configuration options.

23.3.2.1 FlexTDM DPRAM

Each of the FlexTDM DPRAM is a 256x27 dual port RAM array that controls the FlexTDM behavior. The DPRAM can be configured dynamically during FlexTDM operation.

For the various configuration options and field assignments of the FlexTDM DPRAM refer to for FlexTDM Transmit Dual Port RAM (TDPR)<n> Register (n=0–255) (Table 1129 p. 1344) for the transmit settings and to FlexTDM Receive Dual Port RAM (RDPR)<n> Register (n=0–255) (Table 1130 p. 1345) for the receive side.

Note

- When the FlexTDM is disabled, the CPU can access the DPRAM for both reads and writes. When the FlexTDM is enabled, the FlexTDM DPRAM is write only.
- The initial value of the DPRAM is undefined. After reset, an internal logic initiates automatically a value of 0 to all fields. The indication for the completion of this process is registered in <RID> field of MCDMA Global Control Register. SW must poll this bit before trying to change the initial value of the DPRAM.

23.3.2.2 FlexTDM Programming Modes

In addition to the DPRAM configuration, software needs to enable the FlexTDM operation via <TEN> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1347). After enabling the FlexTDM, there are two ways the user can dynamically program the DPRAM:

Single Array Mode: The FlexTDM read pointer ALWAYS returns to entry 0 after receiving a SYNC or reading the last entry (the entry with L bit set). The entire range of 256 entries is available for programming a TDM frame. The user can dynamically make changes in the DPRAM, but caution must be taken not to write to the same address from which the FlexTDM is reading. This can be accomplished by checking the read pointer before accessing the DPRAM array. The read pointer status can be polled though registers:

- FlexTDM0 Transmit Read Pointer Register (Table 1131 p. 1346) for TX-DPRAM
- FlexTDM0 Receive Read Pointer Register (Table 1132 p. 1347) for RX-DPRAM

Split Array Mode: The TDM frame is limited to 128 DPRAM entries. The user programs the FlexTDM frame in entries 0–127 and enables the FlexTDM. When a programming change is required, the user first programs the new frame in entries 128–255 and then sets the RR2HALFand TR2HALF (read R2HALF as Return To Half) bits in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1347). When the next SYNC or last entry occurs, the FlexTDM starts processing the frame as programmed in entries 128–255. The user can then re-program entries 0–127. This process of switching between one half of the DPRAM and the other half simplifies on-the-fly DPRAM changes.

Note

- If a frame structure is changed (for example, a change in frame length) while the FlexTDM is enabled, the FlexTDM will lose synchronization.
- For the recommended programming of FlexTDM for Linear mode, see Section 23.7.4.1, Configuring FlexTDM for Linear Mode, on page 411.

23.3.2.3 FlexTDM Synchronization

After enable, the FlexTDM needs to receive a single FSYNC to achieve synchronization. Until synchronization is attained, the FlexTDM transmit and receive paths are disabled.

After gaining synchronization, the FlexTDM always predicts when the next sync is expected. If the sync signal is not asserted when expected, the FlexTDM executes a synchronization lost procedure and a maskable interrupt is generated.

Note

FlexTDM synchronization is relevant only for FSYNC use as an input.

23.4 Multi-Channel Serial Controller (MCSC)

The Multi-Channel Serial Controller (MCSC) can process data from up to 32 channels. These channels can operate in either of the following modes:

- HDLC Bit-oriented Protocol mode
- Transparent Protocol mode

The FlexTDM can be programmed so that a channel operates at a speed of 64 Kbps. It can also operate at N*64Kbps or at N*8Kbps.

23.4.1 HDLC Mode

In HDLC mode, an MCSC channel performs the following protocol functions:

- Flag generation and stripping
- Bit stuffing and stripping
- Address recognition (8- or 16-bit addresses)
- CRC generation and checking
- Line condition monitoring

Figure 100:Typical HDLC Frame

FLAG	ADDRESS	CONTROL	INFORMATION	CRC	FLAG
8 Bits	8/16/8N Bits	8/16 Bits	8N Bits (Optional)	16/32 Bits	8 Bits

23.4.1.1 MCSC Channel-x Receive Configuration Register (MRCRx)

Each channel has its own RX configuration register. This register fields have different meaning when working in HDLC mode or in transparent mode. User should configure the corresponding channel register according to the mode the channel works in (HDLC or transparent). See MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) for more details about the various configuration options of each mode.

23.4.1.2 Maximum and Short Frame Length Register

Each frame received is checked if it is between the maximum and short frame length boundaries. If the packet does not comply to this restriction, the descriptor is closed with either MFLE or SFE error bits set.

The MCSC Global Configuration Register contains values for both parameters. These parameters are system-specific and not channel-specific.

23.4.1.3 MCSC Channel-x Transmit Configuration Register (MTCRx)

Each channel has its own RX configuration register. This register fields have different meaning when working in HDLC mode or in transparent mode. User should configure the corresponding channel register according to the mode the channel works in (HDLC or transparent). See MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31) for more details about the various configuration options of each mode.

23.4.1.4 HDLC Address Filtering Methods

There are two methods for HDLC-address filtering.

Null/Broadcast Address Filtering method The TDMMC accepts all addresses except for Null (0x0) or Broadcast (0xFFFF) address, in case these are configured to be filtered.

Four 16-bit Address Register method This method is implemented by a set of four 16-bit address registers and a mask register.
The TDMMC compares the received address to the group of addresses defined by the masked address registers and passes the HDLC-received-frame when there is an address match.

Recommended Usage

Null/Broadcast Address Filtering method By default, TDMMC uses the Null/Broadcast Address Filtering method, where *all* HDLC frames are accepted except for Null/Broadcast Address, if they are not enabled. Program the <NrorBrcst> field to enable Broadcast reception, and for null reception, enable the <NullEn> field in the MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Table 1049 p. 1301).

Four 16-bit Address Register method To activate the Four 16-bit Address Registers method:

1. Configure the required addresses in the:
   - <Addr2_MSByte>/<Addr1_MSByte> field in the HDLC Mode Address1 and Address2 Register (Table 1083 p. 1325)
   - <Addr4_MSByte>/<Addr3_MSByte> field in the HDLC Mode Address3 and Address4 Register (Table 1084 p. 1326)
2. Configure the relevant bits to be compared in the <BitCompareEn> field in the HDLC Mode Address Filtering Register (Table 1085 p. 1326) as indicated in the register description.
3. Set the <HdlcAddrMatchMode> field in the MCSC Global Configuration Extension Register (Table 1078 p. 1323) register to 1.
4. In case Null and Broadcast addresses need to be filtered, use <NrorBrcst> field and the <NullEn> field in the MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Table 1049 p. 1301) to set this option.

23.4.2 Transparent Protocol

In transparent mode, the MCSC does not perform any protocol-dependent data processing, but rather transfers the data bytes as they appear on the channel, to the external memory, for the CPU to process. A transparent channel is synchronous and if it is not serviced on time, underrun and overrun errors can occur.

23.4.2.1 MCSC Channel-x Receive Configuration Register (MRCRx)

Each channel has its own RX configuration register. This register fields have different meaning when working in HDLC mode or in transparent mode. User should configure the corresponding channel register according to the mode the channel works in (HDLC or transparent). See MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) for more details about the various configuration options of each mode.

23.4.2.2 MCSC Channel-x Transmit Configuration Register (MTCRx)

Each channel has its own RX configuration register. This register fields have different meaning when working in HDLC mode or in transparent mode. User should configure the corresponding channel register according to the mode the channel works in (HDLC or transparent). See MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31) for more details about the various configuration options of each mode.

23.4.3 MCSC Command Execution Status Register

For each receive channel the CPU can issue one of two commands to the MCSC if it is an HDLC channel, and one of three commands if it is a Transparent channel. For HDLC and Transparent channels the CPU can issue an Enter Hunt command or an Abort Receive command. For Transparent channels the CPU can issue an additional Close Rx Descriptor command.

The CPU can determine whether or not the command was executed, by either programming the MCSC to queue an interrupt entry to the interrupt queue set an interrupt when the command is executed (see Section 25.5 “Interrupt Queue Controller” on page 512) or by polling the channel’s Command Execution Status Register. Each channel has its own register that contains three bits indicating whether or not the MCSC has already executed the command issued by the CPU. The contents of this register are Read Only for the CPU. Only the MCSC can change the value of this register. See Table 1048, MCSC Channel-x Command Execution Status Register Channel <n> (n=0–31) for setting options of this register

23.5 Multi-Channel Direct Memory Access (MCDMA)

The MCDMA is dedicated to moving data between the serial communications channels and memory buffers. The MCDMA can serve up to 32 channels. Each MCDMA channel consists of 2 DMA engines—1 for receiving and 1 for transmitting.

The MCDMA uses only one interface to the device interconnect fabric (Mbus). The MCDMA priority in the Mbus arbitration accessing memory is programmable via the memory controller registers.

Each MCDMA channel has an internal RxFIFO for data buffering. The buffer accumulates data before transferring it to external memory. In addition, each channel has a TxFIFO for pre-fetching data from memory, before transmission. Data is read from the external memory and transferred to the TxFIFO before transmission begins. Size of both RxFIFO and TxFIFO can be dynamically configured using a linked list mechanism. The Dynamic FIFO Size is intended to meet the channel’s buffering requirements derived from its speed.

The MCDMA uses an 4 KB SRAM for implementing a per channel RxFIFO. The SRAM is shared among all channels by using the linked list mechanism. Before enabling a channel, the user must assign part of the SRAM to the channel, for its specific RxFIFO use. The methodology of building the channel’s RxFIFO and restrictions are detailed in Section 23.5.4 “Receive and Transmit FIFO Management Linked List” on page 400. Another 4-KB SRAM is also used for implementing a per channel TxFIFO. The same principals applied to the RxFIFO, also apply to the TxFIFO.

For receive operations, the MCSC moves data into the dedicated FIFO of the corresponding MCDMA channel. By using descriptors set up by the user, the MCDMA moves data into the memory buffers. For transmit operations, the MCDMA uses descriptors set up by the user to move data out of the memory buffers and into the channel’s dedicated FIFO. The MCSC then moves the data down to the serial communications link.

The MCDMA channel descriptors are connected via a chained data structure per channel. These work without CPU interference after appropriate initialization. The MCDMA channels can be programed to generate interrupts on buffer or frame boundaries.

When enabled, the receive MCDMA engines run freely and expect to find a valid descriptor when required. When a receive MCDMA channel accesses an invalid descriptor, the receive MCDMA process halts with a resource error status indication.

When enabled, the transmit MCDMAs runs freely until the end of the descriptor chain is reached. If a transmit MCDMA channel reaches the end of the chain and encounters an invalid descriptor where the preceding descriptor is not marked as an end of frame descriptor, the transmit MCDMA process halts with a resource error status indication.

The MCDMA logic connected to the Mbus arbitrates access to the descriptors and buffers between the various DMA engines. A weighted round-robin scheme is used for arbitration between high priority and low priority channels within the MCDMA channels. Arbitration between MCDMA and other Mbus units is done at the Mbus level, which supports a programmable, weighted pizza arbiter.

The MCDMA buffers and descriptors reside either in SDRAM space or in PCIe space. Address decoding is automatic and does not require user intervention after appropriate initialization.

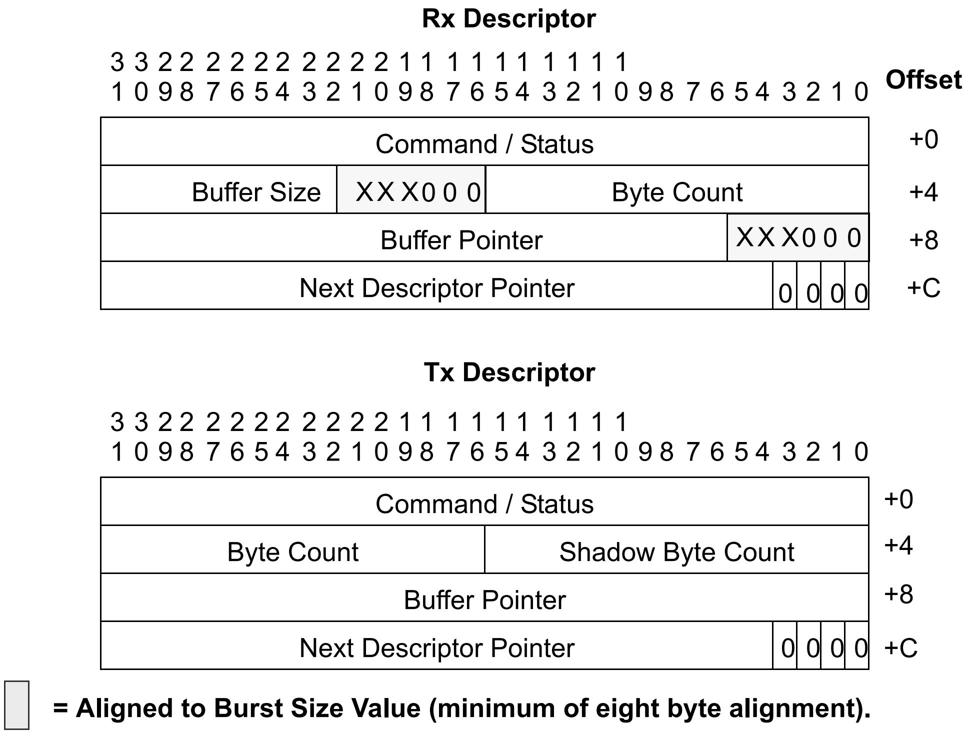
Each MCDMA has a Global Control Register that holds a number of global parameters and commands. The contents of the MGCR are listed in Table 1040, MCDMA Global Control Register MGCR.

23.5.1 MCDMA Descriptors

MCDMA data is transferred via chained linked descriptors. The following rules apply to the device's MCDMA descriptors:

- Descriptor length is 4LW and must be 4LW aligned (i.e. Descriptor_Address[3:0]=0000).
- Descriptors may reside anywhere in the memory address space except for the NULL address used to indicate the end of the descriptor chain.
- Rx buffers size must be a multiple of the channel's Rx Burst Size parameter (RBSZ). Minimum size for Rx buffers is 8 bytes.
- Rx buffers address must be aligned with the channel's Rx Burst Size parameter (RBSZ). For instance, if the Rx Burst Size has a value of 3 (8 double words or 64 bytes bursts) then all of that channel's buffers must have a starting address aligned to 64 bytes.
- Tx buffers associated with Tx descriptors can start in any byte location and are not limited to buffer size.
- MCDMA Rx and Tx buffers are limited to 64 KB.

Figure 101:MCDMA Descriptor Format



The following sections provides information about the various descriptor fields.

23.5.1.1 MCDMA Descriptor—Command/Status Word

Table 103 describes the command/status fields and their usage. The fields may have different meaning if representing HDLC mode descriptor or Transparent mode descriptor as specified in:

- Section 23.5.1.3, MCDMA Channel-x Command/Status Field for Transparent Mode
- Section 23.5.1.2, MCDMA Channel-x Command / Status Field for HDLC Mode.

Table 103: MCDMA Descriptor—Command/Status Word

Bits	Field	Description
NOTE: The Command/Status word, contains command bits that instruct the MCDMA how to process buffer and status bits that have been updated by the MCDMA upon closing a descriptor. The CPU uses the status bits to evaluate the buffer status. Except for bits 31, 30, 23,21, 17 and 16, the definition of the bits vary depending on which mode is being used (see the MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31)).		
15:0	Status	Status Determined by the mode selected.
16	L	Last bit Indicates the last buffer of a frame.
17	F	First bit Indicates the first buffer of a frame.
20:18	Reserved	Reserved
21	EOPI	End Of Packet Interrupt **NOTE:** Only relevant to Tx Descriptors. When the last bit, from a packet with EOPI bit set, leaves the serial port towards the TDM interface, the MCDMA queues a maskable interrupt. If the packet is divided into two or more buffers, the EOPI must be set only in the last packet's descriptor.
22	Tx_HDLC_GC	Generate CRC (GC) **NOTE:** Only relevant for Tx in HDLC mode.
23	EI	Enable Interrupt The MCDMA generates a maskable interrupt when closing descriptor with EI bit set. **NOTE:** When the RIFB bit is set in the MCDMA ChannelX configuration register, a Rx interrupt is generated only if this is the last descriptor associated with a received frame. In this case, EI bit setting is masked for intermediate descriptors.
29:24	Reserved	Reserved
30	AM	Auto Mode When set, the MCDMA does not clear the Owner bit of the descriptor at the end of buffer processing.
31	O	Owner Bit • When set to 1, the buffer can be processed by the MCDMA. • When set to 0, the buffer can be processed by the CPU. **NOTE:** When a descriptor with owner bit set to 0 is fetched, the MCDMA process is halted.

23.5.1.2 MCDMA Channel-x Command / Status Field for HDLC Mode

When an MCSC channel is in HDLC mode, the Command/Status field in the corresponding MCDMA Channel-x descriptor is formatted as described in Table 104.

Table 104: MCDMA Channel-x Command/Status Field for HDLC Mode

Bit	Rx - Function	Tx - Function
0	CE - CRC Error	Reserved
1	Reserved	Reserved
2	Reserved	Reserved
3	NO - Non Octet Frame	Reserved
4	ABR - Abort Sequence	Reserved
5	ORD - Data Overrun in MCDMA	Reserved
6	ORC - Data Overrun in MCSC	UR - Data Underrun
7	MFLE - Max Frame Length Error	Reserved
8	SFE - Short Frame Error	Reserved
9	Reserved	Reserved
13:10	Reserved	Reserved
14	Reserved	Reserved
15	ES - Error Summary ES = CE || NO || ABR || ORC|| ORD || MFLE || SFE[1]	Error Summary ES = UR
16	L - last	L - Last
17	F - First	F - First
20:18	Reserved	Reserved
21	Reserved	EOPI - End Of Packet Interrupt (see bit 21 in Table 103).
22	Reserved	GC - Generate CRC
23	EI - Enable Interrupt	EI - Enable Interrupt
29:24	Reserved	Reserved
30	AM - Auto Mode	AM - Auto Mode
31	O -Owner	O - Owner

[1] "||" means logical OR.

23.5.1.3 MCDMA Channel-x Command/Status Field for Transparent Mode

Table 105 describes the format of the Command/Status field in the corresponding MCDMA Channel-x descriptor, when the channel is in Transparent mode.

Table 105: MCDMA Channel-x Command/Status Field for Transparent Mode

Bit	Rx - Function	Tx - Function
0	Reserved	Reserved
1	Reserved	Reserved
2	Reserved	Reserved
3	Reserved	Reserved
4	Reserved	Reserved
5	ORC - Data Overrun in MCSC	Reserved
6	ORD - Data Overrun in MCDMA	UR - Data Underrun
14:7	Reserved	Reserved
15	ES - Error Summary ES = ORD || ORC	ES - Error Summary ES = UR
16	L - Last	L - Last
17	F - First	F - First
20:18	Reserved	Reserved
21	Reserved	EOPI - End Of Packet Interrupt (see bit 21 in Table 103).
22	Reserved	Reserved
23	EI - Enable Interrupt	EI - Enable Interrupt
29:24	Reserved	Reserved
30	AM - Auto Mode	AM - Auto Mode
31	O - Owner	O - Owner

23.5.1.4 MCDMA Descriptor - Buffer Size, Byte Count, Buffer Pointer and Next Descriptor Pointer

Table 106 through Table 110 outline the command/status fields, the buffer pointer field and the next descriptor pointer field and their usage.

Table 106: MCDMA Descriptor—Buffer Size, Byte Count (Rx Descriptor)

Bits	Field	Description
15:0	Byte Count	The number of bytes actually written by the MCDMA into the buffer. This number is never greater than Buffer Size. The CPU must initialize the Byte Count field with 0x0000.
31:16	Buffer Size	When the buffer byte counter of a MCDMA receive channel reaches the buffer size value, the MCDMA closes the buffer descriptor and moves to the next buffer. The Buffer Size must be a multiple of the channel’s RBSZ parameter, i.e the number of bytes in the burst.

Table 107: MCDMA Descriptor—Byte Count, Shadow Byte Count (Tx Descriptor)

Bits	Field	Description
15:0	Shadow Byte Count	Software must initialize this field with a value identical to the Byte Count field. The MCDMA subtracts the number of bytes actually transmitted from this parameter. Software generally writes 0 in this field when closing a descriptor. However, when the transmit MCDMA Channel halts due to a transmit error, this parameter can be used to determine the number of bytes fetched into the MCDMA buffers. Setting both the Byte Count and Shadow Byte Count to 0 causes the MCDMA to close the descriptor and move to the next descriptor, when both or neither of the F and L bits are set. Setting Byte Count and Buffer Size to 0 in transmit descriptors with one of the F or L bits set will lead to unpredictable behavior.
31:16	Byte Count	Byte count is the number of bytes to be transmitted.

Table 108: MCDMA Descriptor—Buffer Pointer (Rx Descriptor)

Bits	Field	Description
31:0	Buffer Pointer	32-bit pointer to the beginning of the Rx data buffer associated with the descriptor. The buffer resides in memory or PCIe address space. Rx buffer pointers must be aligned to the channel’s Rx Burst Size parameter. When a channel is configured to have a burst size of 32 bytes (4 64-bit words), all of the channel’s buffers must have an address that is a multiple of 32 (5 LSB bits are zero).

Table 109: MCDMA Descriptor—Buffer Pointer (Tx Descriptor)

Bits	Field	Description
31:0	Buffer Pointer	32-bit pointer to the beginning of the Tx data buffer associated with the descriptor. The buffer can reside anywhere in memory or PCIe address space.

Table 110: MCDMA Descriptor—Next Descriptor Pointer

Bits	Field	Description
3:0	Reserved	These bits must be cleared to 0. **NOTE:** The descriptor pointer must be 16-byte aligned.
31:4	Next Descriptor Pointer	32-bit Next Descriptor pointer to the beginning of the next descriptor in the chain. A descriptor can reside anywhere in memory or PCIe space. DMA operation is stopped when a NULL value in the Next Descriptor Pointer is encountered.

23.5.2 MCDMA Channel-x Control Registers

23.5.2.1 Receive MCDMA Channel-x Control Register (RMCCx)

Each Receive MCDMA Channel has a dedicated control register (RMCCx). The RMCC must be initialized in order to enable the DMA channel operation. Refer to MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31) for description of the bit structure of this register and its functionality.

23.5.2.2 Transmit MCDMA Channel-x Control Register (TMCCx)

Each Transmit MCDMA Channel has a dedicated control register (TMCCx). The TMCC must be initialized in order to enable the DMA channel operation. Refer to MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31) for description of the bit structure of this register and its functionality.

23.5.3 MCDMA Descriptor Pointer Registers

Each channel in the MCDMA has two Current Descriptor Pointer Registers; MCRDP for Receive and MCTDP for Transmit. Each Current Descriptor Pointer Register points to the address of the current Descriptor that is being processed by the MCDMA. The contents of these registers are used to fetch and update (close) Receive and Transmit Descriptors to and from external memory.

Before any channel is activated or re-activated, the CPU must initialize the appropriate Descriptor Pointer Register with the address of the first descriptor in the chain. During the normal process of reception or transmission, the CPU must not write to these registers. It can only read its contents if necessary; e.g. for monitoring the Rx/Tx progress.

23.5.3.1 MCDMA Current Receive Descriptor Pointer (MCRDP)

MCRDPx points to the current receive descriptor in memory. The CPU must write this register with the first descriptor address before enabling the MCDMA receive channel. See MCDMA Current Receive Descriptor Pointer (MCRDPx) Channel<n> Register (n=0–31).

Copyright © 2014 Marvell

23.5.3.2 MCDMA Current Transmit Descriptor Pointer (MCTDP)

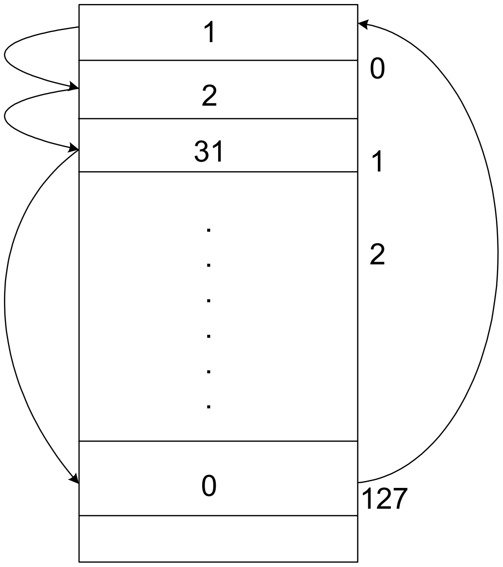
MCTDPx points to the current transmit descriptor in memory. The CPU must write this register with the first descriptor address before enabling the MCDMA transmit channel. See MCDMA Current Transmit Descriptor Pointer (MCTDPx) channel<n> Register (n=0–31).

23.5.4 Receive and Transmit FIFO Management Linked List

The FIFO Management Linked List is a data base used to allocate different Rx/Tx FIFO size on a per channel basis. Since channels may have different throughputs, it is important to provide different buffering capabilities. The higher throughput the channel is, the bigger its FIFO buffer size. The Circular Linked List method enables the management software to maintain and configure such a system when channels are being setup and later on closed. These circular linked lists (one for Receive and one for Transmit) are kept in two of the MCDMA internal SRAMs. Each SRAMs has 128 entries that can be assigned to any of the 32 channels. Each entry in the linked list SRAMs represents the index of the next basic buffering unit in the FIFO, belonging to that same channel. To configure the circular list use the MCDMA Transmit-FIFO Management Linked List<n> Register (n=0–127)

Basic buffering unit in the Rx and Tx FIFO is four 64-bit words.

Figure 102:Circular Linked List Example of Allocated Basic Buffering Units for Channel 0



The Linked List entries are 8 bits wide and are mapped to 32-bit aligned addresses. When accessing the Linked List, the CPU must write the Next Block Pointer in the 8 LSB bits. The CPU must initialize a channel's linked list before enabling the channel. Once the channel is enabled, the CPU **must not** access the channel's entries in the linked list memory for read or write.

Note

- Each channel's linked list must be cyclic. After reset the first 32 entries are initialized by the device so that they point themselves, creating 32 cyclic linked lists. Each list contains one 32-bytes block. The rest of the entries are not initialized and must be programmed before used.
- The MCDMA will not function unless each Channel's FIFO Linked List contains the FIFO Buffer which has the same number as the channel index. Figure 102 describes a list that defines the FIFO buffers belonging to Channel-0's Rx FIFO and illustrates how the buffer number zero is part of the list.
- For Rx channels, the CPU can assign the whole FIFO for one channel usage. For Tx channels, the CPU can assign the whole FIFO minus four 64-bit words for one channel usage.

23.5.5 Rx Service Request Queues

When a channel accumulates enough data in its RxFIFO, a transfer to the external memory, as specified by the channel's current descriptor fields, is required. The channel transfers the data when sufficient data to perform a burst is available, or when at the end of frame. The Channel's DMA enqueues a service request to one of two service queues; High Priority Queue and Low Priority Queue. The <RCP> field in the MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31) (Table 1037 p. 1292) (RMCC) determines which of the two queues the request be inserted in. Alternatively, the queues are serviced in a Weighted Round Robin fashion as specified in the <RSQW> field in the Rx Service Queue Arbiter Weight Register (Table 1041 p. 1294). This field determines the number of High Priority Entries to be serviced before one Low Priority Entry.
A value of 0x1F indicates unlimited number of high priority entries served before one low priority entry is served. A value of 0x0 indicates unlimited number of low priority entries served before one high priority entry is served.

23.5.6 Tx Service Request Queues

Similar to the Receive channels, when a transmit channel has enough space for data in its TxFIFO, it requests to fill it with data from the external memory, as specified by the channel's current descriptor. The channel does so when enough space to perform a burst is available. The Channel's DMA enqueues a service request to one of two service queues; High Priority Queue and a Low Priority Queue. The bit <TPC> field in the MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31) (Table 1043 p. 1296), defines which queue the request be inserted in. Alternatively, the queues are serviced in a Weighted Round Robin fashion as specified in the <TSQW> field in the Tx Service Queue Arbiter Weight Register (Table 1046 p. 1297). This field determines the number of High Priority Entries to be serviced before one Low Priority Entry.
A value of 0x1F indicates unlimited number of high priority entries served before one low priority entry is served. A value of 0x0 indicates unlimited number of low priority entries served before one high priority entry is served.

23.5.7 Transmit MCDMA

23.5.7.1 Transmit MCDMA Definitions

- SOF (Start Of Frame descriptor): Descriptor with F (First) bit set to 1.
- EOF (End Of Frame descriptor): Descriptor with L (Last) bit set to 1.

F and L bits are set by the CPU before releasing a descriptor to the MCDMA for transmission.

A frame starts with an SOF descriptor and ends with an EOF descriptor. A frame may consist of one buffer or may be divided over many buffers. When a frame is stored in one buffer, the associated descriptor has both F and L bits set to '1'.

23.5.7.2 Transmit MCDMA Flow

The following steps are executed during a normal transmit MCDMA process:

1. Before enabling a MCDMA Tx channel, the CPU must prepare a valid descriptor with the owner bit set to '1'.
2. If it is not yet setup, the CPU must create the linked list of the internal TxFIFO blocks. Refer to Section 23.5.4 "Receive and Transmit FIFO Management Linked List" on page 400
3. The CPU must then write the first descriptor address to the MCDMA Current Transmit Descriptor Pointer (MCTDP) register.
4. The CPU issues a Transmit Demand command.
5. When buffer transmission is complete, the MCDMA closes the buffer descriptor by setting the correct transmit status and by writing "0" in the Owner Bit, returning the buffer to the CPU.

23.5.7.3 Transmit MCDMA Notes

The transmit MCDMA stops the DMA process whenever it reaches a descriptor with NULL (0x00000000) value in the NDP (Next Descriptor Pointer) field or when it fetches a descriptor with Owner Bit set to "0". In such a case, the MCDMA controller clears the TxD bit before returning to IDLE state.

In normal operation, the transmit MCDMA never expects to find a NULL NextDescriptorPointer or Not-Owned descriptor in the middle of a frame. When this occurs, the transmit MCDMA controller halts, the TxD bit is cleared and a maskable Channel TX End interrupt is generated into the Interrupt queue, see <ChTxEnd> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).

When the CPU needs to interfere with the transmit process without corrupting the ongoing transmit process, it can issue a STOP command by writing "1" to the STD bit in the MCDMA channel's control register. The transmit MCDMA controller stops after completing the transmission of the active frame.

When issuing an STD command, the TXD is reset to '0' after the current packet transmission to the MCSC is completed. The CPU may then issue a new Transmit Demand command in order to restart the MCDMA process.

23.5.8 Receive MCDMA

Receive MCDMA Definitions

Table 111: MCDMA Definitions

Term	Definition
SOF	Start Of Frame descriptor Descriptor with F (First) bit set to 1.
EOF	End Of Frame descriptor Descriptor with L (Last) bit set to 1.

F and L bits are set by the MCDMA before releasing a descriptor to the CPU.

A frame starts with an SOF descriptor and ends with an EOF descriptor. It may be contained in one buffer or split over many buffers. When a frame is stored in one buffer, the associated descriptor has both F and L bits set to '1'.

Receive MCDMA Flow

The following steps are executed during a normal receive MCDMA process:

1. Before enabling a MCDMA Rx channel the CPU must prepare a valid descriptor with the owner bit set to ‘1’.
2. If it is not yet setup, the CPU must create the linked list of the internal RxFIFO blocks. Refer to Section 23.5.4 “Receive and Transmit FIFO Management Linked List” on page 400.
3. The CPU must then write the descriptor address to the MCDMA Current Receive Descriptor Pointer (MCRDP) register before enabling the receive MCDMA channel.
4. The CPU writes ‘1’ to the ERD bit in the RMCC register, enabling the receive MCDMA channel.
5. Normally, the receive MCDMA controller runs continuously, processing received data from the MCSC.

Note

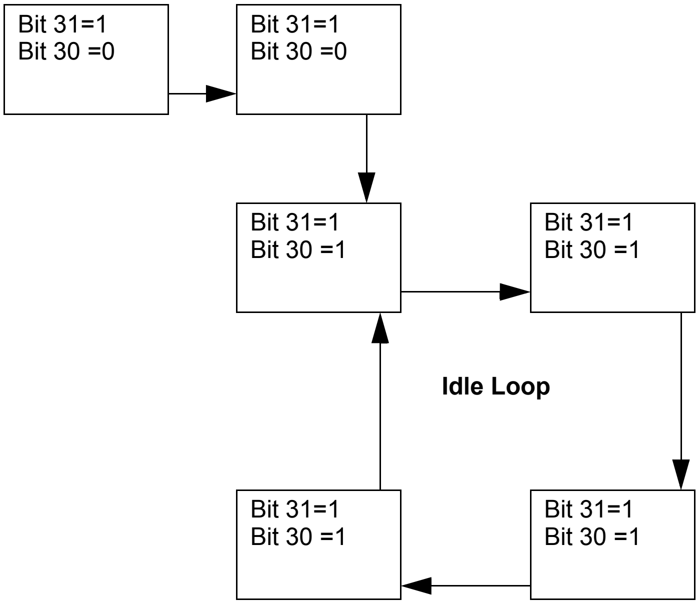
- The receive MCDMA controller never expects to encounter a descriptor with owner bit set to ‘0’ or a NULL value (0x00000000) in the NDP field. If this occurs, the receive MCDMA Channel aborts and a maskable Channel RX Error interrupt is generated into the Interrupt queue, see <ChRxErr> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1313).
- The CPU uses the receive abort command to stop the receive MCDMA Channel. It is the CPU’s responsibility to restart the descriptor chain properly.

23.5.9 MCDMA in Auto Mode

The CPU may set bit 30 in the command/status field of transmit or receive descriptors directing the MCDMA to work in Auto Mode.

When working with an Auto Mode descriptor, the device’s MCDMA works regularly except that it does not clear the Ownership bit when closing the descriptor. An example of usage is when the CPU may use this to cause the MCDMA to transmit endlessly (until CPU intervention).

Figure 103:Using Auto Mode to create Idle Loop



23.6 Interrupts

The following section describes the interrupt assertion mechanism of the TDM controller.

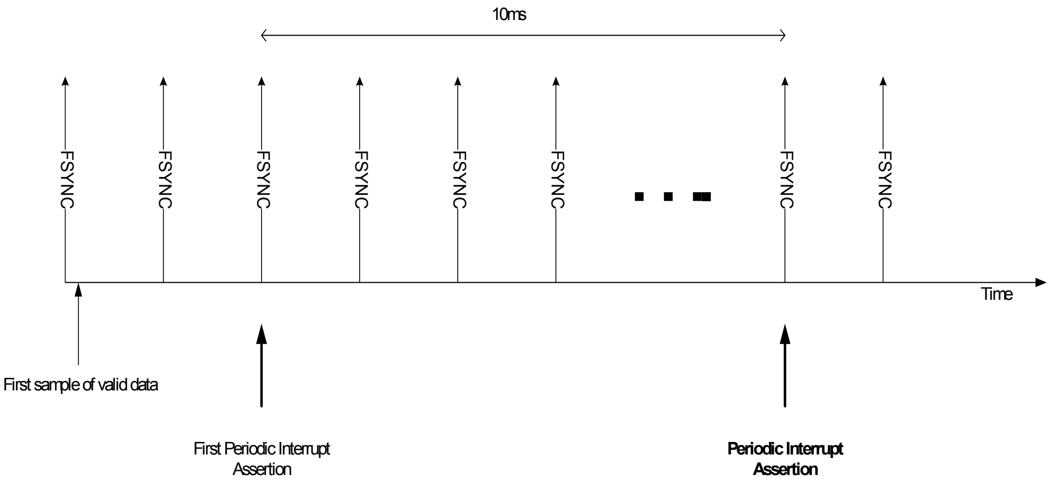
23.6.1 Periodic Interrupt Assertion

The TDM controller implements a periodic interrupt assertion mechanism for proper synchronization between the software voice stack and the hardware engines. This mechanism provides the flexibility of setting a programmable repeated interrupt signal at a programmable offset from FSYNC assertion. This guarantees that by the time the software accesses the voice buffer in memory, data is available for all active channels and is not still buffered in any internal memory.

The timing of the first interrupt assertion is also programmable and is related to the first sample of valid data. The granularity is of FSYNC assertions. For example, one can choose to set the first interrupt four FSYNC assertions after the first sample of valid data.

Figure 104 provides an example for setting a periodic interrupt once every 10 ms (80 FSYNC assertions—a typical voice buffer size) and having it shifted by two FSYNC assertions from the first sample of valid data.

Figure 104:Periodic Interrupt Example


10ms
FSYNC
FSYNC
FSYNC
FSYNC
FSYNC
FSYNC
FSYNC
FSYNC
Time
First sample of valid data
First Periodic Interrupt Assertion
Periodic Interrupt Assertion

There are two periodic interrupt signals for RX and TX synchronization. It is also possible to use only one of the signals to synchronize both RX and TX operations.

The Voice Periodic Interrupt Control Register contains the relevant configuration settings for periodic interrupt generation.
To configure the offset of the first interrupt assertion configure the <TxVoicePeriodIntGap> field and/or the <RxVoicePeriodIntGap> field in the Voice Periodic Interrupt Control Register (Table 1142 p. 1356) to the desired number of FYSNC from the first valid data sample.

To configure the time interval in FSYNC period granularity between any two interrupt assertions, set the <TxVoicePeriodIntLen> field and/or the <RxVoicePeriodIntLen> field in the Voice Periodic Interrupt Control Register (Table 1142 p. 1356) to the desired number of FSYNC assertions that represent the selected time interval.

All interrupt assertions are reported via a maskable event to the CPU core and are logged in the <RxVoiceInterrupt> field in the TDMMC Functional Cause Register (Table 1138 p. 1354).

23.6.2 Single-Interrupt Queue Controller (IQC) Mode

The device integrates an Interrupt Queue Controller (IQC) which handles interrupts that are created and managed on a per channel basis: that is, only channels that are handled by the MCSC and MCDMA. The IQC manages its interrupt queue in the external memory and uses the following four address pointers for that purpose:

- Interrupt Queue First (IQF)—See the IQC0 (Interrupt Queue) First Entry Address - IQF0 Register (Table 1056 p. 1310)
- Interrupt Queue Last (IQL)—See the IQC0 (Interrupt Queue) Last entry Address - IQL0 Register (Table 1057 p. 1310)
- Interrupt Queue Head (IQH)—See the IQC0 (Interrupt Queue) Head Address - IQH0 Register (Table 1058 p. 1311)
- Interrupt Queue Tail (IQT) pointers—See the IQC0 (Interrupt Queue) Tail Address - IQT0 Register (Table 1059 p. 1311)

The Interrupt Queue First (IQF) and Interrupt Queue Last (IQL) define the boundaries of the queue. The defined memory space must be contained in the same DRAM physical rank. The two registers must be setup by the CPU. The interrupt queue controller uses them for read only. The Interrupt Queue Head (IQH) points to the next empty entry in the queue, in which it inserts the interrupt details. This pointer is read and updated by the interrupt queue controller and can be read by the CPU. The Interrupt Queue Tail (IQT) points to the next entry that the CPU reads and processes. The entry is read and updated by the CPU and the interrupt queue controller uses it only to detect queue overflow.

Whenever the boundaries of the interrupt queue in the external memory are changed (either by writing to the Interrupt Queue First or Interrupt Queue Last registers), the Interrupt Queue Head is automatically reset to the value of the Interrupt Queue First register.

Whenever the interrupt queue controller inserts an interrupt entry to the queue, it issues a normal interrupt through the conventional serial interrupt cause registers.

For an overflow caused by an over-loaded CPU or improper handling of the interrupt queue, the IQC issues an interrupt through the conventional serial interrupt cause registers.

Each IQC message (entry) that is reported by the hardware holds the number of the channel that the interrupt relates to and information about the specific event that triggered the interrupt assertion, as listed in Table 112. Enable each of the events to be reported to the IQC via the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1311). For instance, if the CPU sets the bit <ChRxErr>, then for each Rx channel that encounter a resource error, an appropriate interrupt entry is queued to the interrupt queue in memory.

Note

Each entry can hold more than one active bit, indicating two or more events that the channel experienced simultaneously.

Table 112: IQC Interrupt Entry Structure

Bits	Name	Description
4:0	ChId	Channel ID This field contains the channel number that encounters an event causing this entry to be inserted in the interrupt queue.
15:5		Reserved.

Table 112: IQC Interrupt Entry Structure (Continued)

Bits	Name	Description
16	ChRxBuf	MCDMA ChannelX Rx Buffer Return See <ChRxBuf> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1313).
17	ChRxErr	MCDMA ChannelX Rx Error See <ChRxErr> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1313).
18	ChTxBuf	MCDMA ChannelX Tx Buffer Return See <ChTxBuf> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1313).
19	ChTxEnd	MCDMA ChannelX Tx End See <ChTxEnd> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
20	ChRxAbrt	MCSC ChannelX Abort command executed See <ChRxAbrt> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
21	ChRxEH	MCSC ChannelX Enter Hunt Command Executed See <ChRxEH> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
22	ChRxCRD	MCSC ChannelX Close Rx Descriptor Command Executed See <ChRxCRD> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
24:23		Reserved
25	ChRxStts	MCDMA ChannelX Closed Rx Descriptor with Non Zero Status See <ChRxStts> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
26	ChTxEIDL	MCSC ChannelX Tx Entered Idle State See <ChTxEIDL> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
27	ChTxStts	MCDMA ChannelX Closed Tx Descriptor with Non Zero Status See <ChTxStts> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
28	ChRxIL	MCSC ChannelX Rx moved to Idle Line State See <ChRxIL> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1312).
29	ChRxEIL	MCSC ChannelX Rx Exit Idle Line State See <ChRxEIL> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1311).
30	ChTxEOP	MCSC ChannelX Tx End Of Packet See <ChTxEOP> field in the IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1311).
63:31		Reserved.

23.6.3 Multiple-IQC Mode vs. Single-IQC Mode

The device uses up to four CPUs. The Interrupt Queue Control (IQC) mechanism is designed to operate with one master (CPU). To be able to divide the interrupt processing between the four CPUs, there is an option to use up to four IQCs. This mode of operation is called Multiple-IQC mode. The IQC mechanism has its own interrupt causes that notifies the CPU regarding *not Empty* status and alarms in case of *overrun* status of the external buffer. These interrupt causes were duplicated for each CPU. Thus, there are additional interrupt causes that are relevant for this mode.

Recommended Usage

The default case is the Single-IQC mode.

To operate the Multiple-IQC mode,

1. Configure the set of registers (IQC0–IQC3) First, Last, Head, and Tail registers as described in those registers. descriptions) for each one of the IQCs (see registers for IOC0 below.).
   - IQC0 (Interrupt Queue) First Entry Address - IQF0 Register (Table 1056 p. 1310)
   - IQC0 (Interrupt Queue) Last entry Address - IQL0 Register (Table 1057 p. 1310)
   - IQC0 (Interrupt Queue) Head Address - IQH0 Register (Table 1058 p. 1311)
   - IQC0 (Interrupt Queue) Tail Address - IQT0 Register (Table 1059 p. 1311)
2. Configure the required bits in the IQC0–3 enable interrupt registers:
   - IQC0 (Interrupt Queue) Enable Interrupt Register (Table 1060 p. 1311)
   - IQC1 (Interrupt Queue) Enable Interrupt Register (Table 1075 p. 1316)
   - IQC2 (Interrupt Queue) Enable Interrupt Register (Table 1076 p. 1318)
   - IQC3 (Interrupt Queue) Enable Interrupt Register (Table 1077 p. 1320)
3. Set the IQC0-3 Channels responsibility. The user must not set the same channel in two different IQCs. Each IQC should be responsible for a unique group of channels.
   - IQC0 Channels Responsibility Register (Table 1079 p. 1324)
   - IQC1 Channels Responsibility Register (Table 1080 p. 1324)
   - IQC2 Channels Responsibility Register (Table 1081 p. 1324)
   - IQC3 Channels Responsibility Register (Table 1082 p. 1324)
4. Set the <IQCMultEn> field in the MCSC Global Configuration Extension Register (Table 1078 p. 1323).

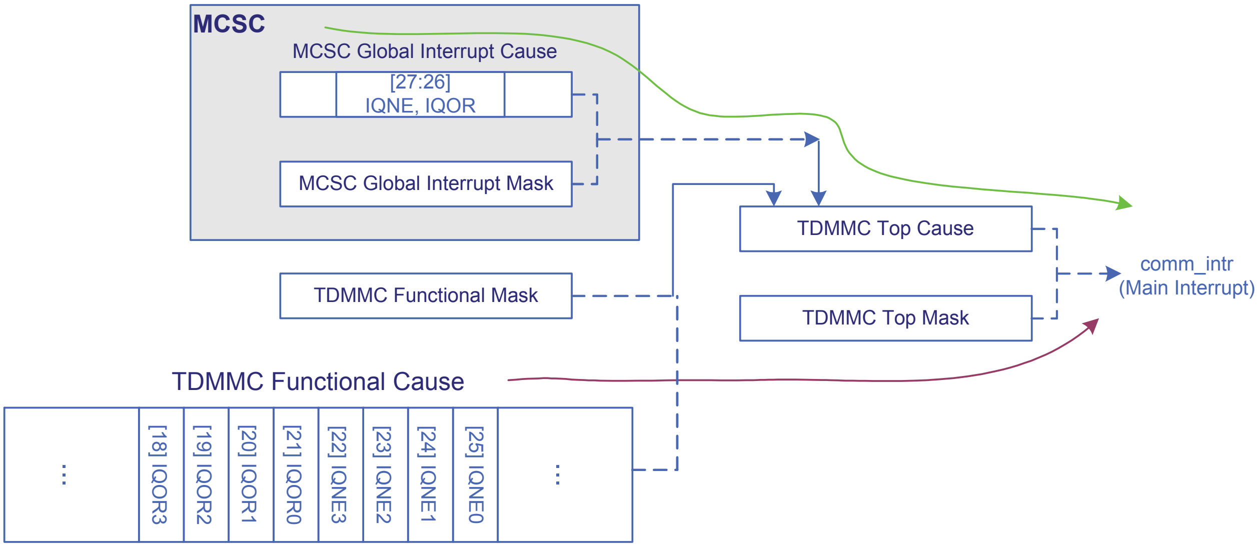
Multiple-IQC mode While using the Multiple-IQC mode, each CPU should relate to its <IQNE> (Interrupt Queue Not Empty) and <IQOR> (Interrupt Queue OverRun) cause field within the TDMMC Functional Cause Register (Table 1138 p. 1352).

Single-IQC mode In Single-IQC mode, these cause bits are active in the MCSC Global Interrupt Cause Register (Table 1052 p. 1306). This cause register effects the TDMMC Top Cause Register (Table 1137 p. 1351).

Figure 105 provides the flow for the relevant interrupt cause and mask registers for Single-IQC mode and Multiple-IQC mode.

- The green pattern is relevant for Single-IQC mode.
- The purple pattern is relevant for the Multiple-IQC mode.

Figure 105:MCSC Global Interrupts and Mask—Flow



23.7 Voice Application Flows

The TDM controller can be used for the Voice over IP (VoIP) application. This application requires that all of the used channels be synchronized by the starting phase and byte alignment. This means that once the TDM controller starts to transmit the data, it must send the first bytes of all channels in the first TDM frame. When using Linear mode, the TDM controller must send the first two bytes for each channel in the first TDM frame. The flows that are described in the following sub-sections should be used to ensure these requirements are met.

In the VoIP application, the TDM controller is configured to transmit and receive 32 channels constantly while data transmission/reception from open phone calls fills the content of specific channels. When there is no expected data transmission/reception from open phone calls, the CPU might want to stop the TDM controller from operating to save power. In this case, the TDM controller runs an abort process.

Note

The Abort sequence must be implemented in a strict manner as described in Section 23.7.2, Aborting Flow, on page 410. Otherwise, channel balancing problems or byte alignment problems in Linear mode or both of these problems can occur following re-initialization of the TDM controller after the Abort process.

Point to consider in voice application flows

Consider the following points when using the voice application flows described in the sections below.

- When using Output-Sync (<TdmFsyncOEn> =0x1 in the TDM Clock and Sync Control Register (Table 1134 p. 1349)), the user must configure the <FsyncBitcount> field in the TDMMC Output Sync Bit Count Register (Table 1141 p. 1355), in accordance with DPRAM

Copyright © 2014 Marvell

programming, to activate the output Fsync signal. The Fsync signal can be activated with no relation to the setting of FTDM-Enable (<TEN> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1347)).

- As a part of the initialization process, the second half of the DPRAM is configured to be the same as the first half of the DPRAM with one change: all entries in the second half should be configured with MASK=0. The default value of the <TR2HALF> field and the <RR2HALF> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1348) are 0x0 and they should stay at this value until a step in one of the follow processes instructs otherwise.
- To avoid redundant access to DRAM by the MCDMA immediately after the TDM controller has initialized (assuming there are no active calls at this stage), initiate a single start/stop sequence to MCSC/MCDMA.
- The <TEN> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1347) should not be reset to 0! TDM should remain active to retain the link with the SLIC.
- When working in Linear mode, the user should configure the TDM with <RPT>=1. This field is located in the in the FlexTDM Receive Dual Port RAM (RDPR)<n> Register (n=0–255) (Table 1130 p. 1345) / FlexTDM Transmit Dual Port RAM (TDPR)<n> Register (n=0–255) (Table 1129 p. 1344)
- When setting the DPRAM <RPT> field to 1, 2, or 3, in either or both of the registers mentioned above, then all non-active entries before the entry marked with L must have the same RPT configuration as the active entries.
- The user can skip the Enter Hunt stage (when using transparent mode—for VoiP applications). If using Enter Hunt, wait 2 Fsync cycles and turn off the Enter Hunt command. Do not try to poll the enter-hunt execution status.

23.7.1 Initialization Flow

The TDM controller consists of several sub-units. For proper operation, after the initialization of the sub-units, the ENABLING SEQUENCE of the sub-units should be implemented in this specific order:

1. Configure the <TxEn> field in the MCSC Global Configuration Register (Table 1051 p. 1305).
2. Configure each channel in:
   - <ET> field in the MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31) (Table 1050 p. 1304)
   - <ERD> field in the MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31) (Table 1037 p. 1292)
3. Configure the <RxEn> field in the MCSC Global Configuration Register (Table 1051 p. 1305).
4. Configure each channel in the:
   - <ER> field in the MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Table 1049 p. 1300)
   - <TXD> field in the MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31) (Table 1043 p. 1295)
5. Configure the <TEN> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1347).

Copyright © 2014 Marvell

23.7.2 Aborting Flow

Note

There are two important things that will be implemented during the abort process:

- The TDM Sync and clock signals (in Output mode) remain running to keep the SLIC synchronized and avoid requiring SLIC re-initialization when resuming TDM controller operation. The SLIC initialization is time consuming, and can negatively effect performance. Therefore it is important that it keep operating.
- Once the TDM controller has aborted operation, it will not access the DDR since the DDR might be in self-refresh. This means that the MCDMA will stop accessing buffers and descriptors.

To stop the TDM controller from operating when working in, in a controllable way, make sure to follow these steps. Working in AutoMode (AM) and deciding to stop:

1. For *each* channel in MCDMA TX:
   a) Reading the current descriptor pointer to get the next descriptor address.
   b) Updating the Next Descriptor to as follows:
   LAST=1
   AM=0
   NextDescAddr=Null
   Ownership=MCDMA
2. Follow the same procedure as in step 1 for each channel in MCDMA RX.
3. Checking that each one of the channels has stopped by polling the ERD and TXD:
   a) <ERD> field in the MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31) (Table 1037 p. 1292) should change from 0x1 to 0x0, indicating that the DMA per each channel has stopped the Receive process.
   b) <TXD> field in the MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31) (Table 1043 p. 1295) should change from 0x1 to 0x0, indicating that the DMA per each channel has stopped the Transmit process.
4. At this stage the RX and TX DMAs in TDM controller have been stopped. This means that the TDM controller will not write or read from the DDR, while its TDM is still active.
5. At this stage, the TDM controller must be moved to the second half of the TDM (configured with MASK=0) following these steps:
   a) Set the <TR2HALF> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1348) to 0x1.
   b) Set the <RR2HALF> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1348) to 0x1.
   c) Wait for at least 1 frame.

Note

The user should avoid using the Abort feature that is available in MCSC-RX and MCDMA-TX registers.

23.7.3 Resuming Flow

After stopping the TDM controller, following the Aborting Flow, follow the these steps to resume TDM controller operation:

1. Re-Initialization of MCDMA by updating descriptors and return to AutoMode.
2. Waiting a period of 1-TDM-sync cycle.
3. For each channel, enable ERD and TXD:
   a) Set <ERD> field in the MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31) (Table 1037 p. 1292) to 0x1.
   b) Set <TXD> field in the MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31) (Table 1043 p. 1295) to 0x1.
4. Clear <TR2HALF> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1348) to 0x0.
5. Clear <RR2HALF> field in the FlexTDM0 Configuration Register (TCR) (Table 1133 p. 1348) to 0x0.

23.7.4 Using Linear Mode

In Linear mode each sample of a channel contains 2B of data.

23.7.4.1 Configuring FlexTDM for Linear Mode

There are two methods to implement the Linear mode in the TDM DPRAM. However, Marvell® recommends only one of them.

Method1 (Recommended) In this method, set the <RPT> field to 0x1 in each entry in the FlexTDM Receive Dual Port RAM (RDPR)<n> Register (n=0–255) (Table 1130 p. 1345) / FlexTDM Transmit Dual Port RAM (TDPR)<n> Register (n=0–255) (Table 1129 p. 1344). This ensures that the specific channel within each entry will have 2 consecutive time slots within the TDM frame.

Method2 (Not Recommended) In this method, 2 DPRAM entries are allocated per channel.

The following diagram can be used to understand the recommended method.

Method1 (Recommended)			Method2 (Not Recommended)	
Entry0	Ch0, RPT=1		Entry0	Ch0, RPT=0
Entry1	Ch1, RPT=1		Entry1	Ch0, RPT=0
Entry2	Ch2, RPT=1		Entry2	Ch1, RPT=0
...	...		Entry3	Ch1, RPT=0
			...	...

23.7.4.2 Using Linear Mode with MSB-First

For SLIC devices, which require receiving the MSbit and MSByte first, when using Linear mode:

The sample should be represented in MSB-first format when using both:

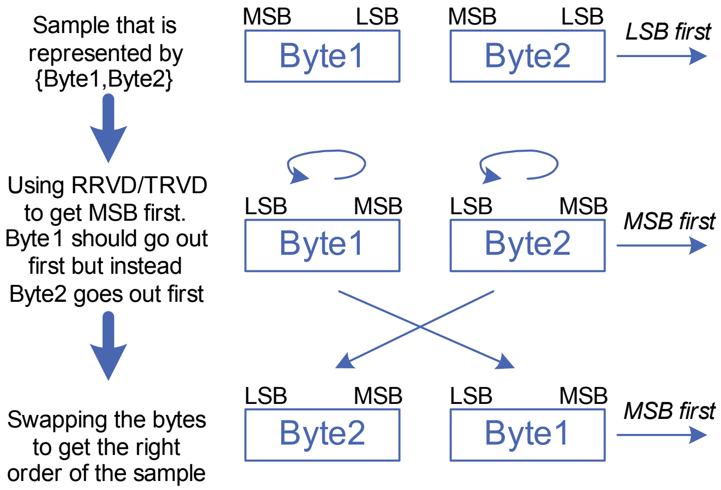
- Linear Mode (one sample represented by 2-bytes per channel)
- The <RRVD> field in the MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Table 1049 p. 1299) or the <TRVD> field in the MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31) (Table 1050 p. 1303).

Copyright © 2014 Marvell

However, the <RRVD> or the <TRVD> functions per byte (arranged MSB2LSB per each byte and not for the results of 2 bytes). Thus, in this case MSB-first is implemented per byte, but the 2 bytes appear to be swapped.

To fix that, the user must set the <LinearRxByteSwap> field and the <LinearTxByteSwap> field in the MCSC Global Configuration Extension Register (Table 1078 p. 1323). Setting those fields to 0x1 will swap the bytes as shown in Figure .

Linear Mode Flow Data Byte Swap



24 Secure Digital Input/Output (SDIO) Controller

This section describes the Secure Digital (SD) and MultiMedia card (MMC) host controller unit integrated into the device.

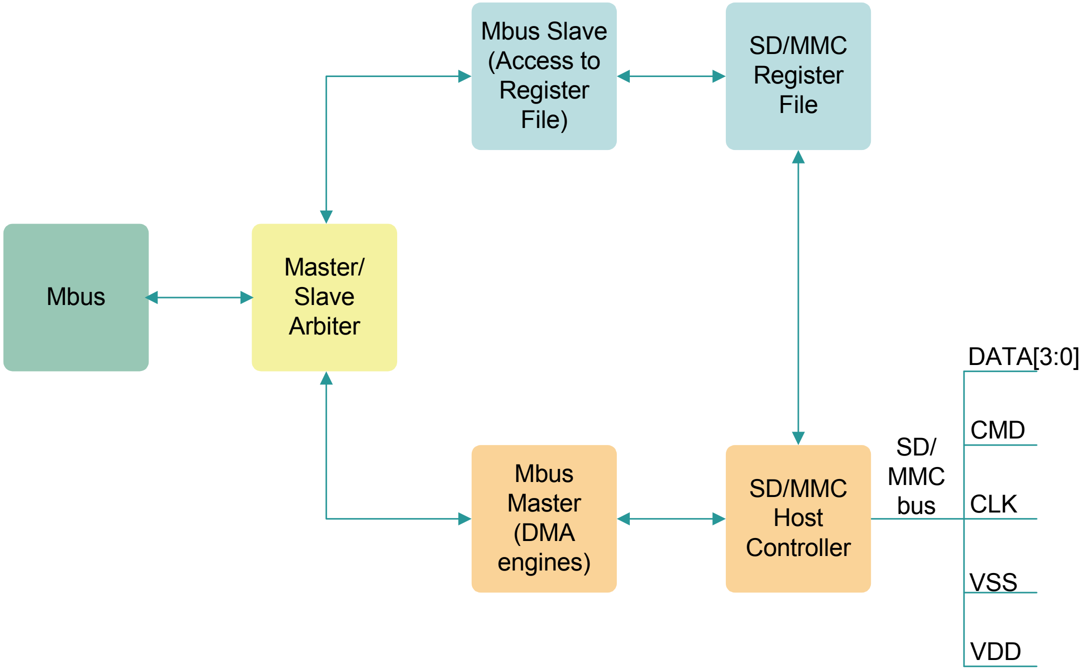
The Secure Digital (SD) and MultiMedia card (MMC) host controller unit functions as a host for the SD/MMC bus to transfer data through the Mbus between SDMem, SDIO, and MMC cards on one side and system buffers (see Figure 106).

One side of the unit interfaces with a standard SD/MMC host bus.

The other side is programmable to interface either:

- Directly with the CPU—Mbus Slave
- From the DMA engines to the DRAM—Mbus Master

Figure 106:SD_MMC Host Controller Hardware Block Diagram



Communication over the SD/MMC bus is based on commands and data bit streams. They are initiated by a start bit and terminated by a stop bit.

Command A command is a token that starts an operation. A command is sent from the host to the card(s). The command is transferred serially on the CMD line and multiplexed on the MPP as SD_CMD. The SD_CMD is bidirectional.

Command response A command response is a token that is sent from an addressed card to the host as an answer to a previously received host command. The response is transferred serially on the CMD line and multiplexed on the MPP as SD_CMD.

Data There are four bidirectional data transfer lines used to transfer data from the card to the host or vice versa. They are multiplexed on the MPP as SD_D[3:0].

SD_CLK This clock synchronizes the SDIO bus of the device.

Note

For more information, refer to the *SD Memory Card Specifications*.

Use the <ClkDvdrMValue> field in the Clock Divider Value Register (Table 1197 p. 1389) to configure the SDIO clock.

For example, to set the SDIO clock to 50 MHz, the <ClkDvdrMValue> field is set to 1, as in:

SDIO clock = 100M/(m + 1) = 50 Mhz

For the maximum SDIO clock frequency, see the “Clocking” section in the device Hardware Specifications.

The SDIO registers are located in Appendix A.14, Secure Digital Input-Output (SDIO) Registers, on page 1360.

24.1 Features

The SDIO features supported by the device, and those features that are not supported are:

Features supported:

- 1-bit/4-bit SDMem, SDIO, and MMC cards
- Up to 50 MHz for SD and MMC
- Multiple MMC cards sharing a common bus
- Interrupts for information exchange between host and cards
- Read wait commands in SD cards
- Hardware generate/check CRC on all command and data transaction on card bus
- Suspend/resume in SDIO cards
- Stream read/write in MMC cards

Features not supported:

- Interrupts mode in the MMC card
- SPI mode

24.2 Functional Description

24.2.1 SDMem, MMC, and SDIO Arbitration Scheme

Host and SD/MMC cards go through two phases following each power on reset, software reset, or addition of a new card (see Figure 107):

Card identification phase The host looks for new cards on the bus. While in this phase, the host resets all the cards that are in Card Identification mode. Any card that is already identified will not be reset. Then the host sends the command to validate the operation voltage range, identify the card, and request the card's Relative Card Address (RCA).

- In SDmem, this operation is done to each card separately on its own CMD line.
- In the MMC system, it is done on a shared CMD line.

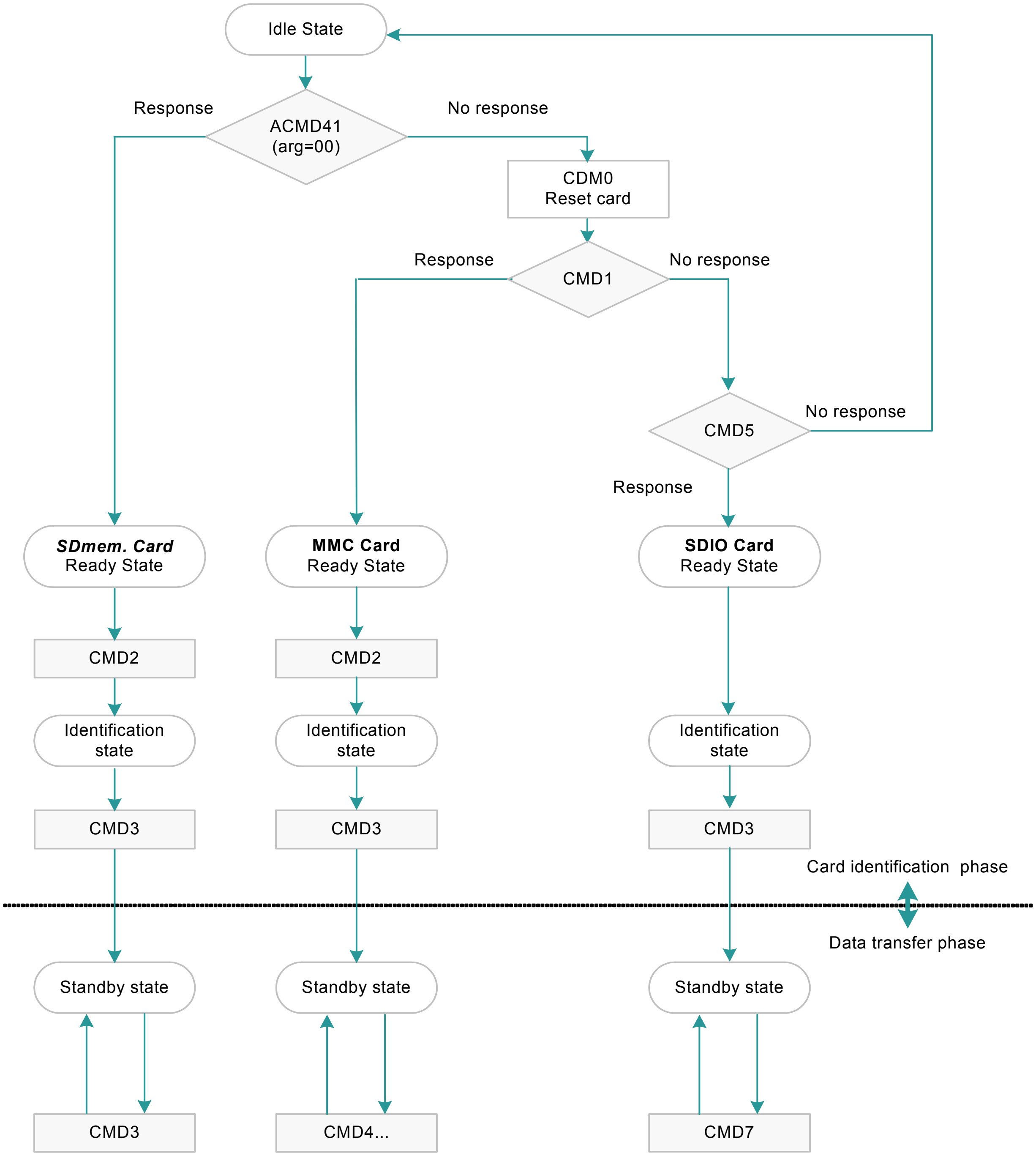
All data communication in the Card Identification phase uses the CMD line only. The host starts the card identification process with an identification clock rate fOD that is slow (see *SD Memory Card Specifications)*.

Data transfer phase The host enters the data transfer phase after identifying all the cards on the bus. In this phase, the host is ready to transfer data.

After the host driver completes the setup of the host controller, it starts data transfer by writing to the Command Register (Table 1156 p. 1365). Therefore, the Command register has to be written last.

Figure 107 shows the Host initialization flow.

Figure 107:Host Initialization Flow



24.2.2 Difference Between SD Cards and MMC Cards

The SD card protocol is designed to be a super-set of the MultiMedia card (MMC) protocol. However, besides some different commands in the initialization protocol, the only difference between the SD and MMC cards is the bus topology.

SD card Each SD card has a dedicated, independent point-to-point connection—containing its own CLK, CMD, DATA lines—to the host. The I/O pads of the SD card are a push-pull type.

MultiMedia card The MMC card bus is a shared bus, between multiple MMC cards with the MMC cards identified serially, one at a time. Therefore, during initialization, MMC cards use open-drain I/O pad types. In the data transfer phase, MMC cards switch to push-pull I/O type just like the SD cards.

To the host, this shared bus makes no difference. However, each of the MMC cards has to monitor the bus to determine whether or not it has won the bus.

To support both bus topologies, the host controller functions as an open-drain during the initialization phase, and functions as a push-pull type during the data transfer phase. For the CMD and DATA lines, for both SD cards and MMC cards, the type needs to be pull-up.

24.2.3 SDIO/SDMem/MMC Host Controller Initialization

If the SD/MMC host controller requires a configuration different from the default, the software should configure the following registers:

- Mbus Control Low Register (Table 1187 p. 1385)
- Mbus Control High Register (Table 1188 p. 1385)
- Window0 Control Register (Table 1189 p. 1386)—size, target, and attributes
- Window0 Base Register (Table 1190 p. 1387)
- Window1 Control Register (Table 1191 p. 1387)—size, target, and attributes
- Window1 Base Register (Table 1192 p. 1387)
- Window2 Control Register (Table 1193 p. 1388)—size, target, and attributes
- Window2 Base Register (Table 1194 p. 1388)
- Window3 Control Register (Table 1195 p. 1388)—size, target, and attributes
- Window3 Base Register (Table 1196 p. 1389)

24.2.4 SDIO/SDMem/MMC Command Execution

The SD/MMC command is activated by configuration of the relevant SD registers. Some commands (e.g., Auto Cmd12) require the setting of dedicated registers before executing the command. Software will set appropriate registers before writing Command to the Command Register (Table 1156 p. 1365). Upon writing Command to the Command register, the hardware automatically starts command execution, data transfer, and CRC generation or checking. Therefore, the Command register is the last register to be written by software. The command completion could be detected by interrupt assertion or by polling Interrupt Status Registers.

Table 113 shows an example of the software flow for the SDIO command execution. Configure the registers listed in this table:

Table 113: Software Flow

Register Name	Field
DMA Buffer Address 16 LSB Register (Table 1149 p. 1361)	<DmaAddrLo>
DMA Buffer Address 16 MSB Register (Table 1150 p. 1361)	<DmaAddrHi>

Table 113: Software Flow (Continued)

Register Name	Field
Data Block Size Register (Table 1151 p. 1362)	<BlockSize>
Data Block Count Register (Table 1152 p. 1362)	<BlockCount>
Argument in Command 16 LSB Register (Table 1153 p. 1362)	<ArgLow>
Argument in Command 16 MSB Register (Table 1154 p. 1363)	<ArgHigh>
Host Control Register (Table 1168 p. 1370)	<PushPullEn> <CardType> <BigEndian> <LsbFirst> <DataWidth> <TimeoutValue> <TimeoutEn>
Data Block Gap Control Register (Table 1169 p. 1372)	<Resume>
Clock Control Register (Table 1170 p. 1374)	<SclkMasterEn>
Argument in Auto Cmd12 Command 16 LSB Transferred Register (Table 1181 p. 1383)	<AutoCmd12ArgLo>
Argument in Auto Cmd12 Command 16 MSB Transferred Register (Table 1182 p. 1384)	<AutoCmd12ArgHi>
Index of Auto Cmd12 Commands Transferred Register (Table 1183 p. 1384)	<AutoCmd12BusyChkEn><AutoCmd12IndexChkEn> <AutoCmd12Index>
Transfer Mode Register (Table 1155 p. 1363)	<SwWrDataStart> <HwWrDataEn> <AutoCMD12En> <IntChkEn> <DataXferTowardHost> <StopClkEn> <HostXferMode>
NOTE: After this sequence, the Command Register (Table 1156 p. 1365) fields must be configured. Writing to the command register is the trigger for SDIO command execution.	
Command Register (Table 1156 p. 1365)	<RespType> <DataCrc16ChkEn> <CmdCrcChkEn> <CmdIndexChkEn> <DataPresent> <UnexpectedRespEn> <CmdIndex>

24.2.5 SDIO/SDMem/MMC Interrupts

The SD/MMC interrupts are registered in two Interrupt Cause registers:

- Normal Interrupt Status Register (Table 1172 p. 1375)
- Error Interrupt Status Register (Table 1173 p. 1376)

The interrupt going from the unit to the CPU is generated as a bitwise OR of all bits of these registers. Upon an interrupt event, the corresponding cause bit is set to 1. It is cleared upon a software write of 0.

Each bit in the Normal Interrupt Status Register can be enabled or disabled by the relevant bit in the Normal Interrupt Status Enable Register (Table 1174 p. 1378).

The impact of each bit on SD/MMC interrupt generation may be enabled or disabled by the relevant bit in Normal Interrupt Status Interrupt Enable Register (Table 1176 p. 1380).

Each bit in Error Interrupt Status Register can be enabled or disabled by the relevant bit in the Error Interrupt Status Enable Register (Table 1175 p. 1379).

The impact of each bit on SD/MMC interrupt generation may be enabled or disabled by the relevant bit in Error Interrupt Status Interrupt Enable Register (Table 1177 p. 1381).

The following interrupt events are supported:

Normal Interrupts:

- Command Complete
- Transfer Complete
- Block gap event
- DMA interrupt
- Tx ready—the FIFO has room for the CPU to write 16 bits of data.
- Rx ready—the FIFO contains at least 1 byte of data ready to be read by the CPU.
- Cards interrupt.
- Read Wait state is on.
- There are at least eight filled entries for data to be read from the FIFO.
- There are at least eight empty entries for data to be written in the FIFO.
- The hardware is suspended.
- Auto_cmd12 is completed.
- Unexpected response from devices detected.

Error Interrupts:

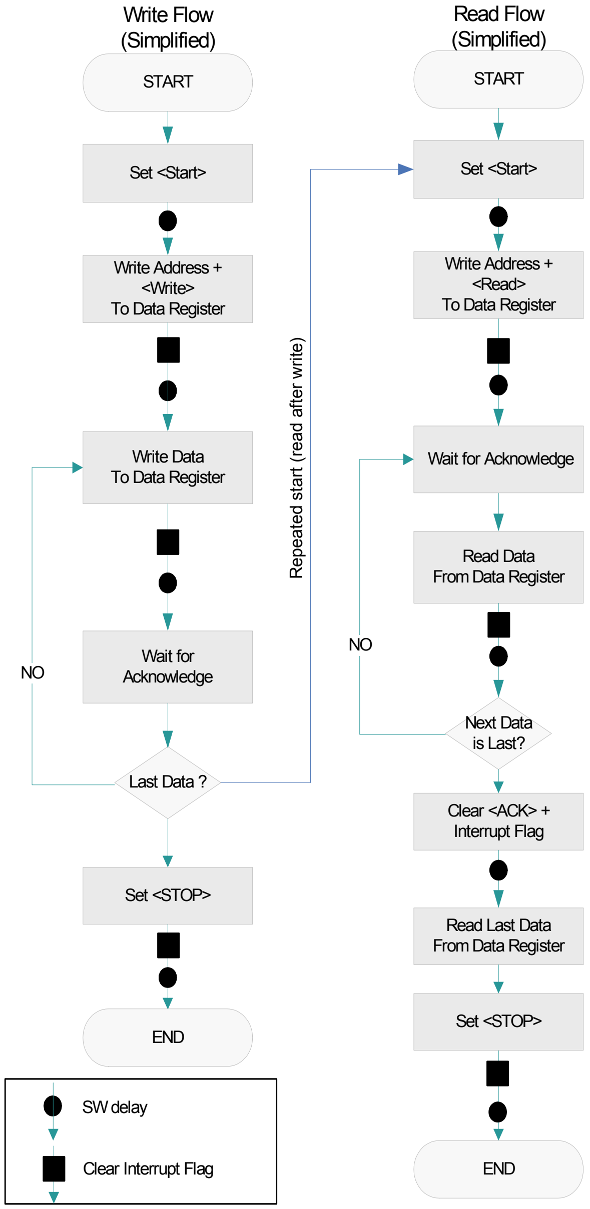
- Command timeout error
- Command CRC error
- Command end bit error
- Command start bit error
- Command index error
- Auto CMD12 error
- Data timeout error
- Read data CRC error
- Read data end bit error
- Transfer size mismatched error
- Response T bit error
- CRC end bit error
- CRC start bit error
- CRC status error

25 Inter-Integrated Circuit Interface (I^2C)

This section describes the device's 2 general purpose Inter-Integrated Circuit Interface (I^2C) ports. Each port can act as an I^2C master or as a slave. Figure 108 provides a high-level (simplified) I^2C transaction flow diagram. Refer to Section 25.3, I^2C Functional Description for full details.

If enabled via reset strap, port0 also supports initialization from external I^2C serial ROM.

Figure 108: I^2C Transaction Flow Diagram


Write Flow (Simplified)
START
Set <Start>
Write Address + <Write> To Data Register
Write Data To Data Register
Wait for Acknowledge
NO
Last Data ?
Set <STOP>
END
Repeated start (read after write)
Read Flow (Simplified)
START
Set <Start>
Write Address + <Read> To Data Register
Wait for Acknowledge
Read Data From Data Register
NO
Next Data is Last?
Clear <ACK> + Interrupt Flag
Read Last Data From Data Register
Set <STOP>
END
SW delay
Clear Interrupt Flag

The I^2C registers are located in Appendix A.15, Inter-Integrated Circuit (I2C) Registers, on page 1391.

25.1 Features

The I^2C interface supports the following features:

- Two I^2C ports
- S-Mode (up to 100 kHz SCL clock rate)
- General purpose I^2C master/slave
- Transaction level I^2C master activation
- EEPROM Serial initialization support

25.2 I^2C Bus Operation

The I^2C port consists of 2 open drain signals:

- SCL (Serial Clock)
- SDA (Serial address/data)

The I^2C master starts a transaction by driving a start condition followed by a 7- or 10-bit slave address and a read/write bit indication. The target I^2C slave responds with acknowledge.

In case of a write access (R/W bit is 0 following the I^2C slave acknowledge), the master drives eight bits of data, and the slave responds with acknowledge. This write access (8-bit data followed by acknowledge) continues until the I^2C master ends the transaction with a stop condition.

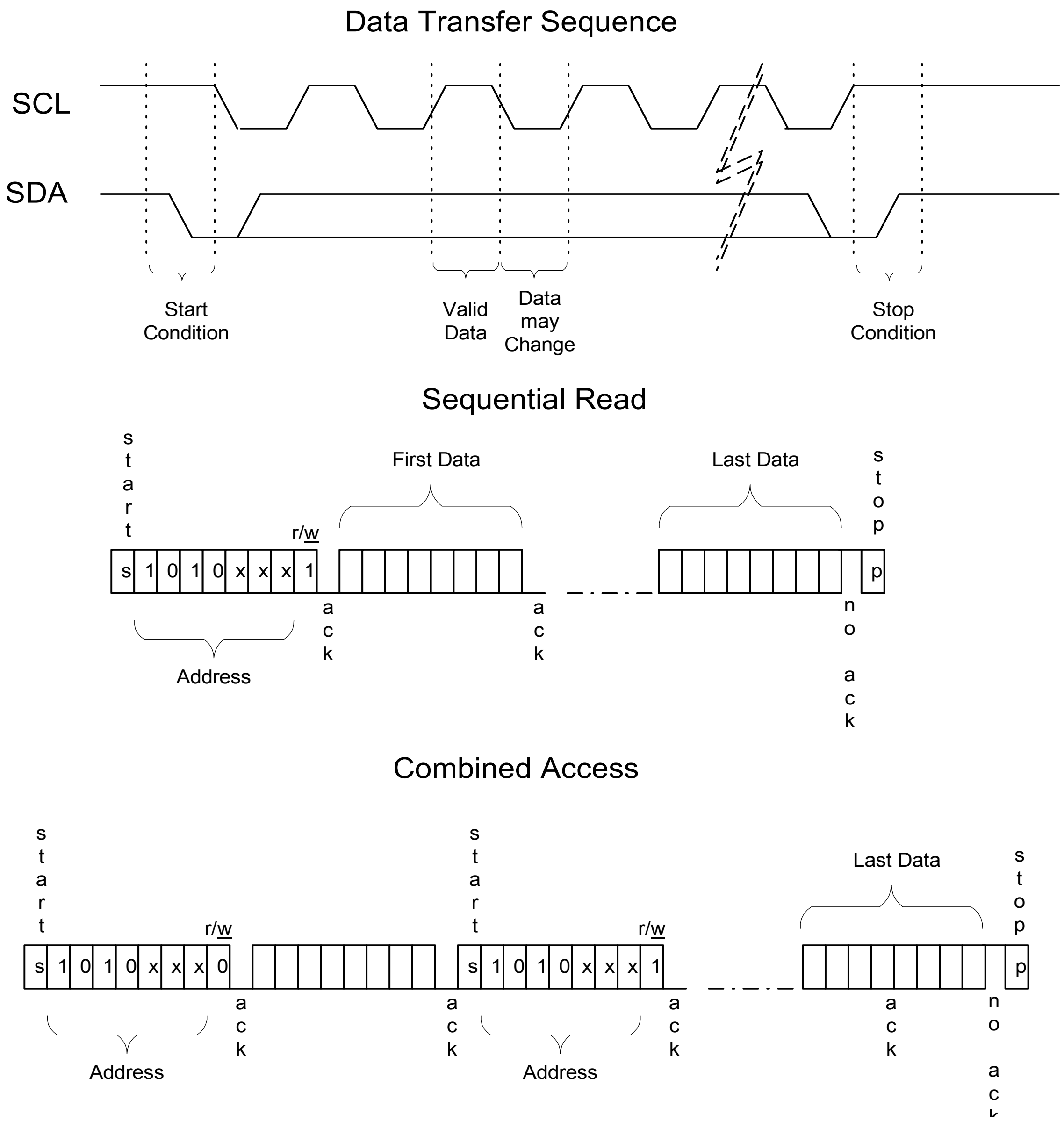
In case of a read access following the I^2C slave address acknowledge, the I^2C slave drives eight bits of data and the master responds with acknowledge. This read access (8-bit data followed by acknowledge) continues until the I^2C master ends the transaction by responding with no acknowledge to the last 8-bit data, followed by a stop condition.

A target slave that cannot drive valid read data right after it received the address, can insert “wait states” by forcing SCL low until it has valid data to drive on the SDA line.

A master is allowed to combine 2 transactions. After the last data transfer, it can drive a new start condition followed by new slave address, rather than drive stop condition. Combining transactions guarantees that the master does not lose arbitration to some other I^2C master.

I^{2}C examples are shown in Figure 109.

Figure 109:I^{2}C Examples



25.3 I^{2}C Functional Description

The port can act as master, generating read/write requests, and as a slave, responding to read/write requests from an external master. It can be used for various applications, and can control other I^{2}C on-board devices such as temp sensors, to read DIMM SPD ROM. It is also used for serial ROM initialization (see the Serial ROM Initialization section in the device *Hardware Specifications*).

The I^{2}C interface master and slave activities are handled by a simple CPU access to internal registers, plus an interrupt interface. The protocol is byte oriented— the core processor is required to handle each transmitted/received byte. The following sections describe I^{2}C registers and receive/transmit operation.

Note

The I^{2}C port is also used for Serial ROM initialization (if enabled). The CPU must not access I^{2}C port registers during the initialization period.

25.3.1 I^{2}C Slave Address Registers

The I^{2}C slave interface supports both 7-bit and 10-bit addressing. The slave address is programmed by the I2C Slave Address Register (t=0–1) (Table 1201 p. 1392) and I2C Extended Slave Address Register (t=0–1) (Table 1206 p. 1396).

When the I^{2}C receives a 7-bit address after a start condition, it compares that address against the value programmed in the Slave Address register. If the address matches, it responds with acknowledge.

If the received 7 address bits are ‘11110xx’, meaning that it is an 10-bit slave address, the I^{2}C compares the received 10-bit address with the 10-bit value programmed in the Slave Address and Extended Slave Address registers. If the address matches, it responds with acknowledge.

The I^{2}C interface also supports slave response to general call transactions. If the <GCE> field in the I2C Slave Address Register (t=0–1) (Table 1201 p. 1392) is set to 1, the I^{2}C also responds to general call address (0x0).

25.3.2 I^{2}C Data Register

The 8-bit data register is used both in master and slave modes.

In master mode, the core processor must place the slave address or write data to be transmitted. In case of read access, it contains received data (need to be read by the core processor).

In slave mode, the Data register contains data received from master on write access, or data to be transmitted (written by the core processor) on read access.

Note

The data register Most Significant bit (MSb) contains the first bit to be transmitted or being received.

25.3.3 I^{2}C Control Register

This 8-bit register contains the following bits:

Table 114: I^{2}C Control Register Bits

Bit	Function	Description
1:0	Reserved	Read only 0.
2	Acknowledge Bit	When set to 1, the I^{2}C drives an acknowledge bit on the bus in response to a received address (slave mode), or in response to a data received (read data in master mod, write data in slave mode). For a master to signal a I^{2}C target a read of last data, the core processor must clear this bit (generating no acknowledge bit on the bus). For the slave to respond, this bit must always be set back to 1.
3	Interrupt Flag	If any of the interrupt events occur, set to 1 by I^{2}C hardware If set to 1 and I^{2}C interrupts are enabled through bit[7], an interrupt is asserted.
4	Stop Bit	When set to 1, the I^{2}C master initiates a stop condition on the bus. The bit is set only. It is cleared by I^{2}C hardware after a stop condition is driven on the bus.
5	Start Bit	When set to 1, the I^{2}C master initiates a start condition on the bus, when the bus is free, or a repeated start condition, if the master already drives the bus. The bit is set only. It is cleared by I^{2}C hardware after a start condition is driven on the bus.
6	I^{2}C Enable	If set to 1, the I^{2}C slave responds to calls to its slave address, and to general calls if enabled. If cleared to 0, SDA and SCL inputs are ignored. The I^{2}C slave does not respond to any address on the bus.
7	Interrupt Enable	If set to 1, I^{2}C interrupts are enabled. Marvell recommends to use the I^{2}C interrupt to interface the I^{2}C module, rather than using the register polling method.
31:8	Reserved	Reserved

25.3.4 I^{2}C Status Register

This 8-bit register contains the current status of the I^{2}C interface. Bits[7:3] are the status code, bits[2:0] are Reserved (read only 0). Table 115 lists all possible status codes.

Table 115: I^{2}C Status Codes

Code	Status
0x00	Bus error. **NOTE:** A bus error occurs if the I^{2}C slave interface drives the bus (Data or Clock) when it should not. To recover from this error, set the <Stop> field in the I2C Control Register (t=0–1) (Table 1203 p. 1393) and clear the interrupt.
0x08	Start condition transmitted.
0x10	Repeated start condition transmitted.
0x18	Address + write bit transmitted, acknowledge received.

Copyright © 2014 Marvell

Table 115: I^2C Status Codes (Continued)

Code	Status
0x20	Address + write bit transmitted, acknowledge not received.
0x28	Master transmitted data byte, acknowledge received.
0x30	Master transmitted data byte, acknowledge not received.
0x38	Master lost arbitration during address or data transfer.
0x40	Address + read bit transmitted, acknowledge received.
0x48	Address + read bit transmitted, acknowledge not received.
0x50	Master received read data, acknowledge transmitted.
0x58	Master received read data, acknowledge not transmitted.
0x60	Slave received slave address, acknowledge transmitted.
0x68	Master lost arbitration during address transmit, address is targeted to the slave (write access), acknowledge transmitted.
0x70	General call received, acknowledge transmitted.
0x78	Master lost arbitration during address transmit, general call address received, acknowledge transmitted.
0x80	Slave received write data after receiving slave address, acknowledge transmitted.
0x88	Slave received write data after receiving slave address, acknowledge not transmitted.
0x90	Slave received write data after receiving general call, acknowledge transmitted.
0x98	Slave received write data after receiving general call, acknowledge not transmitted.
0xA0	Slave received stop or repeated start condition.
0xA8	Slave received address + read bit, acknowledge transmitted.
0xB0	Master lost arbitration during address transmit, address is targeted to the slave (read access), acknowledge transmitted.
0xB8	Slave transmitted read data, acknowledge received.
0xC0	Slave transmitted read data, acknowledge not received.
0xC8	Slave transmitted last read byte, acknowledge received.
0xD0	Second address + write bit transmitted, acknowledge received.
0xD8	Second address + write bit transmitted, acknowledge not received.
0xE0	Second address + read bit transmitted, acknowledge received.
0xE8	Second address + read bit transmitted, acknowledge not received.
0xF8	No relevant status. Interrupt flag is kept 0.

25.3.5 Baud Rate Register

The I²C module contains a clock divider to generate the SCL clock. Setting the <N> field and the <M> field in the I2C Baud Rate Register (t=0–1) (Table 1204 p. 1393) defines SCL frequency (F_{SCL}) as follows:

$$F_{SCL} = \frac{F_{TClk}}{10 \cdot (M+1) \cdot 2^{(N+1)}}$$

Note

Where M is the value represented by bits[6:3] and N is the value represented by bits[2:0]: If for example, M=4 and N=5 (which are the default values), running TCLK at 200 MHz (F_{TClk}) results in a SCL frequency of 62.5 kHz.

As defined in the I²C spec for S-Mode, the maximum supported SCL frequency is 100 kHz. Fast mode (where SCL frequency is 400 kHz) is not supported.

The Baud Rate register must be set properly, even when using the I²C port as a slave only. It should be set such that F_{SCL} will be in the range of x1 to x2 of the I²C bus frequency.

25.3.6 I²C Port Master Operation

Refer also for Figure 108 to illustrate the transaction flow. In that figure, SW delay represent a delay of at least 2 internal clock cycles (x10 SCL clock rate, at 100 kHz SCL, for example, the delay is 2 µs).

A master write access consists of the following steps:

1. The core processor sets the <Start> field in the I2C Control Register (t=0–1) (Table 1203 p. 1393) to 1. The I²C master then generates a start condition as soon as the bus is free, sets an Interrupt flag, and sets the Status register to 0x8.
2. The core processor writes a 7-bit address plus a write bit to the Data register and clears the Interrupt flag for the I²C master interface to drive the slave address on the bus. The target slave responds with acknowledge. This causes an Interrupt flag to be set and a status code of 0x18 is registered in the Status register.

   If the target I²C device has an 10-bit address, the core processor needs to write the remainder 8-bit address bits to the Data register. The core processor then clears the Interrupt flag for the master to drive this address on the bus. The target device responds with acknowledge, causing an Interrupt flag to be set and status code of 0xD0 be registered in the Status register.
3. The core processor writes a data byte to the Data register, and then clears the Interrupt flag for the I²C master interface to drive the data on the bus. The target slave responds with acknowledge, causing the Interrupt flag to be set, and status code of 0x28 be registered in the Status register. The core processor continues this loop of writing new data to the Data register and clearing the Interrupt flag as long as it needs to transmit write data to the target.
4. After the last data transmit, the core processor may terminate the transaction or restart a new transaction. To terminate the transaction, the core processor sets the Control Register <Stop> bit and then clears the Interrupt flag. This causes the I²C master to generate a stop condition on the bus and to return to idle state. To restart a new transaction, the core processor sets the <Start> field in the I2C Control Register (t=0–1) (Table 1203 p. 1393) and clears the Interrupt flag, thus causing the I²C master to generate a new start condition.

Copyright © 2014 Marvell

Note The above sequence describes a normal operation. There are also abnormal cases, such as a slave not responding with acknowledge or arbitration loss. Each of these cases is reported in the Status register and needs to be handled by the core processor.

A master read access consists of the following steps:

1. Generating start condition, exactly the same as in the case of write access (see Section 25.3.1, I^{2}C Slave Address Registers).
2. Drive 7- or 10-bit slave address, exactly the same as in the case of write access, with the exception that the status code after the first address byte transmit is 0x40, and after 2nd address byte transmit (in case of 10-bit address) is 0xE0.
3. Read data being received from the target device is placed in the data register and acknowledge is driven on the bus. Also interrupt flag is set, and status code of 0x50 is registered in the Status register. The core processor reads data from Data register and clears the Interrupt flag to continue receiving next read data byte. This loop is continued as long as the core processor wishes to read data from the target device.
4. To terminate, the read access needs to respond with no acknowledge to the last data. It then generates a stop condition or generates a new start condition to restart a new transaction. With last data, the core processor clears the <ACK> field in the I2C Control Register (t=0–1) (Table 1203 p. 1393) (when clearing the Interrupt bit), causing the I^{2}C master interface to respond with no acknowledge to last received read data. In this case, the Interrupt flag is set with status code of 0x58. Now, the core processor can issue a stop condition or a new start condition.

Note This sequence describes a normal operation. There are also abnormal cases, such as the slave not responding with acknowledge, or arbitration loss. Each of these cases is reported in the Status register and needs to be handled by core processor.

25.3.7 I^{2}C Port Slave Operation

The I^{2}C slave interface can respond to a read access, driving read data back to the master that initiated the transaction, or respond to write access, receiving write data from the master.

Upon detecting a new address driven on the bus with a read bit indication, the I^{2}C slave interface compares the address against the address programmed in the Slave Address register. If it matches, the slave responds with acknowledge. It also sets the Interrupt flag, and sets the status code to 0xA8.

Note If the I^{2}C slave address is 10-bit, the interrupt flag is set, and the status code changes only after receiving and identifying an address match on the second address byte.

The core processor now must write new read data to the Data register and clear the Interrupt flag, causing I^{2}C slave interface to drive the data on the bus. The master responds with acknowledge causing an Interrupt flag to be set, and status code of 0xB8 to be registered in the Status register.

If the master does not respond with acknowledge, the Interrupt flag is set, a status code 0f 0xC0 is registered, and the I^{2}C slave interface returns back to idle state.

If the master generates a stop condition after driving an acknowledge bit, the I^{2}C slave interface returns back to idle state.

Upon detecting a new address driven on the bus with write bit indication, the I^{2}C slave interface compares the address against the address programed in the Slave Address register. If the address matches, it responds with acknowledge.matches, responds with acknowledge. It also sets an Interrupt flag, and sets the status code to 0x60 (0x70 in case of general call address, if general call is enabled).

Following each write byte received, the I^{2}C slave interface responds with acknowledge, sets an Interrupt flag, and sets status code to 0x80 (0x90 in case of general call access). The core processor then reads the received data from Data register and clears Interrupt flag to allow transfer to continue.

If a stop condition or a start condition of a new access is detected after driving the acknowledge bit, an Interrupt flag is set and a status code of 0xA0 is registered.

25.3.8 I^{2}C Port Transaction Generator

To reduce software overhead each I^{2}C port can be activated in a transaction level mode. In this mode, all properties of an I^{2}C Master-Receive or an I^{2}C Master-Transmit are pre-configured. Following the transaction's properties configuration, the transaction generator is signaled to start. When the transaction is completed, an interrupt is asserted.

25.3.8.1 I^{2}C Transaction Generation: Master-Transmitter

The following sequence performs an I^{2}C Master-Transmitter transaction:

1. Set the <Wr> and clear the <Rd> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1398) in the same register.
2. Define the slave address length by writing (0x0 = 7-bit address or 0x1 = 10-bit address) to the <ExtendAddr> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1398).
3. Write the slave address of the peer I^{2}C to the <Addr> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1398).
4. Write the number of data bytes to be transmitted to the <TxSize> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1398).
5. Write the data to be transmitted to the <TxDataLow> field in the I2C Bridge Transmit Data Low Register (t=0–1) (Table 1209 p. 1396) and the <TxDataHigh> field in the I2C Bridge Transmit Data High Register (t=0–1) (Table 1210 p. 1397).
6. Set the <BridgeEn> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1397) to start the transaction.
7. When the transaction is completed an I^{2}C interrupt is set.
8. Read the <Error> field in the I2C Bridge Status Register (t=0–1) (Table 1214 p. 1399) to validate that transmission has been completed successfully.

25.3.8.2 I^{2}C Transaction Generation: Master-Receiver

The following sequence performs an I^{2}C Master-Receiver transaction:

1. Set the <Rd> field and clear the <Wr> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1398).
2. Define the slave address length by writing to the <ExtendAddr> (0x0 = 7-bit address or 0x1 = 10-bit address).
3. Write to the <Addr> field in the slave address of the peer I^{2}C.
4. Write the number of data bytes to be received in the <RxSize> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1398).
5. Set the <BridgeEn> field to start the transaction.

Copyright © 2014 Marvell

6. When the transaction is completed an I^2C interrupt is set.
7. Read the <Error> to validate that data has been received successfully.
8. Read the requested data from the <RxDataLow> field in the I2C Bridge Receive Data Low Register (t=0–1) (Table 1211 p. 1397) and the <RxDataHigh> field in the I2C Bridge Receive Data High Register (t=0–1) (Table 1212 p. 1397).

25.3.8.3 I^2C Transaction Generation: Master-Transmitter Followed by a Master-Receiver

The following sequence performs an I^2C Master-Transmitter transaction that is followed by a Master-Receiver transaction:

1. Set both the <Rd> field and the <Wr> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1398).
2. Define the delimiter between the Master-Transmitter phase and the Master-Receiver phase by writing the required type of Start transmission (0 = Stop followed by Start or 1 = Repeated Start) to the <RepeatedStart> field in the I2C Bridge Control Register (t=0–1) (Table 1213 p. 1397).
3. Define the slave address length by writing to the <ExtendAddr> field (0x0 = 7-bit address or 0x1 = 10-bit address).
4. Write the slave address of the peer I^2C to the <Addr> field.
5. Write the number of data bytes to be transmitted during the Master-Transmitter phase to the <TxSize>.
6. Write the data to be transmitted at the Master-Transmitter phase to the <TxDataLow> field in the I2C Bridge Transmit Data Low Register (t=0–1) (Table 1209 p. 1396) and the <TxDataHigh> field in the I2C Bridge Transmit Data High Register (t=0–1) (Table 1210 p. 1397).
7. Write the number of data bytes to be received during the Master-Receiver phase to the <RxSize>.
8. Set the <BridgeEn> field to start the transaction.
9. When the transaction is completed an I^2C interrupt is set.
10. Read the <Error> field in the I2C Bridge Status Register (t=0–1) (Table 1214 p. 1399) to validate that both phases have completed successfully.
11. Read the requested data from the <RxDataLow> field in the I2C Bridge Receive Data Low Register (t=0–1) (Table 1211 p. 1397) and the **<RxDataHigh>** field in the I2C Bridge Receive Data High Register (t=0–1) (Table 1212 p. 1397).

25.3.8.4 I^2C Transaction Generation Example: Reading from an EPROM

To demonstrate usage of a master-transmitter followed by a master-receiver, the following sequence is an example that performs a read from a standard I^2C EPROM:

1. <Rd>=1, <Wr>=1.
2. <RepeatedStart>=1.
3. <ExtendAddr>=0.
4. <TxSize>=2.
5. <TxDataLow> and <TxDataHigh>=The byte address in the EPROM.
6. <RxSize>=8.
7. <Addr>=0x50 // 7 bit slave address of the EPROM.
8. Set the <BridgeEn> field to start the transaction.
9. When the transaction is completed an I^2C interrupt is set.
10. Read the <Error> field in the I2C Bridge Status Register (t=0–1) (Table 1214 p. 1399) to validate that both phases have completed successfully.
11. Read the 8 bytes from the <RxDataLow> and <RxDataHigh> fields.

25.3.8.5 I²C Transaction Generation Timing Gaps

Timing gaps with in an auto generated transaction can be configured using the I2C Bridge Timing gaps Register (t=0–1) (Table 1217 p. 1400).

25.4 I²C Serial ROM Initialization

I²C port0 can be set at reset to perform serial ROM initialization (act as a I²C master, reading data from external I²C ROM, and writing this data to the chip registers). For further details about serial ROM initialization, refer to the Reset section in the device Hardware Specifications.

25.5 I²C Hardware Slave

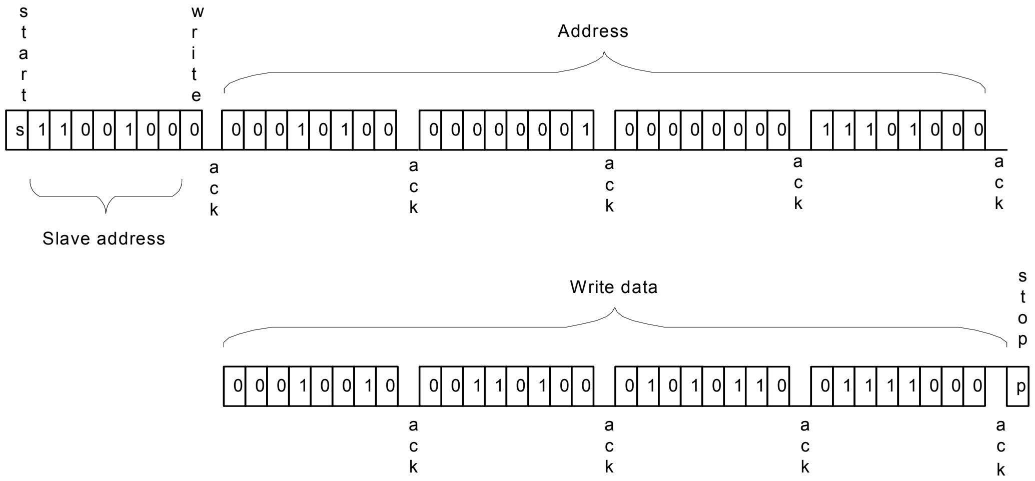
I²C Port1 can be set at reset to act as a slave hardware port.

The hardware slave is a an internal DMA that can receive read and write transactions from the I²C bus, and issues them as internal requests within the device. It then completes the transaction over the I²C port. These operations do not require software intervention.

As a hardware slave port, I²C1 corresponds to a fixed 7-bit address of 'b1100100.

A word write example is shown in Figure 110.

Figure 110:I²C Debug Port Write Example

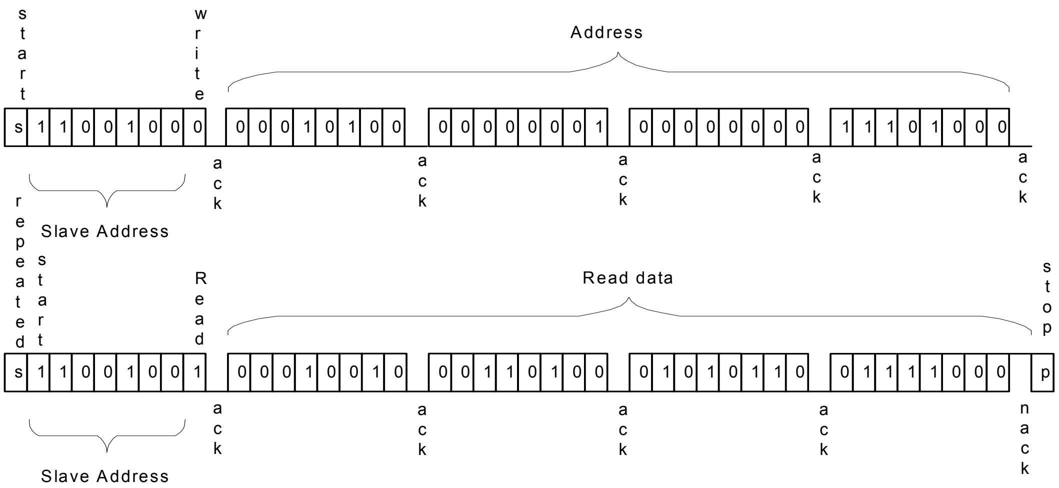


The write sequence is as follows:

1. The external master drives the start bit followed by the slave address (7'b1100100) with the read/write bit cleared to 0 (write). The device responds with acknowledge (ACK).
2. The external master drives 4 bytes that represent the 32-bit address within the device address space (0x1401.00E8 in the above example). The device responds with an acknowledgement to each byte.
3. The external master drives 4 bytes of data (0x1234.5678 in the above example). The device responds with ACK to all 4 bytes.
4. The external master drives a stop bit.
5. The device I²C hardware slave writes the 32-bit data to the location pointed to by the 32-bit address.

A word read example is shown in Figure 111

Figure 111:I^{2}C Debug Port Read Example



The read sequence is as follows:

1. The external master drives the start bit followed by the slave address (7’b1100100) with read/write bit cleared to 0 (write). The device responds with acknowledge (ack).
2. The external master drives 4 bytes that represent a 32-bit address within the device address space (0x1401.00E8 in the above example). The device responds with ACK to each byte.
3. The external master drives a new start bit followed by slave address (7’b1100100) with read/write bit set to 1 (read). The device responds with acknowledge.
4. The device I^{2}C hardware slave controller reads 32-bit of data from the location pointed by the 32-bit address.
5. The device I^{2}C hardware slave controller drives the 4 bytes received from the internal register over the data of the master-receive transaction (0x1234.5678 in the above example). The external master responds with ACK to the first 3 bytes and NACK to the last byte.
6. The external master drives a stop bit.

This sequence is a combined sequence. The external master drives the start bit of the read access right after the acknowledge bit of last write access without a STOP in between. I^{2}C hardware slave also tolerates a non-combined access, in which the external master generates a stop bit at the end of the write access, and follows it by issuing the read access.

26 Universal Asynchronous Receiver/Transmitter (UART) Interface

The device integrates 4 Universal Asynchronous Receiver/Transmitter (UART) ports.

Each Universal Asynchronous Receiver/Transmitter (UART) port performs the following data conversions:

- Serial-to-parallel conversion on data characters received from a peripheral device or a modem
- Parallel-to-serial conversion on data characters received from the device

The device can read the complete UART status from the Line Status (LSR) Register (n=0–3) (Table 1229 p. 1409). Status information includes the type and condition of transfer operations and error conditions (parity, overrun, framing, or break interrupt) associated with the UART.

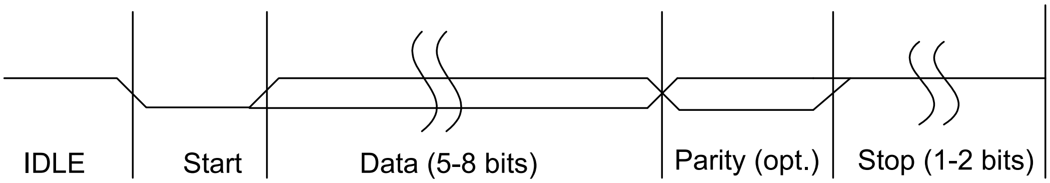
Each serial port operates in either FIFO or Standard mode. In addition, for transmit operations, the UART can operate in DMA-Based mode.

- In FIFO mode, a 16-byte transmit FIFO buffer holds data from the device until it is transmitted on the serial link; a 16-byte FIFO buffer receives data from the serial link until it is read by the device.
- In Standard mode, an 8-bit transmit buffer holds data from the device until it is transmitted on the serial link; an 8-bit buffer receives data from the serial link until it is read by the device.
- In DMA-Based mode, the IDMA buffer is used to transmit data (see Section 26.4, DMA-Based UART, on page 436).

The device supports the UART interface through the UA0/1/2/3_TXD and UA0/1/2/3_RXD pins and provides modem control functions through the UA0/1/2/3_CTS and UA0/1_RTS pins.

Figure 112 shows a UART character format on the serial line.

Figure 112:UART Character Format



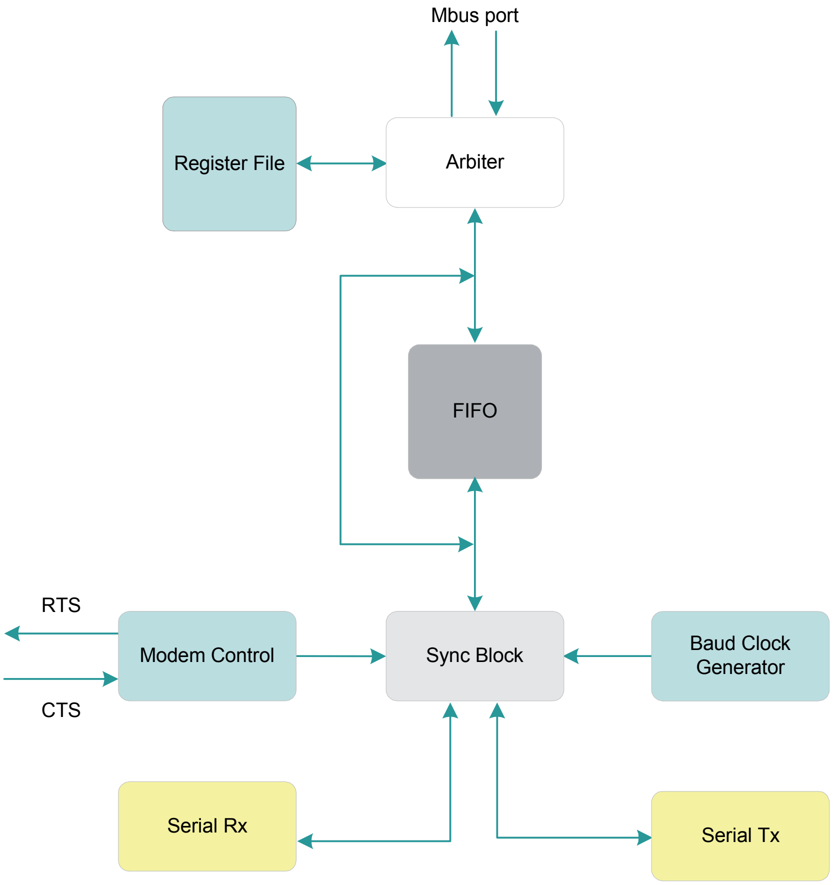
The UART is 16750 compatible with busy functionality. For details about this functionality, see the UART registers.

The UART registers are located in Appendix A.16, UART Registers, on page 1401.

Copyright © 2014 Marvell

Figure 113 presents the main blocks of the UART controller.

Figure 113:UART Controller Block Diagram



26.1 Features

The UART interface incorporates the following features:

- Ability to add or delete standard asynchronous communications bits (start, stop, and parity) in the serial data
- Independently controlled transmit, receive, line-status, and data-set interrupts
- Programmable baud-rate generator (see Section 26.3, Programmable Baud-Rate Generator)
- Modem flow control functions (CTS and RTS).
- Programmable serial interface:
  - 5-, 6-, 7- or 8-bit characters
  - Even, odd, or no parity detection
  - 1 or 2 stop-bit generation
- 16-byte transmit FIFO
- 16-byte receive FIFO
- Complete status-reporting capability
- Ability to generate and detect line breaks

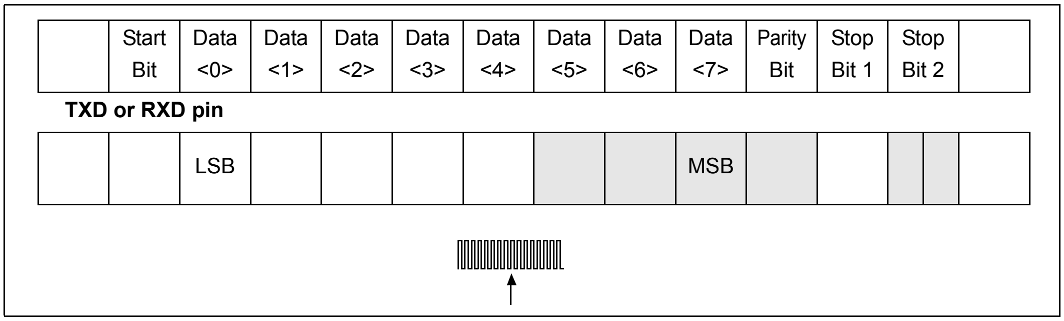
Copyright © 2014 Marvell

- Internal diagnostic capabilities that include:
  - Loopback controls for communication link fault isolation
  - DMA-based UART
  - Break, parity, and framing-error simulation
- Fully prioritized interrupt system controls

26.2 Functional Description

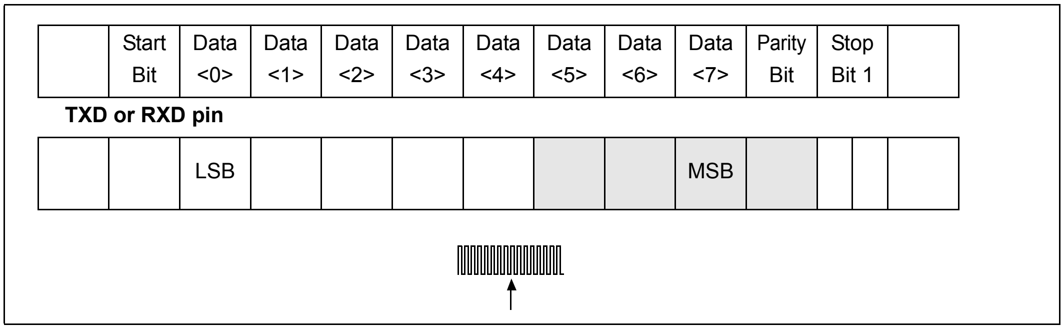
Figure 114 and Figure 115 shows the format of a UART data frame, respectively, with two stop bits and with one stop bit.

Figure 114:UART Character (Two Stop Bits Example)



To ensure bus stability, the receiver samples the serial input data at approximately the mid-point of the bit sequence, once the start bit has been detected.

Figure 115:UART Character (One Stop Bit Example)



The receive-data sample-counter frequency is 16 times the value of the bit frequency. The 16x clock is created by the baud-rate generator. Each bit is sampled three times in the middle of the bit sequence. Shaded bits are optional and can be programmed by software.

The data frame is between 7 and 12 bits long, depending on the size of the data programmed, parity status (enabled/disabled), and the number of stop bits. A data frame begins by transmitting a start bit that is represented by a high to low transition. The start bit is followed by five to eight bits of data that begin with the least significant bit (LSB).

The data bits are followed by an optional parity bit. The parity bit is set if even parity is enabled and the data byte has an odd number of ones, or if odd parity is enabled and the data byte has an even number of ones. The data frame ends with one or two stop bits, as programmed by software. The stop bits are represented by one or two successive bit periods of logic one.

Each UART port has a transmit and a receive FIFO.

- The transmit FIFO is 16 bytes deep.
- The receive FIFO is 16 bytes deep.

26.3 Programmable Baud-Rate Generator

Each UART port includes a programmable baud-rate generator that can take a fixed-input clock and divide it to generate the preferred baud rate. The baud rate is calculated by taking the TCLK frequency and dividing it by a value between 1 and (2^{16} - 1), to produce a 16x clock that is used to drive the internal transmit and receive logic. Each UART operates in an environment that is either controlled by the software and can be polled, or is interrupt driven.

The baud-rate generator output frequency is 16 times the baud rate. The <DivLatchLow> field in the Divisor Latch Low (DLL) Register (n=0–3) (Table 1221 p. 1403) and the <DivLatchHigh> field in the Divisor Latch High (DLH) Register (n=0–3) (Table 1223 p. 1404) make up the two 8-bit divisor latch fields that store the divisor in a 16-bit binary format. Load these divisor latches during initialization to ensure that the baud-rate generator operates properly. The 16x clock stops if both fields are loaded with 0x0.

The baud rate of the data shifted into or out of a UART is given by the formula:

$$\mathbf{BaudRate}=\frac{\text{TCLK freq}}{(16\text{x}\mathbf{Divisor})}$$

Table 116: Typical Baud Rates, TCLK = 200 MHz

Required Baud	Divisor (decimal)	Actual Baud	Error (%)
1200	10,417	1200	0
2400	5208	2400	0
4800	2604	4800	0
9600	1302	9600	0
19,200	651	19,200	0
38,400	326	38,344	0.1
57,600	217	57,604	0.1
76,800	163	76,687	0.1
115,200	109	114,679	0.5
230,400	54	231,481	0.5
460,800	27	462,963	0.5
500,000	25	500,000	0
921,600	14	892,857	3.1
1,000,000	13	961,538	3.8
1,382,400	9	1,388,889	0.5

Table 116: Typical Baud Rates, TCLK = 200 MHz (Continued)

Required Baud	Divisor (decimal)	Actual Baud	Error (%)
1,500,000	8	1,562,500	4.2

Table 117: Typical Baud Rates, TCLK = 250 MHz

Required Baud	Divisor (decimal)	Actual Baud	Error (%)
1200	13,021	1200	0
2400	6510	2400	0
4800	3255	4800	0
9600	1628	9600	0
19,200	814	19,200	0
38,400	407	38,400	0
57,600	271	57,657	0.1
76,800	203	76,970	0.2
115,200	136	114,890	0.27
230,400	68	229,779	0.27
460,800	34	459,559	0.27
500,000	31	504,032	0.81
921,600	17	919,118	0.27

26.4 DMA-Based UART

To enable transfer of large amounts of data, the device supports DMA-based UART transmission. This mode is necessary when using the UART to interface devices that constantly transfer large data buffers.

When this mode is activated, the device IDMA channel 0 is used to transfer a data buffer from memory to the UART. The CPU is interrupted only when the entire IDMA buffer has been transmitted.

Note

- If the DMA-Based UART mode is used, IDMA channel 0 cannot be used for other tasks.
- The DMA-Based UART mode only applies to UART Transmit operation. The Receive operation still operates in either Standard mode or FIFO mode.

Each of the device UART ports can be configured to support DMA-based transmit mode. To set a UART port to DMA-based operation, set the <External_Writes_Enable> field in the UART External Control Register (Table 1219 p. 1402).

1. To activate DMA-based UART transmit, configure IDMA channel 0 as follows:
   - Set the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1492):

Copyright © 2014 Marvell

    - Set <DMAReqDir> bit[15] to 1. The DMAReqn is generated by the destination.
    - Set the <DestHold> bit[5] to 1. The destination address holds the same value, and is not incremented.
    - Set the <ChainMode> bit[9] to 1. Non-chain mode is enabled.
    - Set the <DMAReqMode> bit[16] to 1. Edge sensitive DMA requests are triggered.
    - Clear the <DstBurstLimit> bits[2:0]. The destination byte count is set to 8 bytes.
    - Set the <DAddrOvr> bits[24:23] to 1 for destination address override.
  - Set the IDMA address decode (Base/Size) registers (see Appendix A.23, IDMA Registers, on page 1489) to map the SDRAM as the source address and the UART as the destination address:
    - Set the Base Address register BARx and Size register SRx to map the SDRAM space in which the data buffer to be transmitted is placed.
    - Set the <Target> bits[3:0] to 1 and <Attr> bits[15:8] to 1 of BAR1, BAR2, or BAR3. These settings configure the Base address register to map the BAR to the UART port.
  - Set IDMA descriptor:
    - Set the <ByteCnt> field in the Channel IDMA Byte Count Register (n=0–3) (Table 1406 p. 1497) to the buffer size.
    - Set the <SrcAdd> field in the Channel IDMA Source Address Register (n=0–3) (Table 1407 p. 1497) to the IDMA source address.
    - In the Channel IDMA Destination Address Register (n=0–3) (Table 1408 p. 1497), set the relevant <DestAdd> bit to an address that matches the BAR assigned to the UART unit, and set the <DestAdd> bits[9:8] to one of the UART ports to which the data will be transferred. Use one of the following values to specify the UART port:
      - 0x0 = Port0
      - 0x1 = Port1
      - 0x2 = Port2
      - 0x3 = Port3

2. Set the following UART configurations:
   - In the UART External Control Register (Table 1219 p. 1402), set one of the <DMAMode> bits[11:8] to 1 based on the UART port that is configured to DMA mode (the other 3 bits should be cleared to 0).
     - bit[8] = UART0
     - bit[9] = UART1
     - bit[10] = UART2
     - bit[11] = UART3
   - Set the <AFCE> field and the <RTS> field in the Modem Control (MCR) Register (n=0–3) (Table 1228 p. 1409) to 1.
   - Enable the FIFO mode by setting the <FIFOEn> field in the FIFO Control (FCR) Register (n=0–3) (Table 1225 p. 1406) to 1.
3. To start a DMA transfer, set the <ChanEn> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1494) to 1. The IDMA starts fetching buffer data from memory, and pushes the data towards the selected UART port. The IDMA fetches up to 128B from DRAM for each transaction (according to DMA <SrcBurstLimit> setting) and splits it to multiple 8B writes towards the UART. These bytes are pushed to the UART Tx FIFO and driven on the line.

The DMA is working on demand. Whenever the UART FIFO becomes full, the DMA stops pushing data. Also, if the external UART device uses transmit flow control (CTS), the UART (and DMA) acts accordingly and stops sending data.

When the entire data buffer transmit is finished, the IDMA completion interrupt is asserted in the Main Interrupt Cause register.

Note The IDMA <Comp> field in the Interrupt Cause Register (Table 1411 p. 1499) completion bit is asserted when the last byte is placed in the UART transmit FIFO. It does not reflect actual transmission. To detect actual transmission of last bit, read the <TxEmpty> field in the Line Status (LSR) Register (n=0–3) (Table 1229 p. 1409).

27 Real-Time Clock (RTC)

This section describes the device's real-time clock (RTC).

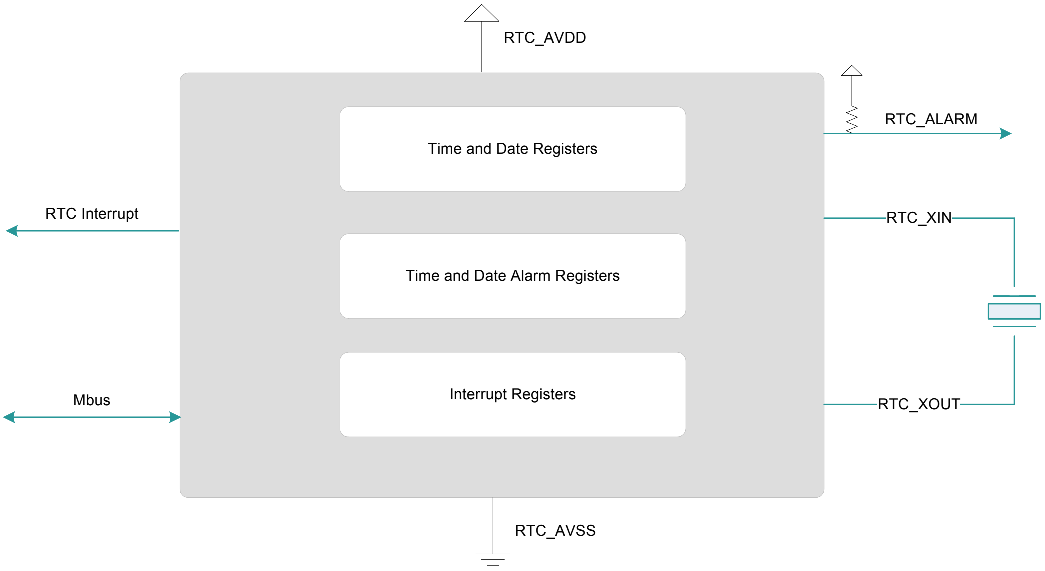
The real-time clock (RTC)/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap years up to the end-of-year 2099. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator.

An internal alarm provides support for a real time periodic interrupt.

There is also an external alarm that provides support for a minimal power consumption mode, which triggers an alarm event on the device interface.

The RTC unit operates with an external 32.768 kHz crystal.

Figure 116:RTC Block Diagram



The RTC registers are located in Appendix 27, Real-Time Clock (RTC), on page 439.

27.1 Features

RTC features include:

- Real-time fields: second, minute, hour, date, day, month, and year
- Leap-year compensation
- Independent power pin (RTC_AVDD) for power by battery
- Total current consumption for typical case: 3 uA
- Internal alarm time fields: second, minute, hour, date, month, and year
- External Alarm indication can be driven to an external host over an open drain pad

Copyright © 2014 Marvell

- Real-time field setting by CPU
- Valid until the end of the century (i.e., 23:59:59, December 31, 2099)

27.2 Functional Description

To set the RTC date and time, write the desired time and date to the respective time and date registers:

- RTC Time Register (Table 1236 p. 1414)
- RTC Date Register (Table 1237 p. 1415)

The values written to the fields in these registers are in BCD (Binary-Coded Decimal) format.

RTC date and time are known by reading the RTC Time Register and RTC Date Register.

27.2.1 Setting the Time

To configure the day of the week and the time:

1. Set the <HourFomat> to indicate if the time value is a 24-hour or a 12-hour presentation.
   - When this field is set to a 12-hour representation, set bit [21] of the <Hour> field to indicate whether the time is AM or PM, and set bits [20:16] to set the hour.
   - When this field is set to a 24-hour representation, set the <Hour> field to indicate the hour.
2. Set the <Minute> field to indicate the minutes within the hour.
3. Set the <Second> field to indicate the seconds within the minutes.

All the above fields are located in the same register, and can be set using a single write transaction.

27.2.2 Setting the Date

To set the date:

1. Set the <Day> field in the RTC Date Register (Table 1237 p. 1416) to indicate the day of the month.
2. Set the <Month> field to indicate the month.
3. Set the <Year> field to indicate the year.

All the above fields are located in the same register, and can be set using a single write transaction.

27.2.3 Setting Both the Time and Date

When setting both the time and the date, Marvell® recommends setting the RTC Time Register prior to setting the RTC Date Register. This avoids the occurrence of a new date increment due to an existing (invalid) time, which may occur if the recommended order is not followed. For example, if the updated date is set first, while the current time is set to 23:59:59, when the time is updated (1 second or more later), the date advances again when the clock reaches 0:0:0, resulting in a date advance of one additional day.

27.2.4 RTC Alarm Operation

The RTC block includes 2 alarm modes: an Internal mode and an External mode.

Internal mode The interrupt generated by the RTC logic, as a result of the alarm counter expiration, is only routed to the local host via the assertion of the <AlarmInterrupt> field in the RTC Interrupt Cause Register (Table 1241 p. 1418).

External mode The interrupt is routed to the local host via the assertion of the <BatteryAlarmInterruptEnable> field in the RTC Interrupt Mask Register (Table 1240 p. 1417), and also to an external host or board logic. The alarm logic runs from the battery domain and can be used as a wake-up event for the entire board (for example, to wake-up the system on a specific day and time). The external interrupt is driven over a dedicated open drain pin.

The RTC internal alarm is not powered by a battery power pin. The internal alarm can only be used when the device power is on (see Section 27.2.4, RTC Alarm Operation, on page 441).

The following sections describe the configuration procedure of both modes.

27.2.4.1 Internal Alarm Mode

To set the RTC internal alarm date and time, write the desired alarm time and date and alarm valid data to the respective alarm time and date registers:

- RTC Alarm Time Configuration Register (Table 1238 p. 1416)
- RTC Alarm Date Configuration Register (Table 1239 p. 1416)

The values written to the fields in these registers are in BCD (Binary-Coded Decimal) format.

RTC alarm date and time are known by reading the RTC Alarm Time Configuration Register and RTC Alarm Date Configuration Register.

These two registers contain the six alarm fields Real-time Clock (RTC), corresponding to the six data and time fields (day in week is excluded). For each such field, a valid bit specifies whether an alarm event does or does not include the matching between the alarm field and the corresponding real-time field.

The match is tested on a once-per-second update of the time and date registers.

Upon a tested match the <AlarmInterrupt> field in the RTC Interrupt Cause Register (Table 1241 p. 1418) is set resulting in an RTC Alarm interrupt.

To clarify the Alarm Interrupt valid bit usage, Table 118 provides a number of examples of configurations and the corresponding alarm behavior.

Table 118: Alarm Interrupt Valid Bit Usage

RTC Alarm Date Configuration Register			RTC Alarm Time Configuration Register			Alarm Rate
Alarm Year Valid	Alarm Month Valid	Alarm Day Valid	Alarm Hour Valid	Alarm Minute Valid	Alarm Second Valid	
0	0	0	0	0	0	Alarm once per second.
0	0	0	0	0	1	Alarm once per minute, when seconds match.
0	0	0	0	1	1	Alarm once per hour, when minutes and seconds match.

Table 118: Alarm Interrupt Valid Bit Usage (Continued)

RTC Alarm Date Configuration Register			RTC Alarm Time Configuration Register			Alarm Rate
Alarm Year Valid	Alarm Month Valid	Alarm Day Valid	Alarm Hour Valid	Alarm Minute Valid	Alarm Second Valid	
0	0	0	1	1	1	Alarm once per day, when hour, minute, and seconds match.
1	1	1	0	0	0	Alarm once per second, in a specific date.
1	1	1	1	0	0	Alarm every second in a specified date and hour.
1	1	1	1	1	0	Alarm every second in a specified date, hour, and minute.
1	1	1	1	1	1	Alarm once, when all fields match.

The alarm registers are initialized at reset, with all Alarm Valid bits being in *not set* state. As shown in the first row of Table 118, this mode will set the alarm upon every second. Without setting the RTC Alarm Time Configuration Register and RTC Alarm Date Configuration Register, the user should not expect a deterministic value at the <AlarmInterrupt> field, since its value depends on the reset de-assertion and the register read of a real-time-second change.

To set the alarm registers, the following sequence must be applied:

1. Disable interrupts by writing 0 to the <AlarmInterruptEnable> field in the RTC Interrupt Mask Register (Table 1240 p. 1418).
2. Set the fields in the RTC Alarm Time Configuration Register (Table 1238 p. 1416) and RTC Alarm Date Configuration Register (Table 1239 p. 1416).
3. Clear the <AlarmInterrupt> field.
4. Enable the interrupt by writing 1 to <AlarmInterruptEnable> field.

27.2.4.2 External Alarm Mode

To set the RTC Alarm date and time for external alarm mode, write the desired alarm time and date and alarm valid data to the respective alarm time and date register:

- Enable alarm programming by setting the <AlarmClr> field in the RTC External Alarm Control Register (Table 1242 p. 1418).
- Configure the desired time and date in RTC External Alarm Config Register (Table 1243 p. 1419). There are two options to calculate the value needed in this field:
  - A real time calculation: For a precise formula of the calculation see Section 27.2.4.3, External Alarm Real Time Calculation, on page 443.
  - A reference time calculation: Read the RTC Count Status Register (Table 1245 p. 1419) and add the number of seconds from current time that will be counted before the alarm is triggered. For example, to configure an external alarm to an hour from now, read the RTC Count Status Register register and write the read value + 3600 into RTC External Alarm Config Register.
- Load the alarm configuration by setting the RTC External Alarm Control Register to 1, followed by a clear to 0.
- Disable alarm programming by clearing the RTC External Alarm Config Register to 0.

Upon an expiration of the <ExternalAlarmStatus> field in the RTC External Alarm Status Register (Table 1244 p. 1419), a maskable interrupt is set internally and externally:

- Internally: The <BatteryAlarmInterrupt> field in the RTC Interrupt Cause Register (Table 1241 p. 1418) is set and the <AlarmStatus> field in the RTC External Alarm Control Register

Copyright © 2014 Marvell

(Table 1242 p. 1418) reflects the interrupt assertion.

- Externally: The RTC_ALARM signal is asserted.

To clear an RTC alarm external interrupt, software must assert the <AlarmClr> field in the RTC External Alarm Control Register (Table 1242 p. 1418) for a duration of at least 1 second after the alarm interrupt was logged, before resetting it back to 0 to proceed with normal operation.

27.2.4.3 External Alarm Real Time Calculation

For a real time YY/MM/DD Hr:Minute:Sec that is intended to be the external alarm time, the corresponding value must be written to the RTC External Alarm Config Register:

```
YY*31536000 +
ceil(YY/4)*86400 +
NSM(MM) +
((YY%4==0) * (MM>2) * 86400) +
(DD-1)*86400 +
(Hr*3600) +
(Min*60) +
Sec
```

NSM is defined according to the following table:

Table 119: External Alarm Real Time Calculation NSM Definition

Month	NSM
1	0
2	2678400
3	5097600
4	7776000
5	10368000
6	13046400
7	15638400
8	18316800
9	20995200
10	23587200
11	26265600
12	28857600

28 General Purpose Input/Output Ports

MV78260 **MV78460**	Contain a 67-bit General Purpose Input/Output (GPIO) port, multiplexed on MPP pins.
MV78230	Contains a 49-bit General Purpose Input/Output (GPIO) port, multiplexed on MPP pins.

The GPIO registers are located in Appendix A.18, General Purpose Input/Output (GPIO) Registers, on page 1420.

28.1 Features

The GPIO interface includes the following features:

- GPIO pin can be set as input or output.
- GPIO inputs are latched and can also be used to register and generate maskable interrupts from external devices.
- GPIO outputs can be configured to the desired driving value (High or Low) and can follow an open drain functionality.
- As GPIO outputs, these pins can be used for driving LEDs, with programmable blinking rates and durations.

28.2 Functional Description

Some of the unused device pins can be used as General Purpose IOs (GPIOs) interface. Each of the GPIO pins can be assigned to act as a general purpose input or output pin and can be used to register external interrupts (when assigned as an input pin). The usage model of input, output and the value can be configured via configuration registers mentioned in the following section.

28.3 GPIO Control Registers

The GPIO Data In register samples the data driven on the GPIO input pins. Data sampled into this register is the data sampled on the pins, or the inverted data, configured via the GPIO_<32*n>_<32*n+31>_Data In Polarity Register (n=0–1) (Table 1250 p. 1423) and GPIO_64_66_Data In Polarity Register (Table 1263 p. 1426) settings.

Each GPIO can also act as an output. Setting the GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1) (Table 1248 p. 1422) and the GPIO_64_66_Data Out Enable Control Register (Table 1261 p. 1425), enables driving data on the corresponding GPIO output pin. The data driven on the pin is the data configured into the GPIO_<32*n>_<32*n+31>_DataOut Register (n=0–1) (Table 1247 p. 1422) and the GPIO_64_66_DataOut Register (Table 1260 p. 1425).

The combination of the GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1) and GPIO_<32*n>_<32*n+31>_DataOut Register (n=0–1), GPIO_64_66_Data Out Enable Control Register and GPIO_64_66_DataOut Register, also allows open drain/open source output implementation. For example, to implement an open drain 0 output, reset the GPIO_<32*n>_<32*n+31>_DataOut Register (n=0–1) field to 0, and toggle the output by setting/clearing the corresponding GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1) field.

The device’s GPIO pins can also be used for driving external LEDs. When the GPIO_<32*n>_<32*n+31>_Blink Enable Register (n=0–1) (Table 1249 p. 1422) and GPIO_64_66_Blink Enable Register (Table 1262 p. 1425) are set, and the corresponding bit in GPIO Data Out Enable is enabled, the GPIO pin blinks.

There are two global sets (set A and set B) of configurable blinking parameters:

- ON duration, in terms of core clock cycles
- OFF duration, in terms of core clock cycles

Each GPIO can be configure to blink according to set A or set B.

28.4 GPIO Interrupts

When configured to act as GPIO inputs, GPIOs can be used for registration of external devices interrupts into the CPU interrupt controller. The device supports both edge and level sensitive external interrupts.

When the external device uses edge sensitive interrupts, each toggle of a GPIO Data In Register bit from 0 to 1 is registered in the GPIO Interrupt Cause register (see below). If the interrupt is not masked by the GPIO Interrupt Mask register, the interrupt is routed to the CPU Main Interrupt Cause register.

GPIO Data In Registers

- GPIO_<32*n>_<32*n+31>_Data In Register (n=0–1)
- GPIO_64_66_Data In Register
- GPIO_<32*n>_<32*n+31>_Data In Polarity Register (n=0–1)
- GPIO_64_66_Data In Polarity Register

GPIO Interrupt Cause Registers

- GPIO_<32*n>_<32*n+31>_Interrupt Cause Register (n=0–1)
- GPIO_64_66_Interrupt Cause Register

To clear an edge sensitive interrupt, the software must clear the corresponding bit in the relevant GPIO Interrupt Cause register (see above).

If the external device uses level sensitive interrupts, the data is registered in the GPIO Data In register. If not masked by the GPIO Interrupt Level Mask register (GPIO_<32*n>_<32*n+31>_Interrupt Level Mask Register (n=0–1) or GPIO_64_66_Interrupt Level Mask Register), the interrupt is routed to the CPU Main Interrupt Cause register.

Note

To clear a level sensitive interrupt, the software clears the interrupt directly on the external device.

28.5 Multi-CPU Support

Note

The following information only applies to the MV78460, MV78230, and the MV78260 devices.

In some multi-CPU applications, GPIO interrupts can be dynamically routed to different CPUs. To support this, the device has a separate masking option of GPIO interrupts per each CPU. Each Mask register controls which of the GPIO interrupts will be routed to the corresponding CPU. The software can dynamically change these masking settings, to change GPIO interrupts routing. For details see Section 9, Multiprocessor Interrupt Controller (MPIC), on page 139.

Also, GPIO outputs can be controlled by multiple CPUs. For one CPU not to interfere with the operation of another CPU, the device also implements, in addition to the GPIO_<32*n>_<32*n+31>_DataOut Register (n=0–1) (Table 1247 p. 1422) and the GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1) (Table 1248 p. 1422), the Set and Clear registers (see Appendix A.18, General Purpose Input/Output (GPIO) Registers, on page 1420).

For example, if CPU0 needs to set GPIO[0] output to 1, it would typically write 0x1 to the GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1) (Table 1248 p. 1422). If CPU1 needs to set GPIO[1] output to 1, it would typically write 0x2 to GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1) (Table 1248 p. 1422). These two writes contradict each other (CPU1 write 0x2 overrides CPU0 setting of GPIO[0]). The device provides a simple hardware mechanism to overcome this race.

Instead of writing to the GPIO Data Out register, the CPU0 writes 0x1 to GPIO Data Out Set register. This write causes bit[0] in the GPIO Data Out register to be set, without affecting any other bits (GPIO Data Out Set register works with write 1 to set, write 0 don't care policy). Similarly, CPU1 writes 0x2 to GPIO Data Out Set register will cause setting of GPIO Data Out Register bit[1], while not affecting the setting of bit[0].

Note

- The Set/Clear registers are only relevant only if multiple CPUs are required to simultaneously control the GPIO outputs. If this is not the case, use the conventional GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1) (Table 1248 p. 1422).
- If the GPIO outputs controlled by each CPU reside in a different byte segment of the GPIOs, the race can be avoided by simply using byte accesses to the GPIO Data Out register. For example, CPU0 controls GPIO[0] and CPU1 controls GPIO[8].
- Similar to GPIO Data Out Set/Clear registers, there are also GPIO Data Output Enable Set/Clear registers. There are useful to solve the above race condition when using the GPIOs as open source/open drain outputs.

Copyright © 2014 Marvell

MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Part 6: System Functions and Engines

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. Moving Forward Faster

Document Classification: Proprietary Information

THIS PAGE IS INTENTIONALLY LEFT BLANK

29 High-Speed Integrated SERDES Interface

The MV78230/78x60 integrates multiple Marvell high-speed SERDES (Serializer/Deserializer) interfaces. These SERDES lanes may be used as the electrical interface for various functionalities such as the SATA II, PCI Express, SGMII, DR-SGMII, QSGMII, and sETM interfaces. This versatility provides a large degree of flexibility for different types of applications.

A set of configuration registers is used to assign a SERDES lane to a specific operation mode. Configuring these registers should be completed before the link on the SERDES is established. The following steps are required to create the SERDES allocation:

- Create the SERDES allocation on the device pins via Shared SERDES 0-7 Selectors Register (Table 1452 p. 1525) and Shared SERDES 8-15 Selectors Register (Table 1453 p. 1526). Each field in this register represents a corresponding SERDES lane. The field value determines if the SERDES is active or not and sets the functionality mode of the SERDES.
- Configure the relevant MAC and SERDES to the specific work mode as described in the following sections.

Note

A full matrix of configuration options for this device is included in the device *Hardware Specifications*.

29.1 Features

The high-speed integrated SERDES interface supports:

- Multiple integrated low-power, high-speed SERDES lanes, based on Marvell® SERDES technology
- Versatile muxing options of PCIe, SATA, SGMII, QSGMII, and sETM interfaces

29.2 Configuring a MAC-SERDES Interface for PCIe Mode

In PCIe mode, the SERDES is accessed through the PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172). A write to this register triggers a read or write operation on the SERDES:

- As specified in the <PhyAccssMd> field
- To the address specified by the <PhyAddr> field
- With the data specified in <PhyData> field

Note

- The upper bits (bits[13:8] of the <PhyAddr> field determine the PCIe lane relevant for this configuration cycle. For x1 lanes select the value as specified in the register's field description. For x4 port, configure these bits as lane 0.
- The lower bits (bits[7:0]) of the same field determine the indirect register offset.
- For PCIe lanes 0 and 1, it is necessary to configure the SERDES ports through the lower index MAC. The upper bits of the <PhyAddr> field are fixed according to the lane number.

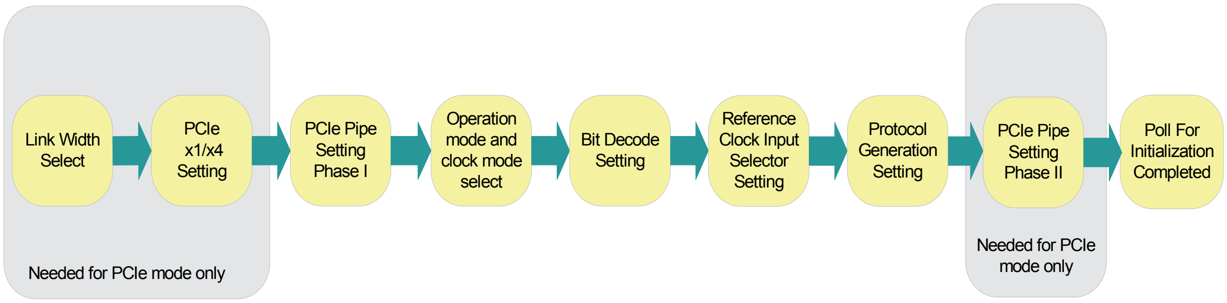
Each of the PCIe units within the device has its own set of indirect registers for SERDES configuration as listed below:

- To configure PCIe lane 0.x: offset 0x00041B00 in the PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172).
- To configure PCIe lane 1.x: offset 0x00081B00 in the PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172)
- To configure PCIe lane 2.x: offset 0x00043B00 in the PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172)
- To configure PCIe lane 3.x: offset 0x00083B00 in the PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172)

29.3 Assigning a SERDES Lane to the Relevant MAC

After selecting the operation mode for a given SERDES via registers Shared SERDES 0-7 Selectors Register (Table 1452 p. 1525) and Shared SERDES 8-15 Selectors Register (Table 1453 p. 1526), the software must configure the relevant MAC and SERDES operation modes. Figure 117 outlines the steps required to complete the MAC and SERDES configuration before the link may be used for an operational mode.

Figure 117:Assigning SERDES to a MAC


Link Width Select
PCIe x1/x4 Setting
Needed for PCIe mode only
PCIe Pipe Setting Phase I
Operation mode and clock mode select
Bit Decode Setting
Reference Clock Input Selector Setting
Protocol Generation Setting
PCIe Pipe Setting Phase II
Needed for PCIe mode only
Poll For Initialization Completed

The following sections describe the exact configurations required for each of these steps.

29.3.1 PCIe Link Width Select

Some of the PCIe units have the option of working in either x4 link width or quad x1 independent links. Set the <PCIenQuadBy1En> field in SoC Control Register to the required operational mode (x4 or quad x1).

Note Not all PCIe units have the option to be configured as x4 or quad x1. A unit may have a single option—either x4 or x1. Refer to Section 19, PCI Express Interface (PCIe) 2.0, on page 333 for more details on the device's PCIe unit setting options.

In addition, configure the width of physical interface (x4 or x1) in each PCIe controller via the <MaxLnkWdth> field in the PCI Express Link Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 823 p. 1123) of the relevant controller.

Note Only values of x4 or x1 are supported. Other values are reserved.

29.3.2 PCIe x4 Setting

Note These settings are only required if the PCIe is used in x4 mode.

To establish PCIe x4 links, configure the Communication PHY Reference Clock Alignment Register as follows:

- To establish an x4 link on PCIe unit 0, write 0xF to bits [3:0]
- To establish an x4 link on PCIe unit 1, write 0xF to bits [7:4]
- To establish an x4 link on PCIe unit 2, write 0xF to bits [11:8]
- To establish an x4 link on PCIe unit 3, write 0xF to bits [15:12]

29.3.3 PCIe Pipe Setting Phase I

Note These settings are only required if the PCIe is used.

The settings in Table 120 are needed to configure the SERDES to work in PCIe mode. Each row in the table defines an indirect access to one of the SERDES registers, as specified in Section 29.2, Configuring a MAC-SERDES Interface for PCIe Mode.

Table 120: PCIe Pipe Configurations

SERDES Register	Operation	Indirect Address[1]	Value[2]
GLOB_CLK_CTRL Reset and Clock Control Register (Table 903 p. 1192)	Apply PIPE soft reset + fixed pclk mode	0xc1	0x25
GLOB_TEST_CTRL Test Mode Control Register (Table 904 p. 1193)	Multicast enable bit configuration [Relevant for x4 links only]	0xc2	0x200
GLOB_CLK_SRC_LO Clock Source Low Register (Table 905 p. 1194)	Clock control (Relevant for x1 links only)	0xc3	0x0f
GLOB_DP_CFG Datapath Configuration Register (Table 907 p. 1195)	Re-write default values	0xc8	0x05
GLOB_PM_CFG0 Power Management Timing Parameter Register (Table 908 p. 1196)	Re-write default values	0xd0	0x100
GLOB_PM_CFG1 Power Management Timing Parameter 2 Register (Table 909 p. 1197)	Re-write default values	0xd1	0x3014
LANE_CFG0 Lane Configuration 0 Register (Table 900 p. 1189)	Re-write default values	0x80	0x1000
LANE_CFG1 Lane Configuration 1 Register (Table 901 p. 1190)	Re-write default values	0x81	0x11

[1] indirect Address corresponds to bits [7:0] of field <PhyAddr> in register PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172)

[2] Write Data corresponds to field <PhyData> in register PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1) (Table 880 p. 1172)

29.3.4 Operational Mode and Reference Clock Setting

To determine the operational mode of the SERDES and the reference clock frequency of the internal SERDES PLL, configure the <PHY_MODE> and <REF_FREF_SEL> fields in the Power and PLL Control register of the corresponding MAC. As each MAC contains a different Power and PLL Control register, only apply the setting to the relevant MAC, as selected by the SERDES multiplex options. For example, if you have chosen to work with PCIe lane 0.x, configure the Power and PLL Control register of PCIe MAC 0 only.

Table 121 lists the required setting per each operational mode and MAC selection.

Table 121: PHY MODE and REF_FREF_SEL Setting

Interface	PHY_MODE field	REF_FREF_SEL field	Register
PCIe	0x3 (PCIe)	0x0 (100 MHz)	Indirect register offset: 0x1 See Section 29.2for details about indirect access to the SERDES in PCIe mode. Write Data [15:0]: 0xFC60
SATA	0x0 (SATA)	0x1 (25 MHz)	Power and PLL Control Register (n=0–1) (Table 1011 p. 1271) Write Data: 0xF801

Table 121: PHY MODE and REF_FREF_SEL Setting (Continued)

Interface	PHY_MODE field	REF_FREF_ SEL field	Register
SGMII	0x4 (SERDES)	0x1 (25 MHz)	Power and PLL Control Register (i=0–3) (Table 558 p. 912) Write Data: 0xF881
QSGMII	0x4 (SERDES)	0x1 (25 MHz)	Power and PLL Control Register (i=0–3) (Table 558 p. 912) -- Configure Port 0 Write Data: 0xF881

29.3.5 Bits Decode Setting

To determine the bits decode of the SERDES to be either 10 or 20-bits according to the required link speed, configure the <SEL_BITS> field in the Digital Loopback Enable register of the corresponding MAC. As each MAC contains a different Digital Loopback Enable register, only apply the setting to the relevant MAC as selected by the SERDES multiplex options. For example, if chosen to work with PCIe lane 0.x, configure Digital Loopback Enable register of PCIe MAC 0 only.

Table 122 lists the required setting per each operational mode and MAC selection.

Table 122: SEL_BITS Setting

Interface	SEL_BITS field	Register
PCIe	NA	NA Done automatically by the SERDES while in PCIe mode
SATA	0x1 (20bits)	Digital Loopback Enable Register (n=0–1) (Table 1016 p. 1275) Write Data: 0x400
SGMII	0x0 (10bits)	Digital Loopback Enable Register (i=0–3) (Table 566 p. 917) Write Data: 0x0
QSGMII	0x1 (20bits)	Digital Loopback Enable Register (i=0–3) (Table 566 p. 917) -- Configure Port 0 Write Data: 0x400

29.3.6 Reference Clock Select Setting

To determine the source clock input to the reference clock of the integrated SERDES PLL, configure the <REFCLK_SEL> field in the Reference Clock Select register of the corresponding MAC. As each MAC contain a different Reference Clock Select register, only apply the setting to the relevant MAC as selected by the SERDES multiplex options. For example, if chosen to work with PCIe lane 0.x, configure Reference Clock Select register of PCIe MAC 0 only.

Table 123 outlines the required setting per each operational mode and MAC selection.

Table 123: REFCLK_SEL Setting

Interface	REFCLK_ SEL field	Register
PCIe	0x0	Indirect register offset: 0x46 See Section 29.2 for details about indirect access to the SERDES in PCIe mode. Write Data [15:0]: 0x0000
SATA	0x1	Reference Clock Select Register (n=0–1) (Table 1018 p. 1276) Write Data: 0x400
SGMII	0x1	Reference Clock Select Register (i=0–3) (Table 568 p. 918) Write Data: 0x400
QSGMII	0x1	Reference Clock Select Register (i=0–3) (Table 568 p. 918)—Configure Port 0. Write Data: 0x400

29.3.7 Impedance Tuning

Enable the high RX impedance setting of the SERDES by the <RX_HIZ> field in the COMPHY Control Register (i=0–3) (Table 569 p. 918). .

Interface	RX_HIZ Field	Register
PCIe	0x0	Indirect register offset: 0x48 See Section 29.2 for details about indirect access to the SERDES in PCIe mode. Write Data [15:0]: 0x8080
SATA	0x0	See the COMPHY Control Register (n=0–1) (Table 1019 p. 1276) (offsets Write Data: 0x8080
SGMII	0x0	See the COMPHY Control Register (i=0–3) (Table 569 p. 918) Write Data: 0x400
QSGMII	0x0	See the COMPHY Control Register (i=0–3) (Table 569 p. 918) (offset 0x72F20)—Configure Port 0. Write Data: 0x400

29.3.8 Protocol Generation Setting

To determine the protocol generation in which the interface will work, configure the <PIN_PHY_GEN> field in the LP_PHY_EXTERNAL_CONTROL register for the SATA ports and the SERDES Configuration register for the SGMII/QSGMII ports of the corresponding MAC. As each MAC contains different such registers, only apply the setting to the relevant MAC, as selected by the SERDES multiplex options. For example, if you have chosen to work with SGMII 0, configure the register of SGMII MAC 0 only.

Table 124 lists the required settings per each operational mode and MAC selection.

Table 124: PIN_PHY_GEN Setting

Interface	PIN_PHY_GEN field	Register
PCIe	NA	PCIe standard defines specific flow to determine the link generation and speed. Refer to the PCIe section for more details.
SATA	Gen II (3Gbps) 0x11 Gen I (1.5Gbps) 0x00	LP Phy External Control Register (n=0–1) (Table 968 p. 1238) Write Data, Gen II: 0x227 Write Data, Gen I: 0x7
SGMII	SGMII (1.25Gbps) 0x6 DRSGMII (3.125Gbps) 0x8	Serdes Configuration (SERDESCFG) Register (i=0–3) (Table 695 p. 1002) Write Data, SGMII: 0xcc7 Write Data, DRSGMII: 0x1107
QSGMII	QSGMII (5Gbps) 0x3	Serdes Configuration (SERDESCFG) Register (i=0–3) (Table 695 p. 1002) -- Configure Port 0 Write Data: 0x667

29.3.9 PCIe Pipe Setting Phase II

Note

These settings are only required if the PCIe is used.

The settings in Table 125 are needed to configure the SERDES to work in PCIe mode. Each row in the table defines an indirect access to one of the SERDES registers, as specified in Section 29.2, Configuring a MAC-SERDES Interface for PCIe Mode.

Table 125: PCIe Pipe Configurations

SERDES Register	Operation	Indirect Address	Value
GLOB_CLK_CTRL Reset and Clock Control Register (Table 903 p. 1192)	Remove PIPE soft reset + fixed pclk mode	0xc1	0x24

29.3.10 Poll for Initialization Completed

To complete the configuration process, the CPU must poll the completion indications from the various SERDES lanes, as listed in Table 126.

Table 126: Completion Indications

Interface	Completion Value (CPU should poll for this value)	Register (Completion status will appear in this register)
PCIe	NA	PCIe standard defines specific flow to determine the link status.
SATA	Bits[2:0] = 0x7	LP Phy External Status Register (n=0–1) (Table 969 p. 1239)

Table 126: Completion Indications (Continued)

Interface	Completion Value (CPU should poll for this value)	Register (Completion status will appear in this register)
SGMII	Bits[2:0] = 0x7	Serdes Status (SERDESSTS) Register (i=0–3) (Table 696 p. 1002)
QSGMII	Bits[2:0] = 0x7	Serdes Status (SERDESSTS) Register (i=0–3) (Table 696 p. 1002) —Poll on Port 0

Once a completion indication has been received, the SERDES and MAC are programmed to work in the desired operation mode.

Note

Completing this stage connects a specific MAC to a SERDES and configures the operational mode of the link on the specific SERDES lane.

To establish the external interface link, the software must complete link establishment, as required by the specific interface protocol specifications.

29.4 Configure a SERDES for sETM Mode

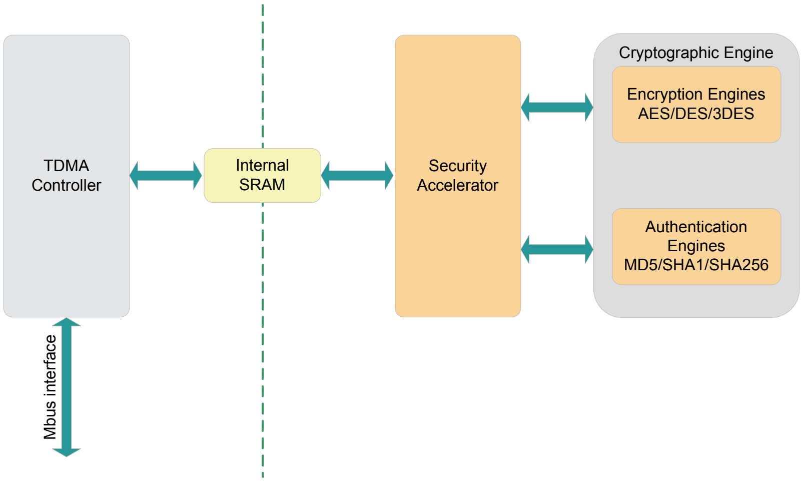
Note

For more information about configuring a SERDES for sETM mode, contact a Marvell Field Application Engineer (FAE).

30 Cryptographic Engines and Security Accelerators (CESA)

The device integrates two hardware based Cryptographic Engines and Security Accelerators (CESA) and a TDMA engine. Since both units are identical, this section describes the structure of a single CESA.

Figure 118:Cryptographic Engines and Security Accelerators Functional Block Diagram



The CESA registers are located in Appendix A.21, Cryptographic Engine and Security Accelerator (CESA) Registers, on page 1439.

Acronyms

See Table 2, Terms and Abbreviations, on page 29 for the definitions of terms used in this section.

30.1 Features

The CESA incorporates the cryptographic engines, TDMA engine, and Security Accelerator as listed below.

30.1.1 Cryptographic Engine Features

There are 4, independently-operating cryptographic engines:

- Authentication MD5/SHA-1/SHA-256 engine (see Section 30.3.1, Authentication, on page 460
- DES encryption/decryption engine (see Section 30.3.2, DES Encryption / Decryption, on page 463)

- AES128 encryption engine (see Section 30.3.3, AES128 Encryption, on page 469)
- AES128 decryption engine (see Section 30.3.4, AES128 Decryption, on page 472)

Each of these four independent cryptographic engines has separate registers for data, control, and operation modes. The address allocation is specified in Table 1298, Summary Map Table for the Cryptographic Engine and Security Accelerator (CESA) Registers, on page 1439.

These engines implement the following algorithms:

- Encryption: DES (ECB and CBC mode) and Triple DES (ECB and CBC mode, EDE and EEE)
- Encryption: AES128/128, AES128/192, AES128/256.
- Authentication: SHA-1, SHA-2 (with 256-bit digest), and MD5

The cryptographic engines implement the following features:

- Authentication in the MD5, SHA-1, or SHA-256 algorithm, selectable by the user
- Authentication Continue mode, enables chaining between blocks
- Authentication Automatic Padding mode
- Encryption and Decryption in Single DES, Single (ECB) or Block (CBC) mode, or 3DES, EEE or EDE mode, selectable by the user
- DES write pipeline
- Optimal external update of Authentication and Encryption (in CBC) initial values, enabling flexibility of use—multi-packet calculation, sharing between resources
- Byte Swap support for Data input and initial values
- Byte Swap support for DES/3DES and AES data output.
- Automatic engine activation when the required data block is loaded
- Authentication and encryption can be done simultaneously
- Authentication and encryption termination interrupts
- Supports DES OFB and CFB modes with additional software
- AES encryption and decryption—completely separate engines that can work simultaneously when TDMA and the Security Accelerator are not used.

30.1.2 TDMA Engine Features

There is one TDMA engine that implements the following features:

- Moves data back and forth between the main memory and the internal SRAM
- Transfers a single data buffer of up to 2 KB
- Supports Chain mode

30.1.3 Security Accelerator Features

The Security Accelerator utilizes the engines to perform one of the following operations per-packet:

- Performs a complete over-the-packet operation with no software intervention
- Supports 4 types of operation:
  - Authentication only (MD5/SHA-1/SHA-256/HMAC-MD5/HMAC-SHA-1/HMAC-SHA-256)
  - Encryption/Decryption only (DES/3DES/AES - both ECB and CBC)
  - Authentication followed by Decryption/Encryption
  - Decryption/Encryption followed by Authentication

30.2 Theory of Operation

The CESA can work in one of the following operational flows:

- The Software directly controls the cryptographic engine per data block.
- The Security Accelerator performs a complete over-the-packet operation with no software intervention. The software or TDMA copies the packet from/to main memory to/from the internal SRAM.
- An Enhanced Flow that attaches the TDMA to the Security Accelerator on a per packet basis or by a Multi-Packet Chain.

The recommended/performance optimized flow is the Enhanced Multi-Packet Chain Software flow described in Section 30.6.2, Multi-Packet Chain Mode, on page 490

In this flow, the TDMA engine copies packets and descriptors into the internal SRAM, the Security Accelerator controls the packets processing according to the descriptors (using the cryptographic engines), and the TDMA engine copies the processed packet back to the main memory. Multiple packets can be chained and processed by the hardware without software intervention.

30.3 Functional Description—Cryptographic Engine

Access

The cryptographic engines can be accessed by the Security Accelerator or by the host[1]. The access is done by writing and reading to specified addresses in the engine.

Commands and Control

The engines' modes of operation (AES, DES, 3DES, and SHA) and the endianess of the input data are controlled by the host. Control is done via four specified command registers:

- SHA-1/SHA-2/MD5 Authentication Command Register (p=0–1) (Table 1367 p. 1467)DES Command Register (p=0–1) (Table 1351 p. 1460)
- AES Encryption Command Register (p=0–1) (Table 1311 p. 1445)
- AES Decryption Command Register (p=0–1) (Table 1324 p. 1448).

These registers also contain flags that are used as status indicators to the host.

The engines also provide interrupts that are set when an operation is completed (see Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1450).

Input Data

The encryption engines operate on data blocks of 64 bits or 128 bits, while the authentication engine requires a block of 16 words (512 bits) as input.

The input data is loaded by writing to data registers.

The engines, excluding DES, do not support multi-tasking. When a data block is written to one of the engines, the host must wait until this engine finishes the calculation before it writes the next input data block.

The engines also require cryptographic parameters, such as keys and initial values, according to the mode of operation used. This is done to by writing to specific registers.

Both the encryption engine and the authentication engine support byte swap of input data.

Output Data The encryption engines return a cipher/decipher data block of 64 or 128 bits. The engines support byte swap of their output data.

The authentication engine returns four, five, or eight words that are the hash signature of the input data block.

The output data is accessed through specific registers in the engines.

Principle of Operation When a host wants to perform cryptographic operations, it writes the cryptographic parameters (for example, IV, keys) and the command to be performed in registers. Then the host writes the data to be processed (in the data registers). This triggers the engine, which starts the processing automatically after the required amount of data was written. When the engine finishes the cryptographic calculation, it sets a termination bit in one of the command registers and activates an interrupt to notify the host that the operation is finished. After the operation is finished, the host can read the result of the operation from the engines' registers.

1. The word host is used as a generic term representing the agent that controls the operation of the cryptographic engines (CPU or Security Accelerator). While the accelerator is working the host cannot work with the cryptographic engines and vise versa.

30.3.1 Authentication

To activate the authentication process, the host should perform the following (some of the stages are optional):

1. Verify the <Termination> field in the SHA-1/SHA-2/MD5 Authentication Command Register (p=0–1) (Table 1367 p. 1467) in the register.

   The host must read the register to verify that the engine is not in the middle of a calculation process. Writing in the middle of a calculation results in erroneous data.

   After reset, the termination bit (bit[31]) is set. Any write that the host performs to the Authentication engine resets the termination bit.

2. Write to the <Mode> field and the <DataByteSwap> field in the SHA-1/SHA-2/MD5 Authentication Command Register (p=0–1) (Table 1367 p. 1467) (This is optional if no change is needed.)

3. Write initial values:

   This stage is only required if multiple packets are processed simultaneously.

   Externally written Initial values are valid only if the <Mode> is set to 1 (Continue mode).

   The initial value is five words long in SHA-1/SHA-256, and in MD5 it is four words long. The IV/digest specified in the SHA-1/SHA-2/MD5 Initial Value/Digest A Register (p=0–1) (Table 1362 p. 1465) through SHA-2 Initial Value/Digest H Register (p=0–1) (Table 1372 p. 1470).

   The host may write to any of these registers. Initial values registers that are not written contain the digest from the previous calculation.

4. Write data words

   NOTE: The host may change the value of the command register during the process of writing the data. This may be done when it is necessary to swap only part of the data.

   SHA-1/SHA-256/MD5 algorithms work in 512-bit chunks, which are equal to 16 words. There are two ways to write the data words to the engine:

   1.Write cycles to Data In register.

   2.Write cycles to the Data In register and to the Byte Count registers.

30.3.1.1 Write Cycles to Data In Register

The host must perform 16 write cycles, first to W0, then to W1 through W15.

30.3.1.2 Write Cycles to the Data In Register and to Byte Count Registers

This type of access is preferred where a packet is small (i.e., less than 14 words) or for the last chunk of a packet whose size is less than 14 words. In these cases, the algorithm requires a zero padding to 14 words in addition to the 2-word padding of the bit count needed in single/last chunks. This requires successive writes of full zero words.

In this type of access, the writing of these zero padding is skipped and thus less then 16 write accesses activate the engine.

This access is done in the following manner:

1. The host writes between 0 to 13 words to the Data In register. The write to the Data In register is done until the word that contains bit N+1 of the data, where N is the place of the last bit of data in the chunk. The last word written must be padded with one bit of 1 and then zeros until the completion of the 32-bit word.
2. Then, the host writes the lower part (MD5) or the higher part (SHA-1/SHA-256) of the bit count value to the SHA-1/SHA-2/MD5 Bit Count Low Register (p=0–1) (Table 1368 p. 1469) —word 14 (see Figure 119).
3. Then the host writes the higher part (MD5) or the lower part (SHA-1/SHA-256) of the bit count value to the SHA-1/SHA-2/MD5 Bit Count High Register (p=0–1) (Table 1369 p. 1469)—word 15 (see Figure 119). After this write, the engine starts working automatically, and all words of the data chunk from the last word written to the Data In register until word 14 are considered as zeros.

Less than 16-Word	16-Word
word 0	word 0
word 1	word 1
:	
:	
:	
word 13 or less	word 13
bit count low	word 14
bit count high	word 15

4. Poll the SHA-1/SHA-2/MD5 Authentication Command Register (p=0–1) (Table 1367 p. 1467) or wait for the interrupt:

   After the engine is loaded with the data chunk or after a write to the two Bit Count registers, the engine starts working automatically.

   When the <Termination> field is set to 1, it is an indication for the host that the engine finished the calculation process, and the digest is ready.

   The host can poll the <Termination> field or wait for the <ZInt0> field in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1452). This interrupt is activated on the rising edge of the <Termination> field. To clear the interrupt, the host must write a 0 to it. Writing a 1 has no effect.

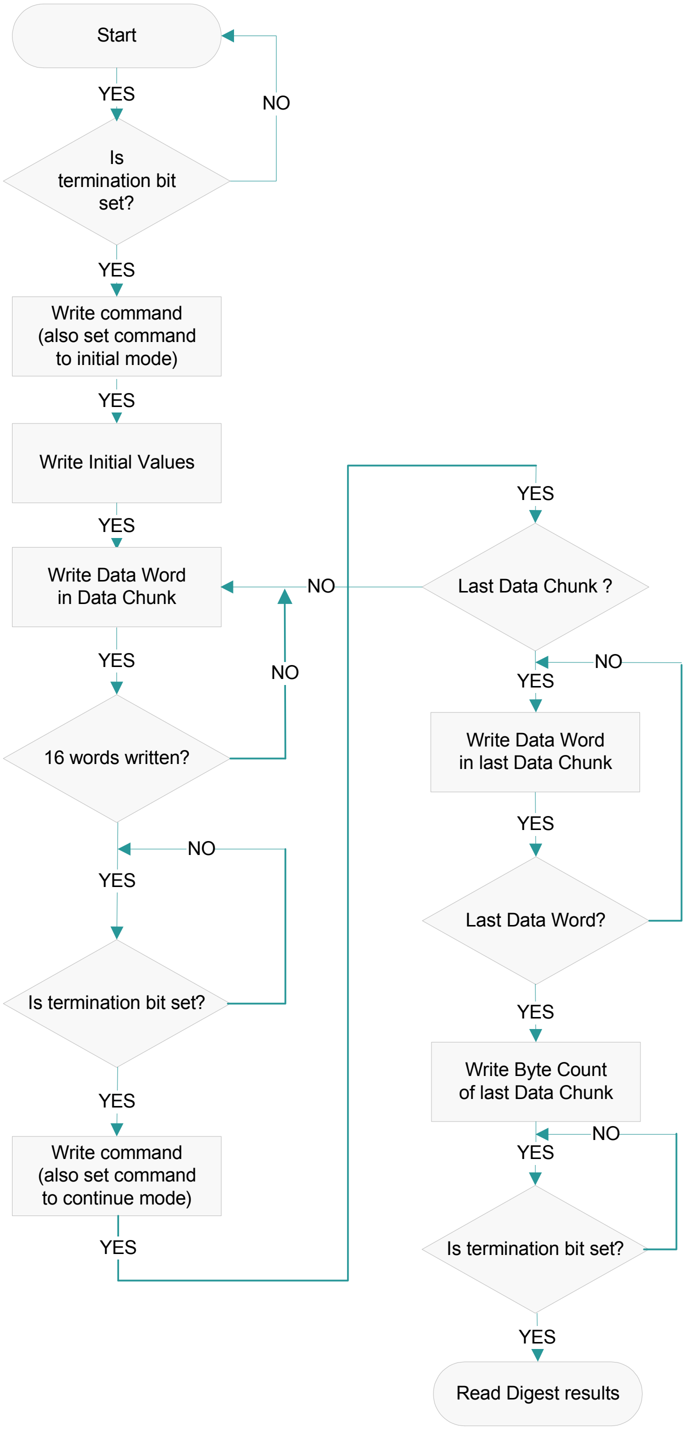
5. Read result:

   Once the <Termination> bit is asserted, the host may read the digest. The digest length is eight words for SHA-256, five words for SHA-1 or four words for MD5. These words are stored in the (see SHA-1/SHA-2/MD5 Initial Value/Digest A Register (p=0–1) (Table 1362 p. 1465) through SHA-1/SHA-2/MD5 Data In Register (p=0–1) (Table 1373 p. 1470)

After reading the result, the host can immediately start the write command again for initial values and data, or just data.

Figure 119 shows a typical authentication flow for a packet.

Figure 119:Typical Authentication Flow for a Packet



30.3.2 DES Encryption / Decryption

The DES encryption algorithm complies with the DES standard as described in FIPS PUB 46-2.

The engine implements two different modes:

ECB	Direct application of the DES standard to encrypt and decrypt data
CBC	An enhanced mode of ECB that chains together blocks of cipher text. The chain's glue is the DES IV (Initial Value) register (see DES Initial Value Low Register (p=0–1) (Table 1345 p. 1459) and DES Initial Value High Register (p=0–1) (Table 1346 p. 1459).

Two other modes, CFB and OFB, may be implemented by the engine, however, additional software is required.

The engine implements two triple DES (3DES) modes, as described in RFC 1851:

- EEE
- EDE

The 3DES modes work with three different keys for high security and can be combined with ECB (EEE or EDE only) or CBC (EDE only) modes.

All the modes are reciprocal, that is, they decipher or cipher data.

To activate the encryption engine, the following steps are required:

1. Verify termination bits in the DES Command Register (p=0–1) (Table 1351 p. 1460).

   The host must read the register, to verify that the engine is not in the middle of a calculation process and that the engine's parameters (i.e., DES operation modes and either the DES key or the 3DES keys) can be updated.

   In the initial operation, it is always necessary to write the DES key or the 3DES keys and possibly to write an operational mode other then the default operational mode. In that case, before the engine's parameters are written, the <AllTermination> field in the DES Command Register (p=0–1) (Table 1351 p. 1460) must be set.

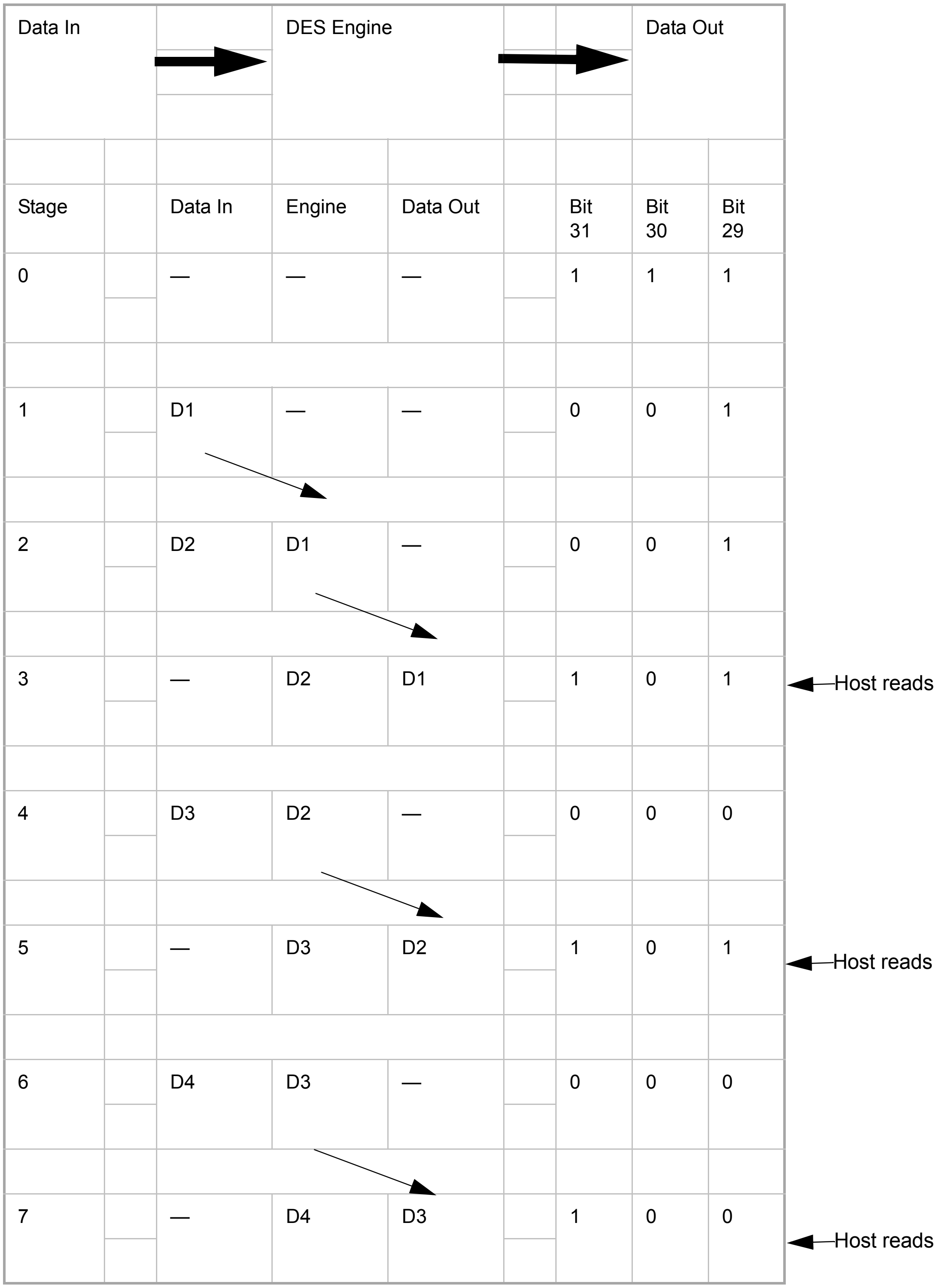
   When the <Termination> field in the DES Command Register (p=0–1) (Table 1351 p. 1460) is set, a 64-bit DES data block can be written to the engine, but this does not necessarily mean that parameters can be updated. Writing to the engine de-asserts this bit.

   However, when the <AllTermination> is set, a DOUBLE 64-bit DES data block can be written to the engine. In addition, when this bit is set the engine's parameters can be updated.

   The DES engine operates on a pipeline principle, see Figure 120: "DES Engine Pipeline", on page 464.

   The host writes to the engine when the <WriteAllow> field in the DES Command Register (p=0–1) (Table 1351 p. 1461) is set, and when a result is ready, the host must read it to enable the engine to process the next data in the pipeline. When <WriteAllow> is set, the host can write one data block to the engine.

Figure 120:DES Engine Pipeline



Stage	Data In	Engine	Data Out	Bit 31	Bit 30	Bit 29	
0	—	—	—	1	1	1	
1	D1	—	—	0	0	1	
2	D2	D1	—	0	0	1	
3	—	D2	D1	1	0	1	Host reads
4	D3	D2	—	0	0	0	
5	—	D3	D2	1	0	1	Host reads
6	D4	D3	—	0	0	0	
7	—	D4	D3	1	0	0	Host reads

Note: A read of Dataout Low resets <WriteAllow>.

Copyright © 2014 Marvell

Table 127 summarizes the meaning of bits [31:30] and the possible states.

Table 127: DES Command Register Status Bits and Their Meaning

Bit 31	Bit 30	Description	Number of Data 64-Bit Blocks That Can Be Written	Can Engine Parameters Be Updated?
0	0	Engine is busy.	1—only if bit 29 is set	No
0	1	Engine is ready.	2	Yes
1	0	First data block waiting for read; pipeline is full.	1—only if bit 29 is set	No
1	1	Engine completed calculations; pipeline is empty.	2	Yes

2. Write operational mode and endianess for fields in the DES Command Register (p=0–1):

Direction	Encryption or decryption
Algorithm	Single DES or 3DES
Triple DES mode	EEE or EDE
Chain	ECB or CBC
Data byte swap	Similar to swapping performed in the authentication engine (see Section 30.3.1, Authentication, on page 460).
IV byte swap	Swapping of data written to the DES IV register. Similar to the data swap.
Data out byte swap	These bits control byte swap of the cipher/decipher output result.

3. Write the keys

   Each key is a 64-bit block. Writing a single key requires two write operations:

Single DES mode	A write to a single key must be performed.The key used in DES mode is the KEY0 register.
3DES mode	Marvell recommends to write to three different keys: KEY0, KEY1, and KEY2 registers.

4. Write the initial value (IV).

   This step is necessary only in the DES/3DES CBC modes. A 64-bit block must be written to the IV register. In CBC, the IV value is XORed with the Data_In. The result is used as the input to the cipher/decipher machine.

   Writing the IV register requires two write operations, one to the DES Initial Value Low Register (p=0–1) and the other to the DES Initial Value High Register (p=0–1). In 64-bit mode, a single write operation is required.

5. Write blocks for the 64-bit data.

   DES encryption requires a 64-bit block of input data that is loaded by writing to one of the DES Data Buffer registers (see DES Data Buffer Low Register (p=0–1) (Table 1354 p. 1462) or DES Data Buffer High Register (p=0–1) (Table 1355 p. 1462). One or two data blocks may be loaded according to the engine status (see Table 127).

   If a data block is shorter than 64-bit length, the specification requires zero padding to 64 bits.

   If the next block to be processed uses the same cryptographic parameters (i.e., the same keys and modes), and if the IV is the output data of the previous block, the host writes only the DES Data Buffer registers (see Table 1354 on page 1462 or Table 1355 on page 1462). This is useful when it is necessary to encrypt a message that consists of multiple 64-bit blocks. In this case, it is efficient to use the DES pipeline option and write two blocks at once as specified in Table 127.

The DES machine starts working automatically when a 64-bit data block is written to it, or when there is data in its pipeline, and the previous result has been read.

Operation starts after the host writes to the DES data buffer addresses. The host must first write to the DES data in/out low address and then to the DES data buffer high address, as shown in the following example.

```
The data block to encrypt is 0x1122334455667788.
The host writes 0x55667788 to address 0xDD70.
Then, the host writes 0x11223344 to address 0xDD74.
```

The result of this write operation are shown in the following table.

Table 128: DES Encryption / Decryption Write Operation Result

Byte Swap	DES Data Buffer Register: Actual data that will be encrypted
0	1122334455667788
1	4433221188776655

NOTE: Other writes to addresses 0xDD70/0xDD74 will cause unexpected results.

6. Poll the DES Command Register (p=0–1) (Table 1351 p. 1460) or wait for the interrupt.

After the engine is loaded with one 64-bit blocks, it starts working automatically. The host must not write anything to the engine until the <WriteAllow> is set.

At this stage, the host must poll the <Termination> field in the DES Command Register (p=0–1) (Table 1351 p. 1460). When the bit is 1, it is an indication to the host that the engine finished the calculation process and that the result is ready.

The host may write one data block each time, or it may perform consecutive writes of two data blocks:

The host writes one data block each time:

The result of the data block is ready, and no more data is in the engine pipeline. Bit [30], the <AllTermination> bit, will be set and engine parameters can be updated.

The host performs consecutive writes of two data blocks:

First data block When the first data block result is ready, the <Termination> field and the <WriteAllow> field are set. At this stage, the engine had the second data in the pipeline. However, the calculation cannot start until the host has read the result of the first data via DES Data Out registers (read the Data High before the Data Low). Here bit [30], the <AllTermination> bit is not set.

Once the first data result is read, the <WriteAllow> field is reset, and the engine starts the calculation of the second data. At this stage, the host can write one data block (the third block) to the engine. This was indicated by <WriteAllow> (see Figure 120, DES Engine Pipeline, on page 464).

Second data block When the second data block result is ready, the <Termination> field is set. At this stage, the engine had completed the second data calculation and the pipeline was empty. In this situation, the <AllTermination> bit is also set. Once the data result was read, the host can alter the DES parameter and then write two data blocks.

The ZInt1 interrupt is set every time the <AllTermination> field in the DES Command Register (p=0–1) (Table 1351 p. 1460) changes from 0 to 1.

The ZInt4 interrupt is set every time the <Termination> field in the DES Command Register (p=0–1) (Table 1351 p. 1460) changes from 0 to 1.

After the interrupt occurs, the host writes 0 to the interrupt relevant bit in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1450) to reset it. Writing 1 has no effect.

7. Read DES result.

   Once a termination bit has been asserted, the host may read the result. The result of the encryption (or decryption) is stored in the DES Data In/Out registers.

   Two read operations are required to read the result—the first read from address 0xDD7C and the second read from address 0xDD78.

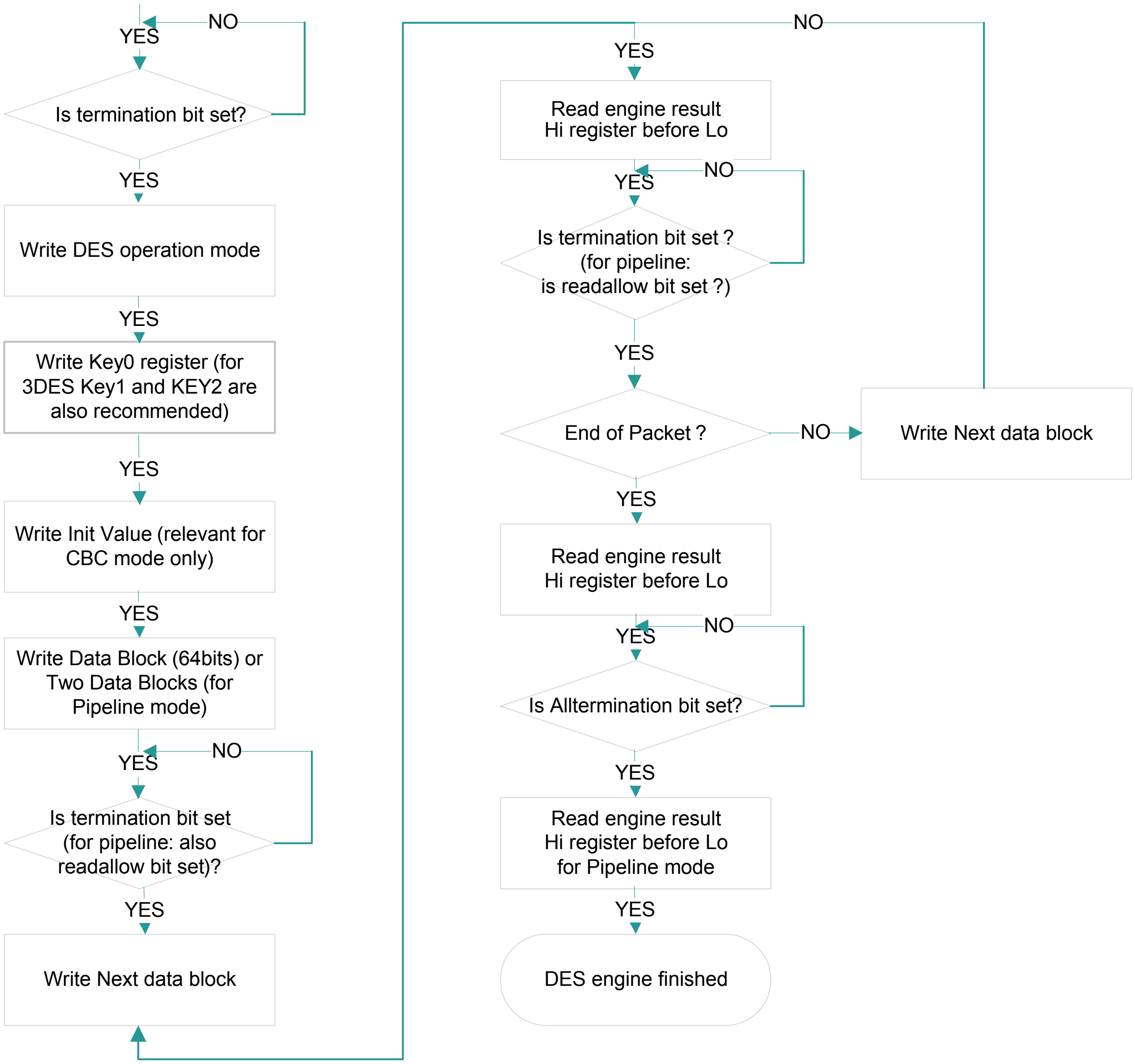
   Byte swap bits have no effect on reads, that is, the engine does not perform any endianess change on the result. To perform an endianess change, use the <DataByteSwap> field in the DES Command Register (p=0–1) (Table 1351 p. 1461).

8. Reading keys and IV

   It is possible to read the key values and IV value at any time by accessing the appropriate registers. In Chain modes (CBC), the IV register is changed by the machine at the end of every DES calculation cycle. In this situation, the IV register is loaded with the calculation result, so it should have the same value as the DES Data Out register (see DES Data Out Low Register (p=0–1) (Table 1356 p. 1462) and DES Data Out High Register (p=0–1) (Table 1357 p. 1463).

Figure 121 illustrates the typical DES/3DES packet encryption flow.

Figure 121:Typical DES/3DES Encryption Flow for Packet


NO
YES
Is termination bit set?
YES
Write DES operation mode
YES
Write Key0 register (for 3DES Key1 and KEY2 are also recommended)
YES
Write Init Value (relevant for CBC mode only)
YES
Write Data Block (64bits) or Two Data Blocks (for Pipeline mode)
NO
YES
Is termination bit set (for pipeline: also readallow bit set)?
YES
Write Next data block
NO
YES
Read engine result Hi register before Lo
NO
YES
Is termination bit set ? (for pipeline: is readallow bit set ?)
YES
End of Packet ?
NO
Write Next data block
YES
Read engine result Hi register before Lo
NO
YES
Is Alltermination bit set?
YES
Read engine result Hi register before Lo for Pipeline mode
YES
DES engine finished

Copyright © 2014 Marvell

30.3.3 AES128 Encryption

The AES128 encryption engine complies with the AES standard as described in the FIPS standard.

This engine does encryption only. The decryption is performed by the AES128 decryption engine (see Section 30.3.4, AES128 Decryption, on page 472).

The engine implements the AES algorithm on a 128-bit data block on three possible Key Length modes:

- 128-bit mode
- 192-bit mode
- 256-bit mode

To activate the AES encryption engine, the following steps are required.

1. Verify the <Termination> field in the AES Encryption Command Register (p=0–1) (Table 1311 p. 1445):

   At this stage the host must read the register, to verify that the engine is not in the middle of a calculation process. Writing in the middle of a calculation results in erroneous data. Unless the host made a write to the engine before, the <Termination> must be set (this is also true after reset).

   Any write that the host performs to the AES encryption engine resets the <Termination> bit.

2. Write operational mode and endianess for fields in the AES Encryption Command Register (p=0–1):

   - Key Length mode—128, 192 or 256 bit
   - Data byte swap—this is similar to the swapping performed in the Authentication engine (see Section 30.3.1, Authentication, on page 460).
   - Data out byte swap—these bits control byte swap of the cipher output result.

3. Write key:

   The AES key length can be changed in accordance to the Key Length mode selected. It may be a 128-, 192-, or 256-bit block. Writing a single key requires four, six, or eight write operations.

   With the key maximum length 256-bit block, the block is structured from eight words, each word is a column in the AES Cipher Key block:

AES key column 0	AES key column 1	AES key column 2	AES key column 3

   Thus, there are eight AES encryption key registers, each of them contains a column of the AES cipher key block.

   Commonly, a 4-word key is used. In this case, the host must write to the Key Column 0, 1, 2, and 3 registers.

   Since keys are changed only occasionally, this step is not always mandatory.

4. Write block for the 128-bit data.

   The AES encryption requires a 128-bit block of input data. This block is loaded by writing to the AES Encryption Data In/Out register.

   If a data block is shorter than 128-bits, the specification requires zero padding to 128 bits.

   If the next block to be processed uses the same key; the host should write only the AES Data In/out encryption registers (see Table 1307 on page 1444 through Table 1310 on page 1445). This is useful when it is necessary to encrypt a message that consists of multiple 128-bit blocks.

   The AES machine starts working automatically when a 128-bit data block is written to it (that is, operation starts after the host writes to all the AES data in/out addresses).

This data is a 128-bit block. This block is structured from four words. Each word is a column in the AES Cipher Data block.

AES Data column 0 (0xDDAC)	AES Data column 1 (0xDDA8)	AES Data column 2 (0xDDA4)	AES Data column 3 (0xDDA0)

Thus, there are four AES Encryption Data In/Out registers, each of them containing a column of the AES cipher block.

Upon writing to all these registers, the AES cipher machine will automatically start working as shown in the following example. The order in which these registers is written is not significant.

> The data block to encrypt is `0x9900AABBCCDDEEFF1122334455667788`.
> This data is a 128 bit block, structured from four words.
> Each word is a column in the AES Cipher data block.
> The data must be loaded to the cipher machine as follows:
>
> > Write 0x55667788 to address 0xDDA0.
> >
> > Write 0x11223344 to address 0xDDA4.
> >
> > Write 0xCCDDEEFF to address 0xDDA8.
> >
> > Write 0x9900AABB to address 0xDDAC.

There is full support for data byte swap to the AES data blocks, similar to the DES engine, but in the AES cipher engine, all data column registers are 1 word in width.

Results of this write operation.

Table 129: AES128 Encryption Write Operation Result

Byte Swap	AES Data In/out
0	9900AABBCCDDEEFF1122334455667788
1	BBAA0099FFEEDDCC4433221188776655

NOTE: Other writes to addresses 0xDDA0, 0xDDA4, 0xDDA8, or 0xDDAC cause unexpected results.

5. Poll the AES Encryption Command register or wait for the interrupt

   After the engine is loaded with the 128-bit block, it starts working automatically. The host must not write anything to the engine until it finishes the calculation.

   At this stage, the host must poll the <Termination> field in the AES Encryption Command Register (p=0–1) (Table 1311 p. 1445). When the bit is 1, it is an indication to the host that the engine finished the calculation process, and the result is ready.

   Authentication calculation termination activates the <ZInt2> interrupt (see the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1450), which can serve as an alternative to host polling. This interrupt is set every time the <Termination> field in the AES Encryption Command Register (p=0–1) (Table 1311 p. 1445) changes from 0 to 1.

   After the interrupt occurs, the host should write 0 to <ZInt2> field in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1452) to reset it. Writing 1 has no effect.

6. Read AES result.

   Once the termination bit has been asserted, the host may read the result. The result of the encryption is stored in the AES Data In/Out registers (see Table 1307 on page 1444 through Table 1310 on page 1445).

Four read operations are required to read the result, i.e., the host must read addresses 0xDDA0, 0xDDA4, 0xDDA8, and 0xDDAC.

Byte swap bits have no effect on reads, i.e., the engine does not perform any endianess change on the result. To perform an endianess change, use the <OutByteSwap> field in the AES Encryption Command Register (p=0–1) (Table 1311 p. 1445).

7. Read keys.

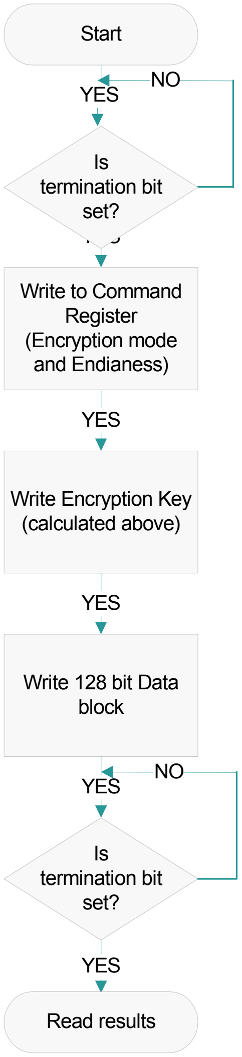
   It is possible to read the key values at any time, by accessing the appropriate registers. To perform an endianess change on key reads, use the <OutByteSwap> field in the AES Encryption Command Register (p=0–1) (Table 1311 p. 1445).

   The AES Encryption Command Register (p=0–1) controls the AES encryption modes. It also flags when the processing is complete.

   NOTE: The AES encryption key reads are necessary for the AES decryption engine (see Section 30.3.4, AES128 Decryption, on page 472).

Figure 122 shows a typical authentication flow for a packet.

Figure 122:Typical AES Encryption Flow for a Data Block



30.3.4 AES128 Decryption

The AES128 decryption engine complies the AES standard as described in the FIPS standard.

This engine performs decryption. The modes of operation and activation are similar to the AES encryption.

In respect to AES encryption, the only difference is that the host must calculate a decryption key before loading the key. This process is performed by loading a key into the AES encryption unit, performing a “dummy” encryption cycle, and then reading the resolved key from that engine. The result is the decryption key.

A decryption key is the last 128/192/256 bits of the key block created by the key expansion algorithm.

As in the encryption engine, the decryption engine implements AES algorithm on a 128-bit data block size in three possible mode sizes:

- 128-bit mode
- 192-bit mode
- 256-bit mode

To activate the AES Decryption engine, the following steps are required:

1. Verify the termination bit in the AES Decryption Command register.
2. Calculate decryption key

   This step is unique to the AES decryption engine, and is only necessary when a new key, which was never used before, is loaded. The AES algorithm uses a complex key schedule. Thus, at the end of the encryption operation the key is changed. To perform AES decryption the engine must actually start from the key at the end of the encryption key schedule. To decrypt a data block with a given key, the host must first load this key into the decryption engine and then, start the key generation process setting the <AesDecMakeKey> field in the AES Decryption Command Register (p=0–1) (Table 1324 p. 1449) bit to 1. At the end of the key generation process, the host reads the key registers from the encryption engine. This decryption key is loaded by the host into the decryption key registers to start the required description process.

   To read the decryption key from the encryption engine, the host must set the <AesDecKeyReady> field in the AES Decryption Command Register (p=0–1) (Table 1324 p. 1448) to 1 prior to the reading of the AES encryption key registers. Setting this bit enables reading of the internal key in the AES encryption engine, which at the end of an encryption process, is the key for the decryption start point.

   The host may store the decryption key in memory, so that the decryption key calculation may be skipped next time, and the same key used.

   NOTE: Steps 3—8 are the same as for AES encryption.
3. Verify the termination bit in the AES Decryption/Encryption Command register.
4. Write operational mode and endianess for fields in the AES Decryption Command Register (p=0–1).
5. Write decryption key.
6. Write block for the 128-bit data.
7. Poll the AES Decryption Command register or wait for the interrupt.

   After the engine is loaded with the 128-bit block, it starts working automatically. The host must not write anything to the engine until it finishes the calculation. At this stage, the host must poll the <Termination> field in the AES Decryption Command Register (p=0–1) (Table 1324 p. 1448). When the bit is 1, it is an indication to the host that the engine finished the calculation process, and the result is ready. Authentication calculation termination activates the <ZInt3> interrupt (see the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1450) which can serve as an alternative to host polling. This interrupt is set every time the <Termination> field in the AES Decryption Command Register (p=0–1)

(Table 1324 p. 1448) changes from 0 to 1. After the interrupt occurs, the host should write 0 to <ZInt3> field in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1452) to reset it. Writing 1 has no effect.

8. Read AES result.
9. Read keys.

   It is possible to read the key values at any time by accessing the appropriate registers. To perform an endianess change on key reads, use the <OutByteSwap> field in the AES Encryption Command Register (p=0–1) (Table 1311 p. 1445) or the AES Decryption Command Register (p=0–1) (Table 1324 p. 1448).

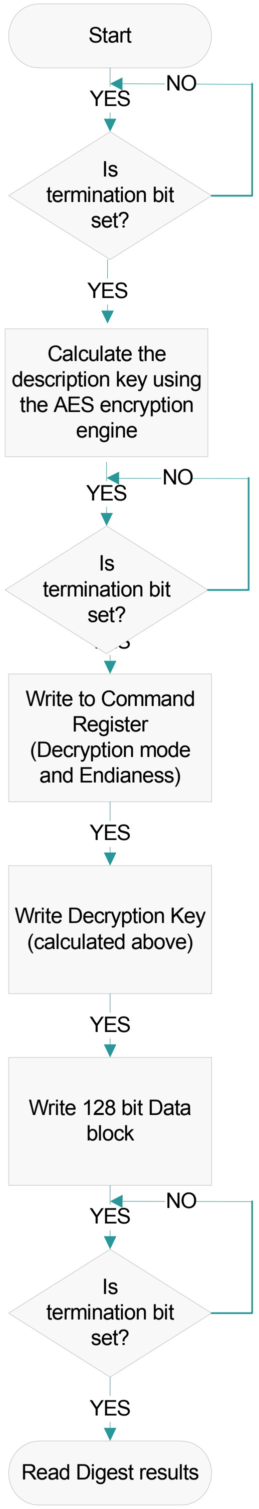
   The AES Decryption Key registers (see Table 1312 on page 1446 through Table 1319 on page 1447) contain the AES key block for the decryption engine. A read to these registers returns the last key written there by the host. The host must load a pre-calculated decryption key to these registers (as described previously, see “Calculate decryption key” on page 472).

   NOTE: Directly loading an encryption key to these registers returns incorrect results!

   The AES Decryption Command Register (p=0–1) (Table 1324 p. 1448) controls the AES decryption operation. It also flags when the processing is complete.

Figure 123 shows a typical authentication flow for a packet.

Figure 123:Typical AES Decryption Flow for a Data Block


Start
NO
YES
Is termination bit set?
YES
Calculate the description key using the AES encryption engine
NO
YES
Is termination bit set?
Write to Command Register (Decryption mode and Endianess)
YES
Write Decryption Key (calculated above)
YES
Write 128 bit Data block
NO
YES
Is termination bit set?
YES
Read Digest results

30.4 Functional Description—TDMA Engine

The device has one independent TDMA engine. The TDMA engine optimizes system performance by moving large amounts of data without significant CPU intervention.

The TDMA engine can move data between the main memory and the internal SRAM, and from the internal SRAM to the main memory. It can transfer a single data buffer of up to 2 KB. It can also run in Chain mode. That mode assigns a unique descriptor to each buffer.

As long as TDMA is active, any software read access to the Security Accelerator internal SRAM is forbidden. A software read access is not expected. See the software flow description in Section 30.5.2, Software Flow Chart, on page 480.

When fetching data into the SRAM, the data is read from the source through the Mbus and written directly into the internal SRAM.

When storing data from the SRAM, the data is read from the SRAM and written to the main memory through the Mbus.

30.4.1 TDMA Descriptors

The TDMA Descriptor consists of four 32-bit registers (see Figure 124 and Table 130).

Figure 124:TDMA Descriptors

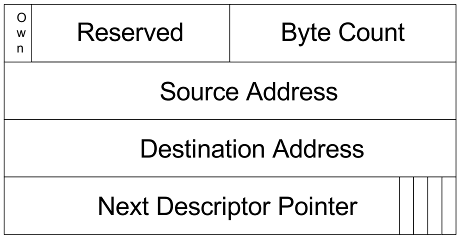


Table 130: TDMA Descriptor Definitions

TDMA Descriptor	Definition
Byte Count	Number of bytes of data to transfer. The maximum number of bytes to which the TDMA controller can be configured to transfer is 2 KB (12-bit register). This register decrements at the end of every burst of transmitted data from the source to the destination. When the byte count register is 0, the TDMA transaction is finished or terminated.
Source Address	Bits [31:0] of the TDMA source address.
Destination Address	Bits [31:0] of the TDMA destination address.
Pointer to the Next Descriptor	Bits [31:0] of the TDMA Next Descriptor address for chained operation. The descriptor must be 16 sequential bytes located at 16-bytes aligned address (bits [3:0] are 0). **NOTE:** This descriptor is used only used when the TDMA is configured to Chained mode.

Note

The source or destination address must be configured to the internal SRAM address space.

30.4.2 TDMA Address Decoding

The TDMA shares four address windows. Each address window can be individually configured.

For each TDMA transaction, the TDMA engine first compares the address (source, destination, or descriptor) against its address decoding registers. Each window can be configured to a different target interface. Address comparison is done to select the correct target interface.

30.4.3 TDMA Control

The TDMA has its own unique control register, where certain TDMA modes are programmed. The following are the bit descriptions for each field in the control registers.

30.4.3.1 Burst Limit

The TDMA byte count is chopped into small bursts.

The burst limit can be 32 or 128 bytes. The limit determines the burst length of the TDMA transaction against the source and destination. There are separate Burst Limit parameters for source and destination.

The burst limit setting is affected by the source and destination characteristics, as well as system bandwidth allocation considerations.

Note

Regardless of the burst limit setting, the fetch of a new descriptor is always a 16-byte burst. This implies that descriptors cannot be located in devices that do not support such bursts.

30.4.3.2 Chain Mode

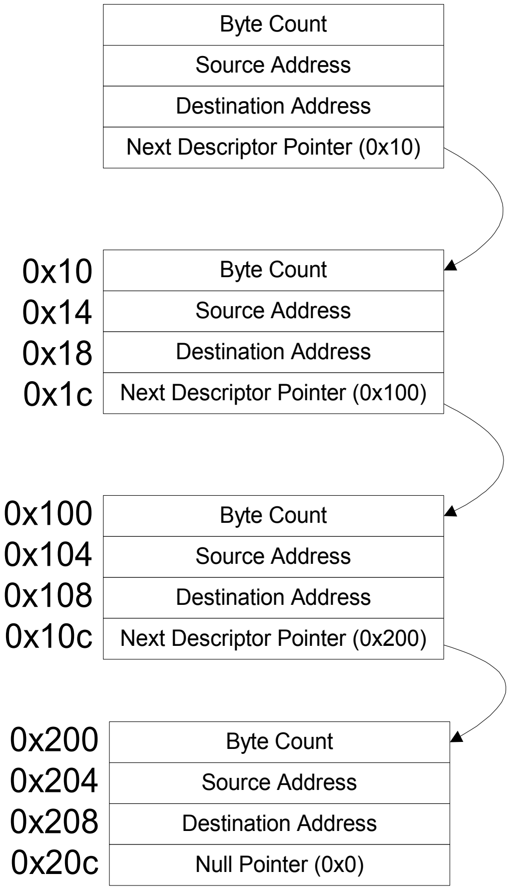
When the <TdmaChainMode> field in the Control Register (p=0–1) (Table 1339 p. 1457) is cleared to 0, Chained mode is enabled.

In Chain mode, at the completion of one buffer transfer, the Pointer to Next Descriptor provides the address of the next TDMA descriptor. If it is a NULL pointer (value of 0), it indicates that this is the last descriptor in the chain. If not, the TDMA engine fetches the new descriptor, and starts transferring the new buffer.

Copyright © 2014 Marvell

Figure 125 shows an example of a TDMA descriptors chain.

Figure 125:Chain Mode TDMA



Fetch next descriptor can be forced by bit [13], the <FetchND> field in the Control Register (p=0–1) (Table 1339 p. 1457).

Setting the <FetchND> to 1 forces a fetch of the next descriptor based on the value in the Pointer to Next Descriptor register. This bit is cleared to 0 after the fetch of the new descriptor is complete. Setting <FetchND> is not allowed if the next descriptor pointer equals Null.

The first descriptor of a chain can be set directly by programming the TDMA registers, or can be fetched from memory, using the <FetchND> bit. If fetched from memory, the next descriptor address must be first written to the Next Descriptor Pointer register of the TDMA. The TDMA then must be enabled by setting bit [12] the <TDMAEn> field in the Control Register (p=0–1) (Table 1339 p. 1457) to 1 (see Section 30.4.3.3, TDMA Activation, on page 478) and setting the <FetchND> field to 1.

When the TDMA transfer is completed, a TDMA completion interrupt is set. When running in chain mode, an interrupt is asserted only upon the completion of the last descriptor byte count.

If the <TdmaChainMode> field in the Control Register (p=0–1) (Table 1339 p. 1457) is set to 1, Chained mode is disabled and the Pointer to Next Descriptor register is not loaded at the completion of the TDMA transaction.

Copyright © 2014 Marvell

Note

In non-chained mode, the Byte Count, Source, and Destination registers must be initialized prior to enabling the TDMA.

30.4.3.3 TDMA Activation

Software TDMA activation is done via the <TDMAEn> field in the Control Register (p=0–1) (Table 1339 p. 1457) as follows:

- When cleared to 0, the TDMA is disabled.
- When set to 1, the TDMA is initiated based on the current setting loaded in the TDMA descriptor (i.e., byte count, source address, and destination address).

An active TDMA can be temporarily stopped by clearing the <TDMAEn> bit and then the active TDMA can be continued from the point it was stopped by setting <TDMAEn> bit back to 1.

Clearing the <TDMAEn> bit during a TDMA operation does not guarantee an immediate TDMA pause. The TDMA engine must complete transferring the last burst it was working on. Software can monitor the TDMA status by reading <TDMA Act>, bit [14].

The <TDMA Act> bit is read only.

- If cleared to 0, the TDMA is not active.
- If set to 1, the TDMA is active.

In Non-chain mode, this bit is de-asserted when the byte count reaches zero.
In Chain mode, this bit is de-asserted when the pointer to the next descriptor is NULL and byte count reaches zero.

30.4.3.4 Source and Destination Addresses Alignment

The TDMA implementation maintains aligned accesses to both source and destination.

If source and destination addresses have different alignments, the TDMA performs multiple reads from the source to execute a write of full burst limit to the destination. For example, if the source address is 0x64 and the destination address is the internal SRAM:

1. The Burst limit of the source is set to 32 bytes.
2. The byte count is 64.
3. The TDMA perform three reads from the source:
   a) 28 bytes from address 0x64 (due to the Mbus transfer limit).
   b) 32 bytes from address 0x80 (due to the source burst limit).
   c) 4 bytes from address 0xA0.

This implementation guarantees that all reads from the source and all writes to the destination have all byte enables asserted (except for the buffer start/end, in case they are not aligned). This is especially important when the source device does not tolerate reads of extra data (destructive reads) or when the destination device does not support write byte enables.

30.4.3.5 Descriptor Ownership

A typical application of chain mode TDMA involves the CPU core preparing a chain of descriptors in memory and then preparing buffers to move the descriptors from source to destination.

The <Own> field in the TDMA Byte Count Register (p=0–1) (Table 1340 p. 1458) (bit [31]) acts as an ownership bit.

- If set to 1, the descriptor is owned by the device TDMA.
- If cleared to 0, it is owned by the CPU. Once the CPU prepares a buffer to be transferred, it sets the ownership bit. This indicates that the buffer is owned by the TDMA.

Copyright © 2014 Marvell

An attempt by the TDMA to fetch a descriptor that is owned by the CPU (which means the CPU did not prepare a new buffer yet) results in an interrupt assertion, and the TDMA stops.

30.4.4 TDMA Interrupts

The TDMA interrupts are registered in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1450) register. Upon an interrupt event, the corresponding cause bit is set to 1. It is cleared upon a software write of 0.

The following interrupt events are supported:

- TDMA completion
- TDMA descriptor ownership violation

30.5 Functional Description—Security Accelerator

The Security Accelerator is activated by the CPU core. It operates on top of the cryptographic engines and sets them to operate properly. The entire operation of the accelerator works with the data and descriptors in the internal SRAM as follows:

CPU

The CPU core performs the following:

1. Prepares TDMA descriptors to copy data to the SRAM, or copies data directly into the SRAM.
2. Prepares Security Accelerator descriptors, which can be placed with the TDMA descriptors, stating the required operation, see Section 30.5.3, Security Accelerator Descriptor Data Structure, on page 484.
3. Activates the accelerator.

Security Accelerator

The Security Accelerator performs the following:

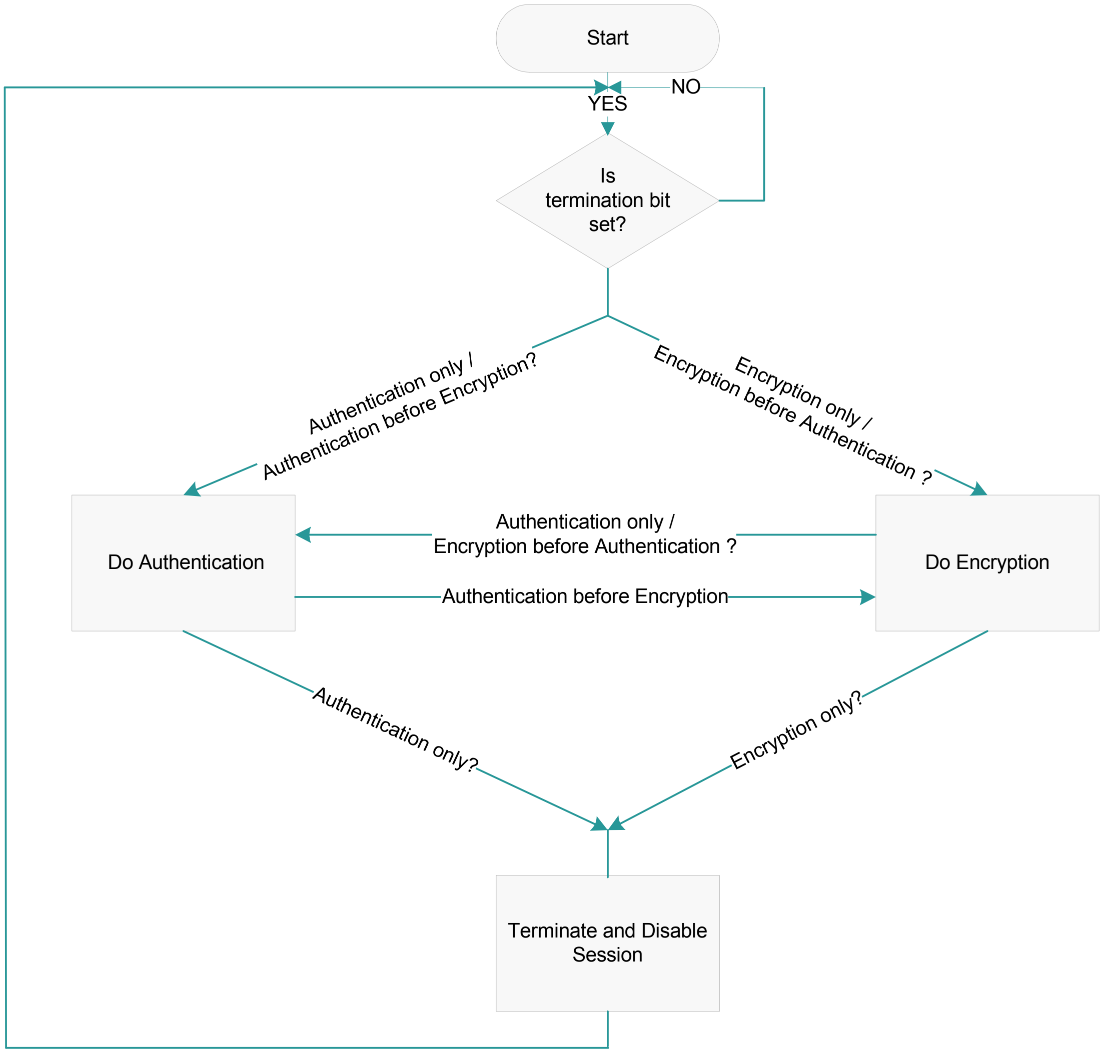
1. Reads the descriptor.
2. Sets up the engines.
3. Starts feeding the packet data into the engine one data block at a time.
4. Waits for completion.
5. Reads the data from the engine.
6. Stores the data in the Security Accelerator internal SRAM.

This procedure is performed until the packet is processed.

30.5.1 Hardware Flow Chart

Figure 126 illustrates the main accelerator decision flow.

Figure 126:Security Accelerator Main Decision Flow


Start
NO
YES
Is termination bit set?
Authentication only / Authentication before Encryption?
Encryption only / Encryption before Authentication ?
Do Authentication
Authentication only / Encryption before Authentication ?
Authentication before Encryption
Do Encryption
Authentication only?
Encryption only?
Terminate and Disable Session

30.5.2 Software Flow Chart

The managing software performs the following:

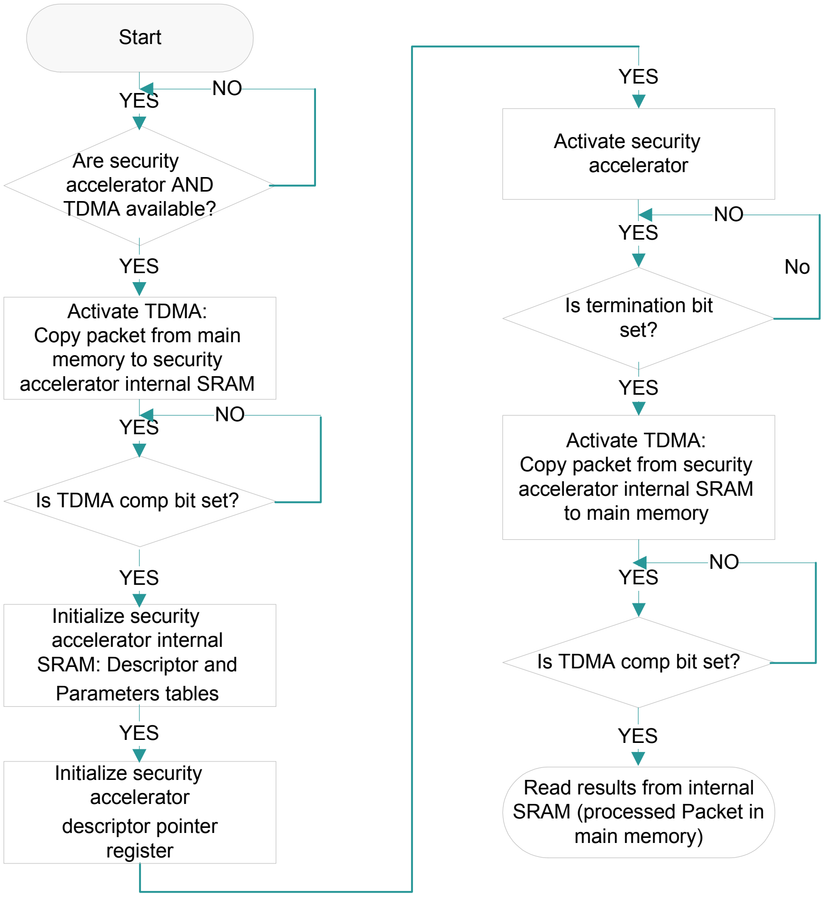
1. Uses the TDMA to copy the packet from main memory into the Security Accelerator internal SRAM. Since the Security Accelerator internal SRAM is only 2 KB, it cannot contain large packets.
2. Stores the cryptographic relevant parameters for the Security Accelerator operation to be undergone by the packet.

Copyright © 2014 Marvell

3. Prepares a descriptor in the Security Accelerator internal SRAM stating the required operation and parameters for the accelerator. The descriptor, which is 8 DWORDs long, is described in Section 30.5.3 "Security Accelerator Descriptor Data Structure".
4. Writes the pointer to this descriptor into the selected session, the <SecurityAcclDescPtr0> field in the Security Accelerator Descriptor Pointer Register (p=0–1) (Table 1359 p. 1463).
5. Activates the programmed session by setting the appropriate bit in the Security Accelerator Command Register (p=0–1) (Table 1358 p. 1463).
6. Waits for the session completion indication either by polling the Security Accelerator Status Register (p=0–1) (Table 1361 p. 1465) or by interrupt.
7. Uses the TDMA to copy the processed packet back into main memory.

Figure 127 illustrates the security acceleration flow for packet processing and Figure 128 illustrates the enhanced mode.

Figure 127:Security Acceleration Flow for Packet Processing


Start
NO
YES
Are security accelerator AND TDMA available?
YES
Activate TDMA: Copy packet from main memory to security accelerator internal SRAM
NO
YES
Is TDMA comp bit set?
YES
Initialize security accelerator internal SRAM: Descriptor and Parameters tables
YES
Initialize security accelerator descriptor pointer register
YES
Activate security accelerator
NO
YES
No
Is termination bit set?
YES
Activate TDMA: Copy packet from security accelerator internal SRAM to main memory
NO
YES
Is TDMA comp bit set?
YES
Read results from internal SRAM (processed Packet in main memory)

30.5.2.1 Encryption Operation

Initialization

Initialize the encryption operation as follows:

1. Reads Encryption mode from the Security Accelerator Data Structure DWORD 0—Configuration (Table 131 p. 484) and the configuration register of the selected cryptographic engine is written.
2. Reads the source and destination pointers from the Security Accelerator Data Structure DWORD 1—Encryption Pointers (Table 132 p. 485).
3. Reads the number of bytes that should be encrypted from the Security Accelerator Data Structure DWORD 2— Encryption Data Length (Table 133 p. 485).
4. Reads the keys for encryption from the pointer specified in Security Accelerator Data Structure DWORD 3—Encryption Keys Pointer (Table 134 p. 486) and then writes to the selected cryptographic engine.
5. When the mode selected is CBC, reads initial values from the pointer specified in Security Accelerator Data Structure DWORD 4—Encryption Initial Values Pointer (Table 135 p. 486). For DES/3DES,writes initial values to the engine.

Data Processing

Data processing must implement the following steps:

1. Reads data from the source pointer block by block for the specified data size (8 bytes for DES/3DES, 16 bytes for AES). If the size is not a multiple of block size, the last block is padded with zeros.
2. Feeds each block to the engine. When the engine finishes processing a block, the result is read and written to the location specified by the destination pointer.
3. When using AES CBC encryption, XORs the first block data with the initial values before the writing data to the engine. Each of the following blocks is first XORed with the result of the previous block processing before it is written to the engine.
4. When using AES CBC decryption, XORs the result of the first block processing with the initial values before writing it to the destination. Then, each processed block is XORed with the previous block of data before it is written to destination.

Termination

Termination is carried out as follows:

1. When using CBC encryption, writes the last block of the destination data to memory, according to the pointer specified in the Security Accelerator Data Structure DWORD 4—Encryption Initial Values Pointer, overwriting the previous data.
2. When using CBC decryption, writes the last block of source data to memory, according to the pointer specified in the Security Accelerator Data Structure DWORD 4—Encryption Initial Values Pointer.

30.5.2.2 Authentication Operation

Initialization

Initialize the authentication operation as follows:

1. Reads Authentication mode from the Security Accelerator Data Structure DWORD 0—Configuration (Table 131 p. 484) and writes the configuration register of the Authentication engine.
2. Reads the source pointer from the Security Accelerator Data Structure DWORD 5—MAC Source Pointer (Table 136 p. 486).

3. Reads the Digest location pointer and the number of bytes in the message from the Security Accelerator Data Structure DWORD 6—MAC Digest (Table 137 p. 487).
4. When the direction is “Decode”, reads the original digest first and stores it internally, then overwrites it with zeros.
5. When using the Hash-based Message Authentication Code (HMAC) operation, reads the inner initial values pointer from the Security Accelerator Data Structure DWORD 7—MAC Initial Values Pointers (Table 138 p. 487), and writes the values to the IV registers of the authentication engine.

Data Processing

Data processing is carried out as follows:

1. Reads data from the source pointer block-by-block (64 bytes per block).
2. Pads the last chunk of data with a single 1 bit, as many zeros as needed, and the original message length. For HMAC modes, packet length is increased by 64 bytes to reflect the ipad string length.
3. For HMAC modes, reads the outer initial values pointer from the Security Accelerator Data Structure DWORD 7—MAC Initial Values Pointers (Table 138 p. 487), and writes the values to the IV registers of the Authentication engine. The digest from the previous section is now used as the input data to the engine, padded again as specified, with the packet length now equal to the key length plus 64 bytes (which now reflects the opad string length).

Termination

Termination is carried out as follows:

1. Reads the resulting digest from the engine, and stores it at the location that has been read from the Security Accelerator Data Structure DWORD 6—MAC Digest.
2. When the direction is “Decode”, the digest is also compared to the copy that was extracted from the original message.
   - If the digests are not identical, an indication bit is set in the Security Accelerator Status Register (p=0–1) (Table 1361 p. 1465).
   - If the <StopOnDecodeDigestErr> field in the Security Accelerator Configuration Register (p=0–1) (Table 1360 p. 1464) was set, the session is immediately aborted.

30.5.2.3 Security Accelerator in Continuous Mode

Continuous mode

The Security Accelerator supports Continuous mode. To enable this mode, set the <MultiPacketChainMode> field in the Security Accelerator Configuration Register (p=0–1) (Table 1360 p. 1464) to 1.

In this mode, the Security Accelerator can process fragmented packets one at a time. Only single packets need to reside in the Security Accelerator internal SRAM, while the total size of the packet is limited to 64 KB.

Field Fragmentation mode

Field Fragmentation mode is used to indicate if the current fragment is the first, middle, or last in the packet.

The fragments must be inserted in order.

Processing the First Fragment

The first fragment is processed as follows:

1. The operation starts in the same manner as in Non-fragmented mode. Finalization is not performed.
2. In encryption, IV is not stored.

3. In authentication, data is not padded, the outer operation in HMAC does not occur, and the digest is not written or compared.

Processing Middle Fragments (Not First and Not Last)

The middle fragments are processed as follows:

1. Initializations are not done.
2. In encryption, keys and IVs are not loaded into the engines.
3. In authentication, the digest is not read or cleared. Inner operation of HMAC is not done. Finalization is not done, as for the first fragment.

Processing the Last fragment

The last fragment is processed as follows:

1. Initializations are not completed, as for a middle fragment.
2. Finalization is completed, as in Non-fragmented mode.

30.5.3 Security Accelerator Descriptor Data Structure

The descriptor data structure is described in Table 131 through Table 138.

Table 131: Security Accelerator Data Structure DWORD 0—Configuration

Bits	Field	Function
1:0	Operation	00 = MAC only 01 = Cryptographic only 10 = MAC then cryptographic 11 = Cryptographic then MAC
3:2	Reserved	Reserved Must be 0.
6:4	MacMode	001 = SHA-2 011 = HMAC-SHA-2 100 = MD5 101 = SHA-1 110 = HMAC-MD5 111 = HMAC-SHA-1 All other combinations are reserved (no operation).
7	AuthResultLen	Authentication result length 0 = Full size (128 bit in MD-5, 160b in SHA-1, 256b in SHA-2) 1 = 96b(The 96b result length is supported for SHA-1 and MD5 only.)
9:8	EncryptMode	00 = Reserved (no operation) 01 = DES 10 = 3DES 11 = AES
11:10	Reserved	Reserved Must be 0.
12	Direction	0 = Encode 1 = Decode
15:13	Reserved	Reserved Must be 0.

Table 131: Security Accelerator Data Structure DWORD 0—Configuration

Bits	Field	Function
16	EncryptConfidentialityMode	0 = ECB 1 = CBC
19:17	Reserved	Reserved Must be 0.
20	3DESMode	0 = EEE 1 = EDE Relevant only in 3DES encryption mode.
23:21	Reserved	Reserved Must be 0.
25:24	AESKeyLength	00 = 128-bits 01 = 192-bits 10 = 256-bits
29:26	Reserved	Reserved Must be 0.
31:30	FragMode	Fragmentation mode 00 = Not fragmented 01 = First Fragment in packet 10 = Last Fragment in packet 11 = Middle Fragment in packet

Table 132: Security Accelerator Data Structure DWORD 1—Encryption Pointers

Bits	Field	Function
10:0	EncrypSourceDataPtr	Pointer to the first DWORD of data to encrypt (DWORD aligned) Bits [2:0] must be 0.
15:11	Reserved	Reserved Must be 0.
26:16	EncrypDesDataPtr	Pointer to the first DWORD of encrypted data (DWORD aligned) Bits [18:16] must be 0.
31:27	Reserved	Reserved Must be 0.

Table 133: Security Accelerator Data Structure DWORD 2— Encryption Data Length

Bits	Field	Function
10:0	EncrypDataLen	Numbers of bytes to encrypt The length should be a multiple of encryption block size (8 bytes for DES/3DES, 16 bytes for AES). Bits [2:0] must be 0. Bit [3] (in AES only) is reserved and assumed to be 0 regardless of programming.
31:11	Reserved	Reserved Must be 0.

Table 134: Security Accelerator Data Structure DWORD 3—Encryption Keys Pointer

Bits	Field	Function

10:0	EncrypKeyPointer	Pointer to an array (EKey) of DWORDs that contains the encryption key (DWORD aligned)
		EKey[0] = Key 0 low of DES/3DES/Key column 0 of AES 128/192/256
		EKey[1] = Key 0 high of DES/3DES/Key column 1 of AES 128/192/256
		EKey[2] = Key 1 low of 3DES/Key column 2 of AES 128/192/256
		EKey[3] = Key 1 high of 3DES/Key column 3 of AES 128/192/256
		EKey[4] = Key 2 low of 3DES/Key column 4 of AES 192/256
		EKey[5] = Key 2 high of 3DES/Key column 5 of AES 192/256
		EKey[6] = Key column 6 of AES 256
		EKey[7] = Key column 7 of AES 256
		Bits [2:0] must be 0.

31:11	Reserved	Reserved Must be 0.

Table 135: Security Accelerator Data Structure DWORD 4—Encryption Initial Values Pointer

Bits	Field	Function

10:0	EncryptIVPointer	Pointer to an array (EIV) of DWORDs that contains the encryption initial values (DWORD aligned)
		EIV[0] = IV low of DES/3DES/IV 0 of AES
		EIV[1] = IV high of DES/3DES/IV 1 of AES
		EIV[2] = IV 2 of AES
		EIV[3] = IV 3 of AES

15:11	Reserved	Reserved Must be 0.
26:16	EncryptIVBufPointer	• In Encryption mode, in the encryption direction: Before encryption starts, the Security Accelerator copies the contents of <EncryptIVPointer> (bit[15:0] in same register) to <EncryptIVBufPointer>. • In Encryption mode, in the decryption direction: Before decryption starts, the Security Accelerator copies the contents of <EncryptIVBufPointer> to <EncryptIVPointer>. Bits [18:16] must be 0.
31:27	Reserved	Reserved Must be 0.

Table 136: Security Accelerator Data Structure DWORD 5—MAC Source Pointer

Bits	Field	Function
10:0	MACSourceDataPointer	Pointer to the first DWORD of data to MAC (DWORD aligned) Bits [2:0] must be 0.
15:11	Reserved	Reserved Must be 0.

Copyright © 2014 Marvell

Table 136: Security Accelerator Data Structure DWORD 5—MAC Source Pointer

Bits	Field	Function
31:16	TotalMacDataLength	In MAC Non Fragment mode, this field should be equal to MacDataLength (see Table 137). In the last MAC fragment, it should be equal to the total data lengths of all the packet fragments.

Table 137: Security Accelerator Data Structure DWORD 6—MAC Digest

Bits	Field	Function
10:0	MACDigestPointer	Byte location in which digest is stored during encoding or should be stored during decoding Bits [2:0] must be 0.
15:11	Reserved	Reserved Must be 0.
26:16	MACDataLength	Numbers of bytes to MAC In the first and middle fragments, the MAC data length should be aligned to 16 words. Bits [5:0] must be 0.
31:27	Reserved	Reserved Must be 0.

Table 138: Security Accelerator Data Structure DWORD 7—MAC Initial Values Pointers

Bits	Field	Function

10:0	MACInnerIVPointer	Pointer to an array (MIIV) of DWORDs that contains the MAC inner initial values (DWORD aligned)
		These values are the outcome of the hash function operation over the 64-byte string that equals the bitwise XOR between the key padded with zeros and the ipad string.
		MIIV[0] = Inner IV 0 of HMAC-MD5/HMAC-SHA-1
		MIIV[1] = Inner IV 1 of HMAC-MD5/HMAC-SHA-1
		MIIV[2] = Inner IV 2 of HMAC-MD5/HMAC-SHA-1
		MIIV[3] = Inner IV 3 of HMAC-MD5/HMAC-SHA-1
		MIIV[4] = Inner IV 4 of HMAC-SHA-1
		Bits [2:0] must be 0.

15:13	Reserved	Reserved Must be 0.

26:16	MACOuterIVPointer	Pointer to an array (MOIV) of DWORDs that contains the MAC outer initial values (DWORD aligned). These values are the outcome of the hash function operation over the 64-byte string that equals the bitwise XOR between the key padded with zeros and the opad string.
		MOIV[0] = Outer IV 0 of HMAC-MD5/HMAC-SHA-1
		MOIV[1] = Outer IV 1 of HMAC-MD5/HMAC-SHA-1
		MOIV[2] = Outer IV 2 of HMAC-MD5/HMAC-SHA-1
		MOIV[3] = Outer IV 3 of HMAC-MD5/HMAC-SHA-1
		MOIV[4] = Outer IV 4 of HMAC-SHA-1
		Bits [18:16] must be 0.

Table 138: Security Accelerator Data Structure DWORD 7—MAC Initial Values Pointers (Continued)

Bits	Field	Function
31:27	Reserved	Reserved Must be 0.

30.6 Enhanced Flow Description

The Enhanced Flow for the Cryptographic Engine attaches the TDMA to the Security Accelerator on a single packet basis or by a multi-packet chain.

30.6.1 Single Packet Mode

To perform the following flow, set the <WaitForTDMA> field and the <ActivateTDMA> field in the Security Accelerator Configuration Register (p=0–1) (Table 1360 p. 1464).

When the Security Accelerator operates in conjunction with the TDMA, the Security Accelerator can operate solely with external main memory. As indicated by the flow shown in Figure 128, the Security Accelerator has the capability to activate the TDMA, determine the status of the TDMA, and provide a single completion interrupt. The first five steps in the flow are performed by the software and the remaining steps are performed by the hardware.

Figure 128:Security Acceleration Flow for Packet Processing—Enhanced Mode

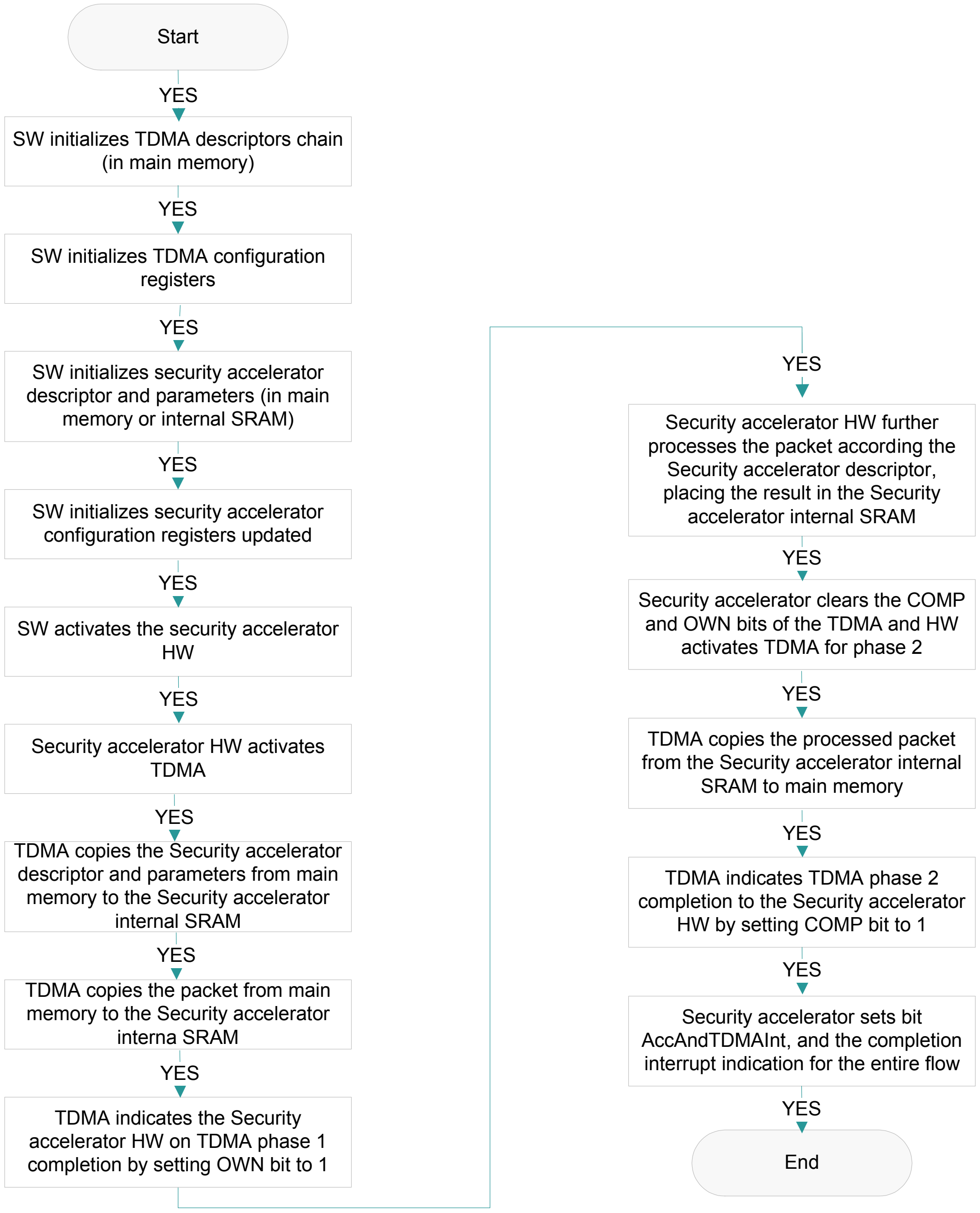
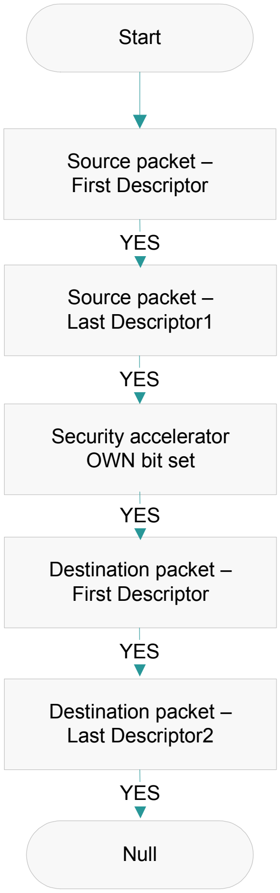

Start
YES
SW initializes TDMA descriptors chain (in main memory)
YES
SW initializes TDMA configuration registers
YES
SW initializes security accelerator descriptor and parameters (in main memory or internal SRAM)
YES
SW initializes security accelerator configuration registers updated
YES
SW activates the security accelerator HW
YES
Security accelerator HW activates TDMA
YES
TDMA copies the Security accelerator descriptor and parameters from main memory to the Security accelerator internal SRAM
YES
TDMA copies the packet from main memory to the Security accelerator interna SRAM
YES
TDMA indicates the Security accelerator HW on TDMA phase 1 completion by setting OWN bit to 1
YES
Security accelerator HW further processes the packet according the Security accelerator descriptor, placing the result in the Security accelerator internal SRAM
YES
Security accelerator clears the COMP and OWN bits of the TDMA and HW activates TDMA for phase 2
YES
TDMA copies the processed packet from the Security accelerator internal SRAM to main memory
YES
TDMA indicates TDMA phase 2 completion to the Security accelerator HW by setting COMP bit to 1
YES
Security accelerator sets bit AccAndTDMAInt, and the completion interrupt indication for the entire flow
YES
End

Figure 129 depicts the TDMA descriptors structure for Security Accelerator packet processing when in the Security Accelerator is operating in enhanced mode.

Figure 129:TDMA Descriptors Structure for Security Accelerator Packet Processing—Enhanced Mode



[1] This step only applies if there is more than one source packet descriptor.

[2] This step applies if there is more than one destination packet descriptor.

30.6.2 Multi-Packet Chain Mode

The device Security Accelerator also supports Multi-Packet Chain mode. In this mode, multiple packets can be chained and processed by the hardware without software interference. To enable this mode, set the <MultiPacketChainMode> field in the Security Accelerator Configuration Register (p=0–1) (Table 1360 p. 1464) to 1.

In Multi-Packet Chain mode, the software prepares multiple packets in memory. For each packet that needs to be processed, the software also prepares a set of descriptors as described in Figure 129, TDMA Descriptors Structure for Security Accelerator Packet Processing—Enhanced Mode, on page 490 except for a small modification. Instead of a NULL pointer at the last descriptor of the destination packet, the software includes a pointer to the next source packet's first descriptor.

Copyright © 2014 Marvell

If the Multi-Packet Chain mode is enabled, after writing the encrypted packet N back to memory:

1. The TDMA fetches the first descriptor of source packet N+1.
2. The Security Accelerator sets the <AccAndTDMAInt_CM> field in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1451).
3. When there are no more packets to process (meaning, the destination packet last descriptor points to a NULL pointer), the Security Accelerator halts, and the <AccAndTDMAInt> field in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1451) is set.

30.7 Interrupt Coalescing

The Multi-Packet Chain mode execution can be accomplished with or without using the coalescing mechanism.

If the interrupt coalescing mechanism is used, initialize the following registers:

- Cryptographic Interrupt Coalescing Threshold Register (p=0–1) (Table 1329 p. 1452)
- Cryptographic Interrupt Time Threshold Register (p=0–1) (Table 1330 p. 1453)

If Interrupt coalescing mechanism is not used, the <EopCoalInt> field in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1451) should be masked.

Note

The use of the <EopCoalInt>and <AccAndTDMAInt> is mutually exclusive.

The Security Accelerator and TDMA engines provide a high data rate, it is important to reduce the number of interrupts that the TDMA may generate. The device provides an interrupt coalescing mechanism that sets the interrupt coalescing bit in the Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Table 1327 p. 1450), and propagates an interrupt indication if one of the following is true:

- The number of TDMA commands reached the Cryptographic interrupt coalescing threshold value (see the Cryptographic Interrupt Coalescing Threshold Register (p=0–1) (Table 1329 p. 1452)).
- At least one TDMA commands has completed and the time that passed since its completion reached the Cryptographic interrupt time threshold value (see the Cryptographic Interrupt Time Threshold Register (p=0–1)).

31 XOR Engines

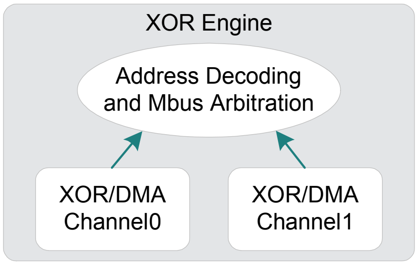
This device integrates 2 dual-channel XOR engines (overall 4 XOR DMA channels). This section describes a single channel XOR engine, but applies for all other channels/engines as well.

The XOR engine is a generic acceleration engine for storage applications that provides a low latency, high throughput XOR calculation capabilities, enabling CPU XOR calculation offloading in various RAID implementations. In addition, the engine provides DMA (Memory copy) functionality, iSCSI CRC32C calculation, memory ECC errors cleanup operation and memory initialization (Memory set, Memory clear) support.

The XOR engine enables to offload or accelerate the CPU and speeds up overall system performance in Networking and Storage applications.

Figure 130 presents an XOR engine design with 2 channels.

Figure 130:XOR Engine Block Diagram



The XOR engine registers are located in Appendix A.22, XOR Registers, on page 1471.

31.1 Features

- Two separate channels for enabling concurrent operation (for example, concurrent XOR and iSCSI CRC32C calculations)
- 512-byte buffer per channel
- Support unaligned data transfers
- XOR calculation for up to eight data block sources
- Data block size up to 16 MB
- Programmable maximum burst size on read and write
- Descriptor chain mechanism
- Hot insertion of new descriptors to chain
- iSCSI CRC32C calculation that is compliant with IPS iSCSI version 13 draft
- DMA operation
- Memory initialization support
- Memory ECC cleanup support
- Write access protection of configuration registers

Copyright © 2014 Marvell

31.2 Functional Description

The XOR engine has 5 main operation modes:

- XOR calculation Mode (XOR)
- iSCSI CRC32C Calculation Mode (CRC)
- DMA Operation Mode (DMA)
- Memory initialization Mode (MemInit)
- Memory ECC error cleanup mode (ECC)

The engine has 2 independent channels. Each channel can be configured in 2 ways.

- One way is to have each channel perform a single operation mode at a time. The operation mode is defined through the <OperationMode> field in the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1479). In this mode the XOR engine can perform the following operations: XOR, CRC and DMA.
- The other way is to have the mode of operation encoded in each of the descriptors in the chain. this way each chain of descriptors can be used to perform different operations, not limiting the use of one operation mode per one channel. The operation mode is defined when the <OperationMode> field in the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1479) is set to *Descriptor*. In this mode the XOR engine can perform the following operations: XOR, CRC, DMA.

For XOR, CRC, DMA, the XOR engine is controlled by chain of descriptors and responds to similar activation scheme. These modes differ only in the interpretation of the chain descriptor fields. In ECC and Memory Initialization (MemInit) modes, the XOR engine responds to different activation schemes. It is controlled by programming internal registers directly. On all operation modes, XOR engine uses the same address decoding scheme.

Upon startup, the 2 XOR channels are in an inactive state and can be configured to any operation mode (XOR, CRC, DMA, ECC, MemInit or Descriptor). After being configured, the XOR engine channel can be activated. It can be stopped or paused by software at any time. After stopped by software, the engine re-enters inactive state and can be configured to another operation mode and re-activated. This also applies if the XOR engine channel finished the operation (reached End Of Chain) without being stopped by the software. Again, the engine re-enters an inactive state and can be configured to another operation mode, and re-activated. After paused by software, the XOR engine channel suspends the current operation at the earliest opportunity. Upon activating the channel again, it resumes executing the same operation.

The two XOR engine channels are independent in their operation modes. The only exception is that both engines must not be configured to ECC or MemInit operation modes. These modes share hardware resources.

Note

Attempting to change the channels operation mode during a pause results in unexpected behavior.

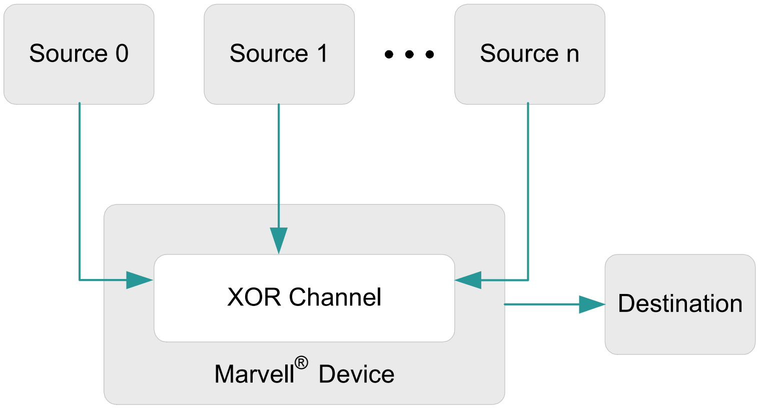
31.3 XOR Engine Modes of Operation

31.3.1 XOR Engine Operation

The XOR engine enables block XOR calculation in hardware. It performs the XOR operation on multiple blocks of source (incoming) data and stores the result back in a destination block. The source and destination addresses are specified through a chain descriptor.

Figure 131 shows how the XOR operation works with multiple blocks of source (incoming) data and stores the result back in a destination block.

Figure 131:XOR Operation with Multiple Incoming Data Blocks



The parameters of the XOR operation are configured by writing the relevant information to a chain descriptor. The relevant parameters consist of source addresses, destination addresses, the number of bytes to transfer, and various control information.

When activated in XOR mode, the XOR engine fetches the first descriptor and starts performing the XOR operation according to its parameters. After finishing the operation, the XOR engine closes the descriptor by writing back the status word to the descriptor and returning the ownership of it to the CPU. The XOR engine checks whether it reached the end of the descriptor chain. If it is the end, the engine enters an inactive state and waits to be re-activated by software. If it did not reach the end of the chain, it progresses to the next descriptor, and so on.

The basic XOR operation algorithm is as follows:

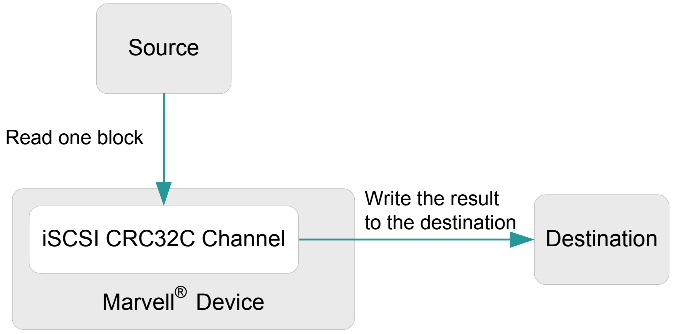
1. Read data from the first enabled source block to the internal buffer.
2. Read from the second enabled source block and calculate XOR with the data from the internal buffer. The intermediate result is stored in the internal buffer.
3. Step 2 is repeated for the rest of the source buffers until all enabled source buffers are handled (up to 8 sources).
4. Write the internal buffer to the destination buffer.
5. Repeat stages 1-4 until the descriptor byte count in is reached.

31.3.2 iSCSI CRC32C Calculation

In addition to the XOR operation, the XOR engine also provides iSCSI CRC32C calculation capabilities. It performs iSCSI CRC32C calculation on a source block and writes the result back to a descriptor, as shown in Figure 132.

Copyright © 2014 Marvell

Figure 132:XOR iSCSI CRC32C Operation



The source blocks are specified through a chain of descriptors. Parameters of the iSCSI CRC32C operation are configured in the same way as in XOR operation - writing the relevant information to a chain descriptor. The relevant parameters consist of source addresses, destination address, size of source block, and 'last block in CRC source chain' indication.

The CRC source block in CRC mode can be scattered over a few target blocks. It can be in non-consecutive memory spaces and even in different interfaces. The CRC Source block is represented by a source block chain of descriptors. Every descriptor represents one consecutive section of the CRC source block.

When activated in CRC mode (see the <OperationMode> field in the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1479) is set to CRC), XOR engine executes iSCSI CRC32C calculation according to the source block chain. The last descriptor in a source block chain is marked as 'last'. After the calculation is finished, the 32-bit result is written to the CRC source block chain's last descriptor.

The chain descriptor operation is slightly different in CRC-32 mode than in XOR mode. In XOR mode, every descriptor stands for a self contained XOR operation. In CRC mode, one CRC operation (one CRC source block) can be represented by a number of chain descriptors, thus enabling concatenation of a few data sources to one block, for CRC calculation. The descriptor chain is constructed of several source block chains.

In all buffers used for iSCSI CRC calculation, align the data size to 4B.

The Basic iSCSI CRC32C operation is as follows:

1. Read data from the source block to internal buffer.
2. Calculate iSCSI CRC32C on the internal buffer and store intermediate result.
3. Repeat step 1-2 for the rest of the source block, until all of the blocks are processed.
4. Repeat steps 1-3 for the rest of the blocks in the source block chain.
5. Write result to the last descriptor.

31.3.3 DMA Operation

The XOR engine also provides generic DMA capabilities—copying of a source block to a destination block. The source blocks are specified through a chain of descriptors. Parameters of the DMA operation are configured in the same way as in XOR operation - by writing the relevant information to a chain descriptor. The relevant parameters consist of source address, destination address, and size of source block.

When activated in DMA mode, the XOR engine fetches the first descriptor and starts performing the DMA operation according to its parameters. After finishing the operation, the XOR engine closes the descriptor by writing back status word to the descriptor and returning the ownership of it to the CPU. The XOR engine checks whether it reached the end of the descriptor chain. If the end has been reached, the engine enters an inactive state and waits to be re-activated by the software. If it did not reach the end of the chain, it progresses to the next descriptor, and so on.

31.3.4 Memory Initialization

The XOR engine provides memory value initialization capabilities. It performs writes of pre-defined values to a destination memory block. The destination address and block size are specified directly by internal registers. The relevant parameters consist of a destination address, an initial memory value, and a size of the destination block.

Note

Only one channel may be configured to MemInit mode at a time. If both channels are configured to MemInit mode, engine behavior is unpredictable.

When activated in MemInit mode, the XOR engine executes a memory initialization operation according to the relevant internal registers. It will write the 64-bit initial value, specified by the XOR Engine Initial Value Low (XEIVRL) Register (m=0–1) (Table 1391 p. 1484) and XOR Engine Initial Value High (XEIVRH) Register (m=0–1) (Table 1392 p. 1485), in a cyclical method to the destination block. Upon completion of the memory initialization operation, the XOR engine channel asserts the EOC interrupt.

31.3.5 Memory ECC Errors Cleanup (Scrubbing)

The XOR engine provides single bit ECC error cleanup capabilities. It scans for ECC errors on a pre-defined destination block in DRAM. If a single bit error is detected, it is fixed. If two or more errors are detected (non-correctable ECC errors), an interrupt is asserted by the memory controller.

The memory ECC error cleanup operation is based on the device DRAM Read-Modify-Write (RMW) feature. Upon a write request to memory, with all byte enables inactive, the DRAM controller initiates an automatic RMW operation, without changing the data. If, during the read portion of the RMW, the controller detects a single bit error (or no error at all), it corrects the data and writes it back to memory. In case of a double-bit ECC error detection, the DRAM controller asserts an interrupt.

Parameters of the memory ECC error cleanup operation are configured in the same way as in memory initialization operation—by writing the relevant information to internal registers. The relevant parameters consist of the destination address and the destination block size.

Note

Only one channel may be configured to ECC mode at a time. If both channels are configured to ECC mode, engine behavior is unexpected.

In addition, the ECC operation supports a timer mode that enables periodic activation of the ECC cleanup operation, to a small target block at a time. This avoids long periods of memory usage by the ECC operation that can interfere with the normal system operation. Moreover, ECC error cleanup can run in the background, without CPU involvement. Once the CPU configures the parameters for the ECC operation, the ECC error cleanup takes place periodically, without any CPU intervention. The destination block is divided into sections according to the <SectionSizeCtrl> field in the XOR Engine Timer Mode Control (XETMCR) Register (m=0–1) (Table 1388 p. 1483).

The period between cleanup of sequential sectors is controlled by the ECC timer—a 32-bit wide timer integrated in the XOR engine. With every expiration of the ECC timer, another section of the target block is cleaned. Both channel0 and channel1 are coupled to the ECC timer. The timer decrements with every TCLK cycle. Upon expiration, cleanup of the relevant sector is initiated. The timer issues a timer expiration interrupt, reloads itself to the programmed initial value, and initiates a new count down. Reads from the timer are done from the counter itself, while writes are done to its register. This means that read results are in the counter's real-time value. The timer is enabled only if one of the XOR engine channels is set to ECC timer mode operation.

If activated in timer mode, the XOR engine:

- Enables timer count-down
- Waits for timer expiration
- Cleans the first memory section
- Asserts EOD interrupt

The XOR engine then waits for the second expiration, cleans the second section and asserts another EOD interrupt, and so on. When the XOR engine reaches the end of the target block, it asserts an EOC interrupt and re-starts cleaning the first memory section. To stop the operation, set the <XEstop> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480).

If activated in non-timer mode, the whole target block is processed at one time and the XOR unit becomes inactive.

If the destination block is not 8 byte aligned, the XOR engine initiates write requests to the smallest 8 byte aligned block that contains the original block.

31.4 Descriptor Chain Format

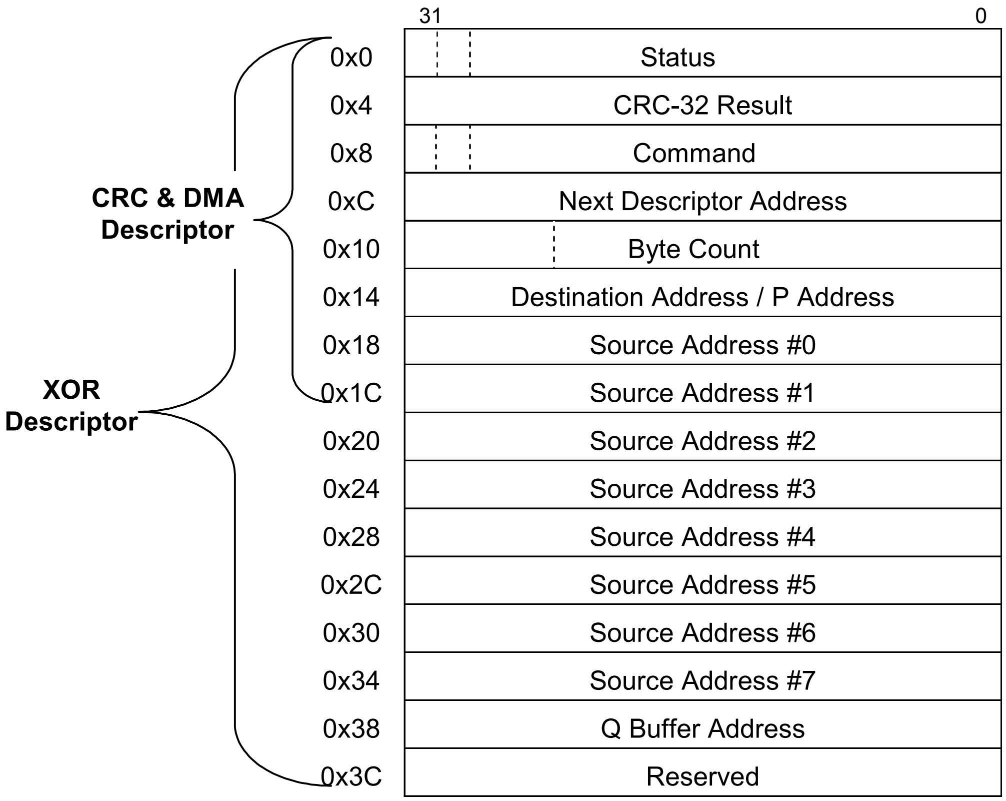
The XOR engine descriptor format supports 32-bit addressing. In XOR mode, the descriptor consists of sixteen 32-bit words which totals the 64B size of each descriptor. In CRC and DMA modes, only the upper 32B of the descriptor is needed. Therefore, the descriptor consists of eight 32-bit words, totalling to a 32B size for each descriptor (see Figure 133).

By fetching a descriptor from memory, the XOR engine gets all the information about the next operation to be performed. When the XOR engine finishes the operation associated with a descriptor, it closes the descriptor by updating the status word. This means the operation completed successfully and returns the ownership of the descriptor to the CPU.

Note

The chain descriptor operation is valid only in XOR, CRC, and DMA operation modes. In ECC and MemInit modes, the XOR engine receives the operation data directly from its internal registers.

Figure 133:XOR Descriptor Format



0x0	Status
0x4	CRC-32 Result
0x8	Command
0xC	Next Descriptor Address
0x10	Byte Count
0x14	Destination Address / P Address
0x18	Source Address #0
0x1C	Source Address #1
0x20	Source Address #2
0x24	Source Address #3
0x28	Source Address #4
0x2C	Source Address #5
0x30	Source Address #6
0x34	Source Address #7
0x38	Q Buffer Address
0x3C	Reserved

Data Alignment

- The XOR descriptors must be 64-byte aligned (Address[5:0]=0).
- The CRC and DMA descriptors must be 32-bytes aligned (Address[4:0]=0).
- There are no restrictions on source or destination data block alignment.
- Source and destination blocks can have different alignments.
- Different source blocks can have different alignments as well.

Copyright © 2014 Marvell

Table 139: Descriptor Status Word Definition

Bit	Field	Description
29:0	Reserved	Reserved.
30	Success	Successful descriptor execution indication. Indicates whether the operation completed successfully. 0 = Completed unsuccessfully. Transfer terminated before the whole byte count was transferred. 1 = Completed successfully. The whole byte count transferred. That field is updated upon closing the descriptor **NOTE:** In ECC cleanup mode the success bit indicates successful execution, even if ECC errors where found and not corrected.
31	Own	Ownership Bit Indicates whether the descriptor is owned by the CPU or the XOR engine. 0 = CPU owned. 1 = XOR engine owned. That field is updated upon closing a descriptor - XOR engine gives back ownership to the CPU by clearing the own bit.

Table 140: Descriptor CRC-32 Result Word Definition

Bit	Field	Description
31:0	CRCresult	Result of CRC-32 calculation Valid only in the last descriptor of a CRC source block chain, after it was closed by the XOR engine. **NOTE:** Valid only in CRC mode.

Table 141: Descriptor Command Word Definition

Bit	Field	Description
0	Src0Cmd	Specifies the type of operation to be carried out on the data pointed by SA#0 (Source Address 0 word of the descriptor). It is relevant only in XOR operation modes. It is disregarded in all other operation modes. 0x0 = Data from the source is disregarded in the current descriptor operation. 0x1 = Data from the source is transferred and is significant in the calculation.
1	Src1Cmd	Specifies the type of operation to be carried out on the data pointed by SA#1 (Source Address 1 word of the descriptor). It is relevant only in XOR operation modes. It is disregarded in all other operation modes. 0x0 = Data from source is disregarded in the current descriptor operation. 0x1 = Data from source is transferred and is significant in the calculation.

Table 141: Descriptor Command Word Definition (Continued)

Bit	Field	Description
2	Src2Cmd	Specifies the type of operation to be carried out on the data pointed by SA#2 (Source Address 2 word of the descriptor). It is relevant only in XOR operation mode. It is disregarded in all other operation modes. 0x0 = Data from source is disregarded in the current descriptor operation. 0x1 = Data from source is transferred and is significant in the calculation.
3	Src3Cmd	Specifies the type of operation to be carried out on the data pointed by SA#3 (Source Address 3 word of the descriptor). It is relevant only in XOR operation mode. It is disregarded in all other operation modes. 0x0 = Data from source is disregarded in the current descriptor operation. 0x1 = Data from source is transferred and is significant in the calculation.
4	Src4Cmd	Specifies the type of operation to be carried out on the data pointed by SA#4 (Source Address 4word of the descriptor). It is relevant only on XOR operation mode. Disregarded in all other operation modes. 0x0 = Data from Source is disregarded in the current descriptor operation. 0x1 = Data from source is transferred and is significant in the calculation.
5	Src5Cmd	Specifies the type of operation to be carried out on the data pointed by SA#5 (Source Address 5 word of the descriptor). It is relevant only on XOR operation mode. It is disregarded in all other operation modes. 0x0 = Data from source is disregarded in the current descriptor operation. 0x1 = Data from source is transferred and is significant in the calculation.
6	Src6Cmd	Specifies the type of operation to be carried out on the data pointed by SA#6 (Source Address 6 word of the descriptor). It is relevant only in XOR operation mode. It is disregarded in all other operation modes. 0x0 = Data from source is disregarded in the current descriptor operation. 0x1 = Data from source is transferred and is significant in the calculation.
7	Src7Cmd	Specifies the type of operation to be carried out on the data pointed by SA#7 (Source Address 7 word of the descriptor). It is relevant only on XOR operation mode. It is disregarded in all other operation modes. 0x0 = Data from source is disregarded in the current descriptor operation. 0x1 = Data from source will be transferred and is significant in the calculation.
11:10	Reserved	Reserved

Table 141: Descriptor Command Word Definition (Continued)

Bit	Field	Description
27:24	OperationMode	This field is valid only if the <OperationMode> field in the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1479) is configured to 0x7 (Descriptor). Specifies the type of operation to be carried out by XOR Engine. 0x0 = XOR calculate operation 0x1 = CRC-32 calculate operation 0x2 = DMA operation Other values are reserved.
29:28	Reserved	Reserved
30	CRCLast	Indicated last descriptor in a CRC-32 calculation chain. 0 = Not last descriptor in a CRC calculation chain. 1 = Last descriptor in a CRC calculation chain. When closing the descriptor, the XOR engine writes the CRC result to its CRC-32 Result word. The next descriptor in the descriptor chain initiates a new CRC calculation. If the source block is represented by one descriptor only, it should be marked as last. **NOTE:** Relevant only in CRC operation mode.
31	EODIntEn	End Of Descriptor Interrupt Enable. Specifies if the EOD interrupt is asserted upon closure of that descriptor. 1 = EOD enabled. 0 = EOD disabled.

Table 142: Descriptor Next Descriptor Address Word

Bits	Field	Description
31:0	NDA	Next descriptor address pointer XOR mode: NDA must be 64-byte aligned (bits[5:0] must be 0x0). CRC/DMA Mode: NDA must be 32-byte aligned (bits[4:0] must be 0x0). NDA field of the last descriptor of a descriptor chain must be NULL.

Table 143: Descriptor Byte Count Word

Bit	Field	Description
23:0	ByteCount	XOR mode: Size of source and destination blocks in bytes. CRC mode: Size of source block part represented by the descriptor. DMA mode: Size of source and destination block in bytes. Minimum blocks' size: 16B. Maximum blocks' size: 16MB-1
31:24	Reserved	Reserved.

Table 144: Descriptor Destination Address Word

Bits	Field	Description
31:0	DA	Destination Block address pointer. XOR Mode: Destination Block address pointer. CRC mode: Not used. DMA mode: Destination Block address pointer.

Table 145: Descriptor Source Address #N Words

Bits	Field	Description
31:0	SA#0 Source Address #0	Source block #0 address pointer. XOR Mode: Source Block #0 address pointer. CRC mode: Address pointer to part of source block represented by the descriptor. DMA Mode: Source Block address pointer.
31:0	SA#N [N=1..7] Source Address #N	Source block #N address pointer. XOR Mode: Source Block #N address pointer. CRC mode: Not used. DMA mode: Not used.

31.5 Address Decoding

The XOR engine has eight address windows that can be individually configured. With each transaction, the XOR engine first compares the address (source, destination, or descriptor) against the address decoding registers. Each window can be configured to a different target interface. Address comparison is done to select the correct target interface. If the address does not match any of the address windows (no hit), an interrupt is generated and the XOR engine is stopped. If the address matches more than one address window (multiple hit), an interrupt is generated and the XOR engine is stopped.

For the XOR engine to avoid accessing forbidden address space (due to a programing bug), each channel uses access protection logic that prevents it from read/write access to specific address windows. In case of access violation, the operation is stopped, the channel becomes inactive, and an interrupt is asserted.

31.5.1 Target Interface

Source data blocks, destination data block, and descriptors can be targeted to any of the chip Interfaces. The unique attributes of each interface are configured per address window through the XOR engine BARs (Base Address Registers) (see Table 1374, Summary Map Table for the XOR Registers, on page 1471).

31.5.2 64-bit Addressing

Four of the eight address windows have an upper 32-bit address register. These are used for accessing interfaces that support more than 4 GB of address space. The address generated on the interface is composed of the 32-bit address issued by the XOR engine, if it hits the relevant address window, concatenated with the High Remap register.

The XOR engine address decoder can map a total of up to a 4 GB address space.

31.5.3 Address Override

The XOR engine also supports an address override feature. Each of the sources, destination, or next descriptor addresses of each channel, can be configured to use the override feature by using the XOR Engine Address Override Control (XEAOCR)<n> Register (m=0–1, n=0–1) (Table 1379 p. 1475). When override is enabled and the respective pointer field is cleared to 0x0, the transaction target interface, and attributes, are taken from the Base Address register 0 (XEBAR0) and the upper 32 bits of the 64 bit address are taken from High Address Remap Register 0 (XEHARR0). When set to 0x1, these items are taken from XEBAR1and XEHARR1 respectively, and so on for pointer values of 0x2 and 0x3.

This address override feature, enables additional address de-coupling. For example, it allows the use of the same source and destination addresses, while the source is targeted to one interface and destination to a different interface.

Copyright © 2014 Marvell

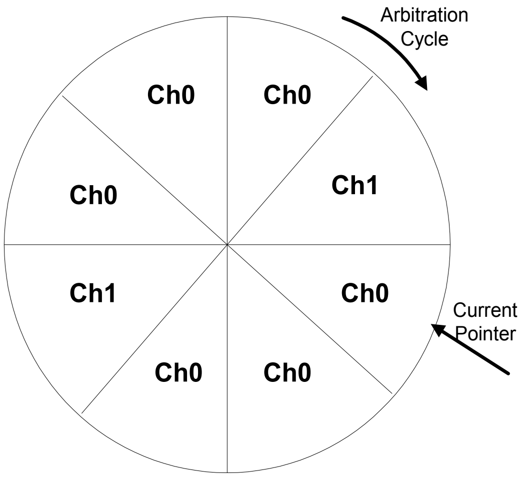
Note When using the address override option, no window access control is performed. For example if override is set to SA#5 of channel 1, any address that is specified in the descriptor as SA#5 is directly accessed without any window access control check.

31.6 Arbitration

The device includes 4 XOR channels in 2 different units, 2 in each unit. The following section describes the arbitration scheme between each 2 channels.

A programmable weighted round robin arbiter controls the bandwidth allocation for each channel on the Mbus port. Each channel can be configured to have a different bandwidth allocation. Figure 134 shows an example of the arbitration cycle.

Figure 134:Programmable Channel Pizza Arbiter



The pizza arbiter has eight slices, each slice can be configured to serve a different channel. In Figure 134, channel0 gets 75% of the bandwidth, and channel1 25% each. At each clock cycle, the arbiter samples all channels requests and gives the bus to the next channel according to the “pizza” setting.

The bandwidth allocation is flexible. The arbiter influences the bandwidth allocation only when 2 ports demand Mbus port service at the same time (congestion conditions). If only 1 channel demands Mbus bandwidth, the channel receives 100% of the bandwidth. For example, in Figure 134, only Channel0 is active and so it gets 100% of the Mbus port bandwidth.

31.7 XOR Engine Programming

31.7.1 Programming in XOR, CRC, and DMA Modes

The XOR engine operation is similar in XOR, CRC, and DMA modes. All of these modes use descriptor chains. The modes differ in their configuration parameters and in their chain descriptor size and field interpretation.

31.7.1.1 Activation

To activate an XOR engine when it is inactive, the software must perform the following sequence:

1. Confirm that the relevant XOR engine channel is inactive (the <XEstatus> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480) is cleared to 0).
2. Initialize the relevant XOR engine channel configuration through the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1478).
3. Prepare the descriptor (or chain of descriptors) in memory.
4. Update the relevant XOR Engine Next Descriptor Pointer (XEnNDPR)<n> Register (m=0–1, n=0–1) (Table 1383 p. 1481) register.
5. Set the <XEStart> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1481).

When the XOR engine starts, the engine activates the relevant channel with the <XEstatus> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480) is set.

When activated (<XEstatus> is set), the XOR engine fetches the descriptor pointed by the XOR Engine Next Descriptor Pointer (XEnNDPR)<n> Register (m=0–1, n=0–1) (Table 1383 p. 1481), and starts performing the operation on it. Upon completion of the operation it progresses to the next descriptor. It continues this operation until it reaches the end of the descriptor chain (Next descriptor Address field of current descriptor = NULL). When it reaches the end of the chain, the XOR engine asserts an interrupt (EOC - End Of Chain interrupt), clears the <XEstatus> bit and enters an inactive state. This inactive state is equal to the initial state of XOR engine upon startup.

31.7.1.2 Update Descriptor Chain

A new descriptor can be added to the chain when the XOR engine is active (<XEstatus>=1).

The software adds new descriptors to the descriptor chain by performing the following:

1. Prepares new descriptors (or chain of descriptors) in memory.
2. Updates the next descriptor address field in the former last descriptor.

Note

If the ownership mechanism is violated, the CPU will write to an XOR engine owned descriptor (the former last one). This does not affect the XOR engine operation.

31.7.1.3 Pause Operation

The pause operation enables a temporary halt of the current descriptor chain processing and then a continuation of it without any impact on the execution, except the delay caused by the pause period. When paused the XOR engine channel does not initiate any requests to the Mbus, releasing its resources to other units.

The pause operation can be used for boosting performance of a mission critical process for a specific time period. After the critical time period is over, the software signals the XOR engine channel to continue processing the current descriptor chain from the point at which it was paused.

Copyright © 2014 Marvell

The software can pause the XOR engine channel operation during an active phase by performing the following:

1. Confirm that the relevant XOR engine channel is active (<XEpause> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480) is set). If it is not active, the pause operation is not necessary.
2. Set the relevant <XEpause>.
3. Check the relevant <XEstatus> field. When it is cleared, the pause operation completed.

When paused (<XEpause> is set), the XOR engine channel suspends the current operation at the earliest opportunity, and enters a pause state. Upon entering a pause state, the XOR engine channel signals the software by clearing the <XEstatus> bit in the activation register and asserting the paused interrupt.

Receipt of an EOC interrupt before a paused interrupt, after initiation of a pause operation, implies that:

- The channel completed the current descriptor chain before the pause operation.
- The channel is in stop mode and not in paused mode.

The software must act accordingly and reactivate the channel according to the Re-Activation After Stop information.

Re-Activation After Pause

To re-activate the channel, the software must set the <XErestart> field in the relevant activation register (XE0ACTR or XE1ACTR). When <XEstatus> field is set, the XOR engine has resumed operation.

After pausing a channel, it is not allowed to stop it. To stop the channel, the software must first perform a re-activation after pause operation. Only after the channel becomes active can it be stopped.

31.7.1.4 Stop Operation

The stop operation terminates processing of an XOR engine channel's current operation. After stop, the current operation cannot be resumed and a new operation must be loaded to the XOR engine channel. The software can stop the XOR engine channel operation while active, by performing the following:

1. Check that the relevant XOR engine channel is active (<XEstatus> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480) is set). If it is not active, the stop operation is not necessary.
2. Set the relevant <XEstop> field.
3. Check the relevant <XEstatus> field. When it is cleared, the stop operation is completed.

When stopped (<XEstop> is set), the XOR engine stops performing the operation at the earliest opportunity and enters an inactive state. Upon entering an inactive state, the XOR engine closes the current descriptor and signals the software by clearing the <XEstatus> field in the activation register and asserting the stopped interrupt. Inactive state is similar to Initial state of XOR engine on startup.

Re-Activation After Stop

Re-activation is similar to the activation procedure in Section 31.7.1.1, Activation, on page 504.

The software must perform the following steps:

1. Confirm that the relevant XOR engine channel is inactive. The <XEstatus> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480) is cleared to 0.
2. Initialize the relevant XOR engine channel through the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1478).

3. Prepare a descriptor (or chain of descriptors) in memory.
4. Update the XOR Engine Next Descriptor Pointer (XEnNDPR)<n> Register (m=0–1, n=0–1) (Table 1383 p. 1481).
5. Set the <XErestart> field.

When the XOR engine starts, the engine activates the relevant channel with the <XEstatus> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480) is set.

When activated (<XEstatus> is set), the XOR engine fetches the descriptor pointed by the XOR Engine Next Descriptor Pointer (XEnNDPR)<n> Register (m=0–1, n=0–1) (Table 1383 p. 1481), and starts performing the operation on it. Upon completion, it progresses to the next descriptor and continues until the end of the descriptor chain is reached, the EOC - Next descriptor Address field of the current descriptor equals NULL. Upon reaching the end of the chain, the XOR engine clears the <XEstatus> bit and enters inactive state. This state is equal to the initial state of XOR engine on startup.

31.7.1.5 Reaching End of Descriptor Chain

Upon reaching the end of the descriptor chain, the XOR engine asserts an EOC (End Of Chain) interrupt and enters an inactive state. It waits to be re-activated by the software (setting <XEStart> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1481)).

Upon receiving an EOC interrupt, two options must be examined by the software:

True EOC XOR engine reaches the end of a descriptor chain.

False EOC Chain was updated and the XOR engine is not in the current EOC. This can occur when software updates the descriptor chain while the XOR engine is processing the former last descriptor in the chain. In this case, the XOR Engine Next Descriptor Pointer (XEnNDPR)<n> Register (m=0–1, n=0–1) (Table 1383 p. 1481) has a NULL value. Although it did not reach a true EOC, the XOR engine enters an inactive state.

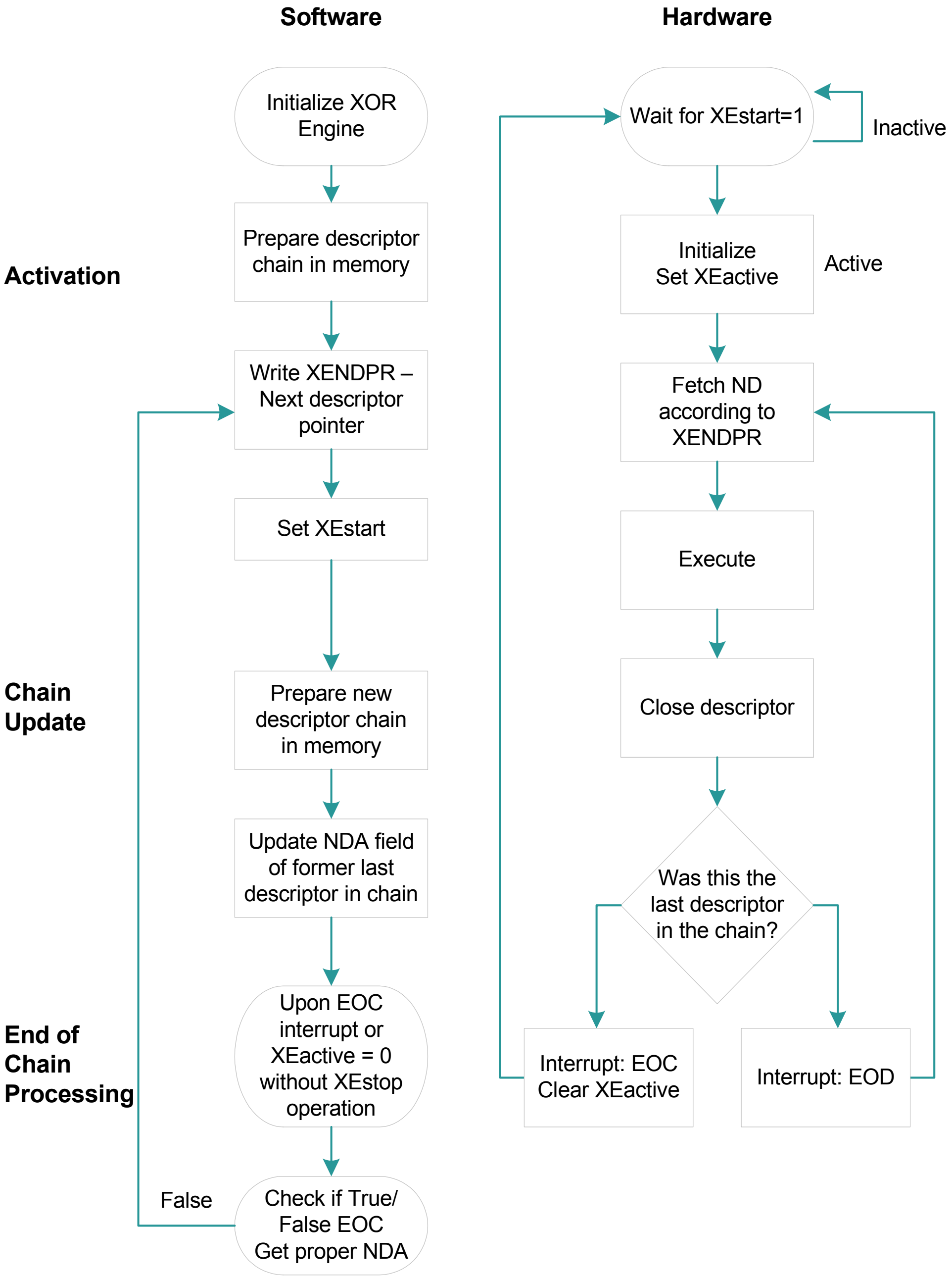
To determine which option is valid, and to act accordingly, the software must check if the XOR engine current descriptor is currently the last descriptor in the chain. For example, read the XOR Engine Current Descriptor Pointer (XEnCDPR)<n> Register (m=0–1, n=0–1) (Table 1384 p. 1481) and match it with the software's current descriptor parameter.

If it is true, EOC acts according to activation after stop.

If it is false EOC forces the XOR engine to re-read the current descriptor. That is done by writing to the current descriptor pointer, that was read from XECDPR, to the Next descriptor Pointer Register (XENDPR), and performing an activation after stop, set the <XEstart> bit in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480).

31.7.1.6 Synchronizing Software and Hardware

Figure 135:Software and Hardware Synchronization


Software
Hardware
Activation
Chain Update
End of Chain Processing
Initialize XOR Engine
Prepare descriptor chain in memory
Write XENDPR – Next descriptor pointer
Set XEstart
Prepare new descriptor chain in memory
Update NDA field of former last descriptor in chain
Upon EOC interrupt or XEactive = 0 without XEstop operation
Check if True/ False EOC Get proper NDA
False
Wait for XEstart=1
Inactive
Initialize Set XEactive
Active
Fetch ND according to XENDPR
Execute
Close descriptor
Was this the last descriptor in the chain?
Interrupt: EOC Clear XEactive
Interrupt: EOD

31.7.2 Programming in ECC and MemInit Modes

The ECC and MemInit modes are programmed and controlled directly through internal registers (without using descriptor chains).

31.7.2.1 Activation

To activate the XOR engine, the software must perform the following sequence:

1. Confirm that XOR engine relevant channel is inactive. The <XEstatus> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480) is cleared to 0.
2. Initialize the relevant XOR engine channel configuration through the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1478).
3. Program the relevant internal registers (XOR engine ECC/MemInit Registers).
4. Set the <XErestart> field.

31.7.2.2 Stop Operation

The stop operation terminates a XOR engine channel's processing of the current operation. After stop, the current operation cannot be resumed. A new operation must be loaded to the XOR engine channel.

To stop the XOR engine channel operation while active, performing the following:

1. Check that the relevant XOR engine channel is active. The <XEstatus> field in the XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Table 1382 p. 1480)Register must be set. If it is not active, the stop operation is not necessary.
2. Set the <XEstop> field in the relevant activation register.
3. Check the relevant <XEstatus> field. When it is cleared, the stop operation is completed.

When stopped (<XEstop> is set), the XOR engine stops performing the operation at the earliest opportunity and enters an inactive state. Upon entering inactive state, the XOR engine signals the software by clearing the <XEstatus> field in the activation register and asserting the stopped interrupt. The inactive state is similar to initial state of the XOR engine on startup.

31.7.3 Internal Registers Write Access Protection

When an XOR engine channel is active, all the registers that are related to that channel, the shared address decoding registers, the shared channel arbitration registers and the shared memory initialization initial value registers, are write access protected. Every write request to those internal registers when the channel is active (<XEstatus> of the relevant channel is set) is silently disregarded. The only channel related registers that can be write accessed during the channel active period are the activation registers, the shared interrupt cause and mask registers, and the debug register.

This design prevents configuration changes during channel operation. Changes during a channel's operation can cause unpredictable results. The register access protection can be de-activated per channel through the relevant <RegAccProtect> field in the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1478).

If at least one of the channels enables register write access protection, write accesses to internal registers shared by channels (for example, address decoding, channel arbitration, and memory initialization initial value registers) are disregarded.

Read requests for all internal registers are enabled at all times, regardless of the channel activation status (except for WO - Write Only registers).

Copyright © 2014 Marvell

31.8 Burst Limit

The maximum burst sizes of different transaction types on the Mbus can be configured through the XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Table 1381 p. 1478).

- Data read (reading a source buffer) maximum burst size: 32B/64B/128B.
- Data write (writing to destination buffer) maximum burst size: 32B/64B/128B.

A descriptor read, fetching a descriptor, is always a 32B burst size. In XOR mode, almost all the 64 bytes of the descriptor are relevant and two 32B read requests are required. In CRC or ECC modes, only the upper 32B of the descriptor are relevant and one 32B read request is sufficient.

Descriptor write, closing a descriptor, is always 8B burst size (Status and iSCSI CRC32C Result words).

31.9 Errors and Interrupts

The XOR engine interrupts are registered in the XOR Engine Interrupt Cause (XEICR1) Register (m=0–1) (Table 1393 p. 1485). Upon an interrupt event, the corresponding cause bit is set to 1. It is cleared upon a software write of 0.

The XOR Engine Interrupt Mask (XEIMR) Register (m=0–1) (Table 1394 p. 1487) controls whether an interrupt event causes an interrupt assertion. The setting of the mask register only affects the interrupt assertion. This setting has no effect on the cause register bits setting.

The XOR engine interrupts are divided into 2 groups:

Error Interrupts Descriptor ownership violation, address miss, multiple hit, window access violation, write protect violation, or parity error.

Operation Completion Interrupts EOD (End of Descriptor), EOC (End of Chain), pause, or stop by software

Table 146 summarizes the interpretation of EOD and EOC interrupts for each operation mode.

Table 146: EOC/EOD interpretation

Operation Mode	Operation Related Interrupt Description
XOR, CRC, DMA	• The EOD interrupt is asserted upon closing each descriptor. If the <EODIntEn> bit of the descriptor is cleared, an EOD interrupt is not asserted when it is closed. • The EOC interrupt is asserted upon reaching end of descriptor chain or upon end of chain processing due to error condition.
MemInit	• The EOC interrupt is asserted upon completing the MemInit operation.
ECC - Non-Timer mode	• The EOC interrupt is asserted after the entire destination block is cleaned.
ECC - Timer mode	• The EOD interrupt is asserted after every section cleanup completion. • The EOC interrupt is asserted after the entire destination block is cleaned.

The following error interrupts are supported:

- Parity error: Internal data path parity error.
- Ownership error: Fetching descriptor that is owned by the CPU (software error).
- Address Miss Error: Accessing an address that is not in one of the address windows, or an address which matches more than one address window.

- Access Protect Error: Accessing an address which is access protected.
- Write Protect Error: Writing to a write protected address.

In all error conditions, the XOR engine halts, as if it is stopped by the software. Also, in all case of an error address, the address is latched in the XOR Engine Error Address (XEEAR) Register (m=0–1) (Table 1396 p. 1488). Once an address is latched, no new addresses (due to additional errors) can be latched until the current address being read.

31.10 Performance Optimization

To increase XOR engine performance:

- Set the source and destination buffers alignment according to the relevant maximum burst size configuration parameters (<SrcBurstLimit> and <DstBurstLimit>). For example, if SrcBurstLimit of Channel0 is 128B and DstBurstLimit is 32B, it is recommended to keep the source buffers in 128B aligned addresses and the destination buffers in 32B aligned addresses.
- Set the buffer sizes as large as possible.

32 Independent DMA (IDMA) Controller

The IDMA Controller has four independent IDMA engines. The IDMA engines optimize system performance by moving large amounts of data without significant CPU intervention.

A typical IDMA usage is to move data from a device I/O interface into the local DRAM. The data in the DRAM is processed by the CPU and driven back by the DMA to the external interface.

Each IDMA engine can move data between any source to any destination. It can transfer a single data buffer of up to 16 MB. It can also run in chain mode, in which each buffer has its own descriptor. The link list of descriptors can be placed in any of the device's interfaces.

The IDMA controller registers are located in Appendix A.23, IDMA Registers, on page 1489.

32.1 Features

The IDMA controller supports the following features:

- 4 IDMA channels, each with a 512-byte buffer
- Single data buffer transfer of up to 16 MB
- Chain mode data buffer transfer
- Chaining via linked-lists of descriptors
- Moves data from any interface to any interface
- Increment or hold on both the source and destination address

32.2 Functional Description

IDMA unit contains four 512-byte buffers, one buffer per DMA channel.

When a channel is activated, data is being read from the source into the buffer, and then written to the destination. Read and write are handled independently. The DMA engine transfers the buffer in chunks of from 8 up to 128 bytes. It reads from the source as long as it has place in the buffer. It writes to the destination, as long as there is valid data in the buffer to be transferred. This independency, results in concurrent reads and writes, and maximum utilization of the DMA interface.

Since the four channels share the same resources, arbitration is required. The four channels use a fixed round-robin arbiter that allows different bandwidth allocation to each channel, see Section 32.6, Arbitration, on page 518.

32.3 IDMA Descriptors

Each IDMA Channel Descriptor consists of four 32-bit registers. Each channel can be configured to work in a 64-KB descriptor mode, or with 16-MB descriptor mode, as shown in Figure 136.

Figure 136:IDMA Descriptors

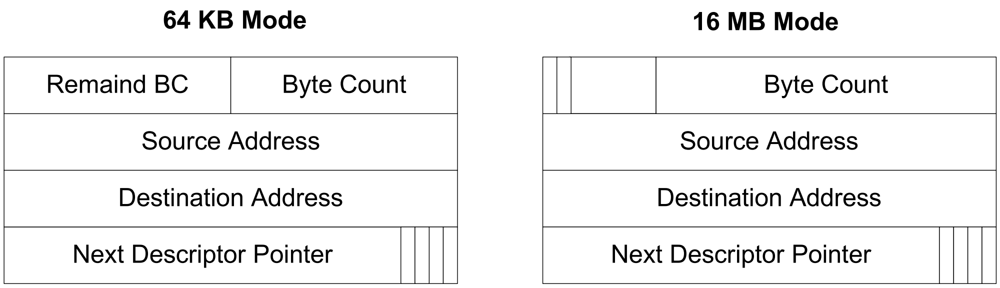


Table 147: DMA Descriptor Definitions

DMA Descriptor	Definition
Byte Count	Number of bytes of data to transfer. The maximum number of bytes which the IDMA controller can be configured to transfer is 64 KB-1 (16-bit register) in 64 KB descriptor mode or 16 MB-1 (24-bit register) in the 16 MB descriptor mode. This register decrements at the end of every burst of transmitted data from the source to the destination. When the byte count register is 0, or the End of Transfer pin is asserted, the IDMA transaction is finished or terminated.
Source Address	Bits[31:0] of the IDMA source address. According to the setting of the Channel Control register, this register either increments or holds the same value.
Destination Address	Bits[31:0] of the IDMA destination address. According to the setting of the Channel Control register, this register either increments or holds the same value.
Pointer to the Next Descriptor	Bits[31:0] of the IDMA Next Descriptor address for chained operation. The descriptor must be 16 sequential bytes located at 16-bytes aligned address (bits[3:0] are 0). **NOTE:** Only used when the channel is configured to Chained Mode.

32.4 IDMA Address Decoding

The four DMA channels share eight address windows. Each address window can be individually configured.

With each IDMA transaction, the IDMA engine first compares the address (source, destination, or descriptor) against its address decoding registers. Each window can be configured to different target interface. Address comparison is done to select the correct target interface (e.g. DRAM CS[0]).

If the address does not match any of the address windows, an interrupt is generated and the IDMA engine is stopped.

Four of the eight address windows have an upper 32-bit address register. These are used for accessing interfaces that support more than 4GB address space. The address generated on the interface is composed of the 32-bit address issued by the IDMA (if it hits the relevant address window) concatenated with the High Remap register.

For the DMA engine to avoid accessing forbidden address space (due to a programing bug), each channel uses access protection logic that prevents it from read/write access to specific address windows. In case of access violation, the IDMA halts and an interrupt is asserted.

The IDMA also supports an address override feature. Each of the source, destination, or next descriptor addresses can be configured to use the override feature by using the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1492) <SAddrOvr> bits[22:21], <DAddrOvr> bits[24:23], and <NAddrOvr> bits[26:25]. When the respective field is set to 0x1, the transaction target interface attributes are taken from Base Address register (BAR) 1, and the upper 32-bit address are taken from High Address Remap register of BAR 1. When set to 0x2, the attributes are taken from BAR 2. When set to 0x3, the attributes are taken from BAR 3.

This address override feature, enables additional address decoupling. For example, it allows the use of the same source and destination addresses while the source is targeted to one interface and destination to a second interface.

Note

For full details on IDMA address decoding, see Section 3.3, IDMA Address Decoding, on page 66.

32.5 IDMA Channel Control

Each IDMA Channel has its own unique control register where certain IDMA modes are programmed. Following are the bit descriptions for each field in the control registers. For detailed registers descriptions, see Appendix A.23.2, IDMA Channel Control, on page 1492.

32.5.1 Address Increment/Hold

The IDMA engine supports both increment and hold modes, on both source and destination addresses.

If the <SrcHold> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1495) is cleared to 0, the IDMA automatically increments the source address with each transfer. If set to 1, the source address remains constant throughout the IDMA burst.

Similarly, If the <DestHold> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1495) is cleared to 0, the IDMA automatically increments the destination address. If the set to 1, the destination address remains constant throughout the IDMA burst.

Setting the <SrcHold> or <DestHold> fields is useful when the source/destination device is accessible through a constant address. For example, if the source/destination device is a FIFO, it is accessed with a single address, while data is being popped/pushed with each IDMA burst.

Note

When using Hold mode, the address is restricted to be aligned to the Burst Limit setting (see the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1492) <DstBurstLimit> and <SrcBurstLimit> fields).

32.5.2 Burst Limit

The whole IDMA byte count is chopped into small bursts.

The burst limit can vary from 8 to 128 bytes in modulo-2 steps (that is, 8, 16, 32, 64, 128 bytes). It determines the burst length of IDMA transaction against the source and destination. There are separate BurstLimit parameters for source and destination. For example, setting the source burst limit to 32 bytes and the destination burst limit to 128 bytes, means that the IDMA reads 32 bytes from the source, and then writes the data to the destination after combining to 128 bytes. The IDMA continues this read/write loop until transfer of the whole byte count is complete.

The burst limit setting is affected by the source and destination characteristics, as well as system bandwidth allocation considerations.

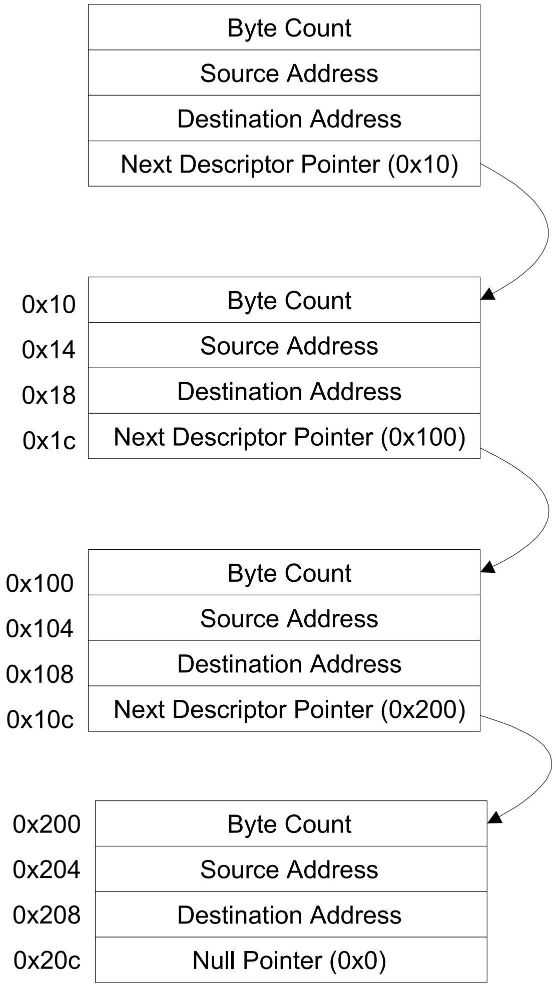
32.5.3 Chain Mode

When the <ChainMode> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1494) bit[9] is cleared to 0, chained mode is enabled.

In chain mode, at the completion of one buffer transfer, the Pointer to Next Descriptor provides the address of a next IDMA descriptor. If it is a NULL pointer (value of 0), it indicates that this is the last descriptor in the chain. If not, the DMA engine fetches the new descriptor, and starts transferring the new buffer.

Figure 137 shows an example of an IDMA descriptors chain.

Figure 137:Chained Mode IDMA



Fetch next descriptor can be forced by the <FetchND> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1494).

Setting this bit to 1 forces a fetch of the next descriptor based on the value in the Pointer to Next Descriptor register.

This bit can be set even if the current IDMA has not yet completed. In this case, the IDMA engine completes the current burst read and write and then fetches the next descriptor. This bit is reset back to 0 after the fetch of the new descriptor is complete. Setting <FetchND> is not allowed if the descriptor equals Null.

Note

If using the <FetchND> bit while the current DMA is in progress, the <Abr> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1493) must be set.

The first descriptor of a chain can be set directly by programing the channels registers, or can be fetched from memory, using the FetchND bit. If fetched from memory, the next descriptor address must be first written to the Next Descriptor Pointer register of the channel. The channel then must be enabled by setting the <ChainMode> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1494) to 1 and setting <FetchND> to 1 (see Section 32.5.4, Channel Activation, on page 516).

When the IDMA transfer is done, an IDMA completion interrupt is set. When running in chain mode, the <IntMode> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1494) controls whether to assert an interrupt on the completion of every byte count transfer or only with last descriptor byte count completion. If cleared to 0, the <Comp> field in the Interrupt Mask Register (Table 1412 p. 1499) is set every time the IDMA byte count reaches 0. If set to 1, <Comp> is asserted when both the Pointer to Next Descriptor Register has a NULL value and byte count is 0.

If <ChainMode> is set to 1, chained mode is disabled and the Pointer to Next Descriptor register is not loaded at the completion of the IDMA transaction.

In non-chained mode the Byte Count, Source, and Destination registers must be initialized prior to enabling the channel.

If reading a new descriptor results in parity/ECC error indicated by the unit from which the descriptor is being read, the channel halts. This is done in order to prevent destructive reads/writes, due to bad source/destination pointers.

If using <FetchND> to fetch the first descriptor and the <IntMode> bit is cleared to 0, a dummy DMA completion interrupt is asserted with this first fetch. Although, no data has been transferred.

32.5.4 Channel Activation

Software channel activation is done via the <ChanEn> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1494).

When cleared to 0, the channel is disabled. When set to 1, the IDMA is initiated based on the current setting loaded in the channel descriptor (i.e. byte count, source address, and destination address). An active channel can be temporarily stopped by clearing the <ChanEn> field, and then continuing later from the point the channel was stopped by setting <ChanEn> back to 1.

Clearing the <ChanEn> field during an IDMA operation does not guarantee an immediate channel pause. The IDMA engine must complete transferring the last burst it was working on. Software can monitor the channel status by reading <ChanAct> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1494).

To restart a suspended channel in non-chained mode, the <ChanEn> field must be set to 1. In Chained mode, the software must find out if the first fetch took place. If the fetch did take place, only <ChanEn> is set to 1. If the fetch did not take place, the <FetchND> field must also be set to 1.

The <ChanAct> field is read only. If cleared to 0, the channel is not active. If set to 1, the channel is active. In non-chain mode, this bit is de-asserted when the byte count reaches zero. In chain-mode, this bit is de-asserted when pointer to next descriptor is NULL and byte count reaches zero.

If <ChanEn> field is cleared to 0 during IDMA transfer, <ChanAct> bit toggles to 0 as soon as the IDMA engine finishes the last burst it is working on.

To abort an IDMA transfer in the middle, the software must set <Abr> field in the Channeln Control (Low) Register (n=0–3) (Table 1403 p. 1493) to 1. Setting this bit has a similar affect to clearing <ChanEn>. However, it guarantees a smooth transfer of the IDMA engine to idle state. As soon as the IDMA is back in idle state, the <Abr> bit gets cleared, allowing the software to re-program the channel.

When a channel is stopped (abort, fetch, pause), it completes the last read from source, and then writes all of the remaining data in the FIFO to the destination.

If the byte count is smaller that the burst limit setting, the source and destination addresses must be 64-bit aligned.

If the close descriptor feature is used, only set the <Abr> field after first clearing <ChanEn> and then <ChanAct>.

32.5.5 Source and Destination Addresses Alignment

The IDMA implementation maintains aligned accesses to both source and destination.

If source and destination addresses have different alignments, the IDMA performs multiple reads from the source to execute a write of full BurstLimit to the destination. For example, if the source address is 0x4, the destination address is 0x100, and BurstLimit of both source and destination is set to 8 bytes, the IDMA perform two reads from the source. First 4 bytes from address 0x4 then 8 bytes from address 0x8, and only then performs a write of 8 bytes to address 0x100.

This implementation guarantees that all reads from the source and all writes to the destination have all byte enables asserted (except for the IDMA buffer start/end, in case they are not aligned). This is especially important when the source device does not tolerate read of extra data (destructive reads) or when the destination device does not support write byte enables.

32.5.6 Descriptor Ownership

A typical application of chain mode IDMA involves the Marvell® core processor preparing a chain of descriptors in memory and then preparing buffers to move from source to destination. Buffers may be dynamically prepared, this means once a buffer was transferred the Marvell® core processor can prepare a new buffer in the same location to be sent. This application requires some handshake between the IDMA engine and the Marvell® core processor.

When working with the 16 MB descriptor mode, <Own> field in the Interrupt Cause Register (Table 1411 p. 1498) acts as an ownership bit. If set to 1, the descriptor is owned by the device IDMA. If cleared to 0, it is owned by the Marvell® core processor. Once the Marvell® core processor prepares a buffer to be transferred, it sets the ownership bit. This indicates that the buffer is owned by the IDMA. Once the IDMA completes transferring the buffer, it closes the descriptor by writing back the upper byte of the Byte Count register (bits[31:24]), with MSB cleared to 0, indicating to the Marvell® core processor the buffer was transferred. When the Marvell® core processor recognizes that it owns the buffer, it is allowed to place a new buffer to be transferred. An attempt by the IDMA to fetch a descriptor that is owned by Marvell® core processor (which means the Marvell® core processor did not prepare a new buffer yet), results in an interrupt assertion and an IDMA channel halt.

Note

This feature is not supported in 64 KB descriptor mode.

The Descriptor is closed when the byte count reaches 0 or when transfer is terminated in the middle via fetch next descriptor command. In this case, the transfer may end with some data remaining in the buffer pointed by the current descriptor.

When working in 64 KB descriptor mode, when closing the descriptor, the IDMA engine writes the left byte count to the upper 16-bit of the byte count field of the descriptor. This is useful if an IDMA is terminated in the middle and a CPU might want to re-transmit the left byte count. In case the IDMA ended properly (all byte count was transferred), a 0 value is written back to the descriptor.

When working with the 16 MB descriptor mode, there is an alternative way to signal to the Marvell® core processor that the descriptor was not completely transferred. In this case, the IDMA engine rather than writing back the remaining byte count, it writes back to only bits[31:24] of the descriptor's

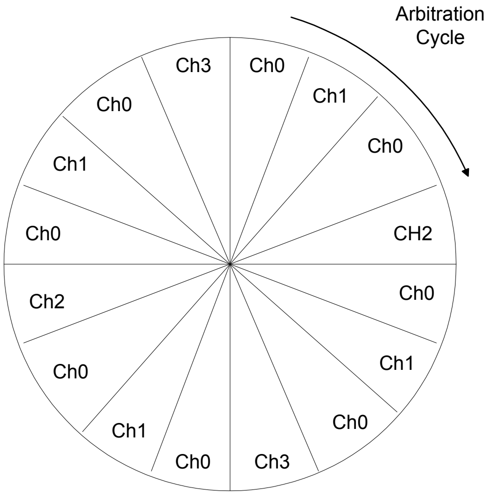
<ByteCount> field, with bit[30] indicating whether the whole byte count was transferred (0) or terminated before transfer completion (1). Bits[29:24] are meaningless.

Each IDMA channel has a Current Descriptor Pointer register (CDPTR) associated with it. This register is used for closing the current descriptor before fetching the next descriptor. The register is a read/write register but the Marvell® core processor must not write to it. When the NPTR (Next pointer) is first programed, the CDPTR reloads itself with the same value written to NPTR. After processing a descriptor, the IDMA channel updates the current descriptor using CDPTR, saves NPTR into the CDPTR, and fetches a new descriptor.

32.6 Arbitration

The IDMA controller has a programmable round-robin arbiter for the four DMA channels. Each channel can be configured to have different priority. Figure 138 shows an example arbitration cycle.

Figure 138:Configurable Weights Arbiter



The user can define each of the 16 slices of this “pizza arbiter”. At each clock cycle, the arbiter samples all channels requests and gives the bus to the next agent according to the “pizza”.

Note

The arbiter setting is effective only when using more than two channels. With two channels, each one will receive half of the BW regardless of arbiter setting.

32.7 DMA Interrupts

The IDMA interrupts are registered in the IDMA Interrupt Cause register. Upon an interrupt event, the corresponding cause bit is set to 1. It is cleared upon a software write of 0.

The IDMA Mask registers controls whether an interrupt event causes an interrupt assertion. The setting of the mask register only affects the interrupt assertion, it has no affect on the cause register bits setting.

The following interrupt events are supported per each channel:

- DMA completion
- DMA address out of range
- DMA access protect violation
- DMA write protect violation
- DMA descriptor ownership violation

In case of an error condition (address out of range, access protect violation, write protect violation, descriptor ownership violation), the IDMA transaction address is latched in the Address Error register. Once an address is latched, no new address (due to additional errors) can be latched, until the current address being read.

33 Electronic Fuse (eFuse)

This section describes the electronic fuse (eFuse), a one-time programmable module.

The eFuse registers are located in Appendix A.24, Electronic Fuse (eFuse) Registers, on page 1501.

33.1 Unique Device ID Read

The eFuse integrates a Unique Device ID number programmed and locked by Marvell®.

For the Unique Device ID value reading, read the following registers:

- Unique Device ID 0 Register (Table 1421 p. 1503)
- Unique Device ID 1 Register (Table 1422 p. 1503)

33.2 eFuse Fields and Mapping to Secured Boot Register Space

Table 149 lists the eFuse locations and addresses for the eFuse Secured Boot registers.

Table 149: Secured Boot Register Space

Field	Register Address
Secured_Boot_Mode	0x10010[0]
Secured_Debug_Disable	0x10010[1]
Secured Boot Cntrl_Spare0	0x10010[3:2]
Secured Boot Boot Device	0x10010[7:4]
Secured Boot Cntrl_Spare1	0x10010[15:8]
Secured Boot Flash ID	0x10010[31:16]
Secured Boot Box ID	0x10014[31:0]
Secured Boot Reserved3	0x10018[31:0]
Secured Boot Reserved4	0x1001C[31:0]
Secured Boot Symmetric Key 0	0x10080[31:0]
Secured Boot Symmetric Key 1	0x10084[31:0]
Secured Boot Symmetric Key 2	0x10088[31:0]
Secured Boot Symmetric Key 3	0x1008C[31:0]
Secured Boot Public Key Digest 0	0x10020[31:0]
Secured Boot Public Key Digest 1	0x10024[31:0]
Secured Boot Public Key Digest 2	0x10028[31:0]
Secured Boot Public Key Digest 3	0x1002C[31:0]
Secured Boot Public Key Digest 4	0x10030[31:0]
Secured Boot Public Key Digest 5	0x10034[31:0]

Copyright © 2014 Marvell

Table 149: Secured Boot Register Space (Continued)

Field	Register Address
Secured Boot Public Key Digest 6	0x10038[31:0]
Secured Boot Public Key Digest 7	0x1003C[31:0]

33.3 Protection Bit Mapping

The following table lists the mapping between the eFuse data registers and the corresponding security bit in the Fuse Protection FSB Register (Table 1425 p. 1505).

Table 150: Protection Bits Mapping

Field	Register Address	eFuse Data Register Address
FSB_GP_1_0	0x1000C[1]	0x100C4, 0x100C0
FSB_GP_3_2	0x1000C[2]	0x100CC, 0x100C8
FSB_SBR_CNTRL	0x1000C[10]	0x10014, 0x10010
FSB_SBR_SPARE	0x1000C[11]	0x1001C, 0x10018
FSB_SBR_SKEY_1_0	0x1000C[12]	0x10084, 0x10080
FSB_SBR_SKEY_1_0	0x1000C[13]	0x1008C, 0x10088
FSB_SBR_PKEY_1_0	0x1000C[14]	0x10024, 0x10020
FSB_SBR_PKEY_3_2	0x1000C[15]	0x1002C, 0x10028
FSB_SBR_PKEY_5_4	0x1000C[16]	0x10034, 0x10030
FSB_SBR_PKEY_7_6	0x1000C[17]	0x1003C, 0x10038

33.4 eFuse Programming

There are 2 methods for programming the eFuse:

- The Software Burn mode is a register access interface that a customer provides the ability to program the Secured Boot eFuse and the GP instances.
- The JTAG eFuse Access is a dedicated JTAG mode in which the eFuse can be accessed with read and write commands via the JTAG Interface (without usage of any other pin). The interface address mapping is equivalent to the internal address mapping.

33.4.1 Software eFuse Access—Burn Sequence

Perform the following sequence to activate the software burning mechanism:

1. Write: <Fuse Access Mode> = 0x1.
2. Set the necessary data bits by writing to the corresponding eFuse register.
3. If the current write is the last programming needed on the target eFuse, set the security bit by writing to the corresponding FSB register.
4. Trigger the burn by writing 0x1 to <Fuse Burn Trigger>, and the corresponding address to <Fuse Burn Address>.
5. Poll the <Fuse Access Done> = 0x1 for an indication that the burn process was completed.

6. Check that <Fuse Burn Success> = 0x1. If not, the burn process has failed. See the note below for possible causes of the failure.
7. Clear <Fuse Access Mode>.

Note

- The above sequence demonstrates a write to a single eFuse register. The user can repeat Steps 2–6 for additional registers.
- The possible reasons that may cause Step 6 to fail are:
  - An attempt to burn an eFuse where the FSB is active.
  - An attempt to write 0 to a bit that has been burned previously.
  - A manufacture defect of the eFuse.

33.4.2 JTAG eFuse Access

The eFuse can be accessed via the JTAG Interface. To do so, configure the 5-bit JTAG Instruction register (JTAG_IREG[4:0]) to 0x14. Following this action, the device still operates in Functional mode. In this mode, the JTAG eFuse Data register is connected between TDI and TDO.

Figure 139 displays the definition of the JTAG eFuse Data Register (JTAG_EFUSE_DREG) content.

Figure 139:JTAG eFuse Data Register Content (JTAG_EFUSE_DREG)

TDI	43	42	41	40:33	32	31:0	TDO
→	Read	Burn	Burn Mode	Address	FSB	Data	→

43:35	34	33	32	31:0
RSVD	Success	Done	FSB	Data

33.4.2.1 JTAG eFuse Burn Sequence

The sequence to activate the JTAG burn mechanism is as follows (see Note 1):

1. Serially Load the 5-bit Instruction register to the value IREG = 0x14 (see Note 2).
2. Serially Load the 44-bit eFuse Data Register (JTAG_EFUSE_DREG) such that:
   a) <Burn> (JTAG_EFUSE_DREG[42]) = 0x1.
   b) <Read> (JTAG_EFUSE_DREG[43]) = 0x0.
   c) <BurnMode> (JTAG_EFUSE_DREG[41])= 0x1.
   d) <ADDR> (JTAG_EFUSE_DREG[40:33]) = bits[7:0] of the register offset.
   e) <FSB> (JTAG_EFUSE_DREG[32]) = set 1 if for the burn process to set the FSB of the eFuse that contains the corresponding data.
   f) <DATA> (JTAG_EFUSE_DREG[31:0]) = the data that should be burned.
3. Poll <DONE> = 0x1 by serially unloading the 44-bit Data Register (JTAG_EFUSE_DREG) and examining JTAG_EFUSE_DREG[33]. During these unload operations, drive 0x0 on TDI, thus maintaining JTAG accesses that do *not* trigger an operation (Read or Burn).
4. Upon an unload that was completed with <DONE> = 0x1, examine the <Success> field (JTAG_EFUSE_DREG[34]) to determine if the operation has finished. If <Success> = 0x1, the burn process has completed successfully. If <Success> = 0x0, the burn process has failed.
5. Return JTAG IREG to its default value by serial loading the value 0x2 to IREG and setting the TAP controller state to RESET (see Note 2).

33.4.2.2 JTAG eFuse Read Sequence

The sequence to activate the JTAG read mechanism is as follows (see Note 1):

1. Serially Load the 5-bit Instruction register to be IREG = 0x14.
2. Serially Load the 44-bit Data Register (JTAG_EFUSE_DREG) such that:
   a) <Read> (JTAG_EFUSE_DREG[43]) = 0x1
   b) <Burn> (JTAG_EFUSE_DREG[42]) = 0x0
   c) <BurnMode> (JTAG_EFUSE_DREG[41]) = 0x1
   d) <ADDR> (JTAG_EFUSE_DREG[40:33]) = bits[7:0] of the register offset to be read
3. Poll <DONE> = 0x1 by serially unloading the 44-bit Data Register (JTAG_EFUSE_DREG) and examining JTAG_EFUSE_DREG[33]. During these unload operations, drive 0x0 on TDI, thus maintaining JTAG accesses that do NOT trigger an operation (Read or Burn)
4. Upon an unload that was completed with <DONE> = 0x1, examine <DATA> (JTAG_EFUSE_DREG[31:0]]) and <FSB> (JTAG_EFUSE_DREG[32]) to obtain the data that exists in the eFuse corresponding to the desired offset and to confirm that offset is locked for further burns.
5. Return JTAG IREG to its default value by serial loading into IREG 0x2 and setting the TAP controller state to RESET (see Note 2).

33.4.2.3 Notes Regarding the JTAG Sequences

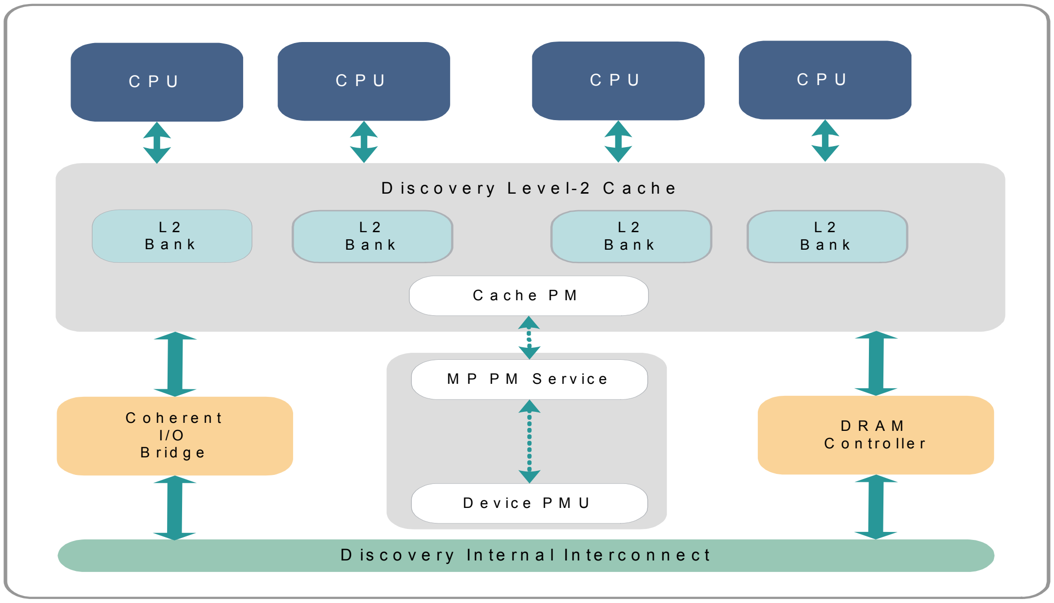
1. The description assumes knowledge of JTAG interface. In particular, signal protocol for loading IREG, loading/unloading Data Register (JTAG_EFUSE_DREG), and TAP SM processing are beyond the scope of this document.
2. Stages 1 and 5 define an entrance and exit of JTAG eFuse mode. In the event of several actions, one can avoid them and resume normal mode following the last action.

34 Power Management

This section describes the power management (PM) capabilities of the MV78230/78x60 device, including programming and signal interfaces.

The Power Management Unit (PMU) administers the power management functions of the MV78230/78x60 device, including reset, clocks, and power domain functional modes. The PMU enables the optimization of the device and system overall power consumption.

Figure 140:Power Management Block Diagram



The Power Management registers are located in Appendix A.20, Power Management Unit (PMU) Registers, on page 1435.

34.1 Features

The MV78230/78x60 supports the following power management features:

- Independent power modes control per CPU with control of on-die, per-CPU power switch for powering down the CPUs
- Run, Idle, and Deep Idle (Power Down) CPU states with coherency management
- Hardware controlled wake-up events and power state transitions
- Dynamic Frequency Scaling (DFS) of each CPU, enabling higher performance power flexibility by dynamically changing CPU frequency
- Functional (dynamic or static) clock gating for the CPU subsystem units and the I/O peripherals
- Programmable thermal sensor controller with overheat detection
- Hardware assist for system idle profiling
- Unit specific power management options

Copyright © 2014 Marvell

34.2 Functional Description

The Power Management functions are provided by two power manager units:

- The Device Power Management Unit (DEV_PMU)
- The Multiprocessor Power Management Service Unit (MP_PMU).

The MP_PMU is responsible for the power management, clocks, and reset functions of the multiprocessor subsystem, including the CPU cores, the Coherency fabric, the L2 cache and the debug units. The MP_PMU delivers programming interface, wake-up controls and parallel independent entry and exit sequences of each CPU's power modes. In addition, the MP_PMU integrates hardware coordination logic for synchronizing between all the CPU cores for handling shared resources power management flows. The MP_PMU communicates with the DEV_PMU which controls all power domains, external power-supply functions, device PLLs, and peripheral power saving modes.

34.3 Individual CPU Power Management

The MV78230/78x60 supports the following advanced CPU Core power management capabilities.

Table 151: CPU Core Power Modes

Mode	Description	Core Responds To		
		Snoop	Interrupts	Debug
RUN	All units operating normally with fine-grained dynamic clock gating	Yes	Yes	Yes
IDLE	CPU is placed in WFI/WFE states. It stops dispatching new instructions and halts all internal clocks.	Yes	Yes	Yes
DEEP IDLE (Power Down)	CPU is entirely power-down; caches are flushed to memory.	No	Yes	No

34.3.1 CPU Run Mode

A normal mode of operation in which all of the functionality of the core is available. The CPU core delivers fine-grained dynamic clock gating to its functional blocks during this mode, lowering the dynamic power consumption during functional operation.

34.3.2 CPU Idle (WFI/WFE) Mode

Wait For Interrupt (WFI) and Wait For Event (WFE) disables most of the clocks of the CPU, while keeping its logic powered up. Transition to these states is performed directly by software through the execution of a WFI instruction and WFE instructions respectively. The CPU core ensures that all memory accesses occur in a program order before the WFI operation has completed, and the core enters an idle state. The transition from WFI mode back to Run mode is caused by:

- An interrupt, on the assertion of the IRQ or FIQ
- A debug request, regardless of whether a debug is enabled
- An imprecise Data Abort
- Cache maintenance, TLB, or Instruction synchronization broadcasted requests from other CPUs
- A reset to the CPU

While in WFI/WFE state, the processor serves snoop requests from the coherent fabric. This includes snoops resulting from other CPU cores or I/O agent. Only the blocks that responsible for the snoop processing inside the processor are temporarily woken up and then placed back to WFI/WFE mode upon completion of the snoop. With the addition of an external snoop filter, waking up the CPU core for serving an inbound snoop request is applied only for snoops that are resolved as hit by the snoop filter of the attached CPU core.

Wait For Event (WFE) instruction is applicable in the multiprocessor. Once a CPU core is being placed in WFE state, it will wake up on any other CPU Send Event notification through execution of the SEV instruction. For example, two CPU cores are racing on same semaphore. The one that does not win can insert itself to WFE state, waiting for the other core SEV to release the semaphore.

34.3.3 CPU Deep Idle

Deep Idle state is a low-power state in which power consumption of the CPU core is completely diminished by powering off this domain. There is no processor activity in this mode, state is lost, and the caches contents are not preserved. Thus all processor states, including its caches, architecture, and debug registers must be saved externally.

The device is brought out of this mode by any enabled wake-up event. Recovery is fully controlled by the PMU. On exit from Deep Idle low-power mode, the CPU core is resumed through an internal reset.

While in Deep Idle mode, the PMU asserts the appropriate power down signal to the internal power switch.

34.3.3.1 Entering Deep Idle Mode

Before entering Deep Idle mode, the state of the CPU and its caches must be stored to the external SDRAM, including the caches, all ARM registers, all CP15 registers and debug related states.

Enter Deep Idle low-power mode, by performing the following steps:

1. Disable the CPU FIQ and IRQ interrupts in the CPU core.
2. Flush the L1 Data cache; save all ARM registers, CP15, VFP, and Debug registers.
3. Issue a Data Synchronization Barrier instruction to ensure that all state saving has been completed.
4. Enable any interrupts to be used as wake-ups from low-power Deep Idle mode. If any interrupt is not necessary, it should be disabled through the main interrupt controller IRQ line or at the unit level. Alternatively, in a system where all functional interrupts are mapped to IRQ interrupt, the FIQ can be utilized for triggering a Deep Idle wake-up event by pre-programming all the wake-up interrupts to assert the FIQ. Wake-up triggering through IRQ, FIQ or both are configured by setting<Enable IRQ Wake Up> and/or <Enable FIQ Wake Up> accordingly.
5. To enable debugger wake up during Deep Idle state, set <Enable Debug Wake Up>.
6. Set <Global CPU IRQ Mask> and <Global CPU FIQ Mask> to 1. The MP_PMU masks the main interrupt pins (FIQ and IRQ) from the Interrupt Controller from being asserted to the CPU core.
7. Set the <L2 Power Down Request> to 1 if powering down the L2 cache is required. L2 cache is flushed and powered down by the hardware. Since the L2 cache is shared among all the processors, powering it down will be performed by hardware only when the all CPUs request it simultaneously.
8. Set the <Deep Idle Power Enable> to 1.
9. Place the CPU to **Wait For Interrupt** mode. When the CPU enters this state, the following steps occur:

   a) CPU activity is halted, outstanding instructions are completed, and the write buffer is drained out.

   b) The CPU clocks are halted.

   c) An internal reset is generated to the CPU.

   d) The PMU power down the CPU core.

   NOTE: If this is the last CPU to enter Deep Idle mode, and all the CPUs requested to power down the L2 cache, the following steps will be performed:

e) The L2 cache is flushed and its clocks are halted.
f) The PMU powers down the L2 cache.

34.3.3.2 Behavior During Deep Idle Mode

In Deep Idle mode the processor is powered down and logically placed in reset state. It does not recognize interrupt assertion during Deep Idle mode. Therefore, the MP_PMU watches for pre-programmed wake-up events configured by the CPU prior to entering Deep Idle mode indication assertion, and invokes exit from Deep Idle mode on any of the device's enabled interrupts including inter processor interrupts. While in Deep Idle state, the <CPU Power Down Status> of the appropriate CPU is set.

34.3.3.3 Exiting Deep Idle Mode

The following occurs after the assertion of any pre-programmed wake-up event:

1. The on-die switch is enabled to resume CPU core voltage. The PMU waits for CPU power to stabilize.
2. The CPU clocks are started.
3. The CPU internal reset is de-asserted, enabling CPU fetching from the reset-vector location. All CPU registers are set to their pre-defined reset values.
4. The CPU begins the required boot sequence, which includes the following items:
   a) Software examines the CPU power status to determine that the reset source was a Deep Idle low-power exit reset.
   b) Software brings the SDRAM out of self-refresh mode, enabling context restoring from SDRAM.
   c) Enable delivering of other CPU cores cache maintenance instruction, TLB and Instruction synchronization to the CPU core by setting <DVM Snoop CPU Enable> to 1. This should be enabled prior to Instruction cache and MMU initialization.
   d) Invalidate the external snoop filter by setting <Invalidate Snoop Filter> to 1.
   e) Enable delivering of snoop requests to the CPU core by setting <Snoop CPU Enable> to 1, the coherency fabric plugs-in the CPU core to the coherency subsystem. This step should be performed prior to enabling the Data cache.
5. The CPU core restores context which was saved before the power down and continues execution.

34.3.4 Coherency in Low Power Modes

When the CPU core is in Idle mode, the CPU core is in the *clock gating* state. It snoops its L1 caches and full coherency is maintained. The CPU core also serves cache maintenance, TLB, and instruction synchronization messages broadcasted by other CPUs. When snooping is completed, the CPU core immediately resumes Idle state. The external snoop filter significantly reduces the total wake-up rate on snoop service, results in temporarily waking-up the CPU core for serving only snoops that hit the L1 cache.

In a Deep Idle state, the L1 Data cache is flushed to lower memory level, Instruction cache and TLB are invalidated so coherency is inherently maintained.

34.3.5 Dynamic Clock Gating

All the CPU core and the FPU coprocessor functional blocks integrate cycle accurate dynamic clock gating for halting a non-active functional block, execution component, or pipe stage.

34.4 Multiprocessor Subsystem Power Management

34.4.1 L2 Cache and Coherency Fabric Power Down

If all the CPUs are already in power down state, it is possible to further decrease the power consumption by powering down the L2 cache and Coherency Fabric.

The actual triggering of this power down flow is done by hardware, following specific configurations settings.

If all the conditions are met, e.g. all the processors request to power down the L2, the hardware initiates the L2 cache power down flow, which includes flushing of the cache content to the DRAM, and disable the sharing. This causes all I/O transactions to be directed to the DRAM.

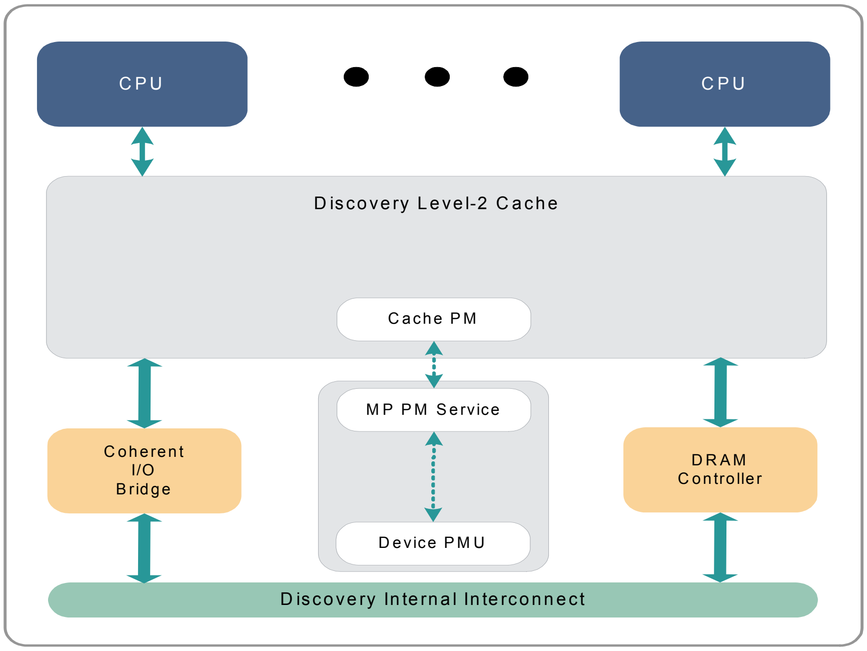
For shutting down the core and its L2 cache, each core powering down can be independent to the other cores. At the point when all the cores have been placed to power-down state, the shared L2 cache can be powered down as well for further power saving. This is fully achieved by the MP_PMU Service Unit and the L2 controller.

The MP_PMU includes hardware coordination logic to detect when all cores reach their power down conditions. It then instructs the shared L2 cache to flush its content through lower level memory, places it in reset state, and communicates with the DEV_PMU to power down the entire multiprocessor microprocessor subsystem. Immediately following power-up, the L2 controller performs self invalidation.

When the L2 cache and Coherency Fabric are powered down the PMU monitors pre-configured wake events of all the CPUs, and if such an event occurs, the PMU first powers up the L2 cache and Coherency Fabric, and only then it powers up the specific CPU.

To complete the flow, the CPU's reset is de-asserted and it returns to run state, performing boot flow (resume from Deep-Idle).

Figure 141:L2 Power Down



Copyright © 2014 Marvell

34.4.1.1 Entering Power Down Mode Flow

When PMU detects that all CPUs are in Deep-Idle state and have been instructed to also power down the shared L2 and coherency fabric, the device starts the procedure for entering Power Down Mode.

Enter Deep Idle low-power mode, by performing the following steps by the PMU hardware to flush the pending I/O coherency traffic and the L2 cache content:

1. The PMU disables all new I/O coherency requests from being allocated to the shared L2 cache, from this point, read and write requests for access to L2 for reading and writing update the values but do not allocate to it.
2. On hit in the L2 cache, all I/O requests are written also to the external memory.
3. The PMU initiates the L2 flush operation. During this operation, the device continues routing the coherent I/O traffic through the L2 cache, steps (1) and (2) above guarantees system data correctness.
4. When L2 flush is completed, the hardware blocks routing of new I/O traffic through the coherency fabric, flushes all pending I/O transactions and routes the I/O traffic to access external memory directly.
5. The PMU powers down the entire block.
6. The PMU tracks all the CPUs’ wake-up events.

34.4.1.2 Exiting Power Down Mode Flow

Upon occurrence of any CPU’s wake-up event, the PMU performs the following steps for waking up the L2 cache and coherency fabric:

1. The PMU operates the power up flow and removes all logic components from reset state.
2. The PMU re-enables the routing of I/O coherent transactions from the L2 cache and coherency fabric.
3. The waked-up CPU core exists from reset state and starts running its Deep Idle resume procedure.

34.4.2 L2 Cache Power Management

The L2 cache supports the following Power Management features:

- L2 Idle state: Dynamic clock gating
- L2 Retention state: Dynamic memory state retention
- L2 Deep Idle state: Cache controller and its memories are powered down, state is stored in lower memory level.

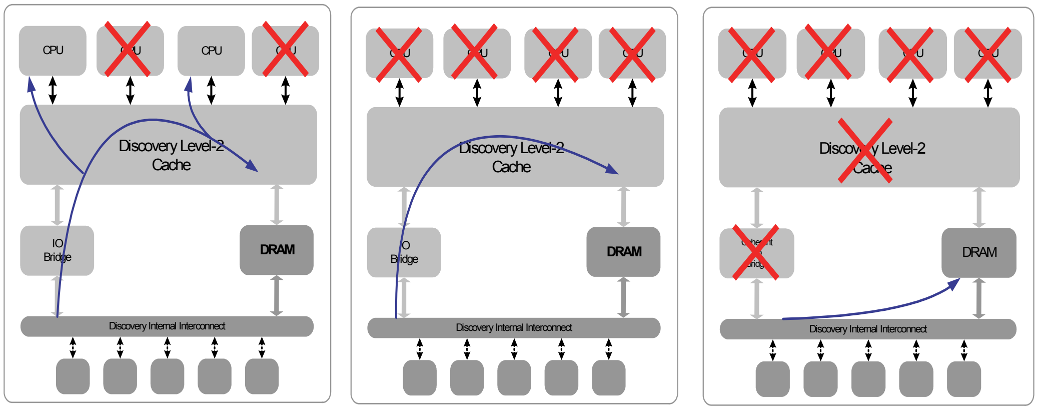
See Section 8.6, Power Management, on page 137 for more details.

34.4.3 I/O Coherency During Power Down Modes

The MV78230/78x60 manages I/O coherency with the CPU subsystem differently for each of the following scenarios:

- Part of the CPUs are in powered down—I/O transactions are routed upstream for snooping only the active CPUs and the L2 cache.
- All CPUs are is in power down but cache is powered on—I/O transactions are routed upstream for snooping only the shared L2 cache.
- The entire subsystem is powered down—I/O transactions are redirected to DRAM by Internal Interconnect.

Figure 142:I/O Coherency During Power Down Modes



The MV78230/78x60 manages I/O coherency with the CPU subsystem when it is powered down—I/O transactions are redirected to the DRAM by the Internal Interconnect.

34.5 Dynamic Frequency Scaling (DFS)

The device allows changing the CPU to a new target frequency in a specified flow triggered by software.

34.5.1 CPU DFS

Each CPU clock frequency can be dynamically changed.

When the CPU runs at full speed, the frequency is set by the initial reset strap setting and is controlled by the boot initialization.

It is also possible that the CPU clock frequency can be lowered to the Coherency Fabric clock frequency.

To set the new CPU frequency, complete the following steps:

1. Set the matching field to 1 in the <CpuClkDivResetMask> field in the CPU Divider Clock Control0 Register (Table 1471 p. 1542).
2. Program the target frequency PLL ratios, using the <DfsRatiox> field in the register for the relevant CPU.

To configure:	**Use register:**
CPU0	PMU DFS Control 1 Register (Table 1292 p. 1437)
CPU1	PMU DFS Control 2 Register (Table 1293 p. 1437)
CPU2	PMU DFS Control 3 Register (Table 1294 p. 1437)
CPU3	PMU DFS Control 4 Register (Table 1295 p. 1438)

Set the <DfsRatiox> based on the <DfsInitRatiox> setting in the register for the relevant CPU.

- If the <DfsInitRatiox> field is set to 1:1, set <DfsRatiox> to 1:2.
- If the <DfsInitRatiox> field is set to 1:2, set <DfsRatiox> to 1:4.

3. Enable the CPU DFS Done Interrupt.
4. Set <CPU DFS Request>.
5. Place the CPU in the Wait For Interrupt mode. When the CPU enters this state, the following steps occur:

Copyright © 2014 Marvell

a) CPU activity is halted, outstanding instructions are completed, and the write buffer is drained out.
b) The CPU clocks are halted.
c) PMU reloads new clocks ratio and set <CPU DFS Done> on completion; a completion interrupt is asserted to CPU.

34.6 Waking Options

The device provides several waking options sourced by different peripherals to take the device out of power save modes:

- Wake on LAN (wake up by specific packets coming from the Ethernet)
- Wake on USB
- Wake on Timer
- Wake on GPIO (an external event such as someone touching a touch screen etc)
- Wake on any other interrupt source

When the system is put in low power mode, it characterizes what events should wake it up and what traffic should be ignored and dropped. Notice that the same traffic may be accepted in normal operational mode, and ignored in low power state. The purpose is only to wake up the system on events that follow the system characterization as awaking events.

34.6.1 Wake On LAN (WOL) Mode

The WOL mode is an option for the system to wake up based on specific packets that enter the Ethernet port.

The Ethernet MAC provides the ability to filter out any packet that does not follow-up the waking packet definition. The Ethernet MAC can also accept the waking packet and wake up the system as a result of this packet.

Once set to WOL mode, the MAC discards all packets that do not pass the WOL filtering.

The device provides the following WOL filtering capability:

The Ethernet Controller MAC supports several filtering options that are useful for WOL. Each of these options is maskable:

- Magic packet (6 bytes of 0xff followed by 16 times the destination MAC address)
- ARP filter wake up by an ARP packet configurable IP address
- Wake up by a user-configured waking frame (three types)
- Wake up due to a change in the link status
- Wake up by a unicast to our MAC address that passed the port’s parsing options
- Wake up by a multicast packet that passed the port’s parsing options
- Wake up by specific configured EtherType (layer 2) or IP type (layer 3) packets (for example wake up by ping packet (IP type = ICMP), that passed the port’s RX parsing options)
- The TCAM provides excessive, flexible and configurable options for filtering out packets during power save modes. Lines in the TCAM may be configured to be operational only during power save state while other lines are operational only during normal operation mode. In low power mode, matching packets (TCAM hit) may be forwarded to memory and wake up the system, while non-matched packets will be dropped and maintain the system in power save mode.

For further information about WOL mode, see Section 15.9, Wake On LAN (WOL) Modes and Events, on page 287.

34.6.2 Wake On USB

The MV78230/78x60 device provides the option to wake up from one of the system power save modes by an event triggered on the USB interface.

When operated as a USB peripheral, the device may be put in suspend mode by the remote host. Suspend is signaled by 3 ms of idle time on the bus. When the remote host signals a resume or reset command on the bus, the USB peripheral unit takes the device out of suspend mode and will set a maskable interrupt when the link is operational to wake up the CPU.

When operating as a USB host and before entering CPU power down mode, the software may choose to put the USB port into low power state without disconnecting the link by signalling the connected devices to go into suspend mode.

Once the link is in suspend mode and CPU is powered down, there are several events that may restore the link:

- Remote resume request by a connected USB peripheral
- Connection of a new device
- Disconnection of a connected device

The USB host controller provides the option to disable waking up by any of the events through a software configuration setting. While sensing any of the enabled events, the USB host controller will initiate a maskable interrupt to wake up the CPU once there is a change in the link status.

Shutting down the PHY forces a link disconnect.

34.6.3 Wake on Timer

Before going into power down mode, the software may set one of the device’s timers to wake up the system after a preset time period. Once timer is expired, an interrupt will assert triggering a wake up event.

34.6.4 Wake on GPIO

External events are one of the most common way to wake up a powered down system, for example, touching a touch screen in a printer, a button that is pushed, a connected device that is connected over one of the powered down interfaces (e.g PCIe) and requests service and so on. When the system is powered down, any Interrupt enabled GPIO assertion will be treated as a waking event to the system.

34.6.5 Other Waking Events

Other system events that may be used as waking alerts are Mbus to DRAM requests for a specific master ID (instead of using the interrupt for reducing the wake latency). Any other system event that triggers an unmasked interrupt cause also wakes the CPU. This includes. for example, SD card insertion, doorbell interrupt in a multi-CPU environment, and MSIx message.

34.7 Thermal Management

The device incorporates a Thermal Management engine for monitoring die temperature. It includes an on-die analog-to-digital thermal sensor, that is used to determine when the maximum specified processor junction temperature has been reached.

Thermal Management functionality is controlled by the Thermal Manager Control and Status Register (Table 1460 p. 1535). The Thermal Manager is disabled by setting <TMDis>.

The current measured on-die temperature is digitally represented by 9-bit data format latched to the <ThermTempOut> field in the Thermal Manager Control and Status Register (Table 1460 p. 1535) for reading.

The Thermal sensor is active in all power modes.

Thermal Manager Interrupt Model

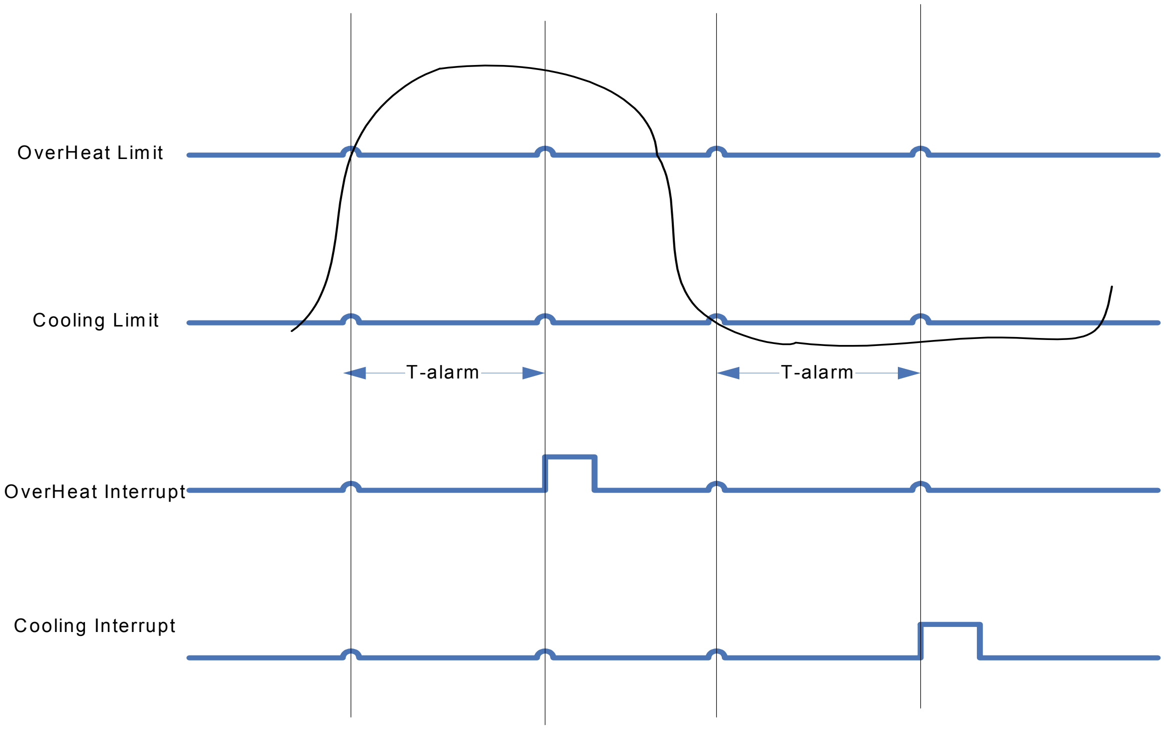
The Thermal Management Unit supplies an over-heating and cooling interrupt model to the software. The software must then direct the device to enter low-power operational modes to lower the die temperature on assertion of an over-heating interrupt, or to exit from the low-power operational state when a cooling interrupt is being asserted.

The <TMOverHeatThr> field in the Thermal Manager Control and Status Register (Table 1460 p. 1535) specifies the over-heat threshold of the on-die measured temperature. Once the measured temperature reaches the level above this value for an uninterrupted duration of <TMOverHitAlarmDelay>, a <TMOverHeatThr> interrupt is asserted, if enabled, informing the software to set the device to low-power mode.

The <TMCoolThr> field in the Thermal Manager Control and Status Register (Table 1460 p. 1535) specifies the cooling threshold of the on-die measured temperature. Once the measured temperature reaches the level below this value for an uninterrupted duration of <TMCoolingAlarmDelay>, a <TMCoolThr> interrupt is asserted, if enabled, informing the software to exit the device from low-power mode.

Figure 143 presents the Thermal Manager Interrupt model.

Figure 143:Thermal Manager Interrupt Model



34.8 Hardware Idle Profiling

The PMU includes hardware-based profilers that monitor the CPU idle time and generate an interrupt when the idle-time ratio crosses the high or low thresholds.

Each CPU includes an independent idle profiler counter, an independent timer, and independent high and low thresholds.

A high-/low-threshold interrupt may be used by software to trigger a CPU DFS procedure to switch between predefined operating frequencies. For example, when the CPU is in its high-frequency state and averaging more then 75% in Idle state, a CPU high-idle threshold interrupt can be generated to trigger CPU DFS to reduce the CPU speed to low-frequency state. When the CPU is in low-frequency state and averaging less then 25% in Idle state, an interrupt can be generated to trigger CPU DFS to restore the CPU to high-frequency state. Since CPU DFS can be conducted with a low latency, the hardware profiling is an attractive feature since it offers a mechanism that can be used effectively with little overhead. It reduces the dynamic power consumption of the device when it is not performing intensive tasks.

The registers that configures the PMU hardware idle profiling are the:

- Idle Profiler Configuration Registers
  - MC Idle Timer Value Register (Table 433 p. 761)
  - MC Idle High Threshold Register (Table 434 p. 761)
  - MC Idle Low Threshold Register (Table 435 p. 762)
  - MC Idle Result Register (Table 436 p. 762)
- Idle Profiler Interrupts: MC PM Event Status and Mask Register (Table 437 p. 762)
- Enable Idle Profiler: <MCStatCountEn> field in the MC Statistic Counter Global Control Register (Table 432 p. 761)

34.9 Unit Specific Power Management Options

The MV78230/78x60 supports the following power saving options in addition to and independent from the CPU's power save options:

- DRAM power save modes: Self refresh, active power down mode and precharged power down mode.
- All PCIe Device, Link, and ASPM Power Modes
- SERDES power down—Each SERDES may be shut down independently for zero current drawing.
- USB PHY can be either shutdown completely for zero current drawing or put in SUSPEND mode for auxiliary current drawing.
- Each unit has the option to gate its clocks and completely halt to save dynamic power.
- EEE (Energy Efficient Ethernet)— LPI mode. The capability to put the Ethernet PHY into a low power state and resume without breaking the link.

34.10 Debug During Power Management (CoreSight)

During Deep-IDLE state, the CPU Core's debug and tracing logic is powered down as well. Any debug or tracing attempt results in slave error indication from the CoreSight DAP controller.

Copyright © 2014 Marvell

34.11 Power Management Registers

Table 152 lists the register fields used for CPU subsystem Power Management.

Table 152: CPU Core Power Management Registers Pointers

Register Field Function	Register Mapping
Snoop CPU Enable	<SnoopCPU0 En> field in the Coherency Fabric Control Register (Table 155 p. 612) <SnoopCPU1 En> field in the Coherency Fabric Control Register (Table 155 p. 611) <SnoopCPU2 En> field in the Coherency Fabric Control Register (Table 155 p. 611) <SnoopCPU3 En> field in the Coherency Fabric Control Register (Table 155 p. 610)
DVM Snoop CPU Enable	**NOTE:** This will be included in a future revision of the device.
Snoop Filter Enable	<SF Enable> field in the Snoop Filter Control Register (N=0–3) (Table 245 p. 659)
Invalidate Snoop Filter	<SF Invalidate> field in the Snoop Filter Control Register (N=0–3) (Table 245 p. 659)
CPU DFS Request	The <CPUnDfsReq> field (where n=0–3) in: • CPU0 PM Control and Configuration Register (Table 390 p. 740) • CPU1 PM Control and Configuration Register (Table 400 p. 745) • CPU2 PM Control and Configuration Register (Table 410 p. 749) • CPU3 PM Control and Configuration Register (Table 420 p. 754)
CPU DFS Done	The <CPUnDfsDone> field (where n=0–3) in: • CPU0 PM Event Status and Mask Register (Table 396 p. 743) • CPU1 PM Event Status and Mask Register (Table 406 p. 748) • CPU2 PM Event Status and Mask Register (Table 416 p. 753) • CPU3 PM Event Status and Mask Register (Table 426 p. 757)
Global CPU IRQ Mask	The <CPUnIntr0Mask> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)
Global CPU FIQ Mask	The <CPUnIntr1Mask> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)
L2 Power Down Request	The <CPUnL2FabPwrDwnEn> field (where n=0–3) in: • CPU0 PM Control and Configuration Register (Table 390 p. 740) • CPU1 PM Control and Configuration Register (Table 400 p. 745) • CPU2 PM Control and Configuration Register (Table 410 p. 749) • CPU3 PM Control and Configuration Register (Table 420 p. 754)
Enable IRQ Wake Up	The <CPUnWake0Mask> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)
Enable FIQ Wake Up	The <CPUnWake1Mask> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)

Table 152: CPU Core Power Management Registers Pointers (Continued)

Register Field Function	Register Mapping
Enable Debug Wake Up	The <CPUnWake2Mask> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)
CPU Idle Wait Mask	The <CPUnIdle0Mask> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)
Snoop Queue Empty Wait Mask	The <CPUnIdle1Mask> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)
Deep Idle Power Enable	The <CPUnPowerDownReq> field (where n=0–3) in: • CPU0 PM Control and Configuration Register (Table 390 p. 740) • CPU1 PM Control and Configuration Register (Table 400 p. 745) • CPU2 PM Control and Configuration Register (Table 410 p. 749) • CPU3 PM Control and Configuration Register (Table 420 p. 754)
CPU Power Down Status	The <CPUnPwrDwnStts> field (where n=0–3) in: • CPU0 PM Status and Mask Register (Table 391 p. 741) • CPU1 PM Status and Mask Register (Table 401 p. 745) • CPU2 PM Status and Mask Register (Table 411 p. 750) • CPU3 PM Status and Mask Register (Table 421 p. 755)

35 Adaptive Voltage Scaling (AVS)

This section describes the Adaptive voltage scaling (AVS) unit.

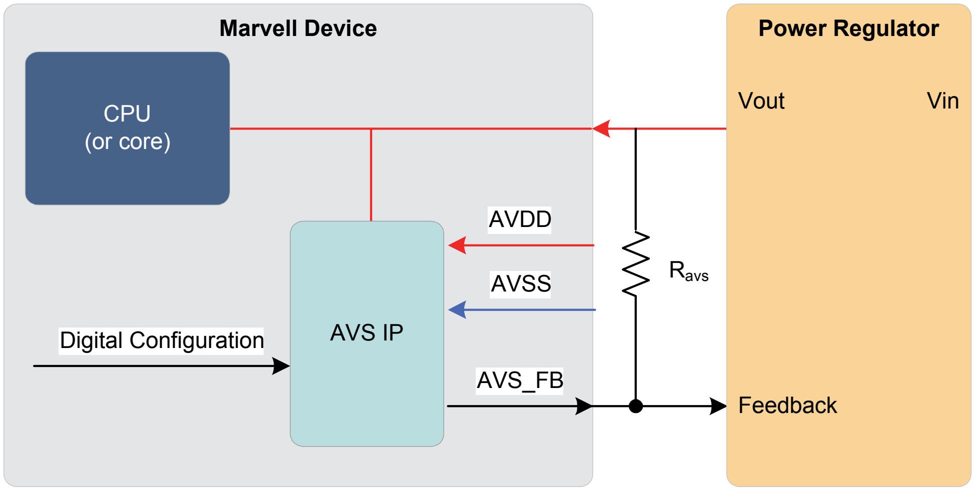
The device integrates 2 AVS units.

- One unit controls the CPU voltage (VDD_CPU) via the CPU_AVS_FB signal.
- The other unit control the Core voltage (VDD) via the CORE_AVS_FB signal.

Each AVS unit is used to manipulating the external power regulator output voltage. The AVS unit samples the on-die voltage levels of the device and uses the relevant AVS_FB signal to fine tune the output voltage levels of the power regulator.

Figure 144 provides a diagram illustrating the AVS unit.

Figure 144:AVS Block Diagram



NOTE: Refer to the device Hardware Design Guide for additional information about the AVS unit.

35.1 Features

This unit integrates the following features:

- On-die voltage level sampling
- Required on-die voltage level setting

35.2 High-Level Functional Description

The AVS unit samples the controlled on-die voltage level, and controls the power regulator feedback pin. This enables the design to achieve the desired on-die voltage.

- The usage of Core AVS is required when the DDR is operating at higher frequencies. Refer to the device Hardware Specifications for the frequencies at which Core AVS is required.
- The usage of CPU AVS is required when the system is operating at higher frequencies. Refer to the device Hardware Specifications for the frequencies at which CPU AVS is required.

The AVS registers are provided in Appendix A.25, Core and CPU AVS Registers, on page 1507.

THIS PAGE IS INTENTIONALLY LEFT BLANK

MV78230, MV78260, and MV78460

ARMADA® XP Family of Highly Integrated Multi-Core ARMv7 Based SoC Processors

Functional Specifications – Unrestricted

Appendix A: Register Set

Doc. No. MV-S107021-U0, Rev. A

May 29, 2014, Preliminary

Marvell. Moving Forward Faster

Document Classification: Proprietary Information

THIS PAGE IS INTENTIONALLY LEFT BLANK

List of Registers

Copyright © 2014 Marvell

Copyright © 2014 Marvell

Copyright © 2014 Marvell

Copyright © 2014 Marvell

A.3 DRAM Controller Registers 763

Copyright © 2014 Marvell

A.7 Buffer Management Controller Registers 1035

Copyright © 2014 Marvell

Copyright © 2014 Marvell

Copyright © 2014 Marvell

##

Copyright © 2014 Marvell

A.12 Serial Peripheral Interface (SPI) Registers 1278

A.16 UART Registers 1401

A.17 Real Time Clock (RTC) Registers 1414

Copyright © 2014 Marvell

Copyright © 2014 Marvell

Document Classification: Proprietary Information
Copyright © 2014 Marvell

A.24 Electronic Fuse (eFuse) Registers 1501

A.25 Core and CPU AVS Registers 1507

Copyright © 2014 Marvell

A MV78230/78x60 Register Set

This section details the MV78230/78x60 registers.

A.1 Register Description

The device internal registers are mapped into a 1 MB address window. However, only part of this space is really populated with registers. The chip registers are distributed among the device's different units (distributed register file). Each unit has its own 64 KB register file space.

All registers are 32 bits wide [31:0]. The device registers use Little Endian byte orientation in which the Most Significant Byte (MSB) of a multi-byte expression is located in the highest address. So, for example, if a register value is 0xAABBCCDD (bits[31:24] are 0xAA), a write of single byte 0x77 to address 0x1, results in a new register value of 0xAABB77DD. The bits within a given byte are always ordered so that bit[7] is the Most Significant bit (MSb) and bit[0] is the Least Significant bit (LSb).

Unless otherwise noted, the registers support byte, half-word, and word accesses.

A.1.1 Register Field Type Codes

The MV78230/78x60 registers are made up of up to 32-bit fields, where each field is associated with one or more bits. Each of these register fields have a unique programming functionality and their operation is defined by the field's type. The following list describes the function of each type:

Table 153: Standard Register Field Type Codes

Type	Description
LH	Register field with latching high function. If status is high, then the register is set to one and remains set until a read operation is performed through the management interface or a reset occurs.
LL	Register field with latching low function. If status is low, then the register is cleared to zero and remains cleared until a read operation is performed through the management interface or a reset occurs.
Retain	The register value is retained after software reset is executed.
RO	Read Only. Writing to this type of field may cause unpredictable results.
ROC	Read Only Clear. After read, register field is cleared to zero.
RSVD	Reserved for future use. All reserved bits are read as zero unless otherwise noted.
RW	Read and Write.
RW0C	Read-only status, Write-0 to clear status register. Register bits indicate status when read, a set bit indicates a status event may be cleared by writing a 0. Writing a 1 to RW0C bits have no effect.
RW1C	Read-only status, Write-1 to clear status register. Register bits indicate status when read, a set bit indicates a status event may be cleared by writing a 1. Writing a 0 to RW1C bits have no effect.
SC	Self-Clear. Writing a one to this register causes the desired function to be immediately executed, then the register field is cleared to zero when the function is complete.

Copyright © 2014 Marvell

Table 153: Standard Register Field Type Codes (Continued)

Type	Description
Special	Used in special cases where the register type functionality does not conform to any of the standard types listed in this table. Refer to the Description Column of the register table for a full definition of the field/register type and functionality.
Update	Value written to the register field does not take effect until a software reset is executed. The value can still be read after it is written.
WO	Write Only. A write to the register field will trigger an internal function and a read will return an undefined value.
X	These bits do not exist.

NOTE: If the type is None, the register is not initialized.

A.2 CPU Sub-System Registers

The following table provides a summarized list of all registers that belong to the CPU Sub-System, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 154: Summary Map Table for the CPU Sub-System Registers

Register Name	Offset	Table and Page
Coherent Fabric Control and Status		
Coherency Fabric Control Register	0x00020200	Table 155, p. 610
Coherency Fabric Configuration Register	0x00020204	Table 156, p. 612
Events Affinity Register	0x00020208	Table 157, p. 613
IO Snoop Affinity Register	0x0002020C	Table 158, p. 614
CDI Configuration Register	0x00020220	Table 159, p. 615
CFU Configuration Register	0x00020228	Table 160, p. 615
SRAM Window n Control Register (n=0–3)	SRAM Window ID0: 0x00020240, SRAM Window ID1: 0x00020244, SRAM Window ID2: 0x00020248, SRAM Window ID3: 0x0002024C	Table 161, p. 617
Coherency Fabric Error Status Register	0x00020258	Table 162, p. 617
Coherency Fabric Error Mask Register	0x0002025C	Table 163, p. 620
Coherency Fabric Local Cause Register	0x00020260	Table 164, p. 620
Coherent IO Bridge Control and Status		
CIB Control and Configuration Register	0x00020280	Table 165, p. 622
CIB Non L1 Snoopable Window0 Control Register	0x00020284	Table 166, p. 623
CIB Non L1 Snoopable Window0 Base Address Register	0x00020288	Table 167, p. 624
CIB Non L1 Snoopable Window1 Control Register	0x0002028C	Table 168, p. 624
CIB Non L1 Snoopable Window1 Base Address Register	0x00020290	Table 169, p. 624
CIB Read Buffer Select Register	0x00020294	Table 170, p. 625
CIB Parity Error Status Register	0x00020298	Table 171, p. 625
SDRAM Address Decoding		
Win 0 Base Address Register	0x00020180	Table 172, p. 625
Win 0 Control Register	0x00020184	Table 173, p. 626
Win 1 Base Address Register	0x00020188	Table 174, p. 626
Win 1 Control Register	0x0002018C	Table 175, p. 627
Win 2 Base Address Register	0x00020190	Table 176, p. 627
Win 2 Control Register	0x00020194	Table 177, p. 628
Win3 Base Address Register	0x00020198	Table 178, p. 628
Win 3 Control Register	0x0002019C	Table 179, p. 629
Mbus Unit Address Decoding		
Window0 Control Register	0x00020000	Table 180, p. 630

Copyright © 2014 Marvell

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
Window0 Base Register	0x00020004	Table 181, p. 631
Window0 Remap Low Register	0x00020008	Table 182, p. 631
Window0 Remap High Register	0x0002000C	Table 183, p. 631
Window1 Control Register	0x00020010	Table 184, p. 632
Window1 Base Register	0x00020014	Table 185, p. 632
Window1 Remap Low Register	0x00020018	Table 186, p. 633
Window1 Remap High Register	0x0002001C	Table 187, p. 633
Window2 Control Register	0x00020020	Table 188, p. 633
Window2 Base Register	0x00020024	Table 189, p. 634
Window2 Remap Low Register	0x00020028	Table 190, p. 634
Window2 Remap High Register	0x0002002C	Table 191, p. 634
Window3 Control Register	0x00020030	Table 192, p. 634
Window3 Base Register	0x00020034	Table 193, p. 635
Window3 Remap Low Register	0x00020038	Table 194, p. 635
Window3 Remap High Register	0x0002003C	Table 195, p. 636
Window4 Control Register	0x00020040	Table 196, p. 636
Window4 Base Register	0x00020044	Table 197, p. 636
Window4 Remap Low Register	0x00020048	Table 198, p. 637
Window4 Remap High Register	0x0002004C	Table 199, p. 637
Window5 Control Register	0x00020050	Table 200, p. 637
Window5 Base Register	0x00020054	Table 201, p. 638
Window5 Remap Low Register	0x00020058	Table 202, p. 638
Window5 Remap High Register	0x0002005C	Table 203, p. 639
Window6 Control Register	0x00020060	Table 204, p. 639
Window6 Base Register	0x00020064	Table 205, p. 639
Window6 Remap Low Register	0x00020068	Table 206, p. 640
Window6 Remap High Register	0x0002006C	Table 207, p. 640
Window7 Control Register	0x00020070	Table 208, p. 640
Window7 Base Register	0x00020074	Table 209, p. 641
Window7 Remap Low Register	0x00020078	Table 210, p. 641
Window7 Remap High Register	0x0002007C	Table 211, p. 642
Internal Registers Base Address	0x00020080	Table 212, p. 642
Internal Units Sync Barrier Control Register	0x00020084	Table 213, p. 642
Window8 Control Register	0x00020090	Table 214, p. 642
Window8 Base Register	0x00020094	Table 215, p. 643
Window9 Control Register	0x00020098	Table 216, p. 644
Window9 Base Register	0x0002009C	Table 217, p. 644
Window10 Control Register	0x000200A0	Table 218, p. 645
Window10 Base Register	0x000200A4	Table 219, p. 645
Window11 Control Register	0x000200A8	Table 220, p. 646

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
Window11 Base Register	0x000200AC	Table 221, p. 646
Window12 Control Register	0x000200B0	Table 222, p. 647
Window12 Base Register	0x000200B4	Table 223, p. 647
Window13 Control Register	0x000200B8	Table 224, p. 648
Window13 Base Register	0x000200BC	Table 225, p. 648
Window14 Control Register	0x000200C0	Table 226, p. 649
Window14 Base Register	0x000200C4	Table 227, p. 649
Window15 Control Register	0x000200C8	Table 228, p. 650
Window15 Base Register	0x000200CC	Table 229, p. 650
Window16 Control Register	0x000200D0	Table 230, p. 651
Window16 Base Register	0x000200D4	Table 231, p. 651
Window17 Control Register	0x000200D8	Table 232, p. 652
Window17 Base Register	0x000200DC	Table 233, p. 652
Window18 Control Register	0x000200E0	Table 234, p. 653
Window18 Base Register	0x000200E4	Table 235, p. 653
Window19 Control Register	0x000200E8	Table 236, p. 654
Window19 Base Register	0x000200EC	Table 237, p. 654
Window13 Remap Low Register	0x000200F0	Table 238, p. 655
Window13 Remap High Register	0x000200F4	Table 239, p. 655
Mbus Bridge Window Control Register	0x00020250	Table 240, p. 655
Mbus Bridge Window Base Register	0x00020254	Table 241, p. 656
CPU		
CPU Configuration Register (N=0–3)	CPU ID0: 0x00021800, CPU ID1: 0x00021900, CPU ID2: 0x00021A00, CPU ID3: 0x00021B00	Table 242, p. 656
CPU Control and Status Register (N=0–3)	CPU ID0: 0x00021808, CPU ID1: 0x00021908, CPU ID2: 0x00021A08, CPU ID3: 0x00021B08	Table 243, p. 658
CPU IO Sync Barrier Control Register (N=0–3)	CPU ID0: 0x00021810, CPU ID1: 0x00021910, CPU ID2: 0x00021A10, CPU ID3: 0x00021B10	Table 244, p. 658
Snoop Filter Control Register (N=0–3)	CPU ID0: 0x00021820, CPU ID1: 0x00021920, CPU ID2: 0x00021A20, CPU ID3: 0x00021B20	Table 245, p. 659
Snoop Filter Interrupt Cause Register (N=0–3)	CPU ID0: 0x00021828, CPU ID1: 0x00021928, CPU ID2: 0x00021A28, CPU ID3: 0x00021B28	Table 246, p. 659

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
Snoop Filter Interrupt Mask Register (N=0–3)	CPU ID0: 0x0002182C, CPU ID1: 0x0002192C, CPU ID2: 0x00021A2C, CPU ID3: 0x00021B2C	Table 247, p. 660
Timers Control Register (N=0–3)	CPU ID0: 0x00021840, CPU ID1: 0x00021940, CPU ID2: 0x00021A40, CPU ID3: 0x00021B40	Table 248, p. 660
Timer0 Reload Register (N=0–3)	CPU ID0: 0x00021850, CPU ID1: 0x00021950, CPU ID2: 0x00021A50, CPU ID3: 0x00021B50	Table 249, p. 662
Timer 0 Register (N=0–3)	CPU ID0: 0x00021854, CPU ID1: 0x00021954, CPU ID2: 0x00021A54, CPU ID3: 0x00021B54	Table 250, p. 662
Timer1 Reload Register (N=0–3)	CPU ID0: 0x00021858, CPU ID1: 0x00021958, CPU ID2: 0x00021A58, CPU ID3: 0x00021B58	Table 251, p. 663
Timer 1 Register (N=0–3)	CPU ID0: 0x0002185C, CPU ID1: 0x0002195C, CPU ID2: 0x00021A5C, CPU ID3: 0x00021B5C	Table 252, p. 663
Watchdog Timer Reload Register (N=0–3)	CPU ID0: 0x00021860, CPU ID1: 0x00021960, CPU ID2: 0x00021A60, CPU ID3: 0x00021B60	Table 253, p. 663
Watchdog Timer Register (N=0–3)	CPU ID0: 0x00021864, CPU ID1: 0x00021964, CPU ID2: 0x00021A64, CPU ID3: 0x00021B64	Table 254, p. 663
Timers Events Status register Register (N=0–3)	CPU ID0: 0x00021868, CPU ID1: 0x00021968, CPU ID2: 0x00021A68, CPU ID3: 0x00021B68	Table 255, p. 664
Cache Lockdown		
L2 CPUn Data Lockdown Register (n=0–3)	CPU ID0: 0x00008900, CPU ID1: 0x00008908, CPU ID2: 0x00008910, CPU ID3: 0x00008918	Table 256, p. 664
L2 CPUn Instruction Lockdown Register (n=0–3)	CPU ID0: 0x00008904, CPU ID1: 0x0000890C, CPU ID2: 0x00008914, CPU ID3: 0x0000891C	Table 257, p. 667
IO Bridge Lockdown Register	0x00008984	Table 258, p. 669
Cache Error Control		
L2 ECC Error Count Register	0x00008600	Table 259, p. 672
L2 ECC Error Threshold Register	0x00008604	Table 260, p. 673

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
L2 Error Attr Capture Register	0x00008608	Table 261, p. 673
L2 Error Address Capture Register	0x0000860C	Table 262, p. 674
L2 Error Way Set Capture Register	0x00008610	Table 263, p. 675
L2 Error Injection Control Register	0x00008614	Table 264, p. 675
L2 ECC Error Injection Mask Register	0x00008618	Table 265, p. 676
Cache Power Management		
L2 Dynamic Memory Power Down Register	0x00008A00	Table 266, p. 676
L2 Dynamic Clock Gating Register	0x00008A04	Table 267, p. 676
L2 Force Power Down Register	0x00008A08	Table 268, p. 677
Cache Maintenance		
L2 Sync Barrier Register	0x00008700	Table 269, p. 677
L2 Maintenance Status Register	0x00008704	Table 270, p. 677
L2 Range Base Address CPU n Register (n=0–3)	CPU_ID0: 0x00008710, CPU_ ID1: 0x00008714, CPU_ ID2: 0x00008718, CPU_ ID3: 0x0000871C	Table 271, p. 678
L2 Range Base Address Virtual Register	0x00008720	Table 272, p. 679
L2 Block Allocation Extend Addr Virtual Register	0x00008724	Table 273, p. 679
L2 Block Allocation Extend Addr CPU n Register (n=0–3)	CPU_ID0: 0x00008740, CPU_ ID1: 0x00008744, CPU_ ID2: 0x00008748, CPU_ ID3: 0x0000874C	Table 274, p. 679
L2 Invalidate by Physical Address Register	0x00008770	Table 275, p. 680
L2 Invalidate Range Register	0x00008774	Table 276, p. 680
L2 Invalidate by Index Way Register	0x00008778	Table 277, p. 680
L2 Invalidate by Way Register	0x0000877C	Table 278, p. 681
L2 Block Allocation Register	0x0000878C	Table 279, p. 681
L2 Clean by Physical Address Register	0x000087B0	Table 280, p. 682
L2 Clean by Range Register	0x000087B4	Table 281, p. 683
L2 Clean by Index Way Register	0x000087B8	Table 282, p. 683
L2 Clean by Way Register	0x000087BC	Table 283, p. 684
L2 Flush by Physical Address Register	0x000087F0	Table 284, p. 684
L2 Flush by Range Register	0x000087F4	Table 285, p. 685
L2 Flush by Index Way Register	0x000087F8	Table 286, p. 685
L2 Flush by Way Register	0x000087FC	Table 287, p. 686
Cache Performance Monitor		
L2 Counters Control Register	0x00008200	Table 288, p. 686
L2 Counter <X>_ Configuration Register (X=0–1)	Counter ID0: 0x00008204, Counter ID1: 0x00008210	Table 289, p. 687
L2 Counter X Value Low Register (X=0–1)	Counter ID0: 0x00008208, Counter ID1: 0x00008214	Table 290, p. 689

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
L2 Counter X Value High Register (X=0–1)	Counter ID0: 0x0000820C, Counter ID1: 0x00008218	Table 291, p. 689
Cache Interrupt Control		
L2 Interrupt Cause Register	0x00008220	Table 292, p. 689
L2 Interrupt Mask Register	0x00008224	Table 293, p. 690
Cache Main Control		
L2 Control Register	0x00008100	Table 294, p. 690
L2 Auxiliary Control Register	0x00008104	Table 295, p. 691
Main Interrupt Controller Control and Configuration		
MPIC Control Register	0x00020A00	Table 296, p. 694
Software Triggered Interrupt Register	0x00020A04	Table 297, p. 694
SOC Main Interrupt Error Cause Register	0x00020A20	Table 298, p. 696
Interrupt Set Enable (ISE) Register	0x00020A30	Table 299, p. 696
Interrupt Clear Enable (ICE) Register	0x00020A34	Table 300, p. 696
Interrupt Source i Control Register (i=0–28)	Interrupt ID0: 0x00020B00, Interrupt ID1: 0x00020B04... Interrupt ID28: 0x00020B70	Table 301, p. 697
Interrupt Source i Control Register (i=29–115)	Interrupt ID29: 0x00020B74, Interrupt ID30: 0x00020B78... Interrupt ID115: 0x00020CCC	Table 302, p. 699
Per CPU Interrupt Controller Control and Configuration		
Outbound Doorbell Register (N=0–3)	CPU ID0: 0x00021870, CPU ID1: 0x00021970, CPU ID2: 0x00021A70, CPU ID3: 0x00021B70	Table 303, p. 701
Outbound Doorbell Mask Register (N=0–3)	CPU ID0: 0x00021874, CPU ID1: 0x00021974, CPU ID2: 0x00021A74, CPU ID3: 0x00021B74	Table 304, p. 701
Inbound Doorbell Register (N=0–3)	CPU ID0: 0x00021878, CPU ID1: 0x00021978, CPU ID2: 0x00021A78, CPU ID3: 0x00021B78	Table 305, p. 701
Inbound Doorbell Mask Register (N=0–3)	CPU ID0: 0x0002187C, CPU ID1: 0x0002197C, CPU ID2: 0x00021A7C, CPU ID3: 0x00021B7C	Table 306, p. 702
Main Interrupt Cause (Vector 0) Register (N=0–3)	CPU ID0: 0x00021880, CPU ID1: 0x00021980, CPU ID2: 0x00021A80, CPU ID3: 0x00021B80	Table 307, p. 702
Main Interrupt Cause (Vector 1) Register (N=0–3)	CPU ID0: 0x00021884, CPU ID1: 0x00021984, CPU ID2: 0x00021A84, CPU ID3: 0x00021B84	Table 308, p. 702

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
Main Interrupt Cause (Vector 2) Register (N=0–3)	CPU ID0: 0x00021888, CPU ID1: 0x00021988, CPU ID2: 0x00021A88, CPU ID3: 0x00021B88	Table 309, p. 702
Main Interrupt Cause (Vector 3) Register (N=0–3)	CPU ID0: 0x0002188C, CPU ID1: 0x0002198C, CPU ID2: 0x00021A8C, CPU ID3: 0x00021B8C	Table 310, p. 703
IRQ Select Cause Register (N=0–3)	CPU ID0: 0x000218A0, CPU ID1: 0x000219A0, CPU ID2: 0x00021AA0, CPU ID3: 0x00021BA0	Table 311, p. 703
FIQ Select Cause Register (N=0–3)	CPU ID0: 0x000218A4, CPU ID1: 0x000219A4, CPU ID2: 0x00021AA4, CPU ID3: 0x00021BA4	Table 312, p. 704
Current Task Priority (CTP) Register (N=0–3)	CPU ID0: 0x000218B0, CPU ID1: 0x000219B0, CPU ID2: 0x00021AB0, CPU ID3: 0x00021BB0	Table 313, p. 705
IRQ Interrupt Acknowledge (IIACK) Register (N=0–3)	CPU ID0: 0x000218B4, CPU ID1: 0x000219B4, CPU ID2: 0x00021AB4, CPU ID3: 0x00021BB4	Table 314, p. 705
Interrupt Set Mask (ISM) Register (N=0–3)	CPU ID0: 0x000218B8, CPU ID1: 0x000219B8, CPU ID2: 0x00021AB8, CPU ID3: 0x00021BB8	Table 315, p. 706
Interrupt Clear Mask (ICM) Register (N=0–3)	CPU ID0: 0x000218BC, CPU ID1: 0x000219BC, CPU ID2: 0x00021ABC, CPU ID3: 0x00021BBC	Table 316, p. 706
SOC Main Interrupt Error Mask Register (N=0–3)	CPU ID0: 0x000218C0, CPU ID1: 0x000219C0, CPU ID2: 0x00021AC0, CPU ID3: 0x00021BC0	Table 317, p. 707
Coherency Fabric Local Interrupt Mask Register (N=0–3)	CPU ID0: 0x000218C4, CPU ID1: 0x000219C4, CPU ID2: 0x00021AC4, CPU ID3: 0x00021BC4	Table 318, p. 707
End Point Interrupt Control		
Main Interrupt Cause (Vec 0) Register	0x00020900	Table 319, p. 707
Main Interrupt Cause (Vec 1) Register	0x00020904	Table 320, p. 707
Main Interrupt Cause (Vec 2) Register	0x00020908	Table 321, p. 708
Main Interrupt Cause (Vec 3) Register	0x0002090C	Table 322, p. 708
Endpoint Sel Cause Register	0x00020928	Table 323, p. 708
EP Coherency Fabric Local Interrupt Mask Register	0x00020940	Table 324, p. 709
EP SOC Main Interrupt Error Mask Register	0x00020944	Table 325, p. 709

Copyright © 2014 Marvell

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
Shared Inbound Doorbell		
Shared Inbound Doorbell0 Register	0x00020400	Table 326, p. 710
Shared Inbound Doorbell1 Register	0x00020404	Table 327, p. 710
Shared Inbound Doorbell2 Register	0x00020408	Table 328, p. 710
Global Timers		
Timers Control Register	0x00020300	Table 329, p. 710
Timers Events Status register Register	0x00020304	Table 330, p. 713
Timer0 Reload Register	0x00020310	Table 331, p. 714
Timer 0 Register	0x00020314	Table 332, p. 714
Timer1 Reload Register	0x00020318	Table 333, p. 714
Timer 1 Register	0x0002031C	Table 334, p. 715
Timer2 Reload Register	0x00020320	Table 335, p. 715
Timer 2 Register	0x00020324	Table 336, p. 715
Timer3 Reload Register	0x00020328	Table 337, p. 715
Timer 3 Register	0x0002032C	Table 338, p. 715
Watchdog Timer Reload Register	0x00020330	Table 339, p. 716
Watchdog Timer Register	0x00020334	Table 340, p. 716
ARM Generic Timer Control Register	0x00020340	Table 341, p. 716
ARM Generic Timer Status Register	0x00020344	Table 342, p. 716
ARM Generic Timer Low Register	0x00020348	Table 343, p. 717
ARM Generic Timer High Register	0x0002034C	Table 344, p. 717
Semaphores		
Semaphore0 Register	0x00020500	Table 345, p. 717
Semaphore%<n> Register (n=1–127)	Semaphore1: 0x00020504, Semaphore2: 0x00020508... Semaphore127: 0x000206FC	Table 346, p. 718
Clocks Resets and Power Management		
WD RSTOUTn Mask Register	0x00020704	Table 347, p. 718
WD Interrupt Mask Register	0x00020708	Table 348, p. 719
CPUn SW Reset Control Register (n=0–3)	CPU ID0: 0x00020800, CPU ID1: 0x00020808, CPU ID2: 0x00020810, CPU ID3: 0x00020818	Table 349, p. 720
General Purpose SW usage 0 Register	0x00020980	Table 350, p. 720
General Purpose SW usage 1 Register	0x00020984	Table 351, p. 721
General Purpose SW usage 2 Register	0x00020988	Table 352, p. 721
General Purpose SW usage 3 Register	0x0002098C	Table 353, p. 721
AVS Control and Status		
AVS Control 0 Register	0x00020860	Table 354, p. 721
AVS Control 2 Register	0x00020868	Table 355, p. 722
Arbitration Control		

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
CFU BW Control Limits Register	0x00020440	Table 356, p. 722
CFU BW Control Threshold Register	0x00020444	Table 357, p. 724
Arbitration Queue Load Control Register	0x00020448	Table 358, p. 724
CoreSight Control		
Coresight Components Base Address Register	0x00020D00	Table 359, p. 726
Coresight CTI Mask Register	0x00020D08	Table 360, p. 727
Coresight Status Register	0x00020D0C	Table 361, p. 727
CoreSight Timestamp		
Timestamp Counter Enable Register	0x00023000	Table 362, p. 728
Timestamp Counter Preload 1 Register	0x00023004	Table 363, p. 728
Timestamp Counter Preload 2 Register	0x00023008	Table 364, p. 728
Timestamp Claim Tag Set Register	0x00023FA0	Table 365, p. 728
Timestamp Claim Tag Clear Register	0x00023FA4	Table 366, p. 729
Timestamp Lock Access Register	0x00023FB0	Table 367, p. 729
Timestamp Lock Status Register	0x00023FB4	Table 368, p. 729
Timestamp Authentication Status Register	0x00023FB8	Table 369, p. 729
Timestamp Device ID Register	0x00023FC8	Table 370, p. 729
Timestamp Device Type Identifier Register	0x00023FCC	Table 371, p. 730
Timestamp Peripheral ID4 Register	0x00023FD0	Table 372, p. 730
Timestamp Peripheral ID5 Register	0x00023FD4	Table 373, p. 730
Timestamp Peripheral ID6 Register	0x00023FD8	Table 374, p. 730
Timestamp Peripheral ID7 Register	0x00023FDC	Table 375, p. 730
Timestamp Peripheral ID0 Register	0x00023FE0	Table 376, p. 731
Timestamp Peripheral ID1 Register	0x00023FE4	Table 377, p. 731
Timestamp Peripheral ID2 Register	0x00023FE8	Table 378, p. 731
Timestamp Peripheral ID3 Register	0x00023FEC	Table 379, p. 731
Timestamp Component ID0 Register	0x00023FF0	Table 380, p. 731
Timestamp Component ID1 Register	0x00023FF4	Table 381, p. 732
Timestamp Component ID2 Register	0x00023FF8	Table 382, p. 732
Timestamp Component ID3 Register	0x00023FFC	Table 383, p. 732
Priority and Group Control		
Mbus Units Priority Control Register	0x00020420	Table 384, p. 732
Fabric Units Priority Control Register	0x00020424	Table 385, p. 734
Mbus Units Prefetch Control Register	0x00020428	Table 386, p. 735
Fabric Units Prefetch Control Register	0x0002042C	Table 387, p. 738
Power Management Service Unit		
PMU General		
Mbus to DDR Request ID Mask Register	0x00022F10	Table 388, p. 739
CPU0 PM Service		
CPU0 Statistic Counter Global Control Register	0x00022100	Table 389, p. 740

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
CPU0 PM Control and Configuration Register	0x00022104	Table 390, p. 740
CPU0 PM Status and Mask Register	0x0002210C	Table 391, p. 741
CPU0 Idle Timer Value Register	0x00022110	Table 392, p. 742
CPU0 Idle High Threshold Register	0x00022114	Table 393, p. 742
CPU0 Idle Low Threshold Register	0x00022118	Table 394, p. 743
CPU0 Idle Result Register	0x0002211C	Table 395, p. 743
CPU0 PM Event Status and Mask Register	0x00022120	Table 396, p. 743
CPU0 Boot Address Redirect Register	0x00022124	Table 397, p. 744
CPU0 Resume Control Register	0x00022128	Table 398, p. 744
CPU1 PM Service		
CPU1 Statistic Counter Global Control Register	0x00022200	Table 399, p. 744
CPU1 PM Control and Configuration Register	0x00022204	Table 400, p. 745
CPU1 PM Status and Mask Register	0x0002220C	Table 401, p. 745
CPU1 Idle Timer Value Register	0x00022210	Table 402, p. 747
CPU1 Idle High Threshold Register	0x00022214	Table 403, p. 747
CPU1 Idle Low Threshold Register	0x00022218	Table 404, p. 747
CPU1 Idle Result Register	0x0002221C	Table 405, p. 748
CPU1 PM Event Status and Mask Register	0x00022220	Table 406, p. 748
CPU1 Boot Address Redirect Register	0x00022224	Table 407, p. 749
CPU1 Resume Control Register	0x00022228	Table 408, p. 749
CPU2 PM Service		
CPU2 Statistic Counter Global Control Register	0x00022300	Table 409, p. 749
CPU2 PM Control and Configuration Register	0x00022304	Table 410, p. 749
CPU2 PM Status and Mask Register	0x0002230C	Table 411, p. 750
CPU2 Idle Timer Value Register	0x00022310	Table 412, p. 752
CPU2 Idle High Threshold Register	0x00022314	Table 413, p. 752
CPU2 Idle Low Threshold Register	0x00022318	Table 414, p. 752
CPU2 Idle Result Register	0x0002231C	Table 415, p. 752
CPU2 PM Event Status and Mask Register	0x00022320	Table 416, p. 753
CPU2 Boot Address Redirect Register	0x00022324	Table 417, p. 753
CPU2 Resume Control Register	0x00022328	Table 418, p. 754
CPU3 PM Service		
CPU3 Statistic Counter Global Control Register	0x00022400	Table 419, p. 754
CPU3 PM Control and Configuration Register	0x00022404	Table 420, p. 754
CPU3 PM Status and Mask Register	0x0002240C	Table 421, p. 755
CPU3 Idle Timer Value Register	0x00022410	Table 422, p. 756
CPU3 Idle High Threshold Register	0x00022414	Table 423, p. 757
CPU3 Idle Low Threshold Register	0x00022418	Table 424, p. 757
CPU3 Idle Result Register	0x0002241C	Table 425, p. 757
CPU3 PM Event Status and Mask Register	0x00022420	Table 426, p. 757

Table 154: Summary Map Table for the CPU Sub-System Registers (Continued)

Register Name	Offset	Table and Page
CPU3 Boot Address Redirect Register	0x00022424	Table 427, p. 758
CPU3 Resume Control Register	0x00022428	Table 428, p. 758
Fabric PM Service		
L2C NFabric PM Control and Configuration Register	0x00022004	Table 429, p. 759
L2C NFabric PM Status and Mask Register	0x0002200C	Table 430, p. 759
L2C NFabric PM Event Status and Mask Register	0x00022020	Table 431, p. 760
Memory Controller PM Service		
MC Statistic Counter Global Control Register	0x00022E00	Table 432, p. 761
MC Idle Timer Value Register	0x00022E10	Table 433, p. 761
MC Idle High Threshold Register	0x00022E14	Table 434, p. 761
MC Idle Low Threshold Register	0x00022E18	Table 435, p. 762
MC Idle Result Register	0x00022E1C	Table 436, p. 762
MC PM Event Status and Mask Register	0x00022E20	Table 437, p. 762

A.2.1 Coherent Fabric Control and Status

Table 155: Coherency Fabric Control Register
Offset: 0x00020200

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27	SnoopCPU3 En	RW 0x0	Snoop CPU Enable Defines if CPU is enabled in the Coherency Fabric When CPU is disabled, no coherency request is sent to the processor by the fabric CPU may be dynamically disabled as part from the Power-Down process, once disabled, the fabric completes all pending snoop requests through the CPU, following coherent request will not be delivered to the associated Core. It is the software's responsibility to flush the Layer-1 Data Cache if coherency is required. 0 = False: Disable Snoop 1 = True: Enable Snoop

Table 155: Coherency Fabric Control Register (Continued)

Offset: 0x00020200

Bit	Field	Type/InitVal	Description
26	SnoopCPU2 En	RW 0x0	Snoop CPU Enable Defines if CPU is enabled in the Coherency Fabric When CPU is disabled, no coherency request is sent to the processor by the fabric CPU may be dynamically disabled as part from the Power-Down process, once disabled, the fabric completes all pending snoop requests through the CPU, following coherent request will not be delivered to the associated Core. It is the software’s responsibility to flush the Layer-1 Data Cache if coherency is required. 0 = False: Disable Snoop 1 = True: Enable Snoop
25	SnoopCPU1 En	RW 0x0	Snoop CPU Enable Defines if CPU is enabled in the Coherency Fabric When CPU is disabled, no coherency request is sent to the processor by the fabric CPU may be dynamically disabled as part from the Power-Down process, once disabled, the fabric completes all pending snoop requests through the CPU, following coherent request will not be delivered to the associated Core. It is the software’s responsibility to flush the Layer-1 Data Cache if coherency is required. 0 = False: Disable Snoop 1 = True: Enable Snoop

Table 155: Coherency Fabric Control Register (Continued)

Offset: 0x00020200

Bit	Field	Type/InitVal	Description
24	SnoopCPU0 En	RW 0x0	Snoop CPU Enable Defines if CPU is enabled in the Coherency Fabric When CPU is disabled, no coherency request is sent to the processor by the fabric CPU may be dynamically disabled as part from the Power-Down process, once disabled, the fabric completes all pending snoop requests through the CPU, following coherent request will not be delivered to the associated Core. It is the software's responsibility to flush the Layer-1 Data Cache if coherency is required. 0 = False: Disable Snoop 1 = True: Enable Snoop
23:9	Reserved	RSVD 0x0	Reserved
8	MBUS Err Prop Enable	RW 0x1	Enable MBUS Error Propagation 0 = False: MBUS Error is not propagated to the CPU on read 1 = True: MBUS Error is propagated to the CPU on read
7:0	Reserved	RSVD 0x0	Reserved

Table 156: Coherency Fabric Configuration Register

Offset: 0x00020204

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27:24	SMP Group0	RW 0x0	SMP Group 0 Defines which CPUs are in SMP group 0. In SMP mode, HW maintains cache coherency between CPUs that exist in same group. When bit #i of this field is set - CPU is present in SMP Group 0
23:20	Reserved	RSVD 0x0	Reserved

Table 156: Coherency Fabric Configuration Register (Continued)

Offset: 0x00020204

Bit	Field	Type/InitVal	Description
19:16	SMP Group1	RW 0x0	SMP Group 1 Defines which CPUs are in SMP group 1. In SMP mode, HW maintains cache coherency between CPUs that exist in same group. When bit #i of this field is set - CPU is present in SMP Group 1
15	Force Data Coherency En	RW 0x0	Force Data Coherent enable. When Enabled, data coherency between all CPUs will be maintained regardless of the SMP_Groups configuration. 0 = False: Shared memory requests will snoop the Data caches of the CPUs sharing the same SMP_Group as the initiator 1 = True: All shared memory requests will snoop the Data cache of all processors
14:2	Reserved	RSVD 0x0	Reserved
1:0	CPU Number	RO 0x3	Number of CPUs present in the Coherency Fabric. 0 = 1 CPU: CPU0 1 = 2 CPUs: CPU0-CPU1 2 = 3 CPUs: CPU0-CPU2 3 = 4 CPUs: CPU0-CPU3

Table 157: Events Affinity Register

Offset: 0x00020208

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:12	CPU3 Event Source	RW 0xF	Defines which of the CPUs can trigger the event_in input of CPU3. An active high value at bit i indicates that when CPUi issues a SEV command CPU3 will sense an event in.
11:8	CPU2 Event Source	RW 0xF	Defines which of the CPUs can trigger the event_in input of CPU2. An active high value at bit i indicates that when CPUi issues a SEV command CPU2 will sense an event in.
7:4	CPU1 Event Source	RW 0xF	Defines which of the CPUs can trigger the event_in input of CPU1. An active high value at bit i indicates that when CPUi issues a SEV command CPU1 will sense an event in.

Table 157: Events Affinity Register (Continued)

Offset: 0x00020208

Bit	Field	Type/InitVal	Description
3:0	CPU0 Event Source	RW 0xF	Defines which of the CPUs can trigger the event_in input of CPU0. An active high value at bit i indicates that when CPUi issues a SEV command CPU0 will sense an event in.

Table 158: IO Snoop Affinity Register

Offset: 0x0002020C

Bit	Field	Type/InitVal	Description
The IO Snoop Affinity Register defines the CPUs groups it serves for IO snoop in the IO Snoop affinity mechanism When CPU is defined as IO Coherent CPU, the CFU snoops this CPU for each IO coherent request. CPU can be snoopable by IO while is AMP mode or belong to any SMP group. The CFU will not snoop a CPU on IO request that is not snoopable by IO			
31:16	Reserved	RSVD 0x0	Reserved
15:12	CPU3_IO_Affinity	RW 0xF	CPU3 IO Affinity Defines the groups for the inbound IO snoop request that CPU3 is coherent with. 1 Current IO group is coherent with CPU 0 Current IO group is not coherent with CPU Bit 0 is for group0 Bit 1 is for group1 Bit 2 is for group2 Bit 3 is for group3
11:8	CPU2_IO_Affinity	RW 0xF	CPU2 IO Affinity Defines the groups for the inbound IO snoop request that CPU2 is coherent with. 1 Current IO group is coherent with CPU 0 Current IO group is not coherent with CPU Bit 0 is for group0 Bit 1 is for group1 Bit 2 is for group2 Bit 3 is for group3

Copyright © 2014 Marvell

Table 158: IO Snoop Affinity Register (Continued)

Offset: 0x0002020C

Bit	Field	Type/InitVal	Description
7:4	CPU1_IO_Affinity	RW 0xF	CPU1 IO Affinity Defines the groups for the inbound IO snoop request that CPU1 is coherent with. 1 Current IO group is coherent with CPU 0 Current IO group is not coherent with CPU Bit 0 is for group0 Bit 1 is for group1 Bit 2 is for group2 Bit 3 is for group3
3:0	CPU0_IO_Affinity	RW 0xF	CPU0 IO Affinity Defines the groups for the inbound IO snoop request that CPU0 is coherent with. 1 Current IO group is coherent with CPU 0 Current IO group is not coherent with CPU Bit 0 is for group0 Bit 1 is for group1 Bit 2 is for group2 Bit 3 is for group3

Table 159: CDI Configuration Register

Offset: 0x00020220

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x3	Reserved
0	DramControllerBus Width	RW 0x0	Coherency Fabric DRAM Controller interface data bus width. Note: The value in this field must correspond to the value in reg 0x14A8[0] 0 = 128bit 1 = 256bit

Table 160: CFU Configuration Register

Offset: 0x00020228

Bit	Field	Type/InitVal	Description
31:19	Reserved	RSVD 0x0	Reserved

Table 160: CFU Configuration Register (Continued)
Offset: 0x00020228

Bit	Field	Type/InitVal	Description
18	PoUtoSL2	RW 0x1	L2 Point of Unification 0 = Disable: L2 is the point of unification. Cache maintenance to point of unification command will not effects the L2. 1 = Enable: DDR is the point of unification. Cache maintenance to point of unification command will effects the L2.
17	PoCtoSL2	RW 0x1	L2 Point of Coherent 0 = Disable: - When enabled: L2 is the point of coherent. Cache maintenance to point of coherent command will not effects the L2. 1 = Enable: DDRis the point of coherent. Cache maintenance to point of coherent command will effects the L2.
16:13	Reserved	RSVD 0x0	Reserved
12	DMB Arb Point	RW 0x0	Control the point were DMB read can progress. 0 = Outstanding: DMB will not progress till all outstanding reads completed 1 = Arbitration: DMB will not progress till all pending reads are forward to the arbitration point.
11	DMB Wr Empty Sync	RW 0x0	Control the option that DMB will have immediate response in case of empty status. 0 = Enable 1 = Disable
10	DMB Sync L2	RW 0x0	Control the option that DMB will sync to the L2 cache. 0 = Enable 1 = Disable
9	DMB Snoop	RW 0x0	DMB snoops other CPUs. 0 = Enable 1 = Disable
8:6	Reserved	RSVD 0x0	Reserved
5	L2 Speculative Acc En	RW 0x0	Defines if to enable the option in which coherent read request will be access L2 speculatively 0 = Enable: L2 will be access befor the corresponding snoop is resolved 1 = Disable: L2 will be accessed after the corresponding snoop is resolved
4:0	Reserved	RSVD 0x0	Reserved

Table 161: SRAM Window n Control Register (n=0–3)

Offset: SRAM Window ID0: 0x00020240, SRAM Window ID1: 0x00020244, SRAM Window ID2: 0x00020248, SRAM Window ID3: 0x0002024C

Offset Formula: 0x00020240+4 * n: where n (0-3) represents SRAM Window ID

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Corresponds to transaction address[31:16]
15:12	BaseExtension	RW 0x0	Base Address Extension Corresponds to transaction address[35:32]
11	Reserved	RSVD 0x0	Reserved
10:8	Size	RW 0x1	Window Size 0 = 64KB 1 = 128KB 3 = 256KB 7 = 512KB
7:1	Reserved	RSVD 0x0	Reserved
0	WinEn	RW 0x0	Window Enable 0 = False: Window is disabled 1 = True: Window is enabled.

Table 162: Coherency Fabric Error Status Register

Offset: 0x00020258

Bit	Field	Type/InitVal	Description
31	CPU3 LSL1 STATE Parity Error	RW0C 0x0	CPU LSL1 State Parity Error Set when parity error is detected on the LSL1 State memories of CPU3
30	CPU2 LSL1 STATE Parity Error	RW0C 0x0	CPU LSL1 State Parity Error Set when parity error is detected on the LSL1 State memories of CPU2
29	CPU1 LSL1 STATE Parity Error	RW0C 0x0	CPU LSL1 State Parity Error Set when parity error is detected on the LSL1 State memories of CPU1
28	CPU0 LSL1 STATE Parity Error	RW0C 0x0	CPU LSL1 State Parity Error Set when parity error is detected on the LSL1 State memories of CPU0

Copyright © 2014 Marvell

Table 162: Coherency Fabric Error Status Register (Continued)
Offset: 0x00020258

Bit	Field	Type/InitVal	Description
27	CPU3 LSL1 ATTR Parity Error	RW0C 0x0	CPU LSL1 Attr Parity Error Set when parity error is detected on the LSL1 Attr memories of CPU3
26	CPU2 LSL1 ATTR Parity Error	RW0C 0x0	CPU LSL1 Attr Parity Error Set when parity error is detected on the LSL1 Attr memories of CPU2
25	CPU1 LSL1 ATTR Parity Error	RW0C 0x0	CPU LSL1 Attr Parity Error Set when parity error is detected on the LSL1 Attr memories of CPU1
24	CPU0 LSL1 ATTR Parity Error	RW0C 0x0	CPU LSL1 Attr Parity Error Set when parity error is detected on the LSL1 Attr memories of CPU0
23	CPU3 MMU Parity Error	RW0C 0x0	CPU MMU Parity Error Set when parity error is detected on the TLB memories of CPU3
22	CPU2 MMU Parity Error	RW0C 0x0	CPU MMU Parity Error Set when parity error is detected on the TLB memories of CPU2
21	CPU1 MMU Parity Error	RW0C 0x0	CPU MMU Parity Error Set when parity error is detected on the TLB memories of CPU1
20	CPU0 MMU Parity Error	RW0C 0x0	CPU MMU Parity Error Set when parity error is detected on the TLB memories of CPU0
19	CPU3 IC Parity Error	RW0C 0x0	CPU’s Instruction Cache Parity Error Set when parity error is detected on the Instruction Cache memories of CPU3
18	CPU2 IC Parity Error	RW0C 0x0	CPU’s Instruction Cache Parity Error Set when parity error is detected on the Instruction Cache memories of CPU2
17	CPU1 IC Parity Error	RW0C 0x0	CPU’s Instruction Cache Parity Error Set when parity error is detected on the Instruction Cache memories of CPU1
16	CPU0 IC Parity Error	RW0C 0x0	CPU’s Instruction Cache Parity Error Set when parity error is detected on the Instruction Cache memories of CPU0
15	CPU3 DC Parity Error	RW0C 0x0	CPU’s Data Cache Parity Error Set when parity error is detected on the Data Cache memories of CPU3

Table 162: Coherency Fabric Error Status Register (Continued)

Offset: 0x00020258

Bit	Field	Type/InitVal	Description
14	CPU2 DC Parity Error	RW0C 0x0	CPU’s Data Cache Parity Error Set when parity error is detected on the Data Cache memories of CPU2
13	CPU1 DC Parity Error	RW0C 0x0	CPU’s Data Cache Parity Error Set when parity error is detected on the Data Cache memories of CPU1
12	CPU0 DC Parity Error	RW0C 0x0	CPU’s Data Cache Parity Error Set when parity error is detected on the Data Cache memories of CPU0
11	ETB3 Parity Error	RW0C 0x0	ETB Array Parity Error Set when a parity error has occurred on ETB (Embedded Trace Buffer) memory of CPU3
10	ETB2 Parity Error	RW0C 0x0	ETB Array Parity Error Set when a parity error has occurred on ETB (Embedded Trace Buffer) memory of CPU2
9	ETB1 Parity Error	RW0C 0x0	ETB Array Parity Error Set when a parity error has occurred on ETB (Embedded Trace Buffer) memory of CPU1
8	ETB0 Parity Error	RW0C 0x0	ETB Array Parity Error Set when a parity error has occurred on ETB (Embedded Trace Buffer) memory of CPU0
7	CPU3 Snoop Parity Error	RW0C 0x0	CPU Snoop Parity Error Interrupt Set when a parity error has occurred on snoop to erroneous Data Cache line in CPU3
6	CPU2 Snoop Parity Error	RW0C 0x0	CPU Snoop Parity Error Interrupt Set when a parity error has occurred on snoop to erroneous Data Cache line in CPU2
5	CPU1 Snoop Parity Error	RW0C 0x0	CPU Snoop Parity Error Interrupt Set when a parity error has occurred on snoop to erroneous Data Cache line in CPU1

Table 162: Coherency Fabric Error Status Register (Continued)

Offset: 0x00020258

Bit	Field	Type/InitVal	Description
4	CPU0 Snoop Parity Error	RW0C 0x0	CPU Snoop Parity Error Interrupt Set when a parity error has occurred on snoop to erroneous Data Cache line in CPU0
3	CIB AccessErr	RW0C 0x0	CIB Access Error Set in the case of a coherent request that is issue by an IO engine and does not hit an SRAM or a DRAM window.
2	CIB Parity Error	RW0C 0x0	Coherent IO Bridge Parity Error Set when Coherent IO Bridge memories detected a soft error. Details of the parity error are captured at <CIB Parity Error Status>
1	AccessErr	RW0C 0x0	Access Error. Set in the following error cases: * CPU burst access to internal register * CPU access to an unmapped address (no match in any of the address decoding windows).
0	Bit64Err	RW0C 0x0	MBUS Bridge Read Error Set when an erroneous read data indication from the Mbus unit is recognized

Table 163: Coherency Fabric Error Mask Register

Offset: 0x0002025C

Bit	Field	Type/InitVal	Description
31:0	FabricErrorMask	RW 0x0	Fabric Error Mask Mask bit per error event in the Coherency Fabric Error Status Register. When error event occurs in the Coherency Fabric, the CoherencyFabricErrSum status bit in the Coherency Fabric Local Cause Register is set if the associated error mask bit is set

Table 164: Coherency Fabric Local Cause Register

Offset: 0x00020260

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved

Table 164: Coherency Fabric Local Cause Register (Continued)
Offset: 0x00020260

Bit	Field	Type/InitVal	Description
27	CoherencyFabricErrorSum	RO 0x0	Coherency Fabric Error Sum Interrupt
26	Global WD Timer	RO 0x0	Global WD Timer Interrupt
25:22	Reserved	RSVD 0x0	Reserved
21	PMU CPU3 Interrupt	RO 0x0	Power Management Unit Interrupt
20	PMU CPU2 Interrupt	RO 0x0	Power Management Unit Interrupt
19	PMU CPU1 Interrupt	RO 0x0	Power Management Unit Interrupt
18	PMU CPU0 Interrupt	RO 0x0	Power Management Unit Interrupt
17	PMU MC Interrupt	RO 0x0	Power Management Unit Memory Controller Interrupt
16	L2 Cache Interrupt	RO 0x0	L2 Cache Events Summary Interrupt
15:12	Reserved	RSVD 0x0	Reserved
11	CPU3 nctiirq	RO 0x0	CPU3 Debug Cross Trigger Interupt
10	CPU2 nctiirq	RO 0x0	CPU2 Debug Cross Trigger Interupt
9	CPU1 nctiirq	RO 0x0	CPU1 Debug Cross Trigger Interupt
8	CPU0 nctiirq	RO 0x0	CPU0 Debug Cross Trigger Interupt
7:4	Reserved	RSVD 0x0	Reserved
3	CPU3 Perf Counter Overflow	RO 0x0	CPU3 Performance Counter Overflow Interrupt
2	CPU2 Perf Counter Overflow	RO 0x0	CPU2 Performance Counter Overflow Interrupt
1	CPU1 Perf Counter Overflow	RO 0x0	CPU1 Performance Counter Overflow Interrupt

Table 164: Coherency Fabric Local Cause Register (Continued)

Offset: 0x00020260

Bit	Field	Type/InitVal	Description
0	CPU0 Perf Counter Overflow	RO 0x0	CPU0 Performance Counter Overflow Interrupt

A.2.2 Coherent IO Bridge Control and Status

Table 165: CIB Control and Configuration Register

Offset: 0x00020280

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x7FB	Reserved
20	NonSecureIOTransfer	RW 0x0	Secured/Non secured IO transfers towards Fabric. 0 = Secured Transfer 1 = Non Secured Transfer
19:14	Reserved	RSVD 0x1C	Reserved
13	CIB Empty Status	RO 0x0	Coherent IO Bridge Empty Specifies that CIB internal buffers are empty, that is: - All shared commands were sent to Fabric. - All shared writes were completed by Fabric domain. - All shared reads were sent back to the external hosts. 0 = Not Empty 1 = Empty
12	Reserved	RSVD 0x1	Reserved
11:10	IgnoreSharingEnable	RW 0x0	Coherent IO Bridge Ignore Sharing Enable When this bit is set, coherent IO requests are treated as non-shared, and therefore, ignored by the Coherent IO Bridge and routed directly to the DRAM Controller. This bit is used when the entire CPU Subsystem's caches are flushed to memory (for example, in power down state). 0 = Disable: Mbus external shared requests are routed to Coherent IO Bridge and external non-shared requests are routed to DDR. 2 = Enable: Mbus external shared and external non-shared requests are routed to DDR.

Table 165: CIB Control and Configuration Register (Continued)

Offset: 0x00020280

Bit	Field	Type/InitVal	Description
9	StopCibAckEnable	RW 0x0	Stop Coherent IO Bridge Acknowledge Enable This bit controls the prevention of CIB acknowledgment to pending Mbus shared requests. 0 = False: CIB acknowledges shared read/write pending requests. 1 = True: CIB does not acknowledge shared read/write pending requests, only used as part of power down operation.
8:5	Reserved	RSVD 0x8	Reserved
4:3	Reserved	RW 0x2	Reserved Must be 0x2.
2:1	Reserved	RW 0x3	Reserved Must be 0x3.
0	CIB Write Back Enable	RW 0x0	Coherent IO Bridge Write Back Mode Enable When this field is set, IO write requests that are written to the L2 (due to a hit or due to a CL allocation) will not be written to DDR. When this field is cleared, IO write requests will always be written to the DDR. 0 = Disable: CIB in Write Through Mode 1 = Enable: CIB in Write Back Mode

Table 166: CIB Non L1 Snoopable Window0 Control Register

Offset: 0x00020284

Bit	Field	Type/InitVal	Description
31:16	Window Size	RW 0x0	Non L1 Snoopable Window Size, Each bit represents 64KB size granularity.
15:1	Reserved	RSVD 0x0	Reserved
0	WinEn	RW 0x0	Non L1 Snoopable Window Enable Enable checking IO requests if residing within non L1 snoopable window0 region. If it matches window address region no L1 snoop will be applied. 0 = False: CIB non L1 Snoop Window disabled 1 = True: CIB non L1 Snoop Window enabled

Table 167: CIB Non L1 Snoopable Window0 Base Address Register

Offset: 0x00020288

Bit	Field	Type/InitVal	Description
31:16	WindowBase	RW 0x0	Non L1 Snoopable Window Base Address. Corresponds to transaction address bits [31:16]
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Non L1 Snoopable Window Base Address Extension. Corresponds to transaction address bits [35:32]

Table 168: CIB Non L1 Snoopable Window1 Control Register

Offset: 0x0002028C

Bit	Field	Type/InitVal	Description
31:16	Window Size	RW 0x0	Non L1 Snoopable Window Size, Each bit represents 64KB size granularity.
15:1	Reserved	RSVD 0x0	Reserved
0	WinEn	RW 0x0	Non L1 Snoopable Window Enable Enable checking IO requests if residing within non L1 snoopable window0 region. If it matches window address region no L1 snoop will be applied. 0 = False: CIB non L1 Snoop Window disabled 1 = True: CIB non L1 Snoop Window enabled

Table 169: CIB Non L1 Snoopable Window1 Base Address Register

Offset: 0x00020290

Bit	Field	Type/InitVal	Description
31:16	WindowBase	RW 0x0	Non L1 Snoopable Window Base Address. Corresponds to transaction address bits [31:16]]
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Non L1 Snoopable Window Base Address Extension. Corresponds to transaction address bits [35:32]

Table 170: CIB Read Buffer Select Register

Offset: 0x00020294

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	RdBuff	RW 0x0	Read buffer assignment per unit. When a bit is set to 0, the corresponding unit is assigned to receive read data from Read buffer 0. When set to 1, the corresponding unit is assigned to receive read data from Read buffer 1. Bit[0] corresponds to unit ID 0x0, bit[1] corresponds to unit ID 0x1, etc. For example, setting to 0x80 stands for unit ID 0x7 receives read data from read buffer 1, while the rest of the units receive data from read buffer 0.

Table 171: CIB Parity Error Status Register

Offset: 0x00020298

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved
3	Port1 read internal RAM error	RO 0x0	Internal Error over Port1 read RAM due to parity error.
2	Port0 read internal RAM error	RO 0x0	Internal Error over Port0 read RAM due to parity error.
1	Port1 write internal RAM error	RO 0x0	Internal Error over Port1 write RAM due to parity error.
0	Port0 write internal RAM error	RO 0x0	Internal Error over Port0 write RAM due to parity error.

A.2.3 SDRAM Address Decoding

Table 172: Win 0 Base Address Register

Offset: 0x00020180

Bit	Field	Type/InitVal	Description
31:24	Base	RW 0x0	DRAM Window0 base address. Corresponds to address bits [31:24].

Table 172: Win 0 Base Address Register (Continued)

Offset: 0x00020180

Bit	Field	Type/InitVal	Description
23:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	DRAM Window0 base address extension Corresponds to address bits [35:32]

Table 173: Win 0 Control Register

Offset: 0x00020184

Bit	Field	Type/InitVal	Description
31:24	Size	RW 0x0f	DRAM Window0 Size Corresponds to Base Address bits [31:24]. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. Changing the value of this field must be done when the window is disabled (<En>=0)
23:5	Reserved	RO 0x7ffff	Reserved
4:2	Win_CS	RW 0x0	DRAM Window0 CS Defines the DRAM CS which will be accessed upon a hit in DRAM Window 0. Changing the value of this field must be done when the window is disabled (<En>=0) 0 = CS[0]: For DRAM Controller 1 = CS[1]: For DRAM Controller 2 = CS[2]: For DRAM Controller 3 = CS[3]: For DRAM Controller
1	Reserved	RSVD 0x0	Reserved
0	En	RW 0x0	DRAM Window0 Enable 0 = False 1 = True

Table 174: Win 1 Base Address Register

Offset: 0x00020188

Bit	Field	Type/InitVal	Description
31:24	Base	RW 0x10	DRAM Window1 base address. Corresponds to address bits [31:24].

Table 174: Win 1 Base Address Register (Continued)

Offset: 0x00020188

Bit	Field	Type/InitVal	Description
23:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	DRAM Window1 base address extension Corresponds to address bits [35:32]

Table 175: Win 1 Control Register

Offset: 0x0002018C

Bit	Field	Type/InitVal	Description
31:24	Size	RW 0x0f	DRAM Window1 Bank Size Corresponds to Base Address bits [31:24]. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s.
23:5	Reserved	RO 0x7ffff	Reserved
4:2	Win_CS	RW 0x1	DRAM Window1 CS Defines the DRAM CS which will be accessed upon a hit in DRAM Window 1. Changing the value of this field must be done when the window is disabled (<En>=0) 0 = CS[0]: For DRAM Controller 1 = CS[1]: For DRAM Controller 2 = CS[2]: For DRAM Controller 3 = CS[3]: For DRAM Controller
1	Reserved	RSVD 0x0	Reserved
0	En	RW 0x0	DRAM Window1 Enable 0 = False 1 = True

Table 176: Win 2 Base Address Register

Offset: 0x00020190

Bit	Field	Type/InitVal	Description
31:24	Base	RW 0x20	DRAM Window2 base address. Corresponds to address bits [31:24].

Table 176: Win 2 Base Address Register (Continued)

Offset: 0x00020190

Bit	Field	Type/InitVal	Description
23:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	DRAM Window2 base address extension Corresponds to address bits [35:32]

Table 177: Win 2 Control Register

Offset: 0x00020194

Bit	Field	Type/InitVal	Description
31:24	Size	RW 0x0f	DRAM Window2 Bank Size Corresponds to Base Address bits [31:24]. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s.
23:5	Reserved	RO 0x7ffff	Reserved
4:2	Win_CS	RW 0x2	DRAM Window2 CS Defines the DRAM CS which will be accessed upon a hit in DRAM Window 2. Changing the value of this field must be done when the window is disabled (<En>=0) 0 = CS[0]: For DRAM Controller 1 = CS[1]: For DRAM Controller 2 = CS[2]: For DRAM Controller 3 = CS[3]: For DRAM Controller
1	Reserved	RSVD 0x0	Reserved
0	En	RW 0x0	DRAM Window2 Enable 0 = False 1 = True

Table 178: Win3 Base Address Register

Offset: 0x00020198

Bit	Field	Type/InitVal	Description
31:24	Base	RW 0x30	DRAM Window3 base address. Corresponds to address bits [31:24].

Document Classification: Proprietary Information

Copyright © 2014 Marvell

Table 178: Win3 Base Address Register (Continued)

Offset: 0x00020198

Bit	Field	Type/InitVal	Description
23:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	DRAM Window3 base address extension Corresponds to address bits [35:32]

Table 179: Win 3 Control Register

Offset: 0x0002019C

Bit	Field	Type/InitVal	Description
31:24	Size	RW 0x0f	DRAM Window3 Size Corresponds to Base Address bits [31:24]. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s.
23:5	Reserved	RO 0x7ffff	Reserved
4:2	Win_CS	RW 0x3	DRAM Window3 CS Defines the DRAM CS which will be accessed upon a hit in DRAM Window 3. Changing the value of this field must be done when the window is disabled (<En>=0) 0 = CS[0]: For DRAM Controller 1 = CS[1]: For DRAM Controller 2 = CS[2]: For DRAM Controller 3 = CS[3]: For DRAM Controller
1	Reserved	RSVD 0x0	Reserved
0	En	RW 0x0	DRAM Window3 Enable 0 = False 1 = True

A.2.4 Mbus Unit Address Decoding

Table 180: Window0 Control Register
Offset: 0x00020000

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window in 64 KB granularity (e.g., a value of 0x00FF specifies 256*64KB = 16 MB). NOTES: A value of 0x0 specifies 64-KB size. Changing the value of this field must be done when <win_en>=0
15:8	Attr	RW 0xE8	Specifies the target interface attributes associated with this window. See the "Target Interface Configurations" section. Changing the value of this field must be done when <win_en>=0
7:4	Target	RW 0x4	Specifies the target interface associated with this window. See the "Target Interface Configurations" section. Changing the value of this field must be done when <win_en>=0
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window0 Enable 0 = Disable: Window is disabled. 1 = Enable: Window is enabled.

Table 181: Window0 Base Register
Offset: 0x00020004

Bit	Field	Type/InitVal	Description
NOTE: If the remap function for this register is not used, the <Remap> field in the Window0 Remap Low Register must be set to same value as the <Base> field in this register!			
31:16	Base	RW 0x8000	Base Address Used with the <Size> field in the Window0 Control Register to set the address window size and location. Corresponds to transaction address[31:16]. Changing the value of this field must be done when <win0_en>=0
15:4	Reserved	RO 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension This field holds base address[35:32] of Window 0.

Table 182: Window0 Remap Low Register
Offset: 0x00020008

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x8000	Remap Address Used with the <Size> field in the Window0 Control Register to specifies address bits[31:0] to be driven to the target interface. Changing the value of this field must be done when <win0_en>=0
15:0	Reserved	RO 0x0	Reserved

Table 183: Window0 Remap High Register
Offset: 0x0002000C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Specifies address bits[63:32] to be driven to the target interface.

Copyright © 2014 Marvell

Table 184: Window1 Control Register
Offset: 0x00020010

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register
15:8	Attr	RW 0xE8	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x4	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window1 Enable See the Window0 Control Register 0 = Disable 1 = Enable

Table 185: Window1 Base Register
Offset: 0x00020014

Bit	Field	Type/InitVal	Description
NOTE: If the remap function for this register is not used, the <Remap> field in the Window0 Remap Low Register must be set to same value as the <Base> field in this register.			
31:16	Base	RW 0x8800	Base Address See description at Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Register

Copyright © 2014 Marvell

Table 186: Window1 Remap Low Register
Offset: 0x00020018

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x8800	Remap Address See the Window0 Remap Low Register
15:0	Reserved	RO 0x0	Reserved

Table 187: Window1 Remap High Register
Offset: 0x0002001C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 188: Window2 Control Register
Offset: 0x00020020

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register
15:8	Attr	RW 0xE8	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x4	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window2 Enable See the Window0 Control Register 0 = Disable 1 = Enable

Table 189: Window2 Base Register

Offset: 0x00020024

Bit	Field	Type/InitVal	Description
NOTE: If the remap function for this register is not used, the <Remap> field in the Window2 Remap Low Register must be set to same value as the <Base> field in this register.			
31:16	Base	RW 0x9000	Base Address See description at Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Register

Table 190: Window2 Remap Low Register

Offset: 0x00020028

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x9000	Remap Address See the Window0 Remap Low Register.
15:0	Reserved	RO 0x0	Reserved

Table 191: Window2 Remap High Register

Offset: 0x0002002C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 192: Window3 Control Register

Offset: 0x00020030

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register
15:8	Attr	RW 0xE8	Target specific attributes depending on the target interface. See the Window0 Control Register

Copyright © 2014 Marvell

Table 192: Window3 Control Register (Continued)

Offset: 0x00020030

Bit	Field	Type/InitVal	Description
7:4	Target	RW 0x4	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window3 Enable See the Window0 Control Register 0 = Disable 1 = Enable

Table 193: Window3 Base Register

Offset: 0x00020034

Bit	Field	Type/InitVal	Description
NOTE: If the remap function for this register is not used, the <Remap> field in the Window3 Remap Low Register must be set to same value as the <Base> field in this register.			
31:16	Base	RW 0x9800	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Register

Table 194: Window3 Remap Low Register

Offset: 0x00020038

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x9800	Remap Address See the Window0 Remap Low Register
15:0	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 195: Window3 Remap High Register
Offset: 0x0002003C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 196: Window4 Control Register
Offset: 0x00020040

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register
15:8	Attr	RW 0xE8	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x8	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window4 Enable See the Window0 Control Register 0 = Disable 1 = Enable

Table 197: Window4 Base Register
Offset: 0x00020044

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xA000	Base Address See the Window0 Base Register.
15:4	Reserved	RSVD 0x0	Reserved

Table 197: Window4 Base Register (Continued)
Offset: 0x00020044

Bit	Field	Type/InitVal	Description
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Register

Table 198: Window4 Remap Low Register
Offset: 0x00020048

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0xA000	Remap Address See the Window0 Remap Low Register
15:0	Reserved	RO 0x0	Reserved

Table 199: Window4 Remap High Register
Offset: 0x0002004C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 200: Window5 Control Register
Offset: 0x00020050

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register
15:8	Attr	RW 0xE8	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x8	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register
3:2	Reserved	RSVD 0x0	Reserved

Table 200: Window5 Control Register (Continued)
Offset: 0x00020050

Bit	Field	Type/InitVal	Description
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x0	Window5 Enable See the Window0 Control Register 0 = Disable 1 = Enable

Table 201: Window5 Base Register
Offset: 0x00020054

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xA800	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Register

Table 202: Window5 Remap Low Register
Offset: 0x00020058

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0xA800	Remap Address See the Window0 Remap Low Register
15:0	Reserved	RO 0x0	Reserved

Table 203: Window5 Remap High Register
Offset: 0x0002005C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 204: Window6 Control Register
Offset: 0x00020060

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register
15:8	Attr	RW 0xE8	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x8	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register
3:2	Reserved	RO 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x0	Window6 Enable See the Window0 Control Register 0 = Disable 1 = Enable

Table 205: Window6 Base Register
Offset: 0x00020064

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xB000	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved

Table 205: Window6 Base Register (Continued)

Offset: 0x00020064

Bit	Field	Type/InitVal	Description
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 206: Window6 Remap Low Register

Offset: 0x00020068

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0xB000	Remap Address See the Window0 Remap Low Register
15:0	Reserved	RO 0x0	Reserved

Table 207: Window6 Remap High Register

Offset: 0x0002006C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 208: Window7 Control Register

Offset: 0x00020070

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register.
15:8	Attr	RW 0xE8	Target specific attributes depending on the target interface. See the Window0 Control Register.
7:4	Target	RW 0x8	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register
3:2	Reserved	RO 0x0	Reserved

Table 208: Window7 Control Register (Continued)

Offset: 0x00020070

Bit	Field	Type/InitVal	Description
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x0	Window7 Enable See the Window0 Control Register 0 = Disable 1 = Enable

Table 209: Window7 Base Register

Offset: 0x00020074

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xB800	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 210: Window7 Remap Low Register

Offset: 0x00020078

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0xB800	Remap Address See the Window0 Remap Low Register
15:0	Reserved	RO 0x0	Reserved

Table 211: Window7 Remap High Register

Offset: 0x0002007C

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 212: Internal Registers Base Address

Offset: 0x00020080

Bit	Field	Type/InitVal	Description
31:20	Base	RW 0xD00	Internal registers Base Address
19:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Internal Space base address extension Corresponds to address bits [35:32]

Table 213: Internal Units Sync Barrier Control Register

Offset: 0x00020084

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	InternalUnitSyncBarrierEnable	RW 0x0	Internal Unit Sync Barrier Enable Per each MBUS Unit, this field contain a bit. When bit#i in this field is set, the data of a read request from an internal address with (addr[19:16] == i) will be returned to the CPU after all previous write requests from the unit with ID=i have been written to the memory.

Table 214: Window8 Control Register

Offset: 0x00020090

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size See the Window0 Control Register.

Table 214: Window8 Control Register (Continued)

Offset: 0x00020090

Bit	Field	Type/InitVal	Description
15:8	Attr	RW 0x9	Target specific attributes depending on the target interface. See the Window0 Control Register.
7:4	Target	RW 0x9	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window8 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 215: Window8 Base Register

Offset: 0x00020094

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xC801	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 216: Window9 Control Register

Offset: 0x00020098

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7FF	Window Size See the Window0 Control Register.
15:8	Attr	RW 0x2F	Target specific attributes depending on the target interface. See the Window0 Control Register.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window9 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 217: Window9 Base Register

Offset: 0x0002009C

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xD800	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 218: Window10 Control Register

Offset: 0x000200A0

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size See the Window0 Control Register.
15:8	Attr	RW 0x3E	Target specific attributes depending on the target interface. See the Window0 Control Register.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window10 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 219: Window10 Base Register

Offset: 0x000200A4

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xE000	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 220: Window11 Control Register
Offset: 0x000200A8

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size See the Window0 Control Register.
15:8	Attr	RW 0x3D	Target specific attributes depending on the target interface. See the Window0 Control Register.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window11 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 221: Window11 Base Register
Offset: 0x000200AC

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xE800	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 222: Window12 Control Register

Offset: 0x000200B0

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size See the Window0 Control Register.
15:8	Attr	RW 0x3B	Target specific attributes depending on the target interface. See the Window0 Control Register.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window12 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 223: Window12 Base Register

Offset: 0x000200B4

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xF000	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Copyright © 2014 Marvell

Table 224: Window13 Control Register

Offset: 0x000200B8

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size See the Window0 Control Register.
15:8	Attr	RW SAR	Target specific attributes depending on the target interface. See the Window0 Control Register. 0x1D Boot from BootRom 0x2F Boot From Device Bus 0x1E Boot From SPI.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window13 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 225: Window13 Base Register

Offset: 0x000200BC

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xF800	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 226: Window14 Control Register

Offset: 0x000200C0

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x03FF	Window Size See the Window0 Control Register.
15:8	Attr	RW 0x1E	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window14 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 227: Window14 Base Register

Offset: 0x000200C4

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xD400	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 228: Window15 Control Register
Offset: 0x000200C8

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size See the Window0 Control Register.
15:8	Attr	RW 0xe0	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x4	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x0	Window15 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 229: Window15 Base Register
Offset: 0x000200CC

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xD030	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 230: Window16 Control Register
Offset: 0x000200D0

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size See the Window0 Control Register.
15:8	Attr	RW 0xe0	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x4	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x0	Window16 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 231: Window16 Base Register
Offset: 0x000200D4

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xD031	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 232: Window17 Control Register

Offset: 0x000200D8

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size See the Window0 Control Register.
15:8	Attr	RW 0xe0	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x4	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x0	Window17 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 233: Window17 Base Register

Offset: 0x000200DC

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xD032	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 234: Window18 Control Register

Offset: 0x000200E0

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x7f	Window Size See the Window0 Control Register.
15:8	Attr	RW 0x0	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0xc	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window18 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 235: Window18 Base Register

Offset: 0x000200E4

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xD380	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Copyright © 2014 Marvell

Table 236: Window19 Control Register

Offset: 0x000200E8

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size See the Window0 Control Register.
15:8	Attr	RW 0x0e	Target specific attributes depending on the target interface. See the Window0 Control Register
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
3:2	Reserved	RSVD 0x0	Reserved
1	SyncBarrierEnable	RW 0x0	Window Sync Barrier Enable When enabled, data associated with read requests that hit the window will be returned to the requesting CPU after all writes issued by the target unit reach the memory. 0 = False 1 = True
0	win_en	RW 0x1	Window19 Enable See the Window0 Control Register. 0 = Disable 1 = Enable

Table 237: Window19 Base Register

Offset: 0x000200EC

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0000	Base Address See the Window0 Base Register
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension See description at Window0 Base Address Reg

Table 238: Window13 Remap Low Register

Offset: 0x000200F0

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0xF800	Remap Address Used with the <Size> field in the Window13 Control Register to specify address bits[31:0] to be driven to the target interface. Changing the value of this field must be done when <win13_en>=0
15:0	Reserved	RO 0x0	Reserved

Table 239: Window13 Remap High Register

Offset: 0x000200F4

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Specifies address bits[63:32] to be driven to the target interface.

Table 240: Mbus Bridge Window Control Register

Offset: 0x00020250

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0000	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window in 64 KB granularity (for example, a value of 0x00FF specifies 256*64KB = 16 MB). **NOTE:** A value of 0x0 specifies 64-KB size. **NOTE:**
15:4	Reserved	RSVD 0x0	Reserved
3:1	Reserved	RO 0x0	Reserved
0	WinEn	RW 0x0	Mbus Bridge Window Enable 0 = False: Window is disabled. 1 = True: Window is enabled.

Table 241: Mbus Bridge Window Base Register

Offset: 0x00020254

Bit	Field	Type/InitVal	Description
NOTE: If the remap function for this register is not used, the <Remap> field in the Window0 Remap Low Register must be set to same value as the <Base> field in this register!			
31:16	Base	RW 0x8000	Base Address Used with the <Size> field in the Mbus Bridge Window Control Register to set the address window size and location. Corresponds to transaction address[31:16].
15:4	Reserved	RSVD 0x0	Reserved
3:0	BaseExtension	RW 0x0	Base Address Extension Corresponds to transaction address[31:16].

A.2.5 CPU

Table 242: CPU Configuration Register (N=0–3)

Offset: CPU ID0: 0x00021800, CPU ID1: 0x00021900, CPU ID2: 0x00021A00, CPU ID3: 0x00021B00

Offset Formula: 0x00021800+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x1	Reserved
27:24	Cluster ID	RW 0x0	CPU Cluster ID Value read in Cluster ID register filed, bits [11:8] of the MPIRD.
23:17	Reserved	RSVD 0x16	Reserved
16	Shared L2 Present	RW 0x1	Defines the corresponding CPU where a Shared L2 exists.
15	Pclk WFI En	RW SAR	Enable wakeup from WFI through debugger Other values are reserved.
14:11	Reserved	RSVD 0x0	Reserved
10:9	Core Mode	RW SAR	Specifies the targeted ARM core behavior 1 = Cortex-A8: (v7, uP) 3 = Cortex-A9: (v7, uP or MP)

Table 242: CPU Configuration Register (N=0–3) (Continued)

Offset: CPU ID0: 0x00021800, CPU ID1: 0x00021900, CPU ID2: 0x00021A00, CPU ID3: 0x00021B00

Offset Formula: 0x00021800+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
8:7	Reserved	RSVD 0x0	Reserved
6	NFMI En	RW SAR	Fast Interrupt Enable Configures fast interrupts to be nonmaskable. This pin is only sampled during reset of the processor. 0 = NMFI Disable: Clears the NMFI bit in the CP15 System Control Register 1 = NMFI Enable: Sets the NMFI bit in the CP15 System Control Register
5	TE Init	RW SAR	Thumb Exception Initialization Controls the state of the TE bit in the CP15 System Control Register. This pin is only sampled during reset of the processor. 0 = ARM Exception: Enable ARM exception generation. On exception entry, CPSR T bit is 0 and J bit is 0. 1 = Thumb Exception: Enable Thumb exception generation. On exception entry, CPSR T bit is 1 and J bit is 0.
4	Reserved	RW 0x1	Reserved Must be 0x1.
3	Endian Init	RW SAR	Endian Initialization Controls the state of the B bit in the CP15 c1 Control register. This pin is only sampled during reset of the processor. 0 = Little Endian: Clears the B bit in the CP15 c1 Control Register, indicating Little Endian memory system support. 1 = Big Endian: Sets the B bit in the CP15 c1 Control Register, indicating Big Endian word-invariant memory system support.
2	Reserved	RSVD 0x0	Reserved
1	Vec Init Loc	RW SAR	Determines the reset location of the boot starting address. 0 = MsbZeros: Boot starting address is 0x00000000 1 = MsbOnes: Boot starting address is 0xFFFF0000.

Table 242: CPU Configuration Register (N=0–3) (Continued)

Offset: CPU ID0: 0x00021800, CPU ID1: 0x00021900, CPU ID2: 0x00021A00, CPU ID3: 0x00021B00
Offset Formula: 0x00021800+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
0	Reserved	RSVD 0x0	Reserved

Table 243: CPU Control and Status Register (N=0–3)

Offset: CPU ID0: 0x00021808, CPU ID1: 0x00021908, CPU ID2: 0x00021A08, CPU ID3: 0x00021B08
Offset Formula: 0x00021808+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x8000	Reserved
1	CPUBigEndianStatus	RO 0x0	Big Endian Reflects the value of the Big Endian field of the CPU CP15 register. 0 = Little 1 = Big
0	Reserved	RSVD 0x0	Reserved

Table 244: CPU IO Sync Barrier Control Register (N=0–3)

Offset: CPU ID0: 0x00021810, CPU ID1: 0x00021910, CPU ID2: 0x00021A10, CPU ID3: 0x00021B10
Offset Formula: 0x00021810+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:1	Reserved	RO 0x0	
0	TriggerAndStatus	RW 0x0	Sync Barrier Trigger and Status Triggers sync barrier operation for the CPU Core. 0 = IDLE 1 = BUSY

Table 245: Snoop Filter Control Register (N=0–3)

Offset: CPU ID0: 0x00021820, CPU ID1: 0x00021920, CPU ID2: 0x00021A20, CPU ID3: 0x00021B20
Offset Formula: 0x00021820+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:9	Reserved	RSVD 0x0	Reserved
8	SF Invalidate	RW0C 0x1	Snoop Filer Invalidate When 0x0 is written , SF will clear all entries. HW sets this bit when invalidation is done .
7:2	Reserved	RSVD 0x0	Reserved
1	SF Self Bypass	RW0C 0x0	Snoop Filter Self Bypass This bit is set when SF identifies a parity error and is cleared by SW. When the bit is set, the Snoop requests will bypass the SF. In addition, SW recieves an interrupt that indicates a parity error. The SF parity interrupt is also an indication that this bit has been automatically set.
0	SF Enable	RW 0x0	Snoop Filter Enable When not active, SF will be bypassed and as such all coherent requests will be forwarded to the CPU. 0 = False: SF is bypassed 1 = True: SF will not forward coherent requests that are known to be non-allocated

Table 246: Snoop Filter Interrupt Cause Register (N=0–3)

Offset: CPU ID0: 0x00021828, CPU ID1: 0x00021928, CPU ID2: 0x00021A28, CPU ID3: 0x00021B28
Offset Formula: 0x00021828+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	SF parity error	RW1C 0x0	SF parity error interrupt

Table 247: Snoop Filter Interrupt Mask Register (N=0–3)

Offset: CPU ID0: 0x0002182C, CPU ID1: 0x0002192C, CPU ID2: 0x00021A2C, CPU ID3: 0x00021B2C

Offset Formula: 0x0002182C+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	SF Parity Mask	RW 0x0	SF parity error mask When set, SF parity error will not be propogated to FIQ/IRQ

Table 248: Timers Control Register (N=0–3)

Offset: CPU ID0: 0x00021840, CPU ID1: 0x00021940, CPU ID2: 0x00021A40, CPU ID3: 0x00021B40

Offset Formula: 0x00021840+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24:22	CPUTimer1 Ratio	RW 0x0	Global Timer 1 Ratio See definition and encoding at <CPUWDTimerRatio>
21:19	CPUTimer0 Ratio	RW 0x0	Global Timer 0 Ratio See definition and encoding at <CPUWDTimerRatio>
18:16	CPUWDTimerRatio	RW 0x0	Global Watchdog Timer Ratio. The field defines the CPU WD Timer interval. Notate the number in this field as R. When activated, the timer will be incremented upon every (2 ^ R) source clocks. When <CPUWDTimer25MhzEn> is set , this field is ignored. 0 = 1: Timer tic occurs every source clock 1 = 2: Timer tic occurs every 2 source clocks 2 = 4: Timer tic occurs every 4 source clocks 3 = 8: Timer tic occurs every 8 source clocks 4 = 16: Timer tic occurs every 16 source clocks 5 = 32: Timer tic occurs every 32 source clocks 6 = 64: Timer tic occurs every 64 source clocks 7 = 128: Timer tic occurs every 128 source clocks
15:13	Reserved	RSVD 0x0	Reserved
12	CPUTimer1 25MhzEn	RW 0x0	CPU Timer1 25Mhz frequency Enable See description at <CPUWDTimer25MhzEn> 0 = False 1 = True

Table 248: Timers Control Register (N=0–3) (Continued)

Offset: CPU ID0: 0x00021840, CPU ID1: 0x00021940, CPU ID2: 0x00021A40, CPU ID3: 0x00021B40

Offset Formula: 0x00021840+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
11	CPUTimer0 25MhzEn	RW 0x0	CPU Timer0 25Mhz frequency Enable See description at <CPUWDTimer25MhzEn> 0 = False 1 = True
10	CPUWDTimer25MhzEn	RW 0x0	CPU Watchdog Timer 25Mhz frequency Enable Defines the count frequency of the CPU watchdog timer. 0 = False: CPU WD timer count frequency is defined by the source clock according to <CPUWDTimerRatio> field 1 = True: CPU WD timer count frequency is 25Mhz
9:6	Reserved	RSVD 0x0	Reserved
5	CPUWDTimerAuto	RW 0x0	CPU Watchdog Timer Auto Mode When this bit is clear to 0 and <CPUWDTimer> in the CPU Watchdog Timer Register has reached zero, <CPUWDTimer> stops counting. When this bit is set to 1 and <CPUWDTimer> has reached zero, <CPUTimer0Rel> in the CPU Timer0 Reload Register is reloaded to <CPUWDTimer>, it then continues to count.
4	CPUWDTimerEn	RW 0x0	CPU Watchdog Timer Enable 0 = False 1 = True
3	CPUTimer1Auto	RW 0x0	CPU Timer 1 Auto Mode When this bit is clear to 0 and <CPUTimer1> in the CPU Timer1 Register has reached to zero, <CPUTimer1> stops counting. When this bit is set to 1 and <CPUTimer1> has reached zero, <CPUTimer1Rel> in the CPU Timer1 Reload Register is reloaded to <CPUTimer1>, it then continues to count.
2	CPUTimer1En	RW 0x0	CPU Timer 1 Enable 0 = False 1 = True
1	CPUTimer0Auto	RW 0x0	CPU Timer 0 Auto Mode When this bit is cleared to 0 and <CPUTimer0> in the CPU Timer0 Register has reached zero, <CPUTimer0> stops counting. When this bit is set to 1 and <CPUTimer0> has reached zero, <CPUTimer0Rel> in the CPU Timer0 Reload Register is reloaded to <CPUTimer0>, it then continues to count.

Table 248: Timers Control Register (N=0–3) (Continued)

Offset: CPU ID0: 0x00021840, CPU ID1: 0x00021940, CPU ID2: 0x00021A40, CPU ID3: 0x00021B40
Offset Formula: 0x00021840+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
0	CPUTimer0En	RW 0x0	CPU Timer 0 Enable 0 = False 1 = True

Table 249: Timer0 Reload Register (N=0–3)

Offset: CPU ID0: 0x00021850, CPU ID1: 0x00021950, CPU ID2: 0x00021A50, CPU ID3: 0x00021B50
Offset Formula: 0x00021850+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	CPUTimer0Rel	RW 0x0	CPU Timer 0 Reload This field contains the reload value of timer 0, it is used as the reload value in Periodic mode.

Table 250: Timer 0 Register (N=0–3)

Offset: CPU ID0: 0x00021854, CPU ID1: 0x00021954, CPU ID2: 0x00021A54, CPU ID3: 0x00021B54
Offset Formula: 0x00021854+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	CPUTimer0	RW 0x0	CPU Timer 0 This 32-bit counter is decremented every device internal clock. When field <CPUTimer0En> in the CPU Timer Control Register = 0,<CPUTimer0> stops. When <CPUTimer0En> = 1 and <CPUTimer0Auto> in the CPU Timer Control Register = 0, <CPUTimer0> is decremented until it reaches to 0, then it stops. When <CPUTimer0En> = 1and <CPUTimer0Auto> = 1, <CPUTimer0> is decremented until it reaches to 0, then the field <CPUTimer0Rel> in the CPU 0 Reload Register is reload to <CPUTimer0> and then <CPUTimer0> continues to count. When <CPUTimer0> reaches 0, bit <CPUTimer0IntReq> in the MBUS-L to Mbus Bridge Interrupt Cause Register is set to 1.

Copyright © 2014 Marvell

Table 251: Timer1 Reload Register (N=0–3)

Offset: CPU ID0: 0x00021858, CPU ID1: 0x00021958, CPU ID2: 0x00021A58, CPU ID3: 0x00021B58

Offset Formula: 0x00021858+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	CPUTimer1Rel	RW 0x0	CPU Timer 1 Reload See the CPU Timer0 Reload Register.

Table 252: Timer 1 Register (N=0–3)

Offset: CPU ID0: 0x0002185C, CPU ID1: 0x0002195C, CPU ID2: 0x00021A5C, CPU ID3: 0x00021B5C

Offset Formula: 0x0002185C+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	CPUTimer1	RW 0x0	CPU Timer 1 See the CPU Timer 0 Register.

Table 253: Watchdog Timer Reload Register (N=0–3)

Offset: CPU ID0: 0x00021860, CPU ID1: 0x00021960, CPU ID2: 0x00021A60, CPU ID3: 0x00021B60

Offset Formula: 0x00021860+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	CPUWDTimerLen	RW 0x0	CPU Watchdog Timer Length See the CPU Timer0 Reload Register.

Table 254: Watchdog Timer Register (N=0–3)

Offset: CPU ID0: 0x00021864, CPU ID1: 0x00021964, CPU ID2: 0x00021A64, CPU ID3: 0x00021B64

Offset Formula: 0x00021864+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	CPUWDTimer	RW 0x7FFFFFFF	CPU Watchdog Timer See the CPU Timer 0 Register.

Table 255: Timers Events Status register Register (N=0–3)

Offset: CPU ID0: 0x00021868, CPU ID1: 0x00021968, CPU ID2: 0x00021A68, CPU ID3: 0x00021B68

Offset Formula: 0x00021868+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24	WD timer expired	RW0C 0x0	Sticky indication for timer expiration
23:9	Reserved	RSVD 0x0	Reserved
8	Timer1 expired	RW0C 0x0	Sticky indication for timer expiration
7:1	Reserved	RSVD 0x0	Reserved
0	Timer0 expired	RW0C 0x0	Sticky indication for timer expiration

A.2.6 Cache Lockdown

Table 256: L2 CPUn Data Lockdown Register (n=0–3)

Offset: CPU ID0: 0x00008900, CPU ID1: 0x00008908, CPU ID2: 0x00008910, CPU ID3: 0x00008918

Offset Formula: 0x00008900+8*n: where n (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31	CPU Data Way31 Lock Down	RW 0x0	CPU Data Way31 Lock Down See definition at Way0 Lock Down
30	CPU Data Way30 Lock Down	RW 0x0	CPU Data Way30 Lock Down See definition at Way0 Lock Down
29	CPU Data Way29 Lock Down	RW 0x0	CPU Data Way29 Lock Down See definition at Way0 Lock Down
28	CPU Data Way28 Lock Down	RW 0x0	CPU Data Way28 Lock Down See definition at Way0 Lock Down
27	CPU Data Way27 Lock Down	RW 0x0	CPU Data Way27 Lock Down See definition at Way0 Lock Down
26	CPU Data Way26 Lock Down	RW 0x0	CPU Data Way26 Lock Down See definition at Way0 Lock Down

Table 256: L2 CPUn Data Lockdown Register (n=0–3) (Continued)

Offset: CPU ID0: 0x00008900, CPU ID1: 0x00008908, CPU ID2: 0x00008910, CPU ID3: 0x00008918

Offset Formula: 0x00008900+8*n: where n (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
25	CPU Data Way25 Lock Down	RW 0x0	CPU Data Way25 Lock Down See definition at Way0 Lock Down
24	CPU Data Way24 Lock Down	RW 0x0	CPU Data Way24 Lock Down See definition at Way0 Lock Down
23	CPU Data Way23 Lock Down	RW 0x0	CPU Data Way23 Lock Down See definition at Way0 Lock Down
22	CPU Data Way22 Lock Down	RW 0x0	CPU Data Way22 Lock Down See definition at Way0 Lock Down
21	CPU Data Way21 Lock Down	RW 0x0	CPU Data Way21 Lock Down See definition at Way0 Lock Down
20	CPU Data Way20 Lock Down	RW 0x0	CPU Data Way20 Lock Down See definition at Way0 Lock Down
19	CPU Data Way19 Lock Down	RW 0x0	CPU Data Way19 Lock Down See definition at Way0 Lock Down
18	CPU Data Way18 Lock Down	RW 0x0	CPU Data Way18 Lock Down See definition at Way0 Lock Down
17	CPU Data Way17 Lock Down	RW 0x0	CPU Data Way17 Lock Down See definition at Way0 Lock Down
16	CPU Data Way16 Lock Down	RW 0x0	CPU Data Way16 Lock Down See definition at Way0 Lock Down
15	CPU Data Way15 Lock Down	RW 0x0	CPU Data Way15 Lock Down See definition at Way0 Lock Down
14	CPU Data Way14 Lock Down	RW 0x0	CPU Data Way14 Lock Down See definition at Way0 Lock Down
13	CPU Data Way13 Lock Down	RW 0x0	CPU Data Way13 Lock Down See definition at Way0 Lock Down
12	CPU Data Way12 Lock Down	RW 0x0	CPU Data Way12 Lock Down See definition at Way0 Lock Down

Copyright © 2014 Marvell

Table 256: L2 CPUn Data Lockdown Register (n=0–3) (Continued)

Offset: CPU ID0: 0x00008900, CPU ID1: 0x00008908, CPU ID2: 0x00008910, CPU ID3: 0x00008918

Offset Formula: 0x00008900+8*n: where n (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
11	CPU Data Way11 Lock Down	RW 0x0	CPU Data Way11 Lock Down See definition at Way0 Lock Down
10	CPU Data Way10 Lock Down	RW 0x0	CPU Data Way10 Lock Down See definition at Way0 Lock Down
9	CPU Data Way9 Lock Down	RW 0x0	CPU Data Way9 Lock Down See definition at Way0 Lock Down
8	CPU Data Way8 Lock Down	RW 0x0	CPU Data Way8 Lock Down See definition at Way0 Lock Down
7	CPU Data Way7 Lock Down	RW 0x0	CPU Data Way7 Lock Down See definition at Way0 Lock Down
6	CPU Data Way6 Lock Down	RW 0x0	CPU Data Way6 Lock Down See definition at Way0 Lock Down
5	CPU Data Way5 Lock Down	RW 0x0	CPU Data Way5 Lock Down See definition at Way0 Lock Down
4	CPU Data Way4 Lock Down	RW 0x0	CPU Data Way4 Lock Down See definition at Way0 Lock Down
3	CPU Data Way3 Lock Down	RW 0x0	CPU Data Way3 Lock Down See definition at Way0 Lock Down
2	CPU Data Way2 Lock Down	RW 0x0	CPU Data Way2 Lock Down See definition at Way0 Lock Down
1	CPU Data Way1 Lock Down	RW 0x0	CPU Data Way1 Lock Down See definition at Way0 Lock Down
0	CPU Data Way0 Lock Down	RW 0x0	CPU Data Way0 Lock Down This field defines if way0 is locked down from being filled by a line fill originated by the CPU Data request 0 = Unlock: CPU Data requests can be allocated in Way0 1 = Lock: CPU Data requests will not be allocated in Way0

Table 257: L2 CPUn Instruction Lockdown Register (n=0–3)

Offset: CPU ID0: 0x00008904, CPU ID1: 0x0000890C, CPU ID2: 0x00008914, CPU ID3: 0x0000891C

Offset Formula: 0x00008904+8*n: where n (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31	CPU Instruction Way31 Lock Down	RW 0x0	CPU Instruction Way31 Lock Down See description at CPU Instruction Way0 Lock Down
30	CPU Instruction Way30 Lock Down	RW 0x0	CPU Instruction Way30 Lock Down See description at CPU Instruction Way0 Lock Down
29	CPU Instruction Way29 Lock Down	RW 0x0	CPU Instruction Way29 Lock Down See description at CPU Instruction Way0 Lock Down
28	CPU Instruction Way28 Lock Down	RW 0x0	CPU Instruction Way28 Lock Down See description at CPU Instruction Way0 Lock Down
27	CPU Instruction Way27 Lock Down	RW 0x0	CPU Instruction Way27 Lock Down See description at CPU Instruction Way0 Lock Down
26	CPU Instruction Way26 Lock Down	RW 0x0	CPU Instruction Way26 Lock Down See description at CPU Instruction Way0 Lock Down
25	CPU Instruction Way25 Lock Down	RW 0x0	CPU Instruction Way25 Lock Down See description at CPU Instruction Way0 Lock Down
24	CPU Instruction Way24 Lock Down	RW 0x0	CPU Instruction Way24 Lock Down See description at CPU Instruction Way0 Lock Down
23	CPU Instruction Way23 Lock Down	RW 0x0	CPU Instruction Way23 Lock Down See description at CPU Instruction Way0 Lock Down
22	CPU Instruction Way22 Lock Down	RW 0x0	CPU Instruction Way22 Lock Down See description at CPU Instruction Way0 Lock Down
21	CPU Instruction Way21 Lock Down	RW 0x0	CPU Instruction Way21 Lock Down See description at CPU Instruction Way0 Lock Down
20	CPU Instruction Way20 Lock Down	RW 0x0	CPU Instruction Way20 Lock Down See description at CPU Instruction Way0 Lock Down
19	CPU Instruction Way19 Lock Down	RW 0x0	CPU Instruction Way19 Lock Down See description at CPU Instruction Way0 Lock Down

Table 257: L2 CPUn Instruction Lockdown Register (n=0–3) (Continued)

Offset: CPU ID0: 0x00008904, CPU ID1: 0x0000890C, CPU ID2: 0x00008914, CPU ID3: 0x0000891C

Offset Formula: 0x00008904+8*n: where n (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
18	CPU Instruction Way18 Lock Down	RW 0x0	CPU Instruction Way18 Lock Down See description at CPU Instruction Way0 Lock Down
17	CPU Instruction Way17 Lock Down	RW 0x0	CPU Instruction Way17 Lock Down See description at CPU Instruction Way0 Lock Down
16	CPU Instruction Way16 Lock Down	RW 0x0	CPU Instruction Way16 Lock Down See description at CPU Instruction Way0 Lock Down
15	CPU Instruction Way15 Lock Down	RW 0x0	CPU Instruction Way15 Lock Down See description at CPU Instruction Way0 Lock Down
14	CPU Instruction Way14 Lock Down	RW 0x0	CPU Instruction Way14 Lock Down See description at CPU Instruction Way0 Lock Down
13	CPU Instruction Way13 Lock Down	RW 0x0	CPU Instruction Way13 Lock Down See description at CPU Instruction Way0 Lock Down
12	CPU Instruction Way12 Lock Down	RW 0x0	CPU Instruction Way12 Lock Down See description at CPU Instruction Way0 Lock Down
11	CPU Instruction Way11 Lock Down	RW 0x0	CPU Instruction Way11 Lock Down See description at CPU Instruction Way0 Lock Down
10	CPU Instruction Way10 Lock Down	RW 0x0	CPU Instruction Way10 Lock Down See description at CPU Instruction Way0 Lock Down
9	CPU Instruction Way9 Lock Down	RW 0x0	CPU Instruction Way9 Lock Down See description at CPU Instruction Way0 Lock Down
8	CPU Instruction Way8 Lock Down	RW 0x0	CPU Instruction Way8 Lock Down See description at CPU Instruction Way0 Lock Down
7	CPU Instruction Way7 Lock Down	RW 0x0	CPU Instruction Way7 Lock Down See description at CPU Instruction Way0 Lock Down
6	CPU Instruction Way6 Lock Down	RW 0x0	CPU Instruction Way6 Lock Down See description at CPU Instruction Way0 Lock Down
5	CPU Instruction Way5 Lock Down	RW 0x0	CPU Instruction Way5 Lock Down See description at CPU Instruction Way0 Lock Down

Table 257: L2 CPUn Instruction Lockdown Register (n=0–3) (Continued)

Offset: CPU ID0: 0x00008904, CPU ID1: 0x0000890C, CPU ID2: 0x00008914, CPU ID3: 0x0000891C

Offset Formula: 0x00008904+8*n: where n (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
4	CPU Instruction Way4 Lock Down	RW 0x0	CPU Instruction Way4 Lock Down See description at CPU Instruction Way0 Lock Down
3	CPU Instruction Way3 Lock Down	RW 0x0	CPU Instruction Way3 Lock Down See description at CPU Instruction Way0 Lock Down
2	CPU Instruction Way2 Lock Down	RW 0x0	CPU Instruction Way2 Lock Down See description at CPU Instruction Way0 Lock Down
1	CPU Instruction Way1 Lock Down	RW 0x0	CPU Instruction Way1 Lock Down See description at CPU Instruction Way0 Lock Down
0	CPU Instruction Way0 Lock Down	RW 0x0	CPU Instruction Way0 Lock Down When this bit is set, Way0 is locked down for allocation of CPU instruction line fetches that have missed. 0 = Unlock: CPU Instruction fetches can be allocated in Way0 1 = Lock: CPU Instruction fetches will not be allocated in Way0

Table 258: IO Bridge Lockdown Register

Offset: 0x00008984

Bit	Field	Type/InitVal	Description
31	IOBridgeWay31LockDown	RW 0x0	IO Bridge Way31 Lock Down See description at IO Bridge Way0 Lock down
30	IOBridgeWay30LockDown	RW 0x0	IO Bridge Way30 Lock Down See description at IO Bridge Way0 Lock down
29	IOBridgeWay29LockDown	RW 0x0	IO Bridge Way29 Lock Down See description at IO Bridge Way0 Lock down
28	IOBridgeWay28LockDown	RW 0x0	IO Bridge Way28 Lock Down See description at IO Bridge Way0 Lock down
27	IOBridgeWay27LockDown	RW 0x0	IO Bridge Way27 Lock Down See description at IO Bridge Way0 Lock down

Table 258: IO Bridge Lockdown Register (Continued)
Offset: 0x00008984

Bit	Field	Type/InitVal	Description
26	IOBridgeWay26LockDown	RW 0x0	IO Bridge Way26 Lock Down See description at IO Bridge Way0 Lock down
25	IOBridgeWay25LockDown	RW 0x0	IO Bridge Way25 Lock Down See description at IO Bridge Way0 Lock down
24	IOBridgeWay24LockDown	RW 0x0	IO Bridge Way24 Lock Down See description at IO Bridge Way0 Lock down
23	IOBridgeWay23LockDown	RW 0x0	IO Bridge Way23 Lock Down See description at IO Bridge Way0 Lock down
22	IOBridgeWay22LockDown	RW 0x0	IO Bridge Way22 Lock Down See description at IO Bridge Way0 Lock down
21	IOBridgeWay21LockDown	RW 0x0	IO Bridge Way21 Lock Down See description at IO Bridge Way0 Lock down
20	IOBridgeWay20LockDown	RW 0x0	IO Bridge Way20 Lock Down See description at IO Bridge Way0 Lock down
19	IOBridgeWay19LockDown	RW 0x0	IO Bridge Way19 Lock Down See description at IO Bridge Way0 Lock down
18	IOBridgeWay18LockDown	RW 0x0	IO Bridge Way18 Lock Down See description at IO Bridge Way0 Lock down
17	IOBridgeWay17LockDown	RW 0x0	IO Bridge Way17 Lock Down See description at IO Bridge Way0 Lock down
16	IOBridgeWay16LockDown	RW 0x0	IO Bridge Way16 Lock Down See description at IO Bridge Way0 Lock down

Table 258: IO Bridge Lockdown Register (Continued)

Offset: 0x00008984

Bit	Field	Type/InitVal	Description
15	IOBridgeWay15LockDown	RW 0x0	IO Bridge Way15 Lock Down See description at IO Bridge Way0 Lock down
14	IOBridgeWay14LockDown	RW 0x0	IO Bridge Way14 Lock Down See description at IO Bridge Way0 Lock down
13	IOBridgeWay13LockDown	RW 0x0	IO Bridge Way13 Lock Down See description at IO Bridge Way0 Lock down
12	IOBridgeWay12LockDown	RW 0x0	IO Bridge Way12 Lock Down See description at IO Bridge Way0 Lock down
11	IOBridgeWay11LockDown	RW 0x0	IO Bridge Way11 Lock Down See description at IO Bridge Way0 Lock down
10	IOBridgeWay10LockDown	RW 0x0	IO Bridge Way10 Lock Down See description at IO Bridge Way0 Lock down
9	IOBridgeWay9LockDown	RW 0x0	IO Bridge Way9 Lock Down See description at IO Bridge Way0 Lock down
8	IOBridgeWay8LockDown	RW 0x0	IO Bridge Way8 Lock Down See description at IO Bridge Way0 Lock down
7	IOBridgeWay7LockDown	RW 0x0	IO Bridge Way7 Lock Down See description at IO Bridge Way0 Lock down
6	IOBridgeWay6LockDown	RW 0x0	IO Bridge Way6 Lock Down See description at IO Bridge Way0 Lock down
5	IOBridgeWay5LockDown	RW 0x0	IO Bridge Way5 Lock Down See description at IO Bridge Way0 Lock down

Table 258: IO Bridge Lockdown Register (Continued)
Offset: 0x00008984

Bit	Field	Type/InitVal	Description
4	IOBridgeWay4Lock Down	RW 0x0	IO Bridge Way4 Lock Down See description at IO Bridge Way0 Lock down
3	IOBridgeWay3Lock Down	RW 0x0	IO Bridge Way3 Lock Down See description at IO Bridge Way0 Lock down
2	IOBridgeWay2Lock Down	RW 0x0	IO Bridge Way2 Lock Down See description at IO Bridge Way0 Lock down
1	IOBridgeWay1Lock Down	RW 0x0	IO Bridge Way1 Lock Down See description at IO Bridge Way0 Lock down
0	IOBridgeWay0Lock Down	RW 0x0	IO Bridge Way0 Lock Down This field defines if way0 is locked down from being filled by a line fill originated by a request issued by one of the IO units (through the IO Bridge) 0 = Unlock: Coherent IO Bridge requests can be allocated in Way0 1 = Lock: Coherent IO Bridge requests will not be allocated in Way0

A.2.7 Cache Error Control

Table 259: L2 ECC Error Count Register
Offset: 0x00008600

Bit	Field	Type/InitVal	Description
31	ClrCnt	RW 0x0	Writing 1 to this register clears the correctable and uncorrectable error counter. Reading always returns 0.
30:16	UncorrectableCnt	RO 0x0	Uncorrectable Error Counter The uncorrectable error interrupt is set when the counter reaches <L2UCTreshold>
15:0	CorrectableCnt	RO 0x0	Correctable Error Counter The correctable error interrupt is set when the counter reaches <L2CorrTreshold>

Table 260: L2 ECC Error Threshold Register

Offset: 0x00008604

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved
23:16	L2UCThreshold	RW 0x0	L2 Uncorrectable ECC Error Threshold When the threshold value is cleared to 0, detected uncorrectable ECC errors will not set L2EccUncorrectable interrupt cause field
15:0	L2CorrThreshold	RW 0x0	L2 Correctable Error Threshold When the threshold value is cleared to 0, detected correctable ECC errors will not set L2EccCorrectable interrupt cause field

Table 261: L2 Error Attr Capture Register

Offset: 0x00008608

Bit	Field	Type/InitVal	Description
31	MultiSrcErrorCapLost	RO 0x0	Multiple Source Error Capture Lost The L2 Controller sets this bit to 1 whenever the Error Capture registers contain valid error information, and a new error originated by another source is detected but cannot be captured
30:21	Reserved	RSVD 0x0	Reserved
20	CapErrDirty	RO 0x0	Dirty bit of Capture Error
19	Reserved	RSVD 0x0	Reserved
18:16	CapErrSource	RO 0x0	Transaction Source of the Captured Error 0 = CPU0 1 = CPU1 2 = CPU2 3 = CPU3 7 = IO

Table 261: L2 Error Attr Capture Register (Continued)

Offset: 0x00008608

Bit	Field	Type/InitVal	Description
15:12	CapTransType	RO 0x0	Transaction Type of the Captured Error 0 = Data Read: Data read operation. 1 = Instruction Read: L1 instruction cache read operation. 2 = Clean-Flush: Clean or flush of dirty line operation. 3 = Eviction: Dirty line eviction result from replacement operation. 4 = Read-Modifed-Write: Read-Modified-Write result from write operation
11:10	Reserved	RSVD 0x0	Reserved
9:8	CapErrType	RO 0x0	Captured Error Type 0 = CorrECC: Correctable ECC Error 1 = UnCorrECC: UnCorrectable ECC Error 2 = TagParity: Tag Parity Error
7:1	Reserved	RSVD 0x0	Reserved
0	CapErrValid	RW1C 0x0	Error Capture Registers Valid Software must clear this bit to enable capturing of new detected error. 0 = Non Valid: L2 capture registers contain no valid information. 1 = Valid: L2 capture registers contain valid information of the last captured error (Data ECC or Tag Parity).

Table 262: L2 Error Address Capture Register

Offset: 0x0000860C

Bit	Field	Type/InitVal	Description
31:5	CapErrAddr	RO 0x0	Captured Error Physical Address Captures the L2 address bits [31:5], corresopnding to the first detected Data ECC or Tag Parity error.
4	Reserved	RSVD 0x0	Reserved
3:0	CapErrAddrExt	RO 0x0	Captured Error Physical Address Extension Captures the L2 address bits [35:32], corresopnding to the first detected Data ECC or Tag Parity error.

Table 263: L2 Error Way Set Capture Register

Offset: 0x00008610

Bit	Field	Type/InitVal	Description
31:20	Reserved	RSVD 0x0	Reserved
19:8	CapErrIndex	RO 0x0	Captured Error Cache Index Captures the Index in which the L2 cache error was discovered. Locked when the <CapErrValid> field is set by the hardware.
7:5	Reserved	RSVD 0x0	Reserved
4:1	CapErrWay	RO 0x0	Captured Error Cache Way Captures the Way in which the L2 cache memory error was discovered. Locked when the <CapErrValid> field is set by the hardware
0	Reserved	RSVD 0x0	Reserved

Table 264: L2 Error Injection Control Register

Offset: 0x00008614

Bit	Field	Type/InitVal	Description
31:5	ErrAddrInject	RW 0x0	Error Address Inject Specifies the Address to inject the Data ECC or Tag parity errors.
4:2	Reserved	RO 0x0	Reserved
1	TagParErrInjectEn	RW 0x0	L2 Tag Parity Error Injection Enable 0 = Disable: L2 Tag Parity Error Injection disabled. 1 = Enable: L2 Tag Parity Error Injection enabled.
0	EccErrInjectEn	RW 0x0	L2 ECC Error Injection Enable 0 = Disable: L2 ECC Error Injection disabled. 1 = Enable: L2 ECC Error Injection enabled.

Table 265: L2 ECC Error Injection Mask Register
Offset: 0x00008618

Bit	Field	Type/InitVal	Description
31:0	Error Injection Mask	RW 0x0	L2 ECC Error Inject Mask When <EccErrInjectEn> is set and a CL that is targeted to be error injected enters L2 due to a linefill, this field determines the bits in the ECC that will be flipped. A value of 1 in bit[i] of this field will cause bit[i] of the stored ECC word to be the negation of bit[i[in the calculated ECC.

A.2.8 Cache Power Management

Table 266: L2 Dynamic Memory Power Down Register
Offset: 0x00008A00

Bit	Field	Type/InitVal	Description
31	DMPDEnable	RW 0x0	Dynamic Memory Power Down Enable When this bit is set, upon an IDLE period of <DMPD_Treshold> L2 clock cycles , L2 memories will enter power down retention mode.
30:16	Reserved	RSVD 0x0	Reserved
15:0	DMPDTreshold	RW 0x0	Dynamic Memory Power Down Threshold

Table 267: L2 Dynamic Clock Gating Register
Offset: 0x00008A04

Bit	Field	Type/InitVal	Description
31	DCPDEnable	RW 0x0	Dynamic Clock Power Down Enable When this bit is set, upon <DCPD_Treshold> of IDLE L2 clock cycles, the L2 clock will be disabled.
30:16	Reserved	RSVD 0x0	Reserved
15:0	DCPDTreshold	RW 0x0	Dynamic Clock Power Down Threshold

Table 268: L2 Force Power Down Register
Offset: 0x00008A08

Bit	Field	Type/InitVal	Description
31:9	Reserved	RSVD 0x0	Reserved
8	FCPD	WO 0x0	L2 Force Clock Down When 0x1 is written to this field, L2 stops its clock. The clock will resume upon the following request that is issued to L2.
7:1	Reserved	RSVD 0x0	Reserved
0	FMPD	WO 0x0	L2 force Memory Power Down When 0x1 is written to this field, L2 inserts the memories into a low leakage state. The memories exit from this state upon the following request that is issued to L2.

A.2.9 Cache Maintenance

Table 269: L2 Sync Barrier Register
Offset: 0x00008700

Bit	Field	Type/InitVal	Description
31:0	Various	WO 0x0	Writing to this register performs a draining operation of the L2 eviction buffer. The write is completed when all maintenance operations of the issuing CPU are finished, and the eviction buffer is empty. This is an automic opetation, the L2 slave ports are stalled till the operation is completed.

Table 270: L2 Maintenance Status Register
Offset: 0x00008704

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved
3	CPU3 Op Pending	RO 0x0	CPU3 Pending Operation Status 0 = Non-Active: There is no active operation. 1 = Active: There is an operation that was issued and has not been completed.

Table 270: L2 Maintenance Status Register (Continued)

Offset: 0x00008704

Bit	Field	Type/InitVal	Description
2	CPU2 Op Pending	RO 0x0	CPU2 Pending Operation Status 0 = Non-Active: There is no active operation. 1 = Active: There is an operation that was issued and has not been completed.
1	CPU1 Op Pending	RO 0x0	CPU1 Pending Operation Status 0 = Non-Active: There is no active operation. 1 = Active: There is an operation that was issued and has not been completed.
0	CPU0 Op Pending	RO 0x0	CPU0 Pending Operation Status 0 = Non-Active: There is no active operation. 1 = Active: There is an operation that was issued and has not been completed.

Table 271: L2 Range Base Address CPU n Register (n=0–3)

Offset: CPU_ID0: 0x00008710, CPU_ID1: 0x00008714, CPU_ID2: 0x00008718, CPU_ ID3: 0x0000871C

Offset Formula: 0x00008710+4*n: where n (0-3) represents CPU_ID

Bit	Field	Type/InitVal	Description
31:0	RangeBaseAddr	RW 0x0	L2 Range Base Address CPUn This register holds the start address of the range to perform the flushing, cleaning, or invalidating range operation. The actual physical address is {bits[3:0],bits[31:5],00000} If LPAE is not used, set bits[4:0] to 0x0. There is a dedicated register per-CPU. This fact enables range operation without synchronization between multiple CPUs.

Copyright © 2014 Marvell

Table 272: L2 Range Base Address Virtual Register

Offset: 0x00008720

Bit	Field	Type/InitVal	Description
31:0	Base Address	RW 0x0	This register is a virtual register that is dedicated per CPU. When CPUx accesses this register for reading or writing, the corresponding BaseAddress is accessed. In practice: For CPU0 accesses, 0x8710 is accessed. For CPU1 accesses, 0x8714 is accessed. For CPU2 accesses, 0x8718 is accessed. For CPU3 accesses, 0x871C is accessed.

Table 273: L2 Block Allocation Extend Addr Virtual Register

Offset: 0x00008724

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved
3:0	BlockAllocationExtendedAddr	RW 0x0	Block Allocation Extended Address Virtual reg This is a service reg that provides SW support of setting the Allocation Extension Address without need of a self identification. When this reg is accesses with a read or a write by a specific CPU the corresponding AllocationBaseAddrExt is accessed. In particular, Upon CPU0 accesses, 0x8740 is accessed Upon CPU1 accesses, 0x8744 is accessed Upon CPU2 accesses, 0x8748 is accessed Upon CPU3 accesses, 0x874C is accessed

Table 274: L2 Block Allocation Extend Addr CPU n Register (n=0–3)

Offset: CPU_ID0: 0x00008740, CPU_ID1: 0x00008744, CPU_ID2: 0x00008748, CPU_ID3: 0x0000874C

Offset Formula: 0x00008740+4*n: where n (0-3) represents CPU_ID

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 274: L2 Block Allocation Extend Addr CPU n Register (n=0–3) (Continued)

Offset: CPU_ID0: 0x00008740, CPU_ID1: 0x00008744, CPU_ID2: 0x00008748, CPU_ID3: 0x0000874C

Offset Formula: 0x00008740+4*n: where n (0-3) represents CPU_ID

Bit	Field	Type/InitVal	Description
3:0	Allocation Base Extended Addr	RW 0x0	L2 Block Allocation Address Extension for CPUn This field defines ADDR[35:32] of block allocations requests that are performed by CPU n

Table 275: L2 Invalidate by Physical Address Register

Offset: 0x00008770

Bit	Field	Type/InitVal	Description
31:0	InvalidatePA	WO 0x0	Writing to this register causes a specific L2 cache line, identified by its address, to be marked as not valid. The base address of the cache line that will be invalidated is: {bits[3:0], bits[31:5], 00000} If LPAE is not used, bits[3:0] should be 0x0

Table 276: L2 Invalidate Range Register

Offset: 0x00008774

Bit	Field	Type/InitVal	Description
31:0	InvalidateRange	WO 0x0	Writing to this register causes a specified range of addresses to be marked as invalid. The range of addresses are all cache-line addresses between the <RangeBaseAddr> associated with the requesting CPU and PA derived from data written to this register, where: PA = {bits[3:0], bits[31:5], 00000} If LPAE is not used, bits[3:0] should be 0x0

Table 277: L2 Invalidate by Index Way Register

Offset: 0x00008778

Bit	Field	Type/InitVal	Description
31:27	Way	WO 0x0	L2 Invalidate Way Defines the target way that should be invalidated. The encoding is the binary value of {bit[27],bits[31:28]}

Table 277: L2 Invalidate by Index Way Register (Continued)
Offset: 0x00008778

Bit	Field	Type/InitVal	Description
26:17	Reserved	RSVD 0x0	Reserved
16:5	Index	WO 0x0	L2 Invalidate Index
4:0	Reserved	RSVD 0x0	Reserved

Table 278: L2 Invalidate by Way Register
Offset: 0x0000877C

Bit	Field	Type/InitVal	Description
31:0	InvalidateWay	WO 0x0	L2 Invalidate Way Writing to this register causes a subset of the Ways to mark all their lines as non valid. The format of the data written is: Bit 0 for Way0 Bit 1 for Way1 ...

Table 279: L2 Block Allocation Register
Offset: 0x0000878C

Bit	Field	Type/InitVal	Description
Writing to this register triggers a 64 KB block allocation sequence to a specific way. All lines that exist in the specified way are flushed and a new block replaces them.			
31:10	Allocation Base Addr	WO 0x0	L2 Block Allocation Base Address Define bits[31:10] of the base address of the allocated block. The allocated block must be 1K aligned (i.e. the actual address is this field padded with 10 0's on the right). Notice that at a system with LPAE , addr[35:32] is defined according to <L2 Block Allocation Extended Addr>.
9	Reserved	RSVD 0x0	Reserved

Table 279: L2 Block Allocation Register (Continued)

Offset: 0x0000878C

Bit	Field	Type/InitVal	Description
8	Allocation Atomicity	WO 0x0	L2 Block Allocation Atomicity Defines if the allocation is done in the background (in which the L2 continues serving requests) or in the foreground (meaning, the L2 completes the allocation before serving an additional request). 0 = Foreground 1 = Background
7:6	Allocation Data	WO 0x0	L2 Block Allocation Data Defines the data that will be written to the Allocated block. 0 = Disable: The data in the allocated way is undefined. 1 = ClearData: The data in the allocated way will be all 0's 3 = SetData: The data in the allocated way will be all 1's
5	Reserved	RSVD 0x0	Reserved
4:0	Allocation Way ID	WO 0x0	L2 Block Allocation Way ID Define the ID of the target way to be allocated.

Table 280: L2 Clean by Physical Address Register

Offset: 0x000087B0

Bit	Field	Type/InitVal	Description
31:0	Clean PA	WO 0x0	Writing to this register causes a specific L2 cache line, identified by its address, to be marked as clean. Where the line is marked as dirty, a copy of the modified line is written to the lower memory hierarchy, through the eviction buffer. The line remains validated in the cache. Where the line does not exist or is not dirty, no action is performed. Writing to this register causes a specific L2 cache line, identified by its address, to be marked as not valid. The base address of the cache line that will be cleaned is: {bits[3:0], bits[31:5], 00000} If LPAE is not used, bits[3:0] should be 0x0

Table 281: L2 Clean by Range Register

Offset: 0x000087B4

Bit	Field	Type/InitVal	Description
31:0	CleanRange	WO 0x0	Writing to this register causes a specified range of addresses to be marked as clean. When a line is marked as valid and dirty, a copy of the modified line is written to the lower memory hierarchy, through the eviction buffer. The line remains validated in the cache. When a line is not valid or is not marked dirty, no action is performed on the line. The range of addresses are all cache-line addresses between the <RangeBaseAddr> associated with the requesting CPU and PA derived from data written to this register, where: PA = {bits[3:0], bits[31:5], 00000} If LPAE is not used, bits[3:0] should be 0x0

Table 282: L2 Clean by Index Way Register

Offset: 0x000087B8

Bit	Field	Type/InitVal	Description
Writing to this register causes a specific L2 cache line, identified by its way and index, to be marked as clean. When the line is marked as dirty, a copy of the modified line is written to the lower memory hierarchy through the eviction buffer. The line remains validated in the cache. When the target line is not valid or is not marked dirty, no action is performed.			
31:27	Way	WO 0x0	L2 Clean Way Defines the target way that should be cleaned. The encoding is the binary value of {bit[27],bits[31:28]}
26:17	Reserved	RSVD 0x0	Reserved
16:5	Index	WO 0x0	L2 Clean Index
4:0	Reserved	RSVD 0x0	Reserved

Table 283: L2 Clean by Way Register

Offset: 0x000087BC

Bit	Field	Type/InitVal	Description
31:0	CleanWay	WO 0x0	L2 Clean Way Writing to this register causes a subset of the L2 Ways to clean all their valid cache lines. When a line is marked as valid and dirty, a copy of the modified line is written to the lower memory hierarchy, through the eviction buffer. The line remains validated in the cache. When a line is not valid or is not marked dirty, no action is performed on the line. The format is: Bit 0 for Way0 Bit 1 for Way1 ...

Table 284: L2 Flush by Physical Address Register

Offset: 0x000087F0

Bit	Field	Type/InitVal	Description
31:0	FlushPA	WO 0x0	Cleans and invalidates the cache line by address. Writing to this register causes a specific L2 cache line, identified by its address, to be marked as invalid and, when the line is marked as dirty, written to the lower memory hierarchy. Ifthe line does not exist, no action is performed. The base address of the cache line that will be flushed is: {bits[3:0], bits[31:5], 00000} If LPAE is not used, bits[3:0] should be 0x0

Table 285: L2 Flush by Range Register

Offset: 0x000087F4

Bit	Field	Type/InitVal	Description
31:0	FlushRange	WO 0x0	Writing to this register causes a specified range of addresses to be cleaned and invalidated. When a line is marked as valid and dirty, a copy of the modified line is written to the lower memory hierarchy, through the eviction buffer. When a specific address in the range is not in the cache, no action is performed on that line. The range of addresses are all cache-line addresses between the <RangeBaseAddr> associated with the requesting CPU and PA derived from data written to this register, where: PA = {bits[3:0], bits[31:5], 00000} If LPAE is not used, bits[3:0] should be 0x0

Table 286: L2 Flush by Index Way Register

Offset: 0x000087F8

Bit	Field	Type/InitVal	Description
Writing to this register causes a specific L2 cache line, identified by its way and index, to be invalidated and cleaned. When the target line is valid, it will be invalidated and if marked as dirty a copy of the modified line is written to the lower memory hierarchy through the eviction buffer. When the target line is not valid, no action is performed.			
31:27	Way	WO 0x0	L2 Flush Way Defines the target way that should be flushed The encoding is the binary value of {bit[27],bits[31:28]}
26:17	Reserved	RSVD 0x0	Reserved
16:5	Index	WO 0x0	L2 Flush Index
4:0	Reserved	RSVD 0x0	Reserved

Table 287: L2 Flush by Way Register
Offset: 0x000087FC

Bit	Field	Type/InitVal	Description
31:0	FlushWay	WO 0x0	L2 Flush Way Writing to this register causes a subset of the L2 ways to be cleaned and invalidated. When a line is marked as valid and dirty, a copy of the modified line is written to the lower memory hierarchy, through the eviction buffer. When a line is not valid or is not marked dirty, no action is performed on the line. The format is: Bit 0 for Way0 Bit 1 for Way1 ...

A.2.10 Cache Performance Monitor

Table 288: L2 Counters Control Register
Offset: 0x00008200

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x0	Reserved
9	Counter1 Enable	RW 0x0	L2 Event Counter1 Enable When this bit is set, event counter1 is enabled. 0 = False: Counter at pause state 1 = True: Counter increments upon an event
8	Counter1 Clear	WO 0x0	Event Counter 1 Reset Writing 1 to this field resets Counter1 Value register.
7:2	Reserved	RSVD 0x0	Reserved
1	Counter0 Enable	RW 0x0	L2 Counter0 Enable When this bit is set, counter0 is enabled. 0 = False: Counter at pause state 1 = True: Counter increments upon an event
0	Counter0 Clear	WO 0x0	Event Counter0 Reset Writing 1 to this field, resets Counter0 Value register.

Table 289: L2 Counter <X>_ Configuration Register (X=0–1)

Offset: Counter ID0: 0x00008204, Counter ID1: 0x00008210

Offset Formula: 0x00008204+12 * X: where X (0-1) represents Counter ID

Bit	Field	Type/InitVal	Description
31	IO Bridge Event Enable	RW 0x0	When the <Event_Type> is configured to an event that can be associated to a Master: If an event that is associated to IO requests has occurred then the counter will increment only if this bit is set. When <event_Type> is configured to an event that can not be associated to a Master, this bit is ignored.
30:20	Reserved	RSVD 0x0	Reserved
19	CPU3 Events Enable	RW 0x0	When the <Event_Type> is configured to an event that can be associated to a Master: If an event that is associated to CPU3 has occured then the counter will increment only if this bit is set. When <event_Type> is configured to an event that can not be associated to a Master, this bit is ignored.
18	CPU2 Events Enable	RW 0x0	When the <Event_Type> is configured to an event that can be associated to a Master: If an event that is associated to CPU2 has occured then the counter will increment only if this bit is set. When <event_Type> is configured to an event that can not be associated to a Master, this bit is ignored.
17	CPU1 Events Enable	RW 0x0	When the <Event_Type> is configured to an event that can be associated to a Master: If an event that is associated to CPU1 has occured then the counter will increment only if this bit is set. When <event_Type> is configured to an event that can not be associated to a Master, this bit is ignored.
16	CPU0 Events Enable	RW 0x0	When the <Event_Type> is configured to an event that can be associated to a Master: If an event that is associated to CPU0 has occured then the counter will increment only if this bit is set. When <event_Type> is configured to an event that can not be associated to a Master, this bit is ignored.
15:8	Reserved	RSVD 0x0	Reserved

Table 289: L2 Counter <X>_ Configuration Register (X=0–1) (Continued)

Offset: Counter ID0: 0x00008204, Counter ID1: 0x00008210

Offset Formula: 0x00008204+12 * X: where X (0-1) represents Counter ID

Bit	Field	Type/InitVal	Description
7:2	L2 Event Type	RW 0x0	Counter Event Source 0 = Counter Disable: Counter Is disabled. 1 = Evict: Eviction or cast out from L2. 2 = DataRdHit: L2 Cache Data Read request hit. 3 = DataRdReq: L2 Cache Data Read request. 4 = DataWrHit: L2 Cache Data Write request hit. 5 = DataWrReq: L2 Cache Data Write request. 6 = DataWrThrReq: L2 Cache Data Write Through request. 7 = InstRdHit: L2 Cache Instruction Read hit. 8 = InstRdReq: L2 Cache Instruction Read request. 11 = WriteWAMiss: L2 Cache eviction due to full cache-line write with WA set. 12 = WriteWACLReq: L2 Cache full cache-line write with WA set. 13 = WriteWANoCLReq: L2 Cache less than full cache-line write with WA set. 16 = SRAMWr: SRAM Write request. 17 = SRAMRd: SRAM Read request. 18 = RMWWrite: Write request that hits a portion of a QW. This results in a Read Modified Write operation. 19 = SpeculativeInstReq: CPU Issues a speculative instruction fetch. 20 = SpeculativeInstHit: CPU issues a speculative instruction fetch that is resolved with a hit. 25 = ClkGated: L2 clock is gated for dynamic power reduction. 26 = MemPwdn: L2 Memories are in leakage reduction state. 27 = RGFStall: L2 Regfile stall due to maintenance operation. 28 = EBStall: Eviction buffer stall. 29 = LRBStall: Line read buffer stall. 30 = Idle: Counts the number of cycles during which the L2 was not accessed. 31 = Active: Free running clock counter.
1:0	Interrupt Gen	RW 0x0	Event Counter Interrupt Generation 0 = Disable: No interrupt is generated for this counter. 1 = Overflow: Interrupt is asserted on counter overflow. 2 = Reserved0 3 = Reserved1

Copyright © 2014 Marvell

Table 290: L2 Counter X Value Low Register (X=0–1)

Offset: Counter ID0: 0x00008208, Counter ID1: 0x00008214

Offset Formula: 0x00008208+12 * X: where X (0-1) represents Counter ID

Bit	Field	Type/InitVal	Description
31:0	Counter Value	RO 0x0	Event Counter Value This register enables the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration register. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register.

Table 291: L2 Counter X Value High Register (X=0–1)

Offset: Counter ID0: 0x0000820C, Counter ID1: 0x00008218

Offset Formula: 0x0000820C+12 * X: where X (0-1) represents Counter ID

Bit	Field	Type/InitVal	Description
31:0	Counter Value	RO 0x0	Event Counter Value This register enables the programmer to read off the counter value. The counter counts an event as specified by the Counter Configuration register. The counter can be preloaded if counting is disabled and reset by the Event Counter Control Register.

A.2.11 Cache Interrupt Control

Table 292: L2 Interrupt Cause Register

Offset: 0x00008220

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved
3	L2TagParity	RW0C 0x0	L2 Tag Parity Error
2	L2EccCorrectable	RW0C 0x0	L2 ECC Correctable Error
1	L2EccUncorrectable	RW0C 0x0	L2 ECC Uncorrectable Error

Table 292: L2 Interrupt Cause Register (Continued)
Offset: 0x00008220

Bit	Field	Type/InitVal	Description
0	Event Counter Overflow	RW0C 0x0	Event Counter overflow Interrupt will occur if one of the 2 counters has overflown

Table 293: L2 Interrupt Mask Register
Offset: 0x00008224

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved
3	L2TagParityMask	RW 0x0	L2 Tag Parity Mask
2	L2EccCorrectable Mask	RW 0x0	L2 ECC Correctable Mask
1	L2EccUncorrectable Mask	RW 0x0	L2 ECC Uncorrectable Error Mask
0	Event Counter Overflow Mask	RW 0x0	Event Counter Overflow Mask

A.2.12 Cache Main Control

Table 294: L2 Control Register
Offset: 0x00008100

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	L2Enable	RW 0x0	L2 Cache Enable Specifies if the L2 Cache is enabled. If the L2 Cache is disabled, only the L2 Regfile is accessible. When all ways are configured as SRAM, SW is encouraged to clear this bit. 0 = Disable: L2 Cache is disabled. 1 = Enable: L2 Cache is enabled.

Table 295: L2 Auxiliary Control Register
Offset: 0x00008104

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28:27	Replacement Strategy	RW 0x3	Configures one of the following replacement strategies: Way-round-robin Pseudo-random using an LFSR Semi-pLRU All strategies fill invalid unlocked ways first. Or, for each line, when all un-locked ways are valid, the victim is chosen according to the policy defined in this field. 1 = LFSR: Pseudo random using an LFSR. 3 = semi_pLRU: Every quad maintains a pLRU decision. The victim quad is chosen randomly using an LFSR
26:25	Reserved	RSVD 0x1	Reserved
24:23	Force Write Allocate	RW 0x0	Force Write Allocation Write-allocation is performed on full cache-line writes that miss the L2 cache. 0 = Requester Attribute: Use the requester attribute, CPU page-attribute, or IO attribute. 1 = Force No Allocate: Write allocate is never performed, regardless of the requester attribute. 2 = Force Allocate: Write allocate is always performed for CPU writes. Non-CPU writes are allocated according to their attribute. 3 = Force Allocate on DT only: Using the requester attribute, the CPU page-attribute, or IO attribute, except for DT which is forced to be allocated. This option is internal since active IO cannot be notifed. (if it does, it is an instance of an IO that does write_allocate while it is not outer cacheable at MMU tables.)
22	L2InvalEvicLineUCErr	RW 0x0	L2 invalidates uncorrectable error-line eviction. 0 = UCErrWrittenBack: Evicted line with uncorrectable error is written back with an error indication. 1 = UCErrInvalidate: Evicted line with uncorrectable error is invalidated.
21	ParityEnable	RW 0x0	Tag Parity Enable 0 = Disable: Tag arrays are not parity protected. 1 = Enable: Tag arrays are parity protected.
20	EccEnable	RW 0x0	Data ECC Enable 0 = Disable: Data arrays are not ECC protected. 1 = Enable: Data arrays are ECC protected.

Table 295: L2 Auxiliary Control Register (Continued)

Offset: 0x00008104

Bit	Field	Type/InitVal	Description
19:17	WaySize	RO 0x4	Way Size 2 = 16KB 3 = 32KB 4 = 64KB 5 = 128KB 6 = 256KB 7 = 512KB
16:13	Associativity	RW 0xf	L2 Cache Way Associative 3 = 4: Ways 7 = 8: Ways 11 = 16: Ways 15 = 32: Ways
12	Reserved	RSVD 0x0	Reserved
11:10	L2Size	RO 0x3	L2 Cache Size 1 = 0.5MB: 8 ways of 64KB 2 = 1MB: 16 ways of 64KB 3 = 2MB: 32 ways of 64KB
9:6	Reserved	RSVD 0xC	Reserved

Table 295: L2 Auxiliary Control Register (Continued)

Offset: 0x00008104

Bit	Field	Type/InitVal	Description
5	L2MaintenanceToPocEnable	RW 0x0	L2 Maintenance to Point of Coherent Enable This field defines rather broadcast maintenance instructions are completed upon address destination (e.g. DRAM) acceptance or upon SL2 acceptance. DMB commands are treated accordingly: When disabled, a DMB issued by a specific master will be completed only when all previous maintenance commands issued by the same master have been completed (in particular, all flushes have reached the target destination) When enabled, a DMB will be responded after all previous transactions have been received to SL2 but with no guarantee on there arrival at the PoC. Notice: When this field is disabled, SW must issue a DMB command following a bulk of maintenance commands to assure there arrival to PoC (e.g DRAM). 0 = False: Maintenance completion provided upon SL2 acceptance and DMB completed when previous maintenance accepted at PoC. 1 = True: Maintenance completion guaranteed when PoC accepted and DMB completion is not directly dependent of previous maintenance PoC acceptance.
4	L2DualEvictionEnable	RW 0x0	L2 Dual Eviction Enable When dual eviction is enabled, upon an eviction that is a result of LF, the complementary CL for achieving a 64B eviction is looked up and cleaned if necessary. For example, if the CL based at address 0x100 is replaced then the CL based at 0x120 will be cleaned too. If CL based at 0x220 is replaced then CL based at 0x200 will be cleaned too. 0 = False 1 = True
3	L2PowerDownAction	RW 0x0	L2 Power Down Action Upon a power down action, L2 dirty cache lines are transfered to their lower level hierarchy. This field defines the actual action for achieving this transfer. 0 = Clean: L2 dirty lines are written to lower level hierarchy. L2 maintains its cache lines allocation until it is physically powered down. 1 = Flush: L2 dirty lines are written to lower level hierarchy and invalidated. L2 will not maintain its cache line allocation.
2	Reserved	RSVD 0x0	Reserved

Table 295: L2 Auxiliary Control Register (Continued)
Offset: 0x00008104

Bit	Field	Type/InitVal	Description
1:0	L2 WBWT Mode	RW 0x0	L2 Write Through and Write Back policy 0 = PageAttribute: Policy rather to write to the lower memory system depends on the requester memory attribute (WB,WT). 1 = Always Write Back: Writes that are stored in L2 are not written to the lower memory system. 2 = Always Write Through: Writes are always written to the lower memory system.

A.2.13 Main Interrupt Controller Control and Configuration

Table 296: MPIC Control Register
Offset: 0x00020A00

Bit	Field	Type/InitVal	Description
31:13	Reserved	RSVD 0x0	Reserved
12:2	NumINT	RO 0x74	Number Of Interrupts Contains binary value of the maximum number of interrupt sources, also defines the highest interrupt ID to be returned on reading the IACK
1	Reserved	RSVD 0x0	Reserved
0	IRQPrioEnable	RW 0x0	IRQ Prioritization Enable 0 = False: IRQ prioritization is disabled, IRQ pin is asserted for any valid interrupt (interrupt is valid when its cause and mask bits are set) 1 = True: IRQ prioritization is enabled, IRQ pin is asserted for any valid interrupt (interrupt is valid when its cause and mask bits are set) with higher priority than Current Task Priority of the CPU

Table 297: Software Triggered Interrupt Register
Offset: 0x00020A04

Bit	Field	Type/InitVal	Description
The Register controls the issuing of software interrupts.			
31:26	Reserved_31_26	RSVD 0x0	

Copyright © 2014 Marvell

Table 297: Software Triggered Interrupt Register (Continued)

Offset: 0x00020A04

Bit	Field	Type/InitVal	Description
25:24	Target List Filter	WO 0x0	Target List Filter Controls the filtering that the distributor performs on software interrupts as follows: 0 = AllAndSelf: Send the interrupt to all CPUs that are defined to be in the Target List 1 = AllNoSelf: Send the interrupt to all CPUs that are defined to be in the Target List except the CPU that issues the request. 2 = OnlySelf: Send the interrupt only to the CPU interface that requested the interrupt 3 = RSVD: Reserved
23:12	Reserved	RSVD 0x0	Reserved
11	CPU3 in Target List	WO 0x0	CPU3 In Interrupt Target List When this bit is set, the distributor includes CPU3 in Target List and signals an interrupt accordinglly. (For usages of the TargetList, see <TargetListFilter> description)
10	CPU2 in Target List	WO 0x0	CPU2 In Interrupt Target List When this bit is set, the distributor includes CPU2 in Target List and signals an interrupt accordinglly. (For usages of the TargetList, see <TargetListFilter> description)
9	CPU1 in Target List	WO 0x0	CPU1 In Interrupt Target List When this bit is set, the distributor includes CPU1 in Target List and signals an interrupt accordinglly. (For usages of the TargetList, see <TargetListFilter> description)
8	CPU0 in Target List	WO 0x0	CPU0 In Interrupt Target List When this bit is set, the distributor includes CPU0 in Target List and signals an interrupt accordinglly. (For usages of the TargetList, see <TargetListFilter> description)
7	Reserved	RSVD 0x0	Reserved
6:5	Inbound Doorbell Sel	WO 0x0	Inbound Doorbell Select Specifies on which Inbound Doorbell register the interrupt specified in the IntID should be set 0 = Doorbell0: Inbound Doorbell 0 register, per-CPU register 1 = Doorbell1: Shared Inbound Doorbell 1 register 2 = Doorbell2: Shared Inbound Doorbell 2 register 3 = Doorbell3: Shared Inbound Doorbell 3 register

Table 297: Software Triggered Interrupt Register (Continued)

Offset: 0x00020A04

Bit	Field	Type/InitVal	Description
4:0	IntID	WO 0x0	Set this to the INTID of the software interrupt that the distributor signals to the processor, the INITID is latched in the CPU Inbound Doorbell Cause register

Table 298: SOC Main Interrupt Error Cause Register

Offset: 0x00020A20

Bit	Field	Type/InitVal	Description
31:0	Err_Int[n]	RO 0x0	Error interrupt line n [n=0..31]. For further information, see the SoC Errors Mapping table. Associated mask register is defined for CPU.

Table 299: Interrupt Set Enable (ISE) Register

Offset: 0x00020A30

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x0	Reserved
9:0	Interrupt ID	WO 0x0	Writing the Interrupt ID to this field, sets the IntEN bit in the Interrupt Source Control Register which associated with the Interrupt ID. This means interrupt delivering is now enabled.

Table 300: Interrupt Clear Enable (ICE) Register

Offset: 0x00020A34

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x0	Reserved
9:0	Interrupt ID	WO 0x0	Writing the Interrupt ID to this field, clears the IntEN bit in the Interrupt Source Control Register which associated with the Interrupt ID. This means interrupt delivering is now disabled.

Table 301: Interrupt Source i Control Register (i=0–28)

Offset: Interrupt ID0: 0x00020B00, Interrupt ID1: 0x00020B04...Interrupt ID28: 0x00020B70

Offset Formula: 0x00020B00+4*i: where i (0-28) represents Interrupt ID

Bit	Field	Type/InitVal	Description
31	EP Mask<i>	RW 0x0	Endpoint Maski When this bit is cleared, interrupt i is masked on reaching PCIe EP.
30:29	Reserved	RSVD 0x0	Reserved
28	Reserved	RO 0x1	must write 0x1
27:24	Priority<i>	RW 0x1	Interrupt Priority Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source.
23:20	Reserved	RSVD 0x0	Reserved
19	EndPoint Seli CPU3	RW 0x0	Interrupt Cause Selector For CPU3 For PCIe EP
18	EndPoint Seli CPU2	RW 0x0	Interrupt Cause Selector For CPU2 For PCIe EP
17	EndPoint Seli CPU1	RW 0x0	Interrupt Cause Selector For CPU1 For PCIe EP
16	EndPoint Seli CPU0	RW 0x0	Interrupt Cause Selector For CPU0 for PCIe EP
15:12	Reserved	RSVD 0x0	Reserved
11	FIQ Maski CPU3	RW 0x0	CPU3 FIQ Mask bit interrupti See Description at FIQ Maski CPU0
10	FIQ Maski CPU2	RW 0x0	CPU2 FIQ Mask bit interrupti See Description at FIQ Maski CPU0
9	FIQ Maski CPU1	RW 0x0	CPU1 FIQ Mask bit interrupti See Description at FIQ Maski CPU0

Table 301: Interrupt Source i Control Register (i=0–28) (Continued)

Offset: Interrupt ID0: 0x00020B00, Interrupt ID1: 0x00020B04...Interrupt ID28: 0x00020B70

Offset Formula: 0x00020B00+4*i: where i (0-28) represents Interrupt ID

Bit	Field	Type/InitVal	Description
8	FIQ Maski CPU0	RW 0x0	CPU0 FIQ Mask bit interrupt i When this field is cleared, interrupt i is masked to the CPU When this field is set, interrupt i is enabled to the CPU.. The mask affects the assertion of FIQ pin of the CPU. It does not affect the setting of bits in the cause register. The associated CPU can update this field by writing to its private IMS and ICM registers as follows: Writing i to Interrupt_ID field and 0x1 to selFIQ in the ISM sets this field Writing i to Interrupt_ID field and 0x1 to selFIQ in the ICM clears this field Notice that this mechanism provides the support for a CPU to handle FIQ interrupts without CPU identification relativity.
7:4	Reserved	RSVD 0x0	Reserved
3	IRQ Maski CPU3	RW 0x0	CPU3 IRQ Mask bit interrupti See Description at IRQ Maski CPU0
2	IRQ Maski CPU2	RW 0x0	CPU2 IRQ Mask bit interrupti See Description at IRQ Maski CPU0
1	IRQ Maski CPU1	RW 0x0	CPU1 IRQ Mask bit interrupti See Description at IRQ Maski CPU0
0	IRQ Maski CPU0	RW 0x0	CPU0 IRQ Mask bit interrupt i When this field is cleared, interrupt i is masked to the CPU When this field is set, interrupt i is enabled to the CPU.. The mask affects the assertion of IRQ pin of the CPU. It does not affect the setting of bits in the cause register. The associated CPU can update this field by writing to its private IMS and ICM registers as follows: Writing i to Interrupt_ID field and 0x0 to selFIQ in the ISM sets this field Writing i to Interrupt_ID field and 0x0 to selFIQ in the ICM clears this field Notice that this mechanism provides the support for a CPU to handle IRQ interrupts without CPU identification relativity.

Table 302: Interrupt Source i Control Register (i=29–115)

Offset: Interrupt ID29: 0x00020B74, Interrupt ID30: 0x00020B78...Interrupt ID115: 0x00020CCC

Offset Formula: 0x00020B00+4*i: where i (29-115) represents Interrupt ID

Bit	Field	Type/InitVal	Description
31	EP Mask<i>	RW 0x0	Endpoint Maski When this bit is cleared, interrupt i is masked on reaching PCIe EP.
30:29	Reserved	RSVD 0x0	Reserved
28	IntEn<i>	RW 0x0	Interrupt Enable i 0 = Disable: Interrupt delivering is disabled from this source. The event is latеched in the main cause but is never sent to any CPU. 1 = Enable: Interrupt delivering is enabled from this source.
27:24	Priority<i>	RW 0x1	Interrupt Priority Specifies the interrupt priority. The lowest priority is 0 and the highest priority is 15. A priority level of 0 disables interrupts from this source
23:12	Reserved	RSVD 0x0	Reserved
11	FIQ Maski CPU3	RW 0x0	CPU3 FIQ Mask i See Description at FIQ Maski CPU0
10	FIQ Maski CPU2	RW 0x0	CPU2 FIQ Mask i See Description at FIQ Maski CPU0
9	FIQ Maski CPU1	RW 0x0	CPU1 FIQ Mask i See Description at FIQ Maski CPU0

Table 302: Interrupt Source i Control Register (i=29–115) (Continued)

Offset: Interrupt ID29: 0x00020B74, Interrupt ID30: 0x00020B78...Interrupt ID115: 0x00020CCC
Offset Formula: 0x00020B00+4*i: where i (29-115) represents Interrupt ID

Bit	Field	Type/InitVal	Description
8	FIQ Mask i CPU0	RW 0x0	CPU0 FIQ Mask bit interrupt i When this field is cleared, interrupt i is masked to the CPU When this field is set, interrupt i is enabled to the CPU.. The mask affects the assertion of FIQ pin of the CPU. It does not affect the setting of bits in the cause register. The associated CPU can update this field by writing to its private IMS and ICM registers as follows: Writing i to Interrupt_ID field and 0x1 to selFIQ in the ISM sets this field Writing i to Interrupt_ID field and 0x1 to selFIQ in the ICM clears this field Notice that this mechanism provides the support for a CPU to handle FIQ interrupts without CPU identification relativity.
7:4	Reserved	RSVD 0x0	Reserved
3	IRQ Maski CPU3	RW 0x0	CPU3 IRQ Mask i See Description at IRQ Maski CPU0
2	IRQ Maski CPU2	RW 0x0	CPU2 IRQ Mask i See Description at IRQ Maski CPU0
1	IRQ Maski CPU1	RW 0x0	CPU1 IRQ Mask i See Description at IRQ Maski CPU0
0	IRQ Mask i CPU0	RW 0x0	CPU0 IRQ Mask bit interrupt i When this field is cleared, interrupt i is masked to the CPU When this field is set, interrupt i is enabled to the CPU.. The mask affects the assertion of IRQ pin of the CPU. It does not affect the setting of bits in the cause register. The associated CPU can update this field by writing to its private IMS and ICM registers as follows: Writing i to Interrupt_ID field and 0x0 to selFIQ in the ISM sets this field Writing i to Interrupt_ID field and 0x0 to selFIQ in the ICM clears this field Notice that this mechanism provides the support for a CPU to handle IRQ interrupts without CPU identification relativity.

A.2.14 Per CPU Interrupt Controller Control and Configuration

Table 303: Outbound Doorbell Register (N=0–3)

Offset: CPU ID0: 0x00021870, CPU ID1: 0x00021970, CPU ID2: 0x00021A70, CPU ID3: 0x00021B70

Offset Formula: 0x00021870+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	OutboundIntCs	RW 0x0	Outbound Interrupt Cause When a bit in this field is set and the corresponding bit in <OutboundIntCsMask> in the Outbound Doorbell Mask Register is also set, bit <OutboundDoorbell> is set in the Main Interrupt Cause Register. A CPU write of 1 set the bits in this field. A CPU write of 0 has no effect. A target write of 0 clear the bits in this field. A target write of 1 has no affect

Table 304: Outbound Doorbell Mask Register (N=0–3)

Offset: CPU ID0: 0x00021874, CPU ID1: 0x00021974, CPU ID2: 0x00021A74, CPU ID3: 0x00021B74

Offset Formula: 0x00021874+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	OutboundIntCsMask	RW 0x0	Outbound Interrupt Cause Mask Mask bit per each cause bit in <OutboundIntCs> field in the Outbound Doorbell Register. The mask only affects the assertion of the interrupt bit in the Main Interrupt Cause Register. It does not affect the setting of bits in the Outbound Doorbell Register.

Table 305: Inbound Doorbell Register (N=0–3)

Offset: CPU ID0: 0x00021878, CPU ID1: 0x00021978, CPU ID2: 0x00021A78, CPU ID3: 0x00021B78

Offset Formula: 0x00021878+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	InboundIntCs	RW 0x0	Inbound Command Cause When this bit is not zero and the corresponding bit <InboundIntCsMask> in the Inbound Doorbell Mask Register is set, bit <InboundDoorbellSumLow> is set in the Main Interrupt Cause Register for the lowest 16 doorbells and <InboundDoorbellSumHigh> is set for the upper 16 doorbells Setting any non-masked doorbell cause bit is performed using the following two mechanisms: * An initiator direct write of 1 sets the bits in this field. An initiator write of 0 has no effect. * Other agent in the writes to the Software Triggered Interrupt Register, targeting current CPU with specified Msg ID A CPU write of 0 clear the bits in this field. A CPU write of 1 has no effect.

Table 306: Inbound Doorbell Mask Register (N=0–3)

Offset: CPU ID0: 0x0002187C, CPU ID1: 0x0002197C, CPU ID2: 0x00021A7C, CPU ID3: 0x00021B7C

Offset Formula: 0x0002187C+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	InboundIntCsMask	RW 0x0	Inbound Interrupt Cause Mask Mask bit per each cause bit in the <InboundIntCs> field in the Inbound Doorbell Register. The mask only affects the assertion of interrupt bit in Main Interrupt Cause Register. It does not affect the setting of bits in the Inbound Doorbell Register.

Table 307: Main Interrupt Cause (Vector 0) Register (N=0–3)

Offset: CPU ID0: 0x00021880, CPU ID1: 0x00021980, CPU ID2: 0x00021A80, CPU ID3: 0x00021B80

Offset Formula: 0x00021880+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:29	Reserved	RO 0x0	
28:0	INT[n]	RO 0x0	Interrupt Line n (n=0..28)

Table 308: Main Interrupt Cause (Vector 1) Register (N=0–3)

Offset: CPU ID0: 0x00021884, CPU ID1: 0x00021984, CPU ID2: 0x00021A84, CPU ID3: 0x00021B84

Offset Formula: 0x00021884+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:29	Reserved_31_29	RO 0x0	Reserved
28:0	INT[n]	RO 0x0	Interrupt line n (n=29..57)

Table 309: Main Interrupt Cause (Vector 2) Register (N=0–3)

Offset: CPU ID0: 0x00021888, CPU ID1: 0x00021988, CPU ID2: 0x00021A88, CPU ID3: 0x00021B88

Offset Formula: 0x00021888+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:29	Reserved_31_29	RO 0x0	
28:0	INT[n]	RO 0x0	Interrupt Line n (n=58..86)

Copyright © 2014 Marvell

Table 310: Main Interrupt Cause (Vector 3) Register (N=0–3)

Offset: CPU ID0: 0x0002188C, CPU ID1: 0x0002198C, CPU ID2: 0x00021A8C, CPU ID3: 0x00021B8C

Offset Formula: 0x0002188C+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:29	Reserved_31_29	RO 0x0	
28:0	INT[n]	RO 0x0	Interrupt Line n (n=87..115)

Table 311: IRQ Select Cause Register (N=0–3)

Offset: CPU ID0: 0x000218A0, CPU ID1: 0x000219A0, CPU ID2: 0x00021AA0, CPU ID3: 0x00021BA0

Offset Formula: 0x000218A0+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31	Stat	RO 0x0	Indicates if there are valid interrupts (interrupt becomes a valid when its cause and mask bits are set) in one of the cause registers. 0 = No valid interrupts: No active non-masked interrupts in all Interrupt Cause registers. 1 = There are active interrupts: There are active non-masked in some of the Interrupt Cause registers.
30:29	VecSel	RO 0x3	Vector Select Indicates if bits[28:0] are a shadow Main Cause Vec0, Vec1, Vec2 or Vec3 after masking 0 = Vec0: Shadow of Cause register Vec 0 1 = Vec1: Shadow of Cause register Vec 1 2 = Vec2: Shadow of Cause register Vec 2 3 = Vec3: Shadow of Cause register Vec 3
28:0	Cause	RO 0x0	A shadow register of the vector0,1,2 or 3 after the masking operation (valid interrupts only) Return the masked Cause Register Vec 0 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 1 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 2 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 3

Table 312: FIQ Select Cause Register (N=0–3)

Offset: CPU ID0: 0x000218A4, CPU ID1: 0x000219A4, CPU ID2: 0x00021AA4, CPU ID3: 0x00021BA4

Offset Formula: 0x000218A4+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31	Stat	RO 0x0	Indicates if there are valid interrupts (interrupt becomes a valid when its cause and mask bits are set) in one of the cause registers. 0 = No valid interrupts: No active non-masked interrupts in all Interrupt Cause registers. 1 = There are active interrupts: There are active non-masked in some of the Interrupt Cause registers.
30:29	VecSel	RO 0x3	Vector Select Indicates if bits[28:0] are a shadow Main Cause Vec0, Vec1, Vec2 or Vec3 after masking 0 = Vec0: Shadow of Cause register Vec 0 1 = Vec1: Shadow of Cause register Vec 1 2 = Vec2: Shadow of Cause register Vec 2 3 = Vec3: Shadow of Cause register Vec 3
28:0	Cause	RO 0x0	A shadow register of the vector0,1,2 or 3 after the masking operation Return the masked Cause Register Vec 0 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 1 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 2 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 3

Table 313: Current Task Priority (CTP) Register (N=0–3)

Offset: CPU ID0: 0x000218B0, CPU ID1: 0x000219B0, CPU ID2: 0x00021AB0, CPU ID3: 0x00021BB0

Offset Formula: 0x000218B0+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:28	Task Priority	RW 0x0	Current Task Priority Indicates the priority level that pending interrupt priorities must exceed for the interrupt request to be served by CPU Core ’b0000 Lowest priority. All interrupts except those whose priority are 0 can be serviced ’b1111 Highest priority. No interrupts are signaled to the CPU Core else... receive valid interrupt (non-masked) with highest priority. Its software responsibility to update the running priority task
27:0	Reserved	RSVD 0x0	Reserved

Table 314: IRQ Interrupt Acknowledge (IIACK) Register (N=0–3)

Offset: CPU ID0: 0x000218B4, CPU ID1: 0x000219B4, CPU ID2: 0x00021AB4, CPU ID3: 0x00021BB4

Offset Formula: 0x000218B4+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
The Interrupt Acknowledge Register is a read-only register. The Interrupt Acknowledge Register is used to determine the source of the interrupt request received by an CPU. If no interrupt is Pending then the Interrupt ID returned is 1023			
31:16	InboundDoorbell_lo	RO 0x0	Returns shadow of the CPU Inbound Doorbell bits [15:0] after masking. Valid only when the interrupt ID specifies Doorbell In interrupt.
15:10	Reserved	RO 0x0	
9:0	Interrupt ID	RO 0x3FF	Interrupt ID Determine the source of the interrupt request received and selected by the interrupt controller If no interrupt is Pending then the Interrupt ID returned is 1023 (spurious interrupt)

Table 315: Interrupt Set Mask (ISM) Register (N=0–3)

Offset: CPU ID0: 0x000218B8, CPU ID1: 0x000219B8, CPU ID2: 0x00021AB8, CPU ID3: 0x00021BB8

Offset Formula: 0x000218B8+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31	SetFIQ	WO 0x0	Select FIQ 0 = Set IRQ Mask: Mask IRQ of the requesting CPU 1 = Set FIQ Mask: Mask FIQ of the requesting CPU
30:10	Reserved	RSVD 0x0	Reserved
9:0	Interrupt ID	WO 0x0	Interrupt ID Writing the Interrupt ID to this field masks only the IRQ/FIQ of the accessing CPU in the Interrupt Source Control Register which associated with the Interrupt ID This means that this interrupt is now disabled for the CPU.

Table 316: Interrupt Clear Mask (ICM) Register (N=0–3)

Offset: CPU ID0: 0x000218BC, CPU ID1: 0x000219BC, CPU ID2: 0x00021ABC, CPU ID3: 0x00021BBC

Offset Formula: 0x000218BC+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31	Clear FIQ	WO 0x0	Select FIQ 0 = Clr IRQ Mask: Unmasks IRQ of the requesting CPU 1 = Clr FIQ Mask: Unmasks FIQ of the requesting CPU
30:10	Reserved	RSVD 0x0	Reserved
9:0	Interrupt ID	WO 0x0	Interrupt ID Writing the Interrupt ID to this field unmasks only the IRQ/FIQ the accessing CPU in the Interrupt Source Control Register which associated with the Interrupt ID This means that this interrupt is now enabled for the CPU.

Table 317: SOC Main Interrupt Error Mask Register (N=0–3)

Offset: CPU ID0: 0x000218C0, CPU ID1: 0x000219C0, CPU ID2: 0x00021AC0, CPU ID3: 0x00021BC0

Offset Formula: 0x000218C0+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	SOC Error Interrupt Mask. Mask bit per SOC Main Interrupt Error event. If bit is set to 0, interrupt is masked; if set to 1 interrupt is enabled. The mask only affects the assertion of the SOCErrSum interrupt line in the associated CPU main cause registers.

Table 318: Coherency Fabric Local Interrupt Mask Register (N=0–3)

Offset: CPU ID0: 0x000218C4, CPU ID1: 0x000219C4, CPU ID2: 0x00021AC4, CPU ID3: 0x00021BC4

Offset Formula: 0x000218C4+0x100*N: where N (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	Coherency Fabric Local Interrupt Mask. Mask bit per Subsystem Interrupt event. If bit is set to 0, interrupt is masked; if set to 1 interrupt is enabled. The mask only affects the assertion of the FabricLocalSum interrupt line in the associated CPU main cause registers.

A.2.15 End Point Interrupt Control

Table 319: Main Interrupt Cause (Vec 0) Register

Offset: 0x00020900

Bit	Field	Type/InitVal	Description
31:29	Reserved	RO 0x0	
28:0	INT[n]	RO 0x0	Interrupt Line n (n=0..28)

Table 320: Main Interrupt Cause (Vec 1) Register

Offset: 0x00020904

Bit	Field	Type/InitVal	Description
31:29	Reserved_31_29	RO 0x0	Reserved

Table 320: Main Interrupt Cause (Vec 1) Register (Continued)

Offset: 0x00020904

Bit	Field	Type/InitVal	Description
28:0	INT[n]	RO 0x0	Interrupt line n (n=29..57)

Table 321: Main Interrupt Cause (Vec 2) Register

Offset: 0x00020908

Bit	Field	Type/InitVal	Description
31:29	Reserved_31_29	RO 0x0	
28:0	INT[n]	RO 0x0	Interrupt Line n (n=58..86)

Table 322: Main Interrupt Cause (Vec 3) Register

Offset: 0x0002090C

Bit	Field	Type/InitVal	Description
31:29	Reserved_31_29	RO 0x0	
28:0	INT[n]	RO 0x0	Interrupt Line n (n=87..115)

Table 323: Endpoint Sel Cause Register

Offset: 0x00020928

Bit	Field	Type/InitVal	Description
31	Stat	RO 0x0	Indicates if there are valid interrupts (interrupt becomes a valid when its cause and mask bits are set) in one of the cause vectors. 0 = No valid interrupts: No active non-masked interrupts in all Interrupt Cause registers. 1 = There are active interrupts: There are active non-masked in some of the Interrupt Cause registers.
30:29	VecSel	RO 0x3	Vector Select Indicates if bits[28:0] are a shadow Main Cause Vec0, Vec1, Vec2 or Vec3 after masking 0 = Vec0: Shadow of Cause register Vec 0 1 = Vec1: Shadow of Cause register Vec 1 2 = Vec2: Shadow of Cause register Vec 2 3 = Vec3: Shadow of Cause register Vec 3

Table 323: Endpoint Sel Cause Register (Continued)
Offset: 0x00020928

Bit	Field	Type/InitVal	Description
28:0	Cause	RO 0x0	A shadow register of the vector0,1,2 or 3 after the masking operation (valid interrupts) Return the EP masked Cause Register Vec 0 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 1 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 2 If any interrupt is still set after masking in this vector, else return masked Cause Register Vec 3

Table 324: EP Coherency Fabric Local Interrupt Mask Register
Offset: 0x00020940

Bit	Field	Type/InitVal	Description
31:0	EPFabricLocaMask	RW 0x0	Endpoint Coherency Fabric Local Interrupt Mask Mask bit per Coherency Fabric Local Interrupt event. If bit is set to 0, interrupt is masked; if set to 1 interrupt is enabled. The mask only affects the assertion of the associated interrupt line to the Endpoint main cause register/

Table 325: EP SOC Main Interrupt Error Mask Register
Offset: 0x00020944

Bit	Field	Type/InitVal	Description
31:0	EPSocErrorMask	RW 0x0	SOC Error Interrupt Mask. Mask bit per SOC Main Interrupt Error event. If bit is set to 0, interrupt is masked; if set to 1 interrupt is enabled. The mask only affects the assertion of the SOCErrSum interrupt line Endpoint main interrupt cause registers.

A.2.16 Shared Inbound Doorbell

Table 326: Shared Inbound Doorbell0 Register

Offset: 0x00020400

Bit	Field	Type/InitVal	Description
31:0	InboundIntCs0	RW 0x0	Inbound Command Cause When this bit is not zero, bit <SharedInboundDoorbellSum> is set in the Main Interrupt Cause Register. Setting any cause bit is performed using the following two mechanisms: * A MBUS Master direct write of 1 sets the bits in this field. A MBUS Master write of 0 has no effect. * Any write to the Software Triggered Interrupt Register, targeting current Inbound Doorbell <Doorbell Sel>, set the associated cause bit <IntID> Any CPU write of 0 clear the bits in this field. A CPU write of 1 has no effect.

Table 327: Shared Inbound Doorbell1 Register

Offset: 0x00020404

Bit	Field	Type/InitVal	Description
31:0	InboundIntCs1	RW 0x0	See Shared Inbound Doorbell0 Register

Table 328: Shared Inbound Doorbell2 Register

Offset: 0x00020408

Bit	Field	Type/InitVal	Description
31:0	InboundIntCs2	RW 0x0	See Shared Inbound Doorbell0 Register

A.2.17 Global Timers

Table 329: Timers Control Register

Offset: 0x00020300

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved
30:28	Global Timer3 Ratio	RW 0x0	Global Timer 3 Ratio See definition and encoding at <GlobalWDTimerRatio>

Table 329: Timers Control Register (Continued)

Offset: 0x00020300

Bit	Field	Type/InitVal	Description
27:25	Global Timer2 Ratio	RW 0x0	Global Timer 2 Ratio See definition and encoding at <GlobalWDTimerRatio>
24:22	Global Timer1 Ratio	RW 0x0	Global Timer 1 Ratio See definition and encoding at <GlobalWDTimerRatio>
21:19	Global Timer0 Ratio	RW 0x0	Global Timer 0 Ratio See definition and encoding at <GlobalWDTimerRatio>
18:16	GlobalWDTimerRati o	RW 0x0	Global Watchdog Timer Ratio. The field defines the Global WD Timer interval. Notate the number in this field as R. When activated, the timer will be incremented upon every (2 ^ R) source clocks. When <GlobalWDTimer25mhz> is enabled, this field is ignored. 0 = 1: Timer tic occurs every source clock 1 = 2: Timer tic occurs every 2 source clocks 2 = 4: Timer tic occurs every 4 source clocks 3 = 8: Timer tic occurs every 8 source clocks 4 = 16: Timer tic occurs every 16 source clocks 5 = 32: Timer tic occurs every 32 source clocks 6 = 64: Timer tic occurs every 64 source clocks 7 = 128: Timer tic occurs every 128 source clocks
15	Reserved	RSVD 0x0	Reserved
14	GlobalTimer3 25MhzEn	RW 0x0	Global Timer3 25Mhz frequency Enable See description at <GlobalWDTimer25MhzEn> 0 = False 1 = True
13	GlobalTimer2 25MhzEn	RW 0x0	Global Timer2 25Mhz frequency Enable See description at <GlobalWDTimer25MhzEn> 0 = False 1 = True
12	GlobalTimer1 25MhzEn	RW 0x0	Global Timer1 25Mhz frequency Enable See description at <GlobalWDTimer25MhzEn> 0 = False 1 = True
11	GlobalTimer0 25MhzEn	RW 0x0	Global Timer0 25Mhz frequency Enable See description at <GlobalWDTimer25MhzEn> 0 = False 1 = True

Table 329: Timers Control Register (Continued)

Offset: 0x00020300

Bit	Field	Type/InitVal	Description
10	GlobalWDTimer25MhzEn	RW 0x0	Global Watchdog Timer 25Mhz frequency Enable Defines the count frequency of the global watchdog timer. 0 = False: Global WD timer count frequency is defined by the source clock according to <GlobalWDTimerRatio> field 1 = True: Global WD timer count frequency is 25Mhz
9	GlobalWDTimerAuto	RW 0x0	Global Watchdog Timer Auto Mode When this bit is clear to 0 and <GlobalWDTimer> in the CPU Watchdog Timer Register has reached zero, <GlobalWDTimer> stops counting. When this bit is set to 1 and <GlobalWDTimer> has reached zero, <GlobalTimer0Rel> in the GlobalTimer0 Reload Register is reloaded to <GlobalWDTimer>, it then continues to count.
8	GlobalWDTimerEn	RW 0x0	Global Watchdog Timer Enable 0 = False 1 = True
7	GlobalTimer3Auto	RW 0x0	Global Timer 3 Auto Mode When this bit is clear to 0 and <GlobalTimer3> in the Global Timer2 Register has reached to zero, <GlobalTimer3> stops counting. When this bit is set to 1 and <GlobalTimer3> has reached zero, <GlobalTimer3Rel> in the Global Timer3 Reload Register is reloaded to <GlobalTimer3>, it then continues to count.
6	GlobalTimer3En	RW 0x0	CPU Timer 3 Enable 0 = False 1 = True
5	GlobalTimer2Auto	RW 0x0	Global Timer 2 Auto Mode When this bit is clear to 0 and <GlobalTimer2> in the Global Timer2 Register has reached to zero, <GlobalTimer2> stops counting. When this bit is set to 1 and <GlobalTimer2> has reached zero, <GlobalTimer2Rel> in the Global Timer1 Reload Register is reloaded to <GlobalTimer2>, it then continues to count.
4	GlobalTimer2En	RW 0x0	CPU Timer 2 Enable 0 = False 1 = True

Copyright © 2014 Marvell

Table 329: Timers Control Register (Continued)

Offset: 0x00020300

Bit	Field	Type/InitVal	Description
3	GlobalTimer1Auto	RW 0x0	Global Timer 1 Auto Mode When this bit is clear to 0 and <GlobalTimer1> in the Global Timer1 Register has reached to zero, <GlobalTimer1> stops counting. When this bit is set to 1 and <GlobalTimer1> has reached zero, <GlobalTimer1Rel> in the Global Timer2 Reload Register is reloaded to <GlobalTimer1>, it then continues to count.
2	GlobalTimer1En	RW 0x0	CPU Timer 1 Enable 0 = False 1 = True
1	GlobalTimer0Auto	RW 0x0	Global Timer 0 Auto Mode When this bit is cleared to 0 and <GlobalTimer0> in the Global Timer0 Register has reached zero, <GlobalTimer0> stops counting. When this bit is set to 1 and <GlobalTimer0> has reached zero, <GlobalTimer0Rel> in the Global Timer0 Reload Register is reloaded to <GlobalTimer0>, it then continues to count.
0	GlobalTimer0En	RW 0x0	Global Timer 0 Enable 0 = False 1 = True

Table 330: Timers Events Status register Register

Offset: 0x00020304

Bit	Field	Type/InitVal	Description
31	WD timer expired	RW0C 0x0	Sticky indication for timer expiration
30:25	Reserved	RSVD 0x0	Reserved
24	Timer3 expired	RW0C 0x0	Sticky indication for timer expiration.
23:17	Reserved	RSVD 0x0	Reserved
16	Timer2 expired	RW0C 0x0	Sticky indication for timer expiration.
15:9	Reserved	RSVD 0x0	Reserved
8	Timer1 expired	RW0C 0x0	Sticky indication for timer expiration.

Table 330: Timers Events Status register Register (Continued)
Offset: 0x00020304

Bit	Field	Type/InitVal	Description
7:1	Reserved	RSVD 0x0	Reserved
0	Timer0 expired	RW0C 0x0	Sticky indication for timer expiration.

Table 331: Timer0 Reload Register
Offset: 0x00020310

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer0Rel	RW 0x0	Global Timer 0 Reload This field contains the reload value of timer 0, it is used as the reload value in Periodic mode.

Table 332: Timer 0 Register
Offset: 0x00020314

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer0	RW 0x0	Global Timer 0 This 32-bit counter is decremented according to the <GlobalTimer0Ratio> When field <GlobalTimer0En> in the Global Timer Control Register = 0,<GlobalTimer0> stops. When <GlobalTimer0En> = 1 and <GlobalTimer0Auto> in the Global Timer Control Register = 0, <GlobalTimer0> is decremented until it reaches to 0, then it stops. When <GlobalTimer0En> = 1 and <GlobalTimer0Auto> = 1, <GlobalTimer0> is decremented until it reaches to 0, then the field <GlobalTimer0Rel> in the Global 0 Reload Register is reload to <GlobalTimer0> and then <GlobalTimer0> continues to count. When <GlobalTimer0> reaches 0, bit <GlobalTimer0IntReq> in the Coherency Fabric Local Cause Register is set to 1.

Table 333: Timer1 Reload Register
Offset: 0x00020318

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer1Rel	RW 0x0	Global Timer 1 Reload See the Global Timer0 Reload Register.

Table 334: Timer 1 Register
Offset: 0x0002031C

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer1	RW 0x0	Global Timer 1 See the Global Timer 0 Register.

Table 335: Timer2 Reload Register
Offset: 0x00020320

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer2Rel	RW 0x0	Global Timer 2 Reload See the Global Timer 0 Reload Register

Table 336: Timer 2 Register
Offset: 0x00020324

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer2	RW 0x0	Global Timer 2 See the Global Timer 0 Register

Table 337: Timer3 Reload Register
Offset: 0x00020328

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer3Rel	RW 0x0	Global Timer 3 Reload See the Global Timer0 Reload Register.

Table 338: Timer 3 Register
Offset: 0x0002032C

Bit	Field	Type/InitVal	Description
31:0	GlobalTimer3	RW 0x0	Global Timer 3 See the Global Timer 0 Register.

Table 339: Watchdog Timer Reload Register

Offset: 0x00020330

Bit	Field	Type/InitVal	Description
31:0	GlobalWDTimerLen	RW 0x0	Global WD Timer Reload See the Global Timer0 Reload Register.

Table 340: Watchdog Timer Register

Offset: 0x00020334

Bit	Field	Type/InitVal	Description
31:0	GlobalWDTimer	RW 0x7FFFFFFF	Global Watchdog Timer See the Global Timer 0 Register.

Table 341: ARM Generic Timer Control Register

Offset: 0x00020340

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	GenericTimerHaltOnDbg	RW 0x0	ARM Generic Timer Halt On Debug 0 = False: Debug signal into the Counter has no effect 1 = True: Debug signal into the Counter halts the Count
0	GenericTimerEnable	RW 0x0	ARM Generic Timer Enable 0 = False: ARM Generic timer is not incrementing 1 = True: ARM Generic timer is incrementing

Table 342: ARM Generic Timer Status Register

Offset: 0x00020344

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	DebugHalted	RO 0x0	ARM Generic Timer Debug Halted
0	Reserved	RSVD 0x0	Reserved

Table 343: ARM Generic Timer Low Register
Offset: 0x00020348

Bit	Field	Type/InitVal	Description
31:0	GenericTimerLow	RW 0x0	ARM Generic Timer Low Bits[31:0] of the Generic Timer

Table 344: ARM Generic Timer High Register
Offset: 0x0002034C

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved
23:0	GenericTimerHigh	RW 0x0	ARM Genric Timer High Bits[55:32] of the Generic Timer

A.2.18 Semaphores

Table 345: Semaphore0 Register
Offset: 0x00020500

Bit	Field	Type/InitVal	Description
31:27	Reserved	RO 0x0	Reserved
26:24	Semaphore 3	RO 0x0	See description at Semaphore 0 field[2:0]
23:19	Reserved	RO 0x0	Reserved
18:16	Semaphore 2	RO 0x0	See description at Semaphore 0 field[2:0]
15:11	Reserved	RO 0x0	Reserved
10:8	Semaphore 1	RO 0x0	See description at Semaphore 0 field[2:0]
7:3	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 345: Semaphore0 Register (Continued)

Offset: 0x00020500

Bit	Field	Type/InitVal	Description
2:0	Semaphore 0	RO 0x0	Semaphore 0 A semphor can be locked by a CPU or an external IO device. The first one to read it before it is locked, receives its ID value - CPU ID or 0x4 for an external IO device. Once locked by one of the agents, a read by any of the other agents will return the owner ID. Unlock the semaphore by writing a value of 0xFF. A write of any other value is ignored.

Table 346: Semaphore%<n> Register (n=1–127)

Offset: Semaphore1: 0x00020504, Semaphore2: 0x00020508...Semaphore127: 0x000206FC
Offset Formula: 0x00020504+(n - 1)*4 : where n (1-127) represents Semaphore

Bit	Field	Type/InitVal	Description
31:27	Reserved	RO 0x0	Reserved
26:24	Semaphore 3	RO 0x0	See description at Semaphor0 register
23:19	Reserved	RO 0x0	Reserved
18:16	Semaphore 2	RO 0x0	See description at Semaphor0 register
15:11	Reserved	RO 0x0	Reserved
10:8	Semaphore 1	RO 0x0	See description at Semaphor0 register
7:3	Reserved	RO 0x0	Reserved
2:0	Semaphore 0	RO 0x0	See description at Semaphor0 register

A.2.19 Clocks Resets and Power Management

Table 347: WD RSTOUTn Mask Register

Offset: 0x00020704

Bit	Field	Type/InitVal	Description
31:20	Reserved	RSVD 0x0	Reserved

Table 347: WD RSTOUTn Mask Register (Continued)
Offset: 0x00020704

Bit	Field	Type/InitVal	Description
19:16	CPU WD Group	RW 0x0	This field defines the CPUs that participate in the watchdog group. For each CPU<i>, when bit[i] in this field is set, the CPU is in the WD group. When the WD group is not empty (i.e. this field is NOT 0x0), at the event that all WD timers of all CPUs in the group have expired, a WD Reset will be triggered. Notice that in addition to this mechanism, Every CPU has its own WD Reset Mask that when cleared , enables it to trigger the WD Reset when its own timer has expired.
15:9	Reserved	RSVD 0x0	Reserved
8	Global WD mask	RW 0x0	When this bit is set, CPU subsystem will assert WD RSTOUTn upon global WD timer expiration.
7:4	Reserved	RSVD 0x0	Reserved
3	CPU3 WD mask	RW 0x0	When this bit is set, CPU subsystem will assert WD RSTOUTn upon CPU3 WD timer expiration.
2	CPU2 WD mask	RW 0x0	When this bit is set, CPU subsystem will assert WD RSTOUTn upon CPU2 WD timer expiration.
1	CPU1 WD mask	RW 0x0	When this bit is set, CPU subsystem will assert WD RSTOUTn upon CPU1 WD timer expiration.
0	CPU0 WD mask	RW 0x0	When this bit is set, CPU subsystem will assert WD RSTOUTn upon CPU0 WD timer expiration.

Table 348: WD Interrupt Mask Register
Offset: 0x00020708

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:12	CPU3 Local WD Interrupt Mask	RW 0x8	CPU3 WD interuupt Mask When bit[i] of this field is set, a CPU[i] WD timer expiration will assert the ARM_WD_INTR of CPU3

Table 348: WD Interrupt Mask Register (Continued)

Offset: 0x00020708

Bit	Field	Type/InitVal	Description
11:8	CPU2 Local WD Interrupt Mask	RW 0x4	CPU2 WD interuupt Mask When bit[i] of this field is set, a CPU[i] WD timer expiration will assert the ARM_WD_INTR of CPU2
7:4	CPU1 Local WD Interrupt Mask	RW 0x2	CPU1 WD interuupt Mask When bit[i] of this field is set, a CPU[i] WD timer expiration will assert the ARM_WD_INTR of CPU1
3:0	CPU0 Local WD Interrupt Mask	RW 0x1	CPU0 WD interuupt Mask When bit[i] of this field is set, a CPU[i] WD timer expiration will assert the ARM_WD_INTR of CPU0

Table 349: CPUn SW Reset Control Register (n=0–3)

Offset: CPU ID0: 0x00020800, CPU ID1: 0x00020808, CPU ID2: 0x00020810, CPU ID3: 0x00020818
Offset Formula: 0x00020800+8*n: where n (0-3) represents CPU ID

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x80	Reserved
0	CPU SW RESET	RW <min(%n,1)>	When this bit is set to 1, CPU will be held at reset. The CPU components held at reset are the CPU core, CPU sETM and CPU debug interface. 0 = Disable: CPU is not held at reset. 1 = Enable: CPU is held at reset.

Table 350: General Purpose SW usage 0 Register

Offset: 0x00020980

Bit	Field	Type/InitVal	Description
31:0	GP SW 0	RW 0x0	General Purpose Read Write register for SW usage

Table 351: General Purpose SW usage 1 Register
Offset: 0x00020984

Bit	Field	Type/InitVal	Description
31:0	GP SW 1	RW 0x0	General Purpose Read Write register for SW usage

Table 352: General Purpose SW usage 2 Register
Offset: 0x00020988

Bit	Field	Type/InitVal	Description
31:0	GP SW 2	RW 0xffffffff	General Purpose Read Write register for SW usage

Table 353: General Purpose SW usage 3 Register
Offset: 0x0002098C

Bit	Field	Type/InitVal	Description
31:0	GP SW 3	RW 0xffffffff	General Purpose Read Write register for SW usage

A.2.20 AVS Control and Status

Table 354: AVS Control 0 Register
Offset: 0x00020860

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x40380	Reserved
7:4	AvsHighVDDLimit	RW SAR	AVS VDD High Limit 0 = 0.9125v 1 = 0.925v 2 = 0.9375v 3 = 0.95v 4 = 0.9625v 5 = 0.975v 6 = 0.9875v 7 = 1v 8 = 1.0125v 9 = 1.025v 10 = 1.0375v 11 = 1.05v 12 = 1.0625v 13 = 1.075v 14 = 1.0875v 15 = 1.1v

Table 354: AVS Control 0 Register (Continued)
Offset: 0x00020860

Bit	Field	Type/InitVal	Description
3:0	AvsLowVDDLimit	RW 0xF	AVS VDD Low Limit 5 = 0.7875v 6 = 0.8v 7 = 0.8125v 8 = 0.825v 9 = 0.8375v 10 = 0.85v 11 = 0.8625v 12 = 0.875v 13 = 0.8875v 14 = 0.9v 15 = 0.9125v

Table 355: AVS Control 2 Register
Offset: 0x00020868

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24	AvsSwRst	RW 0x1	Clearing this bit asserts port rst_b of MISL_0040G_AVS.
23:10	Reserved	RSVD 0x88	Reserved
9	AvsEnable	RW SAR	AVS enable control 0 = OFF: AVS is off, vddfb=vdd. 1 = ON: AVS is on.
8:0	Reserved	RSVD 0x0	Reserved

A.2.21 Arbitration Control

Table 356: CFU BW Control Limits Register
Offset: 0x00020440

Bit	Field	Type/InitVal	Description
31:28	PendingIOJamLimit	RW 0x4	Jammed Period Pending IO Coherent Request Limit Defines the number-1 of coherent requests that Fabric will accept in to its queue from IO units when <OutstndJamTreshold> is exceeded. When it is not exceeded , this field is ignored.

Table 356: CFU BW Control Limits Register (Continued)

Offset: 0x00020440

Bit	Field	Type/InitVal	Description
27:24	PendingCPUJamLimit	RW 0x2	Jammed Period Pending CPU Read Request Limit Defines the number-1 of read requests that Fabric accepts into its queue from each CPU when <OutstndJamTreshold> is exceeded. When it is not exceeded , this field is ignored.
23:20	PendingIOLimit	RW 0x4	Pending Outstanding IO Coherent Request Limit Defines the number-1 of coherent requests that Fabric accepts into its Queue from IO units. When <PendingJamTreshold> is exceeded, this field is ignored and limit is defined by <PendingIOJamLimit>.
19:16	PendingCPULimit	RW 0x4	Pending CPU Read Request Limit Defines the number-1 of read requests that Fabric will accepts into its Queue from each CPU. When <PendingJamTreshold> is exceeded, this field is ignored and the limit is defined by <PendingCPUJamLimit>.
15:12	OutstndIOJamLimit	RW 0x8	Jammed Period Outstanding IO Coherent Request Limit Defines the number-1 of coherent requests that Fabric accepts from IO units when <OutstndJamTreshold> is exceeded. When it is not exceeded , this field is ignored.
11:8	OutstndCPUJamLimit	RW 0x4	Jammed Period Outstanding CPU Read Request Limit Defines the number-1 of read requests that Fabric accepts from each CPU when <OutstndJamTreshold> is exceeded. When it is not exceeded , this field is ignored.
7:4	OutstndIOLimit	RW 0xb	Outstanding IO Coherent Request Limit Defines the number-1 of coherent requests that Fabric accepts from IO units. When <OutstndJamTreshold> is exceeded, this field is ignored and limit is defined by <OutstndIOJamLimit>.
3:0	OutstndCPULimit	RW 0x8	Outstanding CPU Read Request Limit Defines the number-1 of read requests that Fabric accepts from each CPU. When <OutstndJamTreshold> is exceeded, this field is ignored and limit is defined by <OutstndCPUJamLimit>.

Table 357: CFU BW Control Threshold Register

Offset: 0x00020444

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:4	PendingJamTreshold	RW 0x8	Pending Jammed Threshold When the number of requests in fabric queue exceeds the value in this field, Fabric BW control limits acceptance of further requests according to <PendingCPUJamLimit> and <PendingIOJamLimit>.
3:0	OutstndJamTreshold	RW 0xb	Outstanding Jammed Treshhold When the number of outstanding requests exceeds the value in this field, Fabric BW control limits acceptance of further requests according to <OutstndCPUJamLimit> and <OutstndIOJamLimit>

Table 358: Arbitration Queue Load Control Register

Offset: 0x00020448

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:28	CFU WRQ Full Prio	RW 0x0	CFU Write Queue Full Priority
27:24	CFU WRQ Prio Upgrade Treshold	RW 0xF	CFU Write Queue Priority upgrade Threshold When the number of occupied entries CFU Write Queue exceeds the value in this field, Write requests towards DDR will be served with a prio that is at least the value configured in <CFU_WRQ_Full_Prio>
23:22	L2 ARB Age step	RW 0x0	L2 Arbitration Age Step
21:20	CFU RDQ Full Prio	RW 0x0	CFU Read Queue Full Priority
19:16	CFU RDQ Prio Upgrade Treshold	RW 0xF	CFU Read Queue Priority upgrade Threshold When the number of occupied entries in CFU Read Queue exceeds the value in this field, Read requests issued towards the DDR will be served with a priority that is at least the value configured in <CFU_RDQ_Full_Prio>

Table 358: Arbitration Queue Load Control Register (Continued)

Offset: 0x00020448

Bit	Field	Type/InitVal	Description
15:14	CFU EV Arb Age Step	RW 0x0	CFU Write Eviction Arbitration age step
13:12	L2 EVB Full Prio	RW 0x0	L2 Eviction Buffer Full Priority
11:8	L2 EVB Prio Upgrade Treshold	RW 0xF	L2 Eviction Buffer Priority upgrade Threshold When the number of occupied entries in L2 Eviction Buffer exceeds the value in this field, L2 eviction will be served with a prio that is at least the value configured in <L2_EVB_Full_Prio>
7:6	CFU Write Arb Age Step	RW 0x0	CFU Write arbiter age step
5:4	CIB CMDQ Full Prio	RW 0x0	CIB Command Queue Full Priority
3:0	CIB CMDQ Prio Upgrade Treshold	RW 0xF	CIB Command Queue Priority upgrade Threshold When the number of occupied entries in CIB CMDQ exceeds the value in this field, CIB coherent requests will be served with a prio that is at least the value configured in <CIB_CMDQ_Full_Prio>

A.2.22 CoreSight Control

Table 359: Coresight Components Base Address Register

Offset: 0x00020D00

Bit	Field	Type/InitVal	Description
31:12	CoresightComponentBAR	RW 0x42300	Coresight Components BAR as registered within ROM table. Register initial value points to ROM Table base address read from the processor. List of BAR for 4xCPU devices: 0x4230_0000ROM Table 0x4230_1000CPU0 Debug 0x4230_2000CPU0 sETM 0x4230_3000CPU0 CTI 0x4230_4000CPU0 PMU 0x4230_5000ETB0 0x4230_6000CTI1 (ETB0 + ETB1 triggers) 0x4230_7000CPU1 Debug 0x4230_8000CPU1 sETM 0x4230_9000CPU1 CTI 0x4230_A000CPU1 PMU 0x4230_B000ETB1 0x4230_C000CPU2 Debug 0x4230_D000CPU2 sETM 0x4230_E000CPU2 CTI 0x4230_F000CPU2 PMU 0x4231_0000ETB2 0x4231_1000CTI2 (ETB2 + ETB3 triggers) 0x4230_2000CPU3 Debug 0x4231_3000CPU3 sETM 0x4231_4000CPU3 CTI 0x4231_5000CPU3 PMU 0x4231_6000ETB3 0x4231_7000Timestamp 0x4231_8000TPIU Funnel 0x4231_9000TPIU 0x4231_A000sETM 0x4231_B000Lane0 sETM Phy 0x4231_C000Lane1 sETM Phy
11:0	Reserved	RSVD 0x0	Reserved

Table 360: Coresight CTI Mask Register
Offset: 0x00020D08

Bit	Field	Type/InitVal	Description
31:24	MaskCTI1TriggerOut	RW 0x0	Enabling CTI1 Trigger Out outputs according DBGEN state. By configuring any bit to 0b enabling Trigger Out outputs when DBGEN is enabled. By configuring any bit to 1b enabling Trigger Out outputs regardless DBGEN state.
23:16	MaskCTI0TriggerOut	RW 0x0	Enabling CTI0 Trigger Out outputs according DBGEN state. By configuring any bit to 0b enabling Trigger Out outputs when DBGEN is enabled. By configuring any bit to 1b enabling Trigger Out outputs regardless DBGEN state.
15:8	MaskCTI1TriggerIn	RW 0x0	Enabling CTI1 Trigger In inputs according NIDEN or DBGEN states. By configuring any bit to 0b enabling Trigger In inputs when NIDEN or DBGEN are enabled. By configuring any bit to 1b enabling Trigger In input regardless NIDEN & DBGEN state.
7:0	MaskCTI0TriggerIn	RW 0x0	Enabling CTI0 Trigger In inputs according NIDEN or DBGEN states. By configuring any bit to 0b enabling Trigger In inputs when NIDEN or DBGEN are enabled. By configuring any bit to 1b enabling Trigger In input regardless NIDEN & DBGEN state.

Table 361: Coresight Status Register
Offset: 0x00020D0C

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	sETM Trace Stall	RO 0x0	When the sETM is released from reset, there is a delay before it is ready to start transmitting data. This is indicated by the STALL signal. When the STALL signal falls, the transmitter is ready to accept trace data from any CoreSight trace sources. 0 = sETM Ready: sETM is ready to accept trace data and transmit it. 1 = sETM Stalled: sETM cannot transmit any trace data.

A.2.23 CoreSight Timestamp

Table 362: Timestamp Counter Enable Register
Offset: 0x00023000

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	Timestamp Counter Enable	RW 0x0	Enables or disables 48bit timestamp gray counter. 0 = Disable 1 = Enable

Table 363: Timestamp Counter Preload 1 Register
Offset: 0x00023004

Bit	Field	Type/InitVal	Description
31:0	Timestamp Preload value	RW 0x0	Preload the [31:0] bits timestamp counter.

Table 364: Timestamp Counter Preload 2 Register
Offset: 0x00023008

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Timestamp preload value	RW 0x0	Preload the [47:32] bit timestamp counter.

Table 365: Timestamp Claim Tag Set Register
Offset: 0x00023FA0

Bit	Field	Type/InitVal	Description
31:0	Claim TAG Set	RW 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 366: Timestamp Claim Tag Clear Register

Offset: 0x00023FA4

Bit	Field	Type/InitVal	Description
31:0	Claim TAG Clear	RW 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 367: Timestamp Lock Access Register

Offset: 0x00023FB0

Bit	Field	Type/InitVal	Description
31:0	Lock Access	WO 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 368: Timestamp Lock Status Register

Offset: 0x00023FB4

Bit	Field	Type/InitVal	Description
31:0	Lock Status	RO 0x3	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 369: Timestamp Authentication Status Register

Offset: 0x00023FB8

Bit	Field	Type/InitVal	Description
31:0	Authentication Status	RO 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 370: Timestamp Device ID Register

Offset: 0x00023FC8

Bit	Field	Type/InitVal	Description
31:0	Device ID	RO 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 371: Timestamp Device Type Identifier Register

Offset: 0x00023FCC

Bit	Field	Type/InitVal	Description
31:0	Device Type Identifier	RO 0x4	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 372: Timestamp Peripheral ID4 Register

Offset: 0x00023FD0

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID4	RO 0x5	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 373: Timestamp Peripheral ID5 Register

Offset: 0x00023FD4

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID5	RO 0x9A	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 374: Timestamp Peripheral ID6 Register

Offset: 0x00023FD8

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID6	RO 0xE	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 375: Timestamp Peripheral ID7 Register

Offset: 0x00023FDC

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID7	RO 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 376: Timestamp Peripheral ID0 Register

Offset: 0x00023FE0

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID0	RO 0x3	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 377: Timestamp Peripheral ID1 Register

Offset: 0x00023FE4

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID1	RO 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 378: Timestamp Peripheral ID2 Register

Offset: 0x00023FE8

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID2	RO 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 379: Timestamp Peripheral ID3 Register

Offset: 0x00023FEC

Bit	Field	Type/InitVal	Description
31:0	Peripheral ID3	RO 0x0	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 380: Timestamp Component ID0 Register

Offset: 0x00023FF0

Bit	Field	Type/InitVal	Description
31:0	Component ID0	RO 0xD	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 381: Timestamp Component ID1 Register
Offset: 0x00023FF4

Bit	Field	Type/InitVal	Description
31:0	Component ID1	RO 0x90	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 382: Timestamp Component ID2 Register
Offset: 0x00023FF8

Bit	Field	Type/InitVal	Description
31:0	Component ID2	RO 0x5	See registers details over Coresight Design Kit Technical Refernce Manual.

Table 383: Timestamp Component ID3 Register
Offset: 0x00023FFC

Bit	Field	Type/InitVal	Description
31:0	Component ID3	RO 0xB1	See registers details over Coresight Design Kit Technical Refernce Manual.

A.2.24 Priority and Group Control

Table 384: Mbus Units Priority Control Register
Offset: 0x00020420

Bit	Field	Type/InitVal	Description
31:30	Unit 15 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=15 0 = Low 1 = Med 2 = High 3 = Top
29:28	Unit 14 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=14 0 = Low 1 = Med 2 = High 3 = Top

Copyright © 2014 Marvell

Table 384: Mbus Units Priority Control Register (Continued)
Offset: 0x00020420

Bit	Field	Type/InitVal	Description
27:26	Unit 13 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=13 0 = Low 1 = Med 2 = High 3 = Top
25:24	Unit 12 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=12 0 = Low 1 = Med 2 = High 3 = Top
23:22	Unit 11 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=11 0 = Low 1 = Med 2 = High 3 = Top
21:20	Unit 10 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=10 0 = Low 1 = Med 2 = High 3 = Top
19:18	Unit 9 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=9 0 = Low 1 = Med 2 = High 3 = Top
17:16	Unit 8 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=8 0 = Low 1 = Med 2 = High 3 = Top
15:14	Unit 7 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=7 0 = Low 1 = Med 2 = High 3 = Top
13:12	Unit 6 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=6 0 = Low 1 = Med 2 = High 3 = Top

Table 384: Mbus Units Priority Control Register (Continued)

Offset: 0x00020420

Bit	Field	Type/InitVal	Description
11:10	Unit 5 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=5 0 = Low 1 = Med 2 = High 3 = Top
9:8	Unit 4 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=4 0 = Low 1 = Med 2 = High 3 = Top
7:6	Unit 3 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=3 0 = Low 1 = Med 2 = High 3 = Top
5:4	Unit 2 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=2 0 = Low 1 = Med 2 = High 3 = Top
3:2	Unit 1 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=1 0 = Low 1 = Med 2 = High 3 = Top
1:0	Unit 0 Prio	RW 0x0	Defines the priority of requests initialized by unit with source_id=0 0 = Low 1 = Med 2 = High 3 = Top

Table 385: Fabric Units Priority Control Register

Offset: 0x00020424

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved

Table 385: Fabric Units Priority Control Register (Continued)

Offset: 0x00020424

Bit	Field	Type/InitVal	Description
7:6	CPU 3 Prio	RW 0x3	CPU3 Priority Defines the priority of requests that where initialized by CPU3 0 = Low 1 = Med 2 = High 3 = Top
5:4	CPU 2 Prio	RW 0x3	CPU2 Priority Defines the priority of requests that where initialized by CPU2 0 = Low 1 = Med 2 = High 3 = Top
3:2	CPU 1 Prio	RW 0x3	CPU1 Priority Defines the priority of requests that where initialized by CPU1 0 = Low 1 = Med 2 = High 3 = Top
1:0	CPU 0 Prio	RW 0x3	CPU0 Priority Defines the priority of requests that where initialized by CPU0 0 = Low 1 = Med 2 = High 3 = Top

Table 386: Mbus Units Prefetch Control Register

Offset: 0x00020428

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	Unit15PFen	RW 0x0	MBUS Unit15 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit15 0 = False 1 = True

Table 386: Mbus Units Prefetch Control Register (Continued)

Offset: 0x00020428

Bit	Field	Type/InitVal	Description
14	Unit14PFen	RW 0x0	MBUS Unit14 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit14 0 = False 1 = True
13	Unit13PFen	RW 0x0	MBUS Unit13 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit13 0 = False 1 = True
12	Unit12PFen	RW 0x0	MBUS Unit12 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit12 0 = False 1 = True
11	Unit11PFen	RW 0x0	MBUS Unit11 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit11 0 = False 1 = True
10	Unit10PFen	RW 0x0	MBUS Unit10 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit10 0 = False 1 = True
9	Unit9PFen	RW 0x0	MBUS Unit9 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit9 0 = False 1 = True
8	Unit8PFen	RW 0x0	MBUS Unit8 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit8 0 = False 1 = True

Copyright © 2014 Marvell

Table 386: Mbus Units Prefetch Control Register (Continued)

Offset: 0x00020428

Bit	Field	Type/InitVal	Description
7	Unit7PFen	RW 0x0	MBUS Unit7 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit7 0 = False 1 = True
6	Unit6PFen	RW 0x0	MBUS Unit6 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit6 0 = False 1 = True
5	Unit5PFen	RW 0x0	MBUS Unit5 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit5 0 = False 1 = True
4	Unit4PFen	RW 0x0	MBUS Unit4 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit4 0 = False 1 = True
3	Unit3PFen	RW 0x0	MBUS Unit3 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit3 0 = False 1 = True
2	Unit2PFen	RW 0x0	MBUS Unit2 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit2 0 = False 1 = True
1	Unit1PFen	RW 0x0	MBUS Unit1 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit1 0 = False 1 = True

Table 386: Mbus Units Prefetch Control Register (Continued)

Offset: 0x00020428

Bit	Field	Type/InitVal	Description
0	Unit0PFen	RW 0x0	MBUS Unit0 Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MBUS Unit0 0 = False 1 = True

Table 387: Fabric Units Prefetch Control Register

Offset: 0x0002042C

Bit	Field	Type/InitVal	Description
31:12	Reserved	RSVD 0x0	Reserved
11	CPU3 Data PFen	RW 0x0	CPU3 Data Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MMU or DATA of CPU3 0 = False 1 = True
10	CPU2 Data PFen	RW 0x0	CPU2 Data Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MMU or DATA of CPU2 0 = False 1 = True
9	CPU1 Data PFen	RW 0x0	CPU1 Data Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MMU or DATA of CPU1 0 = False 1 = True
8	CPU0 Data PFen	RW 0x0	CPU0 Data Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by MMU or DATA of CPU0 0 = False 1 = True
7:4	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 387: Fabric Units Prefetch Control Register (Continued)
Offset: 0x0002042C

Bit	Field	Type/InitVal	Description
3	CPU3 Instruction PFen	RW 0x0	CPU3 ICACHE Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by Icache of CPU3 0 = False 1 = True
2	CPU2 Instruction PFen	RW 0x0	CPU2 ICACHE Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by Icache of CPU2 0 = False 1 = True
1	CPU1 Instruction PFen	RW 0x0	CPU1 ICACHE Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by Icache of CPU1 0 = False 1 = True
0	CPU0 Instruction PFen	RW 0x0	CPU0 ICACHE Prefetch Enable Defines rather intermediate engines should perform a pre fetch action as a result of transaction issued by Icache of CPU0 0 = False 1 = True

A.2.25 Power Management Service Unit

A.2.25.1 PMU General

Table 388: Mbus to DDR Request ID Mask Register
Offset: 0x00022F10

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Mbus2DdrReqIDMask	RW 0x0	Controls which unit request to the DRAM, will result in a wake-up event. The requesting unit ID is used as the index to this configuration field. If the relevant bit is set to 1, it means this unit’s request should result in a wake-up event.

A.2.25.2 CPU0 PM Service

Table 389: CPU0 Statistic Counter Global Control Register
Offset: 0x00022100

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	CPU0StatCountEn	RW 0x0	CPU0 statistics counter enable control. 0 = Disable: Statistic counter disabled. 1 = Enable: Statistic counter enabled.

Table 390: CPU0 PM Control and Configuration Register
Offset: 0x00022104

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x0	Reserved
20	CPU0L2FabPwrDwnEn	RW 0x0	A value of 1 means the unit CPU0 agrees that the L2$ and NFabric will be powered down, while the unit is powered down.
19	Reserved	RSVD 0x0	Reserved
18	CPU0DfsReq	RW 0x0	CPU0 DFS request. Cleared by HW.
17	Reserved	RSVD 0x0	Reserved
16	CPU0PowerDownReq	RW 0x0	CPU0 power down request. Cleared by HW.
15:8	Reserved	RSVD 0x0	Reserved
7:4	CPU0FrequencyID	RW 0x1	Indicates CPU0 Frequency ID.
3:0	CPU0PowerDomainID	RW 0x1	Indicates CPU0 Power-Domain ID.

Table 391: CPU0 PM Status and Mask Register

Offset: 0x0002210C

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	CPU0Intr1Mask	RW 0x0	Controls the masking control of the unit's wake1 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit's interrupt. 1 = Mask: Mask the unit's interrupt.
24	CPU0Intr0Mask	RW 0x0	Controls the masking control of the unit's wake0 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit's interrupt. 1 = Mask: Mask the unit's interrupt.
23	CPU0Wake3Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
22	CPU0Wake2Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
21	CPU0Wake1Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
20	CPU0Wake0Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
19	Reserved	RSVD 0x0	Reserved
18	CPU0Idle2Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
17	CPU0Idle1Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
16	CPU0Idle0Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
15	CPU0PwrDwnStts	RO 0x0	Indicates CPU0 is at power down 0 = PowerDown 1 = PowerUp
14:8	Reserved	RSVD 0x0	Reserved
7	CPU0Wake3Stts	RO 0x0	Reflects CPU0 Wake3 status

Table 391: CPU0 PM Status and Mask Register (Continued)

Offset: 0x0002210C

Bit	Field	Type/InitVal	Description
6	CPU0Wake2Stts	RO 0x0	Reflects CPU0 Wake2 status
5	CPU0Wake1Stts	RO 0x0	ReflectsCPU0 Wake1 status
4	CPU0Wake0Stts	RO 0x0	Reflects CPU0 Wake0 status
3	Reserved	RSVD 0x0	Reserved
2	CPU0Idle2Stts	RO 0x0	Reflects CPU0 Idle2 status
1	CPU0Idle1Stts	RO 0x0	Reflects CPU0 Idle1 status
0	CPU0Idle0Stts	RO 0x0	Reflects CPU0 Idle0 status

Table 392: CPU0 Idle Timer Value Register

Offset: 0x00022110

Bit	Field	Type/InitVal	Description
31:0	CPU0IdleTimerVal	RW 0x400	This value specifies the time period in which the idle indication will be periodically monitored. At the end of each time period, the counted idle indication will be compared to the configurable thresholds.

Table 393: CPU0 Idle High Threshold Register

Offset: 0x00022114

Bit	Field	Type/InitVal	Description
31:0	CPU0IdleHiThrs	RW 0x300	This value specifies the Idle high threshold. If the number of Idle indications, counted during an “Idle Timer” period, is greater then this threshold, a IdleHighStatus interrupt will be set.

Table 394: CPU0 Idle Low Threshold Register

Offset: 0x00022118

Bit	Field	Type/InitVal	Description
31:0	CPU0IdleLoThrs	RW 0x100	This value specifies the Idle low threshold. If the number of Idle indications, counted during an “Idle Timer” period, is less then this threshold, a IdleLowStatus interrupt will be set.

Table 395: CPU0 Idle Result Register

Offset: 0x0002211C

Bit	Field	Type/InitVal	Description
31:0	CPU0IdleResult	RW 0x0	This value specifies the number of Idle indication which were counted during the “Idle Timer” period. This field is updated after each “Idle Timer” period is completed. If the number of Idle indication, counted during an “Idle Timer” period exceeds the thresholds, the result value is locked, to enable SW to read it. The SW needs to write (clear) this register, to unlock it.

Table 396: CPU0 PM Event Status and Mask Register

Offset: 0x00022120

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	CPU0IdleLowThrshIntMask	RW 0x0	Mask control of the CPU0IdleLowThrshInt event, to generate a unit’s PM interrupt.
24	CPU0IdleHighThrshIntMask	RW 0x0	Mask control of the CPU0IdleHighThrshInt event, to generate a unit’s PM interrupt.
23:18	Reserved	RSVD 0x0	Reserved
17	CPU0DfsDoneMask	RW 0x0	Mask control of the CPU0DfsDone event, to generate a unit’s PM interrupt.
16	CPU0PoweredUpMask	RW 0x0	Mask control of the CPU0PoweredUp event, to generate a unit’s PM interrupt.
15:10	Reserved	RSVD 0x0	Reserved

Table 396: CPU0 PM Event Status and Mask Register (Continued)

Offset: 0x00022120

Bit	Field	Type/InitVal	Description
9	CPU0IdleLowThrshInt	RW1C 0x0	CPU0 Idle statistics counter low threshold interrupt.
8	CPU0IdleHighThrshInt	RW1C 0x0	CPU0 Idle statistics counter high threshold interrupt.
7:2	Reserved	RSVD 0x0	Reserved
1	CPU0DfsDone	RW1C 0x0	Set by the PMU to indicate that the DFS flow has completed.
0	CPU0PoweredUp	RW1C 0x0	Set by the PMU to indicate that the power up flow has completed.

Table 397: CPU0 Boot Address Redirect Register

Offset: 0x00022124

Bit	Field	Type/InitVal	Description
31:0	CPU0BootAddrRedirect	RW 0xffff0000	CPU0 Boot Address Redirect. Software can prepare in this register, the address to which CPU0 is required to boot from. The BootRom firmware should use this register content to redirect the boot address.

Table 398: CPU0 Resume Control Register

Offset: 0x00022128

Bit	Field	Type/InitVal	Description
31:0	ResumeControlInfo	RW 0x0	Resume Control Information, used by software to determine if and what actions needs to be performed before continue to execute from the Boot Address Redirect, when resuming from power down.

A.2.25.3 CPU1 PM Service

Table 399: CPU1 Statistic Counter Global Control Register

Offset: 0x00022200

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved

Table 399: CPU1 Statistic Counter Global Control Register (Continued)

Offset: 0x00022200

Bit	Field	Type/InitVal	Description
0	CPU1StatCountEn	RW 0x0	CPU1 statistics counter enable control. 0 = Disable: Statistic counter disabled. 1 = Enable: Statistic counter enabled.

Table 400: CPU1 PM Control and Configuration Register

Offset: 0x00022204

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x0	Reserved
20	CPU1L2FabPwrDwnEn	RW 0x0	A value of 1 means the unit CPU1 agrees that the L2$ and NFabric will be powered down, while the unit is powered down.
19	Reserved	RSVD 0x0	Reserved
18	CPU1DfsReq	RW 0x0	CPU1 DFS request. Cleared by HW.
17	Reserved	RSVD 0x0	Reserved
16	CPU1PowerDownReq	RW 0x0	CPU1 power down request. Cleared by HW.
15:8	Reserved	RSVD 0x0	Reserved
7:4	CPU1FrequencyID	RW 0x2	Indicates CPU1 Frequency ID.
3:0	CPU1PowerDomainID	RW 0x2	Indicates CPU1 Power-Domain ID.

Table 401: CPU1 PM Status and Mask Register

Offset: 0x0002220C

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved

Table 401: CPU1 PM Status and Mask Register (Continued)

Offset: 0x0002220C

Bit	Field	Type/InitVal	Description
25	CPU1Intr1Mask	RW 0x0	Controls the masking control of the unit's wake1 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit's interrupt. 1 = Mask: Mask the unit's interrupt.
24	CPU1Intr0Mask	RW 0x0	Controls the masking control of the unit's wake0 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit's interrupt. 1 = Mask: Mask the unit's interrupt.
23	CPU1Wake3Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
22	CPU1Wake2Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
21	CPU1Wake1Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
20	CPU1Wake0Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
19	Reserved	RSVD 0x0	Reserved
18	CPU1Idle2Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
17	CPU1Idle1Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
16	CPU1Idle0Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
15	CPU1PwrDwnStts	RO 0x0	Indicates CPU1 is at power down 0 = PowerDown 1 = PowerUp
14:8	Reserved	RSVD 0x0	Reserved
7	CPU1Wake3Stts	RO 0x0	Reflects CPU1 Wake3 status
6	CPU1Wake2Stts	RO 0x0	Reflects CPU1 Wake2 status

Table 401: CPU1 PM Status and Mask Register (Continued)

Offset: 0x0002220C

Bit	Field	Type/InitVal	Description
5	CPU1Wake1Stts	RO 0x0	ReflectsCPU1 Wake1 status
4	CPU1Wake0Stts	RO 0x0	Reflects CPU1 Wake0 status
3	Reserved	RSVD 0x0	Reserved
2	CPU1Idle2Stts	RO 0x0	Reflects CPU1 Idle2 status
1	CPU1Idle1Stts	RO 0x0	Reflects CPU1 Idle1 status
0	CPU1Idle0Stts	RO 0x0	Reflects CPU1 Idle0 status

Table 402: CPU1 Idle Timer Value Register

Offset: 0x00022210

Bit	Field	Type/InitVal	Description
31:0	CPU1IdleTimerVal	RW 0x400	This value specifies the time period in which the idle indication will be periodically monitored. At the end of each time period, the counted idle indication will be compared to the configurable thresholds.

Table 403: CPU1 Idle High Threshold Register

Offset: 0x00022214

Bit	Field	Type/InitVal	Description
31:0	CPU1IdleHiThrs	RW 0x300	This value specifies the Idle high threshold. If the number of Idle indications, counted during an “Idle Timer” period, is greater then this threshold, a IdleHighStatus interrupt will be set.

Table 404: CPU1 Idle Low Threshold Register

Offset: 0x00022218

Bit	Field	Type/InitVal	Description
31:0	CPU1IdleLoThrs	RW 0x100	This value specifies the Idle low threshold. If the number of Idle indications, counted during an “Idle Timer” period, is less then this threshold, a IdleLowStatus interrupt will be set.

Table 405: CPU1 Idle Result Register

Offset: 0x0002221C

Bit	Field	Type/InitVal	Description
31:0	CPU1IdleResult	RW 0x0	This value specifies the number of Idle indication which were counted during the "Idle Timer" period. This field is updated after each "Idle Timer" period is completed. If the number of Idle indication, counted during an "Idle Timer" period exceeds the thresholds, the result value is locked, to enable SW to read it. The SW needs to write (clear) this register, to unlock it.

Table 406: CPU1 PM Event Status and Mask Register

Offset: 0x00022220

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	CPU1IdleLowThrshIntMask	RW 0x0	Mask control of the CPU1IdleLowThrshInt event, to generate a unit's PM interrupt.
24	CPU1IdleHighThrshIntMask	RW 0x0	Mask control of the CPU1IdleHighThrshInt event, to generate a unit's PM interrupt.
23:18	Reserved	RSVD 0x0	Reserved
17	CPU1DfsDoneMask	RW 0x0	Mask control of the CPU1DfsDone event, to generate a unit's PM interrupt.
16	CPU1PoweredUpMask	RW 0x0	Mask control of the CPU1PoweredUp event, to generate a unit's PM interrupt.
15:10	Reserved	RSVD 0x0	Reserved
9	CPU1IdleLowThrshInt	RW1C 0x0	CPU1 Idle statistics counter low threshold interrupt.
8	CPU1IdleHighThrshInt	RW1C 0x0	CPU1 Idle statistics counter high threshold interrupt.
7:2	Reserved	RSVD 0x0	Reserved
1	CPU1DfsDone	RW1C 0x0	Set by the PMU to indicate that the DFS flow has completed.

Table 406: CPU1 PM Event Status and Mask Register (Continued)

Offset: 0x00022220

Bit	Field	Type/InitVal	Description
0	CPU1PoweredUp	RW1C 0x0	Set by the PMU to indicate that the power up flow has completed.

Table 407: CPU1 Boot Address Redirect Register

Offset: 0x00022224

Bit	Field	Type/InitVal	Description
31:0	CPU1BootAddrRedirect	RW 0xffff0000	CPU1 Boot Address Redirect. Software can prepare in this register, the address to which CPU1 is required to boot from. The BootRom firmware should use this register content to redirect the boot address.

Table 408: CPU1 Resume Control Register

Offset: 0x00022228

Bit	Field	Type/InitVal	Description
31:0	ResumeControlInfo	RW 0x0	Resume Control Information, used by software to determine if and what actions needs to be performed before continue to execute from the Boot Address Redirect, when resuming from power down.

A.2.25.4 CPU2 PM Service

Table 409: CPU2 Statistic Counter Global Control Register

Offset: 0x00022300

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	CPU2StatCountEn	RW 0x0	CPU2 statistics counter enable control. 0 = Disable: Statistic counter disabled. 1 = Enable: Statistic counter enabled.

Table 410: CPU2 PM Control and Configuration Register

Offset: 0x00022304

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x0	Reserved

Table 410: CPU2 PM Control and Configuration Register (Continued)

Offset: 0x00022304

Bit	Field	Type/InitVal	Description
20	CPU2L2FabPwrDwnEn	RW 0x0	A value of 1 means the unit CPU2 agrees that the L2$ and NFabric will be powered down, while the unit is powered down.
19	Reserved	RSVD 0x0	Reserved
18	CPU2DfsReq	RW 0x0	CPU2 DFS request. Cleared by HW.
17	Reserved	RSVD 0x0	Reserved
16	CPU2PowerDownReq	RW 0x0	CPU2 power down request. Cleared by HW.
15:8	Reserved	RSVD 0x0	Reserved
7:4	CPU2FrequencyID	RW 0x3	Indicates CPU2 Frequency ID.
3:0	CPU2PowerDomainID	RW 0x3	Indicates CPU2 Power-Domain ID.

Table 411: CPU2 PM Status and Mask Register

Offset: 0x0002230C

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	CPU2Intr1Mask	RW 0x0	Controls the masking control of the unit’s wake1 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit’s interrupt. 1 = Mask: Mask the unit’s interrupt.
24	CPU2Intr0Mask	RW 0x0	Controls the masking control of the unit’s wake0 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit’s interrupt. 1 = Mask: Mask the unit’s interrupt.
23	CPU2Wake3Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
22	CPU2Wake2Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.

Table 411: CPU2 PM Status and Mask Register (Continued)

Offset: 0x0002230C

Bit	Field	Type/InitVal	Description
21	CPU2Wake1Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
20	CPU2Wake0Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
19	Reserved	RSVD 0x0	Reserved
18	CPU2Idle2Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
17	CPU2Idle1Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
16	CPU2Idle0Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
15	CPU2PwrDwnStts	RO 0x0	Indicates CPU2 is at power down 0 = PowerDown 1 = PowerUp
14:8	Reserved	RSVD 0x0	Reserved
7	CPU2Wake3Stts	RO 0x0	Reflects CPU2 Wake3 status
6	CPU2Wake2Stts	RO 0x0	Reflects CPU2 Wake2 status
5	CPU2Wake1Stts	RO 0x0	ReflectsCPU2 Wake1 status
4	CPU2Wake0Stts	RO 0x0	Reflects CPU2 Wake0 status
3	Reserved	RSVD 0x0	Reserved
2	CPU2Idle2Stts	RO 0x0	Reflects CPU2 Idle2 status
1	CPU2Idle1Stts	RO 0x0	Reflects CPU2 Idle1 status
0	CPU2Idle0Stts	RO 0x0	Reflects CPU2 Idle0 status

Table 412: CPU2 Idle Timer Value Register

Offset: 0x00022310

Bit	Field	Type/InitVal	Description
31:0	CPU2IdleTimerVal	RW 0x400	This value specifies the time period in which the idle indication will be periodically monitored. At the end of each time period, the counted idle indication will be compared to the configurable thresholds.

Table 413: CPU2 Idle High Threshold Register

Offset: 0x00022314

Bit	Field	Type/InitVal	Description
31:0	CPU2IdleHiThrs	RW 0x300	This value specifies the Idle high threshold. If the number of Idle indications, counted during an “Idle Timer” period, is greater then this threshold, a IdleHighStatus interrupt will be set.

Table 414: CPU2 Idle Low Threshold Register

Offset: 0x00022318

Bit	Field	Type/InitVal	Description
31:0	CPU2IdleLoThrs	RW 0x100	This value specifies the Idle low threshold. If the number of Idle indications, counted during an “Idle Timer” period, is less then this threshold, a IdleLowStatus interrupt will be set.

Table 415: CPU2 Idle Result Register

Offset: 0x0002231C

Bit	Field	Type/InitVal	Description
31:0	CPU2IdleResult	RW 0x0	This value specifies the number of Idle indication which were counted during the “Idle Timer” period. This field is updated after each “Idle Timer” period is completed. If the number of Idle indication, counted during an “Idle Timer” period exceeds the thresholds, the result value is locked, to enable SW to read it. The SW needs to write (clear) this register, to unlock it.

Table 416: CPU2 PM Event Status and Mask Register

Offset: 0x00022320

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	CPU2IdleLowThrshIntMask	RW 0x0	Mask control of the CPU2IdleLowThrshInt event, to generate a unit's PM interrupt.
24	CPU2IdleHighThrshIntMask	RW 0x0	Mask control of the CPU2IdleHighThrshInt event, to generate a unit's PM interrupt.
23:18	Reserved	RSVD 0x0	Reserved
17	CPU2DfsDoneMask	RW 0x0	Mask control of the CPU2DfsDone event, to generate a unit's PM interrupt.
16	CPU2PoweredUpMask	RW 0x0	Mask control of the CPU2PoweredUp event, to generate a unit's PM interrupt.
15:10	Reserved	RSVD 0x0	Reserved
9	CPU2IdleLowThrshInt	RW1C 0x0	CPU2 Idle statistics counter low threshold interrupt.
8	CPU2IdleHighThrshInt	RW1C 0x0	CPU2 Idle statistics counter high threshold interrupt.
7:2	Reserved	RSVD 0x0	Reserved
1	CPU2DfsDone	RW1C 0x0	Set by the PMU to indicate that the DFS flow has completed.
0	CPU2PoweredUp	RW1C 0x0	Set by the PMU to indicate that the power up flow has completed.

Table 417: CPU2 Boot Address Redirect Register

Offset: 0x00022324

Bit	Field	Type/InitVal	Description
31:0	CPU2BootAddrRedirect	RW 0xffff0000	CPU2 Boot Address Redirect. Software can prepare in this register, the address to which CPU2 is required to boot from. The BootRom firmware should use this register content to redirect the boot address.

Table 418: CPU2 Resume Control Register

Offset: 0x00022328

Bit	Field	Type/InitVal	Description
31:0	ResumeControlInfo	RW 0x0	Resume Control Information, used by software to determine if and what actions needs to be performed before continue to execute from the Boot Address Redirect, when resuming from power down.

A.2.25.5 CPU3 PM Service

Table 419: CPU3 Statistic Counter Global Control Register

Offset: 0x00022400

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	CPU3StatCountEn	RW 0x0	CPU3 statistics counter enable control. 0 = Disable: Statistic counter disabled. 1 = Enable: Statistic counter enabled.

Table 420: CPU3 PM Control and Configuration Register

Offset: 0x00022404

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x0	Reserved
20	CPU3L2FabPwrDwnEn	RW 0x0	A value of 1 means the unit CPU3 agrees that the L2$ and NFabric will be powered down, while the unit is powered down.
19	Reserved	RSVD 0x0	Reserved
18	CPU3DfsReq	RW 0x0	CPU3 DFS request. Cleared by HW.
17	Reserved	RSVD 0x0	Reserved
16	CPU3PowerDownReq	RW 0x0	CPU3 power down request. Cleared by HW.
15:8	Reserved	RSVD 0x0	Reserved
7:4	CPU3FrequencyID	RW 0x4	Indicates CPU3 Frequency ID.

Table 420: CPU3 PM Control and Configuration Register (Continued)

Offset: 0x00022404

Bit	Field	Type/InitVal	Description
3:0	CPU3PowerDomainID	RW 0x4	Indicates CPU3 Power-Domain ID.

Table 421: CPU3 PM Status and Mask Register

Offset: 0x0002240C

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	CPU3Intr1Mask	RW 0x0	Controls the masking control of the unit's wake1 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit's interrupt. 1 = Mask: Mask the unit's interrupt.
24	CPU3Intr0Mask	RW 0x0	Controls the masking control of the unit's wake0 indication, as an interrupt. A value of 1 means that this interrupt signals will be masked from the unit. 0 = NoMask: Do not mask the unit's interrupt. 1 = Mask: Mask the unit's interrupt.
23	CPU3Wake3Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
22	CPU3Wake2Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
21	CPU3Wake1Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
20	CPU3Wake0Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
19	Reserved	RSVD 0x0	Reserved
18	CPU3Idle2Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
17	CPU3Idle1Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
16	CPU3Idle0Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.

Table 421: CPU3 PM Status and Mask Register (Continued)

Offset: 0x0002240C

Bit	Field	Type/InitVal	Description
15	CPU3PwrDwnStts	RO 0x0	Indicates CPU3 is at power down 0 = PowerDown 1 = PowerUp
14:8	Reserved	RSVD 0x0	Reserved
7	CPU3Wake3Stts	RO 0x0	Reflects CPU3 Wake3 status
6	CPU3Wake2Stts	RO 0x0	Reflects CPU3 Wake2 status
5	CPU3Wake1Stts	RO 0x0	ReflectsCPU3 Wake1 status
4	CPU3Wake0Stts	RO 0x0	Reflects CPU3 Wake0 status
3	Reserved	RSVD 0x0	Reserved
2	CPU3Idle2Stts	RO 0x0	Reflects CPU3 Idle2 status
1	CPU3Idle1Stts	RO 0x0	Reflects CPU3 Idle1 status
0	CPU3Idle0Stts	RO 0x0	Reflects CPU3 Idle0 status

Table 422: CPU3 Idle Timer Value Register

Offset: 0x00022410

Bit	Field	Type/InitVal	Description
31:0	CPU3IdleTimerVal	RW 0x400	This value specifies the time period in which the idle indication will be periodically monitored. At the end of each time period, the counted idle indication will be compared to the configurable thresholds.

Table 423: CPU3 Idle High Threshold Register

Offset: 0x00022414

Bit	Field	Type/InitVal	Description
31:0	CPU3IdleHiThrs	RW 0x300	This value specifies the Idle high threshold. If the number of Idle indications, counted during an “Idle Timer” period, is greater then this threshold, a IdleHighStatus interrupt will be set.

Table 424: CPU3 Idle Low Threshold Register

Offset: 0x00022418

Bit	Field	Type/InitVal	Description
31:0	CPU3IdleLoThrs	RW 0x100	This value specifies the Idle low threshold. If the number of Idle indications, counted during an “Idle Timer” period, is less then this threshold, a IdleLowStatus interrupt will be set.

Table 425: CPU3 Idle Result Register

Offset: 0x0002241C

Bit	Field	Type/InitVal	Description
31:0	CPU3IdleResult	RW 0x0	This value specifies the number of Idle indication which were counted during the “Idle Timer” period. This field is updated after each “Idle Timer” period is completed. If the number of Idle indication, counted during an “Idle Timer” period exceeds the thresholds, the result value is locked, to enable SW to read it. The SW needs to write (clear) this register, to unlock it.

Table 426: CPU3 PM Event Status and Mask Register

Offset: 0x00022420

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	CPU3IdleLowThrshIntMask	RW 0x0	Mask control of the CPU3IdleLowThrshInt event, to generate a unit’s PM interrupt.
24	CPU3IdleHighThrshIntMask	RW 0x0	Mask control of the CPU3IdleHighThrshInt event, to generate a unit’s PM interrupt.
23:18	Reserved	RSVD 0x0	Reserved

Table 426: CPU3 PM Event Status and Mask Register (Continued)

Offset: 0x00022420

Bit	Field	Type/InitVal	Description
17	CPU3DfsDoneMask	RW 0x0	Mask control of the CPU3DfsDone event, to generate a unit's PM interrupt.
16	CPU3PoweredUpMask	RW 0x0	Mask control of the CPU3PoweredUp event, to generate a unit's PM interrupt.
15:10	Reserved	RSVD 0x0	Reserved
9	CPU3IdleLowThrshlnt	RW1C 0x0	CPU3 Idle statistics counter low threshold interrupt.
8	CPU3IdleHighThrshlnt	RW1C 0x0	CPU3 Idle statistics counter high threshold interrupt.
7:2	Reserved	RSVD 0x0	Reserved
1	CPU3DfsDone	RW1C 0x0	Set by the PMU to indicate that the DFS flow has completed.
0	CPU3PoweredUp	RW1C 0x0	Set by the PMU to indicate that the power up flow has completed.

Table 427: CPU3 Boot Address Redirect Register

Offset: 0x00022424

Bit	Field	Type/InitVal	Description
31:0	CPU3BootAddrRedirect	RW 0xffff0000	CPU3 Boot Address Redirect. Software can prepare in this register, the address to which CPU3 is required to boot from. The BootRom firmware should use this register content to redirect the boot address.

Table 428: CPU3 Resume Control Register

Offset: 0x00022428

Bit	Field	Type/InitVal	Description
31:0	ResumeControlInfo	RW 0x0	Resume Control Information, used by software to determine if and what actions needs to be performed before continue to execute from the Boot Address Redirect, when resuming from power down.

A.2.25.6 Fabric PM Service

Table 429: L2C NFabric PM Control and Configuration Register
Offset: 0x00022004

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:4	L2C NFabricFrequencyID	RW 0x0	Indicates L2C NFabric Frequency ID.
3:0	L2C NFabricPowerDomainID	RW 0x0	Indicates L2C NFabric Power-Domain ID.

Table 430: L2C NFabric PM Status and Mask Register
Offset: 0x0002200C

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved
23	L2C NFabricWake3Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
22	L2C NFabricWake2Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
21	L2C NFabricWake1Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
20	L2C NFabricWake0Mask	RW 0x0	A value of 1 indicates that the corresponding wake event will trigger the power up flow.
19	Reserved	RSVD 0x0	Reserved
18	L2C NFabricIdle2Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
17	L2C NFabricIdle1Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.
16	L2C NFabricIdle0Mask	RW 0x0	A value of 1 indicates that the PMU will wait for corresponding idle indication before starting the power down flow.

Table 430: L2C NFabric PM Status and Mask Register (Continued)

Offset: 0x0002200C

Bit	Field	Type/InitVal	Description
15	L2C NFabricPwrDwnStts	RO 0x0	Indicates L2C NFabric is at power down 0 = PowerDown 1 = PowerUp
14:8	Reserved	RSVD 0x0	Reserved
7	L2C NFabricWake3Stts	RO 0x0	Reflects L2C NFabric Wake3 status
6	L2C NFabricWake2Stts	RO 0x0	Reflects L2C NFabric Wake2 status
5	L2C NFabricWake1Stts	RO 0x0	ReflectsL2C NFabric Wake1 status
4	L2C NFabricWake0Stts	RO 0x0	Reflects L2C NFabric Wake0 status
3	Reserved	RSVD 0x0	Reserved
2	L2C NFabricIdle2Stts	RO 0x0	Reflects L2C NFabric Idle2 status
1	L2C NFabricIdle1Stts	RO 0x0	Reflects L2C NFabric Idle1 status
0	L2C NFabricIdle0Stts	RO 0x0	Reflects L2C NFabric Idle0 status

Table 431: L2C NFabric PM Event Status and Mask Register

Offset: 0x00022020

Bit	Field	Type/InitVal	Description
31:17	Reserved	RSVD 0x0	Reserved
16	L2C NFabricPoweredUp Mask	RW 0x0	Mask control of the L2C NFabricPoweredUp event, to generate a unit's PM interrupt.
15:1	Reserved	RSVD 0x0	Reserved

Table 431: L2C NFabric PM Event Status and Mask Register (Continued)
Offset: 0x00022020

Bit	Field	Type/InitVal	Description
0	L2C NFabricPoweredUp	RW1C 0x0	Set by the PMU to indicate that the power up flow has completed.

A.2.25.7 Memory Controller PM Service

Table 432: MC Statistic Counter Global Control Register
Offset: 0x00022E00

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	MCStatCountEn	RW 0x0	MC statistics counter enable control. 0 = Disable: Statistic counter disabled. 1 = Enable: Statistic counter enabled.

Table 433: MC Idle Timer Value Register
Offset: 0x00022E10

Bit	Field	Type/InitVal	Description
31:0	MCIdleTimerVal	RW 0x400	This value specifies the time period in which the idle indication will be periodically monitored. At the end of each time period, the counted idle indication will be compared to the configurable thresholds.

Table 434: MC Idle High Threshold Register
Offset: 0x00022E14

Bit	Field	Type/InitVal	Description
31:0	MCIdleHiThrs	RW 0x300	This value specifies the Idle high threshold. If the number of Idle indications, counted during an “Idle Timer” period, is greater then this threshold, a IdleHighStatus interrupt will be set.

Table 435: MC Idle Low Threshold Register

Offset: 0x00022E18

Bit	Field	Type/InitVal	Description
31:0	MCIdleLoThrs	RW 0x100	This value specifies the Idle low threshold. If the number of Idle indications, counted during an “Idle Timer” period, is less then this threshold, a IdleLowStatus interrupt will be set.

Table 436: MC Idle Result Register

Offset: 0x00022E1C

Bit	Field	Type/InitVal	Description
31:0	MCIdleResult	RW 0x0	This value specifies the number of Idle indication which were counted during the “Idle Timer” period. This field is updated after each “Idle Timer” period is completed. If the number of Idle indication, counted during an “Idle Timer” period exceeds the thresholds, the result value is locked, to enable SW to read it. The SW needs to write (clear) this register, to unlock it.

Table 437: MC PM Event Status and Mask Register

Offset: 0x00022E20

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	MCIdleLowThrshInt Mask	RW 0x0	Mask control of the MCIdleLowThrshInt event, to generate a unit’s PM interrupt.
24	MCIdleHighThrshInt Mask	RW 0x0	Mask control of the MCIdleHighThrshInt event, to generate a unit’s PM interrupt.
23:10	Reserved	RSVD 0x0	Reserved
9	MCIdleLowThrshInt	RW1C 0x0	MC Idle statistics counter low threshold interrupt.
8	MCIdleHighThrshInt	RW1C 0x0	MC Idle statistics counter high threshold interrupt.
7:0	Reserved	RSVD 0x0	Reserved

A.3 DRAM Controller Registers

The following table provides a summarized list of all registers that belong to the DRAM Controller, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 438: Summary Map Table for the DRAM Controller Registers

Register Name	Offset	Table and Page
SDRAM Data Protection and Error Report Registers		
SDRAM Error Data (High) Register	0x00001440	Table 439, p. 766
SDRAM Error Data (Low) Register	0x00001444	Table 440, p. 766
SDRAM Received ECC Register	0x00001448	Table 441, p. 767
SDRAM Calculated ECC Register	0x0000144C	Table 442, p. 767
SDRAM Error Address Register	0x00001450	Table 443, p. 767
SDRAM Error Control Register	0x00001454	Table 444, p. 768
Single-Bit Error Counter Register	0x00001458	Table 445, p. 769
Double-Bit Error Counter Register	0x0000145C	Table 446, p. 769
DDR Controller Error Interrupt Cause Register	0x000014D0	Table 447, p. 769
DDR Controller Error Interrupt Mask Register	0x000014D4	Table 448, p. 770
DDR Controller Message Interrupt Cause Register	0x000014D8	Table 449, p. 771
DDR Controller Message Interrupt Mask Register	0x000014DC	Table 450, p. 772
SDRAM Control Registers		
SDRAM Configuration Register	0x00001400	Table 451, p. 773
DDR Controller Control (Low) Register	0x00001404	Table 452, p. 774
SDRAM Timing (Low) Register	0x00001408	Table 453, p. 776
SDRAM Timing (High) Register	0x0000140C	Table 454, p. 777
SDRAM Address Control Register	0x00001410	Table 455, p. 779
SDRAM Open Pages Control Register	0x00001414	Table 456, p. 781
SDRAM Operation Register	0x00001418	Table 457, p. 782
DDR Controller Control (High) Register	0x00001424	Table 458, p. 785
DDR ODT Timing (Low) Register	0x00001428	Table 459, p. 786
DDR3 Timing Register	0x0000142C	Table 460, p. 786
SDRAM Interface Mbus Control (Low) Register	0x00001430	Table 461, p. 787
SDRAM Interface Mbus Control (High) Register	0x00001434	Table 462, p. 788
SDRAM Interface Mbus Timeout Register	0x00001438	Table 463, p. 788
VTT Control Register	0x00001470	Table 464, p. 788
SDRAM Auto Power Save Register	0x00001474	Table 465, p. 789
DDR ODT Timing (High) Register	0x0000147C	Table 466, p. 790
SDRAM Initialization Control Register	0x00001480	Table 467, p. 791
AXI Mbus Arbiter Configuration Register	0x00001488	Table 468, p. 792
SDRAM ODT Control (Low) Register	0x00001494	Table 469, p. 793
SDRAM ODT Control (High) Register	0x00001498	Table 470, p. 797

Copyright © 2014 Marvell

Table 438: Summary Map Table for the DRAM Controller Registers (Continued)

Register Name	Offset	Table and Page
DDR Controller ODT Control Register	0x0000149C	Table 471, p. 798
Read Buffer Select Register	0x000014A4	Table 472, p. 799
AXI Control Register	0x000014A8	Table 473, p. 801
DRAM Controller Miscellaneous Register	0x000014B0	Table 474, p. 802
DRAM Address and Control Driving Strength (Low) Register	0x000014C0	Table 475, p. 802
DRAM Data and DQS Driving Strength (Low) Register	0x000014C4	Table 476, p. 803
DRAM Vertical Calibration Machine Control Register	0x000014C8	Table 477, p. 806
DRAM Main Pads Calibration Machine Control Register	0x000014CC	Table 478, p. 807
SDRAM Interface Mbus Control2 (Low) Register	0x000014F4	Table 479, p. 808
SDRAM Interface Mbus Control2 (High) Register	0x000014F8	Table 480, p. 808
Dynamic Power Save Register	0x00001520	Table 481, p. 809
DDR IO Register	0x00001524	Table 482, p. 810
DFS Register	0x00001528	Table 483, p. 810
Read Data Sample Delays Register	0x00001538	Table 484, p. 812
Read Data Ready Delays Register	0x0000153C	Table 485, p. 813
Training Register	0x000015B0	Table 486, p. 814
Training Software 1 Register	0x000015B4	Table 487, p. 815
Training Software 2 Register	0x000015B8	Table 488, p. 816
Training Patterns Base Address Register	0x000015BC	Table 489, p. 817
Training Debug 2 Register	0x000015C4	Table 490, p. 817
Training Debug 3 Register	0x000015C8	Table 491, p. 818
DDR3 MR0 Register	0x000015D0	Table 492, p. 819
DDR3 MR1 Register	0x000015D4	Table 493, p. 820
DDR3 MR2 Register	0x000015D8	Table 494, p. 822
DDR3 MR3 Register	0x000015DC	Table 495, p. 823
DDR3 Rank Control Register	0x000015E0	Table 496, p. 824
ZQC Configuration Register	0x000015E4	Table 497, p. 824
DRAM PHY Configuration Register	0x000015EC	Table 498, p. 825
ODPG CTRL Control Register	0x00001600	Table 499, p. 826
PHY Lock Mask Register	0x00001670	Table 500, p. 828
DRAM PHY Lock Status Register	0x00001674	Table 501, p. 829
PHY Register File Access	0x000016A0	Table 502, p. 831
Training Write Leveling Register	0x000016AC	Table 503, p. 832
DDR3 Registered DRAM Control	0x000016D0	Table 504, p. 833
DDR3 Registered DRAM Timing	0x000016D4	Table 505, p. 834
DLB Control Register	0x00001700	Table 506, p. 835
DLB Bus Optimization Weights Register	0x00001704	Table 507, p. 836
DLB Aging Register	0x00001708	Table 508, p. 839

Copyright © 2014 Marvell

Table 438: Summary Map Table for the DRAM Controller Registers (Continued)

Register Name	Offset	Table and Page
DLB Eviction Control Register	0x0000170C	Table 509, p. 839
DLB Eviction Timers Register	0x00001710	Table 510, p. 841
DRAM Horizontal Calibration machine control Register	0x000017C8	Table 511, p. 841
DDR3 MR0 CSn Register (n=0–3)	CS number0: 0x00001870, CS number1: 0x00001880, CS number2: 0x00001890, CS number3: 0x000018A0	Table 512, p. 842
DDR3 MR1 CSn Register (n=0–3)	CS number0: 0x00001874, CS number1: 0x00001884, CS number2: 0x00001894, CS number3: 0x000018A4	Table 513, p. 844
DDR3 MR2 CSn Register (n=0–3)	CS number0: 0x00001878, CS number1: 0x00001888, CS number2: 0x00001898, CS number3: 0x000018A8	Table 514, p. 845
DDR3 MR3 CSn Register (n=0–3)	CS number0: 0x0000187C, CS number1: 0x0000188C, CS number2: 0x0000189C, CS number3: 0x000018AC	Table 515, p. 846
SDRAM Scratch Pad		
Scratch Pad 0 Register	0x00001504	Table 516, p. 847
Scratch Pad 1 Register	0x0000150C	Table 517, p. 847
Scratch Pad 2 Register	0x00001514	Table 518, p. 847
Scratch Pad 3 Register	0x0000151C	Table 519, p. 848
DDR Data PHY Registers		
Chip Select (CSn) Write Leveling Register (n=0–3)	M_CS number0: 0x00000000, M_CS number1: 0x00000004, M_CS number2: 0x00000008, M_CS number3: 0x0000000C	Table 520, p. 848
Chip Select (CSn) Write DQS Reference Delay Register (n=0–3)	M_CS number0: 0x00000001, M_CS number1: 0x00000005, M_CS number2: 0x00000009, M_CS number3: 0x0000000D	Table 521, p. 849
Chip Select (CSn) Read Leveling Register (n=0–3)	M_CS number0: 0x00000002, M_CS number1: 0x00000006, M_CS number2: 0x0000000A, M_CS number3: 0x0000000E	Table 522, p. 849
Chip Select (CSn) Read DQS Reference Delay Register (n=0–3)	M_CS number0: 0x00000003, M_CS number1: 0x00000007, M_CS number2: 0x0000000B, M_CS number3: 0x0000000F	Table 523, p. 850
DDR PHY Registers (Internal)		
Data PHY Registers (Internal)		

Copyright © 2014 Marvell

Table 438: Summary Map Table for the DRAM Controller Registers (Continued)

Register Name	Offset	Table and Page
Data Bitn Write Deskew Register (n=0–7)	DQ bit number0: 0x00000010, DQ bit number1: 0x00000011, DQ bit number2: 0x00000012, DQ bit number3: 0x00000013, DQ bit number4: 0x00000014, DQ bit number5: 0x00000015, DQ bit number6: 0x00000016, DQ bit number7: 0x00000017	Table 524, p. 851
DQS Write Deskew Register	0x00000018	Table 525, p. 851
DQS B Write Deskew Register	0x00000019	Table 526, p. 852
DM Write Deskew Register	0x0000001A	Table 527, p. 852
Data Bitn Read Deskew Register (n=0–7)	DQ bit number0: 0x00000030, DQ bit number1: 0x00000031, DQ bit number2: 0x00000032, DQ bit number3: 0x00000033, DQ bit number4: 0x00000034, DQ bit number5: 0x00000035, DQ bit number6: 0x00000036, DQ bit number7: 0x00000037	Table 528, p. 852
DQS Read Deskew Register	0x00000038	Table 529, p. 853
PAD Control Register	0x0000003D	Table 530, p. 853
Control PHY Registers		
Control n Deskew Register (n=0–6)		Table 531, p. 854
Clock n Deskew Register (n=0–1)		Table 532, p. 854

A.3.1 SDRAM Data Protection and Error Report Registers

Table 439: SDRAM Error Data (High) Register

Offset: 0x00001440

Bit	Field	Type/InitVal	Description
31:0	ECCData_High	RO 0x0	Sampled 32 high bits of the last data that contained an ECC error.

Table 440: SDRAM Error Data (Low) Register

Offset: 0x00001444

Bit	Field	Type/InitVal	Description
31:0	ECCData_Low	RO 0x0	Sampled 32 low bits of the last data that contained an ECC error.

Table 441: SDRAM Received ECC Register

Offset: 0x00001448

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7:0	ECCReg	RO 0x0	ECC code read from SDRAM, and reported as an ECC error.

Table 442: SDRAM Calculated ECC Register

Offset: 0x0000144C

Bit	Field	Type/InitVal	Description
31:24	Reserved	RO 0x0	Reserved
23:8	EccRow	RO 0x0	Sampled Row[15:0] address of the latest latched data containing ECC error.
7:0	ECCCalc	RO 0x0	ECC code calculated by the DDR Controller on the data read from DRAM, and reported as containing an ECC error.

Table 443: SDRAM Error Address Register

Offset: 0x00001450

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25:23	EccBA	RO 0x0	Sampled Bank[2:0] address of the latest latched data containing ECC error.
22:8	EccCol	RO 0x0	Sampled Col[14:0] address of the latest latched data containing ECC error.
7:3	Reserved	RO 0x0	Reserved.
2:1	CS	RO 0x0	DRAM chip select Indicates in which of the four, physical DRAM CS the error occurred. Useful for software to reproduce the original address. 0 = CSn[0] 1 = CSn[1] 2 = CSn[2] 3 = CSn[3]
0	ErrType	RO 0x0	Indicates what type of ECC error is currently reported. 0 = Threshold expired: Single-bit ECC error threshold expired. 1 = Error detected: Double-bit ECC error detected.

Table 444: SDRAM Error Control Register

Offset: 0x00001454

Bit	Field	Type/InitVal	Description
31:24	Reserved	RO 0x0	Reserved
23:16	ThrEcc	RW 0x0	Threshold ECC Interrupt Number of single-bit ECC errors that occurred before the DDR Controller generated an interrupt. When the threshold is reached, the counter re-starts counting only when the Cause register has been cleared. **NOTE:** If set to 0x0, the DDR Controller does not generate an interrupt in the case of a single-bit error. The DDR Controller always generates an interrupt when it detects a double-bit ECC error. **NOTE:**
15:12	Reserved	RO 0x0	Reserved
11	DPPar	RW 0x0	DDR Controller data path parity select **NOTE:** Set to even parity for normal operation. Odd parity is for debug only. 0 = Even parity 1 = Odd parity
10	Reserved	RO 0x0	Reserved
9	PerrProp	RW 0x0	Propagate write data parity errors to DRAM according to following rules: ECC Enabled: DRAM controller generates an uncorrectable ECC error on write access to DRAM in the case of parity error indication from the originating interface and this bit is enabled. ECC Disabled: DRAM controller masks the data on write access to DRAM (by asserting the DM signals on the DRAM interface) in the case of parity error indication from the originating interface and this bit is enabled. 0 = Disable: Do not propagate Parity Error towards the DRAM. 1 = Enable: Propagate Parity Error towards the DRAM .
8	ForceEn	RW 0x0	Force user defined ECC byte on SDRAM writes. 0 = Normal: Write calculated ECC byte. 1 = Force: Write user-defined ECC byte.
7:0	ForceECC	RW 0x0	If <ForceEn> is set, user-defined ECC byte is written to the ECC bank.

Copyright © 2014 Marvell

Table 445: Single-Bit Error Counter Register

Offset: 0x00001458

Bit	Field	Type/InitVal	Description
31:0	SBE_Count	RW 0x0	Number of single bit ECC errors detected Counts single bit errors for all four banks [3:0]. If the number of detected errors reaches 2^32, it keeps this maximum value (no wrap to 0). Software must clear the counter by writing 0x0 to it.

Table 446: Double-Bit Error Counter Register

Offset: 0x0000145C

Bit	Field	Type/InitVal	Description
31:0	DBE_Count	RW 0x0	Number of double bit ECC errors detected in all four banks (CS[3:0]). If the number of detected errors reaches 2^32, it keeps this maximum value (no wrap to 0). Software must clear the counter by writing 0x0 to it.

Table 447: DDR Controller Error Interrupt Cause Register

Offset: 0x000014D0

Bit	Field	Type/InitVal	Description
NOTE: All bits are Read/Write Clear only. A cause bit sets upon event occurrence. A write of 0 clears the bit. A write of 1 has no affect.			
31:13	Reserved	RSVD 0x0	Reserved
12	NormalOpResume	RW0C 0x0	DRAM controller resumed normal operation. (Exited Auto Power Down / Auto Self Refresh modes). .
11	AutoPowDwnEntr	RW0C 0x0	DRAM controller entered Auto Power Down mode.
10	AutoSelfRefrEntr	RW0C 0x0	DRAM controller entered Auto Self Refresh mode.
9:4	Reserved	RSVD 0x0	Reserved
3	TrnErr	RW0C 0x0	Training error detected. Reports on errors revealed during the link training procedure.
2	DPErr	RW0C 0x0	DDR Controller internal data path parity error detected.
1	DBit	RW0C 0x0	Double bit ECC error detected.

Table 447: DDR Controller Error Interrupt Cause Register (Continued)

Offset: 0x000014D0

Bit	Field	Type/InitVal	Description
0	SBit	RW0C 0x0	Single bit ECC error threshold expired.

Table 448: DDR Controller Error Interrupt Mask Register

Offset: 0x000014D4

Bit	Field	Type/InitVal	Description
31:13	Reserved	RSVD 0x0	Reserved
12	NormalOpResume Mask	RW 0x0	DRAM controller normal operation resume interrupt mask. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
11	AutoPowDwnEntr Mask	RW 0x0	DRAM controller Auto Power Down mode entrance interrupt mask. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
10	AutoSelfRefrEntr Mask	RW 0x0	DRAM controller Auto Self Refresh mode entrance interrupt mask. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
9:4	Reserved	RSVD 0x0	Reserved
3	TrnErrIntMask	RW 0x0	Training error interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
2	DPErrIntMask	RW 0x0	DDR Controller internal data path parity error interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
1	DBitIntMask	RW 0x0	Double bit ECC error interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.

Table 448: DDR Controller Error Interrupt Mask Register (Continued)

Offset: 0x000014D4

Bit	Field	Type/InitVal	Description
0	SBitIntMask	RW 0x0	Single bit ECC error threshold expired interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.

Table 449: DDR Controller Message Interrupt Cause Register

Offset: 0x000014D8

Bit	Field	Type/InitVal	Description
NOTE: All bits are Read/Write Clear only. A cause bit sets upon event occurrence. A write of 0 clears the bit. A write of 1 has no affect.			
31:13	Reserved	RSVD 0x0	Reserved
12	NormalOpResume	RW0C 0x0	DRAM controller resumed normal operation. (Exited Auto Power Down / Auto Self Refresh modes). .
11	AutoPowDwnEntr	RW0C 0x0	DRAM controller entered Auto Power Down mode.
10	AutoSelfRefrEntr	RW0C 0x0	DRAM controller entered Auto Self Refresh mode.
9:4	Reserved	RSVD 0x0	Reserved
3	TrnErr	RW0C 0x0	Training error detected. Reports on errors revealed during the link training procedure.
2	DPErr	RW0C 0x0	DDR controller internal data path parity error detected.
1	DBit	RW0C 0x0	Double bit ECC error detected.
0	SBit	RW0C 0x0	Single bit ECC error threshold expired.

Table 450: DDR Controller Message Interrupt Mask Register

Offset: 0x000014DC

Bit	Field	Type/InitVal	Description
31:13	Reserved	RSVD 0x0	Reserved
12	NormalOpResume Mask	RW 0x0	DRAM controller normal operation resume interrupt mask. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
11	AutoPowDwnEntr Mask	RW 0x0	DRAM controller Auto Power Down mode entrance interrupt mask. 0 = Mask: Corresponding interrupt is masked 1 = Set: Corresponding interrupt is unmasked.
10	AutoSelfRefrEntr Mask	RW 0x0	DRAM controller Auto Self Refresh mode entrance interrupt mask. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
9:4	Reserved	RSVD 0x0	Reserved
3	TrnErrIntMask	RW 0x0	Training error interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
2	DPErrIntMask	RW 0x0	DDR Controller internal data path parity error interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
1	DBitIntMask	RW 0x0	Double bit ECC error interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.
0	SBitIntMask	RW 0x0	Single bit ECC error threshold expired interrupt mask. If a bit is set to 1, the corresponding event is unmasked. The mask does not affect setting of the Interrupt Cause register bits. It only affects assertion of an interrupt to the CPU core. 0 = Mask: Corresponding interrupt is masked. 1 = Set: Corresponding interrupt is unmasked.

A.3.2 SDRAM Control Registers

Table 451: SDRAM Configuration Register

Offset: 0x00001400

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved
30	DataPupRdRST	RW 0x1	DRAM Data PHY Read path reset. 0 = Enable 1 = Disable
29	DataPupWrRST	RW 0x1	DRAM Data PHY Write path reset. 0 = Enable 1 = Disable
28	PupRstDivider	RW 0x1	Manual override drive of the DRAM PHY's and ADLL's inputs. 0 = Enable: Reset 1 = Disable: Normal
27:26	Reserved	RO 0x0	Reserved
25	Reserved	RW 0x1	Reserved. Must be 0x1.
24	SRMode	RW 0x1	Self Refresh Mode 0 = Battery_backup: Once entered self refresh, exit only upon power on reset. 1 = Power_saving: Exit self refresh, when new DRAM access is requested.
23:20	Reserved	RO 0x0	Reserved
19	IErr	RW 0x0	Ignore ECC errors. 0 = Report: ECC errors are reported 1 = Ignore: ECC errors are ignored. However, RMW is still performed per partial writes to DRAM.
18	ECC	RW 0x0	Enable ECC When ECC is enabled, every partial write to the DRAM results in RMW. The DRAM controller writes an ECC byte to the DRAM along with the write data. 0 = None: ECC is disabled 1 = Support: ECC is enabled

Table 451: SDRAM Configuration Register (Continued)

Offset: 0x00001400

Bit	Field	Type/InitVal	Description
17	RegDIMM	RW 0x0	Enable Registered DIMM or Equivalent Sampling Logic **NOTE:** Even if on-board DRAM devices are used, this register can be used to buffer DRAM address/control signals. For that configuration, set the <RegDIMM> bit to 1. 0 = UnBuffered: Non buffered 1 = Registered: Registered DIMM or equivalent sampling logic
16	P2DWr	RW 0x0	CPU to Dram Write buffer policy **NOTE:** If any of the following device configurations are used, clear this bit: - CAS Latency = 3 0 = Store: Write buffer configure to store & forward mode 1 = Cut: Write buffer configure to cut through mode
15	DDRBusInUse	RW 0x1	DDR Data bus width in use 0 = Half: 32 bits Data bus 1 = Full: 64 bits Data bus
14	Reserved	RW 0x0	Reserved **NOTE:** Must be 0x1.
13:0	Refresh	RW 0x0400	Refresh interval count value. Defines the gap in clock-cycles between consecutive Refresh commands. Should be configured to (tREFI / (HCLK period))

Table 452: DDR Controller Control (Low) Register

Offset: 0x00001404

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved
30	Reserved	RW 0x0	Reserved - Must be 0x0.
29	Reserved	RW 0x1	Reserved - Must be 0x1.
28	Reserved	RW 0x1	Reserved - Must be 0x1.
27:20	Reserved	RSVD 0x63	Reserved

Table 452: DDR Controller Control (Low) Register (Continued)

Offset: 0x00001404

Bit	Field	Type/InitVal	Description
19	Reserved	RO 0x0	Reserved
18	Reserved	RSVD 0x0	Reserved
17:16	Reserved	RO 0x0	Reserved
15	Clk3Drv	RW 0x1	M_CLK_OUT[3] Drive **NOTE:** When not using M_CLK_OUT[3], set to high-z. 0 = Disable: M_CLK_OUT[3] and M_CLK_OUTn[3] are high-z. 1 = Enable: M_CLK_OUT[3] and M_CLK_OUTn[3] are driven normally.
14	DPDEn	RW 0x0	DRAM interface input buffers power down enable **NOTE:** Even when set to 0, input buffer is turned off during reset and during self refresh (whenever address/control pins output enables are in-active) 0 = Disable: Input buffer is always enabled (never powered down). 1 = Enable: Input buffer is turned off (power down) whenever there is no active read transaction.
13	Clk2Drv	RW 0x1	M_CLK_OUT[2] Drive **NOTE:** When not using M_CLK_OUT[2], set to high-z. 0 = Disable: M_CLK_OUT[2] and M_CLK_OUTn[2] are high-z. 1 = Enable: M_CLK_OUT[2] and M_CLK_OUTn[2] are driven normally.
12	Clk1Drv	RW 0x1	M_CLK_OUT[1] Drive **NOTE:** When not using M_CLK_OUT[1], set to high-z. 0 = Disable: M_CLK_OUT[1] and M_CLK_OUTn[1] are high-z. 1 = Enable: M_CLK_OUT[1] and M_CLK_OUTn[1] are driven normally.
11	Clk0Drv	RW 0x1	M_CLK_OUT[0] Drive **NOTE:** When not using M_CLK_OUT[0], set to high-z. 0 = Disable: M_CLK_OUT[0] and M_CLK_OUTn[0] are high-z. 1 = Enable: M_CLK_OUT[0] and M_CLK_OUTn[0] are driven normally.
10:7	Reserved	RO 0x0	Reserved
6	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 452: DDR Controller Control (Low) Register (Continued)

Offset: 0x00001404

Bit	Field	Type/InitVal	Description
5	SRClk	RW 0x0	Clock drive upon self refresh **NOTE:** If using clock buffers for distributing clock to the DRAM devices, the clock must be kept driven during Self Refresh mode unless these buffers can tolerate clock gating. 0 = ClkDrive In SR: Clock is kept driven during self refresh. 1 = ClkGate In SR: Clock is gated during self refresh.
4:3	2T	RW 0x0	2T/3T Mode Useful when interfacing heavy DRAM load (multiple DRAM physical banks) at high frequency. 0x3 is Reserved 0 = Single cycle: Address and command are driven for a single cycle 1 = Two cycles (2T): Address and command are driven for two cycles (2T) (qualified with M_CS that is still driven for one cycle) 2 = Three cycles (3T): Address and command are driven for three cycles (3T) (qualified with M_CS that is still driven for one cycle) Only for PHY to Unit clocking mode of 2:1
2:0	Reserved	RO 0x0	Reserved

Table 453: SDRAM Timing (Low) Register

Offset: 0x00001408

Bit	Field	Type/InitVal	Description
NOTE: The default values corresponds to working with DDR533 (4-4-4). Change these timing parameters according to DRAM type and operating frequency.			
31:28	tRTP	RW 0x1	Read Command to Precharge. Value 0 means one cycle, value 1 means two cycles, and so on.
27:24	tRRD	RW 0x1	Activate Bank A to Activate Bank B Value 0 means one cycle, value 1 means two cycles, and so on.
23:22	Reserved	RO 0x0	Reserved
21:20	tRAS High	RW 0x0	tRAS field extension bit. See bits[3:0] for tRAS field description.
19:16	tWTR	RW 0x1	Write Command to Read Command Value 0 means one cycle, value 1 means two cycles, and so on.
15:12	tWR	RW 0x3	Write Command to Precharge Value 0 means one cycle, value 1 means two cycles, and so on.

Table 453: SDRAM Timing (Low) Register (Continued)

Offset: 0x00001408

Bit	Field	Type/InitVal	Description
11:8	tRP	RW 0x3	Precharge Period (precharge to active) Value 0 means one cycle, value 1 means two cycles, and so on.
7:4	tRCD	RW 0x3	Activate to Command Value 0 means one cycle, value 1 means two cycles, and so on.
3:0	tRAS	RW 0xB	Minimum Row Active Time (active to precharge) Value 0 means one cycle, value 1 means two cycles, and so on. tRAS_High bit is an extension of this field (see bit 20 in this register).

Table 454: SDRAM Timing (High) Register

Offset: 0x0000140C

Bit	Field	Type/InitVal	Description
31:30	Reserved	RO 0x0	Reserved
29	AddRdSampDelta2tR2R	RW 0x1	Add Read Sample Delta to tR2R. When enabled, the tR2R timer is added with the delta between smallest and largest RdSamplDel value. (See register 0x1538 - "Read Data Sample Delays Register") When disabled, the software needs to add this value manually to the tR2R configuration. 0 = Disable 1 = Enable
28:25	tMOD	RW 0xb	tMOD (MRS command to Non-MRS command delay) value. Value 0 means one cycle, value 1 means two cycles, etc. **NOTE:** The maximal value for DDR3 Registered DRAM is 0xE. **NOTE:**
24:22	tR2W_W2R_High	RW 0x0	Higher 3 bits of Minimum Gap Between DRAM Read and Write Accesses.. This field is an extension of <tR2W_W2R> . This timing parameter is not part of the JEDEC standard. Used to prevent contention between read and write data driven from two different DRAM physical banks (same parameter for read after write and write after read).

Table 454: SDRAM Timing (High) Register (Continued)

Offset: 0x0000140C

Bit	Field	Type/InitVal	Description
21:19	tR2R_High	RW 0x0	Higher 3 bits of time between DRAM Read accesses to different DRAM physical banks (tR2R[4:2]). This field is an extension of <tR2R>. This timing parameter is not part of the JEDEC standard. It is used to prevent contention between read data drive from two different DRAM physical banks (DIMMs).
18:16	tRFC_High	RW 0x0	Higher 3 bits of Refresh Command Period (tRFC[9:7]). This field is an extension of tRFC[6:0] (bits [6:0] in this register). NOTE: For DDR3 - tRFC value must be greater than 0x18 (>=25).
15:11	tW2W	RW 0x0	Minimum Gap between Write to Write to different DRAM physical banks. The gap is between last data of first command to first data of consecutive command. Value 0 means no gap, value 1 means one cycle gap, and so on.
10:9	tR2W_W2R	RW 0x0	Low 2 bits of Minimum Gap Between DRAM Read and Write Accesses. The gap is between last data of first command to first data of consecutive command. tR2W_W2R_High field (bits[24:22] in this register) is an extension of this field. This timing parameter is not part of the JEDEC standard. Used to prevent contention between read and write data driven from two different devices (same parameter for read after write and write after read). Value 0 means one cycle, value 1 means two cycles, and so on.
8:7	tR2R	RW 0x0	Lower 2 bits of minimum gap between DRAM Read accesses to different DRAM physical banks (tR2R[1:0]) The gap is between last data of first command to first data of consecutive command. tR2R_High field (bits[21:19] in this register) is an extension of this field. This timing parameter is not part of the JEDEC standard. It is used to prevent contention between read data drive from two different DRAM physical banks (DIMMs). Value 0 means one cycle, value 1 means two cycles, and so on.
6:0	tRFC	RW 0x34	Lower 7 bits of Refresh Command Period (tRFC[6:0]). tRFC_High field (bits[18:16] in this register) is an extension of this field. Value 0 means one cycle, value 1 means two cycles, and so on. **NOTE:** For DDR3 - tRFC value must be greater than 0x18 (>=25).

Table 455: SDRAM Address Control Register
Offset: 0x00001410

Bit	Field	Type/InitVal	Description
31	Reserved	RO 0x0	Reserved
30:24	tFAW	RW 0x11	tFAW (Four Activate Window) DDR timing parameter value, in clock cycles. Value 0 means one cycle, value 1 means two cycles, etc.
23	CS3SingleDeviceSize_High	RW 0x0	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS3 DRAM Size is defined by this field along with the <CS3SingleDeviceSize> field (bits [15:14]). See the <CS3SingleDeviceSize> field description for configuration values.
22	CS2SingleDeviceSize_High	RW 0x0	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS2 DRAM Size is defined by this field along with the <CS2SingleDeviceSize> field (bits [11:10]). See the <CS2SingleDeviceSize> field description for configuration values.
21	CS1SingleDeviceSize_High	RW 0x0	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS1 DRAM Size is defined by this field along with the <CS1SingleDeviceSize> field (bits [7:6]). See the <CS1SingleDeviceSize> field description for configuration values.
20	CS0SingleDeviceSize_High	RW 0x0	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS0 DRAM Size is defined by this field along with the <CS0SingleDeviceSize> field (bits [3:2]). See the <CS0SingleDeviceSize> field description for configuration values.
19	CS3AddrSel	RW 0x0	Address Select Mode for CS3 Defines how the original master address is translated into DRAM address (Row, Col, BA). **NOTE:** See DDR SDRAM Addressing .
18	CS2AddrSel	RW 0x0	Address Select Mode for CS2 Defines how the original master address is translated into DRAM address (Row, Col, BA). **NOTE:** See DDR SDRAM Addressing.

Table 455: SDRAM Address Control Register (Continued)

Offset: 0x00001410

Bit	Field	Type/InitVal	Description
17	CS1AddrSel	RW 0x0	Address Select Mode for CS1 Defines how the original master address is translated into DRAM address (Row, Col, BA). **NOTE:** See DDR SDRAM Addressing.
16	CS0AddrSel	RW 0x0	Address Select Mode for CS0 Defines how the original master address is translated into DRAM address (Row, Col, BA). **NOTE:** See DDR SDRAM Addressing.
15:14	CS3SingleDeviceSize	RW 0x1	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS3 Dram size is defined by the <CS3SingleDeviceSize_High> and <CS3SingleDeviceSize> fields, bits{[23],[15:14]} as follows: 0 = 2Gbit: DDR2: 2 Gb DDR3: 2 Gb 1 = 256Mbit: Reserved. 2 = 512Mbit: DDR2: 512 Mb DDR3: 512 Mb 3 = 1Gbit: DDR2: 1 Gb DDR3: 1 Gb 4 = 4Gbit: DDR2: 4 Gb DDR3: 4 Gb 5 = 8Gbit: DDR2: Reserved DDR3: 8 Gb
13:12	CS3Struct	RW 0x0	SDRAM Device structure. Affects DRAM row and column address bits multiplexing. **NOTE:** See DDR SDRAM Address Multiplex. 0 = X8: The CS is assembled from X8 devices 1 = X16: The CS is assembled from X16 devices
11:10	CS2SingleDeviceSize	RW 0x1	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS2 DRAM size is defined by the <CS2SingleDeviceSize_High> and <CS2SingleDeviceSize> fields, bits{[22],[11:10]} as follows: 0 = 2Gbit: DDR2: 2 Gb DDR3: 2 Gb 1 = 256Mbit: Reserved. 2 = 512Mbit: DDR2: 512 Mb DDR3: 512 Mb 3 = 1Gbit: DDR2: 1 Gb DDR3: 1 Gb 4 = 4Gbit: DDR2: 4 Gb DDR3: 4 Gb 5 = 8Gbit: DDR2: Reserved DDR3: 8 Gb
9:8	CS2Struct	RW 0x0	SDRAM Device structure. Affects DRAM row and column address bits multiplexing. **NOTE:** See DDR SDRAM Address Multiplex. 0 = X8: The CS is assembled from X8 devices 1 = X16: The CS is assembled from X16 devices

Copyright © 2014 Marvell

Table 455: SDRAM Address Control Register (Continued)

Offset: 0x00001410

Bit	Field	Type/InitVal	Description
7:6	CS1SingleDeviceSize	RW 0x1	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS1 DRAM size is defined by the <CS1SingleDeviceSize_High> and <CS1SingleDeviceSize> fields, bits{[21],[7:6]} as follows: 0 = 2Gbit: DDR2: 2 Gb DDR3: 2 Gb 1 = 256Mbit: Reserved. 2 = 512Mbit: DDR2: 512 Mb DDR3: 512 Mb 3 = 1Gbit: DDR2: 1 Gb DDR3: 1 Gb 4 = 4Gbit: DDR2: 4 Gb DDR3: 4 Gb 5 = 8Gbit: DDR2: Reserved DDR3: 8 Gb
5:4	CS1Struct	RW 0x0	SDRAM Device structure. Affects DRAM row and column address bits multiplexing. **NOTE:** See DDR SDRAM Address Multiplex. 0 = X8: The CS is assembled from X8 devices 1 = X16: The CS is assembled from X16 devices
3:2	CS0SingleDeviceSize	RW 0x1	SDRAM Device Configuration Affects DRAM row and column address bits multiplexing. See DDR SDRAM Addressing. CS0 DRAM size is defined by the <CS0SingleDeviceSize_High> and <CS0SingleDeviceSize> fields, bits{[20],[3:2]} as follows: 0 = 2Gbit: DDR2: 2 Gb DDR3: 2 Gb 1 = 256Mbit: Reserved. 2 = 512Mbit: DDR2: 512 Mb DDR3: 512 Mb 3 = 1Gbit: DDR2: 1 Gb DDR3: 1 Gb 4 = 4Gbit: DDR2: 4 Gb DDR3: 4 Gb 5 = 8Gbit: DDR2: Reserved DDR3: 8 Gb
1:0	CS0Struct	RW 0x0	SDRAM Device structure. Affects DRAM row and column address bits multiplexing. **NOTE:** See DDR SDRAM Address Multiplex. 0 = X8: The CS is assembled from X8 devices. 1 = X16: The CS is assembled from X16 devices.

Table 456: SDRAM Open Pages Control Register

Offset: 0x00001414

Bit	Field	Type/InitVal	Description
31:10	Reserved	RO 0x0	Reserved.
9:8	Reserved	RSVD 0x3	Reserved
7:2	Reserved	RW 0x0	Reserved.

Table 456: SDRAM Open Pages Control Register (Continued)

Offset: 0x00001414

Bit	Field	Type/InitVal	Description
1	Reserved	RSVD 0x0	Reserved
0	OPEn	RW 0x0	Open Page Enable 0 = Open: DDR Controller keeps the corresponding bank page open whenever it can. 1 = Close: DDR Controller always closes the page at the end of the transaction.

Table 457: SDRAM Operation Register

Offset: 0x00001418

Bit	Field	Type/InitVal	Description
31:25	Reserved	RO 0x0	Reserved.
24	CWA Delay Sel	RW 0x0	CWA command delay select. Defines the delay after a CWA command. **NOTE:** Select tSTAB for configurations that affect the clock timing. 0 = tMRD: tMRD as defined by JEDEC JESD82-29A 1 = tSTAB: tSTAB as defined by JEDEC JESD82-29A
23:20	CWA Data	RW 0x0	DDR3 register control word data. Relevant for the CWA command only. Defines the CWA command RC Data ({M_BA[1], M_BA[0], M_A[4], M_A[3]}).
19:16	CWA RC	RW 0x0	DDR3 register control word number (RC0-RC15) Relevant for the CWA command only. Defines the CWA command RC number ({M_BA[2], M_A[2], M_A[1], M_A[0]}).
15:12	Dram Controller Status	RO 0x0	This field represents the status of the memory controller, thus can be used by SW to monitor the DRAM status (eg. Self refresh or power down). 0 = Normal: The DRAM is at normal operation mode 1 = Power Down: The DRAM is at Power Down mode 2 = Self Refresh: The DRAM is at Self Refresh mode 3 = Vtt powering up: The DRAM controller is performing Self Refresh exit procedure, and wait for the VTT termination power regulator to restore stable power state. 4 = Init: The DRAM controller is performing Initialization sequence on the DRAM

Copyright © 2014 Marvell

Table 457: SDRAM Operation Register (Continued)

Offset: 0x00001418

Bit	Field	Type/InitVal	Description
11	Cmd_CS3	RW 0x0	Define if the Operation command will be applied to CS3 or not. (Active Low) This bit only influences following commands: MR0, MR1, MR2, MR3, ZQCL, ZQCS, CWA and NOP. All other commands are always targeted to all CSs. 0 = Enable: M_CSn[3] is active during Operation Command execution. 1 = Disable: M_CSn[3] is not active during Operation Command execution.
10	Cmd_CS2	RW 0x0	Define if the Operation command will be applied to CS2 or not. (Active Low) This bit only influences following commands: MR0, MR1, MR2, MR3, ZQCL, ZQCS, CWA and NOP. All other commands are always targeted to all CSs. 0 = Enable: M_CSn[2] is active during Operation Command execution. 1 = Disable: M_CSn[2] is not active during Operation Command execution.
9	Cmd_CS1	RW 0x0	Define if the Operation command will be applied to CS1 or not. (Active Low) This bit only influences following commands: MR0, MR1, MR2, MR3, ZQCL, ZQCS, CWA and NOP. All other commands are always targeted to all CSs. 0 = Enable: M_CSn[1] is active during Operation Command execution. 1 = Disable: M_CSn[1] is not active during Operation Command execution.
8	Cmd_CS0	RW 0x0	Define if the Operation command will be applied to CS0 or not. (Active Low) This bit only influences following commands: MR0, MR1, MR2, MR3, ZQCL, ZQCS, CWA and NOP. All other commands are always targeted to all CSs. 0 = Enable: M_CSn[0] is active during Operation Command execution. 1 = Disable: M_CSn[0] is not active during Operation Command execution.
7:5	Reserved	RW 0x0	Reserved.

Table 457: SDRAM Operation Register (Continued)

Offset: 0x00001418

Bit	Field	Type/InitVal	Description
4:0	Cmd	SC 0x0	DRAM Mode Select Setting Cmd results in DDR Controller execution of the required command to the DRAM. NOTES: 1. In case of one of following commands - MR0, MR1, MR2, MR3, ZQCL, ZQCS, CWA and NOP, the command target CS is defined by Cmd_CS#i, see bits [11:8] for details. All active CSs must be of the same address-mirroring structure - i.e. all mirrored or all not mirrored.(DDR3). DDR3 Rank control register define each CS mirroring structure. All other commands are always targeted to all CSs. 2. ZQ command must be performed to one chip select (CS) at a time. Therefore, in ZQCL and ZQCS commands, only one of Cmd_CS<i> (i= 0 to 3) fields may be active. (Bits [11:8]). 3. CWA command is specified for configuring the DDR3 register module. (JEDEC SSTE32882) The CWA command must be perfromed to 2 or 4 CSs at a time. (DDR3 register control word access definition). It is the responsibility of the user to activate the relevant Cmd_CS<i> (i= 0 to 3) fields (bits [11:8]) according to the Registered DDR3 topology. The CWA RC number, RC Data and RC delay - are defined by bits [24:16] in this register. The DDR Controller then resets Cmd to the default 0x0 value. 0x0 = Normal: Normal SDRAM Mode, not generating any special command . 0x1 = Precharge: Precharge all banks command. 0x2 = Refresh: Refresh all banks command 0x3 = MR0: MR0 command. 0x4 = MR1: MR1 command. 0x5 = NOP: NOP command 0x7 = SelfRefresh: Enter self refresh command 0x8 = MR2: MR2 command. 0x9 = MR3: MR3 command. 0xA = ACT_PDE: Active Power Down Entry command (Power Down Entry while Not all Banks are closed) 0xB = PRE_PDE: Precharge Power Down Entry command (Power Down Entry while all Banks are closed) 0xC = ZQCL: ZQ calibration Long command 0xD = ZQCS: ZQ calibration Short command. 0xE = CWA: DR3 Register Control Word Access. (RC) Valid only in case of Registered DDR3. Reserved configuration for unbuffered DDR3.

Table 458: DDR Controller Control (High) Register

Offset: 0x00001424

Bit	Field	Type/InitVal	Description
31:25	Reserved	RO 0x0	Reserved
24	Reserved	RW 0x1	Reserved - must be set to 0x0
23:22	Reserved	RSVD 0x1	Reserved
21	Reserved	RW 0x1	Reserved - Must be 0x1
20:12	Reserved	RSVD 0xF	Reserved
11	WrMeshDelay	RW 0x1	Add 1/4 cc delay Set this bit when AddrCntrlToClkSkew (0x15ec[5:4]) is “Zero”, otherwise this bit should be 0. 0 = Disable: No Delay. 1 = Enable: Add 1/4 cc delay.
10	Reserved	RW 0x0	Reserved must be 0x0.
9	Reserved	RW 0x1	Reserved
8	Reserved	RW 0x0	Reserved Must be 0x1.
7	Reserved	RW 0x0	Reserved **NOTE:** Must be 0x1.
6:4	Reserved	RSVD 0x7	Reserved
3	CPU_Interjection	RW 0x1	CPU Interjection Enables chop of a long Mbus access to DRAM to several accesses (giving the CPU a chance to interject in the middle and access the DRAM). The chop boundary is 32 bytes. 0 = Split: Enable CPU interjection 1 = SplitBypass: Disable CPU interjection
2:0	Reserved	RW 0x7	Reserved - Must be 0x7.

Table 459: DDR ODT Timing (Low) Register

Offset: 0x00001428

Bit	Field	Type/InitVal	Description
NOTE: The default values corresponds to working with DDR533 (4-4-4). Change these timing parameters according to DRAM type and operating frequency.			
31:22	Reserved	RO 0x0	Reserved.
21	tODT_OFF_RD_ High	RW 0x0	Extension Bit[4] of tODT_OFF_RD field. See tODT_OFF_RD field description.
20:16	tODT_OFF_CTL_ RD	RW 0x7	Number of cycles from Read command to de-assertion of the internal ODT signal to the DRAM controller I/O buffer. In DDR3 mode: Set to CL + 6.
15:12	tODT_ON_CTL_RD	RW 0x4	Number of cycles from Read command to assertion of the internal ODT signal to the DRAM controller I/O buffer. Set to CL -1.
11:8	tODT_OFF_RD	RW 0x4	Number of cycles from Read command to de-assertion of the M_ODT signal. tODT_OFF_RD has extension bit tODT_OFF_RD_High (bit [21] in this register) In DDR3 mode: Set to CL - CWL + 6.
7:4	tODT_ON_RD	RW 0x1	Number of cycles from Read command to assertion of the M_ODT signal. Value 0 means same cycle as read command, value 1 means one cycle later, etc. In DDR3 mode: Set to CL -CWL + 1.
3:0	Reserved	RO 0x0	Reserved

Table 460: DDR3 Timing Register

Offset: 0x0000142C

Bit	Field	Type/InitVal	Description
31:30	Reserved	RO 0x0	Reserved.
29	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 460: DDR3 Timing Register (Continued)

Offset: 0x0000142C

Bit	Field	Type/InitVal	Description
28:21	tWLDELAY	RW 0x0	tWLO + tWLOE + board trace delay. Where board trace delay(ns) = (max clock trace length (inch) + max DQS trace length(inch0)*180(ps/inch) /1000. The minimal value is 0x0A. Value 0 means one cycle, value 1 means two cycles, etc.
20:15	Reserved	RW 0x18	Reserved.
14:9	tWLMRD	RW 0x27	Time, in cycles, between WL entry and first DQS pulse. Value 0 means one cycle, value 1 means two cycles, etc.
8:4	tXPDLL	RW 0x13	Power down exit time, in cycles. Value 0 means one cycle, value 1 means two cycles, etc.
3:0	tPD	RW 0x8	Time (in Refresh periods) from PD (Power down) entry to PD exit. **NOTE:** If a Read, Write, or OP command was issued, the PD exit will be performed immediately regardless of the tPD value. tPD must not be configured to 0x0. The refresh interval is set by the <RefresH> field in the SDRAM Configuration register.

Table 461: SDRAM Interface Mbus Control (Low) Register

Offset: 0x00001430

Bit	Field	Type/InitVal	Description
31:28	Arb7	RW 0x7	Pizza Arbiter slice 7 unit ID
27:24	Arb6	RW 0x6	Pizza Arbiter slice 6 unit ID.
23:20	Arb5	RW 0x5	Pizza Arbiter slice 5 unit ID.
19:16	Arb4	RW 0x4	Pizza Arbiter slice 4 unit ID.
15:12	Arb3	RW 0x3	Pizza Arbiter slice 3 unit ID.
11:8	Arb2	RW 0x2	Pizza Arbiter slice 2 unit ID.
7:4	Arb1	RW 0x1	Pizza Arbiter slice 1 unit ID.
3:0	Arb0	RW 0x0	Pizza Arbiter slice 0 unit ID.

Table 462: SDRAM Interface Mbus Control (High) Register

Offset: 0x00001434

Bit	Field	Type/InitVal	Description
31:28	Arb15	RW 0xF	Pizza Arbiter slice 15 unit ID.
27:24	Arb14	RW 0xE	Pizza Arbiter slice 14 unit ID.
23:20	Arb13	RW 0xD	Pizza Arbiter slice 13 unit ID.
19:16	Arb12	RW 0xC	Pizza Arbiter slice 12 unit ID.
15:12	Arb11	RW 0xB	Pizza Arbiter slice 11 unit ID.
11:8	Arb10	RW 0xA	Pizza Arbiter slice 10 unit ID.
7:4	Arb9	RW 0x9	Pizza Arbiter slice 9 unit ID.
3:0	Arb8	RW 0x8	Pizza Arbiter slice 8 unit ID.

Table 463: SDRAM Interface Mbus Timeout Register

Offset: 0x00001438

Bit	Field	Type/InitVal	Description
31:18	Reserved	RSVD 0x0	Reserved
17	Reserved	RW 0x0	Reserved
16	TimeoutEn	RW 0x1	Mbus Arbiter Timer enable 0 = Enable: Timer is enabled 1 = Disable: Timer is disabled
15:8	Reserved	RO 0x0	Reserved
7:0	Timeout	RW 0xFF	Mbus Arbiter Timeout Preset Value **NOTE:** Must keep value of 0xFF as long as Timeout counter is disabled. **NOTE:**

Table 464: VTT Control Register

Offset: 0x00001470

Bit	Field	Type/InitVal	Description
31:28	Reserved	RW 0x0	Reserved, Set to 0

Table 464: VTT Control Register (Continued)

Offset: 0x00001470

Bit	Field	Type/InitVal	Description
27:16	M_VTT_CTRL power off counter	RW 0x1	The minimal duration of time between power off (de-assertion) of the external VTT regulator to the power on (assertion) state. Number of 1k DRAM cycles from the de-assertion of M_VTT_CTRL signal to re-assertion of this signal. Setting this bit to 0 means no delay. Setting this bit to 1 means 1K cycles, 2 means 2K cycles, etc. Set this counter to a time duration that will gurantee that the VTT regulator has reached power off state, before having the option of taking it out of that state again.
15:4	M_VTT_CTRL Power up counter	RW 0x320	Number of 1k DRAM cycles from the assertion of M_VTT_CTRL output signal to a stable power out from the external VTT power regulator. Setting this bit to 0 means zero delay. setting this bit to 1 means 1K DRAM cycles, 2 means 2K DRAM cycles, etc. Set this counter according to the power restore time specification of the VTT regulator in usage.
3	Reserved	RW 0x0	Reserved Must be 0x0.
2	Reserved	RW 0x1	Reserved **NOTE:** Must be 0x1.
1	M_VTT_CTRL polarity	RW 0x0	Set this bit to produce an active-high M_VTT_CTRL signal. 0 = Active_Low: M_VTT_CTRL signal is active low (A value of 0 means power up). 1 = Active_High: M_VTT_CTRL signal is active high (A value of 1 means power up).
0	VTT CTRL Enable	RW 0x0	Enable the driving of an enable control signal to the external DRAM address/command VTT power regulator (M_VTT_CTRL) 0 = Disable: VTT control feature is disabled. M_VTT signal value equal to M_VTT_CTRL Polarity bit value. 1 = Enable: VTT control feature is enabled., VTT power is controlled by the memory controller according to this register configuration.

Table 465: SDRAM Auto Power Save Register

Offset: 0x00001474

Bit	Field	Type/InitVal	Description
31:18	Reserved	RSVD 0x0	Reserved

Table 465: SDRAM Auto Power Save Register (Continued)

Offset: 0x00001474

Bit	Field	Type/InitVal	Description
17:12	IdleTH	RW 0x0	Number of transactions allowed within the PowerSaveCnt window. When the number of transactions received by the DRAM controller exceeds this threshold, the AutoPowerSaveCnt counter will reset. Number of 0 means one transaction, 1 means two transactions etc.
11:2	AutoPowerSaveCnt	RW 0xc3	Counts the number of DRAM clock cycles (in 1024 cycles chunks) after which the DRAM will enter Power Save or Self Refresh mode, according to AutoPowerSaveOption configuration. 0 - 1024 cycles, 1 - 2048 cycles, etc.
1:0	AutoPowerSaveOption	RW 0x0	set these bits to allow automatic DRAM Power Down or Self Refresh entry 0 = Disable: Automatic Power Save feature is disabled 1 = AutoPD: When counter expired, go into Power Down mode 2 = AutoSR: When counter expired go into Self Refresh mode 3 = AutoPDSR: When counter expired go to Power Down mode, when it expired again go out of Power Down mode and get into Self Refresh mode.

Table 466: DDR ODT Timing (High) Register

Offset: 0x0000147C

Bit	Field	Type/InitVal	Description
NOTE: The default values corresponds to working with DDR533 (4-4-4). Change these timing parameters according to DRAM type and operating frequency.			
31:18	Reserved	RO 0x0	Reserved
17	tODT_OFF_WR_High	RW 0x0	Extension bIt[4] of tODT_OFF_WR. See tODT_OFF_WR field description.
16:12	tODT_OFF_CTL_WR	RW 0x7	Number of cycles from Write command to de-assertion of the internal ODT signal to the DRAM controller I/O buffer. In DDR3 mode: Set to CWL + 5.
11:8	tODT_ON_CTL_WR	RW 0x4	Number of cycles from Write command to assertion of the internal ODT signal to the DRAM controller I/O buffer. In DDR3 mode: Set to CWL - 1.

Table 466: DDR ODT Timing (High) Register (Continued)

Offset: 0x0000147C

Bit	Field	Type/InitVal	Description
7:4	tODT_OFF_WR	RW 0x4	Number of cycles from Write command to de-assertion of the M_ODT signal. tODT_OFF_WR has extension bit - tODT_OFF_WR_High (bit[17] in this register) In DDR3 mode: Set to 0x7.
3:0	tODT_ON_WR	RW 0x1	Number of cycles from the Write command to assertion of the M_ODT signal. A value of 0 means one cycle before write command, a value of 1 means the same cycle as write command, a value of 2 means one cycle later ,etc. In DDR3 mode: Set to 0x1.

Table 467: SDRAM Initialization Control Register

Offset: 0x00001480

Bit	Field	Type/InitVal	Description
31:4	Reserved	RO 0x0	Reserved
3	Init RESET Deassert	SC 0x0	De-assert M_RESETn and wait 500 us. DRAM controller clears this bit once M_RESETn is de-asserted and 500 us have passed. **NOTE:** Use only for DDR3 RegDRAM . Use this bit to preconfigure a SSTE32882 device prior to starting the DDR3 Init Sequence (by setting bit InitEn). Set this bit only after configuring all DRAM and controller timing parameters. 0 = Disable 1 = Enable
2	Init CKE Assert	SC 0x0	De-assert M_RESETn and assert M_CKE. DRAM controller clears this bit once CKE is asserted. **NOTE:** Use only for DDR3 RegDRAM. Use this bit to preconfigure a SSTE32882 device prior to starting the DDR3 Init Sequence (by setting bit InitEn). Set this bit only after configuring all DRAM and controller timing parameters. 0 = Disable 1 = Enable
1	Reserved	RW 0x0	Reserved must be 0x0.

Table 467: SDRAM Initialization Control Register (Continued)

Offset: 0x00001480

Bit	Field	Type/InitVal	Description
0	InitEn	SC 0x0	Initialization Enable DRAM initialization sequence starts upon setting <InitEn >to 1. DRAM controller clears the bit upon initialization completion. **NOTE:** Set this bit only after configuring all DRAM and Controller timing parameters. 0 = Disable 1 = Enable: Initialization Enable

Table 468: AXI Mbus Arbiter Configuration Register

Offset: 0x00001488

Bit	Field	Type/InitVal	Description
31:14	Reserved	RSVD 0x0	Reserved
13:12	AxiCpuArbStep	RW 0x0	Setting this field sets the throughput of the two ports of the AXI-Mbus arbiter. 0 = Step 0 1 = Step 1 2 = Step 2 3 = Step 3
11:8	QafHighThr	RW 0x5	When number of valid transactions in the MBUS Q has reached the high threshold the MBUS Q drives the MbusQafPrio priority to the AXI-MBUS Arbiter, when number of transaction goes below the low threshold the MBUS Q drives the normal priority according to its current transactions.
7:4	QafLowThr	RW 0x4	When number of valid transactions in the Mbus Q has reached the high threshold the Mbus Q drives the <MbusQafPrio> field priority to the AXI-Mbus Arbiter, when the number of transaction goes below the low threshold the Mbus Q drives the normal priority according to its current transactions.
3:2	Reserved	RSVD 0x0	Reserved
1:0	MbusQafPrio	RW 0x2	This field represent the priority of the Mbus Q when it has reached the QAF (Queue Almost Full) threshold. 0 = Low: Low priority 1 = Med: Medium Priority 2 = High: High Priority 3 = Top: Top Priority

Table 469: SDRAM ODT Control (Low) Register

Offset: 0x00001494

Bit	Field	Type/InitVal	Description
31	ODT3_WRITE_CS3	RW 0x0	M_ODT[3] control for write transactions to DRAM CS3 0 = OFF: M_ODT[3] is de-asserted during write to DRAM CS3 1 = Active: M_ODT[3] is asserted during write to DRAM CS3
30	ODT3_WRITE_CS2	RW 0x0	M_ODT[3] control for write transactions to DRAM CS2 0 = OFF: M_ODT[3] is de-asserted during write to DRAM CS2. 1 = Active: M_ODT[3] is asserted during write to DRAM CS2
29	ODT3_WRITE_CS1	RW 0x0	M_ODT[3] control for write transactions to DRAM CS1 0 = OFF: M_ODT[3] is de-asserted during write to DRAM CS1. 1 = Active: M_ODT[3] is asserted during write to DRAM CS1.
28	ODT3_WRITE_CS0	RW 0x0	M_ODT[3] control for write transactions to DRAM CS0 0 = OFF: M_ODT[3] is de-asserted during write to DRAM CS0. 1 = Active: M_ODT[3] is asserted during write to DRAM CS0.
27	ODT2_WRITE_CS3	RW 0x0	M_ODT[2] control for write transactions to DRAM CS3 0 = OFF: M_ODT[2] is de-asserted during write to DRAM CS3. 1 = Active: M_ODT[2] is asserted during write to DRAM CS3.
26	ODT2_WRITE_CS2	RW 0x0	M_ODT[2] control for write transactions to DRAM CS2 0 = OFF: M_ODT[2] is de-asserted during write to DRAM CS2. 1 = Active: M_ODT[2] is asserted during write to DRAM CS2.
25	ODT2_WRITE_CS1	RW 0x0	M_ODT[2] control for write transactions to DRAM CS1 0 = OFF: M_ODT[2] is de-asserted during write to DRAM CS1. 1 = Active: M_ODT[2] is asserted during write to DRAM CS1.
24	ODT2_WRITE_CS0	RW 0x0	M_ODT[2] control for write transactions to DRAM CS0 0 = OFF: M_ODT[2] is de-asserted during write to DRAM CS0. 1 = Active: M_ODT[2] is asserted during write to DRAM CS0.
23	ODT1_WRITE_CS3	RW 0x0	M_ODT[1] control for write transactions to DRAM CS3 0 = OFF: M_ODT[1] is de-asserted during write to DRAM CS3. 1 = Active: M_ODT[1] is asserted during write to DRAM CS3.

Table 469: SDRAM ODT Control (Low) Register (Continued)
Offset: 0x00001494

Bit	Field	Type/InitVal	Description
22	ODT1_WRITE_CS2	RW 0x0	M_ODT[1] control for write transactions to DRAM CS2 0 = OFF: M_ODT[1] is de-asserted during write to DRAM CS2. 1 = Active: M_ODT[1] is asserted during write to DRAM CS2.
21	ODT1_WRITE_CS1	RW 0x0	M_ODT[1] control for write transactions to DRAM CS1 0 = OFF: M_ODT[1] is de-asserted during write to DRAM CS1. 1 = Active: M_ODT[1] is asserted during write to DRAM CS1.
20	ODT1_WRITE_CS0	RW 0x0	M_ODT[1] control for write transactions to DRAM CS0 0 = OFF: M_ODT[1] is de-asserted during write to DRAM CS0. 1 = Active: M_ODT[1] is asserted during write to DRAM CS0.
19	ODT0_WRITE_CS3	RW 0x0	M_ODT[0] control for write transactions to DRAM CS3 0 = OFF: M_ODT[0] is de-asserted during write to DRAM CS3. 1 = Active: M_ODT[0] is asserted during write to DRAM CS3.
18	ODT0_WRITE_CS2	RW 0x0	M_ODT[0] control for write transactions to DRAM CS2 0 = OFF: M_ODT[0] is de-asserted during write to DRAM CS2 1 = Active: M_ODT[0] is asserted during write to DRAM CS2
17	ODT0_WRITE_CS1	RW 0x0	M_ODT[0] control for write transactions to DRAM CS1 0 = OFF: M_ODT[0] is de-asserted during write to DRAM CS1. 1 = Active: M_ODT[0] is asserted during write to DRAM CS1.
16	ODT0_WRITE_CS0	RW 0x0	M_ODT[0] control for write transactions to DRAM CS0 0 = OFF: M_ODT[0] is de-asserted during write to DRAM CS0. 1 = Active: M_ODT[0] is asserted during write to DRAM CS0.
15	ODT3_READ_CS3	RW 0x0	M_ODT[3] control for read transactions from DRAM CS3 **NOTE:** This bit must be 0 in case of DDR3. (DDR3 protocol prohibits ODT assertion during Read response) 0 = OFF: M_ODT[3] is de-asserted during read from DRAM CS3. 1 = Active: M_ODT[3] is asserted during read from DRAM CS3.

Table 469: SDRAM ODT Control (Low) Register (Continued)
Offset: 0x00001494

Bit	Field	Type/InitVal	Description
14	ODT3_READ_CS2	RW 0x0	M_ODT[3] control for read transactions from DRAM CS2 0 = OFF: M_ODT[3] is de-asserted during read from DRAM CS2. 1 = Active: M_ODT[3] is asserted during read from DRAM CS2.
13	ODT3_READ_CS1	RW 0x0	M_ODT[3] control for read transactions from DRAM CS1 0 = OFF: M_ODT[3] is de-asserted during read from DRAM CS1. 1 = Active: M_ODT[3] is asserted during read from DRAM CS1.
12	ODT3_READ_CS0	RW 0x0	M_ODT[3] control for read transactions from DRAM CS0 0 = OFF: M_ODT[3] is de-asserted during read from DRAM CS0. 1 = Active: M_ODT[3] is asserted during read from DRAM CS0.
11	ODT2_READ_CS3	RW 0x0	M_ODT[2] control for read transactions from DRAM CS3 0 = OFF: M_ODT[2] is de-asserted during read from DRAM CS3. 1 = Active: M_ODT[2] is asserted during read from DRAM CS3.
10	ODT2_READ_CS2	RW 0x0	M_ODT[2] control for read transactions from DRAM CS2 **NOTE:** This bit must be 0 in case of DDR3. (DDR3 protocol prohibits ODT assertion during Read response) 0 = OFF: M_ODT[2] is de-asserted during read from DRAM CS2. 1 = Active: M_ODT[2] is asserted during read from DRAM CS2.
9	ODT2_READ_CS1	RW 0x0	M_ODT[2] control for read transactions from DRAM CS1 0 = OFF: M_ODT[2] is de-asserted during read from DRAM CS1. 1 = Active: M_ODT[2] is asserted during read from DRAM CS1.
8	ODT2_READ_CS0	RW 0x0	M_ODT[2] control for read transactions from DRAM CS0 0 = OFF: M_ODT[2] is de-asserted during read from DRAM CS0. 1 = Active: M_ODT[2] is asserted during read from DRAM CS0.
7	ODT1_READ_CS3	RW 0x0	M_ODT[1] control for read transactions from DRAM CS3 0 = OFF: M_ODT[1] is de-asserted during read from DRAM CS3. 1 = Active: M_ODT[1] is asserted during read from DRAM CS3.

Table 469: SDRAM ODT Control (Low) Register (Continued)

Offset: 0x00001494

Bit	Field	Type/InitVal	Description
6	ODT1_READ_CS2	RW 0x0	M_ODT[1] control for read transactions from DRAM CS2 0 = OFF: M_ODT[1] is de-asserted during read from DRAM CS2. 1 = Active: M_ODT[1] is asserted during read from DRAM CS2.
5	ODT1_READ_CS1	RW 0x0	M_ODT[1] control for read transactions from DRAM CS1 **NOTE:** This bit must be 0 in case of DDR3. (DDR3 protocol prohibits ODT assertion during Read response) 0 = OFF: M_ODT[1] is de-asserted during read from DRAM CS1. 1 = Active: M_ODT[1] is asserted during read from DRAM CS1.
4	ODT1_READ_CS0	RW 0x0	M_ODT[1] control for read transactions from DRAM CS0 0 = OFF: M_ODT[1] is de-asserted during read from DRAM CS0. 1 = Active: M_ODT[1] is asserted during read from DRAM CS0.
3	ODT0_READ_CS3	RW 0x0	M_ODT[0] control for read transactions from DRAM CS3 0 = OFF: M_ODT[0] is de-asserted during read from DRAM CS3. 1 = Active: M_ODT[0] is asserted during read from DRAM CS3.
2	ODT0_READ_CS2	RW 0x0	M_ODT[0] control for read transactions from DRAM CS2 0 = OFF: M_ODT[0] is de-asserted during read from DRAM CS2. 1 = Active: M_ODT[0] is asserted during read from DRAM CS2.
1	ODT0_READ_CS1	RW 0x0	M_ODT[0] control for read transactions from DRAM CS1 0 = OFF: M_ODT[0] is de-asserted during read from DRAM CS1. 1 = Active: M_ODT[0] is asserted during read from DRAM CS1.
0	ODT0_READ_CS0	RW 0x0	M_ODT[0] control for read transactions from DRAM CS0 **NOTE:** This bit must be 0 in case of DDR3. (DDR3 protocol prohibits ODT assertion during Read response) 0 = OFF: M_ODT[0] is de-asserted during read from DRAM CS0. 1 = Active: M_ODT[0] is asserted during read from DRAM CS0.

Copyright © 2014 Marvell

Table 470: SDRAM ODT Control (High) Register

Offset: 0x00001498

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7	ODT3_OVRD_val	RW 0x0	M_ODT[3] assertion control - valid only when ODT3_OVRD bit is 1. 0 = OFF: M_ODT[3] is never active. 1 = Active: M_ODT[3] is always active.
6	ODT3_OVRD	RW 0x0	M_ODT[3] override control 0 = Normal: M_ODT[3] assertion/de-assertion is controlled by SDRAM ODT Control Low register. 1 = Ovrd: M_ODT[3] assertion/de-assertion is controlled by ODT3_ OVRD_val control bit.
5	ODT2_OVRD_val	RW 0x0	M_ODT[2] assertion control - valid only when ODT2_OVRD bit is 1. 0 = OFF: M_ODT[2] is never active. 1 = Active: M_ODT[2] is always active.
4	ODT2_OVRD	RW 0x0	M_ODT[2] override control 0 = Normal: M_ODT[2] assertion/de-assertion is controlled by SDRAM ODT Control Low register. 1 = Ovrd: M_ODT[2] assertion/de-assertion is controlled by the <ODT2_ OVRD_val> control bit.
3	ODT1_OVRD_val	RW 0x0	M_ODT[1] assertion control - valid only when ODT1_OVRD bit is 1 0 = OFF: M_ODT[1] is never active. 1 = Active: M_ODT[1] is always active.
2	ODT1_OVRD	RW 0x0	M_ODT[1] override control 0 = Normal: M_ODT[1] assertion/de-assertion is controlled by SDRAM ODT Control Low register. 1 = Ovrd: M_ODT[1] assertion/de-assertion is controlled by the <ODT1_ OVRD_val> control bit.
1	ODT0_OVRD_val	RW 0x0	M_ODT[0] assertion control - valid only when ODT0_OVRD bit is 1 0 = OFF: M_ODT[0] is never active. 1 = Active: M_ODT[0] is always active.
0	ODT0_OVRD	RW 0x0	M_ODT[0] override control 0 = Normal: M_ODT[0] assertion/de-assertion is controlled by SDRAM ODT Control Low register. 1 = Ovrd: M_ODT[0] assertion/de-assertion is controlled by the <ODT0_ OVRD_val> control bit.

Table 471: DDR Controller ODT Control Register

Offset: 0x0000149C

Bit	Field	Type/InitVal	Description
31:10	Reserved	RO 0x0	Reserved
9	ODT Cont OVRD val	RW 0x0	Controller I/O buffer ODT assertion control - valid only when ODT_Cont_OVRD bit is 1. 0 = Not_Active: Internal ODT is never active. 1 = Active: Internal ODT is always active.
8	ODT Cont OVRD	RW 0x0	DDR Controller I/O buffer ODT override control. 0 = Normal: Internal ODT assertion/de-assertion is controlled by ODT_Cont_Read_CS[3:0]. 1 = Ovrd: Internal ODT assertion/de-assertion is controlled by the <ODT_Cont_OVRD_val> control bit.
7	Reserved_7	RW 0x0	Reserved
6	Reserved_6	RW 0x0	Reserved
5	Reserved_5	RW 0x0	Reserved
4	Reserved_4	RW 0x0	Reserved
3	ODT_Cont_READ_CS3	RW 0x0	DDR Controller I/O buffer ODT control during read transactions from DRAM CS3 0 = Not_Active: Internal ODT is de-asserted during read from DRAM CS3. 1 = Active: Internal ODT is asserted during read from DRAM CS3.
2	ODT_Cont_READ_CS2	RW 0x0	DDR Controller I/O buffer ODT control during read transactions from DRAM CS2 0 = Not_Active: Internal ODT is de-asserted during read from DRAM CS2. 1 = Active: Internal ODT is asserted during read from DRAM CS2.
1	ODT_Cont_READ_CS1	RW 0x0	DDR Controller I/O buffer ODT control during read transactions from DRAM CS1 0 = Not_Active: Internal ODT is de-asserted during read from DRAM CS1. 1 = Active: Internal ODT is asserted during read from DRAM CS1.

Copyright © 2014 Marvell

Table 471: DDR Controller ODT Control Register (Continued)

Offset: 0x0000149C

Bit	Field	Type/InitVal	Description
0	ODT_Cont_READ_CS0	RW 0x0	DDR Controller I/O buffer ODT control during read transactions from DRAM CS0 0 = Not_Active: Internal ODT is de-asserted during read from DRAM CS0. 1 = Active: Internal ODT is asserted during read from DRAM CS0.

Table 472: Read Buffer Select Register

Offset: 0x000014A4

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	U15_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0xF 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
14	U14_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0xE 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
13	U13_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0xD 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
12	U12_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0xC 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
11	U11_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0xB 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.

Table 472: Read Buffer Select Register (Continued)

Offset: 0x000014A4

Bit	Field	Type/InitVal	Description
10	U10_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0xA 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
9	U9_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x9 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
8	U8_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x8 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
7	U7_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x7 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
6	U6_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x6 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
5	U5_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x5 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
4	U4_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x4 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
3	U3_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x3 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.

Table 472: Read Buffer Select Register (Continued)

Offset: 0x000014A4

Bit	Field	Type/InitVal	Description
2	U2_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x2 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
1	U1_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x1 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.
0	U0_RdBuff_Sel	RW 0x0	Mbus Read response port select for read transactions sourced by UNIT ID 0x0 0 = RetOnX0: Read response returns on Mbus port0. 1 = RetOnX1: Read response returns on Mbus port1.

Table 473: AXI Control Register

Offset: 0x000014A8

Bit	Field	Type/InitVal	Description
31:11	Reserved	RO 0x0	Reserved.
10:8	Reserved	RSVD 0x1	Reserved
7:5	RdWrChArbWeight	RW 0x0	Coherency Bus DRAM Controller interface read and write channels arbiter weight. 0 = 1Rd1Wr: 1 Read 1 Write 1 = 2Rd1Wr: 2 Reads 1 Write 2 = 3Rd1Wr: 3 Reads 1 Write 3 = 4Rd1Wr: 4 Reads 1 Write 4 = 5Rd1Wr: 5 Reads 1 Write 5 = 6Rd1Wr: 6 Reads 1 Write 6 = 7Rd1Wr: 7 Reads 1 Write 7 = 8Rd1Wr: 8 Reads 1 Write
4:2	Reserved	RO 0x0	Reserved.
1	Reserved	RW 0x0	Reserved must be 0.
0	AxiDataBusWidth	RW 0x0	Coherency Fabric DRAM Controller interface data bus width. 0 = 128bit: Coherency Fabric data bus width is 128 bits. 1 = 256bit: Coherency Fabric data bus width is 256 bits.

Table 474: DRAM Controller Miscellaneous Register

Offset: 0x000014B0

Bit	Field	Type/InitVal	Description
31:1	Reserved	RW 0x0	Reserved
0	RetryMask	RW 0x1	DRAM Controller to Mbus retry 0 = Disable: Normal operation of Mbus retry. 1 = Enable: Mask Mbus retry.

Table 475: DRAM Address and Control Driving Strength (Low) Register

Offset: 0x000014C0

Bit	Field	Type/InitVal	Description
NOTE: The recommend values for this register may be updated in the future, following the results of silicon production testing.			
31:29	Reserved	RO 0x0	Reserved.
28:16	Reserved	RSVD 0x1924	Reserved
15:14	Reserved	RW 0x0	Reserved Must be 0x0.
13:7	ZPR	RW 0x69	Controls the PMOS driving strength of the Address/Control pads. Select the appropriate setting from the values listed below: DDR3 (1.5V): 0x7D = 30 ohm 0x69 = 35 ohm 0x5B = 40 ohm 0x51 = 45 ohm 0x49 = 50 ohm 0x42 = 55 ohm 0x3D = 56 ohm DDR3 (1.35V): 0x7F = 25 ohm 0x7D = 30 ohm 0x6B = 35 ohm 0x5D = 40 ohm 0x53 = 45 ohm 0x49 = 50 ohm All other values are reserved.

Table 475: DRAM Address and Control Driving Strength (Low) Register (Continued)

Offset: 0x000014C0

Bit	Field	Type/InitVal	Description
6:0	ZNR	RW 0x69	Controls the NMOS driving strength of the Address/Control pads. Select the appropriate setting from the values listed below: DDR3 (1.5V): 0x7D = 30 ohm 0x69 = 35 ohm 0x5B = 40 ohm 0x51 = 45 ohm 0x49 = 50 ohm 0x42 = 55 ohm 0x3D = 56 ohm DDR3 (1.35V): 0x7F = 25 ohm 0x7D = 30 ohm 0x6B = 35 ohm 0x5D = 40 ohm 0x53 = 45 ohm 0x49 = 50 ohm All other values are reserved.

Table 476: DRAM Data and DQS Driving Strength (Low) Register

Offset: 0x000014C4

Bit	Field	Type/InitVal	Description
NOTE: The recommend values for this register may be updated in the future, following the results of silicon production testing.			
31:28	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 476: DRAM Data and DQS Driving Strength (Low) Register (Continued)

Offset: 0x000014C4

Bit	Field	Type/InitVal	Description
27:22	D_DATA_ ZPODT	RW 0x24	Controls the PMOS termination value of the DQ pads. Select the appropriate setting from the values listed below: DDR3 (1.5V) : 0x3F = 40 ohm 0x3E = 45 ohm 0x3B = 50 ohm 0x36 = 55 ohm 0x32 = 60 ohm 0x2A = 75 ohm 0x15 = 150 ohm DDR3 (1.35V): 0x3F = 50 ohm 0x33 = 65 ohm 0x2F = 70 ohm 0x2C = 75 ohm 0x29 = 80 ohm 0x27 = 85 ohm 0x17 = 150 ohm All other values are reserved.
21:16	D_DATA_ZNODT	RW 0x24	Controls the NMOS termination value of the DQ pads. Select the appropriate setting from the values listed below: DDR3 (1.5V) : 0x3F = 40 ohm 0x3E = 45 ohm 0x3B = 50 ohm 0x36 = 55 ohm 0x32 = 60 ohm 0x2A = 75 ohm 0x15 = 150 ohm DDR3 (1.35V): 0x3F = 50 ohm 0x33 = 65 ohm 0x2F = 70 ohm 0x2C = 75 ohm 0x29 = 80 ohm 0x27 = 85 ohm 0x17 = 150 ohm All other values are reserved.

Copyright © 2014 Marvell

Table 476: DRAM Data and DQS Driving Strength (Low) Register (Continued)
Offset: 0x000014C4

Bit	Field	Type/InitVal	Description
15:14	Reserved	RW 0x0	Reserved Must be 0x0.
13:7	ZPR	RW 0x69	Controls the PMOS driving strength of the Data and DQS pads. Select the appropriate setting from the values listed below: DDR3 (1.5V): 0x7D = 30 ohm 0x69 = 35 ohm 0x5B = 40 ohm 0x51 = 45 ohm 0x49 = 50 ohm 0x42 = 55 ohm 0x3D = 56 ohm DDR3 (1.35V): 0x7F = 25 ohm 0x7D = 30 ohm 0x6B = 35 ohm 0x5D = 40 ohm 0x53 = 45 ohm 0x49 = 50 ohm All other values are reserved.

Table 476: DRAM Data and DQS Driving Strength (Low) Register (Continued)

Offset: 0x000014C4

Bit	Field	Type/InitVal	Description
6:0	ZNR	RW 0x69	Controls the NMOS driving strength of the Data and DQS pads. Select the appropriate setting from the values listed below: DDR3 (1.5V): 0x7D = 30 ohm 0x69 = 35 ohm 0x5B = 40 ohm 0x51 = 45 ohm 0x49 = 50 ohm 0x42 = 55 ohm 0x3D = 56 ohm DDR3 (1.35V): 0x7F = 25 ohm 0x7D = 30 ohm 0x6B = 35 ohm 0x5D = 40 ohm 0x53 = 45 ohm 0x49 = 50 ohm All other values are reserved.

Table 477: DRAM Vertical Calibration Machine Control Register

Offset: 0x000014C8

Bit	Field	Type/InitVal	Description
31:28	Reserved	RO 0x0	Reserved.
27:22	ManNgCalVal	RW 0x14	Manual ng_cal value.
21:16	ManPgCalVal	RW 0x14	Manual pg_cal value.
15:10	AutoNgCalVal	RO 0x0	Automatic ng_cal value.
9:4	AutoPgCalVal	RO 0x0	Automatic pg_cal value.
3	CalValManOvrd	RW 0x0	Calibration values manual override. 0 = Disable: Use automatic calibration values. 1 = Enable: Use manual calibration values.

Table 477: DRAM Vertical Calibration Machine Control Register (Continued)

Offset: 0x000014C8

Bit	Field	Type/InitVal	Description
2	SinglePfirst	RW 0x0	Single PAD calibration order control. 0 = N_first: Calibrate N then P. 1 = P_first: Calibrate P then N.
1	SinglePadCal	RW 0x0	Single or dual PAD calibration select. 0 = Disable: Calibration is done against 2 PADS. 1 = Enable: Calibration is done against 1 PAD.
0	Reserved	RSVD 0x1	Reserved

Table 478: DRAM Main Pads Calibration Machine Control Register

Offset: 0x000014CC

Bit	Field	Type/InitVal	Description
31	CalMachStat	RO 0x0	Calibration machine status. 0 = Busy: Calibration machine is busy (re-calibrating). 1 = Ready: Calibration machine is ready for re-calibration.
30:13	Dynamic Pads Calibration Counter	RW 0x1e848	The number of 1024 core clock cycles between each dynamic calibration. 0 = Reserved. 1 means 1024 cycles, 2 means 2048 cycles, etc. The minimal value is 10.
12:2	Reserved	RSVD 0x0	Reserved
1	Recalibrate	SC 0x0	Set this bit to allow SW re-calibration of the DRAM pads before setting this bit, must make sure that the calibration machine is ready for re-calibration. 0 = Normal 1 = Recalibrate
0	Dynamic Pads Calibration Enable	RW 0x0	Set this bit to enable dynamic pads calibration. Dynamic pads calibration is intended to compensate for VT variations during functional working mode. 0 = Disable: Dynamic pads calibration is disabled 1 = Enable: Dynamic pads calibration is enabled

Table 479: SDRAM Interface Mbus Control2 (Low) Register

Offset: 0x000014F4

Bit	Field	Type/InitVal	Description
31:28	Arb23	RW 0x7	Pizza Arbiter slice 23 unit ID.
27:24	Arb22	RW 0x6	Pizza Arbiter slice 22 unit ID.
23:20	Arb21	RW 0x5	Pizza Arbiter slice 21 unit ID.
19:16	Arb20	RW 0x4	Pizza Arbiter slice 20 unit ID.
15:12	Arb19	RW 0x3	Pizza Arbiter slice 19 unit ID.
11:8	Arb18	RW 0x2	Pizza Arbiter slice 18 unit ID.
7:4	Arb17	RW 0x1	Pizza Arbiter slice 17 unit ID.
3:0	Arb16	RW 0x0	Pizza Arbiter slice 16 unit ID.

Table 480: SDRAM Interface Mbus Control2 (High) Register

Offset: 0x000014F8

Bit	Field	Type/InitVal	Description
31:28	Arb31	RW 0xF	Pizza Arbiter slice 31 unit ID.
27:24	Arb30	RW 0xE	Pizza Arbiter slice 30 unit ID.
23:20	Arb29	RW 0xD	Pizza Arbiter slice 29 unit ID.
19:16	Arb28	RW 0xC	Pizza Arbiter slice 28 unit ID.
15:12	Arb27	RW 0xB	Pizza Arbiter slice 27 unit ID.
11:8	Arb26	RW 0xA	Pizza Arbiter slice 26 unit ID.
7:4	Arb25	RW 0x9	Pizza Arbiter slice 25 unit ID.
3:0	Arb24	RW 0x8	Pizza Arbiter slice 24 unit ID.

Table 481: Dynamic Power Save Register

Offset: 0x00001520

Bit	Field	Type/InitVal	Description
31:20	DynIdleThreshT	RW 0xB4	Dynamic power save idle threshold - for core-domain. Defines the idle period (in core-domain clock cycles) required for triggering clock gating and/or memory power down (as defined by DynClkGateT and DynMemPwdnT).
19:18	Reserved	RSVD 0x0	Reserved
17	DynMemPwdnT	RW 0x0	Dynamic internal memory power down - for core-domain. When enabled, core-domain internal memories will be powered down once idle threshold (DynIdleThreshT) has been reached, in order to save power. 0 = Disable 1 = Enable
16	DynClkGateT	RW 0x0	Dynamic clock gating - for core-domain. When enabled, core-domain clock will be gated once idle threshold (DynIdleThreshT) has been reached, in order to save power. 0 = Disable 1 = Enable
15:4	DynIdleThreshD	RW 0x12C	Dynamic power save idle threshold - for D-domain. Defines the idle period (in D-domain clock cycles) required for triggering clock gating and/or memory power down (as defined by DynClkGateD and DynMemPwdnD).
3:2	Reserved	RSVD 0x0	Reserved
1	DynMemPwdnD	RW 0x0	Dynamic internal memory power down - for D-domain. When enabled, D-domain internal memories will be powered down once idle threshold (DynIdleThreshD) has been reached, in order to save power. 0 = Disable 1 = Enable
0	DynClkGateD	RW 0x0	Dynamic clock gating - for D-domain. When enabled, D-domain clock will be gated once idle threshold (DynIdleThreshD) has been reached, in order to save power. 0 = Disable 1 = Enable

Table 482: DDR IO Register

Offset: 0x00001524

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	Phy2UnitClkRatio	RW SAR	DRAM PHY to Unit clock ratio. 0 = 1to1: DRAM PHY to Unit clock ratio working in 1:1 clocking mode. 1 = 2to1: DRAM PHY to Unit clock ratio working in 2:1 clocking mode.
14:0	Reserved	RSVD 0x800	Reserved

Table 483: DFS Register

Offset: 0x00001528

Bit	Field	Type/InitVal	Description
31:15	Reserved	RSVD 0x6	Reserved
14:12	DfsCWLNextState	RW 0x0	Indicates the required CWL configuration after DFS is complete. Used only when DfsReconf2 is Enabled. For a list of valid values, please refer to field CWL in register DDR3 MR2.
11:8	DfsCLNextState	RW 0x2	Indicates the required CL configuration after DFS is complete. Used only when DfsReconf0 is Enabled. For a list of valid values, please refer to field CL in register DDR3 MR0.
7	DfsZQCL	RW 0x1	Issue ZQCL command upon completion of the DFS procedure. If this bit is set, the controller commits four ZQCL commands, one per each CS. 0 = Disable 1 = Enable
6	DfsReconf2	RW 0x1	Reconfigure timing parameters in MR2 (or EMRS2) upon completion of the DFS procedure. If this bit is set, the controller commits four MRS commands, one per each CS. Their value is based on the value in register DDR3 MR2 (or Extended DRAM Mode 2) of the relevant CS, and modified according to field: DfsCWLNextState. Software is required to update the aforementioned fields with the appropriate post-DFS values prior to initiating the DFS process. 0 = Disable 1 = Enable

Table 483: DFS Register (Continued)

Offset: 0x00001528

Bit	Field	Type/InitVal	Description
5	DfsReconf1	RW 0x1	Reconfigure timing parameters in MR1 (or EMRS) upon completion of the DFS procedure. If this bit is set, the controller commits four MRS commands, one per each CS. Their value is based on the value in register DDR3 MR1 (or Extended DRAM Mode) of the relevant CS. Software is required to update the aforementioned fields with the appropriate post-DFS values prior to initiating the DFS process. 0 = Disable 1 = Enable
4	DfsReconf0	RW 0x1	Reconfigure timing parameters in MR0 (or MRS) upon completion of the DFS procedure. If this bit is set, the controller commits four MRS commands, one per each CS. Their value is based on the value in register DDR3 MR0 (or SDRAM Mode) of the relevant CS, and modified according to fields: DfsDllNextState, DfsCLNextState, DfsWRNextState. Software is required to update the aforementioned fields with the appropriate post-DFS values prior to initiating the DFS process. 0 = Disable 1 = Enable
3	DfsAtSR	RO 0x0	Indicates that all DRAM devices on all ranks have been transitioned into Self Refresh mode. DFS may be executed once this bit has asserted.
2	DfsSR	RW 0x0	Transition all DRAM devices on all ranks to Self Refresh mode. After setting this bit, software should wait for DfsAtSR to assert. This bit should be asserted before DFS, and only after DfsBlock has been asserted. 0 = Disable 1 = Enable
1	DfsBlock	RW 0x0	Stop committing new transactions to the DDR bus. The controller will continue to accept new transactions from the internal interfaces, but will accumulate and block them. This bit should be asserted to initiate a DFS procedure. 0 = Disable 1 = Enable
0	DfsDllNextState	RW 0x1	Indicates the required DRAM DLL state after DFS is complete. 0 = Enable 1 = Disable

Table 484: Read Data Sample Delays Register

Offset: 0x00001538

Bit	Field	Type/InitVal	Description
31:29	Reserved	RW 0x0	Reserved
28:24	RdSmplDel_3	RW 0x0	Defines the total delay from read command until opening the read mask, when accessing CS3. This field defines the delay in DDR cycles granularity. Finer granularity is achieved in conjunction with register “Chip Select (CS%n) Read Leveling” (see the DDR PHY Registers section): Field CS%n_Rd_Lvl_Ph_Sel defines additional delay in half DDR cycles (or Phases) granularity. Field CS%n_Rd_Lvl_Ref_Dly defines additional delay in granularity of 1/32 parts of a DDR clock cycle. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).
23:21	Reserved	RW 0x0	Reserved
20:16	RdSmplDel_2	RW 0x0	Defines the total delay from read command until opening the read mask, when accessing CS2. This field defines the delay in DDR cycles granularity. Finer granularity is achieved in conjunction with register “Chip Select (CS%n) Read Leveling” (see the DDR PHY Registers section): Field CS%n_Rd_Lvl_Ph_Sel defines additional delay in half DDR cycles (or Phases) granularity. Field CS%n_Rd_Lvl_Ref_Dly defines additional delay in granularity of 1/32 parts of a DDR clock cycle. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).
15:13	Reserved	RW 0x0	Reserved
12:8	RdSmplDel_1	RW 0x0	Defines the total delay from read command until opening the read mask, when accessing CS1. This field defines the delay in DDR cycles granularity. Finer granularity is achieved in conjunction with register “Chip Select (CS%n) Read Leveling” (see the DDR PHY Registers section): Field CS%n_Rd_Lvl_Ph_Sel defines additional delay in half DDR cycles (or Phases) granularity. Field CS%n_Rd_Lvl_Ref_Dly defines additional delay in granularity of 1/32 parts of a DDR clock cycle. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).

Table 484: Read Data Sample Delays Register (Continued)

Offset: 0x00001538

Bit	Field	Type/InitVal	Description
7:5	Reserved	RW 0x0	Reserved
4:0	RdSmplDel_0	RW 0x0	Defines the total delay from read command until opening the read mask, when accessing CS0. This field defines the delay in DDR cycles granularity. Finer granularity is achieved in conjunction with register “Chip Select (CS%n) Read Leveling” (see the DDR PHY Registers section): Field CS%n_Rd_Lvl_Ph_Sel defines additional delay in half DDR cycles (or Phases) granularity. Field CS%n_Rd_Lvl_Ref_Dly defines additional delay in granularity of 1/32 parts of a DDR clock cycle. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).

Table 485: Read Data Ready Delays Register

Offset: 0x0000153C

Bit	Field	Type/InitVal	Description
31:29	Reserved	RW 0x0	Reserved
28:24	RdReadyDel_3	RW 0x0	Number of DDR cycles from read command until data is ready to be fetched from the PHY, when accessing CS3. Configure according to the furthest PUP, i.e. the PUP with the largest read leveling delay, in order to allow fetching data on the same cycle from all PUPs. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).
23:21	Reserved	RW 0x0	Reserved
20:16	RdReadyDel_2	RW 0x0	Number of DDR cycles from read command until data is ready to be fetched from the PHY, when accessing CS2. Configure according to the furthest PUP, i.e. the PUP with the largest read leveling delay, in order to allow fetching data on the same cycle from all PUPs. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).
15:13	Reserved	RW 0x0	Reserved

Table 485: Read Data Ready Delays Register (Continued)

Offset: 0x0000153C

Bit	Field	Type/InitVal	Description
12:8	RdReadyDel_1	RW 0x0	Number of DDR cycles from read command until data is ready to be fetched from the PHY, when accessing CS1. Configure according to the furthest PUP, i.e. the PUP with the largest read leveling delay, in order to allow fetching data on the same cycle from all PUPs. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).
7:5	Reserved	RW 0x0	Reserved
4:0	RdReadyDel_0	RW 0x0	Number of DDR cycles from read command until data is ready to be fetched from the PHY, when accessing CS0. Configure according to the furthest PUP, i.e. the PUP with the largest read leveling delay, in order to allow fetching data on the same cycle from all PUPs. Valid value is CAS latency (see field CL in register DDR3 MR0) or greater. Value may be updated by HW during training (read leveling).

Table 486: Training Register

Offset: 0x000015B0

Bit	Field	Type/InitVal	Description
31	TrnStart	SC 0x0	Starts automatic DDR3 training sequence. This bit is self-cleared once the training sequence is complete. 0 = Disable: Training sequence is not active. 1 = Enable: Start automatic training sequence.
30	TrnError	RO 0x1	DDR3 training sequence error status. Cleared by training restart. 0 = Error encountered during training sequence. 1 = Training sequence finished successfully.
29	TrnDfsReq	RW0C 0x0	Indicates training logic requests a DFS (Dynamic Frequency Change). The requested frequency appears in TrnDfsFreq. SW should clear this bit to zero once the DFS is complete. 0 = Done: DFS is complete. 1 = Request: DFS request is pending.

Table 486: Training Register (Continued)

Offset: 0x000015B0

Bit	Field	Type/InitVal	Description
28:27	TrnDfsFreq	RO 0x0	Requested target frequency for DFS (Dynamic Frequency Change). Valid only when TrnDfsReq is set. 0 = Preload frequency: Frequency used to perload training patterns in DRAM, before any training sequence is performed. 1 = Low frequency: Lowest possble DDR3 frequency. Used for per-bit deskew. 3 = Nominal frequency: Nominal working frequency.
26:24	TrnRetestNum	RW 0x1	Number of retest iterations to confirm a successful delay setup. (Total number of iterations = TrnRetestNum + 1)
23	Trn_CS3	RW 0x0	Chip Select 3 (rank3) DDR3 training sequence enable. 0 = Disable: CS3 is excluded from the DDR3 training sequence. 1 = Enable: CS3 is included in the DDR3 training sequence.
22	Trn_CS2	RW 0x0	Chip Select 2 (rank2) DDR3 training sequence enable. 0 = Disable: CS2 is excluded from the DDR3 training sequence. 1 = Enable: CS2 is included in the DDR3 training sequence.
21	Trn_CS1	RW 0x0	Chip Select 1 (rank1) DDR3 training sequence enable. 0 = Disable: CS1 is excluded from the DDR3 training sequence. 1 = Enable: CS1 is included in the DDR3 training sequence.
20	Trn_CS0	RW 0x0	Chip Select 0 (rank0) DDR3 training sequence enable. 0 = Disable: CS0 is excluded from the DDR3 training sequence. 1 = Enable: CS0 is included in the DDR3 training sequence.
19:18	Reserved	RSVD 0x0	Reserved
17	Reserved	RW 0x0	Reserved
16:0	TrnAutoSeq	RW 0x0	DDR3 Training Sequence Phases Mask A value of one in each bit enables the corresponding phase in the link training procedure.

Table 487: Training Software 1 Register

Offset: 0x000015B4

Bit	Field	Type/InitVal	Description
31:17	Reserved	RSVD 0x7C00	Reserved

Table 487: Training Software 1 Register (Continued)

Offset: 0x000015B4

Bit	Field	Type/InitVal	Description
16	TrnBPointAck	SC 0x0	Indicates end of Software Training, resume HW operation.
15:0	Reserved	RSVD 0xFF	Reserved

Table 488: Training Software 2 Register

Offset: 0x000015B8

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0xD	Reserved
25:17	TrnRLFifoOvrrun	RO 0x0	Indication from PHY that an overrun error has occured during Read Leveling. Each bit corresponds to one DRAM Data PHY. 0 = no overrun. 1 = overrun detected.
16:9	TrnOctMask	RW 0x0	Mask (ignore) specific data octets when performing HW training. Bit per data octet (the ECC octet may be masked by disabling <ECC>), where: 0x0 = Enable octet 0x1 = Mask (ignore) octet
8	Reserved	RW 0x0	Reserved
7:5	TrnRLPtrnBit	RW 0x0	Defines which bit within each 8-bit octet is toggled for the Read Leveling pattern.
4	TrnRLRstFifo	SC 0x0	Reset PHY read FIFO during read leveling. Valid only when <TrnSwOvrd> is enabled. 0 = Disable 1 = Enable
3	TrnRLMode	RW 0x1	Set Read Leveling mode in the DDR3 controller and PHY. Valid only when <TrnSwOvrd> is enabled. 0 = Enable 1 = Disable

Table 488: Training Software 2 Register (Continued)

Offset: 0x000015B8

Bit	Field	Type/InitVal	Description
2	TrnWLMode	RW 0x1	Set Write Leveling mode in the DDR3 controller and PHY. Valid only when <TrnSwOvrd> is enabled; any change to this value must be done while <TrnSwOvrd> is enabled. 0 = Enable 1 = Disable
1	TrnEccMux	RW 0x0	Multiplex ECC read data for ECC DRAM training. During a Read operation, M_CB[7:0] inputs are delivered to the controller as rd_data[7:0] bits. During a Write operation, write data[7:0] bits from the controller are placed on M_CB[7:0] outputs, regardless of this configuration. Valid only when <TrnSwOvrd> is enabled. 0 = Disable 1 = Enable
0	TrnSwOvrd	RW 0x0	Software override of the training sequence. 0 = Disable: Don’t override. 1 = Enable: Override.

Table 489: Training Patterns Base Address Register

Offset: 0x000015BC

Bit	Field	Type/InitVal	Description
31:3	TrnPatternsBAR	RW 0x0	Training patterns base address in DRAM.
2:0	Reserved	RW 0x0	Reserved

Table 490: Training Debug 2 Register

Offset: 0x000015C4

Bit	Field	Type/InitVal	Description
31:28	TrnDbgInitRdSmplInc	RW 0x0	Increment over DDR3_CL (in SDR cycles), for setting the initial <RdSmplDel> for read leveling.
27:25	TrnDbgRdyIncPh5_1to1	RW 0x3	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 5.

Copyright © 2014 Marvell

Table 490: Training Debug 2 Register (Continued)

Offset: 0x000015C4

Bit	Field	Type/InitVal	Description
24:22	TrnDbgRdyIncPh4_1to1	RW 0x2	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 4. Valid when <UnitxPhyClkRatioMode> is set to 1to1 mode.
21:19	TrnDbgRdyIncPh1_1to1	RW 0x2	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 1.
18:16	TrnDbgRdyIncPh0_1to1	RW 0x1	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 0.
15:0	TrnDbgInitDly	RW 0x0	Initial Delay

Table 491: Training Debug 3 Register

Offset: 0x000015C8

Bit	Field	Type/InitVal	Description
31:24	TrnDbgRLFifoRstVal	RW 0xFE	Reset value for the DRAM Data PHYs’ read FIFO.
23:21	TrnDbgRdyIncPh7_2to1	RW 0x6	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 7. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.
20:18	TrnDbgRdyIncPh6_2to1	RW 0x6	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 6. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.
17:15	TrnDbgRdyIncPh5_2to1	RW 0x4	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 5. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.
14:12	TrnDbgRdyIncPh4_2to1	RW 0x4	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 4. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.
11:9	TrnDbgRdyIncPh3_2to1	RW 0x4	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 3. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.
8:6	TrnDbgRdyIncPh2_2to1	RW 0x4	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 2. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.

Table 491: Training Debug 3 Register (Continued)
Offset: 0x000015C8

Bit	Field	Type/InitVal	Description
5:3	TrnDbgRdyIncPh1_2to1	RW 0x2	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 1. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.
2:0	TrnDbgRdyIncPh0_2to1	RW 0x2	Increment over <RdSmplDel> (in DDR cycles), for setting <RdReadyDel> at read leveling phase 0. Valid when <UnitxPhyClkRatioMode> is set to 2to1 mode.

Table 492: DDR3 MR0 Register
Offset: 0x000015D0

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR0 register during a MRS0 command..			
31:16	Reserved	RO 0x0	Reserved.
15:13	RFU	RW 0x0	Reserved for future use. Must be 0.
12	PPD	RW 0x0	DLL control for Precharge Power Down. 0 = Slow Exit: DLL Off. 1 = Fast Exit: DLL On.
11:9	WR	RW 0x3	Write recovery for auto-precharge NOTE: This device does not support auto-precharge. Must be 0x3. 0 = 16cc 1 = 5cc 2 = 6cc 3 = 7cc 4 = 8cc 5 = 10cc 6 = 12cc 7 = 14cc
8	DLL Reset	RW 0x0	DLL Reset. 0 = Normal: No DLL Reset. 1 = reset_DLL: Reset DLL.
7	TM	RW 0x0	Test Mode 0 = Disable: Normal Operation. 1 = Enable: Test Mode.

Table 492: DDR3 MR0 Register (Continued)

Offset: 0x000015D0

Bit	Field	Type/InitVal	Description
6:4	CL_High	RW 0x1	CL + CL_High bits (DDR3_MR0_Reg{[6:4],[2]}) defines the CAS Latency. See CL field for details.
3	RBT	RW 0x0	Read Burst Type 0 = Nibble Sequential: See DDR3 spec burst type and Burst order table. 1 = Interleave: See DDR3 spec burst type and Burst order table.
2	CL	RW 0x0	CL + CL_High bits (DDR3_MR0_Reg{[6:4],[2]}) defines the CAS Latency value. Following values refers to CL[3:0] (i.e. bits[6:4,2]) 1 = CL_12 2 = CL_5 3 = CL_13 4 = CL_6 5 = CL_14 6 = CL_7 8 = CL_8 10 = CL_9 12 = CL_10 14 = CL_11: (optional for DDR3-1600)
1:0	BL	RW 0x0	DDR Data Burst Length 0 = Fixed to 8: BL is always 8. 1 = 4 or 8 OTF: BL is 4 or 8 On The Fly

Table 493: DDR3 MR1 Register

Offset: 0x000015D4

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR1 register during a MRS1 command.			
31:16	Reserved	RO 0x0	Reserved for future use. Must be 0.
15:13	RFU_15_13	RW 0x0	Reserved for future use. Must be 0.
12	Qoff	RW 0x0	Output buffer Enable 0 = On: Output buffer is enabled. 1 = Off: Output buffer is disabled.
11	TDQS_Enable	RW 0x0	TDQS Enable. 0 = Disable: TDQS is Disabled 1 = Enable: TDQS is Enabled

Table 493: DDR3 MR1 Register (Continued)

Offset: 0x000015D4

Bit	Field	Type/InitVal	Description
10	RFU_10	RW 0x0	Reserved for future use. Must be 0.
9	Rtt_nom_9	RW 0x0	Nominal Rtt. Rtt_nom is defined by bits 9,6 and 2 of this register. See MR1 register description in the DDR3 standard for detailed description.
8	RFU_8	RW 0x0	Reserved for future use. Must be 0.
7	Write Leveling	RW 0x0	Write leveling enable. 0 = Disable: Write leveling is disabled. 1 = Enable: Write leveling is enabled.
6	Rtt_nom_6	RW 0x0	Nominal Rtt. Rtt_nom is defined by bits 9,6 and 2 of this register. See MR1 register description in the DDR3 standard for detailed description.
5	DIC_5	RW 0x0	Output Driver Impedance Control. DIC is defined by bits 5 and 1 in this register. See MR1 register description in the DDR3 standard for detailed description.
4:3	AL	RW 0x0	Additive Latency **NOTE:** The device does not support additive latency. 0 = Disable: AL is disabled 1 = AL1: AL = CL -1. 2 = AL2: AL = CL -2. 0x3 is reserved.
2	Rtt_nom	RW 0x0	Nominal Rtt. Rtt_nom is defined by bits 9,6 and 2 of this register. See MR1 register description in the DDR3 standard for detailed description. 0 = Disable 1 = RZQ/4 2 = RZQ/2 3 = RZQ/6 4 = RZQ/12 5 = RZQ/8: All other values are Reserved.

Table 493: DDR3 MR1 Register (Continued)

Offset: 0x000015D4

Bit	Field	Type/InitVal	Description
1	DIC	RW 0x0	Output Driver Impedance Control (DIC). DIC is defined by bits [5] and [1] in this register. **NOTE:** RZQ = 240 ohm. 0 = RZQ_6: Output Driver Impedance = RZQ/6 1 = RZQ_7: Output Driver Impedance = RZQ/7 All other values are reserved.
0	DLL Enable	RW 0x0	DLL Enable 0 = Enable: Enable DLL. 1 = Disable: Disable DLL.

Table 494: DDR3 MR2 Register

Offset: 0x000015D8

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR2 register during a MRS2 command.			
31:16	Reserved	RO 0x0	Reserved, must be 0.
15:11	RFU_15_11	RW 0x0	Reserved for future use, must be 0.
10:9	Rtt_WR	RW 0x0	Dynamic ODT setings. 0x3 is Reserved 0 = Off: Dynamic ODT is Off (Write does not affect Rtt Value) 1 = RZQ/4: Rtt = RZQ/4 2 = RZQ/2: Rtt = RZQ/2
8	RFU_8	RW 0x0	Reserved for future use, must be 0.
7	SRT	RW 0x0	Self-Refresh Temperature range. 0 = Normal: Normal operating temperature range. 1 = Extended: Extended (optional) operating temperature range.
6	ASR	RW 0x0	Auto Self-Refresh. 0 = Disable: Manual SR Reference 1 = Enable: Auto Self Refresh enable.

Table 494: DDR3 MR2 Register (Continued)

Offset: 0x000015D8

Bit	Field	Type/InitVal	Description
5:3	CWL	RW 0x0	CAS Write Latency. 0 = CWL_5 1 = CWL_6 2 = CWL_7 3 = CWL_8 4 = CWL_9 5 = CWL_10 6 = CWL_11 7 = CWL_12
2:0	PASR	RW 0x0	Partial Array Self-Refresh (Optional) 0 = Full Array 1 = Half Array BA_0to3: BA[2:0] = 000,001,010, and 011 2 = Quarter Array BA_0and1: BA[2:0] = 000,& 001 3 = 1/8th Array: BA[2:0] = 000 4 = 3/4 Array: BA[2:0] = 010,011,100,101,110 and 111 5 = Half Array BA_4to7: BA[2:0] = 100,101,110 and 111 6 = Quarter Array BA_6and7: BA[2:0] = 110 and 111 7 = 1/8th Array BA_7: BA[2:0] = 111

Table 495: DDR3 MR3 Register

Offset: 0x000015DC

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR3 register during a MRS3 command.			
31:16	Reserved	RO 0x0	Reserved, must be 0.
15:3	RFU	RW 0x0	Reserved for future use, must be 0.
2	MPR	RW 0x0	MPR Operation control. 0 = Disable: Normal Operation. 1 = Enable: Dataflow from MPR.
1:0	MPR Loc	RW 0x0	MPR Location, must be 0x0. 0 = Predefined: Predefined pattern. 1 = RFU_1: Reserved. 2 = RFU_2: Reserved. 3 = RFU_3: Reserved.

Table 496: DDR3 Rank Control Register

Offset: 0x000015E0

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved, must be 0.
7	CS3 Mirror	RW 0x0	Indicates if CS3 address/control lines are mirrored. 0 = Disable: Non-Mirrored rank. 1 = Enable: Mirrored rank.
6	CS2 Mirror	RW 0x0	Indicates if CS2 address/control lines are mirrored. 0 = Disable: Non-Mirrored rank. 1 = Enable: Mirrored rank.
5	CS1 Mirror	RW 0x0	Indicates if CS1 address/control lines are mirrored. 0 = Disable: Non-Mirrored rank. 1 = Enable: Mirrored rank.
4	CS0 Mirror	RW 0x0	Indicates if CS0 address/control lines are mirrored. 0 = Disable: Non-Mirrored rank. 1 = Enable: Mirrored rank.
3	CS3 Exist	RW 0x0	Indicates that CS3 is populated with Dram device. 0 = Disable: CS3 slot is not populated. 1 = Enable: CS3 slot is populated with Dram device.
2	CS2 Exist	RW 0x0	Indicates that CS2 is populated with Dram device. 0 = Disable: CS2 slot is not populated. 1 = Enable: CS2 slot is populated with Dram device.
1	CS1 Exist	RW 0x0	Indicates that CS1 is populated with Dram device. 0 = Disable: CS1 slot is not populated. 1 = Enable: CS1 slot is populated with Dram device.
0	CS0 Exist	RW 0x1	Indicates that CS0 is populated with Dram device. 0 = Disable: CS0 slot is not populated. 1 = Enable: CS0 slot is populated with Dram device.

Table 497: ZQC Configuration Register

Offset: 0x000015E4

Bit	Field	Type/InitVal	Description
31:23	Reserved	RO 0x0	Reserved.
22	Reserved	RSVD 0x0	Reserved

Table 497: ZQC Configuration Register (Continued)
Offset: 0x000015E4

Bit	Field	Type/InitVal	Description
21	SR Exit ZQCS	RW 0x1	Enable ZQCS command after exiting Self Refresh. 0 = Disable 1 = Enable
20	Auto ZQC select	RW 0x0	Periodic ZQC command type. 0 = ZQCS: ZQ Calibration Short. 1 = ZQCL: ZQ Calibration Long.
19:0	Auto ZQC timing	RW 0x3c18	Number of Refresh commands between consecutive ZQC commands.

Table 498: DRAM PHY Configuration Register
Offset: 0x000015EC

Bit	Field	Type/InitVal	Description
31	DRAM Data PHYs ADLL Reset	RW 0x1	DRAM Data PHYs ADLL Reset. 0 = Reset 1 = Normal
30	Cntrl DRAM PHYs ADLL Reset	RW 0x1	DRAM Control PHYs ADLL Reset. 0 = Reset 1 = Normal
29	Parallel ADLL Conf	SC 0x1	Parallel ADLL Configuration. This bit is self cleared to 0 when all ADLLs are locked. 0 = Clear: Indication that all ADLLs are locked (Self clear) 1 = Enable: Select Parallel configuration.
28:27	Preamble Length	RW 0x3	Number of clock cycles to open write OEn before the preamble. 0 = FullDDRcc: Open the write OEn 1 DDR cycle before the preamble. 1 = HalfDDRcc: Open the write OEn 1/2 DDR cycle before the preamble. 3 = Normal: Normal preamble timing. 0x2 is reserved.
26:6	Reserved	RSVD 0x0	Reserved
5:4	AddrCntrlToClkSkew	RW 0x0	Address / Control to clock skew Mode. **NOTE:** DDR3 or On-Board DRAM devices must use Half clock cycle skew. 0 = Zero: Zero skew. 1 = Quarter: 1/4 cycle skew. 2 = Half: 1/2 cycle skew.
3:2	Reserved	RSVD 0x1	Reserved

Copyright © 2014 Marvell

Table 498: DRAM PHY Configuration Register (Continued)

Offset: 0x000015EC

Bit	Field	Type/InitVal	Description
1	Reserved	RW 0x1	Reserved **NOTE:** Must be 0x0.
0	Reserved	RSVD 0x1	Reserved

Table 499: ODPG CTRL Control Register

Offset: 0x00001600

Bit	Field	Type/InitVal	Description
31	ODPG Select	RW 0x0	ODPG Select 0 = Data: Data Path 1 = Control: Control Path
30:23	Reserved	RO 0x0	Reserved
22	ODPG CNTRL Mbus0 Connect	RW 0x0	Mbus0_Bypass for ODPG mode Must be 0x1 (Bypass) in ODPG mode. Must be 0x0 in Normal mode.
21	ODPG CTRL Auto Refresh Dis	RW 0x0	Auto Refresh Disable (DRAM controller test mode) 0 = Enable: Auto Refresh Normal mode 1 = Disable: Auto Refresh Disable
20:15	ODPG CTRL CMD Burst Del	RW 0x0	ODPG Command Burst Delay Determines number of delay cycles between command bursts. Maximum of 64 delay cycles. (Every burst includes 1-16 consecutive commands, depending on the configurable 4 bit <ODPG CTRL CMD Burst Size> field). 0 means no delay (consecutive bursts), 1 means a 1 cycle delay, etc.
14:11	ODPG CTRL CMD Burst Size	RW 0x0	ODPG Command Burst Size Determines number of Tx commands in a single burst: Number of commands = ODPG_CNTRL_CBSize + 1 Maximum of 16 commands. (Command Burst: consecutive Tx commands with no delay cycles between them.)

Copyright © 2014 Marvell

Table 499: ODPG CTRL Control Register (Continued)

Offset: 0x00001600

Bit	Field	Type/InitVal	Description
10:5	ODPG CTRL Pat Size	RW 0x0	ODPG Pattern Size Determines number of bit phases in a single pattern: Number of phases = ODPG_CNTRL_PSize + 1 Maximum of 64 phases.
4:2	ODPG CTRL _Stop	RW 0x0	ODPG Stop condition Determines ODPG stopping condition. ODPG stops its Tx/Rx operation upon occurrence of the condition determined in this field, unless it is stopped by writing 1'b1 to <ODPG Dis> bit. 0 = Single_Pattern: ODPG will stop upon end of pattern. 1 = Erroneous_Data: ODPG will stop upon receiving erroneous data, only in Rx mode; in Tx mode – the same as 3'b1xx. 2 = Predifined_Addr: Predefined address space (ODPG will stop upon Tx/Rx end of predefined address space determined by buffer_offset[31:0], and buffer_size[31:0]). 3 = Predifined_Num: Predefined number of phases (ODPG will stop upon reaching Tx/Rx of N bit phases determined by phase_length[31:0]). 4 = Continuous: No stop, continuous operation (ODPG will stop only upon write of 1'b1 to odpg_dis bit).
1	ODPG Dis	SC 0x0	Deactivates ODPG operation. Writing 1'b1 to this bit terminates ODPG without any relation to the stop_condition[2:0] configuration. Termination occurs as soon as the ODPG reaches proper completion (last data transmitted in Tx). This bit is cleared upon ODPG assertion of the ODPG_done signal. 0 = Normal: ODPG termination depends on stop_condition[2:0] configuration. 1 = Disable: Disable ODPG.
0	ODPG En	SC 0x0	Activates ODPG operation (by writing 1'b1 to this bit). Also, functions as a status bit, indicating whether or not ODPG is active. This bit is cleared upon ODPG assertion of the ODPG_done signal. 0 = Disable: ODPG is not active (status). 1 = Enable: Activate ODPG (control) / ODPG is active (status).

Table 500: PHY Lock Mask Register

Offset: 0x00001670

Bit	Field	Type/InitVal	Description
31:12	Reserved	RSVD 0x0	Reserved
11	SB_IN ADLL Cal Mask	RW 0x1	Mask START BURST IN ADLL calibration done indication. 0 = Mask: Ignore START BURST IN ADLL calibration done indication (Refer ADLL as calibrated). 1 = Enable: Use START BURST IN ADLL calibration done indication.
10	Cntrl DRAM PHYs ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from all control DRAM PHYs. 0 = Mask: Ignore DRAM Control PHYs calibration done indications (Refer DRAM PHYs as calibrated). 1 = Enable: Use DRAM Control PHYs calibration done indications.
9	DDR APLL Lock Mask	RW 0x1	Mask APLL lock indication. 0 = Mask: Ignore APLL Lock indication (Refer APLL as locked). 1 = Enable: Use APLL Lock indication.
8	DRAM Data PHY 8 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 8. 0 = Mask: Ignore DRAM PHY’s calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY’s calibration done indication.
7	DRAM Data PHY 7 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 7. 0 = Mask: Ignore DRAM PHY’s calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY’s calibration done indication.
6	DRAM Data PHY 6 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 6. 0 = Mask: Ignore DRAM PHY’s calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY’s calibration done indication.
5	DRAM Data PHY 5 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 5. 0 = Mask: Ignore DRAM PHY’s calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY’s calibration done indication.

Table 500: PHY Lock Mask Register (Continued)

Offset: 0x00001670

Bit	Field	Type/InitVal	Description
4	DRAM Data PHY 4 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 4. 0 = Mask: Ignore DRAM PHY's calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY's calibration done indication.
3	DRAM Data PHY 3 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 3. 0 = Mask: Ignore DRAM PHY's calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY's calibration done indication.
2	DRAM Data PHY 2 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 2. 0 = Mask: Ignore DRAM PHY's calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY's calibration done indication.
1	DRAM Data PHY 1 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 1. 0 = Mask: Ignore DRAM PHY's calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY's calibration done indication.
0	DRAM Data PHY 0 ADLL Cal Mask	RW 0x1	Mask ADLL calibration done indication from DRAM Data PHY number 0. 0 = Mask: Ignore DRAM PHY's calibration done indication (Refer DRAM PHY as calibrated). 1 = Enable: Use DRAM PHY's calibration done indication.

Table 501: DRAM PHY Lock Status Register

Offset: 0x00001674

Bit	Field	Type/InitVal	Description
31:12	Reserved	RSVD 0x0	Reserved
11	SB_IN ADLL Cal Status	RO 0x0	START BURST IN ADLL calibration done indication. Active high.
10	Cntrl DRAM PHYs ADLL Cal Status	RO 0x0	ADLL calibration done indication from all control DRAM PHYs. Active high.

Table 501: DRAM PHY Lock Status Register (Continued)

Offset: 0x00001674

Bit	Field	Type/InitVal	Description
9	DDR APLL Lock Status	RO 0x0	APLL lock indication. Active high.
8	DRAM Data PHY 8 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 8. Active high.
7	DRAM Data PHY 7 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 7. Active high.
6	DRAM Data PHY 6 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 6. Active high.
5	DRAM Data PHY 5 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 5. Active high.
4	DRAM Data PHY 4 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 4. Active high.
3	DRAM Data PHY 3 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 3. Active high.
2	DRAM Data PHY 2 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 2. Active high.
1	DRAM Data PHY 1 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 1. Active high.
0	DRAM Data PHY 0 ADLL Cal Status	RO 0x0	ADLL calibration done indication from DRAM Data PHY number 0. Active high.

Table 502: PHY Register File Access
Offset: 0x000016A0

Bit	Field	Type/InitVal	Description
31	PrfaReq	SC 0x0	Issue indirect access request to PHY register file. This bit is self-cleared upon completion of the operation. All other fields in this register must be valid and stable once this bit is asserted. It is not possible to start a new operation before the prior one has been completed. In addition, it is not possible to regret a committed request. Once this bit is asserted, it must not be cleared by SW, but rather by the self-clear hardware mechanism. In read operation, the self-clearing of this bit qualifies valid data in PrfaData. 0 = Disable: Request completed. 1 = Enable: Request pending.
30	PrfaType	RW 0x0	Operation type. 0 = Read 1 = Write
29:28	Reserved	RW 0x0	Reserved Must be 0.
27	PrfaBC	RW 0x0	Broadcast write operation. If enabled and PrfaPupCtrlData=0x1, the write will be broadcast to all DRAM Control PHYs. If enabled and PrfaPupCtrlData=0x0, the write will be broadcast to all DRAM Data PHYs. If enabled and PrfaPupCtrlData=0x1 and PrfaPupNum=0xF, the write will be broadcast to all DRAM PHYs. 0 = Disable 1 = Enable
26	PrfaPupCtrlData	RW 0x0	DRAM PHY type select. 0 = Data: DRAM Data PHY. 1 = Cntrl: DRAM Control PHY.
25:22	PrfaPupNum	RW 0x0	PuP number in the PHY.
21:16	PrfaRegNum	RW 0x0	Register number in the PHY register file.

Copyright © 2014 Marvell

Table 502: PHY Register File Access (Continued)

Offset: 0x000016A0

Bit	Field	Type/InitVal	Description
15:0	PrfaData	RW 0x0	During Write operation: The data to write into the PHY register file. During Read operation: The data read from the PHY register file (qualified by the clearing of PrfaReq).

Table 503: Training Write Leveling Register

Offset: 0x000016AC

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29	TrnWLDelayExp	RO 0x0	Write Leveling <TWLDelay> timer has expired (DQS-to-DQ-sample)
28:20	TrnWLResult	RO 0x0	Write Leveling result, as sampled on DQ pins.
19:16	TrnWLRefNum	RW 0x9	Number of Refresh commands to execute before moving the DRAM into Write Leveling mode.
15:8	TrnWrLvlPrmDqMask	RW 0xFF	Select DQ bits in each byte as prime for Write Leveling. Each bit in this field corresponds to the matching DQ bit, where: 0 is Mask 1 is Prime
7:4	TrnWLDqs	SC 0x0	Generates a DQS pulse (one bit per delay phase). For frequencies up to 400 MHz: 0x5 = Enable. 0x0 = Disable. For frequencies above 400 MHz: 0x1 = Enable. 0x0 = Disable.
3	TrnWLDeUpd	SC 0x0	Delay Element update in the PHY. Required when WL training of each CS is complete.
2	TrnWLCsUpd	SC 0x0	Update write leveling delay phase in the PHY. Must be set whenever TrnWLCs is changed.

Table 503: Training Write Leveling Register (Continued)

Offset: 0x000016AC

Bit	Field	Type/InitVal	Description
1:0	TrnWLCs	RW 0x0	Chip Select number for Write Leveling. Controls the write leveling delay used by the PHY.

Table 504: DDR3 Registered DRAM Control

Offset: 0x000016D0

Bit	Field	Type/InitVal	Description
31:28	CKE3 CS Grp	RW 0x8	Indicates the group of CS's that are connected to CKE3. Used for SRE and PDE commands to DDR3 registered DRAM. (LSB=CS0, MSB=CS3)
27:24	CKE2 CS Grp	RW 0x4	Indicates the group of CS's that are connected to CKE2. Used for SRE and PDE commands to DDR3 registered DRAM. (LSB=CS0, MSB=CS3)
23:20	CKE1 CS Grp	RW 0x2	Indicates the group of CS's that are connected to CKE1. Used for SRE and PDE commands to DDR3 registered DRAM. (LSB=CS0, MSB=CS3)
19:16	CKE0 CS Grp	RW 0x1	Indicates the group of CS's that are connected to CKE0. Used for SRE and PDE commands to DDR3 registered DRAM. (LSB=CS0, MSB=CS3)
15	SRFloatEn	RW 0x1	Controls whether to float control and clock outputs during Self Refresh. This bit should be disabled in cases where a CWA command is required while the DRAM is in Self Refresh. **NOTE:** If this bit is enabled, floating the clock outputs is subject to the SRClk configuration. If this bit is disabled, clock outputs will not be floated anyway. 0 = Disable: Do not float outputs. 1 = Enable: Float outputs.
14:9	Reserved	RW 0x38	Reserved
8	Reserved	RSVD 0x1	Reserved

Table 504: DDR3 Registered DRAM Control (Continued)

Offset: 0x000016D0

Bit	Field	Type/InitVal	Description
7	Register 3T mode	RW 0x0	The register 1T/3T timing mode. Must match the configuration of the SSTE32882 (RC2 bit DA4). 0 = Disable: Operates in 1T mode. 1 = Enable: This sets the mode to 3T mode.
6	CWA Parity	RW 0x0	Parity Calculation This field should match the PAR_IN input to the SSTE32882.
5:4	CWA WE	RW 0x1	Value of M_WEn during CWA command **NOTE:** Only one of <CWA RAS>, <CWA CAS>, or <CWA WE> can be set to ParityComplement. 0 = Low: Drive 0x0 1 = High: Drive 0x1 2 = ParityComplement: Drive 0x0 or 0x1 to meet Parity calculation on the CWA command according to <CWA Parity> field. 0x3 = Reserved;
3:2	CWA CAS	RW 0x1	Value of M_CASn during CWA (Control Word Access) command **NOTE:** Only one of <CWA RAS>, <CWA CAS>, or <CWA WE> can be set to ParityComplement. 0 = Low: Drive 0x0 1 = High: Drive 0x1 2 = ParityComplement: Drive 0x0 or 0x1 in order to meet Parity calculation on the CWA command according to <CWA Parity> field. 0x3 = Reserved;
1:0	CWA RAS	RW 0x2	Value of M_RASn during CWA command **NOTE:** Only one of <CWA RAS>, <CWA CAS>, or <CWA WE> can be set to ParityComplement. 0 = Low: Drive 0x0 1 = High: Drive 0x1 2 = ParityComplement: Drive 0x0 or 0x1 in order to meet Parity calculation on the CWA command according to <CWA Parity> field. 0x3 = Reserved;

Table 505: DDR3 Registered DRAM Timing

Offset: 0x000016D4

Bit	Field	Type/InitVal	Description
31:20	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 505: DDR3 Registered DRAM Timing (Continued)

Offset: 0x000016D4

Bit	Field	Type/InitVal	Description
19:16	CWA tMRD	RW 0x8	tMRD as defined by JEDEC JESD82-29A (SSTE32882 Registering clock driver) Value 0 means 1 cycle, value 1 means 2 cycles, etc.
15:0	tSTAB	RW 0x12C0	tSTAB as defined by JEDEC JESD82-29A (SSTE32882 Registering clock driver) Value 0 means 1 cycle, value 1 means 2 cycles, etc.

Table 506: DLB Control Register

Offset: 0x00001700

Bit	Field	Type/InitVal	Description
31:9	Reserved	RSVD 0x100000	Reserved
8:7	PreFetchNLnAddrTrig	RW 0x0	Next-line Pre-Fetch Address Trigger Current read line address hits a line quarter equal or bigger than this field. 0 = First: 1st 32B line quarter 1 = Second: 2nd 32B line quarter 2 = Third: 3rd 32B line quarter 3 = Fourth: 4th 32B line quarter
6	PreFetchNLnSzTr	RW 0x0	Next-line Pre-Fetch Size Trigger Current read size = full line (128B) 0 = Disable: Next-line prefetch full size trigger is disabled 1 = Enable: Next-line prefetch full size trigger is enabled
5	PreFetchILnSzTr	RW 0x0	In-line Pre-Fetch Size Trigger Current read size >= 32B 0 = Disable: Inline prefetch min size trigger is disabled 1 = Enable: Inline prefetch min size trigger is enabled
4	MbusPreFetchEn	RW 0x0	Mbus Line Pre-Fetch Enable Mbus master originated read executes a DLB line pre-fetch from DRAM. 0 = Disable: Mbus line pre-fetch is disabled. 1 = Enable: Mbus line pre-fetch is enabled.
3	AxiPreFetchEn	RW 0x0	AXI Line Pre-Fetch Enable AXI master originated read executes a DLB line pre-fetch from DRAM. 0 = Disable: AXI line pre-fetch is disabled. 1 = Enable: AXI line pre-fetch is enabled.

Table 506: DLB Control Register (Continued)
Offset: 0x00001700

Bit	Field	Type/InitVal	Description
2	WrCoalescingEn	RW 0x0	Write Coalescing Enable 0 = Disable: Data of a write command that hits a DLB modified line flushes the original line, it is before stored in DLB. 1 = Enable: Data of a write command that hits a DLB modified line is merged with the existing line without flushing it.
1	EvictModifiedFull	RW 0x0	Evict fully modified line immediately. 0 = Disable: Fully modified line is not evicted immediately. 1 = Enable: Fully modified line is evicted immediately.
0	DlbEn	RW 0x0	DLB Enable 0 = Disable: DLB is disabled. 1 = Enable: DLB is enabled.

Table 507: DLB Bus Optimization Weights Register
Offset: 0x00001704

Bit	Field	Type/InitVal	Description
NOTE: The sum of each formula below should not exceed 7: 1. SameBankOPageWeight + SameCmdWeight + SameCsWeight + RdCmdWeight 2. DiffBankOPageWeight + SameCmdWeight + SameCsWeight + RdCmdWeight 3. DiffBankCPageWeight + SameCmdWeight + SameCsWeight + RdCmdWeight 4. SameBankOPageWeight + SameCmdWeight + SameCsWeight + WrCmdWeight 5. DiffBankOPageWeight + SameCmdWeight + SameCsWeight + WrCmdWeight 7. DiffBankCPageWeight + SameCmdWeight + SameCsWeight + WrCmdWeight 8. OpenPageWeight + SameCmdWeight + SameCsWeight + RdCmdWeight 9. OpenPageWeight + SameCmdWeight + SameCsWeight + WrCmdWeight			
31:22	Reserved	RSVD 0x0	Reserved
21:20	PfPrio	RW 0x0	Next line prefetch command priority setting. Valid only when field PfPrioSrc = Priority_programmed. 0 = Low priority: Next line prefetch command priority setting = low priority. 1 = Med priority: Next line prefetch command priority setting = medium priority. 2 = High priority: Next line prefetch command priority setting = high priority. 3 = Top priority: Next line prefetch command priority setting = top priority.

Table 507: DLB Bus Optimization Weights Register (Continued)

Offset: 0x00001704

Bit	Field	Type/InitVal	Description
19	PfPrioSrc	RW 0x0	Next line prefetch command priority source. 0 = Priority programmed: Next line prefetch command priority is programmed (copied from PfPrio field). 1 = Priority same as read: Next line Prefetch command priority is the same as the priority of the read command that initiated it.
18:16	PfRspWeight	RW 0x0	The scheduling weight of a prefetch response command **NOTE:** Prefetch response represents a read response from a valid line that was prefetched and imported to DLB. **NOTE:**
15:14	OpenPageWeight	RW 0x0	Open Page Weight The scheduling weight of command to an open page for a given bank. **NOTE:** The difference between this weight and the first two weight fields (SameBankOPageWeight and DiffBankOPageWeight) is that this weight differentiates the total weight of commands targeting to a given bank. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4
13:12	SameCsWeight	RW 0x0	Same CS Weight The scheduling weight of the command to the same CS as the previous command. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4
11:10	WrCmdWeight	RW 0x0	Write Command Weight The scheduling weight of the write command. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4
9:8	RdCmdWeight	RW 0x0	Read Command Weight The scheduling weight of the read command. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4

Copyright © 2014 Marvell

Table 507: DLB Bus Optimization Weights Register (Continued)

Offset: 0x00001704

Bit	Field	Type/InitVal	Description
7:6	SameCmdWeight	RW 0x0	Same Command Type Weight The scheduling weight for a command of the same read/write type as the previous command. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4
5:4	DiffBankCPageWeight	RW 0x0	Different Bank and Closed Page Weight The scheduling weight for a command targeting a different bank than the previous command and that does not hit an open page in its target bank address. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4
3:2	DiffBankOPageWeight	RW 0x0	Different Bank and Open Page Weight The scheduling weight for a command targeting a different bank than the previous command and that hits an open page in its target bank address. **NOTE:** The difference between this weight and the last weight field (OpenPageWeight) is that this weight differentiates the total weight of commands targeting to different banks. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4
1:0	SameBankOPageWeight	RW 0x0	Same Bank and Open Page Weight The scheduling weight for a command targeting the same bank and same page as the previous command. Applicable only if the accessed page is open. **NOTE:** The difference between this weight and the last weight field (OpenPageWeight) is that this weight differentiates the total weight of commands targeting to different banks. 0 = Zero: Configured to weight 0 1 = One: Configured to weight 1 2 = Two: Configured to weight 2 3 = Four: Configured to weight 4

Table 508: DLB Aging Register

Offset: 0x00001708

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27:24	AgePrioCntThrCmp	RW 0x1	Aging Priority Counter Threshold - Read Completions. <Description Missing> Configuration range is 0x1-0xF (0x0 is reserved).
23	Reserved	RSVD 0x0	Reserved
22:16	AgePrioCntThrRd	RW 0x7F	Aging Priority Counter Threshold - Reads. This value determines time periods between line aging priority level incrementation. The counter counts in DRAM clock cycles. When it reaches the threshold value, it expires, reloads, and starts counting again. Each expiration increments the line’s priority by one. **NOTE:** The same threshold is set for all banks, yet each line is managed separately. Configuration range is 0x1-0xFFFF (0x0 is reserved).
15:7	Reserved	RSVD 0x0	Reserved
6:0	AgePrioCntThrWr	RW 0x7F	Aging Priority Counter Threshold - Writes. This value determines time periods between line aging priority level incrementation. The counter counts in DRAM clock cycles. When it reaches the threshold value, it expires, reloads, and starts counting again. Each expiration increments the line’s priority by one. **NOTE:** The same threshold is set for all banks, yet each line is managed separately. Configuration range is 0x1-0xFFFF (0x0 is reserved).

Table 509: DLB Eviction Control Register

Offset: 0x0000170C

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x0	Reserved

Table 509: DLB Eviction Control Register (Continued)

Offset: 0x0000170C

Bit	Field	Type/InitVal	Description
9	EvictIdleTimeEn	RW 0x0	Eviction Idle Enable Offline modified lines flush (eviction) **NOTE:** Relevant window timer located in register "DLB Eviction Timers" 0 = Disable: Offline modified lines flush is disabled. 1 = Enable: Offline modified lines flush is enabled.
8	EvictBlockEn	RW 0x0	Eviction Threshold High Window Enable **NOTE:** Relevant window timer located in register "DLB Eviction Timers" 0 = Disable: Eviction threshold high window is disabled. 1 = Enable: Eviction threshold high window is enabled.
7:4	EvictThrLow	RW 0x4	Eviction Threshold Low When number of enqueued modified lines targeting a specific bank reaches this threshold - they are forwarded for execution with priority status = Low. **NOTE:** Same threshold for all banks, yet each bank is managed separately. Value 0x0 means 1 line, value 0x1 means 2 lines, etc.
3:0	EvictThrHigh	RW 0x9	Eviction Threshold High When number of enqueued modified lines targeting a specific bank reaches this threshold - they are forwarded for execution with priority status = High. **NOTE:** Same threshold for all banks, yet each bank is managed separately. Value 0x0 means 1 line, value 0x1 means 2 lines, etc.

Table 510: DLB Eviction Timers Register

Offset: 0x00001710

Bit	Field	Type/InitVal	Description
31:16	EvictIdleTimeThr	RW 0xFF	Eviction Idle Profiler Used for offline modified lines flush (eviction). When this timer expires, all modified lines currently in DLB are evicted. Configuration range = 0x1 - 0xFFFF. Value 0x0 is reserved.
15:10	EvictBlockWinLow	RW 0xF	Eviction Low-Priority Window. This value determines the time period in which evicted commands priority is lowered to lowest priority level. **NOTE:** This period prevents blockage of new read commands that arrive during low/high-threshold eviction periods and allows new commands to bypass the evicted ones. Configuration range = 0x1 - 0xFFFF Value 0x0 is reserved.
9:0	EvictBlockWin	RW 0x1F	Eviction Threshold Priority Window. This value determines the time period in which evicted commands receive their regular priority according to low/high thresholds and aging. Configuration range = 0x1 - 0xFFFF Value 0x0 is reserved.

Table 511: DRAM Horizontal Calibration machine control Register

Offset: 0x000017C8

Bit	Field	Type/InitVal	Description
NOTE: This register controls the calibration machine that handles the rotated pads in the chip (90 degrees)			
31:28	Reserved	RO 0x0	Reserved.
27:22	ManNgCalVal	RW 0x14	Manual ng_cal value.
21:16	ManPgCalVal	RW 0x14	Manual pg_cal value.
15:10	AutoNgCalVal	RO 0x0	Automatic ng_cal value.
9:4	AutoPgCalVal	RO 0x0	Automatic pg_cal value.

Table 511: DRAM Horizontal Calibration machine control Register (Continued)

Offset: 0x000017C8

Bit	Field	Type/InitVal	Description
3	CalValManOvrd	RW 0x0	Calibration values manual override. 0 = Disable: Use automatic calibration values. 1 = Enable: Use manual calibration values.
2	SinglePfirst	RW 0x0	Single PAD calibration order control. 0 = N_first: Calibrate N then P. 1 = P_first: Calibrate P then N.
1	SinglePadCal	RW 0x0	Single or dual PAD calibration select. 0 = Disable: Calibration is done against 2 PADS. 1 = Enable: Calibration is done against 1 PAD.
0	Reserved	RSVD 0x1	Reserved

Table 512: DDR3 MR0 CSn Register (n=0–3)

Offset: CS number0: 0x00001870, CS number1: 0x00001880, CS number2: 0x00001890, CS number3: 0x000018A0

Offset Formula: 0x00001870+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR0 register during a MRS0 command to M_CS%n			
31:16	Reserved	RO 0x0	Reserved.
15:13	RFU	RW 0x0	Reserved for future use, must be 0.
12	PPD	RW 0x0	DLL control for Precharge Power Down. **NOTE:** This field must be configured identically for all chip selects. 0 = Slow Exit: DLL Off. 1 = Fast Exit: DLL On.
11:9	WR	RW 0x3	Write recovery for auto-precharge NOTE: This device does not support auto-precharge. Must be 0x3. 0 = 16cc 1 = 5cc 2 = 6cc 3 = 7cc 4 = 8cc 5 = 10cc 6 = 12cc 7 = 14cc

Copyright © 2014 Marvell

Table 512: DDR3 MR0 CSn Register (n=0–3) (Continued)

Offset: CS number0: 0x00001870, CS number1: 0x00001880, CS number2: 0x00001890, CS number3: 0x000018A0

Offset Formula: 0x00001870+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
8	DLL Reset	RW 0x0	DLL Reset. 0 = Normal: No DLL Reset. 1 = reset_DLL: Reset DLL.
7	TM	RW 0x0	Test Mode 0 = Disable: Normal Operation. 1 = Enable: Test Mode.
6:4	CL_High	RW 0x1	CL + CL_High bits (DDR3_MR0_Reg{[6:4],[2]}) defines the CAS Latency. See CL field for details. **NOTE:** This field must be configured identically for all chip selects.
3	RBT	RW 0x0	Read Burst Type **NOTE:** This field must be configured identically for all chip selects. 0 = Nibble Sequential: See DDR3 spec burst type and Burst order table. 1 = Interleave: See DDR3 spec burst type and Burst order table.
2	CL	RW 0x0	CL + CL_High bits (DDR3_MR0_Reg{[6:4],[2]}) defines the CAS Latency value. Following values refer to CL[3:0] (i.e. bits[6:4,2]) **NOTE:** This field must be configured identically for all chip selects. 1 = CL_12 2 = CL_5 3 = CL_13 4 = CL_6 5 = CL_14 6 = CL_7 8 = CL_8 10 = CL_9 12 = CL_10 14 = CL_11: (optional for DDR3-1600)
1:0	BL	RW 0x0	DDR Data Burst Length **NOTE:** This field must be configured identically for all chip selects. 0 = Fixed to 8: BL is always 8. 1 = 4 or 8 OTF: BL is 4 or 8 On The Fly

Table 513: DDR3 MR1 CSn Register (n=0–3)

Offset: CS number0: 0x00001874, CS number1: 0x00001884, CS number2: 0x00001894, CS number3: 0x000018A4

Offset Formula: 0x00001874+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR1 register during a MRS1 command to M_ CS%n.			
31:16	Reserved	RO 0x0	Reserved, must be 0.
15:13	RFU_15_13	RW 0x0	Reserved for future use, must be 0.
12	Qoff	RW 0x0	Output buffer Enable 0 = On: Output buffer is enabled. 1 = Off: Output buffer is disabled.
11	TDQS_Enable	RW 0x0	TDQS Enable. 0 = Disable: TDQS is Disabled 1 = Enable: TDQS is Enabled
10	RFU_10	RW 0x0	Reserved for future use, must be 0.
9	Rtt_nom_9	RW 0x0	Nominal Rtt. Rtt_nom is defined by bits 9,6 and 2 of this register. See MR1 register description in the DDR3 standard for detailed description.
8	RFU_8	RW 0x0	Reserved for future use, must be 0.
7	Write Leveling	RW 0x0	Write leveling enable. 0 = Disable: Write leveling is disabled. 1 = Enable: Write leveling is enabled.
6	Rtt_nom_6	RW 0x0	Nominal Rtt. Rtt_nom is defined by bits 9,6 and 2 of this register. See MR1 register description in the DDR3 standard for detailed description.
5	DIC_5	RW 0x0	Output Driver Impedance Control. DIC is defined by bits 5 and 1 in this register. See MR1 register description in the DDR3 standard for detailed description.
4:3	AL	RW 0x0	Additive Latency **NOTE:** The device does not support additive latency. 0 = Disable: AL is disabled 1 = AL1: AL = CL -1. 2 = AL2: AL = CL -2. 0x3 is reserved.

Table 513: DDR3 MR1 CSn Register (n=0–3) (Continued)

Offset: CS number0: 0x00001874, CS number1: 0x00001884, CS number2: 0x00001894, CS number3: 0x000018A4

Offset Formula: 0x00001874+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
2	Rtt_nom	RW 0x0	Nominal Rtt. Rtt_nom is defined by bits 9,6 and 2 of this register. See MR1 register description in the DDR3 standard for detailed description. 0 = Disable 1 = RZQ/4 2 = RZQ/2 3 = RZQ/6 4 = RZQ/12 5 = RZQ/8: All other values are Reserved.
1	DIC	RW 0x0	Output Driver Impedance Control (DIC). DIC is defined by bits [5] and [1] in this register. **NOTE:** RZQ = 240 ohm 0 = RZQ_6: Output Driver Impedance = RZQ/6 1 = RZQ_7: Output Driver Impedance = RZQ/7 All other values are reserved.
0	DLL Enable	RW 0x0	DLL Enable 0 = Enable: Enable DLL. 1 = Disable: Disable DLL.

Table 514: DDR3 MR2 CSn Register (n=0–3)

Offset: CS number0: 0x00001878, CS number1: 0x00001888, CS number2: 0x00001898, CS number3: 0x000018A8

Offset Formula: 0x00001878+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR2 register during a MRS2 command to M_ CS%n.			
31:16	Reserved	RO 0x0	Reserved, must be 0.
15:11	RFU_15_11	RW 0x0	Reserved for future use, must be 0.
10:9	Rtt_WR	RW 0x0	Dynamic ODT setings. 0x3 is Reserved 0 = Off: Dynamic ODT is Off (Write does not affect Rtt Value) 1 = RZQ/4: Rtt = RZQ/4 2 = RZQ/2: Rtt = RZQ/2
8	RFU_8	RW 0x0	Reserved for future use, must be 0.

Table 514: DDR3 MR2 CSn Register (n=0–3) (Continued)

Offset: CS number0: 0x00001878, CS number1: 0x00001888, CS number2: 0x00001898, CS number3: 0x000018A8

Offset Formula: 0x00001878+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
7	SRT	RW 0x0	Self-Refresh Temperature range. 0 = Normal: Normal operating temperature range. 1 = Extended: Extended (optional) operating temperature range.
6	ASR	RW 0x0	Auto Self-Refresh. 0 = Disable: Manual SR Reference 1 = Enable: Auto Self Refresh enable.
5:3	CWL	RW 0x0	CAS Write Latency. **NOTE:** This field must be configured identically for all chip selects. 0 = CWL_5 1 = CWL_6 2 = CWL_7 3 = CWL_8 4 = CWL_9 5 = CWL_10 6 = CWL_11 7 = CWL_12
2:0	PASR	RW 0x0	Partial Array Self-Refresh (optional) 0 = Full Array 1 = Half Array BA_0to3: BA[2:0] = 000,001,010, & 011 2 = Quarter Array BA_0and1: BA[2:0] = 000,& 001 3 = 1/8th Array: BA[2:0] = 000 4 = 3/4 Array: BA[2:0] = 010,011,100,101,110 & 111 5 = Half Array BA_4to7: BA[2:0] = 100,101,110 & 111 6 = Quarter Array BA_6and7: BA[2:0] = 110 & 111 7 = 1/8th Array BA_7: BA[2:0] = 111

Table 515: DDR3 MR3 CSn Register (n=0–3)

Offset: CS number0: 0x0000187C, CS number1: 0x0000188C, CS number2: 0x0000189C, CS number3: 0x000018AC

Offset Formula: 0x0000187C+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
NOTE: Data written to this register will be configured to the DDR3 DRAM MR3 register during a MRS3 command to M_ CS%n			
31:16	Reserved	RO 0x0	Reserved, must be 0.
15:3	RFU	RW 0x0	Reserved for future use, must be 0.

Table 515: DDR3 MR3 CSn Register (n=0–3) (Continued)

Offset: CS number0: 0x0000187C, CS number1: 0x0000188C, CS number2: 0x0000189C, CS number3: 0x000018AC

Offset Formula: 0x0000187C+n*0x10: where n (0-3) represents CS number

Bit	Field	Type/InitVal	Description
2	MPR	RW 0x0	MPR Operation control. 0 = Disable: Normal Operation. 1 = Enable: Dataflow from MPR.
1:0	MPR Loc	RW 0x0	MPR Location, must be 0x0. 0 = Predefined: Predefined pattern. 1 = RFU_1: Reserved. 2 = RFU_2: Reserved. 3 = RFU_3: Reserved.

A.3.3 SDRAM Scratch Pad

Table 516: Scratch Pad 0 Register

Offset: 0x00001504

Bit	Field	Type/InitVal	Description
31:24	Scratch Pad 0	RW 0x0f	Scratch byte for software usage.
23:0	Reserved	RSVD 0xFFFFF1	Reserved

Table 517: Scratch Pad 1 Register

Offset: 0x0000150C

Bit	Field	Type/InitVal	Description
31:24	Scratch Pad 1	RW 0x0f	Scratch byte for software usage.
23:0	Reserved	RSVD 0xFFFFF5	Reserved

Table 518: Scratch Pad 2 Register

Offset: 0x00001514

Bit	Field	Type/InitVal	Description
31:24	Scratch Pad 2	RW 0x0f	Scratch byte for software usage.
23:0	Reserved	RSVD 0xFFFFF9	Reserved

Table 519: Scratch Pad 3 Register

Offset: 0x0000151C

Bit	Field	Type/InitVal	Description
31:24	Scratch Pad 3	RW 0x0f	Scratch byte for software usage.
23:0	Reserved	RSVD 0xFFFFFD	Reserved

A.3.4 DDR Data PHY Registers

Table 520: Chip Select (CSn) Write Leveling Register (n=0–3)

Offset: M_CS number0: 0x00000000, M_CS number1: 0x00000004, M_CS number2: 0x00000008, M_CS number3: 0x0000000C

Offset Formula: 0x00000000+n*4: where n (0-3) represents M_CS number

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	CSn_Wr_DQS_Ref_Dly_Inv	WO 0x0	M_CS[n] DQS out delay element output inversion control. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer. 0 = Disable: Do not invert the M_CS[n] DQS out delay element output. 1 = Enable: Invert the M_CS[n] DQS out delay element output.
14:10	CSn_Wr_DQS_Ref_Dly	WO 0x0	Reference Delay of M_CS[n] DQS out delay element. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer.
9:8	CSn_Wr_Lvl_PH_Sel	RW 0x0	Required phase of the DDR clock to start write leveling of DQS_out/DQS_out_b signals during access to M_CS[n]. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer. 0 = Ph0: First phase 1 = Ph1: Second phase 2 = Ph2: Third phase 3 = Ph3: Fourth phase
7:6	Reserved	RSVD 0x0	Reserved
5	CSn_Wr_Lvl_Ref_Dly_Inv	RW 0x0	M_CS[n] write leveling delay element output inversion control. 0 = Disable: Do not invert M_CS[n] write leveling delay element output. 1 = Enable: Invert M_CS[n] write leveling delay element output.

Table 520: Chip Select (CSn) Write Leveling Register (n=0–3) (Continued)

Offset: M_CS number0: 0x00000000, M_CS number1: 0x00000004, M_CS number2: 0x00000008, M_CS number3: 0x0000000C

Offset Formula: 0x00000000+n*4: where n (0-3) represents M_CS number

Bit	Field	Type/InitVal	Description
4:0	CSn_Wr_Lvl_Ref_Dly	RW 0x0	Reference delay of M_CS[n] write leveling delay element. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer. **NOTE:**

Table 521: Chip Select (CSn) Write DQS Reference Delay Register (n=0–3)

Offset: M_CS number0: 0x00000001, M_CS number1: 0x00000005, M_CS number2: 0x00000009, M_CS number3: 0x0000000D

Offset Formula: 0x00000001+4*n: where n (0-3) represents M_CS number

Bit	Field	Type/InitVal	Description
31:6	Reserved	RSVD 0x0	Reserved
5	CSn_Wr_DQS_Ref_Dly_Inv	RW 0x0	M_CS[n] DQS_out delay element output inversion control. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer. 0 = Disable: Do not invert the M_CS0[n] DQS_out delay element output. 1 = Enable: Invert the M_CS[n] DQS_out delay element output.
4:0	CSn_Wr_DQS_Ref_Dly	RW 0x0	Reference delay of M_CS[n] DQS_out delay element. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer.

Table 522: Chip Select (CSn) Read Leveling Register (n=0–3)

Offset: M_CS number0: 0x00000002, M_CS number1: 0x00000006, M_CS number2: 0x0000000A, M_CS number3: 0x0000000E

Offset Formula: 0x00000002+4*n: where n (0-3) represents M_CS number

Bit	Field	Type/InitVal	Description
31:11	Reserved	RSVD 0x0	Reserved

Table 522: Chip Select (CSn) Read Leveling Register (n=0–3) (Continued)

Offset: M_CS number0: 0x00000002, M_CS number1: 0x00000006, M_CS number2: 0x0000000A, M_CS number3: 0x0000000E

Offset Formula: 0x00000002+4*n: where n (0-3) represents M_CS number

Bit	Field	Type/InitVal	Description
10:8	CSn_Rd_Lvl_Ph_Sel	RW 0x0	Required phase of the ddr_clk to start read leveling for M_CS[n]. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer. 0 = Ph0: 1st phase of ddr_clk (1st phase of sdr_clk) 1 = Ph1: 2nd phase of ddr_clk (1st phase of sdr_clk) 2 = Ph2: 3rd phase of ddr_clk allowed in 2:1 mode only (2nd phase of sdr_clk) 3 = Ph3: 4th phase of ddr_clk allowed in 2:1 mode only (2nd phase of sdr_clk) 4 = Ph4: 5th phase of ddr_clk in 2:1 mode 3rd phase of ddr_clk in 1:1 mode. 5 = Ph5: 6th phase of ddr_clk in 2:1 mode 4th phase of ddr_clk in 1:1 mode. 6 = Ph6: 7th phase of ddr_clk in 2:1 mode allowed in 2:1 mode only 7 = Ph7: 8th phase of ddr_clk in 2:1 mode allowed in 2:1 mode only
7:6	Reserved	RSVD 0x0	Reserved
5	CSn_Rd_Lvl_Ref_Dly_Inv	RW 0x0	M_CS[n] reads the leveling delay element output inversion control. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer. 0 = Disable: Do not invert M_CS[n] read leveling delay element output. 1 = Enable: Invert M_CS[n] read leveling delay element output.
4:0	CSn_Rd_Lvl_Ref_Dly	RW 0x0	Reference delay of M_CS[n] read leveling delay element. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer.

Table 523: Chip Select (CSn) Read DQS Reference Delay Register (n=0–3)

Offset: M_CS number0: 0x00000003, M_CS number1: 0x00000007, M_CS number2: 0x0000000B, M_CS number3: 0x0000000F

Offset Formula: 0x00000003+4*n: where n (0-3) represents M_CS number

Bit	Field	Type/InitVal	Description
31:6	Reserved	RSVD 0x0	Reserved

Table 523: Chip Select (CSn) Read DQS Reference Delay Register (n=0–3) (Continued)

Offset: M_CS number0: 0x00000003, M_CS number1: 0x00000007, M_CS number2: 0x0000000B, M_CS number3: 0x0000000F

Offset Formula: 0x00000003+4*n: where n (0-3) represents M_CS number

Bit	Field	Type/InitVal	Description
5	CSn_Rd_DQS_Ref_ Dly_Inv	RW 0x0	M_CS[n] DQS_in delay element output inversion control. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer. 0 = Disable: Do not invert the M_CS[n] DQS_in delay element output 1 = Enable: Invert the M_CS[n] DQS_in delay element output
4:0	CSn_Rd_DQS_Ref_ Dly	RW 0x0	Reference delay of M_CS[n] DQS_in delay element. **NOTE:** Reserved for Marvell use. Do not change the default value without approval from a Marvell Field Applications Engineer.

A.3.5 DDR PHY Registers (Internal)

A.3.5.1 Data PHY Registers (Internal)

Table 524: Data Bitn Write Deskew Register (n=0–7)

Offset: DQ bit number0: 0x00000010, DQ bit number1: 0x00000011, DQ bit number2: 0x00000012, DQ bit number3: 0x00000013, DQ bit number4: 0x00000014, DQ bit number5: 0x00000015, DQ bit number6: 0x00000016, DQ bit number7: 0x00000017

Offset Formula: 0x00000010+n: where n (0-7) represents DQ bit number

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4:0	M_DQn_WR_ DESKEW_VAL	RW 0x0	M_DQ[n] bit deskew value for write operations. **NOTE:**

Table 525: DQS Write Deskew Register

Offset: 0x00000018

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved

Table 525: DQS Write Deskew Register (Continued)

Offset: 0x00000018

Bit	Field	Type/InitVal	Description
4:0	DQS_WR_ DESKEW_VAL	RW 0x0	M_DQS deskew value for write operations. **NOTE:**

Table 526: DQS B Write Deskew Register

Offset: 0x00000019

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4:0	DQS_B_WR_ DESKEW_VAL	RW 0x0	M_DQSn deskew value for write operation. **NOTE:**

Table 527: DM Write Deskew Register

Offset: 0x0000001A

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4:0	DM_WR_DESKEW_ VAL	RW 0x0	M_DM deskew value for write operation. **NOTE:**

Table 528: Data Bitn Read Deskew Register (n=0–7)

Offset: DQ bit number0: 0x00000030, DQ bit number1: 0x00000031, DQ bit number2: 0x00000032, DQ bit number3: 0x00000033, DQ bit number4: 0x00000034, DQ bit number5: 0x00000035, DQ bit number6: 0x00000036, DQ bit number7: 0x00000037

Offset Formula: 0x00000030+n: where n (0-7) represents DQ bit number

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4:0	DQn_RD_ DESKEW_VAL	RW 0x0	M_DQ[n] data deskew value for read operation. **NOTE:**

Table 529: DQS Read Deskew Register

Offset: 0x00000038

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4:0	DQS_RD_ DESKEW_VAL	RW 0x0	M_DQS deskew value for read operation. **NOTE:**

Table 530: PAD Control Register

Offset: 0x0000003D

Bit	Field	Type/InitVal	Description
31:9	Reserved	RSVD 0x0	Reserved
8	DM PAD Enable	RW 0x0	When disabled, DM pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable
7	DQ7 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable
6	DQ6 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable
5	DQ5 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable
4	DQ4 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable

Table 530: PAD Control Register (Continued)

Offset: 0x0000003D

Bit	Field	Type/InitVal	Description
3	DQ3 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable
2	DQ2 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable
1	DQ1 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable
0	DQ0 PAD Enable	RW 0x0	When disabled, DQx pad is non-functional and power consumption is minimized (IO PAD pin state as following: pd_comsb, odt_en = 0, oe = 1). 0 = Enable 1 = Disable

A.3.5.2 Control PHY Registers

Table 531: Control n Deskew Register (n=0–6)

Offset:

Offset Formula:

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4:0	Reserved	RW 0x0	Reserved for Marvell use. **NOTE:** Do not change the default value without approval from a Marvell Field Applications Engineer.

Table 532: Clock n Deskew Register (n=0–1)

Offset:

Offset Formula:

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved

Table 532: Clock n Deskew Register (n=0–1) (Continued)

Offset:

Offset Formula:

Bit	Field	Type/InitVal	Description
4:0	Reserved	RW 0x0	Reserved for Marvell use. **NOTE:** Do not change the default value without approval from a Marvell Field Applications Engineer.

A.4 NAND Flash Registers

The following table provides a summarized list of all registers that belong to the NAND Flash, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 533: Summary Map Table for the NAND Flash Registers

Register Name	Offset	Table and Page
Data Flash Control Register (NDCR)	0x000D0000	Table 534, p. 857
NAND Interface Timing Parameter 0 Register (NDTR0CS0)	0x000D0004	Table 535, p. 865
NAND Interface Timing Parameter 1 Register (NDTR1CS0)	0x000D000C	Table 536, p. 868
NAND Controller Status Register (NDSR)	0x000D0014	Table 537, p. 871
NAND Controller Page Count Register (NDPCR)	0x000D0018	Table 538, p. 877
NAND Controller Bad Block Registers (NDBBRx)	0x000D001C	Table 539, p. 878
NAND ECC Control (NDECCCTRL) Register	0x000D0028	Table 540, p. 878
NAND Busy Length Count (NDBZCNT) Register	0x000D002C	Table 541, p. 879
NAND Controller Data Buffer (NDDB) Register	0x000D0040	Table 542, p. 880
NAND Controller Command Buffer 0 (NDCB0) Register	0x000D0048	Table 543, p. 881
NAND Controller Command Buffer 1 (NDCB1) Register	0x000D004C	Table 544, p. 885
NAND Controller Command Buffer 2 (NDCB2) Register	0x000D0050	Table 545, p. 886
NAND Controller Command Buffer 3 (NDCB3) Register	0x000D0054	Table 546, p. 887

Copyright © 2014 Marvell

Table 534: Data Flash Control Register (NDCR)

Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
Program the Timing Parameter registers (NDTR0CSx and NDTR1CSx) and Control registers before writing to this register. This register must be programmed before the partition registers are written to because writing any protection regions into the protection registers overrides updates to the PG_PER_BLK and RA_START fields. The reason for this blocking is to prevent an incorrect update of the control register that can “confuse” the partition address matching functions. This is a read/write register. Ignore reads from reserved bits. Write 0 to reserved bits.			
31	SPARE_EN	RW 0x0	Spare Area Enable Set the spare-area enable (SPARE_EN) bit to use the spare area of the NAND flash for data storage. If the SPARE_EN bit is set and ECC is not enabled (ECC_EN cleared), it is possible to write to and read from the entire data area and spare area. If SPARE_EN is clear, the spare area is not available to users. If SPARE_EN and ECC_EN are set, the remaining spare bytes after ECC is written are available to software. NDTR0CSx and NDTR1CSx registers specify the data bytes available for program/read, to the user per page for different SPARE_EN and ECC_EN settings. The ECCCTRL register defines the number of bytes used by ECC in ECCCTRL.ECC_SPARE. For Hamming ECC, the compatible definition is used. For extended ECC additional bytes are required to achieve the needed error recovery. 0 = Disable: Write/Read to available spare area disabled. 1 = Enable: Write/Read to available spare area enabled.
30	ECC_EN	RW 0x0	ECC Enable When the ECC enable (ECC_EN) bit is set, it enables the computation of ECC and error detection and correction. When ECC_EN bit is clear, ECC computation and error checks are not performed. The ECCCTRL register defines the type of ECC to be performed. The compatible algorithm is to use a single bit error correct, double bit detect Hamming code where a syndrome is generated for each 512 bytes. For extended correction a BCH algorithm with 16-bit per 2K byte page correcting code is used. 0 = Disable: ECC is disabled for write and read data. 1 = Enable: ECC is enabled for write and read data.
29	DMA_EN	RW 0x0	DMA Request Enable When this bit is clear, no DMA requests are asserted. **NOTE:** When PDMA is set to OFF, this bit must be cleared to 0x0. 0 = Disable: Data and Command DMA are disabled. 1 = Enable: Data and Command DMA are enabled.

Table 534: Data Flash Control Register (NDCR) (Continued)

Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
28	ND_RUN	RW 0x0	NAND Controller Run Mode When the NAND-controller run mode (ND_RUN) bit is set, the NAND controller starts to execute the command in the command buffer. When PDMA is set to ON, if the command buffer is empty when the ND_RUN bit is set, the NAND controller asserts the command-DMA request if the DMA request is enabled. The ND_RUN bit is cleared when the NAND flash controller finishes the execution of a command with the next command (NC) bit clear, indicating the end of the command sequence. For a bad-block detect, ND_RUN is cleared after the command buffer is emptied. Clearing the ND_RUN bit during a command execution results in an immediate termination of transactions to the flash device and clears the data and command buffers. 0 = Disable: NAND controller is not in run mode. 1 = Enable: NAND controller is in run mode.
27	DWIDTH_C	RW 0x0	Data Bus Width of the NAND Flash Controller Flash-controller data width (DWIDTH_C) defines the data-bus width used by the NAND flash controller to communicate to the flash memory device(s). When DWIDTH_C = 1 and DWIDTH_M = 0, two flash devices are connected to the same device. 0 = 8-bits: Data Bus width is 8 bits. 1 = 16-bits: When 16b_NFC is set to ON, Data Bus width is 16 bits. Otherwise, this is reserved.
26	DWIDTH_M	RW 0x0	Data Bus Width of the NAND Flash Memory Flash-memory data width (DWIDTH_M) defines the data bus width of the NAND flash-memory devices used in the memory system. DWIDTH_M and DWIDTH_C determine the way NAND flash devices are interfaced. DWIDTH_C = 0 and DWIDTH_M = 1 is a non-valid setting. If used, the flash controller behavior defaults to DWIDTH_C = 0 and DWIDTH_M = 0. When DWIDTH_C = 1 and DWIDTH_M = 0, two flash devices are connected to the same device. 0 = 8-bits: Data Bus width is 8 bits. 1 = 16-bits: When 16b_NFC is set to ON, Data Bus width is 16 bits. Otherwise, this is reserved.
25:24	PAGE_SZ	RW 0x0	Page Size of the Flash device The page-size (PAGE_SZ) field defines the page size of the flash memory. All memory devices used in the flash-memory system must have the same bus width, page size, command set, and timing parameters.

Copyright © 2014 Marvell

Table 534: Data Flash Control Register (NDCR) (Continued)

Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
23	SEQ_DIS	RW 0x0	Sequential page read disable This bit should be set if sequential page read is not wanted. When SEQ_DIS bit is clear, all multi-page reads are completed sequentially. The device is addressed only when the first page is read, and subsequent page reads are performed by toggling NF_REn when the device is ready. Consult the data sheet of the flash part under “Sequential Row Read” or similar for details. **NOTE:** Sequential read is supported only when the entire page (including the spare area) is read by the NAND flash controller. The cases where sequential read is supported are: {PAGE_SZ = 00, SPARE_EN = 1, ECC_EN = 0}, {PAGE_SZ = 01, SPARE_EN = 1, ECC_EN = 0} and {PAGE_SZ = 01, SPARE_EN = 1, ECC_EN = 1}. For all other settings of PAGE_SZ, SPARE_EN and ECC_EN sequential read are not supported, the NCSX bit must be set, and multi-page reads are performed after addressing each page individually. If SEQ_DIS is clear, in between multiple read operations will wait for NDTR1CS0.tR_NFC before transferring more data. If NDTR1CS0.WAIT_MODE is set, the value in NDTR1CS0.tR_NFC can be “small” and the NFC will wait for a minimum time before transferring more data. If WAIT_MODE is not set, the value in NDTR1CS0.tR_NFC will need to be set to the tR parameter from the NAND flash device’s data sheet. In this case the data throughput may not be optimal. 0 = No Command Cycle: Sequential page reads (without commands) are supported. 1 = Command Cycle Required: All sequential page reads require command cycles.
22	Reserved	RSVD 0x0	Reserved
21	FORCE_CSX	RW 0x0	Force chip select false on busy If the FORCE_CSX bit is set, then the NF_CSn for the addressed flash must be de-asserted during the time where Busy is expected to be asserted. In addition, during all operations DFI (Data Flash Interface) arbitration must be allowed during the command and/or data transfer phases. Under normal operation, NF_CSn shall stay asserted throughout the command and also throughout data transfer phases, but if the DFI is being arbitrated away from the NFC, the command sequence or data sequence shall be interrupted at the beginning of the current cycle by first de-asserting NF_CSn and then relinquishing the DFI to the other slave. During an address phase where NF_ALE is asserted, NF_ALE is de-asserted along with NF_CSn. When the NFC reacquires the DFI, NF_ALE and NF_CSn will assert and address cycles will continue with NF_WEn strobes. When NOR/NAND hardware arbiter enabled, this bit must be set. 0 = Assert NF_CSn: Assert chip select during entire command, except during command overlap. 1 = De-assert NF_CSn: De-assert chip select during busy phases.

Copyright © 2014 Marvell

Table 534: Data Flash Control Register (NDCR) (Continued)

Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
20	CLR_PG_CNT	RW 0x0	Clear Page Count The clear-page-count (CLR_PG_CNT) bit, when set, clears the Page Count register. After the NAND Interface Page Count register is cleared, the CLR_PG_CNT bit is reset. When a program failure occurs, CLR_PG_CNT can be used to clear the page count values after software reads the number of pages programmed correctly. 0 = Not Cleared: Page count not cleared 1 = Clear: Clear page count
19	STOP_ON_UNCOR	RW 0x0	Stop On Uncorrectable Error If an uncorrectable error is received on a read and this bit is set, NDCB0.NC is reset and when PDMA is set to ON, further command DMA processing is halted. 0 = Continue: Do not stop on uncorrectable error. 1 = Stop: Stop on uncorrectable error.
18:16	RD_ID_CNT	RW 0x0	Read ID Byte Count The read-ID byte-count (RD_ID_CNT) bit specifies the number of data bytes to read from the flash device when a read-ID command is issued. Valid values for RD_ID_CNT are 1–7. For a read ID command, the page count should be set to 1 (NDCB2[Page_Count] should be zero). Values from 1 to 7. Specifies the number of bytes of read ID data to be read from the flash device
15	RA_START	RW 0x0	Row Address Start Position The row-address start-position (RA_START) bit specifies the address cycle where row address starts in the addressing sequence. When RA_START is clear, NDCB1[ADDR2] (address sent out in the second addressing cycle) contains the first row address. When RA_START is set, NDCB1[ADDR3] (address sent out in the third addressing cycle) contains the first row address. 0 = 2nd Address Cycle: Row address supplied to flash from second address cycle onwards 1 = 3rd Address Cycle: Row address supplied to flash from third address cycle onwards
14:13	PG_PER_BLK	RW 0x0	Pages Per Block The pages-per-block (PG_PER_BLK) bit specifies the number of pages in one block of the flash device. The PG_PER_BLK bit, when cleared, indicates 32 pages/block, and when set, indicates 64 pages/block. PG_PER_BLK along with RA_START is used by the NAND flash controller to increment the row address in multi-page commands. 00 = Flash device has 32 pages per block 10 = Flash device has 64 pages per block 01 = Flash device has 128 pages per block 11 = Flash device has 256 pages per block **NOTE:** The order of bits seems backwards due to compatibility. Previously bit 13 was reserved. **NOTE:**

Table 534: Data Flash Control Register (NDCR) (Continued)

Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
12	Reserved	RSVD 0x1	Reserved
11	RDYM	RW 0x1	Flash Device Ready Interrupt Mask The flash-device ready-mask (RDYM) bit is used to mask an interrupt request that is asserted when NF_RnB (Ready/Busy_) makes a transition from low to high, as indicated by NDSR.RDY. When RDYM is cleared, the interrupt is enabled, and whenever NDSR[RDY] is set, an interrupt request is made to the interrupt controller. When RDYM is cleared, the interrupt is masked and the state of the RDY status bit does not generate an interrupt. If more than one NF_RnB signal is used, this mask bit enables/disables interrupts for both. 0 = Enable: Flash device ready interrupt is enabled. 1 = Disable: Flash device ready interrupt is disabled.
10	CS0_PAGEDM	RW 0x1	NF_CSn[0,2] Page Done Interrupt Mask The CS0 page-done mask (CS0_PAGEDM) bit is used to mask or enable an interrupt request that is asserted when a page transaction (read or write) to a flash device interfaced using NF_CSn[0] is completed, as indicated by NDSR[CS0_PAGED]. When CS0_PAGEDM is cleared, the interrupt is enabled; whenever NDSR[CS0_PAGED] is set, an interrupt request is made to the interrupt controller. When CS0_PAGEDM is set, the interrupt is masked and the state of the PAGE_DN status bit does not generate an interrupt. **NOTE:** All reference to NF_CSn[2] is only applicable when 16b_NFC is set to ON. 0 = Enable: NF_CSn[0,2] page done interrupt is enabled. 1 = Disable: NF_CSn[0,2] page done interrupt is disabled.
9	CS1_PAGEDM	RW 0x1	NF_CSn[1,3] Page Done Interrupt Mask When 16b_NFC is set to ON, CS1 Page Done Mask (CS1_PAGEDM) is used to mask or enable an interrupt request that is asserted when a page transaction (read or write) to flash device interfaced using NF_CSn[1] is completed as indicated by NDSR[CS1_PAGED]. When CS1_PAGEDM is cleared, the interrupt is enabled, and whenever NDSR[CS1_PAGED] is set, an interrupt request is made to the interrupt controller. When CS1_ PAGEDM is set, the interrupt is masked and the state of the PAGE_DN status bit does not generate an interrupt. **NOTE:** When 16b_NFC is set to OFF, this bit must be set to 1. 0 = Enable: NF_CSn[1,3] page done interrupt is enabled. 1 = Disable: NF_CSn[1,3] page done interrupt is disabled.

Table 534: Data Flash Control Register (NDCR) (Continued)

Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
8	CS0_CMDDM	RW 0x1	NF_CSn[0,2] Command Done Interrupt Mask The CS0 command-done mask (CS0_CMDDM) bit is used to mask or enable an interrupt request that is asserted when the command for the flash device interfaced using NF_CSn[0] has been executed, as indicated by NDSR[CS0_CMDD]. When CS0_CMDDM is cleared, the interrupt is enabled; whenever NDSR[CS0_CMDD] is set, an interrupt request is made to the interrupt controller. When CS0_CMDDM is set, the interrupt is masked and the state of the CS0_CMDD status bit does not generate an interrupt. **NOTE:** All reference to NF_CSn[2] is only applicable when 16b_NFC is set to ON. 0 = Enable: NF_CSn[0,2] command done interrupt is enabled. 1 = Disable: NF_CSn[0,2] command done interrupt is disabled.
7	CS1_CMDDM	RW 0x1	NF_CSn[1,3] Command Done Interrupt Mask When 16b_NFC is set to ON, the CS1 command-done mask (CS1_CMDDM) bit is used to mask or enable an interrupt request that is asserted when the command for the flash device interfaced using ND_nCS1 has been executed, as indicated by NDSR[CS1_CMDD]. When CS1_CMDDM is cleared, the interrupt is enabled, and whenever NDSR[CS1_CMDD] is set, an interrupt request is made to the interrupt controller. When CS1_CMDDM is set, the interrupt is masked and the state of the CS1_CMDD status bit does not generate an interrupt. **NOTE:** When 16b_NFC is set to OFF, this bit must be set to 1. 0 = Enable: NF_CSn[1,3] command done interrupt is enabled. 1 = Disable: NF_CSn[1,3] command done interrupt is disabled.
6	CS0_BBDM	RW 0x1	NF_CSn[0,2] Bad Block Detect Interrupt Mask The CS0 bad-block-detect mask (CS0_BBDM) bit is used to mask or enable an interrupt request that is asserted when the status check at the end of a page write or block erase to a device interfaced using ND_CSn[0,2] returns a bad-block error, as indicated by NDSR[CS0_BBD]. When CS0_BBDM = 0, the interrupt is enabled; whenever NDSR[CS0_BBD] is set, an interrupt request is made to the interrupt controller. When CS0_BBDM=1, the interrupt is masked and the state of the CS0_BBD status bit does not generate an interrupt. **NOTE:** All reference to NF_CSn[2] is only applicable when 16b_NFC is set to ON. 0 = Enable: NF_CSn[0,2] bad block detect interrupt is enabled. 1 = Disable: NF_CSn[0,2] bad block detect interrupt is disabled.

Copyright © 2014 Marvell

Table 534: Data Flash Control Register (NDCR) (Continued)
Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
5	CS1_BBDM	RW 0x1	NF_CSn[1,3] Bad Block Detect Interrupt Mask When 16b_NFC is set to ON, CS1 Bad Block Detect Mask (CS1_BBDM) is used to mask or enable an interrupt request that is asserted when the status check at the end of a page write or block erase to a device interfaced using NF_CSn[1,3] returns a bad block error, as indicated by NDSR[CS1_BBD]. When CS1_BBDM is cleared, the interrupt is enabled, and whenever NDSR[CS1_BBD] is set, an interrupt request is made to the interrupt controller. When CS1_BBDM is set, the interrupt is masked and the state of the CS1_BBD status bit does not generate an interrupt. **NOTE:** For the reliable operation of flash media, always enable CS0_ BBD and CS1_BBD interrupts (CS0_BBDM and CS1_BBDM cleared), and have software perform block replacement based on these interrupts. When 16b_NFC is set to OFF, this bit must be set to 1. 0 = Enable: NF_CSn[1,3] bad block detect interrupt is enabled. 1 = Disable: NF_CSn[1,3] bad block detect interrupt is disabled.
4	UNCERRM	RW 0x1	ECC Failure Error Interrupt Mask The ECC Failure error mask (UNCERRM) bit is used to mask or enable an interrupt request that is asserted when an uncorrectable Hamming double-bit error is detected in any of the data streams as indicated by NDSR[UNCERR]. When in extended ECC mode, this bit indicates a failure with the BCH ECC. When UNCERRM is cleared, the interrupt is enabled; whenever NDSR[UNCERR] is set, an interrupt request is made to the interrupt controller. When UNCERRM is set, the interrupt is masked and the state of the UNCERR status bit does not generate an interrupt. No data correction is possible for either kind of failure. 0 = Enable: ECC Failure error interrupt is enabled. 1 = Disable: ECC Failure error interrupt is disabled.
3	CORERRM	RW 0x1	Correctable Error Interrupt Mask The correctable error mask (CORERRM) bit is used to mask or enable an interrupt request that is asserted when either a single-bit error is detected in any of the data streams when in Hamming ECC mode) or when the number of error exceeds the threshold defined in the ECCCTRL register when in BCH mode. This condition is indicated by NDSR[CORERR]. When CORERRM is cleared, the interrupt is enabled; whenever NDSR[CORERR] is set, an interrupt request is made to the interrupt controller. When CORERRM is set, the interrupt is masked and the state of the UNCERR status bit does not generate an interrupt. This interrupt is an indication only and can be used to determine if read disturbs require a block to be rewritten. 0 = Enable: Single-bit error interrupt is enabled. 1 = Disable: Single-bit error interrupt is disabled.

Table 534: Data Flash Control Register (NDCR) (Continued)

Offset: 0x000D0000

Bit	Field	Type/InitVal	Description
2	WRDREQM	RW 0x1	Write Data Request Interrupt Mask The write-data-request mask (WRDREQM) bit is used to mask or enable an interrupt request that is asserted when a write command is loaded into command buffer and data buffer is empty, as indicated by NDSR[WRDREQ]. When WRDREQM is cleared, the interrupt is enabled; whenever NDSR[WRDREQ] is set, an interrupt request is made to the interrupt controller. When WRDREQM is set, the interrupt is masked and the state of the WRDREQ status bit does not generate an interrupt. 0 = Enable: Write data request interrupt is enabled. 1 = Disable: Write data request interrupt is disabled.
1	RDDREQM	RW 0x1	Read Data Request Interrupt Mask Read data request mask (RDDREQM) is used to mask or enable an interrupt request that is asserted when the data buffer has been loaded with a page of read data, as indicated by NDSR[RDDREQ]. When RDDREQM is cleared, the interrupt is enabled, and whenever NDSR[RDDREQ] is set, an interrupt request is made to the interrupt controller. When RDDREQM is set, the interrupt is masked and the state of the RDDREQ status bit does not generate an interrupt. 0 = Enable: Read data request interrupt is enabled. 1 = Disable: Read data request interrupt is disabled.
0	WRCMDREQM	RW 0x1	Write Command Request Interrupt Mask Write command request mask (WRCMDREQM) is used to mask or enable an interrupt request that is asserted when a write to the command buffer is requested, as indicated by NDSR[WRCMDREQ]. When WRCMDREQM is cleared, the interrupt is enabled, and whenever NDSR[WRCMDREQ] is set, an interrupt request is made to the interrupt controller. When WRCMDREQM is set, the interrupt is masked and the state of the RDDREQ status bit does not generate an interrupt. 0 = Enable: Write command request interrupt is enabled. 1 = Disable: Write command request interrupt is disabled.

Table 535: NAND Interface Timing Parameter 0 Register (NDTR0CS0)

Offset: 0x000D0004

Bit	Field	Type/InitVal	Description
Program this register to configure the setup and hold times of outputs driven by the NAND flash controller (NF_ALE, NF_ CLE, NF_CSn) and the pulse widths and cycle times of NF_REn and NF_WEn. Accesses to NAND flash devices interfaced using NF_CSn are controlled by the settings in NDTR0CS0. Programmable timing parameters enable the interface to a wide variety of NAND flash devices. All timing parameters in NDTR0CS0 are set in terms of NAND controller clock periods. The NAND controller clock runs at the ND_CLK frequency. Refer to the Hardware Specifications for frequency limitations of the NAND flash controller timing parameters. The NDTR0CS0 and NDTR1CS0 registers define the timings as integer multiples of the NFC clock. This is a read/write register. Ignore reads from reserved bits. Write 0 to reserved bits.			
31:27	tADL_NFC	RW 0x1	Address to Write Data Delay This parameter controls the time between the rising edge of the NF_WEn strobe for the last address cycle until the first rising edge of the NF_WEn strobe for a write operation. Value is from 0 to 31 and specifies in terms of ND_CLK periods. The fields must be set to the difference between the NAND's device tADL requirement and the Max(tWH_NFC,tCH_NFC) * t(ND_CLK) ns. Requirement (tADL refer to the parameters of the NAND flash data sheet): tADL_NFC * t(ND_CLK) > (tADL - Max(tWH_NFC,tCH_NFC) * t(ND_ CLK))
26	SelCntr	RW 0x1	Select Read Counter This bit determines the source of the read strobe. SW must set this field to "Strobe". 0 = Reserved 1 = Strobe: Strobe is Rd_Cnt_Del+1 ND_CLK clocks from the rising edge of NF_REn.
25:22	Rd_Cnt_Del	RW 0x8	Read Strobe Count Delay In read commands, this counter used to latch the read data into the NFC relatively to the NF_REn edge. The NF_REn falling or rising edge is the trigger to this counter (The edge can be selected by sel_nRE_edge field). This value is the number of ND_CLK clocks after the rising/falling edge of NF_REn to latch read data into the NFC from the DFI. Note: When NOR/NAND hardware arbiter used (enabled), SW must add value of 3 to this field in order to compensate on internal sampling stages.
21:19	tCH_NFC	RW 0x4	Enable Signal Hold Time Enable signal hold time (tCH) defines the hold time (with respect to the rising edge of NF_WEn) of NF_CSn, NF_ALE and NF_CLE. Value from 0 to 7. Specifies the hold time for NF_CSn, NF_ALE, and NF_ CLE outputs in terms of NAND controller clock periods. Requirement (tCH, tCLH and tALH refer to the parameters of the NAND flash data sheet): (tCH_NFC + 1) * t(ND_CLK) > Max(tCH, tCLH, tALH)

Table 535: NAND Interface Timing Parameter 0 Register (NDTR0CS0) (Continued)

Offset: 0x000D0004

Bit	Field	Type/InitVal	Description
18:16	tCS_NFC	RW 0x1	Enable Signal Setup Time Enable signal setup time (tCS) defines the setup time (with respect to the falling edge of NF_WEn) of NF_CSn, NF_ALE and NF_CLE. Value from 0 to 7. Specifies the setup time for NF_CSn, NF_ALE, and NF_CLE outputs in terms of NAND controller clock periods. Requirements (tCS, tCLS, tALS, tREA and tCEA refer to the parameters of the NAND flash data sheet): 1. (tCS_NFC + 1) * t(ND_CLK) >= tCS - (tWP_NFC + 1) * t(ND_CLK) 2. (tCS_NFC + 1) * t(ND_CLK) >= Max(tCLS,tALS) - (tWP_NFC + 1) * t(ND_CLK) 3. When FORCE_CSX enabled: (tCS_NFC +1) * t(ND_CLK) >= (tCEA - tREA)
15:14	Reserved	RSVD 0x0	Reserved
13:11	tWH_NFC	RW 0x1	NF_WEn high duration The period for which NF_WEn remains high, while address or data is being input to the NAND flash device in multiple NF_WEn cycles, is specified by NF_WEn high duration (tWH). The valid values for tWH_NFC are 0 through 7. Actual high duration of NF_WEn is (tWH_NFC +1) * t(ND_CLK). Value from 0 to 7. Specifies the NF_WEn high duration, in terms of ND_CLK periods. NF_WEn high duration = tWH_NFC+1 for a series of write pulses. For the last write pulse of a command, the duration is max(tWH_NFC,tCH_NFC)+1. Also note that this is the minimum duration that is guaranteed. The maximum duration for which the NF_WEn is high is greater than (tWH_NFC + 1) * t(ND_CLK) if the DFI bus is granted to the NOR during this time. Requirements (tWH and tWC refer to the parameters of the NAND flash data sheet): 1. (tWH_NFC + 1)*t(ND_CLK) >= tWH 2. (tWH_NFC + 1)*t(ND_CLK) >= tWC - (tWP_NFC + 1)*t(ND_CLK) 3. If NDCR.ECC_EN is true and NDECCCTRL.BCH_EN is true, tWH_NFC + tWP_NFC must always be greater or equal to 2.

Table 535: NAND Interface Timing Parameter 0 Register (NDTR0CS0) (Continued)
Offset: 0x000D0004

Bit	Field	Type/InitVal	Description
10:8	tWP_NFC	RW 0x1	NF_WEn pulse width The period for which NF_WEn is asserted low is specified in WE_Pulse width (tWP). The valid values for tWP_NFC are 0 through 7. Actual high duration of NF_WEn pulse would be (tWH_NFC +1) * t(ND_CLK). The NF_WEn cycle time is (tWH_NFC+tWP_NFC+2)*t(ND_CLK). Value from 0 to 7. Specifies the NF_WEn pulse width, in terms of NAND controller clock periods. NF_WEn pulse width = tWP_NFC + 1. Requirements (tCS, tCLS, tALS, tREA and tCEA refer to the parameters of the NAND flash data sheet): 1. (tWP_NFC + 1)*t(ND_CLK) >= tWP 2. (tWP_NFC + 1)*t(ND_CLK) >= tWC - (tWP_NFC + 1)*t(ND_CLK) 3. If NDCR.ECC_EN is true and NDECCCTRL.BCH_EN is true, tWH_NFC + tWP_NFC must always be greater or equal to 2.
7	sel_nRE_edge	RW 0x1	Select Read Strobe edge for Read data capture. This configuration select the relative edge that the Read Strobe Count Delay will start to count. 0 = Rising Edge: Select rising edge of NF_REn as reference to latch read data 1 = Falling Edge: Select falling edge of NF_REn as reference to latch read data.
6	etRP_NFC	RW 0x1	Extended tRP_NFC This bit is the MSB for the tRP_NFC value.
5:3	tRH_NFC	RW 0x1	NF_REn high duration The period for which NF_REn remains high while NF_REn is being toggled to access data during a read operation is specified by NF_REn high duration (tRH). The valid values for tRH_NFC are 0 through 7. Actual high duration of NF_REn is (tRH_NFC +1) * t(ND_CLK). Value from 0 to 7. Specifies the NF_REn high duration, in terms of NAND controller clock periods. NF_REn high duration = tRH_NFC+1. **NOTE:** This is the minimum duration that is guaranteed. The maximum duration for which the NF_REn is high is greater than ((tRH_NFC + 1)*t(ND_CLK)) if the DFI bus is granted to SMC during this time. Requirements (tRH and tRC refer to the parameters of the NAND flash data sheet): 1. (tRH_NFC + 1)*t(ND_CLK) >= tRH 2. (tRH_NFC + 1)*t(ND_CLK) >= tRC - ({etRP_NFC,tRP_NFC}+1)*t(ND_CLK) 3. If NDCR.ECC_EN is true and NDECCCTRL.BCH_EN is true, tRH_NFC + {etRP_NFC,tRP_NFC} must always be greater or equal to 2.

Table 535: NAND Interface Timing Parameter 0 Register (NDTR0CS0) (Continued)

Offset: 0x000D0004

Bit	Field	Type/InitVal	Description
2:0	tRP_NFC	RW 0x1	NF_REn pulse width These 3 bits in conjunction with bit 6 above (etRP_NFC) determine the period for which NF_REn is asserted low. The valid values for tRP_NFC are 0 through 16. Actual high duration of NF_REn pulse is {etRP_NFC,tRP_NFC}+1 NAND controller clock cycles. The NF_REn cycle time would be tRH_NFC+tRP_NFC+2 NAND controller clock cycles. Value from 0 to 16. Specifies the NF_REn pulse width, in terms of NAND controller clock periods. NF_REn pulse width = {etRP_NFC, tRP_NFC} +1. Requirements (tRP and tRC refer to the parameters of the NAND flash data sheet): 1. ({etRP_NFC,tRP_NFC} + 1)*t(ND_CLK) >= tRP 2. ({etRP_NFC,tRP_NFC} + 1)*t(ND_CLK) >= tRC - (tRH_NFC + 1)*t(ND_CLK) 3. If NDCR.ECC_EN is true and NDECCCTRL.BCH_EN is true, tRH_NFC + {etRP_NFC,tRP_NFC} must always be greater or equal to 2.

Table 536: NAND Interface Timing Parameter 1 Register (NDTR1CS0)

Offset: 0x000D000C

Bit	Field	Type/InitVal	Description
This register can be programmed to set the timing information for read, status read, and read-ID commands. Accesses to NAND flash devices interfaced using NF_CEn are controlled by the settings in NDTR1CS0. All timing parameters in NDTR1CS0 are set in terms of NAND controller-clock periods. The NAND controller clock runs at the frequency of the NFC clock (ND_MCLK). Refer to the Hardware Specifications for frequency limitation of the NAND flash controller timing parameters. When 16b_NFC is set to ON, although there is only one definition register, because it is possible to overlap operations on the chip selects, there are two actual timers, one for chip select 0, 2 and one for chip select 1, 3. This is a read/write register. Ignore reads from reserved bits. Write 0 to reserved bits.			
31:16	tR_NFC	RW 0x1	NF_WEn high to NF_REn Low for Read Value from 1 to 65535. Specifies additional delay between NF_WEn high and NF_REn low for read, in terms of NAND controller clock periods. This delay depends on the value programmed in NDTR0CS0.tCH_NFC in addition to tR_NFC programmed value. Requirements (tR and tWB refer to the parameters of the NAND flash data sheet). 1. (tR_NFC+2)*t(ND_CLK) >= tR - (tCH_NFC + 1)*t(ND_CLK) 2. If NDTR1CS0.WAIT_MODE is true then (tR_NFC >= tWB)

Table 536: NAND Interface Timing Parameter 1 Register (NDTR1CS0) (Continued)

Offset: 0x000D000C

Bit	Field	Type/InitVal	Description
15	Wait_Mode	RW 0x0	Determines how the information in this register is used to define the state machine transitions on read operations. For all reads, the timer always counts after the read command has been issued and this bit only determines the actual state machine proceed condition. **NOTE:** This bit may be changed from a 1 to a 0 while an operation is in progress. The reason for this is to provide a mechanism to reset in the event of RnB deadlock. If for some reason, RnB should stay de-asserted, resetting this bit after the time has expired will allow the state machine to proceed. This will release the controller and the NAND device should be issued a Reset command. In the event of RnB lockup, the result of the read operation is not defined. 0 = Specific: The tR_NFC time value specified in this register solely determines the duration. When this count has not expired, the state machine stalls and when the count reaches tR_NFC, regardless of the state of RnB, the operation will proceed. In other words RnB is ignored. 1 = Minimum: The tR_NFC time value specified in this register determines the minimum duration. When this count has not expired, the state machine stalls. When this count has expired, there may be an additional stall if RnBx is still indicating busy. The proceed condition is the AND of the time out from the defined timer and RnBx i.e. to proceed, both the time value must have expired and RnBx must be signaling “ready”.
14	PRESCALE	RW 0x0	Prescale tR_NFC Regardless of the setting of the PRESCALE bit, a tR_NFC value of zero produces a minimum of ND_CLK clock delay. A value for tR_NFC of 1 produces a two ND_CLK clock additional delay without PRESCALE and a 17 ND_CLK clock additional delay with PRESCALE. 0 = Whole NFC Clocks: The tR_NFC time value specified in this register is the number of NFC clocks to wait to begin to sample tR (in Wait_ Mode) or to simply wait (when not in Wait_Mode). 1 = Fraction of NFC Clocks: The tR_NFC time value specified in this register is 1/16 of the number of NFC clocks i.e. the tR_NFC value is prescaled by 16.
13:10	Reserved	RSVD 0x0	Reserved

Table 536: NAND Interface Timing Parameter 1 Register (NDTR1CS0) (Continued)

Offset: 0x000D000C

Bit	Field	Type/InitVal	Description
9:8	tRHW_NFC	RW 0x0	Read to Write Delay The tRHW parameter is the delay between the rising edge of a NF_REn strobe to the falling edge of a NF_WEn strobe. This situation occurs when a read command (of any type) is followed by another command. The NF_REn generates a read cycle that must be completely removed from the bus (i.e. tri-state) before the host can drive a command to write. The value of this field is multiplied by 16 to determine the number of clocks to guarantee between reads and the next command. 0 = None: 0 ND_CLK periods added 1 = 16: 16 ND_CLK periods added 2 = 32: 32 ND_CLK periods added 3 = 48: 48 ND_CLK periods added
7:4	tWHR_NFC	RW 0x1	NF_WEn High to NF_REn Low for a Read Status For a status-read command, NF_WEn high to NF_REn Low Delay (tWHR) specifies the delay between de-assertion of NF_WEn and assertion of NF_REn. Value from 0 to 15. Specifies the delay between NF_CLE high and NF_REn low for status read, in terms of ND_CLK periods. Requirement (tWHR refer to the parameter of the NAND flash data sheet). The actual delay depends on tAR_NFC, tWH_NFC, tWHR_NFC, and tCH_NFC: (max(tWH_NFC, tCH_NFC) + max(tAR_NFC, max(0, tWHR_NFC-max(tWH_NFC,tCH_NFC))) + 2) * t(ND_CLK) >= tWHR
3:0	tAR_NFC	RW 0x1	NF_ALE Low to NF_REn Low Delay For a Read-ID command, NF_ALE Low to NF_REn Low delay (tAR) specifies the delay between de-assertion of NF_ALE and assertion of NF_REn. Requirement (tAR refer to the parameter of the NAND flash data sheet). The actual delay depends on tAR_NFC, tWH_NFC, tWHR_NFC, and tCH_NFC: (max(tWH_NFC, tCH_NFC) + max(tAR_NFC, max(0, tWHR_NFC-max(tWH_NFC,tCH_NFC))) + 2) * t(ND_CLK) >= tAR

Table 537: NAND Controller Status Register (NDSR)

Offset: 0x000D0014

<table>
<tr><th>Bit</th><th>Field</th><th>Type/InitVal</th><th>Description</th></tr>
<tr><td colspan="4">This register contains bits that signal flash ready status, bad block detected, read-data-error conditions, data-read/write requests, completion of a page, and command. Unless masked, each of these hardware-detected events signals an interrupt request to the interrupt controller. Once set, each of the status bits is cleared by writing 1’b1 to the corresponding bit position. Writing 1’b0 has no effect on the status bits. Each of the NDSR bits signals an interrupt request as long as the bit is set and the corresponding interrupt is not masked. Once the bit is cleared, the interrupt is cleared.
NDSR status bits are set in both DMA and non-DMA modes of operation, and they can be polled and cleared by software in either mode.
This is a read/write register. Ignore reads from reserved bits. Write 1’b0 to reserved bits.
<b>NOTE:</b> Care must be taken to only clear bits known to be set because it is possible that between the NDSR read and the subsequent write to clear the status, additional conditions will be set. If NDCR.ND_RUN is true, it is almost never correct to simply write 0x9FFF to the register (i.e. reset all bits). Doing so will most likely create the situation where the next expected event happens to assert just before the write to clear a previous event. This next event then gets cleared, and a software deadlock will occur.
The following examples show how the status register should be cleared
NDSR = NDSR_WRCMDREQ; // reset the write command request status
NDSR = NDSR_WRCMDREQ | NDSR_CS0_CMDD; // reset command req and cmdd
NDSR = *NDSR; // reset all currently set status bits</td></tr>
<tr><td>31:21</td><td>Reserved</td><td>RSVD
0x0</td><td>Reserved</td></tr>
<tr><td>20:16</td><td>ERR_CNT</td><td>RO
0x0</td><td>Block Error Count
If NDCR.ECC_EN is true, NDECCCTRL.BCH_EN is true an ECC check has been performed and (ERR_CNT > NDECCCTRL.ECC_THRESH), these bits indicate the number of errors encountered. If ECC_EN or BCH_EN are false, or (ERR_CNT <= ECC_THRESH) this field is 0x00. The error count is the number of non-zero syndromes. Writing 0x1f into this field will reset the error count. If the ERR_CNT field is 0x1f the errors are not correctable and UNCERR will also be set.
<b>NOTE:</b> If it is desired to see the BCH error count in all cases, it will be necessary to set the ECC_THRESH to 0.
<b>NOTE:</b></td></tr>
<tr><td>15</td><td>TRUSTVIO</td><td>RW
0x0</td><td>Trust Violation
This bit is set when a trusted partition (or illegal operation) access violation occurs as defined by the NDCMDMAT{0,1,2} and NDPT{0..3}CS{0,1} registers.</td></tr>
<tr><td>14:13</td><td>Reserved</td><td>RSVD
0x0</td><td>Reserved</td></tr>
</table>

Table 537: NAND Controller Status Register (NDSR) (Continued)

Offset: 0x000D0014

Bit	Field	Type/InitVal	Description
12	RDY1	RW 0x0	NAND Flash Ready CS1 The NAND flash-ready (RDY) status bit is set when NF_RnB[1] (NAND Flash Ready/Busy) has made a low-to-high transition, indicating the ready state entry of flash device(s). The NAND flash-ready condition can be programmed to cause an interrupt by clearing the flash-device-ready interrupt mask (RDYM). This status bit is updated continuously each time NF_RnB[1] input makes a low-to-high transition, regardless of the type of command sent to the flash device. When 16b_NFC is set to ON, in a ganged configuration (two 8-bit parts connected as a logical 16-bit part, where the RnB signals of each part are wire ANDed) the RDY reflects the worst case time from the two ganged parts. **NOTE:** Although the reset value of RDY1 is 0, this bit will be updated as soon as the NFC interface clock begins pulsing so that NDSR will most likely be set to 0x00001800 by the time it is captured. Both bits [12] and [11] are set together and should be reset together when there is not dual NF_RnB pins. 0 = No Transtition: The Ready/Busy_ input has not transitioned from low to high. 1 = Transition: The Ready/Busy_ input has transitioned from low to high.
11	RDY0	RW 0x0	NAND Flash Ready CS0 The NAND flash-ready (RDY) status bit is set when NF_RnB[0] (NAND Flash Ready/Busy) has made a low-to-high transition, indicating the ready state entry of flash device(s). The NAND flash-ready condition can be programmed to cause an interrupt by clearing the flash-device-ready interrupt mask (RDYM). This status bit is updated continuously each time NF_RnB[0] input makes a low-to-high transition, regardless of the type of command sent to the flash device. **NOTE:** Although the reset value of RDY0 is 0, this bit will be updated as soon as the NFC interface clock begins pulsing so that NDSR will most likely be set to 0x00001800 by the time it is captured. In a ganged configuration (two 8-bit parts connected as a logical 16-bit part, where the NF_RnB signals of each part are wire ANDed) the RDY reflects the worst case time from the two ganged parts. 0 = No Transtition: The Ready/Busy_ input has not transitioned from low to high. 1 = Transition: The Ready/Busy_ input has transitioned from low to high.

Table 537: NAND Controller Status Register (NDSR) (Continued)
Offset: 0x000D0014

Bit	Field	Type/InitVal	Description
10	CS0_PAGED	RW 0x0	NF_CSn[0,2] Page Done The CS0 page-done (CS0_PAGED) status bit is set when the read or write transaction to flash device(s) interfaced using NF_CSn[0,2] has completed a page of access including any user-defined spare-area access and ECC. CS0_PAGED is not set if an auto-status check after a program command results in a failure. CS0_PAGED bit is not set for read ID, read status, and erase commands. Page-done condition can be programmed to cause an interrupt by clearing the page- done interrupt mask (CS0_PAGEDM). 0 = No Boundary: The current read/write transaction on NF_CEn[0] has not reached a page boundary. 1 = Boundary: The current read/write transaction on NF_CEn[0] has reached a page boundary.
9	CS1_PAGED	RW 0x0	NF_CEn[1,3] Page Done The NF_CEn[1,3] page-done (CS1_PAGED) status bit is set when the read or write transaction to flash device(s) interfaced using NF_CEn[1,3] has completed a page of access including any user- defined spare-area access and ECC. CS1_PAGED is not set if an auto-status check after a program command results in a failure. The CS1_PAGED bit is not set for read ID, read status, and erase commands. Page-done condition can be programmed to cause an interrupt by clearing the page- done interrupt mask (CS1_PAGEDM). **NOTE:** Page-done status bits (CS0_PAGED and CS1_PAGED) should be used only for a page-related command like read or write. 0 = No Boundary: The current read/write transaction on NF_CEn[1] has not reached a page boundary. 1 = Boundary: The current read/write transaction on NF_CEn[1] has reached a page boundary.
8	CS0_CMDD	RW 0x0	NF_CSn[0,2] Command Done The NF_CSn[0,2] command-done (CS0_CMDD) bit is set when the execution of the command sent to the flash device(s) on NF_CSn[0,2] is completed successfully. CS0_CMDD is not set if an auto-status check after a program/erase command returns a failure. Command-done condition can be programmed to cause an interrupt by clearing the NF_ CSn[0,2] command-done mask (CS0_CMDDM). Command done is set when the command has finished being sequenced to the DFI, but whether or not a particular command is finished depends on the specific command. A read type command is done when it has been sequenced but it is still required that the data buffer be emptied, so setting CS0_CMDD does not signify that the NFC is ready for the next operation. 0 = Incomplete: The command execution on NF_CEn[0] has not successfully completed. 1 = Complete: The command execution on NF_CEn[0] has successfully completed.

Table 537: NAND Controller Status Register (NDSR) (Continued)

Offset: 0x000D0014

Bit	Field	Type/InitVal	Description
7	CS1_CMDD	RW 0x0	NF_CEn[1,3] Command Done The NF_CEn[1,3] command-done (CS1_CMDD) bit is set when the execution of the command sent to the flash device(s) on NF_CEn[1,3] is completed successfully. CS1_CMDD is not set if an auto-status check after a program/erase command returns a failure. Command-done condition can be programmed to cause an interrupt by clearing the NF_CEn[1,3] command-done mask (CS1_CMDDM). Command done is set when the command has finished being sequenced to the DFI, but whether or not a particular command is finished depends on the specific command. A read type command is done when it has been sequenced but it is still required that the data buffer be emptied, so setting CS0_CMDD does not signify that the NFC is ready for the next operation. 0 = Incomplete: The command execution on NF_CEn[1] has not successfully completed. 1 = Complete: The command execution on NF_CEn[1] has successfully completed.
6	CS0_BBD	RW 0x0	NF_CSn[0,2] Bad Block Detect The NF_CSn[0,2] bad-block detect (CS0_BBD) status bit is set when the status read after a program/erase operation to flash device(s) interfaced using NF_CSn[0,2] returns a program/erase failure. A bad-block detect condition can be programmed to cause an interrupt by clearing the NF_CSn[0,2] bad-block detect interrupt mask (CS0_BBDM). When two 8-bit NAND flash devices are interfaced to the 16-bit NAND flash controller data bus (NDCR[DWIDTH_C] = 1 and NDCR[DWIDTH_M] = 0) using NF_CSn[0,2], a program/erase failure in any one of the two devices results in the CS0_BBD bit getting set. Because both flash devices are addressed at the same time, software must mark the corresponding blocks as non-valid. 0 = No Bad Block: No bad block is encountered while a write/erase on NF_CEn[0]. 1 = Bad Block: Bad block is encountered while a write/erase on NF_CEn[0].
5	CS1_BBD	RW 0x0	NF_CEn[1,3] Bad Block Detect The NF_CEn[1,3] bad-block detect (CS1_BBD) status bit is set when the status read after a program/erase operation to flash device(s) interfaced using NF_CEn[1,3] returns a program/erase failure. A bad-block detect condition can be programmed to cause an interrupt by clearing the NF_CEn[1,3] bad block detect interrupt mask (CS1_BBDM). When two 8-bit NAND flash devices are interfaced to the 16-bit NAND flash controller data bus (NDCR[DWIDTH_C]= 1 and NDCR[DWIDTH_M] = 0) using NF_CEn[1,3], a program/erase failure in any one of the two devices results in CS1_BBD bit getting set. Because both flash devices are addressed at the same time, software can mark the corresponding blocks as non-valid. 0 = No Bad Block: No bad block is encountered while a write/erase on NF_CEn[1]. 1 = Bad Block: Bad block is encountered while a write/erase on NF_CEn[1].

Copyright © 2014 Marvell

Table 537: NAND Controller Status Register (NDSR) (Continued)
Offset: 0x000D0014

Bit	Field	Type/InitVal	Description
4	UNCERR	RW 0x0	Uncorrectable Error The uncorrectable error (UNCERR) status bit is set when an uncorrectable error is detected in any of the read data streams. Uncorrectable error condition can be programmed to cause an interrupt by clearing the uncorrectable error mask (UNCERRM). For NDECCCTRL.BCH_EN == 0 && NDCR.ECC_EN == 1 (i.e. Hamming ECC is enabled) an uncorrectable error is the detection of two errors in a syndrome range. Hamming codes are guaranteed to detect all possible double bit ECC error over their syndrome range. For NDECCCTRL.BCH_EN == 1&& NDCR.ECC_EN == 1 (i.e. BCH ECC is enabled) an uncorrectable error is the detection of more errors than be corrected. BCH codes are not guaranteed to detect all possible error cases in excess of their correction ability, and it is possible (although highly unlikely) for the BCH engine to report a false correction success. When an uncorrectable error is detected, the bad-block detect bits (CS0_BBD, CS1_BBD) are not set and NAND Control Bad Block registers (NDBBRx) are not updated. Software must take corrective action in this scenario. Software could read NDCBx registers to get the block address that returned erroneous data and mark this block as invalid. 0 = No error: No uncorrectable error is encountered in any of the page read data streams 1 = Error: Uncorrectable error is encountered in one of the page read data streams
3	CORERR	RW 0x0	Correctable Error The CORERR status bit is set when a correctable error is detected in any of the read data streams. A correctable error condition can be programmed to cause an interrupt by clearing the correctable error mask (CORERRM). For NDECCCTRL.BCH_EN == 0 && NDCR.ECC_EN == 1 (i.e. Hamming ECC is enabled) a correctable error is the detection of one error in a syndrome range. For NDECCCTRL.BCH_EN == 1 && NDCR.ECC_EN == 1 (i.e. BCH ECC is enabled) a correctable error is the detection of greater than NDECCCTRL.ECC_THRESH errors and less than or equal to the maximum number of errors the BCH can correct. **NOTE:** If CORERR is set, ERR_CNT will be set to 0x1f. 0 = No Error: No correctable error is encountered in any of the page read data streams. 1 = Error: Correctable error is encountered in one of the page read data streams.

Table 537: NAND Controller Status Register (NDSR) (Continued)

Offset: 0x000D0014

Bit	Field	Type/InitVal	Description
2	WRDREQ	RW 0x0	Write Data Request The write-data-request (WRDREQ) status bit is set when a write command is loaded into the NAND flash controller command buffer and data buffer is empty. The write data request status bit can be programmed to cause an interrupt by clearing the write-data-request mask (WRDREQM). 0 = No Write: No write to the data buffer is required. 1 = Write: Current command is a page write and data buffer has not been loaded.
1	RDDREQ	RW 0x0	Read Data Request The read-data-request (RDDREQ) status bit is set when the data buffer is loaded with a complete page (including spare data if SPARE_EN is set) of read data for a read operation. For read ID and read-status operations, RDDREQ is set when the data buffer is loaded with the requested number of read data bytes as specified in the command. When ECC is enabled for Hamming, RDDREQ is set once per entire data buffer transfer. When BCH ECC is enabled, RDDREQ is set once per 32 bytes of read data transfer. The processor must check that this bit is set after each 32 byte read from NDDB. 0 = No read: No read from the data buffer is required. 1 = Read: Data buffer has read data available.
0	WRCMDREQ	RW 0x0	Write Command Request The write-command-request (WRCMDREQ) status bit is set when one of the following conditions is met, indicating a request to write a command to the NAND flash controller command buffer. NDCR[ND_RUN] is set and the command buffer is empty. The Ready/Busy_ (R/B_) input is asserted low after the transfer of any command except read ID and read status, with 3 (next command) it set, to the addressed NAND flash device. After the completion of read-ID or read-status commands with next command (NC) bit set. Write command request can be programmed to cause an interrupt by clearing the write-command request mask (WRCMDREQM). 0 = No Write Required: No write to the command buffer is required. 1 = Write Required: New Command needs to be written to the command Buffer.

Table 538: NAND Controller Page Count Register (NDPCR)

Offset: 0x000D0018

Bit	Field	Type/InitVal	Description
This register can be read to determine the completed number of pages for multi-page program/read operations. Writes to NDPCR are ignored. This is a read-only register. Ignore reads from reserved bits.			
31:24	Reserved	RSVD 0x0	Reserved
23:16	PG_CNT_1	RO 0x0	Page count for device interfaced using NF_CSn[1,3] The page count for a flash device on NF_CSn[1,3] (PG_CNT_1) indicates the number of pages completed (programmed or read) for a given NAND flash-controller command. NDCB0[AUTO_RS] must be set for a multi-page program or multi-page read command. The page count gets reset when a command execution is finished. PG_CNT_1 is incremented after each page of data is programmed or read. However, if the status check after programming a page indicates a failure, PG_CNT_1 is not incremented and software can read this value to find out the last page programmed correctly and then set NDCR[CLR_PG_CNT] to clear the page count values.
15:8	Reserved	RSVD 0x0	Reserved
7:0	PG_CNT_0	RO 0x0	Page count for device interfaced using NF_CSn[0,2] Page count for Flash Device on NF_CSn[0,2] (PG_CNT_0) indicates the number of pages completed (programmed or read) for a given NAND flash controller command. NDCB0[AUTO_RS] must be set for a multi-page program or multi-page read command. The page count gets reset when a command execution is finished. PG_CNT_0 is incremented after each page of data is programmed or read. However, if the status check after programming a page indicates a failure, PG_CNT_0 is not incremented and software can read this value to find out the last page programmed correctly and then set NDCR[CLR_PG_CNT] to clear the page count values.

Table 539: NAND Controller Bad Block Registers (NDBBRx)

Offset: 0x000D001C

Bit	Field	Type/InitVal	Description
This register holds the bad-block information when a bad block is detected. NDBBR0 holds the bad-block information for a bad- block detected for a device interfaced using NF_CEn[0] (NDSR[CS0_BBD] is set). Similarly, NDBBR1 holds the bad-block information for a device interfaced using NF_CEn[1] (NDSR[CS1_BBD] set). These are read-only registers. Ignore reads from reserved bits.			
31:0	Bad_Block_ Information	RO 0x0	For a program command, bad block information contained in these registers consists of ADDR5, ADDR4, ADDR3, and ADDR2 for the command that resulted in a bad block detection. ADDR5 occupies the most significant byte and ADDR2 the least significant byte in NDBDRx. For an erase command bad block information contained in these registers consists of ADDR4, ADDR3, ADDR2, and ADDR1 for the command that resulted in a bad block detection. ADDR4 occupies the most significant byte and ADDR1 the least significant byte in NDBDRx. When a bad block is detected (the CS0_BBD or CS1_BBD bits in NDSR gets set), software reads the corresponding NDBDRx registers, uses the relevant portions of the bad-block information to get the bad-block address, and mark these blocks addresses as non-valid. Writes to the NDBBRx registers are ignored.

Table 540: NAND ECC Control (NDECCCTRL) Register

Offset: 0x000D0028

Bit	Field	Type/InitVal	Description
31:15	Reserved	RSVD 0x0	Reserved
14:7	ECC_SPARE	RO 0x0	ECC Bytes used in Spare Area This read-only value determines the number of bytes in the spare area that are dedicated to ECC storage by the hardware. The number of bytes is determined by NDCR.ECC_EN and BCH_EN. To compute the number of available spare bytes available to the programmer, take the spare size (16 bytes for a small block and 64 bytes for a large block) and subtract the value in ECC_SPARE. Note that this value is per chunk, so for a 4K page size that is read and written in two chunks this value specifies the ECC storage per chunk, not in total.
6:1	ECC_THRESH	RW 0x0	ECC Warning Threshold ECC threshold. When BCH_EN is true and ECC_EN in the NDCR is true, and the number of corrected errors exceeds this value, the NDSR.CORERR status is set and NDSR.ECC_CNT contains the number of corrected error bits. Currently, with a maximum of 16 errors correctable, values greater than 15 have the effect of disabling CORERR status. **NOTE:** If it is desired to always see all correctable errors, this field should be set to 0. **NOTE:**

Table 540: NAND ECC Control (NDECCCTRL) Register (Continued)

Offset: 0x000D0028

Bit	Field	Type/InitVal	Description
0	BCH_EN	RW 0x0	BCH Enable When ECC_EN in the NDCR is true, reads from a page will perform ECC to the following degrees of correctability: **NOTE:** There is an interaction between the ganging configuration and the ECC choice in that ganging 512 byte page size devices is only supported with the Hamming ECC engine and the ganging of 2048 (or 4096) byte page size devices is only supported with the BCH ECC engine. 0 = 2/8-bits: 2 bits per 512 byte page and 8 bits per 2048 byte page, distributed as one error in every other bit per 512 byte sector using a Hamming algorithm. 1 = 16-bits: 16 bits per 2048 byte page (plus spare area, if enabled).

Table 541: NAND Busy Length Count (NDBZCNT) Register

Offset: 0x000D002C

Bit	Field	Type/InitVal	Description
31:16	ND_RnBCNT1	RO 0x0	RnB Busy Count 1 This field is cleared to zero on the 1 -> 0 transition of NF_RnB[1] and while NF_RnB[1] is low, the field counts basic NFC controller clocks. When NF_RnB[1] has a 0 -> 1 transition, the count freezes.
15:0	ND_RnBCNT0	RO 0x0	RnB Busy Count 0 This field is cleared to zero on the 1 -> 0 transition of NF_RnB[1] and while NF_RnB[0] is low, the field counts basic NFC controller clocks. When NF_RnB[0] has a 0 -> 1 transition, the count freezes.

Table 542: NAND Controller Data Buffer (NDDB) Register
Offset: 0x000D0040

Bit	Field	Type/InitVal	Description
NAND controller data buffer (NDDB) is the read/write port of the NAND controller data buffer. NDDB is the source for read-data, and the destination for write-data. Only four byte writes and four byte reads are supported for software accesses to NDDB, mainly because the application processor cannot perform an 8-byte read. Read/write operations to NDDB must occur in the following order: The first write corresponds to the lower four bytes of the data buffer and the following operation corresponds to the upper four bytes. This register behaves as though it were a FIFO to the actual 2176 byte data buffer. The buffer contains 2KB for the “main” data area plus an additional 128 bytes to accommodate the currently defined spare size as well as allow for increases in the future. When ECC is enabled for Hamming, NDSR.RDDREQ is set once per entire data buffer transfer. When BCH ECC is enabled, NDSR.RDDREQ is set once per 32 bytes of data read transfer. The processor must check that this bit is set after each 32 byte read from NDDB. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.			
31:0	NAND_flash_data	RW 0x0	Holds the write/read data

Table 543: NAND Controller Command Buffer 0 (NDCB0) Register

Offset: 0x000D0048

Bit	Field	Type/InitVal	Description
The NAND controller-command buffer 0 (NDCB0) holds the commands (CMD1, CMD2) to be sent to the NAND flash device and command-control (CMD_CTRL) information. This register determines the overall semantic structure of the command sequence. This is a read/write register. Ignore reads from reserved bits. Write 0b0 to reserved bits.			
31:29	CMD_XTYPE	RW 0x0	Command Extended Type This field extends the command type semantic field and is defined only for read and write semantics. The read semantic is divided into three sections, command dispatch, data transfer (after ready is high) and DMA transfer. If the third section is disabled, the operation being performed is referred to as “read sniffing”. Here the purpose is to read a page and compute the number of ECC correction bits to make sure that the total correction is within the safe range of the NAND management algorithm. The write semantic is divided into three sections, initial command dispatch, data transfer and command issue. The reason for this extended field is to enable larger block support. All blocks larger than 2KB will composed of 2KB chunks, and to simplify the controller logic, these chunks are moved to and from the NAND as a separate command semantic. Read Semantics: 0b000 = Monolithic Read 0b001 = Last Naked Read 0b010 = Illegal 0b011 = Illegal 0b100 = Read 0b101 = Naked Read 0b110 = Command dispatch 0b111 = Illegal Write Semantics: 0b000 = Monolithic Write 0b001 = Naked Write with final command 0b010 = Illegal 0b011 = Final Command 0b100 = Command dispatch with write 0b101 = Naked Write 0b110 = Command dispatch 0b111 = Illegal

Table 543: NAND Controller Command Buffer 0 (NDCB0) Register (Continued)

Offset: 0x000D0048

Bit	Field	Type/InitVal	Description
28	LEN_OVRD	RW 0x0	Length Override Length override is used for non-standard length operations such as the ONFI parameter page read. 0 = Three Writes: Use the lengths defined by NDCR.PAGE_SZ for the operation. A command block consists of three writes and the NDLENCNT value in NDCB3 may not be written. 1 = Four Writes: Use the length defined in NDCB3.NDLENCNT for the operation, regardless of the programming of NDCR.PAGE_SZ. A command block consists of four writes and the NDCB3 register must be written.
27	RDY_BYP	RW 0x0	Ready Bypass The purpose for this function is to enable a real-time schedule of operations to the two chip selects where operations on one NAND can continue while operations on the other NAND are in progress. This is useful to enable striping where multiple identical operations are scheduled to the two chip selects. This bit is in addition to the capability mentioned in Command Sequence and Parallel Execution register, and allows virtually any operation pair to be overlapped. 0 = Wait: The current command will wait for the tR timeout as specified in NDTR1CS0.tR_NFC and NDTR1CS0.WAIT_MODE for the indicated CSEL to be true before proceeding according to the bypass rules. (In other words, the current command will wait for the selected RDY to be true.) 1 = Continue: The current command will continue to the next command (if there is one as defined by the NC bit), regardless of the state of RDY.
26	ST_ROW_EN	RW 0x0	Status Row Address Enable For ONFI NAND and other NAND parts, the status command may be performed on a per logical unit basis. The logical unit is addressed by the row address defined in the read, program or erase command. If this bit is set, both the automatic status cycle at the end of each read or write or any “manual” status read will supply the row address where the row address is defined by the ADDR_CYC bits and the RA_START bit in the NDCR. For read and program commands the ADDR_CYC defines the total number of address cycles and the RA_START bit defines which address bytes form the row address. For the erase command, the ADDR_CYC itself defines the number of row address cycles and the RA_START bit is ignored. It is expected that this bit is set for status commands and not for AUTO_RS. If multiple LUN program and erase commands are being issued, it will be necessary to issue LUN status commands with this bit enabled to determine precisely which LUN caused the failure.

Copyright © 2014 Marvell

Table 543: NAND Controller Command Buffer 0 (NDCB0) Register (Continued)

Offset: 0x000D0048

Bit	Field	Type/InitVal	Description
25	AUTO_RS	RW 0x0	Auto Read Status When set, the auto-read status bit (AUTO_RS) enables the automatic checking of the program/erase status by issuing a read status command. The bit pattern to be used for the status command is defined in NDCB2. When AUTO_RS is clear, no automatic status check is performed. When automatic status check is performed, status-read data is not written into the data buffer. The status-read data is checked by the NAND flash controller for the success of program/erase operation, and in case of failure, the appropriate bad-block-detected bits are updated in NDSR, the address of the bad block is saved in the NDBBRLx registers. Set AUTO_RS for program and erase commands only. Setting AUTO_RS for commands other than program or erase may result in incorrect operation by the NAND flash controller. AUTO_RS must be set for multipage program (when NDCB2[Page_ Count] is non-zero) to ensure that a read-status command is issued after every page program to verify the success of the program operation. 0 = No Automatic Read: No automatic read status command execution after program/erase. 1 = Automatic Read: Automatic read status command execution after program/erase.
24	CSEL	RW 0x0	CS Select The CS-select bit selects the chip-select signal (NF_CSn[0] or NF_ CSn[1]) to be activated for the command execution. 0 = NF_CSn[0]: NF_CSn[0] is asserted for the access. 1 = NF_CSn[1]: NF_CSn[1] is asserted for the access.

Copyright © 2014 Marvell

Document Classification: Proprietary Information

Table 543: NAND Controller Command Buffer 0 (NDCB0) Register (Continued)

Offset: 0x000D0048

Bit	Field	Type/InitVal	Description
23:21	CMD_TYPE	RW 0x0	Command Type Command type (CMD_TYPE) defines the type of NAND flash command represented by CMD1 (and CMD2 if for a double byte command). **NOTE:** No data transfer occurs when erase or reset commands are executed. The Naked Command issues CMD1 as a command cycle and continues. The reason for this semantic is to enable the creation of “outlier” command semantics such as “Read Unique ID”. The Naked Address issues ADDDR1 as a command cycle and continues. The reason for this semantic is to enable the creation of “outlier” command semantics such as “Read Unique ID”. The format of the data provided by a Read ID and Status Read is defined by the NAND manufacturer. Note that when devices are ganged, both the status and ID information is replicated on each half of each 16-bit word. Note also that for status, the value for each part in the gang is device dependent. 0 = Read 1 = Program (Write) 2 = Erase 3 = Read ID 4 = Status Read 5 = Reset 6 = Naked Command 7 = Naked Address
20	NC	RW 0x0	Next Command The next-command (NC) bit (if set) indicates the presence of another valid command following the current command. The last command of a command sequence must have the NC bit clear. 0 = No Valid Command: No valid command following the current command. 1 = Valid Command: Valid command following the current command.
19	DBC	RW 0x0	Double Byte Command The double-byte command (DBC) bit (if set) indicates that the current command involves the transfer of two commands to the NAND flash. DBC must be clear for a single-byte command. 0 = Single-Byte: Current command is a single-byte command. 1 = Double-Byte: Current command is a double-byte command.

Copyright © 2014 Marvell

Table 543: NAND Controller Command Buffer 0 (NDCB0) Register (Continued)

Offset: 0x000D0048

Bit	Field	Type/InitVal	Description
18:16	ADDR_CYC	RW 0x0	Number of Address Cycles Address cycles (ADDR_CYC) specifies the number of address cycles in which the NAND flash is to be addressed. Valid values for address cycles are 1 to 5 for NAND commands Program, Read, Erase and Read-Id. For example, if ADDR_CYC value is programmed as 4, ADDR1, ADDR2, ADDR3, and ADDR4 are sent to the flash device in successive address cycles, while ADDR5 is ignored. Valid values for address cycles for Reset and Read-status commands is 0. If the SW programs a non-zero value for these commands the NAND controller will simply ignore it.
15:8	CMD2	RW 0x0	Second Command The second command (CMD2) contains the second byte of command sent to NAND flash after CMD1 and address in a double-byte command.
7:0	CMD1	RW 0x0	First command The first command (CMD1) contains the first byte command sent to the NAND flash when the command execution begins.

Table 544: NAND Controller Command Buffer 1 (NDCB1) Register

Offset: 0x000D004C

Bit	Field	Type/InitVal	Description
This register contains the addresses to be sent to the NAND flash in multiple cycles. In NAND-flash mode, I/O<7:0> pins are used to send ADDRx<7:0>, in as many cycles as defined by NDCB0[ADDR_CYC]. The ADDR5 field in NAND controller command buffer 2 (NDCB2) is used to specify the fifth cycle of address for devices supporting five addressing cycles. Contents of the ADDRx fields are sent out sequentially by the NAND controller during the addressing phase. Program these fields with address values based on the cycle-by-cycle addressing specified by the NAND-flash device. When programming an erase command the contents of ADDR1 must be replicated in ADDR5 in order to save the full address during a bad block detect since only fields ADDR2 through ADDR5 are saved. When programming a manual read status command the user must write the ADDRx fields with the same address as the previous command in order to save a valid address to NDBDRx during a bad block detect. This is a read-only register. Ignore reads from reserved bits.			
31:24	ADDR4	RO 0x0	Address sent out to the flash device on the fourth addressing cycle.
23:16	ADDR3	RO 0x0	Address sent out to the flash device on the third addressing cycle.
15:8	ADDR2	RO 0x0	Address sent out to the flash device on the second addressing cycle.
7:0	ADDR1	RO 0x0	Address sent out to the flash device on the first addressing cycle.

Table 545: NAND Controller Command Buffer 2 (NDCB2) Register
Offset: 0x000D0050

Bit	Field	Type/InitVal	Description
This register holds the address (ADDR5) for the fifth addressing cycle in NAND-flash mode and the page count for the command. This is a read-only register. Ignore reads from reserved bits.			
31:24	ST_MASK	RO 0x0	Status Mask This is the mask applied to the status field to determine failure. This field value is ANDed with the status word received from the NAND device and the resulting byte is bitwise ORed over the [7:0] field. If the result is true (i.e. any bit allowed by ST_MASK is set), the status is indicating failure. A Status Mask of 0x00 is singular in that it defines a mask of 0x01 instead. If software issues 0x00 to this field, the value actually loaded will be 0x01. Receiving bad status sets one of the CSx_BBD bits in NDSR (depending on the active chip select) when AUTO_RS is set. Generally, this field should only be set to a non-standard value when performing non-standard status commands as a stands-alone command.
23:16	ST_CMD	RO 0x0	Status Command The default status command is 0x70 but ONFI and other devices allow logical unit status checks. This field allows the non-standard status command to be defined for both status commands as well as auto status reads. A Status Command of 0x00 is singular in that it defines a status command of 0x70 instead. If software issues 0x00 to this field, the value actually loaded will be 0x70. While it is possible to use LUN status for AUTO_RS, there is only one ST_CMD, ST_MASK and ST_ROW_EN value, but there are two AUTO_RS for overlapped operations. This means that one cannot have different values for these fields on any pair of operations that can potentially overlap. For this reason, the use of LUN status during multiple chained operations is discouraged. Instead, the NFC should perform simple status operations, which will perform the “OR” of all logical operations in the NAND and when a bad block is detected, software can issue LUN status commands to determine exactly which LUN operation actually failed.

Copyright © 2014 Marvell

Table 545: NAND Controller Command Buffer 2 (NDCB2) Register (Continued)

Offset: 0x000D0050

Bit	Field	Type/InitVal	Description
15:8	Page_Count	RO 0x0	Page Count Page_Count specifies the number of pages of data to be transferred for a program or read command. Set this field to zero for all commands except multipage program/read commands (commands where more than one page is programmed or read), in which case Page_Count + 1 specifies the number of pages serviced. Value from 0 to 255. Specifies the number of pages of data to be transferred for the current command. This field must not be set to a larger value than allowed by NDCR.PG_PER_BLK. Number of pages to be transferred = Page_Count +1 **NOTE:** While multi-page transfers reduce the number of command cycle that need to be programmed into NDCBx, they do not necessarily improve throughput because each operation to the NAND is treated separately. (Refer to the description of <SEQ_DIS> in Table 381 on page 299). To truly optimize throughput requires cache instructions and/or LUN commands and a carefully constructed real time schedule. **NOTE:**
7:0	ADDR5	RO 0x0	Address sent out to the flash device on the fifth addressing cycle. 0 = CS0_1: CS0 or CS1 selected according to the value NDCB0 <CSEL> bit[24]. 1 = CS2_3: CS2 or CS3 selected according to the value NDCB0 <CSEL> bit[24].

Table 546: NAND Controller Command Buffer 3 (NDCB3) Register

Offset: 0x000D0054

Bit	Field	Type/InitVal	Description
For increased flexibility and to support ONFI, this register allows arbitrary length read data transfers. For any transfer that uses this arbitrary length count, NDCR.SPARE_EN must be disabled or the result will be unpredictable. If NDCR.ECC_EN is enabled, ECC is applied as described elsewhere. The register address is after the NDCBx registers and the register is written as an optional extension of writing a command block to NDCB0. If NDCB0.LEN_OVRD is false and NDCB0.ADDR_CYC < 6, this register must not be written and a fourth write to NDCB0 is ignored. If NDCB0.LEN_OVRD is true or NDCB0.ADDR_CYC >= 6, this register must be written, even if it is not changing value.			
31:24	ADDR7	RO 0x0	Address sent out to the flash device on the seventh addressing cycle.
23:16	ADDR6	RO 0x0	Address sent out to the flash device on the sixth addressing cycle.

Table 546: NAND Controller Command Buffer 3 (NDCB3) Register (Continued)

Offset: 0x000D0054

Bit	Field	Type/InitVal	Description
15:0	NDLENCNT	RO 0x0	When the arbitrary length count is selected, this value is used to count the data cycles for any and all selected operations. As a practical matter, this value should not exceed the 2176 byte buffer size or data will overflow the data FIFO. When ECC is enabled and BCH is selected, this value must be a multiple of 32 bytes. **NOTE:** The value 0 is defined to be valid, and when this field is zero, no data transfer is performed. **NOTE:**

Copyright © 2014 Marvell

A.5 Device Bus Registers

The following table provides a summarized list of all registers that belong to the Device Bus, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 547: Summary Map Table for the Device Bus Registers

Register Name	Offset	Table and Page
DEV_BOOTCSn Read Parameters Register	0x00010400	Table 548, p. 889
DEV_BOOTCSn Write Parameters Register	0x00010404	Table 549, p. 890
DEV_CSn[n] Read Parameters Register (n=0–3)	CS0: 0x00010408, CS1: 0x00010410, CS2: 0x00010418, CS3: 0x00010420	Table 550, p. 891
DEV_CSn[n] WriteParameters Register (n=0–3)	CS0: 0x0001040C, CS1: 0x00010414, CS2: 0x0001041C, CS3: 0x00010424	Table 551, p. 892
Device Bus Interface Control Register	0x000104C0	Table 552, p. 893
Device Bus Error Address Register	0x000104C4	Table 553, p. 893
Device Bus Sync Control Register	0x000104C8	Table 554, p. 894
Device Bus Interrupt Cause Register	0x000104D0	Table 555, p. 895
Device Bus Interrupt Mask Register	0x000104D4	Table 556, p. 896

Table 548: DEV_BOOTCSn Read Parameters Register

Offset: 0x00010400

Bit	Field	Type/InitVal	Description
31:30	DevWidth	RW SAR	Boot Device Width 0 = 8-bit 1 = 16-bit 2 = 32-bit 3 = Reserved
29:28	Reserved	RSVD 0x0	Reserved
27:23	RdHold	RW 0x0	Defined the number of TCLK cycles in read accesses between de-assertion of DEV_OEn and de-assertion of DEV_BOOTCSn. Set <RdHold> to a value smaller than <TurnOff>.- 2
22:17	Acc2Next	RW 0x3f	Defines the number of TCLK cycles between the cycle that samples data N and the cycle that samples data N+1 (in burst accesses). Minimal value = 0x2

Copyright © 2014 Marvell

Table 548: DEV_BOOTCSn Read Parameters Register (Continued)

Offset: 0x00010400

Bit	Field	Type/InitVal	Description
16:12	RdSetup	RW 0x0	Defined the number of TCLK cycles in read accesses between the end of the address phase and the assertion of DEV_OEn. End of address phase occurs one cycle after DEV_ALE[0] negation.
11:6	Acc2First	RW 0x3f	Defines the number of TCLK cycles in a read access between the negation of DEV_ALE[0] and the cycle containing the first data sampled by the device. Number of cycles = <Acc2First> - 2 * <Addr2Ale> - 1. Minimal value = 2 * <Addr2Ale> + 2 + <RdSetup> + 2
5:0	TurnOff	RW 0x0f	Defines the minimal gap guaranteed between two consecutive device accesses. The number of TCLK cycles between de-assertion of DEV_BOOTCSn and the beggining of the next address phase will be atleast <TurnOff> - <RdHold>. Minimal value for <TurnOff> is <RdHold>+2

Table 549: DEV_BOOTCSn Write Parameters Register

Offset: 0x00010404

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24	DevSyncEnable	RW 0x0	Synchronous device enable 0 = False 1 = True
23:22	Reserved	RSVD 0x0	Reserved
21:16	WrHigh	RW 0xf	The number of TCLK cycles in a burst write access that DEV_WEn is kept de-asserted. Minimal value = 0x1
15:14	Reserved	RSVD 0x0	Reserved
13:8	WrLow	RW 0xf	The number of TCLK cycles in a write access that DEV_WEn is kept active. Minimal value = 0x1
7:6	Reserved	RSVD 0x0	Reserved

Table 549: DEV_BOOTCSn Write Parameters Register (Continued)

Offset: 0x00010404

Bit	Field	Type/InitVal	Description
5:0	ALE2Wr	RW 0xf	Defines the number of TCLK cycles in a write access between DEV_ ALE[0] negation and the assertion of DEV_WEn. Number of cycles = <ALE2Wr> - 2 * <Addr2ALE> - 1. Minimal value = 2 * <Addr2ALE> + 2

Table 550: DEV_CSn[n] Read Parameters Register (n=0–3)

Offset: CS0: 0x00010408, CS1: 0x00010410, CS2: 0x00010418, CS3: 0x00010420

Offset Formula: 0x00010408+n * 8: where n (0-3) represents CS

Bit	Field	Type/InitVal	Description
31:30	DevWidth	RW 0x0	Device Width 0 = 8-bit 1 = 16-bit 2 = 32-bit 3 = Reserved
29:28	Reserved	RSVD 0x0	Reserved
27:23	RdHold	RW 0x0	Defined the number of TCLK cycles in read accesses between de-assertion of DEV_OEn and de-assertion of DEV_CSn. Minimum value=0 Maximal value=<TurnOff> - 2
22:17	Acc2Next	RW 0x1F	Defines the number of TCLK cycles between the cycle that samples data N and the cycle that samples data N+1 (in burst accesses). Minimal value = 0x2
16:12	RdSetup	RW 0x0	Defined the number of TCLK cycles in read accesses between the end of the address phase and the assertion of DEV_OEn. End of address phase occurs one cycle after DEV_ALE[0] negation.
11:6	Acc2First	RW 0x1F	Defines the number of TCLK cycles in a read access between the negation of DEV_ALE[0] and the cycle containing the first data sampled by the device. Number of cycles = Acc2First - 2*Addr2Ale - 1 Minimal value = 2 * Addr2Ale + 2 + RdSetup + 2

Copyright © 2014 Marvell

Table 550: DEV_CSn[n] Read Parameters Register (n=0–3) (Continued)

Offset: CS0: 0x00010408, CS1: 0x00010410, CS2: 0x00010418, CS3: 0x00010420

Offset Formula: 0x00010408+n * 8: where n (0-3) represents CS

Bit	Field	Type/InitVal	Description
5:0	TurnOff	RW 0x0f	Defines the minimal gap guaranteed between two consecutive device accesses. The number of TCLK cycles between de-assertion of DEV_CSn and the beggining of the next address phase will be atleast <TurnOff> - <RdHold>. Minimal value for <TurnOff> is <RdHold>+2

Table 551: DEV_CSn[n] WriteParameters Register (n=0–3)

Offset: CS0: 0x0001040C, CS1: 0x00010414, CS2: 0x0001041C, CS3: 0x00010424

Offset Formula: 0x0001040C+ 8 * n: where n (0-3) represents CS

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24	DevSyncEnable	RW 0x0	Synchronous device enable 0 = False 1 = True
23:22	Reserved	RSVD 0x0	Reserved
21:16	WrHigh	RW 0xf	The number of TCLK cycles in a burst write access that the DEV_WEn signal is kept de-asserted. Minimal value = 0x1
15:14	Reserved	RSVD 0x0	Reserved
13:8	WrLow	RW 0xf	The number of TCLK cycles in a write access that the DEV_WEn signal is kept active. Minimal value = 0x1
7:6	Reserved	RSVD 0x0	Reserved
5:0	ALE2Wr	RW 0xf	Defines the number of TCLK cycles in a write access between DEV_ALE[0] negation and the assertion of DEV_WEn. Number of cycles = ALE2Wr - 2*Addr2ALE - 1. Minimal value = 2*Addr2ALE + 2

Copyright © 2014 Marvell

Table 552: Device Bus Interface Control Register

Offset: 0x000104C0

Bit	Field	Type/InitVal	Description
31:20	Reserved	RSVD 0x0	Reserved
19	ForceParityErr	RW 0x0	By setting this bit, the device controller internal buffer will act as if the memory is corrupted. Thus, upon the use of these buffers, DPErrInt will be set. This bit is a debug ability and is provided to support software development for implementing interrupt handling of an internal SRAM parity detection.
18:17	Addr2ALE	RW 0x2	Defines the number of TCLK cycles in which address is steady before ALE assertions. **NOTE:** Addr2ALE imposes a limitation of the timing parameters that exist for each CS, since Acc2First and ALE2First must be configured such that they are expired after address latching is done. 2 = 3cycles: Address will be steady 3 core clock cycle before ALE assertion 3 = Reserved: Invalid value
16	OEWE_shared	RW SAR	Defines if OE and WE are latched at first ALE cycle as A[15] and A[16]. This fact influences the OEn and WEn signal as following: 0 - OE and WE are not shared whenever CS is inactive OE and WE are inactive. 1 - OE and WE are shared with A[16:15]. Whenever CS is inactive and ALE[1:0] are high, OE and WE are inactive.
15:0	Timeout	RW 0xFFFF	Timeout Timer Preset Value If the device access is not completed within the period of this preset value (due to a lack of READYn assertion), the Device controller completes the transaction as if READYn was asserted, and it asserts an interrupt. **NOTE:** If set to 0x0, the Device controller waits for READYn assertion forever. The minimum value for the timeout is 0x40, thus exceeding all timing parameters values.

Table 553: Device Bus Error Address Register

Offset: 0x000104C4

Bit	Field	Type/InitVal	Description
31:0	Addr	RO 0x0	Latched address upon DReadyErr event After the address is latched, no new address is latched until the register is being read.

Table 554: Device Bus Sync Control Register

Offset: 0x000104C8

Bit	Field	Type/InitVal	Description
This register provides support for a Synchronios device.			
31	TclkDivLoadActive	RO 0x0	When updating the Tclk Divider Value, this bit indicates that action has been completed, and DEV_CLK_OUT is indeed driving the core clock with the requested division. Software must poll this bit after reloading a new TCLK_DIV_VAL. 0 = LoadNonActive: Tclk Divider is updated. 1 = LoadActive: Tclk Divider is not updated.
30:26	Reserved	RSVD 0x0	Reserved
25	CS3 Ready Polarity	RW 0x0	See BootCS Polarity 0 = Active Low 1 = Active High
24	CS3 Ready Ignore	RW 0x0	See BootCS Ignore
23:21	Reserved	RSVD 0x0	Reserved
20	CS2 Ready Polarity	RW 0x0	See BootCS Polarity 0 = Active Low 1 = Active High
19	CS2 Ready Ignore	RW 0x0	See BootCS Ignore
18:16	Reserved	RSVD 0x0	Reserved
15	CS1 Ready Polarity	RW 0x0	See BootCS Polarity 0 = Active Low 1 = Active High
14	CS1 Ready Ignore	RW 0x0	See BootCS Ignore
13:11	Reserved	RSVD 0x0	Reserved
10	CS0 Ready Polarity	RW 0x0	See BootCS Polarity 0 = Active Low 1 = Active High
9	CS0 Ready Ignore	RW 0x0	See BootCS Ignore
8:6	Reserved	RSVD 0x0	Reserved

Table 554: Device Bus Sync Control Register (Continued)

Offset: 0x000104C8

Bit	Field	Type/InitVal	Description
5	BootCS Ready Polarity	RW 0x0	Defines the polarity of DEV_READYn when BootCS is accessed. 0 = Active Low 1 = Active High
4	BootCS Ready Ignore	RW 0x0	When set, DEV_READYn is ignored.
3:0	Tclk Divide Value	RW 0x1	Defines the DEV_CLK_OUT frequency in terms of the core clock divider values. For example, If the core clock frequency is 166 MHz, the following encoding holds: 0 = Disable: DEV_CLK_OUT will not toggle. 2 = 1to2 3 = 1to3 4 = 1to4 5 = 1to5 6 = 1to 6 7 = 1to7 8 = 1to8 9 = 1to9 10 = 1to10 11 = 1to11 12 = 1to12 13 = 1to13 14 = 1to14 15 = 1to15

Table 555: Device Bus Interrupt Cause Register

Offset: 0x000104D0

Bit	Field	Type/InitVal	Description
NOTE: All cause bits are clear only. They are set upon error condition cleared upon a value write of 0. Writing a value of 1 has no affect.			
31:2	Reserved	RSVD 0x0	Reserved
1	DRdyErr	RW0C 0x0	Ready Timer expired
0	RuDPErr	RW0C 0x0	Internal Device Controller SRAM’s Parity Error.

Copyright © 2014 Marvell

Table 556: Device Bus Interrupt Mask Register

Offset: 0x000104D4

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	MaskReadyErr	RW 0x0	Masks the assertion of an interrupt in case of a Ready Timeout event Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register.
0	MaskParityErr	RW 0x0	Masks the assertion of an interrupt in case of an internal SRAM parity error event. Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register.

Copyright © 2014 Marvell

A.6 Ethernet Networking Controller and MAC Registers

The following table provides a summarized list of all registers that belong to the Ethernet Networking Controller and MAC, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
COM PHY		
Power and PLL Control Register (i=0–3)	Port2: 0x00032E04, Port3: 0x00036E04, Port0: 0x00072E04, Port1: 0x00076E04	Table 558, p. 912
KVCO Calibration Control Register (i=0–3)	Port2: 0x00032E08, Port3: 0x00036E08, Port0: 0x00072E08, Port1: 0x00076E08	Table 559, p. 913
Generation 1 Setting 0 Register (i=0–3)	Port2: 0x00032E34, Port3: 0x00036E34, Port0: 0x00072E34, Port1: 0x00076E34	Table 560, p. 913
Generation 2 Setting 0 Register (i=0–3)	Port2: 0x00032E3C, Port3: 0x00036E3C, Port0: 0x00072E3C, Port1: 0x00076E3C	Table 561, p. 913
PHY Test Control 0 Register (i=0–3)	Port2: 0x00032E54, Port3: 0x00036E54, Port0: 0x00072E54, Port1: 0x00076E54	Table 562, p. 914
PHY Test PRBS Error Counter 0 Register (i=0–3)	Port2: 0x00032E7C, Port3: 0x00036E7C, Port0: 0x00072E7C, Port1: 0x00076E7C	Table 563, p. 915
PHY Test PRBS Error Counter 1 Register (i=0–3)	Port2: 0x00032E80, Port3: 0x00036E80, Port0: 0x00072E80, Port1: 0x00076E80	Table 564, p. 915
PHY Test OOB 0 Register (i=0–3)	Port2: 0x00032E84, Port3: 0x00036E84, Port0: 0x00072E84, Port1: 0x00076E84	Table 565, p. 916
Digital Loopback Enable Register (i=0–3)	Port2: 0x00032E8C, Port3: 0x00036E8C, Port0: 0x00072E8C, Port1: 0x00076E8C	Table 566, p. 917
PHY Isolation Mode Control Register (i=0–3)	Port2: 0x00032E98, Port3: 0x00036E98, Port0: 0x00072E98, Port1: 0x00076E98	Table 567, p. 917

Copyright © 2014 Marvell

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Reference Clock Select Register (i=0–3)	Port2: 0x00032F18, Port3: 0x00036F18, Port0: 0x00072F18, Port1: 0x00076F18	Table 568, p. 918
COMPHY Control Register (i=0–3)	Port2: 0x00032F20, Port3: 0x00036F20, Port0: 0x00072F20, Port1: 0x00076F20	Table 569, p. 918
Address Decoder Registers		
Base Address<n> Register (i=0–3, n=0–5)	Port2BA0: 0x00032200, Port2BA1: 0x00032208...Port1BA5 : 0x00076228	Table 570, p. 919
Size (S)<n> Register (i=0–3, n=0–5)	Port2SR0: 0x00032204, Port2SR1: 0x0003220C...Port1SR5: 0x0007622C	Table 571, p. 920
High Address Remap (HA)1<n> Register (i=0–3, n=0–3)	Port2HARR0: 0x00032280, Port2HARR1: 0x00032284...Port1HARR3: 0x0007628C	Table 572, p. 920
Base Address Enable (BARE) Register (i=0–3)	Port2: 0x00032290, Port3: 0x00036290, Port0: 0x00072290, Port1: 0x00076290	Table 573, p. 920
Ethernet Port Access Protect (EPAP) Register (i=0–3)	Port2: 0x00032294, Port3: 0x00036294, Port0: 0x00072294, Port1: 0x00076294	Table 574, p. 921
Global Miscellaneous Registers		
PHY Address Register (i=0–3)	Port2: 0x00032000, Port3: 0x00036000, Port0: 0x00072000, Port1: 0x00076000	Table 575, p. 922
SMI Register (i=0–3)	Port2: 0x00032004, Port3: 0x00036004, Port0: 0x00072004, Port1: 0x00076004	Table 576, p. 922
Ethernet Unit Default Address (EUDA) Register (i=0–3)	Port2: 0x00032008, Port3: 0x00036008, Port0: 0x00072008, Port1: 0x00076008	Table 577, p. 923
Ethernet Unit Default ID (EUDID) Register (i=0–3)	Port2: 0x0003200C, Port3: 0x0003600C, Port0: 0x0007200C, Port1: 0x0007600C	Table 578, p. 923
Ethernet Unit SMI Speed Register (i=0–3)	Port2: 0x00032014, Port3: 0x00036014, Port0: 0x00072014, Port1: 0x00076014	Table 579, p. 924

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Ethernet Unit Interrupt Cause (EUIC) Register (i=0–3)	Port2: 0x00032080, Port3: 0x00036080, Port0: 0x00072080, Port1: 0x00076080	Table 580, p. 924
Ethernet Unit Interrupt Mask (EUIM) Register (i=0–3)	Port2: 0x00032084, Port3: 0x00036084, Port0: 0x00072084, Port1: 0x00076084	Table 581, p. 926
Ethernet Unit Error Address (EUEA) Register (i=0–3)	Port2: 0x00032094, Port3: 0x00036094, Port0: 0x00072094, Port1: 0x00076094	Table 582, p. 927
Ethernet Unit Internal Address Error (EUIAE) Register (i=0–3)	Port2: 0x00032098, Port3: 0x00036098, Port0: 0x00072098, Port1: 0x00076098	Table 583, p. 927
Ethernet Unit Control (EUC) Register (i=0–3)	Port2: 0x000320B0, Port3: 0x000360B0, Port0: 0x000720B0, Port1: 0x000760B0	Table 584, p. 927
Miscellaneous Registers		
SDMA Configuration (SDC) Register (i=0–3)	Port2: 0x0003241C, Port3: 0x0003641C, Port0: 0x0007241C, Port1: 0x0007641C	Table 585, p. 928
Networking Controller Miscellaneous Registers		
Port Acceleration Mode (PACC) Register (i=0–3)	Port2: 0x00032500, Port3: 0x00036500, Port0: 0x00072500, Port1: 0x00076500	Table 586, p. 930
Port BM Address (PBMADDR) Register (i=0–3)	Port2: 0x00032504, Port3: 0x00036504, Port0: 0x00072504, Port1: 0x00076504	Table 587, p. 930
Port Version (PVersion) Register (i=0–3)	Port2: 0x000325BC, Port3: 0x000365BC, Port0: 0x000725BC, Port1: 0x000765BC	Table 588, p. 930
RX DMA Hardware Parser Registers		
VLAN EtherType (EVLANE) Register (i=0–3)	Port2: 0x00032410, Port3: 0x00036410, Port0: 0x00072410, Port1: 0x00076410	Table 589, p. 931
MAC Address Low (MACAL) Register (i=0–3)	Port2: 0x00032414, Port3: 0x00036414, Port0: 0x00072414, Port1: 0x00076414	Table 590, p. 931

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
MAC Address High (MACAH) Register (i=0–3)	Port2: 0x00032418, Port3: 0x00036418, Port0: 0x00072418, Port1: 0x00076418	Table 591, p. 932
IP Differentiated Services CodePoint 0 to Priority (DSCP0) Register (i=0–3)	Port2: 0x00032420, Port3: 0x00036420, Port0: 0x00072420, Port1: 0x00076420	Table 592, p. 932
IP Differentiated Services CodePoint 1 to Priority (DSCP1) Register (i=0–3)	Port2: 0x00032424, Port3: 0x00036424, Port0: 0x00072424, Port1: 0x00076424	Table 593, p. 932
IP Differentiated Services CodePoint 2 to Priority (DSCP2) Register (i=0–3)	Port2: 0x00032428, Port3: 0x00036428, Port0: 0x00072428, Port1: 0x00076428	Table 594, p. 933
IP Differentiated Services CodePoint 3 to Priority (DSCP3) Register (i=0–3)	Port2: 0x0003242C, Port3: 0x0003642C, Port0: 0x0007242C, Port1: 0x0007642C	Table 595, p. 934
IP Differentiated Services CodePoint 4 to Priority (DSCP4) Register (i=0–3)	Port2: 0x00032430, Port3: 0x00036430, Port0: 0x00072430, Port1: 0x00076430	Table 596, p. 934
IP Differentiated Services CodePoint 5 to Priority (DSCP5) Register (i=0–3)	Port2: 0x00032434, Port3: 0x00036434, Port0: 0x00072434, Port1: 0x00076434	Table 597, p. 935
IP Differentiated Services CodePoint 6 to Priority (DSCP6) Register (i=0–3)	Port2: 0x00032438, Port3: 0x00036438, Port0: 0x00072438, Port1: 0x00076438	Table 598, p. 935
VLAN Priority Tag to Priority (VPT2P) Register (i=0–3)	Port2: 0x00032440, Port3: 0x00036440, Port0: 0x00072440, Port1: 0x00076440	Table 599, p. 936
Ethernet Type Priority Register (i=0–3)	Port2: 0x000324BC, Port3: 0x000364BC, Port0: 0x000724BC, Port1: 0x000764BC	Table 600, p. 936
Destination Address Filter Special Multicast Table (DFSMT)<n> Register (i=0–3, n=0–63)	Port2Register0: 0x00033400, Port2Register1: 0x00033404...Port 1Register63: 0x000774FC	Table 601, p. 937
Destination Address Filter Other Multicast Table (DFOMT)<n> Register (i=0–3, n=0–63)	Port2Register0: 0x00033500, Port2Register1: 0x00033504...Port 1Register63: 0x000775FC	Table 602, p. 939
Destination Address Filter Unicast Table (DFUT)<n> Register (i=0–3, n=0–3)	Port2Register0: 0x00033600, Port2Register1: 0x00033604...Port 1Register3: 0x0007760C	Table 603, p. 940
RX DMA Miscellaneous Registers		

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Register (i=0–3)	Port2: 0x00031D00, Port3: 0x00035D00, Port0: 0x00071D00, Port1: 0x00075D00	Table 604, p. 942
Port Rx Minimal Frame Size (PxMFS) Register (i=0–3)	Port2: 0x0003247C, Port3: 0x0003647C, Port0: 0x0007247C, Port1: 0x0007647C	Table 605, p. 942
Port Rx Discard Frame Counter (PxDFC) Register (i=0–3)	Port2: 0x00032484, Port3: 0x00036484, Port0: 0x00072484, Port1: 0x00076484	Table 606, p. 942
Port Overrun Frame Counter (PxOFC) Register (i=0–3)	Port2: 0x00032488, Port3: 0x00036488, Port0: 0x00072488, Port1: 0x00076488	Table 607, p. 943
Receive Queue Command (RQC) Register (i=0–3)	Port2: 0x00032680, Port3: 0x00036680, Port0: 0x00072680, Port1: 0x00076680	Table 608, p. 943
RX DMA Networking Controller Miscellaneous Registers		
Port RX Queues Configuration (PRXC)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x00031400, Port2RX Queue1: 0x00031404...Port1RX Queue7: 0x0007541C	Table 609, p. 944
Port RX Queues Snoop (PRXSNP)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x00031420, Port2RX Queue1: 0x00031424...Port1RX Queue7: 0x0007543C	Table 610, p. 946
Port RX Prefetch 0_1 (PRXF01)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x00031440, Port2RX Queue1: 0x00031444...Port1RX Queue7: 0x0007545C	Table 611, p. 947
Port RX Queues Descriptors Queue Address (PRXDQA) <q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x00031480, Port2RX Queue1: 0x00031484...Port1RX Queue7: 0x0007549C	Table 612, p. 948
Port RX Queues Descriptors Queue Size (PRXDQS)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x000314A0, Port2RX Queue1: 0x000314A4...Port1RX Queue7: 0x000754BC	Table 613, p. 948
Port RX Queues Descriptors Queue Threshold (PRXDQTH)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x000314C0, Port2RX Queue1: 0x000314C4...Port1RX Queue7: 0x000754DC	Table 614, p. 949
Port RX Queues Status (PRXS)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x000314E0, Port2RX Queue1: 0x000314E4...Port1RX Queue7: 0x000754FC	Table 615, p. 950

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Port RX Queues Status Update (PRXSU)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x00031500, Port2RX Queue1: 0x00031504...Port1RX Queue7: 0x0007551C	Table 616, p. 950
Port RX Descriptor Index (PRXDI)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x00031520, Port2RX Queue1: 0x00031524...Port1RX Queue7: 0x0007553C	Table 617, p. 951
Port Pool0 Buffer Size (PPL0BSZ) Register (i=0–3)	Port2: 0x00031700, Port3: 0x00035700, Port0: 0x00071700, Port1: 0x00075700	Table 618, p. 952
Port Pool1 Buffer Size (PPL1BSZ) Register (i=0–3)	Port2: 0x00031704, Port3: 0x00035704, Port0: 0x00071704, Port1: 0x00075704	Table 619, p. 952
Port Pool2 Buffer Size (PPL2BSZ) Register (i=0–3)	Port2: 0x00031708, Port3: 0x00035708, Port0: 0x00071708, Port1: 0x00075708	Table 620, p. 952
Port Pool3 Buffer Size (PPL3BSZ) Register (i=0–3)	Port2: 0x0003170C, Port3: 0x0003570C, Port0: 0x0007170C, Port1: 0x0007570C	Table 621, p. 953
Reserved_1710 Register (i=0–3)	Port2: 0x00031710, Port3: 0x00035710, Port0: 0x00071710, Port1: 0x00075710	Table 622, p. 953
Port RX Initialization (PRXINIT) Register (i=0–3)	Port2: 0x00031CC0, Port3: 0x00035CC0, Port0: 0x00071CC0, Port1: 0x00075CC0	Table 623, p. 954
RX DMA Wake on LAN Registers		
Arp IP0 Address (AIP0ADR) Register (i=0–3)	Port2: 0x00032498, Port3: 0x00036498, Port0: 0x00072498, Port1: 0x00076498	Table 624, p. 954
Arp IP1 Address (AIP1ADR) Register (i=0–3)	Port2: 0x0003249C, Port3: 0x0003649C, Port0: 0x0007249C, Port1: 0x0007649C	Table 625, p. 954
WOL Sleep (WOLSLP) Register (i=0–3)	Port2: 0x00033690, Port3: 0x00037690, Port0: 0x00073690, Port1: 0x00077690	Table 626, p. 955
WOL Wakeup Event Enable (WOLWEE) Register (i=0–3)	Port2: 0x00033694, Port3: 0x00037694, Port0: 0x00073694, Port1: 0x00077694	Table 627, p. 955

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
WOL Sleep Interrupt Cause (WOLSIC) Register (i=0–3)	Port2: 0x00033698, Port3: 0x00037698, Port0: 0x00073698, Port1: 0x00077698	Table 628, p. 957
WOL Sleep Interrupt Mask (WOLSIM) Register (i=0–3)	Port2: 0x0003369C, Port3: 0x0003769C, Port0: 0x0007369C, Port1: 0x0007769C	Table 629, p. 959
WOL Wakeup Events Counter (WOLWEC) Register (i=0–3)	Port2: 0x000336A0, Port3: 0x000376A0, Port0: 0x000736A0, Port1: 0x000776A0	Table 630, p. 960
WOL Type1 (WOLT1) Register (i=0–3)	Port2: 0x000336A4, Port3: 0x000376A4, Port0: 0x000736A4, Port1: 0x000776A4	Table 631, p. 960
WOL Type2 (WOLT2) Register (i=0–3)	Port2: 0x000336A8, Port3: 0x000376A8, Port0: 0x000736A8, Port1: 0x000776A8	Table 632, p. 961
WOL Type3 (WOLT3) Register (i=0–3)	Port2: 0x000336AC, Port3: 0x000376AC, Port0: 0x000736AC, Port1: 0x000776AC	Table 633, p. 961
WOL Wakeup Frame Location (WOLWFL) Register (i=0–3)	Port2: 0x000336B0, Port3: 0x000376B0, Port0: 0x000736B0, Port1: 0x000776B0	Table 634, p. 962
WOL Wakeup Frame Size (WOLWFS) Register (i=0–3)	Port2: 0x000336B4, Port3: 0x000376B4, Port0: 0x000736B4, Port1: 0x000776B4	Table 635, p. 963
WOL Wakeup Frame Select (WOLWFSEL) Register (i=0–3)	Port2: 0x000336B8, Port3: 0x000376B8, Port0: 0x000736B8, Port1: 0x000776B8	Table 636, p. 963
WOL Wakeup Frame Data (WOLWFD)<n> Register (i=0–3, n=0–31)	Port2pattern_dword_number0: 0x00033700, Port2pattern_dword_number1: 0x00033704...Port1pattern_dword_number31: 0x0007777C	Table 637, p. 964
WOL Wakeup Frame Mask (WOLWFM)<n> Register (i=0–3, n=0–31)	Port2pattern_dword_no0: 0x00033780, Port2pattern_dword_no1: 0x00033784...Port1pattern_dword_no31: 0x000777FC	Table 638, p. 965
TX DMA Miscellaneous Registers		

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Transmit Queue Command (TQC) Register (i=0–3)	Port2: 0x00030048, Port3: 0x00034048, Port0: 0x00070048, Port1: 0x00074048	Table 639, p. 966
TX Bad FCS Transmitted Packets Counter (TXBADFCS) Register (i=0–3)	Port2: 0x000318C0, Port3: 0x000358C0, Port0: 0x000718C0, Port1: 0x000758C0	Table 640, p. 967
TX Dropped Packets Counter (TXDROPPED) Register (i=0–3)	Port2: 0x000318C4, Port3: 0x000358C4, Port0: 0x000718C4, Port1: 0x000758C4	Table 641, p. 967
Port Tx FIFO Threshold (PxTFTT) Register (i=0–3)	Port2: 0x00032478, Port3: 0x00036478, Port0: 0x00072478, Port1: 0x00076478	Table 642, p. 968
TX DMA Networking Controller Miscellaneous Registers		
Port TX Queues Descriptors Queue Address (PTXDQA)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031800, Port2Q1: 0x00031804...Port1Q7: 0 x0007581C	Table 643, p. 968
Port TX Queues Descriptors Queue Size (PTXDQS)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031820, Port2Q1: 0x00031824...Port1Q7: 0 x0007583C	Table 644, p. 968
Port TX Queues Status (PTXS)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031840, Port2Q1: 0x00031844...Port1Q7: 0 x0007585C	Table 645, p. 969
Port Tx Queues Status Update (PTXSU)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031860, Port2Q1: 0x00031864...Port1Q7: 0 x0007587C	Table 646, p. 970
Port TX Queues Descriptor Index (PTXDI)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031880, Port2Q1: 0x00031884...Port1Q7: 0 x0007589C	Table 647, p. 970
TX Transmitted Buffers Counter (TXTBC)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x000318A0, Port2Q1: 0x000318A4...Port1Q7: 0x000758BC	Table 648, p. 971
Port TX Initialization (PTXINIT) Register (i=0–3)	Port2: 0x000318F0, Port3: 0x000358F0, Port0: 0x000718F0, Port1: 0x000758F0	Table 649, p. 971
TX DMA Packet Modification Registers		
TX Marvell Header Reg1 (TX_MH_reg1) Register (i=0–3)	Port2: 0x00031944, Port3: 0x00035944, Port0: 0x00071944, Port1: 0x00075944	Table 650, p. 972
TX Marvell Header Reg2 (TX_MH_reg2) Register (i=0–3)	Port2: 0x00031948, Port3: 0x00035948, Port0: 0x00071948, Port1: 0x00075948	Table 651, p. 972

Copyright © 2014 Marvell

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
TX Marvell Header Reg3 (TX_MH_reg3) Register (i=0–3)	Port2: 0x0003194C, Port3: 0x0003594C, Port0: 0x0007194C, Port1: 0x0007594C	Table 652, p. 973
TX Marvell Header Reg4 (TX_MH_reg4) Register (i=0–3)	Port2: 0x00031950, Port3: 0x00035950, Port0: 0x00071950, Port1: 0x00075950	Table 653, p. 973
TX Marvell Header Reg5 (TX_MH_reg5) Register (i=0–3)	Port2: 0x00031954, Port3: 0x00035954, Port0: 0x00071954, Port1: 0x00075954	Table 654, p. 974
TX Marvell Header Reg6 (TX_MH_reg6) Register (i=0–3)	Port2: 0x00031958, Port3: 0x00035958, Port0: 0x00071958, Port1: 0x00075958	Table 655, p. 974
TX Marvell Header Reg7 (TX_MH_reg7) Register (i=0–3)	Port2: 0x0003195C, Port3: 0x0003595C, Port0: 0x0007195C, Port1: 0x0007595C	Table 656, p. 975
TX Marvell Header Reg8 (TX_MH_reg8) Register (i=0–3)	Port2: 0x00031960, Port3: 0x00035960, Port0: 0x00071960, Port1: 0x00075960	Table 657, p. 975
TX Marvell Header Reg9 (TX_MH_reg9) Register (i=0–3)	Port2: 0x00031964, Port3: 0x00035964, Port0: 0x00071964, Port1: 0x00075964	Table 658, p. 976
TX Marvell Header Reg10 (TX_MH_reg10) Register (i=0–3)	Port2: 0x00031968, Port3: 0x00035968, Port0: 0x00071968, Port1: 0x00075968	Table 659, p. 976
TX Marvell Header Reg11 (TX_MH_reg11) Register (i=0–3)	Port2: 0x0003196C, Port3: 0x0003596C, Port0: 0x0007196C, Port1: 0x0007596C	Table 660, p. 977
TX Marvell Header Reg12 (TX_MH_reg12) Register (i=0–3)	Port2: 0x00031970, Port3: 0x00035970, Port0: 0x00071970, Port1: 0x00075970	Table 661, p. 977
TX Marvell Header Reg13 (TX_MH_reg13) Register (i=0–3)	Port2: 0x00031974, Port3: 0x00035974, Port0: 0x00071974, Port1: 0x00075974	Table 662, p. 978
TX Marvell Header Reg14 (TX_MH_reg14) Register (i=0–3)	Port2: 0x00031978, Port3: 0x00035978, Port0: 0x00071978, Port1: 0x00075978	Table 663, p. 978

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
TX Marvell Header Reg15 (TX_MH_reg15) Register (i=0–3)	Port2: 0x0003197C, Port3: 0x0003597C, Port0: 0x0007197C, Port1: 0x0007597C	Table 664, p. 979
TX MTU (TX_MTU) Register (i=0–3)	Port2: 0x00031988, Port3: 0x00035988, Port0: 0x00071988, Port1: 0x00075988	Table 665, p. 979
TX DMA Queues Arbiter Registers		
Transmit Queue Fixed Priority Configuration - Arbiter ver1 (TQFPC_AV1) Register (i=0–3)	Port2: 0x000300DC, Port3: 0x000340DC, Port0: 0x000700DC, Port1: 0x000740DC	Table 666, p. 980
Port Transmit Token-Bucket Rate Configuration - Arbiter ver1 (PTTBRC_AV1) Register (i=0–3)	Port2: 0x000300E0, Port3: 0x000340E0, Port0: 0x000700E0, Port1: 0x000740E0	Table 667, p. 980
Maximum Transmit Unit - Arbiter ver1 (MTU_AV1) Register (i=0–3)	Port2: 0x000300E8, Port3: 0x000340E8, Port0: 0x000700E8, Port1: 0x000740E8	Table 668, p. 981
Port Maximum Token Bucket Size - Arbiter ver1 (PMTBS_AV1) Register (i=0–3)	Port2: 0x000300EC, Port3: 0x000340EC, Port0: 0x000700EC, Port1: 0x000740EC	Table 669, p. 981
Transmit Queue Token-Bucket Counter - Arbiter ver1 (TQxTBC_AV1)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00030300, Port2Q1: 0x00030310...Port1Q7: 0 x00074370	Table 670, p. 982
Transmit Queue Token Bucket Configuration - Arbiter ver1 (TQxTBCFG_AV1)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00030304, Port2Q1: 0x00030314...Port1Q7: 0 x00074374	Table 671, p. 982
Transmit Queue Arbiter Configuration - Arbiter ver1 (TQxAC_AV1)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00030308, Port2Q1: 0x00030318...Port1Q7: 0 x00074378	Table 672, p. 983
Port Transmit Token-Bucket Counter - Arbiter ver1 (PTTBC_AV1) Register (i=0–3)	Port2: 0x00030380, Port3: 0x00034380, Port0: 0x00070380, Port1: 0x00074380	Table 673, p. 983
Transmit Queue Command1 - Arbiter ver3 (TQC1_AV3) Register (i=0–3)	Port2: 0x00031A00, Port3: 0x00035A00, Port0: 0x00071A00, Port1: 0x00075A00	Table 674, p. 984
Transmit Queue Fixed Priority Configuration - Arbiter ver3 (TQFPC_AV3) Register (i=0–3)	Port2: 0x00031A04, Port3: 0x00035A04, Port0: 0x00071A04, Port1: 0x00075A04	Table 675, p. 985
Basic Refill No of Clocks - Arbiter ver3 (BRC_AV3) Register (i=0–3)	Port2: 0x00031A08, Port3: 0x00035A08, Port0: 0x00071A08, Port1: 0x00075A08	Table 676, p. 985

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Maximum Transmit Unit - Arbiter ver3 (MTU_AV3) Register (i=0–3)	Port2: 0x00031A0C, Port3: 0x00035A0C, Port0: 0x00071A0C, Port1: 0x00075A0C	Table 677, p. 986
Port Bucket Refill - Arbiter ver3 (PRefill_AV3) Register (i=0–3)	Port2: 0x00031A10, Port3: 0x00035A10, Port0: 0x00071A10, Port1: 0x00075A10	Table 678, p. 986
Port Maximum Token Bucket Size - Arbiter ver3 (PMTBS_AV3) Register (i=0–3)	Port2: 0x00031A14, Port3: 0x00035A14, Port0: 0x00071A14, Port1: 0x00075A14	Table 679, p. 987
Port Transmit Token-Bucket Counter - Arbiter ver3 (PTTBC_AV3) Register (i=0–3)	Port2: 0x00031A18, Port3: 0x00035A18, Port0: 0x00071A18, Port1: 0x00075A18	Table 680, p. 988
Queue Bucket Refill - Arbiter ver3 (QRefill_AV3)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031A20, Port2Q1: 0x00031A24...Port1Q7: 0x00075A3C	Table 681, p. 988
Queue Maximum Token Bucket Size - Arbiter ver3 (QMTBS_AV3)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031A40, Port2Q1: 0x00031A44...Port1Q7: 0x00075A5C	Table 682, p. 989
Queue Transmit Token-Bucket Counter - Arbiter ver3 (QxTTBC_AV3)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031A60, Port2Q1: 0x00031A64...Port1Q7: 0x00075A7C	Table 683, p. 990
Transmit Queue Arbiter Configuration - Arbiter ver3 (TQxAC_AV3)<q> Register (i=0–3, q=0–7)	Port2Q0: 0x00031A80, Port2Q1: 0x00031A84...Port1Q7: 0x00075A9C	Table 684, p. 990
Transmission Queue IPG - Arbiter ver3 (TQxIPG_AV3)<q> Register (i=0–3, q=2–3)	Port2Q2: 0x00031AA8, Port2Q3: 0x00031AAC, Port3Q2: 0x00035AA8, Port3Q3: 0x00035AAC, Port0Q2: 0x00071AA8, Port0Q3: 0x00071AAC, Port1Q2: 0x00075AA8, Port1Q3: 0x00075AAC	Table 685, p. 991
High Token in Low Packet - Arbiter ver3 (HITKNinLOPKT_AV3) Register (i=0–3)	Port2: 0x00031AB0, Port3: 0x00035AB0, Port0: 0x00071AB0, Port1: 0x00075AB0	Table 686, p. 991
High Token in Asynchronous Packet - Arbiter ver3 (HITKNinASYNCPKT_AV3) Register (i=0–3)	Port2: 0x00031AB4, Port3: 0x00035AB4, Port0: 0x00071AB4, Port1: 0x00075AB4	Table 687, p. 992
Low Token in Asynchronous Packet - Arbiter ver3 (LOTKNinASYNCPKT_AV3) Register (i=0–3)	Port2: 0x00031AB8, Port3: 0x00035AB8, Port0: 0x00071AB8, Port1: 0x00075AB8	Table 688, p. 993

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Transmission Speed - Arbiter ver3 (TS_AV3) Register (i=0–3)	Port2: 0x00031ABC, Port3: 0x00035ABC, Port0: 0x00071ABC, Port1: 0x00075ABC	Table 689, p. 993
RX_TX DMA Registers		
Port Configuration (PxC) Register (i=0–3)	Port2: 0x00032400, Port3: 0x00036400, Port0: 0x00072400, Port1: 0x00076400	Table 690, p. 994
Port Configuration Extend (PxCX) Register (i=0–3)	Port2: 0x00032404, Port3: 0x00036404, Port0: 0x00072404, Port1: 0x00076404	Table 691, p. 996
Marvell Header Register (i=0–3)	Port2: 0x00032454, Port3: 0x00036454, Port0: 0x00072454, Port1: 0x00076454	Table 692, p. 997
Serial Registers		
Port Serial Control0 (PSC0) Register (i=0–3)	Port2: 0x0003243C, Port3: 0x0003643C, Port0: 0x0007243C, Port1: 0x0007643C	Table 693, p. 1000
Ethernet Port Status 0 (PS0) Register (i=0–3)	Port2: 0x00032444, Port3: 0x00036444, Port0: 0x00072444, Port1: 0x00076444	Table 694, p. 1001
Serdes Configuration (SERDESCFG) Register (i=0–3)	Port2: 0x000324A0, Port3: 0x000364A0, Port0: 0x000724A0, Port1: 0x000764A0	Table 695, p. 1002
Serdes Status (SERDESSTS) Register (i=0–3)	Port2: 0x000324A4, Port3: 0x000364A4, Port0: 0x000724A4, Port1: 0x000764A4	Table 696, p. 1002
One mS Clock Divider Register (i=0–3)	Port2: 0x000324F4, Port3: 0x000364F4, Port0: 0x000724F4, Port1: 0x000764F4	Table 697, p. 1002
Periodic Flow Control Clock Divider Register (i=0–3)	Port2: 0x000324F8, Port3: 0x000364F8, Port0: 0x000724F8, Port1: 0x000764F8	Table 698, p. 1003
Gigabit Ethernet MAC Serial Parameters Configuration Registers		
Port Serial Parameters Configuration Register (i=0–3)	Port2: 0x00032C14, Port3: 0x00036C14, Port0: 0x00072C14, Port1: 0x00076C14	Table 699, p. 1003

Copyright © 2014 Marvell

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Port Serial Parameters 1 Configuration Register (i=0–3)	Port2: 0x00032C94, Port3: 0x00036C94, Port0: 0x00072C94, Port1: 0x00076C94	Table 700, p. 1004
Gigabit Ethernet MAC Auto-Negotiation Configuration Registers		
Port Auto-Negotiation Configuration Register (i=0–3)	Port2: 0x00032C0C, Port3: 0x00036C0C, Port0: 0x00072C0C, Port1: 0x00076C0C	Table 701, p. 1004
Gigabit Ethernet MAC Control Registers		
Port MAC Control Register0 (i=0–3)	Port2: 0x00032C00, Port3: 0x00036C00, Port0: 0x00072C00, Port1: 0x00076C00	Table 702, p. 1007
Port MAC Control Register1 (i=0–3)	Port2: 0x00032C04, Port3: 0x00036C04, Port0: 0x00072C04, Port1: 0x00076C04	Table 703, p. 1008
Port MAC Control Register2 (i=0–3)	Port2: 0x00032C08, Port3: 0x00036C08, Port0: 0x00072C08, Port1: 0x00076C08	Table 704, p. 1009
Port MAC Control Register3 (i=0–3)	Port2: 0x00032C48, Port3: 0x00036C48, Port0: 0x00072C48, Port1: 0x00076C48	Table 705, p. 1011
CCFC Port Speed Timerp Register (i=0–3, p=0–7)	Port2Port Speed Index0: 0x00032C58, Port2Port Speed Index1: 0x00032C5C...Port1Port Speed Index7: 0x00076C74	Table 706, p. 1011
Port MAC Control Register4 (i=0–3)	Port2: 0x00032C90, Port3: 0x00036C90, Port0: 0x00072C90, Port1: 0x00076C90	Table 707, p. 1012
Gigabit Ethernet MAC Interrupt Registers		
Port Interrupt Cause Register (i=0–3)	Port2: 0x00032C20, Port3: 0x00036C20, Port0: 0x00072C20, Port1: 0x00076C20	Table 708, p. 1012
Port Interrupt Mask Register (i=0–3)	Port2: 0x00032C24, Port3: 0x00036C24, Port0: 0x00072C24, Port1: 0x00076C24	Table 709, p. 1013
Gigabit Ethernet MAC Low Power Idle Registers		
LPI Control 0 Register (i=0–3)	Port2: 0x00032CC0, Port3: 0x00036CC0, Port0: 0x00072CC0, Port1: 0x00076CC0	Table 710, p. 1014

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
LPI Control 1 Register (i=0–3)	Port2: 0x00032CC4, Port3: 0x00036CC4, Port0: 0x00072CC4, Port1: 0x00076CC4	Table 711, p. 1014
LPI Control 2 Register (i=0–3)	Port2: 0x00032CC8, Port3: 0x00036CC8, Port0: 0x00072CC8, Port1: 0x00076CC8	Table 712, p. 1015
LPI Status Register (i=0–3)	Port2: 0x00032CCC, Port3: 0x00036CCC, Port0: 0x00072CCC, Port1: 0x00076CCC	Table 713, p. 1015
LPI Counter Register (i=0–3)	Port2: 0x00032CD0, Port3: 0x00036CD0, Port0: 0x00072CD0, Port1: 0x00076CD0	Table 714, p. 1016
Gigabit Ethernet MAC PRBS Check Status Registers		
Port PRBS Status Register (i=0–3)	Port2: 0x00032C38, Port3: 0x00036C38, Port0: 0x00072C38, Port1: 0x00076C38	Table 715, p. 1016
Port PRBS Error Counter Register (i=0–3)	Port2: 0x00032C3C, Port3: 0x00036C3C, Port0: 0x00072C3C, Port1: 0x00076C3C	Table 716, p. 1016
Gigabit Ethernet MAC Status Registers		
Port Status Register0 (i=0–3)	Port2: 0x00032C10, Port3: 0x00036C10, Port0: 0x00072C10, Port1: 0x00076C10	Table 717, p. 1017
Networking Controller Interrupt Registers		
Port CPU0 to Queue (PCP02Q) Register (i=0–3)	Port2: 0x00032540, Port3: 0x00036540, Port0: 0x00072540, Port1: 0x00076540	Table 718, p. 1020
Port CPU1 to Queue (PCP12Q) Register (i=0–3)	Port2: 0x00032544, Port3: 0x00036544, Port0: 0x00072544, Port1: 0x00076544	Table 719, p. 1021
Port CPU2 to Queue (PCP22Q) Register (i=0–3)	Port2: 0x00032548, Port3: 0x00036548, Port0: 0x00072548, Port1: 0x00076548	Table 720, p. 1022
Port CPU3 to Queue (PCP32Q) Register (i=0–3)	Port2: 0x0003254C, Port3: 0x0003654C, Port0: 0x0007254C, Port1: 0x0007654C	Table 721, p. 1022

Table 557: Summary Map Table for the Ethernet Networking Controller and MAC Registers

Register Name	Offset	Table and Page
Port RX Interrupt Threshold (PRXITTH)<q> Register (i=0–3, q=0–7)	Port2RX Queue0: 0x00032580, Port2RX Queue1: 0x00032584...Port1RX Queue7: 0x0007659C	Table 722, p. 1023
Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3)	Port2: 0x000325A0, Port3: 0x000365A0, Port0: 0x000725A0, Port1: 0x000765A0	Table 723, p. 1024
Port RX_TX Threshold Interrupt Mask (PRXTXThIM) Register (i=0–3)	Port2: 0x000325A4, Port3: 0x000365A4, Port0: 0x000725A4, Port1: 0x000765A4	Table 724, p. 1026
Port RX_TX Interrupt Cause (PRXTXIC) Register (i=0–3)	Port2: 0x000325A8, Port3: 0x000365A8, Port0: 0x000725A8, Port1: 0x000765A8	Table 725, p. 1026
Port RX_TX Interrupt Mask (PRXTXIM) Register (i=0–3)	Port2: 0x000325AC, Port3: 0x000365AC, Port0: 0x000725AC, Port1: 0x000765AC	Table 726, p. 1028
Port Misc Interrupt Cause (PMiscIC) Register (i=0–3)	Port2: 0x000325B0, Port3: 0x000365B0, Port0: 0x000725B0, Port1: 0x000765B0	Table 727, p. 1028
Port Misc Interrupt Mask (PMiscIM) Register (i=0–3)	Port2: 0x000325B4, Port3: 0x000365B4, Port0: 0x000725B4, Port1: 0x000765B4	Table 728, p. 1031
Port Interrupt Enable (PIntEnb) Register (i=0–3)	Port2: 0x000325B8, Port3: 0x000365B8, Port0: 0x000725B8, Port1: 0x000765B8	Table 729, p. 1031
Miscellaneous Interrupt Registers		
Port Internal Address Error (EUIAE) Register (i=0–3)	Port2: 0x00032494, Port3: 0x00036494, Port0: 0x00072494, Port1: 0x00076494	Table 730, p. 1032

A.6.1 COM PHY

Table 558: Power and PLL Control Register (i=0–3)

Offset: Port2: 0x00032E04, Port3: 0x00036E04, Port0: 0x00072E04, Port1: 0x00076E04

Offset Formula: 0x00072E04+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	pu_ivref	RW 0x1	Power Up Reference. Analog current and voltage reference circuits power control. 0 = Power down 1 = Power up 0 = Power down 1 = Power up
14	PU_PLL	RW 0x1	Power Up PLL. This bit only works in the PHY Isolate Mode. 0 = Power down 1 = Power up 0 = Power down 1 = Power up
13	PU_RX	RW 0x1	Power Up Receiver. This bit only works in the PHY Isolate Mode. 0 = Power down 1 = Power up 0 = Power down 1 = Power up
12	PU_TX	RW 0x1	Power Up Transmitter. This bit only works in the PHY Isolate Mode. 0 = Power down 1 = Power up 0 = Power down 1 = Power up
11:8	Reserved	RSVD 0x8	Reserved
7:5	PHY_MODE	RW 0x3	PHY_MODE These bits control the mode selection. 0x0: SATA. 0x3: PCIe. 0x4: SERDES (GbE).

Table 558: Power and PLL Control Register (i=0–3) (Continued)

Offset: Port2: 0x00032E04, Port3: 0x00036E04, Port0: 0x00072E04, Port1: 0x00076E04

Offset Formula: 0x00072E04+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
4:0	REF_FREF_SEL	RW 0x0	Reference Frequency Select. These bits indicate the reference clock speed. SATA 0: 20.0 MHz 1: 25.0 MHz SerDes (GbE) 0x0: 20.0 MHz 0x1: 25.0 MHz PCIe PHY 0x0: 100 MHz

Table 559: KVCO Calibration Control Register (i=0–3)

Offset: Port2: 0x00032E08, Port3: 0x00036E08, Port0: 0x00072E08, Port1: 0x00076E08

Offset Formula: 0x00072E08+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x43	Reserved

Table 560: Generation 1 Setting 0 Register (i=0–3)

Offset: Port2: 0x00032E34, Port3: 0x00036E34, Port0: 0x00072E34, Port1: 0x00076E34

Offset Formula: 0x00072E34+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0xC918	Reserved

Table 561: Generation 2 Setting 0 Register (i=0–3)

Offset: Port2: 0x00032E3C, Port3: 0x00036E3C, Port0: 0x00072E3C, Port1: 0x00076E3C

Offset Formula: 0x00072E3C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0xAA21	Reserved

Table 562: PHY Test Control 0 Register (i=0–3)

Offset: Port2: 0x00032E54, Port3: 0x00036E54, Port0: 0x00072E54, Port1: 0x00076E54

Offset Formula: 0x00072E54+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	PT_EN	RW 0x0	PHY Test Enable 0: PHY Test disable 1: PHY Test enable Pattern must be selected while PT_EN (R15h [15]) = 0. When set to 1, this signal clears the pattern and error counts and begins looking for pattern lock. When set to zero this stops the testing and freeze the error and pattern counts.
14	PT_RST	RW 0x0	PHY Test Reset. This is a reset signal to PHY Test. Once its set to 1, all registers in PHY Test are cleared.
13	PT_PASS	RO 0x0	PHY Test Pass 0: 1 or more errors or the pattern is not locked 1: Pattern is locked and no errors detected.
12	PT_LOCK	RO 0x0	PHY Test Pattern Lock 0: Pattern detector is not locked onto the pattern 1: Patter detector had locked onto the pattern If the pattern doesn't lock then either the signal quality is low or the pattern provided to the receiver doesn't match the programmed pattern.
11	PT_XFER_DIFF	RW 0x0	PHY Test Transmit Jitter Pattern Select Option 0: PT_DATA[7:0] selects jitter test pattern for both transmit and receive data 1: PT_DATA [95:88] selects jitter test pattern for transmit data and PT_DATA [7:0] selects pattern for receive data comparison When this signal is 1 the transmitter and receiver can be programmed to use different jitter patterns. This field only has meaning when using the jitter patterns PT_PATTERN_SEL (R15h [7:4]) = 1110b. Both fields PT_DATA [95:88] and PT_DATA [7:0] have the same encoding.
10	PT_PATSYN_EN	RW 0x0	PHY Test Pattern Sync Enable If enabled, a pattern sync signal is generated and pulsed at the pattern repeat point. This can be used as a pattern trigger for test equipment.

Copyright © 2014 Marvell

Table 562: PHY Test Control 0 Register (i=0–3) (Continued)

Offset: Port2: 0x00032E54, Port3: 0x00036E54, Port0: 0x00072E54, Port1: 0x00076E54

Offset Formula: 0x00072E54+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
9	PT_START_RD	RW 0x0	PHY Test Start Running Disparity Selects the initial running disparity for SATA test patterns. 0: Initial disparity for pattern is negative 1: Initial disparity for pattern is positive
8	PT_LONG_SHORT	RW 0x0	PHY Test Long SATA Patterns 0: Short version of SATA patterns 1: Long version of SATA patterns
7:4	PT_PATTERN_SEL	RW 0x0	PHY Test Pattern Select This selects the pattern from the table of patterns or specifies an alternate table to use for additional jitter test patterns. Patterns can only be selected when PT_EN (R15h [15]) = 0.
3:0	PT_LOCK_CNT	RW 0x0	Pattern Lock Count Program with the number of pattern signatures to be found -1. Once n + 1 of these signatures has been found PT_LOCK (R15h [12]) is asserted and the bit error test begins.

Table 563: PHY Test PRBS Error Counter 0 Register (i=0–3)

Offset: Port2: 0x00032E7C, Port3: 0x00036E7C, Port0: 0x00072E7C, Port1: 0x00076E7C

Offset Formula: 0x00072E7C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	PT_PRBS_ERR_ CNT[31:16]	RO 0x0	PHY Test Error Count [31:16] Indicates how may bit errors were encountered after obtaining pattern lock.

Table 564: PHY Test PRBS Error Counter 1 Register (i=0–3)

Offset: Port2: 0x00032E80, Port3: 0x00036E80, Port0: 0x00072E80, Port1: 0x00076E80

Offset Formula: 0x00072E80+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved

Table 564: PHY Test PRBS Error Counter 1 Register (i=0–3) (Continued)

Offset: Port2: 0x00032E80, Port3: 0x00036E80, Port0: 0x00072E80, Port1: 0x00076E80

Offset Formula: 0x00072E80+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	PT_PRBS_ERR_CNT[15:0]	RO 0x0	PHY Test Error Count [15:0] The error count indicates how may bit errors were encountered after obtaining pattern lock.

Table 565: PHY Test OOB 0 Register (i=0–3)

Offset: Port2: 0x00032E84, Port3: 0x00036E84, Port0: 0x00072E84, Port1: 0x00076E84

Offset Formula: 0x00072E84+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	PT_OOB_EN	RW 0x0	PHY Test Out Of Band (OOB) Enable This allows PHY Test to control the idle signal to generate pattern that resembles OOB. Must set PT_EN (R54h [15]) to run.
14	Reserved	RSVD 0x0	Reserved
13:12	PT_TESTMODE	RW 0x0	Test Mode. 0: Test Disable 1: Test Enable
11:10	PT_OOB_SPEED	RW 0x0	PHY Test Out Of Band (OOB) Speed 0h: Every bit of the OOB pattern is transmitted 1 time (Use when programmed to 1.5G). 1h: Every bit of the OOB pattern is transmitted 2 times (Use when programmed to 3G). 2h: Every bit of the OOB pattern is transmitted 4 times (Use when programmed to 6G). 4h: Every bit of the OOB pattern is transmitted 8 times (reserved for future speed).
9:8	PT_OOB_PAT_SEL	RW 0x0	PHY Test Out Of Band (OOB) Pattern Select Selects which pattern is transmitted during a emulated OOB burst. 00: Transmit the align primitive. 01: Transmit D24.3 pattern. 10: Reserved - do not use. 11: Reserved - do not use.

Table 565: PHY Test OOB 0 Register (i=0–3) (Continued)

Offset: Port2: 0x00032E84, Port3: 0x00036E84, Port0: 0x00072E84, Port1: 0x00076E84

Offset Formula: 0x00072E84+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
7:0	PT_OOB_IDLE_ LEN	RW 0x0	Out Of Band (OOB) Idle Length. Specifies the emulated OOB gap width. IDLE period = pt_oob_idle_len (R84h [7:0]) × 40 UI.

Table 566: Digital Loopback Enable Register (i=0–3)

Offset: Port2: 0x00032E8C, Port3: 0x00036E8C, Port0: 0x00072E8C, Port1: 0x00076E8C

Offset Formula: 0x00072E8C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:12	Reserved	RSVD 0x0	Reserved
11:10	SEL_BITS	RW 0x1	Select Data Bit Width. 00: 10-bit 01: 20-bit 10: 40-bit 11: Reserved
9	CDR_LOCK_MODE	RW 0x0	CDR Lock Mode. 0: CDR lock with clock only. 1: CDR lock with both clock and data.
8	CDR_LOCK_DET_ EN	RW 0x0	CDR Lock Detection Enable. 0: CDR lock detection is disabled. 1: CDR lock detection is enabled.
7:0	Reserved	RSVD 0x0	Reserved

Table 567: PHY Isolation Mode Control Register (i=0–3)

Offset: Port2: 0x00032E98, Port3: 0x00036E98, Port0: 0x00072E98, Port1: 0x00076E98

Offset Formula: 0x00072E98+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 567: PHY Isolation Mode Control Register (i=0–3) (Continued)

Offset: Port2: 0x00032E98, Port3: 0x00036E98, Port0: 0x00072E98, Port1: 0x00076E98

Offset Formula: 0x00072E98+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15	PHY Isolate Mode	RW 0x0	PHY Isolation Mode Enable. Enable is used to isolate the PHY from the outside logic to test the stand-alone mode. 0 = Disable. 1 = Enable. 0 = Disable 1 = Enable
14	EOM_CK_ALIGN_EN	RW 0x0	EOM Clock Align Enable. To enable EOM clock and data clock alignment circuits.
13:0	Reserved	RSVD 0x100	Reserved

Table 568: Reference Clock Select Register (i=0–3)

Offset: Port2: 0x00032F18, Port3: 0x00036F18, Port0: 0x00072F18, Port1: 0x00076F18

Offset Formula: 0x00072F18+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:11	Reserved	RSVD 0x0	Reserved
10	REFCLK_SEL	RW 0x0	0x0: RefClk0 (100MHz for PCIe). 0x1: RefClk1 (25MHz for SATA & GbE).
9:0	Reserved	RSVD 0x0	Reserved

Table 569: COMPHY Control Register (i=0–3)

Offset: Port2: 0x00032F20, Port3: 0x00036F20, Port0: 0x00072F20, Port1: 0x00076F20

Offset Formula: 0x00072F20+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x1210	Reserved
2	RX_HIZ	RW 0x1	RX High Impedance Mode. 0 = Termination: Serdes-RX-PAD are connected to termination 1 = HighZ: Serdes-RX-PAD are not connected to termination

Table 569: COMPHY Control Register (i=0–3) (Continued)

Offset: Port2: 0x00032F20, Port3: 0x00036F20, Port0: 0x00072F20, Port1: 0x00076F20

Offset Formula: 0x00072F20+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
1:0	Reserved	RSVD 0x0	Reserved

A.6.2 Address Decoder Registers

Table 570: Base Address<n> Register (i=0–3, n=0–5)

Offset: Port2BA0: 0x00032200, Port2BA1: 0x00032208...Port1BA5: 0x00076228

Offset Formula: 0x00072200+n*8 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-5) represents BA

Bit	Field	Type/InitVal	Description
31:16	Base Address	RW 0x0	Window Base address Used together with the size register to set the address window size and location within the device’s address space. An address driven by the networking controller is considered as a window hit if: (address | size) == (base | size).
15:8	Attr	RW 0x0	Specifies target specific attributes depending on the target interface. See Address Decoding chapter for full details.
7:4	Reserved	RW 0x0	Reserved
3:0	Target	RW 0x0	Specifies the target resource associated with this window. See Address Decoding chapter for full details

Table 571: Size (S)<n> Register (i=0–3, n=0–5)

Offset: Port2SR0: 0x00032204, Port2SR1: 0x0003220C...Port1SR5: 0x0007622C

Offset Formula: 0x00072204+n*0x8 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-5) represents SR

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window size Used together with the Base Address register to set the address window size and location within the range of 4 GB space. Must be programmed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB). An address driven by networking controller is considered as a window hit if: (address | size) == (base | size).
15:0	Reserved	RO 0x0	Reserved

Table 572: High Address Remap (HA)1<n> Register (i=0–3, n=0–3)

Offset: Port2HARR0: 0x00032280, Port2HARR1: 0x00032284...Port1HARR3: 0x0007628C

Offset Formula: 0x00072280+n*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-3) represents HARR

Bit	Field	Type/InitVal	Description
Remap 0 corresponds to Base Address register 0; Remap 1 to Base Address register 1; Remap 2 to Base Address register 2; and Remap 3 to Base Address register 3.			
31:0	Remap	RW 0x0	Remap address Specifies address bits[63:32] to be driven to the target interface. Relevant only for target interfaces that supports more than 4 GB of address space.

Table 573: Base Address Enable (BARE) Register (i=0–3)

Offset: Port2: 0x00032290, Port3: 0x00036290, Port0: 0x00072290, Port1: 0x00076290

Offset Formula: 0x00072290+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:6	Reserved	RO 0x0	Reserved

Table 573: Base Address Enable (BARE) Register (i=0–3) (Continued)

Offset: Port2: 0x00032290, Port3: 0x00036290, Port0: 0x00072290, Port1: 0x00076290

Offset Formula: 0x00072290+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
5:0	En	RW 0x3F	Address window enable This is one bit per window. If it is set to 0, the corresponding address window is enabled. (Bit[0] matches to Window0; bit[1] matches to Window1, etc.) 0 = Enable 1 = Disable

Table 574: Ethernet Port Access Protect (EPAP) Register (i=0–3)

Offset: Port2: 0x00032294, Port3: 0x00036294, Port0: 0x00072294, Port1: 0x00076294

Offset Formula: 0x00072294+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:12	Reserved	RO 0x0	Reserved
11:10	Win5	RW 0x3	Window5 access control (the same as Win0 access control)
9:8	Win4	RW 0x3	Window4 access control (the same as Win0 access control)
7:6	Win3	RW 0x3	Window3 access control (the same as Win0 access control)
5:4	Win2	RW 0x3	Window2 access control (the same as Win0 access control)
3:2	Win1	RW 0x3	Window1 access control (the same as Win0 access control)
1:0	Win0	RW 0x3	Window0 access control In case of access violation (for example, write data to a read only region), an interrupt is set, and the transaction is written or read from the default address, as specified in the default address register. 0 = No Access 1 = Read Only 2 = Reserved 3 = Full Access: (Read or Write)

A.6.3 Global Miscellaneous Registers

Table 575: PHY Address Register (i=0–3)

Offset: Port2: 0x00032000, Port3: 0x00036000, Port0: 0x00072000, Port1: 0x00076000

Offset Formula: 0x00072000+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:6	Reserved	RO 0x0	Reserved
5	Reserved	RW 0x0	Reserved. Must write 0.
4:0	PhyAd	RW 0x8	PHY Device Address This field defines the PHY address of the PHY connected to the port. This PHY address is used during Auto-Negotiation

Table 576: SMI Register (i=0–3)

Offset: Port2: 0x00032004, Port3: 0x00036004, Port0: 0x00072004, Port1: 0x00076004

Offset Formula: 0x00072004+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:29	Reserved	RO 0x0	Reserved
28	Busy	RO 0x0	1 = Indicates that an CPU SMI register access operation is in progress and that the CPU should not write to the SMI register during this time.
27	ReadValid	RO 0x0	1 = Indicates that the CPU SMI Read operation has completed, and that the data is valid on the <Data> field.
26	Opcode	RW 0x1	Defines the type of the CPU SMI register access: 0 = Write 1 = Read
25:21	RegAd	RW 0x0	PHY Device Register Address This field defines the PHY Device Register Address used during the CPU SMI READ or WRITE access.
20:16	PhyAd	RW 0x0	PHY Device Address This field defines the PHY address used during the CPU SMI READ or WRITE access.

Copyright © 2014 Marvell

Table 576: SMI Register (i=0–3) (Continued)

Offset: Port2: 0x00032004, Port3: 0x00036004, Port0: 0x00072004, Port1: 0x00076004

Offset Formula: 0x00072004+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	Data	RW 0x0	SMI READ operation sequence: (1) Write to SMI register where <Opcode> = 1, <PhyAd>=destnation PHY address, <RegAd>=destination register address . <Data> field is ignored. (2) Read SMI register. (3) If <ReadValid> is not set, goto step (2) (4) When <ReadValid> is set, the <Data> is the addressed PHY register content. <Data> filed remains undefined as long as <ReadValid> is 0. SMI WRITE operation (single register write): Write to SMI register where <Opcode> = 0, <PhyAd>=destnation PHY address, <RegAd>=destination register address,<Data> = data to be written to the addressed PHY register.

Table 577: Ethernet Unit Default Address (EUDA) Register (i=0–3)

Offset: Port2: 0x00032008, Port3: 0x00036008, Port0: 0x00072008, Port1: 0x00076008

Offset Formula: 0x00072008+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:3	DAR[31:3]	RW 0x0	Default Address The Default Address to which the Networking Controller directs no match, multiple address hits, and address protect violations. Occurrence of this event may be due to programming errors of the descriptor pointers or buffer pointers. **NOTE:** This field is in 8 bytes units. **NOTE:** The Default Address should by a multiple of 128 (bits [6:3] should be 0x0) **NOTE:**
2:0	Reserved	RO 0x0	Reserved

Table 578: Ethernet Unit Default ID (EUDID) Register (i=0–3)

Offset: Port2: 0x0003200C, Port3: 0x0003600C, Port0: 0x0007200C, Port1: 0x0007600C

Offset Formula: 0x0007200C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:12	Reserved	RO 0x0	Reserved

Table 578: Ethernet Unit Default ID (EUDID) Register (i=0–3) (Continued)

Offset: Port2: 0x0003200C, Port3: 0x0003600C, Port0: 0x0007200C, Port1: 0x0007600C

Offset Formula: 0x0007200C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
11:4	DATTR	RW 0xE	Specifies the Default Attribute of the target unit to which the Ethernet unit directs no match and address protect violations. Identical to Base Address register’s <Attr> field encoding.
3:0	DIDR	RW 0x0	Specifies the ID of the target unit to which the Ethernet unit directs no match and address protect violations. Identical to Base Address register’s <Target> field encoding.

Table 579: Ethernet Unit SMI Speed Register (i=0–3)

Offset: Port2: 0x00032014, Port3: 0x00036014, Port0: 0x00072014, Port1: 0x00076014

Offset Formula: 0x00072014+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:2	Reserved	RO 0x0	Reserved
1:0	MDC Frequency	RW 0x0	MDC Clock frequency Define MDC clock frequency as following: 0 = Normal mode: Clock frequency is core clock frequency divided by 128. 1 = Fast mode: Clock frequency is core clock frequency divided by 16 2 = Accelerate mode: Clock frequency is core clock frequency divided by 8 3 = Reserved

Table 580: Ethernet Unit Interrupt Cause (EUIC) Register (i=0–3)

Offset: Port2: 0x00032080, Port3: 0x00036080, Port0: 0x00072080, Port1: 0x00076080

Offset Formula: 0x00072080+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.			
31:14	Reserved	RO 0x0	Reserved
13	Reserved	RW0C 0x0	Reserved
12	MDPErr	RW0C 0x0	Port MIB SRAM parity error. 0 = No Error: No MIB counters SRAM error was detected. 1 = Error: MIB counters SRAM error was detected.

Table 580: Ethernet Unit Interrupt Cause (EUIC) Register (i=0–3) (Continued)

Offset: Port2: 0x00032080, Port3: 0x00036080, Port0: 0x00072080, Port1: 0x00076080

Offset Formula: 0x00072080+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
11:10	Reserved	RO 0x0	Reserved.
9	PDPErr	RW0C 0x0	Port DMA internal data path (SRAMs) parity error. 0 = No Error: No DMA internal data path error was detected. 1 = Error: DMA Internal data path error was detected.
8	Reserved	RO 0x0	Reserved
7	InternalAddrError	RW0C 0x0	Internal Address Error is set when an access to an illegal offset of the internal registers is identified. When set, the Internal Address Error register locks the address that caused the error.
6	Reserved	RW0C 0x0	Reserved
5	CountWA	RW0C 0x0	Counters Wrap Around Indication MIB Counter Wrap Around Interrupt is set if one of the MIB counters wrapped around (passed 32 bits).
4	SMIdone	RW0C 0x0	SMI Command Done Indicates the SMI completed a MII management command (either read or write) initiated by the CPU writing to the SMI register.
4	Reserved	RSVD 0x0	Reserved
3	AddressNoMatch	RW0C 0x0	This bit is set if an Ethernet DMA address does not match any of the Ethernet address decode windows.
2	AddressViolation	RW0C 0x0	This bit is set if an Ethernet DMA violates a window access protection.

Table 580: Ethernet Unit Interrupt Cause (EUIC) Register (i=0–3) (Continued)

Offset: Port2: 0x00032080, Port3: 0x00036080, Port0: 0x00072080, Port1: 0x00076080

Offset Formula: 0x00072080+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
1	Parity	RW0C 0x0	Parity Error This bit is set if parity error is detected during a DMA read operation. Effect on Tx DMA operation: - If a parity error occurs upon descriptor fetch, the Tx DMA asserts the Tx Error interruption. - If the parity error occurred on the first descriptor fetch, the DMA stops and disables the queue. - If parity error is detected on a packet's data, the Tx DMA continues with the transmission but does not ask to generate CRC at the end of the packet. Effect on Rx DMA operation: - If a parity error occurs upon descriptor fetch, the Rx_DMA asserts the Rx Error interrupt. - If the parity error occurred on the first descriptor fetch, the DMA stops and disables the queue.
0	EtherIntSum	RO 0x0	Ethernet Unit Interrupt Summary This bit is a logical OR of the unmasked bits[13:1] in the register.

Table 581: Ethernet Unit Interrupt Mask (EUIM) Register (i=0–3)

Offset: Port2: 0x00032084, Port3: 0x00036084, Port0: 0x00072084, Port1: 0x00076084

Offset Formula: 0x00072084+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:14	Reserved	RO 0x0	Reserved
13:1	Various	RW 0x0	Mask bits for Unit Interrupt Cause register 0 = Mask 1 = Do not mask
0	Reserved	RO 0x0	Reserved

Table 582: Ethernet Unit Error Address (EUEA) Register (i=0–3)

Offset: Port2: 0x00032094, Port3: 0x00036094, Port0: 0x00072094, Port1: 0x00076094

Offset Formula: 0x00072094+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	Error Address	RO 0x0	Locks the address, if there is an address violation of the DMA such as: Multiple Address window hit, No Hit, or Access Violations. The Address is locked until the register is read. (Used for software debug after address violation interrupt is raised.) This field is read only

Table 583: Ethernet Unit Internal Address Error (EUIAE) Register (i=0–3)

Offset: Port2: 0x00032098, Port3: 0x00036098, Port0: 0x00072098, Port1: 0x00076098

Offset Formula: 0x00072098+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:2	Internal Address	RO 0x0	If there is an address violation of an unmapped access to the Networking Controller top registers, locks the relevant internal address bits (bits 15:2). The Address is locked until the register is read. **NOTE:** This field is used for software debugging after an address violation interrupt is raised. **NOTE:**
1:0	Reserved	RO 0x0	Reserved

Table 584: Ethernet Unit Control (EUC) Register (i=0–3)

Offset: Port2: 0x000320B0, Port3: 0x000360B0, Port0: 0x000720B0, Port1: 0x000760B0

Offset Formula: 0x000720B0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:26	Reserved	RO 0x0	Reserved
25	RamsInitializationCompleted	RO 0x0	RAMs initialization completed This bit, when set to 0x1, indicates that the initialization of the port various RAMs was completed. The RAMs initialization is done following reset deassertion. **NOTE:** The CPU should delay configuration of the various port register, till the completion of RAMs initialization **NOTE:**

Copyright © 2014 Marvell

Table 584: Ethernet Unit Control (EUC) Register (i=0–3) (Continued)

Offset: Port2: 0x000320B0, Port3: 0x000360B0, Port0: 0x000720B0, Port1: 0x000760B0

Offset Formula: 0x000720B0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
24	Port Reset	RW 0x0	Port reset **NOTE:** Following port reset all the port registers should be configured again 0 = Inactive: No reset to the port 1 = Active: Port in software reset
23:4	Reserved	RO 0x0	Reserved
3	Reserved	SC 0x0	Reserved
2	Reserved	RO 0x0	Reserved
1	Polling	RW 0x0	Polling Enable When set to "1", it enables polling, via the SMI interface, of the PHY various status registers. 0 = Disable 1 = Enable
0	Reserved	RW 0x0	Reserved. Must write 0.

A.6.4 Miscellaneous Registers

Table 585: SDMA Configuration (SDC) Register (i=0–3)

Offset: Port2: 0x0003241C, Port3: 0x0003641C, Port0: 0x0007241C, Port1: 0x0007641C

Offset Formula: 0x0007241C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:26	Reserved	RO 0x0	Reserved
25	Reserved	RSVD 0x0	Reserved

Table 585: SDMA Configuration (SDC) Register (i=0–3) (Continued)

Offset: Port2: 0x0003241C, Port3: 0x0003641C, Port0: 0x0007241C, Port1: 0x0007641C

Offset Formula: 0x0007241C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
24:22	TxBSZ	RW 0x4	Tx Burst Size Sets the maximum burst size for Tx DMA transactions: **NOTE:** This field effects only data-transfers. 0 = 1 64-bit: Burst limited to 1 64-bit words 1 = 2 64-bit: Burst limited to 2 64-bit words 2 = 4 64-bit: Burst limited to 4 64-bit words 3 = 8 64-bit: Burst limited to 8 64-bit words 4 = 16 64-bit: Burst limited to 16 64-bit words
21:7	Reserved	RSVD 0x0	Reserved
6	SwapMode	RW 0x0	Rx and Tx Descriptors Swap mode 0 = No swapping 1 = Byte swap: In every 64-bit word of the descriptor, the byte order is swapped such that byte 0 is placed in byte 7, byte 7 is placed in byte 0, byte 1 is placed in byte 6, byte 6 is placed in byte 1, byte 2 is placed in byte 5, etc.
5	BLMT	RW 0x1	Big/Little Endian Transmit Mode The <BLMT> field only affects a data transfer from memory. 0 = Byte swap 1 = No swap
4	BLMR	RW 0x1	Big/Little Endian Receive Mode The <BLMR> field only affects a data transfer to memory. 0 = Byte swap 1 = No swap
3:1	RxBSZ	RW 0x4	Rx Burst Size Sets the maximum burst size for Rx DMA transactions: **NOTE:** This field effects only data-transfers. 0 = 1 64-bit: Burst limited to 1 64-bit words 1 = 2 64-bit: Burst limited to 2 64-bit words 2 = 4 64-bit: Burst limited to 4 64-bit words 3 = 8 64-bit: Burst limited to 8 64-bit words 4 = 16 64-bit: Burst limited to 16 64-bit words
0	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

A.6.5 Networking Controller Miscellaneous Registers

Table 586: Port Acceleration Mode (PACC) Register (i=0–3)

Offset: Port2: 0x00032500, Port3: 0x00036500, Port0: 0x00072500, Port1: 0x00076500

Offset Formula: 0x00072500+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:3	Reserved	RO 0x0	Reserved
2:0	AccelerationMode	RW 0x0	Controls the acceleration mode of the port. This field should be configured during initialization. **NOTE:** 1 = Enhanced mode 1: Queue management: counters mode Hardware parsing Buffer management by software 2 = Enhanced mode 2: Queue management: counters mode Hardware parsing Buffer management by hardware or software

Table 587: Port BM Address (PBMADDR) Register (i=0–3)

Offset: Port2: 0x00032504, Port3: 0x00036504, Port0: 0x00072504, Port1: 0x00076504

Offset Formula: 0x00072504+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the buffer manager is used, as defined by the Port Acceleration mode (PACC) register.			
31:11	BMAddress	RW 0x0	Buffer Management Unit Address (bits[31:11]) For enabling the Networking Controller accessing the BM unit, one of the Networking Controller address decoding windows has to be configured for accessing the BM unit. The window size has to be configured to 64KB. In this field should be configured bits [31:11] of the base address of the window used for BM unit access.
10:0	Reserved	RO 0x0	Reserved

Table 588: Port Version (PVersion) Register (i=0–3)

Offset: Port2: 0x000325BC, Port3: 0x000365BC, Port0: 0x000725BC, Port1: 0x000765BC

Offset Formula: 0x000725BC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 588: Port Version (PVersion) Register (i=0–3) (Continued)

Offset: Port2: 0x000325BC, Port3: 0x000365BC, Port0: 0x000725BC, Port1: 0x000765BC

Offset Formula: 0x000725BC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
7:0	Version	RO 0x10	Networking controller version

A.6.6 RX DMA Hardware Parser Registers

Table 589: VLAN EtherType (EVLANE) Register (i=0–3)

Offset: Port2: 0x00032410, Port3: 0x00036410, Port0: 0x00072410, Port1: 0x00076410

Offset Formula: 0x00072410+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved
15:0	VL_EtherType	RW 0x8100	The EtherType for packets carrying the VLAN tag, for 802.1p priority field processing, and for continued parsing of the received frames layer3/4 headers.

Table 590: MAC Address Low (MACAL) Register (i=0–3)

Offset: Port2: 0x00032414, Port3: 0x00036414, Port0: 0x00072414, Port1: 0x00076414

Offset Formula: 0x00072414+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Read Only
15:0	MAC[15:0]	RW 0x0	The least significant bits of the MAC Address Used for both flow-control Pause frames as a source address, as well as for address filtering.

Table 591: MAC Address High (MACAH) Register (i=0–3)

Offset: Port2: 0x00032418, Port3: 0x00036418, Port0: 0x00072418, Port1: 0x00076418

Offset Formula: 0x00072418+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	MAC[47:16]	RW 0x0	The most significant bits of the MAC Address Used for both flow-control Pause frames as source address, as well as for address filtering. **NOTE:** <MAC[40]> is the Multicast/Unicast bit.

Table 592: IP Differentiated Services CodePoint 0 to Priority (DSCP0) Register (i=0–3)

Offset: Port2: 0x00032420, Port3: 0x00036420, Port0: 0x00072420, Port1: 0x00076420

Offset Formula: 0x00072420+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved
29:0	TOS_Q[29:0]	RW 0x0	DSCP(6-bits) to queue number mapping. In IPv4, DSCP is in TOS field. In IPv6, DSCP is in Traffic Class field. The Priority queue mapping of received frames with DSCP values 0 (corresponding to TOS_Q[2:0]) through 9 (corresponding to TOS_Q[29:27]). Each queue mapping field contains 3 bits, indicating queues 0 to 7. **NOTE:** The initial value means that ToS does not effect queue decisions. NOTE: IPv6 based queue mapping is not supported in basic mode(<AccelerationMode> field = 0)

Table 593: IP Differentiated Services CodePoint 1 to Priority (DSCP1) Register (i=0–3)

Offset: Port2: 0x00032424, Port3: 0x00036424, Port0: 0x00072424, Port1: 0x00076424

Offset Formula: 0x00072424+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 593: IP Differentiated Services CodePoint 1 to Priority (DSCP1) Register (i=0–3) (Continued)

Offset: Port2: 0x00032424, Port3: 0x00036424, Port0: 0x00072424, Port1: 0x00076424

Offset Formula: 0x00072424+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
29:0	TOS_Q[59:30]	RW 0x0	DSCP(6-bits) to queue number mapping. In IPv4, DSCP is in TOS field. In IPv6, DSCP is in Traffic Class field. The Priority queue mapping of received frames with DSCP values 10 (corresponding to TOS_Q[32:30]) through 19 (corresponding to TOS_Q[59:57]). Each queue mapping field contains 3 bits, indicating queues 0 to 7. **NOTE:** The initial value means that ToS does not effect queue decisions. NOTE: IPv6 based queue mapping is not supported in basic mode(<AccelerationMode> field = 0)

Table 594: IP Differentiated Services CodePoint 2 to Priority (DSCP2) Register (i=0–3)

Offset: Port2: 0x00032428, Port3: 0x00036428, Port0: 0x00072428, Port1: 0x00076428

Offset Formula: 0x00072428+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved
29:0	TOS_Q[89:60]	RW 0x0	DSCP(6-bits) to queue number mapping. In IPv4, DSCP is in TOS field. In IPv6, DSCP is in Traffic Class field. The Priority queue mapping of received frames with DSCP values 20 (corresponding to TOS_Q[62:60]) through 29 (corresponding to TOS_Q[89:87]). Each queue mapping field contains 3 bits, indicating queues 0 to 7. **NOTE:** The initial value means that ToS does not effect queue decisions. NOTE: IPv6 based queue mapping is not supported in basic mode(<AccelerationMode> field = 0)

Table 595: IP Differentiated Services CodePoint 3 to Priority (DSCP3) Register (i=0–3)

Offset: Port2: 0x0003242C, Port3: 0x0003642C, Port0: 0x0007242C, Port1: 0x0007642C

Offset Formula: 0x0007242C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved
29:0	TOS_Q[119:90]	RW 0x0	DSCP(6-bits) to queue number mapping. In IPv4, DSCP is in TOS field. In IPv6, DSCP is in Traffic Class field. The Priority queue mapping of received frames with DSCP values 30 (corresponding to TOS_Q[92:90]) through 39 (corresponding to TOS_Q[119:117]). Each queue mapping field contains 3 bits, indicating queues 0 to 7. **NOTE:** The initial value means that ToS does not effect queue decisions. NOTE: IPv6 based queue mapping is not supported in basic mode(<AccelerationMode> field = 0)

Table 596: IP Differentiated Services CodePoint 4 to Priority (DSCP4) Register (i=0–3)

Offset: Port2: 0x00032430, Port3: 0x00036430, Port0: 0x00072430, Port1: 0x00076430

Offset Formula: 0x00072430+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved
29:0	TOS_Q[149:120]	RW 0x0	DSCP(6-bits) to queue number mapping. In IPv4, DSCP is in TOS field. In IPv6, DSCP is in Traffic Class field. The Priority queue mapping of received frames with DSCP values 40 (corresponding to TOS_Q[122:120]) through 49 (corresponding to TOS_Q[149:147]). Each queue mapping field contains 3 bits, indicating queues 0 to 7. **NOTE:** The initial value means that ToS does not effect queue decisions. NOTE: IPv6 based queue mapping is not supported in basic mode(<AccelerationMode> field = 0)

Table 597: IP Differentiated Services CodePoint 5 to Priority (DSCP5) Register (i=0–3)

Offset: Port2: 0x00032434, Port3: 0x00036434, Port0: 0x00072434, Port1: 0x00076434

Offset Formula: 0x00072434+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved
29:0	TOS_Q[179:150]	RW 0x0	DSCP(6-bits) to queue number mapping. In IPv4, DSCP is in TOS field. In IPv6, DSCP is in Traffic Class field. The Priority queue mapping of received frames with DSCP values 50 (corresponding to TOS_Q[152:150]) through 59 (corresponding to TOS_Q[179:177]). Each queue mapping field contains 3 bits, indicating queues 0 to 7. **NOTE:** The initial value means that ToS does not effect queue decisions. NOTE: IPv6 based queue mapping is not supported in basic mode(<AccelerationMode> field = 0)

Table 598: IP Differentiated Services CodePoint 6 to Priority (DSCP6) Register (i=0–3)

Offset: Port2: 0x00032438, Port3: 0x00036438, Port0: 0x00072438, Port1: 0x00076438

Offset Formula: 0x00072438+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:12	Reserved	RO 0x0	Reserved
11:0	TOS_Q[191:180]	RW 0x0	DSCP(6-bits) to queue number mapping. In IPv4, DSCP is in TOS field. In IPv6, DSCP is in Traffic Class field. The Priority queue mapping of received frames with DSCP values 60 (corresponding to TOS_Q[2:0]) through 63 (corresponding to TOS_Q[191:180]). Each queue mapping field contains 3 bits, indicating queues 0 to 7. **NOTE:** The initial value means that ToS does not effect queue decisions. NOTE: IPv6 based queue mapping is not supported in basic mode(<AccelerationMode> field = 0)

Table 599: VLAN Priority Tag to Priority (VPT2P) Register (i=0–3)

Offset: Port2: 0x00032440, Port3: 0x00036440, Port0: 0x00072440, Port1: 0x00076440

Offset Formula: 0x00072440+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:24	Reserved	RO 0x0	Reserved
23:0	Priority[23:0]	RW 0x0	The Priority queue mapping of received frames with 802.1p priority field values 0 (corresponding to Priority[2:0]) through 7 (corresponding to Priority[23:21]). **NOTE:** The initial value means that Priority does not effect the queue decisions.

Table 600: Ethernet Type Priority Register (i=0–3)

Offset: Port2: 0x000324BC, Port3: 0x000364BC, Port0: 0x000724BC, Port1: 0x000764BC

Offset Formula: 0x000724BC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.			
31:22	Reserved	RO 0x0	Reserved
21	ForceUnicstHit	RW 0x0	Force Unicast Hit When set, each received Unicast packet is handled as it matches the Unicast destination address filtering.
20:5	EtherTypePriVal	RW 0x0806	EtherType priority value EtherType value for which the priority will be given when enabled by when EtherTypePriEn=0x1 or EtherTypePriFrstEn = 0x1.
4:2	EtherTypePriQ	RW 0x0	EtherType Priority Queue The queue number for packets that match the <EtherTypePri> value, when enabled by <EtherTyePriEn> (set to 1) or <EtherTypePriFrstEn>(set to 1).

Table 600: Ethernet Type Priority Register (i=0–3) (Continued)

Offset: Port2: 0x000324BC, Port3: 0x000364BC, Port0: 0x000724BC, Port1: 0x000764BC

Offset Formula: 0x000724BC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
1	EtherTypePriFrstEn	RW 0x0	The EtherType Priority Enable and queuing decision is upon <EtherTypePriQ> if hit, regardless of the other Ethernet packet parameters (e.g., DSCP). If enabled, this field overrides <EtherTypePriEn> . 0 = Disable: No EtherType priority 1 = Enable: EtherType priority enabled, and queueing is determined only upon it.
0	EtherTypePriEn	RW 0x0	EtherType Priority Enable. 0 = Disable: No EtherType priority 1 = Enable: EtherType priority enabled

Table 601: Destination Address Filter Special Multicast Table (DFSMT)<n> Register (i=0–3, n=0–63)

Offset: Port2Register0: 0x00033400, Port2Register1: 0x00033404...Port1Register63: 0x000774FC

Offset Formula: 0x00073400+n*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-63) represents Register

Bit	Field	Type/InitVal	Description
NOTE: Every register holds four entries. A total of 64 registers appear in the table in consecutive order. This register is used for address filtering of IP multicast packets. The 8 least significant bits of the packet destination address are used as pointer to the coresponding entry. When using the DSA tag with the queuing mode that is based on CPU_Code (<DAPrefix> = 0x2), this register is no longer used for address filtering. The CPU_Code value is used as pointer to the coresponding entry.			
31:29	Reserved	RSVD 0x0	Reserved
28	Reserved[3]	RW 0x0	Reserved Must be set to 0
27:25	Queue[4*n+3]	RW 0x0	For pointer equal to 4*n+3: Determines the Queue number if Pass[4*n+3]=1.
24	Pass[4*n+3]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+3: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
23:21	Reserved	RSVD 0x0	Reserved

Table 601: Destination Address Filter Special Multicast Table (DFSMT)<n> Register (i=0–3, n=0–63) (Continued)

Offset: Port2Register0: 0x00033400, Port2Register1: 0x00033404...Port1Register63: 0x000774FC

Offset Formula: 0x00073400+n*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3)

Bit	Field	Type/InitVal	Description
20	Reserved[2]	RW 0x0	Reserved Must be set to 0
19:17	Queue[4*n+2]	RW 0x0	For pointer equal to 4*n+2: Determines the Queue number if Pass[4*n+2]=1.
16	Pass[4*n+2]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+2: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
15:13	Reserved	RSVD 0x0	Reserved
12	Reserved[1]	RW 0x0	Reserved Must be set to 0
11:9	Queue[4*n+1]	RW 0x0	For pointer equal to 4*n+1: Determines the Queue number if Pass[4*n+1]=1.
8	Pass[4*n+1]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+1: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
7:5	Reserved	RSVD 0x0	Reserved
4	Reserved[0]	RW 0x0	Reserved Must be set to 0
3:1	Queue[4*n+0]	RW 0x0	For pointer equal to 4*n+0: Determines the Queue number if Pass[4*n+0]=1.
0	Pass[4*n+0]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+0: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame

Table 602: Destination Address Filter Other Multicast Table (DFOMT)<n> Register (i=0–3, n=0–63)

Offset: Port2Register0: 0x00033500, Port2Register1: 0x00033504...Port1Register63: 0x000775FC

Offset Formula: 0x00073500+n*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-63) represents Register

Bit	Field	Type/InitVal	Description
NOTE: Every register holds four entries. A total of 64 registers appear in this table in consecutive order. The CRC of the packet destination address is used as pointer to the coresponding entry.			
31:29	Reserved	RSVD 0x0	Reserved
28	Reserved[3]	RW 0x0	Reserved Must be set to 0
27:25	Queue[4*n+3]	RW 0x0	For pointer equal to 4*n+3: Determines the Queue number if Pass[4*n+3]=1.
24	Pass[4*n+3]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+3: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
23:21	Reserved	RSVD 0x0	Reserved
20	Reserved[2]	RW 0x0	Reserved Must be set to 0
19:17	Queue[4*n+2]	RW 0x0	For pointer equal to 4*n+2: Determines the Queue number if Pass[4*n+2]=1.
16	Pass[4*n+2]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+2: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
15:13	Reserved	RSVD 0x0	Reserved
12	Reserved[1]	RW 0x0	Reserved Must be set to 0
11:9	Queue[4*n+1]	RW 0x0	For pointer equal to 4*n+1: Determines the Queue number if Pass[4*n+1]=1.
8	Pass[4*n+1]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+1: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame

Table 602: Destination Address Filter Other Multicast Table (DFOMT)<n> Register (i=0–3, n=0–63) (Continued)

Offset: Port2Register0: 0x00033500, Port2Register1: 0x00033504...Port1Register63: 0x000775FC

Offset Formula: 0x00073500+n*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3)

Bit	Field	Type/InitVal	Description
7:5	Reserved	RSVD 0x0	Reserved
4	Reserved[0]	RW 0x0	Reserved Must be set to 0
3:1	Queue[4*n+0]	RW 0x0	For pointer equal to 4*n+0: Determines the Queue number if Pass[4*n+0]=1.
0	Pass[4*n+0]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+0: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame

Table 603: Destination Address Filter Unicast Table (DFUT)<n> Register (i=0–3, n=0–3)

Offset: Port2Register0: 0x00033600, Port2Register1: 0x00033604...Port1Register3: 0x0007760C

Offset Formula: 0x00073600+n*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-3) represents Register

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register. **NOTE:** Every register holds four entries. A total of four registers appear in this table in consecutive order. This register is used for address filtering of unicast packets. The 4 least significant bits of the packet destination address are used as pointer to the coresponding entry.			
31:29	Unused[3]	RO 0x0	Reserved
28	Reserved[3]	RW 0x0	Reserved Must be set to 0
27:25	Queue[4*n+3]	RW 0x0	For pointer equal to 4*n+3: Determines the Queue number if Pass[4*n+3]=1.
24	Pass[4*n+3]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+3: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
23:21	Unused[2]	RO 0x0	Reserved

Table 603: Destination Address Filter Unicast Table (DFUT)<n> Register (i=0–3, n=0–3) (Continued)

Offset: Port2Register0: 0x00033600, Port2Register1: 0x00033604...Port1Register3: 0x0007760C

Offset Formula: 0x00073600+n*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-3) represents Register

Bit	Field	Type/InitVal	Description
20	Reserved[2]	RW 0x0	Reserved Must be set to 0
19:17	Queue[4*n+2]	RW 0x0	For pointer equal to 4*n+2: Determines the Queue number if Pass[4*n+2]=1.
16	Pass[4*n+2]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+2: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
15:13	Unused[1]	RO 0x0	Reserved
12	Reserved[1]	RW 0x0	Reserved Must be set to 0
11:9	Queue[4*n+1]	RW 0x0	For pointer equal to 4*n+1: Determines the Queue number if Pass[4*n+1]=1.
8	Pass[4*n+1]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+1: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame
7:5	Unused[0]	RO 0x0	Reserved
4	Reserved[0]	RW 0x0	Reserved Must be set to 0
3:1	Queue[4*n+0]	RW 0x0	For pointer equal to 4*n+0: Determines the Queue number if Pass[4*n+0]=1.
0	Pass[4*n+0]	RW 0x0	Determines whether to filter or accept, for pointer equal to 4*n+0: 0 = Reject: Reject (discard) frame 1 = Accept: Accept frame

A.6.7 RX DMA Miscellaneous Registers

Table 604: Register (i=0–3)

Offset: Port2: 0x00031D00, Port3: 0x00035D00, Port0: 0x00071D00, Port1: 0x00075D00

Offset Formula: 0x00071D00+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

Table 605: Port Rx Minimal Frame Size (PxMFS) Register (i=0–3)

Offset: Port2: 0x0003247C, Port3: 0x0003647C, Port0: 0x0007247C, Port1: 0x0007647C

Offset Formula: 0x0007247C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:7	Reserved	RO 0x0	Reserved
6:2	RxMFS[6:2]	RW 0x10	Contains the Receive Minimal Frame Size in units of 4 bytes. Valid Range: 10-16 (equivalent to 40-64 bytes). For packets that include Marvell Header (when MHEn is set in Marvell Header Register), the packets are dropped if the received packets size < MFS. For packets without Marvell Header (when MHEn is cleared in Marvell Header Register), the packets are dropped if the received packets size < (MFS-2).
1:0	Reserved	RO 0x0	Reserved

Table 606: Port Rx Discard Frame Counter (PxDFC) Register (i=0–3)

Offset: Port2: 0x00032484, Port3: 0x00036484, Port0: 0x00072484, Port1: 0x00076484

Offset Formula: 0x00072484+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	Rx Discard[31:0]	ROC 0x0	Number of frames that were discarded because of address filtering or resource error This register is reset every time the CPU reads from it.

Copyright © 2014 Marvell

Table 607: Port Overrun Frame Counter (PxOFC) Register (i=0–3)

Offset: Port2: 0x00032488, Port3: 0x00036488, Port0: 0x00072488, Port1: 0x00076488

Offset Formula: 0x00072488+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	Rx Overrun[31:0]	RO 0x0	Number of frames that were received and could not be processed because the RX FIFO was full. This register is reset every time the CPU reads from it.

Table 608: Receive Queue Command (RQC) Register (i=0–3)

Offset: Port2: 0x00032680, Port3: 0x00036680, Port0: 0x00072680, Port1: 0x00076680

Offset Formula: 0x00072680+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Read Only
15:8	DISQ	RW 0x0	Disable Queue[7:0] One bit per each queue.Writing these bits set to 1 disables the queue. The receive DMA will stop the Receive process to this queue, on the next packet boundary. Writing 1 to DISQ bit resets the matching ENQ bit after the RxDMA finished processing the queue, if the ENQ bit was used while the CPU wrote the DISQ for it. Writing 0 to DISQ bit has no effect. When receive DMA encounters a descriptor with a parity error, the DMA will disable the queue but not set the DISQ bit for that queue, thus reading DISQ and ENQ bits discriminates between queues disabled by CPU and those stopped by DMA due to a parity error on descriptor.

Table 608: Receive Queue Command (RQC) Register (i=0–3) (Continued)

Offset: Port2: 0x00032680, Port3: 0x00036680, Port0: 0x00072680, Port1: 0x00076680

Offset Formula: 0x00072680+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
7:0	ENQ	RW 0x0	Enable Queue[7:0] One bit per queue. Writing these bits set to 1 enables the queue. The Receive DMA fetches the first descriptor from the address programmed in the PRXDQA register for that queue and starts the Receive process (in case of SW buffers management). Writing 1 to ENQ bit resets the matching <DISQ> bit. Writing 1 to ENQ bit of a DMA that is already in enable state, has no effect. Writing 0 to ENQ bit has no effect. When the receive DMA encounters a descriptor with a parity error, the DMA clears the ENQ bit for that queue. Reading these bits reports the active enable status for each queue. **NOTE:** A descriptor with parity error results in closing the status of the packet with a resource error condition. **NOTE:**

A.6.8 RX DMA Networking Controller Miscellaneous Registers

Table 609: Port RX Queues Configuration (PRXC)<q> Register (i=0–3, q=0–7)

Offset: Port2RX Queue0: 0x00031400, Port2RX Queue1: 0x00031404...Port1RX Queue7: 0x0007541C

Offset Formula: 0x00071400+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
31:17	Reserved	RO 0x0	Reserved
16	UserPrefetchCmnd0	RW 0x0	Controls the Prefetch Command data to use. 1 = Prefetch0: Use PrefetchCommand0 field of the PRXF01 register.
16	Reserved	RSVD 0x0	Reserved
15:12	Reserved	RO 0x0	Reserved

Table 609: Port RX Queues Configuration (PRXC)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2RX Queue0: 0x00031400, Port2RX Queue1: 0x00031404...Port1RX Queue7: 0x0007541C

Offset Formula: 0x00071400+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
11:8	PacketOffset	RW 0x0	RX Packet Offset This field is valid only if Queues Management - counters mode is selected by Accelerationmode field of the PACC register. This field defines the offset, from start of buffer, where the packet should be stored. The offset is in units of qwords (8 bytes). This offset is only implemented for the first buffer used to store the packet (indicated by the <F> bit of the descriptor). For all other buffers offset will be as following: 8 bytes - for queues using buffers from the BMU. 0 bytes - for all other queues NOTE: For all queues configured to use the BMU, the minimum legal value of this field is 8.
7:6	PoolIdLong	RW 0x1	Pool ID Long This field is valid only if buffers for this RX queue are allocated by the BM unit (HWBufAlloc==1) and HW buffer management is selected by AccelerationMode field of the PACC register. It defines the BM pool no. to be used for packets that can not be stored in a single buffer of the pool pointed by the Pool Id Short field. **NOTE:** This field is valid only in Enhanced Modes that use hardware buffers management - as defined by the Port Acceleration Mode (PACC) register. **NOTE:**
7:4	Reserved	RSVD 0x4	Reserved
5:4	PoolIdShort	RW 0x0	Pool ID Short This field is valid only if buffers for this RX queue are allocated by the BM unit (HWBufAlloc==1) and HW buffer management is selected by AcceleratinMode field of the PACC register. It defines the BM pool no. to be used for packets that can be stored in a single buffer of this pool. **NOTE:** This field is valid only in Enhanced Modes that use hardware buffers management - as defined by the Port Acceleration Mode (PACC) register. **NOTE:**
3:1	Reserved	RO 0x0	Reserved

Table 609: Port RX Queues Configuration (PRXC)<q> Register (i=0–3, q=0–7) (Continued)

Offset: **Port2RX Queue0: 0x00031400, Port2RX Queue1: 0x00031404...Port1RX Queue7: 0x0007541C**

Offset Formula: **0x00071400+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue**

Bit	Field	Type/InitVal	Description
0	HwBuffAlloc	RW 0x0	Hardware Buffer Allocation This bit defines the buffer allocation method for packets in this RX queue. It is used only if the AccelerationMode field of the PACC register, is configured for buffer management by BM unit. 0 = By SW: The buffer pointer is provided in the descriptor 1 = By BM: The buffer pointer is provided by the BM unit.
0	Reserved	RSVD 0x0	Reserved

Table 610: Port RX Queues Snoop (PRXSNP)<q> Register (i=0–3, q=0–7)

Offset: **Port2RX Queue0: 0x00031420, Port2RX Queue1: 0x00031424...Port1RX Queue7: 0x0007543C**

Offset Formula: **0x00071420+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue**

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:30	Reserved	RO 0x0	Reserved
29:16	L2DepositNoOfBytes	RW 0x0	L2 Deposit Number of Bytes It defines the number of bytes from start of packet, for which the "L2 deposit enable" indication will be set, during the packet data transfer. **NOTE:** Actually the "L2 deposit enable" indication will be set for an integer number of MBus bursts. This number is minimum number of MBus bursts that are required to transfer the configured number of bytes. The MBus burst size is configured by the RxBSZ field of the SDMA Configuration (SDC) register.
15:14	Reserved	RO 0x0	Reserved

Table 610: Port RX Queues Snoop (PRXSNP)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2RX Queue0: 0x00031420, Port2RX Queue1: 0x00031424...Port1RX Queue7: 0x0007543C

Offset Formula: 0x00071420+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
13:0	SnoopNoOfBytes	RW 0x0	Snoop Number of Bytes It defines the number of bytes from start of packet, for which the "Snoop enable" indication will be set, during the packet data transfer. **NOTE:** Actually the "Snoop enable" indication will be set for an integer number of MBus bursts. This number is minimum number of MBus bursts that are required to transfer the configured number of bytes. The MBus burst size is configured by the RxBSZ field of the SDMA Configuration (SDC) register.

Table 611: Port RX Prefetch 0_1 (PRXF01)<q> Register (i=0–3, q=0–7)

Offset: Port2RX Queue0: 0x00031440, Port2RX Queue1: 0x00031444...Port1RX Queue7: 0x0007545C

Offset Formula: 0x00071440+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:16	PrefetchCommand1	RW 0x0	Prefetch Command1 This field is contains prefetch command data, to be used by a dedicated prefetch HW, to prefetch parts of the packet. The data from this field or from one of the other three Prefetch Command fields is written to the descriptor. When using the hardware parser (as defined by the AccelerationMode field of the PACC register) the data contained in PrefetchCommand0 field is written to the descriptor.
31:16	Reserved	RSVD 0x0	Reserved
15:0	PrefetchCommand0	RW 0x0	Prefetch Command0 This field is contains prefetch command data, to be used by a dedicated prefetch HW, to prefetch parts of the packet. The data from this field is written to the descriptor. When using the hardware parser (as defined by the AccelerationMode field of the PACC register) the data contained in PrefetchCommand0 field is written to the descriptor.

Table 612: Port RX Queues Descriptors Queue Address (PRXDQA) <q> Register (i=0–3, q=0–7)

Offset: Port2RX Queue0: 0x00031480, Port2RX Queue1: 0x00031484...Port1RX Queue7: 0x0007549C

Offset Formula: 0x00071480+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:5	DescriptorsQueueStartingAddress	RW 0x0	Descriptors Queue Starting Address This field is valid only if one of the enhanced modes is selected by the AccelerationMode field of the PACC register. The field defines the starting address of the descriptors queue. The starting address is in units of 32 bytes. Therefore the descriptors will be actually aligned to 32 bytes. .
4:0	Reserved	RO 0x0	Reserved

Table 613: Port RX Queues Descriptors Queue Size (PRXDQS)<q> Register (i=0–3, q=0–7)

Offset: Port2RX Queue0: 0x000314A0, Port2RX Queue1: 0x000314A4...Port1RX Queue7: 0x000754BC

Offset Formula: 0x000714A0+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:19	BufferSize	RW 0x0	Buffer Size This field is valid as following: In acceleration modes that are not using the buffer manager, as defined by the Port Acceleration mode (PACC) register - for all queues In acceleration modes that are using the buffer manager, as defined by the Port Acceleration mode (PACC) register - for queues with software buffer allocation, as configured by the <HWBufAlloc> in the PRXC register.
18:14	Reserved	RO 0x0	Reserved
13:0	DescriptorsQueueSize	RW 0x0	Descriptors Queue Size This field is valid only if one of the enhanced modes is selected by the AccelerationMode field of the PACC register. It contains the number of descriptors that can be stored in the descriptors queue. **NOTE:** Since the descriptor size is 32 bytes, the interval between successive descriptors will be also 32 bytes.

Copyright © 2014 Marvell

Table 614: Port RX Queues Descriptors Queue Threshold (PRXDQTH)<q> Register (i=0–3, q=0–7)

Offset: Port2RX Queue0: 0x000314C0, Port2RX Queue1: 0x000314C4...Port1RX Queue7: 0x000754DC

Offset Formula: 0x000714C0+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:30	Reserved2	RO 0x0	Reserved
29:16	NonOccupiedDescriptorsThreshold	RW 0x0	Non Occupied Descriptors Threshold This field is only valid in Enhanced modes, as selected by the AccelerationMode field of the PACC register. When the number of descriptors not yet used by the Networking Controller, as reflected by the <NonOccupiedlDescriptorsCounter> field in the PRXS register, is less than the value of this field, then the relevant RxDescriptorThresholdAlertQueue bit in PRXTXThIC Register is set.
15:14	Reserverd1	RO 0x0	Reserved
13:0	OccupiedDescriptorsThreshold	RW 0x0	Occupied Descriptors Threshold This field is only valid in Enhanced modes, as selected by the AccelerationMode field of the PACC register. When the number of buffers that contain received packets data, and not yet processed by CPU, as reflected by the <OccupiedDescriptorsCounter> field in the PRXS register, is greater than the value of this field, then the relevant RxBufferInterruptCoaliascingThresholdAlertPriorityQueue bit in PRXTXThIC Register is set.

Table 615: Port RX Queues Status (PRXS)<q> Register (i=0–3, q=0–7)

Offset: **Port2RX Queue0: 0x000314E0, Port2RX Queue1: 0x000314E4...Port1RX Queue7: 0x000754FC**

Offset Formula: **0x000714E0+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue**

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:30	Reserved	RO 0x0	Reserved
29:16	NonOccupiedDescriptorsCounter	RO 0x0	Non Occupied Descriptors Counter This field indicates the number of RX descriptors received from CPU and not yet fetched by the Networking Controller. It is incremented by CPU, by writing to the "NoOfNewDescriptors" field in the PRXSU register, and decremented by Networking Controller with the completion of each packet reception and transfer to data buffer.
15:14	Reserved	RO 0x0	Reserved
13:0	OccupiedDescriptorsCounter	RO 0x0	Occupied Descriptors Counter This field indicates the number of descriptors that point to buffers that contain packets data transferred by the Networking Controller and not processed yet by the CPU. This number is incremented by the Networking Controller, with the completion of each packet reception and transfer to data buffer. This number is decremented by CPU, following RX descriptors processing. The decrement is done by writing to the "NoOfProcessedDescriptors" field in the PRXSU register. The decrement is by the value written to this field.

Table 616: Port RX Queues Status Update (PRXSU)<q> Register (i=0–3, q=0–7)

Offset: **Port2RX Queue0: 0x00031500, Port2RX Queue1: 0x00031504...Port1RX Queue7: 0x0007551C**

Offset Formula: **0x00071500+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue**

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:24	Reserved	RO 0x0	Reserved

Table 616: Port RX Queues Status Update (PRXSU)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2RX Queue0: 0x00031500, Port2RX Queue1: 0x00031504...Port1RX Queue7: 0x0007551C

Offset Formula: 0x00071500+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
23:16	NoOfNewDescriptors	WO 0x0	Number Of New Descriptors To this field the CPU writes the number of new descriptors added to the descriptors queue. Following writing to this field, the NonOccupiedlDescriptorsCounter field of the PxRXyS register is updated (incremented).
15:8	Reserved	RO 0x0	Reserved
7:0	NoOfProcessedDescriptors	WO 0x0	Number Of Processed Descriptors To this field the CPU writes the number of descriptors with packets information received from the Networking Controller, that where processed by CPU and can be released to the Networking Controller for new packets usage. Following writing to this field, the OccupiedDescriptorsCounter field of the PRXS register is updated (decremented).

Table 617: Port RX Descriptor Index (PRXDI)<q> Register (i=0–3, q=0–7)

Offset: Port2RX Queue0: 0x00031520, Port2RX Queue1: 0x00031524...Port1RX Queue7: 0x0007553C

Offset Formula: 0x00071520+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:14	Reserved	RO 0x0	Reserved
13:0	NextDescriptorIndex	RO 0x0	Next Descriptor Index This field indicates the index of the next descriptor that is going to be used. The valid values are 0 - "DescriptorsQueueSize" - 1.

Table 618: Port Pool0 Buffer Size (PPL0BSZ) Register (i=0–3)

Offset: Port2: 0x00031700, Port3: 0x00035700, Port0: 0x00071700, Port1: 0x00075700

Offset Formula: 0x00071700+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This field is valid only in modes in which the buffer manager is used, as defined by the Port Acceleration mode (PACC) register			
31:16	Reserved	RO 0x0	Reserved
15:3	Pool0BufferSize	RW 0x0	Pool #0 Buffer Size This field defines the size of the buffers from pool #0. The size is in units of 8 bytes.
2:0	Reserved	RO 0x0	Reserved

Table 619: Port Pool1 Buffer Size (PPL1BSZ) Register (i=0–3)

Offset: Port2: 0x00031704, Port3: 0x00035704, Port0: 0x00071704, Port1: 0x00075704

Offset Formula: 0x00071704+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This field is valid only in modes in which the buffer manager is used, as defined by the Port Acceleration mode (PACC) register			
31:16	Reserved	RO 0x0	Reserved
15:3	Pool1BufferSize	RW 0x0	Pool 1 Buffer Size This field defines the size of the buffers from <Pool1BufferSize>. The size is in units of 8 bytes.
2:0	Reserved	RO 0x0	Reserved

Table 620: Port Pool2 Buffer Size (PPL2BSZ) Register (i=0–3)

Offset: Port2: 0x00031708, Port3: 0x00035708, Port0: 0x00071708, Port1: 0x00075708

Offset Formula: 0x00071708+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This field is valid only in modes in which the buffer manager is used, as defined by the Port Acceleration mode (PACC) register			
31:16	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 620: Port Pool2 Buffer Size (PPL2BSZ) Register (i=0–3) (Continued)

Offset: Port2: 0x00031708, Port3: 0x00035708, Port0: 0x00071708, Port1: 0x00075708

Offset Formula: 0x00071708+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:3	Pool2BufferSize	RW 0x0	Pool #2 Buffer Size This field defines the size of the buffers from pool #2. The size is in units of 8 bytes.
2:0	Reserved	RO 0x0	Reserved

Table 621: Port Pool3 Buffer Size (PPL3BSZ) Register (i=0–3)

Offset: Port2: 0x0003170C, Port3: 0x0003570C, Port0: 0x0007170C, Port1: 0x0007570C

Offset Formula: 0x0007170C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This field is valid only in modes in which the buffer manager is used, as defined by the Port Acceleration mode (PACC) register			
31:16	Reserved	RO 0x0	Reserved
15:3	Pool3BufferSize	RW 0x0	Pool #3 Buffer Size This field defines the size of the buffers from pool #3. The size is in units of 8 bytes.
2:0	Reserved	RO 0x0	Reserved

Table 622: Reserved_1710 Register (i=0–3)

Offset: Port2: 0x00031710, Port3: 0x00035710, Port0: 0x00071710, Port1: 0x00075710

Offset Formula: 0x00071710+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1:0	Reserved	RW 0x0	Reserved. Must write 0x3.

Table 623: Port RX Initialization (PRXINIT) Register (i=0–3)

Offset: Port2: 0x00031CC0, Port3: 0x00035CC0, Port0: 0x00071CC0, Port1: 0x00075CC0

Offset Formula: 0x00071CC0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:1	Reserved	RO 0x0	Reserved
0	RXDMAInit	RW 0x0	RX DMA INIT When this bit is set to "1", then the RX DMA module is initialized, including the various Descriptor counters, Fifo etc. Following RX DMA initialization and TX DMA initialization (by the TXDMAInit bit of the PTXINIT register), the acceleration mode may be changed (by the AccelerationMode field of the PACC register). 0 = Normal Operation 1 = RX DMA Initialization

A.6.9 RX DMA Wake on LAN Registers

Table 624: Arp IP0 Address (AIP0ADR) Register (i=0–3)

Offset: Port2: 0x00032498, Port3: 0x00036498, Port0: 0x00072498, Port1: 0x00076498

Offset Formula: 0x00072498+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register			
31:0	ArpIPAddr	RW 0x0	This field contains the value of IPV4 address, for which MAC ARP packets should be accepted, when in the PxC register bit RBArp is cleared and bit RBArpF is set.

Table 625: Arp IP1 Address (AIP1ADR) Register (i=0–3)

Offset: Port2: 0x0003249C, Port3: 0x0003649C, Port0: 0x0007249C, Port1: 0x0007649C

Offset Formula: 0x0007249C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register			
31:0	ArpIPAddr	RW 0x0	This field contains the value of IPV4 address, for which MAC ARP packets should be accepted, when in the PxC register bit RBArp is cleared and bit RBArpF is set.

Table 626: WOL Sleep (WOLSLP) Register (i=0–3)

Offset: Port2: 0x00033690, Port3: 0x00037690, Port0: 0x00073690, Port1: 0x00077690

Offset Formula: 0x00073690+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:2	Reserved	RO 0x0	Reserved
1	SleepModeActive	RO 0x0	Sleep Mode Active The 0 setting means that the port is not in Sleep mode. When this field is set to 1, the port is in Sleep mode.
0	SleepMode	RW 0x0	Sleep Mode 0x0 = Non-sleep mode 0x1 = Sleep mode **NOTE:** In Sleep mode, if a non-masked wake-up event is identified, then this bit is cleared (to non sleep mode). 0 = Non-sleep mode: Normal mode of operation 1 = Sleep mode: In this state received packets are discarded. Only non-masked wakup packets are accepted.

Table 627: WOL Wakeup Event Enable (WOLWEE) Register (i=0–3)

Offset: Port2: 0x00033694, Port3: 0x00037694, Port0: 0x00073694, Port1: 0x00077694

Offset Formula: 0x00073694+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:13	Reserved	RO 0x0	Reserved
12	Type3	RW 0x0	Type3 enable This bit controls identification of reception of packets as defined by the WOL Type3 register 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
11	Type2	RW 0x0	Type2 enable This bit controls identification of reception of packets as defined by the WOL Type2 register 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled

Table 627: WOL Wakeup Event Enable (WOLWEE) Register (i=0–3) (Continued)

Offset: Port2: 0x00033694, Port3: 0x00037694, Port0: 0x00073694, Port1: 0x00077694

Offset Formula: 0x00073694+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
10	Type1	RW 0x0	Type1 enable This bit controls identification of reception of packets as defined by the WOL Type1 register 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
9	Multicast	RW 0x0	Multicast enable This bit controls identification of reception of multicast packets 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
8	Unicast	RW 0x0	Unicast enable This bit controls identification of reception of unicast packets 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
7	ARPIP1Address	RW 0x0	ARP IP1 address enable This bit controls identification of reception of ARP packets, with IPV4 address equal with IP address programmed in the ARP IP1address register 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
6	ARPIP0Address	RW 0x0	ARP IP0 address enable This bit controls identification of reception of ARP packets, with IPV4 address equal with IP address programmed in the ARP IP0 address register 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
5	LinkChange	RW 0x0	Link Change enable This bit controls identification of link status change 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
4	MagicPattern	RW 0x0	Magic Pattern enable This bit controls identification of packets containing the magic pattern. 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
3	WakeupFrame3	RW 0x0	Wake up Frame 3 Enable This bit controls identification of packets containing wake up frame 3. 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled

Table 627: WOL Wakeup Event Enable (WOLWEE) Register (i=0–3) (Continued)

Offset: Port2: 0x00033694, Port3: 0x00037694, Port0: 0x00073694, Port1: 0x00077694

Offset Formula: 0x00073694+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
2	WakeupFrame2	RW 0x0	Wake up Frame 2 Enable This bit controls identification of packets containing wake up frame 2. 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
1	WakeupFrame1	RW 0x0	Wake up Frame 1 Enable This bit controls identification of packets containing wake up frame 1. 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled
0	WakeupFrame0	RW 0x0	Wake up Frame 0 Enable This bit controls identification of packets containing wake up frame 0. 0 = Wake up event identification is disabled 1 = Wake up event identification is enabled

Table 628: WOL Sleep Interrupt Cause (WOLSIC) Register (i=0–3)

Offset: Port2: 0x00033698, Port3: 0x00037698, Port0: 0x00073698, Port1: 0x00077698

Offset Formula: 0x00073698+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:13	Reserved	RO 0x0	Reserved
12	Type3	RW0C 0x0	Packet of Type 3 was identified. Packet of type as defined in the WOL Type3 (WOLT3) register was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified: register.
11	Type2	RW0C 0x0	Packet of Type 2 was identified. Packet of type as defined in the WOL Type2 (WOLT2) register was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified
10	Type1	RW0C 0x0	Packet of Type 1 was identified. Packet of type as defined in the WOL Type1 (WOLT1) register was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified: register.

Table 628: WOL Sleep Interrupt Cause (WOLSIC) Register (i=0–3) (Continued)

Offset: Port2: 0x00033698, Port3: 0x00037698, Port0: 0x00073698, Port1: 0x00077698

Offset Formula: 0x00073698+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
9	Multicast	RW0C 0x0	Multicast packet was identified Multicast packet that passed the non-sleep RX packets filter was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified
8	Unicast	RW0C 0x0	Unicast packet was identified Unicast packet that passed the non-sleep RX packets filter was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified: register.
7	ArpIP1Address	RW0C 0x0	Arp packet with IP address number 1 was identified. Arp packet with IP address equal to the IP address stored in the Arp IP1 Address (AIP1ADR) register was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified
6	ArpIP0Address	RW0C 0x0	Arp packet with IP address number 0 was identified. Arp packet with IP address equal to the IP address stored in the Arp IP0 Address (AIP0ADR) register was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified
5	LinkChange	RW0C 0x0	Link change was identified. This bit indicates that the link status change was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified
4	MagicPattern	RW0C 0x0	Magic pattern was identified. This bit indicates that the magic pattern was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified
3	WakeupFrame3	RW0C 0x0	Wake up frame 3 identified. This bit indicates that wake up frame 3 was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified
2	WakeupFrame2	RW0C 0x0	Wake up frame 2 identified. This bit indicates that wake up frame 2 was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified

Table 628: WOL Sleep Interrupt Cause (WOLSIC) Register (i=0–3) (Continued)

Offset: Port2: 0x00033698, Port3: 0x00037698, Port0: 0x00073698, Port1: 0x00077698

Offset Formula: 0x00073698+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
1	WakeupFrame1	RW0C 0x0	Wake up frame 1 identified. This bit indicates that wake up frame 1 was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified: register.
0	WakeupFrame0	RW0C 0x0	Wake up frame 0 identified. This bit indicates that wake up frame 0 was identified in sleep mode. 0 = Wake up event was not identified 1 = Wake up event was identified

Table 629: WOL Sleep Interrupt Mask (WOLSIM) Register (i=0–3)

Offset: Port2: 0x0003369C, Port3: 0x0003769C, Port0: 0x0007369C, Port1: 0x0007769C

Offset Formula: 0x0007369C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:13	Reserved	RO 0x0	Reserved
12:0	SleepModeInterruptsMask	RW 0x0	Sleep mode interrupts mask Mask bits to the corresponding bits of the WOL Sleep Interrupt Cause register (WOLSIC). It only effects interrupt generation to the Power Management Unit (PMU). 0 = Interrupt disabled: The corresponding wake up event is masked and it will not generate an interrupt. 1 = Interrupt enabled: The corresponding wake up event is enabled and will generate an interrupt.

Table 630: WOL Wakeup Events Counter (WOLWEC) Register (i=0–3)

Offset: Port2: 0x000336A0, Port3: 0x000376A0, Port0: 0x000736A0, Port1: 0x000776A0

Offset Formula: 0x000736A0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	EventsCounter	ROC 0x0	Events counter This register counts the number of identified and enabled wake-up events, while in wake mode. NOTEs: 1. Link change is not counted 2. If for the same packet more than one wake-up event type is identified, then the following groups of events are counted separately: a. Magic pattern b. Wake-up frames c. All other types of wake-up events

Table 631: WOL Type1 (WOLT1) Register (i=0–3)

Offset: Port2: 0x000336A4, Port3: 0x000376A4, Port0: 0x000736A4, Port1: 0x000776A4

Offset Formula: 0x000736A4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:25	Reseved	RW 0x0	Reserved
24	CompareMode	RW 0x0	Type no. 1 compare mode. This bit defines the packet field that will compared in order to identify type no. 1 wake up event. 0 = Ethernet type: The packet Ethernet type is compared with the register EtherType field 1 = IP protocol: The IPV4 protocol field (if IPV4 is detected) is compared with the register IPProtocol field
23:16	IPProtocol	RW 0x0	IP Protocol This field contains the IP Protocol to which the packet IPV4 Protocol field is compared (if IPV4 is detected), for identification of type no. 1 wake up event. This compare is made only if the register CompareMode bit equals to 1.
15:0	EtherType	RW 0x0	Ethernet Type This field contains the Ethernet type to which the packet Ethernet type is compared, for identification of type no. 1 wake up event. This compare is made only if the register CompareMode bit equals to 0.

Table 632: WOL Type2 (WOLT2) Register (i=0–3)

Offset: Port2: 0x000336A8, Port3: 0x000376A8, Port0: 0x000736A8, Port1: 0x000776A8

Offset Formula: 0x000736A8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:25	Reseved	RW 0x0	Reserved
24	CompareMode	RW 0x0	Type no. 2 compare mode. This bit defines the packet field that will compared in order to identify type no. 2 wake up event. 0 = Ethernet type: The packet Ethernet type is compared with the register EtherType field 1 = IP protocol: The IPV4 protocol field (if IPV4 is detected) is compared with the register IPProtocol field
23:16	IPProtocol	RW 0x0	IP Protocol This field contains the IP Protocol to which the packet IPV4 Protocol field is compared (if IPV4 is detected), for identification of type no. 2 wake up event. This compare is made only if the register CompareMode bit equals to 1.
15:0	EtherType	RW 0x0	Ethernet Type This field contains the Ethernet type to which the packet Ethernet type is compared, for identification of type no. 2 wake up event. This compare is made only if the register CompareMode bit equals to 0.

Table 633: WOL Type3 (WOLT3) Register (i=0–3)

Offset: Port2: 0x000336AC, Port3: 0x000376AC, Port0: 0x000736AC, Port1: 0x000776AC

Offset Formula: 0x000736AC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:25	Reseved	RW 0x0	Reserved
24	CompareMode	RW 0x0	Type no. 3 compare mode. This bit defines the packet field that will compared in order to identify type no. 3 wake up event. 0 = Ethernet type: The packet Ethernet type is compared with the register EtherType field 1 = IP protocol: The IPV4 protocol field (if IPV4 is detected) is compared with the register IPProtocol field

Copyright © 2014 Marvell

Table 633: WOL Type3 (WOLT3) Register (i=0–3) (Continued)

Offset: Port2: 0x000336AC, Port3: 0x000376AC, Port0: 0x000736AC, Port1: 0x000776AC

Offset Formula: 0x000736AC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
23:16	IPProtocol	RW 0x0	IP Protocol This field contains the IP Protocol to which the packet IPV4 Protocol field is compared (if IPV4 is detected), for identification of type no. 3 wake up event. This compare is made only if the register CompareMode bit equals to 1.
15:0	EtherType	RW 0x0	Ethernet Type This field contains the Ethernet type to which the packet Ethernet type is compared, for identification of type no. 3 wake up event. This compare is made only if the register CompareMode bit equals to 0.

Table 634: WOL Wakeup Frame Location (WOLWFL) Register (i=0–3)

Offset: Port2: 0x000336B0, Port3: 0x000376B0, Port0: 0x000736B0, Port1: 0x000776B0

Offset Formula: 0x000736B0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:19	Reserved	RO 0x0	Reserved
18:9	WakeupFrameOffset	RW 0x0	Wake up frame offset. It defines the offset of the first packet byte that is compared to the wake up frame. The offset is defined in units of words (2 bytes/word). The configured offset should be limited, so that the offset of the first packet byte that is compared to the wake up frame is less than 1400 bytes.
8:2	Reserved	RO 0x0	Reserved
1:0	WakeupFrameStartLocation	RW 0x0	Wakeup frame start location This field defines the starting location from where the packed data is compared to the wake up frame patterns. 0 = Start of packet: The starting compare point is WakeupFrameOffset words (2 bytes) after start of packet. 1 = Layer 3 header: The starting compare point is WakeupFrameOffset words (2 bytes) from start of layer 3 header. 2 = Layer 4 header: The starting compare point is WakeupFrameOffset words (2 bytes) from start of layer 4 header 3 = Reserved

Table 635: WOL Wakeup Frame Size (WOLWFS) Register (i=0–3)

Offset: Port2: 0x000336B4, Port3: 0x000376B4, Port0: 0x000736B4, Port1: 0x000776B4

Offset Formula: 0x000736B4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:24	Pattern3Size	RW 0x0	Wake up frame - pattern no. 3 size This field defines the number of bytes in wake frame pattern no. 3. **NOTE:** If in the WOL Wakeup Frame Location (WOLWFL) register, the WakeupFrameOffset and the WakeupframeStartLocation fields are configured to "0", then the minimum legal pattern size is 3 bytes.
23:16	Pattern2Size	RW 0x0	Wake up frame - pattern no. 2 size This field defines the number of bytes in wake frame pattern no. 2. **NOTE:** If in the WOL Wakeup Frame Location (WOLWFL) register, the WakeupFrameOffset and the WakeupframeStartLocation fields are configured to "0", then the minimum legal pattern size is 3 bytes.
15:8	Pattern1Size	RW 0x0	Wake up frame - pattern no. 1 size This field defines the number of byte in wake frame pattern no. 1. **NOTE:** If in the WOL Wakeup Frame Location (WOLWFL) register, the WakeupFrameOffset and the WakeupframeStartLocation fields are configured to "0", then the minimum legal pattern size is 3 bytes.
7:0	Pattern0Size	RW 0x0	Wake up frame - pattern no. 0 size This field defines the number of bytes in wake frame pattern no. 0. **NOTE:** If in the WOL Wakeup Frame Location (WOLWFL) register, the WakeupFrameOffset and the WakeupframeStartLocation fields are configured to "0", then the minimum legal pattern size is 3 bytes.

Table 636: WOL Wakeup Frame Select (WOLWFSEL) Register (i=0–3)

Offset: Port2: 0x000336B8, Port3: 0x000376B8, Port0: 0x000736B8, Port1: 0x000776B8

Offset Formula: 0x000736B8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:2	Reserved	RO 0x0	Reserved

Table 636: WOL Wakeup Frame Select (WOLWFSEL) Register (i=0–3) (Continued)

Offset: Port2: 0x000336B8, Port3: 0x000376B8, Port0: 0x000736B8, Port1: 0x000776B8

Offset Formula: 0x000736B8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
1:0	PatternSelect	RW 0x0	Pattern Select This field selects the pattern number that will be accessed by CPU via the WOL Wakeup Frame Data and WOL Wakeup Frame Mask registers.

Table 637: WOL Wakeup Frame Data (WOLWFD)<n> Register (i=0–3, n=0–31)

Offset: Port2pattern_dword_number0: 0x00033700, Port2pattern_dword_number1: 0x00033704...Port1pattern_dword_number31: 0x0007777C

Offset Formula: 0x00073700+4 * n + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-31) represents pattern_dword_number

Bit	Field	Type/InitVal	Description
CPU access to the wake-up frame pattern memory is allowed only if all the WakeupFrame* bits in the WOL Wakeup Event Enable (WOLWEE) register are cleared.			
31:24	Byte3	RW 0x0	Wake-up frame data byte This field contains a byte of the wake-up frame pattern, defined by the WOL Wakeup Frame Select register. The byte is byte number 4*n+3, where n is from this register offset formula.
23:16	Byte2	RW 0x0	Wake-up frame data byte This field contains a byte of the wake-up frame pattern, defined by the WOL Wakeup Frame Select register. The byte is byte number 4*n+2, where n is from this register offset formula.
15:8	Byte1	RW 0x0	Wake-up frame data byte This field contains a byte of the wake-up frame pattern, defined by the WOL Wakeup Frame Select register. The byte is byte number 4*n+1, where n is from this register offset formula.
7:0	Byte0	RW 0x0	Wake-up frame data byte This field contains a byte of the wake-up frame pattern, defined by the WOL Wakeup Frame Select register. The byte is byte number 4*n+0, where n is from this register offset formula.

Table 638: WOL Wakeup Frame Mask (WOLWFM)<n> Register (i=0–3, n=0–31)

Offset: Port2pattern_dword_no0: 0x00033780, Port2pattern_dword_no1: 0x00033784...Port1pattern_dword_no31: 0x000777FC

Offset Formula: 0x00073780+4 * n + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-31) represents pattern_dword_no

Bit	Field	Type/InitVal	Description
CPU access to the wake-up frame pattern memory is allowed only if all the WakeupFrame* bits in the WOL Wakeup Event Enable (WOLWEE) register are cleared.			
31:25	Reserved	RO 0x0	Reserved
24	Byte3Mask	RW 0x0	Byte3 Mask This bit defines if the byte of pattern pointed by the WOL Wakeup Frame Select register, located at offset 4*n +3, where n is from the register offset formula, is compared with the corresponding packet byte: 0 = No compare: The frame pattern byte is considered as don’t care. 1 = Compare: The frame pattern byte will be compared with the packet data.
23:17	Reserved	RO 0x0	Reserved
16	Byte2Mask	RW 0x0	Byte2 Mask This bit defines if the byte of pattern pointed by the WOL Wakeup Frame Select register, located at offset 4*n +2, where n is from the register offset formula, is compared with the corresponding packet byte: 0 = No compare: The frame pattern byte is considered as don’t care. 1 = Compare: The frame pattern byte will be compared with the packet data.
15:9	Reserved	RO 0x0	Reserved
8	Byte1Mask	RW 0x0	Byte1 Mask This bit defines if the byte of pattern pointed by the WOL Wakeup Frame Select register, located at offset 4*n +1, where n is from the register offset formula, is compared with the corresponding packet byte: 0 = No compare: The frame pattern byte is considered as don’t care. 1 = Compare: The frame pattern byte will be compared with the packet data.
7:1	Reserved	RO 0x0	Reserved

Table 638: WOL Wakeup Frame Mask (WOLWFM)<n> Register (i=0–3, n=0–31) (Continued)

Offset: Port2pattern_dword_no0: 0x00033780, Port2pattern_dword_no1: 0x00033784...Port1pattern_dword_no31: 0x000777FC

Offset Formula: 0x00073780+4 * n + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where n (0-31) represents pattern_dword_no

Bit	Field	Type/InitVal	Description
0	Byte0Mask	RW 0x0	Byte0 Mask This bit defines if the byte of pattern pointed by the WOL Wakeup Frame Select register, located at offset 4*n +0, where n is from the register offset formula, is compared with the corresponding packet byte: 0 = No compare: The frame pattern byte is considered as don’t care. 1 = Compare: The frame pattern byte will be compared with the packet data.

A.6.10 TX DMA Miscellaneous Registers

Table 639: Transmit Queue Command (TQC) Register (i=0–3)

Offset: Port2: 0x00030048, Port3: 0x00034048, Port0: 0x00070048, Port1: 0x00074048

Offset Formula: 0x00070048+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:8	DISQ	RW 0x0	Disable Queue One bit per each queue. Writing 1 disables the queue. The transmit DMA will stop the transmit process from this queue on next packet boundary. Writing 1 to the DISQ bit resets the matching ENQ bit (when the DMA is finished with the queue and if the current active queue has been disabled). Writing 0 to the DISQ bit has no effect. When transmit DMA encounters a descriptor with a parity error, the DMA will disable the queue but not set the DISQ bit for the queue. Thus reading DISQ and ENQ bits discriminates between queues disabled by the CPU and those stopped by DMA due to parity error on descriptor.

Table 639: Transmit Queue Command (TQC) Register (i=0–3) (Continued)

Offset: Port2: 0x00030048, Port3: 0x00034048, Port0: 0x00070048, Port1: 0x00074048

Offset Formula: 0x00070048+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
7:0	ENQ	RW 0x0	Enable Queue One bit per queue. Writing these bits set to 1 enables the queue. The Transmit DMA will fetch descriptors starting from the first descriptor from the address programmed in the PTXDQA register for that queue and start the Transmit process. Writing 1 to ENQ bit resets the matching <DISQ> bit. Writing 1 to ENQ bit of a DMA that is already in enable state, has no effect. Writing 0 to ENQ bit has no effect. When the receive DMA encounters a descriptor with a parity error, the DMA will clear the ENQ bit for that queue. Thus reading these bits reports the active enable status for each queue.

Table 640: TX Bad FCS Transmitted Packets Counter (TXBADFCS) Register (i=0–3)

Offset: Port2: 0x000318C0, Port3: 0x000358C0, Port0: 0x000718C0, Port1: 0x000758C0

Offset Formula: 0x000718C0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	BadFcsPacketsCounter	ROC 0x0	This counter counts the packets transmitted with bad FCS, due to any of the following events: - Error indications from DRAM controller during packet read - Underrun in the TX DMA FIFO

Table 641: TX Dropped Packets Counter (TXDROPPED) Register (i=0–3)

Offset: Port2: 0x000318C4, Port3: 0x000358C4, Port0: 0x000718C4, Port1: 0x000758C4

Offset Formula: 0x000718C4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	DroppedPacketsCounter	ROC 0x0	This counter counts the dropped packets by the TX controller, due to any of the following events: - Error indications from DRAM controller during descripror read

Table 642: Port Tx FIFO Threshold (PxTFTT) Register (i=0–3)

Offset: Port2: 0x00032478, Port3: 0x00036478, Port0: 0x00072478, Port1: 0x00076478

Offset Formula: 0x00072478+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7:0	Reserved	RW 0xF0	Reserved. Must write 0xF0

A.6.11 TX DMA Networking Controller Miscellaneous Registers

Table 643: Port TX Queues Descriptors Queue Address (PTXDQA)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031800, Port2Q1: 0x00031804...Port1Q7: 0x0007581C

Offset Formula: 0x00071800+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31:5	DescriptorsQueueStartingAddress	RW 0x0	Descriptors Queue Starting Address **NOTE:** This field is only valid when descriptor management by counters is used (as defined by the PxACC register). The field defines the starting address of the descriptors queue. The starting address is in units of 32 bytes. Therefore the descriptors queue is actually aligned to 32 bytes. The size of the TX descriptors is 16 bytes. In order to allow future expansion of the TX descriptor size to 32 bytes, the descriptors in the TX queue are located at locations aligned to 32 bytes.
4:0	Reserved	RO 0x0	Reserved

Table 644: Port TX Queues Descriptors Queue Size (PTXDQS)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031820, Port2Q1: 0x00031824...Port1Q7: 0x0007583C

Offset Formula: 0x00071820+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
NOTE: This register is only valid in Enhanced modes, as defined by the Port Acceleration Mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved

Table 644: Port TX Queues Descriptors Queue Size (PTXDQS)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2Q0: 0x00031820, Port2Q1: 0x00031824...Port1Q7: 0x0007583C

Offset Formula: 0x00071820+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3)

Bit	Field	Type/InitVal	Description
29:16	transmittedBuffersThreshold	RW 0x0	Transmitted Buffers Threshold When the number of buffers that were transmitted by the Network Controller, and not yet released by the CPU, as reflected by the <TransmittedBuffersCounter> field in the PTXS register, is larger than the value of this field, the relevant bit in the Tx <BufferThresholdCrossQueue> field of the Port TX Threshold Interrupt Cause (PTXThIC) register is set. NOTE: This field is valid only when descriptor management by counters is used (as defined by the PxACC register).
15:14	Reserved	RO 0x0	Reserved
13:0	DescriptorsQueueSize	RW 0x0	Descriptors Queue Size It contains the number of descriptors that can be stored in the descriptors queue. **NOTE:** In order to allow future increase of the TX descriptors to 32 bytes, the descriptors are located at locations aligned to 32 bytes. **NOTE:**

Table 645: Port TX Queues Status (PTXS)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031840, Port2Q1: 0x00031844...Port1Q7: 0x0007585C

Offset Formula: 0x00071840+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
NOTE: This register is only valid in Enhanced modes, as defined by the Port Acceleration Mode (PACC) register.			
31:30	Reserved	RO 0x0	Reserved
29:16	TransmittedBuffersCounter	RO 0x0	Transmitted Buffers Counter This field indicates the number of buffers that were transmitted and not yet released by the software. This field is incremented by Networking Controller following closing of the last descriptor of the packet, and decremented by CPU by writing to the NoOfReleasedBuffers field in the PTXSU register.
15:14	Reserved	RO 0x0	Reserved

Table 645: Port TX Queues Status (PTXS)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2Q0: 0x00031840, Port2Q1: 0x00031844...Port1Q7: 0x0007585C

Offset Formula: 0x00071840+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
13:0	PendingDescriptors Counter	RO 0x0	Pending Descriptors Counter This field indicates the number of descriptors that point to data to be transmitted. This number is incremented by CPU, by writing to the NoOfWrittenDescriptors field in the PTXSU register. The Networking Controller decrements this field, following closing of the last descriptor in packet. The decrement is by the number of descriptors in this packet.

Table 646: Port Tx Queues Status Update (PTXSU)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031860, Port2Q1: 0x00031864...Port1Q7: 0x0007587C

Offset Formula: 0x00071860+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
NOTE: This register is only valid in Enhanced modes, as defined by the Port Acceleration Mode (PACC) register.			
31:24	Reserved	RO 0x0	Reserved
23:16	NoOfReleasedBuffers	WO 0x0	Number Of Released Buffers To this field, the CPU writes the number of buffers that were released. The TransmittedBuffersCounter field, in the PTXS register is decremented by this number.
15:8	Reserved	RO 0x0	Reserved
7:0	NoOfWrittenDescriptors	WO 0x0	Number Of Written Descriptors To this field the CPU writes the number of new descriptors prepared by CPU for packets transmission.

Table 647: Port TX Queues Descriptor Index (PTXDI)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031880, Port2Q1: 0x00031884...Port1Q7: 0x0007589C

Offset Formula: 0x00071880+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
NOTE: This register is only valid in Enhanced modes, as defined by the Port Acceleration Mode (PACC) register.			
31:14	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 647: Port TX Queues Descriptor Index (PTXDI)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2Q0: 0x00031880, Port2Q1: 0x00031884...Port1Q7: 0x0007589C

Offset Formula: 0x00071880+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
13:0	NextDescriptorIndex	RO 0x0	Next Descriptor Index This field indicates the index of the next descriptor that is going to be fetched. The valid values are 0 to <DescriptorsQueueSize>- 1.

Table 648: TX Transmitted Buffers Counter (TXTBC)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x000318A0, Port2Q1: 0x000318A4...Port1Q7: 0x000758BC

Offset Formula: 0x000718A0+4*q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:30	Reserved	RO 0x0	Reserved
29:16	TransmittedBuffersCounter	ROC 0x0	Transmitted Buffers Counter This field indicates the number of buffers that were transmitted and not yet released by SW to the operating system. This field is incremented by Networking Controller following transmission of the packet (and descriptor closing), and decremented by CPU by writing to the NoOfReleasedBuffers field in the PxTXySU register. Following the read of this field, the counter is cleared. The data read from this field is identical with the data read from the TransmittedBuffersCounter field of the Port TX queues status (PTXS) register.
15:0	Reserved	RO 0x0	Reserved

Table 649: Port TX Initialization (PTXINIT) Register (i=0–3)

Offset: Port2: 0x000318F0, Port3: 0x000358F0, Port0: 0x000718F0, Port1: 0x000758F0

Offset Formula: 0x000718F0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is only valid in Enhanced modes, as defined by the Port Acceleration Mode (PACC) register.			
31:1	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 649: Port TX Initialization (PTXINIT) Register (i=0–3) (Continued)

Offset: Port2: 0x000318F0, Port3: 0x000358F0, Port0: 0x000718F0, Port1: 0x000758F0

Offset Formula: 0x000718F0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
0	TxDMAInit	RW 0x0	TX DMA Init This bit when set to "1" it initializes the TX DMA engines. That includes clearing of the ENQ bits, Initialization of the Descriptor counters etc.

A.6.12 TX DMA Packet Modification Registers

Table 650: TX Marvell Header Reg1 (TX_MH_reg1) Register (i=0–3)

Offset: Port2: 0x00031944, Port3: 0x00035944, Port0: 0x00071944, Port1: 0x00075944

Offset Formula: 0x00071944+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 651: TX Marvell Header Reg2 (TX_MH_reg2) Register (i=0–3)

Offset: Port2: 0x00031948, Port3: 0x00035948, Port0: 0x00071948, Port1: 0x00075948

Offset Formula: 0x00071948+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved

Table 651: TX Marvell Header Reg2 (TX_MH_reg2) Register (i=0–3) (Continued)

Offset: Port2: 0x00031948, Port3: 0x00035948, Port0: 0x00071948, Port1: 0x00075948

Offset Formula: 0x00071948+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 652: TX Marvell Header Reg3 (TX_MH_reg3) Register (i=0–3)

Offset: Port2: 0x0003194C, Port3: 0x0003594C, Port0: 0x0007194C, Port1: 0x0007594C

Offset Formula: 0x0007194C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 653: TX Marvell Header Reg4 (TX_MH_reg4) Register (i=0–3)

Offset: Port2: 0x00031950, Port3: 0x00035950, Port0: 0x00071950, Port1: 0x00075950

Offset Formula: 0x00071950+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RW 0x0	Reserved

Table 653: TX Marvell Header Reg4 (TX_MH_reg4) Register (i=0–3) (Continued)

Offset: Port2: 0x00031950, Port3: 0x00035950, Port0: 0x00071950, Port1: 0x00075950

Offset Formula: 0x00071950+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	Marvell Header	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 654: TX Marvell Header Reg5 (TX_MH_reg5) Register (i=0–3)

Offset: Port2: 0x00031954, Port3: 0x00035954, Port0: 0x00071954, Port1: 0x00075954

Offset Formula: 0x00071954+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 655: TX Marvell Header Reg6 (TX_MH_reg6) Register (i=0–3)

Offset: Port2: 0x00031958, Port3: 0x00035958, Port0: 0x00071958, Port1: 0x00075958

Offset Formula: 0x00071958+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RW 0x0	Reserved

Table 655: TX Marvell Header Reg6 (TX_MH_reg6) Register (i=0–3) (Continued)

Offset: Port2: 0x00031958, Port3: 0x00035958, Port0: 0x00071958, Port1: 0x00075958

Offset Formula: 0x00071958+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 656: TX Marvell Header Reg7 (TX_MH_reg7) Register (i=0–3)

Offset: Port2: 0x0003195C, Port3: 0x0003595C, Port0: 0x0007195C, Port1: 0x0007595C

Offset Formula: 0x0007195C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 657: TX Marvell Header Reg8 (TX_MH_reg8) Register (i=0–3)

Offset: Port2: 0x00031960, Port3: 0x00035960, Port0: 0x00071960, Port1: 0x00075960

Offset Formula: 0x00071960+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved

Table 657: TX Marvell Header Reg8 (TX_MH_reg8) Register (i=0–3) (Continued)

Offset: Port2: 0x00031960, Port3: 0x00035960, Port0: 0x00071960, Port1: 0x00075960

Offset Formula: 0x00071960+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 658: TX Marvell Header Reg9 (TX_MH_reg9) Register (i=0–3)

Offset: Port2: 0x00031964, Port3: 0x00035964, Port0: 0x00071964, Port1: 0x00075964

Offset Formula: 0x00071964+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 659: TX Marvell Header Reg10 (TX_MH_reg10) Register (i=0–3)

Offset: Port2: 0x00031968, Port3: 0x00035968, Port0: 0x00071968, Port1: 0x00075968

Offset Formula: 0x00071968+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RW 0x0	Reserved

Table 659: TX Marvell Header Reg10 (TX_MH_reg10) Register (i=0–3) (Continued)

Offset: Port2: 0x00031968, Port3: 0x00035968, Port0: 0x00071968, Port1: 0x00075968

Offset Formula: 0x00071968+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	Marvell Header	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 660: TX Marvell Header Reg11 (TX_MH_reg11) Register (i=0–3)

Offset: Port2: 0x0003196C, Port3: 0x0003596C, Port0: 0x0007196C, Port1: 0x0007596C

Offset Formula: 0x0007196C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 661: TX Marvell Header Reg12 (TX_MH_reg12) Register (i=0–3)

Offset: Port2: 0x00031970, Port3: 0x00035970, Port0: 0x00071970, Port1: 0x00075970

Offset Formula: 0x00071970+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RW 0x0	Reserved

Table 661: TX Marvell Header Reg12 (TX_MH_reg12) Register (i=0–3) (Continued)

Offset: Port2: 0x00031970, Port3: 0x00035970, Port0: 0x00071970, Port1: 0x00075970

Offset Formula: 0x00071970+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 662: TX Marvell Header Reg13 (TX_MH_reg13) Register (i=0–3)

Offset: Port2: 0x00031974, Port3: 0x00035974, Port0: 0x00071974, Port1: 0x00075974

Offset Formula: 0x00071974+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RW 0x0	Reserved
15:0	Marvell Header	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 663: TX Marvell Header Reg14 (TX_MH_reg14) Register (i=0–3)

Offset: Port2: 0x00031978, Port3: 0x00035978, Port0: 0x00071978, Port1: 0x00075978

Offset Formula: 0x00071978+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RO 0x0	Reserved

Table 663: TX Marvell Header Reg14 (TX_MH_reg14) Register (i=0–3) (Continued)

Offset: Port2: 0x00031978, Port3: 0x00035978, Port0: 0x00071978, Port1: 0x00075978

Offset Formula: 0x00071978+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 664: TX Marvell Header Reg15 (TX_MH_reg15) Register (i=0–3)

Offset: Port2: 0x0003197C, Port3: 0x0003597C, Port0: 0x0007197C, Port1: 0x0007597C

Offset Formula: 0x0007197C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register.			
31:16	Reserved	RW 0x0	Reserved
15:0	MarvellHeader	RW 0x0	Marvell Header This field contains the Marvell header that should used as packet header (instead of the present packet header). **NOTE:** This field is used only if the MHEn field in the legacy Marvell Header register is set, and if selected by the MH-select field of the TX descriptor. **NOTE:**

Table 665: TX MTU (TX_MTU) Register (i=0–3)

Offset: Port2: 0x00031988, Port3: 0x00035988, Port0: 0x00071988, Port1: 0x00075988

Offset Formula: 0x00071988+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved

Table 665: TX MTU (TX_MTU) Register (i=0–3) (Continued)

Offset: Port2: 0x00031988, Port3: 0x00035988, Port0: 0x00071988, Port1: 0x00075988

Offset Formula: 0x00071988+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:0	MaximumTransmissionUnit	RW 0xffff	Maximum Transmission Unit This field defines the maximum allowed number of bytes in transmitted packet. If the TX DMA will attempt to transmit a packet larger then MTU then the packet will be dropped (by generating a bad FCS). Note: The value configured in this field should be an even number. **NOTE:** The minimum legal value is 256.

A.6.13 TX DMA Queues Arbiter Registers

Table 666: Transmit Queue Fixed Priority Configuration - Arbiter ver1 (TQFPC_AV1) Register (i=0–3)

Offset: Port2: 0x000300DC, Port3: 0x000340DC, Port0: 0x000700DC, Port1: 0x000740DC

Offset Formula: 0x000700DC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7:0	FIXPR	RW 0xFF	Fixed priority queue [7:0] Setting this bit to 1 configures the transmit queue as fixed priority. When the value is 0 value the queue is subject to the WRR algorithm.

Table 667: Port Transmit Token-Bucket Rate Configuration - Arbiter ver1 (PTTBRC_AV1) Register (i=0–3)

Offset: Port2: 0x000300E0, Port3: 0x000340E0, Port0: 0x000700E0, Port1: 0x000740E0

Offset Formula: 0x000700E0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:10	Reserved	RO 0x0	Reserved

Table 667: Port Transmit Token-Bucket Rate Configuration - Arbiter ver1 (PTTBRC_AV1) Register (i=0–3) (Continued)

Offset: Port2: 0x000300E0, Port3: 0x000340E0, Port0: 0x000700E0, Port1: 0x000740E0

Offset Formula: 0x000700E0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
9:0	PTKNRT	RW 0x3FF	Port Token Rate A configurable value of 0-1023, in units of 1/64 byte. This value is added to queues Token-Bucket every 8 system clock cycles. The port-Token-Bucket is used to limit the port transmit bandwidth, see "Weighted Round-Robin Priority Mode" and "Transmit Queue Bandwidth Limitation". **NOTE:** The initial value actually means that there is no bandwidth limitation on the port. **NOTE:**

Table 668: Maximum Transmit Unit - Arbiter ver1 (MTU_AV1) Register (i=0–3)

Offset: Port2: 0x000300E8, Port3: 0x000340E8, Port0: 0x000700E8, Port1: 0x000740E8

Offset Formula: 0x000700E8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:6	Reserved	RO 0x0	Reserved
5:0	MTU	RW 0x24	Maximum Transmit Unit 9 KB/256 = 36 (=0x24) Configurable value in 256-byte units, up to 16 KB, used to implement the Token-Bucket bandwidth limitation. For port/queue where the value in field <TKNBKT> in the Transmit Queue Token-Bucket Counter - Arbiter ver1 (TQxTBC_AV1) Register is greater than <MTU>, the port/queue bandwidth limitation is OFF (transmit enabled).

Table 669: Port Maximum Token Bucket Size - Arbiter ver1 (PMTBS_AV1) Register (i=0–3)

Offset: Port2: 0x000300EC, Port3: 0x000340EC, Port0: 0x000700EC, Port1: 0x000740EC

Offset Formula: 0x000700EC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved

Table 669: Port Maximum Token Bucket Size - Arbiter ver1 (PMTBS_AV1) Register (i=0–3) (Continued)

Offset: Port2: 0x000300EC, Port3: 0x000340EC, Port0: 0x000700EC, Port1: 0x000740EC

Offset Formula: 0x000700EC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3)

Bit	Field	Type/InitVal	Description
15:0	PMTBS	RW 0xFFFF	Port Maximum Token Bucket Size Configurable value, in 256-byte units, used to implement the port Token-Bucket bandwidth limitation. The port token bucket is incremented by the values of field <PTKNRT> (in the Port Transmit Token-Bucket Rate Configuration - Arbiter ver1 (PTTBRC_AV1) Register) up to a value of PMTBS*256*64. (Maximum accumulated transmit credit.)

Table 670: Transmit Queue Token-Bucket Counter - Arbiter ver1 (TQxTBC_AV1)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00030300, Port2Q1: 0x00030310...Port1Q7: 0x00074370

Offset Formula: 0x00070300+q*16 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31:30	Reserved	RO 0x0	Read only
29:0	TKNBKT	RW 0x0	Token Bucket Byte counter, in units of 1/64 byte, incremented by the token rate and decremented by the transmitted bytes from this queue x 64. The CPU may write to this counter to override with Token-bucket operation.

Table 671: Transmit Queue Token Bucket Configuration - Arbiter ver1 (TQxTBCFG_AV1)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00030304, Port2Q1: 0x00030314...Port1Q7: 0x00074374

Offset Formula: 0x00070304+q*16 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31:26	Reserved	RO 0x0	Read only
25:10	MTBS	RW 0x0	Maximum Token Bucket Size Configurable value in 256-byte units, used to implement the Token-Bucket bandwidth limitation. The token bucket is incremented by token rate up to MTBS*256*64. (Maximum accumulated transmit credit.)

Table 671: Transmit Queue Token Bucket Configuration - Arbiter ver1 (TQxTBCFG_AV1)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2Q0: 0x00030304, Port2Q1: 0x00030314...Port1Q7: 0x00074374

Offset Formula: 0x00070304+q*16 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
9:0	TKNRT	RW 0x0	Token Rate Legal values are between 0-1023. The Token Rate, in units of 1/64 byte, is added to queues Token-Bucket every 8 system clock cycles. The Token-Bucket is used to limit the transmit bandwidth per queue.

Table 672: Transmit Queue Arbiter Configuration - Arbiter ver1 (TQxAC_AV1)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00030308, Port2Q1: 0x00030318...Port1Q7: 0x00074378

Offset Formula: 0x00070308+q*16 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31:25	Reserved	RO 0x0	Reserved
24:8	WRR_BC[16:0]	RW 0x0	Internal counter for implementing the WRR algorithm The CPU should not modify this field after writing the initialization value and enabling the port.
7:0	WRRWGT	RW 0x0	Transmit Weighted-Round-Robin Weight The configurable value of WRR weight is 256 bytes per unit, range 0-64 Kbyte, in 256-bytes increments. The transmit bandwidth assigned to a queue is equal to WRRWGT/(Sum of all "non fixed priority" plus "maximum limited" WRRWGTs) part of remaining bandwidth (not consumed by the fixed priority).

Table 673: Port Transmit Token-Bucket Counter - Arbiter ver1 (PTTBC_AV1) Register (i=0–3)

Offset: Port2: 0x00030380, Port3: 0x00034380, Port0: 0x00070380, Port1: 0x00074380

Offset Formula: 0x00070380+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:30	Reserved	RO 0x0	Read only

Table 673: Port Transmit Token-Bucket Counter - Arbiter ver1 (PTTBC_AV1) Register (i=0–3) (Continued)

Offset: Port2: 0x00030380, Port3: 0x00034380, Port0: 0x00070380, Port1: 0x00074380

Offset Formula: 0x00070380+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3)

Bit	Field	Type/InitVal	Description
29:0	PTKNBKT	RW 0x3FFFFFFF	Port Token Bucket Byte counter, in units of 1/64 byte, incremented by the field <PTKNRT> in the Port Transmit Token-Bucket Rate Configuration (PTTBRC) Register (up to a value of PMTBS*256*64) and decremented by transmitted bytes from this port x 64. The CPU may write to this counter to override with Token-bucket operation. **NOTE:** The initial value is set to its maximum value so the port can start transmission immediately. **NOTE:**

Table 674: Transmit Queue Command1 - Arbiter ver3 (TQC1_AV3) Register (i=0–3)

Offset: Port2: 0x00031A00, Port3: 0x00035A00, Port0: 0x00071A00, Port1: 0x00075A00

Offset Formula: 0x00071A00+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:4	Reserved	RO 0x0	Reserved
3	BW_Lim_Sel	RW 0x1	TX queues bandwidth limitation/arbiter mechanism select **NOTE:** this bit must be set to 0, before any other configuration of the TX queues Bandwidth limitation/arbiter mechanism version 3 registers. 0 = Version 3: TX queues bandwidth limitation/arbiter mechanism version 3 1 = Version 1: TX queues bandwidth limitation/arbiter mechanism version 1
2	EJP_ENB	RW 0x0	This bit enables the EJP mechanism: 0 = Fixed priority/WRR 1 = EJP
1	PTP_SYNC_ENB	RW 0x0	PTP Sync Enable This bit defines the port and queue refill periods as described in the description of the PortRefillPeriod field in the PRefill_AV3 register and in the description of the QueueRefillPeriod field in the QRefill_AV3 register. <F_PP_no_PTP>Reserved. Must write 0x0.</F_PP_no_PTP>
1	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 674: Transmit Queue Command1 - Arbiter ver3 (TQC1_AV3) Register (i=0–3) (Continued)

Offset: Port2: 0x00031A00, Port3: 0x00035A00, Port0: 0x00071A00, Port1: 0x00075A00

Offset Formula: 0x00071A00+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
0	WRR_EJP_INIT	RW 0x1	This bit, when set to "1", it acts as a software reset to the module. When this bit is set to "1", it initializes the WRR/EJP module: - The WRR_BYTE_COUNT variable of all queues is cleared to 0x0. - The TIME_FROM_LAST_PKT_TX variable is initialized to 0x3FFFF. - Queue token bucket update is disabled. The token buckets should be initialized in this state.

Table 675: Transmit Queue Fixed Priority Configuration - Arbiter ver3 (TQFPC_AV3) Register (i=0–3)

Offset: Port2: 0x00031A04, Port3: 0x00035A04, Port0: 0x00071A04, Port1: 0x00075A04

Offset Formula: 0x00071A04+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7:0	FIXPR	RW 0xFF	Fixed priority queue [7:0] Setting this bit to 1 configures the transmit queue as fixed priority. A 0 value means the queue is subject to the WRR algorithm.

Table 676: Basic Refill No of Clocks - Arbiter ver3 (BRC_AV3) Register (i=0–3)

Offset: Port2: 0x00031A08, Port3: 0x00035A08, Port0: 0x00071A08, Port1: 0x00075A08

Offset Formula: 0x00071A08+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:0	BasicRefillNoOfClocks	RW 0x40	Basic Refill number of Clocks This field defines the basic period used for the rate limitation tocken buckets update. For the actual token bucket update see the PortTokenRefillValue field of the Prefill_AV3 register and the QueueTokenRefillValue field of the Qrefill_AV3 register. **NOTE:** The minimul legal value is 16. **NOTE:**

Table 677: Maximum Transmit Unit - Arbiter ver3 (MTU_AV3) Register (i=0–3)

Offset: Port2: 0x00031A0C, Port3: 0x00035A0C, Port0: 0x00071A0C, Port1: 0x00075A0C

Offset Formula: 0x00071A0C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:19	Reserved	RO 0x0	Reserved
18:0	MTU	RW 0x12000	Maximum Transmit Unit This field defines the minimum number of tokens in both port and queue token buckets, in order to allow packet transmission. **NOTE:** One token represents 1 packet bit

Table 678: Port Bucket Refill - Arbiter ver3 (PRefill_AV3) Register (i=0–3)

Offset: Port2: 0x00031A10, Port3: 0x00035A10, Port0: 0x00071A10, Port1: 0x00075A10

Offset Formula: 0x00071A10+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:30	Reserved	RO 0x0	Reserved
29:20	PortRefillPeriod	RW 0x1	Port Refill Period This field defines the interval at which the port token bucket is incremented by the value of the PortTokenRefillValue field. The refill interval is as following: If clock signal generated by the PTP module is selected (PTP_SYNC_ENB == 0x1 and EJP_ENB == 0x1): PortRefillPeriod - in units of PTP clock period. If TCLK (core clock) is selected (PTP_SYNC_ENB == 0x0 or EJP_ENB == 0x0): BasicRefillNoOfClocks * PortRefillPeriod, in units of core clock period Therefore, if the core clock is selected (PTP_SYNC_ENB == 0x0 or EJP_ENB == 0x0), the port refill period (in units of usec) is: port_refill_period = BasicRefillNoOfClocks * PortRefillPeriod * (1/C) where: C - is the TCLK (core clock) frequency in MHz **NOTE:** One token is defined as one packet bit. The value of this field should be > 0.
19	Reserved	RO 0x0	Reserved

Table 678: Port Bucket Refill - Arbiter ver3 (PRefill_AV3) Register (i=0–3) (Continued)

Offset: Port2: 0x00031A10, Port3: 0x00035A10, Port0: 0x00071A10, Port1: 0x00075A10

Offset Formula: 0x00071A10+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
18:0	PortTokenRefillValue	RW 0x3ff	Port Token Refill Value This field defines how many tokens will be added to the port token bucket at an interval as defined by the PortRefillPeriod field. Therefore the equation of the port token accumulation rate in units of token/usec is: PortTokenRefillValue/port_refill_period where: port_refill_period is the port refill period as defined in the description of the PortRefillPeriod field **NOTE:** One token is defined as one packet bit. The maximum value of this field should be as follows: PortBucketSize + PortTokenRefillValue <= 0xFFFFFFFF **NOTE:** The value of this field should be > 0. **NOTE:**

Table 679: Port Maximum Token Bucket Size - Arbiter ver3 (PMTBS_AV3) Register (i=0–3)

Offset: Port2: 0x00031A14, Port3: 0x00035A14, Port0: 0x00071A14, Port1: 0x00075A14

Offset Formula: 0x00071A14+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	PortBucketSize	RW 0x7FFF800	Port Token Bucket Size This field defines the size of the port token bucket. **NOTE:** One token is defined as one packet bit. The maximum value of this field should be as follows: PortBucketSize + PortTokenRefillValue <= 0xFFFFFFFF The value of this field should be > MTU.

Table 680: Port Transmit Token-Bucket Counter - Arbiter ver3 (PTTBC_AV3) Register (i=0–3)

Offset: Port2: 0x00031A18, Port3: 0x00035A18, Port0: 0x00071A18, Port1: 0x00075A18

Offset Formula: 0x00071A18+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	PTKNBKT	RW 0x7FFFFFF	Port token Bucket. This field represents the number of tokens in the port token bucket. **NOTE:** 1 token is defined as 1 packet bit The byte counter is incremented by PortTokenRefillValue field of the PRefill_AV3 register, at an interval as defined by the PortRefillPeriod field of the PRefill_AV3 register. The CPU may write to this counter to override with Token-bucket operation.

Table 681: Queue Bucket Refill - Arbiter ver3 (QRefill_AV3)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031A20, Port2Q1: 0x00031A24...Port1Q7: 0x00075A3C

Offset Formula: 0x00071A20+4*q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31:30	Reserved	RO 0x0	Reserved
29:20	QueueRefillPeriod	RW 0x1	Queue Refill Period This field defines the interval at which the queue token bucket is incremented by the value of the QueueTokenRefillValue field. The refill interval is as following: If the clock signal generated by the PTP module is selected (PTP_SYNC_ENB == 0x1 and EJP_ENB == 0x1): QueueRefillPeriod - in units of PTP clock period. If TCLK (core clock) is selected (PTP_SYNC_ENB == 0x0 or EJP_ENB == 0x0): BasicRefillNoOfClocks * QueueRefillPeriod, in units of core clock period. Therefore, if core clock is selected (PTP_SYNC_ENB == 0x0 or EJP_ENB == 0x0), the queue refill period (in units of usec) is: queue_refill_period = BasicRefillNoOfClocks * QueueRefillPeriod * (1/C) where: C - is the TCLK (core clock) frequency in MHz **NOTE:** One token is defined as one packet bit. The value of this field should be > 0.
19	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 681: Queue Bucket Refill - Arbiter ver3 (QRefill_AV3)<q> Register (i=0–3, q=0–7) (Continued)

Offset: Port2Q0: 0x00031A20, Port2Q1: 0x00031A24...Port1Q7: 0x00075A3C

Offset Formula: 0x00071A20+4*q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
18:0	QueueTokenRefillValue	RW 0x3FF	Queue Token Refill Value This field defines how many tokens will be added to the queue token bucket at an interval as defined by the QueueRefillPeriod field. Therefore the equation of the queue token accumulation rate in units of token/usec is: QueueTokenRefillValue/queue_refill_period where: queue_refill_period is the queue refill period as defined in the description of the QueueRefillPeriod field **NOTE:** One token is defined as one packet bit. The maximum value of this field should be as follows: QueueBucketSize + QueueTokenRefillValue <= 0x7FFFFFFF The value of this field should be > 0.

Table 682: Queue Maximum Token Bucket Size - Arbiter ver3 (QMTBS_AV3)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031A40, Port2Q1: 0x00031A44...Port1Q7: 0x00075A5C

Offset Formula: 0x00071A40+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31	Reserved	RO 0x0	Reserved
30:0	QueueBucketSize	RW 0x7FFF800	Queue Token Bucket Size This field defines the size of the queue token bucket. **NOTE:** One token is defined as one packet bit. The maximum value of this field should be as follows: QueueBucketSize + QueueTokenRefillValue <= 0x7FFFFFFF The value of this field should be > MTU.

Table 683: Queue Transmit Token-Bucket Counter - Arbiter ver3 (QxTTBC_AV3)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031A60, Port2Q1: 0x00031A64...Port1Q7: 0x00075A7C

Offset Formula: 0x00071A60+q*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31:0	QTKNBKT	RW 0x0	Queue token Bucket. This field represents the number of tokens in the queue token bucket. Note: 1 token is defined as 1 packet bit The byte counter is incremented by QueueTokenRefillValue field of the QRefill_AV3 register, at an interval as defined by the QueueRefillPeriod field of the QRefill_AV3 register. This field is in two’s complement format. That means that the Queue Token bucket may contain a negative number of tokens. **NOTE:** The CPU may write to this counter to initialize or update the Queue Token-bucket.. It is recommended that the CPU will write to the token bucket, only when the EJP_INIT bit is set. **NOTE:**

Table 684: Transmit Queue Arbiter Configuration - Arbiter ver3 (TQxAC_AV3)<q> Register (i=0–3, q=0–7)

Offset: Port2Q0: 0x00031A80, Port2Q1: 0x00031A84...Port1Q7: 0x00075A9C

Offset Formula: 0x00071A80+q*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents Q

Bit	Field	Type/InitVal	Description
31:26	Reserved	RO 0x0	Reserved
25:8	WRR_BC[17:0]	RO 0x0	Internal counter for implementing the WRR algorithm. The data format is as following: Bits[15:0]: byte_count % WRRWGT Bits[17:16]: int (byte_count / WRRWGT). This field is initialized to 0x0, when the EJP_INIT is set to 0x1.
7:0	WRRWGT	RW 0x0	Transmit Weighted-Round-Robin Weight The configurable value of WRR weight is 256 bytes per unit, spans 0-64 Kbyte in 256-bytes increments. The transmit bandwidth assigned to a queue is equal to WRRWGT/(Sum of all "non fixed priority" plus "maximum limited" WRRWGTs) part of remaining bandwidth (not consumed by fixed priority).

Table 685: Transmission Queue IPG - Arbiter ver3 (TQxIPG_AV3)<q> Register (i=0–3, q=2–3)

Offset: Port2Q2: 0x00031AA8, Port2Q3: 0x00031AAC, Port3Q2: 0x00035AA8, Port3Q3: 0x00035AAC, Port0Q2: 0x00071AA8, Port0Q3: 0x00071AAC, Port1Q2: 0x00075AA8, Port1Q3: 0x00075AAC

Offset Formula: 0x00071AA0+q*4 + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (2-3) represents Q

Bit	Field	Type/InitVal	Description
31:14	Reserved	RO 0x0	Read only
13:0	IPG	RW 0x0	IPG is the minimum required gap from the end of a packet to the beginning of the next packet in the same queue. This field is applicable only for IsoLo(queue#2) and IsoHi(queue#3). The gap is measured in units of TCLK (core clock) cycles. The equation of the IPG is as following: IPG = GAP/(1/C) Where: GAP - is the minimum required gap in usec. C - is the TCLK (core clock) frequency in MHz IPG in units of system clock cycles. This field is used for TX queue selection in EJP mode, for queues 2,3.. **NOTE:** The required GAP for a specific rate can be calculated with the equation: GAP = 8 * PKT_SIZE * (1/MRATE - 1/T) where: PKT_SIZE - The number of bytes in the shortest packet MRATE - Maximum allowed line rate for packets in this queue in Mbit/sec T - line data rate in Mbit/sec

Table 686: High Token in Low Packet - Arbiter ver3 (HITKNinLOPKT_AV3) Register (i=0–3)

Offset: Port2: 0x00031AB0, Port3: 0x00035AB0, Port0: 0x00071AB0, Port1: 0x00075AB0

Offset Formula: 0x00071AB0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:20	Reserved	RO 0x0	Reserved

Table 686: High Token in Low Packet - Arbiter ver3 (HITKNinLOPKT_AV3) Register (i=0–3) (Continued)

Offset: Port2: 0x00031AB0, Port3: 0x00035AB0, Port0: 0x00071AB0, Port1: 0x00075AB0

Offset Formula: 0x00071AB0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3)

Bit	Field	Type/InitVal	Description
19:0	HiTKNinLoPkt	RW 0x0	Number of tokens that are accumulated in the IsoHi (queue#3) token bucket, during transmission of the longest IsoLo (queue#2) packet. This field is used for TX queue selection in EJP mode. The equation of this field is as following: HiTKNinLoPkt = ((8 * L)/T) * queue_token_accumulation_rate where: L - is the longest expected packet in the IsoLo (queue#2) queue T - line data rate in Mbit/sec queue_token_accumulation_rate - the queue token accumulation rate of the IsoHi (queue#3) queue, as defined in the QueueTokenRefillValue field, of the Qrefill_AV3 register **NOTE:** The maximum value of this field should be so that: QueueBucketSize(of queue#3) + HiTKNinLoPkt <= 0x7FFFFFFF

Table 687: High Token in Asynchronous Packet - Arbiter ver3 (HITKNinASYNCPKT_AV3) Register (i=0–3)

Offset: Port2: 0x00031AB4, Port3: 0x00035AB4, Port0: 0x00071AB4, Port1: 0x00075AB4

Offset Formula: 0x00071AB4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:20	Reserved	RO 0x0	Reserved
19:0	HiTKNinAsyncPkt	RW 0x0	Number of tokens that are accumulated in the IsoHi (queue#3) token bucket, during transmission of the longest Async (queue#1 & #0) packet. This field is used for TX queue selection in EJP mode. The equation of this field is as following: HiTKNinAsyncPkt = ((8 * L)/T) * queue_token_accumulation_rate where: L - is the longest expected packet in the Async (queue#0 & #1) queue T - line data rate in Mbit/sec queue_token_accumulation_rate - the queue token accumulation rate of the IsoHi (queue#3) queue, as defined in the QueueTokenRefillValue field, of the Qrefill_AV3 register **NOTE:** The maximum value of this field should be so that: QueueBucketSize(of queue#3) + HiTKNinAsyncPkt <= 0x7FFFFFFF

Table 688: Low Token in Asynchronous Packet - Arbiter ver3 (LOTKNinASYNCPKT_AV3) Register (i=0–3)

Offset: Port2: 0x00031AB8, Port3: 0x00035AB8, Port0: 0x00071AB8, Port1: 0x00075AB8

Offset Formula: 0x00071AB8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:20	Reserved	RO 0x0	Reserved
19:0	LoTKNinAsyncPkt	RW 0x0	Number of tokens that are accumulated in the IsoLo (queue#2) token bucket, during transmission of the longest Async (queue#1 & #0) packet. This field is used for TX queue selection in EJP mode. The equation of this field is as following: LoTKNinAsyncPkt = ((8 * L)/T) * queue_token_accumulation_rate where: L - is the longest expected packet in the Async (queue#0 & #1) queue T - line data rate in Mbit/sec queue_token_accumulation_rate - the queue token accumulation rate of the IsoLo (queue#2) queue, as defined in the QueueTokenRefillValue field, of the Qrefill_AV3 register **NOTE:** The maximum value of this field should be so that: QueueBucketSize(of queue#2) + LoTKNinAsyncPkt <= 0x7FFFFFFF

Table 689: Transmission Speed - Arbiter ver3 (TS_AV3) Register (i=0–3)

Offset: Port2: 0x00031ABC, Port3: 0x00035ABC, Port0: 0x00071ABC, Port1: 0x00075ABC

Offset Formula: 0x00071ABC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:15	Reserved	RO 0x0	Reserved
14:0	TS	RW 0x0	Ethernet Transmission Speed in units of system clocks/1024 bytes This field is used in EJP mode for packets duration calculation. The equation of TS is as following: TS = (1024 * 8 * 1/T) / (1/C) where: C - is the TCLK (core clock) frequency in MHz T - line data rate in Mbit/sec

A.6.14 RX_TX DMA Registers

Table 690: Port Configuration (PxC) Register (i=0–3)

Offset: Port2: 0x00032400, Port3: 0x00036400, Port0: 0x00072400, Port1: 0x00076400

Offset Formula: 0x00072400+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:26	Reserved	RO 0x0	Reserved
25	RxCS	RW 0x1	Rx TCP/UDP checksum mode 0 = No header: Calculate without pseudo header. 1 = With header: Calculation include pseudo header.
24:22	BPDUQ	RW 0x7	Captured BPDU frames (if the Port Configuration Extended register <Span> field is set) are directed to this Queue number. **NOTE:** This field is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register.
21:19	UDPQ	RW 0x0	Captured UDP frames are directed to this Queue number. **NOTE:** This field is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register.
18:16	TCPQ	RW 0x0	Captured TCP frames are directed to this Queue number. **NOTE:** This field is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.
15	UDP_ CapEn/ChecksumDrop	RW 0x0	Capture UDP frames to <UDPQ> / Drop on Checksum Error In Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register, this field controls the capture of UDP frames to <UDPQ>: 0 - UDP frames are not captured 1 - UDP frames are captured **NOTE:**
14	TCP_CapEn	RW 0x0	Capture TCP frames to <TCPQ> **NOTE:** This field is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register. 0 = Disable 1 = Enable

Table 690: Port Configuration (PxC) Register (i=0–3) (Continued)

Offset: Port2: 0x00032400, Port3: 0x00036400, Port0: 0x00072400, Port1: 0x00076400

Offset Formula: 0x00072400+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
13	RBArpF	RW 0x0	When the RBArp bit is cleared, then it selects the ARP packets that are accepted. **NOTE:** This field is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register. 0 = All: All ARP packets are accepted 1 = Filter: All broadcast ARP packets with IP address equal to ArpIP0Addr or to ArpIP1Addr, and all non-broadcast ARP packets are accepted.
12	AMNoTxES	RW 0x0	Automatic mode not updating Error Summary in Tx descriptor. When set, TX descriptors are not updated following packets transmission
11	Reserved	RW 0x0	Reserved
10	Reserved	RW 0x0	Reserved. Must write 0.
9	RBArp	RW 0x0	Reject mode of MAC packets that are ARP (EtherType 0x806). **NOTE:** This field is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register. 0 = Receive: To RXQArp queue. The received packets are accepted according to RBArpF bit. 1 = Reject: The received packets are discarded
8	RBIP	RW 0x0	Reject mode of MAC Broadcasts that are IP (EtherType 0x800). **NOTE:** This field is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register. 0 = Receive: To RXQ queue 1 = Reject: The received packets are discarded
7	RB	RW 0x0	Reject mode of MAC Broadcasts that are not IP or ARP Broadcast. **NOTE:** This field is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register. 0 = Receive: To RXQ queue 1 = Reject: The received packets are discarded

Table 690: Port Configuration (PxC) Register (i=0–3) (Continued)

Offset: Port2: 0x00032400, Port3: 0x00036400, Port0: 0x00072400, Port1: 0x00076400

Offset Formula: 0x00072400+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
6:4	RXQArp	RW 0x0	Default Rx Queue for ARP Broadcasts, if receiving ARP Broadcasts is enabled. **NOTE:** This field is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register.
3:1	RXQ	RW 0x0	Default Rx Queue Is the Default Rx Queue for not matched Unicast frames when UPM bit is set. It is also the default Rx Queue for all MAC broadcast (except for ARP broadcast that has a different field for default queue) if receiving them is enabled. **NOTE:** This field is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register.
0	UPM	RW 0x0	Unicast Promiscuous mode **NOTE:** This field is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register. 0 = Normal: Unicast frames are received only if the destination address is found in the DA-filter table and DA is matched against the port DA MAC Address base. 1 = Promiscuous: Unicast unmatched frames are received to the Rx queue.

Table 691: Port Configuration Extend (PxCX) Register (i=0–3)

Offset: Port2: 0x00032404, Port3: 0x00036404, Port0: 0x00072404, Port1: 0x00076404

Offset Formula: 0x00072404+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:4	Reserved	RO 0x0	Reserved
3	TxCRCDis	RW 0x0	Tx CRC generation 0 = Enable: Port generates good CRC to all Tx packets (except for error conditions).
2	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 691: Port Configuration Extend (PxCX) Register (i=0–3) (Continued)

Offset: Port2: 0x00032404, Port3: 0x00036404, Port0: 0x00072404, Port1: 0x00076404

Offset Formula: 0x00072404+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
1	Span	RW 0x0	Spanning Tree packets capture enable **NOTE:** This field is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register. 0 = Treated: BPDU packets are treated as normal Multicast packets. 1 = Trapped: BPDU packets are trapped and sent to the Port Configuration register BPDU queue.
0	Reserved	RO 0x0	Reserved

Table 692: Marvell Header Register (i=0–3)

Offset: Port2: 0x00032454, Port3: 0x00036454, Port0: 0x00072454, Port1: 0x00076454

Offset Formula: 0x00072454+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:21	Reserved	RO 0x0	Reserved
20:16	SDID	RW 0x0	Source Device ID setting to match the switch SrcDev reported in DSA tag. Used for queuing decision **NOTE:** This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.
15	Reserved	RW 0x0	Reserved
14	SDIDEn	RW 0x0	**NOTE:** This register is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register. Source Device ID enable 0 = Ignore: Ignore DSA tag Source Port ID field in queuing decision 1 = Not Ignore: Take into account DSA tag Source Port ID field in queuing decision
13:12	SPID[5:4]	RW 0x0	MSB bits of Source Port ID (for the case of a switch that supports more than 16 ports) **NOTE:** This register is valid only in Modes in which the hardware parser is used, as defined by the Port Acceleration Mode (PACC) register.

Table 692: Marvell Header Register (i=0–3) (Continued)

Offset: Port2: 0x00032454, Port3: 0x00036454, Port0: 0x00072454, Port1: 0x00076454

Offset Formula: 0x00072454+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
11:10	DSAEn	RW 0x0	DSA Tag enable **NOTE:** DSA tag and Marvell Header can not work simultaneously. If using DSA tag, <MHEn> bit[0] must be 0. 0 = Disable 1 = Non-extended: Non-extended (4 byte) DSA tag 2 = Extended: Extended (8 byte) DSA tag 3 = Reserved
9:8	MHMask	RW 0x0	**NOTE:** This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register. When enabling receive queuing based on Marvell header (DAPrefix ! = 0x0), defines how many queues will be used: 0 = 8 queues: 8 queues (0-7) 1 = 4 queues: 4 queues (0-3) 3 = 2 queues: 2 queues (0-1)
7:4	SPID[3:0]	RW 0x0	Source Port ID Setting to match the Switch SPID reported in Marvell Header for frame receive queuing. **NOTE:** This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.
3	Reserved	RO 0x0	Reserved

Table 692: Marvell Header Register (i=0–3) (Continued)

Offset: Port2: 0x00032454, Port3: 0x00036454, Port0: 0x00072454, Port1: 0x00076454

Offset Formula: 0x00072454+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
2:1	DAPrefix	RW 0x0	Rx queuing policy setting. If set to 0x0, regular Rx priority queuing. 0x1 - 0x3, queuing based on Marvell Header or DSA tag fields. It also depends on the number of queues available, as configured in the MHMask field. If using Marvell Header, configure as follows: DAPrefix = 0x1: 8 queues: queue # = PRI[2:0] 4 queues: queue # = PRI[2:1] 2 queues: queue# = PRI[2] DAPrefix = 0x2: 8 queues: queue# = (DBNum[0], PRI[2:1]) 4 queues: queue # = (DBNum[0], PRI[2) 2 queues: queue# = DBNUM[0] DAPrefix = 0x3: 8 queues: queue# = (SPID[3:0]==<SPID>,PRI[2:1]) 4 queues: queue# = (SPID[3:0]==<SPID>,PRI[2]) 2 queues: queue# = (SPID[3:0]==<SPID>) If using DSA tag, configure as follows: DAPrefix = 0x1: 8 queues: queue # = UP[2:0] 4 queues: queue # = UP[2:1] 2 queues: queue# = UP[2] DAPrefix = 0x2: If the received packet is a TO_CPU type, receive queue is determined according to CPU_Code field of the DSA tag. The mapping from CPU_Code to Rx queue is extracted from DFSMT table. If the received packet is FORWARD format, Rx queue is calculated like in the case of DAPrefix = 0x1. DAPrefix = 0x3: 8 queues: queue# = (SrcPort==<SPID> & SrcDev = <SDID>,UP[2:1]) 4 queues: queue# = (SrcPort==<SPID> & SrcDev = <SDID>,UP[2]) 2 queues: queue# = (SrcPort==<SPID> & SrcDev = <SDID>) **NOTE:** When using <DA_PREFIX> = 0x3, if there is no SrcPort (FORWARD format, with SrcIsTrunk bit set), Rx queue is determined according to UP[2:0] field like in the case of DA_PREFIX = 0x1. If the received packet is not TO_CPU nor FORWARD format, packet is queued as if DA_PREFIX is 0x0 (regular queuing). **NOTE:** This register is valid only in modes in which the hardware parser is used, as defined by the Port Acceleration mode (PACC) register.
0	MHEn	RW 0x0	Marvell Header Enable 0 = Disable 1 = Enable: 2 octets of Marvell header are placed between SFD and DA.

A.6.15 Serial Registers

Table 693: Port Serial Control0 (PSC0) Register (i=0–3)

Offset: Port2: 0x0003243C, Port3: 0x0003643C, Port0: 0x0007243C, Port1: 0x0007643C

Offset Formula: 0x0007243C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31	Reserved	RO 0x0	Reserved
30	IgnoreCarrierSense	RW 0x0	This bit controls if the carrier sense input pin is ignored: 0 = Not ignored 1 = Ignored
29	IgnoreCollision	RW 0x0	This bit controls if the collision input pin is ignored: 0 = Not ignored 1 = Ignored
28	IgnoreRXError	RW 0x0	This bit controls if RX error input pin is ignored:
27:15	Reserved	RO 0x0	Reserved
14	DTEAdvert	RW 0x0	DTE advertise The value of this bit is written to bit 9.10 of the 1000BaseT PHY device after power up or detection of a link failure. **NOTE:** The PHY register bit is updated, only if the AnFcEn bit of the Port Auto-Negotiation Configuration Register is set.
13:12	Reserved	RO 0x0	Reserved
11	Reserved	RSVD 0x0	Reserved
10:9	Reserved	RO 0x0	Reserved
8:7	ForceBPMode	RW 0x0	When this bit is set, the port will start transmitting JAM on the line (Backpressure) in half-duplex (which is only supported in 10/100 Mbps) according to the following settings: **NOTE:** When the link falls, this field goes to disabled 0x00 and must be reprogrammed only after the link is up. 0 = No JAM: (no backpressure) 1 = JAM: Is transmitted continuously, on next frame boundary.

Copyright © 2014 Marvell

Table 693: Port Serial Control0 (PSC0) Register (i=0–3) (Continued)

Offset: Port2: 0x0003243C, Port3: 0x0003643C, Port0: 0x0007243C, Port1: 0x0007643C

Offset Formula: 0x0007243C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
6:5	ForceFCMode	RW 0x0	The port will transmit Pause enable and disable frames depending on the CPU writing 00 and 01 values, conditioned with the flow-control operation enable as reflected in the Port Status Register0 <TxFcEn> bit being set in the following values: **NOTE:** Only one mode is supported for enabling flow-control. When the link falls, this field goes to disabled 0x00 and must be reprogrammed only after the link is up. 0 = No Pause: No Pause disable frames are sent. However, when the value of the field is changed by the CPU from 01 to 00, the port will send a single Pause enable packet (timer 0x0000) to enable the other side to transmit. 1 = Pause: When this field is set to 01 value, and Flow Control is enabled (The Port Status Register0 <TxFcEn> bit is set) then Pause disable frames (timer 0xFFFF) are retransmitted at a period as defined by the PFCDiv field of the Periodic Flow Control Clock Divider Register.
4:0	Reserved	RO 0x0	Reserved

Table 694: Ethernet Port Status 0 (PS0) Register (i=0–3)

Offset: Port2: 0x00032444, Port3: 0x00036444, Port0: 0x00072444, Port1: 0x00076444

Offset Formula: 0x00072444+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:17	Reserved	RO 0x0	Reserved
16	RxFIFOEmpty	RO 0x1	RX Fifo Empty When set to "1", indicates that the port Receive FIFO is empty.
15:9	Reserved	RO 0x0	Reserved
8	TxFIFOEmp	RO 0x0	TX Fifo Empty When set to "1", indicates that the port Transmit FIFO is empty.
7:1	Reserved	RO 0x0	Reserved
0	TxInProg	RO 0x0	Transmit in Progress When equal to "1", indicates that the port’s transmitter is in an active transmission state.

Table 695: Serdes Configuration (SERDESCFG) Register (i=0–3)

Offset: Port2: 0x000324A0, Port3: 0x000364A0, Port0: 0x000724A0, Port1: 0x000764A0

Offset Formula: 0x000724A0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	SerdesConfiguration Data	RW 0x02887	Serdes Configuration Data

Table 696: Serdes Status (SERDESSTS) Register (i=0–3)

Offset: Port2: 0x000324A4, Port3: 0x000364A4, Port0: 0x000724A4, Port1: 0x000764A4

Offset Formula: 0x000724A4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	SerdesStatusData	RO 0x0	Serdes Status Data

Table 697: One mS Clock Divider Register (i=0–3)

Offset: Port2: 0x000324F4, Port3: 0x000364F4, Port0: 0x000724F4, Port1: 0x000764F4

Offset Formula: 0x000724F4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31	En	RW 0x0	Enables 1ms clock generation. This clock is used for SGMII Auto-Negotiation. **NOTE:** In SGMII mode, this clock must be enabled. 0 = Disable 1 = Enable
30:0	Div	RW 0x28B0A	Controls the period of the 1ms clock. This clock is used for SGMII Auto-Negotion. The value to be configured in this field, as a function of the core clock frequency is as follows: 166 MHz - 166666 (0x28B0A) 200 MHz - 200000 (0x30D40) 250 MHz - 250000 (0x3D090) **NOTE:** For a 1-ms period, the data to be configured to this field is according to the formula: 0.001 * (Core clock frequency)

Table 698: Periodic Flow Control Clock Divider Register (i=0–3)

Offset: Port2: 0x000324F8, Port3: 0x000364F8, Port0: 0x000724F8, Port1: 0x000764F8

Offset Formula: 0x000724F8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:0	PFCDiv	RW 0x2660CE	Periodic Flow Control Divider This field defines the period of the transmitted flow control packets. The value to be configured for a 15MSec period, as a function of core clock frequency is as following: 166MHz - 2500000 (0x2625A0) 200MHz - 3000000 (0x2DC6C0) 250MHz - 3750000 (0x393870) Note: The generated period is according to the formula: PFCDiv/(Core clock frequency)

A.6.16 Gigabit Ethernet MAC Serial Parameters Configuration Registers

Table 699: Port Serial Parameters Configuration Register (i=0–3)

Offset: Port2: 0x00032C14, Port3: 0x00036C14, Port0: 0x00072C14, Port1: 0x00076C14

Offset Formula: 0x00072C14+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	Reserved	RW 0x0	Reserved. Must write 0x0.
14	Reserved	RW 0x0	Reserved. Must write 0x0.
13	Reserved	RW 0x0	Reserved. Must write 0x0.
12	Reserved	RW 0x0	Reserved. Must write 0x0.
11:6	Reserved	RW 0x23	Reserved. Must write 0x23.
5	Reserved	RW 0x1	Reserved. Must write 0x1.
4	Reserved	RW 0x1	Reserved. Must write 0x1.
3	Reserved	RW 0x1	Reserved. Must write "1".

Table 699: Port Serial Parameters Configuration Register (i=0–3) (Continued)

Offset: Port2: 0x00032C14, Port3: 0x00036C14, Port0: 0x00072C14, Port1: 0x00076C14

Offset Formula: 0x00072C14+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
2	Reserved	RW 0x0	Reserved. Must write 0x0.
1	Reserved	RW 0x0	Reserved. Must write 0x0.
0	Reserved	RSVD 0x0	Reserved

Table 700: Port Serial Parameters 1 Configuration Register (i=0–3)

Offset: Port2: 0x00032C94, Port3: 0x00036C94, Port0: 0x00072C94, Port1: 0x00076C94

Offset Formula: 0x00072C94+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:3	Reserved	RO 0x0	Reserved
2	Reserved	RW 0x1	Reserved. Must write 0x1.
1	Reserved	RW 0x1	Reserved. Must write 0x1.
0	Reserved	RW 0x1	Reserved. Must write 0x1.

A.6.17 Gigabit Ethernet MAC Auto-Negotiation Configuration Registers

Table 701: Port Auto-Negotiation Configuration Register (i=0–3)

Offset: Port2: 0x00032C0C, Port3: 0x00036C0C, Port0: 0x00072C0C, Port1: 0x00076C0C

Offset Formula: 0x00072C0C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	Reserved	RW 0x1	Reserved. Must write 0x1.
14	Reserved	RW 0x0	Reserved. Must write 0.

Table 701: Port Auto-Negotiation Configuration Register (i=0–3) (Continued)

Offset: Port2: 0x00032C0C, Port3: 0x00036C0C, Port0: 0x00072C0C, Port1: 0x00076C0C

Offset Formula: 0x00072C0C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
13	AnDuplexEn	RW 0x1	Enable Auto-Negotiation for duplex mode **NOTE:** Half-Duplex mode is not supported in 1000 Mbps mode. 0 = AnDuplexDisable.: Duplex mode is according to Set_FullDx bit 1 = AnDuplexEnable.: Duplex mode is according to Auto-Negotiation result
12	Set_FullDx	RW 0x1	Half/Full Duplex Mode When <AnDuplexEn> bit is set to "Disable", then this bit defines the port half/full duplex mode: 0 = Half duplex: Port operates in half-duplex mode. 1 = Full duplex: Port operates in full-duplex mode.
11	AnFcEn	RW 0x0	Enable Auto-Negotiation for Flow Control When enabled, the port can either advertise no flow control or symmetric flow-control according to the <Pause_Adv> bit. Asymmetric flow-control advertisement is not supported. 0 = AnFcDisable.: Flow control mode is according to SetFCEn bit. 1 = AnFcEnable.: Flow control mode is according to Auto-Negotiation result
10	Reserved	RW 0x0	Reserved. Must write "0"
9	Pause_Adv	RW 0x1	Flow control advertise This bit contains the port flow control capability to be advertised. It is valid only if flow control mode is defined by Auto-Negotiation (as defined by the <AnFcEn> bit). **NOTE:** The port does not modify this bit as result of the Auto-Negotiation process 0 = No Flow Control: Advertise no Flow Control. 1 = Symmetric Flow Control: Advertise symmetric flow control support in Auto-Negotiation.
8	SetFcEn	RW 0x1	Flow Control Enable When <AnFcEn> bit is set to "Disable", then this bit enables receiving and transmitting of 802.3 and per priority Flow Control frames in full duplex, or enabling of backpressure in half duplex: 0 = Disabled: Transmission and reception of flow control frames is disabled. 1 = Enabled: Transmission and reception of flow control frames is enabled.

Table 701: Port Auto-Negotiation Configuration Register (i=0–3) (Continued)

Offset: Port2: 0x00032C0C, Port3: 0x00036C0C, Port0: 0x00072C0C, Port1: 0x00076C0C

Offset Formula: 0x00072C0C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
7	AnSpeedEn	RW 0x1	Enable Auto-Negotiation of interface speed 0 = Disable: Port speed is according to SetGMIISpeed and SetMIISpeed bits 1 = Enable: Port speed is according to Auto-negotiation result
6	SetGMIISpeed	RW 0x1	GMII Speed When <ANSpeedEn> bit is set to "Disable" then this bit (together with the SetMIISpeed bit) defines the speed of the port: 0 = 10/100 Mbps: Port works at 10/100 Mbps (as defined by SetMIISpeed bit) 1 = 1000 Mbps.: Port works at 1000 Mbps.
5	SetMIISpeed	RW 0x1	MII Speed When <ANSpeedEn> bit is set to "Disable" and <SetGMIISpeed> bit is set to "10/100 Mbps" then this bit defines the speed of the port: 0 = 10 Mbps Speed: Port works in 10 Mbps 1 = 100 Mbps Speed: Port works in 100 Mbps
4	InBandReStartAn	RW 0x0	In Band Restart Auto-Negotiation Relevant only when Inband Auto Negotiation is enabled (<InBandAnEn>= 1). Writing "1" to this bit restarts In Band Auto-Negotiation.
3	InBandAnByPassEn	RW 0x1	In Band Auto-Negotiation Bypass Enable Relevant only when Inband Auto Negotiation is enabled (<InBandAnEn>= 1). When this bit is set to1, if the link partner does not respond to the Auto-Negotiation process, the link is established by bypassing the Auto-Negotiation procedure. After bypassing Auto-Negotiation, Link is set to be UP, and duplex is set according to <SetFullDx>. **NOTE:** A change of this field must be done when the port’s link is down. 0 = No Bypass: Auto-Negotiation cannot be bypassed 1 = Bypass: Auto-Negotiation bypassed

Table 701: Port Auto-Negotiation Configuration Register (i=0–3) (Continued)

Offset: Port2: 0x00032C0C, Port3: 0x00036C0C, Port0: 0x00072C0C, Port1: 0x00076C0C

Offset Formula: 0x00072C0C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
2	InBandAnEn	RW 0x0	In Band Auto-Negotiation Enable In Band Auto Negotiation is done if port is 1000Base-X for link and flow control support. When <Port MAC Control Register0>/<PortType> = 1 (1000Base-X) this field must be set to 1 When <Port MAC Control Register0>/<PortType> = 0 In band flow control auto negotiation is not supported (only speed and duplex are negotiated). In this case, flow control support is resolved according to SMI auto negotiation if it is enabled (<ANFCEn> = 1) or according to <SetFCEn> if it is disabled. A change of this field must be done when the port’s link is down. 0 = Disabled 1 = Enabled
1	ForceLinkPass	RW 0x0	Force Link status on port to Link UP state **NOTE:** When both <ForceLink Down> and <ForceLink Pass> are HIGH, the link is forced UP. When both ForceLinkDown and ForceLinkPass are LOW, the port link state is determined according to the line state that is determined by Auto-Negotiation 0 = Not Force: Do NOT Force Link Pass 1 = Force: Force Link Pass
0	ForceLinkFail	RW 0x0	Force Link status on port to Link DOWN state **NOTE:** When both <ForceLink Down> and <ForceLink Pass> are HIGH, the link is forced UP. When both ForceLinkDown and ForceLinkPass are LOW, the port link state is determined according to the line state that is determined by Auto-Negotiation 0 = Not Force: Do NOT Force Link Fail 1 = Force: Force Link Fail

A.6.18 Gigabit Ethernet MAC Control Registers

Table 702: Port MAC Control Register0 (i=0–3)

Offset: Port2: 0x00032C00, Port3: 0x00036C00, Port0: 0x00072C00, Port1: 0x00076C00

Offset Formula: 0x00072C00+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved

Table 702: Port MAC Control Register0 (i=0–3) (Continued)

Offset: Port2: 0x00032C00, Port3: 0x00036C00, Port0: 0x00072C00, Port1: 0x00076C00

Offset Formula: 0x00072C00+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15	Reserved	RW 0x1	Reserved. Must write 0x1.
14:2	FrameSizeLimit	RW 0x2f9	Maximal Received Packet Size (MRU) Packets received on the port that exceed this configured size are considered over-sized (if packet CRC is good) or jabber (if packet CRC is bad) and are dropped. The resolution of this field is two bytes and the default value is 1522 bytes **NOTE:**
1	PortType	RW 0x0	This bit indicates the port type. This field may only be changed when the port link is down. Note: 1000Base-X mode is available only when PCS layer is connected directly to LP Serdes PHY. 0 = SGMII: Port is in SGMII mode and is connected to a copper/Fiber PHY with the ability to operate at speeds of 10, 100, and 1000 Mbps 1 = 1000Base-X_mode: Port is in 1000Base-X mode and is connected to a fiber transceiver or it is routed on a backplane to another SERDES
0	PortEn	RW 0x1	Port Enable This bit controls packets reception by the port: 0 = Port Disabled: Port packets reception is disabled 1 = Port Enabled: Port packets reception is enabled

Table 703: Port MAC Control Register1 (i=0–3)

Offset: Port2: 0x00032C04, Port3: 0x00036C04, Port0: 0x00072C04, Port1: 0x00076C04

Offset Formula: 0x00072C04+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	Reserved	RW 0x0	Reserved. Must write 0x0.
14:7	Reserved	RW 0x0	Reserved

Table 703: Port MAC Control Register1 (i=0–3) (Continued)

Offset: Port2: 0x00032C04, Port3: 0x00036C04, Port0: 0x00072C04, Port1: 0x00076C04

Offset Formula: 0x00072C04+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
6	PCS_LB	RW 0x0	Loop back on the PCS Block **NOTE:** A change of this field must be done when the port's link is down. 0 = No loopback: Normal operation, no loopback 1 = Loopback: Loopback on the PCS Block, the PCS Tx is connected directly to the PCS Rx, the PCS TX clock is used ad PCS RX clock
5	Reserved	RW 0x0	Reserved. Must write 0x0.
4	Reserved	RW 0x1	Reserved. Must write 0x1.
3	Reserved	RW 0x0	Reserved. Must write 0x0.
2	Reserved	RW 0x0	Reserved. Must write 0x0.
1	Reserved	RW 0x0	Reserved. Must write 0x0.
0	Reserved	RW 0x1	Reserved. Must write 0x1.

Table 704: Port MAC Control Register2 (i=0–3)

Offset: Port2: 0x00032C08, Port3: 0x00036C08, Port0: 0x00072C08, Port1: 0x00076C08

Offset Formula: 0x00072C08+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	Reserved	RW 0x1	Reserved. Must write 0x1.
14	Reserved	RW 0x1	Reserved. Must write 0x1.
13:12	SelectData ToTransmit	RW 0x0	The data forwarded to the SERDES. A changes to this field must be made only when the port link is down. **NOTE:** Not relevant for CPU port. 0 = Regular Mode: 1.25 Gbps input is from the MAC PCS Tx block. 1 = PRBS Mode: 1.25 Gbps input is from the PRBS Generator. 2 = Zeros Constant 3 = Ones Constant

Table 704: Port MAC Control Register2 (i=0–3) (Continued)

Offset: Port2: 0x00032C08, Port3: 0x00036C08, Port0: 0x00072C08, Port1: 0x00076C08

Offset Formula: 0x00072C08+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
11	PRBSGenEn	RW 0x0	Relevant when <SelectData ToTransmit> is set to 1. Enables generation of the pseudo-random bit-stream. **NOTE:** Not relevant for the CPU port. This field may be changed only when the port link is down. 0 = PRBS Generator Disabled 1 = PRBS Generator Enabled
10	PRBSCheckEn	RW 0x0	PRBS Checkers Enable When a PRBS checker shifts from the disabled to enabled state, it starts an initialization phase. When this phase is done, <PrbsCheckRdy> is set to 1. After the initialization phase is done, the PRBS checker starts searching for the initial alignment pattern on the bit stream received from the SERDES. When alignment lock is reached, <PrbsCheckRdy> is set to 1 and <PrbsBitErrCnt> counts mismatches in the received bit stream. **NOTE:** Not relevant for the CPU port. 0 = PRBS Checker Disabled 1 = PRBS Checker Enabled
9	Reserved	RW 0x0	Reserved. Must write 0x0.
8	Reserved	RW 0x0	Reserved. Must write 0x0.
7	Reserved	RW 0x0	Reserved. Must write 0x0.
6	PortMACReset	RW 0x1	Port MAC Reset. Not relevant when the port is powered up at reset. 0 = Port MAC Not_reset 1 = Port MAC Reset
5	PaddingDis	RW 0x0	Disable padding of transmitted packets shorter than 64B. 0 = Pad: Pad short packet on Tx. 1 = Disable: Disable padding in short packets.
4	RGMIIEn	RW 0x0	RGMII Enable 0 = MII Interface 1 = RGMII interface
3	PcsEn	RW 0x0	Port PCS block is active 0 = Not_working with PCS 1 = Working with PCS
2:1	Reserved	RW 0x0	Reserved. Must write 0x0.

Table 704: Port MAC Control Register2 (i=0–3) (Continued)

Offset: Port2: 0x00032C08, Port3: 0x00036C08, Port0: 0x00072C08, Port1: 0x00076C08

Offset Formula: 0x00072C08+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
0	Reserved	RW 0x0	Reserved. Must write 0x0.

Table 705: Port MAC Control Register3 (i=0–3)

Offset: Port2: 0x00032C48, Port3: 0x00036C48, Port0: 0x00072C48, Port1: 0x00076C48

Offset Formula: 0x00072C48+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	Reserved	RW 0x0	Reserved. Must write 0x0.
14:6	IPG	RW 0xc	Minimal Inter-packet gap Minimal IPG between two consecutive transmitted packets. Resolution of this field is 8-bits and its default value is the standard 96-bits.
5:0	Reserved	RW 0x1	Reserved. Must write 0x1.

Table 706: CCFC Port Speed Timerp Register (i=0–3, p=0–7)

Offset: Port2Port Speed Index0: 0x00032C58, Port2Port Speed Index1: 0x00032C5C...Port1Port Speed Index7: 0x00076C74

Offset Formula: 0x00072C58+4 * p + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where p (0-7) represents Port Speed Index

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:0	PortSpeedTimer	RW 0x0	Timer value bits [15:0] embedded in the IEEE 802.3x frame transmitted by the port when triggered by the CCFC/Global Pause engines.

Table 707: Port MAC Control Register4 (i=0–3)

Offset: Port2: 0x00032C90, Port3: 0x00036C90, Port0: 0x00072C90, Port1: 0x00076C90

Offset Formula: 0x00072C90+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:2	Reserved	RO 0x0	Reserved
1	Reserved	RW 0x0	Reserved. Must write 0x0.
0	Reserved	RW 0x0	Reserved. Must write 0.

A.6.19 Gigabit Ethernet MAC Interrupt Registers

Table 708: Port Interrupt Cause Register (i=0–3)

Offset: Port2: 0x00032C20, Port3: 0x00036C20, Port0: 0x00072C20, Port1: 0x00076C20

Offset Formula: 0x00072C20+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	Reserved	ROC 0x0	Reserved **NOTE:**
14	Reserved	ROC 0x0	Reserved **NOTE:**
13	MAC Rx path received LPI	ROC 0x0	Rx MAC has received LP_IDLE
13:8	Reserved	RSVD 0x0	Reserved
12	PCS Tx path received LPI	ROC 0x0	Tx PCS has received LP_IDLE
11	PCS Rx path received LPI	ROC 0x0	Rx PCS has received LP_IDLE.
10	QSGMII PRBS error	ROC 0x0	QSGMII PRBS Error
7	PRBSError OnPort	ROC 0x0	Asserted when the PRBS checker is enabled and an error is detected. **NOTE:** Not relevant for the CPU port. **NOTE:**

Table 708: Port Interrupt Cause Register (i=0–3) (Continued)

Offset: Port2: 0x00032C20, Port3: 0x00036C20, Port0: 0x00072C20, Port1: 0x00076C20

Offset Formula: 0x00072C20+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
6	SyncStatus Change	ROC 0x0	Asserted when a change in the port PCS layer Sync status is detected. **NOTE:** Not relevant for the CPU port. **NOTE:**
5	AddressOut OfRange	ROC 0x0	Set when the CPU attempts to read from or write to an illegal address in one of the port registers.
4:3	Reserved	RSVD 0x0	Reserved
2	AnCompleted OnPort	ROC 0x0	Asserted when the Auto-Negotiation process is completed. **NOTE:** Not relevant for the CPU port. **NOTE:**
1	LinkStatus Change	ROC 0x0	Asserted when a change in the link status is detected.
0	IntSum	RO 0x0	Sum of all unmasked interrupts in this register.

Table 709: Port Interrupt Mask Register (i=0–3)

Offset: Port2: 0x00032C24, Port3: 0x00036C24, Port0: 0x00072C24, Port1: 0x00076C24

Offset Formula: 0x00072C24+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:1	InterruptMask	RW 0x0	Mask bit for the corresponding bit in the Port<n> Interrupt Cause register. 0 = Interrupt Masked 1 = Interrupt Unmasked
0	Reserved	RSVD 0x0	Reserved

A.6.20 Gigabit Ethernet MAC Low Power Idle Registers

Table 710: LPI Control 0 Register (i=0–3)

Offset: Port2: 0x00032CC0, Port3: 0x00036CC0, Port0: 0x00072CC0, Port1: 0x00076CC0

Offset Formula: 0x00072CC0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:8	Ts limit	RW 0x10	The minimum time from emptying of Tx FIFO and LPI assert. Granularity is 1 uSec for 1000 Mbps and 10 uSec for 100 Mbps
7:0	Li limit	RW 0x4	Minimum time since going into LPI and allowing LPI deassert. Granularity is 1uSec for 1000Mbps and 10uSec for 100Mbps **NOTE:** Maximum allowed value is 0xFE **NOTE:**

Table 711: LPI Control 1 Register (i=0–3)

Offset: Port2: 0x00032CC4, Port3: 0x00036CC4, Port0: 0x00072CC4, Port1: 0x00076CC4

Offset Formula: 0x00072CC4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:4	Tw limit	RW 0x10	The minimum time from LPI deassertion until valid data can be sent. Granularity is 1 uSec for 1000 Mbps and 10 uSec for 100 Mbps
3	Reserved	RSVD 0x0	Reserved
2	LPI manual mode	RW 0x0	In manual mode, the CPU can force the Tx port to go into LP_IDLE mode. The LP state is kept until the CPU deasserts the manual mode or lp_idle_request_force. The Tw_limit is still kept.
1	LPI request force	RW 0x0	Force LPI signaling on Tx. Force happens only in IDLE state of the LPI state machine.
0	LPI request enable	RW 0x0	Enable assertion or deassertion of LPI.

Table 712: LPI Control 2 Register (i=0–3)

Offset: Port2: 0x00032CC8, Port3: 0x00036CC8, Port0: 0x00072CC8, Port1: 0x00076CC8

Offset Formula: 0x00072CC8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: The fields of this register should be initialized to the mentioned values.			
31:9	Reserved	RO 0x0	Reserved
8	Reserved	RW 0x1	Reserved. Must write "1".
7	Reserved	RW 0x1	Reserved. Must write "0".
6:0	Reserved	RW 0x7d	Reserved. Must write 0x7d.

Table 713: LPI Status Register (i=0–3)

Offset: Port2: 0x00032CCC, Port3: 0x00036CCC, Port0: 0x00072CCC, Port1: 0x00076CCC

Offset Formula: 0x00072CCC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:5	Reserved	RO 0x0	Reserved
4	MAC Tx path LP idle status	RO 0x0	Indicates that the MAC is sending LPI to the Tx PHY
3	MAC Tx path LP wait status	RO 0x0	Indicates that the MAC is sending IDLE to the Tx PHY and waiting for Tw timer to end.
2	MAC Rx path LP idle status	RO 0x0	Indicates that the PHY has detected the assertion / deassertion of LP_ IDLE from link partner (through the PCS).
1	PCS Tx path LPI status	RO 0x0	PCS TX path LPI status 0 = Does not receive: Tx PCS does not receive LP_IDLE. 1 = Receives: Tx PCS receives LP_IDLE.
0	PCS Rx path LPI status	RO 0x0	PCS RX path LPI status 0 = Does not receive: Rx PCS does not receive LP_IDLE 1 = Receives: Rx PCS receives LP_IDLE.

Table 714: LPI Counter Register (i=0–3)

Offset: Port2: 0x00032CD0, Port3: 0x00036CD0, Port0: 0x00072CD0, Port1: 0x00076CD0

Offset Formula: 0x00072CD0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:0	LPI counter	RW0C 0x0	This counter is incremented each time the MAC LPI state machine moves from IDLE to LP_IDLE.

A.6.21 Gigabit Ethernet MAC PRBS Check Status Registers

Table 715: Port PRBS Status Register (i=0–3)

Offset: Port2: 0x00032C38, Port3: 0x00036C38, Port0: 0x00072C38, Port1: 0x00076C38

Offset Formula: 0x00072C38+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:2	Reserved	RO 0x0	Reserved
1	PrbsCheckRdy	RO 0x0	PRBS Checker Ready Indicates that the PRBS checker has completed the initialization phase. The PRBS generator at the transmit side may be enabled.
0	PrbsCheck Locked	RO 0x0	PRBS Checkers Alignment Locked Indicates that the PRBS checker has locked alignment to the bit-stream received from the SERDES PRBS generator.

Table 716: Port PRBS Error Counter Register (i=0–3)

Offset: Port2: 0x00032C3C, Port3: 0x00036C3C, Port0: 0x00072C3C, Port1: 0x00076C3C

Offset Formula: 0x00072C3C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15:0	PrbsBitErrCnt	ROC 0x0	PRBS Bit Error Count This counter represents the number of bit mismatches detected since the PRBS checker of the SERDES port reached alignment lock.

A.6.22 Gigabit Ethernet MAC Status Registers

Table 717: Port Status Register0 (i=0–3)

Offset: Port2: 0x00032C10, Port3: 0x00036C10, Port0: 0x00072C10, Port1: 0x00076C10

Offset Formula: 0x00072C10+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	Reserved	RO 0x0	Reserved
14	SyncOk	RO 0x0	This bit, when set, indicates if the PCS has Detected three comma patterns and is synchronized with its peer PCS Layer 0 = No comma: PCS layer has not detected comma patters. 1 = Comma: PCS has detected comma patterns and is synchronized with its peer PCS layer.
13	Reserved	RO 0x1	Reserved
12	InBand AutoNeg BypassAct	RO 0x0	When set, this bit indicates that In Band Auto-Negotiation Bypass was activated. This happens when the <InBandAnByPassEn> and <InBandAnEn> bits in the Port Auto-Negotiation Configuration register are set and during Inband Auto-Negotiation the peer does not reply. If the link partner does not respond to the Auto-Negotiation process, the link is established by bypassing the Auto-Negotiation procedure. After bypassing Auto-Negotiation, the link is set to be UP. Speed is set according to <SetGMIISpeed> and <SetMIISpeed> in the Port Auto-Negotiation Configuration Register. Duplex is set according to <Set_FullDx> and FC is set according to <SetFcEn> in that same register. **NOTE:** When port is In 1000Base-T, (<Port MAC Control Register0>/<PortType> = 0) in band flow control auto negotiation is not supported. In this case, flow control support is resolved according to SMI auto negotiation if it is enabled (<Port Auto-Negotiation Configuration Register>/<AnFcEn>= 1) or according to <Port Auto-Negotiation Configuration Register>/<SetFcEn> if it is disabled. 0 = No Bypass: Auto-Negotiation was not bypassed 1 = Bypassed: Auto-Negotiation was bypassed
11	AnDone	RO 0x0	Auto-Negotiation done 0 = In Progress: Auto-Negotiation is in progress. 1 = Done: Auto-Negotiation process is done.

Table 717: Port Status Register0 (i=0–3) (Continued)

Offset: Port2: 0x00032C10, Port3: 0x00036C10, Port0: 0x00072C10, Port1: 0x00076C10

Offset Formula: 0x00072C10+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
10	SyncFail10ms	RO 0x0	Indicates if the 10-bit state machine has been in <SyncOk> = FAIL for more than 10 ms. Refer to IEEE 802.3 clause 36. 0 = In Sync 1 = Sync Down: Sync is down for more than 10ms.
9	Reserved	RO 0x0	Reserved
8	PortIs DoingPressure	RO 0x0	Port is Doing Back-Pressure This bit indicates if the port is doing back-pressure: **NOTE:** Back-Pressure is done in half duplex when the <ForceBPMode> field of the Port Serial Control0 (PSC0) register is configured to "JAM". 0 = No Back-Pressure: The port is not doing back-pressure 1 = Back-Pressure: The port is doing back-pressure
7	PortTxPause	RO 0x0	Relevant when the port is in full-duplex mode, <FullDx> = 1, and <TxFcEn> = 1. The port is sending IEEE 802.3x Flow Control pause frames due to lack of buffers. 0 = No_Pause: Port has buffers for reception of new packets (in Xon Status). 1 = Pause: Number of buffers allocated for this port is above its configured Xoff threshold, and it is sending 802.3x Flow Control pause frames with Timer 0xFFFF. New packets are received until the port reaches its configured buffers limit threshold.
6	PortRxPause	RO 0x0	Relevant when the port is in full-duplex mode, <FullDx> = 1, and <RxFcEn> = 1. The port received an IEEE 802.3x pause frame and stopped transmission of packets. 0 = No_Pause: Port is in normal state and is sending packets. Either an XON packet was received or the pause Timer has expired. 1 = Pause: Port received a pause and pause Timer has not expired yet. The port halts its transmission until the timer is expired.

Table 717: Port Status Register0 (i=0–3) (Continued)

Offset: Port2: 0x00032C10, Port3: 0x00036C10, Port0: 0x00072C10, Port1: 0x00076C10

Offset Formula: 0x00072C10+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
5	TxFcEn	RO 0x0	TX Flow Control Enable This bit indicates if transmission of flow control frames is enabled or disabled. The state of this bit is defined according to the <AnFcEn> bit of the Port Auto-Negotiation Configuration Register either by Auto-Negotiation or by the <SetFcEn> bit of the Port Auto-Negotiation Configuration Register 0 = Disabled: Transmission of flow control frames is disabled 1 = Enabled: Transmission of flow control frames is enabled
4	RxFcEn	RO 0x0	RX Flow Control Enable This bit indicates if reception of flow control frames is enabled or disabled. The state of this bit is defined according to the <AnFcEn> bit of the Port Auto-Negotiation Configuration Register either by Auto-Negotiation or by the <SetFcEn> bit of the Port Auto-Negotiation Configuration Register 0 = Disabled: Reception of flow control frames is disabled 1 = Enabled: Reception of flow control frames is enabled
3	FullDx	RO 0x0	Half-/Full-Duplex mode. This bit indicates the port duplex mode. The duplex mode is defined according to the <AnDuplexEn> bit of the Port Auto-Negotiation Configuration Register, either by Auto-Negotiation or by the <Set_FullDx> bit of the Port Auto-Negotiation Configuration Register. **NOTE:** Half-Duplex is not supported in 1000 Mbps. 0 = half-duplex: Port is in half-duplex mode. 1 = full-duplex: Port is in full-duplex mode.
2	MIISpeed	RO 0x0	MII Speed This bit, when the <GMIISpeed> bit is set to 10/100Mbps, indicates the port speed (only when the interface mode is GMII/MII). The port speed is defined according to the <ANSpeedEn> bit of the Port Auto-Negotiation Configuration Register, either by Auto-Negotiation or by the <SetGMIISpeed> and <SetMIISpeed> bits of the Port Auto-Negotiation Configuration Register. 0 = 10: Port speed is 10 Mbps. 1 = 100: Port speed is 100 Mbps.

Table 717: Port Status Register0 (i=0–3) (Continued)

Offset: Port2: 0x00032C10, Port3: 0x00036C10, Port0: 0x00072C10, Port1: 0x00076C10

Offset Formula: 0x00072C10+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
1	GMIISpeed	RO 0x0	GMII Speed This bit, together with the MIISpeed bit, indicates the port speed (only when the interface mode is GMII/MII). The port speed is defined according to the <ANSpeedEn> bit of the Port Auto-Negotiation Configuration Register, either by Auto-Negotiation or by the <SetGMIISpeed> and <SetMIISpeed> bits of the Port Auto-Negotiation Configuration Register. 0 = 10/100 Mbps: Port speed is 10 Mbps or 100 Mbps according to <MIISpeed> . 1 = 1000 Mbps: Port speed is 1000 Mbps.
0	LinkUp	RO 0x0	Indicates the port link status. 0 = False: Link is down. 1 = True: Link is up.

A.6.23 Networking Controller Interrupt Registers

Table 718: Port CPU0 to Queue (PCP02Q) Register (i=0–3)

Offset: Port2: 0x00032540, Port3: 0x00036540, Port0: 0x00072540, Port1: 0x00076540

Offset Formula: 0x00072540+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:16	Reserved	RO 0x0	Reserved
15:8	TXQueueAccessEnable[7:0]	RW 0xff	TX Queues Access Enable. It contains 1 bit per queue as following: bit 8 for queue no. 0, bit 9 for queue no. 1 etc. Each bit controls access of CPU#0 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC, PRXTXIC & PMiscIC) and interrupt mask register (PRXTXThIM, PRXTXIM & PMiscIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.

Table 718: Port CPU0 to Queue (PCP02Q) Register (i=0–3) (Continued)

Offset: Port2: 0x00032540, Port3: 0x00036540, Port0: 0x00072540, Port1: 0x00076540

Offset Formula: 0x00072540+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
7:0	RXQueueAccessEnable[7:0]	RW 0xff	RX Queues Access Enable. It contains 1 bit per queue as following: bit 0 for queue no. 0, bit 1 for queue no. 1 etc. Each bit controls access of CPU#0 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC & PRXTXIC) and interrupt mask register (PRXTXThIM & PRXTXIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.

Table 719: Port CPU1 to Queue (PCP12Q) Register (i=0–3)

Offset: Port2: 0x00032544, Port3: 0x00036544, Port0: 0x00072544, Port1: 0x00076544

Offset Formula: 0x00072544+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register **NOTE:** This register is used only for multi-cpu devices			
31:16	Reserved	RO 0x0	Reserved
15:8	TXQueueAccessEnable[7:0]	RW 0x0	TX Queues Access Enable. It contains 1 bit per queue as following: bit 8 for queue no. 0, bit 9 for queue no. 1 etc. Each bit controls access of CPU#1 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC, PRXTXIC & PMiscIC) and interrupt mask register (PRXTXThIM, PRXTXIM & PMiscIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.
7:0	RXQueueAccessEnable[7:0]	RW 0x0	RX Queues Access Enable. It contains 1 bit per queue as following: bit 0 for queue no. 0, bit 1 for queue no. 1 etc. Each bit controls access of CPU#1 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC & PRXTXIC) and interrupt mask register (PRXTXThIM & PRXTXIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.

Table 720: Port CPU2 to Queue (PCP22Q) Register (i=0–3)

Offset: Port2: 0x00032548, Port3: 0x00036548, Port0: 0x00072548, Port1: 0x00076548

Offset Formula: 0x00072548+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register **NOTE:** This register is used only for multi-cpu devices			
31:16	Reserved	RO 0x0	Reserved
15:8	TXQueueAccessEnable[7:0]	RW 0x0	TX Queues Access Enable. It contains 1 bit per queue as following: bit 8 for queue no. 0, bit 9 for queue no. 1 etc. Each bit controls access of CPU#2 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC, PRXTXIC & PMiscIC) and interrupt mask register (PRXTXThIM, PRXTXIM & PMiscIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.
7:0	RXQueueAccessEnable[7:0]	RW 0x0	RX Queues Access Enable. It contains 1 bit per queue as following: bit 0 for queue no. 0, bit 1 for queue no. 1 etc. Each bit controls access of CPU#2 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC & PRXTXIC) and interrupt mask register (PRXTXThIM & PRXTXIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.

Table 721: Port CPU3 to Queue (PCP32Q) Register (i=0–3)

Offset: Port2: 0x0003254C, Port3: 0x0003654C, Port0: 0x0007254C, Port1: 0x0007654C

Offset Formula: 0x0007254C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register **NOTE:** This register is used only for multi-cpu devices			
31:16	Reserved	RO 0x0	Reserved

Table 721: Port CPU3 to Queue (PCP32Q) Register (i=0–3) (Continued)

Offset: Port2: 0x0003254C, Port3: 0x0003654C, Port0: 0x0007254C, Port1: 0x0007654C

Offset Formula: 0x0007254C+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:8	TXQueueAccessEnable[7:0]	RW 0x0	TX Queues Access Enable. It contains 1 bit per queue as following: bit 8 for queue no. 0, bit 9 for queue no. 1 etc. Each bit controls access of CPU#3 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC, PRXTXIC & PMiscIC) and interrupt mask register (PRXTXThIM, PRXTXIM & PMiscIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.
7:0	RXQueueAccessEnable[7:0]	RW 0x0	RX Queues Access Enable. It contains 1 bit per queue as following: bit 0 for queue no. 0, bit 1 for queue no. 1 etc. Each bit controls access of CPU#3 to bits of the coresponding TX queue in the interrupt cause registers (PRXTXThIC & PRXTXIC) and interrupt mask register (PRXTXThIM & PRXTXIM): 0 = Access disabled: The CPU access of the bits of this queue is disabled. During a read access "0" is returned for these bits. 1 = Access enabled: The CPU access to these bits is enabled.

Table 722: Port RX Interrupt Threshold (PRXITTH)<q> Register (i=0–3, q=0–7)

Offset: Port2RX Queue0: 0x00032580, Port2RX Queue1: 0x00032584...Port1RX Queue7: 0x0007659C

Offset Formula: 0x00072580+4 * q + 0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000 : where i (0-3) represents Port, where q (0-7) represents RX Queue

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:24	Reserved	RO 0x0	Reserved
23:0	RxInterruptTimeThreshold	RW 0x0	Rx Interrupt Time Threshold. This field defines the maximum time (in unit of core clocks) from reception of an RX packet till the corresponding RxBufIntCoalThAlertQ bit in the PRXTXThIC register is set.

Table 723: Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3)

Offset: Port2: 0x000325A0, Port3: 0x000365A0, Port0: 0x000725A0, Port1: 0x000765A0

Offset Formula: 0x000725A0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31	PMiscICSummary	RO 0x0	PMiscIC register summary This bit is a summary bit of all non-masked PMiscIC register bits (excluding the TxErrorQueue[7:0] bits).
30	PTxErrorSummary	RO 0x0	PMiscIC register, TXErrorQueue[7:0] bits summary This bit is a summary bit of all non-masked TXErrorQueue[7:0] bits of the PMiscIC register.
29	PRXTXICSummary	RO 0x0	PRXTXIC register summary This bit is a summary bit of all non-masked PRXTXIC register bits (excluding the register summary bits - bits 29,30,31).
28:24	Reserved	RO 0x0	Reserved
23:16	RxDescriptorThresholdAlertQueue[7:0]	RO 0x0	Rx Descriptor Threshold Alert Priority Queue[7:0] It contains 1 bit per queue as following: bit 16 for queue no. 0, bit 17 for queue no. 1 etc Each bit is set if the number of RX descriptors in corresponding queue, that are not yet used by the Networking Controller (as indicated by NonOccupiedDescriptorsCounter field in the PRXS register), is less then the value of NonOccupiedDescriptorsThreshold field (in the PRXDQTH register). NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU.

Table 723: Port RX_TX Threshold Interrupt Cause (PRXTXThIC) Register (i=0–3) (Continued)

Offset: Port2: 0x000325A0, Port3: 0x000365A0, Port0: 0x000725A0, Port1: 0x000765A0

Offset Formula: 0x000725A0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
15:8	RxBufferInterruptCoaliascingThresholdAlertPriorityQueue[7:0]	RO 0x0	Rx Buffer Interrupt Coaliascing Threshold Alert Priority Queue[7:0] It contains 1 bit per queue as following: bit 8 for queue no. 0, bit 9 for queue no. 1 etc Each bit, when set indicates that in the corresponding RX queue one of the following is true: 1. The number of buffers in this queue, with received packets that where not yet processed by CPU (as indicated by OccupiedDescriptorsCounter field in the PRXS register), is larger then the value of the OccupiedDescriptorsThreshold field (in the PRXDQTH register). 2. Following that OccupiedDescriptorsCounter==0 or the CPU updated the OccupiedDescriptorsCounter (by accessing the PRXSU register), the time during which OccupiedDescriptorsCounter!=0 is larger then the value of InterruptTimeThreshold field of the PRXITTH register. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU.
7:0	Tx BufferThresholdCrossQueue[7:0]	RO 0x0	Tx Buffer Threshold Cross in Priority Queues[7:0] It contains 1 bit per queue as following: bit 0 for queue no. 0, bit 1 for queue no. 1 etc. Each bit, when set indicates that in the coresponding TX queue, the number of buffers that were transmitted by the Networking Controller and not yet released by CPU, as reflected by the TransmittedBuffersCounter field in the PTXS register, is larger than the value of TransmittedBuffersThreshold field (in the PTXDQS register). NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU.

Table 724: Port RX_TX Threshold Interrupt Mask (PRXTXThIM) Register (i=0–3)

Offset: Port2: 0x000325A4, Port3: 0x000365A4, Port0: 0x000725A4, Port1: 0x000765A4

Offset Formula: 0x000725A4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:0	RxTxThresholdMask	RW 0x0	RX & TX Threshold Interrupt Cause register mask. This field enables/disables interrupt signal generation, in case that the corresponding bit in the PRXTXThIC register is set. **NOTE:** For reserved bits in the interrupt cause register, must write 0. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0, and cannot be overwritten. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU. 0 = Disabled: Interrupt generation is disabled 1 = Enabled: Interrupt generation is enabled

Table 725: Port RX_TX Interrupt Cause (PRXTXIC) Register (i=0–3)

Offset: Port2: 0x000325A8, Port3: 0x000365A8, Port0: 0x000725A8, Port1: 0x000765A8

Offset Formula: 0x000725A8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31	PMiscICSummary	RO 0x0	PMiscIC register Summary This bit is a summary bit of all non-masked PMiscIC register bits (excluding the TxErrorQueue[7:0] bits).
30	PTxErrorSummary	RO 0x0	PMiscIC register, TXErrorQueue[7:0] bits summary This bit is a summary bit of all non-masked TXErrorQueue[7:0] bits of the PMiscIC register.
29	PRXTXThICSummary	RO 0x0	PRXTXThIC register summary This bit is a summary bit of all non-masked PRXTXThIC register bits (excluding the register summary bits - bits 29,30,31).
28:24	Reserved	RO 0x0	Reserved

Table 725: Port RX_TX Interrupt Cause (PRXTXIC) Register (i=0–3) (Continued)

Offset: Port2: 0x000325A8, Port3: 0x000365A8, Port0: 0x000725A8, Port1: 0x000765A8

Offset Formula: 0x000725A8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
23:16	RxResourceErrorQueue[7:0]	RW0C 0x0	Rx Resource Error in Queue[7:0] It contains 1 bit per queue as following: bit 16 for queue no. 0, bit 17 for queue no. 1 etc Each bit, if set, indicates a Rx resource error event in the corresponding priority queue. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0, and cannot be cleared. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU.
15:8	RxPacketQueue[7:0]	RW0C 0x0	Rx Packet in Priority Queue[7:0] It contains 1 bit per queue as following: bit 8 for queue no. 0, bit 9 for queue no. 1 etc Each bit is set following reception of packet in the corresponding queue. **NOTE:** This bit is set only if enabled by the coresponding bit in the RxPktIntrptEnb field of the PIntEnb register. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0, and cannot be cleared. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU.
7:0	TXBufferReturnQueue[7:0]	RW0C 0x0	Tx Buffer Return in Priority Queue[7:0] It contains 1 bit per queue as following: bit 0 for queue no. 0, bit 1 for queue no. 1 etc Each bit, if set, indicates that for the corresponding queue number, the port finished transmission of data of a packet or a buffer was released to the BM unit. **NOTE:** This bit is set only if enabled by the coresponding bit in the TxPktIntrptEnb field of the PIntEnb register. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0, and cannot be cleared. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU.

Table 726: Port RX_TX Interrupt Mask (PRXTXIM) Register (i=0–3)

Offset: Port2: 0x000325AC, Port3: 0x000365AC, Port0: 0x000725AC, Port1: 0x000765AC

Offset Formula: 0x000725AC+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:0	RxTxMask	RW 0x0	RX & TX Interrupt Cause register mask. This field enables/disables interrupt signal generation, in case that the corresponding bit in the PRXTXIC register is set. **NOTE:** For reserved bits in the interrupt cause register, must write 0. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0, and cannot be cleared. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU. 0 = Disabled: Interrupt generation disabled 1 = Enabled: Interrupt generation enabled

Table 727: Port Misc Interrupt Cause (PMiscIC) Register (i=0–3)

Offset: Port2: 0x000325B0, Port3: 0x000365B0, Port0: 0x000725B0, Port1: 0x000765B0

Offset Formula: 0x000725B0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:24	TXResourceErrorQueue[7:0]	RW0C 0x0	Tx Resource Error in Queue[7:0] It contains 1 bit per queue as following: bit 24 for queue no. 0, bit 25 for queue no. 1 etc Each bit, if set, indicates a Tx resource error event in the corresponding priority queue. Resource error is defined as parity error during descriptor fetch. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0, and cannot be cleared. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU.
23:20	Reserved	RO 0x0	Reserved
19:16	Reserved	RSVD 0x0	Reserved

Table 727: Port Misc Interrupt Cause (PMiscIC) Register (i=0–3) (Continued)

Offset: Port2: 0x000325B0, Port3: 0x000365B0, Port0: 0x000725B0, Port1: 0x000765B0

Offset Formula: 0x000725B0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
19:16	RxNoBuffersToPool[3:0]	RW0C 0x0	RX No Buffers to Pool[3:0] It contains 1 bit per pool as following: bit 16 for pool no. 0, bit 17 for pool no. 1 etc Each bit, if set, indicates that the BM unit failed during software forwarding, to allocate buffers from the corresponding pool. **NOTE:** These bits may be asserted only when buffer management by BM unit is selected by "AccelerationMode" field in the PACC register. **NOTE:**
15	Reserved	RO 0x0	Reserved
14	SerdesRealignSyncError	RW0C 0x0	Serdes Realign Sync Error. When set, this bit indicates a Serdes realign sync error detection.
13	Reserved	RW0C 0x0	Reserved. Must write 0.
12	PRBSError	RW0C 0x0	PRBS Error. This bit is asserted when PRBS checker is enabled and an error was detected.
11	TxUndrn	RW0C 0x0	TX FIFO Underrun When set, this bit indicates that an TX FIFO underrun error event was detected during transmission of a packet.
10	RXLargePacket	RW0C 0x0	RX Large Packet. When set, this bit indicates that a packet larger then maximum legal packet size, as defined in the FrameSizeLimit field of the Port Mac Control Register0 was accepted.
9	RXCRCError	RW0C 0x0	RX CRC Error. When set, this bit indicates that an RX packet CRC error was detected.
8	RXOverrun	RW0C 0x0	RX Fifo Overrun. When set, this bit indicates that an RX FIFO overrun error was detected.
7	InternalAddressError	RW0C 0x0	Internal Address Error This bit is set when an access to an illegal offset of the internal registers is identified. When set, the Internal Address Error register locks the address that caused the error. **NOTE:** This bit is not set for accesses to all illegal offsets.

Table 727: Port Misc Interrupt Cause (PMiscIC) Register (i=0–3) (Continued)

Offset: Port2: 0x000325B0, Port3: 0x000365B0, Port0: 0x000725B0, Port1: 0x000765B0

Offset Formula: 0x000725B0+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
6	Reserved	RSVD 0x0	Reserved
5	Reserved	RO 0x0	Reserved
4	PTP	RW0C 0x0	PTP identification This bit is set following a PTP event, like PTP packet identification. Before clearing this bit, make sure that the following bits are not set: - TrigGenInt bit in the TAI Global Status0 Register - EventInt bit in the TAI Global Status1 Register - PTPInt bits in the PTP Global Status0 Register
4	Reserved	RSVD 0x0	Reserved
3:2	Reserved	RO 0x0	Reserved
1	LinkChange	RW0C 0x0	Link Status Change This bit is set following detection of the link status change.
0	PhyStatusChng	RW0C 0x0	Phy Status Change Indicates a status change reported by the PHY connected to this port. This bit will be set for any change in the link, speed, duplex mode, or flow control capability as reported by the PHY, via the MDIO or SGMII interface. NOTES: 1. Speed change identified via the SGMII interface is reported only if the AnSpeedEN bit of the Port Auto-Negotiation Configuration register is set. 2. Speed change identified via the MDIO interface is reported only if the following conditions are true: a. The AnSpeedEN bit of the Port Auto-Negotiation Configuration register is set. b. The link is up. c. Auto-Negotiation was completed.

Table 728: Port Misc Interrupt Mask (PMiscIM) Register (i=0–3)

Offset: Port2: 0x000325B4, Port3: 0x000365B4, Port0: 0x000725B4, Port1: 0x000765B4

Offset Formula: 0x000725B4+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:0	MiscMask	RW 0x0	Misc Interrupt Cause register mask. This field enables/disables interrupt signal generation, in case that the corresponding bit in the PMiscIC register is set. **NOTE:** For reserved bits in the interrupt cause register, must write 0. NOTE: In a multi-CPU system, bits of RX queues for which the access by the reading CPU is disabled are read as 0, and cannot be overwritten. The access to the queue is configured by the PCP02Q, PCP12Q, PCP22Q, or PCP32Q registers that correspond to the reading CPU. 0 = Disabled: Interrupt generation is disabled 1 = Enabled: Interrupt generation is enabled

Table 729: Port Interrupt Enable (PIntEnb) Register (i=0–3)

Offset: Port2: 0x000325B8, Port3: 0x000365B8, Port0: 0x000725B8, Port1: 0x000765B8

Offset Formula: 0x000725B8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
NOTE: This register is valid only in all Enhanced modes, as defined by the Port Acceleration mode (PACC) register			
31:16	Reserved	RO 0x0	Reserved
15:8	TxPktIntrptEnb	RW 0x0	TX Packet Interrupt Enable: In this field, there is 1 bit/TX queue, that controls interrupt generation transmission of a packet in this TX queue. Actually it enables setting of the TxBufferQueue field in the PRXTXIC register. 0 = Disabled: Interrupt generation following packet transmission in this queue is disabled. 1 = Enabled: Interrupt generation following packet transmission in this queue is enabled.

Table 729: Port Interrupt Enable (PIntEnb) Register (i=0–3) (Continued)

Offset: Port2: 0x000325B8, Port3: 0x000365B8, Port0: 0x000725B8, Port1: 0x000765B8

Offset Formula: 0x000725B8+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
7:0	RxPktIntrptEnb	RW 0x0	RX Packet Interrupt Enable: In this field, there is 1 bit/RX queue, that controls interrupt generation following reception of a packet in this RX queue. Actually it enables setting of the RxPacketQueue field in the PRXTXIC register. 0 = Disabled: Interrupt generation following packet reception in this queue is disabled. 1 = Enabled: Interrupt generation following packet reception in this queue is enabled

A.6.24 Miscellaneous Interrupt Registers

Table 730: Port Internal Address Error (EUIAE) Register (i=0–3)

Offset: Port2: 0x00032494, Port3: 0x00036494, Port0: 0x00072494, Port1: 0x00076494

Offset Formula: 0x00072494+0x4000*(i%2)+floor((3-i) / 2)*0x40000 - 0x40000: where i (0-3) represents Port

Bit	Field	Type/InitVal	Description
31:14	Reserved	RO 0x0	Reserved
13:0	InternalAddress	RO 0x0	Locks bits 15:2 of the internal address bits, if there is an address violation of unmapped access to the port registers. The Address is locked until the register is read.

A.6.25 Gigabit Ethernet MAC MIB Counters

Table 731: Gigabit Ethernet MAC MIB Counters

Offset: Port0: 0x73000–0x7307C, Port1: 0x77000–0x7707C

Offset	Width	Counter Name	Description
0x0	64	GoodOctetsReceived	The sum of lengths of all good Ethernet frames received: frames that are not Bad frames NOR MAC Control frames. **NOTE:** This does *not* include IEEE 802.3x pause messages, but, does include bridge control packets like LCAP and BPDU.
0x8	32	BadOctetsReceived	The sum of lengths of all bad Ethernet frames received
0x10	32	GoodFramesReceived	The number of Ethernet frames received that are not Bad Ethernet frames or MAC Control packets. **NOTE:** This does include Bridge Control packets.
0xC	32	MACTransError	The number of frames not transmitted correctly or dropped due to internal MAC transmit error, for example, underrun.
0x14	32	BadFramesReceived	The number of bad Ethernet frames received
0x18	32	BroadcastFramesReceived	The number of good frames received that had a Broadcast destination MAC address
0x1C	32	MulticastFramesReceived	The number of good frames received that had a Multicast destination MAC address **NOTE:** This does *not* include IEEE 802.3 Flow Control messages as they are considered MAC Control messages.
0x20	32	Frames64Octets	The total number of received and transmitted, Good and Bad frames that are 64 bytes in size or are between the minimum-size (as specified in RxMFS in the PxMFS register) and 64 bytes. **NOTE:** This does *not* include MAC Control frames.
0x24	32	Frames65to127Octets	The total number of received and transmitted, Good and Bad frames that are 65 to 127 bytes in size. **NOTE:** This does *not* include MAC Control frames.
0x28	32	Frames128to255Octets	The total number of received and transmitted, Good and Bad frames that are 128 to 255 bytes in size. **NOTE:** This does *not* include MAC Control frames.
0x2C	32	Frames256to511Octets	The total number of received and transmitted, Good and Bad frames that are 256 to 511 bytes in size. **NOTE:** This does *not* include MAC Control frames.
0x30	32	Frames512to1023Octets	The total number of received and transmitted, Good and Bad frames that are 512 to 1023 bytes in size. **NOTE:** This does *not* include MAC Control frames.
0x34	32	Frames1024toMaxOctets	The total number of received and transmitted, Good and Bad frames that are more than 1023 bytes in size and less than the MRU. **NOTE:** This does not include MAC Control frames.
0x38	64	GoodOctetsSent	The sum of lengths of all good Ethernet frames sent from this MAC. This does not include IEEE 802.3 Flow Control frames NOR packets dropped due to excessive collision NOR packets with an Tx Error Event.

Table 731: Gigabit Ethernet MAC MIB Counters (Continued)
Offset: Port0: 0x73000–0x7307C, Port1: 0x77000–0x7707C

Offset	Width	Counter Name	Description
0x40	32	GoodFramesSent	The number of Ethernet frames sent from this MAC. This does not include IEEE 802.3 Flow Control frames NOR packets dropped due to excessive collision NOR packets with an Tx Error Event.
0x44	32	ExcessiveCollision	The number of frames dropped in the transmit MAC due to excessive collision condition. This counter will not be incremented if the PSCR's <Retr_forever> bit is set. This is applicable for half-duplex mode only.
0x48	32	MulticastFramesSent	The number of good frames sent that had a Multicast destination MAC address. **NOTE:** This does NOT include IEEE 802.3 Flow Control messages, as they are considered MAC Control messages, NOR does it include packets with an Tx Error Event. This counter counts frames of all sizes, including frames smaller than the Minimal Frame Size and frames larger than the MRU.
0x4C	32	BroadcastFramesSent	The number of good frames sent that had a Broadcast destination MAC address. This does not include IEEE 802.3 Flow Control frames NOR packets dropped due to excessive collision NOR packets with an Tx Error Event. **NOTE:** This counter counts frames of all sizes, including frames smaller than the Minimal Frame Size and frames larger than the MRU.
0x50	32	UnrecogMACControl Received	The number of received MAC Control frames that have an opcode different than 00-01.
0x58	32	GoodFCReceived	The number of good flow control messages received
0x5C	32	BadFCReceived	The number of bad flow control frames received
0x60	32	Undersize	The number of undersize packets received
0x54	32	FCSent	The number of flow control frames sent
0x64	32	Fragments	The number of fragments received
0x68	32	Oversize	The number of oversize packets received
0x6C	32	Jabber	The number of jabber packets received
0x70	32	MACRcvError	The number of Rx Error events seen by the receive side of the MAC
0x74	32	BadCRC	The number CRC error events
0x78	32	Collisions	The number of collision events seen by the MAC
0x7C	32	Late Collision	The number of late collisions seen by the MAC

Note

The 802dot3SingleCollisionFrames and 802dot3MultipleCollisionFrames are not implemented.

A.7 Buffer Management Controller Registers

The following table provides a summarized list of all registers that belong to the Buffer Management Controller, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 732: Summary Map Table for the Buffer Management Controller Registers

Register Name	Offset	Table and Page
BMCR Register	0x000C0000	Table 733, p. 1036
BMACTR Register	0x000C0004	Table 734, p. 1037
BMXR_P01 Register	0x000C0008	Table 735, p. 1038
BMXR_P23 Register	0x000C000C	Table 736, p. 1038
BMBPPAR_0 Register	0x000C0010	Table 737, p. 1039
BMBPPRP_0 Register	0x000C0014	Table 738, p. 1039
BMBPPWP_0 Register	0x000C0018	Table 739, p. 1040
BMBPPSR_0 Register	0x000C001C	Table 740, p. 1040
BMBPPAR_1 Register	0x000C0020	Table 741, p. 1041
BMBPPRP_1 Register	0x000C0024	Table 742, p. 1041
BMBPPWP_1 Register	0x000C0028	Table 743, p. 1041
BMBPPSR_1 Register	0x000C002C	Table 744, p. 1042
BMBPPAR_2 Register	0x000C0030	Table 745, p. 1042
BMBPPRP_2 Register	0x000C0034	Table 746, p. 1043
BMBPPWP_2 Register	0x000C0038	Table 747, p. 1043
BMBPPSR_2 Register	0x000C003C	Table 748, p. 1043
BMBPPAR_3 Register	0x000C0040	Table 749, p. 1044
BMBPPRP_3 Register	0x000C0044	Table 750, p. 1044
BMBPPWP_3 Register	0x000C0048	Table 751, p. 1045
BMBPPSR_3 Register	0x000C004C	Table 752, p. 1045
BMICR Register	0x000C0050	Table 753, p. 1045
BMIMR Register	0x000C0054	Table 754, p. 1047
BMABPS_0 Register	0x000C008C	Table 755, p. 1049
BMABPS_1 Register	0x000C0090	Table 756, p. 1049
BMABPS_2 Register	0x000C0094	Table 757, p. 1050
BMABPS_3 Register	0x000C0098	Table 758, p. 1050
BMRBPS_0 Register	0x000C009C	Table 759, p. 1050
BMRBPS_1 Register	0x000C00A0	Table 760, p. 1050
BMRBPS_2 Register	0x000C00A4	Table 761, p. 1051
BMRBPS_3 Register	0x000C00A8	Table 762, p. 1051

Copyright © 2014 Marvell

Table 733: BMCR Register

Offset: 0x000C0000

Bit	Field	Type/InitVal	Description
BM Configuration Register			
31:19	Reserved	RSVD 0x0	Reserved
18:17	MaxInBurstSize	RW 0x2	Define the maximum possible size of allocate & release request bursts driven towards the BM. 0 = 32 BPs 1 = 24 BPs 2 = 16 BPs 3 = 8 BPs
16	BppiFullMode	RW 0x0	Must be zero.
15:12	BppiHighThreshold	RW 0xb	HighThreshold of BPPI<i>, triggering BPs Pushing towards BPPE from BM internal memory. The threshold is (<BHT>+1) x 16 BPs. Maximal value depends upon the value configured to MaxInBurstSize : When MaxInBurstSize = 0, maximal value is 10; When MaxInBurstSize = 1, maximal value is 11; When MaxInBurstSize = 2, maximal value is 12; When MaxInBurstSize = 3, maximal value is 13 is 12, or if MaxInBurstSize is 2 or 3 - 13. Must be larger at least by 3 than the value of BLT.
11:8	BppiLowThreshold	RW 0x5	Low Threshold of BPPI<i>, triggering BPs Pulling from BPPE into BM internal memory. The threshold is (<BLT>+1) x 16 BPs. Must be at least 4, or if MaxInBurstSize is 2 or 3 - at least 3. Must be smaller at least by 3 than the value of BHT.
7:6	Reserved	RSVD 0x0	Reserved

5	SrcSwap	RW
		0x0
			Endianess Swap control of incoming data towards BM. Value Must be equal to DestSwap.
			If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc.
			0 = No Swap;
			1 = Swap;

Table 733: BMCR Register (Continued)
Offset: 0x000C0000

Bit	Field	Type/InitVal	Description

4	DestSwap	RW
		0x0
			Endianess Swap control of data going out of BM. Value Must be equal to SrcSwap.
			If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc.
			0 = No Swap;
			1 = Swap;

3:2	DstBurstSize	RW 0x0	Burst size of write requests access (push commands) that the BM generates over the Internal Mbus 0 = 128 Bytes 1 = 32 Bytes 2 = Reserved 2 3 = Reserved 3
1:0	SrcBurstSize	RW 0x0	Burst size of read requests access (pull commands) that the BM generates over the Internal Mbus. The returning pull response accesses will be in this size. 0 = 128 Bytes 1 = 32 Bytes 2 = Reserved 2 3 = Reserved 3

Table 734: BMACTR Register
Offset: 0x000C0004

Bit	Field	Type/InitVal	Description
BM Activation Register			
31:6	Reserved	RSVD 0x0	Reserved

5:4	BmStatus	RO
		0x0
			Reflects BM activity status:
			0 = not active;
			1 = active;
			2 = paused;
			3 = Reserved;

3	Reserved	RSVD 0x0	Reserved
2	BmPause	RW 0x0	BM activity put on hold. Setting this bit pauses the BM operation. BM does not perfrom BP releasing & returns null BPs as respond to allocate requests. BM will suspend at the earliest opportunity. BM enters Pause condition only if BmStart & BmStop are in inactive. BM should be placed in pause condition only if it was formerly active.

Table 734: BMACTR Register (Continued)

Offset: 0x000C0004

Bit	Field	Type/InitVal	Description
1	BmStop	RW 0x0	BM deactivation. Operation is stopped. BM is reset to its pre-start configuration. BM enters Stop condition only if BmStart in inactive.
0	BmStart	RW 0x0	BM activation. Setting this bit activates the entire BM, regardless of BmStop & BmPause bits. Yet each BP Pool is activated by BMBPPAR<BPP_Enable>. Used also to return back from stop and pause states into active state.

Table 735: BMXR_P01 Register

Offset: 0x000C0008

Bit	Field	Type/InitVal	Description
BM Xbar interface			
31:28	Reserved	RSVD 0x0	Reserved
27:20	XbarAttr_P1	RW 0xfe	Specifies contents of Xbar attributes for the target when driving Xbar transactions of pool #1 over the Xbar. By default - DRAM controller attributes.
19:16	TrgtID_P1	RW 0x1	Specifies the target interface over Xbar by its Unit ID for pool #1 transactions. By default - DRAM controller.
15:12	Reserved	RSVD 0x0	Reserved
11:4	XbarAttr_P0	RW 0xfe	Specifies contents of Xbar attributes for the target when driving Xbar transactions of pool #0 over the Xbar. By default - DRAM controller attributes.
3:0	TrgtID_P0	RW 0x1	Specifies the target interface over Xbar by its Unit ID for pool #0 transactions. By default - DRAM controller.

Table 736: BMXR_P23 Register

Offset: 0x000C000C

Bit	Field	Type/InitVal	Description
BM Xbar interface			
31:28	Reserved	RSVD 0x0	Reserved

Table 736: BMXR_P23 Register (Continued)
Offset: 0x000C000C

Bit	Field	Type/InitVal	Description
27:20	XbarAttr_P3	RW 0xfe	Specifies contents of Xbar attributes for the target when driving Xbar transactions of pool #3 over the Xbar. By default - DRAM controller attributes.
19:16	TrgtID_P3	RW 0x1	Specifies the target interface over Xbar by its Unit ID for pool #3 transactions. By default - DRAM controller.
15:12	Reserved	RSVD 0x0	Reserved
11:4	XbarAttr_P2	RW 0xfe	Specifies contents of Xbar attributes for the target when driving Xbar transactions of pool #2 over the Xbar. By default - DRAM controller attributes.
3:0	TrgtID_P2	RW 0x1	Specifies the target interface over Xbar by its Unit ID for pool #2 transactions. By default - DRAM controller.

Table 737: BMBPPAR_0 Register
Offset: 0x000C0010

Bit	Field	Type/InitVal	Description
Address of External Buffer Pointers Pool #0			
31:2	BPPA_0	RW 0x0	Address pointer for the BM External Buffer Pointers Pool, Bppa_0[31:2]. Bits [1:0] of the address are considered 00.
1	Reserved	RSVD 0x0	Reserved
0	BPP_Enable_0	RW 0x0	Enable the usage of BP Pool #0. 1 = Enable, 0 = Disable

Table 738: BMBPPRP_0 Register
Offset: 0x000C0014

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool RD pointer, pool #0			
31:18	BppRdPtrCurr_0	RO 0x0	Address reflection of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #0 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved

Table 738: BMBPPRP_0 Register (Continued)

Offset: 0x000C0014

Bit	Field	Type/InitVal	Description
15:2	BppRdPtr_0	RW 0x0	Address of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #0 for SW Initialization. Actual address is this address + base address in BMBPPAR_0. Bits [1:0] are considered 00.
1:0	Reserved	RSVD 0x0	Reserved

Table 739: BMBPPWP_0 Register

Offset: 0x000C0018

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool WR pointer, pool #0			
31:18	BppWrPtrCurr_0	RO 0x0	Address reflection of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #0 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved
15:2	BppWrPtr_0	RW 0x0	Address of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #0, for SW Initialization. Actual address is this address + base address in BMBPPAR_0. Bits [1:0] are considered 00. Value of BppWrPtr_i should be smaller at least by 8 than the value of BppSize_i.
1:0	Reserved	RSVD 0x0	Reserved

Table 740: BMBPPSR_0 Register

Offset: 0x000C001C

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool Size, pool #0			
31:14	Reserved	RSVD 0x0	Reserved
13:0	BppSize_0	RW 0x100	Size of External Buffer Pointers Pool #0, in number of BPs. Actual size in bytes is 4 times larger as each BP is 4B. Minimum size is 128 BPs. The last address of the BPPE is BppSize[13:0] x4 - 1.

Table 741: BMBPPAR_1 Register

Offset: 0x000C0020

Bit	Field	Type/InitVal	Description
Address of External Buffer Pointers Pool #1			
31:2	BPPA_1	RW 0x0	Address pointer for the BM External Buffer Pointers Pool, Bppa_1[31:2]. Bits [1:0] of the address are considered 00.
1	Reserved	RSVD 0x0	Reserved
0	BPP_Enable_1	RW 0x0	Enable the usage of BP Pool #1. 1 = Enable, 0 = Disable

Table 742: BMBPPRP_1 Register

Offset: 0x000C0024

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool RD pointer, pool #1			
31:18	BppRdPtrCurr_1	RO 0x0	Address reflection of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #1 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved
15:2	BppRdPtr_1	RW 0x0	Address of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #1 for SW Initialization. Actual address is this address + base address in BMBPPAR_1. Bits [1:0] are considered 00.
1:0	Reserved	RSVD 0x0	Reserved

Table 743: BMBPPWP_1 Register

Offset: 0x000C0028

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool WR pointer, pool #1			
31:18	BppWrPtrCurr_1	RO 0x0	Address reflection of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #1 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved

Table 743: BMBPPWP_1 Register (Continued)

Offset: 0x000C0028

Bit	Field	Type/InitVal	Description
15:2	BppWrPtr_1	RW 0x0	Address of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #1, for SW Initialization. Bits[31:16] equal to bits [31:16] of the base address - BPPAR. Bits [1:0] are considered 00. Value of BppWrPtr_i should be smaller at least by 8 than the value of BppSize_i.
1:0	Reserved	RSVD 0x0	Reserved

Table 744: BMBPPSR_1 Register

Offset: 0x000C002C

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool Size, pool #1			
31:14	Reserved	RSVD 0x0	Reserved
13:0	BppSize_1	RW 0x100	Size of External Buffer Pointers Pool #1, in number of BPs. Actual size in bytes is 4 times larger as each BP is 4B. Minimum size is 128 BPs. The last address of the BPPE is BppSize[13:0] x4 - 1.

Table 745: BMBPPAR_2 Register

Offset: 0x000C0030

Bit	Field	Type/InitVal	Description
Address of External Buffer Pointers Pool #1			
31:2	BPPA_2	RW 0x0	Address pointer for the BM External Buffer Pointers Pool, Bppa_2[31:2]. Bits [1:0] of the address are considered 00.
1	Reserved	RSVD 0x0	Reserved
0	BPP_Enable_2	RW 0x0	Enable the usage of BP Pool #2. 1 = Enable, 0 = Disable

Table 746: BMBPPRP_2 Register

Offset: 0x000C0034

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool RD pointer, pool #2			
31:18	BppRdPtrCurr_2	RO 0x0	Address reflection of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #2 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved
15:2	BppRdPtr_2	RW 0x0	Address of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #2 for SW Initialization. Actual address is this address + base address in BMBPPAR_2. Bits [1:0] are considered 00.
1:0	Reserved	RSVD 0x0	Reserved

Table 747: BMBPPWP_2 Register

Offset: 0x000C0038

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool WR pointer, pool #2			
31:18	BppWrPtrCurr_2	RO 0x0	Address reflection of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #2 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved
15:2	BppWrPtr_2	RW 0x0	Address of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #2, for SW Initialization. Actual address is this address + base address in BMBPPAR_2.. Bits [1:0] are considered 00. Value of BppWrPtr_i should be smaller at least by 8 than the value of BppSize_i.
1:0	Reserved	RSVD 0x0	Reserved

Table 748: BMBPPSR_2 Register

Offset: 0x000C003C

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool Size, pool #2			
31:14	Reserved	RSVD 0x0	Reserved

Table 748: BMBPPSR_2 Register (Continued)

Offset: 0x000C003C

Bit	Field	Type/InitVal	Description
13:0	BppSize_2	RW 0x100	Size of External Buffer Pointers Pool #2, in number of BPs.. Actual size in bytes is 4 times larger as each BP is 4B. Minimum size is 128 BPs. The last address of the BPPE is BppSize[13:0] x4 - 1.

Table 749: BMBPPAR_3 Register

Offset: 0x000C0040

Bit	Field	Type/InitVal	Description
Address of External Buffer Pointers Pool #3			
31:2	BPPA_3	RW 0x0	Address pointer for the BM External Buffer Pointers Pool, Bppa_3[31:2]. Bits [1:0] of the address are considered 00.
1	Reserved	RSVD 0x0	Reserved
0	BPP_Enable_3	RW 0x0	Enable the usage of BP Pool #3. 1 = Enable, 0 = Disable

Table 750: BMBPPRP_3 Register

Offset: 0x000C0044

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool RD pointer, pool #3			
31:18	BppRdPtrCurr_3	RO 0x0	Address reflection of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #3 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved
15:2	BppRdPtr_3	RW 0x0	Address of the RD pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #3 for SW Initialization. Actual address is this address + base address in BMBPPAR_3. Bits [1:0] are considered 00.
1:0	Reserved	RSVD 0x0	Reserved

Table 751: BMBPPWP_3 Register

Offset: 0x000C0048

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool WR pointer, pool #3			
31:18	BppWrPtrCurr_3	RO 0x0	Address reflection of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #3 during BM process. Bits [1:0] are considered 00.
17:16	Reserved	RSVD 0x0	Reserved
15:2	BppWrPtr_3	RW 0x0	Address of the WR pointer of the External Buffer Pointers Pool, bits<15:2> of BPP #3, for SW Initialization. Actual address is this address + base address in BMBPPAR_3. Bits [1:0] are considered 00. Value of BppWrPtr_i should be smaller at least by 8 than the value of BppSize_i.
1:0	Reserved	RSVD 0x0	Reserved

Table 752: BMBPPSR_3 Register

Offset: 0x000C004C

Bit	Field	Type/InitVal	Description
External Buffer Pointers Pool Size, pool #3			
31:14	Reserved	RSVD 0x0	Reserved
13:0	BppSize_3	RW 0x100	Size of External Buffer Pointers Pool #3, in number of BPs. Actual size in bytes is 4 times larger as each BP is 4B. Minimum size is 128 BPs. The last address of the BPPE is BppSize[13:0] x4 - 1.

Table 753: BMICR Register

Offset: 0x000C0050

Bit	Field	Type/InitVal	Description
BM Interrupt Cause			
31	BmIntrSum	RO 0x0	BM Unit Interrupt Summary This bit is a logical OR of the unmasked bits in the BMICR register.
30	BmPause	RW0C 0x0	BM completed Pause routine, after receiving pause command (by BMpause). Entered paused state.

Table 753: BMICR Register (Continued)
Offset: 0x000C0050

Bit	Field	Type/InitVal	Description
29	BmStop	RW0C 0x0	BM completed stop routine, after receiving stop command (by BMstop). Entered Inactive state
28	XbarParErr	RW0C 0x0	Mbus Parity Error Caused by erroneous read response from the Mbus.
27	IntParErr	RW0C 0x0	Parity Error Caused by erroneous internal buffer read.
26:23	Reserved	RSVD 0x0	Reserved
22	AvailableBpLow_3	RW0C 0x0	The sum of available BPs in BPPE_3 and BPPI_3 is below the configured threshold.
21	BppeFull_3	RW0C 0x0	External BP Pool #3 has become full.
20	BppeEmpty_3	RW0C 0x0	External BP Pool #3 has become empty.
19	AllocFail_3	RW0C 0x0	BPs allocation failed as no BPs were available in BPPI #3. Null address was supplied to requesting agent.
18	RlsFail_3	RW0C 0x0	BPs release by agent delayed since BPPI #3 was full.
17	Reserved	RSVD 0x0	Reserved
16	AvailableBpLow_2	RW0C 0x0	The sum of available BPs in BPPE_2 and BPPI_2 is below the configured threshold.
15	BppeFull_2	RW0C 0x0	External BP Pool #2 has become full.
14	BppeEmpty_2	RW0C 0x0	External BP Pool #2 has become empty.
13	AllocFail_2	RW0C 0x0	BPs allocation failed as no BPs were available in BPPI #2. Null address was supplied to requesting agent.
12	RlsFail_2	RW0C 0x0	BPs release by agent delayed since BPPI #2 was full.
11	Reserved	RSVD 0x0	Reserved
10	AvailableBpLow_1	RW0C 0x0	The sum of available BPs in BPPE_1 and BPPI_1 is below the configured threshold.

Table 753: BMICR Register (Continued)

Offset: 0x000C0050

Bit	Field	Type/InitVal	Description
9	BppeFull_1	RW0C 0x0	External BP Pool #1 has become full.
8	BppeEmpty_1	RW0C 0x0	External BP Pool #1 has become empty.
7	AllocFail_1	RW0C 0x0	BPs allocation failed as no BPs were available in BPPI #1. Null address was supplied to requesting agent.
6	RlsFail_1	RW0C 0x0	BPs release by agent delayed since BPPI #1 was full.
5	Reserved	RSVD 0x0	Reserved
4	AvailableBpLow_0	RW0C 0x0	The sum of available BPs in BPPE_0 and BPPI_0 is below the configured threshold.
3	BppeFull_0	RW0C 0x0	External BP Pool #0 has become full.
2	BppeEmpty_0	RW0C 0x0	External BP Pool #0 has become empty.
1	AllocFail_0	RW0C 0x0	BPs allocation failed as no BPs were available in BPPI #0. Null address was supplied to requesting agent.
0	RlsFail_0	RW0C 0x0	BPs release by agent delayed since BPPI #0 was full.

Table 754: BMIMR Register

Offset: 0x000C0054

Bit	Field	Type/InitVal	Description
BM interrupt Mask			
31	BmIntrSumMask	RW 0x0	BM Unit Interrupt Summary Mask
30	BmPauseMask	RW 0x0	BM Pause Interrupt Mask
29	BmStopMask	RW 0x0	BM Stop Interrupt Mask
28	XbarParErrMask	RW 0x0	Xbar Parity Error Interrupt Mask
27	IntParErrMask	RW 0x0	Parity Error Interrupt Mask

Table 754: BMIMR Register (Continued)
Offset: 0x000C0054

Bit	Field	Type/InitVal	Description
26:23	Reserved	RSVD 0x0	Reserved
22	AvlbBpLowMask_3	RW 0x0	AvailableBpLow_3 Interrupt mask.
21	BppeFullMask_3	RW 0x0	Bppe Full Interrupt Mask for BPPE #3
20	BppeEmptyMask_3	RW 0x0	Bppe Empty Interrupt Mask for BPPE #3
19	AllocFailMask_3	RW 0x0	Allocate failure Interrupt Mask for BPPI #3
18	RlsFailMask_3	RW 0x0	Release failure Interrupt Mask for BPPI #3
17	Reserved	RSVD 0x0	Reserved
16	AvlbBpLowMask_2	RW 0x0	AvailableBpLow_2 Interrupt mask.
15	BppeFullMask_2	RW 0x0	Bppe Full Interrupt Mask for BPPE #2
14	BppeEmptyMask_2	RW 0x0	Bppe Empty Interrupt Mask for BPPE #2
13	AllocFailMask_2	RW 0x0	Allocate failure Interrupt Mask for BPPI #2
12	RlsFailMask_2	RW 0x0	Release failure Interrupt Mask for BPPI #2
11	Reserved	RSVD 0x0	Reserved
10	AvlbBpLowMask_1	RW 0x0	AvailableBpLow_1 Interrupt mask.
9	BppeFullMask_1	RW 0x0	Bppe Full Interrupt Mask for BPPE #1
8	BppeEmptyMask_1	RW 0x0	Bppe Empty Interrupt Mask for BPPE #1
7	AllocFailMask_1	RW 0x0	Allocate failure Interrupt Mask for BPPI #1
6	RlsFailMask_1	RW 0x0	Release failure Interrupt Mask for BPPI #1
5	Reserved	RSVD 0x0	Reserved
4	AvlbBpLowMask_0	RW 0x0	AvailableBpLow_0 Interrupt mask.

Table 754: BMIMR Register (Continued)
Offset: 0x000C0054

Bit	Field	Type/InitVal	Description
3	BppeFullMask_0	RW 0x0	Bppe Full Interrupt Mask for BPPE #0
2	BppeEmptyMask_0	RW 0x0	Bppe Empty Interrupt Mask for BPPE #0
1	AllocFailMask_0	RW 0x0	Allocate failure Interrupt Mask for BPPI #0
0	RlsFailMask_0	RW 0x0	Release failure Interrupt Mask for BPPI #0

Table 755: BMABPS_0 Register
Offset: 0x000C008C

Bit	Field	Type/InitVal	Description
The number of BPs allocated from this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	
29:0	BM_Allocated_BPs_counter	ROC 0x0	The number of BPs allocated from this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

Table 756: BMABPS_1 Register
Offset: 0x000C0090

Bit	Field	Type/InitVal	Description
The number of BPs allocated from this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	
29:0	BM_Allocated_BPs_counter	ROC 0x0	The number of BPs allocated from this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

Table 757: BMABPS_2 Register
Offset: 0x000C0094

Bit	Field	Type/InitVal	Description
The number of BPs allocated from this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	
29:0	BM_Allocated_BPs_counter	ROC 0x0	The number of BPs allocated from this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

Table 758: BMABPS_3 Register
Offset: 0x000C0098

Bit	Field	Type/InitVal	Description
The number of BPs allocated from this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	
29:0	BM_Allocated_BPs_counter	ROC 0x0	The number of BPs allocated from this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

Table 759: BMRBPS_0 Register
Offset: 0x000C009C

Bit	Field	Type/InitVal	Description
The number of BPs released to this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	
29:0	BM_Released_BPs_counter	ROC 0x0	The number of BPs released to this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

Table 760: BMRBPS_1 Register
Offset: 0x000C00A0

Bit	Field	Type/InitVal	Description
The number of BPs released to this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	

Table 760: BMRBPS_1 Register (Continued)

Offset: 0x000C00A0

Bit	Field	Type/InitVal	Description
29:0	BM_Released_BPs_ counter	ROC 0x0	The number of BPs released to this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

Table 761: BMRBPS_2 Register

Offset: 0x000C00A4

Bit	Field	Type/InitVal	Description
The number of BPs released to this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	
29:0	BM_Released_BPs_ counter	ROC 0x0	The number of BPs released to this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

Table 762: BMRBPS_3 Register

Offset: 0x000C00A8

Bit	Field	Type/InitVal	Description
The number of BPs released to this pool since the last read of this register.			
31:30	Reserved	RSVD 0x0	
29:0	BM_Released_BPs_ counter	ROC 0x0	The number of BPs released to this pool since the last read of this register, in resolution of 2 BPs (in case of an odd number of BPs the result will be rounded down)

A.8 Precise Time Protocol (PTP) Registers

The following are the register bits used for configuration and status information to and from the software/CPU sub-system for Precise Time Protocol logic for audio-video bridging (AVB) applications.

Note

- These registers are accessed using the two Global registers:
  - PTP Command Register (Port 0 and 1: 0x7C000, Port 2 and 3: 0x3C000)
  - PTP Data Register (Port 0 and 1: 0x7C008, Port 2 and 3: 0x3C008).
- For the registers in Table 764, Table 765, Table 768–Table 776, Table 778, Table 779, Table 781, Table 782, Table 784–Table 794, bits [31:16] are not implemented and will return undefined values.

Table 763: Summary Map Table for the PTP Registers

Register Name	Offset	Table and Page
PTP Global Core Access Registers		
PTP Command Register	Port 0 and 1: 0x7C000, Port 2 and 3: 0x3C000	Table 764, p. 1053
PTP Data Register	Port 0 and 1: 0x7C008, Port 2 and 3: 0x3C008	Table 765, p. 1054
PTP Configuration Registers		
PTP Reset Register	Port 0 and 1: 0x7C010	Table 766, p. 1055
PTP Clock Select Register	Port 0 and 1: 0x7C018	Table 767, p. 1055
PTP Global Configuration Registers		
PTP Global Configuration0 Register	0x0, or Decimal 0	Table 768, p. 1056
PTP Global Configuration1 Register	0x1, or Decimal 1	Table 769, p. 1056
PTP Global Configuration2 Register	0x2, or Decimal 2	Table 770, p. 1057
PTP Global Configuration3 Register	0x3, or Decimal 3	Table 771, p. 1057
PTP Global Status Data Registers		
PTP Global Status0 Register	0x8, or Decimal 8	Table 772, p. 1058
PTP Port Configuration Data Registers		
PTP Port Configuration0 Register	0x0, or Decimal 0	Table 773, p. 1059
PTP Port Configuration1 Register	0x1, or Decimal 1	Table 774, p. 1059
PTP Port Configuration2 Register	0x 2, or Decimal 2	Table 775, p. 1060
PTP Port Status Registers		
PTP Port Status0 Register	0x8, or Decimal 8	Table 776, p. 1063
PTP Port Status1 Register	0x9 and 0xA, or Decimal 9 and Decimal 10	Table 777, p. 1064
PTP Port Status2 Register	0xB, or Decimal 11	Table 778, p. 1064
PTP Port Status3 Register	0xC, or Decimal 12	Table 779, p. 1064
PTP Port Status4 Register	0x D and 0xE, or Decimal 13 and 14	Table 780, p. 1066
PTP Port Status5 Register	0xF, or Decimal 15	Table 781, p. 1066
PTP Port Status6 Register	0x10, or Decimal 16	Table 782, p. 1066
PTP Port Status7 Register	0x11 and 0x12, or Decimal 17 and 18	Table 783, p. 1068
PTP Port Status8 Register	0x13, or Decimal 19	Table 784, p. 1068

Copyright © 2014 Marvell

Table 763: Summary Map Table for the PTP Registers (Continued)

Register Name	Offset	Table and Page
PTP Port Status9 Register	0x15, or Decimal 21	Table 785, p. 1068
TAI Global Configuration Data Registers		
TAI Global Configuration, PTP Port = 0xE	0x0, or Decimal 0	Table 786, p. 1070
TAI Global Configuration Register, PTP Port = 0xE and 0xF	0x1, or Decimal 1	Table 787, p. 1072
TAI Global Configuration0 Register	0x2 and 0x3, or Decimal 2 and 3	Table 788, p. 1073
TAI Global Configuration1 Register	0x4, or Decimal 4	Table 789, p. 1073
TAI Global Configuration2 Register	0x5, or Decimal 5	Table 790, p. 1073
TAI Global Configuration3 Register	0x6, or Decimal 6	Table 791, p. 1074
TAI Global Configuration4 Register	0x7, or Decimal 7	Table 792, p. 1074
TAI Global Status Registers		
TAI Global Status1 Register	0x8, or Decimal 8	Table 793, p. 1075
TAI Global Status1 Register	0x9, or Decimal 9	Table 794, p. 1076
TAI Global Status2 Register	0xA and 0xB, or Decimal 10 and 11	Table 795, p. 1076
PTP Global Status1 Register	0xE and 0xF, or Decimal 14 and 15	Table 796, p. 1077

A.8.1 PTP Core Access Registers

Table 764: PTP Command Register

Offset: Port 0 and 1: 0x7C000, Port 2 and 3: 0x3C000

Bits	Field	Type	Description
15	PTPBusy	SC 0x0	PTP Unit Busy. This bit must be set to 0x1 to start an PTP operation (see PTPOp below). Only one PTP operation can be executing at a time, so this bit must be 0x0 before setting it to a 0x1. When the requested PTP operation completes, this bit will automatically be cleared to a 0x0.
14:12	PTPOp	RWR 0x0	PTP Unit Operation Code. The devices support the following PTP operations (all of these operations can be executed while frames are transiting through the device): 000 = No Operation 001 = Reserved 010 = Reserved 011 = Write to the register pointed by PTPAddr below. PTPData registers content gets written into the selected register. 100 = Read from the register pointed to by PTPAddr below. The data read from the selected register is transferred to PTPData register. 101 = Reserved 110 = Read with post increment of register address defined in bits PTPAddr bits below. For PTP data structures, this command instructs the hardware to take a snap-shot of four consecutive data registers, starting with the PTPAddr location. This is used for capturing time counter values that are more than 16-bits wide, along with the sequence identifier information. 111 = Reserved

Table 764: PTP Command Register (Continued)

Offset: Port 0 and 1: 0x7C000, Port 2 and 3: 0x3C000

Bits	Field	Type	Description
11:8	PTPPort	RWR 0x0	This indicates the physical port of this device. These bits indicate the PTP port that is being accessed in the PTP Command register. For example, if this field is programmed to a 0x1, it indicates that PTP registers belonging to port number 1 are being accessed. To access PTP global registers, set the PTPBlock to 0x0 and set PTPPort to 0xF. To access Time Application Interface (TAI) Global registers, set the PTPBlock to 0x0 and set PTPPort to 0xE. To access PTP Policy global registers, set the PTPBlock to 0x1 and set PTPPort to 0xF. To access QAV global registers, set the PTPBlock to 0x2 and set PTPPort to 0xF.
7:5	PTPBlock	RWR 0x0	This field indicates the block of addresses within the device. For example within the PTP register space, this field selects the block like PTP, SRP, and QAV. The PTPAddr field below selects the specific address within a selected block. 0x0 = To select PTP register space 0x1 = To select PTP Policy register space 0x2 = To select QAV register space 0x3–0x7 = Reserved for future use. **NOTE:** Accessing registers in the reserved range of the PTPBlock will return all zeroes back for the PTP Command register.
4:0	PTPAddr	RWR 0x0	These bits indicate the address bits for the register operation being specified in the PTPOp bits specified above.

Table 765: PTP Data Register

Offset: Port 0 and 1: 0x7C008, Port 2 and 3: 0x3C008

Bits	Field	Type	Description
15:0	PTPData	RWR 0x0	PTP Data bits These data bits indicate either the read data or the write data bits, depending on the PTP Command register. For a read operation, the hardware logic fetches the data bits from the specified address in PTP Command register and stores them into these bits. For a write operation, the hardware logic utilizes the data bits in this field to write to the specified address location in PTP Command register.

A.8.2 PTP Configuration Registers

Table 766: PTP Reset Register

Offset: Port 0 and 1: 0x7C010

Bits	Field	Type	Description
31:1	Reserved	RSVD 0x0	Reserved
0	PTPReset	RW 0x0	PTP Reset 0 = PTP is reset. 1 = PTP is enabled.

Table 767: PTP Clock Select Register

Offset: Port 0 and 1: 0x7C018

Bits	Field	Type	Description
31:1	Reserved	RSVD 0x0	Reserved
0	PTPClkSelect	RW 0x0	PTP Clock Select 0 = An internal 125 MHz clock is used as the PTP module clock. 1 = An external PTP clock is used as the PTP module clock.

A.8.3 PTP Global Configuration Registers

Figure 145:PTP Configuration Data Structure Registers

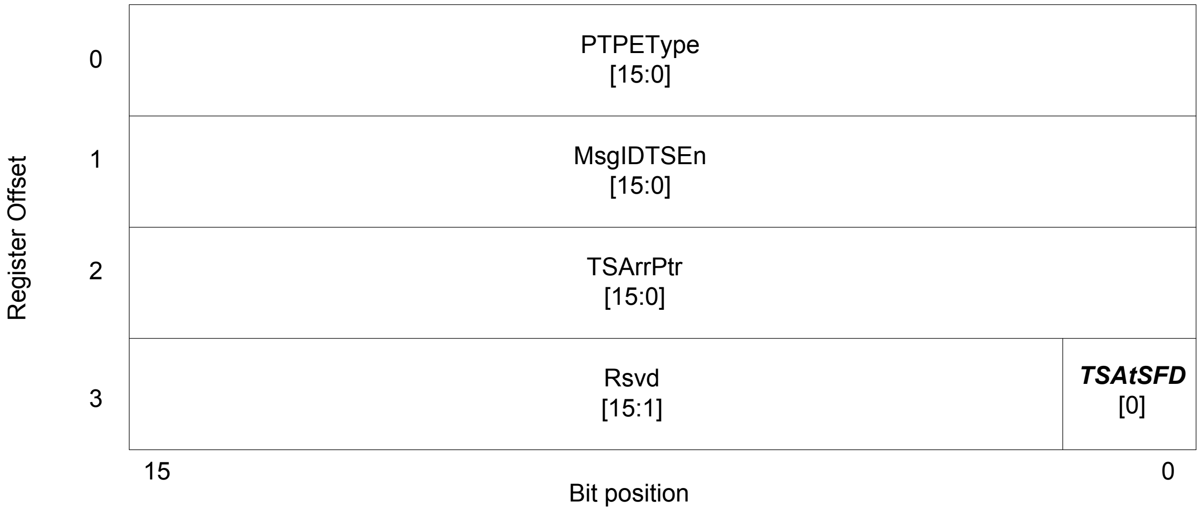


Table 768: PTP Global Configuration0 Register

Offset: 0x0, or Decimal 0

Bits	Field	Type	Description
15:0	PTPEType	RWR 0x0	Precise Time Protocol EtherType. All PTP frames are recognized using a combination of a specific EtherType and MessageID values (part of the PTP Common Header). The actual numeric value is not yet defined in the IEEE802.1AS standard. It is possible that all IEEE1588 time sync frames and IEEE802.11 wireless LAN location estimation time sync messages follow the same EtherType but varying Ether subtypes (aka messageID). The MsgIDTSEn (specified below) qualifies the types of frames that the hardware needs to time stamp.

Table 769: PTP Global Configuration1 Register

Offset: 0x1, or Decimal 1

Bits	Field	Type	Description
15:0	MsgIDTSEn	RWR 0x0	Message Identifier Time Stamp Enable. MessageID is part of the PTP common header for time sync frames. There are PTP frames which need to be time stamped by hardware and some that do not need to be. This field identifies the PTP frame types that need to be time stamped by the hardware. The MsgIDTSEn refers to the bit mask enables, where each bit indicates whether the vectorized MessageID value needs to be time stamped or not. 0x0 = Indicates to hardware to NOT time stamp both incoming and/or outgoing PTP frames which match the MessageID. 0x1= Indicates to hardware to time stamp both incoming and/or outgoing PTP frames which match the MessageID. For example, if MessageID field (in the PTP common header) with a value of 0x4 ought to be time stamped in hardware, then MsgIDTSEn[4] should be configured to a 0x1. Then for the incoming PTP frame with the MessageID field of 0x4, one of the two arrival counters get updated (PTPArr0Time or PTPArr1Time). The exact time counter is identified by the TSArrPtr field below). For an outgoing PTP frame with the MessageID field of 0x4, the TSDepTime counter gets updated for an incoming frame with the MessageId field from the PTPCommon header matching 0x4.

Table 770: PTP Global Configuration2 Register
Offset: 0x2, or Decimal 2

Bits	Field	Type	Description
15:0	TSArrPtr	RWR 0x0	Time Stamp Arrival Time Counter Pointer. If the incoming PTP frame needs to be time stamped (based on MsgIDTSEn), this field determines whether the hardware logic should use PTPArr0Time or PTPArr1Time for storing the arriving frame’s time stamp information. Each bit in this field corresponds to the sixteen combinations of the vectorized MessageID field. For example, if TSArrPtr[2] is set to a 0x1, it indicates to the hardware that if MsgIDTSEn[2] is set. Then use the PTPArr1Time counter for storing the incoming PTP frame’s time stamp. However, if TSArrPtr[2] is set to a 0x0 that indicates to the hardware that if MsgIDTSEn[2] is set. Then use the PTPArr0Time counter for storing the incoming PTP frame’s time stamp. Vectorized: Vectorized term here refers to converting the hexadecimal MessageID field into a sixteen bit binary number.

Table 771: PTP Global Configuration3 Register
Offset: 0x3, or Decimal 3

Bits	Field	Type	Description
15:0	Reserved	RSVD	Reserved for future use.

A.8.4 PTP Global Status Data Structure Registers

Figure 146:PTP Global Status Data Structure Registers

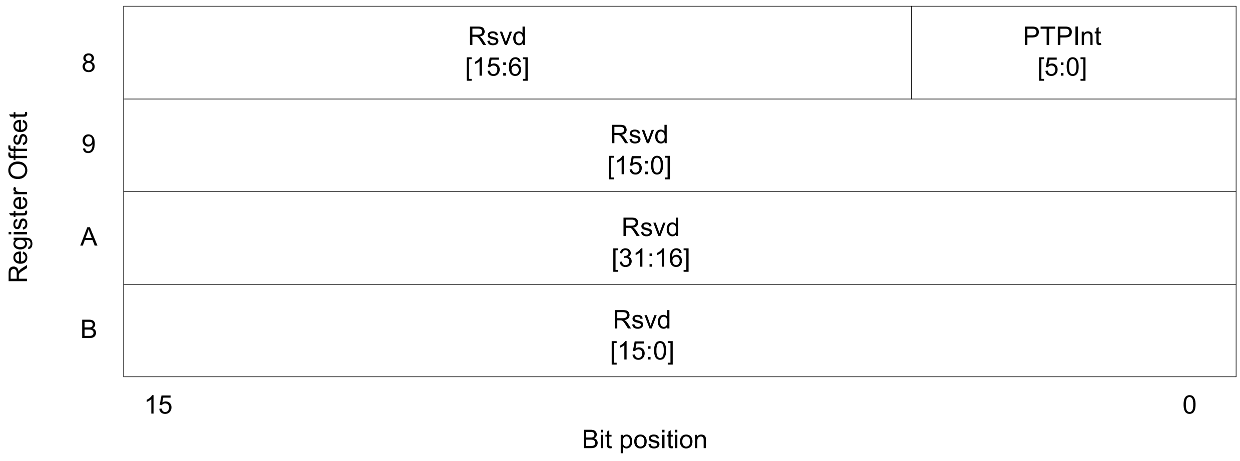


Table 772: PTP Global Status0 Register
Offset: 0x8, or Decimal 8

Bits	Field	Type	Description
15:6	Reserved	RSVD	Reserved for future use.
5:0	PTPInt	RWR 0x0	Precise Time Protocol Interrupt The PTP Interrupt bit gets set for a given port when an incoming PTP frame is time stamped and PTPArrIntEn for that port is set to 0x1. Similarly PTP Interrupt bit gets set for a given port when an outgoing PTP frame is time stamped and PTPDepIntEn for that port is set to 0x1. The hardware logic sets this per port bit, based on the above criteria, and it is cleared upon software reading and clearing the corresponding time counter valid bits that are valid for that port.

A.8.5 PTP Port Configuration Data Structure Registers

Figure 147:PTP Port Configuration Data Structure Registers

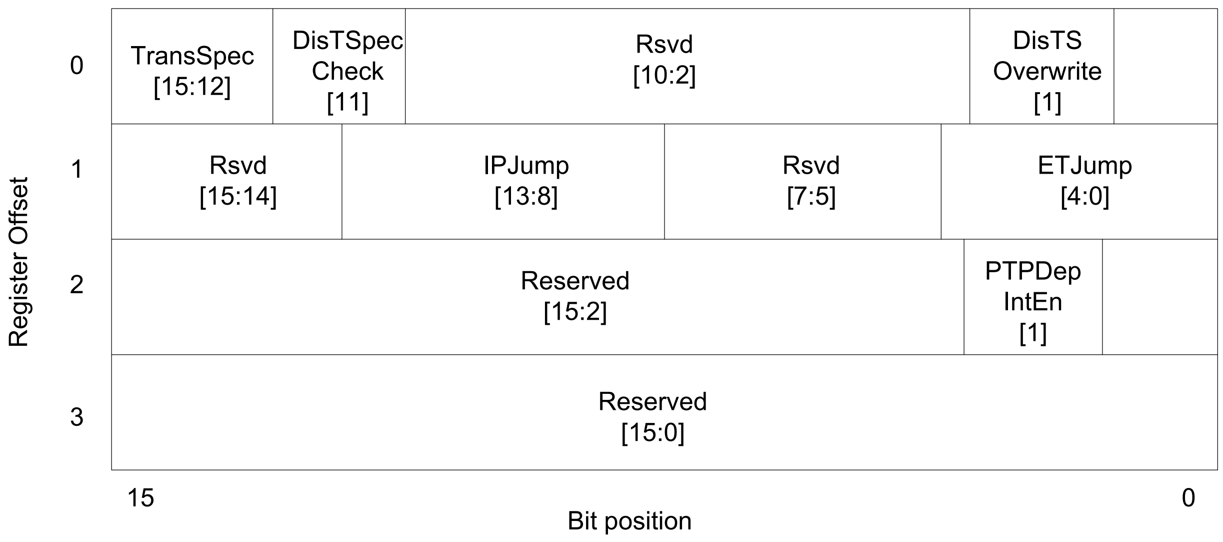


Table 773: PTP Port Configuration0 Register

Offset: 0x0, or Decimal 0

Bits	Field	Type	Description
15:12	TransSpec	RWS 0x1	PTP Transport Specific value. The Transport Specific bits present in PTP Common header are used to differentiate between IEEE1588, IEEE802.1AS frames. This is to differentiate between various timing protocols running on either Layer2 or higher protocol layers. In hardware, in addition to comparing the EtherType to determine that the incoming frame is a PTP frame, it compares the TransSpec bits to the incoming PTP common header's Transport Specific bits. If there is a match, then hardware logic time stamps the frames indicated by MsgIDTSEn and optionally interrupts the CPU. If there is no match, then the hardware does not perform any operations in the PTP core. For IEEE 1588 networks, this bit is expected to be configured to a 0x0. For IEEE 802.1AS networks, this is expected to be configured to 0x1.
11	DisTSpecCheck	RWR 0x0	Disable Transport Specific Check. 0x1= Disables checking for the Transport Specific segment of the PTP Common header, between the incoming packet data and the configured TransSpec (PTP). 0x0 = Enables checking for the Transport Specific segment of the PTP Common header between the incoming packet data and the configured TransSpec (PTP Port Configuration, Offset 0x0).
10:2	Reserved	RSVD	Reserved for future use.
1	DisTSOverwrite	RWR 0x0	Disable Time Stamp Counter Overwriting. When set to 0x1, PTPArr0Time, PTPArr1Time, and PTPDepTime values do not get overwritten until their corresponding valid bits (defined in PTP Port Status Data Structure below) are not cleared. This situation only arises when a port based time stamp counter is written by hardware logic, but the software layer has not read the data. When set to 0x0, PTPArr0Time, PTPArr1Time, and PTPDepTime values do get overwritten, even though their corresponding valid bits (defined in PTP Port Status Data Structure below), are not cleared.
0	DisPTP	RWR 0x0	Disable Precise Time Stamp logic (per-port bit). 0x0 = PTP logic within the chip is enabled. 0x1 = PTP logic is disabled, i.e., hardware logic does not recognize or timestamp PTP frames. Even interrupt generation logic is disabled.

Table 774: PTP Port Configuration1 Register

Offset: 0x1, or Decimal 1

Bits	Field	Type	Description
15:14	Reserved	RSVD	Reserved for future use.

Table 774: PTP Port Configuration1 Register (Continued)

Offset: 0x1, or Decimal 1

Bits	Field	Type	Description
13:8	IPJump	RWS 0x2	Internet Protocol Jump. This field specifies to the PTP hardware logic how many bytes it should skip, starting at the value specified by ETJump (PTP Port Configuration register, Offset 0x1). **NOTE:** This specifies the jump to the beginning of the IPv4 or IPv6 headers for the hardware parser. This allows flexibility in the hardware to skip past the protocol chains that are specific to customer networks including MPLS. For example, if ETJump is programmed to 0xC and IPJump is programmed to 0x16, this indicates to hardware to skip 0x22 bytes to get to the IP header. It can be either an IPv4 or IPv6 header.
7:5	Reserved	RSVD	Reserved for future use.
4:0	ETJump	RWS 0xC	EtherType Jump. This field specifies to the PTP hardware logic how many bytes should it skip starting from MAC-DA of the frame to get to the EtherType of the frame. The hardware would skip that many bytes, and then compare the next 2 bytes to the configured PTPEType (PTP Global Configuration Register, Offset 0x0). This allows flexibility in the hardware to skip past the protocol chains that are specific to customer networks including IEEE802.1q tag, Provider tag.

Table 775: PTP Port Configuration2 Register

Offset: 0x 2, or Decimal 2

Bits	Field	Type	Description
15:2	Reserved	RSVD	Reserved for future use.
1	PTPDepIntEn	RWR 0x0	Precise Time Protocol Port Departure Interrupt enable. This field indicates the per-port interrupt enable for outgoing PTP frame from a given port. When a bit is enabled in this field, it indicates that whenever hardware logic time stamps a PTP frame to this port, it needs to send an interrupt to the CPU. 0x0 = Disable PTP departure counter, based interrupt generation. Even if the PTPDepTimeValid is set to 0x1, PTPInt does not get generated by hardware logic for outgoing PTP frames. 0x1 = Enable PTP departure counter, based interrupt generation. If the PTPDepTimeValid is set to 0x1, PTPInt does get generated by hardware logic for outgoing PTP frames. **NOTE:** Hardware logic only time stamps the PTP frames when configured to do so by MsgIDTSEn field (as specified above).

Table 775: PTP Port Configuration2 Register (Continued)

Offset: 0x 2, or Decimal 2

Bits	Field	Type	Description
0	PTPArrIntEn	RWR 0x0	Precise Time Protocol Port Arrival Interrupt enable. This field indicates the per-port interrupt enable for incoming PTP frames from a given port. When a bit is enabled in this field it indicates that whenever hardware logic time stamps a PTP frame from this port, it needs to send an interrupt to the CPU. 0x0 = Disable PTP arrival counter, based interrupt generation. Even if the PTPArr0TimeValid or PTPArr1TimeValid is set to 0x1 for that port, PTPInt does not get generated by hardware logic for incoming PTP frames from this port. 0x1 = Enable PTP arrival counter, based interrupt generation. If the PTPArr0TimeValid or PTPArr1TimeValid is set to 0x1, PTPInt does get generated by hardware logic for incoming PTP frames. **NOTE:** Hardware logic only time stamps the PTP frames when configured to do so by MsgIDTSEn field (specified above).

A.8.6 PTP Port Status Registers

Figure 148:PTP Port Status Data Structure Registers

Register Offset				
8	Rsvd [15:3]	PTPArr0Int Status [2:1]	PTPArr0 TimeValid [0]	
9	PTPArr0Time [15:0]			
A	PTPArr0Time Contd. [15:0]			
B	PTPArr0SeqId [15:0]			
C	Rsvd [15:3]	PTPArr1Int Status [2:1]	PTPArr1 TimeValid [0]	
D	PTPArr1Time [15:0]			
E	PTPArr1Time Contd. [15:0]			
F	PTPArr1SeqId [15:0]			
10	Rsvd [15:3]	PTPDepInt Status [2:1]	PTPDep TimeValid [0]	
11	PTPDepTime [15:0]			
12	PTPDepTime Contd. [15:0]			
13	PTPDepSeqId [15:0]			
14	Rsvd [15:0]			
15	PTPTS DepDisCtr [3:0]	PTPNonTS DepDisCtr [3:0]	PTPTS ArrDisCtr [3:0]	PTPNonTS ArrDisCtr [3:0]

15 0
Bit position

Copyright © 2014 Marvell

Table 776: PTP Port Status0 Register

Offset: 0x8, or Decimal 8

Bits	Field	Type	Description
15:3	Reserved	RSVD	Reserved for future use.
2:1	PTPArr0IntStatus	RWR 0x0	Precise Time Protocol Arrival Time 0 Interrupt Status The PTP Arrival time 0 Interrupt bit gets set for a given port, when an incoming PTP frame is time stamped in PTPArr0Time counter. 0x0 = Normal, i.e., none of the error conditions stated below are valid for this packet. 0x1 = If the PTPArr0Time counter with its associated valid and SequenceID was overwritten since more than one PTP frame that needed to use the arrival0 counters, arrived at the device through this port before CPU cleared the corresponding valid and counter bits for the previous PTP frame(s). 0x2 = If the incoming frame could not be time stamped in hardware because the DisTSOverwrite was set to a 0x1 and PTPArr0TimeValid was 0x1, when this PTPFrame was processed in hardware logic. This can occur, when more than one PTP frame that needs time stamping into the arrival0 counters, arrives at the device before CPU clears the valid bits for the previous frame. 0x3 = Reserved **NOTE:** If the PTP frame gets discarded inside the device, for policy, CRC, queue congestion, or any other reasons then one of the PTP arrival discard counters is updated (PTPNonTSArrDisCtr or PTPTSArrDisCtr). See the discard counter description for further details.
0	PTPArr0TimeValid	RWR 0x0	Precise Time Protocol Arrival 0 Time Valid When the PTPArr0Time value is updated by hardware, this bit is set to a 0x1 validating the time counter. 0x0 = PTPArr0Time is not valid. 0x1 = PTPArr0Time is valid and PTPArr0IntStatus represents the status information for the PTPArr0Time counter. This is set by hardware for the frames that are assured of reaching the CPU. For frames with a CRC error, this bit is not set, but either PTPNonTSArrCtr or PTPTSArrCtr is updated. **NOTE:** This valid bit needs to be cleared by software, after reading the value, and hardware does not provide any auto-clearing mechanisms. This is because hardware has no way to figure out if software is finished reading all the relevant registers for a particular Time counter before clearing the valid bit.

Table 777: PTP Port Status1 Register

Offset: 0x9 and 0xA, or Decimal 9 and Decimal 10

Bits	Field	Type	Description
31:0	PTPArr0Time	RWR 0x0	Precise Time Protocol Arrival 0 Time counter. This indicates the PTP Arrival 0 time stamp value that is captured by the PTP logic for a PTP frame that needs to be time stamped. The captured time stamp value is from a Global Timer counter running off of a device internal clock. The value in this counter is validated by the PTPArr0TimeValid bit, and PTPArr0IntStatus indicates the status of the PTP frame through the device, as described above. **NOTE:** The maximum Jitter associated with time stamping within the hardware is one TSClkPer value.

Table 778: PTP Port Status2 Register

Offset: 0xB, or Decimal 11

Bits	Field	Type	Description
15:0	PTPArr0SeqId	RWR 0x0	Precise Time Protocol Arrival 0 Sequence Identifier. This indicates the sequence identifier (extracted in hardware from the incoming frames PTP Common Header) for the frame whose time stamp information has been captured by hardware logic in PTPArr0Time register.

Table 779: PTP Port Status3 Register

Offset: 0xC, or Decimal 12

Bits	Field	Type	Description
15:3	Reserved	RSVD	Reserved for future use.

Copyright © 2014 Marvell

Table 779: PTP Port Status3 Register (Continued)

Offset: 0xC, or Decimal 12

Bits	Field	Type	Description
2:1	PTPArr1IntStatus	RWR 0x0	Precise Time Protocol Arrival Time 1 Interrupt Status The PTP Arrival time 1 Interrupt bit gets set for a given port when an incoming PTP frame is time stamped in PTPArr1Time counter. 0x0 = Normal, i.e., none of the error conditions stated below are valid for this packet. 0x1 = If the PTPArr1Time counter with its associated valid and SequenceID was overwritten since more than one PTP frame that needed to use the arrival1 counters, arrived at the device through this port before CPU cleared the corresponding valid and counter bits for the previous PTP frame(s). 0x2 = If the incoming frame could not be time stamped in hardware because the DisTSOverwrite was set to a 0x1 and PTPArr1TimeValid was 0x1, when this PTPFrame was processed in hardware logic. This can occur, when more than one PTP frame that needs time stamping into the arrival0 counters, arrives at the device before CPU clears the valid bits for the previous frame. 0x3 = Reserved **NOTE:** If the PTP frame gets discarded inside the device for policy, CRC, queue congestion, or any other reasons, then one of the PTP arrival discard counters is updated (PTPNonTSArrDisCtr or PTPTSArrDisCtr). See the discard counter description for further details.
0	PTPArr1TimeValid	RWR 0x0	Precise Time Protocol Arrival 1 Time Valid When the PTPArr1Time value is updated by hardware, this bit is set to a 0x1 validating the time counter. 0x0 = PTPArr1Time is not valid. 0x1 = PTPArr1Time is valid, and PTPArr1IntStatus represents the status information for the PTPArr1Time counter. This is set by hardware for the frames that are assured of reaching the CPU. For frames with a CRC error, this bit is not set, but either PTPNonTSArrCtr or PTPTSArrCtr is updated. **NOTE:** This valid bit needs to be cleared by software, after reading the value, and hardware does not provide any auto-clearing mechanisms. This is because hardware has no way to figure out if software is finished reading all the relevant registers for a particular Time counter before clearing the valid bit.

Table 780: PTP Port Status4 Register

Offset: 0x D and 0xE, or Decimal 13 and 14

Bits	Field	Type	Description
31:0	PTPArr1Time	RWR 0x0	Precise Time Protocol Arrival 1 Time counter. This indicates the PTP Arrival 1 time stamp value that is captured by the PTP logic for a PTP frame that needs to be time stamped. The captured time stamp value is from a Global Timer counter running off of a device internal clock. The value in this counter is validated by the PTPArr1TimeValid bit and the PTPArr1IntStatus bit indicates the status of the PTP frame through the device described above. **NOTE:** The maximum Jitter associated with time stamping within the hardware is one TSClkPer value.

Table 781: PTP Port Status5 Register

Offset: 0xF, or Decimal 15

Bits	Field	Type	Description
15:0	PTPArr1SeqId	RWR 0x0	Precise Time Protocol Arrival 1 Sequence Identifier. This indicates the sequence identifier (extracted in hardware from incoming frames PTPCommonHeader) for the frame whose time stamp information has been captured by hardware logic in PTPArr1Time register.

Table 782: PTP Port Status6 Register

Offset: 0x10, or Decimal 16

Bits	Field	Type	Description
15:3	Reserved	RSVD	Reserved for future use.

Table 782: PTP Port Status6 Register (Continued)
Offset: 0x10, or Decimal 16

Bits	Field	Type	Description
2:1	PTPDepIntStatus	RWR 0x0	Precise Time Protocol Departure Time Interrupt Status The PTP Departure time Interrupt bit gets set for a given port when an incoming PTP frame is time stamped in the PTPDepTime counter. 0x0 = Normal, i.e., none of the error conditions stated below are valid for this packet. 0x1 = The PTPDepTime counter, with its associated valid and SequenceID, was overwritten, since more than one PTP frame that needed to use departure counter exited the device through this port before CPU cleared the corresponding valid and counter bits for the previous PTP frame(s). 0x2 = The outgoing frame could not be time stamped in hardware because the DisTSOverwrite was set to a 0x1 and PTPDepTimeValid was 0x1 when this PTPFrame was processed in hardware logic. This can occur when more than one PTP frame that needs time stamping into the departure counter leaves the device, before CPU clears the valid bits for the previous frame. 0x3 = Reserved **NOTE:** If the PTP frame gets discarded inside the device, for CRC reasons, then the PTP departure discard counter gets updated (PTPNonTSDepDisCtr or PTPTSDepDisCtr). See the discard counter description for further details.
0	PTPDepTimeValid	RWR 0x0	Precise Time Protocol Departure Time Valid When the PTPDepTime value is updated by hardware, this bit is set to a 0x1 validating the time counter. 0x0 = PTPDepTime is not valid. 0x1 = PTPDepTime is valid and PTPDepIntStatus represents the status information for the PTPDepTime counter. This is set by hardware for the frames that are assured to depart the port. For frames with a CRC error, this bit is not set, but either PTPNonTSDepCtr or PTPTSDepCtr is updated. **NOTE:** This valid bit needs to be cleared by software, after reading the value, and hardware does not provide any auto-clearing mechanisms. This is because hardware has no way to figure out if software is finished reading all the relevant registers for a particular Time counter before clearing the valid bit.

Table 783: PTP Port Status7 Register

Offset: 0x11 and 0x12, or Decimal 17 and 18

Bits	Field	Type	Description
31:0	PTPDepTime	RWR 0x0	Precise Time Protocol Departure Time counter. This indicates the PTP Departure time stamp value that is captured by the PTP logic for a PTP frame that needs to be time stamped. The captured time stamp value is from a Global Timer counter running off of a device internal clock. The value in this counter is validated by the PTPDepTimeValid bit and the PTPDepIntStatus indicates the status of the PTP frame through the device described above. **NOTE:** The maximum Jitter associated with time stamping within the hardware is one TSClkPer value.

Table 784: PTP Port Status8 Register

Offset: 0x13, or Decimal 19

Bits	Field	Type	Description
15:0	PTPDepSeqId	RWR 0x0	Precise Time Protocol Departure Sequence Identifier. This indicates the sequence identifier (extracted in hardware from the incoming frames PTP Common Header) for the frame whose time stamp information has been captured by hardware logic in the PTPDepTime register.

Table 785: PTP Port Status9 Register

Offset: 0x15, or Decimal 21

Bits	Field	Type	Description
15:12	PTPTSDepDisCtr	RWR 0x0	Precise Time Protocol Departure frame discard counter for PTP frames that need hardware time stamping. This counter is incremented by the hardware logic whenever it discards a PTP frame that needs hardware time stamping (i.e., PTPEtype is a match, and the MsgIDTSEn bit for the corresponding MsgID field in the outgoing PTP frame is set). The PTP frame could be discarded because of CRC reasons in the egress pipe. This counter wraps around in hardware.

Table 785: PTP Port Status9 Register (Continued)

Offset: 0x15, or Decimal 21

Bits	Field	Type	Description
11:8	PTPNonTSDepDisCtr	RWR 0x0	Precise Time Protocol Departure frame discard counter for PTP frames that do not need hardware time stamping. This counter is incremented by the hardware logic whenever it discards a PTP frame that does not need to be time stamped (i.e., PTPEtype is a match, but the MsgIDTSEn bit for the corresponding MsgID field in the outgoing PTP frame is not set). The PTP frame could be discarded because of CRC reasons in the egress pipe. This counter wraps around in hardware.
7:4	PTPTSArrDisCtr	RWR 0x0	Precise Time Protocol arrival frame discard counter for PTP frames that need hardware time stamping. This counter is incremented by the hardware logic whenever it discards a PTP frame that needs hardware time stamping (i.e., PTPEtype is a match, and the MsgIDTSEn bit for the corresponding MsgID field in the outgoing PTP frame is set). The PTP frame could be discarded because of CRC, Policy, Queue congestion, or any other reason inside the device. This counter wraps around in hardware.
3:0	PTPNonTSArrDisCtr	RWR 0x0	Precise Time Protocol Non time stamp Arrival frame discard counter. This counter is incremented by the hardware logic whenever it discards a PTP frame that does not need hardware time stamping (i.e., PTPEtype is a match, but the MsgIDTSEn bit for the corresponding MsgID field in the outgoing PTP frame is not set). The PTP frame could be discarded because of CRC, Policy, Queue congestion, or any other reason inside the device. This counter wraps around in hardware.

A.8.7 Timing Applications Interface Registers

Figure 149:TAI Global Configuration Data Structure

<table>
<tr><th>Register Offset</th><th colspan="10"></th></tr>
<tr><td>0</td><td>Event CapOv [15]</td><td>EventCtr Start [14]</td><td>Rsvd [13:10]</td><td>TrigGen IntEn [9]</td><td>EventCap IntEn [8]</td><td>Rsvd [7:4]</td><td>TimeInc DecEn [3]</td><td>Rsvd [2]</td><td>Trig Mode [1]</td><td>TrigGen Req [0]</td></tr>
<tr><td>1</td><td colspan="10">TSClkPer [15:0]</td></tr>
<tr><td>2</td><td colspan="10">TrigGenAmt bits 15 to 0 [15:0]</td></tr>
<tr><td>3</td><td colspan="10">TrigGenAmt bits 31 to 16 [15:0]</td></tr>
<tr><td>4</td><td colspan="10">TrigClkComp [15:0]</td></tr>
<tr><td>5</td><td colspan="3">PulseWidth [15:12]</td><td colspan="2">TimeInc DecOp [11]</td><td colspan="5">TimeIncDecAmt [7:0]</td></tr>
<tr><td>6</td><td colspan="5">Rsvd [15:9]</td><td colspan="5">SoCClkPer [8:0]</td></tr>
<tr><td>7</td><td colspan="10">SoCClkComp [15:0]</td></tr>
<tr><td></td><td>15</td><td colspan="8">Bit position</td><td>0</td></tr>
</table>

Table 786: TAI Global Configuration, PTP Port = 0xE
Offset: 0x0, or Decimal 0

Bits	Field	Type	Description
15	EventCapOv	RWR 0x0	Event Capture Overwrite When 0x1, this bit enables overwriting the EventCapRegister (PTP Global Status Register, Offset 0xE). The hardware would only overwrite the EventCapRegister if the previously captured event register has not been read by the software. When 0x0, this bit specifies to hardware logic to capture an event namely, take a snapshot of PTP Global Timer value at the rising edge of PTP_ EVENT_REQ (a input into the core) and wait for software to read the EventCapRegister before capturing another event.

Table 786: TAI Global Configuration, PTP Port = 0xE (Continued)
Offset: 0x0, or Decimal 0

Bits	Field	Type	Description
14	EventCtrStart	RWR 0x0	Event Counter Start When 0x1, this bit enables the hardware logic to start incrementing the EventCapCtr register (PTP Global Status Register, 0xD) whenever it captures a low-to-high transition on the PTP_EVENT_REQ signal. When 0x0, the EventCapCtr is not modified by hardware logic, even when it captures a low-to-high transition on the PTP_EVENT_REQ signal.
13:10	Reserved	RSVD	Reserved for future use.
9	TrigGenIntEn	RWR 0x0	Trigger Generator Interrupt Enable. When 0x1, the TAI block would generate an interrupt whenever a PTP_ TRIG_GEN event has been sent out on the PTP_TRIG_GEN signal output. When 0x0, no interrupts are generated by the PTP_TRIG_GEN logic.
8	EventCapIntEn	RWR 0x0	Event Cap Interrupt Enable When 0x1, the TAI block would generate an interrupt whenever an event has been captured on the PTP_EVENT_REQ signal. When 0x0, no interrupts are generated by the EventCap logic.
7:4	Reserved	RSVD	Reserved for future use.
3	TimeIncDecEn	SC 0x0	Time Increment Decrement Enable This is used to adjust the PTP Global Time counter value, with respect to the phase offset computed by the PTP firmware, using the PTP control messages like Sync, PDelayReq, and PDelayResp. The assumption is that the software maintains the 64-bit seconds field of the PTP time of day and hardware maintains the 32-bit field (in PTP clock increments) in the PTP Global Time counter (TAI Global Status Register, Offset 0xE and 0xF). When 0x1, this bit enables the hardware logic to increment or decrement the PTP Global Time counter by the value specified by TimeIncDecAmt (PTP Global Configuration Register, Offset 0x6). When 0x0, the hardware logic does not modify the PTP Global Time counter value. **NOTE:** This function is executed once by the hardware, and upon execution this bit is cleared to 0x0.
2	Reserved	RSVD	Reserved for future use.

Table 786: TAI Global Configuration, PTP Port = 0xE (Continued)

Offset: 0x0, or Decimal 0

Bits	Field	Type	Description
1	TrigMode	RWR 0x0	Trigger Mode When 0x1, the hardware logic matches the PTP Global Timer (TAI Global Status, Offset 0xE and 0xF) with the TrigGenAmt (TAI Global Configuration, Offset 0x2 and 0x3) and generates a pulse on the PTP_TRIG_GEN output signal. The pulse width for this is specified by PulseWidth (TAI Global Configuration, Offset 0x5). **NOTE:** The minimum pulse width that can be generated is one TSClkPer value, and the maximum pulse width is 15 times the TSClkPer value. When 0x0, the hardware logic uses the value specified in the TrigGenAmt as the period for generating periodic pulses on the PTP_TRIG_GEN signal with a 50% duty cycle clock. **NOTE:** The minimum clock period that can be generated on the PTP_TRIG_GEN output signal is 2 times the TSClkPer value. For example, if a 1 pps signal needs to be generated, the TrigMode is set to 0x0, if the TSClkPer is set to 8 ns and the TrigGenAmt is set to 125 x 10^6 cycles.
0	TrigGenReq	RWR 0x0	Trigger Generation Request When 0x1, it validates the TrigGenAmt, TrigMode, and TrigClkComp fields. This enables the hardware logic to generate either a trigger, based on the TrigGenAmt, or a clock, based on the period specified in TrigGenAmt.

Table 787: TAI Global Configuration Register, PTP Port = 0xE and 0xF

Offset: 0x1, or Decimal 1

Bits	Field	Type	Description
15:0	TSClkPer	RWS 0x1F40	Time Stamping Clock Period in pico seconds. This field specifies the clock period for the time stamping clock supplied to the PTP hardware logic. This is the clock that is used by the hardware logic to update the PTP Global Time counter (PTP Global register, Offset 0x9 and 0xA). The default is calculated based on 125 MHz period clock.

Table 788: TAI Global Configuration0 Register

Offset: 0x2 and 0x3, or Decimal 2 and 3

Bits	Field	Type	Description
15:0	TrigGenAmt	RWR 0x0	Trigger Generation Amount This field specifies the PTP Time Application Interface trigger generation time value. When TrigMode is 0x1, the value specified in this field is compared with the PTP Global Timer (PTP Global Status register, Offset 0x9 and 0xA) and, whenever it matches, a pulse is generated whose width is configured using the PulseWidth field (TAI Global Configuration register, Offset 0x5). When TrigMode is 0x0, the value is used as a clock period in TSClkPer increments to generate an output clock on the PTP_TRIG_GEN signal. When TrigMode is 0x0, the TrigClkComp value constantly gets accumulated internally, and when this accumulated value exceeds the value specified in TSClkPer, a TSClkPer value gets added to the clock output momentarily.

Table 789: TAI Global Configuration1 Register

Offset: 0x4, or Decimal 4

Bits	Field	Type	Description
15:0	TrigClkComp	RWR 0x0	Trigger mode Clock Compensation Amount in pico seconds This field is valid only when TrigGenReq is 0x1 and TrigMode is set to 0x0. The field specifies the remainder amount for the clock that is being generated with a period specified by the TrigGenAmt. When TrigMode is 0x0, the TrigClkComp value constantly gets accumulated internally, and when this accumulated value exceeds, the value specified in TSClkPer, a TSClkPer value gets added to the clock output momentarily.

Table 790: TAI Global Configuration2 Register

Offset: 0x5, or Decimal 5

Bits	Field	Type	Description
15:12	PulseWidth	RWS 0xF	Clock high pulse width in units of TSClkPer. This specifies the pulse width of the clock that gets generated on the TrigModeResp, when TrigMode is 0x1 in units defined by the TSClkPer. **NOTE:** When configured to a 0x0, the results are un-deterministic. Therefore, do not program this field to 0x0.

Table 790: TAI Global Configuration2 Register (Continued)

Offset: 0x5, or Decimal 5

Bits	Field	Type	Description
11	TimeIncDecOp	RWR 0x0	Time increment decrement operation. When 0x0, TimeIncDecAmt is considered as an increment value that needs to be added to the PTP Global Time Counter when TimeIncDecEn is 0x1. When 0x1, TimeIncDecAmt is considered as a decrement value that needs to be subtracted from the PTP Global Time Counter when TimeIncDecEn is 0x1. All updates are completed within the same cycle of TimeIncDecEn changing state from 0x0 to 0x1.
10:0	TimeIncDecAmt	RWR 0x0	Time Increment Decrement amount. This field is valid only when TimeIncDecEn is 0x1. This field specifies the number of units of PTP Global Time that need to be incremented or decremented based on the TimeIncDecOp field. This is used for adjusting the PTP Global Time counter value.

Table 791: TAI Global Configuration3 Register

Offset: 0x6, or Decimal 6

Bits	Field	Type	Description
15:9	Reserved	RSVD	Reserved for future use.
8:0	SoCClkPer	RWS 0x186	System on a Chip Clock Period. This specifies the clock period for the PTP Reference Clock. The period is specified in TSClkPer increments. For example, if the TSClkPer is 8 ns, and the device clock needs to be toggling every 3.125 us periods, then this field needs to be programmed to 0x186 and the SoCClkComp field needs to be programmed to 0x1388. 3125 ns/8 ns = 390.625 TSClkPer cycles. Decimal 390 in hexadecimal is 0x186 and 0.625 x 8000 ps = 0x1388 ps.

Table 792: TAI Global Configuration4 Register

Offset: 0x7, or Decimal 7

Bits	Field	Type	Description
15:0	SoCClkComp	RWS 0x1388	System on a Chip Clock Compensation Amount in pico seconds The field specifies the remainder amount, when the clock is generated with a period specified by the SoCClkPer field. The hardware logic keeps track of the remainder for every clock tick generation and compensates for it.

A.8.8 PTP Time Application Interface Registers

Figure 150:PTP Time Application Interface Global Status Data Structure

Register Offset					
8	TrigGenInt [15]	Rsvd [14:0]			
9	EventInt [15]	Rsvd [14:10]	EventCapErr [9]	EventCapValid [8]	EventCapCtr [7:0]
A	EventCapRegister bits 15 to 0 [15:0]				
B	EventCapRegister bits 31 to 16 [15:0]				
C	Rsvd [15:0]				
D	Rsvd [15:0]				
E	PTP Global Time bits 15 to 0 [15:0]				
F	PTP Global Time bits 31 to 16 [15:0]				
	15	Bit position			0

Table 793: TAI Global Status0 Register

Offset: 0x8, or Decimal 8

Bits	Field	Type	Description
15	TrigGenInt	RWR 0x0	Trigger generate mode Interrupt. The TrigGenInt bit is set by the TAI block, when the TrigGenIntEn is 0x1 and when the hardware logic captures a trigger on the PTP_TRIG_GEN signal. This interrupt gets tied to the device interrupt pin.
14:0	Reserved	RSVD	Reserved for future use.

Copyright © 2014 Marvell

Table 794: TAI Global Status1 Register

Offset: 0x9, or Decimal 9

Bits	Field	Type	Description
15	EventInt	RWR 0x0	Event Capture Interrupt. This bit is set by the TAI block when the EventIntEn field is 0x1 and when the hardware logic captures an event in the EventCapRegister. This interrupt gets tied to the device interrupt pin.
14:10	Reserved	RSVD	Reserved for future use.
9	EventCapErr	RWR 0x0	Event Capture Error. This bit is set by the hardware logic when an event has been observed on the PTP_EVENT_REQ signal, but the EventCapValid field is already set to 0x1. This condition can occur when events are observed on the PTP_EVENT_REQ signal faster that the local CPU can read the captured event related counter values.
8	EventCapValid	RWR 0x0	Event Capture Valid. When 0x1, this bit validates the EventCapRegister.
7:0	EventCapCtr	RSVD	Event Capture Counter. This field is incremented by TAI block when the EventCtrStart field is set to 0x1. This field is incremented, whenever an event (a low-to-high transition) has been registered on the PTP_EVENT_REQ signal. **NOTE:** There is no special logic provided to detect this counter's wrap arounds.

Table 795: TAI Global Status2 Register

Offset: 0xA and 0xB, or Decimal 10 and 11

Bits	Field	Type	Description
31:0	EventCap Register	RWR 0x0	Event Capture Register This register captures the value of the PTP Global Timer when an event (a low-to-high transition) has been registered by the TAI block on the PTP_EVENT_REQ signal. If the EventCapOv field is 0x1, then this register indicates the time captured, for the last event in the hardware. When the EventCapErr field is 0x1, the contents in this register are invalid. **NOTE:** The maximum jitter for the EventCapRegister time value, with respect to the rising edge of the PTP_EVENT_REQ input signal, is one TSClkPer value. The minimum PTP_EVENT_REQ signal high or low width has to be equal to or greater than 1.5 times the TSClkPer value. For hardware to capture the Event request on the PTP_EVENT_REQ signal, the minimum gap between two consecutive events must be 150 ns ***(7 times clk_sysp)*** plus 5 times the TSClkPer value.

Table 796: PTP Global Status1 Register
Offset: 0xE and 0xF, or Decimal 14 and 15

Bits	Field	Type	Description
31:0	PTPGlobalTime	RWR 0x0	Precise Time Protocol Global Timer. This indicates the global timer value that is running off of the free running device clock. Based on PTP protocol time of day computations, this field can be either incremented or decremented, using the TimeIncDecAmt field (TAI Global Configuration, Offset 0x5) and by turning on the TimeIncDecEn field (TAI Global Configuration, Offset 0x0) to 0x1 and by selecting an increment or a decrement operation using the TimeIncDecOp field (TAI Global Configuration, Offset 0x5). **NOTE:** This register is updated in the same cycle when the TimeIncDecEn field goes high. This counter wraps around in hardware.

A.9 PCI Express 2.0 Registers

The following table provides a summarized list of all registers that belong to the PCI Express 2.0, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 797: Summary Map Table for the PCI Express 2.0 Registers

Register Name	Offset	Table and Page
PCI Express Configuration Cycles Generation		
PCI Express Configuration Address Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438F8, x4_port3x1_port0: 0x000838F8, x4_port0x1_port0: 0x000418F8, x4_port0x1_port1: 0x000458F8, x4_port0x1_port2: 0x000498F8, x4_port0x1_port3: 0x0004D8F8, x4_port1x1_port0: 0x000818F8, x4_port1x1_port1: 0x000858F8, x4_port1x1_port2: 0x000898F8, x4_port1x1_port3: 0x0008D8F8	Table 798, p. 1097
PCI Express Configuration Data Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438FC, x4_port3x1_port0: 0x000838FC, x4_port0x1_port0: 0x000418FC, x4_port0x1_port1: 0x000458FC, x4_port0x1_port2: 0x000498FC, x4_port0x1_port3: 0x0004D8FC, x4_port1x1_port0: 0x000818FC, x4_port1x1_port1: 0x000858FC, x4_port1x1_port2: 0x000898FC, x4_port1x1_port3: 0x0008D8FC	Table 799, p. 1098
PCI Express Configuration Header		
PCI Express Device and Vendor ID Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042000, x4_port3x1_port0: 0x00082000, x4_port0x1_port0: 0x00040000, x4_port0x1_port1: 0x00044000, x4_port0x1_port2: 0x00048000, x4_port0x1_port3: 0x0004C000, x4_port1x1_port0: 0x00080000, x4_port1x1_port1: 0x00084000, x4_port1x1_port2: 0x00088000, x4_port1x1_port3: 0x0008C000	Table 800, p. 1099
PCI Express Command and Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042004, x4_port3x1_port0: 0x00082004, x4_port0x1_port0: 0x00040004, x4_port0x1_port1: 0x00044004, x4_port0x1_port2: 0x00048004, x4_port0x1_port3: 0x0004C004, x4_port1x1_port0: 0x00080004, x4_port1x1_port1: 0x00084004, x4_port1x1_port2: 0x00088004, x4_port1x1_port3: 0x0008C004	Table 801, p. 1099

Copyright © 2014 Marvell

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Class Code and Revision ID Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042008, x4_port3x1_port0: 0x00082008, x4_port0x1_port0: 0x00040008, x4_port0x1_port1: 0x00044008, x4_port0x1_port2: 0x00048008, x4_port0x1_port3: 0x0004C008, x4_port1x1_port0: 0x00080008, x4_port1x1_port1: 0x00084008, x4_port1x1_port2: 0x00088008, x4_port1x1_port3: 0x0008C008	Table 802, p. 1102
PCI Express BIST Header Type and Cache Line Size Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004200C, x4_port3x1_port0: 0x0008200C, x4_port0x1_port0: 0x0004000C, x4_port0x1_port1: 0x0004400C, x4_port0x1_port2: 0x0004800C, x4_port0x1_port3: 0x0004C00C, x4_port1x1_port0: 0x0008000C, x4_port1x1_port1: 0x0008400C, x4_port1x1_port2: 0x0008800C, x4_port1x1_port3: 0x0008C00C	Table 803, p. 1103
PCI Express BAR0 Internal Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042010, x4_port3x1_port0: 0x00082010, x4_port0x1_port0: 0x00040010, x4_port0x1_port1: 0x00044010, x4_port0x1_port2: 0x00048010, x4_port0x1_port3: 0x0004C010, x4_port1x1_port0: 0x00080010, x4_port1x1_port1: 0x00084010, x4_port1x1_port2: 0x00088010, x4_port1x1_port3: 0x0008C010	Table 804, p. 1104
PCI Express BAR0 Internal (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042014, x4_port3x1_port0: 0x00082014, x4_port0x1_port0: 0x00040014, x4_port0x1_port1: 0x00044014, x4_port0x1_port2: 0x00048014, x4_port0x1_port3: 0x0004C014, x4_port1x1_port0: 0x00080014, x4_port1x1_port1: 0x00084014, x4_port1x1_port2: 0x00088014, x4_port1x1_port3: 0x0008C014	Table 805, p. 1105
PCI Express BAR1 Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042018, x4_port3x1_port0: 0x00082018, x4_port0x1_port0: 0x00040018, x4_port0x1_port1: 0x00044018, x4_port0x1_port2: 0x00048018, x4_port0x1_port3: 0x0004C018, x4_port1x1_port0: 0x00080018, x4_port1x1_port1: 0x00084018, x4_port1x1_port2: 0x00088018, x4_port1x1_port3: 0x0008C018	Table 806, p. 1105

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express BAR1 (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004201C, x4_port3x1_port0: 0x0008201C, x4_port0x1_port0: 0x0004001C, x4_port0x1_port1: 0x0004401C, x4_port0x1_port2: 0x0004801C, x4_port0x1_port3: 0x0004C01C, x4_port1x1_port0: 0x0008001C, x4_port1x1_port1: 0x0008401C, x4_port1x1_port2: 0x0008801C, x4_port1x1_port3: 0x0008C01C	Table 807, p. 1106
PCI Express BAR2 Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042020, x4_port3x1_port0: 0x00082020, x4_port0x1_port0: 0x00040020, x4_port0x1_port1: 0x00044020, x4_port0x1_port2: 0x00048020, x4_port0x1_port3: 0x0004C020, x4_port1x1_port0: 0x00080020, x4_port1x1_port1: 0x00084020, x4_port1x1_port2: 0x00088020, x4_port1x1_port3: 0x0008C020	Table 808, p. 1106
PCI Express BAR2 (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042024, x4_port3x1_port0: 0x00082024, x4_port0x1_port0: 0x00040024, x4_port0x1_port1: 0x00044024, x4_port0x1_port2: 0x00048024, x4_port0x1_port3: 0x0004C024, x4_port1x1_port0: 0x00080024, x4_port1x1_port1: 0x00084024, x4_port1x1_port2: 0x00088024, x4_port1x1_port3: 0x0008C024	Table 809, p. 1107
PCI Express Subsystem Device and Vendor ID Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004202C, x4_port3x1_port0: 0x0008202C, x4_port0x1_port0: 0x0004002C, x4_port0x1_port1: 0x0004402C, x4_port0x1_port2: 0x0004802C, x4_port0x1_port3: 0x0004C02C, x4_port1x1_port0: 0x0008002C, x4_port1x1_port1: 0x0008402C, x4_port1x1_port2: 0x0008802C, x4_port1x1_port3: 0x0008C02C	Table 810, p. 1107
PCI Express Expansion ROM BAR Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042030, x4_port3x1_port0: 0x00082030, x4_port0x1_port0: 0x00040030, x4_port0x1_port1: 0x00044030, x4_port0x1_port2: 0x00048030, x4_port0x1_port3: 0x0004C030, x4_port1x1_port0: 0x00080030, x4_port1x1_port1: 0x00084030, x4_port1x1_port2: 0x00088030, x4_port1x1_port3: 0x0008C030	Table 811, p. 1108

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Capability List Pointer Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042034, x4_port3x1_port0: 0x00082034, x4_port0x1_port0: 0x00040034, x4_port0x1_port1: 0x00044034, x4_port0x1_port2: 0x00048034, x4_port0x1_port3: 0x0004C034, x4_port1x1_port0: 0x00080034, x4_port1x1_port1: 0x00084034, x4_port1x1_port2: 0x00088034, x4_port1x1_port3: 0x0008C034	Table 812, p. 1108
PCI Express Interrupt Pin and Line Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004203C, x4_port3x1_port0: 0x0008203C, x4_port0x1_port0: 0x0004003C, x4_port0x1_port1: 0x0004403C, x4_port0x1_port2: 0x0004803C, x4_port0x1_port3: 0x0004C03C, x4_port1x1_port0: 0x0008003C, x4_port1x1_port1: 0x0008403C, x4_port1x1_port2: 0x0008803C, x4_port1x1_port3: 0x0008C03C	Table 813, p. 1109
PCI Express Power Management Capability Header Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042040, x4_port3x1_port0: 0x00082040, x4_port0x1_port0: 0x00040040, x4_port0x1_port1: 0x00044040, x4_port0x1_port2: 0x00048040, x4_port0x1_port3: 0x0004C040, x4_port1x1_port0: 0x00080040, x4_port1x1_port1: 0x00084040, x4_port1x1_port2: 0x00088040, x4_port1x1_port3: 0x0008C040	Table 814, p. 1110
PCI Express Power Management Control and Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042044, x4_port3x1_port0: 0x00082044, x4_port0x1_port0: 0x00040044, x4_port0x1_port1: 0x00044044, x4_port0x1_port2: 0x00048044, x4_port0x1_port3: 0x0004C044, x4_port1x1_port0: 0x00080044, x4_port1x1_port1: 0x00084044, x4_port1x1_port2: 0x00088044, x4_port1x1_port3: 0x0008C044	Table 815, p. 1111
PCI Express MSI Message Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042050, x4_port3x1_port0: 0x00082050, x4_port0x1_port0: 0x00040050, x4_port0x1_port1: 0x00044050, x4_port0x1_port2: 0x00048050, x4_port0x1_port3: 0x0004C050, x4_port1x1_port0: 0x00080050, x4_port1x1_port1: 0x00084050, x4_port1x1_port2: 0x00088050, x4_port1x1_port3: 0x0008C050	Table 816, p. 1112

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express MSI Message Address Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042054, x4_port3x1_port0: 0x00082054, x4_port0x1_port0: 0x00040054, x4_port0x1_port1: 0x00044054, x4_port0x1_port2: 0x00048054, x4_port0x1_port3: 0x0004C054, x4_port1x1_port0: 0x00080054, x4_port1x1_port1: 0x00084054, x4_port1x1_port2: 0x00088054, x4_port1x1_port3: 0x0008C054	Table 817, p. 1114
PCI Express MSI Message Address (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042058, x4_port3x1_port0: 0x00082058, x4_port0x1_port0: 0x00040058, x4_port0x1_port1: 0x00044058, x4_port0x1_port2: 0x00048058, x4_port0x1_port3: 0x0004C058, x4_port1x1_port0: 0x00080058, x4_port1x1_port1: 0x00084058, x4_port1x1_port2: 0x00088058, x4_port1x1_port3: 0x0008C058	Table 818, p. 1114
PCI Express MSI Message Data Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004205C, x4_port3x1_port0: 0x0008205C, x4_port0x1_port0: 0x0004005C, x4_port0x1_port1: 0x0004405C, x4_port0x1_port2: 0x0004805C, x4_port0x1_port3: 0x0004C05C, x4_port1x1_port0: 0x0008005C, x4_port1x1_port1: 0x0008405C, x4_port1x1_port2: 0x0008805C, x4_port1x1_port3: 0x0008C05C	Table 819, p. 1114
PCI Express Capability Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042060, x4_port3x1_port0: 0x00082060, x4_port0x1_port0: 0x00040060, x4_port0x1_port1: 0x00044060, x4_port0x1_port2: 0x00048060, x4_port0x1_port3: 0x0004C060, x4_port1x1_port0: 0x00080060, x4_port1x1_port1: 0x00084060, x4_port1x1_port2: 0x00088060, x4_port1x1_port3: 0x0008C060	Table 820, p. 1115
PCI Express Device Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042064, x4_port3x1_port0: 0x00082064, x4_port0x1_port0: 0x00040064, x4_port0x1_port1: 0x00044064, x4_port0x1_port2: 0x00048064, x4_port0x1_port3: 0x0004C064, x4_port1x1_port0: 0x00080064, x4_port1x1_port1: 0x00084064, x4_port1x1_port2: 0x00088064, x4_port1x1_port3: 0x0008C064	Table 821, p. 1116

Copyright © 2014 Marvell

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Device Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042068, x4_port3x1_port0: 0x00082068, x4_port0x1_port0: 0x00040068, x4_port0x1_port1: 0x00044068, x4_port0x1_port2: 0x00048068, x4_port0x1_port3: 0x0004C068, x4_port1x1_port0: 0x00080068, x4_port1x1_port1: 0x00084068, x4_port1x1_port2: 0x00088068, x4_port1x1_port3: 0x0008C068	Table 822, p. 1118
PCI Express Link Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004206C, x4_port3x1_port0: 0x0008206C, x4_port0x1_port0: 0x0004006C, x4_port0x1_port1: 0x0004406C, x4_port0x1_port2: 0x0004806C, x4_port0x1_port3: 0x0004C06C, x4_port1x1_port0: 0x0008006C, x4_port1x1_port1: 0x0008406C, x4_port1x1_port2: 0x0008806C, x4_port1x1_port3: 0x0008C06C	Table 823, p. 1121
PCI Express Link Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042070, x4_port3x1_port0: 0x00082070, x4_port0x1_port0: 0x00040070, x4_port0x1_port1: 0x00044070, x4_port0x1_port2: 0x00048070, x4_port0x1_port3: 0x0004C070, x4_port1x1_port0: 0x00080070, x4_port1x1_port1: 0x00084070, x4_port1x1_port2: 0x00088070, x4_port1x1_port3: 0x0008C070	Table 824, p. 1124
PCI Express Device Capabilities 2 Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042084, x4_port3x1_port0: 0x00082084, x4_port0x1_port0: 0x00040084, x4_port0x1_port1: 0x00044084, x4_port0x1_port2: 0x00048084, x4_port0x1_port3: 0x0004C084, x4_port1x1_port0: 0x00080084, x4_port1x1_port1: 0x00084084, x4_port1x1_port2: 0x00088084, x4_port1x1_port3: 0x0008C084	Table 825, p. 1127
PCI Express Device Control Status 2 Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042088, x4_port3x1_port0: 0x00082088, x4_port0x1_port0: 0x00040088, x4_port0x1_port1: 0x00044088, x4_port0x1_port2: 0x00048088, x4_port0x1_port3: 0x0004C088, x4_port1x1_port0: 0x00080088, x4_port1x1_port1: 0x00084088, x4_port1x1_port2: 0x00088088, x4_port1x1_port3: 0x0008C088	Table 826, p. 1128

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Link Control Status 2 Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042090, x4_port3x1_port0: 0x00082090, x4_port0x1_port0: 0x00040090, x4_port0x1_port1: 0x00044090, x4_port0x1_port2: 0x00048090, x4_port0x1_port3: 0x0004C090, x4_port1x1_port0: 0x00080090, x4_port1x1_port1: 0x00084090, x4_port1x1_port2: 0x00088090, x4_port1x1_port3: 0x0008C090	Table 827, p. 1129
PCI Express Advanced Error Report Header Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042100, x4_port3x1_port0: 0x00082100, x4_port0x1_port0: 0x00040100, x4_port0x1_port1: 0x00044100, x4_port0x1_port2: 0x00048100, x4_port0x1_port3: 0x0004C100, x4_port1x1_port0: 0x00080100, x4_port1x1_port1: 0x00084100, x4_port1x1_port2: 0x00088100, x4_port1x1_port3: 0x0008C100	Table 828, p. 1131
PCI Express Uncorrectable Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042104, x4_port3x1_port0: 0x00082104, x4_port0x1_port0: 0x00040104, x4_port0x1_port1: 0x00044104, x4_port0x1_port2: 0x00048104, x4_port0x1_port3: 0x0004C104, x4_port1x1_port0: 0x00080104, x4_port1x1_port1: 0x00084104, x4_port1x1_port2: 0x00088104, x4_port1x1_port3: 0x0008C104	Table 829, p. 1132
PCI Express Uncorrectable Error Mask Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042108, x4_port3x1_port0: 0x00082108, x4_port0x1_port0: 0x00040108, x4_port0x1_port1: 0x00044108, x4_port0x1_port2: 0x00048108, x4_port0x1_port3: 0x0004C108, x4_port1x1_port0: 0x00080108, x4_port1x1_port1: 0x00084108, x4_port1x1_port2: 0x00088108, x4_port1x1_port3: 0x0008C108	Table 830, p. 1133
PCI Express Uncorrectable Error Severity Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004210C, x4_port3x1_port0: 0x0008210C, x4_port0x1_port0: 0x0004010C, x4_port0x1_port1: 0x0004410C, x4_port0x1_port2: 0x0004810C, x4_port0x1_port3: 0x0004C10C, x4_port1x1_port0: 0x0008010C, x4_port1x1_port1: 0x0008410C, x4_port1x1_port2: 0x0008810C, x4_port1x1_port3: 0x0008C10C	Table 831, p. 1135

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Correctable Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042110, x4_port3x1_port0: 0x00082110, x4_port0x1_port0: 0x00040110, x4_port0x1_port1: 0x00044110, x4_port0x1_port2: 0x00048110, x4_port0x1_port3: 0x0004C110, x4_port1x1_port0: 0x00080110, x4_port1x1_port1: 0x00084110, x4_port1x1_port2: 0x00088110, x4_port1x1_port3: 0x0008C110	Table 832, p. 1136
PCI Express Correctable Error Mask Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042114, x4_port3x1_port0: 0x00082114, x4_port0x1_port0: 0x00040114, x4_port0x1_port1: 0x00044114, x4_port0x1_port2: 0x00048114, x4_port0x1_port3: 0x0004C114, x4_port1x1_port0: 0x00080114, x4_port1x1_port1: 0x00084114, x4_port1x1_port2: 0x00088114, x4_port1x1_port3: 0x0008C114	Table 833, p. 1137
PCI Express Advanced Error Capability and Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042118, x4_port3x1_port0: 0x00082118, x4_port0x1_port0: 0x00040118, x4_port0x1_port1: 0x00044118, x4_port0x1_port2: 0x00048118, x4_port0x1_port3: 0x0004C118, x4_port1x1_port0: 0x00080118, x4_port1x1_port1: 0x00084118, x4_port1x1_port2: 0x00088118, x4_port1x1_port3: 0x0008C118	Table 834, p. 1138
PCI Express Header Log First DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004211C, x4_port3x1_port0: 0x0008211C, x4_port0x1_port0: 0x0004011C, x4_port0x1_port1: 0x0004411C, x4_port0x1_port2: 0x0004811C, x4_port0x1_port3: 0x0004C11C, x4_port1x1_port0: 0x0008011C, x4_port1x1_port1: 0x0008411C, x4_port1x1_port2: 0x0008811C, x4_port1x1_port3: 0x0008C11C	Table 835, p. 1139
PCI Express Header Log Second DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042120, x4_port3x1_port0: 0x00082120, x4_port0x1_port0: 0x00040120, x4_port0x1_port1: 0x00044120, x4_port0x1_port2: 0x00048120, x4_port0x1_port3: 0x0004C120, x4_port1x1_port0: 0x00080120, x4_port1x1_port1: 0x00084120, x4_port1x1_port2: 0x00088120, x4_port1x1_port3: 0x0008C120	Table 836, p. 1139

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Header Log Third DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042124, x4_port3x1_port0: 0x00082124, x4_port0x1_port0: 0x00040124, x4_port0x1_port1: 0x00044124, x4_port0x1_port2: 0x00048124, x4_port0x1_port3: 0x0004C124, x4_port1x1_port0: 0x00080124, x4_port1x1_port1: 0x00084124, x4_port1x1_port2: 0x00088124, x4_port1x1_port3: 0x0008C124	Table 837, p. 1140
PCI Express Header Log Fourth DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042128, x4_port3x1_port0: 0x00082128, x4_port0x1_port0: 0x00040128, x4_port0x1_port1: 0x00044128, x4_port0x1_port2: 0x00048128, x4_port0x1_port3: 0x0004C128, x4_port1x1_port0: 0x00080128, x4_port1x1_port1: 0x00084128, x4_port1x1_port2: 0x00088128, x4_port1x1_port3: 0x0008C128	Table 838, p. 1140
PCI Express Root Error Command Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004212C, x4_port3x1_port0: 0x0008212C, x4_port0x1_port0: 0x0004012C, x4_port0x1_port1: 0x0004412C, x4_port0x1_port2: 0x0004812C, x4_port0x1_port3: 0x0004C12C, x4_port1x1_port0: 0x0008012C, x4_port1x1_port1: 0x0008412C, x4_port1x1_port2: 0x0008812C, x4_port1x1_port3: 0x0008C12C	Table 839, p. 1140
PCI Express Root Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042130, x4_port3x1_port0: 0x00082130, x4_port0x1_port0: 0x00040130, x4_port0x1_port1: 0x00044130, x4_port0x1_port2: 0x00048130, x4_port0x1_port3: 0x0004C130, x4_port1x1_port0: 0x00080130, x4_port1x1_port1: 0x00084130, x4_port1x1_port2: 0x00088130, x4_port1x1_port3: 0x0008C130	Table 840, p. 1141
PCI Express Error Source Identification Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00042134, x4_port3x1_port0: 0x00082134, x4_port0x1_port0: 0x00040134, x4_port0x1_port1: 0x00044134, x4_port0x1_port2: 0x00048134, x4_port0x1_port3: 0x0004C134, x4_port1x1_port0: 0x00080134, x4_port1x1_port1: 0x00084134, x4_port1x1_port2: 0x00088134, x4_port1x1_port3: 0x0008C134	Table 841, p. 1143
PCI Express Address Window Control		

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Window0 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043820, x4_port3x1_port0: 0x00083820, x4_port0x1_port0: 0x00041820, x4_port0x1_port1: 0x00045820, x4_port0x1_port2: 0x00049820, x4_port0x1_port3: 0x0004D820, x4_port1x1_port0: 0x00081820, x4_port1x1_port1: 0x00085820, x4_port1x1_port2: 0x00089820, x4_port1x1_port3: 0x0008D820	Table 842, p. 1143
PCI Express Window0 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043824, x4_port3x1_port0: 0x00083824, x4_port0x1_port0: 0x00041824, x4_port0x1_port1: 0x00045824, x4_port0x1_port2: 0x00049824, x4_port0x1_port3: 0x0004D824, x4_port1x1_port0: 0x00081824, x4_port1x1_port1: 0x00085824, x4_port1x1_port2: 0x00089824, x4_port1x1_port3: 0x0008D824	Table 843, p. 1144
PCI Express Window0 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004382C, x4_port3x1_port0: 0x0008382C, x4_port0x1_port0: 0x0004182C, x4_port0x1_port1: 0x0004582C, x4_port0x1_port2: 0x0004982C, x4_port0x1_port3: 0x0004D82C, x4_port1x1_port0: 0x0008182C, x4_port1x1_port1: 0x0008582C, x4_port1x1_port2: 0x0008982C, x4_port1x1_port3: 0x0008D82C	Table 844, p. 1145
PCI Express Window1 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043830, x4_port3x1_port0: 0x00083830, x4_port0x1_port0: 0x00041830, x4_port0x1_port1: 0x00045830, x4_port0x1_port2: 0x00049830, x4_port0x1_port3: 0x0004D830, x4_port1x1_port0: 0x00081830, x4_port1x1_port1: 0x00085830, x4_port1x1_port2: 0x00089830, x4_port1x1_port3: 0x0008D830	Table 845, p. 1145
PCI Express Window1 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043834, x4_port3x1_port0: 0x00083834, x4_port0x1_port0: 0x00041834, x4_port0x1_port1: 0x00045834, x4_port0x1_port2: 0x00049834, x4_port0x1_port3: 0x0004D834, x4_port1x1_port0: 0x00081834, x4_port1x1_port1: 0x00085834, x4_port1x1_port2: 0x00089834, x4_port1x1_port3: 0x0008D834	Table 846, p. 1146

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Window1 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004383C, x4_port3x1_port0: 0x0008383C, x4_port0x1_port0: 0x0004183C, x4_port0x1_port1: 0x0004583C, x4_port0x1_port2: 0x0004983C, x4_port0x1_port3: 0x0004D83C, x4_port1x1_port0: 0x0008183C, x4_port1x1_port1: 0x0008583C, x4_port1x1_port2: 0x0008983C, x4_port1x1_port3: 0x0008D83C	Table 847, p. 1147
PCI Express Window2 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043840, x4_port3x1_port0: 0x00083840, x4_port0x1_port0: 0x00041840, x4_port0x1_port1: 0x00045840, x4_port0x1_port2: 0x00049840, x4_port0x1_port3: 0x0004D840, x4_port1x1_port0: 0x00081840, x4_port1x1_port1: 0x00085840, x4_port1x1_port2: 0x00089840, x4_port1x1_port3: 0x0008D840	Table 848, p. 1147
PCI Express Window2 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043844, x4_port3x1_port0: 0x00083844, x4_port0x1_port0: 0x00041844, x4_port0x1_port1: 0x00045844, x4_port0x1_port2: 0x00049844, x4_port0x1_port3: 0x0004D844, x4_port1x1_port0: 0x00081844, x4_port1x1_port1: 0x00085844, x4_port1x1_port2: 0x00089844, x4_port1x1_port3: 0x0008D844	Table 849, p. 1148
PCI Express Window2 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004384C, x4_port3x1_port0: 0x0008384C, x4_port0x1_port0: 0x0004184C, x4_port0x1_port1: 0x0004584C, x4_port0x1_port2: 0x0004984C, x4_port0x1_port3: 0x0004D84C, x4_port1x1_port0: 0x0008184C, x4_port1x1_port1: 0x0008584C, x4_port1x1_port2: 0x0008984C, x4_port1x1_port3: 0x0008D84C	Table 850, p. 1149
PCI Express Window3 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043850, x4_port3x1_port0: 0x00083850, x4_port0x1_port0: 0x00041850, x4_port0x1_port1: 0x00045850, x4_port0x1_port2: 0x00049850, x4_port0x1_port3: 0x0004D850, x4_port1x1_port0: 0x00081850, x4_port1x1_port1: 0x00085850, x4_port1x1_port2: 0x00089850, x4_port1x1_port3: 0x0008D850	Table 851, p. 1149

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Window3 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043854, x4_port3x1_port0: 0x00083854, x4_port0x1_port0: 0x00041854, x4_port0x1_port1: 0x00045854, x4_port0x1_port2: 0x00049854, x4_port0x1_port3: 0x0004D854, x4_port1x1_port0: 0x00081854, x4_port1x1_port1: 0x00085854, x4_port1x1_port2: 0x00089854, x4_port1x1_port3: 0x0008D854	Table 852, p. 1150
PCI Express Window3 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004385C, x4_port3x1_port0: 0x0008385C, x4_port0x1_port0: 0x0004185C, x4_port0x1_port1: 0x0004585C, x4_port0x1_port2: 0x0004985C, x4_port0x1_port3: 0x0004D85C, x4_port1x1_port0: 0x0008185C, x4_port1x1_port1: 0x0008585C, x4_port1x1_port2: 0x0008985C, x4_port1x1_port3: 0x0008D85C	Table 853, p. 1151
PCI Express Window4 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043860, x4_port3x1_port0: 0x00083860, x4_port0x1_port0: 0x00041860, x4_port0x1_port1: 0x00045860, x4_port0x1_port2: 0x00049860, x4_port0x1_port3: 0x0004D860, x4_port1x1_port0: 0x00081860, x4_port1x1_port1: 0x00085860, x4_port1x1_port2: 0x00089860, x4_port1x1_port3: 0x0008D860	Table 854, p. 1151
PCI Express Window4 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043864, x4_port3x1_port0: 0x00083864, x4_port0x1_port0: 0x00041864, x4_port0x1_port1: 0x00045864, x4_port0x1_port2: 0x00049864, x4_port0x1_port3: 0x0004D864, x4_port1x1_port0: 0x00081864, x4_port1x1_port1: 0x00085864, x4_port1x1_port2: 0x00089864, x4_port1x1_port3: 0x0008D864	Table 855, p. 1152
PCI Express Window4 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004386C, x4_port3x1_port0: 0x0008386C, x4_port0x1_port0: 0x0004186C, x4_port0x1_port1: 0x0004586C, x4_port0x1_port2: 0x0004986C, x4_port0x1_port3: 0x0004D86C, x4_port1x1_port0: 0x0008186C, x4_port1x1_port1: 0x0008586C, x4_port1x1_port2: 0x0008986C, x4_port1x1_port3: 0x0008D86C	Table 856, p. 1152

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Window4 Remap (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043870, x4_port3x1_port0: 0x00083870, x4_port0x1_port0: 0x00041870, x4_port0x1_port1: 0x00045870, x4_port0x1_port2: 0x00049870, x4_port0x1_port3: 0x0004D870, x4_port1x1_port0: 0x00081870, x4_port1x1_port1: 0x00085870, x4_port1x1_port2: 0x00089870, x4_port1x1_port3: 0x0008D870	Table 857, p. 1153
PCI Express Window5 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043880, x4_port3x1_port0: 0x00083880, x4_port0x1_port0: 0x00041880, x4_port0x1_port1: 0x00045880, x4_port0x1_port2: 0x00049880, x4_port0x1_port3: 0x0004D880, x4_port1x1_port0: 0x00081880, x4_port1x1_port1: 0x00085880, x4_port1x1_port2: 0x00089880, x4_port1x1_port3: 0x0008D880	Table 858, p. 1154
PCI Express Window5 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043884, x4_port3x1_port0: 0x00083884, x4_port0x1_port0: 0x00041884, x4_port0x1_port1: 0x00045884, x4_port0x1_port2: 0x00049884, x4_port0x1_port3: 0x0004D884, x4_port1x1_port0: 0x00081884, x4_port1x1_port1: 0x00085884, x4_port1x1_port2: 0x00089884, x4_port1x1_port3: 0x0008D884	Table 859, p. 1155
PCI Express Window5 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004388C, x4_port3x1_port0: 0x0008388C, x4_port0x1_port0: 0x0004188C, x4_port0x1_port1: 0x0004588C, x4_port0x1_port2: 0x0004988C, x4_port0x1_port3: 0x0004D88C, x4_port1x1_port0: 0x0008188C, x4_port1x1_port1: 0x0008588C, x4_port1x1_port2: 0x0008988C, x4_port1x1_port3: 0x0008D88C	Table 860, p. 1155
PCI Express Window5 Remap (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043890, x4_port3x1_port0: 0x00083890, x4_port0x1_port0: 0x00041890, x4_port0x1_port1: 0x00045890, x4_port0x1_port2: 0x00049890, x4_port0x1_port3: 0x0004D890, x4_port1x1_port0: 0x00081890, x4_port1x1_port1: 0x00085890, x4_port1x1_port2: 0x00089890, x4_port1x1_port3: 0x0008D890	Table 861, p. 1156

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Default Window Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438B0, x4_port3x1_port0: 0x000838B0, x4_port0x1_port0: 0x000418B0, x4_port0x1_port1: 0x000458B0, x4_port0x1_port2: 0x000498B0, x4_port0x1_port3: 0x0004D8B0, x4_port1x1_port0: 0x000818B0, x4_port1x1_port1: 0x000858B0, x4_port1x1_port2: 0x000898B0, x4_port1x1_port3: 0x0008D8B0	Table 862, p. 1156
PCI Express Expansion ROM Window Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438C0, x4_port3x1_port0: 0x000838C0, x4_port0x1_port0: 0x000418C0, x4_port0x1_port1: 0x000458C0, x4_port0x1_port2: 0x000498C0, x4_port0x1_port3: 0x0004D8C0, x4_port1x1_port0: 0x000818C0, x4_port1x1_port1: 0x000858C0, x4_port1x1_port2: 0x000898C0, x4_port1x1_port3: 0x0008D8C0	Table 863, p. 1157
PCI Express Expansion ROM Window Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438C4, x4_port3x1_port0: 0x000838C4, x4_port0x1_port0: 0x000418C4, x4_port0x1_port1: 0x000458C4, x4_port0x1_port2: 0x000498C4, x4_port0x1_port3: 0x0004D8C4, x4_port1x1_port0: 0x000818C4, x4_port1x1_port1: 0x000858C4, x4_port1x1_port2: 0x000898C4, x4_port1x1_port3: 0x0008D8C4	Table 864, p. 1157
PCI Express Mbus Control		
PCI Express Mbus Adapter Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438D0, x4_port3x1_port0: 0x000838D0, x4_port0x1_port0: 0x000418D0, x4_port0x1_port1: 0x000458D0, x4_port0x1_port2: 0x000498D0, x4_port0x1_port3: 0x0004D8D0, x4_port1x1_port0: 0x000818D0, x4_port1x1_port1: 0x000858D0, x4_port1x1_port2: 0x000898D0, x4_port1x1_port3: 0x0008D8D0	Table 865, p. 1158
PCI Express Mbus Arbiter Control Register (Low) (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438E0, x4_port3x1_port0: 0x000838E0, x4_port0x1_port0: 0x000418E0, x4_port0x1_port1: 0x000458E0, x4_port0x1_port2: 0x000498E0, x4_port0x1_port3: 0x0004D8E0, x4_port1x1_port0: 0x000818E0, x4_port1x1_port1: 0x000858E0, x4_port1x1_port2: 0x000898E0, x4_port1x1_port3: 0x0008D8E0	Table 866, p. 1159

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Mbus Arbiter Control Register (High) (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438E4, x4_port3x1_port0: 0x000838E4, x4_port0x1_port0: 0x000418E4, x4_port0x1_port1: 0x000458E4, x4_port0x1_port2: 0x000498E4, x4_port0x1_port3: 0x0004D8E4, x4_port1x1_port0: 0x000818E4, x4_port1x1_port1: 0x000858E4, x4_port1x1_port2: 0x000898E4, x4_port1x1_port3: 0x0008D8E4	Table 867, p. 1159
PCI Express Mbus Arbiter Timeout Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x000438E8, x4_port3x1_port0: 0x000838E8, x4_port0x1_port0: 0x000418E8, x4_port0x1_port1: 0x000458E8, x4_port0x1_port2: 0x000498E8, x4_port0x1_port3: 0x0004D8E8, x4_port1x1_port0: 0x000818E8, x4_port1x1_port1: 0x000858E8, x4_port1x1_port2: 0x000898E8, x4_port1x1_port3: 0x0008D8E8	Table 868, p. 1160
PCI Express Control and Status		
PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A00, x4_port3x1_port0: 0x00083A00, x4_port0x1_port0: 0x00041A00, x4_port0x1_port1: 0x00045A00, x4_port0x1_port2: 0x00049A00, x4_port0x1_port3: 0x0004DA00, x4_port1x1_port0: 0x00081A00, x4_port1x1_port1: 0x00085A00, x4_port1x1_port2: 0x00089A00, x4_port1x1_port3: 0x0008DA00	Table 869, p. 1161
PCI Express Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A04, x4_port3x1_port0: 0x00083A04, x4_port0x1_port0: 0x00041A04, x4_port0x1_port1: 0x00045A04, x4_port0x1_port2: 0x00049A04, x4_port0x1_port3: 0x0004DA04, x4_port1x1_port0: 0x00081A04, x4_port1x1_port1: 0x00085A04, x4_port1x1_port2: 0x00089A04, x4_port1x1_port3: 0x0008DA04	Table 870, p. 1163
PCI Express Root Complex Set Slot Power Limit (SSPL) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A0C, x4_port3x1_port0: 0x00083A0C, x4_port0x1_port0: 0x00041A0C, x4_port0x1_port1: 0x00045A0C, x4_port0x1_port2: 0x00049A0C, x4_port0x1_port3: 0x0004DA0C, x4_port1x1_port0: 0x00081A0C, x4_port1x1_port1: 0x00085A0C, x4_port1x1_port2: 0x00089A0C, x4_port1x1_port3: 0x0008DA0C	Table 871, p. 1164

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Completion Timeout Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A10, x4_port3x1_port0: 0x00083A10, x4_port0x1_port0: 0x00041A10, x4_port0x1_port1: 0x00045A10, x4_port0x1_port2: 0x00049A10, x4_port0x1_port3: 0x0004DA10, x4_port1x1_port0: 0x00081A10, x4_port1x1_port1: 0x00085A10, x4_port1x1_port2: 0x00089A10, x4_port1x1_port3: 0x0008DA10	Table 872, p. 1166
PCI Express Root Complex Power Management Event (PME) Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A14, x4_port3x1_port0: 0x00083A14, x4_port0x1_port0: 0x00041A14, x4_port0x1_port1: 0x00045A14, x4_port0x1_port2: 0x00049A14, x4_port0x1_port3: 0x0004DA14, x4_port1x1_port0: 0x00081A14, x4_port1x1_port1: 0x00085A14, x4_port1x1_port2: 0x00089A14, x4_port1x1_port3: 0x0008DA14	Table 873, p. 1166
PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A18, x4_port3x1_port0: 0x00083A18, x4_port0x1_port0: 0x00041A18, x4_port0x1_port1: 0x00045A18, x4_port0x1_port2: 0x00049A18, x4_port0x1_port3: 0x0004DA18, x4_port1x1_port0: 0x00081A18, x4_port1x1_port1: 0x00085A18, x4_port1x1_port2: 0x00089A18, x4_port1x1_port3: 0x0008DA18	Table 874, p. 1167
PCI Express Flow Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A20, x4_port3x1_port0: 0x00083A20, x4_port0x1_port0: 0x00041A20, x4_port0x1_port1: 0x00045A20, x4_port0x1_port2: 0x00049A20, x4_port0x1_port3: 0x0004DA20, x4_port1x1_port0: 0x00081A20, x4_port1x1_port1: 0x00085A20, x4_port1x1_port2: 0x00089A20, x4_port1x1_port3: 0x0008DA20	Table 875, p. 1168
PCI Express Replay Timeout (TO) Timer Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A24, x4_port3x1_port0: 0x00083A24, x4_port0x1_port0: 0x00041A24, x4_port0x1_port1: 0x00045A24, x4_port0x1_port2: 0x00049A24, x4_port0x1_port3: 0x0004DA24, x4_port1x1_port0: 0x00081A24, x4_port1x1_port1: 0x00085A24, x4_port1x1_port2: 0x00089A24, x4_port1x1_port3: 0x0008DA24	Table 876, p. 1169

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Ack Latency Timer Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A28, x4_port3x1_port0: 0x00083A28, x4_port0x1_port0: 0x00041A28, x4_port0x1_port1: 0x00045A28, x4_port0x1_port2: 0x00049A28, x4_port0x1_port3: 0x0004DA28, x4_port1x1_port0: 0x00081A28, x4_port1x1_port1: 0x00085A28, x4_port1x1_port2: 0x00089A28, x4_port1x1_port3: 0x0008DA28	Table 877, p. 1170
PCI Express Secondary Bus Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A2C, x4_port3x1_port0: 0x00083A2C, x4_port0x1_port0: 0x00041A2C, x4_port0x1_port1: 0x00045A2C, x4_port0x1_port2: 0x00049A2C, x4_port0x1_port3: 0x0004DA2C, x4_port1x1_port0: 0x00081A2C, x4_port1x1_port1: 0x00085A2C, x4_port1x1_port2: 0x00089A2C, x4_port1x1_port3: 0x0008DA2C	Table 878, p. 1170
PCI Express Dynamic Width Management Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A30, x4_port3x1_port0: 0x00083A30, x4_port0x1_port0: 0x00041A30, x4_port0x1_port1: 0x00045A30, x4_port0x1_port2: 0x00049A30, x4_port0x1_port3: 0x0004DA30, x4_port1x1_port0: 0x00081A30, x4_port1x1_port1: 0x00085A30, x4_port1x1_port2: 0x00089A30, x4_port1x1_port3: 0x0008DA30	Table 879, p. 1171
PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043B00, x4_port3x1_port0: 0x00083B00, x4_port0x1_port0: 0x00041B00, x4_port0x1_port1: 0x00045B00, x4_port0x1_port2: 0x00049B00, x4_port0x1_port3: 0x0004DB00, x4_port1x1_port0: 0x00081B00, x4_port1x1_port1: 0x00085B00, x4_port1x1_port2: 0x00089B00, x4_port1x1_port3: 0x0008DB00	Table 880, p. 1172
PCI Express BAR Control		
PCI Express BAR1 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043804, x4_port3x1_port0: 0x00083804, x4_port0x1_port0: 0x00041804, x4_port0x1_port1: 0x00045804, x4_port0x1_port2: 0x00049804, x4_port0x1_port3: 0x0004D804, x4_port1x1_port0: 0x00081804, x4_port1x1_port1: 0x00085804, x4_port1x1_port2: 0x00089804, x4_port1x1_port3: 0x0008D804	Table 881, p. 1173

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express BAR2 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043808, x4_port3x1_port0: 0x00083808, x4_port0x1_port0: 0x00041808, x4_port0x1_port1: 0x00045808, x4_port0x1_port2: 0x00049808, x4_port0x1_port3: 0x0004D808, x4_port1x1_port0: 0x00081808, x4_port1x1_port1: 0x00085808, x4_port1x1_port2: 0x00089808, x4_port1x1_port3: 0x0008D808	Table 882, p. 1173
PCI Express Expansion ROM BAR Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x0004380C, x4_port3x1_port0: 0x0008380C, x4_port0x1_port0: 0x0004180C, x4_port0x1_port1: 0x0004580C, x4_port0x1_port2: 0x0004980C, x4_port0x1_port3: 0x0004D80C, x4_port1x1_port0: 0x0008180C, x4_port1x1_port1: 0x0008580C, x4_port1x1_port2: 0x0008980C, x4_port1x1_port3: 0x0008D80C	Table 883, p. 1174
PCI Express Interrupt		
PCI Express Interrupt Cause Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043900, x4_port3x1_port0: 0x00083900, x4_port0x1_port0: 0x00041900, x4_port0x1_port1: 0x00045900, x4_port0x1_port2: 0x00049900, x4_port0x1_port3: 0x0004D900, x4_port1x1_port0: 0x00081900, x4_port1x1_port1: 0x00085900, x4_port1x1_port2: 0x00089900, x4_port1x1_port3: 0x0008D900	Table 884, p. 1174
PCI Express Interrupt Mask Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043910, x4_port3x1_port0: 0x00083910, x4_port0x1_port0: 0x00041910, x4_port0x1_port1: 0x00045910, x4_port0x1_port2: 0x00049910, x4_port0x1_port3: 0x0004D910, x4_port1x1_port0: 0x00081910, x4_port1x1_port1: 0x00085910, x4_port1x1_port2: 0x00089910, x4_port1x1_port3: 0x0008D910	Table 885, p. 1178
PCI Express Debug		
PCI Express Debug Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A60, x4_port3x1_port0: 0x00083A60, x4_port0x1_port0: 0x00041A60, x4_port0x1_port1: 0x00045A60, x4_port0x1_port2: 0x00049A60, x4_port0x1_port3: 0x0004DA60, x4_port1x1_port0: 0x00081A60, x4_port1x1_port1: 0x00085A60, x4_port1x1_port2: 0x00089A60, x4_port1x1_port3: 0x0008DA60	Table 886, p. 1179

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
PCI Express Debug Status Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A64, x4_port3x1_port0: 0x00083A64, x4_port0x1_port0: 0x00041A64, x4_port0x1_port1: 0x00045A64, x4_port0x1_port2: 0x00049A64, x4_port0x1_port3: 0x0004DA64, x4_port1x1_port0: 0x00081A64, x4_port1x1_port1: 0x00085A64, x4_port1x1_port2: 0x00089A64, x4_port1x1_port3: 0x0008DA64	Table 887, p. 1181
PCI Express Debug MAC Status 1 Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043A84, x4_port3x1_port0: 0x00083A84, x4_port0x1_port0: 0x00041A84, x4_port0x1_port1: 0x00045A84, x4_port0x1_port2: 0x00049A84, x4_port0x1_port3: 0x0004DA84, x4_port1x1_port0: 0x00081A84, x4_port1x1_port1: 0x00085A84, x4_port1x1_port2: 0x00089A84, x4_port1x1_port3: 0x0008DA84	Table 888, p. 1181
PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1)	x4_port2x1_port0: 0x00043AB0, x4_port3x1_port0: 0x00083AB0, x4_port0x1_port0: 0x00041AB0, x4_port0x1_port1: 0x00045AB0, x4_port0x1_port2: 0x00049AB0, x4_port0x1_port3: 0x0004DAB0, x4_port1x1_port0: 0x00081AB0, x4_port1x1_port1: 0x00085AB0, x4_port1x1_port2: 0x00089AB0, x4_port1x1_port3: 0x0008DAB0	Table 889, p. 1182
PCI Express Communication PHY		
Power and PLL Control Register	0x00000004	Table 890, p. 1183
KVCO Calibration Control Register	0x00000008	Table 891, p. 1184
PHY Test Control 0 Register	0x00000054	Table 892, p. 1184
PHY Test PRBS Error Counter 0 Register	0x0000007C	Table 893, p. 1186
PHY Test PRBS Error Counter 1 Register	0x00000080	Table 894, p. 1186
PHY Test OOB 0 Register	0x00000084	Table 895, p. 1186
Digital Loopback Enable Register	0x0000008C	Table 896, p. 1187
PHY Isolation Mode Control Register	0x00000098	Table 897, p. 1188
Reference Clock Select Register	0x00000118	Table 898, p. 1188
COMPHY Control Register	0x00000120	Table 899, p. 1189
PCI Express PHY Pipe		
LANE_CFG0 Lane Configuration 0 Register	0x00000080	Table 900, p. 1189
LANE_CFG1 Lane Configuration 1 Register	0x00000081	Table 901, p. 1190
LANE_BEACON Lane Beacon Control Register	0x00000085	Table 902, p. 1191
GLOB_CLK_CTRL Reset and Clock Control Register	0x000000C1	Table 903, p. 1192
GLOB_TEST_CTRL Test Mode Control Register	0x000000C2	Table 904, p. 1193

Table 797: Summary Map Table for the PCI Express 2.0 Registers (Continued)

Register Name	Offset	Table and Page
GLOB_CLK_SRC_LO Clock Source Low Register	0x000000C3	Table 905, p. 1194
GLOB_TRIGGER Register	0x000000C5	Table 906, p. 1194
GLOB_DP_CFG Datapath Configuration Register	0x000000C8	Table 907, p. 1195
GLOB_PM_CFG0 Power Management Timing Parameter Register	0x000000D0	Table 908, p. 1196
GLOB_PM_CFG1 Power Management Timing Parameter 2 Register	0x000000D1	Table 909, p. 1197
GLOB_COUNTER_CTRL Counter Lane and Type Register		Table 910, p. 1197
GLOB_COUNTER_LO Counter Low Register		Table 911, p. 1198
GLOB_COUNTER_HI Counter High Register		Table 912, p. 1198
LANE_CFG0 Lane Configuration 0 Register	0x00000180	Table 913, p. 1198
LANE_CFG1 Lane Configuration 1 Register	0x00000181	Table 914, p. 1199
LANE_BEACON Lane Beacon Control Register	0x00000185	Table 915, p. 1200
LANE_CFG0 Lane Configuration 0 Register	0x00000280	Table 916, p. 1201
LANE_CFG1 Lane Configuration 1 Register	0x00000281	Table 917, p. 1202
LANE_BEACON Lane Beacon Control Register	0x00000285	Table 918, p. 1203
LANE_CFG0 Lane Configuration 0 Register	0x00000380	Table 919, p. 1204
LANE_CFG1 Lane Configuration 1 Register	0x00000381	Table 920, p. 1205
LANE_BEACON Lane Beacon Control Register	0x00000385	Table 921, p. 1206

A.9.1 PCI Express Configuration Cycles Generation

Table 798: PCI Express Configuration Address Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x000438F8, x4_port3x1_port0: 0x000838F8, x4_port0x1_port0: 0x000418F8, x4_port0x1_port1: 0x000458F8, x4_port0x1_port2: 0x000498F8, x4_port0x1_port3: 0x0004D8F8, x4_port1x1_port0: 0x000818F8, x4_port1x1_port1: 0x000858F8, x4_port1x1_port2: 0x000898F8, x4_port1x1_port3: 0x0008D8F8

Offset Formula: 0x000418F8+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x000418F8+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31	ConfigEn	RW 0x0	Configuration Enable Bit
30:28	Reserved	RSVD 0x0	Reserved
27:24	ExtRegNum	RW 0x0	Extended Register Number

Table 798: PCI Express Configuration Address Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x000438F8, x4_port3x1_port0: 0x000838F8, x4_port0x1_port0: 0x000418F8, x4_port0x1_port1: 0x000458F8, x4_port0x1_port2: 0x000498F8, x4_port0x1_port3: 0x0004D8F8, x4_port1x1_port0: 0x000818F8, x4_port1x1_port1: 0x000858F8, x4_port1x1_port2: 0x000898F8, x4_port1x1_port3: 0x0008D8F8

Offset Formula: 0x000418F8+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x000418F8+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
23:16	BusNum	RW 0x0	Bus Number
15:11	DevNum	RW 0x0	Device Number
10:8	FunctNum	RW 0x0	Function Number
7:2	RegNum	RW 0x0	Register Number
1:0	Reserved	RSVD 0x0	Reserved

Table 799: PCI Express Configuration Data Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x000438FC, x4_port3x1_port0: 0x000838FC, x4_port0x1_port0: 0x000418FC, x4_port0x1_port1: 0x000458FC, x4_port0x1_port2: 0x000498FC, x4_port0x1_port3: 0x0004D8FC, x4_port1x1_port0: 0x000818FC, x4_port1x1_port1: 0x000858FC, x4_port1x1_port2: 0x000898FC, x4_port1x1_port3: 0x0008D8FC

Offset Formula: 0x000418FC+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x000418FC+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	ConfigData	RW 0x0	Write access to this register generates a corresponding Configuration TLP on the PCI Express port or an access to the PCI Express port configuration header registers.

A.9.2 PCI Express Configuration Header

Table 800: PCI Express Device and Vendor ID Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042000, x4_port3x1_port0: 0x00082000, x4_port0x1_port0: 0x00040000, x4_port0x1_port1: 0x00044000, x4_port0x1_port2: 0x00048000, x4_port0x1_port3: 0x0004C000, x4_port1x1_port0: 0x00080000, x4_port1x1_port1: 0x00084000, x4_port1x1_port2: 0x00088000, x4_port1x1_port3: 0x0008C000

Offset Formula: 0x00040000+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040000+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	DevID	RO 0x7846	Device ID
15:0	VenID	RO 0x11AB	Vendor ID This field identifies Marvell as the Vendor of the device.

Table 801: PCI Express Command and Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042004, x4_port3x1_port0: 0x00082004, x4_port0x1_port0: 0x00040004, x4_port0x1_port1: 0x00044004, x4_port0x1_port2: 0x00048004, x4_port0x1_port3: 0x0004C004, x4_port1x1_port0: 0x00080004, x4_port1x1_port1: 0x00084004, x4_port1x1_port2: 0x00088004, x4_port1x1_port3: 0x0008C004

Offset Formula: 0x00040004+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040004+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31	DetParErr	RW1C 0x0	Detected Parity Error This bit is set when the device receives a poisoned TLP. **NOTE:** The bit is set regardless of the state of the <PErrEn> bit in this register. Write 1 to clear.
30	SSysErr	RW1C 0x0	Signalled System Error This bit is set when the device sends an ERR_FATAL or ERR_NONFATAL message. This bit is not set if the <SErrEn> field in this register is de-asserted. Write 1 to clear.
29	RMAbort	RW1C 0x0	Received Master Abort This bit is set when the device, as a requester (master), receives a completion with the status Unsupported Request. Write 1 to clear.

Table 801: PCI Express Command and Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042004, x4_port3x1_port0: 0x00082004, x4_port0x1_port0: 0x00040004, x4_port0x1_port1: 0x00044004, x4_port0x1_port2: 0x00048004, x4_port0x1_port3: 0x0004C004, x4_port1x1_port0: 0x00080004, x4_port1x1_port1: 0x00084004, x4_port1x1_port2: 0x00088004, x4_port1x1_port3: 0x0008C004**

Offset Formula: **0x00040004+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040004+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
28	RTAbort	RW1C 0x0	Received Target Abort This bit is set when the device, as a requester (master), receives a completion with the status Completer Abort. Write 1 to clear.
27	STarAbort	RW1C 0x0	Signaled Target Abort This bit is set when the device, as a completer (target), completes a transaction as a Completer Abort. Write 1 to clear.
26:25	Reserved	RSVD 0x0	Does not apply to PCI Express. These bits are hardwired to 0.
24	MasDataPerr	RW1C 0x0	Master Data Parity Error This bit is set when the device has poisoned data detected as a requestor (master). Set when the <PErrEn> field (bit[6]) is set and either: Poisoned completion is received for the PCI Express port. or Poisoned TLP is transmitted to the PCI Express port. Write 1 to clear.
23:21	Reserved	RSVD 0x0	These bits are hardwired to 0.
20	CapList	RO 0x1	Capability List Support This bit indicates that the device configuration header includes capability list. This bit is hardwired to 1, since this is always supported in PCI Express.
19	IntStat	RO 0x0	Interrupt Status When set, this bit indicates that an interrupt message is pending internally in the device. 0 = Disable: No interrupt asserted. 1 = Enable: Interrupt asserted.
18:11	Reserved	RSVD 0x0	These bits are hardwired to 0.

Table 801: PCI Express Command and Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042004, x4_port3x1_port0: 0x00082004, x4_port0x1_port0: 0x00040004, x4_port0x1_port1: 0x00044004, x4_port0x1_port2: 0x00048004, x4_port0x1_port3: 0x0004C004, x4_port1x1_port0: 0x00080004, x4_port1x1_port1: 0x00084004, x4_port1x1_port2: 0x00088004, x4_port1x1_port3: 0x0008C004

Offset Formula: 0x00040004+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040004+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
10	IntDis	RW 0x0	Interrupt Disable This bit controls the ability of the device to generate interrupt emulation messages. When set, interrupt messages are not generated. 0 = Enable: Interrupt messages enabled. 1 = Disable: Interrupt messages disabled.
9	Reserved	RSVD 0x0	Does not apply to PCI Express. This bit is hardwired to 0.
8	SErrEn	RW 0x0	This bit controls the ability of the device to report fatal and non-fatal errors to the Root Complex. This bit affects both assertion of SSysErr status bit[30] in this register and uncorrectable error message generation. **NOTE:** PCI Express uncorrectable error messages are reported if enabled either through this field or through fields <NFErrRepEn> or <FErrRepEn> in the PCI Express Device Control Status Register. 0 = Disable: SSysErr assertion is disabled. 1 = Enable: SSysErr assertion is enabled. In addition uncorrectable error messages generation is enabled.
7	Reserved	RSVD 0x0	Does not apply to PCI Express. This bit is hardwired to 0.
6	PErrEn	RW 0x0	Parity Error Enable This bit controls the ability of the device to respond to poisoned data errors as a requestor (master) on the PCI Express port. Controls MasDataPerr status bit assertion in the PCMDSTT register. **NOTE:** The setting of this bit does not affect the DetectedPErr status bit. 0 = Disable: MasDataPerr assertion is disabled. 1 = Enable: MasDataPerr assertion is enabled.
5:3	Reserved	RSVD 0x0	Does not apply to PCI Express devices. These bits are hardwired to 0.

Table 801: PCI Express Command and Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042004, x4_port3x1_port0: 0x00082004, x4_port0x1_port0: 0x00040004, x4_port0x1_port1: 0x00044004, x4_port0x1_port2: 0x00048004, x4_port0x1_port3: 0x0004C004, x4_port1x1_port0: 0x00080004, x4_port1x1_port1: 0x00084004, x4_port1x1_port2: 0x00088004, x4_port1x1_port3: 0x0008C004**

Offset Formula: **0x00040004+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040004+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
2	MasEn	RW 0x0	Master Enable This bit controls the ability of the device to act as a master on the PCI Express port. When set to 0, no memory or I/O read/write request packets are generated to PCI Express. **NOTE:** Messages and completions are transmitted to the PCI Express regardless of the setting of this bit. 0 = Disable: Neither memory nor I/O requests are transmitted to the PCI-E port. 1 = Enable: Memory and I/O requests are transmitted to the PCI-E port.
1	MemEn	RW 0x0	Memory Space Enable Controls the device response to PCI Express memory accesses. 0 = Disable: All Memory accesses from PCI Express are completed as Unsupported Requests. 1 = Enable
0	IOEn	RW 0x0	I/O Space Enable The I/O space is not enabled; therefore, this bit is not functional. 0 = Disable 1 = Enable

Table 802: PCI Express Class Code and Revision ID Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042008, x4_port3x1_port0: 0x00082008, x4_port0x1_port0: 0x00040008, x4_port0x1_port1: 0x00044008, x4_port0x1_port2: 0x00048008, x4_port0x1_port3: 0x0004C008, x4_port1x1_port0: 0x00080008, x4_port1x1_port1: 0x00084008, x4_port1x1_port2: 0x00088008, x4_port1x1_port3: 0x0008C008**

Offset Formula: **0x00040008+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040008+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:24	BaseClass	RO 0x05	Device Base Class - Memory Controller
23:16	SubClass	RO 0x80	Device Sub class--Other Memory Controller

Copyright © 2014 Marvell

Table 802: PCI Express Class Code and Revision ID Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042008, x4_port3x1_port0: 0x00082008, x4_port0x1_port0: 0x00040008, x4_port0x1_port1: 0x00044008, x4_port0x1_port2: 0x00048008, x4_port0x1_port3: 0x0004C008, x4_port1x1_port0: 0x00080008, x4_port1x1_port1: 0x00084008, x4_port1x1_port2: 0x00088008, x4_port1x1_port3: 0x0008C008

Offset Formula: 0x00040008+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040008+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port,

Bit	Field	Type/InitVal	Description
15:8	ProgIF	RO 0x0	Register Level Programming Interface
7:0	RevID	RO 0x0	Device Revision Number

Table 803: PCI Express BIST Header Type and Cache Line Size Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004200C, x4_port3x1_port0: 0x0008200C, x4_port0x1_port0: 0x0004000C, x4_port0x1_port1: 0x0004400C, x4_port0x1_port2: 0x0004800C, x4_port0x1_port3: 0x0004C00C, x4_port1x1_port0: 0x0008000C, x4_port1x1_port1: 0x0008400C, x4_port1x1_port2: 0x0008800C, x4_port1x1_port3: 0x0008C00C

Offset Formula: 0x0004000C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004000C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31	BISTCap	RO 0x0	BIST Capable Bit BIST is not supported. **NOTE:** This bit must be set to 0. **NOTE:**
30	BISTAct	RO 0x0	BIST Activate bit BIST is not supported.
29:28	Reserved	RSVD 0x0	Reserved
27:24	BISTComp	RO 0x0	BIST Completion Code 0x0 = BIST passed. Other = BIST failed.
23:16	HeadType	RO 0x0	Device Configuration Header Type The header is a Type 0 single-function configuration header.
15:8	Reserved	RSVD 0x0	Does not apply to PCI Express. These bits are hardwired to 0.

Table 803: PCI Express BIST Header Type and Cache Line Size Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x0004200C, x4_port3x1_port0: 0x0008200C, x4_port0x1_port0: 0x0004000C, x4_port0x1_port1: 0x0004400C, x4_port0x1_port2: 0x0004800C, x4_port0x1_port3: 0x0004C00C, x4_port1x1_port0: 0x0008000C, x4_port1x1_port1: 0x0008400C, x4_port1x1_port2: 0x0008800C, x4_port1x1_port3: 0x0008C00C**

Offset Formula: **0x0004000C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x0004000C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
7:0	CacheLine	RW 0x00	Device Cache Line Size **NOTE:**

Table 804: PCI Express BAR0 Internal Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042010, x4_port3x1_port0: 0x00082010, x4_port0x1_port0: 0x00040010, x4_port0x1_port1: 0x00044010, x4_port0x1_port2: 0x00048010, x4_port0x1_port3: 0x0004C010, x4_port1x1_port0: 0x00080010, x4_port1x1_port1: 0x00084010, x4_port1x1_port2: 0x00088010, x4_port1x1_port3: 0x0008C010**

Offset Formula: **0x00040010+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040010+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:20	Base	RW 0xD00	Internal register memory Base Address Indicates 1-MByte address space. Corresponds to address bits[31:20].
19:4	Reserved	RSVD 0x0	Reserved
3	Prefetch	RO 0x1	Prefetch Enable Indicates that prefetching is enabled.
2:1	Type	RO 0x2	BAR Type. BAR can be located in 64-bit memory address space.
0	Space	RO 0x0	Memory Space Indicator.

Table 805: PCI Express BAR0 Internal (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042014, x4_port3x1_port0: 0x00082014, x4_port0x1_port0: 0x00040014, x4_port0x1_port1: 0x00044014, x4_port0x1_port2: 0x00048014, x4_port0x1_port3: 0x0004C014, x4_port1x1_port0: 0x00080014, x4_port1x1_port1: 0x00084014, x4_port1x1_port2: 0x00088014, x4_port1x1_port3: 0x0008C014**

Offset Formula: **0x00040014+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040014+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits[63:32].

Table 806: PCI Express BAR1 Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042018, x4_port3x1_port0: 0x00082018, x4_port0x1_port0: 0x00040018, x4_port0x1_port1: 0x00044018, x4_port0x1_port2: 0x00048018, x4_port0x1_port3: 0x0004C018, x4_port1x1_port0: 0x00080018, x4_port1x1_port1: 0x00084018, x4_port1x1_port2: 0x00088018, x4_port1x1_port3: 0x0008C018**

Offset Formula: **0x00040018+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040018+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base/Reserved	RW 0x0000	Base address Defined according to PCI Express BAR1 Control Register. Indicates 64-KByte to 4-GByte address space. Corresponds to address bits[31:16].
15:4	Reserved	RSVD 0x0	Reserved
3	Prefetch	RO 0x1	Prefetch Enable Indicates that prefetching is enabled.
2:1	Type	RO 0x2	BAR Type BAR can be located in 64-bit memory address space.
0	Space	RO 0x0	Memory Space Indicator

Table 807: PCI Express BAR1 (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x0004201C, x4_port3x1_port0: 0x0008201C, x4_port0x1_port0: 0x0004001C, x4_port0x1_port1: 0x0004401C, x4_port0x1_port2: 0x0004801C, x4_port0x1_port3: 0x0004C01C, x4_port1x1_port0: 0x0008001C, x4_port1x1_port1: 0x0008401C, x4_port1x1_port2: 0x0008801C, x4_port1x1_port3: 0x0008C01C**

Offset Formula: **0x0004001C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x0004001C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits[63:32].

Table 808: PCI Express BAR2 Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042020, x4_port3x1_port0: 0x00082020, x4_port0x1_port0: 0x00040020, x4_port0x1_port1: 0x00044020, x4_port0x1_port2: 0x00048020, x4_port0x1_port3: 0x0004C020, x4_port1x1_port0: 0x00080020, x4_port1x1_port1: 0x00084020, x4_port1x1_port2: 0x00088020, x4_port1x1_port3: 0x0008C020**

Offset Formula: **0x00040020+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040020+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base/Reserved	RW 0xF000	Base address Defined according to PCI Express BAR2 Control Register. Indicates 64-KByte up to 4-GByte address space. Corresponds to address bits[31:16].
15:4	Reserved	RSVD 0x0	Reserved
3	Prefetch	RO 0x1	Prefetch Enable Indicates that pre-fetching is enabled.
2:1	Type	RO 0x2	BAR Type BAR can be located in 64-bit memory address space.
0	Space	RO 0x0	Memory Space Indicator

Table 809: PCI Express BAR2 (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042024, x4_port3x1_port0: 0x00082024, x4_port0x1_port0: 0x00040024, x4_port0x1_port1: 0x00044024, x4_port0x1_port2: 0x00048024, x4_port0x1_port3: 0x0004C024, x4_port1x1_port0: 0x00080024, x4_port1x1_port1: 0x00084024, x4_port1x1_port2: 0x00088024, x4_port1x1_port3: 0x0008C024**

Offset Formula: **0x00040024+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040024+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits[63:32].

Table 810: PCI Express Subsystem Device and Vendor ID Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x0004202C, x4_port3x1_port0: 0x0008202C, x4_port0x1_port0: 0x0004002C, x4_port0x1_port1: 0x0004402C, x4_port0x1_port2: 0x0004802C, x4_port0x1_port3: 0x0004C02C, x4_port1x1_port0: 0x0008002C, x4_port1x1_port1: 0x0008402C, x4_port1x1_port2: 0x0008802C, x4_port1x1_port3: 0x0008C02C**

Offset Formula: **0x0004002C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x0004002C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	SSDevID	RO 0x11AB	Subsystem Device ID Number The default is the Marvell Device ID.
15:0	SSVenID	RO 0x11AB	Subsystem Manufacturer ID Number The default is the Marvell Vendor ID.

Table 811: PCI Express Expansion ROM BAR Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042030, x4_port3x1_port0: 0x00082030, x4_port0x1_port0: 0x00040030, x4_port0x1_port1: 0x00044030, x4_port0x1_port2: 0x00048030, x4_port0x1_port3: 0x0004C030, x4_port1x1_port0: 0x00080030, x4_port1x1_port1: 0x00084030, x4_port1x1_port2: 0x00088030, x4_port1x1_port3: 0x0008C030**

Offset Formula: **0x00040030+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040030+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
If expansion ROM is not enabled via the ExpROMEn bit[0] the PCI Express Expansion ROM BAR Control Register, this register is reserved.			
31:22	RomBase	RW 0x0	Expansion ROM Base Address[31:22]
21:19	RomBase/Reserved	RW 0x0	These bits are defined according to field <ExpROMSz> in the PCI Express Expansion ROM BAR Control Register. When ROM Size is: 0 = 512 KByte Space: Bits[21:19] are used as Expansion ROM Base Address[21:19]. 1 = 1 MByte Space: Bits[21:20] are used as Expansion ROM Base Address[21:20], and bit[19] is reserved. 2 = 2 MByte Space: Bits[21] used as Expansion ROM Base Address[21], and bits[20:19] are reserved. 3 = 4 MByte Space: Bits[21:19] are reserved.
18:1	Reserved	RSVD 0x0	Reserved
0	RomEn	RW 0x0	Expansion ROM Enable **NOTE:** The device expansion ROM address space is enabled only if both this bit (RomEn) and <MemEn> in the PCI Express Command and Status Register are set. 0 = Disable 1 = Enable

Table 812: PCI Express Capability List Pointer Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042034, x4_port3x1_port0: 0x00082034, x4_port0x1_port0: 0x00040034, x4_port0x1_port1: 0x00044034, x4_port0x1_port2: 0x00048034, x4_port0x1_port3: 0x0004C034, x4_port1x1_port0: 0x00080034, x4_port1x1_port1: 0x00084034, x4_port1x1_port2: 0x00088034, x4_port1x1_port3: 0x0008C034**

Offset Formula: **0x00040034+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040034+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved

Table 812: PCI Express Capability List Pointer Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042034, x4_port3x1_port0: 0x00082034, x4_port0x1_port0: 0x00040034, x4_port0x1_port1: 0x00044034, x4_port0x1_port2: 0x00048034, x4_port0x1_port3: 0x0004C034, x4_port1x1_port0: 0x00080034, x4_port1x1_port1: 0x00084034, x4_port1x1_port2: 0x00088034, x4_port1x1_port3: 0x0008C034

Offset Formula: 0x00040034+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040034+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
7:0	CapPtr	RO 0x40	Capability List Pointer The current value in this field points to the PCI Power Management capability set in the PCI Express Power Management Capability Header Register at offset 0x40.

Table 813: PCI Express Interrupt Pin and Line Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004203C, x4_port3x1_port0: 0x0008203C, x4_port0x1_port0: 0x0004003C, x4_port0x1_port1: 0x0004403C, x4_port0x1_port2: 0x0004803C, x4_port0x1_port3: 0x0004C03C, x4_port1x1_port0: 0x0008003C, x4_port1x1_port1: 0x0008403C, x4_port1x1_port2: 0x0008803C, x4_port1x1_port3: 0x0008C03C

Offset Formula: 0x0004003C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004003C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Does not apply to PCI Express This field is hardwired to 0.
15:8	IntPin	RO 0x01	Indicates that the device is using INTA (Interrupt A) in the interrupt emulation messages.
7:0	IntLine	RW 0x00	Provides interrupt line routing information.

Table 814: PCI Express Power Management Capability Header Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042040, x4_port3x1_port0: 0x00082040, x4_port0x1_port0: 0x00040040, x4_port0x1_port1: 0x00044040, x4_port0x1_port2: 0x00048040, x4_port0x1_port3: 0x0004C040, x4_port1x1_port0: 0x00080040, x4_port1x1_port1: 0x00084040, x4_port1x1_port2: 0x00088040, x4_port1x1_port3: 0x0008C040

Offset Formula: 0x00040040+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040040+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:27	PMESup	RO 0x00	This field indicates whether the device supports PM Event generation. The device does not support the PMEn pin.
26	D2Sup	RO 0x1	This bit indicates whether the device supports D2 Power Management state. The device does not support D2 state.
25	D1Sup	RO 0x1	This bit indicates whether the device supports D1 Power Management state. The device does not support D1 state.
24:22	AuxCur	RO 0x0	Auxiliary Current Requirements
21	DSI	RO 0x0	Device Specific Initialization The device does not requires device specific initialization.
20:19	Reserved	RSVD 0x0	Does not apply to PCI Express This field must be hardwired to 0.
18:16	PMCVer	RO 0x3	PCI Power Management Capability Version
15:8	NextPtr	RO 0x50	Next Item Pointer Current value points to MSI capability.
7:0	CapID	RO 0x01	Capability ID Current value identifies the PCI Power Management capability.

Table 815: PCI Express Power Management Control and Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042044, x4_port3x1_port0: 0x00082044, x4_port0x1_port0: 0x00040044, x4_port0x1_port1: 0x00044044, x4_port0x1_port2: 0x00048044, x4_port0x1_port3: 0x0004C044, x4_port1x1_port0: 0x00080044, x4_port1x1_port1: 0x00084044, x4_port1x1_port2: 0x00088044, x4_port1x1_port3: 0x0008C044**

Offset Formula: **0x00040044+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040044+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:24	PMData	RO 0x0	State Data This field is used to report the state dependent data requested by the <PMDataSel> field of this register. The value of this field is scaled by the value reported by the <PMDataScale> field of this register.
23:16	Reserved	RSVD 0x0	Does not apply to PCI Express. This field must be hardwired to 0.
15	PMEStat	RW1C 0x0	PME Status **NOTE:** Sticky bit--not initialized by hot reset.
14:13	PMDataScale	RO 0x0	Data Scale Indicates the scaling factor to be used when interpreting the value of the <PMData> field of this register. The read value depends on the setting of the <PMDataSel> field of this register.
12:9	PMDataSel	RW 0x0	Data Select This 4-bit field is used to select which data is to be reported through the <PMDataScale> and <PMData> fields of this register.
8	PME_en	RW 0x0	PM_PME Message Generation Enable **NOTE:** Sticky bit--not initialized by hot reset. 0 = Disable 1 = Enable
7:4	Reserved	RSVD 0x0	Reserved

Table 815: PCI Express Power Management Control and Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042044, x4_port3x1_port0: 0x00082044, x4_port0x1_port0: 0x00040044, x4_port0x1_port1: 0x00044044, x4_port0x1_port2: 0x00048044, x4_port0x1_port3: 0x0004C044, x4_port1x1_port0: 0x00080044, x4_port1x1_port1: 0x00084044, x4_port1x1_port2: 0x00088044, x4_port1x1_port3: 0x0008C044**

Offset Formula: **0x00040044+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040044+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
3	No_Soft_Reset	RO 0x0	When set (1), this bit indicates that devices transitioning from D3hot to D0 because of PowerState commands do not perform an internal reset. Configuration Context is preserved. Upon transition from the D3hot to the D0 Initialized state, no additional operating system intervention is required to preserve Configuration Context beyond writing the PowerState bits. When clear (0), devices do perform an internal reset upon transitioning from D3hot to D0 via software control of the PowerState bits. Configuration Context is lost when performing the soft reset. Upon transition from the D3hot to the D0 state, full reinitialization sequence is needed to return the device to D0 Initialized. Regardless of this bit, devices that transition from D3hot to D0 by a system or bus segment reset will return to the device state D0 uninitialized with only PME context preserved if PME is supported and enabled.
2	Reserved	RSVD 0x0	Reserved
1:0	PMState	RW 0x0	Power State This field controls the Power Management state of the device. The device supports all Power Management states. 0 = D0 1 = D1 2 = D2 3 = D3

Table 816: PCI Express MSI Message Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042050, x4_port3x1_port0: 0x00082050, x4_port0x1_port0: 0x00040050, x4_port0x1_port1: 0x00044050, x4_port0x1_port2: 0x00048050, x4_port0x1_port3: 0x0004C050, x4_port1x1_port0: 0x00080050, x4_port1x1_port1: 0x00084050, x4_port1x1_port2: 0x00088050, x4_port1x1_port3: 0x0008C050**

Offset Formula: **0x00040050+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040050+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved

Table 816: PCI Express MSI Message Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042050, x4_port3x1_port0: 0x00082050, x4_port0x1_port0: 0x00040050, x4_port0x1_port1: 0x00044050, x4_port0x1_port2: 0x00048050, x4_port0x1_port3: 0x0004C050, x4_port1x1_port0: 0x00080050, x4_port1x1_port1: 0x00084050, x4_port1x1_port2: 0x00088050, x4_port1x1_port3: 0x0008C050

Offset Formula: 0x00040050+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040050+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
23	Addr64	RO 0x1	64-bit Addressing Capable This field indicates whether the device is capable of generating a 64-bit message address. 0 = NotCapable 1 = Capable
22:20	MultiEn	RW 0x0	Multiple Messages Enable The number of messages the system allocates to the device. This number must be smaller or equal to the value that is in the <MultiCap> field.
19:17	MultiCap	RO 0x0	Multiple Message Capable The device is capable of driving a single message.
16	MSIEn	RW 0x0	MSI Enable Controls the device interrupt signaling mechanism. 0 = Disable: Interrupts are signaled through the interrupt emulation messages. 1 = Enable: Interrupts are signaled through MSI mechanism.
15:8	NextPtr	RO 0x60	Next Item Pointer The current value of this field points to PCI Express capability.
7:0	CapID	RO 0x5	Capability ID The current value of this field identifies the PCI MSI capability.

Table 817: PCI Express MSI Message Address Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042054, x4_port3x1_port0: 0x00082054, x4_port0x1_port0: 0x00040054, x4_port0x1_port1: 0x00044054, x4_port0x1_port2: 0x00048054, x4_port0x1_port3: 0x0004C054, x4_port1x1_port0: 0x00080054, x4_port1x1_port1: 0x00084054, x4_port1x1_port2: 0x00088054, x4_port1x1_port3: 0x0008C054

Offset Formula: 0x00040054+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040054+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:2	MSIAddr	RW 0x0	Message Address This field corresponds to Address[31:2] of the MSI MWr TLP.
1:0	Reserved	RSVD 0x0	Reserved

Table 818: PCI Express MSI Message Address (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042058, x4_port3x1_port0: 0x00082058, x4_port0x1_port0: 0x00040058, x4_port0x1_port1: 0x00044058, x4_port0x1_port2: 0x00048058, x4_port0x1_port3: 0x0004C058, x4_port1x1_port0: 0x00080058, x4_port1x1_port1: 0x00084058, x4_port1x1_port2: 0x00088058, x4_port1x1_port3: 0x0008C058

Offset Formula: 0x00040058+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040058+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	MSIAddrh	RW 0x0	Message Upper Address This field corresponds to Address[63:32] of the MSI MWr TLP.

Table 819: PCI Express MSI Message Data Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004205C, x4_port3x1_port0: 0x0008205C, x4_port0x1_port0: 0x0004005C, x4_port0x1_port1: 0x0004405C, x4_port0x1_port2: 0x0004805C, x4_port0x1_port3: 0x0004C05C, x4_port1x1_port0: 0x0008005C, x4_port1x1_port1: 0x0008405C, x4_port1x1_port2: 0x0008805C, x4_port1x1_port3: 0x0008C05C

Offset Formula: 0x0004005C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004005C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved

Table 819: PCI Express MSI Message Data Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x0004205C, x4_port3x1_port0: 0x0008205C, x4_port0x1_port0: 0x0004005C, x4_port0x1_port1: 0x0004405C, x4_port0x1_port2: 0x0004805C, x4_port0x1_port3: 0x0004C05C, x4_port1x1_port0: 0x0008005C, x4_port1x1_port1: 0x0008405C, x4_port1x1_port2: 0x0008805C, x4_port1x1_port3: 0x0008C05C

Offset Formula: 0x0004005C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004005C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
15:0	MSIData	RW 0x0	Message Data **NOTE:**

Table 820: PCI Express Capability Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042060, x4_port3x1_port0: 0x00082060, x4_port0x1_port0: 0x00040060, x4_port0x1_port1: 0x00044060, x4_port0x1_port2: 0x00048060, x4_port0x1_port3: 0x0004C060, x4_port1x1_port0: 0x00080060, x4_port1x1_port1: 0x00084060, x4_port1x1_port2: 0x00088060, x4_port1x1_port3: 0x0008C060

Offset Formula: 0x00040060+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040060+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:25	IntMsgNum	RO 0x0	Interrupt Message Number
24	SlotImp	RO 0x0	Slot Implemented
23:20	DevType	RO 0x4	Device/Port Type 1 = EndPoint: Legacy PCI Express Endpoint 4 = RootComplex: Root Port of PCI Express Root Complex
19:16	CapVer	RO 0x2	Capability Version This field indicates the PCI Express Base spec 2.0 version of the PCI Express capability.
15:8	NextPtr	RO 0x0	Next Item Pointer The current value of this field points to the end of the capability list (NULL).

Table 820: PCI Express Capability Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042060, x4_port3x1_port0: 0x00082060, x4_port0x1_port0: 0x00040060, x4_port0x1_port1: 0x00044060, x4_port0x1_port2: 0x00048060, x4_port0x1_port3: 0x0004C060, x4_port1x1_port0: 0x00080060, x4_port1x1_port1: 0x00084060, x4_port1x1_port2: 0x00088060, x4_port1x1_port3: 0x0008C060

Offset Formula: 0x00040060+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040060+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
7:0	CapID	RO 0x10	Capability ID The current value of this field identifies the PCI PE capability.

Table 821: PCI Express Device Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042064, x4_port3x1_port0: 0x00082064, x4_port0x1_port0: 0x00040064, x4_port0x1_port1: 0x00044064, x4_port0x1_port2: 0x00048064, x4_port0x1_port3: 0x0004C064, x4_port1x1_port0: 0x00080064, x4_port1x1_port1: 0x00084064, x4_port1x1_port2: 0x00088064, x4_port1x1_port3: 0x0008C064

Offset Formula: 0x00040064+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040064+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28	FuncLevelRstCpb	RO 0x0	Function Level Reset Capability. A value of 1b indicates the Function supports the optional Function Level Reset mechanism.
27:26	CapSPLScl	RO 0x0	Captured Slot Power Limit Scale
25:18	CapSPLVal	RO 0x0	Captured Slot Power Limit Value
17:16	Reserved	RSVD 0x0	Reserved
15	RuleBaseErrorRep	RO 0x1	Role-Based Error Reporting => This bit, when set, indicates that the device implements the functionality originally defined in the Error Reporting ECN for PCI Express Base Specification, Revision 1.0a, and later incorporated into PCI Express Base Specification, Revision 1.1. This bit must be set by all devices conforming to the ECN, PCI Express Base Specification, Revision 1.1., or subsequent PCI Express Base Specification revisions.

Table 821: PCI Express Device Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042064, x4_port3x1_port0: 0x00082064, x4_port0x1_port0: 0x00040064, x4_port0x1_port1: 0x00044064, x4_port0x1_port2: 0x00048064, x4_port0x1_port3: 0x0004C064, x4_port1x1_port0: 0x00080064, x4_port1x1_port1: 0x00084064, x4_port1x1_port2: 0x00088064, x4_port1x1_port3: 0x0008C064

Offset Formula: 0x00040064+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040064+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
14	Reserved	RO 0x0	Reserved
13	Reserved	RO 0x0	Reserved
12	Reserved	RO 0x0	Reserved
11:9	EPL1AccLat	RO 0x0	Endpoint L1 Acceptable Latency This field indicates the amount of latency that can be absorbed by the device due to the transition from L1 to L0. The current value specifies less than 1 second. 000b Maximum of 1 µs 001b Maximum of 2 µs 010b Maximum of 4 µs 011b Maximum of 8 µs 100b Maximum of 16 µs 101b Maximum of 32 µs 110b Maximum of 64 µs 111b No limit
8:6	EPL0sAccLat	RO 0x2	Endpoint L0s Acceptable Latency This field indicates the amount of latency that can be absorbed by the device due to the transition from L0s to L0. 0 = Under64ns: Less than 64 ns. 1 = 64to128ns: Less than 128 ns. 2 = 128to256ns: Less than 256 ns. 3 = 256to512ns: Less than 512 ns. 4 = 512nsto1us: Less than 1 us. 5 = 1to2us: Less than 2 us. 6 = 2to4us: Less than 4 us. 7 = Above4us: No limit
5:3	Reserved	RO 0x0	Reserved These bits are hardwired to 0.
2:0	MaxPldSizeSup	RO 0x0	Maximum Payload Size Supported 128B MPS support. 0 = 128

Table 822: PCI Express Device Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042068, x4_port3x1_port0: 0x00082068, x4_port0x1_port0: 0x00040068, x4_port0x1_port1: 0x00044068, x4_port0x1_port2: 0x00048068, x4_port0x1_port3: 0x0004C068, x4_port1x1_port0: 0x00080068, x4_port1x1_port1: 0x00084068, x4_port1x1_port2: 0x00088068, x4_port1x1_port3: 0x0008C068

Offset Formula: 0x00040068+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040068+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:22	Reserved	RSVD 0x0	Reserved
21	TransPend	RO 0x0	Transactions Pending Indicates that the device has issued Non-Posted requests that have not been completed. 0 = Completed: All NP requests have been completed or terminated by the Completion Timeout Mechanism. 1 = Uncompleted: Not all NP requests have been completed or terminated.
20	Reserved	RSVD 0x0	Reserved
19	URDet	RW1C 0x0	Unsupported Request Detected Indicates that the device received an unsupported request. Write 1 to clear.
18	FErrDet	RW1C 0x0	Fatal Error Detected Indicates the status of the fatal errors detected by the device. Write 1 to clear.
17	NFErrDet	RW1C 0x0	Non-Fatal Error Detected Indicates the status of the non-fatal errors detected by the device. Write 1 to clear.
16	CorErrDet	RW1C 0x0	Correctable Error Detected Indicates the status of the correctable errors detected by the device. Write 1 to clear.
15	Reserved	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 822: PCI Express Device Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042068, x4_port3x1_port0: 0x00082068, x4_port0x1_port0: 0x00040068, x4_port0x1_port1: 0x00044068, x4_port0x1_port2: 0x00048068, x4_port0x1_port3: 0x0004C068, x4_port1x1_port0: 0x00080068, x4_port1x1_port1: 0x00084068, x4_port1x1_port2: 0x00088068, x4_port1x1_port3: 0x0008C068

Offset Formula: 0x00040068+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040068+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
14:12	MaxRdRqSz	RW 0x2	Maximum Read Request Size Limits the device maximum read request size as a requestor (master). Other = Reserved 0 = 128B 1 = 256B 2 = 512B 3 = 1 KByte 4 = 2 KByte 5 = 4 KByte
11	EnNS	RO 0x0	Enable No Snoop The device never sets the No Snoop attribute in transactions it initiates as a requester. Hardwired to 0x0. 0 = Disable 1 = Enable
10	Reserved	RSVD 0x0	Reserved
9:8	Reserved	RO 0x0	Reserved These bits are hardwired to 0.
7:5	MaxPldSz	RW 0x0	Maximum Payload Size The maximum payload size supported is 128B (refer to bit <MaxPldSizeSup> in the PCI Express Device Capabilities Register). Other values = Reserved 0 = 128B: 0
4	EnRO	RO 0x0	Enable Relaxed Ordering The device never sets the Relaxed Ordering attribute in transactions it initiates as a requester. Hardwired to 0x0. 0 = Disable 1 = Enable

Table 822: PCI Express Device Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042068, x4_port3x1_port0: 0x00082068, x4_port0x1_port0: 0x00040068, x4_port0x1_port1: 0x00044068, x4_port0x1_port2: 0x00048068, x4_port0x1_port3: 0x0004C068, x4_port1x1_port0: 0x00080068, x4_port1x1_port1: 0x00084068, x4_port1x1_port2: 0x00088068, x4_port1x1_port3: 0x0008C068**

Offset Formula: **0x00040068+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040068+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
3	URRepEn	RW 0x0	Unsupported Request (UR) Reporting Enable In Root Complex mode, reporting of errors is internal to status registers only. An external error message must not be generated, therefore always write 0x0. **NOTE:** UR related error messages are still enabled when URRepEn=0, if <SErrEn> bit in PCI Express Command and Status Register is set. 0 = Disabled: UR related error messages are masked. Status bit is not masked. 1 = Enabled: UR related error messages enabled.
2	FErrRepEn	RW 0x0	Fatal Error Reporting Enable In Root Complex mode, reporting of errors is internal to status registers only. An external error message must not be generated; therefore, always write 0x0. **NOTE:** ERR_FATAL error messages are still enabled when this field is 0, if the <SErrEn> bit in PCI Express Command and Status Register is set. 0 = Disable: ERR_FATAL error messages are masked. Status bit is still affected. 1 = Enable: ERR_FATAL error messages enabled.
1	NFErrRepEn	RW 0x0	Non-Fatal Error Reporting Enable In Root Complex mode, reporting of errors is internal to status registers only. An external error message must not be generated, therefore always write 0x0. **NOTE:** ERR_NONFATAL error messages are still enabled when this field is 0, if the <SErrEn> bit in PCI Express Command and Status Register is set. 0 = Disable: ERR_NONFATAL error messages are masked. Status bit is not masked. 1 = Enable: ERR_NONFATAL error messages enabled.

Table 822: PCI Express Device Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042068, x4_port3x1_port0: 0x00082068, x4_port0x1_port0: 0x00040068, x4_port0x1_port1: 0x00044068, x4_port0x1_port2: 0x00048068, x4_port0x1_port3: 0x0004C068, x4_port1x1_port0: 0x00080068, x4_port1x1_port1: 0x00084068, x4_port1x1_port2: 0x00088068, x4_port1x1_port3: 0x0008C068

Offset Formula: 0x00040068+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040068+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
0	CorErrRepEn	RW 0x0	Correctable Error Reporting Enable In Root Complex mode, reporting of errors is internal to status registers only. An external error message must not be generated; therefore, always write 0x0. **NOTE:** ERR_NONFATAL error messages are still enabled when this field is 0, if the <SErrEn> bit in the PCI Express Command and Status Register is set. 0 = Disable: ERR_COR error messages are masked. Status bit is not masked. 1 = Enable: ERR_COR error messages enabled.

Table 823: PCI Express Link Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004206C, x4_port3x1_port0: 0x0008206C, x4_port0x1_port0: 0x0004006C, x4_port0x1_port1: 0x0004406C, x4_port0x1_port2: 0x0004806C, x4_port0x1_port3: 0x0004C06C, x4_port1x1_port0: 0x0008006C, x4_port1x1_port1: 0x0008406C, x4_port1x1_port2: 0x0008806C, x4_port1x1_port3: 0x0008C06C

Offset Formula: 0x0004006C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004006C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:24	PortNum	RO 0x0	In Endpoint mode, indicates the PCI Express port number as was advertised by the Root Complex during the link training process.
23:19	Reserved	RSVD 0x0	Reserved

Table 823: PCI Express Link Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x0004206C, x4_port3x1_port0: 0x0008206C, x4_port0x1_port0: 0x0004006C, x4_port0x1_port1: 0x0004406C, x4_port0x1_port2: 0x0004806C, x4_port0x1_port3: 0x0004C06C, x4_port1x1_port0: 0x0008006C, x4_port1x1_port1: 0x0008406C, x4_port1x1_port2: 0x0008806C, x4_port1x1_port3: 0x0008C06C**

Offset Formula: **0x0004006C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x0004006C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
18	ClockPowerMng	RO 0x1	Clock Power Management A value of 1b in this bit indicates that the component tolerates the removal of any reference clock(s) via the clock request (CLKREQ#) mechanism when the link is in the L1 and L2/3 Ready link states. A value of 0b indicates the component does not have this capability and that reference clock(s) must not be removed in these link states. This capability is applicable only in form factors that support clock request (CLKREQ#) capability. For a multi-function device, each function indicates its capability independently. Power Management configuration software must only permit reference clock removal if all functions of the multifunction device indicate a 1b in this bit.
17:15	L1ExtLat	RO 0x7	L1 Exit Latency This field indicates the L1 exit latency for the given PCI Express Link. The value reported indicates the length of time this port requires to complete transition from L1 to L0. Defined encodings are: 000b = Less than 1µs 001b = 1 µs to less than 2 µs 010b = 2 µs to less than 4 µs 011b = 4 µs to less than 8 µs 100b = 8 µs to less than 16 µs 101b = 16 µs to less than 32 µs 110b = 32 µs-64 µs 111b = More than 64 µs **NOTE:** The exit latencies can be influenced by the PCI Express reference clock configuration, depending upon whether a component uses a common or separate reference clock.

Table 823: PCI Express Link Capabilities Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x0004206C, x4_port3x1_port0: 0x0008206C, x4_port0x1_port0: 0x0004006C, x4_port0x1_port1: 0x0004406C, x4_port0x1_port2: 0x0004806C, x4_port0x1_port3: 0x0004C06C, x4_port1x1_port0: 0x0008006C, x4_port1x1_port1: 0x0008406C, x4_port1x1_port2: 0x0008806C, x4_port1x1_port3: 0x0008C06C**

Offset Formula: **0x0004006C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x0004006C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
14:12	L0sExtLat	RO 0x2	L0s Exit Latency The time required by the device to transition its Rx lanes from L0s to L0. 0 = Under64ns: Less than 64 ns 1 = 64to128ns: 64 ns-128 ns 2 = 128to256ns: 128 ns-256 ns 3 = 256to512ns: 256 ns-512 ns 4 = 512nsto1µs: 512 ns-1 µs 5 = 1to2µs: 1 µs-2 µs 6 = 2to4µs: 2 us-4 us 7 = Reserved
11:10	AspmSup	RO 0x3	Active State Link PM Support The current value identifies L0s and L1 Supported. 1 = L0s_Entry_Supported 3 = L0s_and_L1_Supported
9:4	MaxLnkWdth	RO SAR	Maximum Link Width supported by the port. The legal values are 1 for X1 link and 4 for X4 link. The default value depends on the PCIe port number as follows: For ports 0.0 and 1.0 = SAR For ports x.1, x.2, x.3 = 0x1 For ports 2.0 and 3.0 = 0x4
3:0	MaxLinkSpd	RO 0x2	Maximum Link Speed The current value identifies the 2.5 Gbps link. 1 = 2.5G Support: 2.5 GTb/s Link speed supported. 2 = 5G Support: 5.0 GT/s and 2.5 GT/s Link speeds supported.

Table 824: PCI Express Link Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042070, x4_port3x1_port0: 0x00082070, x4_port0x1_port0: 0x00040070, x4_port0x1_port1: 0x00044070, x4_port0x1_port2: 0x00048070, x4_port0x1_port3: 0x0004C070, x4_port1x1_port0: 0x00080070, x4_port1x1_port1: 0x00084070, x4_port1x1_port2: 0x00088070, x4_port1x1_port3: 0x0008C070**

Offset Formula: **0x00040070+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040070+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31	PCIeLinkAutoBwStt	RW1C 0x0	Link Autonomous Bandwidth Status This bit is set by hardware to indicate that hardware has autonomously changed the link speed or width, without the port transitioning through DL_Down status, for reasons other than to attempt to correct unreliable link operation. This bit must be set if the Physical Layer reports a speed or width change was initiated by the Downstream component that was indicated as an autonomous change.
30	PCIeLinkBwMngStt	RW1C 0x0	Link Bandwidth Management Status – This bit is set by hardware to indicate that either of the following has occurred without the port transitioning through DL_Down status: - A Link retraining has completed following a write of 1b to the Retrain Link bit. **NOTE:** This bit is set following any write of 1b to the Retrain Link bit, including when the link is in the process of retraining for some other reason. - Hardware has changed the link speed or width to attempt to correct unreliable link operation, either through an LTSSM timeout or a higher level process. This bit must be set if the Physical Layer reports a speed or width change was initiated by the Downstream component that was not indicated as an autonomous change.
29	Reserved	RSVD 0x0	Reserved
28	SltClkCfg	RO 0x1	Slot Clock Configuration 0 = Independent: The device uses an independent clock, irrespective of the presence of a reference clock on the connector. 1 = Reference: The device uses the reference clock that the platform provides.
27	LnkTrn	RO 0x0	Link Training This bit is not applicable and reserved for Endpoints.
26	Reserved	RO 0x0	Reserved

Table 824: PCI Express Link Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042070, x4_port3x1_port0: 0x00082070, x4_port0x1_port0: 0x00040070, x4_port0x1_port1: 0x00044070, x4_port0x1_port2: 0x00048070, x4_port0x1_port3: 0x0004C070, x4_port1x1_port0: 0x00080070, x4_port1x1_port1: 0x00084070, x4_port1x1_port2: 0x00088070, x4_port1x1_port3: 0x0008C070**

Offset Formula: **0x00040070+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040070+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
25:20	NegLnkWdth	RO 0x0	Negotiated Link Width This field indicates the negotiated link width. This field is initialized by the hardware. The value in this field is undefined when the link is not up. 1 = X1 4 = X4
19:16	LnkSpd	RO 0x1	Link Speed This field indicates the negotiated link speed. 1 = 2.5G: 2.5 GT/s PCI Express Link 2 = 5G: 5.0 GT/s PCI Express Link
15:10	Reserved	RSVD 0x0	Reserved
9	HW_autonomous_width_disable	RW 0x0	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.
8	EnClkPwrMng	RW 0x0	Enable Clock Power Management Applicable only for form factors that support a Clock Request (CLKREQ#) mechanism, this enable functions as follows: 0b = Clock power management is disabled and device must hold CLKREQ# signal low. 1b = When this bit is set to 1, the device is permitted to use CLKREQ# signal to power manage link clock according to protocol defined in appropriate form factor specification. Default value of this field is 0b. Components that do not support Clock Power Management (as indicated by a 0b value in the Clock Power Management bit of the Link Capabilities register) must hardwire this bit to 0b.
7	ExtdSnc	RW 0x0	Extended Sync When set, this bit forces extended transmission of 4096 FTS ordered sets followed by a single skip ordered set in exit from L0s and extra (1024) TS1 at exit from L1. **NOTE:** This bit is used for test and measurement. **NOTE:**

Table 824: PCI Express Link Control Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042070, x4_port3x1_port0: 0x00082070, x4_port0x1_port0: 0x00040070, x4_port0x1_port1: 0x00044070, x4_port0x1_port2: 0x00048070, x4_port0x1_port3: 0x0004C070, x4_port1x1_port0: 0x00080070, x4_port1x1_port1: 0x00084070, x4_port1x1_port2: 0x00088070, x4_port1x1_port3: 0x0008C070

Offset Formula: 0x00040070+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040070+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
6	CmnClkCfg	RW 0x0	Common Clock Configuration When set by the software, this bit indicates that both devices on the link use a distributed common reference clock.
5	RetrnLnk	RW 0x0	Retrain Link This bit forces the device to initiate link retraining. Always returns 0 when read.
4	LnkDis	RW 0x0	Link Disable **NOTE:** If configured as an Endpoint, this field is reserved and has no effect. Activation procedure: 1. Set this bit to trigger link disable. 2. Poll <DLDown> de-assertion (PCI Express Status Register, bit 0) ensure the link is disabled. 3. Clear the bit to exit to detect and enable the link again.
3	RCB	RW 0x0	Read Completion Boundary **NOTE:** This bit has no effect on the device behavior. Completions are always returned with 128B read completion boundary. 0 = 64B: 64 Bytes 1 = 128B: 128 Bytes
2	Reserved	RSVD 0x0	Reserved
1:0	AspmCnt	RW 0x0	Active State Link PM Control Controls the level of active state PM supported on the link. 0 = Disable 1 = L0 Support: L0s entry supported. 2 = Reserved 3 = L0L1 Support: L0s and L1 entry supported.

Table 825: PCI Express Device Capabilities 2 Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042084, x4_port3x1_port0: 0x00082084, x4_port0x1_port0: 0x00040084, x4_port0x1_port1: 0x00044084, x4_port0x1_port2: 0x00048084, x4_port0x1_port3: 0x0004C084, x4_port1x1_port0: 0x00080084, x4_port1x1_port1: 0x00084084, x4_port1x1_port2: 0x00088084, x4_port1x1_port3: 0x0008C084

Offset Formula: 0x00040084+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040084+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:5	Reserved	RO 0x0	Reserved
4	CplTODisSup	RO 0x0	Completion Timeout Disable Supported A value of 1b indicates support for the Completion Timeout Disable mechanism.
3:0	CplTORangesSup	RO 0x0	Completion Timeout Ranges Supported This field indicates device function support for the optional Completion Timeout programmability mechanism. This mechanism allows system software to modify the Completion Timeout value. Four time value ranges are defined: Range A: 50 us to 10 ms Range B: 10 ms to 250 ms Range C: 250 ms to 4s Range D: 4 s to 64 s Bits are set according to the table below to show the timeout value ranges supported. 0000b = Completion Timeout programming not supported. The function must implement a timeout value in the range 50 us to 50 ms. 0001b = Range A 0010b = Range B 0011b = Ranges A and B 0110b = Ranges B and C 0111b = Ranges A, B, and C 1110b = Ranges B, C and D 1111b = Ranges A, B, C, and D All other values are reserved.

Table 826: PCI Express Device Control Status 2 Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042088, x4_port3x1_port0: 0x00082088, x4_port0x1_port0: 0x00040088, x4_port0x1_port1: 0x00044088, x4_port0x1_port2: 0x00048088, x4_port0x1_port3: 0x0004C088, x4_port1x1_port0: 0x00080088, x4_port1x1_port1: 0x00084088, x4_port1x1_port2: 0x00088088, x4_port1x1_port3: 0x0008C088

Offset Formula: 0x00040088+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040088+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4	CplTODis	RW 0x0	Completion Timeout Disable When set, this bit disables the Completion Timeout mechanism. Software is permitted to set or clear this bit at any time. When set, the Completion Timeout detection mechanism is disabled. If there are outstanding requests when the bit is cleared, it is permitted but not required for hardware to apply the completion timeout mechanism to the outstanding requests. If this is done, it is permitted to base the start time for each request on either the time this bit was cleared or the time each request was issued.
3:0	CplTOValue	RW 0x0	Completion Timeout Value Defined encoding: 0000b Default range: 12 ms to 48 ms Values available if Range B (10 ms to 250 ms) programmability range is supported: 0101b = 16 ms to 55 ms 0110b = 65 ms to 210 ms Values available if Range C (250 ms to 4 s) programmability range is supported: 1001b = 260 ms to 900 ms 1010b = 1s to 3.5s Values not defined above are reserved.

Table 827: PCI Express Link Control Status 2 Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042090, x4_port3x1_port0: 0x00082090, x4_port0x1_port0: 0x00040090, x4_port0x1_port1: 0x00044090, x4_port0x1_port2: 0x00048090, x4_port0x1_port3: 0x0004C090, x4_port1x1_port0: 0x00080090, x4_port1x1_port1: 0x00084090, x4_port1x1_port2: 0x00088090, x4_port1x1_port3: 0x0008C090

Offset Formula: 0x00040090+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040090+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:17	Reserved	RSVD 0x0	Reserved
16	CurrentDeemphasisLevel	RO 0x0	Current De-emphasis Level When the Link operates at 5 GT/s speed, this bit reflects the level of de-emphasis. Values: 1b -3.5 dB 0b -6 dB The value in this bit is undefined when the link operates at 2.5GT/s speed.
15:13	Reserved	RSVD 0x0	Reserved
12	ComplianceDeemphasis	RW 0x0	Compliance De-emphasis This bit sets the de-emphasis level in Polling. Compliance state if the entry occurred due to the Enter Compliance bit being 1b. Values: 1b -3.5 dB 0b -6 dB This bit is intended for debug, compliance testing purposes. System firmware and software are allowed to modify this bit only during debug or compliance testing.
11	ComplianceSOS	RW 0x0	Compliance SOS When set to 1b, the LTSSM is required to send SKP Ordered Sets periodically in between the (modified) compliance patterns.
10	EnterModifiedCompliance	RW 0x0	Enter Modified Compliance When this bit is set to 1b, the device transmits Modified Compliance Pattern if the LTSSM enters Polling.Compliance substate. This fiel is intended for debug, compliance testing purposes only. System firmware and software is allowed to modify this field only during debug or compliance testing. In all other cases, the system must ensure that this field is set to the default value.

Table 827: PCI Express Link Control Status 2 Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042090, x4_port3x1_port0: 0x00082090, x4_port0x1_port0: 0x00040090, x4_port0x1_port1: 0x00044090, x4_port0x1_port2: 0x00048090, x4_port0x1_port3: 0x0004C090, x4_port1x1_port0: 0x00080090, x4_port1x1_port1: 0x00084090, x4_port1x1_port2: 0x00088090, x4_port1x1_port3: 0x0008C090

Offset Formula: 0x00040090+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040090+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
9:7	TransmitMargin	RO 0x0	Transmit Margin This field controls the value of the nonde-emphasized voltage level at the Transmitter pins. This field is reset to 000b on entry to the LTSSM Polling.Configuration substate (see the relevant section in the Functional Specifications about how the transmitter voltage level is determined in various states). Values: a) 000b = Normal operating range b) 001b = 800-1200 mV for full swing and 400-700 mV for half-swing c) 010b - (n-1) = Values must be monotonic with a non-zero slope. The value of n must be greater than 3 and less than 7. At least two of these must be below the normal operating range of n: 200-400 mV for full-swing and 100-200 mV for half-swing d) (n) - 111b = Reserved **NOTE:** This register can be written to, although it is appears as RO. This register is intended for debug, compliance testing purposes only. System firmware and software are allowed to modify this register only during debug or compliance testing. In all other cases, the system must ensure that this register is set to the default value.
6	SelectDeemphasis	RW 0x0	Selectable De-emphasis When the link is operating at 5.0 GT/s speed, this bit selects the level of de-emphasis for an Upstream component. Values: 1b -3.5 dB 0b -6 dB When the Link is operating at 2.5 GT/s speed, the setting of this bit has no effect. Components that support only the 2.5 GT/s speed are permitted to hardwire this bit to 0b. This bit is not applicable and reserved for Endpoints, PCI Express to PCI/PCI-X bridges, and Upstream ports of switches. This field can be changed from the Mbus.
5	HardwareAutoSpeedDis	RO 0x0	Hardware Autonomous Speed Disable When set, this bit disables hardware from changing the link speed for device specific reasons other than attempting to correct unreliable link operation by reducing link speed. Initial transition to the highest supported common link speed is not blocked by this bit. Functions that do not implement the associated mechanism are permitted to hardwire this bit to 0b. PCIe does not implement this feature.

Table 827: PCI Express Link Control Status 2 Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042090, x4_port3x1_port0: 0x00082090, x4_port0x1_port0: 0x00040090, x4_port0x1_port1: 0x00044090, x4_port0x1_port2: 0x00048090, x4_port0x1_port3: 0x0004C090, x4_port1x1_port0: 0x00080090, x4_port1x1_port1: 0x00084090, x4_port1x1_port2: 0x00088090, x4_port1x1_port3: 0x0008C090**

Offset Formula: **0x00040090+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040090+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
4	EnterCompliance	RW 0x0	Enter Compliance Software is permitted to force a link to enter Compliance mode at the speed indicated in the Target Link Speed field by setting this bit to 1b in both components on a link and then initiating a hot reset on the link.
3:0	TargetLinkSpeed	RW 0x2	Target Link Speed - For Downstream Ports, this field sets an upper limit on link operational speed by restricting the values advertised by the Upstream component in its training sequences. Defined values: 0001b = 2.5 GT/s Target Link Speed 0010b = 5.0 GT/s Target Link Speed All other values are reserved. If a value is written to this field that does not correspond to a speed included in the Supported Link Speeds field, the result is undefined. The default value of this field is the highest link speed supported by the component (as reported in the Supported Link Speeds field of the Link Capabilities register) unless the corresponding platform/form factor requires a different default value. For both Upstream and Downstream ports, this field is used to set the target compliance mode speed when software is using the Enter Compliance bit to force a Link into compliance mode.

Table 828: PCI Express Advanced Error Report Header Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042100, x4_port3x1_port0: 0x00082100, x4_port0x1_port0: 0x00040100, x4_port0x1_port1: 0x00044100, x4_port0x1_port2: 0x00048100, x4_port0x1_port3: 0x0004C100, x4_port1x1_port0: 0x00080100, x4_port1x1_port1: 0x00084100, x4_port1x1_port2: 0x00088100, x4_port1x1_port3: 0x0008C100**

Offset Formula: **0x00040100+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040100+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:20	NextPtr	RO 0x0	Next Item Pointer This field indicates the last item in the extended capabilities linked list.

Table 828: PCI Express Advanced Error Report Header Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042100, x4_port3x1_port0: 0x00082100, x4_port0x1_port0: 0x00040100, x4_port0x1_port1: 0x00044100, x4_port0x1_port2: 0x00048100, x4_port0x1_port3: 0x0004C100, x4_port1x1_port0: 0x00080100, x4_port1x1_port1: 0x00084100, x4_port1x1_port2: 0x00088100, x4_port1x1_port3: 0x0008C100

Offset Formula: 0x00040100+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040100+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port,

Bit	Field	Type/InitVal	Description
19:16	CapVer	RO 0x1	Capability Version
15:0	PECapID	RO 0x1	Extended Capability ID The current value of this field identifies the Advanced Error Reporting capability.

Table 829: PCI Express Uncorrectable Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042104, x4_port3x1_port0: 0x00082104, x4_port0x1_port0: 0x00040104, x4_port0x1_port1: 0x00044104, x4_port0x1_port2: 0x00048104, x4_port0x1_port3: 0x0004C104, x4_port1x1_port0: 0x00080104, x4_port1x1_port1: 0x00084104, x4_port1x1_port2: 0x00088104, x4_port1x1_port3: 0x0008C104

Offset Formula: 0x00040104+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040104+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky, not initialized or modified by hot reset. All fields in this register (except for reserved fields) are SC. Write 1 to clear. A write of 0 has no effect.			
31:21	Reserved	RSVD 0x0	Reserved
20	URErr	RW1C 0x0	Unsupported Request Error Status **NOTE:** Sticky bit--not initialized by hot reset. **NOTE:**
19	Reserved	RSVD 0x0	Reserved
18	MalfTlpErr	RW1C 0x0	Malformed TLP Status **NOTE:** Sticky bit--not initialized by hot reset. **NOTE:**
17	Reserved	RW1C 0x0	Reserved **NOTE:** Sticky bit--not initialized by hot reset. **NOTE:**
16	UnexpCmpErr	RW1C 0x0	Unexpected Completion Status **NOTE:** Sticky bit--not initialized by hot reset. **NOTE:**

Table 829: PCI Express Uncorrectable Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042104, x4_port3x1_port0: 0x00082104, x4_port0x1_port0: 0x00040104, x4_port0x1_port1: 0x00044104, x4_port0x1_port2: 0x00048104, x4_port0x1_port3: 0x0004C104, x4_port1x1_port0: 0x00080104, x4_port1x1_port1: 0x00084104, x4_port1x1_port2: 0x00088104, x4_port1x1_port3: 0x0008C104**

Offset Formula: **0x00040104+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040104+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port,

Bit	Field	Type/InitVal	Description
15	CAErr	RW1C 0x0	Completer Abort Status **NOTE:** Sticky bit--not initialized by hot reset. **NOTE:**
14	CmpTOErr	RW1C 0x0	Completion Timeout Status **NOTE:** Sticky bit--not initialized by hot reset. **NOTE:**
13	Reserved	RSVD 0x0	Reserved **NOTE:** Sticky bit--not initialized by hot reset. **NOTE:**
12	RPsnTlpErr	RW1C 0x0	Poisoned TLP Status **NOTE:** Hot sticky bit--not initialized by hot reset.
11:5	Reserved	RSVD 0x0	Reserved
4	DLPrtErr	RW1C 0x0	Data Link Protocol Error Status **NOTE:** Hot sticky bit--not initialized by hot reset. **NOTE:**
3:0	Reserved	RSVD 0x0	Reserved

Table 830: PCI Express Uncorrectable Error Mask Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042108, x4_port3x1_port0: 0x00082108, x4_port0x1_port0: 0x00040108, x4_port0x1_port1: 0x00044108, x4_port0x1_port2: 0x00048108, x4_port0x1_port3: 0x0004C108, x4_port1x1_port0: 0x00080108, x4_port1x1_port1: 0x00084108, x4_port1x1_port2: 0x00088108, x4_port1x1_port3: 0x0008C108**

Offset Formula: **0x00040108+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040108+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky not initialized or modified by hot reset.			
31:21	Reserved	RSVD 0x0	Reserved

Table 830: PCI Express Uncorrectable Error Mask Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00042108, x4_port3x1_port0: 0x00082108, x4_port0x1_port0: 0x00040108, x4_port0x1_port1: 0x00044108, x4_port0x1_port2: 0x00048108, x4_port0x1_port3: 0x0004C108, x4_port1x1_port0: 0x00080108, x4_port1x1_port1: 0x00084108, x4_port1x1_port2: 0x00088108, x4_port1x1_port3: 0x0008C108

Offset Formula: 0x00040108+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040108+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port,

Bit	Field	Type/InitVal	Description
20	URErrMsk	RW 0x0	Unsupported Request Error Mask
19	Reserved	RSVD 0x0	Reserved.
18	MalfTlpErrMsk	RW 0x0	Malformed TLP Mask
17	Reserved	RW 0x0	Reserved
16	UnexpCmpErrMsk	RW 0x0	Unexpected Completion Mask
15	CAErrMsk	RW 0x0	Completer Abort Mask
14	CmpTOErrMsk	RW 0x0	Completion Timeout Mask
13	Reserved	RSVD 0x0	Reserved
12	RPsnTlpErrMsk	RW 0x0	Poisoned TLP Error Mask
11:5	Reserved	RSVD 0x0	Reserved
4	DLPrtErrMsk	RW 0x0	Data Link Protocol Error Mask When an error is indicated in the PCI Express Uncorrectable Error Status Register and the corresponding bit is set: - The header is not logged in the Header Log Register. - The First Error Pointer is not updated. - An error message is not generated. Status bit is set regardless of the mask setting. 0 = NotMasked 1 = Masked
3:0	Reserved	RSVD 0x0	Reserved

Table 831: PCI Express Uncorrectable Error Severity Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004210C, x4_port3x1_port0: 0x0008210C, x4_port0x1_port0: 0x0004010C, x4_port0x1_port1: 0x0004410C, x4_port0x1_port2: 0x0004810C, x4_port0x1_port3: 0x0004C10C, x4_port1x1_port0: 0x0008010C, x4_port1x1_port1: 0x0008410C, x4_port1x1_port2: 0x0008810C, x4_port1x1_port3: 0x0008C10C

Offset Formula: 0x0004010C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004010C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are hot sticky not initialized or modified by reset.			
31:21	Reserved	RSVD 0x0	Reserved
20	URErrSev	RW 0x0	Unsupported Request Error Severity
19	Reserved	RSVD 0x0	Reserved
18	MalfTlpErrSev	RW 0x1	Malformed TLP Severity
17	Reserved	RW 0x1	Reserved
16	UnexpCmpErrSev	RW 0x0	Unexpected Completion Severity
15	CAErrSev	RW 0x0	Completer Abort Severity
14	CmpTOErrSev	RW 0x0	Completion Timeout Severity
13	Reserved	RSVD 0x0	Reserved
12	RPsnTlpErrSev	RW 0x0	Poisoned TLP Error Severity
11:5	Reserved	RSVD 0x0	Reserved
4	DLPrtErrSev	RW 0x1	Data Link Protocol Error Severity Controls the severity indication of the Uncorrectable errors. Each bit controls the error type of the corresponding bit in the PCI Express Uncorrectable Error Status Register. 0 = NonFatal: Error type is Non-Fatal. 1 = Fatal: Error type is Fatal.
3:0	Reserved	RSVD 0x0	Reserved

Table 832: PCI Express Correctable Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00042110, x4_port3x1_port0: 0x00082110, x4_port0x1_port0: 0x00040110, x4_port0x1_port1: 0x00044110, x4_port0x1_port2: 0x00048110, x4_port0x1_port3: 0x0004C110, x4_port1x1_port0: 0x00080110, x4_port1x1_port1: 0x00084110, x4_port1x1_port2: 0x00088110, x4_port1x1_port3: 0x0008C110**

Offset Formula: **0x00040110+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040110+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky not initialized or modified by hot reset. All fields in this register (except for reserved fields) are RW1C write 1 to clear.			
31:14	Reserved	RSVD 0x0	Reserved
13	AdvNonFatalErr	RW1C 0x0	Advisory Non-Fatal Error Status
12	RplyTOErr	RW1C 0x0	Replay Timer Timeout Status
11:9	Reserved	RSVD 0x0	Reserved
8	RplyRllovrErr	RW1C 0x0	Replay Number Rollover Status
7	BadDllpErr	RW1C 0x0	Bad DLLP Status
6	BadTlpErr	RW1C 0x0	Bad TLP Status
5:1	Reserved	RSVD 0x0	Reserved
0	RcvErr	RW1C 0x0	Receiver Error Status **NOTE:** When set, this bit indicates that a Receiver error has occurred.

Table 833: PCI Express Correctable Error Mask Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042114, x4_port3x1_port0: 0x00082114, x4_port0x1_port0: 0x00040114, x4_port0x1_port1: 0x00044114, x4_port0x1_port2: 0x00048114, x4_port0x1_port3: 0x0004C114, x4_port1x1_port0: 0x00080114, x4_port1x1_port1: 0x00084114, x4_port1x1_port2: 0x00088114, x4_port1x1_port3: 0x0008C114

Offset Formula: 0x00040114+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040114+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky not initialized or modified by hot reset.			
31:14	Reserved	RSVD 0x0	Reserved
13	AdvNonFatalMsk	RW 0x1	Advisory Non-Fatal Error Mask This bit is set by default to enable compatibility with software that does not comprehend Role-Based Error Reporting.
12	RplyTOMsk	RW 0x0	Replay Timer Timeout Mask
11:9	Reserved	RSVD 0x0	Reserved
8	RplyRllovrMsk	RW 0x0	Replay Number Rollover Mask
7	BadDllpErrMsk	RW 0x0	Bad DLLP Mask
6	BadTlpMsk	RW 0x0	Bad TLP Mask
5:1	Reserved	RSVD 0x0	Reserved
0	RcvMsk	RW 0x0	Receiver Error Mask If set, an error message is not generated upon occurrence of a Receiver error. 0 = NotMasked 1 = Masked

Table 834: PCI Express Advanced Error Capability and Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042118, x4_port3x1_port0: 0x00082118, x4_port0x1_port0: 0x00040118, x4_port0x1_port1: 0x00044118, x4_port0x1_port2: 0x00048118, x4_port0x1_port3: 0x0004C118, x4_port1x1_port0: 0x00080118, x4_port1x1_port1: 0x00084118, x4_port1x1_port2: 0x00088118, x4_port1x1_port3: 0x0008C118

Offset Formula: 0x00040118+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040118+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved **NOTE:** Bits [5] and [7] are Read Only and are hardwired to 0. **NOTE:**
4:0	FrstErrPtr	RO 0x0	First Error Pointer This field reports the bit position of the first error reported in the PCI Express Uncorrectable Error Status Register. This field locks upon receipt of the first uncorrectable error that is not masked. It remains locked until software clears it by writing 1 to the corresponding status bit. Upon receipt of the next uncorrectable error that is not masked, the field locks again until cleared as described above. This lock and clear process continues to repeat itself. **NOTE:** The bits in this field are sticky bits--they are not initialized or modified by reset. 4 = DLP: Data Link Protocol Error 12 = TLP: Poisoned TLP Error 13 = FCP: Flow Control Protocol Error 14 = CT: Completion Timeout Error 15 = CAS: Completer Abort Status 16 = UCE: Unexpected Completion Error 17 = ROE: Receiver Overflow Error 18 = Mutant TLP: Malformed TLP Error 19 = Reserved 20 = URE: Unsupported Request Error Other = Reserved

Table 835: PCI Express Header Log First DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004211C, x4_port3x1_port0: 0x0008211C, x4_port0x1_port0: 0x0004011C, x4_port0x1_port1: 0x0004411C, x4_port0x1_port2: 0x0004811C, x4_port0x1_port3: 0x0004C11C, x4_port1x1_port0: 0x0008011C, x4_port1x1_port1: 0x0008411C, x4_port1x1_port2: 0x0008811C, x4_port1x1_port3: 0x0008C11C

Offset Formula: 0x0004011C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004011C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky, not initialized or modified by hot reset.			
31:0	Hdrlog1DW	RO 0x0	Header Log First DWORD Logs the header of the first error reported in the PCI Express Uncorrectable Error Status Register (PEUEST). This field locks upon receipt of the first uncorrectable error that is not masked. It remains locked until software clears it by writing 1 to the corresponding status bit. Upon receipt of the next uncorrectable error that is not masked, the field locks again until cleared as described above. This lock and clear process continues to repeat itself. **NOTE:**

Table 836: PCI Express Header Log Second DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042120, x4_port3x1_port0: 0x00082120, x4_port0x1_port0: 0x00040120, x4_port0x1_port1: 0x00044120, x4_port0x1_port2: 0x00048120, x4_port0x1_port3: 0x0004C120, x4_port1x1_port0: 0x00080120, x4_port1x1_port1: 0x00084120, x4_port1x1_port2: 0x00088120, x4_port1x1_port3: 0x0008C120

Offset Formula: 0x00040120+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040120+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky, not initialized or modified by hot reset.			
31:0	Hdrlog2DW	RO 0x0	Header Log Second DWORD **NOTE:**

Table 837: PCI Express Header Log Third DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042124, x4_port3x1_port0: 0x00082124, x4_port0x1_port0: 0x00040124, x4_port0x1_port1: 0x00044124, x4_port0x1_port2: 0x00048124, x4_port0x1_port3: 0x0004C124, x4_port1x1_port0: 0x00080124, x4_port1x1_port1: 0x00084124, x4_port1x1_port2: 0x00088124, x4_port1x1_port3: 0x0008C124

Offset Formula: 0x00040124+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040124+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky, not initialized or modified by hot reset.			
31:0	Hdrlog3DW	RO 0x0	Header Log Third DWORD **NOTE:**

Table 838: PCI Express Header Log Fourth DWORD Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042128, x4_port3x1_port0: 0x00082128, x4_port0x1_port0: 0x00040128, x4_port0x1_port1: 0x00044128, x4_port0x1_port2: 0x00048128, x4_port0x1_port3: 0x0004C128, x4_port1x1_port0: 0x00080128, x4_port1x1_port1: 0x00084128, x4_port1x1_port2: 0x00088128, x4_port1x1_port3: 0x0008C128

Offset Formula: 0x00040128+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040128+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All fields in this register are sticky, not initialized or modified by hot reset.			
31:0	Hdrlog4DW	RO 0x0	Header Log Fourth DWORD **NOTE:** All fields in this register are Hot sticky bit -- not initialized or modified by hot reset. **NOTE:**

Table 839: PCI Express Root Error Command Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004212C, x4_port3x1_port0: 0x0008212C, x4_port0x1_port0: 0x0004012C, x4_port0x1_port1: 0x0004412C, x4_port0x1_port2: 0x0004812C, x4_port0x1_port3: 0x0004C12C, x4_port1x1_port0: 0x0008012C, x4_port1x1_port1: 0x0008412C, x4_port1x1_port2: 0x0008812C, x4_port1x1_port3: 0x0008C12C

Offset Formula: 0x0004012C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004012C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:3	Reserved	RO 0x0	Reserved

Table 839: PCI Express Root Error Command Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x0004212C, x4_port3x1_port0: 0x0008212C, x4_port0x1_port0: 0x0004012C, x4_port0x1_port1: 0x0004412C, x4_port0x1_port2: 0x0004812C, x4_port0x1_port3: 0x0004C12C, x4_port1x1_port0: 0x0008012C, x4_port1x1_port1: 0x0008412C, x4_port1x1_port2: 0x0008812C, x4_port1x1_port3: 0x0008C12C

Offset Formula: 0x0004012C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004012C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
2	Fatal Error Reporting Enable	RW 0x0	Fatal Error Reporting Enable When set, this bit enables the generation of an interrupt when a fatal error is reported by any of the functions in the hierarchy associated with this Root Port.
1	Non-Fatal Error Reporting Enable	RW 0x0	Non-Fatal Error Reporting Enable When set, this bit enables the generation of an interrupt when a non-fatal error is reported by any of the functions in the hierarchy associated with this Root Port.
0	Correctable Error Reporting Enable	RW 0x0	Correctable Error Reporting Enable When set, this bit enables the generation of an interrupt when a correctable error is reported by any of the functions in the hierarchy associated with this Root Port.

Table 840: PCI Express Root Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042130, x4_port3x1_port0: 0x00082130, x4_port0x1_port0: 0x00040130, x4_port0x1_port1: 0x00044130, x4_port0x1_port2: 0x00048130, x4_port0x1_port3: 0x0004C130, x4_port1x1_port0: 0x00080130, x4_port1x1_port1: 0x00084130, x4_port1x1_port2: 0x00088130, x4_port1x1_port3: 0x0008C130

Offset Formula: 0x00040130+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040130+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:27	Advanced Error Interrupt Message Number	RO 0x0	Advanced Error Interrupt Message Number This register indicates which MSI/MSI-X vector is used for the interrupt message generated in association with any of the status bits of this capability.
26:7	Reserved	RO 0x0	RsvdZ
6	Fatal Error Messages Received	RW1C 0x0	Fatal Error Messages Received Set when one or more fatal uncorrectable error messages have been received.

Table 840: PCI Express Root Error Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00042130, x4_port3x1_port0: 0x00082130, x4_port0x1_port0: 0x00040130, x4_port0x1_port1: 0x00044130, x4_port0x1_port2: 0x00048130, x4_port0x1_port3: 0x0004C130, x4_port1x1_port0: 0x00080130, x4_port1x1_port1: 0x00084130, x4_port1x1_port2: 0x00088130, x4_port1x1_port3: 0x0008C130**

Offset Formula: **0x00040130+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00040130+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
5	Non-Fatal Error Messages Received	RW1C 0x0	Non-Fatal Error Messages Received Set when one or more non-fatal uncorrectable error messages have been received.
4	First Uncorrectable Fatal	RW1C 0x0	First Uncorrectable Fatal Set when the first uncorrectable error message received is for a fatal error.
3	Multiple ERR_FATAL/NONFATAL Received	RW1C 0x0	Multiple ERR_FATAL/NONFATAL Received Set when either a fatal or a non-fatal error is received and ERR_FATAL/NONFATAL Received is already set.
2	ERR_FATAL/NONFATAL Received	RW1C 0x0	ERR_FATAL/NONFATAL Received Set when either a fatal or a non-fatal error message is received and this bit is not already set.
1	Multiple ERR_COR Received	RW1C 0x0	Multiple ERR_COR Received Set when a correctable error message is received and ERR_COR Received is already set.
0	ERR_COR Received	RW1C 0x0	ERR_COR Received Set when a correctable error message is received and this bit is not already set.

Table 841: PCI Express Error Source Identification Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00042134, x4_port3x1_port0: 0x00082134, x4_port0x1_port0: 0x00040134, x4_port0x1_port1: 0x00044134, x4_port0x1_port2: 0x00048134, x4_port0x1_port3: 0x0004C134, x4_port1x1_port0: 0x00080134, x4_port1x1_port1: 0x00084134, x4_port1x1_port2: 0x00088134, x4_port1x1_port3: 0x0008C134

Offset Formula: 0x00040134+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00040134+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	ERR_FATAL/NONFATAL Source Identification	ROC 0x0	ERR_FATAL/NONFATAL Source Identification Loaded with the Requester ID indicated in the received ERR_FATAL or ERR_NONFATAL message when the ERR_FATAL/NONFATAL Received bit is not already set.
15:0	ERR_COR Source Identification	ROC 0x0	ERR_COR Source Identification Loaded with the Requester ID indicated in the received ERR_COR Message when the ERR_COR Received bit is not already set.

A.9.3 PCI Express Address Window Control

Table 842: PCI Express Window0 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043820, x4_port3x1_port0: 0x00083820, x4_port0x1_port0: 0x00041820, x4_port0x1_port1: 0x00045820, x4_port0x1_port2: 0x00049820, x4_port0x1_port3: 0x0004D820, x4_port1x1_port0: 0x00081820, x4_port1x1_port1: 0x00085820, x4_port1x1_port2: 0x00089820, x4_port1x1_port3: 0x0008D820

Offset Formula: 0x00041820+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041820+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KByte granularity. (A value of 0x0FFF specifies 256 MByte).
15:8	Attr	RW 0x0E	Target specific attributes depending on the target interface. See the "PCI Express Address Decoding" section.
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See the "PCI Express Address Decoding" section.

Table 842: PCI Express Window0 Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043820, x4_port3x1_port0: 0x00083820, x4_port0x1_port0: 0x00041820, x4_port0x1_port1: 0x00045820, x4_port0x1_port2: 0x00049820, x4_port0x1_port3: 0x0004D820, x4_port1x1_port0: 0x00081820, x4_port1x1_port1: 0x00085820, x4_port1x1_port2: 0x00089820, x4_port1x1_port3: 0x0008D820

Offset Formula: 0x00041820+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041820+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
3	Reserved	RSVD 0x0	Reserved
2	SlvWrSpltCnt	RW 0x0	Slave Write Split Control Controls the address boundary on which slave write transactions are split, when forwarded to the internal interface. Only relevant for transactions that hit the respective address window. 0 = SplitTo128: Write split on 128-Byte address boundary. 1 = SplitTo32: Write split on 32-Byte address boundary.
1	BarMap	RW 0x0	Mapping to BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
0	WinEn	RW 0x1	Window0 Enable 0 = Disable: Window is disabled. 1 = Enable: Window is enabled.

Table 843: PCI Express Window0 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043824, x4_port3x1_port0: 0x00083824, x4_port0x1_port0: 0x00041824, x4_port0x1_port1: 0x00045824, x4_port0x1_port2: 0x00049824, x4_port0x1_port3: 0x0004D824, x4_port1x1_port0: 0x00081824, x4_port1x1_port1: 0x00085824, x4_port1x1_port2: 0x00089824, x4_port1x1_port3: 0x0008D824

Offset Formula: 0x00041824+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041824+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0000	Base Address Used with the <Size> field to set the address window size and location. Corresponds to transaction address [31:16].
15:0	Reserved	RSVD 0x0	Reserved

Table 844: PCI Express Window0 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004382C, x4_port3x1_port0: 0x0008382C, x4_port0x1_port0: 0x0004182C, x4_port0x1_port1: 0x0004582C, x4_port0x1_port2: 0x0004982C, x4_port0x1_port3: 0x0004D82C, x4_port1x1_port0: 0x0008182C, x4_port1x1_port1: 0x0008582C, x4_port1x1_port2: 0x0008982C, x4_port1x1_port3: 0x0008D82C

Offset Formula: 0x0004182C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004182C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x0	Remap Address Used with the <Size> field to specifies address bits[31:0] to be driven to the target interface.
15:1	Reserved	RSVD 0x0	Reserved
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disable: Remap disabled. 1 = Enable: Remap enabled.

Table 845: PCI Express Window1 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043830, x4_port3x1_port0: 0x00083830, x4_port0x1_port0: 0x00041830, x4_port0x1_port1: 0x00045830, x4_port0x1_port2: 0x00049830, x4_port0x1_port3: 0x0004D830, x4_port1x1_port0: 0x00081830, x4_port1x1_port1: 0x00085830, x4_port1x1_port2: 0x00089830, x4_port1x1_port3: 0x0008D830

Offset Formula: 0x00041830+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041830+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size (A value of 0x0FFF specifies 256 MByte).
15:8	Attr	RW 0x0D	Target specific attributes depending on the target interface.
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window.
3	Reserved	RSVD 0x0	Reserved

Table 845: PCI Express Window1 Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043830, x4_port3x1_port0: 0x00083830, x4_port0x1_port0: 0x00041830, x4_port0x1_port1: 0x00045830, x4_port0x1_port2: 0x00049830, x4_port0x1_port3: 0x0004D830, x4_port1x1_port0: 0x00081830, x4_port1x1_port1: 0x00085830, x4_port1x1_port2: 0x00089830, x4_port1x1_port3: 0x0008D830**

Offset Formula: **0x00041830+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041830+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
2	SlvWrSpltCnt	RW 0x0	Slave Write Split Control Controls the address boundary on which slave write transactions are split, when forwarded to the internal interface. Only relevant for transactions that hit the respective address window. 0 = SplitTo128: Write split on 128-Byte address boundary. 1 = SplitTo32: Write split on 32-Byte address boundary.
1	BarMap	RW 0x0	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
0	WinEn	RW 0x1	Window Enable. 0 = Disable 1 = Enable

Table 846: PCI Express Window1 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043834, x4_port3x1_port0: 0x00083834, x4_port0x1_port0: 0x00041834, x4_port0x1_port1: 0x00045834, x4_port0x1_port2: 0x00049834, x4_port0x1_port3: 0x0004D834, x4_port1x1_port0: 0x00081834, x4_port1x1_port1: 0x00085834, x4_port1x1_port2: 0x00089834, x4_port1x1_port3: 0x0008D834**

Offset Formula: **0x00041834+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041834+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x1000	Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 847: PCI Express Window1 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x0004383C, x4_port3x1_port0: 0x0008383C, x4_port0x1_port0: 0x0004183C, x4_port0x1_port1: 0x0004583C, x4_port0x1_port2: 0x0004983C, x4_port0x1_port3: 0x0004D83C, x4_port1x1_port0: 0x0008183C, x4_port1x1_port1: 0x0008583C, x4_port1x1_port2: 0x0008983C, x4_port1x1_port3: 0x0008D83C**

Offset Formula: **0x0004183C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x0004183C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x0	Remap Address
15:1	Reserved	RSVD 0x0	Reserved
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disable: Remap disabled. 1 = Enable: Remap enabled.

Table 848: PCI Express Window2 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043840, x4_port3x1_port0: 0x00083840, x4_port0x1_port0: 0x00041840, x4_port0x1_port1: 0x00045840, x4_port0x1_port2: 0x00049840, x4_port0x1_port3: 0x0004D840, x4_port1x1_port0: 0x00081840, x4_port1x1_port1: 0x00085840, x4_port1x1_port2: 0x00089840, x4_port1x1_port3: 0x0008D840**

Offset Formula: **0x00041840+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041840+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size
15:8	Attr	RW 0x0B	Target specific attributes depending on the target interface.
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window.
3	Reserved	RSVD 0x0	Reserved
2	SlvWrSpltCnt	RW 0x0	Slave Write Split Control Controls the address boundary on which slave write transactions are split, when forwarded to the internal interface. Only relevant for transactions that hit the respective address window. 0 = SplitTo128: Write split on 128-Byte address boundary. 1 = SplitTo32: Write split on 32-Byte address boundary.

Table 848: PCI Express Window2 Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043840, x4_port3x1_port0: 0x00083840, x4_port0x1_port0: 0x00041840, x4_port0x1_port1: 0x00045840, x4_port0x1_port2: 0x00049840, x4_port0x1_port3: 0x0004D840, x4_port1x1_port0: 0x00081840, x4_port1x1_port1: 0x00085840, x4_port1x1_port2: 0x00089840, x4_port1x1_port3: 0x0008D840

Offset Formula: 0x00041840+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041840+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
1	BarMap	RW 0x0	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
0	WinEn	RW 0x1	Window Enable 0 = Disable 1 = Enable

Table 849: PCI Express Window2 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043844, x4_port3x1_port0: 0x00083844, x4_port0x1_port0: 0x00041844, x4_port0x1_port1: 0x00045844, x4_port0x1_port2: 0x00049844, x4_port0x1_port3: 0x0004D844, x4_port1x1_port0: 0x00081844, x4_port1x1_port1: 0x00085844, x4_port1x1_port2: 0x00089844, x4_port1x1_port3: 0x0008D844

Offset Formula: 0x00041844+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041844+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x2000	Base Address
15:0	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 850: PCI Express Window2 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004384C, x4_port3x1_port0: 0x0008384C, x4_port0x1_port0: 0x0004184C, x4_port0x1_port1: 0x0004584C, x4_port0x1_port2: 0x0004984C, x4_port0x1_port3: 0x0004D84C, x4_port1x1_port0: 0x0008184C, x4_port1x1_port1: 0x0008584C, x4_port1x1_port2: 0x0008984C, x4_port1x1_port3: 0x0008D84C

Offset Formula: 0x0004184C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004184C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x0	Remap Address
15:1	Reserved	RSVD 0x0	Reserved
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disable: Remap disabled. 1 = Enable: Remap enabled.

Table 851: PCI Express Window3 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043850, x4_port3x1_port0: 0x00083850, x4_port0x1_port0: 0x00041850, x4_port0x1_port1: 0x00045850, x4_port0x1_port2: 0x00049850, x4_port0x1_port3: 0x0004D850, x4_port1x1_port0: 0x00081850, x4_port1x1_port1: 0x00085850, x4_port1x1_port2: 0x00089850, x4_port1x1_port3: 0x0008D850

Offset Formula: 0x00041850+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041850+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size
15:8	Attr	RW 0x07	Target specific attributes depending on the target interface.
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window.
3	Reserved	RSVD 0x0	Reserved
2	SlvWrSpltCnt	RW 0x0	Slave Write Split Control Controls the address boundary on which slave write transactions are split, when forwarded to the internal interface. Only relevant for transactions that hit the respective address window. 0 = 128B: Write split on 128-Byte address boundary. 1 = 32B: Write split on 32-Byte address boundary.

Table 851: PCI Express Window3 Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043850, x4_port3x1_port0: 0x00083850, x4_port0x1_port0: 0x00041850, x4_port0x1_port1: 0x00045850, x4_port0x1_port2: 0x00049850, x4_port0x1_port3: 0x0004D850, x4_port1x1_port0: 0x00081850, x4_port1x1_port1: 0x00085850, x4_port1x1_port2: 0x00089850, x4_port1x1_port3: 0x0008D850

Offset Formula: 0x00041850+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041850+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
1	BarMap	RW 0x0	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
0	WinEn	RW 0x1	Window Enable. Window is disabled by default. 0 = Disable 1 = Enable

Table 852: PCI Express Window3 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043854, x4_port3x1_port0: 0x00083854, x4_port0x1_port0: 0x00041854, x4_port0x1_port1: 0x00045854, x4_port0x1_port2: 0x00049854, x4_port0x1_port3: 0x0004D854, x4_port1x1_port0: 0x00081854, x4_port1x1_port1: 0x00085854, x4_port1x1_port2: 0x00089854, x4_port1x1_port3: 0x0008D854

Offset Formula: 0x00041854+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041854+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x3000	Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 853: PCI Express Window3 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004385C, x4_port3x1_port0: 0x0008385C, x4_port0x1_port0: 0x0004185C, x4_port0x1_port1: 0x0004585C, x4_port0x1_port2: 0x0004985C, x4_port0x1_port3: 0x0004D85C, x4_port1x1_port0: 0x0008185C, x4_port1x1_port1: 0x0008585C, x4_port1x1_port2: 0x0008985C, x4_port1x1_port3: 0x0008D85C

Offset Formula: 0x0004185C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004185C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x0	Remap Address
15:1	Reserved	RSVD 0x0	Reserved
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disable: Remap disabled. 1 = Enable: Remap enabled.

Table 854: PCI Express Window4 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043860, x4_port3x1_port0: 0x00083860, x4_port0x1_port0: 0x00041860, x4_port0x1_port1: 0x00045860, x4_port0x1_port2: 0x00049860, x4_port0x1_port3: 0x0004D860, x4_port1x1_port0: 0x00081860, x4_port1x1_port1: 0x00085860, x4_port1x1_port2: 0x00089860, x4_port1x1_port3: 0x0008D860

Offset Formula: 0x00041860+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041860+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size (A value of 0x07FF specifies 128 MByte).
15:8	Attr	RW 0x3B	Target specific attributes depending on the target interface.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window.
3	Reserved	RSVD 0x0	Reserved
2	SlvWrSpltCnt	RW 0x0	Slave Write Split Control Controls the address boundary on which slave write transactions are split, when forwarded to the internal interface. Only relevant for transactions that hit the respective address window. 0 = 128B: Write split on 128-Byte address boundary. 1 = 32B: Write split on 32-Byte address boundary.

Table 854: PCI Express Window4 Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043860, x4_port3x1_port0: 0x00083860, x4_port0x1_port0: 0x00041860, x4_port0x1_port1: 0x00045860, x4_port0x1_port2: 0x00049860, x4_port0x1_port3: 0x0004D860, x4_port1x1_port0: 0x00081860, x4_port1x1_port1: 0x00085860, x4_port1x1_port2: 0x00089860, x4_port1x1_port3: 0x0008D860

Offset Formula: 0x00041860+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041860+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
1	BarMap	RW 0x1	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
0	WinEn	RW 0x1	Window Enable 0 = Disable 1 = Enable

Table 855: PCI Express Window4 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043864, x4_port3x1_port0: 0x00083864, x4_port0x1_port0: 0x00041864, x4_port0x1_port1: 0x00045864, x4_port0x1_port2: 0x00049864, x4_port0x1_port3: 0x0004D864, x4_port1x1_port0: 0x00081864, x4_port1x1_port1: 0x00085864, x4_port1x1_port2: 0x00089864, x4_port1x1_port3: 0x0008D864

Offset Formula: 0x00041864+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041864+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xF000	Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 856: PCI Express Window4 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004386C, x4_port3x1_port0: 0x0008386C, x4_port0x1_port0: 0x0004186C, x4_port0x1_port1: 0x0004586C, x4_port0x1_port2: 0x0004986C, x4_port0x1_port3: 0x0004D86C, x4_port1x1_port0: 0x0008186C, x4_port1x1_port1: 0x0008586C, x4_port1x1_port2: 0x0008986C, x4_port1x1_port3: 0x0008D86C

Offset Formula: 0x0004186C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004186C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x0	Remap Address

Table 856: PCI Express Window4 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x0004386C, x4_port3x1_port0: 0x0008386C, x4_port0x1_port0: 0x0004186C, x4_port0x1_port1: 0x0004586C, x4_port0x1_port2: 0x0004986C, x4_port0x1_port3: 0x0004D86C, x4_port1x1_port0: 0x0008186C, x4_port1x1_port1: 0x0008586C, x4_port1x1_port2: 0x0008986C, x4_port1x1_port3: 0x0008D86C

Offset Formula: 0x0004186C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004186C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
15:1	Reserved	RSVD 0x0	Reserved
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disable: Remap disabled. 1 = Enable: Remap enabled.

Table 857: PCI Express Window4 Remap (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043870, x4_port3x1_port0: 0x00083870, x4_port0x1_port0: 0x00041870, x4_port0x1_port1: 0x00045870, x4_port0x1_port2: 0x00049870, x4_port0x1_port3: 0x0004D870, x4_port1x1_port0: 0x00081870, x4_port1x1_port1: 0x00085870, x4_port1x1_port2: 0x00089870, x4_port1x1_port3: 0x0008D870

Offset Formula: 0x00041870+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041870+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap High Address Specifies address bits[63:32] to be driven to the target interface. Only relevant for target interfaces that support a more than 4-GByte address space. When using target interface that do not support a more than 4-GByte address space, this register must be cleared.

Table 858: PCI Express Window5 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043880, x4_port3x1_port0: 0x00083880, x4_port0x1_port0: 0x00041880, x4_port0x1_port1: 0x00045880, x4_port0x1_port2: 0x00049880, x4_port0x1_port3: 0x0004D880, x4_port1x1_port0: 0x00081880, x4_port1x1_port1: 0x00085880, x4_port1x1_port2: 0x00089880, x4_port1x1_port3: 0x0008D880

Offset Formula: 0x00041880+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041880+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size (A value of 0x07FF specifies 128 MByte).
15:8	Attr	RW 0x2F	Target specific attributes depending on the target interface.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window.
3	Reserved	RSVD 0x0	Reserved
2	SlvWrSpltCnt	RW 0x0	Slave Write Split Control Controls the address boundary on which slave write transactions are split, when forwarded to the internal interface. Only relevant for transactions that hit the respective address window. 0 = 128B: Write split on 128-Byte address boundary. 1 = 32B: Write split on 32-Byte address boundary.
1	BarMap	RW 0x1	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
0	WinEn	RW 0x1	Window Enable 0 = Disable 1 = Enable

Table 859: PCI Express Window5 Base Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043884, x4_port3x1_port0: 0x00083884, x4_port0x1_port0: 0x00041884, x4_port0x1_port1: 0x00045884, x4_port0x1_port2: 0x00049884, x4_port0x1_port3: 0x0004D884, x4_port1x1_port0: 0x00081884, x4_port1x1_port1: 0x00085884, x4_port1x1_port2: 0x00089884, x4_port1x1_port3: 0x0008D884**

Offset Formula: **0x00041884+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041884+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0xF800	Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 860: PCI Express Window5 Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x0004388C, x4_port3x1_port0: 0x0008388C, x4_port0x1_port0: 0x0004188C, x4_port0x1_port1: 0x0004588C, x4_port0x1_port2: 0x0004988C, x4_port0x1_port3: 0x0004D88C, x4_port1x1_port0: 0x0008188C, x4_port1x1_port1: 0x0008588C, x4_port1x1_port2: 0x0008988C, x4_port1x1_port3: 0x0008D88C**

Offset Formula: **0x0004188C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x0004188C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x0	Remap Address
15:1	Reserved	RSVD 0x0	Reserved
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disable: Remap disabled. 1 = Enable: Remap enabled.

Table 861: PCI Express Window5 Remap (High) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043890, x4_port3x1_port0: 0x00083890, x4_port0x1_port0: 0x00041890, x4_port0x1_port1: 0x00045890, x4_port0x1_port2: 0x00049890, x4_port0x1_port3: 0x0004D890, x4_port1x1_port0: 0x00081890, x4_port1x1_port1: 0x00085890, x4_port1x1_port2: 0x00089890, x4_port1x1_port3: 0x0008D890

Offset Formula: 0x00041890+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041890+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap High Address Specifies address bits[63:32] to be driven to the target interface. Only relevant for target interfaces that support a more than 4-GByte address space. When using target interface that do not support a more than 4-GByte address space, this register must be cleared.

Table 862: PCI Express Default Window Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x000438B0, x4_port3x1_port0: 0x000838B0, x4_port0x1_port0: 0x000418B0, x4_port0x1_port1: 0x000458B0, x4_port0x1_port2: 0x000498B0, x4_port0x1_port3: 0x0004D8B0, x4_port1x1_port0: 0x000818B0, x4_port1x1_port1: 0x000858B0, x4_port1x1_port2: 0x000898B0, x4_port1x1_port3: 0x0008D8B0

Offset Formula: 0x000418B0+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x000418B0+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:8	Attr	RW 0x0	Target specific attributes depending on the target interface.
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window.
3	Reserved	RSVD 0x0	Reserved
2	SlvWrSpltCnt	RW 0x0	Slave Write Split Control Controls the address boundary on which slave write transactions are split, when forwarded to the internal interface. Only relevant for transactions that hit the respective address window. 0 = SplitTo128: Write split on 128-Byte address boundary. 1 = SplitTo32: Write split on 32-Byte address boundary.
1:0	Reserved	RSVD 0x0	Reserved

Table 863: PCI Express Expansion ROM Window Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x000438C0, x4_port3x1_port0: 0x000838C0, x4_port0x1_port0: 0x000418C0, x4_port0x1_port1: 0x000458C0, x4_port0x1_port2: 0x000498C0, x4_port0x1_port3: 0x0004D8C0, x4_port1x1_port0: 0x000818C0, x4_port1x1_port1: 0x000858C0, x4_port1x1_port2: 0x000898C0, x4_port1x1_port3: 0x0008D8C0**

Offset Formula: **0x000418C0+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x000418C0+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:8	Attr	RW 0x2F	Target specific attributes depending on the target interface. Default corresponds to DEV_BOOTCSn.
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. Default represents the Device Bus controller.
3:0	Reserved	RSVD 0x0	Reserved

Table 864: PCI Express Expansion ROM Window Remap Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x000438C4, x4_port3x1_port0: 0x000838C4, x4_port0x1_port0: 0x000418C4, x4_port0x1_port1: 0x000458C4, x4_port0x1_port2: 0x000498C4, x4_port0x1_port3: 0x0004D8C4, x4_port1x1_port0: 0x000818C4, x4_port1x1_port1: 0x000858C4, x4_port1x1_port2: 0x000898C4, x4_port1x1_port3: 0x0008D8C4**

Offset Formula: **0x000418C4+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x000418C4+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Remap	RW 0x0	Remap Address
15:1	Reserved	RSVD 0x0	Reserved
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disable: Remap disabled. 1 = Enable: Remap enabled.

A.9.4 PCI Express Mbus Control

Table 865: PCI Express Mbus Adapter Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x000438D0, x4_port3x1_port0: 0x000838D0, x4_port0x1_port0: 0x000418D0, x4_port0x1_port1: 0x000458D0, x4_port0x1_port2: 0x000498D0, x4_port0x1_port3: 0x0004D8D0, x4_port1x1_port0: 0x000818D0, x4_port1x1_port1: 0x000858D0, x4_port1x1_port2: 0x000898D0, x4_port1x1_port3: 0x0008D8D0

Offset Formula: 0x000418D0+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x000418D0+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x0	Reserved
20	TxDPPropEn	RW 0x1	Tx Data Poisoning Error Propagation Enable 0 = Disable 1 = Enable
19	RxDPPropEn	RW 0x1	Rx Data Poisoning Error Propagation Enable 0 = Disable 1 = Enable
18:6	Reserved	RSVD 0x0	Reserved
5	Reserved	RW 0x0	Reserved
4	Reserved	RW 0x0	Reserved
3:2	Reserved	RW 0x0	Reserved
1	Reserved	RW 0x0	Reserved
0	Reserved	RW 0x0	Reserved

Copyright © 2014 Marvell

Table 866: PCI Express Mbus Arbiter Control Register (Low) (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x000438E0, x4_port3x1_port0: 0x000838E0, x4_port0x1_port0: 0x000418E0, x4_port0x1_port1: 0x000458E0, x4_port0x1_port2: 0x000498E0, x4_port0x1_port3: 0x0004D8E0, x4_port1x1_port0: 0x000818E0, x4_port1x1_port1: 0x000858E0, x4_port1x1_port2: 0x000898E0, x4_port1x1_port3: 0x0008D8E0**

Offset Formula: **0x000418E0+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x000418E0+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:28	Arb7	RW 0x7	Slice 7 of arbiter.
27:24	Arb6	RW 0x6	Slice 6 of arbiter.
23:20	Arb5	RW 0x5	Slice 5 of arbiter.
19:16	Arb4	RW 0x4	Slice 4 of arbiter.
15:12	Arb3	RW 0x3	Slice 3 of arbiter.
11:8	Arb2	RW 0x2	Slice 2 of arbiter.
7:4	Arb1	RW 0x1	Slice 1 of arbiter.
3:0	Arb0	RW 0x0	Slice 0 of arbiter.

Table 867: PCI Express Mbus Arbiter Control Register (High) (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x000438E4, x4_port3x1_port0: 0x000838E4, x4_port0x1_port0: 0x000418E4, x4_port0x1_port1: 0x000458E4, x4_port0x1_port2: 0x000498E4, x4_port0x1_port3: 0x0004D8E4, x4_port1x1_port0: 0x000818E4, x4_port1x1_port1: 0x000858E4, x4_port1x1_port2: 0x000898E4, x4_port1x1_port3: 0x0008D8E4**

Offset Formula: **0x000418E4+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x000418E4+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:28	Arb15	RW 0xf	Slice 15 of arbiter.
27:24	Arb14	RW 0xe	Slice 14 of arbiter.
23:20	Arb13	RW 0xd	Slice 13 of arbiter.

Table 867: PCI Express Mbus Arbiter Control Register (High) (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x000438E4, x4_port3x1_port0: 0x000838E4, x4_port0x1_port0: 0x000418E4, x4_port0x1_port1: 0x000458E4, x4_port0x1_port2: 0x000498E4, x4_port0x1_port3: 0x0004D8E4, x4_port1x1_port0: 0x000818E4, x4_port1x1_port1: 0x000858E4, x4_port1x1_port2: 0x000898E4, x4_port1x1_port3: 0x0008D8E4**

Offset Formula: **0x000418E4+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x000418E4+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port,

Bit	Field	Type/InitVal	Description
19:16	Arb12	RW 0xc	Slice 12 of arbiter.
15:12	Arb11	RW 0xb	Slice 11 of arbiter.
11:8	Arb10	RW 0xa	Slice 10 of arbiter.
7:4	Arb9	RW 0x9	Slice 9 of arbiter.
3:0	Arb8	RW 0x8	Slice 8 of arbiter.

Table 868: PCI Express Mbus Arbiter Timeout Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x000438E8, x4_port3x1_port0: 0x000838E8, x4_port0x1_port0: 0x000418E8, x4_port0x1_port1: 0x000458E8, x4_port0x1_port2: 0x000498E8, x4_port0x1_port3: 0x0004D8E8, x4_port1x1_port0: 0x000818E8, x4_port1x1_port1: 0x000858E8, x4_port1x1_port2: 0x000898E8, x4_port1x1_port3: 0x0008D8E8**

Offset Formula: **0x000418E8+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x000418E8+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:17	Reserved	RSVD 0x0	Reserved
16	TimeoutEn	RW 0x1	Mbus Arbiter Timeout Timer Enable 0 = Enable 1 = Disable
15:8	Reserved	RSVD 0x0	Reserved
7:0	Timeout	RW 0xFF	Mbus Arbiter Timeout Preset Value **NOTE:** Must keep the value of 0xFF as long as the Timeout counter is not enabled. **NOTE:**

A.9.5 PCI Express Control and Status

Table 869: PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043A00, x4_port3x1_port0: 0x00083A00, x4_port0x1_port0: 0x00041A00, x4_port0x1_port1: 0x00045A00, x4_port0x1_port2: 0x00049A00, x4_port0x1_port3: 0x0004DA00, x4_port1x1_port0: 0x00081A00, x4_port1x1_port1: 0x00085A00, x4_port1x1_port2: 0x00089A00, x4_port1x1_port3: 0x0008DA00**

Offset Formula: **0x00041A00+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A00+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31	CRS Enable	RW 0x0	When this bit is set, all configuration access returns with status=<CRS>. 0 = Disable 1 = Enable
30	Conf Training Disable	RW 0x1	MAC Training Disable
29	Reserved	RSVD 0x0	Reserved
28	ConfMove2DetectFromModCompl	RW 0x0	If asserted, causes link restart while in Polling.MOD.COMPL state.
27	ConfMstrDis Scrmb	RW 0x0	Master Disable Scrambling When set, a scrambling disable command is transmitted downstream, causing the scrambling to be disabled on both sides of the link. **NOTE:** This should be programmed before the start of link initialization.
26	ConfMstrLb	RW 0x0	Master Loopback When set, a loopback command is transmitted downstream, causing the downstream device to transmit back the traffic that it receives. This bit can be set only if bits[24] of this register and <LnkDis> in the PCI Express Link Control Status Register are cleared. **NOTE:** Use this bit only when working in Root Complex mode.
25	Compliance Receive	RW 0x0	Compliance Receive When set, a compliance command (Bit 4 of Symbol 5 in TS1 OS) is asserted in TS1 OS, causing the device connected to move to Polling.Compliance state.

Table 869: PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043A00, x4_port3x1_port0: 0x00083A00, x4_port0x1_port0: 0x00041A00, x4_port0x1_port1: 0x00045A00, x4_port0x1_port2: 0x00049A00, x4_port0x1_port3: 0x0004DA00, x4_port1x1_port0: 0x00081A00, x4_port1x1_port1: 0x00085A00, x4_port1x1_port2: 0x00089A00, x4_port1x1_port3: 0x0008DA00**

Offset Formula: **0x00041A00+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A00+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
24	ConfMstrHot Reset	RW 0x0	Master Hot-Reset When set, a hot-reset command is transmitted downstream, causing the downstream PCI Express hierarchy to enter a hot-reset cycle. This bit can be set only if LnkDis in the PCI Express Link Control Status Register and bit[26] of this register are cleared. Activation procedure: 1. Set this bit to trigger a hot-reset cycle. 2. Poll DLDown de-assertion (PCI Express Status Register, bit 0) to ensure hot-reset cycle is done. 3. Clear this bit to exit to detect and re-establish the PCI-Express link. This bit should be used only when working in Root Complex mode.
23:16	Reserved	RW 0x14	Reserved Must write 0x14. **NOTE:**
15:13	Reserved	RSVD 0x0	Reserved
12	Reserved	RW 0x0	Reserved
11	Reserved	RW 0x0	Reserved
10	conf_speed_change	RW 0x0	Auto Speed change. when set , link will issue link speed change to the max link speed possibel. 0x0 do not initiate a change. 0x1 do initiate a change.
9:8	Reserved_9_8	RW SAR	Reserved Please do not change default value at any time. The default value depends on the PCIe port number as follows: Ports 0.0, 1.0, 2.0, 3.0 = 0x3 Other ports = 0x2
7:3	Reserved	RSVD 0x5	Reserved

Table 869: PCI Express Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043A00, x4_port3x1_port0: 0x00083A00, x4_port0x1_port0: 0x00041A00, x4_port0x1_port1: 0x00045A00, x4_port0x1_port2: 0x00049A00, x4_port0x1_port3: 0x0004DA00, x4_port1x1_port0: 0x00081A00, x4_port1x1_port1: 0x00085A00, x4_port1x1_port2: 0x00089A00, x4_port1x1_port3: 0x0008DA00**

Offset Formula: **0x00041A00+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A00+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
2	CfgMapTo MemEn	RW 0x1	Configuration Header Mapping to Memory Space Enable When enabled, the configuration header registers can be accessed directly through the memory space. Access is enabled both from the internal bus and from the PCI Express port. 0 = Disable: Mapping disabled. 1 = Enable: Mapping enabled.
1	Reserved	RSVD 0x1	Reserved
0	Reserved	RW 0x0	Reserved

Table 870: PCI Express Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043A04, x4_port3x1_port0: 0x00083A04, x4_port0x1_port0: 0x00041A04, x4_port0x1_port1: 0x00045A04, x4_port0x1_port2: 0x00049A04, x4_port0x1_port3: 0x0004DA04, x4_port1x1_port0: 0x00081A04, x4_port1x1_port1: 0x00085A04, x4_port1x1_port2: 0x00089A04, x4_port1x1_port3: 0x0008DA04**

Offset Formula: **0x00041A04+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A04+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27	PCIeSlvDis Scrmb	RO 0x0	Slave Disable Scrambling Indication This field sets when the opposite device on the PCI Express port acts as a disable scrambling master, and when a scrambling disabled indication is received.
26	PCIeSlvLb	RO 0x0	Slave Loopback Indication This field sets when the opposite device on the PCI Express port acts as a loopback master, and when a loopback indication is received.
25	PCIeSlvDisLink	RO 0x0	Slave Disable Link Indication This field sets when the opposite device on the PCI Express port acts as a disable link master, and when a link disabled indication is received.

Table 870: PCI Express Status Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043A04, x4_port3x1_port0: 0x00083A04, x4_port0x1_port0: 0x00041A04, x4_port0x1_port1: 0x00045A04, x4_port0x1_port2: 0x00049A04, x4_port0x1_port3: 0x0004DA04, x4_port1x1_port0: 0x00081A04, x4_port1x1_port1: 0x00085A04, x4_port1x1_port2: 0x00089A04, x4_port1x1_port3: 0x0008DA04

Offset Formula: 0x00041A04+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A04+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
24	PCIeSlvHot Reset	RO 0x0	Slave Hot Reset Indication This field sets when the opposite device on the PCI Express port acts as a hot-reset master, and a when hot-reset indication is received.
23:21	Reserved	RSVD 0x0	Reserved
20:16	PCIeDevNum	RW 0x0	Device Number Indication This field is used in the RequesterID field of the transmitted TLPs. In Endpoint mode, the field updates whenever a CfgWr access is received.
15:8	PCIeBusNum	RW 0x0	Bus Number Indication This field is used in the RequesterID field of the transmitted TLPs. In Endpoint mode, the field updates whenever a CfgWr access is received.
7:5	Reserved	RSVD 0x0	Reserved
4:1	Reserved	RSVD 0x0	Reserved
0	DLDown	RO 0x0	DL_Down Indication 0 = Active: DL is up. 1 = Inactive: DL is down.

Table 871: PCI Express Root Complex Set Slot Power Limit (SSPL) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A0C, x4_port3x1_port0: 0x00083A0C, x4_port0x1_port0: 0x00041A0C, x4_port0x1_port1: 0x00045A0C, x4_port0x1_port2: 0x00049A0C, x4_port0x1_port3: 0x0004DA0C, x4_port1x1_port0: 0x00081A0C, x4_port1x1_port1: 0x00085A0C, x4_port1x1_port2: 0x00089A0C, x4_port1x1_port3: 0x0008DA0C

Offset Formula: 0x00041A0C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A0C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:17	Reserved	RO 0x0	Reserved

Table 871: PCI Express Root Complex Set Slot Power Limit (SSPL) Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043A0C, x4_port3x1_port0: 0x00083A0C, x4_port0x1_port0: 0x00041A0C, x4_port0x1_port1: 0x00045A0C, x4_port0x1_port2: 0x00049A0C, x4_port0x1_port3: 0x0004DA0C, x4_port1x1_port0: 0x00081A0C, x4_port1x1_port1: 0x00085A0C, x4_port1x1_port2: 0x00089A0C, x4_port1x1_port3: 0x0008DA0C

Offset Formula: 0x00041A0C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A0C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
16	SsplMsgEnable	RW 0x0	SSPL Message Enable 0 = Disable: No SSPL message is send. 1 = Enable: SSPL message is send when the PCI Express is set to Root Complex and: 1) Dl-Down -> Dl-Up or 2) There is a Write to the <SlotPowerLimitValue> or <SlotPowerLimitScale> fields.
15:10	Reserved	RO 0x0	Reserved
9:8	SlotPowerLimitScale	RW 0x0	Slot Power Limit Scale Specifies the scale used for the Slot Power Limit Value (in the PCI Express Interface section see the sub-section "Messages"). Range of Values: 00b = 1.0x 01b = 0.1x 10b = 0.01x 11b = 0.001x Writes to this register when the PCI Express interface is set to Root Complex also cause the port to send the Set Slot Power Limit message.
7:0	SlotPowerLimitValue	RW 0x0	Slot Power Limit Value In combination with the <SlotPowerLimitScale> value, specifies the upper limit on power supplied by the slot. The power limit (in watts) calculated by multiplying the value in this field by the value in the <SlotPowerLimitScale> field. Writes to this register when the PCI Express interface is set to Root Complex also cause the port to send the Set Slot Power Limit message.

Table 872: PCI Express Completion Timeout Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A10, x4_port3x1_port0: 0x00083A10, x4_port0x1_port0: 0x00041A10, x4_port0x1_port1: 0x00045A10, x4_port0x1_port2: 0x00049A10, x4_port0x1_port3: 0x0004DA10, x4_port1x1_port0: 0x00081A10, x4_port1x1_port1: 0x00085A10, x4_port1x1_port2: 0x00089A10, x4_port1x1_port3: 0x0008DA10

Offset Formula: 0x00041A10+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A10+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:17	Reserved	RSVD 0x0	Reserved
16	CPL To Range Offset	RW 0x0	Completion Range Offset This field is used to select a value from the range indicated in the <CplTOValue> field in the PCI Express Control Status 2 register. For example, the <CplTOValue> field provides a default range: 50 us to 50 ms and this register indicates whether the value is close to the minimum or maximum value in this range. 0 = Max Value: Maximum value in the selected range. 1 = Min Value: Minimum value in the selected range.
15:0	Reserved	RW 0x2710	Reserved

Table 873: PCI Express Root Complex Power Management Event (PME) Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A14, x4_port3x1_port0: 0x00083A14, x4_port0x1_port0: 0x00041A14, x4_port0x1_port1: 0x00045A14, x4_port0x1_port2: 0x00049A14, x4_port0x1_port3: 0x0004DA14, x4_port1x1_port0: 0x00081A14, x4_port1x1_port1: 0x00085A14, x4_port1x1_port2: 0x00089A14, x4_port1x1_port3: 0x0008DA14

Offset Formula: 0x00041A14+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A14+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:18	Reserved	RSVD 0x0	Reserved
17	PME Pending	RO 0x0	PME Pending This read-only bit indicates that another PME is pending when the PME Status bit is set. When the PME Status bit is cleared by software, the PME is delivered by hardware by setting the PME Status bit again and updating the Requestor ID appropriately. The PME pending bit is cleared by hardware if no more PMEs are pending.

Table 873: PCI Express Root Complex Power Management Event (PME) Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043A14, x4_port3x1_port0: 0x00083A14, x4_port0x1_port0: 0x00041A14, x4_port0x1_port1: 0x00045A14, x4_port0x1_port2: 0x00049A14, x4_port0x1_port3: 0x0004DA14, x4_port1x1_port0: 0x00081A14, x4_port1x1_port1: 0x00085A14, x4_port1x1_port2: 0x00089A14, x4_port1x1_port3: 0x0008DA14**

Offset Formula: **0x00041A14+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A14+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
16	PME Status	RO 0x0	PME Status This bit indicates that PME was asserted by the requestor ID indicated in the PME Requestor ID field. This bit can be cleared by a write of 0x1 to the <RcvPmPme> field in the PCI Express Interrupt Cause register (0x1900[28]).
15:0	PME Requester ID	RO 0x0	PME Requestor ID This field indicates the PCI requestor ID of the last locked PME requestor. When PME_Status=0x0, the value in the PME_Requester_ID is regard as reserved.

Table 874: PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043A18, x4_port3x1_port0: 0x00083A18, x4_port0x1_port0: 0x00041A18, x4_port0x1_port1: 0x00045A18, x4_port0x1_port2: 0x00049A18, x4_port0x1_port3: 0x0004DA18, x4_port1x1_port0: 0x00081A18, x4_port1x1_port1: 0x00085A18, x4_port1x1_port2: 0x00089A18, x4_port1x1_port3: 0x0008DA18**

Offset Formula: **0x00041A18+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A18+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:17	Reserved	RW 0x0	Reserved
16	Ref_clk_off_en	RW 0x0	The system enables the hardware to kill the REFCLK and Core Clock. This input is also a condition for gating the PCI Express unit.
15:6	Reserved	RSVD 0x0	Reserved
5	Send Turn Off Msg	RW 0x0	Host Request PCI Express unit to send Turn Off message.
4	Send Turn Off Ack Msg	RW 0x0	Endpoint Ready for Turn Off and ready to send Turn Off ack message and start sending DLLP Enter L2/3.

Table 874: PCI Express Power Management Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043A18, x4_port3x1_port0: 0x00083A18, x4_port0x1_port0: 0x00041A18, x4_port0x1_port1: 0x00045A18, x4_port0x1_port2: 0x00049A18, x4_port0x1_port3: 0x0004DA18, x4_port1x1_port0: 0x00081A18, x4_port1x1_port1: 0x00085A18, x4_port1x1_port2: 0x00089A18, x4_port1x1_port3: 0x0008DA18**

Offset Formula: **0x00041A18+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A18+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
3:2	Reserved	RSVD 0x0	Reserved
1	L1 ASPM Ack	RW 0x1	Only for Root Complex. The system allows acknowledgment of an L1 ASPM request by an Endpoint. 0 = SendAspmNackTlp: Stop L1-ASPM flow. 1 = SendAspmAckDllp: Continue L1-ASPM flow.
0	L1 ASPM En	RW 0x0	Only for Endpoint. Enables the hardware to initialize the L1 ASPM process. 0 = IDLE: No l1-Aspm flow (as EndPoint) 1 = StartL1AspmFlow

Table 875: PCI Express Flow Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043A20, x4_port3x1_port0: 0x00083A20, x4_port0x1_port0: 0x00041A20, x4_port0x1_port1: 0x00045A20, x4_port0x1_port2: 0x00049A20, x4_port0x1_port3: 0x0004DA20, x4_port1x1_port0: 0x00081A20, x4_port1x1_port1: 0x00085A20, x4_port1x1_port2: 0x00089A20, x4_port1x1_port3: 0x0008DA20**

Offset Formula: **0x00041A20+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A20+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31	Disable_UpdateFC_TO	RW 0x0	Disable Update Flow Control due to 30 usec timeout. 0 = EnableUpdateFC: There will be at least one Update Flow Control each 30 usec. 1 = DisableUpdateFC: There will be no Update Flow Control due to 30 usec timer expiration.
30:24	Reserved	RW 0x0	Reserved
23:16	ConfChInitFc	RW 0x0	Completion Headers Flow Control Credit Initial Value Infinite.

Copyright © 2014 Marvell

Table 875: PCI Express Flow Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043A20, x4_port3x1_port0: 0x00083A20, x4_port0x1_port0: 0x00041A20, x4_port0x1_port1: 0x00045A20, x4_port0x1_port2: 0x00049A20, x4_port0x1_port3: 0x0004DA20, x4_port1x1_port0: 0x00081A20, x4_port1x1_port1: 0x00085A20, x4_port1x1_port2: 0x00089A20, x4_port1x1_port3: 0x0008DA20**

Offset Formula: **0x00041A20+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A20+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
15:8	ConfNphInitFc	RW SAR	Non-Posted Headers Flow Control Credit Initial Value The default value depends on the PCIe port number as follows: Ports 0.0, 1.0, 2.0, 3.0 = 0xb Other ports = 0x8
7:0	ConfPhInitFc	RW SAR	Posted Headers Flow Control Credit Initial Value The default value depends on the PCIe port number as follows: Ports 0.0, 1.0, 2.0, 3.0 = 0x5 Other ports = 0x4

Table 876: PCI Express Replay Timeout (TO) Timer Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043A24, x4_port3x1_port0: 0x00083A24, x4_port0x1_port0: 0x00041A24, x4_port0x1_port1: 0x00045A24, x4_port0x1_port2: 0x00049A24, x4_port0x1_port3: 0x0004DA24, x4_port1x1_port0: 0x00081A24, x4_port1x1_port1: 0x00085A24, x4_port1x1_port2: 0x00089A24, x4_port1x1_port3: 0x0008DA24**

Offset Formula: **0x00041A24+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A24+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x0	Reserved
9	Disable Replay TO	RW 0x0	Disable Replay TO 1 = No Replay TO: There will be no Replay TO.
8	Use Change Replay TO	RW 0x0	Use Change Replay TO 0 = Fixed: The design value is taken from the fixed initial value. 1 = Bit[7:0] Value: The design value is taken from the <Change Replay TO> bits[7:0].
7:0	Change Replay TO	RW 0x0	Change Replay TO value used when <Use Change Replay TO> is asserted.

Table 877: PCI Express Ack Latency Timer Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A28, x4_port3x1_port0: 0x00083A28, x4_port0x1_port0: 0x00041A28, x4_port0x1_port1: 0x00045A28, x4_port0x1_port2: 0x00049A28, x4_port0x1_port3: 0x0004DA28, x4_port1x1_port0: 0x00081A28, x4_port1x1_port1: 0x00085A28, x4_port1x1_port2: 0x00089A28, x4_port1x1_port3: 0x0008DA28

**Offset Formula: 0x00041A28+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A28+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port**

Bit	Field	Type/InitVal	Description
31:9	Reserved	RSVD 0x0	Reserved
8	Use Change Ack TO	RW 0x0	Use Change Ack TO 0 => The design value is taken from fixed init-val. 1 => The design value is taken from change_ack_to[7:0]. 0 = Fixed: The design value is taken from fixed initial value. 1 = Bit[7:0] Value: The design value is taken from the <Use Change Ack TO> field bits[7:0].
7:0	Change Ack TO	RW 0x0	The value used for ACT_TO when the <Use Change Ack TO> field is asserted.

Table 878: PCI Express Secondary Bus Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A2C, x4_port3x1_port0: 0x00083A2C, x4_port0x1_port0: 0x00041A2C, x4_port0x1_port1: 0x00045A2C, x4_port0x1_port2: 0x00049A2C, x4_port0x1_port3: 0x0004DA2C, x4_port1x1_port0: 0x00081A2C, x4_port1x1_port1: 0x00085A2C, x4_port1x1_port2: 0x00089A2C, x4_port1x1_port3: 0x0008DA2C

**Offset Formula: 0x00041A2C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A2C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port**

Bit	Field	Type/InitVal	Description
31:9	Reserved	RO 0x0	
8	Secondary Bus Number Enable	RW 0x0	When 1, the CFG TLP TYPE is defined using Secondary Bus number. Otherwise the Primary bus number is used.
7:0	Secondary Bus Number	RW 0x0	The Secondary Bus number field is used to decide on type on CFG request. If the Destination Bus number equals the Secondary Bus number then type 0; otherwise type 1. This field is used if the <Secondary Bus Number Enable> field is 1'b1; otherwise the Primary Bus is used for the type calculation.

Copyright © 2014 Marvell

Table 879: PCI Express Dynamic Width Management Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043A30, x4_port3x1_port0: 0x00083A30, x4_port0x1_port0: 0x00041A30, x4_port0x1_port1: 0x00045A30, x4_port0x1_port2: 0x00049A30, x4_port0x1_port3: 0x0004DA30, x4_port1x1_port0: 0x00081A30, x4_port1x1_port1: 0x00085A30, x4_port1x1_port2: 0x00089A30, x4_port1x1_port3: 0x0008DA30**

Offset Formula: **0x00041A30+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041A30+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4	Unused Lanes Turnoff En	RW 0x0	This bit controls the state of unused lanes during the Dynamic Width process. If asserted, unused lanes are completely turned off, otherwise they remain in TX Electrical Idle state. **NOTE:** Currently only Electrical Idle state is supported. Keep this bit de-asserted at all times. 0 = Disable: Lane turn off is disabled. 1 = Enable: Lane turn off is enabled.
3	Link Bandwidth Notific En	RW 0x0	Enable the Link Bandwidth Notification Capability 0 = Disable 1 = Enable
2	Upconfig Capability	RW 0x0	This field is used in TS ordered sets to indicate to other side of the link that Dynamic Width Capability us supported. 0 = Disable: Dynamic width not supported. 1 = Enable: Dynamic width supported.
1	Change Width Autonomous	RW 0x0	Trigger the autonomous width change. When set, the device tries to negotiate the width configured in the <Target Link Width> field. 0 = Disable: Disable width change. 1 = Enable: Start width change
0	Target Link Width	RW 0x00	This field is used to indicate the target width when the width change is initiated by the current device for autonomous reasons. When the <Change Width Autonomous> field is de-asserted, this field is ignored. Valid values: 0: x1 1: x4 Other values are illegal and can cause unpredicted results. 0 = Target width x1 1 = Target width x4

Table 880: PCI Express PHY Indirect Access Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043B00, x4_port3x1_port0: 0x00083B00, x4_port0x1_port0: 0x00041B00, x4_port0x1_port1: 0x00045B00, x4_port0x1_port2: 0x00049B00, x4_port0x1_port3: 0x0004DB00, x4_port1x1_port0: 0x00081B00, x4_port1x1_port1: 0x00085B00, x4_port1x1_port2: 0x00089B00, x4_port1x1_port3: 0x0008DB00**

Offset Formula: **0x00041B00+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041B00+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
NOTE: PHY registers default values must never be changed, unless explicitly requested in one of the device documents.			
31	PhyAccssMd	RW 0x0	PCI Express PHY Access Mode Controls the indirect access type: Read or Write. 0 = Write: The data in the <PhyData> field is written to the PHY register pointed by the <PhyAddr> field. 1 = Read: Read preparation access. The value of the PHY register pointed by the <PhyAddr> field is stored in the <PhyData> field.
30	Reserved	RSVD 0x0	Reserved
29:16	PhyAddr	RW 0x0	14-bit PCI Express PHY Register Address Bits[7:0] select the register offset within the specific lane. Bits[13:8] select which lane: 0x00: Lane0 0x01: Lane1 0x02: Lane2 0x03: Lane3 0x2X: Broadcast to all lanes.
15:0	PhyData	RW 0x0	PCI Express PHY Register Data

Copyright © 2014 Marvell

A.9.6 PCI Express BAR Control

Table 881: PCI Express BAR1 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043804, x4_port3x1_port0: 0x00083804, x4_port0x1_port0: 0x00041804, x4_port0x1_port1: 0x00045804, x4_port0x1_port2: 0x00049804, x4_port0x1_port3: 0x0004D804, x4_port1x1_port0: 0x00081804, x4_port1x1_port1: 0x00085804, x4_port1x1_port2: 0x00089804, x4_port1x1_port3: 0x0008D804

Offset Formula: 0x00041804+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041804+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Bar1Size	RW 0x3FFF	BAR1 Size BAR sizes range from 64 KByte to 4 GByte in powers of 2. Default value: 03FFF = 1 GByte
15:1	Reserved	RSVD 0x0	Reserved
0	Bar1En	RW 0x1	BAR1 Enable 0 = Disable 1 = Enable

Table 882: PCI Express BAR2 Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043808, x4_port3x1_port0: 0x00083808, x4_port0x1_port0: 0x00041808, x4_port0x1_port1: 0x00045808, x4_port0x1_port2: 0x00049808, x4_port0x1_port3: 0x0004D808, x4_port1x1_port0: 0x00081808, x4_port1x1_port1: 0x00085808, x4_port1x1_port2: 0x00089808, x4_port1x1_port3: 0x0008D808

Offset Formula: 0x00041808+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041808+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:16	Bar2Size	RW 0x0FFF	BAR2 Size BAR sizes range from 64 KByte to 4 GByte in powers of 2. Default value: 0x0FFF = 256 MByte
15:1	Reserved	RSVD 0x0	Reserved
0	Bar2En	RW 0x1	BAR2 Enable 0 = Disable 1 = Enable

Table 883: PCI Express Expansion ROM BAR Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x0004380C, x4_port3x1_port0: 0x0008380C, x4_port0x1_port0: 0x0004180C, x4_port0x1_port1: 0x0004580C, x4_port0x1_port2: 0x0004980C, x4_port0x1_port3: 0x0004D80C, x4_port1x1_port0: 0x0008180C, x4_port1x1_port1: 0x0008580C, x4_port1x1_port2: 0x0008980C, x4_port1x1_port3: 0x0008D80C

Offset Formula: 0x0004180C+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x0004180C+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:22	Reserved	RSVD 0x0	Reserved
21:19	ExpROMSz	RW 0x0	Expansion ROM Address Bank Size 0 = 512 KByte 1 = 1024 KByte 2 = 2048 KByte 3 = 4096 KByte
18:1	Reserved	RSVD 0x0	Reserved
0	ExpROMEn	RW 0x0	Expansion ROM BAR Enable 0 = Disable 1 = Enable

A.9.7 PCI Express Interrupt

Table 884: PCI Express Interrupt Cause Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043900, x4_port3x1_port0: 0x00083900, x4_port0x1_port0: 0x00041900, x4_port0x1_port1: 0x00045900, x4_port0x1_port2: 0x00049900, x4_port0x1_port3: 0x0004D900, x4_port1x1_port0: 0x00081900, x4_port1x1_port1: 0x00085900, x4_port1x1_port2: 0x00089900, x4_port1x1_port3: 0x0008D900

Offset Formula: 0x00041900+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041900+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
All bits, except bits[27:24], are RW0C only. A cause bit sets upon an event occurrence. A write of 0 clears the bit. A write of 1 has no effect. Bits[24:27] are set and cleared upon reception of interrupt emulation messages.			
31	RcvMsi	RW0C 0x0	RC received MemWr that hit the MSI attributes.
30	Reserved	RW0C 0x0	Reserved

Table 884: PCI Express Interrupt Cause Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043900, x4_port3x1_port0: 0x00083900, x4_port0x1_port0: 0x00041900, x4_port0x1_port1: 0x00045900, x4_port0x1_port2: 0x00049900, x4_port0x1_port3: 0x0004D900, x4_port1x1_port0: 0x00081900, x4_port1x1_port1: 0x00085900, x4_port1x1_port2: 0x00089900, x4_port1x1_port3: 0x0008D900

Offset Formula: 0x00041900+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041900+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
29	RcvTurnOff	RW0C 0x0	If PCIe is Endpoint, RcvTurnOffMsg: Receive Turn Off MSG from the RC. If PCIe is RootComplex, Ready4TurnOff: RC notifies the Host that the Turn Off Process is completed (whether normally or by a timer).
28	RcvPmPme	RW0C 0x0	Received PM_PME message.
27	RcvIntD	RO 0x0	IntDn status Reflects IntD Interrupt message emulation status. Set when IntD_Assert message received. Cleared when IntD_Deassert message received, or upon link failure scenario (Dl_down). **NOTE:** This bit is not RW0C as other bits in this register, since it is cleared by the interrupting device downstream. Relevant for Root Complex only.
26	RcvIntC	RO 0x0	IntC status Reflects IntC Interrupt message emulation status. Set when IntC_Assert message received. Cleared when IntC_Deassert message received, or upon link failure scenario (Dl_down). **NOTE:** This bit is not RW0C as other bits in this register, since it is cleared by the interrupting device downstream. Relevant for Root Complex only.
25	RcvIntB	RO 0x0	IntB status Reflects IntB Interrupt message emulation status. Set when IntB_Assert message received. Cleared when IntB_Deassert message received, or upon link failure scenario (Dl_down). **NOTE:** This bit is not RW0C as other bits in this register, since it is cleared by the interrupting device downstream. Relevant for Root Complex only.

Table 884: PCI Express Interrupt Cause Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043900, x4_port3x1_port0: 0x00083900, x4_port0x1_port0: 0x00041900, x4_port0x1_port1: 0x00045900, x4_port0x1_port2: 0x00049900, x4_port0x1_port3: 0x0004D900, x4_port1x1_port0: 0x00081900, x4_port1x1_port1: 0x00085900, x4_port1x1_port2: 0x00089900, x4_port1x1_port3: 0x0008D900**

Offset Formula: **0x00041900+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041900+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
24	RcvIntA	RO 0x0	IntA status Reflects IntA Interrupt message emulation status. Set when IntA_Assert message received. Cleared when IntA_Deassert message received, or upon link failure scenario (Dl_down). **NOTE:** This bit is not RW0C as other bits in this register, since it is cleared by the interrupting device downstream. Relevant for Root Complex only.
23	PCIeLinkFail	RW0C 0x0	Link Failure indication PCI Express link dropped from active state (L0, L0s or L1) to Detect state due to link errors. **NOTE:** When dropping to Detect via Hot Reset, Disable Link or Loopback states, the interrupt is not asserted. Sticky bit--not initialized by reset.
22	PCIeSlvLb	RW0C 0x0	Slave Loopback Indication The bit sets when the opposite device on the PCI Express port is acting as a loopback master, and loopback mode was entered. **NOTE:** Sticky bit--not initialized by reset. **NOTE:**
21	PCIeSlvDisLink	RW0C 0x0	Slave Disable Link Indication The bit sets when the opposite device on the PCI Express port is acting as a disable link master, and link was disabled. **NOTE:** Sticky bit--not initialized by reset. **NOTE:**
20	PCIeSlvHot Reset	RW0C 0x0	Received Hot Reset Indication The bit sets when a hot-reset indication is received from the opposite device on the PCI Express port. **NOTE:** Sticky bit--not initialized by reset. **NOTE:**
19	RcvCRS	RW0C 0x0	Received CRS completion status. A downstream PCI Express device can respond to a configuration request with CRS (Configuration Request Retry Status) if it is not ready yet to serve the request. RcvCRS interrupt is set when such a completion status is received.

Table 884: PCI Express Interrupt Cause Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043900, x4_port3x1_port0: 0x00083900, x4_port0x1_port0: 0x00041900, x4_port0x1_port1: 0x00045900, x4_port0x1_port2: 0x00049900, x4_port0x1_port3: 0x0004D900, x4_port1x1_port0: 0x00081900, x4_port1x1_port1: 0x00085900, x4_port1x1_port2: 0x00089900, x4_port1x1_port3: 0x0008D900

Offset Formula: 0x00041900+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041900+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
18	RcvErrCor	RW0C 0x0	Received ERR_COR message. Set when a downstream device detected the error and sent an error message upstream. Relevant for Root Complex only.
17	RcvErrNonFatal	RW0C 0x0	Received ERR_NONFATAL message. Set when a downstream device detected the error and sent an error message upstream. Relevant for Root Complex only.
16	RcvErrFatal	RW0C 0x0	Received ERR_FATAL message. Set when a downstream device detected the error and sent an error message upstream. Relevant for Root Complex only.
15	RcvUrCaErr	RW0C 0x0	Received either UR or CA completion status.
14	FlowCtrlProtocol	RW0C 0x0	Flow Control Protocol Error Set when there is no FC DLLP for more than 200 usec.
13	Reserved	RW0C 0x0	Reserved
12	BIST	RW0C 0x0	PCI Express BIST Enable BIST is not supported.
11	DstateChange	RW0C 0x0	Dstate Change Indication Any change in the Dstate asserts this bit. **NOTE:** This bit is relevant only for Endpoint. **NOTE:**
10	FErrDet	RW0C 0x0	Fatal Error Detected Indicates status of Fatal errors detected by the device.
9	NFErrDet	RW0C 0x0	Non-Fatal Error Detected Indicates status of Non-Fatal errors detected by the device.
8	CorErrDet	RW0C 0x0	Correctable Error Detected Indicates status of correctable errors detected by the device.

Table 884: PCI Express Interrupt Cause Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: **x4_port2x1_port0: 0x00043900, x4_port3x1_port0: 0x00083900, x4_port0x1_port0: 0x00041900, x4_port0x1_port1: 0x00045900, x4_port0x1_port2: 0x00049900, x4_port0x1_port3: 0x0004D900, x4_port1x1_port0: 0x00081900, x4_port1x1_port1: 0x00085900, x4_port1x1_port2: 0x00089900, x4_port1x1_port3: 0x0008D900**

Offset Formula: **0x00041900+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041900+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
7	TxRamParErr	RW0C 0x0	Tx RAM Parity Error
6	RxRamParErr	RW0C 0x0	Rx RAM Parity Error
5	Reserved	RW0C 0x0	Reserved
4	HitDfltWinErr	RW0C 0x0	Hit Default Window Error
3	ErrWrToReg	RW0C 0x0	Erroneous write attempt to internal register. Set when an erroneous write request to PCI Express internal register is received, either from the PCI Express (EP set) or from the internal bus (bit[64] set).
2	Reserved	RW0C 0x0	Reserved
1	MDis	RW0C 0x0	Attempt to generate a PCI transaction while master is disabled.
0	TxReqInDldownErr	RW0C 0x0	Receive TxReq while PCI Express is in Dl-Down.

Table 885: PCI Express Interrupt Mask Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: **x4_port2x1_port0: 0x00043910, x4_port3x1_port0: 0x00083910, x4_port0x1_port0: 0x00041910, x4_port0x1_port1: 0x00045910, x4_port0x1_port2: 0x00049910, x4_port0x1_port3: 0x0004D910, x4_port1x1_port0: 0x00081910, x4_port1x1_port1: 0x00085910, x4_port1x1_port2: 0x00089910, x4_port1x1_port3: 0x0008D910**

Offset Formula: **0x00041910+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port**
0x00041910+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:24	Mask	RW 0x0	Mask bit per cause bit. If a bit is set to 1, the corresponding event is enabled. Mask does not affect setting of the Interrupt Cause register bits; it only affects the assertion of the interrupt.

Copyright © 2014 Marvell

Table 885: PCI Express Interrupt Mask Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043910, x4_port3x1_port0: 0x00083910, x4_port0x1_port0: 0x00041910, x4_port0x1_port1: 0x00045910, x4_port0x1_port2: 0x00049910, x4_port0x1_port3: 0x0004D910, x4_port1x1_port0: 0x00081910, x4_port1x1_port1: 0x00085910, x4_port1x1_port2: 0x00089910, x4_port1x1_port3: 0x0008D910

Offset Formula: 0x00041910+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041910+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
23:20	Mask	RW 0x0	Mask bit per cause bit. If a bit is set to 1, the corresponding event is enabled. Mask does not affect setting of the Interrupt Cause register bits; it only affects the assertion of the interrupt.
19:0	Mask	RW 0x0	Mask bit per cause bit. If a bit is set to 1, the corresponding event is enabled. Mask does not affect setting of the Interrupt Cause register bits; it only affects the assertion of the interrupt.

A.9.8 PCI Express Debug

Table 886: PCI Express Debug Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A60, x4_port3x1_port0: 0x00083A60, x4_port0x1_port0: 0x00041A60, x4_port0x1_port1: 0x00045A60, x4_port0x1_port2: 0x00049A60, x4_port0x1_port3: 0x0004DA60, x4_port1x1_port0: 0x00081A60, x4_port1x1_port1: 0x00085A60, x4_port1x1_port2: 0x00089A60, x4_port1x1_port3: 0x0008DA60

Offset Formula: 0x00041A60+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A60+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0xF	Reserved
23	ForcePhyReset	RW 0x0	**NOTE:** that in case of quad pex with one PHY reset will effects all quad links. **NOTE:** Sticky bit -- not initialized by reset. 0 = Deassert PHY Reset 1 = Assert PHY Reset
22	Reserved	RSVD 0x1	Reserved
21	DisHotResetRegRst	RW 0x1	Disable PCIe Register File reset upon hot reset. **NOTE:** Sticky bit -- not initialized by reset. If set to 0x0 (reset is enabled), Regfile initialization is necessary again, if it overwrites the PCIe configurations. 0 = Disable: Mask Disable 1 = Enable: Mask Enable

Table 886: PCI Express Debug Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043A60, x4_port3x1_port0: 0x00083A60, x4_port0x1_port0: 0x00041A60, x4_port0x1_port1: 0x00045A60, x4_port0x1_port2: 0x00049A60, x4_port0x1_port3: 0x0004DA60, x4_port1x1_port0: 0x00081A60, x4_port1x1_port1: 0x00085A60, x4_port1x1_port2: 0x00089A60, x4_port1x1_port3: 0x0008DA60

Offset Formula: 0x00041A60+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A60+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
20	SoftReset	RW 0x0	PCI Express Unit Soft Reset When set, generates an internal reset in the PCIe unit. This bit should be cleared after the link is re-negotiated. In RC mode, by default, this reset will not reset the register file or generate pcie_rst_out_. Before setting this bit, make sure there are no pending TLP in the unit. 0 = Disable 1 = Enable
19	DisLinkRestartRegRst	RW 0x1	Disable PCIe Register File reset upon Link restart (as a result of link disable prosedure or link instability) Note: Sticky bit - not initialized by reset 0 = Disable: Mask Disable 1 = Enable: Mask Enable
18	Reserved	RSVD 0x0	Reserved
17	ConfMskHotReset	RW 0x1	Mask Hot Reset from PCIe Reset Output **NOTE:** Sticky bit -- not initialized by reset. 0 = Disable: Mask Disable 1 = Enable: Mask Enable
16	ConfMskLnkRestart	RW 0x1	Mask Reset propagation to PCIE Reset Output in case of link restart event (as a result of link disable prosedure or link instability) Note: Sticky bit - not initialized by reset 0 = Disable: Mask Disable 1 = Enable: Mask Enable
15:0	Reserved	RSVD 0xF0C0	Reserved

Table 887: PCI Express Debug Status Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A64, x4_port3x1_port0: 0x00083A64, x4_port0x1_port0: 0x00041A64, x4_port0x1_port1: 0x00045A64, x4_port0x1_port2: 0x00049A64, x4_port0x1_port3: 0x0004DA64, x4_port1x1_port0: 0x00081A64, x4_port1x1_port1: 0x00085A64, x4_port1x1_port2: 0x00089A64, x4_port1x1_port3: 0x0008DA64

Offset Formula: 0x00041A64+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A64+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

Table 888: PCI Express Debug MAC Status 1 Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043A84, x4_port3x1_port0: 0x00083A84, x4_port0x1_port0: 0x00041A84, x4_port0x1_port1: 0x00045A84, x4_port0x1_port2: 0x00049A84, x4_port0x1_port3: 0x0004DA84, x4_port1x1_port0: 0x00081A84, x4_port1x1_port1: 0x00085A84, x4_port1x1_port2: 0x00089A84, x4_port1x1_port3: 0x0008DA84

Offset Formula: 0x00041A84+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041A84+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27:24	TrnCntlBits	RO 0x0	Training control status bits sent on TX: Bit 27 - Scramble disable Bit 26 - Master loopback Bit 25 - Link disable Bit 24 - Hot reset
23:0	Reserved	RSVD 0x0	Reserved

Table 889: PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1)

Offset: x4_port2x1_port0: 0x00043AB0, x4_port3x1_port0: 0x00083AB0, x4_port0x1_port0: 0x00041AB0, x4_port0x1_port1: 0x00045AB0, x4_port0x1_port2: 0x00049AB0, x4_port0x1_port3: 0x0004DAB0, x4_port1x1_port0: 0x00081AB0, x4_port1x1_port1: 0x00085AB0, x4_port1x1_port2: 0x00089AB0, x4_port1x1_port3: 0x0008DAB0

Offset Formula: 0x00041AB0+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041AB0+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
31:14	Reserved	RSVD 0x0	Reserved
13:10	PDedicateBufferSize	RW 0x3	Indicates the number of entries used for posted tlps allocation. The maximum number for X1 mode is 5 , for X4 mode is 11. The number of CPL entries is calculated as 6 - <PDedicateBufferSize> for X1 mode and 12 - <PDedicateBufferSize> for X4 mode.
9	PCplDedicateAlloc	RW 0x0	If asserted, indicates that the dynamic buffer allocation is disabled and the dedicated buffer sizes are used for completion (CPL) and posted (P) buffers. 0 = Dynamic allocation: Dynamic allocation is enabled. 1 = Dedicated allocation: CPL and P tlps each have their own resources.
8	UpdateDedicatedBufferAlloc	RW 0x0	To latch the allocation of Posted and Completion buffers from the <PDedicateBufferSize> field of current register, the user must write 1’b1 to this field. Otherwise, the design will assume that the size of Completion and Posted buffers are equal. To update the size of buffers again, without initiating a fundamental reset, write 1’b0 to this field and then 1’b1 again.
7	Reserved	RSVD 0x0	Reserved
6	Reserved	RW 0x0	Reserved
5	Reserved	RW 0x0	Reserved
4	Reserved	RW 0x0	Reserved
3	TxNpPushDis	RW 0x0	Tx Transaction Ordering Control (Non Posted vs. Posted) 0 = Disable 1 = Enable: Tx non posted may bypass posted.
2	TxCmplPushDis	RW 0x0	Tx Transaction Ordering Control (Completion vs. Posted) 0 = Disable 1 = Enable: Tx completion may bypass posted.

Copyright © 2014 Marvell

Table 889: PCI Express TL Control Register (x=0–0, x=0–3, y=2–3, y=0–1) (Continued)

Offset: x4_port2x1_port0: 0x00043AB0, x4_port3x1_port0: 0x00083AB0, x4_port0x1_port0: 0x00041AB0, x4_port0x1_port1: 0x00045AB0, x4_port0x1_port2: 0x00049AB0, x4_port0x1_port3: 0x0004DAB0, x4_port1x1_port0: 0x00081AB0, x4_port1x1_port1: 0x00085AB0, x4_port1x1_port2: 0x00089AB0, x4_port1x1_port3: 0x0008DAB0

Offset Formula: 0x00041AB0+y*0x40000+x*0x4000: where y (0-1) represents x4_port, where x (0-3) represents x1_port
0x00041AB0+(y-2)*0x40000+x*0x4000+0x2000: where x (0-0) represents x1_port, where y (2-3) represents x4_port

Bit	Field	Type/InitVal	Description
1	RxNpPushDis	RW 0x0	Rx Transaction Ordering Control (Non Posted vs. Posted) 0 = Disable 1 = Enable: Rx non posted may bypass posted.
0	RxCmplPushDis	RW 0x0	Rx Transaction Ordering Control (Completion vs. Posted) 0 = Disable 1 = Enable: Rx completion may bypass posted.

A.9.9 PCI Express Communication PHY

Table 890: Power and PLL Control Register

Offset: 0x00000004

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	pu_ivref	RW 0x1	Power Up Reference. Analog current and voltage reference circuits power control. 0 = Power down 1 = Power up
14	PU_PLL	RW 0x1	Power Up PLL. This bit only works in the PHY Isolate Mode. 0 = Power down 1 = Power up
13	PU_RX	RW 0x1	Power Up Receiver. This bit only works in the PHY Isolate Mode. 0 = Power down 1 = Power up
12	PU_TX	RW 0x1	Power Up Transmitter. This bit only works in the PHY Isolate Mode. 0 = Power down 1 = Power up
11:8	Reserved	RSVD 0x8	Reserved

Table 890: Power and PLL Control Register (Continued)

Offset: 0x00000004

Bit	Field	Type/InitVal	Description
7:5	PHY_MODE	RW 0x3	PHY_MODE These bits control the mode selection. 0x0: SATA. 0x3: PCIe. 0x4: SERDES (GbE).
4:0	REF_FREF_SEL	RW 0x0	Reference Frequency Select. These bits indicate the reference clock speed. SATA 0: 20.0 MHz 1: 25.0 MHz SerDes (GbE) 0x0: 20.0 MHz 0x1: 25.0 MHz PCIe PHY 0x0: 100 MHz

Table 891: KVCO Calibration Control Register

Offset: 0x00000008

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x43	Reserved

Table 892: PHY Test Control 0 Register

Offset: 0x00000054

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	PT_EN	RW 0x0	PHY Test Enable 0: PHY Test disable 1: PHY Test enable Pattern must be selected while PT_EN (R15h [15]) = 0. When set to 1, this signal clears the pattern and error counts and begins looking for pattern lock. When set to zero this stops the testing and freeze the error and pattern counts.

Table 892: PHY Test Control 0 Register (Continued)

Offset: 0x00000054

Bit	Field	Type/InitVal	Description
14	PT_RST	RW 0x0	PHY Test Reset. This is a reset signal to PHY Test. Once its set to 1, all registers in PHY Test are cleared.
13	PT_PASS	RO 0x0	PHY Test Pass 0: 1 or more errors or the pattern is not locked 1: Pattern is locked and no errors detected.
12	PT_LOCK	RO 0x0	PHY Test Pattern Lock 0: Pattern detector is not locked onto the pattern 1: Patter detector had locked onto the pattern If the pattern doesn't lock then either the signal quality is low or the pattern provided to the receiver doesn't match the programmed pattern.
11	PT_XFER_DIFF	RW 0x0	PHY Test Transmit Jitter Pattern Select Option 0: PT_DATA[7:0] selects jitter test pattern for both transmit and receive data 1: PT_DATA [95:88] selects jitter test pattern for transmit data and PT_DATA [7:0] selects pattern for receive data comparison When this signal is 1 the transmitter and receiver can be programmed to use different jitter patterns. This field only has meaning when using the jitter patterns PT_PATTERN_SEL (R15h [7:4]) = 1110b. Both fields PT_DATA [95:88] and PT_DATA [7:0] have the same encoding.
10	PT_PATSYN_EN	RW 0x0	PHY Test Pattern Sync Enable If enabled, a pattern sync signal is generated and pulsed at the pattern repeat point. This can be used as a pattern trigger for test equipment.
9	PT_START_RD	RW 0x0	PHY Test Start Running Disparity Selects the initial running disparity for SATA test patterns. 0: Initial disparity for pattern is negative 1: Initial disparity for pattern is positive
8	PT_LONG_SHORT	RW 0x0	PHY Test Long SATA Patterns 0: Short version of SATA patterns 1: Long version of SATA patterns
7:4	PT_PATTERN_SEL	RW 0x0	PHY Test Pattern Select This selects the pattern from the table of patterns or specifies an alternate table to use for additional jitter test patterns. Patterns can only be selected when PT_EN (R15h [15]) = 0.

Table 892: PHY Test Control 0 Register (Continued)

Offset: 0x00000054

Bit	Field	Type/InitVal	Description
3:0	PT_LOCK_CNT	RW 0x0	Pattern Lock Count Program with the number of pattern signatures to be found -1. Once n + 1 of these signatures has been found PT_LOCK (R15h [12]) is asserted and the bit error test begins.

Table 893: PHY Test PRBS Error Counter 0 Register

Offset: 0x0000007C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	PT_PRBS_ERR_CNT[31:16]	RO 0x0	PHY Test Error Count [31:16] Indicates how may bit errors were encountered after obtaining pattern lock.

Table 894: PHY Test PRBS Error Counter 1 Register

Offset: 0x00000080

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	PT_PRBS_ERR_CNT[15:0]	RO 0x0	PHY Test Error Count [15:0] The error count indicates how may bit errors were encountered after obtaining pattern lock.

Table 895: PHY Test OOB 0 Register

Offset: 0x00000084

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	PT_OOB_EN	RW 0x0	PHY Test Out Of Band (OOB) Enable This allows PHY Test to control the idle signal to generate pattern that resembles OOB. Must set PT_EN (R54h [15]) to run.
14	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 895: PHY Test OOB 0 Register (Continued)

Offset: 0x00000084

Bit	Field	Type/InitVal	Description
13:12	PT_TESTMODE	RW 0x0	Test Mode. 0: Test Disable 1: Test Enable
11:10	PT_OOB_SPEED	RW 0x0	PHY Test Out Of Band (OOB) Speed 0h: Every bit of the OOB pattern is transmitted 1 time (Use when programmed to 1.5G). 1h: Every bit of the OOB pattern is transmitted 2 times (Use when programmed to 3G). 2h: Every bit of the OOB pattern is transmitted 4 times (Use when programmed to 6G). 4h: Every bit of the OOB pattern is transmitted 8 times (reserved for future speed).
9:8	PT_OOB_PAT_SEL	RW 0x0	PHY Test Out Of Band (OOB) Pattern Select Selects which pattern is transmitted during a emulated OOB burst. 00: Transmit the align primitive. 01: Transmit D24.3 pattern. 10: Reserved - do not use. 11: Reserved - do not use.
7:0	PT_OOB_IDLE_LEN	RW 0x0	Out Of Band (OOB) Idle Length. Specifies the emulated OOB gap width. IDLE period = pt_oob_idle_len (R84h [7:0]) × 40 UI.

Table 896: Digital Loopback Enable Register

Offset: 0x0000008C

Bit	Field	Type/InitVal	Description
31:12	Reserved	RSVD 0x0	Reserved
11:10	SEL_BITS	RW 0x1	Select Data Bit Width. 00: 10-bit 01: 20-bit 10: 40-bit 11: Reserved
9	CDR_LOCK_MODE	RW 0x0	CDR Lock Mode. 0: CDR lock with clock only. 1: CDR lock with both clock and data.

Table 896: Digital Loopback Enable Register (Continued)

Offset: 0x0000008C

Bit	Field	Type/InitVal	Description
8	CDR_LOCK_DET_ EN	RW 0x0	CDR Lock Detection Enable. 0: CDR lock detection is disabled. 1: CDR lock detection is enabled.
7:0	Reserved	RSVD 0x0	Reserved

Table 897: PHY Isolation Mode Control Register

Offset: 0x00000098

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	PHY Isolate Mode	RW 0x0	PHY Isolation Mode Enable. Enable is used to isolate the PHY from the outside logic to test the stand-alone mode. 0 = Disable. 1 = Enable.
14	EOM_CK_ALIGN_ EN	RW 0x0	EOM Clock Align Enable. To enable EOM clock and data clock alignment circuits.
13:0	Reserved	RSVD 0x100	Reserved

Table 898: Reference Clock Select Register

Offset: 0x00000118

Bit	Field	Type/InitVal	Description
31:11	Reserved	RSVD 0x0	Reserved
10	REFCLK_SEL	RW 0x0	0x0: RefClk0 (100MHz for PCIe). 0x1: RefClk1 (25MHz for SATA & GbE).
9:0	Reserved	RSVD 0x0	Reserved

Table 899: COMPHY Control Register
Offset: 0x00000120

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x1210	Reserved
2	RX_HIZ	RW 0x1	RX High Impedance Mode.
1:0	Reserved	RSVD 0x0	Reserved

A.9.10 PCI Express PHY Pipe

Table 900: LANE_CFG0 Lane Configuration 0 Register
Offset: 0x00000080

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	CfgForceRxPresent	RW 0x0	Receiver Detection Override 0 = UNIPHY: Use value from UNIPHY. 1 = NotDetected: Force receiver NOT detected for this lane. 2 = Force: Force receiver detected for this lane. 3 = Force1: Force receiver detected for this lane.
13	CfgFastSynch	RW 0x0	Fast Synchronization at 125 MHz PCLK Frequency 1 = Single-cycle: Single-cycle synchronization 1 = Half-cycle: Half-cycle synchronization
12:9	CfgElbThreshold	RW 0x8	Elastic Buffer Quiescent Threshold **NOTE:** All other values are reserved. 5 = Minimum: Minimum value for common REFCLK 8 = 600ppm: 600 ppm adjustment 9 = >600ppm: Beyond 600 ppm adjustment 10 = Max600ppm: Maximum value for 600 ppm adjustment 13 = MaxCommon: Maximum value for common REFCLK

Table 900: LANE_CFG0 Lane Configuration 0 Register (Continued)

Offset: 0x00000080

Bit	Field	Type/InitVal	Description
8:5	CfgTxAlignPos	RW 0x0	TX alignment shift position (for dphy_ana_txdata) 0x0 = Shift by 0 bits. 0x1, 0x2 = Shift by 1 bit. 0x3, 0x4 = Shift by 2 bits. 0x5, 0x6 = Shift by 3 bits. 0x7 = Shift by 4 bits. 0x8 = Shift by 5 bits. 0x9, 0xA = Shift by 6 bits. 0xB, 0xC = Shift by 7 bits. 0xD, 0xE = Shift by 8 bits. 0xF = Shift by 9 bits.
4	PrdTxSwing	RW 0x0	Replaces mac_phy_txswing when mode_margin_override = 1 (bit2 in GLOB_TEST_CTRL register 0xc2),
3:1	PrdTxMargin	RW 0x0	Replaces mac_phy_txmargin when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2),
0	PrdTxDeEmph	RW 0x0	Replaces mac_phy_txdeemph when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2).

Table 901: LANE_CFG1 Lane Configuration 1 Register

Offset: 0x00000081

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	Reserved	RO 0x0	Reserved
14:11	Reserved	RW 0x0	Reserved
10	CfgUseGen2PllCal	RW 0x0	Use Gen2 speed for PLL calibration. 0 = Gen1: Use Gen1 speed for PLL calibration. 1 = Gen2: Use Gen2 speed for PLL calibration.
9	CfgUseMaxPllRate	RW 0x0	Use maximum PLL rate mode at the Common PHY. 0 = Non-maximum: Non-maximum PLL rate mode at Common PHY 1 = Maximum: Maximum PLL rate mode at Common PHY mode
8	CfgSpdChangeWait	RW 0x0	Speed change wait period when in maximum PLL rate mode. 0 = 3.2us 1 = 6.4us

Table 901: LANE_CFG1 Lane Configuration 1 Register (Continued)
Offset: 0x00000081

Bit	Field	Type/InitVal	Description
7	CfgDisableTxDetVal	RW 0x0	Disable txdetrx valid signal during TX Detect RX. 0 = Enable: Enable txdetrx valid signal for TX Detect RX 1 = Disable: Disable txdetrx valid signal for TX Detect RX.
6	CfgTxDetRxMode	RW 0x0	Tx Detect Rx Mode 1 = Low-z: Tx Detect Rx at low-z mode 1 = High-z: Tx Detect Rx at high-z mode
5	CfgAlignIdleHiZ	RW 0x0	Align the phase of the internal idle high-z control signal with the idle low-z control signal. 0 = Non-Aligned: High-z off before low-z off. 1 = Aligned: Both high-z off and low-off are in the same phase.
4:3	CfgEn2TxDataDly	RW 0x2	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen2 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.
2:1	CfgGen1TxDataDly	RW 0x0	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen1 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.
0	CfgTxElecIdleMode	RW 0x1	Tx Electrical Idle Mode Enable 0 = LOw: Transmitter is at low impedance mode during Electrical Idle. 1 = High: Transmitter is at high impedance mode during Electrical Idle.

Table 902: LANE_BEACON Lane Beacon Control Register
Offset: 0x00000085

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:8	Reserved	RO 0x0	Reserved
7	BeaconDetected	RO 0x0	A low-frequency beacon is detected.
6	PmoBeaconTxEn	RW 0x0	Replaces dphy_ana_beacon_tx_en at PM override mode.

Table 902: LANE_BEACON Lane Beacon Control Register (Continued)
Offset: 0x00000085

Bit	Field	Type/InitVal	Description
5:2	CfgBeaconTxLoZWait	RW 0x5	Beacon transmit low-z wait period, in units of 20 us.
1	CfgBeaconRxEn	RW 0x0	Rx Beacon Mode Enable
0	CfgBeaconTxEn	RW 0x0	Tx Beacon Mode Enable

Table 903: GLOB_CLK_CTRL Reset and Clock Control Register
Offset: 0x000000C1

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:9	Reserved	RO 0x0	Reserved
8	PhyReset	RW 0x0	Common PHY Reset
7	ModePclkCtrl	RW 0x0	PCLK Output Control 0 = Regular: Pclk is not inverted 1 = Inverted: Pclk is inverted.
6	ModeP2PhyStatus	RW 0x0	PHY Status Behavior During P2 0 = Low: Phy_mac_phystatus is low. 1 = High: Phy_mac_phystatus reamains high.
5:4	ModeRefDiv	RW 0x2	Reference Clock Divisor This bit determines how the sclk is generated from the refclk_int. 0 = 1: Divided by 1 1 = 2: Divided by 2 2 = 4: Divided by 4 3 = 8: Divided by 8
3	ModeP1ClkReqN	RW 0x0	Clock Removal in P1 This bit controls whether the pclk is replaced by the oscclk and clk_req_n asserted when mac_phy_clk_req_n is asserted during P1 0 = NotReplaced: The pclk is not replaced by oscclk in P1. 1 = Replaced: The pclk is replaced by oscclk in P1.

Table 903: GLOB_CLK_CTRL Reset and Clock Control Register (Continued)
Offset: 0x000000C1

Bit	Field	Type/InitVal	Description
2	ModeFixedPclk	RW 0x0	PHY Datapath Width Mode 0 = 16b/16b: 16-bit @125 MHz at 2.5 GT/s 16-bit @ 250 MHz at 5.0 GT/s 1 = 8b/16b: 8-bit @ 250 MHz at 2.5 GT/s 16-bit @ 250 MHz at 5.0 GT/s
1	RegReset	RW 0x0	PIPE Register Reset When asserted, this bit resets all the registers to their default values (except the reset bits). Any register writes will have no effect.
0	SoftReset	RW 0x1	PIPE Soft Reset This bit is OR-ed with the power-on reset of the PHY. This bit does not effect register values.

Table 904: GLOB_TEST_CTRL Test Mode Control Register
Offset: 0x000000C2

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	Reserved	RO 0x0	Reserved
13	ModeRstOverride	RW 0x0	Override Common PHY Reset 0 = Normal: Normal operation 1 = Override: Override Common PHY reset. The reset sequence is backward compatible with rev 1.0.
12	ModeLbSerdes	RW 0x0	Enable SERDES Loopback 0 = Disable 1 = Enable
11	ModeLbDeep	RW 0x0	Enable Deep Loopback 0 = Disable 1 = Enable
10	ModeLbShallow	RW 0x0	Enable Shallow Loopback 0 = Disable 1 = Enable
9	ModeMulticast	RW 0x0	Select Multicast Register Mode 0 = Normal: Normal mode 1 = WriteAll: A write to a lane-specific register will result in a write to all lanes. The lane address bits will be ignored in this mode.

Table 904: GLOB_TEST_CTRL Test Mode Control Register (Continued)

Offset: 0x000000C2

Bit	Field	Type/InitVal	Description
8:3	DbgTestbusSel	RW 0x0	Select Test Bus Lane 0x00-0x0C = PIPE testbus 0x0D = PHY testbus 0x0E-0x0F = PIPE testbus
2	ModeMarginOverride	RW 0x0	Override Margining Controls from the MAC 0 = MAC: Use margining signals from MAC. 1 = Configuration: Use margining signals from lane configuration.
1	ModePmOverride	RW 0x0	Override PM Control Outputs 0 = Normal: Normal operation 1 = Override: Override PM control outputs
0	mode_bist	RW 0x0	BIST Mode Enable When BIST mode is selected, common PCLK must be used. The BIST logic clock is enabled. 0 = Normal: Normal mode 1 = BIST: BIST mode

Table 905: GLOB_CLK_SRC_LO Clock Source Low Register

Offset: 0x000000C3

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	ModeClkSrc	RW 0x1	Each bit indicates that PCLK is generated from that lane. Exactly one lane must be selected within each link. This is the selection for lanes 0 through 15. When no lane is selected (including lanes 16 through 31), the REFCLK is used as the PCLK for all lanes.

Table 906: GLOB_TRIGGER Register

Offset: 0x000000C5

Bit	Field	Type/InitVal	Description
This register is used to generate various trigger pulses. The trigger pulse length should be long enough to guarantee that the sampling clock domain will be able to detect a rising edge of the pulse.			
31:0	Reserved	RSVD 0x0	Reserved

Table 907: GLOB_DP_CFG Datapath Configuration Register

Offset: 0x000000C8

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:13	Reserved	RO 0x0	Reserved
12	CfgTxElecIdleAssert	RW 0x0	Common PHY Tx Idle HighZ (tx_idle_hiz) Assert Timing 0 = 2ClockCycleDelay: Assert tx_idle_hiz two clock cycles later after sending the last symbol of EIOS. 1 = Immediate: Assert tx_idle_hiz immediately after sending the last symbol of EIOS.
11:10	CfgGen2TxElecIdleDly	RW 0x0	Common PHY Tx Idle HighZ (tx_idle_hiz) Timing Gen2 0 = Synchronous: Tx_idle_hiz is synchronous with txdata. 1 = 1ClkCycleDelay: Tx_idle_hiz is delayed by 1 clock cycle. 2 = 2ClkCycleDelay: Tx_idle_hiz is delayed by 2 clock cycles. 3 = 3ClkCycleDelay: Tx_idle_hiz is delayed by 3 clock cycles.
9	CfgSalFreeze	RW 0x0	Symbol Alignment State Machine Freeze 0 = Normal: Normal operation 1 = Modified: Symbol alignment state machine stops. This mode is meant to be used temporarily while symbol alignment weights are being modified.
8	CfgAlwaysAlign	RW 0x0	Symbol Alignment Mode 0 = Normal: Normal operation: Symbol alignment is readjusted only when COM received and SYNC_FAIL state. 1 = Readjusted: Symbol alignment is always readjusted when COM symbol is received.
7	CfgDisableSkp	RW 0x0	SKP Ordered Set Handling Mode 0 = Normal: Normal operation 1 = SKPDetectDisabled: SKP detection is disabled for the purposed of elastic buffer adjustment (overflow and underflow are still handled).
6	CfgMaskErrors	RW 0x0	8b/10b Decoder Error Masking 0 = Normal: Normal operation 1 = Masked: 8b/10b decoder errors are masked (off). This is only used for symbol alignment state machine.
5	CfgDisableEdb	RW 0x0	EDB Replacement of Error Symbols 0 = Replaced: All symbols with 8b/10b errors are replaced by EDB. 1 = NotReplaced: Symbols are never replaced by EDB.

Table 907: GLOB_DP_CFG Datapath Configuration Register (Continued)
Offset: 0x000000C8

Bit	Field	Type/InitVal	Description
4	CfgNoDispError	RW 0x0	Disparity Error Handling Mode 0 = Separately: Disparity errors are handled separately from 8b/10b errors. 1 = All: All decoder errors (including disparity errors) are handled as 8b/10b errors, including EDB.
3	CfgPassRxInfo	RW 0x0	Rx Data Mode 0 = Zero: Rxdata, rxdatak, and rxstatus are 0 when rxvalid=0. 1 = Retain: Rxdata, rxdatak, and rxstatus retains their values regardless of the value of rxvalid.
2:1	CfgGen1TxElecIdleDly	RW 0x2	Common PHY Tx Idle HighZ (tx_idle_hiz) Timing Gen1 0 = Synchronous: Tx_idle_hiz is synchronous with txdata. 1 = 1ClkCycleDelay: Tx_idle_hiz is delayed by 1 clock cycle. 2 = 2ClkCycleDelay: Tx_idle_hiz is delayed by 2 clock cycles. 3 = 3ClkCycleDelay: Tx_idle_hiz is delayed by 3 clock cycles.
0	CfgIgnorePhyRdy	RW 0x1	Common PHY phy_rdy Handling 0 = Positive: Positive edge on phy_rdy is required before rxvalid is asserted. 1 = Ignored: Rxvalid logic ignores phy_rdy.

Table 908: GLOB_PM_CFG0 Power Management Timing Parameter Register
Offset: 0x000000D0

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:12	CfgPmOscClkWait	RW 0x0	The amount of time spent in PLLON power state waiting for the completion of the last oscclk pulse on pclk/aux_clk.
11:8	CfgPmRxDenWait	RW 0x1	The amount of time spent in the decision state (to determine if Rx is present or not) after seeing the internal signal TXDETRX_VALID asserted, in units of 1.28 us.
7:0	CfgPmRxDloZWait	RW 0x0	The amount of time spent in LoZ state before receiver detection is initiated, in units of 5.12 us.

Table 909: GLOB_PM_CFG1 Power Management Timing Parameter 2 Register

Offset: 0x000000D1

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:8	CfgPmShortWait	RW 0x30	Length of the SHORT_WAIT, in sclk cycles.
7:0	CfgPmRxDetWait	RW 0x14	The amount of time to wait while rxdet=1 before rxpresent is sampled, in units of 5.12 us.

Table 910: GLOB_COUNTER_CTRL Counter Lane and Type Register

Offset:

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:12	Reserved	RO 0x0	Reserved
11:8	CounterType	RW 0x0	Counter Type 0xD-0xF are reserved. 0 = 8b/10bErr: 8b/10b error 1 = DispErr: Disparity error 2 = 8b/10b/DispErr: 8b/10b or disparity error 3 = 8b/10bErrRxValid: 8b/10b error (only during rxvalid) 4 = DispErrRxValid: Disparity error (only during rxvalid) 5 = 8b/10b/DispErrRxValid: 8b/10b or disparity error (only during rxvalid) 6 = Rxvalid1: Rxvalid = 1 7 = COM-SKP: COM-SKP pairs 8 = SKPAdded: SKP added. 9 = SKPDeleted: SKP deleted 10 = Underflow 11 = Overflow 12 = RxvalidTrans: Rxvalid transitions
7:5	Reserved	RW 0x0	Reserved
4:0	CounterLane	RW 0x0	Selects the lane from which the counter will be computed.

Table 911: GLOB_COUNTER_LO Counter Low Register
Offset:

Bit	Field	Type/InitVal	Description
This register contains the sampled value of the counter obtained by counter_sample or counter_sample_clear.			
31:16	Reserved	RSVD 0x0	Reserved
15:0	CounterSampled	RO 0x0	Low 16 bits of the sampled counter value.

Table 912: GLOB_COUNTER_HI Counter High Register
Offset:

Bit	Field	Type/InitVal	Description
This register contains the sampled value of the counter obtained by counter_sample or counter_sample_clear.			
31:16	Reserved	RSVD 0x0	Reserved
15:0	CounterSampled	RO 0x0	High 16 bits of the sampled counter value.

Table 913: LANE_CFG0 Lane Configuration 0 Register
Offset: 0x00000180

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	CfgForceRxPresent	RW 0x0	Receiver Detection Override 0 = UNIPHY: Use value from UNIPHY. 1 = NotDetected: Force receiver NOT detected for this lane. 2 = Force: Force receiver detected for this lane. 3 = Force1: Force receiver detected for this lane.
13	CfgFastSynch	RW 0x0	Fast Synchronization at 125 MHz PCLK Frequency 1 = Single-cycle: Single-cycle synchronization 1 = Half-cycle: Half-cycle synchronization
12:9	CfgElbThreshold	RW 0x8	Elastic Buffer Quiescent Threshold **NOTE:** All other values are reserved. 5 = Minimum: Minimum value for common REFCLK 8 = 600ppm: 600 ppm adjustment 9 = >600ppm: Beyond 600 ppm adjustment 10 = Max600ppm: Maximum value for 600 ppm adjustment 13 = MaxCommon: Maximum value for common REFCLK

Table 913: LANE_CFG0 Lane Configuration 0 Register (Continued)

Offset: 0x00000180

Bit	Field	Type/InitVal	Description
8:5	CfgTxAlignPos	RW 0x0	TX alignment shift position (for dphy_ana_txdata) 0x0 = Shift by 0 bits. 0x1, 0x2 = Shift by 1 bit. 0x3, 0x4 = Shift by 2 bits. 0x5, 0x6 = Shift by 3 bits. 0x7 = Shift by 4 bits. 0x8 = Shift by 5 bits. 0x9, 0xA = Shift by 6 bits. 0xB, 0xC = Shift by 7 bits. 0xD, 0xE = Shift by 8 bits. 0xF = Shift by 9 bits.
4	PrdTxSwing	RW 0x0	Replaces mac_phy_txswing when mode_margin_override = 1 (bit2 in GLOB_TEST_CTRL register 0xc2),
3:1	PrdTxMargin	RW 0x0	Replaces mac_phy_txmargin when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2),
0	PrdTxDeEmph	RW 0x0	Replaces mac_phy_txdeemph when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2).

Table 914: LANE_CFG1 Lane Configuration 1 Register

Offset: 0x00000181

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	Reserved	RO 0x0	Reserved
14:11	Reserved	RW 0x0	Reserved
10	CfgUseGen2PllCal	RW 0x0	Use Gen2 speed for PLL calibration. 0 = Gen1: Use Gen1 speed for PLL calibration. 1 = Gen2: Use Gen2 speed for PLL calibration.
9	CfgUseMaxPllRate	RW 0x0	Use maximum PLL rate mode at the Common PHY. 0 = Non-maximum: Non-maximum PLL rate mode at Common PHY 1 = Maximum: Maximum PLL rate mode at Common PHY mode
8	CfgSpdChangeWait	RW 0x0	Speed change wait period when in maximum PLL rate mode. 0 = 3.2us 1 = 6.4us

Table 914: LANE_CFG1 Lane Configuration 1 Register (Continued)

Offset: 0x00000181

Bit	Field	Type/InitVal	Description
7	CfgDisableTxDetVal	RW 0x0	Disable txdetrx valid signal during TX Detect RX. 0 = Enable: Enable txdetrx valid signal for TX Detect RX 1 = Disable: Disable txdetrx valid signal for TX Detect RX.
6	CfgTxDetRxMode	RW 0x0	Tx Detect Rx Mode 1 = Low-z: Tx Detect Rx at low-z mode 1 = High-z: Tx Detect Rx at high-z mode
5	CfgAlignIdleHiZ	RW 0x0	Align the phase of the internal idle high-z control signal with the idle low-z control signal. 0 = Non-Aligned: High-z off before low-z off. 1 = Aligned: Both high-z off and low-off are in the same phase.
4:3	CfgEn2TxDataDly	RW 0x2	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen2 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.
2:1	CfgGen1TxDataDly	RW 0x0	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen1 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.
0	CfgTxElecIdleMode	RW 0x1	Tx Electrical Idle Mode Enable 0 = LOw: Transmitter is at low impedance mode during Electrical Idle. 1 = High: Transmitter is at high impedance mode during Electrical Idle.

Table 915: LANE_BEACON Lane Beacon Control Register

Offset: 0x00000185

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:8	Reserved	RO 0x0	Reserved
7	BeaconDetected	RO 0x0	A low-frequency beacon is detected.
6	PmoBeaconTxEn	RW 0x0	Replaces dphy_ana_beacon_tx_en at PM override mode.

Table 915: LANE_BEACON Lane Beacon Control Register (Continued)

Offset: 0x00000185

Bit	Field	Type/InitVal	Description
5:2	CfgBeaconTxLoZWait	RW 0x5	Beacon transmit low-z wait period, in units of 20 us.
1	CfgBeaconRxEn	RW 0x0	Rx Beacon Mode Enable
0	CfgBeaconTxEn	RW 0x0	Tx Beacon Mode Enable

Table 916: LANE_CFG0 Lane Configuration 0 Register

Offset: 0x00000280

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	CfgForceRxPresent	RW 0x0	Receiver Detection Override 0 = UNIPHY: Use value from UNIPHY. 1 = NotDetected: Force receiver NOT detected for this lane. 2 = Force: Force receiver detected for this lane. 3 = Force1: Force receiver detected for this lane.
13	CfgFastSynch	RW 0x0	Fast Synchronization at 125 MHz PCLK Frequency 1 = Single-cycle: Single-cycle synchronization 1 = Half-cycle: Half-cycle synchronization
12:9	CfgElbThreshold	RW 0x8	Elastic Buffer Quiescent Threshold **NOTE:** All other values are reserved. 5 = Minimum: Minimum value for common REFCLK 8 = 600ppm: 600 ppm adjustment 9 = >600ppm: Beyond 600 ppm adjustment 10 = Max600ppm: Maximum value for 600 ppm adjustment 13 = MaxCommon: Maximum value for common REFCLK
8:5	CfgTxAlignPos	RW 0x0	TX alignment shift position (for dphy_ana_txdata) 0x0 = Shift by 0 bits. 0x1, 0x2 = Shift by 1 bit. 0x3, 0x4 = Shift by 2 bits. 0x5, 0x6 = Shift by 3 bits. 0x7 = Shift by 4 bits. 0x8 = Shift by 5 bits. 0x9, 0xA = Shift by 6 bits. 0xB, 0xC = Shift by 7 bits. 0xD, 0xE = Shift by 8 bits. 0xF = Shift by 9 bits.

Table 916: LANE_CFG0 Lane Configuration 0 Register (Continued)

Offset: 0x00000280

Bit	Field	Type/InitVal	Description
4	PrdTxSwing	RW 0x0	Replaces mac_phy_txswing when mode_margin_override = 1 (bit2 in GLOB_TEST_CTRL register 0xc2),
3:1	PrdTxMargin	RW 0x0	Replaces mac_phy_txmargin when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2),
0	PrdTxDeEmph	RW 0x0	Replaces mac_phy_txdeemph when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2).

Table 917: LANE_CFG1 Lane Configuration 1 Register

Offset: 0x00000281

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	Reserved	RO 0x0	Reserved
14:11	Reserved	RW 0x0	Reserved
10	CfgUseGen2PllCal	RW 0x0	Use Gen2 speed for PLL calibration. 0 = Gen1: Use Gen1 speed for PLL calibration. 1 = Gen2: Use Gen2 speed for PLL calibration.
9	CfgUseMaxPllRate	RW 0x0	Use maximum PLL rate mode at the Common PHY. 0 = Non-maximum: Non-maximum PLL rate mode at Common PHY 1 = Maximum: Maximum PLL rate mode at Common PHY mode
8	CfgSpdChangeWait	RW 0x0	Speed change wait period when in maximum PLL rate mode. 0 = 3.2us 1 = 6.4us
7	CfgDisableTxDetVal	RW 0x0	Disable txdetrx valid signal during TX Detect RX. 0 = Enable: Enable txdetrx valid signal for TX Detect RX 1 = Disable: Disable txdetrx valid signal for TX Detect RX.
6	CfgTxDetRxMode	RW 0x0	Tx Detect Rx Mode 1 = Low-z: Tx Detect Rx at low-z mode 1 = High-z: Tx Detect Rx at high-z mode

Table 917: LANE_CFG1 Lane Configuration 1 Register (Continued)

Offset: 0x00000281

Bit	Field	Type/InitVal	Description
5	CfgAlignIdleHiZ	RW 0x0	Align the phase of the internal idle high-z control signal with the idle low-z control signal. 0 = Non-Aligned: High-z off before low-z off. 1 = Aligned: Both high-z off and low-off are in the same phase.
4:3	CfgEn2TxDataDly	RW 0x2	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen2 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.
2:1	CfgGen1TxDataDly	RW 0x0	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen1 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.
0	CfgTxElecIdleMode	RW 0x1	Tx Electrical Idle Mode Enable 0 = LOw: Transmitter is at low impedance mode during Electrical Idle. 1 = High: Transmitter is at high impedance mode during Electrical Idle.

Table 918: LANE_BEACON Lane Beacon Control Register

Offset: 0x00000285

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:8	Reserved	RO 0x0	Reserved
7	BeaconDetected	RO 0x0	A low-frequency beacon is detected.
6	PmoBeaconTxEn	RW 0x0	Replaces dphy_ana_beacon_tx_en at PM override mode.
5:2	CfgBeaconTxLoZWait	RW 0x5	Beacon transmit low-z wait period, in units of 20 us.
1	CfgBeaconRxEn	RW 0x0	Rx Beacon Mode Enable
0	CfgBeaconTxEn	RW 0x0	Tx Beacon Mode Enable

Table 919: LANE_CFG0 Lane Configuration 0 Register
Offset: 0x00000380

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	CfgForceRxPresent	RW 0x0	Receiver Detection Override 0 = UNIPHY: Use value from UNIPHY. 1 = NotDetected: Force receiver NOT detected for this lane. 2 = Force: Force receiver detected for this lane. 3 = Force1: Force receiver detected for this lane.
13	CfgFastSynch	RW 0x0	Fast Synchronization at 125 MHz PCLK Frequency 1 = Single-cycle: Single-cycle synchronization 1 = Half-cycle: Half-cycle synchronization
12:9	CfgElbThreshold	RW 0x8	Elastic Buffer Quiescent Threshold **NOTE:** All other values are reserved. 5 = Minimum: Minimum value for common REFCLK 8 = 600ppm: 600 ppm adjustment 9 = >600ppm: Beyond 600 ppm adjustment 10 = Max600ppm: Maximum value for 600 ppm adjustment 13 = MaxCommon: Maximum value for common REFCLK
8:5	CfgTxAlignPos	RW 0x0	TX alignment shift position (for dphy_ana_txdata) 0x0 = Shift by 0 bits. 0x1, 0x2 = Shift by 1 bit. 0x3, 0x4 = Shift by 2 bits. 0x5, 0x6 = Shift by 3 bits. 0x7 = Shift by 4 bits. 0x8 = Shift by 5 bits. 0x9, 0xA = Shift by 6 bits. 0xB, 0xC = Shift by 7 bits. 0xD, 0xE = Shift by 8 bits. 0xF = Shift by 9 bits.
4	PrdTxSwing	RW 0x0	Replaces mac_phy_txswing when mode_margin_override = 1 (bit2 in GLOB_TEST_CTRL register 0xc2),
3:1	PrdTxMargin	RW 0x0	Replaces mac_phy_txmargin when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2),
0	PrdTxDeEmph	RW 0x0	Replaces mac_phy_txdeemph when mode_margin_override=1 (bit2 in GLOB_TEST_CTRL register 0xc2).

Table 920: LANE_CFG1 Lane Configuration 1 Register

Offset: 0x00000381

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	Reserved	RO 0x0	Reserved
14:11	Reserved	RW 0x0	Reserved
10	CfgUseGen2PllCal	RW 0x0	Use Gen2 speed for PLL calibration. 0 = Gen1: Use Gen1 speed for PLL calibration. 1 = Gen2: Use Gen2 speed for PLL calibration.
9	CfgUseMaxPllRate	RW 0x0	Use maximum PLL rate mode at the Common PHY. 0 = Non-maximum: Non-maximum PLL rate mode at Common PHY 1 = Maximum: Maximum PLL rate mode at Common PHY mode
8	CfgSpdChangeWait	RW 0x0	Speed change wait period when in maximum PLL rate mode. 0 = 3.2us 1 = 6.4us
7	CfgDisableTxDetVal	RW 0x0	Disable txdetrx valid signal during TX Detect RX. 0 = Enable: Enable txdetrx valid signal for TX Detect RX 1 = Disable: Disable txdetrx valid signal for TX Detect RX.
6	CfgTxDetRxMode	RW 0x0	Tx Detect Rx Mode 1 = Low-z: Tx Detect Rx at low-z mode 1 = High-z: Tx Detect Rx at high-z mode
5	CfgAlignIdleHiZ	RW 0x0	Align the phase of the internal idle high-z control signal with the idle low-z control signal. 0 = Non-Aligned: High-z off before low-z off. 1 = Aligned: Both high-z off and low-off are in the same phase.
4:3	CfgEn2TxDataDly	RW 0x2	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen2 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.
2:1	CfgGen1TxDataDly	RW 0x0	Tx datapath delayed latency (to accommodate high-z off latency of internal analog circuit in Gen1 mode. 0 = None: No delay 1 = 1ClkCycles: Delayed by 1 clock cycle. 2 = 2ClkCycles: Delayed by 2 clock cycles. 3 = 3ClkCycles: Delayed by 3 clock cycles.

Table 920: LANE_CFG1 Lane Configuration 1 Register (Continued)

Offset: 0x00000381

Bit	Field	Type/InitVal	Description
0	CfgTxElecIdleMode	RW 0x1	Tx Electrical Idle Mode Enable 0 = LOw: Transmitter is at low impedance mode during Electrical Idle. 1 = High: Transmitter is at high impedance mode during Electrical Idle.

Table 921: LANE_BEACON Lane Beacon Control Register

Offset: 0x00000385

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:8	Reserved	RO 0x0	Reserved
7	BeaconDetected	RO 0x0	A low-frequency beacon is detected.
6	PmoBeaconTxEn	RW 0x0	Replaces dphy_ana_beacon_tx_en at PM override mode.
5:2	CfgBeaconTxLoZWait	RW 0x5	Beacon transmit low-z wait period, in units of 20 us.
1	CfgBeaconRxEn	RW 0x0	Rx Beacon Mode Enable
0	CfgBeaconTxEn	RW 0x0	Tx Beacon Mode Enable

A.10 USB 2.0 Registers

The following table provides a summarized list of all registers that belong to the USB 2.0, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 922: Summary Map Table for the USB 2.0 Registers

Register Name	Offset	Table and Page
USB 2.0 Bridge Interrupt and Error Registers		
USB 2.0 Bridge Interrupt Cause Register (n=0–2)	port0: 0x00050310, port1: 0x00051310, port2: 0x00052310	Table 923, p. 1208
USB 2.0 Bridge Interrupt Mask Register (n=0–2)	port0: 0x00050314, port1: 0x00051314, port2: 0x00052314	Table 924, p. 1208
USB 2.0 Core Interrupt Mask Register (n=0–2)	port0: 0x00050318, port1: 0x00051318, port2: 0x00052318	Table 925, p. 1209
USB 2.0 Bridge Error Address Register (n=0–2)	port0: 0x0005031C, port1: 0x0005131C, port2: 0x0005231C	Table 926, p. 1209
USB 2.0 Bridge Control and Status Registers		
USB 2.0 Bridge Control Register (n=0–2)	port0: 0x00050300, port1: 0x00051300, port2: 0x00052300	Table 927, p. 1209
USB 2.0 Bridge Address Decoding Registers		
USB 2.0 Window0 Control Register (n=0–2)	port0: 0x00050320, port1: 0x00051320, port2: 0x00052320	Table 928, p. 1210
USB 2.0 Window0 Base Register (n=0–2)	port0: 0x00050324, port1: 0x00051324, port2: 0x00052324	Table 929, p. 1211
USB 2.0 Window1 Control Register (n=0–2)	port0: 0x00050330, port1: 0x00051330, port2: 0x00052330	Table 930, p. 1211
USB 2.0 Window1 Base Register (n=0–2)	port0: 0x00050334, port1: 0x00051334, port2: 0x00052334	Table 931, p. 1212
USB 2.0 Window2 Control Register (n=0–2)	port0: 0x00050340, port1: 0x00051340, port2: 0x00052340	Table 932, p. 1212
USB 2.0 Window2 Base Register (n=0–2)	port0: 0x00050344, port1: 0x00051344, port2: 0x00052344	Table 933, p. 1213
USB 2.0 Window3 Control Register (n=0–2)	port0: 0x00050350, port1: 0x00051350, port2: 0x00052350	Table 934, p. 1213

Table 922: Summary Map Table for the USB 2.0 Registers (Continued)

Register Name	Offset	Table and Page
USB 2.0 Window3 Base Register (n=0–2)	port0: 0x00050354, port1: 0x00051354, port2: 0x00052354	Table 935, p. 1214
USB 2.0 PHY Registers		
USB 2.0 Power Control Register (n=0–2)	port0: 0x00050400, port1: 0x00051400, port2: 0x00052400	Table 936, p. 1214
USB 2.0 PHY Suspend/Resume Control Register (n=0–2)	port0: 0x00050404, port1: 0x00051404, port2: 0x00052404	Table 937, p. 1214

A.10.1 USB 2.0 Bridge Interrupt and Error Registers

Table 923: USB 2.0 Bridge Interrupt Cause Register (n=0–2)

Offset: port0: 0x00050310, port1: 0x00051310, port2: 0x00052310

Offset Formula: 0x00050310+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
NOTE: All cause bits are clear only. They are set to `1? upon an interrupt event and cleared when the software writes a value of 0. Writing 1 has no affect.			
31:7	Reserved	RO 0x0	Reserved
6:4	Reserved	RW0C 0x0	Reserved
3	DUF	RW0C 0x0	USB Device underflow
2	DOF	RW0C 0x0	USB Host/Device overflow
1	HOF	RW0C 0x0	USB Host Underflow
0	AddrDecErr	RW0C 0x0	Address Decoding Error Asserted upon address decoding error

Table 924: USB 2.0 Bridge Interrupt Mask Register (n=0–2)

Offset: port0: 0x00050314, port1: 0x00051314, port2: 0x00052314

Offset Formula: 0x00050314+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:7	Reserved	RSVD 0x0	Reserved
6:4	Reserved	RW 0x0	Reserved

Table 924: USB 2.0 Bridge Interrupt Mask Register (n=0–2) (Continued)

Offset: port0: 0x00050314, port1: 0x00051314, port2: 0x00052314

Offset Formula: 0x00050314+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
3:0	Mask	RW 0x0	If set to 1, the related interrupt is enabled.

Table 925: USB 2.0 Core Interrupt Mask Register (n=0–2)

Offset: port0: 0x00050318, port1: 0x00051318, port2: 0x00052318

Offset Formula: 0x00050318+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	UsbIntrMask	RW 0x1	When set to 1, USB port interrupts are routed the CPU core. If set to 0, USB port interrupts are not routed to the CPU core. 0 = Mask 1 = Not_Mask

Table 926: USB 2.0 Bridge Error Address Register (n=0–2)

Offset: port0: 0x0005031C, port1: 0x0005131C, port2: 0x0005231C

Offset Formula: 0x0005031C+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:0	ErrAddr	RO 0x0	Error Address Latched upon any of the address decoding errors (address miss, multiple hit). Once the address is latched, no new address is latched until SW reads it (Read access to USB 2.0 Bridge Error Address Register).

A.10.2 USB 2.0 Bridge Control and Status Registers

Table 927: USB 2.0 Bridge Control Register (n=0–2)

Offset: port0: 0x00050300, port1: 0x00051300, port2: 0x00052300

Offset Formula: 0x00050300+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x0	Reserved
9:6	Reserved	RW 0x0	Reserved

Table 927: USB 2.0 Bridge Control Register (n=0–2) (Continued)

Offset: port0: 0x00050300, port1: 0x00051300, port2: 0x00052300

Offset Formula: 0x00050300+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
5	Reserved	RW 0x0	Reserved 0 = reserved0 1 = reserved1
4	BS	RW 0x1	USB Core Byte Swap 0 = Byte swap: Byte swap over 64-bit qword, upon USB core read/write access from memory. 1 = No byte swap
3	Reserved	RW 0x0	Reserved
2	Reserved	RW 0x0	Reserved 0 = reserved0 1 = reserved1
1	Reserved	RW 0x0	0 = reserved0 1 = reserved1
0	Reserved	RW 0x0	Reserved 0 = reserved0 1 = reserved1

A.10.3 USB 2.0 Bridge Address Decoding Registers

Table 928: USB 2.0 Window0 Control Register (n=0–2)

Offset: port0: 0x00050320, port1: 0x00051320, port2: 0x00052320

Offset Formula: 0x00050320+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as sequence of 1s followed by sequence of 0's. The number of 1s specifies the size of the window (for example, a value of 0x00FF specifies 256 x 64k = 16 MB).
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this window. **NOTE:** See the "PCI Express Address Decoding" section. **NOTE:**
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See the Units IDs and Attributes table. **NOTE:** See the "PCI Express Address Decoding" section. Must be configured to the SDRAM Controller. **NOTE:**

Table 928: USB 2.0 Window0 Control Register (n=0–2) (Continued)

Offset: port0: 0x00050320, port1: 0x00051320, port2: 0x00052320

Offset Formula: 0x00050320+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
3:2	Reserved	RSVD 0x0	Reserved
1	WrBL	RW 0x0	USB to Mbus Burst Write Limit 0 = No limit 1 = Limit: Limit writes not to cross 32B boundary
0	win_en	RW 0x0	Window0 Enable 0 = Disable 1 = Enable

Table 929: USB 2.0 Window0 Base Register (n=0–2)

Offset: port0: 0x00050324, port1: 0x00051324, port2: 0x00052324

Offset Formula: 0x00050324+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Used with the size field to set the address window size and location. Corresponds to transaction address[31:16]
15:0	Reserved	RSVD 0x0	Reserved

Table 930: USB 2.0 Window1 Control Register (n=0–2)

Offset: port0: 0x00050330, port1: 0x00051330, port2: 0x00052330

Offset Formula: 0x00050330+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this window. **NOTE:** See the "PCI Express Address Decoding" section. **NOTE:**
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See the Units IDs and Attributes table. **NOTE:** See the "PCI Express Address Decoding" section. Must be configured to the SDRAM Controller. **NOTE:**

Table 930: USB 2.0 Window1 Control Register (n=0–2) (Continued)

Offset: port0: 0x00050330, port1: 0x00051330, port2: 0x00052330

Offset Formula: 0x00050330+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
3:2	Reserved	RSVD 0x0	Reserved
1	WrBL	RW 0x0	USB to Mbus Burst Write Limit 0 = No limit: No limit 1 = Limit: Limit writes not to cross 32B boundary
0	win_en	RW 0x0	Window1 Enable 0 = Disable 1 = Enable

Table 931: USB 2.0 Window1 Base Register (n=0–2)

Offset: port0: 0x00050334, port1: 0x00051334, port2: 0x00052334

Offset Formula: 0x00050334+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Used with the size field to set the address window size and location. Corresponds to transaction address[31:16]
15:0	Reserved	RSVD 0x0	Reserved

Table 932: USB 2.0 Window2 Control Register (n=0–2)

Offset: port0: 0x00050340, port1: 0x00051340, port2: 0x00052340

Offset Formula: 0x00050340+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this windo. **NOTE:** See the "PCI Express Address Decoding" section. **NOTE:**
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See the Units IDs and Attributes table. **NOTE:** See the "PCI Express Address Decoding" section. Must be configured to the SDRAM Controller. **NOTE:**
3:2	Reserved	RSVD 0x0	Reserved

Table 932: USB 2.0 Window2 Control Register (n=0–2) (Continued)

Offset: port0: 0x00050340, port1: 0x00051340, port2: 0x00052340

Offset Formula: 0x00050340+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
1	WrBL	RW 0x0	USB to Mbus Burst Write Limit 0 = No limit 1 = Limit: Limit writes not to cross 32B boundary
0	win_en	RW 0x0	Window2 Enable 0 = Disable 1 = Enable

Table 933: USB 2.0 Window2 Base Register (n=0–2)

Offset: port0: 0x00050344, port1: 0x00051344, port2: 0x00052344

Offset Formula: 0x00050344+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Used with the size field to set the address window size and location. Corresponds to transaction address[31:16]
15:0	Reserved	RSVD 0x0	Reserved

Table 934: USB 2.0 Window3 Control Register (n=0–2)

Offset: port0: 0x00050350, port1: 0x00051350, port2: 0x00052350

Offset Formula: 0x00050350+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1's specifies the size of the window (for example, a value of 0x00ff specifies 256x64k = 16 MB).
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this window. **NOTE:** See the "PCI Express Address Decoding" section. **NOTE:**
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See the Units IDs and Attributes table. **NOTE:** See the "PCI Express Address Decoding" section. Must be configured to the SDRAM Controller. **NOTE:**
3:1	Reserved	RSVD 0x0	Reserved

Table 934: USB 2.0 Window3 Control Register (n=0–2) (Continued)

Offset: port0: 0x00050350, port1: 0x00051350, port2: 0x00052350

Offset Formula: 0x00050350+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
0	win_en	RW 0x0	Window3 Enable 0 = Disable 1 = Enable

Table 935: USB 2.0 Window3 Base Register (n=0–2)

Offset: port0: 0x00050354, port1: 0x00051354, port2: 0x00052354

Offset Formula: 0x00050354+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Used with the size field to set the address window size and location. Corresponds to transaction address[31:16]
15:0	Reserved	RSVD 0x0	Reserved

A.10.4 USB 2.0 PHY Registers

Table 936: USB 2.0 Power Control Register (n=0–2)

Offset: port0: 0x00050400, port1: 0x00051400, port2: 0x00052400

Offset Formula: 0x00050400+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x1FE0002C	Reserved
2	SUSPENDM	RW 0x1	Input SUSPENDM Note:This bit is valid only when bit 1 in register suspend/resume control (0x404) is set to 0x1. 0 = Force PHY to suspend 1 = Normal operation
1:0	Reserved	RSVD 0x3	Reserved

Table 937: USB 2.0 PHY Suspend/Resume Control Register (n=0–2)

Offset: port0: 0x00050404, port1: 0x00051404, port2: 0x00052404

Offset Formula: 0x00050404+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved

Table 937: USB 2.0 PHY Suspend/Resume Control Register (n=0–2) (Continued)

Offset: port0: 0x00050404, port1: 0x00051404, port2: 0x00052404

Offset Formula: 0x00050404+n*0x1000: where n (0-2) represents port

Bit	Field	Type/InitVal	Description
1	Enable force suspend	RW 0x0	Force Suspend Enable 0 = Disable force suspend 1 = Enable force suspend: (See PHY Power Control register bit[2])
0	Reserved	RSVD 0x0	Reserved

A.11 Serial-ATA Host Controller (SATAHC) Registers

The following table provides a summarized list of all registers that belong to the Serial-ATA Host Controller (SATAHC), including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 938: Summary Map Table for the Serial-ATA Host Controller (SATAHC) Registers

Register Name	Offset	Table and Page
SATAHC Arbiter		
SATAHC Configuration Register	0x000A0000	Table 939, p. 1219
SATAHC Request Queue Out-Pointer Register	0x000A0004	Table 940, p. 1220
SATAHC Response Queue In-Pointer Register	0x000A0008	Table 941, p. 1220
SATAHC Interrupt Coalescing Threshold Register	0x000A000C	Table 942, p. 1221
SATAHC Interrupt Time Threshold Register	0x000A0010	Table 943, p. 1221
SATAHC Interrupt Cause Register	0x000A0014	Table 944, p. 1222
SATAHC Main Interrupt Cause Register	0x000A0020	Table 945, p. 1224
SATAHC Main Interrupt Mask 0 Register	0x000A0024	Table 946, p. 1225
SATAHC LED Configuration Register	0x000A002C	Table 947, p. 1226
Window0 Control Register	0x000A0030	Table 948, p. 1227
Window0 Base Register	0x000A0034	Table 949, p. 1227
Window1 Control Register	0x000A0040	Table 950, p. 1228
Window1 Base Register	0x000A0044	Table 951, p. 1228
Window2 Control Register	0x000A0050	Table 952, p. 1228
Window2 Base Register	0x000A0054	Table 953, p. 1229
Window3 Control Register	0x000A0060	Table 954, p. 1229
Window3 Base Register	0x000A0064	Table 955, p. 1230
SATAHC Interrupt Coalescing Threshold 0 Register	0x000A010C	Table 956, p. 1230
SATAHC Interrupt Time Threshold 0 Register	0x000A0110	Table 957, p. 1230
SATAHC Main Interrupt Mask 1 Register	0x000A0124	Table 958, p. 1231
SATAHC Interrupt Coalescing Threshold 1 Register	0x000A020C	Table 959, p. 1232
SATAHC Interrupt Time Threshold 1 Register	0x000A0210	Table 960, p. 1232
Basic DMA		
Basic DMA Command Register (n=0–1)	Port0: 0x000A2224, Port1: 0x000A4224	Table 961, p. 1233
Basic DMA Status Register (n=0–1)	Port0: 0x000A2228, Port1: 0x000A4228	Table 962, p. 1234
Descriptor Table Low Base Address Register (n=0–1)	Port0: 0x000A222C, Port1: 0x000A422C	Table 963, p. 1236
Descriptor Table High Base Address Register (n=0–1)	Port0: 0x000A2230, Port1: 0x000A4230	Table 964, p. 1237
Data Region Low Address Register (n=0–1)	Port0: 0x000A2234, Port1: 0x000A4234	Table 965, p. 1237
Data Region High Address Register (n=0–1)	Port0: 0x000A2238, Port1: 0x000A4238	Table 966, p. 1237

Copyright © 2014 Marvell

Table 938: Summary Map Table for the Serial-ATA Host Controller (SATAHC) Registers (Continued)

Register Name	Offset	Table and Page
Serial-ATA Interface		
Serial-ATA Interface Configuration Register (n=0–1)	Port0: 0x000A2050, Port1: 0x000A4050	Table 967, p. 1238
LP Phy External Control Register (n=0–1)	Port0: 0x000A2058, Port1: 0x000A4058	Table 968, p. 1238
LP Phy External Status Register (n=0–1)	Port0: 0x000A205C, Port1: 0x000A405C	Table 969, p. 1239
SStatus Register (n=0–1)	Port0: 0x000A2300, Port1: 0x000A4300	Table 970, p. 1239
SError Register (n=0–1)	Port0: 0x000A2304, Port1: 0x000A4304	Table 971, p. 1240
SControl Register (n=0–1)	Port0: 0x000A2308, Port1: 0x000A4308	Table 972, p. 1242
LTMode Register (n=0–1)	Port0: 0x000A230C, Port1: 0x000A430C	Table 973, p. 1243
BIST Control Register (n=0–1)	Port0: 0x000A2334, Port1: 0x000A4334	Table 974, p. 1245
BIST-Dword1 Register (n=0–1)	Port0: 0x000A2338, Port1: 0x000A4338	Table 975, p. 1245
BIST-Dword2 Register (n=0–1)	Port0: 0x000A233C, Port1: 0x000A433C	Table 976, p. 1246
SError Interrupt Mask Register (n=0–1)	Port0: 0x000A2340, Port1: 0x000A4340	Table 977, p. 1246
Serial-ATA Interface Control Register (n=0–1)	Port0: 0x000A2344, Port1: 0x000A4344	Table 978, p. 1246
Serial-ATA Interface Test Control Register (n=0–1)	Port0: 0x000A2348, Port1: 0x000A4348	Table 979, p. 1248
Serial-ATA Interface Status Register (n=0–1)	Port0: 0x000A234C, Port1: 0x000A434C	Table 980, p. 1249
Vendor Unique Register (n=0–1)	Port0: 0x000A235C, Port1: 0x000A435C	Table 981, p. 1251
FIS Configuration Register (n=0–1)	Port0: 0x000A2360, Port1: 0x000A4360	Table 982, p. 1251
FIS Interrupt Cause Register (n=0–1)	Port0: 0x000A2364, Port1: 0x000A4364	Table 983, p. 1252
FIS Interrupt Mask Register (n=0–1)	Port0: 0x000A2368, Port1: 0x000A4368	Table 984, p. 1254
FIS Dword0 Register (n=0–1)	Port0: 0x000A2370, Port1: 0x000A4370	Table 985, p. 1255
FIS Dword1 Register (n=0–1)	Port0: 0x000A2374, Port1: 0x000A4374	Table 986, p. 1255
FIS Dword2 Register (n=0–1)	Port0: 0x000A2378, Port1: 0x000A4378	Table 987, p. 1255
FIS Dword3 Register (n=0–1)	Port0: 0x000A237C, Port1: 0x000A437C	Table 988, p. 1255

Table 938: Summary Map Table for the Serial-ATA Host Controller (SATAHC) Registers (Continued)

Register Name	Offset	Table and Page
FIS Dword4 Register (n=0–1)	Port0: 0x000A2380, Port1: 0x000A4380	Table 989, p. 1255
FIS Dword5 Register (n=0–1)	Port0: 0x000A2384, Port1: 0x000A4384	Table 990, p. 1256
FIS Dword6 Register (n=0–1)	Port0: 0x000A2388, Port1: 0x000A4388	Table 991, p. 1256
PHY Configuration Register (n=0–1)	Port0: 0x000A23A0, Port1: 0x000A43A0	Table 992, p. 1256
PHYTCTL Register (n=0–1)	Port0: 0x000A23A4, Port1: 0x000A43A4	Table 993, p. 1257
PHY Mode 10 Register (n=0–1)	Port0: 0x000A23A8, Port1: 0x000A43A8	Table 994, p. 1257
PHY Mode 12 Register (n=0–1)	Port0: 0x000A23B0, Port1: 0x000A43B0	Table 995, p. 1258
EDMA		
EDMA Configuration Register (n=0–1)	Port0: 0x000A2000, Port1: 0x000A4000	Table 996, p. 1258
EDMA Interrupt Error Cause Register (n=0–1)	Port0: 0x000A2008, Port1: 0x000A4008	Table 997, p. 1262
EDMA Interrupt Error Mask Register (n=0–1)	Port0: 0x000A200C, Port1: 0x000A400C	Table 998, p. 1265
EDMA Request Queue Base Address High Register (n=0–1)	Port0: 0x000A2010, Port1: 0x000A4010	Table 999, p. 1265
EDMA Request Queue In-Pointer Register (n=0–1)	Port0: 0x000A2014, Port1: 0x000A4014	Table 1000, p. 1265
EDMA Request Queue Out-Pointer Register (n=0–1)	Port0: 0x000A2018, Port1: 0x000A4018	Table 1001, p. 1266
EDMA Response Queue Base Address High Register (n=0–1)	Port0: 0x000A201C, Port1: 0x000A401C	Table 1002, p. 1266
EDMA Response Queue In-Pointer Register (n=0–1)	Port0: 0x000A2020, Port1: 0x000A4020	Table 1003, p. 1266
EDMA Response Queue Out-Pointer Register (n=0–1)	Port0: 0x000A2024, Port1: 0x000A4024	Table 1004, p. 1267
EDMA Command Register (n=0–1)	Port0: 0x000A2028, Port1: 0x000A4028	Table 1005, p. 1267
EDMA Status Register (n=0–1)	Port0: 0x000A2030, Port1: 0x000A4030	Table 1006, p. 1269
EDMA IORdy Timeout Register (n=0–1)	Port0: 0x000A2034, Port1: 0x000A4034	Table 1007, p. 1270
EDMA Command Delay Threshold Register (n=0–1)	Port0: 0x000A2040, Port1: 0x000A4040	Table 1008, p. 1270
EDMA Halt Conditions Register (n=0–1)	Port0: 0x000A2060, Port1: 0x000A4060	Table 1009, p. 1271
EDMA NCQ0 Done/TCQ0 Outstanding Status Register (n=0–1)	Port0: 0x000A2094, Port1: 0x000A4094	Table 1010, p. 1271
SATA Communication PHY		

Table 938: Summary Map Table for the Serial-ATA Host Controller (SATAHC) Registers (Continued)

Register Name	Offset	Table and Page
Power and PLL Control Register (n=0–1)	Port0: 0x000A2804, Port1: 0x000A4804	Table 1011, p. 1271
PHY Test Control 0 Register (n=0–1)	Port0: 0x000A2854, Port1: 0x000A4854	Table 1012, p. 1272
PHY Test PRBS Error Counter 0 Register (n=0–1)	Port0: 0x000A287C, Port1: 0x000A487C	Table 1013, p. 1273
PHY Test PRBS Error Counter 1 Register (n=0–1)	Port0: 0x000A2880, Port1: 0x000A4880	Table 1014, p. 1274
PHY Test OOB 0 Register (n=0–1)	Port0: 0x000A2884, Port1: 0x000A4884	Table 1015, p. 1274
Digital Loopback Enable Register (n=0–1)	Port0: 0x000A288C, Port1: 0x000A488C	Table 1016, p. 1275
PHY Isolation Mode Control Register (n=0–1)	Port0: 0x000A2898, Port1: 0x000A4898	Table 1017, p. 1276
Reference Clock Select Register (n=0–1)	Port0: 0x000A2918, Port1: 0x000A4918	Table 1018, p. 1276
COMPHY Control Register (n=0–1)	Port0: 0x000A2920, Port1: 0x000A4920	Table 1019, p. 1276

A.11.1 SATAHC Arbiter

Table 939: SATAHC Configuration Register

Offset: 0x000A0000

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	Port1CoalDis	RW 0x0	Port1 Coalescing Disable 0 = CoalEnable: When command complete it can assert <SaIntCoal> in the SATAHC Interrupt Cause Register by using the logic in <SATAHC Interrupt Coalescing Threshold> or <SATAHC Interrupt Time Threshold>. 1 = CoalDisable: <SaIntCoal> in the SATAHC Interrupt Cause will never be asserted by this port command complete.
24	Port0CoalDis	RW 0x0	Port0 Coalescing Disable 0 = CoalEnable: When command complete it can assert <SaIntCoal> in the SATAHC Interrupt Cause Register by using the logic in <SATAHC Interrupt Coalescing Threshold> or <SATAHC Interrupt Time Threshold>. 1 = CoalDisable: <SaIntCoal> in the SATAHC Interrupt Cause will never be asserted by this port command complete.

Table 939: SATAHC Configuration Register (Continued)

Offset: 0x000A0000

Bit	Field	Type/InitVal	Description
23:17	Reserved	RSVD 0x0	Reserved
16	TimeoutEn	RW 0x1	Mbus Arbiter Timer Enable 0 = Enable 1 = Disable
15:11	Reserved	RSVD 0x0	Reserved
10	PrdpBS	RW 0x1	PRDP Byte swap 0 = Byte swap 1 = No byte swap
9	EDmaBS	RW 0x1	EDMA Byte swap 0 = Byte swap 1 = No byte swap
8	DmaBS	RW 0x1	Basic DMA Byte swap 0 = Byte swap 1 = No byte swap
7:0	Timeout	RW 0xFF	SATAHC interface Mbus Arbiter Timeout Preset Value

Table 940: SATAHC Request Queue Out-Pointer Register

Offset: 0x000A0004

Bit	Field	Type/InitVal	Description
31:7	Reserved	RSVD 0x0	Reserved
6:5	Reserved	RO 0x0	Reserved
4:0	eRQQOP0	RO 0x0	EDMA Request Queue Out-Pointer This field reflects the value of bits [9:5] in the EDMA Request Queue Out-Pointer Register located in the port.

Table 941: SATAHC Response Queue In-Pointer Register

Offset: 0x000A0008

Bit	Field	Type/InitVal	Description
31:7	Reserved	RSVD 0x0	Reserved

Table 941: SATAHC Response Queue In-Pointer Register (Continued)

Offset: 0x000A0008

Bit	Field	Type/InitVal	Description
6:5	Reserved	RO 0x0	Reserved
4:0	eRPQIP0	RO 0x0	EDMA Response Queue In-Pointer Register This field reflects the value of bits [7:3] in the EDMA Response Queue In-Pointer Register located in the port.

Table 942: SATAHC Interrupt Coalescing Threshold Register

Offset: 0x000A000C

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	SAICOALT	RW 0x0	SATA Interrupt Coalescing Threshold This field provides a way to minimize the number of interrupts to off load the CPU. It defines the number of SaCrpbXDone indications before asserting the <SaIntCoal> bit in the SATAHC Interrupt Cause Register. Once the accumulated number of SaCrpbXDone indications provided by all SATAHC ports reaches the SAICOALT value, a <SaIntCoal> interrupt is asserted. When SaIntCoal is negated or when the SATAHC Interrupt Coalescing Threshold Register is written, the interrupts counter is cleared. 0x0 => Assertion of SaCrpbXDone causes an immediate assertion of <SaIntCoal>. 0x1-0xFF => <SaIntCoal> assertion is provided for every ‚ÄòSAICOALT’ times of SaCrpbXDone assertion.

Table 943: SATAHC Interrupt Time Threshold Register

Offset: 0x000A0010

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved

Table 943: SATAHC Interrupt Time Threshold Register (Continued)

Offset: 0x000A0010

Bit	Field	Type/InitVal	Description
23:0	SAITMTH	RW 0x0	SATA Interrupt Time Threshold This field provides a way to ensure maximum delay between <SaCrpb0Done> assertion and assertion bit <SaIntCoal> in SATAHC Interrupt Cause Register (even if the number of <SaCrpb0Done> indications did not reach the <SAICOALT> value). When <SaIntCoal> is negated or when the SATAHC Interrupt Time Threshold Register is written, the down counter is cleared. A new count is enabled in the assertion of the next <SaCrpb0Done> indication. When set to 0 = Assertion of <SaCrpb0Done> causes an immediate assertion of bit <SaIntCoal>. When set from 1-23 = Up to <n> internal clocks between assertion of <SaCrpb0Done> and assertion of <SaIntCoal>.

Table 944: SATAHC Interrupt Cause Register

Offset: 0x000A0014

Bit	Field	Type/InitVal	Description
NOTE: A corresponding cause bit is set every time that an interrupt occurs. A write of 0 clears the bit. A write of 1 has no effect.			
31:10	Reserved	RSVD 0x0	Reserved
9	SaDevInterrupt1	RW0C 0x0	SATA Device Interrupt This bit is set if bit <eEnEDMA> in the EDMA Command Register is cleared, and the ATA interrupt line in port 1 is active. 0 = Inactive: The ATA interrupt line was not active. 1 = Active: The ATA interrupt line was active.
8	SaDevInterrupt0	RW0C 0x0	SATA Device Interrupt This bit is set if bit <eEnEDMA> in the EDMA Command Register is cleared, and the ATA interrupt line in port 0 is active. 0 = Inactive: The ATA interrupt line was not active. 1 = Active: The ATA interrupt line was active.
7	SaIntCoal1	RW0C 0x0	SATA Interrupt Coalescing for port1 This bit is set: When the accumulated number of <SaCrpb1Done> indications from the ports that participate in the coalescing mechanism according to <CoalDis> setting in the SATAHC Configuration Register - since the last <SaIntCoal1> negation - reaches the value set in the SATAHC Interrupt Coalescing Threshold1 Register, or When the time from first <SaCrpb1Done> assertion after <SaIntCoal1> negation reaches the value set in the SATAHC Interrupt Time Threshold1 Register. 0 = No occurrence: Cause for Interrupt Coalescing did not occur. 1 = Occurs: Cause for Interrupt Coalescing occurs.

Table 944: SATAHC Interrupt Cause Register (Continued)

Offset: 0x000A0014

Bit	Field	Type/InitVal	Description
6	SaIntCoal0	RW0C 0x0	SATA Interrupt Coalescing for port0 This bit is set: When the accumulated number of <SaCrpb0Done> indications from the ports that participate in the coalescing mechanism according to <CoalDis> setting in the SATAHC Configuration Register - since the last <SaIntCoal0> negation - reaches the value set in the SATAHC Interrupt Coalescing Threshold0 Register, or When the time from first <SaCrpb0Done> assertion after <SaIntCoal0> negation reaches the value set in the SATAHC Interrupt Time Threshold0 Register. 0 = No occurrence: Cause for Interrupt Coalescing did not occur. 1 = Occurs: Cause for Interrupt Coalescing occurs.
5	Reserved	RSVD 0x0	Reserved
4	SaIntCoal	RW0C 0x0	SATA Interrupt Coalescing This bit is set: When the accumulated number of <SaCrpbXDone> indications from the ports that participate in the coalescing mechanism according to <CoalDis> setting in the SATAHC Configuration Register - since the last <SaIntCoal> negation - reaches the value set in the SATAHC Interrupt Coalescing Threshold Register, or When the time from first <SaCrpbXDone> assertion after <SaIntCoal> negation reaches the value set in the SATAHC Interrupt Time Threshold Register. 0 = No occurrence: Cause for Interrupt Coalescing did not occur. 1 = Occurs: Cause for Interrupt Coalescing occurs.
3:2	Reserved	RSVD 0x0	Reserved
1	SaCrpb1Done/DMA 1Done	RW0C 0x0	SATA CRPB 1 Done / Basic DMA 1 Done When EDMA is enable: (Bit <eEnEDMA> in the EDMA Command Register is set.) This bit is set when the EDMA in the port places a new CRPB in the response queue. 0 = A new CRPB was not placed in the response queue. 1 = A new CRPB was placed in the response queue. When EDMA is disabled: (Bit eEnEDMA in the EDMA Command Register is cleared.) This bit is set when the Basic DMA in the port completes the data transfer, clears bit <BasicDMAActive>, and moves to idle state. 0 = Basic DMA has not completed the data transfer. 1 = Basic DMA completed the data transfer. 0 = NotCompleted: Basic DMA has not completed the data transfer. 1 = Completed: Basic DMA completed the data transfer.

Table 944: SATAHC Interrupt Cause Register (Continued)

Offset: 0x000A0014

Bit	Field	Type/InitVal	Description
0	SaCrpb0Done/DMA0Done	RW0C 0x0	SATA CRPB 0 Done / Basic DMA 0 Done When EDMA is enable: (Bit <eEnEDMA> in the EDMA Command Register is set.) This bit is set when the EDMA in the port places a new CRPB in the response queue. 0 = A new CRPB was not placed in the response queue. 1 = A new CRPB was placed in the response queue. When EDMA is disabled: (Bit eEnEDMA in the EDMA Command Register is cleared.) This bit is set when the Basic DMA in the port completes the data transfer, clears bit <BasicDMAActive>, and moves to idle state. 0 = Basic DMA has not completed the data transfer. 1 = Basic DMA completed the data transfer. 0 = NotCompleted: Basic DMA has not completed the data transfer. 1 = Completed: Basic DMA completed the data transfer.

Table 945: SATAHC Main Interrupt Cause Register

Offset: 0x000A0020

Bit	Field	Type/InitVal	Description
NOTE: The bits in this register mirror the interrupt indications coming from the other SATAHC interrupt cause registers.			
31:9	Reserved	RO 0x0	Reserved
8	SataCoalDone	RO 0x0	SATA ports coalescing done This bit is set when bit <SaIntCoal> in the SATAHC Interrupt Cause Register, of SATAHC is set. **NOTE:** SATA completion bit <SataDone> must be mask to use this bit. **NOTE:**
7	SataCoalDone1	RO 0x0	SATA ports coalescing done for port1 This bit is set when bit <SaIntCoal1> in the SATAHC Interrupt Cause Register, of SATAHC is set. **NOTE:** SATA completion bit <Sata1Done> must be mask to use this bit. **NOTE:**
6	SataCoalDone0	RO 0x0	SATA ports coalescing done for port0 This bit is set when bit <SaIntCoal0> in the SATAHC Interrupt Cause Register, of SATAHC is set. **NOTE:** SATA completion bit <Sata0Done> must be mask to use this bit. **NOTE:**
5	Sata1DmaDone	RO 0x0	SATA port1 DMA done This bit is set when both of the following occur: <SaCrpb1Done/DMA1Done> bit in SATAHC Interrupt Cause Register of SATAHC is set. <SaDevInterrupt1> in SATAHC Interrupt Cause Register of SATAHC is set.

Table 945: SATAHC Main Interrupt Cause Register (Continued)

Offset: 0x000A0020

Bit	Field	Type/InitVal	Description
4	Sata0DmaDone	RO 0x0	SATA port0 DMA done This bit is set when both of the following occur: <SaCrpb0Done/DMA0Done> bit in SATAHC Interrupt Cause Register of SATAHC is set. <SaDevInterrupt0> in SATAHC Interrupt Cause Register of SATAHC is set.
3	Sata1Done	RO 0x0	SATA port1 command done This bit is set when one of the following occurs: <SaCrpb1Done/DMA1Done> bit in SATAHC Interrupt Cause Register is set. <SaDevInterrupt1> in SATAHC Interrupt Cause Register is set.
2	Sata1Err	RO 0x0	SATA port1 error. When this bit is assert then it will assert also the sunit1_err in the device Main_Error_Cause.
1	Sata0Done	RO 0x0	SATA port0 command done This bit is set when one of the following occurs: <SaCrpb0Done/DMA0Done> bit in SATAHC Interrupt Cause Register is set. <SaDevInterrupt0> in SATAHC Interrupt Cause Register is set.
0	Sata0Err	RO 0x0	SATA port0 error When this bit is assert then it will assert also the sunit0_err in the device Main_Error_Cause.

Table 946: SATAHC Main Interrupt Mask 0 Register

Offset: 0x000A0024

Bit	Field	Type/InitVal	Description
NOTE: Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register			
31:9	Reserved	RO 0x0	Reserved
8	SataCoalDone	RW 0x0	Mask SATA ports coalescing done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
7	Reserved	RSVD 0x0	Reserved
6	SataCoalDone0	RW 0x0	Mask SATA 0 port coalescing done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.

Table 946: SATAHC Main Interrupt Mask 0 Register (Continued)

Offset: 0x000A0024

Bit	Field	Type/InitVal	Description
5	Sata1DmaDone	RW 0x0	Mask SATA port1 DMA done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
4	Sata0DmaDone	RW 0x0	Mask SATA port0 DMA done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
3	Sata1Done	RW 0x0	Mask SATA port1 command done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
2	Sata1Err	RW 0x0	Mask SATA port1 error 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
1	Sata0Done	RW 0x0	Mask SATA port0 command done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
0	Sata0Err	RW 0x0	Mask SATA port0 error 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.

Table 947: SATAHC LED Configuration Register

Offset: 0x000A002C

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved
3	led_polarity	RW 0x0	LED Polarity 0 = Invert: Invert the active indication. 1 = No change: Do not change the active indication.
2	act_presence	RW 0x0	Active Presence 0 = Only active indication: Use active indication only. 1 = Multiplex/presence indication: Multiplex active and presence indication on the same LED.
1	Reserved	RW 0x0	Reserved. Write only 0x1.

Table 947: SATAHC LED Configuration Register (Continued)
Offset: 0x000A002C

Bit	Field	Type/InitVal	Description
0	act_led_blink	RW 0x0	Active LED Blink 0 = As is: Use indication as is. 1 = Blinking: Set indication to blinking.

Table 948: Window0 Control Register
Offset: 0x000A0030

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as sequence of 1's followed by sequence of 0's. The number of 1's specifies the size of the window in 64 KByte granularity (e.g., a value of 0x0FFF specifies 256 MByte). **NOTE:** A value of 0x0 specifies 64-KByte size. **NOTE:**
15:8	Attr	RW 0x0E	Specifies the target interface attributes associated with this window.
7:4	Target	RW 0x0	Specifies the target interface associated with this window. **NOTE:** Must be configured to the SDRAM Controller. **NOTE:**
3:2	Reserved	RSVD 0x0	Reserved
1	WrBL	RW 0x0	Write to Mbus Burst Limit 0 = No Limit: No limit (up to 128Bytes burst). 1 = Limit: Limit (up to 32Bytes burst).
0	WinEn	RW 0x1	Window 0 Enable 0 = Disable 1 = Enable

Table 949: Window0 Base Register
Offset: 0x000A0034

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Used with the <Size> field to set the address window size and location. Corresponds to transaction address[31:16].
15:0	Reserved	RSVD 0x0	Reserved

Table 950: Window1 Control Register

Offset: 0x000A0040

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size See "Window0 Control Register".
15:8	Attr	RW 0x0D	Target specific attributes depending on the target interface. See "Window0 Control Register".
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See "Window0 Control Register".
3:2	Reserved	RSVD 0x0	Reserved
1	WrBL	RW 0x0	Write to Mbus Burst Limit 0 = No Limit: No limit (up to 128Bytes burst). 1 = Limit: Limit (up to 32Bytes burst).
0	WinEn	RW 0x1	Window1 Enable See "Window0 Control Register".

Table 951: Window1 Base Register

Offset: 0x000A0044

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x1000	Base Address See "Window0 Base Register".
15:0	Reserved	RSVD 0x0	Reserved

Table 952: Window2 Control Register

Offset: 0x000A0050

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size See "Window0 Control Register".
15:8	Attr	RW 0x0B	Target specific attributes depending on the target interface. See "Window0 Control Register".
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See "Window0 Control Register".

Table 952: Window2 Control Register (Continued)

Offset: 0x000A0050

Bit	Field	Type/InitVal	Description
3:2	Reserved	RSVD 0x0	Reserved
1	WrBL	RW 0x0	Write to Mbus burst limit 0 = No Limit: No limit (up to 128B burst). 1 = Limit: Limit (up to 32Bytes burst).
0	WinEn	RW 0x1	Window2 Enable See "Window0 Control Register".

Table 953: Window2 Base Register

Offset: 0x000A0054

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x2000	Base Address See "Window0 Base Register".
15:0	Reserved	RSVD 0x0	Reserved

Table 954: Window3 Control Register

Offset: 0x000A0060

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size See "Window0 Control Register".
15:8	Attr	RW 0x07	Target specific attributes depending on the target interface. See "Window0 Control Register".
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See "Window0 Control Register".
3:2	Reserved	RSVD 0x0	Reserved
1	WrBL	RW 0x0	Write to Mbus Burst Limit 0 = No Limit: No limit (up to 128B burst). 1 = Limit: Limit (up to 32Bytes burst).
0	WinEn	RW 0x1	Window3 Enable See "Window0 Control Register".

Table 955: Window3 Base Register

Offset: 0x000A0064

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x3000	Base Address See "Window0 Base Register".
15:0	Reserved	RSVD 0x0	Reserved

Table 956: SATAHC Interrupt Coalescing Threshold 0 Register

Offset: 0x000A010C

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	SAICOALT0	RW 0x0	SATA Interrupt Coalescing Threshold This field provides a way to minimize the number of interrupts to off load the CPU. It defines the number of SaCrpb0Done indications before asserting the <SaIntCoal0> bit in the SATAHC Interrupt Cause Register. Once the accumulated number of SaCrpb0Done indications provided by all SATAHC ports reaches the SAICOALT0 value, a <SaIntCoal0> interrupt is asserted. When SaIntCoal0 is negated or when the SATAHC Interrupt Coalescing Threshold Register is written, the interrupts counter is cleared. 0x0 => Assertion of SaCrpb0Done causes an immediate assertion of <SaIntCoal0>. 0x1-0xFF => <SaIntCoal0> assertion is provided for every ‚ÄòSAICOALT0’ times of SaCrpb0Done assertion.

Table 957: SATAHC Interrupt Time Threshold 0 Register

Offset: 0x000A0110

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved

Table 957: SATAHC Interrupt Time Threshold 0 Register (Continued)

Offset: 0x000A0110

Bit	Field	Type/InitVal	Description
23:0	SAITMTH0	RW 0x0	SATA Interrupt Time Threshold This field provides a way to ensure maximum delay between <SaCrpb0Done> assertion and assertion bit <SaIntCoal0> in SATAHC Interrupt Cause Register (even if the number of <SaCrpb0Done> indications did not reach the <SAICOALT0> value). When <SaIntCoal0> is negated or when the SATAHC Interrupt Time Threshold Register is written, the down counter is cleared. A new count is enabled in the assertion of the next <SaCrpb0Done> indication. When set to 0 = Assertion of <SaCrpb0Done> causes an immediate assertion of bit <SaIntCoal0>. When set from 1-23 = Up to <n> internal clocks between assertion of <SaCrpb0Done> and assertion of <SaIntCoal0>.

Table 958: SATAHC Main Interrupt Mask 1 Register

Offset: 0x000A0124

Bit	Field	Type/InitVal	Description
NOTE: Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register			
31:9	Reserved	RO 0x0	Reserved
8	Reserved	RSVD 0x0	Reserved
7	SataCoalDone1	RW 0x0	Mask SATA 1 port coalescing done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
6	Reserved	RSVD 0x0	Reserved
5	Sata1DmaDone	RW 0x0	Mask SATA port1 DMA done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
4	Sata0DmaDone	RW 0x0	Mask SATA port0 DMA done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
3	Sata1Done	RW 0x0	Mask SATA port1 command done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
2	Sata1Err	RW 0x0	Mask SATA port1 error 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.

Table 958: SATAHC Main Interrupt Mask 1 Register (Continued)

Offset: 0x000A0124

Bit	Field	Type/InitVal	Description
1	Sata0Done	RW 0x0	Mask SATA port0 command done 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.
0	Sata0Err	RW 0x0	Mask SATA port0 error 0 = Mask: Interrupt is masked. 1 = Enable: Interrupt is enabled.

Table 959: SATAHC Interrupt Coalescing Threshold 1 Register

Offset: 0x000A020C

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	SAICOALT1	RW 0x0	SATA Interrupt Coalescing Threshold This field provides a way to minimize the number of interrupts to off load the CPU. It defines the number of SaCrpb1Done indications before asserting the <SaIntCoal1> bit in the SATAHC Interrupt Cause Register. Once the accumulated number of SaCrpb1Done indications provided by all SATAHC ports reaches the SAICOALT1 value, a <SaIntCoal1> interrupt is asserted. When SaIntCoal1 is negated or when the SATAHC Interrupt Coalescing Threshold Register is written, the interrupts counter is cleared. 0x0 => Assertion of SaCrpbXDone causes an immediate assertion of <SaIntCoal1>. 0x1-0xFF => <SaIntCoal1> assertion is provided for every ‚ÄòSAICOALT1’ times of SaCrpb1Done assertion.

Table 960: SATAHC Interrupt Time Threshold 1 Register

Offset: 0x000A0210

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved

Table 960: SATAHC Interrupt Time Threshold 1 Register (Continued)
Offset: 0x000A0210

Bit	Field	Type/InitVal	Description
23:0	SAITMTH1	RW 0x0	SATA Interrupt Time Threshold This field provides a way to ensure maximum delay between <SaCrpb1Done> assertion and assertion bit <SaIntCoal1> in SATAHC Interrupt Cause Register (even if the number of <SaCrpb1Done> indications did not reach the <SAICOALT1> value). When <SaIntCoal1> is negated or when the SATAHC Interrupt Time Threshold Register is written, the down counter is cleared. A new count is enabled in the assertion of the next <SaCrpb1Done> indication. When set to 0 = Assertion of <SaCrpb1Done> causes an immediate assertion of bit <SaIntCoal1>. When set from 1-23 = Up to <n> internal clocks between assertion of <SaCrpb1Done> and assertion of <SaIntCoal1>.

A.11.2 Basic DMA

Table 961: Basic DMA Command Register (n=0–1)
Offset: Port0: 0x000A2224, Port1: 0x000A4224
Offset Formula: 0x000A2224+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	DataRegionByteCount	RW 0x0	Data Region Byte Count This field indicates the count of the data region in bytes. Bit [0] is force to 0. There is a 64 KB maximum. A value of 0 indicates 64 KB. The data in the buffer must not cross the boundary of the 32-bit address space, that is, the 32-bit high address of all data in the buffer must be identical. This field value is updated by the DMA and indicates the completion status of the DMA when bit <BasicDMAActive> in the Basic DMA Status Register is cleared to 0. The host must not write to this bit when the Basic DMA is active.
15:11	Reserved	RW 0x0	Reserved
10	ContFromPrev	RW 0x0	This bit indicates if the Basic DMA needs to continue from the point where it stopped on its previous transaction or if it needs to load a new PRD table. The Basic DMA ignores the value of this bit, if the <BasicDMAPaused> bit in the Basic DMA Status Register is cleared to 0 and a new PRD table is loaded. This bit must not be changed when the Basic DMA is active. 0 = New table: Load new PRD table 1 = Old table: Continue from the PRD table associated with its previous DMA transaction

Table 961: Basic DMA Command Register (n=0–1) (Continued)

Offset: Port0: 0x000A2224, Port1: 0x000A4224
Offset Formula: 0x000A2224+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
9	DataRegionLast	RW 0x0	This bit is valid only if bit[8] <DRegionValid> is set to 1. This bit indicates if the data region described by the <DataRegionByteCount> field [31:16] in the Data Region Low Address and Data Region High Address registers are the last data region to be transferred. This bit must not be changed when the Basic DMA is active. 0 = More data: Not last data region. 1 = Last data: Last data region to be transferred in the PRD table.
8	DRegionValid	RW 0x0	This bit indicates if the <DataRegionByteCount> field [31:16] in this register, Data Region Low Address Register and Data Region High Address Register are valid and to be used by the DMA. This bit must not be changed when the Basic DMA is active. 0 = Invalid: Data Region is not valid 1 = Valid: Data Region is valid
7:4	Reserved	RW 0x0	Reserved
3	Read	RW 0x0	This bit sets the direction of the Basic DMA transfer: This bit must not be changed when the Basic DMA is active. 0 = Read: Memory Basic reads are performed. Read from system memory and store in the device. 1 = Write: Memory Basic writes are performed. Load from the device and write to system memory.
2:1	Reserved	RW 0x0	Reserved
0	Start	RW 0x0	Basic DMA operation is enabled by setting this bit to 1. Basic DMA operation begins when this bit is detected changing from a "0" to a 1. The Basic DMA transfers data between the ATA device and memory only when this bit is set. The Basic DMA operation can be halted by writing a "0" to this bit. All state information is lost when a "0" is written. The Basic DMA operation cannot be stopped and then resumed. This bit is intended to be reset by the CPU after the data transfer is completed. If a Basic DMA operation is aborted during command execution to the drive, the host software must set bit <eAtaRst>(bit [2]) to recover.

Table 962: Basic DMA Status Register (n=0–1)

Offset: Port0: 0x000A2228, Port1: 0x000A4228
Offset Formula: 0x000A2228+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved

Table 962: Basic DMA Status Register (n=0–1) (Continued)

Offset: Port0: 0x000A2228, Port1: 0x000A4228

Offset Formula: 0x000A2228+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
3	BasicDMALast	RO 0x0	This bit is valid when bit[0] <BasicDMAActive> in this register is cleared. This bit is set when: - Bit <eEDMAFBS> in EDMA Configuration Register is set and - Bit <BasicDMAActive> is cleared and - EOT for that region is set in the region descriptor This bit is cleared when: - Bit <BasicDMAActive> is cleared and - EOT for that region is cleared in the region descriptor. 0 = Halts before: DMA halts before the last data region 1 = Halts in: DMA halts in the last data region
2	BasicDMAPaused	RO 0x0	This bit is valid when bit[0] <BasicDMAActive> in this register is cleared. This bit is set when: - Bit <eEDMAFBS> in the EDMA Configuration Register is set and - Bit[0] <BasicDMAActive> is cleared and - The last transfer for the region is not yet performed, or EOT for that region is not set in the region descriptor. This bit is cleared when: -Bit[0] <BasicDMAActive> is cleared and -The last transfer for the region is performed, where EOT for that region is set in the region descriptor OR -The start bit is cleared in the command register. 0 = Idle state: The DMA is in idle state 1 = Paused: The DMA is paused
1	BasicDMAError	RO 0x0	This bit is valid when bit[0] <BasicDMAActive> in this register is cleared. This bit is set when an error is encounters or when the DMA halts abnormally 0 = No error 1 = Error

Copyright © 2014 Marvell

Table 962: Basic DMA Status Register (n=0–1) (Continued)

Offset: Port0: 0x000A2228, Port1: 0x000A4228

Offset Formula: 0x000A2228+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
0	BasicDMAActive	RO 0x0	This bit is set when the start bit is written to the command register. This bit is cleared when the last transfer for the region is performed, where EOT for that region is set in the region descriptor or a complete FIS is transferred. It is also cleared when the start bit is cleared in the command register. When this bit is read as a 0, all data transferred form the drive during the previous Basic DMA is visible in system memory, unless the Basic DMA command was aborted. This bit is set when the start bit is written to the command register. This bit is cleared when: - The last transfer for the region is performed, where EOT for that region is set in the region descriptor OR - Bit <eEDMAFBS> in EDMA Configuration Register is set and a complete FIS is received or a complete FIS is transmitted OR -The start bit is cleared in the command register. When bit <eEarlyCompletionEn> is set, this bit is cleared as soon as last data leaves the DMA. When bit <eEarlyCompletionEn> is cleared and this bit is read as a 0, all data transferred form the drive in a read transaction during the previous Basic DMA is visible in system memory, unless the Basic DMA command was aborted. 0 = Idle DMA 1 = Active DMA

Table 963: Descriptor Table Low Base Address Register (n=0–1)

Offset: Port0: 0x000A222C, Port1: 0x000A422C

Offset Formula: 0x000A222C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: Enhanced Physical Region Descriptors are in use to enable 64-bit memory addressing. See "EDMA Physical Region Descriptors (ePRD) Table Data Structure". The CPU accesses this register for direct access to the device when the EDMA is disabled.			
31:4	Descriptor Table Base Address	RW 0x0	The Descriptor Table Base Address corresponds to address A[31:4]. The descriptor table must be 16-byte aligned.
3:0	Reserved	RSVD 0x0	Reserved

Table 964: Descriptor Table High Base Address Register (n=0–1)

Offset: Port0: 0x000A2230, Port1: 0x000A4230

Offset Formula: 0x000A2230+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: Enhanced Physical Region Descriptors are in use to enable 64-bit memory addressing. See "EDMA Physical Region Descriptors (ePRD) Table Data Structure".			
31:0	Descriptor Table Base Address	RW 0x0	The Descriptor Table Base Address corresponds to address A[63:32]

Table 965: Data Region Low Address Register (n=0–1)

Offset: Port0: 0x000A2234, Port1: 0x000A4234

Offset Formula: 0x000A2234+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: The CPU accesses this register for direct access to the device when the EDMA is disabled. Bit <eEnEDMA> in EDMA Command Register (see Table 310 on page 380) is cleared. While the EDMA is enabled, the host must not write this register. If any of these bits are written when bit <eEnEDMA> in EDMA Command Register is set, the write transaction will cause unpredictable behavior.			
31:0	DataRegion[31:0]	RW 0x0	Data Region This DWORD contains bit [31:1] of the current physical region starting address. Bit 0 must be 0. This field is updated by the DMA and indicates the completion status of the DMA when bit <BasicDMAActive> in the Basic DMA Status Register is cleared to 0. The host must not write to this bit when the Basic DMA is active.

Table 966: Data Region High Address Register (n=0–1)

Offset: Port0: 0x000A2238, Port1: 0x000A4238

Offset Formula: 0x000A2238+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: he CPU accesses this register for direct access to the device when the EDMA is disabled. Bit <eEnEDMA> in EDMA Command Register is cleared. While the EDMA is enabled, the host must not write this register. If any of these bits are written when bit <eEnEDMA> in EDMA Command Register is set, the write transaction will cause unpredictable behavior			
31:0	DataRegion[63:32]	RW 0x0	Data Region This DWORD contains bits [64:32] of the current physical region starting address. This field is updated by the DMA and indicates the completion status of the DMA when bit <BasicDMAActive> in the Basic DMA Status Register is cleared to 0. The host must not write to this bit when the Basic DMA is active.

A.11.3 Serial-ATA Interface

Table 967: Serial-ATA Interface Configuration Register (n=0–1)

Offset: Port0: 0x000A2050, Port1: 0x000A4050

Offset Formula: 0x000A2050+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: After any modification in this register, the host must set bit[2]<eAtaRst> in EDMA Command Register.			
31:12	Reserved	RSVD 0x186	Reserved
11	ComChannel	RW 0x0	Communication Channel Operating Mode. This bit defines if the Serial-ATA port functions in Target mode operation. **NOTE:** In Target mode, the ATA task registers are updated when the Register Device to Host FIS is received, regardless to the value of <BSY> bit in the ATA Status register (see "Shadow Register Block Registers Map"). 0 = Disk controller 1 = Target mode operation
10	TargetMode	RW 0x0	Target Mode. This bit defines the Serial-ATA port that functions as a target during Target mode operation. To move to Target mode, you need to clear this field and set bit[11] <ComChannel>. Any other option will result in the mode being Initiator mode. 0 = Target 1 = Initiator
9	Reserved	RSVD 0x0	Reserved
8	CommEn	RW 0x0	PHY communication enable signal to override field <DET> (see "SControl Register" setting). 0 = No override: DET setting is not overridden. 1 = Override: If DET value is 0x4, its value is overridden with a value of 0x0.
7	Gen2En	RW 0x1	Generation 2 communication speed support. 0 = Disable 1 = Enable
6:0	Reserved	RSVD 0x0	Reserved

Table 968: LP Phy External Control Register (n=0–1)

Offset: Port0: 0x000A2058, Port1: 0x000A4058

Offset Formula: 0x000A2058+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:13	Reserved	RSVD 0x0	Reserved

Table 968: LP Phy External Control Register (n=0–1) (Continued)

Offset: Port0: 0x000A2058, Port1: 0x000A4058

Offset Formula: 0x000A2058+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
12:9	PIN_PHY_GEN_RX	RW 0x1	Receiver Generation Select. SATA: 0h: 1.5 Gbps, 1h: 3 Gbps Note: Binary settings not mentioned here are not to be used.
8:5	PIN_PHY_GEN_TX	RW 0x1	Transmitter Generation Select. SATA: 0h: 1.5 Gbps, 1h: 3 Gbps Note: Binary settings not mentioned here are not to be used.
4:3	Reserved	RSVD 0x0	Reserved
2	PIN_PU_TX	RW 0x1	Power Up Transmitter. 0: Power down 1: Power up
1	PIN_PU_RX	RW 0x1	Power Up Receiver. 0: Power down 1: Power up
0	PIN_PU_PLL	RW 0x1	Power Up PLL. 0: Power down 1: Power up

Table 969: LP Phy External Status Register (n=0–1)

Offset: Port0: 0x000A205C, Port1: 0x000A405C

Offset Formula: 0x000A205C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

Table 970: SStatus Register (n=0–1)

Offset: Port0: 0x000A2300, Port1: 0x000A4300

Offset Formula: 0x000A2300+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: See the Serial-ATA specification for a detailed description.			
31:12	Reserved	RSVD 0x0	Reserved

Table 970: SStatus Register (n=0–1) (Continued)

Offset: Port0: 0x000A2300, Port1: 0x000A4300

Offset Formula: 0x000A2300+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
11:8	IPM	RO 0x0	These bits set the current interface power management state. All other values are reserved. 0 = No communication: Device not present, or communication not established. 1 = Active: Interface in active state. 2 = Partial: Interface in PARTIAL power management state. 6 = Slumber: Interface in SLUMBER power management state.
7:4	SPD	RO 0x0	These bits set whether the negotiated interface communication speed is established. All other values are reserved. 0 = No negotiation: No negotiated speed. The device is not present or communication is not established. 1 = Gen1 negotiation: Generation 1 communication rate negotiated. 2 = Gen2 negotiation: Generation 2 communication rate negotiated.
3:0	DET	RO 0x4	These bits set the interface device detection and PHY state. All other values are reserved. 0 = No communication: No device detected, and PHY communication is not established. 1 = Device with no comm: Device presence detected, but PHY communication is not established. 3 = Device with comm: Device presence detected, and PHY communication is established. 4 = PHY offline: PHY in offline mode as a result of the interface being disabled or running in a loopback mode.

Table 971: SError Register (n=0–1)

Offset: Port0: 0x000A2304, Port1: 0x000A4304

Offset Formula: 0x000A2304+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: A write of 1 clears the bits in this register. A write of 0 has no effect.			
31:27	Reserved	RSVD 0x0	Reserved
26	X	RW 0x0	Exchanged When set to 1 this bit indicates that device presence has changed since the last time this bit was cleared. The means by which the implementation determines that the device presence has changed is vendor specific. This bit may be set anytime a PHY reset initialization sequence occurs as determined by reception of the COMINIT signal whether in response to: a new device being inserted, a COMRESET having been issued, or power-up.

Table 971: SError Register (n=0–1) (Continued)

Offset: Port0: 0x000A2304, Port1: 0x000A4304

Offset Formula: 0x000A2304+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
25	Reserved	RSVD 0x0	Reserved
24	T	RW 0x0	Transport State Transition Error When set to 1, this bit indicates that an error has occurred in the transition from one state to another within the Transport layer since the last time this bit was cleared.
23	S	RW 0x0	Link Sequence Error When set to 1, this bit indicates that one or more Link state machine error conditions was encountered since the last time this bit was cleared. The Link Layer state machine defines the conditions under which the link layer detects an erroneous transition.
22	H	RW 0x0	Handshake Error When set to one, this bits indicates that one or more R_ERR handshake responses was received in response to frame transmission. Such errors may be the result of: - A CRC error detected by the recipient - A disparity or 10-bit to 8-bit decoding error - Other error conditions leading to a negative handshake on a transmitted frame.
21	C	RW 0x0	CRC Error When set to 1, this bit indicates that one or more CRC errors occurred with the Link Layer since the bit was last cleared.
20	D	RW 0x0	Disparity Error When set to one, this bit indicates that incorrect disparity was detected one or more times since the last time the bit was cleared.
19	B	RW 0x0	10-bit to 8-bit Decode Error When set to 1, this bit indicates that one or more 10-bit to 8-bit decoding errors occurred since the bit was last cleared.
18	W	RW 0x0	Comm Wake When set to 1, this bit indicates that a Comm Wake signal was detected by the PHY since the last time this bit was cleared.
17	Reserved	RSVD 0x0	Reserved
16	N	RW 0x0	PhyRdy change When set to 1, this bit indicates that the PhyRdy changed state since the last time this bit was cleared.

Table 971: SError Register (n=0–1) (Continued)

Offset: Port0: 0x000A2304, Port1: 0x000A4304
Offset Formula: 0x000A2304+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
15:11	Reserved	RSVD 0x0	Reserved
10:2	Reserved	RSVD 0x0	Reserved
1	M	RW 0x0	Recovered Communication Error Communication between the device and host was temporarily lost but was re-established. This can arise from: - A device temporarily being removed. - From a temporary loss of PHY synchronization. - From other causes and may be derived from the PhyNRdy signal between the PHY and Link layers. No action is required by the host software since the operation ultimately succeeded. However, the host software may elect to track such recovered errors to gauge overall communications integrity and potentially step-down the negotiated communication speed.
0	Reserved	RSVD 0x0	Reserved

Table 972: SControl Register (n=0–1)

Offset: Port0: 0x000A2308, Port1: 0x000A4308
Offset Formula: 0x000A2308+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: See the Serial-ATA specification for a detailed description.			
31:16	Reserved	RSVD 0x0	Reserved
15:12	SPM	RW 0x0	<SPM> field is used to select a power management state. A value written to this field is treated as a one-shot. This field will be read as 0000. All other values are reserved. This field is read only 0000. 0 = No request: No power management state transition requested. 1 = Partial: Transition to the PARTIAL power management state initiated. 2 = Slumber: Transition to the SLUMBER power management state initiated. 3 = Active: Transition to the active power management state initiated.

Table 972: SControl Register (n=0–1) (Continued)

Offset: Port0: 0x000A2308, Port1: 0x000A4308

Offset Formula: 0x000A2308+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
11:8	IPM	RW 0x0	<IPM> field represents the enabled interface power management states that can be invoked via the Serial-ATA interface power management capabilities. All other values are reserved. 0 = No restrictions: No interface power management state restrictions. 1 = Partial: Transition to the PARTIAL power management state disabled. 2 = Slumber: Transition to the SLUMBER power management state disabled. 3 = Partial/Slumber: Transition to both the PARTIAL and SLUMBER power management states disabled.
7:4	SPD	RW 0x0	<SPD> field represents the highest allowed communication speed the interface is able to negotiate. All other values are reserved. 0 = No restrictions: No speed negotiation restrictions. 1 = Less than Gen1 rate: Limit speed negotiation to a rate not greater than Generation 1 communication rate. 2 = Less than Gen2 rate: Limit speed negotiation to a rate not greater than Generation 2 communication rate.
3:0	DET	RW 0x4	<DET> field controls the host adapter device detection and interface initialization. All other values are reserved. 0 = No request: No device detection or initialization action requested. 1 = Interface comm: Perform interface communication initialization sequence to establish communication. 4 = Disable PHY: Disable the Serial-ATA interface and put the PHY in offline mode.

Table 973: LTMode Register (n=0–1)

Offset: Port0: 0x000A230C, Port1: 0x000A430C

Offset Formula: 0x000A230C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:26	Reserved	RW 0x0	Reserved
25	Reserved	RW 0x1	Reserved
24	Reserved	RW 0x1	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.

Table 973: LTMode Register (n=0–1) (Continued)

Offset: Port0: 0x000A230C, Port1: 0x000A430C

Offset Formula: 0x000A230C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
23:21	Reserved	RW 0x0	Reserved
20	Reserved	RW 0x0	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
19	Reserved	RW 0x0	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
18:17	Reserved	RW 0x1	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
16	Reserved	RW 0x1	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
15	Reserved	RW 0x0	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
14	Reserved	RW 0x0	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
13:9	Reserved	RW 0x0	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
8	Reserved	RW 0x0	Reserved. **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
7	NearEndLBEn	RW 0x1	Near-End Loopback Enable. 0 = Disable: Near-end loopback (BIST) is disabled. 1 = Enable: Near-end loopback (BIST) is enabled.
6	Reserved	RSVD 0x0	Reserved
5:0	RcvWaterMark	RW 0x30	WaterMark Receiving flow control settings (values in Dwords). When the available entry of the Rx FIFO is less than this value, Serial-ATA flow control is performed by sending HOLD primitives.

Copyright © 2014 Marvell

Table 974: BIST Control Register (n=0–1)

Offset: Port0: 0x000A2334, Port1: 0x000A4334

Offset Formula: 0x000A2334+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x80	Reserved
15:11	Reserved	RSVD 0x0	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed. **NOTE:**
10	BISTResult	RO 0x0	BIST Test Pass 0 = Passed 1 = Failed
9	BISTEn	RW 0x0	BIST Test enable On the assertion of the signal, BIST mode starts. 0 = Disable 1 = Enable
8	BISTMode	RW 0x0	BIST mode Test direction. 0 = Receiver: BIST Activate FIS Receiver mode. 1 = Transmitter: BIST Activate FIS Transmitter mode.
7:0	BISTPattern	RW 0x0	BIST Pattern Test pattern, refer to bits [15:8] of the first Dword of the BIST Activate FIS.

Table 975: BIST-Dword1 Register (n=0–1)

Offset: Port0: 0x000A2338, Port1: 0x000A4338

Offset Formula: 0x000A2338+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	BistDw1	RW 0x0	In BIST mode: (Bit <BISTEn> is set to 1 in BIST Control Register). This field is the second Dword of the BIST Activate FIS. In PHY Loopback mode: (Bit <LBEnable> is set to 1 in BIST Control Register). This field is the high Dword [63:32] of the user specified loopback pattern of the BIST Activate FIS.

Table 976: BIST-Dword2 Register (n=0–1)

Offset: Port0: 0x000A233C, Port1: 0x000A433C

Offset Formula: 0x000A233C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	BistDw2	RW 0x0	In BIST mode: (Bit <BISTEn> is set to 1 in BIST Control Register). This field is the third Dword of the BIST Activate FIS. In PHY Loopback mode: (Bit <LBEnable> is set to 1 in BIST Control Register). This field is the low Dword [31:0] of the user specified loopback pattern of the BIST Activate FIS.

Table 977: SError Interrupt Mask Register (n=0–1)

Offset: Port0: 0x000A2340, Port1: 0x000A4340

Offset Formula: 0x000A2340+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	eSErrIntMsk	RW 0x019C0000	SError Interrupt Mask Bits Each of these bits checks the corresponding bit in SError Register, and if these bits are disabled (0), they mask the interrupt. 0 = Mask 1 = No mask

Table 978: Serial-ATA Interface Control Register (n=0–1)

Offset: Port0: 0x000A2344, Port1: 0x000A4344

Offset Formula: 0x000A2344+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: When bit <eEnEDMA> in the EDMA Command Register is set, this register must not be written. If this register is written when bit <eEnEDMA> is set, the write transaction will cause unpredictable behavior.			
31:26	Reserved	RSVD 0x0	Reserved
25	SendSftRst	SC 0x0	Self-Negate When this bit is set to 1, the transport layer sends Register - Host to Device control FIS to the device.
24	ClearStatus	SC 0x0	Status Self-Clear This bit clears bits [16], [30], and [31] in the Serial-ATA Interface Status Register. 0 = No clear: Does not clear bits. 1 = Clear: Clears the bits.
23:17	Reserved	RSVD 0x0	Reserved

Table 978: Serial-ATA Interface Control Register (n=0–1) (Continued)

Offset: Port0: 0x000A2344, Port1: 0x000A4344

Offset Formula: 0x000A2344+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
16	eDMAActivate	RW 0x0	DMA Activate This bit has an effect only if the Serial-ATA port is in Target mode operation (i.e., bit <ComChannel> in the EDMA Configuration Register, is set). When this bit is set the transport layer sends (multiple) data FISs, as long as the DMA is active, to complete the data transaction associated with the command. When the port functions as a target in Target mode operation, this bit is set by the target host software to activate read data transactions from the target to the initiator. When the port functions as an initiator in Target mode operation, this bit is set when a DMA Activate FIS is received to activate the write data transaction from the initiator to the target. This bit is cleared when the DMA completes the data transaction associated with the command. 0 = Not sent: Transport layer does not send DMA data FISs. 1 = Sent: Transport layer keeps sending (multiple) DMA data FISs until the data transaction associated with the command is completed.
15:10	Reserved	RSVD 0x0	Reserved
9	VendorUqSend	RW 0x0	Send vendor Unique FIS. This bit is set to 1 by the host SW to indicate that the next DWORD written to the Vendor Unique Register is the last DWORD in the payload. Bit <VendorUqDn> of the Serial-ATA Interface Status Register is set when the transmission is completed; bit <VendorUqErr> of that same register is also set if the transmission ends with an error. When the host SW writes to the Vendor Unique Register with this bit set to 1, the Serial-ATA transport layer closes the FIS and clears this bit.
8	VendorUqMd	RW 0x0	Vendor Unique mode This bit is set to 1 to indicate that the next FIS, going to be transmitted, is a Vendor Unique FIS. Only when this bit is set, the host may write to the Vendor Unique Register. When this bit is cleared, bits <VendorUqDn> and <VendorUqErr> of the Serial-ATA Interface Status Register are also cleared. Before setting this bit the Host must verify: - No pending commands are in progress. - The EDMA is disabled, bit <eEnEDMA> in the EDMA Command Register is cleared.
7:4	Reserved	RW 0x0	Reserved
3:0	PMportTx	RW 0x0	Port Multiplier Transmit. This field specifies the Port Multiplier (bits [11:8] in DW0 of the FIS header) of any transmitted FIS.

Table 979: Serial-ATA Interface Test Control Register (n=0–1)

Offset: Port0: 0x000A2348, Port1: 0x000A4348

Offset Formula: 0x000A2348+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	PortNumDevErr	RO 0x0	Each bit in this field is set to 1 when Register - Device to Host FIS or Set Device Bits FIS is received with bit <ERR> from the corresponding PM port set to 1. All bits in this field are cleared to 0 when the value of the <eEnEDMA> bit in the EDMA Command Register is changed from 0 to 1.
15:14	TransFrmSiz	RW 0x0	Maximum Transmit Frame Size When<TransFrmSizExt> is 0: 0 = Maximum Transmit Frame Size is 8 KB 1 = Maximum Transmit Frame Size is 512B 2 = Maximum Transmit Frame Size is 64B 3 = Maximum Transmit Frame Size is 128B When <TransFrmSizExt> is 1: 0 = Maximum Transmit Frame Size is 256B 1 = Maximum Transmit Frame Size is 1 KB 2 = Maximum Transmit Frame Size is 2 KB 3 = Maximum Transmit Frame Size is 4 KB
13	LBStartRd	RW 0x0	Loopback Start BIST-DW1 Register is used as User-specified test pattern low DWORD. BIST-DW2 Register is used as User-specified test pattern high DWORD. These registers must be initiated before this bit is set. 0 = Disable 1 = Enable
12:9	LBPattern	RW 0x0	PHY Loopback pattern
8	LBEnable	RW 0x0	PHY Loopback Enable This bit enables SATA near-end PHY loopback. 0 = Disable 1 = Enable
7:4	Reserved	RSVD 0x0	Reserved
3:2	Reserved	RW 0x0	Reserved
1	TransFrmSizExt	RW 0x0	Maximum Transmit Frame Size Extended See field <TransFrmSiz> [15:14].
0	MBistEn	RW 0x0	Memory BIST Enable Start Memory BIST test in MBIST mode. 0 = Disable: Memory BIST test disabled. 1 = Enable: Memory BIST test enabled.

Table 980: Serial-ATA Interface Status Register (n=0–1)

Offset: Port0: 0x000A234C, Port1: 0x000A434C

Offset Formula: 0x000A234C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31	N	RO 0x0	Set to 1 when the Set Devices Bits FIS is received with the Notification (N) bit set to 1. **NOTE:** This bit is cleared when the <ClearStatus> bit[24] in the Serial-ATA Interface Control Register is set to 1). **NOTE:**
30	RxBIST	RO 0x0	Set to 1 when BIST FIS is received **NOTE:** This bit is cleared when the <ClearStatus> bit[24] in the Serial-ATA Interface Control Register is set to 1. **NOTE:**
29	Reserved	RSVD 0x0	Reserved
28:24	TransFsmSts	RO 0x0	Transport Layer FSM status 0x00 = Transport layer is idle. 0x00-0x1F = Transport layer is not idle.
23	LinkDown	RO 0x1	SATA communication is not ready. Primitives or FISs are not able to be transmitted or received. 0 = Ready: Link is ready. 1 = Not Ready: Link is not ready.
22	PlugIn	RO 0x0	Cable plug-in indicator and device presence indication. This signal becomes invalid when the core is in SATA Power Management modes. This indicator is also reflected in the <X> bit in SError Register. 0 = No detection: Device presence not detected. 1 = Detection: Device presence detected.
21	TxHdAct	RO 0x0	Tx Header Active This bit indicates if the transport Tx Header FSM is active. 0 = Idle 1 = Active
20	RxHdAct	RO 0x0	Rx Header Active This bit indicates if the transport Rx Header FSM is active. 0 = Idle 1 = Active
19	PIOAct	RO 0x0	PIO Active This bit indicates if the transport PIO FSM is active. 0 = Idle 1 = Active

Table 980: Serial-ATA Interface Status Register (n=0–1) (Continued)

Offset: Port0: 0x000A234C, Port1: 0x000A434C

Offset Formula: 0x000A234C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
18	DMAAct	RO 0x0	DMA Active This bit indicates if the transport DMA FSM is active. 0 = Idle 1 = Active
17	LBPass	RO 0x0	Near-end Loopback Pass This bit indicates if the Near-end Loopback test passed. 0 = Fail 1 = Pass
16	AbortCommand	RO 0x0	Abort Command This bit indicates if the transport has aborted a command as a response to collision with incoming FIS. **NOTE:** This bit is cleared when the <ClearStatus> bit[24] in the Serial-ATA Interface Control Register is set to 1. 0 = Not aborted: Command was not aborted. 1 = Abort: Command was aborted.
15	MBistFail	RO 0x0	Memory BIST Fail This bit indicates if the memory BIST test passed. It is valid when bit <MBistRdy> of this register is set. 0 = Pass 1 = Fail
14	MBistRdy	RO 0x1	Memory BIST Ready This bit indicates when the memory BIST test is completed. 0 = Incomplete: Memory BIST test is not completed. 1 = Complete: Memory BIST test is completed.
13	VendorUqErr	RO 0x0	Vendor Unique FIS Transmission Error This bit indicates if the Vendor Unique FIS transmission has completed successfully. This bit is valid when bit <VendorUqDn> of this register is set. 0 = Completed: Vendor Unique FIS transmission has completed successfully. 1 = Error: Vendor Unique FIS transmission has completed with error.
12	VendorUqDn	RO 0x0	Vendor Unique FIS Transmission Done This bit is set when the Vendor Unique FIS transmission has completed. 0 = Incomplete: Vendor Unique FIS transmission has not completed. 1 = Complete: Vendor Unique FIS transmission has completed.

Table 980: Serial-ATA Interface Status Register (n=0–1) (Continued)

Offset: Port0: 0x000A234C, Port1: 0x000A434C

Offset Formula: 0x000A234C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
11:8	PMportRx	RO 0x0	Port Multiplier Received This field specifies the Port Multiplier (bits [11:8] in DW0 of the FIS header) of the last received FIS. It also specifies the Port Multiplier (bits [11:8] in DW0 of the FIS header) of the next Data - Host to Device transmit FISs.
7:0	FISTypeRx	RO 0x0	FIS Type Received This field specifies the FIS Type of the last received FIS.

Table 981: Vendor Unique Register (n=0–1)

Offset: Port0: 0x000A235C, Port1: 0x000A435C

Offset Formula: 0x000A235C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	VendorUqDw	RW 0x0	Vendor Unique DWORD The data written to this register is transmitted as a vendor unique FIS. This Data includes the FIS header as well as the payload.

Table 982: FIS Configuration Register (n=0–1)

Offset: Port0: 0x000A2360, Port1: 0x000A4360

Offset Formula: 0x000A2360+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:18	Reserved	RSVD 0x0	Reserved
17	FISUnrecTypeCont	RW 0x0	When this bit is set, the transport layer state machine ignores incoming FIS with unrecognized FIS type. 0 = Error state: When an unrecognized FIS type is received, the transport layer goes into error state and asserts a protocol error. 1 = No error state: When an unrecognized FIS type is received, the transport layer does not go into error state and does not assert a protocol error.
16	FISDMAActiveSyncResp	RW 0x0	This bit identifies whether the transport layer responses with a single SYNC primitive after the DMA activates FIS reception. 0 = Normal: Normal response 1 = SYNC primitive: Response with single SYNC primitive. Must be set to 1 when bit <eEDMAFBS> in EDMA Configuration Register is set to 1.

Table 982: FIS Configuration Register (n=0–1) (Continued)

Offset: Port0: 0x000A2360, Port1: 0x000A4360

Offset Formula: 0x000A2360+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
15:8	FISWait4HostRdyEn	RW 0x0	This field identifies whether the transport layer waits for the upper layer (host) to acknowledge reception of the current FIS before enabling the link layer to response with R_RDY for the next FIS. The function of each bit in this field is: <FISWait4HostRdyEn>[0]: Register - Device to Host FIS with <ERR> or <DF> bit set to 1. <FISWait4HostRdyEn>[1]: SDB FIS is received with <N> bit set to 1. <FISWait4HostRdyEn>[2]: SDB FIS is received with <ERR> bit set to 1. <FISWait4HostRdyEn>[3]: BIST activate FIS <FISWait4HostRdyEn>[4]: PIO Setup FIS <FISWait4HostRdyEn>[5]: Data FIS with Link error <FISWait4HostRdyEn>[6]: Unrecognized FIS type <FISWait4HostRdyEn>[7]: Any FIS 0 = No wait: Do not wait for host ready. 1 = Wait: Wait for host ready.
7:0	FISWait4RdyEn	RW 0x0	This field identifies whether the transport layer waits for the upper layer (EDMA or host) to acknowledge reception of the current FIS before enabling the link layer to response with R_RDY for the next FIS. When bit <eEnEDMA> is set to 1 in EDMA Command Register, the transport layer ignores the value of bits [4:0] of this field and assume a value of 0x1F, i.e., it waits for the EDMA to acknowledge reception of the current FIS before enabling the link layer to response with R_RDY for the next FIS. The function of each bit in this field is: <FISWait4RdyEn>[0]: Register - Device to Host FIS <FISWait4RdyEn>[1]: SDB FIS is received with <N> bit cleared to 0. <FISWait4RdyEn>[2]: DMA Activate FIS <FISWait4RdyEn>[3]: DMA Setup FIS <FISWait4RdyEn>[4]: Data FIS first DW <FISWait4RdyEn>[5]: Data FIS entire FIS <FISWait4RdyEn>[7:6]: Reserved 0 = No wait: Do not wait for host ready. 1 = Wait: Wait for host ready.

Table 983: FIS Interrupt Cause Register (n=0–1)

Offset: Port0: 0x000A2364, Port1: 0x000A4364

Offset Formula: 0x000A2364+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
A corresponding cause bit is set every time that an interrupt occurs. A write of 0 clears the bit. A write of 1 has no effect.			
31:26	Reserved	RO 0x0	Reserved

Table 983: FIS Interrupt Cause Register (n=0–1) (Continued)

Offset: Port0: 0x000A2364, Port1: 0x000A4364

Offset Formula: 0x000A2364+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
25	FISTxErr	RW 0x0	This bit is valid when bit[24] <FISTxDone> is set to 1. This bit is set to 1 when the FIS transmission is done, bit[24] <FISTxDone> is set to 1 and one of the following occurs: FIS transmission is aborted due to collision with the received FIS. FIS transmission is completed with R_ERR. 0 = Complete: Frame transmission was completed successful with R_OK. 1 = Incomplete: Frame transmission was not completed successful.
24	FISTxDone	RW 0x0	This bit is set to 1 when the FIS transmission is done, either aborted or completed with R_OK or R_ERR. 0 = Continues: Frame transmission continues. 1 = Completed: Frame transmission completed, either with R_ERR or R_ OK, or frame transmission aborted.
23:16	Reserved	RW 0x0	Reserved.
15:8	FISWait4HostRdy	RW 0x0	This field indicates the reception of the following FISs. <FISWait4HostRdy>[0]: Register Device to Host FIS with <ERR> bit set to 1. <FISWait4HostRdy>[1]: SDB FIS is received with <N> bit set to 1. <FISWait4HostRdy>[2]: SDB FIS is received with <ERR> bit set to 1. <FISWait4HostRdy>[3]: BIST activates FIS <FISWait4HostRdy>[4]: PIO Setup FIS <FISWait4HostRdy>[5]: Data FIS with Link error <FISWait4HostRdy>[6]: Unrecognized FIS type <FISWait4HostRdy>[7]: Any FIS For any FIS other than data FIS, the corresponding bit is set when the FIS is received from the link layer without an error, that is, FIS Dword0 Register through FIS Dword6 Register are updated with the content of the FIS up to the FIS length. For the data FIS, the corresponding bit is set when the entire FIS is received from the link layer, if a link error occurs. Only the FIS Dword0 Register is updated with the content of the FIS. FIS Dword1 Register through FIS Dword6 Register are not updated. If the non-data FIS length is shorter than 7 Dwords, only the relevant registers are updated with the content of the FIS. If the non-data FIS length is longer than 7 Dwords, FIS Dword0 Register through FIS Dword6 Register are updated with the content of the FIS. The rest of FIS is dropped. When at least one bit in this field is set and the corresponding bit in the <FISWait4HostRdyEn> field of the FIS Configuration Register is enabled (set to 1), the transport layer prevents assertion of the primitive R_RDY and the reception of the next FIS. 0 = No interrupt: No interrupt indication. 1 = Interrupt: Corresponding interrupt occurs.

Table 983: FIS Interrupt Cause Register (n=0–1) (Continued)

Offset: Port0: 0x000A2364, Port1: 0x000A4364

Offset Formula: 0x000A2364+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
7:0	FISWait4Rdy	RW 0x0	This field indicates the reception of the following FISs. <FISWait4Rdy>[0]: Register Device to Host FIS <FISWait4Rdy>[1]: SDB FIS is received with <N> bit cleared to 0. <FISWait4Rdy>[2]: DMA Activate FIS <FISWait4Rdy>[3]: DMA Setup FIS <FISWait4Rdy>[4]: Data FIS first Dword <FISWait4Rdy>[5]: Data FIS entire FIS <FISWait4Rdy>[7:6]: Reserved For any FIS other than data FIS, the corresponding bit is set when the entire FIS is received from the link layer without an error, that is, FIS Dword0 Register through FIS Dword6 Register are updated with the content of the FIS. If the non-data FIS length is shorter than 7 Dwords, only the relevant registers are updated with the content of the FIS. If the non-data FIS length is longer than 7 Dwords, FIS Dword0 Register through FIS Dword6 Register are updated with the content of the FIS. The rest of FIS is dropped. For data FIS, <FISWait4Rdy> [4] is set when the first Dword of the FIS is received from the link layer and <FISWait4Rdy> [5] is set when the entire FIS is received from the link layer, regardless of whether or not an error occurred. Only FIS Dword0 Register is updated with the content of the FIS. FIS Dword1 Register through FIS Dword6 Register are not updated. When at least one bit in this field is set and the corresponding bit in the <FISWait4RdyEn> field of the FIS Configuration Register) is enabled (set to 1), the transport layer prevents assertion of the primitive R_RDY and the reception of the next FIS. 0 = No interrupt: No interrupt indication. 1 = Interrupt: Corresponding interrupt occurs.

Table 984: FIS Interrupt Mask Register (n=0–1)

Offset: Port0: 0x000A2368, Port1: 0x000A4368

Offset Formula: 0x000A2368+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:26	Reserved	RO 0x0	Reserved
25:0	FISIntMask	RW 0x0000A00	FIS Interrupt Error Mask Bits Each of these bits mask the corresponding bit in the FIS Interrupt Cause Register. If a bit in the FIS Interrupt Cause Register is set and the corresponding bit in this register is set to 1, bit <eTransInt> in EDMA Interrupt Error Cause Register is also set to 1. 0 = Mask 1 = Do not mask

Table 985: FIS Dword0 Register (n=0–1)

Offset: Port0: 0x000A2370, Port1: 0x000A4370

Offset Formula: 0x000A2370+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	RxFISDW0	RO 0x0	This field contains Dword 0 of the incoming data or non-data FIS.

Table 986: FIS Dword1 Register (n=0–1)

Offset: Port0: 0x000A2374, Port1: 0x000A4374

Offset Formula: 0x000A2374+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	RxFISDW1	RO 0x0	This field contains Dword 1 of the incoming non-data FIS.

Table 987: FIS Dword2 Register (n=0–1)

Offset: Port0: 0x000A2378, Port1: 0x000A4378

Offset Formula: 0x000A2378+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	RxFISDW2	RO 0x0	This field contains Dword 2 of the incoming non-data FIS.

Table 988: FIS Dword3 Register (n=0–1)

Offset: Port0: 0x000A237C, Port1: 0x000A437C

Offset Formula: 0x000A237C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	RxFISDW3	RO 0x0	This field contains Dword 3 of the incoming non-data FIS.

Table 989: FIS Dword4 Register (n=0–1)

Offset: Port0: 0x000A2380, Port1: 0x000A4380

Offset Formula: 0x000A2380+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	RxFISDW4	RO 0x0	This field contains Dword 4 of the incoming non-data FIS.

Table 990: FIS Dword5 Register (n=0–1)

Offset: Port0: 0x000A2384, Port1: 0x000A4384

Offset Formula: 0x000A2384+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	RxFISDW5	RO 0x0	This field contains Dword 5 of the incoming non-data FIS.

Table 991: FIS Dword6 Register (n=0–1)

Offset: Port0: 0x000A2388, Port1: 0x000A4388

Offset Formula: 0x000A2388+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	RxFISDW6	RO 0x0	This field contains Dword 6 of the incoming non-data FIS.

Table 992: PHY Configuration Register (n=0–1)

Offset: Port0: 0x000A23A0, Port1: 0x000A43A0

Offset Formula: 0x000A23A0+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:9	Reserved	RW 0x1	Reserved
8:4	VTH_DISCON	RW 0x7	Host disconnect’s voltage level threshold (25 mVp-p step) 0: 25 mVp-p 1: 50 mVp-p 2: 75 mVp-p 3: 100 mVp-p 4: 125 mVp-p 5: 150 mVp-p 6: 175 mVp-p 7: 200 mVp-p 8: 225 mVp-p 9: 250 mVp-p 10: 275 mVp-p 11: 300 mVp-p 12: 325 mVp-p 13: 350 mVp-p 14: 375 mVp-p 15: 400 mVp-p 31: 800mVp-p

Table 992: PHY Configuration Register (n=0–1) (Continued)

Offset: Port0: 0x000A23A0, Port1: 0x000A43A0

Offset Formula: 0x000A23A0+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
3:0	SQ_THRESHOLD	RW 0x3	Squelch Detector Threshold 0 = 50 mVp-p 1 = 70 mVp-p 2 = 90 mVp-p 3 = 110 mVp-p 4 = 130 mVp-p 5 = 150 mVp-p 6 = 170 mVp-p 7 = 190 mVp-p 8 = 50 mVp-p 9 = 80 mVp-p 10 = 110 mVp-p 11 = 140 mVp-p 12 = 170 mVp-p 13 = 200 mVp-p 14 = 230 mVp-p 15 = 260 mVp-p

Table 993: PHYTCTL Register (n=0–1)

Offset: Port0: 0x000A23A4, Port1: 0x000A43A4

Offset Formula: 0x000A23A4+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
PHY-LINK Test Control			
31:16	Reserved	RSVD 0x102	Reserved
15:12	LINK_TESTSEL	RW 0x0	Link Test Port Select (MONITORSEL)
11:6	ND	RW 0x0	Not Defined
5:0	TEST_ANA	RW 0x10	Analog Test Port Select

Table 994: PHY Mode 10 Register (n=0–1)

Offset: Port0: 0x000A23A8, Port1: 0x000A43A8

Offset Formula: 0x000A23A8+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
TX Clock Frequency Offset Manual Update			
31:28	ND	RW 0x0	Not Defined

Table 994: PHY Mode 10 Register (n=0–1) (Continued)

Offset: Port0: 0x000A23A8, Port1: 0x000A43A8

Offset Formula: 0x000A23A8+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
27	PEAK_READY	RO 0x0	When set, the NXT_AVG value is valid (this signal will toggle per update window as defined in AVG_WINDOW[1:0])
26	Reserved	RO 0x0	Reserved
25:16	NXT_AVG	RO 0x0	Tx Frequency Offset
15:14	ND	RW 0x0	Not Defined
13	Reserved	RW 0x0	Reserved
12	AVG_READ_EN	RW 0x0	When toggled, it triggers the TX Frequency Offset calculation. This can only be triggered once after power-up. To retrigger, it requires a PHY RESET.
11	AVG_READY	RO 0x0	When set, the AVG field is valid
10	Reserved	RSVD 0x0	Reserved
9:0	AVG	RO 0x0	Tx Frequency Offset

Table 995: PHY Mode 12 Register (n=0–1)

Offset: Port0: 0x000A23B0, Port1: 0x000A43B0

Offset Formula: 0x000A23B0+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	Reserved	RW 0x0	Reserved

A.11.4 EDMA

Table 996: EDMA Configuration Register (n=0–1)

Offset: Port0: 0x000A2000, Port1: 0x000A4000

Offset Formula: 0x000A2000+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:27	Reserved	RSVD 0x0	Reserved

Table 996: EDMA Configuration Register (n=0–1) (Continued)

Offset: Port0: 0x000A2000, Port1: 0x000A4000

Offset Formula: 0x000A2000+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
26	eDMAFBS	RW 0x0	Port Multiplier, FIS Based Switching mode. When cleared to 0, the Basic DMA continues the data transfer until the end of the PRD table. Then, it clears bit <BasicDMAActive>, clears bit <BasicDMAPaused>, and halts. These two bits are in the Basic DMA Status Register. When this bit is set to 1, the Basic DMA stops on the FIS boundary. During a write command, the Basic DMA: 1. Completes the 8-KB FIS transmission to the drive (Max frame transmission size can be change by the <TransFrmSiz> field in the Serial-ATA Interface Test Control Register). 2. Clears bit <BasicDMAActive>. 3. Sets bit <BasicDMAPaused> if the command was not completed. 4. Halts. During a read command, the Basic DMA: 1. Completes the data transfer associates with a single FIS reception. 2. If bit <eEarlyCompletionEn> is set it waits for data visible in the system memory. 3. Clears bit <BasicDMAActive>. 4. Sets bit <BasicDMAPaused> if command was not completed. 5. Halts. **NOTE:** This bit must be set to 1 when FIS based switching mode is used. See "Port Multiplier--FIS-Based Switching". **NOTE:** For the best Performance when a single drive is connected, clear this bit to 0. When bit[16] <eEDMAFBS> in this register is set to 1, the Basic DMA ignores the value of this bit, and a value of 1 is assumed. 0 = Disable: FIS based Switching mode disabled. Basic DMA stops on a command (PRD) boundary. 1 = Enable: FIS Based Switching mode enabled. Basic DMA stops on a FIS boundary.
25	Reserved	RW 0x0	Reserved
24	ResumeDis	RW 0x0	When set to 1, the EDMA never sets bit <ContFromPrev in the Basic DMA Command Register. When cleared to 0, the EDMA sets bit <ContFromPrev> in the Basic DMA Command Register whenever possible.
23	eMaskRxPM	RW 0x0	This bit masks the PM field in the incoming FISs 0 = Mask: Mask the PM field in incoming FISs 1 = No mask: Do not mask

Table 996: EDMA Configuration Register (n=0–1) (Continued)

Offset: Port0: 0x000A2000, Port1: 0x000A4000

Offset Formula: 0x000A2000+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
22	eHostQueueCacheEn	RW 0x0	EDMA Host Queue Cache Enable When set to 1, the EDMA is capable of storing up to four commands internally, before sending the commands to the drive. This bit enables the EDMA to send multiple commands across multiple drives much faster. It also enables the EDMA to send commands to any available drive, while other drives are busy executing previous commands. 0 = Disable: Host Queue Cache is disabled. EDMA stores a single command internally. Only when the command is sent to the drive, it fetches a new command from the CRQB. 1 = Enable: Host Queue Cache is enabled.
21:20	Reserved	RW 0x0	Reserved
19	Reserved	RO 0x0	Reserved
18	eEarlyCompletionEn	RW 0x0	This bit enables Basic DMA Early completion. When this bit is set to 1, Basic DMA Early completion is enabled. The DMA completes operation when all data is transferred from the disk to the memory, and it updates the following bits immediately, instead of waiting for this data to be visible in the system memory: Bit <BasicDMAActive> in the Basic DMA Status Register is cleared to 0. When EDMA is disabled, Bits[3:0] <DMAxDone> in the SATAHC Interrupt Cause Register is set to 1. Regardless to this bit value, when EDMA is enabled, the EDMA ensures all data is visible in system memory and only then it sets bit <SaCrpb0Done> in the EDMA Interrupt Error Cause Register. When EDMA controls the Basic DMA, it is recommended to set this bit to 1. When the CPU controls the Basic DMA directly, it is recommended to clear this bit to 0. 0 = Disable: Early DMA completion disabled 1 = Enable: Early DMA completion enabled
17	eCutThroughEn	RW 0x0	This bit enables the Basic DMA cut-though operation when writing data to the drive. When the maximum read burst size initiated by the SATAHC-DMA to the memory in a read is limited to 128B (bits [8] <eRdBSz> and [11] <eRdBSzExt> in this register are both 0), the value of the <eCutThroughEn> bit is ignored and the cut-though operation is disabled. 0 = Store/forward 1 = Cut through
16	eEDMAFBS	RW 0x0	EDMA FIS (Frame Information Structure) Based Switching When cleared to 0, the EDMA issue a new command only when the previous command was completed or released by the drive. When set to 1, EDMA issues a new command to the drive as soon as the target device is available, regardless to the status of all of the other devices.

Copyright © 2014 Marvell

Table 996: EDMA Configuration Register (n=0–1) (Continued)

Offset: Port0: 0x000A2000, Port1: 0x000A4000

Offset Formula: 0x000A2000+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
15:14	Reserved	RW 0x0	Reserved
13	eWrBufferLen	RO 0x0	EDMA Write Buffers Length This bit defines the length of the write buffers. **NOTE:** This field must be set to 0. 0 = 256B: Write buffer is 256B 1 = Reserved
12	Reserved	RW 0x1	Reserved This field must be set to 0x1
11	eRdBSzExt	RO 0x0	EDMA Burst Size Ext. This bit sets the maximum burst size initiated by the SATAHC-DMA to the memory. This bit is related to the read operation. **NOTE:** This field must be set to 0. 0 = Configured: EDMA Burst size is configured according to bit <eRdBSz> 1 = Reserved
10	Reserved	RW 0x0	Reserved
9	eQue	RW 0x0	EDMA Queued When EDMA uses queued DMA commands, this bit must be set to 1. 0 = Non-Queued: Non-Queued DMA commands are in use 1 = Queued: Only Queued DMA commands are in use
8	eRdBSz	RW 0x0	EDMA Burst Size This bit sets the maximum burst size initiated by the SATAHC-DMA to the memory. This bit is related to read operation. **NOTE:** This field must be set to 0. 0 = 128B 1 = Reserved
7:6	Reserved	RW 0x0	Reserved
5	eSATANatvCmdQue	RW 0x0	EDMA SATA Native Command Queuing. When EDMA uses SATA Native Command Queuing, this bit must be set to 1. When this bit is set, bit[9] <eQue> is ignored. 0 = Disable: Native Command Queuing is not in use. 1 = Enable: Native Command Queuing is in use.
4:0	Reserved	RW 0x1F	Reserved

Table 997: EDMA Interrupt Error Cause Register (n=0–1)

Offset: Port0: 0x000A2008, Port1: 0x000A4008

Offset Formula: 0x000A2008+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: A corresponding cause bit is set every time that an interrupt occurs. A write of 0 clears the bit. A write of 1 has no effect.			
31	TransProtErr	RW0C 0x0	Transport Protocol Error This bit is set when a control FIS is received with a non transient transport protocol error. This bit is set when a: - Link error indication is set during reception of a DMA/PIO data frame or control frame (i.e., when the Link Layer is reset to Idle state by the reception of SYNC primitives from the device). During control frame reception, bit [15] in this register is also set to 1. During data frame reception, bit [19] in this register is also set to 1. - Control frame is too short or too long. - Incorrect FIS type -Illegal transfer count on a PIO transaction See "Serial-ATA II Transport Layer Protocol Non Transient Errors".
30:26	LinkDataTxErr	RW0C 0x0	Link Data Transmit Error This field indicates when a data FIS is transmitted with errors. Bit [0] of this field (i.e., bit [26] of this register) is set to 1 when a SATA CRC error occurs. Bit [1] of this field is set to 1 when an internal FIFO error occurs. Bit [2] of this field is set to 1 when the Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [3] of this field is set to 1 when the Link Layer accepts a DMAT primitive from the device. Bit [4] of this field is set to 1 to indicate when FIS transmission is aborted due to collision with Rx traffic. See "Serial-ATA II Link Layer Error During Transmission of a Data Frame". .
25:21	LinkCtlTxErr	RW0C 0x0	Link Control Transmit Error This field indicates when a control FIS is transmitted with errors. Bit [0] of this field (i.e., bit [21] of this register) is set to 1 when a SATA CRC error occurs. Bit [1] of this field is set to 1 when an internal FIFO error occurs. Bit [2] of this field is set to 1 when the Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [3] of this field is set to 1 when the Link Layer accepts a DMAT primitive from the device. Bit [4] of this field is set to 1 to indicate when FIS transmission is aborted due to collision with Rx traffic. See Section "Serial-ATA II Link Layer Error During Transmission of a Control Frame".

Table 997: EDMA Interrupt Error Cause Register (n=0–1) (Continued)

Offset: Port0: 0x000A2008, Port1: 0x000A4008

Offset Formula: 0x000A2008+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
20:17	LinkDataRxErr	RW0C 0x0	Link Data Receive Error This field indicates when a data FIS is received with errors. Bit [0] of this field (i.e., bit [17] of this register) is set to 1 when a SATA CRC error occurs. Bit [1] of this field is set to 1 when an internal FIFO error occurs. Bit [2 of this field] is set to 1 when the Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [3] of this field is set to 1 when Link state errors, coding errors, or running disparity errors occur during FIS reception. See "Serial-ATA II Link Layer Error During Reception of a Data Frame".
16:13	LinkCtlRxErr	RW0C 0x0	Link Control Receive Error This field indicates when a control FIS is received with errors. Bit [0] of this field (i.e., bit [13] of this register) is set to 1 when a SATA CRC error occurs. Bit [1] of this field is set to 1 when an internal FIFO error occurs. Bit [2] of this field is set to 1 when the Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [3] of this field is set to 1 when Link state errors, coding errors, or running disparity errors occur during FIS reception. See "Serial-ATA II Link Layer Error During Reception of a Control Frame".
12	eIORdyErr	RW0C 0x0	EDMA IORdy Error IORdy timeout occurred. See "EDMA IORdy Timeout Register". **NOTE:** This bit is only set when the EDMA is disabled.
11	ePrtIntErr	RW0C 0x0	EDMA Internal RAM Parity Error 0 = NoError 1 = RamError: Internal error was found EDMA ram.
10	ePrtCRPBErr	RW0C 0x0	EDMA CRPB Parity Error 0 = NoError 1 = CrpbError: Internal parity errorwas found when updating new entry in CRPB.
9	ePrtCRQBErr	RW0C 0x0	EDMA CRQB Parity Error 0 = NoError 1 = CrqbError: Internal parity errorwas found when fetching new command (CRQB)

Table 997: EDMA Interrupt Error Cause Register (n=0–1) (Continued)

Offset: Port0: 0x000A2008, Port1: 0x000A4008

Offset Formula: 0x000A2008+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
8	eTransInt	RW0C 0x0	Transport Layer Interrupt This bit is set when at least one bit in the FIS Interrupt Cause Register is set to 1 and the corresponding bit in the FIS Interrupt Mask Register is set to 1 (enabled). **NOTE:** This bit should be cleared only after clearing the FIS Interrupt Cause Register. 0 = No wait: Transport layer does not wait for host 1 = Wait: Transport layer waits for host
7	eSelfDis	RW0C 0x0	EDMA Self Disable This bit is set to 1 if the EDMA encounters error and clears bit <eEnEDMA>. 0 = No self disable: EDMA was not self disabled 1 = Self disabled: EDMA was self disabled
6	Reserved	RW0C 0x0	Reserved
5	SerrInt	RW0C 0x0	This bit is set to 1 when at least one bit in SError Register is set to 1 and the corresponding bit in SError Interrupt Mask Register is enabled. See "SError Register Errors". **NOTE:** This bit should be cleared only after clearing the SError Register. 0 = Not set: A bit in SError Register was not set to 1. 1 = Set: A bit in SError Register was set to 1.
4	eDevCon	RW0C 0x0	EDMA Device Connected This bit is set to 1 if the device was disconnected and is connected again. 0 = Not reconnected: Device was not reconnected. 1 = Reconnected: Device was reconnected.
3	eDevDis	RW0C 0x0	EDMA Device Disconnect This bit is set to 1 if the device is disconnected. **NOTE:** After DevDis interrupt, device hard reset is required. Set the <eAtaRst>bit[2] in the EDMA Command Register to 1. See "Device Disconnect". 0 = Not disconnected: Device was not disconnected. 1 = Disconnected: Device was disconnected.
2	eDevErr	RW0C 0x0	EDMA Device Error This bit is set to 1 when: 1. The EDMA is enabled (i.e., bit <eEnEDMA> in the EDMA Command Register is set to 1) and 2. Register - Device to Host FIS (Frame Information Structure) or Set Device Bits FIS is received with bit <ERR> set to 1. See "Device Error Indications".

Table 997: EDMA Interrupt Error Cause Register (n=0–1) (Continued)

Offset: Port0: 0x000A2008, Port1: 0x000A4008

Offset Formula: 0x000A2008+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
1	ePrtPrdErr	RW0C 0x0	EDMA ATA UDMA PRD Parity Error 0 = NoError 1 = PrdError: Internal error was found when fetching a new PRD.
0	ePrtDataErr	RW0C 0x0	EDMA ATA UDMA Data Parity Error 0 = NoError 1 = DmaError: Internal error was found during ATA UDMA data transfer.

Table 998: EDMA Interrupt Error Mask Register (n=0–1)

Offset: Port0: 0x000A200C, Port1: 0x000A400C

Offset Formula: 0x000A200C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	eIntErrMsk	RW 0x0	EDMA Interrupt Error Mask Bits Each of these bits checks the corresponding bit in the EDMA Interrupt Error Cause Register, and if these bits are set (0), they mask the interrupt.

Table 999: EDMA Request Queue Base Address High Register (n=0–1)

Offset: Port0: 0x000A2010, Port1: 0x000A4010

Offset Formula: 0x000A2010+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	eRqQBA[63:32]	RW 0x0	The EDMA Request Queue Base Address corresponds to bits [63:32].

Table 1000:EDMA Request Queue In-Pointer Register (n=0–1)

Offset: Port0: 0x000A2014, Port1: 0x000A4014

Offset Formula: 0x000A2014+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:10	eRqQBA[31:10]	RW 0x0	EDMA Request Queue Base Address corresponds to bits [31:10].
9:5	eRqQIP	RW 0x0	EDMA Request Queue In-Pointer The EDMA request queue in-pointer is written/increment by the system driver to indicate that a new CRQB has been placed on the request queue. The EDMA compares the EDMA Request Queue In-Pointer to the EDMA Request Queue Out-Pointer to determine when the request queue is empty. If there are CRQBs in the request queue, then, when the EDMA is ready, it process the commands.

Table 1000:EDMA Request Queue In-Pointer Register (n=0–1) (Continued)

Offset: Port0: 0x000A2014, Port1: 0x000A4014

Offset Formula: 0x000A2014+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
4:0	Reserved	RSVD 0x0	Reserved

Table 1001:EDMA Request Queue Out-Pointer Register (n=0–1)

Offset: Port0: 0x000A2018, Port1: 0x000A4018

Offset Formula: 0x000A2018+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:12	Reserved	RSVD 0x0	Reserved
11:10	Reserved	RW 0x0	Reserved
9:5	eRqQOP	RW 0x0	EDMA Request Queue Out Pointer The EDMA request queue out-pointer is updated/increment by the EDMA each time a CRQB is copied from the command queue into the EDMA internal memory. The system driver reads this register to determine if the request queue is full.
4:0	Reserved	RSVD 0x0	Reserved

Table 1002:EDMA Response Queue Base Address High Register (n=0–1)

Offset: Port0: 0x000A201C, Port1: 0x000A401C

Offset Formula: 0x000A201C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	eRpQBA[63:32]	RW 0x0	The EDMA Response Queue Base Address corresponds to bits [63:32].

Table 1003:EDMA Response Queue In-Pointer Register (n=0–1)

Offset: Port0: 0x000A2020, Port1: 0x000A4020

Offset Formula: 0x000A2020+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x0	Reserved
9:8	Reserved	RW 0x0	Reserved

Table 1003:EDMA Response Queue In-Pointer Register (n=0–1) (Continued)

Offset: Port0: 0x000A2020, Port1: 0x000A4020

Offset Formula: 0x000A2020+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
7:3	eRpQIP	RW 0x0	EDMA Response Queue In-Pointer The EDMA response queue in-pointer is updated/increment by the EDMA each time a command execution is completed and a new CRPB is moved from the EDMA internal memory to the response queue by the EDMA. The system driver compares the EDMA Response Queue In-Pointer with the EDMA Response Queue Out-Pointer to determine if there is a CRPB in the response queue that needs processing.
2:0	Reserved	RSVD 0x0	Reserved

Table 1004:EDMA Response Queue Out-Pointer Register (n=0–1)

Offset: Port0: 0x000A2024, Port1: 0x000A4024

Offset Formula: 0x000A2024+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:8	eRPQBA[31:8]	RW 0x0	The EDMA Response Queue base address corresponds to bits [31:8]
7:3	eRPQOP	RW 0x0	EDMA Response Queue Out-Pointer The EDMA response queue out-pointer is updated/increment by the system driver after the driver completes processing the CRPB pointed to by this register. The EDMA compares the EDMA Response Queue Out-Pointer to the EDMA Response Queue In-Pointer to determine if the response queue is full.
2:0	Reserved	RSVD 0x0	Reserved

Table 1005:EDMA Command Register (n=0–1)

Offset: Port0: 0x000A2028, Port1: 0x000A4028

Offset Formula: 0x000A2028+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4	eEDMAFrz	RW 0x0	EDMA Freeze When this bit is set, EDMA does not pull new commands from CRQB. If commands are pending in EDMA cache, these commands are sent to the drive regardless to this bit value. 0 = Pull: Pull new commands from CRQB and insert them into the drive. 1 = Do not pull: Do not pull new commands from CRQB.
3	Reserved	RW 0x0	Write only 0x0.

Table 1005:EDMA Command Register (n=0–1) (Continued)

Offset: Port0: 0x000A2028, Port1: 0x000A4028

Offset Formula: 0x000A2028+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
2	eAtaRst	RW 0x0	ATA Device Reset When this bit is set to 1, Serial-ATA transport, link, and physical layers are reset. All Serial-ATA Interface Registers (see "Serial-ATA Interface Registers Map") are reset, except the Serial-ATA Interface Configuration Register, and the OOB COMRESET signal is sent to the device, after configuring the <DET> field in the SControl Register. Host must not read/write Serial-ATA Interface Registers. If Host initiates a read/write to Serial-ATA Interface Registers when this bit is set, the transaction gets stuck until EDMA IORdy Timeout Register expires. - When this bit is set and EDMA is enabled (bit <eEnEDMA> in the EDMA Command Register is set to 1) the EDMA operation must be aborted: Set bit <eDsEDMA> in the EDMA Command Register to 1. - When this bit is set and the Basic DMA is enabled (bit <Start> in Basic DMA Command Register is set to 1) the Basic DMA operation must be aborted: Clear bit <Start> in Basic DMA Command Register to 0. 0 = Normal: Normal operation. 1 = Reset: Device reset.
1	eDsEDMA	SC 0x0	Disable EDMA This bit is self-negated. When this bit is set to 1, the EDMA aborts the current Command and then clears bit <eEnEDMA> (bit [0]). If the EDMA operation is aborted during command execution, the host software must set bit <eAtaRst> (bit [2]) to recover.
0	eEnEDMA	RW 0x0	Enable EDMA When this bit is set to 1, the EDMA is activated. All control registers, request queues, and response queues must be initialized before this bit is set to 1. The EDMA clears this bit when any of the following events occur: - Bit <eDsEDMA> (bit [1]) is set. - An error condition occurs and the corresponding bit in EDMA Halt Conditions Register is not masked. - Link is down and the corresponding bit in EDMA Halt Conditions Register is not masked . Writing 0 to this bit is ignored. 0 = Idle: The EDMA is idle. 1 = Active: The EDMA is active.

Copyright © 2014 Marvell

Table 1006:EDMA Status Register (n=0–1)

Offset: Port0: 0x000A2030, Port1: 0x000A4030

Offset Formula: 0x000A2030+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:22	Reserved	RSVD 0x0	Reserved
21:16	eIOId	RO 0x0	EDMA IO ID This field indicates the IO ID of the last command used by the EDMA. The EDMA updates this field when a new command is written to the device; or, When the device sends a completion FIS (NonQ, TCQ -- RegD2H with <REL> bit cleared and <DRQ> bit cleared. NCQ -- reception of SDB FIS); or, When the DMA is activated (after a reception of DMA Activate FIS or a reception of a DATA FIS).
15:8	eSTATE	RO 0x11	EDMA State These bits indicate the current state of the machine.
7	EDMAIdle	RO 0x0	This bit is set to 1 when all of the following conditions occur: - No commands are pending in EDMA request queue AND - All command EDMAs taken from the EDMA request queue were delivered to the drive AND - All command completions EDMA received from the drive were delivered to the host response queue AND - All commands sent to the drives were either completed or released AND - EDMA is in idle state. - EDMA is enabled. **NOTE:** This bit may be set when a command is still pending in the drive. **NOTE:**
6	eCacheEmpty	RO 0x1	Cache Empty This field indicates if EDMA cache is empty. 0 = Not empty 1 = Empty
5	eDevDir	RO 0x0	Device Direction The EDMA updates this field with field <cDIR> of the CRQB when a new command is written to the device, or when the tag is read from the device after a service command (TCQ), or when the tag is read from the device in DMA Setup FIS (NCQ). 0 = Mem to device: System memory to device 1 = Device to mem: Device to system memory
4:0	eDevQueTAG	RO 0x0	EDMA Tag This field indicates the tag of the last command used by the EDMA. The EDMA updates this field with field <cDeviceQueTag> of the CRQB: - When a new command is written to the device, OR - When the tag is read from the device after a service command (TCQ), or - When the tag is read from the device in DMA Setup (SU) FIS (NCQ).

Table 1007:EDMA IORdy Timeout Register (n=0–1)

Offset: Port0: 0x000A2034, Port1: 0x000A4034

Offset Formula: 0x000A2034+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	eIORdyTimeout	RW 0xBC	EDMA IORdy Signal Timeout Value If the SATAHC unit is cannot complete the transaction within the number of TCLK cycles set in this field, a timeout occurs, and the transaction is terminated. If the transaction is terminated as a result of the IORdy timeout, the <eIORdyErr> field is set. The Serial-ATA specification defines the IORdy time-out value to be 1250 ns. The default value (0xBC) is configured as if TCLK equals 150 MHz (6.666 ns). The default is calculated as 1250/6.666 ns = 188 (0xBC). When the TCLK is 200 MHz (5 ns), the value of this field should be 0xFA (1250/5 ns). Therefore, set this field to 0xFA. When the TCLK is 166 MHz (6 ns), the value of this field should be 0xD1 (1250/6 ns). Therefore, set this field to 0xD1. When clearing this field to 0, the timeout is ignored, and the transaction will never be aborted. When setting this field to non-zero, the timeout occurs after x* TCLK cycles.

Table 1008:EDMA Command Delay Threshold Register (n=0–1)

Offset: Port0: 0x000A2040, Port1: 0x000A4040

Offset Formula: 0x000A2040+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31	CMDDataoutDelayEn	RW 0x0	When this bit is set to 1, when the DMA completes write data transaction, the content of field [15:0] <CmdDelayThrshd> is written to a 16-bit counter. This counter is decrement every 16 clock cycles until it reaches 0. Then it stops. A new command is issue to the drive only when the value of this counter is 0 (i.e., a delay of <CmdDelayThrshd> *16 clock cycles is inserted after the data transaction from the system memory to the drive completes and before it issues a new command to the drive). 0 = Disabled: No delay is inserted before command retransmission. 1 = Enabled: A delay is inserted before command retransmission.
30:16	Reserved	RSVD 0x0	Reserved
15:0	CmdDelayThrshd	RW 0x0	This field indicates the length of delay to insert before sending a command to the drive when [31], <CMDDataoutDelayEn> is enabled.

Table 1009:EDMA Halt Conditions Register (n=0–1)

Offset: Port0: 0x000A2060, Port1: 0x000A4060

Offset Formula: 0x000A2060+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:0	eHaltMask	RW 0xFC1E0E1F	EDMA halts when any bit is set to 1 in the EDMA Interrupt Error Cause Register, and is not masked by the corresponding bit in this field. 0 = Mask 1 = Do not mask

Table 1010:EDMA NCQ0 Done/TCQ0 Outstanding Status Register (n=0–1)

Offset: Port0: 0x000A2094, Port1: 0x000A4094

Offset Formula: 0x000A2094+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
NOTE: When the EDMA is disabled, the fields in this register are Read Only.			
31:0	NCQDone_ TCQOutstand	RW 0x0	When in NCQ mode: NCQ Completion Status Each bit represents a completed NCQ command corresponding to the host command ID. When set the NCQ command corresponding to the host command ID has been completed but not yet updated in the host response queue in system memory [CRPB]. When in TCQ mode: TCQ Outstanding Commands Status Each bit represents an outstanding command corresponding to the host command ID. When set, the command was sent to the device but it has not yet completed and updated in the host response queue in system memory [CRPB]. EDMA sets the corresponding bit when sent a new command, EDMA clears the corresponding bit when the command is completed and updated in the host response queue in system memory [CRPB].

A.11.5 SATA Communication PHY

Table 1011:Power and PLL Control Register (n=0–1)

Offset: Port0: 0x000A2804, Port1: 0x000A4804

Offset Formula: 0x000A2804+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0xF8	Reserved
7:5	PHY_MODE	RW 0x3	PHY_MODE These bits control the mode selection. 0x0: SATA. 0x3: PCIe. 0x4: SERDES (GbE).

Table 1011:Power and PLL Control Register (n=0–1) (Continued)

Offset: Port0: 0x000A2804, Port1: 0x000A4804

Offset Formula: 0x000A2804+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
4:0	REF_FREF_SEL	RW 0x0	Reference Frequency Select. These bits indicate the reference clock speed. SATA 0: 20.0 MHz 1: 25.0 MHz SerDes (GbE) 0x0: 20.0 MHz 0x1: 25.0 MHz PCIe PHY 0x0: 100 MHz

Table 1012:PHY Test Control 0 Register (n=0–1)

Offset: Port0: 0x000A2854, Port1: 0x000A4854

Offset Formula: 0x000A2854+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	PT_EN	RW 0x0	PHY Test Enable 0: PHY Test disable 1: PHY Test enable Pattern must be selected while PT_EN (R15h [15]) = 0. When set to 1, this signal clears the pattern and error counts and begins looking for pattern lock. When set to zero this stops the testing and freeze the error and pattern counts.
14	PT_RST	RW 0x0	PHY Test Reset. This is a reset signal to PHY Test. Once its set to 1, all registers in PHY Test are cleared.
13	PT_PASS	RO 0x0	PHY Test Pass 0: 1 or more errors or the pattern is not locked 1: Pattern is locked and no errors detected.
12	PT_LOCK	RO 0x0	PHY Test Pattern Lock 0: Pattern detector is not locked onto the pattern 1: Patter detector had locked onto the pattern If the pattern doesn’t lock then either the signal quality is low or the pattern provided to the receiver doesn’t match the programmed pattern.

Table 1012:PHY Test Control 0 Register (n=0–1) (Continued)

Offset: Port0: 0x000A2854, Port1: 0x000A4854

Offset Formula: 0x000A2854+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
11	PT_XFER_DIFF	RW 0x0	PHY Test Transmit Jitter Pattern Select Option 0: PT_DATA[7:0] selects jitter test pattern for both transmit and receive data 1: PT_DATA [95:88] selects jitter test pattern for transmit data and PT_DATA [7:0] selects pattern for receive data comparison When this signal is 1 the transmitter and receiver can be programmed to use different jitter patterns. This field only has meaning when using the jitter patterns PT_PATTERN_SEL (R15h [7:4]) = 1110b. Both fields PT_DATA [95:88] and PT_DATA [7:0] have the same encoding.
10	PT_PATSYN_EN	RW 0x0	PHY Test Pattern Sync Enable If enabled, a pattern sync signal is generated and pulsed at the pattern repeat point. This can be used as a pattern trigger for test equipment.
9	PT_START_RD	RW 0x0	PHY Test Start Running Disparity Selects the initial running disparity for SATA test patterns. 0: Initial disparity for pattern is negative 1: Initial disparity for pattern is positive
8	PT_LONG_SHORT	RW 0x0	PHY Test Long SATA Patterns 0: Short version of SATA patterns 1: Long version of SATA patterns
7:4	PT_PATTERN_SEL	RW 0x0	PHY Test Pattern Select This selects the pattern from the table of patterns or specifies an alternate table to use for additional jitter test patterns. Patterns can only be selected when PT_EN (R15h [15]) = 0.
3:0	PT_LOCK_CNT	RW 0x0	Pattern Lock Count Program with the number of pattern signatures to be found -1. Once n + 1 of these signatures has been found PT_LOCK (R15h [12]) is asserted and the bit error test begins.

Table 1013:PHY Test PRBS Error Counter 0 Register (n=0–1)

Offset: Port0: 0x000A287C, Port1: 0x000A487C

Offset Formula: 0x000A287C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved

Table 1013:PHY Test PRBS Error Counter 0 Register (n=0–1) (Continued)

Offset: Port0: 0x000A287C, Port1: 0x000A487C

Offset Formula: 0x000A287C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
15:0	PT_PRBS_ERR_ CNT[31:16]	RO 0x0	PHY Test Error Count [31:16] Indicates how may bit errors were encountered after obtaining pattern lock.

Table 1014:PHY Test PRBS Error Counter 1 Register (n=0–1)

Offset: Port0: 0x000A2880, Port1: 0x000A4880

Offset Formula: 0x000A2880+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	PT_PRBS_ERR_ CNT[15:0]	RO 0x0	PHY Test Error Count [15:0] The error count indicates how may bit errors were encountered after obtaining pattern lock.

Table 1015:PHY Test OOB 0 Register (n=0–1)

Offset: Port0: 0x000A2884, Port1: 0x000A4884

Offset Formula: 0x000A2884+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	PT_OOB_EN	RW 0x0	PHY Test Out Of Band (OOB) Enable This allows PHY Test to control the idle signal to generate pattern that resembles OOB. Must set PT_EN (R54h [15]) to run.
14	Reserved	RSVD 0x0	Reserved
13:12	PT_TESTMODE	RW 0x0	Test Mode. 0: Test Disable 1: Test Enable

Table 1015:PHY Test OOB 0 Register (n=0–1) (Continued)

Offset: Port0: 0x000A2884, Port1: 0x000A4884

Offset Formula: 0x000A2884+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
11:10	PT_OOB_SPEED	RW 0x0	PHY Test Out Of Band (OOB) Speed 0h: Every bit of the OOB pattern is transmitted 1 time (Use when programmed to 1.5G). 1h: Every bit of the OOB pattern is transmitted 2 times (Use when programmed to 3G). 2h: Every bit of the OOB pattern is transmitted 4 times (Use when programmed to 6G). 4h: Every bit of the OOB pattern is transmitted 8 times (reserved for future speed).
9:8	PT_OOB_PAT_SEL	RW 0x0	PHY Test Out Of Band (OOB) Pattern Select Selects which pattern is transmitted during a emulated OOB burst. 00: Transmit the align primitive. 01: Transmit D24.3 pattern. 10: Reserved - do not use. 11: Reserved - do not use.
7:0	PT_OOB_IDLE_ LEN	RW 0x0	Out Of Band (OOB) Idle Length. Specifies the emulated OOB gap width. IDLE period = pt_oob_idle_len (R84h [7:0]) × 40 UI.

Table 1016:Digital Loopback Enable Register (n=0–1)

Offset: Port0: 0x000A288C, Port1: 0x000A488C

Offset Formula: 0x000A288C+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:12	Reserved	RSVD 0x0	Reserved
11:10	SEL_BITS	RW 0x1	Select Data Bit Width. 00: 10-bit 01: 20-bit 10: 40-bit 11: Reserved
9	CDR_LOCK_MODE	RW 0x0	CDR Lock Mode. 0: CDR lock with clock only. 1: CDR lock with both clock and data.
8	CDR_LOCK_DET_ EN	RW 0x0	CDR Lock Detection Enable. 0: CDR lock detection is disabled. 1: CDR lock detection is enabled.
7:0	Reserved	RSVD 0x0	Reserved

Table 1017:PHY Isolation Mode Control Register (n=0–1)

Offset: Port0: 0x000A2898, Port1: 0x000A4898

Offset Formula: 0x000A2898+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:15	Reserved	RSVD 0x0	Reserved
14	EOM_CK_ALIGN_EN	RW 0x0	EOM Clock Align Enable. To enable EOM clock and data clock alignment circuits.
13:0	Reserved	RSVD 0x100	Reserved

Table 1018:Reference Clock Select Register (n=0–1)

Offset: Port0: 0x000A2918, Port1: 0x000A4918

Offset Formula: 0x000A2918+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:11	Reserved	RSVD 0x0	Reserved
10	REFCLK_SEL	RW 0x0	0x0: RefClk0 (100MHz for PCIe). 0x1: RefClk1 (25MHz for SATA & GbE).
9:0	Reserved	RSVD 0x0	Reserved

Table 1019:COMPHY Control Register (n=0–1)

Offset: Port0: 0x000A2920, Port1: 0x000A4920

Offset Formula: 0x000A2920+0x2000*n: where n (0-1) represents Port

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x1210	Reserved
2	RX_HIZ	RW 0x1	RX High Impedance Mode.
1:0	Reserved	RSVD 0x0	Reserved

A.11.6 Shadow Register Block Map

Table 1020:Shadow Register Block Registers Map

Bits 31-24	Bits 23-16	Bits 15-8	Bits 7-0	Offset	Comment
Reserved	Reserved	Device PIO Data register		Port 0: 0xA2100	see ATA/ATAPI specification
Reserved	Reserved	Reserved	Device - Features (Features Current)/Error register	Port 0: 0xA2104	
Reserved	Reserved	Reserved	Device - Sector Count (Sector Count Current) register	Port 0: 0xA2108	
Reserved	Reserved	Reserved	Device - LBA Low register	Port 0: 0xA210C	
Reserved	Reserved	Reserved	Device - LBA Mid register	Port 0: 0xA2110	
Reserved	Reserved	Reserved	Device - LBA High register	Port 0: 0xA2114	
Reserved	Reserved	Reserved	Device - Device/Head (Device) register	Port 0: 0xA2118	
Reserved	Reserved	Reserved	Device - Command/Status register	Port 0: 0xA211C	
Reserved	Reserved	Reserved	Device - Control/ Alternate Status register	Port 0: 0xA2120	

A.12 Serial Peripheral Interface (SPI) Registers

The following table provides a summarized list of all registers that belong to the SPI, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1021:Summary Map Table for the SPI Registers

Register Name	Offset	Table and Page
SPI Control Register (N=0–1)	SPI_Controller0: 0x00010600, SPI_Controller1: 0x00010680	Table 1022, p. 1278
SPI Interface Configuration Register (N=0–1)	SPI_Controller0: 0x00010604, SPI_Controller1: 0x00010684	Table 1023, p. 1279
SPI Data Out Register (N=0–1)	SPI_Controller0: 0x00010608, SPI_Controller1: 0x00010688	Table 1024, p. 1281
SPI Data In Register (N=0–1)	SPI_Controller0: 0x0001060C, SPI_Controller1: 0x0001068C	Table 1025, p. 1281
SPI Interrupt Cause Register (N=0–1)	SPI_Controller0: 0x00010610, SPI_Controller1: 0x00010690	Table 1026, p. 1281
SPI Interrupt Mask Register (N=0–1)	SPI_Controller0: 0x00010614, SPI_Controller1: 0x00010694	Table 1027, p. 1282
SPI Timing Parameters 1 Register (N=0–1)	SPI_Controller0: 0x00010618, SPI_Controller1: 0x00010698	Table 1028, p. 1282
SPI Timing Parameters 2 Register (N=0–1)	SPI_Controller0: 0x0001061C, SPI_Controller1: 0x0001069C	Table 1029, p. 1283
SPI Direct Write Configuration Register (N=0–1)	SPI_Controller0: 0x00010620, SPI_Controller1: 0x000106A0	Table 1030, p. 1284
SPI Direct Write Header Register (N=0–1)	SPI_Controller0: 0x00010624, SPI_Controller1: 0x000106A4	Table 1031, p. 1284
SPI Direct Read Header Register (N=0–1)	SPI_Controller0: 0x00010628, SPI_Controller1: 0x000106A8	Table 1032, p. 1285
SPI CS Address Decode Register (N=0–1)	SPI_Controller0: 0x0001062C, SPI_Controller1: 0x000106AC	Table 1033, p. 1285
SPI CSn Timing Parameters Register (n=0–7, N=0–1)	SPI_Controller0SPI CS0: 0x00010630, SPI_Controller0SPI CS1: 0x00010634...SPI_Controller1SPI CS7: 0x000106CC	Table 1034, p. 1286
SPI Controller Version Register (N=0–1)	SPI_Controller0: 0x00010650, SPI_Controller1: 0x000106D0	Table 1035, p. 1287

Table 1022:SPI Control Register (N=0–1)

Offset: SPI_Controller0: 0x00010600, SPI_Controller1: 0x00010680

Offset Formula: 0x00010600+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved

Table 1022:SPI Control Register (N=0–1) (Continued)

Offset: SPI_Controller0: 0x00010600, SPI_Controller1: 0x00010680

Offset Formula: 0x00010600+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
4:2	SPI CS num	RW 0x0	This field defines which SPI_CS will be asserted when <CSnAct>=1. It has a binary format, thus, at any given point of time at most one SPI_CS will be active. 0 = SPI_CSn[0] 1 = SPI_CSn[1] 2 = SPI_CSn[2] 3 = SPI_CSn[3] 4 = SPI_CSn[4] 5 = SPI_CSn[5] 6 = SPI_CSn[6] 7 = SPI_CSn[7]
1	SMemRdy	RO 0x1	Serial Memory Data Transfer Ready
0	CSnAct	RW 0x0	Serial Memory Control 0 = Deactivated: Serial Memory Interface is deactivated. The SPI_CSn output is driven HIGH. 1 = Activated: Serial Memory Interface is activated. The SPI_CSn output is driven LOW.

Table 1023:SPI Interface Configuration Register (N=0–1)

Offset: SPI_Controller0: 0x00010604, SPI_Controller1: 0x00010684

Offset Formula: 0x00010604+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	Reserved_1	RW 0x0	Must be 0.
14	RxLSBF	RW 0x0	Receive LSB-First Enable 1 = Data Recieves is treaeated as least significant bit first. 0 = Data Recieves is treaeated as most significant bit first.
13	TxLSBF	RW 0x0	Trammit LSB-First Enable 1 = Data is transferred least significant bit first. 0 = Data is transferred most significant bit first.

Table 1023:SPI Interface Configuration Register (N=0–1) (Continued)

Offset: SPI_Controller0: 0x00010604, SPI_Controller1: 0x00010684

Offset Formula: 0x00010604+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
12	CPHA	RW 0x0	SPI Clock Phase Bit This bit is used to shift the SCK serial clock. 0 = half_cycle_in: The first SCK edge is issued one-half cycle into the transfer operation 1 = beggining: The first SCK edge is issued at the beginning of the transfer operation
11	CPOL	RW 0x0	SPI Clock Polarity Bit This bit selects an inverted or non-inverted SPI clock. To transmit data between SPI modules, the SPI modules must have identical CPOL values. 0 = Active_High: Active-high clocks selected; SCK idles low 1 = Active_Low: Active-low clocks selected; SCK idles high
10	DirectRdHighSpeed	RW 0x0	SPI Direct Read High Speed Transaction Defines rather the SPI issues a dummy byte between the last byte of the address phase and the first byte of the data phase. For example, to activate an SPI Flash Fast_Read Command, USER should set this bit and configure <DirectRdHeader>to be 0x0B 0 = Normal: Data will be driven on the byte following the last address byte 1 = HS: A dummy byte will be issued between last Address byte and first data byte
9:8	DirectAddrLen	RW 0x2	Address length of SPI. This field defines the number of bytes of the address phase in a direct accesses (READ or WRITE) to SPI. For direct Writes, this field is considered only if <Direct_Wr_Addr_En> is active. 0 = 1Byte: SPI has a 1-byte address length. 1 = 2Byte: SPI has a 2-byte address length. 2 = 3Byte: SPI has a 3-byte address length. 3 = 4Byte: SPI has a 4-byte address length.
7:6	SPI_SPPR_HI	RW 0x0	SPI Baud Rate Pre-selection Bits 2,1 This filed together with SPI_SPPR0 combine the SPPR field. See SPR field description for a defitnition of the baude rate calculation.
5	BYTE_LEN	RW 0x0	Number of bytes in each serial flash I/O transfer. 0 = 1Byte 1 = 2Byte
4	SPI_SPPR0	RW 0x1	SPI Baud Rate Pre-selection Bit0 This filed together with SPI_SPPR_HI combine the SPPR field. See SPR field description for a defitnition of the baude rate calculation.

Table 1023:SPI Interface Configuration Register (N=0–1) (Continued)

Offset: SPI_Controller0: 0x00010604, SPI_Controller1: 0x00010684

Offset Formula: 0x00010604+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
3:0	SPI_SPR	RW 0xd	SPI Baud Rate Selection Bits The SPR together with the SPPR define the SPI CLK frequency as follows: SPI actaul frequency = core_clk / (SPR * (2 ^ SPPR))

Table 1024:SPI Data Out Register (N=0–1)

Offset: SPI_Controller0: 0x00010608, SPI_Controller1: 0x00010688

Offset Formula: 0x00010608+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	SMemOut	RW 0x0	Writing to this register initiates a transfer out to external serial memory. The content of this register will be shifted out serially. When BYTE_LEN = 1, bits 7:0 are shifted out. When BYTE_LEN=2, bits 7:0 are shifted out, followed by bits 15:8. To shift in data from the serial memory, a dummy write can be used to advance SPI_SCK.

Table 1025:SPI Data In Register (N=0–1)

Offset: SPI_Controller0: 0x0001060C, SPI_Controller1: 0x0001068C

Offset Formula: 0x0001060C+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	SMemIn	RO 0x0	While SPI Data Out register bits are shifted out on SPI_MOSI, the value sampled on SPI_MISO is shifted into this register. When BYTE_LEN = 1, data is shifted into bit 0 and left shifted. When BYTE_LEN = 2, data is shifted into bits7:0 first and then bits [15:8].

Table 1026:SPI Interrupt Cause Register (N=0–1)

Offset: SPI_Controller0: 0x00010610, SPI_Controller1: 0x00010690

Offset Formula: 0x00010610+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.			
31:1	Reserved	RSVD 0x0	Reserved

Table 1026:SPI Interrupt Cause Register (N=0–1) (Continued)

Offset: SPI_Controller0: 0x00010610, SPI_Controller1: 0x00010690

Offset Formula: 0x00010610+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
0	SMemRdiInt	RW0C 0x0	Serial memory data transfer ready interrupt.

Table 1027:SPI Interrupt Mask Register (N=0–1)

Offset: SPI_Controller0: 0x00010614, SPI_Controller1: 0x00010694

Offset Formula: 0x00010614+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	SMemRdiIntMask	RW 0x1	Mask bit for the serial memory data transfer ready interrupt. Mask only affects the assertion of interrupt pin. It does not affect the setting of bits in the Cause register 0 = Masked: Interrupt masked 1 = Enabled: Interrupt enabled

Table 1028:SPI Timing Parameters 1 Register (N=0–1)

Offset: SPI_Controller0: 0x00010618, SPI_Controller1: 0x00010698

Offset Formula: 0x00010618+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:12	SPI_TCS_HOLD	RW 0x4	Spi Timing CS Hold This field defines the minimal gap between last SPI_CLK edge and CS de-assertion The gap is calculated in core_clk cycles.
11:8	SPI_TCS_SETUP	RW 0x4	Spi Timing CS Setup This field defines the minimal gap between CS assertion and first SPI_CLK edge The gap is calculated in core_clk cycles.

Table 1028:SPI Timing Parameters 1 Register (N=0–1) (Continued)

Offset: SPI_Controller0: 0x00010618, SPI_Controller1: 0x00010698

Offset Formula: 0x00010618+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
7:6	SPI_TMISO_ SAMPLE	RW 0x1	This field defines the gap between SPI CLK sampling edge (as determined by CPOL and CPHA) and the actual core clock cycle that MISO is captured. 0x0 - MISO sample occurs upon SPI_CLK edge 0x1 - MISO sample occurs 1 core_clk after SPI_CLK edge 0x2 - MISO sample occurs 2 core_clk after SPI_CLK edge 0x3 - MISO sample occurs 3 core_clk after SPI_CLK edge User is restricted to set this field such that the sample does not occur befor the following SPI_CLK edge. In other words, SPI_CLK frequency as defined by <SPR> and <SPPR> and this field need to maintain: 1/2 SPI_CLK > TMISO_SAMPLE
5:0	SP_TCSH	RW 0x3f	SPI Timing CS High minimum gap. This field , joint with <SPI_TCSH_9_6>, define the minimal gap between CS de-assertion and CS assertion. The gap is calculated in number of core_clk cycles. Notice that the init value is maximal to enable a wide variety of Boot flash devices. To increase performance, Marvell encourages to add to the Boot code a configuartion write that will cahnge this field to be the minimal values suitable to the corresponding Boot Device.

Table 1029:SPI Timing Parameters 2 Register (N=0–1)

Offset: SPI_Controller0: 0x0001061C, SPI_Controller1: 0x0001069C

Offset Formula: 0x0001061C+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0xF00	Reserved
3:0	SPI_TCSH_9_6	RW 0x0	Spi Timing CS High bits [9:6]. This field , together with SPI_TCSH define the required gap between CS assertion and CS de-assertion. The actual gap is {SPI_TCSH_9_6, SPI_TCSH}.

Table 1030:SPI Direct Write Configuration Register (N=0–1)

Offset: SPI_Controller0: 0x00010620, SPI_Controller1: 0x000106A0

Offset Formula: 0x00010620+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:20	Reserved	RSVD 0x0	Reserved
19:17	Direct Wr Cs Hold	RW 0x0	In Direct Writes, this field defines the minimal number of core clock cycles between assertion of CS and the first SPI_CLK.
16	Direct Wr Deassert Cs	RW 0x1	When this bit is active (high), direct writes will assert SPI_CS prior to the data transmission and de-assert SPI_CS upon completion of the data transmission. If this feature is de-activated (set to 0) , then prior to any direct write, SPI_CS assertion should be guaranteed by software.
15:9	Reserved	RSVD 0x0	Reserved
8	Direct Wr Addr Enable	RW 0x0	When this bit is active (high), every direct write access to SPI will be prefixed with the first <DirectAddrLen> bytes of the request address.
7:3	Reserved	RSVD 0x0	Reserved
2:1	Direct Wr Hdr Size	RW 0x0	This field specifies the number of bytes to be transmitted as the data prefix. 0 = 1 Bytes: Hdr[7:0] will be transmitted 1 = 2 Bytes: Hdr[15:0] will be transmitted 2 = 3 Bytes: Hdr[23:0] will be transmitted 3 = 4 Bytes: Hdr[31:0] will be transmitted
0	Direct Write Hdr Enable	RW 0x0	When this bit is active (high), every direct write access to SPI will be prefixed with the first <Direct Wr Hdr Size> bytes of <DirectWrHdrData> .

Table 1031:SPI Direct Write Header Register (N=0–1)

Offset: SPI_Controller0: 0x00010624, SPI_Controller1: 0x000106A4

Offset Formula: 0x00010624+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:0	SpiDirectHdr	RW 0x0	Upon a direct access to SPI, the value in this field will be added as a header to the data. The number of bytes that will be added is defined according to <Sirect_Wr_Hdr_Size>.

Table 1032:SPI Direct Read Header Register (N=0–1)

Offset: SPI_Controller0: 0x00010628, SPI_Controller1: 0x000106A8

Offset Formula: 0x00010628+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	DirectRdHeader	RW 0x3	SPI Direct Read Header Defines the SPI command byte that will be issued upon a Direct Read transaction. Default value is the standard SPI Read Command. User should align configuration of this field with <DirectRdHighSpeed> and <DirectAddrLen> for a correct correlation with the target's protocol.

Table 1033: SPI CS Address Decode Register (N=0–1)

Offset: SPI_Controller0: 0x0001062C, SPI_Controller1: 0x000106AC

Offset Formula: 0x0001062C+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x0	Reserved
20:16	SPI Window1IndexLSB	RW 0x0	SPI Window1 Index LSB When <SPI DecodeMode> is configured to DoubleWindow, this field defines the LSB Index of the two address bits that determines the target CS (of CS[7:4]) When <SPI DecodeMode> is configured to AttrPerCS or to SingleWindow , the value in this field has no affect.
15:13	Reserved	RSVD 0x0	Reserved
12:8	SPI Window0IndexLSB	RW 0x0	SPI Window0 Index LSB When <SPI DecodeMode> is configured to SingleWindow, this field defines the LSB Index of the three address bits that determines the target CS (of CS[7:0]) When <SPI DecodeMode> is configured to DoubleWindow, this field defines the LSB Index of the two address bits that determines the target CS (of CS[3:0]) When <SPI DecodeMode> is configured to AttrPerCS the value in this field has no affect.
7:2	Reserved	RSVD 0x0	Reserved

Table 1033: SPI CS Address Decode Register (N=0–1) (Continued)
Offset: SPI_Controller0: 0x0001062C, SPI_Controller1: 0x000106AC
Offset Formula: 0x0001062C+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
1:0	SPI DecodeMode	RW 0x0	SPI CS Decode Mode Define the mapping mode between an SPI direct access and its target CS. 0 = AttrPerCs: Every CS has its dedicated ATTR. The target CS is the binary encoding of {Attr[0],Attr[7:6]} 1 = SingleWindow: The CS is defined according to three consecutive bits of the address. The LSB index of the 3 bits is defined by <SPI_Window0IndexLSB>. 3 = DoubleWindow: There are two SPI Windows. SPI Window0 is mapped to CS0-3 and SPI Window1 is mapped to CS4-7. The Window is determined by the ATTR as follows: ATTR[0]=0 : SPI Window0 ATTR[0]=1 : SPI Window1 For Window0, the target CS (of CS[3:0]) is defined according to two consecutive bits of the address. The LSB of these two bits is defined by <SPI_Window0IndexLSB> For Window1, the target CS (of CS[7:4]) is defined according to two consecutive bits of the address. The LSB of these two bits is defined by <SPI_Window1IndexLSB>

Table 1034: SPI CSn Timing Parameters Register (n=0–7, N=0–1)
Offset: SPI_Controller0SPI CS0: 0x00010630, SPI_Controller0SPI CS1: 0x00010634...SPI_Controller1SPI CS7: 0x000106CC
Offset Formula: 0x00010630+4*n + 128*N : where N (0-1) represents SPI_Controller, where n (0-7) represents SPI CS

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved
23:20	SPI_CSn_TCS_HOLD	RW 0x0	SPI CSn Hold See description at <TCS_HOLD> in register <SPI Timing Paramters 1>
19:16	SPI_CSn_TCS_SETUP	RW 0x0	SPI CSn Setup See description at <TCS_SETUP> in register <SPI Timing Paramters 1>
15	Reserved	RSVD 0x0	Reserved
14:13	SPI_CSn_TMISO_SAMPLE	RW 0x0	SPI CSn Timing Miso Sample See description at <TMISO_SAMPLE> in register <SPI Timing Paramters 1>
12	SPI_CSn_CPHA	RW 0x0	SPI CSn Clock Phase See description at <CPHA> in register <SPI Interface Configuration>
11	SPI_CSn_CPOL	RW 0x0	SPI CSn Clock Polarity See description at <CPOL> in register <SPI Interface Configuration>

Table 1034: SPI CSn Timing Parameters Register (n=0–7, N=0–1) (Continued)
Offset: SPI_Controller0SPI CS0: 0x00010630, SPI_Controller0SPI CS1: 0x00010634...SPI_Controller1SPI CS7: 0x000106CC
Offset Formula: 0x00010630+4*n + 128*N : where N (0-1) represents SPI_Controller, where n (0-7) represents SPI CS

Bit	Field	Type/InitVal	Description
10	SPI_CSn_ DirectRdHighSpeed	RW 0x0	SPI CSn Read Direct accesses high speed See description at <DirectRdHighSpeed> in register <SPI Interface Configuration>
9:8	SPI_CSn_ DirectAddrLen	RW 0x0	SPI CSn Direct accesses Address length See description at <DirectAddrLen> in register <SPI Interface Configuration>
7:5	SPI_CSn_SPPR	RW 0x0	SPI CSn SPPR See description at <SPI_SPR> in register <SPI Interface Configuration>
4:1	SPI_CSn_SPR	RW 0x0	SPI CSn SPR See description at <SPI_SPR> in register <SPI Interface Configuration>
0	SpiParamMode	RW 0x0	SPI CSn parameter mode 0 = Global: Transaction to SPI CS%n act according to the global set of paramters which are located at: <SPI Interface Configurations>, <SPI Timing Parameters 1>, <SPI Timing Parameters 2> 1 = Local: Transaction to SPI CS%n act accoriding to its dedicated parameters

Table 1035: SPI Controller Version Register (N=0–1)
Offset: SPI_Controller0: 0x00010650, SPI_Controller1: 0x000106D0
Offset Formula: 0x00010650+128*N: where N (0-1) represents SPI_Controller

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:4	SpiMainVersion	RO 0x1	Spi controller Version - Main field
3:0	SpiSubVersion	RO 0x0	Spi Controller Sub Version

A.13 Time Division Multiplexing (TDM) Controller Registers

The following table provides a summarized list of all registers that belong to the Time Division Multiplexing (TDM) Controller, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1036:Summary Map Table for the Time Division Multiplexing (TDM) Controller Registers

Register Name	Offset	Table and Page
Multi Channel DMA (MCDMA)		
MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31)	ChNum0: 0x000B3000, ChNum1: 0x000B3004... ChNum31: 0x000B307C	Table 1037, p. 1291
MCDMA Receive-FIFO Management Linked List <n> Register (n=0–127)	ChNum0: 0x000B3C00, ChNum1: 0x000B3C04... ChNum127: 0x000B3DFC	Table 1038, p. 1293
MCDMA Current Receive Descriptor Pointer (MCRDPx) Channel<n> Register (n=0–31)	ChNum0: 0x000B4000, ChNum1: 0x000B4004... ChNum31: 0x000B407C	Table 1039, p. 1293
MCDMA Global Control Register MGCR	0x000B4400	Table 1040, p. 1293
Rx Service Queue Arbiter Weight Register	0x000B4408	Table 1041, p. 1294
IQC Rx Buffer Status Interrupt Enable Register	0x000B440C	Table 1042, p. 1294
MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31)	ChNum0: 0x000B5000, ChNum1: 0x000B5004... ChNum31: 0x000B507C	Table 1043, p. 1295
MCDMA Transmit-FIFO Management Linked List<n> Register (n=0–127)	ChNum0: 0x000B5C00, ChNum1: 0x000B5C04... ChNum127: 0x000B5DFC	Table 1044, p. 1297
MCDMA Current Transmit Descriptor Pointer (MCTDPx) channel<n> Register (n=0–31)	ChNum0: 0x000B7000, ChNum1: 0x000B7004... ChNum31: 0x000B707C	Table 1045, p. 1297
Tx Service Queue Arbiter Weight Register	0x000B7408	Table 1046, p. 1297
IQC Tx Buffer Status Interrupt Enable Register	0x000B740C	Table 1047, p. 1298
Multi Channel Serial Controller (MCSC)		
MCSC Channel-x Command Execution Status Register Channel <n> (n=0–31)	Chnum0: 0x000B0000, Chnum1: 0x000B0004... Chnum31: 0x000B007C	Table 1048, p. 1298
MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31)	ChNum0: 0x000B0400, ChNum1: 0x000B0404... ChNum31: 0x000B047C	Table 1049, p. 1299
MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31)	ChNum0: 0x000B1800, ChNum1: 0x000B1804... ChNum31: 0x000B187C	Table 1050, p. 1303
MCSC Global Configuration Register	0x000B2800	Table 1051, p. 1305
MCSC Global Interrupt Cause Register	0x000B2804	Table 1052, p. 1306
MCSC Extended Interrupt Cause Register	0x000B2808	Table 1053, p. 1308
MCSC Global Interrupt Mask Register	0x000B280C	Table 1054, p. 1310

Table 1036:Summary Map Table for the Time Division Multiplexing (TDM) Controller Registers

Register Name	Offset	Table and Page
MCSC Extended Interrupt Mask Register	0x000B2810	Table 1055, p. 1310
IQC0 (Interrupt Queue) First Entry Address - IQF0 Register	0x000B2814	Table 1056, p. 1310
IQC0 (Interrupt Queue) Last entry Address - IQL0 Register	0x000B2818	Table 1057, p. 1310
IQC0 (Interrupt Queue) Head Address - IQH0 Register	0x000B281C	Table 1058, p. 1311
IQC0 (Interrupt Queue) Tail Address - IQT0 Register	0x000B2820	Table 1059, p. 1311
IQC0 (Interrupt Queue) Enable Interrupt Register	0x000B2824	Table 1060, p. 1311
MCSC TX Channel Balancing Mask Register	0x000B284C	Table 1061, p. 1313
MCSC RX Channel Balancing Mask Register	0x000B2850	Table 1062, p. 1313
IQC1 (Interrupt Queue) First Entry Address - IQF1 Register	0x000B2854	Table 1063, p. 1313
IQC1 (Interrupt Queue) Last entry Address - IQL1 Register	0x000B2858	Table 1064, p. 1314
IQC1 (Interrupt Queue) Head Address - IQH1 Register	0x000B285C	Table 1065, p. 1314
IQC1 (Interrupt Queue) Tail Address - IQT1 Register	0x000B2860	Table 1066, p. 1314
IQC2 (Interrupt Queue) First Entry Address - IQF2 Register	0x000B2864	Table 1067, p. 1314
IQC2 (Interrupt Queue) Last entry Address - IQL2 Register	0x000B2868	Table 1068, p. 1315
IQC2 (Interrupt Queue) Head Address - IQH2 Register	0x000B286C	Table 1069, p. 1315
IQC2 (Interrupt Queue) Tail Address - IQT2 Register	0x000B2870	Table 1070, p. 1315
IQC3 (Interrupt Queue) First Entry Address - IQF3 Register	0x000B2874	Table 1071, p. 1315
IQC3 (Interrupt Queue) Last entry Address - IQL3 Register	0x000B2878	Table 1072, p. 1316
IQC3 (Interrupt Queue) Head Address - IQH3 Register	0x000B287C	Table 1073, p. 1316
IQC3 (Interrupt Queue) Tail Address - IQT3 Register	0x000B2880	Table 1074, p. 1316
IQC1 (Interrupt Queue) Enable Interrupt Register	0x000B2884	Table 1075, p. 1316
IQC2 (Interrupt Queue) Enable Interrupt Register	0x000B2888	Table 1076, p. 1318
IQC3 (Interrupt Queue) Enable Interrupt Register	0x000B288C	Table 1077, p. 1320
MCSC Global Configuration Extension Register	0x000B2890	Table 1078, p. 1322
IQC0 Channels Responsibility Register	0x000B2894	Table 1079, p. 1324
IQC1 Channels Responsibility Register	0x000B2898	Table 1080, p. 1324
IQC2 Channels Responsibility Register	0x000B289C	Table 1081, p. 1324
IQC3 Channels Responsibility Register	0x000B28A0	Table 1082, p. 1324
HDLC Mode Address1 and Address2 Register	0x000B28A4	Table 1083, p. 1325
HDLC Mode Address3 and Address4 Register	0x000B28A8	Table 1084, p. 1325
HDLC Mode Address Filtering Register	0x000B28AC	Table 1085, p. 1326
Shared Bus to Mbus Bridge		
TDMMC Window0 Control Register	0x000B8A00	Table 1086, p. 1327
TDMMC Window0 Size Register	0x000B8A04	Table 1087, p. 1327

Table 1036:Summary Map Table for the Time Division Multiplexing (TDM) Controller Registers

Register Name	Offset	Table and Page
TDMMC Window1 Control Register	0x000B8A08	Table 1088, p. 1327
TDMMC Window1 Size Register	0x000B8A0C	Table 1089, p. 1328
TDMMC Window2 Control Register	0x000B8A10	Table 1090, p. 1328
TDMMC Window2 Size Register	0x000B8A14	Table 1091, p. 1329
TDMMC Window3 Control Register	0x000B8A18	Table 1092, p. 1329
TDMMC Window3 Size Register	0x000B8A1C	Table 1093, p. 1329
TDMMC Window4 Control Register	0x000B8A20	Table 1094, p. 1330
TDMMC Window4 Size Register	0x000B8A24	Table 1095, p. 1330
TDMMC Window5 Control Register	0x000B8A28	Table 1096, p. 1330
TDMMC Window5 Size Register	0x000B8A2C	Table 1097, p. 1331
TDMMC Window6 Control Register	0x000B8A30	Table 1098, p. 1331
TDMMC Window6 Size Register	0x000B8A34	Table 1099, p. 1331
TDMMC Window7 Control Register	0x000B8A38	Table 1100, p. 1332
TDMMC Window7 Size Register	0x000B8A3C	Table 1101, p. 1332
TDMMC Window8 Control Register	0x000B8A40	Table 1102, p. 1332
TDMMC Window8 Size Register	0x000B8A44	Table 1103, p. 1333
TDMMC Window9 Control Register	0x000B8A48	Table 1104, p. 1333
TDMMC Window9 Size Register	0x000B8A4C	Table 1105, p. 1333
TDMMC Window10 Control Register	0x000B8A50	Table 1106, p. 1334
TDMMC Window10 Size Register	0x000B8A54	Table 1107, p. 1334
TDMMC Window11 Control Register	0x000B8A58	Table 1108, p. 1334
TDMMC Window11 Size Register	0x000B8A5C	Table 1109, p. 1335
TDMMC Window0 High Address Register	0x000B8A80	Table 1110, p. 1335
TDMMC Window1 High Address Register	0x000B8A84	Table 1111, p. 1335
TDMMC Window2 High Address Register	0x000B8A88	Table 1112, p. 1335
TDMMC Window3 High Address Register	0x000B8A8C	Table 1113, p. 1335
TDMMC Window4 High Address Register	0x000B8A90	Table 1114, p. 1336
TDMMC Window5 High Address Register	0x000B8A94	Table 1115, p. 1336
TDMMC Windows Access Protect Register	0x000B8B00	Table 1116, p. 1336
TDMMC Window0 Enable Register	0x000B8B04	Table 1117, p. 1337
TDMMC Window1 Enable Register	0x000B8B08	Table 1118, p. 1338
TDMMC Window2 Enable Register	0x000B8B0C	Table 1119, p. 1338
TDMMC Window3 Enable Register	0x000B8B10	Table 1120, p. 1339
TDMMC Window4 Enable Register	0x000B8B14	Table 1121, p. 1339
TDMMC Window5 Enable Register	0x000B8B18	Table 1122, p. 1340
TDMMC Window6 Enable Register	0x000B8B1C	Table 1123, p. 1340
TDMMC Window7 Enable Register	0x000B8B20	Table 1124, p. 1341
TDMMC Window8 Enable Register	0x000B8B24	Table 1125, p. 1342
TDMMC Window9 Enable Register	0x000B8B28	Table 1126, p. 1342
TDMMC Window10 Enable Register	0x000B8B2C	Table 1127, p. 1343

Table 1036:Summary Map Table for the Time Division Multiplexing (TDM) Controller Registers

Register Name	Offset	Table and Page
TDMMC Window11 Enable Register	0x000B8B30	Table 1128, p. 1343
Time Division Multiplexing (TDM) Control and Configuration		
FlexTDM Transmit Dual Port RAM (TDPR)<n> Register (n=0–255)	TxDPEntry0: 0x000B8000, TxDPEntry1: 0x000B8004... TxDPEntry255: 0x000B83FC	Table 1129, p. 1344
FlexTDM Receive Dual Port RAM (RDPR)<n> Register (n=0–255)	TxDPEntry0: 0x000B8400, TxDPEntry1: 0x000B8404... TxDPEntry255: 0x000B87FC	Table 1130, p. 1345
FlexTDM0 Transmit Read Pointer Register	0x000B8800	Table 1131, p. 1346
FlexTDM0 Receive Read Pointer Register	0x000B8804	Table 1132, p. 1347
FlexTDM0 Configuration Register (TCR)	0x000B8808	Table 1133, p. 1347
TDM Clock and Sync Control Register	0x000B881C	Table 1134, p. 1349
TDM Clock Divider Control Register	0x000B8820	Table 1135, p. 1351
TDM Reserved Clock Divider Control Register	0x000B8824	Table 1136, p. 1351
Time Division Multiplexing Interrupt Controller		
TDMMC Top Cause Register	0x000B8C00	Table 1137, p. 1351
TDMMC Functional Cause Register	0x000B8C40	Table 1138, p. 1352
TDMMC Error Cause Register	0x000B8C44	Table 1139, p. 1354
TDMMC Top Mask Register	0x000B8C80	Table 1140, p. 1355
TDMMC Output Sync Bit Count Register	0x000B8C8C	Table 1141, p. 1355
Voice Periodic Interrupt Control Register	0x000B8C90	Table 1142, p. 1356
TDMMC Functional Mask Register	0x000B8CC0	Table 1143, p. 1356
TDMMC Error Mask Register	0x000B8CC4	Table 1144, p. 1356
TDMMC Data Delay and Clock Control Register	0x000B8CD0	Table 1145, p. 1357
TDMMC Plus Minus Delay Control for FsyncOut Register	0x000B8CD4	Table 1146, p. 1357
TDMMC Plus Minus Delay Control for FsyncIn Register	0x000B8CD8	Table 1147, p. 1358

A.13.1 Multi Channel DMA (MCDMA)

Table 1037:MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31)

Offset: ChNum0: 0x000B3000, ChNum1: 0x000B3004...ChNum31: 0x000B307C

Offset Formula: 0x000B3000+0x4*n: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
31:7	Reserved	RSVD 0x0	Reserved

Table 1037:MCDMA Receive Control Register (RMCCx) Channel<n> (n=0–31) (Continued)

Offset: ChNum0: 0x000B3000, ChNum1: 0x000B3004...ChNum31: 0x000B307C

Offset Formula: 0x000B3000+0x4*n: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
6	ERD	RW 0x0	Enable Rx DMA Channel When set to 1, the Rx DMA is ready for a receive frame. The device resets configuration bits to the reset value, and clears ERD when the receive DMA Channel has a resource error, or when the CPU issues an Abort command (see " MCSC Channel-x Receive Configuration Register (MRCRx) channels n" (n=0-31)) and the MCSC finishes executing the command.
5	RCP	RW 0x0	Receive Channel Priority There is a High Service Priority queue and a Low Service Priority queue. The Receive Channel Priority determines to which service queue the requests go. 0 = Low: Low Priority Channel. 1 = High: High Priority Channel.
4	RIFB	RW 0x0	Receive Interrupt on Frame Boundaries When set, the Rx MCDMA generates interrupts only on frame boundaries (i.e., after writing the frame status to the descriptor).
3	Reserved	RSVD 0x0	Reserved
2	BLMR	RW 0x1	Big/Little Endian Receive part. The device supports Big or Little Endian configuration per channel for maximum system flexibility. The BLMR bit only affects data movements. 0 = Big Endian 1 = Little Endian
1:0	RBSZ	RW 0x0	Rx Burst Size Sets the maximum burst size for Rx MCDMA transactions. **NOTE:** This is also the Rx Threshold parameter that sets the point from where the DMA starts transferring data to the memory. **NOTE:** The burst size must be equal to or smaller than the size of the linked Rx buffer. 0 = 1-word: Burst is limited to one 64-bit word. 1 = 2-words: Burst is limited to two 64-bit words. 2 = 4-words: Burst is limited to four 64-bit words. 3 = 8-words: Burst is limited to eight 64-bit words.

Table 1038:MCDMA Receive-FIFO Management Linked List <n> Register (n=0–127)

Offset: ChNum0: 0x000B3C00, ChNum1: 0x000B3C04...ChNum127: 0x000B3DFC

Offset Formula: 0x000B3C00+0x4*n: where n (0-127) represents ChNum

Bit	Field	Type/InitVal	Description
31:0	LinkListEntry	RW %n	Next Entry Pointer. Each entry in the linked list SRAMs represents the index of the next basic buffering unit in the FIFO that belongs to the same channel. Each channel’s linked list must be cyclic. The reset value is n, meaning: Channel ID (entry0 will be 0x0, entry1 will be 0x1 ... entry31 will be 0x1F). Since we have 32-channels, the reset value per entry is valid ONLY for entry0-entry31. (All other entries are not initialized)

Table 1039:MCDMA Current Receive Descriptor Pointer (MCRDPx) Channel<n> Register (n=0–31)

Offset: ChNum0: 0x000B4000, ChNum1: 0x000B4004...ChNum31: 0x000B407C

Offset Formula: 0x000B4000+0x4*n: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
31:0	MCRDPx	RW 0x0	The MCRDPx Stands for MCDMA-RX-Current Descriptor Pointer There are 32 such pointers, one per channel. Each Current Descriptor Pointer Register points to the address of the current Descriptor that is processed by the MCDMA. The CPU must write this register with the first descriptor address before enabling the MCDMA receive channel.

Table 1040:MCDMA Global Control Register MGCR

Offset: 0x000B4400

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	RID	RW 0x0	Reset Initialization Completed After an external reset or a software reset, the MCDMA and the MCSC initialize the different channel parameters and resources to initial values. This bit indicates when the process is complete. The CPU must poll for this bit before it starts programming the TDM-MC (Multi-Channel), and activating channels. 0 = Incomplete: The Reset Initialization is not complete. 1 = Complete: The Reset Initialization is complete.
0	Reserved	RSVD 0x0	Reserved

Table 1041:Rx Service Queue Arbiter Weight Register

Offset: 0x000B4408

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28:24	RSQW	RW 0x04	Rx Service Queue Arbiter Weight This field determines the number of high-priority entries to be serviced before one low-priority entry is served. The value of 0x1F indicates an unlimited number of high-priority entries served before one low-priority entry is served. The value of 0x0 indicates an unlimited number of low-priority entries served before one high-priority entry is served.
23:16	HPRT	RW 0x30	High-Priority Rx Threshold This field determines the threshold of the Rx high-priority service queue. Whenever the number of entries in the queue exceeds the value of this field, an interrupt indication is sent to the CPU. The interrupt cause bit is described in the MCSC Extended Interrupt Cause Register.
15:8	LPRT	RW 0x30	Low-Priority Rx Threshold This field determines the threshold of the Rx low-priority service queue. Whenever the number of entries in the queue exceeds the value of this field, an interrupt indication is sent to the CPU. The interrupt cause bit is described in the MCSC Extended Interrupt Cause Register.
7:0	Reserved	RSVD 0x40	Reserved

Table 1042:IQC Rx Buffer Status Interrupt Enable Register

Offset: 0x000B440C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	RxSttsInterruptEn	RW 0x0	Rx Buffer Status Interrupt Enable This field defines which bits in the Rx buffer status field are monitored and reported to the CPU when the descriptor is closed with a status bit set to 1. For instance if bit[6] is set to 1 and the <ChRxStts> bit is set in the IQC Enable Interrupt Register, each Rx Buffer Descriptor that is closed with an overrun status bit set is reported to the CPU by inserting an appropriate interrupt entry to the interrupt queue. **NOTE:** Bit[0] in this register enables monitoring bit[0] in the Rx buffer status field, bit[1] in this register enables monitoring bit[1] in the Rx buffer status field, and so on.

Table 1043:MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31)

Offset: ChNum0: 0x000B5000, ChNum1: 0x000B5004...ChNum31: 0x000B507C

Offset Formula: 0x000B5000+0x4*n: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
31:18	Reserved	RSVD 0x0	Reserved
17	TXD	RW 0x0	Tx Demand When this bit is set to 1, the Tx DMA fetches the first descriptor, and starts the transmission process. The device clears the TXD when it successfully ends a DMA transmit process. It also resets configuration bits to the reset value, and clears the TXD when a resource error occurs after the CPU issues an abort command or a Stop Tx command.
16	AT	RW 0x0	Abort Transmit The CPU sets the AT bit to 1 when it needs to abort a transmit DMA channel operation. When the AT bit is set, the DMA aborts its operation. The device clears both the AT and TXD bits. The CPU can poll this bit. When the CPU reads the bit as 0, it indicates that the device has completed the abort sequence. Besides polling, the CPU can also standby for a "ChTxEnd" interrupt. After the abort command is executed, the CPU must write the first descriptor address, and then set TxD bit to 1. When the CPU issues the Abort command to change the channel’s TDM configuration or the channel’s MCSC configuration, it must also wait for a ChTxEIDL interrupt from MCSC, indicating transmission of an abort pattern to the line, and that CSC channel has moved to idle state.
15	STD	RW 0x0	Stop Tx The MCDMA Channel stops transmission at the end of the frame, that is, at the end of the buffer, with the L bit set to 1. After transmitting the last buffer, the device clears the STD and TXD bits. After the MCDMA Channel stops, the CPU must write the first descriptor address, and then set the TxD bit to 1. The device signals the CPU with a "ChTxEnd" interrupt when the stop procedure is accomplished.
14	URPM	RW 0x0	Under Run Protection Mode Setting this bit to 1 lowers the Under Run probability in highly loaded systems, but can slightly lower the link utilization. If not set, the MCDMA will not start transferring data to the MCSC before half of the TxFIFO is full, or the whole chain of buffers is transferred to the FIFO. When the FIFO is half full, the MCDMA starts transferring data to the MCSC, and continues filling the FIFO simultaneously until the chain is transmitted, or an Under Run occurs. When set, the MCDMA does not start transferring data to the MCSC before half of the TxFIFO is full, or the whole chain of buffers is transferred to the FIFO. When the FIFO is half full, the MCDMA starts transferring data to the MCSC and continues filling the FIFO simultaneously until the first frame is transmitted. It then ensures that the FIFO is at least half full before it continues transmitting the next frame. The same is applicable to the third and fourth frames, and so on.

Table 1043:MCDMA Transmit Control Register (TMCCx) Channel<n> (n=0–31) (Continued)

Offset: ChNum0: 0x000B5000, ChNum1: 0x000B5004...ChNum31: 0x000B507C

Offset Formula: 0x000B5000+0x4*n: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
13	TPC	RW 0x0	Transmit Channel Priority There is a High Service Priority queue, and a Low Service Priority queue. The Transmit Channel Priority determines to which Service Queue the requests go. 0 = Low: Low Priority 1 = High: High Priority
12	NDUR	RW 0x0	Non-Destructive Underrun This bit is relevant only to Transparent channels. For HDLC channels, the bit must be set to 0. When this bit is set, an Underrun (UR) event does not cause the MCDMA to stop transferring data to the MCSC and does not cause it to stop transmitting data bits. When the Tx Descriptor closes, it is marked with the UR flag set.
11	Reserved	RSVD 0x0	Reserved
10	BLMT	RSVD 0x1	Big/Little Endian in Transmit Part. The device supports Big or Little Endian configuration per channel for maximum system flexibility. The BLMT bit only affects data movement. 0 = Big Endian 1 = Little Endian
9:8	TBSZ	RW 0x0	Tx Burst Size Sets the maximum burst size for Tx MCDMA transactions. **NOTE:** This is also the Tx Threshold parameter, which sets the point where the DMA starts transferring data from the external memory. 0 = 1-words: Burst is limited to one 64-bit word. 1 = 2-words: Burst is limited to two 64- bit words. 2 = 4-words: Burst is limited to four 64-bit words. 3 = 8-words: Burst is limited to eight 64-bit words.
7:0	FSize	RW 0x1	FIFO Size The user must initialize this field according to and equal to the number of blocks in the channel TxFIFO. Each block contains four 64-bit lines, or 32-byte size. **NOTE:** The MCDMA starts transferring data to the MCSC only after the TxFIFO of the channel is half full, or when the Tx buffer chain is consumed (all buffers are transferred to the FIFO). **NOTE:** The burst size must be equal to or smaller than the size of the linked Tx buffer.

Table 1044:MCDMA Transmit-FIFO Management Linked List<n> Register (n=0–127)

Offset: ChNum0: 0x000B5C00, ChNum1: 0x000B5C04...ChNum127: 0x000B5DFC

Offset Formula: 0x000B5C00+0x4*n: where n (0-127) represents ChNum

Bit	Field	Type/InitVal	Description
31:0	LinkListEntry	RW %n	Next Entry Pointer Each entry in the linked list SRAMs represents the index of the next basic buffering unit in the FIFO, that belong to that same channel. Each channel's linked list must be cyclic. Each entry in the Link List holds its value (Ch-ID). The reset value is n, meaning: Channel ID (entry0 will be 0x0, entry1 will be 0x1 ... entry31 will be 0x1F). Since we have 32-channels, the reset value per entry is valid ONLY for entry0-entry31. (All other entries are not initialized)

Table 1045:MCDMA Current Transmit Descriptor Pointer (MCTDPx) channel<n> Register (n=0–31)

Offset: ChNum0: 0x000B7000, ChNum1: 0x000B7004...ChNum31: 0x000B707C

Offset Formula: 0x000B7000+0x4*n: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
31:0	MCTDPx	RW 0x0	The MCTDPx Stands for MCDMA-TX-Current Descriptor Pointer There are 32 such pointers, one per channel. Each Current Descriptor Pointer Register points to the address of the current Descriptor that is being processed by the MCDMA. The CPU must write this register with the first descriptor address before enabling the MCDMA transmit channel.

Table 1046:Tx Service Queue Arbiter Weight Register

Offset: 0x000B7408

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28:24	TSQW	RW 0x04	Tx Service Queue Arbiter Weight This field determines the number of high-priority entries to be serviced before one low-priority entry is served. The value of 0x1F indicates an unlimited number of high-priority entries served before one low-priority entry is served. The value of 0x0 indicates an unlimited number of low-priority entries served before one high-priority entry is served.
23:16	HPTT	RW 0x30	High-Priority Tx Threshold This field determines the threshold of the Tx high-priority service queue. Whenever the number of entries in the queue exceeds the value of this field, an interrupt indication is sent to the CPU. The interrupt cause bit is described in the MCSC Extended Interrupt Cause Register.

Table 1046:Tx Service Queue Arbiter Weight Register (Continued)

Offset: 0x000B7408

Bit	Field	Type/InitVal	Description
15:8	LPTT	RW 0x30	Low-Priority Tx Threshold This field determines the threshold of the Tx low-priority service queue. Whenever the number of entries in the queue exceeds the value of this field, an interrupt indication is sent to the CPU. The interrupt cause bit is described in the MCSC Extended Interrupt Cause Register.
7:0	Reserved	RSVD 0x40	Reserved

Table 1047:IQC Tx Buffer Status Interrupt Enable Register

Offset: 0x000B740C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	TxSttsInterruptEn	RW 0x0	Tx Buffer Status Interrupt Enable This field defines which bits in the Tx buffer status field is monitored and reported to the CPU when the descriptor is closed with a status bit set to 1. For instance if bit[6] is set to 1 and the <ChTxStts> bit is set in the IQC Enable Interrupt Register, then each Tx Buffer Descriptor that is closed with an underrun status bit set, is reported to the CPU by inserting an appropriate interrupt entry to the interrupt queue. **NOTE:** Bit[0] in this register enables monitoring bit[0] in the Tx buffer status field, bit[1] in this register enables monitoring bit[1] in the Tx buffer status field, and so on.

A.13.2 Multi Channel Serial Controller (MCSC)

Table 1048:MCSC Channel-x Command Execution Status Register Channel <n> (n=0–31)

Offset: Chnum0: 0x000B0000, Chnum1: 0x000B0004...Chnum31: 0x000B007C

Offset Formula: 0x000B0000+n*0x4: where n (0-31) represents Chnum

Bit	Field	Type/InitVal	Description
31:28	Reserved	RO 0x0	Reserved
27	CrdExec	RO 0x0	Close Rx Descriptor Command Executed When the MCSC encounters a CPU's Close Rx Descriptor command, it executes it and sets this bit to 1. The CPU can poll this bit to determine when the command is executed. This bit is cleared automatically by the MCSC as a result of clearing the original command bit in the channel's configuration register. **NOTE:** This bit and the CRD Command are relevant only to Transparent channels. For HDLC channels, this bit is defined as Reserved. **NOTE:**

Copyright © 2014 Marvell

Table 1048: MCSC Channel-x Command Execution Status Register Channel <n> (n=0–31) (Continued)

Offset: Chnum0: 0x000B0000, Chnum1: 0x000B0004...Chnum31: 0x000B007C

Bit	Field	Type/InitVal	Description
26	EhExec	RO 0x0	Enter Hunt Command Executed When the MCSC encounters a CPU’s Enter Hunt command, it executes it and sets this bit to 1. The CPU can poll this bit to find out when the command is executed. This bit is cleared automatically by the MCSC as a result of clearing the original command bit in the channel’s configuration register.
25	AbortExec	RO 0x0	Abort Command Executed When the MCSC encounters a CPU’s Abort command, it executes it and sets this bit to 1. The CPU can poll this bit to find out when the command is executed. This bit is cleared automatically by the MCSC as a result of clearing the original command bit in the channel’s configuration register.
24:0	Reserved	RO 0x0	Reserved

Table 1049: MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31)

Offset: ChNum0: 0x000B0400, ChNum1: 0x000B0404...ChNum31: 0x000B047C

Offset Formula: 0x000B0400+n*0x4: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
31	MODE	RW 0x0	Mode **NOTE:** No reset value is valid, the register must be programmed! 0 = HDLC protocol 1 = Transparent protocol
30	RRVD	RW 0x0	In Transparent / HDLC Mode: RRVD - Receive Reverse Data **NOTE:** No reset value is valid, register must be programmed! 0 = Normal Mode: Default 1 = Reverse Data Mode: MSB is shifted in first.

Copyright © 2014 Marvell

Table 1049:MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Continued)

Offset: ChNum0: 0x000B0400, ChNum1: 0x000B0404...ChNum31: 0x000B047C

Bit	Field	Type/InitVal	Description
29:28	CbsCrcm	RW 0x0	In Transparent Mode: bit[29] - Reserved bit[28] - Continuous Byte Sync: This bit enables flexible support of ADPCM and other similar techniques: 0 = The MCSC uses the byte sync signaling from the TDM only upon first appearance. It is used to start receiving data at the correct point in the time slot, otherwise, the sync signaling is ignored. 1 = When the MCSC receives the byte sync signaling from the TDM, it transfers the current bits received to the MCDMA. Received bits are shifted right, and the last received bit is inserted in bit seven. When only a portion of a byte is transferred to the MCDMA, the valid bits are aligned to the left, and the MSB bit is in bit seven of each byte. If the RRVD field (bit 30) is set, the received bits are shifted left, and the last received bit is inserted in bit zero. When only a portion of a byte is transferred to the MCDMA, the valid bits are aligned to the right, and the MSB bit is in bit zero of each byte. In HDLC Mode: bit[29:28] - CRC Mode 00 = CRC16-CCITT (HDLC based protocols, for example, X.25) (Default) 01 = Reserved 10 = CRC32-CCITT (HDLC-based protocols, for example, LAP-D. Identical to the Ethernet CRC) 11 = Reserved **NOTE:** No reset value is valid, register must be programmed!
27	ER	RW 0x0	In Transparent/HDLC Mode: Enable Receive This bit must be set before any operation related to the receive side of the channel is performed. When reset, the receive channel remains in the Idle state. **NOTE:** No reset value is valid, register must be programmed!
26	RINV	RW 0x0	In Transparent /HDLC Mode: Receive bit stream inversion. **NOTE:** No reset value is valid, register must be programmed! 0 = No invert 1 = Invert: Invert the data before it is processed in the MCSC data path.

Table 1049:MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Continued)

Offset: ChNum0: 0x000B0400, ChNum1: 0x000B0404...ChNum31: 0x000B047C

Bit	Field	Type/InitVal	Description
25	NrorBrcst	RW 0x0	In Transparent Mode: Non Reactive Over Run When this bit is not set, an Overrun event causes the MCSC to transfer the next byte to the MCDMA with a status word, with the OR bit set. The Rx Descriptor is closed, with the OR flag set, and the descriptor is incremented. The MCSC continues processing Rx bits and transferring them to the MCDMA, which transfers them to the next buffers. When this bit is set, the overrun event does not cause any change in the MCSC data flow, and is ignored. In HDLC Mode: Broadcast Enable Enables the reception of HDLC broadcast address (0xFFFF or 0xFF, depending on the AF setting). **NOTE:** No reset value is valid, register must be programmed!
24	NullEn	RW 0x0	In Transparent Mode: Reserved. In HDLC Mode: Null Enable Enables the reception of HDLC NULL address (0x0000 or 0x00 depending on the AF setting). **NOTE:** No reset value is valid, register must be programmed!
23	CrdAf	RW 0x0	In Transparent Mode: Close Rx Descriptor When the CPU issues a CRD command, the current receive descriptor is closed, and the following received data is transferred into a new buffer. If there is no active receive in progress, no action takes place. The CPU is responsible for clearing this bit after it receives an indication that it was executed, either by polling the channel’s Command Execution Status Register, or after receiving an interrupt indicating so (see Interrupt Queue Controller). In HDLC Mode: Address Filtering Mode Used to set the mode of Broadcast and Null Address Filtering: 0 = For 8 bit HDLC/LAPB like address filtering. 1 = For 16 bit LAPD-like address filtering. **NOTE:** No reset value is valid, register must be programmed!

Table 1049:MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Continued)
Offset: ChNum0: 0x000B0400, ChNum1: 0x000B0404...ChNum31: 0x000B047C

Bit	Field	Type/InitVal	Description
22	EH	RW 0x0	In Transparent Mode: Enter Hunt Upon receiving an Enter Hunt command, the receive machine moves to a HUNT state, and continuously searches for an Fsync. If the Enter Hunt command is received during a frame reception, the data that is already in the MCDMA FIFO is written to the buffers, and the last descriptor is closed with a Last indication. The CPU is responsible for clearing this bit after it receives an indication that it is executed by polling the channel's Command Execution Status register. In HDLC Mode: Enter Hunt Upon receiving the Enter Hunt command, the receive machine moves to the HUNT state and continuously searches for an opening flag. If Enter Hunt mode command is issued during frame reception, the current descriptor is closed with Last indication and CRC error indication. The CPU is responsible for clearing this bit after it receives an indication that it was executed by polling the channel's Command Execution Status Register. **NOTE:** No reset value is valid, register must be programmed!
21	AbortRx	RW 0x0	In Transparent /HDLC Mode: Abort Reception Abort is received immediately, and goes to the IDLE state. The data that is already in the MCDMA FIFO is written to the buffers. The MCDMA generates a REINT (Resource Error Interrupt), clears the ERD bit,l and goes to IDLE state, only after the MCDMA ChannelX has moved the data to the memory. To enable reception, the processor must issue an Enter Hunt command after the abort command. The CPU is responsible for clearing this bit after it receives an indication that it was executed by polling the channel's Command Execution Status register. **NOTE:** No reset value is valid, register must be programmed!

Table 1049:MCSC Channel-x Receive Configuration Register (MRCRx) Channels<n> (n=0–31) (Continued)

Offset: ChNum0: 0x000B0400, ChNum1: 0x000B0404...ChNum31: 0x000B047C

Bit	Field	Type/InitVal	Description
20	B2bHDLC	RW 0x0	In Transparent Mode: Reserved In HDLC Mode: Back to Back - 0 = The MCSC receive channel expects at least one opening flag following the packet's closing flag, before it allows receiving a new packet. It ignores any data appearing between the closing flag of a packet and the opening flag of the next packet. Flag sharing is not allowed. 1 = The MCSC receive channel enables back-to-back packet reception. Two packets that arrive with a shared flag are received correctly. **NOTE:** No reset value is valid, register must be programmed!
19:0	Reserved	RW 0x0	Reserved in both HDLC and Transparent Modes.

Table 1050:MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31)

Offset: ChNum0: 0x000B1800, ChNum1: 0x000B1804...ChNum31: 0x000B187C

Offset Formula: 0x000B1800+n*0x4: where n (0-31) represents ChNum

Bit	Field	Type/InitVal	Description
31	MODE	RW 0x0	Mode 0 = HDLC protocol 1 = Transparent protocol **NOTE:** No reset value is valid, register must be programmed! 0 = HDLC protocol 1 = Transparent protocol
30	TRVD	RW 0x0	In Transparent/HDLC Mode: TRVD - Transmit Reverse Data **NOTE:** No reset value is valid, register must be programmed! 0 = Normal mode: Default 1 = Reverse Data mode: MSB is shifted out first.

Table 1050:MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31) (Continued)

Offset: ChNum0: 0x000B1800, ChNum1: 0x000B1804...ChNum31: 0x000B187C

Bit	Field	Type/InitVal	Description
29:28	CbcCrcm	RW 0x0	In Transparent Mode: bit[29] - Reserved. bit[28] - Continuous Byte Sync: This bit enables flexible support of ADPCM and other similar techniques: 0 = The MCSC uses the byte sync indication from the TDM only upon first appearance. It is used to start transmitting data at the correct point in the time slot; otherwise, the sync indication is ignored. 1 = When the MCSC receives the byte sync indication from the TDM, it stops transmitting the rest of the byte, and moves on to the next byte. The lower bits of the byte are transmitted first (from right to left). The valid bits which the CPU intends to transmit must be shifted right, and aligned to bit zero of each byte in memory. If the TRVD field (bit 30) is set, the higher bits of the byte are transmitted first (from left to right). The valid bits which the CPU intends to transmit must be shifted left, and aligned to bit seven of each byte in memory. In HDLC Mode: bits[29:28] - CRC Mode 00 = CRC16-CCITT (HDLC based protocols, for example, X.25) 01 = Reserved 10 = CRC32-CCITT (HDLC based protocols, for example, LAP-D. Identical to the Ethernet CRC) 11 = Reserved **NOTE:** No reset value is valid, register must be programmed!
27	ET	RW 0x0	In Transparent/HDLC Mode: Enable Transmit This bit must be set before any operation related to the transmit side of the channel is performed. When reset, the transmit channel remains in Idle state. **NOTE:** No reset value is valid, register must be programmed!
26	TINV	RW 0x0	In Transparent/HDLC Mode: Transmit bit stream inversion. **NOTE:** No reset value is valid, register must be programmed! 0 = No invert. 1 = Invert: Invert the data after it is processed in the MCSC data path.
25	Reserved	RW 0x0	Reserved

Table 1050:MCSC Channel-x Transmit Configuration Register (MTCRx) Channels<n> (n=0–31) (Continued)

Offset: ChNum0: 0x000B1800, ChNum1: 0x000B1804...ChNum31: 0x000B187C

Bit	Field	Type/InitVal	Description
24:21	NofHDLC	RW 0x0	In Transparent Mode: Reserved. In HDLC Mode: Number of Flags NOF specifies the number of flags that are transmitted between consecutive frames. 0 specifies shared flag mode. In shared flag mode, the closing flag of a frame is used as the opening flag of the following frame. **NOTE:** No reset value is valid, register must be programmed!
20	IpsHDLC	RW 0x0	In Transparent Mode: Reserved. In HDLC Mode: Idle Pattern Select Determines whether a sequence of flags is transmitted or Idle pattern is transmitted. This is when the transmit channel is in Idle state, and has no packets to transmit: 0 = Idle pattern is transmitted (sequence of 0xFF). 1 = Flag pattern is transmitted (sequence of 0x7E). **NOTE:** No reset value is valid, register must be programmed!
19:0	Reserved	RW 0x0	Reserved

Table 1051:MCSC Global Configuration Register

Offset: 0x000B2800

Bit	Field	Type/InitVal	Description
31	TxEn	RW 0x0	Global Enable of MCSC Tx.
30	RxEn	RW 0x0	Global Enable of MCSC Rx.
29	Reserved	RW 0x0	Reserved. Must be 0x0.
28	Reserved	RSVD 0x0	Reserved
27	Reserved	RSVD 0x0	Reserved

Table 1051:MCSC Global Configuration Register (Continued)

Offset: 0x000B2800

Bit	Field	Type/InitVal	Description
26	BSZ	RW 0x0	Burst size When set, the interrupt queue controller maximum size of write burst to the memory is 8 double words. When reset, the maximum size of write burst is 4 double words (this value must be used when allocating the interrupt queue buffer in 32-bit-wide SDRAM).
25:23	Reserved	RSVD 0x0	Reserved
22:21	Reserved	RSVD 0x0	Reserved
20	Reserved	RW 0x0	Reserved. This bit must be programmed to 0x1 for proper operation.
19	Reserved	RSVD 0x0	Reserved
18:16	SHFR	RW 0x0	Short Frame Register Holds the minimum allowed data length in an HDLC frame. Short Frames are frames with a byte count less than SHFR + CRC-Bytes. If CRC16-CCITT is used, then the short frames are those with a frame length less than SHFR+2. If CRC32-CCITT is used, then short frames are those with a frame length less than SHFR+4. Short frames are closed with bit 8 (SFE), set in the descriptor’s status field.
15:0	FLBR	RW 0xFFFF	Frame Length Buffer Register Holds the maximum allowed data length in an HDLC frame. When a frame exceeds the number written in the FLBR, the remainder of the frame is discarded. The HDLC controller waits for a closing flag, and then returns the frame status with bit 7 (MFLE) set to 1.

Table 1052:MCSC Global Interrupt Cause Register

Offset: 0x000B2804

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved
30	MGIS	RO 0x0	MCSC Global Interrupt Register Summary Logical OR of (unmasked) bits 0-28 below. This bit is read only.
29	Reserved	RSVD 0x0	Reserved

Table 1052:MCSC Global Interrupt Cause Register (Continued)

Offset: 0x000B2804

Bit	Field	Type/InitVal	Description
28	MDIL	RW0C 0x0	MCDMA Interrupt Lost This interrupt occurs if the MCDMA tries to insert an interrupt to the Interrupt Queue, and the Interrupt Queue Controller cannot do so (in this case, the interrupt is lost). This can happen if the MCDMA tries to insert too many interrupt entries to the interrupt queue in a short period of time, and the interrupt queue controller does not succeed in transferring the entries to the memory fast enough.
27	IQNE	RW0C 0x0	Interrupt Queue Not Empty This interrupt indicates to the CPU that the Interrupt Queue Controller has written an interrupt entry to the Interrupt Queue in the external memory, and the CPU should handle this entry. Note that during the time between setting this bit and until it is handled by the CPU, more interrupt entries might be added to the queue.
26	IQOR	RW0C 0x0	Interrupt Queue Over Run This interrupt indicates that the interrupt queue in the external memory has suffered an overrun event. Such an event can happen when the CPU does not process the interrupt entries fast enough, and the interrupt queue gets full. The loss of an interrupt can have severe implications on the management of the channels, and should be avoided by correct system design.
25	InitDone	RW0C 0x0	RAM initialization Done This bit when set, indicates that the hardware completed resetting all the internal memories, and it is ready to start operating. The CPU must monitor this bit after reset (Hard or Soft) before it begins normal MCSC operation.
24	RxbOr	RW0C 0x0	Overrun In the Rx byte Machine When set, indicates that the MCSC Rx byte machine tries to send data to the MCDMA, while the MCDMA is not ready to accept data.
23:17	Reserved	RSVD 0x0	Reserved
16	TxOr	RW0C 0x0	Overrun In Tx Requests Synchronization FIFO When set, indicates that an overrun occurred in the Tx request FIFO.
15:9	Reserved	RSVD 0x0	Reserved

Table 1052:MCSC Global Interrupt Cause Register (Continued)

Offset: 0x000B2804

Bit	Field	Type/InitVal	Description
8	TxUr	RW0C 0x0	Underrun In the Tx Output Data Buffer When set, indicates that the TDM request for data was not acknowledged in time.
7:1	Reserved	RSVD 0x0	Reserved
0	RxOr	RW0C 0x0	Overrun In the Rx Data Synchronization FIFO When set, indicates that too much data was gathered in the Rx FIFO without being read by the bit machine. Possible reasons: Incoming data rate is too high.

Table 1053:MCSC Extended Interrupt Cause Register

Offset: 0x000B2808

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved
30	MEIS	RO 0x0	MCSC Extended Interrupt Register Summary Logical OR of (unmasked) bits 15-0 below. This bit is read only.
29:16	Reserved	RSVD 0x0	Reserved
15:9	Reserved	RSVD 0x0	Reserved
8	RxBitMachineOR	RW0C 0x0	OverrunI in the Rx Bit Machines Output Buffers. For Rx-bit-machine0 that is connected to Flex-TDM.
7	THPT	RW 0x0	Threshold Warning of Tx High Priority Request Queue This interrupt indicates that the number of entries in the Tx high priority request queue has exceeded the programmable threshold described in the "MCDMA Tx Service Queue Arbiter Weight Register".
6	TLPT	RW0C 0x0	Threshold Warning of Tx Low Priority Request Queue This interrupt indicates that the number of entries in the Tx low priority request queue has exceeded the programmable threshold described in the "MCDMA Tx Service Queue Arbiter Weight Register".

Table 1053:MCSC Extended Interrupt Cause Register (Continued)

Offset: 0x000B2808

Bit	Field	Type/InitVal	Description
5	RHPT	RW0C 0x0	Threshold Warning of Rx High Priority Request Queue This interrupt indicates that the number of entries in the Rx high priority request queue has exceeded the programmable threshold described in the "MCDMA Rx Service Queue Arbiter Weight Register".
4	RLPT	RW0C 0x0	Threshold Warning of Rx Low Priority Request Queue This interrupt indicates that the number of entries in the Rx low priority request queue has exceeded the programmable threshold described in the "MCDMA Rx Service Queue Arbiter Weight Register".
3	TxHPUR	RW0C 0x0	Tx High Priority Global Data Stream Under Run This interrupt indicates that a global under run has occurred. A global under run means that data was not brought to the MCSC in time for more than one channel simultaneously, without being able to indicate the under run on a separate channel basis. Such an event might force the CPU to reset, and re-initialize all High Priority Transmit channels in MCSC0. This interrupt is unlikely to occur in a well load-balanced system.
2	TxLPUR	RW0C 0x0	Tx Low Priority Global Data Stream Under Run This interrupt indicates that a global under run has occurred, a global under run means that data was not brought to the MCSC in time for more than one channel simultaneously, without being able to indicate the under run on a separate channel basis. Such an event might force the CPU to reset, and re-initialize all Low Priority Transmit channels in MCSC0. This interrupt is unlikely to occur in a well load-balanced system.
1	RxHPOR	RW0C 0x0	Rx High Priority Global Data Stream Over Run This interrupt indicates that a global overrun has occurred. A global overrun means that data was lost from more than one channel simultaneously, without being able to indicate the overrun on a separate channel basis. Such an event can force the CPU to reset, and re-initialize all High Priority Receive channels in MCSC0. This interrupt is unlikely to occur in a well load-balanced system.
0	RxLPOR	RW0C 0x0	Rx Low Priority Global Data Stream Over Run This interrupt indicates that a global overrun has occurred. A global overrun means that data was lost from more than one channel simultaneously, without being able to indicate the overrun on a separate channel basis. Such an event can force the CPU to reset, and re-initialize all Low Priority Receive channels in MCSC0. This interrupt is unlikely to occur in a well load-balanced system.

Table 1054:MCSC Global Interrupt Mask Register

Offset: 0x000B280C

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28:0	MaskBits	RW 0x0	This register masks the interrupt events detailed in MCSC Global Interrupt Cause Register. Each bit in this register masks the corresponding event in the aforementioned cause register.

Table 1055:MCSC Extended Interrupt Mask Register

Offset: 0x000B2810

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	MaskBits	RW 0x0	This register masks the interrupt events detailed in the MCSC Extended Interrupt cause register. Each bit in this register masks the corresponding event in the aforementioned cause register.

Table 1056:IQC0 (Interrupt Queue) First Entry Address - IQF0 Register

Offset: 0x000B2814

Bit	Field	Type/InitVal	Description
31:6	FA	RW 0x0	First Entry Address The interrupt queue first entry address set by the CPU must be 64-byte aligned.
5:0	ConstLowAddr	RO 0x0	Constant Low Address

Table 1057:IQC0 (Interrupt Queue) Last entry Address - IQL0 Register

Offset: 0x000B2818

Bit	Field	Type/InitVal	Description
31:6	LA	RW 0x0	Last Entry Address The interrupt queue last entry address set by the CPU must be 64-byte aligned.
5:0	ConstLowAddr	RO 0x38	Constant Low Address

Table 1058:IQC0 (Interrupt Queue) Head Address - IQH0 Register

Offset: 0x000B281C

Bit	Field	Type/InitVal	Description
31:3	HA	RO 0x0	Head Address The interrupt queue head address represents the entry address, where IQC0 inserts the next interrupt entry. This register is Read-Only, and is reset by the Interrupt Queue Controller to the IQF0 value when programming IQF0 or IQL0.
2:0	Reserved	RSVD 0x0	Reserved

Table 1059:IQC0 (Interrupt Queue) Tail Address - IQT0 Register

Offset: 0x000B2820

Bit	Field	Type/InitVal	Description
31:3	TA	RW 0x0	Tail Address The interrupt queue tail address represents the address from which the CPU fetches the next interrupt entry. If this register equals the IQH0 register, it means that the queue is empty. This field must be initialized by the CPU to the IQF0 Register value before the interrupt queue controller is enabled. This field must also be updated by the CPU when it empties an entry from CPU.
2:0	Reserved	RSVD 0x0	Reserved

Table 1060:IQC0 (Interrupt Queue) Enable Interrupt Register

Offset: 0x000B2824

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved
30	ChTxEOP	RW 0x0	MCSC ChannelX Tx End Of Packet Enables Interrupt insertion when the Tx machine of ChannelX transmits the last bit of a packet from which the descriptor (last descriptor, if the packet is divided to two or more buffers) was created with the EOPI (End Of Packet Interrupt) bit set. This interrupt can be used to determine the exact point in time that the last bit of a specific packet leaves the serial port.
29	ChRxEIL	RW 0x0	ChRxEIL MCSC ChannelX Rx Exit Idle Line State Enables interrupt insertion when the Rx machine of ChannelX monitorsat least one 0 after 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.

Table 1060:IQC0 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B2824

Bit	Field	Type/InitVal	Description
28	ChRxIL	RW 0x0	ChRxIL MCSC ChannelX Rx moved to Idle Line state Enables interrupt insertion in case the Rx machine of ChannelX monitors 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.
27	ChTxStts	RW 0x0	ChTxStts MCDMA ChannelX Closed Tx Descriptor with Non Zero Status Enables interrupt insertion when the ChannelX Tx Buffer Descriptor is closed with a status bit that is set to 1, and is not masked by the Tx Buffer Status Interrupt Mask Register. **NOTE:** Bit[27] of the IQC-RAM entries will be active according to three interrupt-enable fields from this register: <ChTxStts> (bit 27), <ChTxEnd> (bit 19) and <ChTxBuf> (bit 18). **NOTE:**
26	ChTxEIDL	RW 0x0	ChTxEIDL MCSC ChannelX Tx Entered Idle State Enables interrupt insertion when the MCSC ChannelX Tx completes transmission of an abort sequence in response to the CPUs abort command (issued to the MCDMA channel), and enters the IDLE state.
25	ChRxStts	RW 0x0	ChRxStts MCDMA ChannelX Closed Rx Descriptor With Non Zero Status Enables interrupt insertion when the ChannelX Rx Buffer Descriptor closes with a status bit that is set to 1, and is not masked by the Rx Buffer Status Interrupt Mask Register.
24:23	Reserved	RSVD 0x0	Reserved
22	ChRxCRD	RW 0x0	MCSC ChannelX Close Rx Descriptor Command Executed Enables interrupt insertion when the MCSC executes the CPU’s Close Rx Descriptor command. **NOTE:** The CRD Command and this interrupt are relevant only to Transparent channels.
21	ChRxEH	RW 0x0	MCSC ChannelX Rx Enter Hunt Command Executed Enables interrupt insertion when the MCSC executes the CPU’s Enter Hunt command.
20	ChRxAbrt	RW 0x0	ChRxAbrt MCSC ChannelX Rx Abort Command Executed Enables interrupt insertion when the MCSC executes the CPU’s Abort command.
19	ChTxEnd	RW 0x0	MCDMA ChannelX Tx End Enables interrupt insertion when a Tx resource error occurs, or the Tx DMA moves to IDLE after a Stop command and also, when the Tx DMA moves to IDLE after an Abort command. This interrupt is issued only after the channel’s FIFO is cleared of any valid data. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[19]. **NOTE:**

Table 1060:IQC0 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B2824

Bit	Field	Type/InitVal	Description
18	ChTxBuf	RW 0x0	MCDMA ChannelX Tx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Tx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued, but the descriptor's owner bit remains unchanged. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[18]. **NOTE:**
17	ChRxErr	RW 0x0	MCDMA ChannelX Rx Error Enables interrupt insertion when an Rx resource error occurs, or the Rx DMA moves to IDLE after an abort command is issued to the MCSC.
16	ChRxBuf	RW 0x0	MCDMA ChannelX Rx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Rx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued but the descriptor's owner bit remains unchanged.
15:0	Reserved	RSVD 0x0	Reserved

Table 1061:MCSC TX Channel Balancing Mask Register

Offset: 0x000B284C

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0xFFFFFFFF	Reserved

Table 1062:MCSC RX Channel Balancing Mask Register

Offset: 0x000B2850

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0xFFFFFFFF	Reserved

Table 1063:IQC1 (Interrupt Queue) First Entry Address - IQF1 Register

Offset: 0x000B2854

Bit	Field	Type/InitVal	Description
31:6	FA	RW 0x0	First Entry Address The interrupt queue first entry address set by the CPU must be 64byte aligned.
5:0	ConstLowAddr	RO 0x0	Constant Low Address

Table 1064:IQC1 (Interrupt Queue) Last entry Address - IQL1 Register

Offset: 0x000B2858

Bit	Field	Type/InitVal	Description
31:6	LA	RW 0x0	Last Entry Address The interrupt queue last entry address set by the CPU must be 64-byte aligned.
5:0	ConstLowAddr	RO 0x38	Constant Low Address

Table 1065:IQC1 (Interrupt Queue) Head Address - IQH1 Register

Offset: 0x000B285C

Bit	Field	Type/InitVal	Description
31:3	HA	RO 0x0	Head Address The interrupt queue head address represents the entry address, where IQC1 inserts the next interrupt entry. This register is Read-Only, and is reset by the Interrupt Queue Controller to the IQF1 value when programming IQF1 or IQL1.
2:0	Reserved	RSVD 0x0	Reserved

Table 1066:IQC1 (Interrupt Queue) Tail Address - IQT1 Register

Offset: 0x000B2860

Bit	Field	Type/InitVal	Description
31:3	TA	RW 0x0	Tail Address The interrupt queue tail address represents the address from which the CPU fetches the next interrupt entry. If this register equals the IQH1 register, it means that the queue is empty. This field must be initialized by the CPU to the IQF1 Register value before the interrupt queue controller is enabled. This field must also be updated by the CPU when it empties an entry from CPU.
2:0	Reserved	RSVD 0x0	Reserved

Table 1067:IQC2 (Interrupt Queue) First Entry Address - IQF2 Register

Offset: 0x000B2864

Bit	Field	Type/InitVal	Description
31:6	FA	RW 0x0	First Entry Address The interrupt queue first entry address set by the CPU must be 64byte aligned.
5:0	ConstLowAddr	RO 0x0	Constant Low Address

Table 1068:IQC2 (Interrupt Queue) Last entry Address - IQL2 Register

Offset: 0x000B2868

Bit	Field	Type/InitVal	Description
31:6	LA	RW 0x0	Last Entry Address The interrupt queue last entry address set by the CPU must be 64-byte aligned.
5:0	ConstLowAddr	RO 0x38	Constant Low Address

Table 1069:IQC2 (Interrupt Queue) Head Address - IQH2 Register

Offset: 0x000B286C

Bit	Field	Type/InitVal	Description
31:3	HA	RO 0x0	Head Address The interrupt queue head address represents the entry address, where IQC2 inserts the next interrupt entry. This register is Read-Only, and is reset by the Interrupt Queue Controller to the IQF2 value when programming IQF2 or IQL2.
2:0	Reserved	RSVD 0x0	Reserved

Table 1070:IQC2 (Interrupt Queue) Tail Address - IQT2 Register

Offset: 0x000B2870

Bit	Field	Type/InitVal	Description
31:3	TA	RW 0x0	Tail Address The interrupt queue tail address represents the address from which the CPU fetches the next interrupt entry. If this register equals the IQH2 register, it means that the queue is empty. This field must be initialized by the CPU to the IQF2 Register value before the interrupt queue controller is enabled. This field must also be updated by the CPU when it empties an entry from CPU.
2:0	Reserved	RSVD 0x0	Reserved

Table 1071:IQC3 (Interrupt Queue) First Entry Address - IQF3 Register

Offset: 0x000B2874

Bit	Field	Type/InitVal	Description
31:6	FA	RW 0x0	First Entry Address The interrupt queue first entry address set by the CPU must be 64byte aligned.
5:0	ConstLowAddr	RO 0x0	Constant Low Address

Table 1072:IQC3 (Interrupt Queue) Last entry Address - IQL3 Register

Offset: 0x000B2878

Bit	Field	Type/InitVal	Description
31:6	LA	RW 0x0	Last Entry Address The interrupt queue last entry address set by the CPU must be 64-byte aligned.
5:0	ConstLowAddr	RO 0x38	Constant Low Address

Table 1073:IQC3 (Interrupt Queue) Head Address - IQH3 Register

Offset: 0x000B287C

Bit	Field	Type/InitVal	Description
31:3	HA	RO 0x0	Head Address The interrupt queue head address represents the entry address, where IQC3 inserts the next interrupt entry. This register is Read-Only, and is reset by the Interrupt Queue Controller to the IQF3 value when programming IQF3 or IQL3.
2:0	Reserved	RSVD 0x0	Reserved

Table 1074:IQC3 (Interrupt Queue) Tail Address - IQT3 Register

Offset: 0x000B2880

Bit	Field	Type/InitVal	Description
31:3	TA	RW 0x0	Tail Address The interrupt queue tail address represents the address from which the CPU fetches the next interrupt entry. If this register equals the IQH3 register, it means that the queue is empty. This field must be initialized by the CPU to the IQF3 Register value before the interrupt queue controller is enabled. This field must also be updated by the CPU when it empties an entry from CPU.
2:0	Reserved	RSVD 0x0	Reserved

Table 1075:IQC1 (Interrupt Queue) Enable Interrupt Register

Offset: 0x000B2884

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved

Table 1075:IQC1 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B2884

Bit	Field	Type/InitVal	Description
30	ChTxEOP	RW 0x0	MCSC ChannelX Tx End Of Packet Enables Interrupt insertion when the Tx machine of ChannelX transmits the last bit of a packet from which the descriptor (last descriptor, if the packet is divided to two or more buffers) was created with the EOPI (End Of Packet Interrupt) bit set. This interrupt can be used to determine the exact point in time that the last bit of a specific packet leaves the serial port.
29	ChRxEIL	RW 0x0	ChRxEIL MCSC ChannelX Rx Exit Idle Line State Enables interrupt insertion when the Rx machine of ChannelX monitorsat least one 0 after 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.
28	ChRxIL	RW 0x0	ChRxIL MCSC ChannelX Rx moved to Idle Line state Enables interrupt insertion in case the Rx machine of ChannelX monitors 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.
27	ChTxStts	RW 0x0	ChTxStts MCDMA ChannelX Closed Tx Descriptor With Non Zero Status Enables interrupt insertion when the ChannelX Tx Buffer Descriptor is closed with a status bit that is set to 1, and is not masked by the Tx Buffer Status Interrupt Mask Register. **NOTE:** Bit[27] of the IQC-RAM entries will be active according to three interrupt-enable fields from this register: <ChTxStts> (bit 27), <ChTxEnd> (bit 19) and <ChTxBuf> (bit 18). **NOTE:**
26	ChTxEIDL	RW 0x0	ChTxEIDL MCSC ChannelX Tx Entered Idle State Enables interrupt insertion when the MCSC ChannelX Tx completes transmission of an abort sequence in response to the CPUs abort command (issued to the MCDMA channel), and enters the IDLE state.
25	ChRxStts	RW 0x0	ChRxStts MCDMA ChannelX Closed Rx Descriptor With Non Zero Status Enables interrupt insertion when the ChannelX Rx Buffer Descriptor closes with a status bit that is set to 1, and is not masked by the Rx Buffer Status Interrupt Mask Register.
24:23	Reserved	RSVD 0x0	Reserved
22	ChRxCRD	RW 0x0	MCSC ChannelX Close Rx Descriptor Command Executed Enables interrupt insertion when the MCSC executes the CPU’s Close Rx Descriptor command. **NOTE:** The CRD Command and this interrupt are relevant only to Transparent channels.

Table 1075:IQC1 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B2884

Bit	Field	Type/InitVal	Description
21	ChRxEH	RW 0x0	MCSC ChannelX Rx Enter Hunt Command Executed Enables interrupt insertion when the MCSC executes the CPU's Enter Hunt command.
20	ChRxAbrt	RW 0x0	ChRxAbrt MCSC ChannelX Rx Abort Command Executed Enables interrupt insertion when the MCSC executes the CPU's Abort command.
19	ChTxEnd	RW 0x0	MCDMA ChannelX Tx End Enables interrupt insertion when a Tx resource error occurs, or the Tx DMA moves to IDLE after a Stop command. Also, when the Tx DMA moves to IDLE after an Abort command. This interrupt is issued only after the channel's FIFO is cleared of any valid data. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[19]. **NOTE:**
18	ChTxBuf	RW 0x0	MCDMA ChannelX Tx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Tx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued, but the descriptor's owner bit remains unchanged. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[18]. **NOTE:**
17	ChRxErr	RW 0x0	MCDMA ChannelX Rx Error Enables interrupt insertion when an Rx resource error occurs, or the Rx DMA moves to IDLE after an abort command is issued to the MCSC.
16	ChRxBuf	RW 0x0	MCDMA ChannelX Rx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Rx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued but the descriptor's owner bit remains unchanged.
15:0	Reserved	RSVD 0x0	Reserved

Table 1076:IQC2 (Interrupt Queue) Enable Interrupt Register

Offset: 0x000B2888

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved

Table 1076:IQC2 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B2888

Bit	Field	Type/InitVal	Description
30	ChTxEOP	RW 0x0	MCSC ChannelX Tx End Of Packet Enables Interrupt insertion when the Tx machine of ChannelX transmits the last bit of a packet from which the descriptor (last descriptor, if the packet is divided to two or more buffers) was created with the EOPI (End Of Packet Interrupt) bit set. This interrupt can be used to determine the exact point in time that the last bit of a specific packet leaves the serial port.
29	ChRxEIL	RW 0x0	ChRxEIL MCSC ChannelX Rx Exit Idle Line State Enables interrupt insertion when the Rx machine of ChannelX monitorsat least one 0 after 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.
28	ChRxIL	RW 0x0	ChRxIL MCSC ChannelX Rx moved to Idle Line state Enables interrupt insertion in case the Rx machine of ChannelX monitors 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.
27	ChTxStts	RW 0x0	ChTxStts MCDMA ChannelX Closed Tx Descriptor With Non Zero Status Enables interrupt insertion when the ChannelX Tx Buffer Descriptor is closed with a status bit that is set to 1, and is not masked by the Tx Buffer Status Interrupt Mask Register. **NOTE:** Bit[27] of the IQC-RAM entries will be active according to three interrupt-enable fields from this register: <ChTxStts> (bit 27), <ChTxEnd> (bit 19) and <ChTxBuf> (bit 18). **NOTE:**
26	ChTxEIDL	RW 0x0	ChTxEIDL MCSC ChannelX Tx Entered Idle State Enables interrupt insertion when the MCSC ChannelX Tx completes transmission of an abort sequence in response to the CPUs abort command (issued to the MCDMA channel), and enters the IDLE state.
25	ChRxStts	RW 0x0	ChRxStts MCDMA ChannelX Closed Rx Descriptor With Non Zero Status Enables interrupt insertion when the ChannelX Rx Buffer Descriptor closes with a status bit that is set to 1, and is not masked by the Rx Buffer Status Interrupt Mask Register.
24:23	Reserved	RSVD 0x0	Reserved
22	ChRxCRD	RW 0x0	MCSC ChannelX Close Rx Descriptor Command Executed Enables interrupt insertion when the MCSC executes the CPU’s Close Rx Descriptor command. **NOTE:** The CRD Command and this interrupt are relevant only to Transparent channels.

Table 1076:IQC2 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B2888

Bit	Field	Type/InitVal	Description
21	ChRxEH	RW 0x0	MCSC ChannelX Rx Enter Hunt Command Executed Enables interrupt insertion when the MCSC executes the CPU's Enter Hunt command.
20	ChRxAbrt	RW 0x0	ChRxAbrt MCSC ChannelX Rx Abort Command Executed Enables interrupt insertion when the MCSC executes the CPU's Abort command.
19	ChTxEnd	RW 0x0	MCDMA ChannelX Tx End Enables interrupt insertion when a Tx resource error occurs, or the Tx DMA moves to IDLE after a Stop command. Also, when the Tx DMA moves to IDLE after an Abort command. This interrupt is issued only after the channel's FIFO is cleared of any valid data. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[19]. **NOTE:**
18	ChTxBuf	RW 0x0	MCDMA ChannelX Tx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Tx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued, but the descriptor's owner bit remains unchanged. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[18]. **NOTE:**
17	ChRxErr	RW 0x0	MCDMA ChannelX Rx Error Enables interrupt insertion when an Rx resource error occurs, or the Rx DMA moves to IDLE after an abort command is issued to the MCSC.
16	ChRxBuf	RW 0x0	MCDMA ChannelX Rx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Rx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued but the descriptor's owner bit remains unchanged.
15:0	Reserved	RSVD 0x0	Reserved

Table 1077:IQC3 (Interrupt Queue) Enable Interrupt Register

Offset: 0x000B288C

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved

Table 1077:IQC3 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B288C

Bit	Field	Type/InitVal	Description
30	ChTxEOP	RW 0x0	MCSC ChannelX Tx End Of Packet Enables Interrupt insertion when the Tx machine of ChannelX transmits the last bit of a packet from which the descriptor (last descriptor, if the packet is divided to two or more buffers) was created with the EOPI (End Of Packet Interrupt) bit set. This interrupt can be used to determine the exact point in time that the last bit of a specific packet leaves the serial port.
29	ChRxEIL	RW 0x0	ChRxEIL MCSC ChannelX Rx Exit Idle Line State Enables interrupt insertion when the Rx machine of ChannelX monitorsat least one 0 after 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.
28	ChRxIL	RW 0x0	ChRxIL MCSC ChannelX Rx moved to Idle Line state Enables interrupt insertion in case the Rx machine of ChannelX monitors 15 or more consecutive received 1 data bits. This is relevant for HDLC mode only.
27	ChTxStts	RW 0x0	ChTxStts MCDMA ChannelX Closed Tx Descriptor With Non Zero Status Enables interrupt insertion when the ChannelX Tx Buffer Descriptor is closed with a status bit that is set to 1, and is not masked by the Tx Buffer Status Interrupt Mask Register. **NOTE:** Bit[27] of the IQC-RAM entries will be active according to three interrupt-enable fields from this register: <ChTxStts> (bit 27), <ChTxEnd> (bit 19) and <ChTxBuf> (bit 18).
26	ChTxEIDL	RW 0x0	ChTxEIDL MCSC ChannelX Tx Entered Idle State Enables interrupt insertion when the MCSC ChannelX Tx completes transmission of an abort sequence in response to the CPUs abort command (issued to the MCDMA channel), and enters the IDLE state.
25	ChRxStts	RW 0x0	ChRxStts MCDMA ChannelX Closed Rx Descriptor With Non Zero Status Enables interrupt insertion when the ChannelX Rx Buffer Descriptor closes with a status bit that is set to 1, and is not masked by the Rx Buffer Status Interrupt Mask Register.
24:23	Reserved	RSVD 0x0	Reserved
22	ChRxCRD	RW 0x0	MCSC ChannelX Close Rx Descriptor Command Executed Enables interrupt insertion when the MCSC executes the CPU’s Close Rx Descriptor command. **NOTE:** The CRD Command and this interrupt are relevant only to Transparent channels.

Table 1077:IQC3 (Interrupt Queue) Enable Interrupt Register (Continued)

Offset: 0x000B288C

Bit	Field	Type/InitVal	Description
21	ChRxEH	RW 0x0	MCSC ChannelX Rx Enter Hunt Command Executed Enables interrupt insertion when the MCSC executes the CPU's Enter Hunt command.
20	ChRxAbrt	RW 0x0	ChRxAbrt MCSC ChannelX Rx Abort Command Executed Enables interrupt insertion when the MCSC executes the CPU's Abort command.
19	ChTxEnd	RW 0x0	MCDMA ChannelX Tx End Enables interrupt insertion when a Tx resource error occurs, or the Tx DMA moves to IDLE after a Stop command. Also, when the Tx DMA moves to IDLE after an Abort command. This interrupt is issued only after the channel's FIFO is cleared of any valid data. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[19].
18	ChTxBuf	RW 0x0	MCDMA ChannelX Tx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Tx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued, but the descriptor's owner bit remains unchanged. **NOTE:** The indication for this interrupt cause is at bit[27] within the entries of the IQC-Memory and NOT in bit[18].
17	ChRxErr	RW 0x0	MCDMA ChannelX Rx Error Enables interrupt insertion when an Rx resource error occurs, or the Rx DMA moves to IDLE after an abort command is issued to the MCSC.
16	ChRxBuf	RW 0x0	MCDMA ChannelX Rx Buffer Return Enables interrupt insertion when the MCDMA ChannelX Rx closes a descriptor, and returns the associated buffer to CPU ownership. In the case of AutoMode descriptors, the interrupt is issued but the descriptor's owner bit remains unchanged.
15:0	Reserved	RSVD 0x0	Reserved

Table 1078:MCSC Global Configuration Extension Register

Offset: 0x000B2890

Bit	Field	Type/InitVal	Description
31:4	Reserved	RSVD 0x0	Reserved.

Table 1078:MCSC Global Configuration Extension Register (Continued)

Offset: 0x000B2890

Bit	Field	Type/InitVal	Description
3	LinearRxByteSwap	RW 0x0	Linear Rx Byte Swapping The Byte Swapping feature enables MSByte first out in TDM, so that when using Linear mode (2 bytes per channel) and RRVD, which makes MSbit first, the entire 2 bytes are MSbit first. 0 = Disable: Byte swap is disabled. 1 = Enable: Byte swap is executed. This feature should be used ONLY in Linear mode. It is transparent, for voice applications.
2	LinearTxByteSwap	RW 0x0	Linear Tx Byte Swapping The Byte Swapping feature enables MSByte first out in TDM, so that when using Linear mode (2 bytes per channel) and TRVD, which makes MSbit first, the entire 2 bytes are MSbit first. 0 = Disable: Byte swap is disabled. 1 = Enable: Byte swap is executed. This feature should be used ONLY in linear mode. It is transparent, for voice applications.
1	HdlcAddrMatchMode	RW 0x0	HDLC Address Match Mode When this bit is set to 1, the HDLC Frames will be accepted by the RX according to an Address Matching method. Only addresses that match the addresses defined within the HDLC Mode Address1 and Address2 register and the HDLC Mode Address3 and Address4 register will be passed to the Rx-Buffer. In this case, the N/B fields in the MRCRx Register are also relevant. When this bit is 0, ALL HDLC frames will be accepted except for Null/Broadcast Address, if they are not enabled (as defined in N/B fields at the MRCRx register).
0	IQCMultEn	RW 0x1	IQC Multiplication Enable 0 = Enable: Multiplication is enabled. All four IQCs are active. Each IQC is responsible for specific channels that are configured in the IQC0-3 Channel Responsibility registers). 1 = Disable: Multiplication is disabled. Only IQC0 is active. It is responsible for all channels (IQC0 Channel Responsiblility register is not relevant in this case).

Table 1079:IQC0 Channels Responsibility Register

Offset: 0x000B2894

Bit	Field	Type/InitVal	Description
31:0	Channel_masked_in_IQC0	RW 0x0	Each bit represents a channel. (bit0 = Channel0, bit1=Channel1... bit31=Channel31) When setting bit to 1, it means that the channel that it represents will be masked by IQC0 (This channel will not cause an interrupt). When writing 0 to any bit, it means that the channel that it represents will be active in IQC0.

Table 1080:IQC1 Channels Responsibility Register

Offset: 0x000B2898

Bit	Field	Type/InitVal	Description
31:0	Channel_masked_in_IQC1	RW 0x0	Each bit represents a channel. (bit0 = Channel0, bit1=Channel1... bit31=Channel31) When setting bit to 1, it means that the channel that it represents will be masked by IQC1 (This channel will not cause an interrupt). When writing 0 to any bit, it means that the channel that it represents will be active in IQC1.

Table 1081:IQC2 Channels Responsibility Register

Offset: 0x000B289C

Bit	Field	Type/InitVal	Description
31:0	Channel_masked_in_IQC2	RW 0x0	Each bit represents a channel. (bit0 = Channel0, bit1=Channel1... bit31=Channel31) When setting bit to 1, it means that the channel that it represents will be masked by IQC2 (This channel will not cause an interrupt). When writing 0 to any bit, it means that the channel that it represents will be active in IQC2.

Table 1082:IQC3 Channels Responsibility Register

Offset: 0x000B28A0

Bit	Field	Type/InitVal	Description
31:0	Channel_masked_in_IQC3	RW 0x0	Each bit represents a channel. (bit0 = Channel0, bit1=Channel1... bit31=Channel31) When setting bit to 1, it means that the channel that it represents will be masked by IQC3 (This channel will not cause an interrupt). When writing 0 to any bit, it means that the channel that it represents will be active in IQC3.

Table 1083:HDLC Mode Address1 and Address2 Register

Offset: 0x000B28A4

Bit	Field	Type/InitVal	Description
31:24	Addr2_LSByte	RW 0x0	Least significant Byte of Address 2 This field is used when using 8-bit address or 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used with the BitCompareEn[15:8] field defined in the HDLC Mode Address Filtering register.
23:16	Addr2_MSByte	RW 0x0	Most significant Byte of Address 2 This field is used only when using 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used with the BitCompareEn[7:0] field defined in the HDLC Mode Address Filtering register.
15:8	Addr1_LSByte	RW 0x0	Least significant Byte of Address 1 This field is used when using 8-bit address or 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used with the BitCompareEn[15:8] field defined in the HDLC Mode Address Filtering register.
7:0	Addr1_MSByte	RW 0x0	Most significant Byte of Address 1 This field is used only when using 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used along with the BitCompareEn[7:0] field defined in the HDLC Mode Address Filtering register.

Table 1084:HDLC Mode Address3 and Address4 Register

Offset: 0x000B28A8

Bit	Field	Type/InitVal	Description
31:24	Addr4_LSByte	RW 0x0	Least significant Byte of Address 4 This field is used when using 8-bit address or 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used with the BitCompareEn[15:8] field defined in the HDLC Mode Address Filtering register.

Table 1084:HDLC Mode Address3 and Address4 Register (Continued)

Offset: 0x000B28A8

Bit	Field	Type/InitVal	Description
23:16	Addr4_MSByte	RW 0x0	Most significant Byte of Address 4 This field is used only when using 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used along with the BitCompareEn[7:0] field defined in the HDLC Mode Address Filtering register.
15:8	Addr3_LSByte	RW 0x0	Least significant Byte of Address 3 This field is used when using 8-bit address or 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used with the BitCompareEn[15:8] field defined in the HDLC Mode Address Filtering register.
7:0	Addr3_MSByte	RW 0x0	Most significant Byte of Address 3 This field is used only when using 16-bit address that can be used for receive address recognition in HDLC mode. This address is relevant only when the HdlcAddrMatchMode field in the MCSC Global Configuration Extension register is set to 1. This address is used along with the BitCompareEn[7:0] field defined in the HDLC Mode Address Filtering register.

Table 1085:HDLC Mode Address Filtering Register

Offset: 0x000B28AC

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	BitCompareEn	RW 0x0	Bit Comparison Enable Setting 1 in one of the Bit Comparison Enable bits enables the address comparison for that bit: - For 16-bit LAP-D-like address recognition, write 0xFFFF. - For 8-bit HDLC/LAP-B-like address recognition, write 0xFF00. - For reception of a predefined address group, write 0 to the appropriate bits to disable address comparison on those bits.

A.13.3 Shared Bus to Mbus Bridge

Table 1086:TDMMC Window0 Control Register

Offset: 0x000B8A00

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On theTarget Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1087:TDMMC Window0 Size Register

Offset: 0x000B8A04

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window0 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1088:TDMMC Window1 Control Register

Offset: 0x000B8A08

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.

Table 1088:TDMMC Window1 Control Register (Continued)

Offset: 0x000B8A08

Bit	Field	Type/InitVal	Description
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1089:TDMMC Window1 Size Register

Offset: 0x000B8A0C

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window1 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1090:TDMMC Window2 Control Register

Offset: 0x000B8A10

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1091:TDMMC Window2 Size Register

Offset: 0x000B8A14

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window2 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1092:TDMMC Window3 Control Register

Offset: 0x000B8A18

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending on the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1093:TDMMC Window3 Size Register

Offset: 0x000B8A1C

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window3 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1094:TDMMC Window4 Control Register

Offset: 0x000B8A20

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1095:TDMMC Window4 Size Register

Offset: 0x000B8A24

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window4 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1096:TDMMC Window5 Control Register

Offset: 0x000B8A28

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1097:TDMMC Window5 Size Register

Offset: 0x000B8A2C

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window5 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0's. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1098:TDMMC Window6 Control Register

Offset: 0x000B8A30

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1099:TDMMC Window6 Size Register

Offset: 0x000B8A34

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window6 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0's. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1100:TDMMC Window7 Control Register

Offset: 0x000B8A38

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1101:TDMMC Window7 Size Register

Offset: 0x000B8A3C

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window7 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1102:TDMMC Window8 Control Register

Offset: 0x000B8A40

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1103:TDMMC Window8 Size Register

Offset: 0x000B8A44

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window8 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0's. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1104:TDMMC Window9 Control Register

Offset: 0x000B8A48

Bit	Field	Type/InitVal	Description
31:16	Base Address	RW 0x0	BaseAddress Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1105:TDMMC Window9 Size Register

Offset: 0x000B8A4C

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window9 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1106:TDMMC Window10 Control Register

Offset: 0x000B8A50

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending on the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1107:TDMMC Window10 Size Register

Offset: 0x000B8A54

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window10 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1108:TDMMC Window11 Control Register

Offset: 0x000B8A58

Bit	Field	Type/InitVal	Description
31:16	BaseAddress	RW 0x0	Base Address Used with the size register to set the address window size and location. An address driven by one of the DMAs is considered a window hit when it falls inside the range of BaseAddress+WindowSize.
15:8	Attribute	RW 0x0	Specifies Target Specific Attributes Depending On the Target Interface.
7:4	Reserved	RSVD 0x0	Reserved
3:0	TargetID	RW 0x0	Specifies the Target ID of the unit associated with this window.

Table 1109:TDMMC Window11 Size Register

Offset: 0x000B8A5C

Bit	Field	Type/InitVal	Description
31:16	WindowSize	RW 0x0	Window11 Size Used with the size register to set the address window size and location. Must be programmed from LSb to MSb as a sequence of 1s followed by a sequence of 0s. The number of 1s specifies the size of the window in 64-KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB).
15:0	Reserved	RSVD 0x0	Reserved

Table 1110:TDMMC Window0 High Address Register

Offset: 0x000B8A80

Bit	Field	Type/InitVal	Description
31:0	HighAddress	RW 0x0	High Address Bits [63:32] of the address for Mem are greater than 4 GB.

Table 1111:TDMMC Window1 High Address Register

Offset: 0x000B8A84

Bit	Field	Type/InitVal	Description
31:0	HighAddress	RW 0x0	High Address Bits [63:32] of the address for Mem are greater than 4 GB.

Table 1112:TDMMC Window2 High Address Register

Offset: 0x000B8A88

Bit	Field	Type/InitVal	Description
31:0	HighAddress	RW 0x0	High Address Bits [63:32] of the address for Mem are greater than 4 GB.

Table 1113:TDMMC Window3 High Address Register

Offset: 0x000B8A8C

Bit	Field	Type/InitVal	Description
31:0	HighAddress	RW 0x0	High Address Bits [63:32] of the address for Mem are greater than 4 GB.

Table 1114:TDMMC Window4 High Address Register

Offset: 0x000B8A90

Bit	Field	Type/InitVal	Description
31:0	HighAddress	RW 0x0	High Address Bits [63:32] of the address for Mem are greater than 4 GB.

Table 1115:TDMMC Window5 High Address Register

Offset: 0x000B8A94

Bit	Field	Type/InitVal	Description
31:0	HighAddress	RW 0x0	High Address Bits [63:32] of the address for Mem are greater than 4 GB.

Table 1116:TDMMC Windows Access Protect Register

Offset: 0x000B8B00

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved
23:22	Win11Protect	RW 0x3	Window11 Access Control Same definition as Win0Protect.
21:20	Win10Protect	RW 0x3	Window10 Access Control Same definition as Win0Protect.
19:18	Win9Protect	RW 0x3	Window9 Access Control Same definition as Win0Protect.
17:16	Win8Protect	RW 0x3	Window8 Access Control Same definition as Win0Protect.
15:14	Win7Protect	RW 0x3	Window7 Access Control Same definition as Win0Protect.
13:12	Win6Protect	RW 0x3	Window6 Access Control Same definition as Win0Protect.
11:10	Win5Protect	RW 0x3	Window5 Access Control Same definition as Win0Protect.
9:8	Win4Protect	RW 0x3	Window4 Access Control Same definition as Win0Protect.

Table 1116:TDMMC Windows Access Protect Register (Continued)

Offset: 0x000B8B00

Bit	Field	Type/InitVal	Description
7:6	Win3Protect	RW 0x3	Window3 Access Control Same definition as Win0Protect.
5:4	Win2Protect	RW 0x3	Window2 Access Control Same definition as Win0Protect.
3:2	Win1Protect	RW 0x3	Window1 Access Control Same definition as Win0Protect.
1:0	Win0Protect	RW 0x3	Window0 Access Control If an access violation occurs (e.g. write data to a read only region), the transaction is not driven to the target interface. 0 = No access: No access allowed 1 = Read Only 2 = Reserved 3 = Full access: Full access (read or write)

Table 1117:TDMMC Window0 Enable Register

Offset: 0x000B8B04

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit’s address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1118:TDMMC Window1 Enable Register

Offset: 0x000B8B08

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1119:TDMMC Window2 Enable Register

Offset: 0x000B8B0C

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved

Table 1119:TDMMC Window2 Enable Register (Continued)

Offset: 0x000B8B0C

Bit	Field	Type/InitVal	Description
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1120:TDMMC Window3 Enable Register

Offset: 0x000B8B10

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit’s address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1121:TDMMC Window4 Enable Register

Offset: 0x000B8B14

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit’s address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True

Table 1121:TDMMC Window4 Enable Register (Continued)

Offset: 0x000B8B14

Bit	Field	Type/InitVal	Description
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1122:TDMMC Window5 Enable Register

Offset: 0x000B8B18

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1123:TDMMC Window6 Enable Register

Offset: 0x000B8B1C

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved

Table 1123:TDMMC Window6 Enable Register (Continued)

Offset: 0x000B8B1C

Bit	Field	Type/InitVal	Description
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1124:TDMMC Window7 Enable Register

Offset: 0x000B8B20

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit’s address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1125:TDMMC Window8 Enable Register

Offset: 0x000B8B24

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1126:TDMMC Window9 Enable Register

Offset: 0x000B8B28

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1127:TDMMC Window10 Enable Register

Offset: 0x000B8B2C

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

Table 1128:TDMMC Window11 Enable Register

Offset: 0x000B8B30

Bit	Field	Type/InitVal	Description
The user needs to enable a sub unit (TDM unit, MCSC unit, MCDMA unit) to use the specific address window in its address decoding scheme. For example, if Window0 enbale register bit[5] is 0, than this window will not appear in the TDM unit's address decoding scheme.			
31:5	Reserved	RSVD 0xFF	Reserved
4	McscEn	RW 0x1	Enable access for MCSC Unit to this window. 0 = False 1 = True
3	McdmaEn	RW 0x1	Enable access for MCDMA unit to this window. 0 = False 1 = True
2:1	Reserved	RSVD 0x3	Reserved
0	CxEn	RW 0x1	Reserved Must be set to 0x1.

A.13.4 Time Division Multiplexing (TDM) Control and Configuration

Table 1129:FlexTDM Transmit Dual Port RAM (TDPR)<n> Register (n=0–255)

Offset: TxDPEntry0: 0x000B8000, TxDPEntry1: 0x000B8004...TxDPEntry255: 0x000B83FC

Offset Formula: 0x000B8000+n*0x4: where n (0-255) represents TxDPEntry

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28:27	ERPT	RW 0x0	Extended Repeat This field in addition to the RPT field indicates the number of times (up to 16 times) that the entry is repeated before moving to next entry. 00 = Entry repeated according to RPT field setting 01 = Entry repeated four times + the RPT field setting 10 = Entry repeated eight times + the RPT field setting 11 = Entry repeated 12 times + the RPT field setting
26	FINT	RW 0x0	FlexTDM Interrupt An interrupt is generated if the FTINT bit is set in the current, entry and reset in the last TDM entry that was executed.
25	L	RW 0x0	Last Entry in Frame The TDM read pointer returns to entry 0 or entry 128 (address 0 or 128 of the TDM DPRAM) after reading this entry. Control of which entry is executed next - either 0 or 128 - is achieved through the R2HALF and T2HALF bits in the TCR (see FlexTDM0 Configuration Register (TCR)).
24:23	RPT	RW 0x0	Number of times an Entry is repeated before moving to the next Entry. The FlexTDM repeats execution of this entry according to the value programmed in RPT. 0 = 1x: Entry not repeated (that is, it is executed once). 1 = 2x: Entry repeated once (that is, it is executed 2 times). 2 = 3x: Entry repeated twice (that is, it is executed 3 times). 3 = 4x: Entry repeated three times (that is, it is executed 4 times).
22	Reserved	RW 0x0	Reserved. Must be set to 0x1.
21	Reserved	RW 0x0	Reserved. Must be 0x0.
20:19	Reserved	RW 0x0	Reserved Must be 0x0.
18	Reserved	RW 0x0	Reserved Must be 0x1.

Table 1129:FlexTDM Transmit Dual Port RAM (TDPR)<n> Register (n=0–255) (Continued)

Offset: TxDPEntry0: 0x000B8000, TxDPEntry1: 0x000B8004...TxDPEntry255: 0x000B83FC

Offset Formula: 0x000B8000+n*0x4: where n (0-255) represents TxDPEntry

Bit	Field	Type/InitVal	Description
17:16	Reserved	RW 0x0	Reserved Must be 0x1.
15:8	CH	RW 0x0	Channel Number. 32-channels are supported. This field should hold a number between 0 to 31.
7:0	MASK	RW 0x0	Mask Pattern for the Current Data 0 = Tri-state: TDM transmit data is not driven out (transmit output is tri-stated), receive data is ignored. 255 = Driven out: TDM transmit data is driven out, receive data is processed normally.

Table 1130:FlexTDM Receive Dual Port RAM (RDPR)<n> Register (n=0–255)

Offset: TxDPEntry0: 0x000B8400, TxDPEntry1: 0x000B8404...TxDPEntry255: 0x000B87FC

Offset Formula: 0x000B8400+n*0x4: where n (0-255) represents TxDPEntry

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28:27	ERPT	RW 0x0	Extended Repeat This field in addition to the RPT field indicates the number of times (up to 16 times) that the entry is repeated before moving to next entry. 00 = Entry repeated according to RPT field setting 01 = Entry repeated four times + the RPT field setting 10 = Entry repeated eight times + the RPT field setting 11 = Entry repeated 12 times + the RPT field setting
26	FINT	RW 0x0	FlexTDM Interrupt An interrupt is generated if the FTINT bit is set in the current entry, and reset in the last TDM entry that was executed.
25	L	RW 0x0	Last Entry in Frame The TDM read pointer returns to entry 0 or entry 128 (address 0 or 128 of the TDM DPRAM) after reading this entry. Control of which entry is executed next - either 0 or 128 - is achieved through the R2HALF and T2HALF bits in the TCR (see FlexTDM0 Configuration Register (TCR)).

Table 1130:FlexTDM Receive Dual Port RAM (RDPR)<n> Register (n=0–255) (Continued)

Offset: TxDPEntry0: 0x000B8400, TxDPEntry1: 0x000B8404...TxDPEntry255: 0x000B87FC

Offset Formula: 0x000B8400+n*0x4: where n (0-255) represents TxDPEntry

Bit	Field	Type/InitVal	Description
24:23	RPT	RW 0x0	Number of times an Entry is repeated before moving to the next Entry. The FlexTDM repeats execution of this entry according to the value programmed in RPT. 0 = 1x: Entry not repeated (that is, it is executed once). 1 = 2x: Entry repeated once (that is, it is executed 2 times). 2 = 3x: Entry repeated twice (that is, it is executed 3 times). 3 = 4x: Entry repeated three times (that is, it is executed 4 times).
22	Reserved	RW 0x0	Reserved Must be 0x1.
21	Reserved.	RW 0x0	Reserved. Must be 0x0.
20:19	Reserved	RW 0x0	Reserved Must be 0x0.
18	Reserved	RW 0x0	Reserved Must be 0x1.
17:16	Reserved	RW 0x0	Reserved Must be 0x1.
15:8	CH	RW 0x0	Channel Number. 32-channels are supported. This field should hold a number between 0 to 31.
7:0	MASK	RW 0x0	Mask Pattern for the Current Data 0 = Tri-state: TDM transmit data is not driven out (transmit output is tri-stated), receive data is ignored. 255 = Driven out: TDM transmit data is driven out, receive data is processed normally.

Table 1131:FlexTDM0 Transmit Read Pointer Register

Offset: 0x000B8800

Bit	Field	Type/InitVal	Description
31:11	Reserved	RSVD 0x0	Reserved
10:0	TxRdPointer	RO 0x0	Synchronized Sampling of TX Read Pointer

Table 1132:FlexTDM0 Receive Read Pointer Register

Offset: 0x000B8804

Bit	Field	Type/InitVal	Description
31:11	Reserved	RSVD 0x0	Reserved
10:0	RxRdPointer	RO 0x0	Synchronized Sampling of RX Read Pointer

Table 1133:FlexTDM0 Configuration Register (TCR)

Offset: 0x000B8808

Bit	Field	Type/InitVal	Description
31	TEN	RW 0x0	Enable FlexTDM 0 = Disable: FlexTDM disabled. Transmit output is Z. 1 = Enable: FlexTDM enabled.
30:29	TDIAG	RW 0x0	TDM Diagnostic **NOTE:** Proper operation of the echo and loopback modes requires that an identical clock is supplied to the TDM's transmit and receive sections. 0 = Normal Operation: The received input is connected to the FlexTDM receive pin (TRXD), and the transmit output is connected to the FlexTDM transmit pin (TTXD). 1 = Echo: TDM receive input is echoed on the TDM output with one clock delay. The received bit stream is processed normally according to DPRAM programming. 2 = Loopback: Transmit data is driven on TDM output, as in normal operation, and is also connected internally to the TDM receive line. The received bit stream is processed normally according to DPRAM programming. Transactions on TRXD are not seen by the FlexTDM. 3 = Internal Loopback: (transmit output is inactive). TDM transmit output is internally connected to the TDM receive input. This mode is useful for TDM loopback testing without affecting the external lines.
28:27	Reserved	RW 0x0	Reserved. Must be 0x0.
26:25	Reserved	RW 0x0	Reserved. Must be 0x0.
24	Reserved	RW 0x0	Reserved. Must be 0x0.
23	Reserved	RW 0x0	Reserved. Must be 0x1.

Table 1133:FlexTDM0 Configuration Register (TCR) (Continued)

Offset: 0x000B8808

Bit	Field	Type/InitVal	Description
22	Reserved	RSVD 0x0	Reserved
21	DE	RW 0x0	Driving Edge. 0 = Tx rising edge: Transmit data is sent on the rising edge, receive data is sampled on the falling edge of the clock. 1 = Tx falling edge: Transmit data is sent on the falling edge, receive data is sampled on the rising edge of the clock.
20	SE	RW 0x0	Sync Edge (for Fsync used in TX and RX) When Frame Sync is sourced from an external input: 0 = Sync signals are sampled on the falling edge of the clock. 1 = Sync signals are sampled on the rising edge of the clock. When Frame Sync is generated internally within the device: 0 = Sync signals are driven on the falling edge of the clock. 1 = Sync signals are driven on the rising edge of the clock.
19:17	Reserved	RW 0x0	Reserved. Must be 0x0.
16	TR2HALF	RW 0x0	Transmit Return to Half Used for dynamic programming of the TDM transmit frame. 0 = Return to 0: After sync or last, the FlexTDM transmit read pointer returns to the entry 0 in the Tx DPRAM. 1 = Return to 128: After sync or last, the FlexTDM transmit read pointer returns to the entry 128 in the Tx DPRAM.
15	RR2HALF	RW 0x0	RR2HALF Receive Return to Half. This bit is used for dynamic programming of the TDM receive frame. 0 = Return to 0: After sync or last, the FlexTDM receive read pointer returns to the entry 0 in the Rx DPRAM. 1 = Return to 128: After Sync or Last, the FlexTDM receive read pointer returns to the entry 128 in the Rx DPRAM.
14	TTM	RW 0x0	TDM Transparent Mode TTM must be set to 0x0 for proper operation.
13:11	TDTR	RW 0x4	TDM Delay for Transparent Receive TDTR must be set to 0x4 for proper operation.
10:8	TDTT	RW 0x5	TDM Delay for Transparent Transmit TDTT must be set to 0x5 for proper operation.
7	Reserved	RW 0x1	Reserved. Must be 0x1.

Table 1133:FlexTDM0 Configuration Register (TCR) (Continued)

Offset: 0x000B8808

Bit	Field	Type/InitVal	Description
6:5	Reserved	RW 0x0	Reserved. Must be 0x0.
4:2	Reserved	RW 0x0	Reserved. Must be 0x0.
1:0	Reserved	RSVD 0x0	Reserved. Must be 0x0.

Table 1134:TDM Clock and Sync Control Register

Offset: 0x000B881C

Bit	Field	Type/InitVal	Description
31	ProgTdmSlicRst	RW 0x0	Programmable Reset: The programmed value of this field is connected directly to the SLIC-Reset-Input.
30:27	Reserved	RSVD 0xF	Reserved
26:25	FsyncLength	RW 0x0	Fsync Length Modes: 0 = Short1: Fsync Length is 1-bit 1 = Long2: Fsync Length is 2-bits 2 = Long8: Fsync Length is 8-bits 3 = Long16: Fsync Length is 16-bits
24	Reserved	RW 0x1	Reserved Must be 0x1.
23:22	Reserved	RSVD 0x3	Reserved
21	TdmDivBypass	RW 0x1	TDM Refclk Divider Bypass: Bypass Control for the TDM RefClock-Divider that is used in Output Mode. The Internally-generated Clock (Refclk) is being divided before taken out to the TDM-interface. If the user would like to avoid this division, he may configure the bypass to 1. 0 = Active: Clock divider is active. 1 = Bypassed: Clock divider is bypassed.
20	Reserved	RW 0x1	Reserved The value of this field must be the same as the <TdmDivBypass> field.

Table 1134:TDM Clock and Sync Control Register (Continued)

Offset: 0x000B881C

Bit	Field	Type/InitVal	Description
19	FsyncPolarity	RW 0x0	This bit influences the Fsync polarity of RX/TX in Input and Output Modes. 0 = Active-High: Fsync is Active-High. 1 = Active-Low: Fsync is Active-Low.
18:12	TdmRefclkDiv	RW 0x0	TDM Refclk division When working in Output-clock mode, an internal clock is generated from the unit's Refclk. Refclk is driven from the SoC according to the TdmClkSrcSel configuration Field. Refclk is divided, and the divided clock is used within the TDMMC (RX and TX parts) and sent out from the unit as the output clock. This field can be programmed to give six division-ratios. All other combination are reserved. **NOTE:** When TdmDivBypass field in this register is set, these configurations are bypassed. 1 = div64: division by 64. 2 = div32: division by 32. 4 = div16: division by 16. 8 = div8: division by 8. 16 = div4: division by 4. 32 = div2: division by 2.
11:4	Reserved	RW 0x0	Reserved The value of this field must be the same as the <TdmRefclkDiv> field.
3	TdmClockOEn	RW 0x1	TDM Clock Output Enable: 0 = Internal: Clock is generated internally within the device, and goes out through its output port. 1 = External: Clock source is from an external input.
2	Reserved	RW 0x1	Reserved The value of this field must be the same as “TdmClockOEn”.
1	TdmFsyncOEn	RW 0x1	TDM Fsync Output Enable 0 = Internal: Frame Sync is generated internally within the device, and goes out through its output port. 1 = External: Frame Sync source is from an external input.
0	Reserved	RW 0x1	Reserved The value of this field must be the same as the <TdmFsyncOEn> field.

Table 1135:TDM Clock Divider Control Register

Offset: 0x000B8820

Bit	Field	Type/InitVal	Description
31	Reserved	RW 0x0	Reserved. Must be 0x1 for proper operation.
30:8	Reserved	RSVD 0x0	Reserved
7:0	Reserved	RSVD 0x0	Reserved

Table 1136:TDM Reserved Clock Divider Control Register

Offset: 0x000B8824

Bit	Field	Type/InitVal	Description
31	Reserved	RW 0x0	Reserved. Must be 0x1 for proper operation.
30:8	Reserved	RSVD 0x0	Reserved
7:0	Reserved	RSVD 0x0	Reserved

A.13.5 Time Division Multiplexing Interrupt Controller

Table 1137:TDMMC Top Cause Register

Offset: 0x000B8C00

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28	McscSum	RO 0x0	Summary of the Masked Cause Vector from the MCSC Global Interrupt Cause Register. This bit is read only, since the cleaning of the cause is made through the cause register itself.
27	ErrCauseSum	RO 0x0	Summary of masked indications from Error Cause Register
26:19	Reserved	RSVD 0x0	Reserved
18:7	Reserved	RSVD 0x0	Reserved

Table 1137:TDMMC Top Cause Register (Continued)

Offset: 0x000B8C00

Bit	Field	Type/InitVal	Description
6	FuncCauseSum	RO 0x0	Summary of the Masked Cause Vector from the Functional Cause Register. This bit is read only, since the cleaning of the cause is made through the cause register itself.
5:0	Reserved	RSVD 0x0	Reserved

Table 1138:TDMMC Functional Cause Register

Offset: 0x000B8C40

Bit	Field	Type/InitVal	Description
31:26	Reserved	RSVD 0x0	Reserved
25	IQNE0	RW0C 0x0	Interrupt Queue Not Empty (IQC0 - main IQC) This interrupt indicates to the CPU that the Interrupt Queue Controller has written an interrupt entry to the Interrupt Queue in the external memory, and the CPU should handle this entry. Note that during the time between setting this bit and until it is handled by the CPU, more interrupt entries might be added to the queue. This Interrupt cause exist also in 'MCSC Global Interrupt Cause" Register, IQNE field.
24	IQNE1	RW0C 0x0	Interrupt Queue Not Empty (IQC1) This interrupt indicates to the CPU that the Interrupt Queue Controller has written an interrupt entry to the Interrupt Queue in the external memory, and the CPU should handle this entry. Note that during the time between setting this bit and until it is handled by the CPU, more interrupt entries might be added to the queue.
23	IQNE2	RW0C 0x0	Interrupt Queue Not Empty (IQC2) This interrupt indicates to the CPU that the Interrupt Queue Controller has written an interrupt entry to the Interrupt Queue in the external memory, and the CPU should handle this entry. Note that during the time between setting this bit and until it is handled by the CPU, more interrupt entries might be added to the queue.
22	IQNE3	RW0C 0x0	Interrupt Queue Not Empty (IQC3) This interrupt indicates to the CPU that the Interrupt Queue Controller has written an interrupt entry to the Interrupt Queue in the external memory, and the CPU should handle this entry. Note that during the time between setting this bit and until it is handled by the CPU, more interrupt entries might be added to the queue.

Table 1138:TDMMC Functional Cause Register (Continued)

Offset: 0x000B8C40

Bit	Field	Type/InitVal	Description
21	IQOR0	RW0C 0x0	Interrupt Queue Over Run (IQC0 - main IQC) This interrupt indicates that the interrupt queue in the external memory has suffered an overrun event. Such an event can happen when the CPU does not process the interrupt entries fast enough, and the interrupt queue gets full. The loss of an interrupt can have severe implications on the management of the channels, and should be avoided by correct system design. This Interrupt cause exist also in 'MCSC Global Interrupt Cause" Register, IQOR field.
20	IQOR1	RW0C 0x0	Interrupt Queue Over Run (IQC1) This interrupt indicates that the interrupt queue in the external memory has suffered an overrun event. Such an event can happen when the CPU does not process the interrupt entries fast enough, and the interrupt queue gets full. The loss of an interrupt can have severe implications on the management of the channels, and should be avoided by correct system design.
19	IQOR2	RW0C 0x0	Interrupt Queue Over Run (IQC2) This interrupt indicates that the interrupt queue in the external memory has suffered an overrun event. Such an event can happen when the CPU does not process the interrupt entries fast enough, and the interrupt queue gets full. The loss of an interrupt can have severe implications on the management of the channels, and should be avoided by correct system design.
18	IQOR3	RW0C 0x0	Interrupt Queue Over Run (IQC3) This interrupt indicates that the interrupt queue in the external memory has suffered an overrun event. Such an event can happen when the CPU does not process the interrupt entries fast enough, and the interrupt queue gets full. The loss of an interrupt can have severe implications on the management of the channels, and should be avoided by correct system design.
17	SLIC7Int	RW0C 0x0	External Interrupt from SLIC7
16	SLIC6Int	RW0C 0x0	External Interrupt from SLIC6
15	SLIC5Int	RW0C 0x0	External Interrupt from SLIC5
14	SLIC4Int	RW0C 0x0	External Interrupt from SLIC4
13	SLIC3Int	RW0C 0x0	External Interrupt from SLIC3

Table 1138:TDMMC Functional Cause Register (Continued)

Offset: 0x000B8C40

Bit	Field	Type/InitVal	Description
12	SLIC2Int	RW0C 0x0	External Interrupt from SLIC2
11	SLIC1Int	RW0C 0x0	External Interrupt from SLIC1
10	SLIC0Int	RW0C 0x0	External Interrupt from SLIC0
9	TxVoiceInterrupt	RW0C 0x0	This is a periodic Interrupt for Voice application in TX path. TX Voice Interrupt will rise for every configurable number of Fsyncs, as defined in "Voice Periodic Interrupt Control" Register.
8	RxVoiceInterrupt	RW0C 0x0	This is a periodic Interrupt for Voice application in RX path. RX Voice Interrupt will rise for every configurable number of Fsyncs, as defined in "Voice Periodic Interrupt Control" Register.
7	FlexTdmTxSyncLoss	RW0C 0x0	FlexTDM Tx Synchronization Loss indication.
6	FlexTdmTxInterrupt	RW0C 0x0	Interrupt indication from FlexTDM Tx path according to configuration in "FlexTDM Transmit Dual Port RAM" register.
5:4	Reserved	RSVD 0x0	Reserved
3	FlexTdmRxSyncLoss	RW0C 0x0	FlexTDM Rx Synchronization Loss Indication.
2	FlexTDMRxInterrupt	RW0C 0x0	FlexTDM0 Rx Interrupt Interrupt indication from FlexTDM Rx path according to configuration in "FlexTDM Receive Dual Port RAM" register.
1:0	Reserved	RSVD 0x0	Reserved

Table 1139:TDMMC Error Cause Register

Offset: 0x000B8C44

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4	McdmaParityErr	RW0C 0x0	MCDMA internal data path Parity Error Summary

Table 1139:TDMMC Error Cause Register (Continued)

Offset: 0x000B8C44

Bit	Field	Type/InitVal	Description
3	McscParityErr	RW0C 0x0	MCSC internal data path Parity Error Summary
2	TdmTxParityErr	RW0C 0x0	TDM TX internal data Parity Error Summary
1	TdmRxParityErr	RW0C 0x0	TDM RX internal data path Parity Error Summary
0	TDMMCParityErr	RW0C 0x0	Multi Channel TDM internal data path Parity Error Summary

Table 1140:TDMMC Top Mask Register

Offset: 0x000B8C80

Bit	Field	Type/InitVal	Description
31:0	MaskiBits	RW 0x0	Each masking bit serves as a masking bit for the cause bit in the TDMMC Top Cause Register. 0 = Enable: Mask Enabled The interrupt is activated due to this cause bit. 1 = Disable: The interrupt is NOT masked.

Table 1141:TDMMC Output Sync Bit Count Register

Offset: 0x000B8C8C

Bit	Field	Type/InitVal	Description
31:16	FsyncBitcount	RW 0x0	This field contains the number of clocks (bits) between two FSyncs that are generated in output mode. This number should be programmed in accordance with DPRAM programming (RPT and ERPT): Number of bits (in case of 8-bits per entry) = 8x(Number_of_Entries)x[(RPT+1)+ERPTx4]. If the value of this field is cleared to 0 (default value), the output sync signal is not active. In all other cases, the number set in this field is the number of clocks between two Fsync pulses. Once this register is programmed to a value other than 0, the output Fsync signal will be activated according to the programmed rate. The user should program this register to the relevant value after all Fsync parameters are set (sync-inversion/sync-length) since this programming enables the output sync generation. To program the sync to another value, after it has been already programmed, first clear this register (program to 0) and then program the new value. It is strongly recommended to activate TEN field in TCR register only after the Fsync output signal is active.
15:0	Reserved	RW 0x0	Reserved. The user should program this field to the same value as in FsyncBitcoount field.

Table 1142:Voice Periodic Interrupt Control Register

Offset: 0x000B8C90

Bit	Field	Type/InitVal	Description
31:24	TxVoicePeriodIntGap	RW 0xFF	TX Voice Periodic Interrupt First Gap: The Number of Fsync Pulses Counted Before the First Interrupt Appears for TX. 0xFF = Wait, No count.
23:16	RxVoicePeriodIntGap	RW 0xFF	RX Voice Periodic Interrupt First Gap: The Number of Fsync Pulses Counted Before the First Interrupt Appears for RX. 0xFF = Wait, No count.
15:8	TxVoicePeriodIntLen	RW 0xFF	Tx Voice Periodic Interrupt Length: The Number of Required Fsync Pulses Between Periodic Interrupts for TX 0xFF = Disable
7:0	RxVoicePeriodIntLen	RW 0xFF	Rx Voice Periodic Interrupt Length: The Number of Required Fsync Pulses Between Each Periodic Interrupt for RX. 0xFF = Disable

Table 1143:TDMMC Functional Mask Register

Offset: 0x000B8CC0

Bit	Field	Type/InitVal	Description
31:0	MaskBits	RW 0x0	Each masking bit serves as a masking bit for the cause bit in the TDMMC Cause Register. 0 = Enable: Mask Enabled Interrupt is not activated due to this cause bit. 1 = Disable: Interrupt is NOT masked.

Table 1144:TDMMC Error Mask Register

Offset: 0x000B8CC4

Bit	Field	Type/InitVal	Description
31:0	ErrMaskBits	RW 0x0	This mask affects the Error interrupt. Each masking bit serves as a masking bit for the cause bit in the TDMMC Error Cause Register. 0 = Enable: Mask Enabled Interrupt is not activated due to this cause bit. 1 = Disable: Interrupt is NOT masked.

Table 1145:TDMMC Data Delay and Clock Control Register

Offset: 0x000B8CD0

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	Reserved	RSVD 0x0	Reserved
13:12	TdmmcAcTimFix	RW 0x0	AC-Timing Fix in Adding delay for AC-Timing purpose in TXD and FSYNC output pins. 0 = Delay1: Adding Delay of one to two core clock cycles. 1 = NoDelay: Using the Original signals (No Delay fix). 2 = Delay2: Adding Delay of two to three core clock cycles 3 = Delay3: Adding Delay of three to four core clock cycles.
11:2	Reserved	RSVD 0x1	Reserved
1	ClockOutEn	RW 0x0	Clock Output Enable (used in TX and RX) This bit will enable the activation of TDMMC-Line Clock when TDMMC is working in output mode (master mode, when the clock is generated by TDMMC). 0 = Disable: Clock is not active. 1 = Enable: Clock is active.
0	Reserved	RSVD 0x0	Reserved

Table 1146:TDMMC Plus Minus Delay Control for FsyncOut Register

Offset: 0x000B8CD4

Bit	Field	Type/InitVal	Description
31	RestartCalcSyncDelay	RW 0x0	Fsync delay out - restart calculation The restart delay calculation bit is used for restating the delay calculation. The User should write 1 to this bit and then clear it in order to restart the calculation. This bit must be used whenever a new sync delay calculation is executed.

Table 1146:TDMMC Plus Minus Delay Control for FsyncOut Register (Continued)

Offset: 0x000B8CD4

Bit	Field	Type/InitVal	Description
30	SyncDelayOutPolarity	RW 0x1	Fsync Delay out Polarity: PLUS/MINUS Example: If Fsync appears on b0 of RXD (when SyncDelayOutVal=0x0): Using SyncDelayOutPolarity=1, SyncDelayOutVal=0x1 will make Fsync appear on b-1 of RXD. Using SyncDelayOutPolarity=0, SyncDelayOutVal=0x1 will make Fsync appear on b+1 of RXD. **NOTE:** The Fsync appears one TDM-Line-Clock cycle before the first data-bit in RXD/TXD Frame, when no delay is executed, under the following conditions: In Output Mode (when Fsync is generated by the TDMMC) and when Sync and Data are driven at the same clock edge (SE and SE fields are programmed in "FlexTDM0 Configuration Register (TCR)"). 0 = Plus: The delay is in the positive direction, meaning that the Fsync will move forward. 1 = Minus: The delay is in the negative direction, meaning that the Fsync will move backwards.
29:16	SyncDelayOutVal	RW 0x0	Fsync Delay out Value: Absolute number of delay in TDM-Line-Clock cycles.
15:0	Reserved	RSVD 0x4000	Reserved

Table 1147:TDMMC Plus Minus Delay Control for FsyncIn Register

Offset: 0x000B8CD8

Bit	Field	Type/InitVal	Description
31	RestartCalcSyncInDelay	RW 0x0	Fsync delay in - restart calculation The restart delay calculation bit is used for restarting the delay calculation. The User should write 1 to this bit and then clear it in order to restart the calculation. This bit must be used whenever a new sync delay calculation is executed.

Table 1147:TDMMC Plus Minus Delay Control for FsyncIn Register (Continued)

Offset: 0x000B8CD8

Bit	Field	Type/InitVal	Description
30	SyncDelayInPolarity	RW 0x1	Fsync Delay In Polarity: PLUS/MINUS Example: If Fsync appears on b0 of RXD at the I/F (when SyncDelayInVal=0x0): Using SyncDelayInPolarity=1, SyncDelayInVal=0x1 will make Fsync for TDMMC appear on b-1 of RXD. Using SyncDelayInPolarity=0, SyncDelayInVal=0x1 will make Fsync for TDMMC appear on b+1 of RXD. 0 = Plus: The delay is in the positive direction, meaning that the sync will move forward. 1 = Minus: The delay is in the negative direction, meaning that the sync will move backwards.
29:16	SyncDelayInVal	RW 0x0	Fsync Delay In Value: Absolute number of delay in TDM-Line-Clock cycles.
15:0	Reserved	RSVD 0x4000	Reserved

A.14 Secure Digital Input-Output (SDIO) Registers

The following table provides a summarized list of all registers that belong to the SDIO, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1148:Summary Map Table for the SDIO Registers

Register Name	Offset	Table and Page
DMA Buffer Address 16 LSB Register	0x000D4000	Table 1149, p. 1361
DMA Buffer Address 16 MSB Register	0x000D4004	Table 1150, p. 1361
Data Block Size Register	0x000D4008	Table 1151, p. 1362
Data Block Count Register	0x000D400C	Table 1152, p. 1362
Argument in Command 16 LSB Register	0x000D4010	Table 1153, p. 1362
Argument in Command 16 MSB Register	0x000D4014	Table 1154, p. 1363
Transfer Mode Register	0x000D4018	Table 1155, p. 1363
Command Register	0x000D401C	Table 1156, p. 1365
Response Halfword 0 Register	0x000D4020	Table 1157, p. 1366
Response Halfword 1 Register	0x000D4024	Table 1158, p. 1366
Response Halfword 2 Register	0x000D4028	Table 1159, p. 1366
Response Halfword 3 Register	0x000D402C	Table 1160, p. 1367
Response Halfword 4 Register	0x000D4030	Table 1161, p. 1367
Response Halfword 5 Register	0x000D4034	Table 1162, p. 1367
Response Halfword 6 Register	0x000D4038	Table 1163, p. 1367
Response Halfword 7 Register	0x000D403C	Table 1164, p. 1368
16-bit Data Word Accessed by CPU Register	0x000D4040	Table 1165, p. 1368
CRC7 of I Response Register	0x000D4044	Table 1166, p. 1368
Host Present State 16 LSB Register	0x000D4048	Table 1167, p. 1369
Host Control Register	0x000D4050	Table 1168, p. 1370
Data Block Gap Control Register	0x000D4054	Table 1169, p. 1372
Clock Control Register	0x000D4058	Table 1170, p. 1374
Software Reset Register	0x000D405C	Table 1171, p. 1374
Normal Interrupt Status Register	0x000D4060	Table 1172, p. 1375
Error Interrupt Status Register	0x000D4064	Table 1173, p. 1376
Normal Interrupt Status Enable Register	0x000D4068	Table 1174, p. 1378
Error Interrupt Status Enable Register	0x000D406C	Table 1175, p. 1379
Normal Interrupt Status Interrupt Enable Register	0x000D4070	Table 1176, p. 1380
Error Interrupt Status Interrupt Enable Register	0x000D4074	Table 1177, p. 1381
Auto CMD12 Interrupt Status Register	0x000D4078	Table 1178, p. 1382
Current Number of Bytes Remaining in Data Block Register	0x000D407C	Table 1179, p. 1383
Current Number of Data Blocks Left to Be Transferred Register	0x000D4080	Table 1180, p. 1383

Copyright © 2014 Marvell

Table 1148:Summary Map Table for the SDIO Registers (Continued)

Register Name	Offset	Table and Page
Argument in Auto Cmd12 Command 16 LSB Transferred Register	0x000D4084	Table 1181, p. 1383
Argument in Auto Cmd12 Command 16 MSB Transferred Register	0x000D4088	Table 1182, p. 1384
Index of Auto Cmd12 Commands Transferred Register	0x000D408C	Table 1183, p. 1384
Auto Cmd12 Response Halfword 0 Register	0x000D4090	Table 1184, p. 1384
Auto Cmd12 Response Halfword 1 Register	0x000D4094	Table 1185, p. 1385
Auto Cmd12 Response Halfword 2 Register	0x000D4098	Table 1186, p. 1385
Mbus Control Low Register	0x000D4100	Table 1187, p. 1385
Mbus Control High Register	0x000D4104	Table 1188, p. 1385
Window0 Control Register	0x000D4108	Table 1189, p. 1386
Window0 Base Register	0x000D410C	Table 1190, p. 1387
Window1 Control Register	0x000D4110	Table 1191, p. 1387
Window1 Base Register	0x000D4114	Table 1192, p. 1387
Window2 Control Register	0x000D4118	Table 1193, p. 1388
Window2 Base Register	0x000D411C	Table 1194, p. 1388
Window3 Control Register	0x000D4120	Table 1195, p. 1388
Window3 Base Register	0x000D4124	Table 1196, p. 1389
Clock Divider Value Register	0x000D4128	Table 1197, p. 1389
Address Decoder Error Register	0x000D412C	Table 1198, p. 1389
Address Decoder Error Mask Register	0x000D4130	Table 1199, p. 1390

Table 1149:DMA Buffer Address 16 LSB Register

Offset: 0x000D4000

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	DmaAddrLo	RW 0x0	16 LSB of the DMA system buffer starting byte address, word-aligned.

Table 1150:DMA Buffer Address 16 MSB Register

Offset: 0x000D4004

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	DmaAddrHi	RW 0x0	16 MSB of the DMA system buffer starting byte address.

Table 1151:Data Block Size Register

Offset: 0x000D4008

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:12	RSVD	RO 0x0	Reserved Read only.
11:0	BlockSize	RW 0x0	This register specifies the block size for data transfers. 0 = If <BlockCount> =0, it is an infinite transfer. The block count should be 0 if <BlockSize> = 0. 1 = 1 byte . . . 2048 = 2048 bytes The current value of this <BlockSize> field is reflected in Current Number of Bytes Remaining n Data Block Register.

Table 1152:Data Block Count Register

Offset: 0x000D400C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	BlockCount	RW 0x0	This field holds the initial value of the number of blocks. It is not decremented during the transfer. Instead, the remaining blocks left to send are listed in the Current Number of Bytes Remaining in Data Block Register (offset: 0x8007C) 0 = Infinite block count until the Host driver stops the transfer. If <BlockSize> is 0, it is an infinite transfer. If <BlockSize> is not 0, the Host controller will act as if it transferred multiple blocks, with the block count infinite. 1 = 1 block . . . 65535 = 65535 blocks

Table 1153:Argument in Command 16 LSB Register

Offset: 0x000D4010

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved

Table 1153:Argument in Command 16 LSB Register (Continued)

Offset: 0x000D4010

Bit	Field	Type/InitVal	Description
15:0	ArgLow	RW 0x0	16 LSB of the command argument. This value is inserted into the 48-bit command token (bits [23:8]).

Table 1154:Argument in Command 16 MSB Register

Offset: 0x000D4014

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	ArgHigh	RW 0x0	16 MSB of the command argument. This value is inserted into the 48-bit command token bits[39:24].

Table 1155:Transfer Mode Register

Offset: 0x000D4018

Bit	Field	Type/InitVal	Description
rcp controls the pause when recording.			
31:16	Reserved	RSVD 0x0	Reserved
15:7	RSVD	RO 0x0	Reserved Read only.
6	HostXferMode	RW 0x0	Host Transfer Mode 0 = DMA 1 = SW Write
5	StopClkEn	RW 0x0	This bit is set to 1 to stop the SD clocks in the following cases, even if clk_en is set to 1. 1) A command with no response: stop 16 clocks after the host command end bit 2) A command with response without data: stop 16 clocks after the card response end bit. 3) A read data transaction: 16 clocks after the end bit of the last data block. 4) A write data transaction: 16 clocks after the CRC status token returned by card for last data block. If this bit is set to 0, SDCLK is always active 0 = SDClocksActive: SD clocks are always active 1 = StopSDClocks: Stops the SD clocks, even clk_en is set to 1.

Table 1155:Transfer Mode Register (Continued)

Offset: 0x000D4018

Bit	Field	Type/InitVal	Description
4	DataXferTowardHost	RW 0x0	Data transfer direction selection This bit defines the direction of DAT line data transfer. The bit is set to 1 by Host Driver to transfer data from SD card to SD host controller, and it is set to 0 for all other commands. 0 = OtherCommands: All other commands 1 = XferDataToSDHost: Transfers data from the SD card to the SD host controller
3	IntChkEn	RW 0x0	In case of SDIO, if this bit is set to 1, the Host controller checks for interrupts on DAT[1] in both 1-bit an 4-bits mode. 0 = NoInterruptCheck: The Host controller does not check for interrupts on DAT[1]. 1 = CheckInterrupts: The Host controller checks for interrupts on DAT[1] in both 1-bit an 4-bits mode.
2	AutoCMD12En	RW 0x0	Multiple block transfer for memory requires CMD12 to stop the transaction. When this bit is 1, Host controller automatically issues CMD12 when the last block transfer is completed. If this bit is set to 0, the software is responsible for issuing the CMD12 to stop the transfer, and soft reset the host controller. 0 = HostIssuesCMD12: Host controller automatically issues CMD12 when the last block transfer is completed. 1 = SWIssuesCMD12: The software must issue the CMD12 to stop the transfer, and soft reset the host controller.
1	HwWrDataEn	RW 0x0	When the Host controller hardware receives response from the card, it has a option to initiate a write data transfer to the card or to wait for the software to enable the write data transfer by writing 1 to bit[0] of this register. For read data transfer, the hardware always looks for the start bit of the read data from the card. 0 = HWInitWrDataXfer: The hardware always initiates the write data transfer after receiving a response, regardless whether there is an error response or not. 1 = SWInitWrDataXfer: The hardware waits for the software to initiate the write data transfer by writing to bit[0] of this register.
0	SwWrDataStart	RW 0x0	Write 1 to this bit after software decode of the response initiates the write data transfer. 0 = NoTransfer: No write data transfer. 1 = InitWrDataXfer: After software decodes of the response, initiates the write data transfer.

Table 1156:Command Register

Offset: 0x000D401C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	RSVD	RO 0x0	Reserved Read only.
13:8	CmdIndex	RW 0x0	Command index These bits will be inserted into Command token bits[45:40] of the response pattern (for more details, see the SD Memory Card Specifications).
7	UnexpectedRespEn	RW 0x0	Enable hardware to detects unexpected response from device
6	RSVD	RO 0x0	Reserved Read only.
5	DataPresent	RW 0x0	This bit is set to 1 to indicate that data is present and will be transferred using the DATA line. It is set to 0 for the followings: 1) Commands using only CMD lines 2) Commands with no data transfer, but using busy signal on DAT[0] line (for example, CMD38). 0 = NoDataTransfer: 1) Commands using only CMD lines OR 2) Commands with no data transfer, but using busy signal on DAT[0] line (for example, CMD38). 1 = DataAwaitsTransfer: Indicates that data is present and will be transferred using the DATA line.
4	CmdIndexChkEn	RW 0x0	Command index check enable. If this bit is set to 1, the Host controller checks the index field in the response to see if it has the same value as the command index. If the value does not match, it is reported as a Command Index Error. 0 = NoCheck: Host controller does not check the index field. 1 = HCChkIndex: Host controller checks the index field.
3	CmdCrcChkEn	RW 0x0	If this bit is set to 1, Host controller checks the CRC field in the response. If an error is detected, it is reported as a Command CRC error. The number of bits checked by CRC field value varies according to the length of response. 0 = NoCheck: No checking of CRC. 1 = HCCrcChk: Host controller checks the CRC field in the response. If an error is detected, it is reported as a Command CRC error.
2	DataCrc16ChkEn	RW 0x0	Data CRC16 Check Enable 0 = DisableCrc16Chk: Disable read data crc16 check. 1 = EnableCrc16Chk: Enable read data crc16 check.

Table 1156:Command Register (Continued)

Offset: 0x000D401C

Bit	Field	Type/InitVal	Description
1:0	RespType	RW 0x0	Response type select. Note: CRC field for R3 and R4 is expected to be all 1 bits. CRC check should be disabled for these response types. 0 = NoResponse: No response. 1 = 136BitResponse: Response length is 136 bits. 2 = 48BitResponse: Response length is 48 bits. 3 = 48BitResponseChkBusy: Response length is 48 bits and check busy after response.

Table 1157:Response Halfword 0 Register

Offset: 0x000D4020

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Resp0	RO 0x0	1) For 48-bit response token: This register contains bits[45:30] of the response token. 2) For 136-bit response token: This register contains bits[133:118] of the response token.

Table 1158:Response Halfword 1 Register

Offset: 0x000D4024

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Resp1	RO 0x0	1) For 48-bit response token: This register contains bits[29:14] of the response token. 2) For 136-bit response token: This register contains bits[117:102] of the response token.

Table 1159:Response Halfword 2 Register

Offset: 0x000D4028

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Resp2	RO 0x0	1) For 48-bit response token: This register contains bits[13:8] of the response token. 2) For 136-bit response token: This register contains bits[101:86] of the response token.

Table 1160: Response Halfword 3 Register

Offset: 0x000D402C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Resp3	RO 0x0	1) For 48-bit response token: Not relevant. 2) For 136-bit response token: This register contains bits[85:70] of the response token.

Table 1161: Response Halfword 4 Register

Offset: 0x000D4030

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Resp4	RO 0x0	1) For 48-bit response token: Not relevant. 2) For 136-bit response token: This register contains bits[69:54] of the response token.

Table 1162: Response Halfword 5 Register

Offset: 0x000D4034

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Resp5	RO 0x0	1) For 48-bit response token: Not relevant. 2) For 136-bit response token: This register contains bits[53:38] of the response token.

Table 1163: Response Halfword 6 Register

Offset: 0x000D4038

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	Resp6	RO 0x0	1) For 48-bit response token: Not relevant. 2) For 136-bit response token: This register contains bits[37:22] of the response token.

Table 1164:Response Halfword 7 Register

Offset: 0x000D403C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	RSVD	RO 0x0	Reserved Read only.
13:0	Resp7	RO 0x0	1) For 48-bit response token: Not relevant. 2) For 136-bit response token: This register contains bits[21:8] of the response token.

Table 1165:16-bit Data Word Accessed by CPU Register

Offset: 0x000D4040

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	CpuData	RW 0x0	The CPU can access the SD data in the FIFO through this data port register. The first access is to the 16 LSB of the FIFO word, and the second access is to the 16 MSB of FIFO word. **NOTE:** The firmware needs to write or read multiple numbers of WORDS to/from this register even though the transfer size is not a multiple of words. The hardware expects that the CPU write/read multiple of words to/from this register. SD side knows when to stop the transfer of data to/from the SD cards because it has the byte and block count. **NOTE:**

Table 1166:CRC7 of I Response Register

Offset: 0x000D4044

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:7	RSVD	RO 0x0	Reserved Read only.
6:0	CRC7RespToken	RW 0x0	This field contains bits[7:1] (CRC7) of the response token.

Table 1167:Host Present State 16 LSB Register

Offset: 0x000D4048

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	RSVD	RO 0x0	Reserved
14	AutoCmd12Active	RO 0x0	Indicates that auto_cmd12 is active.
13	FifoEmpty	RO 0x1	FIFO Empty
12	FifoFull	RO 0x0	FIFO Full
11:10	RSVD	RO 0x0	Reserved
9	RxActive	RO 0x0	Indicates that the read transfer is active. Note: Sop_at_blk_gap status is set to1, if this bit is changed from 1 to 0. 0 = RxEnabled: This bit is cleared to 0 for the followings: 1) When the last data block as specified by block length is transferred to the system. Transfer complete status is set to 1, if this bit is changed from 1 to 0. 2) When all valid data blocks have been transferred to the system, and no current block transfers are being sent as a result of the Stop At Block Gap. 1 = RxDisabled: This bit is set to 1 for the followings: 1) After the end bit of the read command 2) When writing 1 to Continue Request in the Block Gap Control register to restart a read transfer.
8	TxActive	RO 0x0	Indicates that write transfer is active. If this bit is 0, it means no valid write data exists in the Host controller. A transfer complete interrupt is generated when all write data is out. Besides, during a write transaction, a Block Gap Event interrupt generated when this bit is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the Host driver in determining when to issue commands during write busy. 0 = TxDisabled: No valid write data exists in the Host controller. This bit is cleared in either of the following cases: 1) After getting the CRC status of the last data block as specified by the transfer count (single and multiple). 2) After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request. 1 = TxEnabled: This bit is set either of the following cases: 1) After the end bit of the write command. 2) When writing a 1 to Continue Request in the Block Gap Control register, to restart a write transfer.
7	CmdLevel	RO 0x1	CMD line Signal Level This status is used to check the CMD line level to recover from errors, and for debugging.

Table 1167:Host Present State 16 LSB Register (Continued)

Offset: 0x000D4048

Bit	Field	Type/InitVal	Description
6:3	DatLevel	RO 0xf	DAT[3:0] Line Signal Level. This status is used to check the DAT line level to recover from errors, and for debugging. This field is especially useful in detecting the busy signal level from DAT[0].
2	RSVD	RO 0x0	Reserved
1	CardBusy	RO 0x0	Card busy This bit tells the host that Card is busy, It is set and clear by Host controller.
0	CmdInhibitCmd	RO 0x0	If this bit is 0, it indicates the CMD line is not in use, and the Host controller can issue a command using the CMD line. This bit is set after the Command register is written. This bit is cleared when the command response is received. Even if the CmdInhibitDat is set to 1, commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a command complete interrupt in the Normal Interrupt status register. If the Host controller cannot issue the command because of a command conflict error, this bit remains 1, and the command complete is not set. 0 = CmdRespRecvd: This bit is cleared when the command response is received. 1 = CmdRegWrite: This bit is set after the command register is written.

Table 1168:Host Control Register

Offset: 0x000D4050

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	TimeoutEn	RW 0x0	Enable timeout value counter. If 0, host controller never receives a timeout. 0 = NoTimeout: Host controller never receives a timeout. 1 = Timeout: Host controller receives a timeout.

Table 1168:Host Control Register (Continued)

Offset: 0x000D4050

Bit	Field	Type/InitVal	Description
14:11	TimeoutValue	RW 0xF	Determines the interval by which the DAT line timeouts are detected. This timeout is initiated in these cases: 1) Read transaction: waiting for data from cards. This is Nac timing value in the SD specification, specifying the maximum timing from a read command to a read data (card data access time). 2) Write transaction: waiting for data from the IMB slave, IMB Master, or CPU. For others transaction, there are fixed timeout as followings (unit in SDCLK cycles) -Card side Ncr=64: maximum timing value from command to response. Nid=64 (5 in specification): maximum timing value from command to OCR response. - Host side Nrc=8: minimum timing value from response to next command. Ncc=8: minimum timing value from command to next command. Nwr=2: minimum timing value from data CRC status (from card in write transaction) to next write data in multiple write blocks Nst=2: minimum timing from STOP command to end of write data 0 = SDCLK x 2^12: Timeout value = SDCLK x 2^12 1 = SDCLK x 2^13: Timeout value = SDCLK x 2^13 2 = SDCLK x 2^14: Timeout value = SDCLK x 2^14 3 = SDCLK x 2^15: Timeout value = SDCLK x 2^15 4 = SDCLK x 2^16: Timeout value = SDCLK x 2^16 5 = SDCLK x 2^17: Timeout value = SDCLK x 2^17 6 = SDCLK x 2^18: Timeout value = SDCLK x 2^18 7 = SDCLK x 2^19: Timeout value = SDCLK x 2^19 8 = SDCLK x 2^20: Timeout value = SDCLK x 2^20 9 = SDCLK x 2^21: Timeout value = SDCLK x 2^21 10 = SDCLK x 2^22: Timeout value = SDCLK x 2^22 11 = SDCLK x 2^23: Timeout value = SDCLK x 2^23 12 = SDCLK x 2^24: Timeout value = SDCLK x 2^24 13 = SDCLK x 2^25: Timeout value = SDCLK x 2^25 14 = SDCLK x 2^26: Timeout value = SDCLK x 2^26 15 = SDCLK x 2^27: Timeout value = SDCLK x 2^27
10	HiSpeedEn	RW 0x0	High speed enable. If this bit is set to 0 (default), the Host controller outputs CMD and DAT lines at the falling edge of the SD clock (up to 25 MHz). If this bit is set to 1, the Host controller outputs CMD and DAT lines at the rising edge of the SD clock (up to 50 MHz). 0 = NormalSpeed: Normal speed (high speed disable) 1 = HighSpeedEnable: High speed enable
9	DataWidth	RW 0x0	Data width. 0 = 1BitDataMode: 1-bit data mode, using only DAT[0]. 1 = 4BitDataMode: 4-bit data mode enabled.

Table 1168:Host Control Register (Continued)

Offset: 0x000D4050

Bit	Field	Type/InitVal	Description
8:5	RSVD	RO 0x0	Reserved Read only.
4	LsbFirst	RW 0x0	After bit[3] of this register <BigEndian> does its job, this bit will do the following for both transmitting and receiving: 0 = MSB: Shifts MSB (bit[32] of a Word) first. 1 = LSB: Shifts LSB (bit[0] of a Word) first.
3	BigEndian	RW 0x0	Big Endian Enable 0 = LittleEndian: Little Endian: for transmitting, the host controller gets 32 bits of data from the internal buffer and shifts out the MSB bit first. For receiving, it does the same thing. 1 = BigEndian: Big Endian: for transmitting, the host controller swaps data bytes within a 32-bit data word, i.e., byte0 becomes byte3, byte 1 becomes byte 2, byte2 becomes byte 1, and byte 3 becomes byte 0. For receiving, the host controller swaps as was done for transmitting before writing to the internal buffer.
2:1	CardType	RW 0x0	Card type 0 = MemoryOnly: Memory only SD card 1 = IoOnly: IO only SD card 2 = IoMemCombo: IO and memory combo card 3 = MMC: MMC card
0	PushPullEn	RW 0x0	Push-pull enable. The host driver should program this bit to 1 (push-pull type) after the Card ID phase is completed. 0 = OpenDrain: Open drain on CMD line 1 = PushPullEn: Push-pull on CMD line

Table 1169:Data Block Gap Control Register

Offset: 0x000D4054

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:6	RSVD	RO 0x0	Reserved

Copyright © 2014 Marvell

Table 1169:Data Block Gap Control Register (Continued)

Offset: 0x000D4054

Bit	Field	Type/InitVal	Description
5	Suspend	Special 0x0	Write 1 to this register to suspend the operation at the block gap. This bit is cleared by hardware when the hardware is suspended. When suspended, which is indicated in status register, the firmware needs to store all register values, and the remaining block counts in the Current Number of Data Blocks Left to be Transferred Register are restored to the Data Block Count Register when resumed. **NOTE:** Write 1 and clear. **NOTE:**
4	Resume	Special 0x0	Write 1 to this register to resume the operation. Note: A write to the Command register is required to start the resume operation. This bit is cleared by hardware when hardware resumes. **NOTE:** Write 1 and clear. **NOTE:**
3	StopDatXfer	RW 0x0	Stop data transfer 0 = Disable: Data transfer continues. 1 = StopDataXferEn: Stops data transfer immediately.
2	RdWaitCtl	RW 0x0	Enable read wait protocol If the card supports read wait, set this bit to enable use of the read wait protocol to stop reading data using the DAT[2] line by Host hardware. Otherwise, the Host controller has to stop the SD clock to hold read data. When the Host driver detects a card insertion, it sets this bit according to the CCCR of the SDIO card. **NOTE:** This bit is looked at only at block gap. Within a block, hardware stalls the clock top stop read data if the host cannot accept additional data because of the FIFO buffer is full. **NOTE:** When this bit is cleared by firmware, the operation continues. During read wait, firmware can issue a different command for a different operation, as long as it does not require DATA lines. To continue the waiting operation, firmware needs to write 0 to this register. 0 = Disable: Disable read wait protocol 1 = EnableRdWait: Enable read wait protocol

Table 1169:Data Block Gap Control Register (Continued)

Offset: 0x000D4054

Bit	Field	Type/InitVal	Description
1	ContReq	RW1C 0x0	Continue request. This bit is used to restart a transaction which was stopped using the Stop At Block Gap Request. To cancel stop at the block gap, set Stop At Block Gap Request to 0 and set this bit to 1 to restart the transfer. The host controller automatically clears this bit in either of the following cases: 1) During a read transaction, the DAT Line Active changes from 0 to 1 as a read transaction restart. 2) During a write transaction, the Write Transfer Active changes from 0 to 1 as the write transaction restarts. Therefore, it is not necessary for Host driver to set this bit to 0. If Stop At Block Gap Request is set to 1, any write to this bit is ignored.
0	StopAtBlockGapReq	RW 0x0	Stop at block gap request. This it is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the transfer complete is set to 1, indicating a transfer completion, the Host driver leaves this bit set to 1. Clearing both the Stop At Block Gap Request and the Continue Request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The host controller stops the clock At Block Gap Request for write transfer, but for read transfer, it stops the clock if <RdWaitCtl> is 0. Otherwise, the host controller issues a Read Wait command to stop read data.

Table 1170:Clock Control Register

Offset: 0x000D4058

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:1	RSVD	RO 0x0	Reserved Read only.
0	SclkMasterEn	RW 0x0	SD clock master enable. For valid operation, this field must be 0x0. 0 = SdclkNotSet: The SD clock toggles sync with valid data. 1 = SdclkEn: The SD clock toggles always.

Table 1171:Software Reset Register

Offset: 0x000D405C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved

Table 1171:Software Reset Register (Continued)

Offset: 0x000D405C

Bit	Field	Type/InitVal	Description
15:9	RSVD	RO 0x0	Reserved Read only.
8	SwReset	Special 0x0	Software reset for all. This reset affects the status, state machine, and FIFOs synchronously. The configuration is not affected. **NOTE:** This register is Read Write and Clear. **NOTE:**
7:0	RSVD	RO 0x0	Reserved Read only.

Table 1172:Normal Interrupt Status Register

Offset: 0x000D4060

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	ErrInt	RO 0x0	Error interrupt. If any of bits in the Error Interrupt status register are set, then this bit is set.
14	UnexpectedRespDet	RW1C 0x0	Unexpected response from devices detected.
13	AutoCmd12Complete	RW1C 0x0	Auto_cmd12 is completed.
12	SuspenseOn	RW1C 0x0	Indicates that the hardware is suspended as the result of writing 1 to the <Suspend> field of the Transfer Mode register.
11	ImbFifo8wAvail	RO 0x1	There are at least eight empty entries for data to be written in the FIFO.
10	ImbFifo8wFull	RO 0x0	There are at least eight filled entries for data to be read from the FIFO.
9	ReadWaitOn	RW1C 0x0	Read Wait state is on. This bit is set to 1 to indicate to the firmware that the host controller is in Read Wait state (for SDIO cards). The host writes 1 to this bit to clear and restart data transfer, and to exit the host controller from Read Wait state.
8	CardInt	RW1C 0x0	Card interrupt. This bit is set to 1 if the host controller detects an interrupt from the card.
7:6	RSVD	RO 0x0	Reserved Read only.

Table 1172:Normal Interrupt Status Register (Continued)

Offset: 0x000D4060

Bit	Field	Type/InitVal	Description
5	RxRdy	RO 0x0	At least 1 byte of data is in the FIFO and ready to be read by the CPU.
4	TxRdy	RO 0x1	The FIFO has room for the CPU to write 16 bits of data.
3	DmaInt	RW1C 0x0	DMA interrupt. This status is set if the host controller detects DMA end.
2	BlockGapEv	RW1C 0x0	Block gap event. If the Stop At Block Gap Request in the Block Gap Control register is set, this bit is set when both a read or a write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this bit is not set to 1.
1	XferComplete	RW1C 0x0	Transfer Complete This bit is set when a read or write transaction is completed. 1) Read transaction: This bit is set at the falling edge of Read Transfer Active Status. There are two cases when this occurs: - Data transfer is completed as specified by the data length. - Data transfer is stopped at the block gap, and data transfer is completed by setting the Stop At Block Gap Request. 2) Write transaction: This bit is set at the falling edge of the DAT Line Active status. There are two cases when this occurs: - Data transfer is completed as specified by the data length and the busy signal is released. - Data transfer stopped at the block gap, and data transfer is completed by setting the Stop At Block Gap Request.
0	CmdComplete	RW1C 0x0	Command Complete This bit is set when the end bit of the command response is received (except for the Auto CMD12). **NOTE:** The Command Timeout Error has higher priority than the Command complete. **NOTE:**

Table 1173:Error Interrupt Status Register

Offset: 0x000D4064

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	RSVD	RO 0x0	Reserved Read only.
14	CrcStatErr	RW1C 0x0	CRC status error The CRC status returned from the card is not good in a write transaction.

Table 1173:Error Interrupt Status Register (Continued)

Offset: 0x000D4064

Bit	Field	Type/InitVal	Description
13	CrcStartBitErr	RW1C 0x0	CRC start bit error The CRC status start bit is not at the expected logic level in a write transaction.
12	CrcEndBitErr	RW1C 0x0	CRC end bit error The CRC status end bit is not at the expected logic level in a write transaction.
11	RespTbitErr	RW1C 0x0	Response T bit error.
10	XferSizeErr	RW1C 0x0	Transfer size mismatched error. This bit indicate that the transferred data size on the SD and on the host size is mismatched. For instance, SD side keeps track of the block count and block size. It knows when the data ends. The host side, that is the IMB Slave and CPU access, is supposed to know the transferred data size as well. In the normal case, both sides end with no extra byte remaining. If there is any data left in the FIFO when the block count and block size are exhausted, this interrupt Is generated. In the case of an IMB Master Read (read from cards), the SD side will indicate the end of the packet in FIFO data bit [33]. In the case of an IMB Master Write, because of synchronization, the SD side may end while the IMB Master still requests data from the SDRAM. In this case, the SD side will tell the DMA to stop the request and flush the FIFO.
9	CmdStartBitErr	RW 0x0	Command start bit error
8	AutoCmd12Err	RW1C 0x0	Auto CMD12 error. This error occurs when detecting that one of the bits in the Auto CMD12 Error Status register has changed from 0 to 1.
7	RSVD	RO 0x0	Reserved Read only.
6	RdDataEndBitErr	RW1C 0x0	Read data end bit error This bit is set to 1 when detecting 0 at the end bit position of the read data, which uses the DAT line, or at the end bit position of the CRC status.
5	RdDataCRCErr	RW1C 0x0	Read data CRC error This bit is set to 1 when transferring read data, which uses the DAT line, or when detecting the write CRC status having a value of other than 010.

Table 1173:Error Interrupt Status Register (Continued)

Offset: 0x000D4064

Bit	Field	Type/InitVal	Description
4	DataTimeoutErr	RW1C 0x0	Data timeout error This bit is set when detecting one of the followings: - Busy timeout after Write CRC status - Write CRC status timeout - Read data timeout
3	CmdIndexErr	RW1C 0x0	Command Index Error This bit is set when a command index error occurs in the command response.
2	CmdEndBitErr	RW1C 0x0	Command end bit error This bit is set when detecting that the end bit of a command response is 0.
1	CmdCrcErr	RW1C 0x0	Command CRC Error This bit is set in two cases: 1) Upon detecting a CRC error in the command response. 2) When the host controller detects a CMD line conflict, by monitoring the CMD line when a command is issued. The host controller aborts the command (stops driving the CMD line). The Command Timeout Error is also set to 1 to distinguish the CMD line conflict.
0	CmdTimeoutErr	RW1C 0x0	Command timeout error. This bit is set when no response is returned within 64 SDCLK cycles from the end bit of the command.

Table 1174:Normal Interrupt Status Enable Register

Offset: 0x000D4068

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	RSVD	RO 0x0	Reserved Read only.
14	UnexpectedRespDetEn	RW 0x0	Unexpected response from devices detected enable
13	AutoCmd12CompleteEn	RW 0x0	Auto_cmd12 complete enable.
12	SuspenseOnEn	RW 0x0	Suspense on enable.

Table 1174:Normal Interrupt Status Enable Register (Continued)

Offset: 0x000D4068

Bit	Field	Type/InitVal	Description
11:10	RSVD	RO 0x0	Reserved Read only.
9	RdWaitOnEn	RW 0x0	Read Wait on enable.
8	CardIntEn	RW 0x0	Card interrupt enable.
7:6	RSVD	RO 0x0	Reserved Read only.
5	RdRdyEn	RW 0x0	Buffer read ready enable.
4	WrRdyEn	RW 0x0	Buffer write ready enable.
3	DmaIntEn	RW 0x0	DMA interrupt enable.
2	BlockGapEvEn	RW 0x0	Block gap event enable.
1	XferCompleteEn	RW 0x0	Transfer complete enable.
0	CmdCompleteEn	RW 0x0	Command complete enable.

Table 1175:Error Interrupt Status Enable Register

Offset: 0x000D406C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	RSVD	RO 0x0	Reserved Read only.
14	CrcStatusErrEn	RW 0x0	CRC status error enable.
13	CrcStartBitErrEn	RW 0x0	CRC status start bit error enable.
12	CrcEndBitErrEn	RW 0x0	CRC status end bit error enable.
11	RespTbitErrEn	RW 0x0	Response T bit error enable.

Table 1175:Error Interrupt Status Enable Register (Continued)

Offset: 0x000D406C

Bit	Field	Type/InitVal	Description
10	SizeErrEn	RW 0x0	Size error enable.
9	CmdStartBitErrEn	RW 0x0	Command start bit error enable.
8	AutoCmd12ErrEn	RW 0x0	Auto CMD12 error enable.
7	RSVD	RO 0x0	Reserved Read only.
6	RdDataEndBitErrEn	RW 0x0	Data end bit error enable.
5	RdDataCrcErrEn	RW 0x0	Data CRC error enable.
4	DataTimeoutErrEn	RW 0x0	Data timeout error enable.
3	CmdIndexErrEn	RW 0x0	Command index error enable.
2	CmdEndBitErrEn	RW 0x0	Command end bit error enable.
1	CmdCrcErrEn	RW 0x0	Command CRC error enable.
0	CmdTimeoutErrEn	RW 0x0	Command timeout error enable.

Table 1176:Normal Interrupt Status Interrupt Enable Register

Offset: 0x000D4070

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	RSVD	RW 0x0	Reserved Read only.
14	UnexpectedRespDetIntEn	RO 0x0	Unexpected response from devices detected interrupt enable.
13	AutoCmd12CompleteIntEn	RO 0x0	Auto Cmd12 complete interrupt enable.
12	SuspenseOnIntEn	RW 0x0	Suspense on interrupt enable.
11	ImbFifo8wAvailIntEn	RO 0x0	IMB FIFO 8 word available interrupt enable.

Table 1176:Normal Interrupt Status Interrupt Enable Register (Continued)

Offset: 0x000D4070

Bit	Field	Type/InitVal	Description
10	ImbFifo8wFullIntEn	RO 0x0	IMB FIFO 8 word filled interrupt enable.
9	RdWaitOnIntEn	RW 0x0	Read Wait on interrupt enable.
8	CardIntIntEn	RW 0x0	Card interrupt interrupt enable.
7:6	RSVD	RW 0x0	Reserved Read only.
5	RxRdyIntEn	RW 0x0	Buffer read ready interrupt enable.
4	TxRdyIntEn	RW 0x0	Buffer write ready interrupt enable.
3	DmaIntIntEn	RW 0x0	DMA interrupt interrupt enable.
2	BlockGapEvtIntEn	RW 0x0	Block gap event interrupt enable.
1	XferCompleteIntEn	RW 0x0	Transfer complete interrupt enable.
0	CmdCompleteIntEn	RW 0x0	Command complete interrupt enable.

Table 1177:Error Interrupt Status Interrupt Enable Register

Offset: 0x000D4074

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15	RSVD	RO 0x0	Reserved Read only.
14	CrcStatusErrIntEn	RW 0x0	CRC status error interrupt enable.
13	CrcStatusStartBitErrIntEn	RW 0x0	CRC status start bit error interrupt enable.
12	CrcStatusEndBitErrIntEn	RW 0x0	CRC status end bit error interrupt enable.
11	RespTbitErrIntEn	RW 0x0	Response T bit error interrupt enable.
10	SizeErrIntEn	RW 0x0	Size error interrupt enable.

Table 1177:Error Interrupt Status Interrupt Enable Register (Continued)

Offset: 0x000D4074

Bit	Field	Type/InitVal	Description
9	CmdStartBitErrIntEn	RW 0x0	Command start bit error interrupt enable.
8	AutoCmd12ErrIntEn	RW 0x0	Auto CMD12 error interrupt enable.
7	RSVD	RO 0x0	Reserved Read only.
6	RdDataEndBitErrIntEn	RW 0x0	Data end bit error interrupt enable.
5	RdDataCrcErrIntEn	RW 0x0	Data CRC error interrupt enable.
4	DataTimeoutErrIntEn	RW 0x0	Data timeout error interrupt enable.
3	CmdIndexErrIntEn	RW 0x0	Command index error interrupt enable.
2	CmdEndBitErrIntEn	RW 0x0	Command end bit error interrupt enable.
1	CmdCrcErrIntEn	RW 0x0	Command CRC error interrupt enable.
0	CmdTimeoutErrIntEn	RW 0x0	Command timeout error interrupt enable.

Table 1178:Auto CMD12 Interrupt Status Register

Offset: 0x000D4078

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:7	RSVD	RW1C 0x0	Reserved Read only.
6	AutoCmd12RespStartBitEr	RW1C 0x0	Auto Cmd12 response start bit not detected, and timeout.
5	AutoCmd12RespTBitEr	RW1C 0x0	Auto Cmd12 Response T bit error.
4	AutoCmd12IndexEr	RW1C 0x0	Occurs if the command index error occurs in response to a command error.
3	AutoCmd12EndBitEr	RW1C 0x0	Occurs when detecting that the end bit of command response is 0.

Table 1178:Auto CMD12 Interrupt Status Register (Continued)

Offset: 0x000D4078

Bit	Field	Type/InitVal	Description
2	AutoCmd12CrcEr	RW1C 0x0	Occurs when detecting CRC error in the command response.
1	AutoCmd12TimeoutEr	RW1C 0x0	Occurs if no response is returned within 64 SDCLK cycles from the end bit of the command.
0	AutoCmd12NotExe	RW1C 0x0	Occurs when host controller cannot issue Auto Cmd12 to stop multiple block data transfer due to some errors.

Table 1179:Current Number of Bytes Remaining in Data Block Register

Offset: 0x000D407C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:12	RSVD	RO 0x0	Reserved Read only.
11:0	BlockSize	RO 0x0	Block Size This field shows the remaining number of bytes in the current block.

Table 1180:Current Number of Data Blocks Left to Be Transferred Register

Offset: 0x000D4080

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	BlockCount	RO 0x0	Block count. This field shows the remaining number of blocks in the current transfer.

Table 1181:Argument in Auto Cmd12 Command 16 LSB Transferred Register

Offset: 0x000D4084

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	AutoCmd12ArgLo	RW 0x0	The 16 LSB of the auto cmd12 command argument. This value is inserted into the 48-bit auto cmd12 command token bits[23:8].

Table 1182:Argument in Auto Cmd12 Command 16 MSB Transferred Register

Offset: 0x000D4088

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	AutoCmd12ArgHi	RW 0x0	The 16 MSB of the auto cmd12 command argument. This value is inserted into the 48-bit auto cmd12 command token bits[39:24].

Table 1183:Index of Auto Cmd12 Commands Transferred Register

Offset: 0x000D408C

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	RSVD	RO 0x0	Reserved Read only.
13:8	AutoCmd12Index	RW 0x0	Auto cmd12 command index. These bits will be inserted into the auto cmd12 command token bits[45:40].
7:2	RSVD	RO 0x0	Reserved Read only.
1	AutoCmd12IndexChkEn	RW 0x0	Index check for auto cmd12.
0	AutoCmd12BusyChkEn	RW 0x0	Checks for busy after the auto cmd12 response.

Table 1184:Auto Cmd12 Response Halfword 0 Register

Offset: 0x000D4090

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	AutoCmd12Resp0	RO 0x0	This register contains bits[45:30] of the cmd12 response token.

Table 1185:Auto Cmd12 Response Halfword 1 Register

Offset: 0x000D4094

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	AutoCmd12Resp1	RO 0x0	This register contains bits[29:14] of cmd12 response token.

Table 1186:Auto Cmd12 Response Halfword 2 Register

Offset: 0x000D4098

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	AutoCmd12Resp2	RO 0x0	This register contains bits[13:8] of the cmd12 response token.

Table 1187:Mbus Control Low Register

Offset: 0x000D4100

Bit	Field	Type/InitVal	Description
31:28	SdArbEntry7	RW 0x7	SD Arbitrator Entry 7.
27:24	SdArbEntry6	RW 0x6	SD Arbitrator Entry 6.
23:20	SdArbEntry5	RW 0x5	SD Arbitrator Entry 5.
19:16	SdArbEntry4	RW 0x4	SD Arbitrator Entry 4.
15:12	SdArbEntry3	RW 0x3	SD Arbitrator Entry 3.
11:8	SdArbEntry2	RW 0x2	SD Arbitrator Entry 2.
7:4	SdArbEntry1	RW 0x1	SD Arbitrator Entry 1.
3:0	SdArbEntry0	RW 0x0	SD Arbitrator Entry 0.

Table 1188:Mbus Control High Register

Offset: 0x000D4104

Bit	Field	Type/InitVal	Description
31:28	SdArbEntry15	RW 0x7	SD Arbitrator Entry 15.

Table 1188:Mbus Control High Register (Continued)

Offset: 0x000D4104

Bit	Field	Type/InitVal	Description
27:24	SdArbEntry14	RW 0x6	SD Arbitrator Entry 14.
23:20	SdArbEntry13	RW 0x5	SD Arbitrator Entry 13.
19:16	SdArbEntry12	RW 0x4	SD Arbitrator Entry 12.
15:12	SdArbEntry11	RW 0x3	SD Arbitrator Entry 11.
11:8	SdArbEntry10	RW 0x2	SD Arbitrator Entry 10.
7:4	SdArbEntry9	RW 0x1	SD Arbitrator Entry 9.
3:0	SdArbEntry8	RW 0x0	SD Arbitrator Entry 8.

Table 1189:Window0 Control Register

Offset: 0x000D4108

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window size. Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's. The number of 1's specifies the size of the window in 64 KByte granularity (e.g., a value of 0x00FF specifies 256 = 16 MByte). **NOTE:** A value of 0x0 specifies 64-KByte size. **NOTE:**
15:8	Attr	RW 0x0E	Specifies the target interface attributes associated with this window.
7:4	Target	RW 0x0	Specifies the target interface associated with this window.
3:1	Reserved	RSVD 0x0	Reserved
0	WinEn	RW 0x1	Window 0 Enable 0 = DisableWindow: Window is disabled. 1 = EnableWindow: Window is enabled.

Table 1190:Window0 Base Register

Offset: 0x000D410C

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base address Used with the <Size> field of the Window0 Control Register to set the address window size and location. Corresponds to transaction address[31:16].
15:0	Reserved	RSVD 0x0	Reserved

Table 1191:Window1 Control Register

Offset: 0x000D4110

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window size. Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's. The number of 1's specifies the size of the window in 64 KByte granularity (e.g., a value of 0x00FF specifies 256 = 16 MByte). **NOTE:** A value of 0x0 specifies 64-KByte size. **NOTE:**
15:8	Attr	RW 0x0D	Specifies the target interface attributes associated with this window.
7:4	Target	RW 0x0	Specifies the target interface associated with this window.
3:1	Reserved	RSVD 0x0	Reserved
0	WinEn	RW 0x1	Window 1 Enable 0 = DisableWindow: Window is disabled. 1 = EnableWindow: Window is enabled.

Table 1192:Window1 Base Register

Offset: 0x000D4114

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x1000	Base address Used with the <Size> field of the Window1 Control Register to set the address window size and location. Corresponds to transaction address[31:16].
15:0	Reserved	RSVD 0x0	Reserved

Table 1193:Window2 Control Register

Offset: 0x000D4118

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window size. Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's. The number of 1's specifies the size of the window in 64 KByte granularity (e.g., a value of 0x00FF specifies 256 = 16 MByte). **NOTE:** A value of 0x0 specifies 64-KByte size. **NOTE:**
15:8	Attr	RW 0x0B	Specifies the target interface attributes associated with this window.
7:4	Target	RW 0x0	Specifies the target interface associated with this window.
3:1	Reserved	RSVD 0x0	Reserved
0	WinEn	RW 0x1	Window 2 Enable 0 = DisableWindow: Window is disabled. 1 = EnableWindow: Window is enabled.

Table 1194:Window2 Base Register

Offset: 0x000D411C

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x2000	Base address Used with the <Size> field of the Window2 Control Register to set the address window size and location. Corresponds to transaction address[31:16].
15:0	Reserved	RSVD 0x0	Reserved

Table 1195:Window3 Control Register

Offset: 0x000D4120

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0xFFF	Window size. Used with the Base register to set the address window size and location. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's. The number of 1's specifies the size of the window in 64 KByte granularity (e.g., a value of 0x00FF specifies 256 = 16 MByte). **NOTE:** A value of 0x0 specifies 64-KByte size. **NOTE:**
15:8	Attr	RW 0x07	Specifies the target interface attributes associated with this window.

Table 1195:Window3 Control Register (Continued)

Offset: 0x000D4120

Bit	Field	Type/InitVal	Description
7:4	Target	RW 0x0	Specifies the target interface associated with this window.
3:1	Reserved	RSVD 0x0	Reserved
0	WinEn	RW 0x1	Window 3 Enable 0 = DisableWindow: Window is disabled. 1 = EnableWindow: Window is enabled.

Table 1196:Window3 Base Register

Offset: 0x000D4124

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x3000	Base address Used with the <Size> field of the Window3 Control Register to set the address window size and location. Corresponds to transaction address[31:16].
15:0	Reserved	RSVD 0x0	Reserved

Table 1197:Clock Divider Value Register

Offset: 0x000D4128

Bit	Field	Type/InitVal	Description
31:11	Reserved	RSVD 0x0	Reserved
10:0	ClkDvdrMValue	RW 0x7CF	Clock divider value (m): SDIO clock is: 100/(m+1)

Table 1198:Address Decoder Error Register

Offset: 0x000D412C

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	Add_Dec_Multi_Err	RW0C 0x0	Address Decoding Error Asserted upon address decoding multiple hit error.
0	Add_Dec_Miss_Err	RW0C 0x0	Address Decoding Error Asserted upon address decoding miss error.

Table 1199:Address Decoder Error Mask Register

Offset: 0x000D4130

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1:0	Various	RW 0x0	Mask bits for Address Decoder Error register. 0 = Mask 1 = Do not mask

A.15 Inter-Integrated Circuit (I2C) Registers

The following table provides a summarized list of all registers that belong to the Inter-Integrated Circuit (I2C), including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1200:Summary Map Table for the Inter-Integrated Circuit (I2C) Registers

Register Name	Offset	Table and Page
I2C Port		
I2C Slave Address Register (t=0–1)	I2C Port0: 0x00011000 I2C Port1: 0x00011100	Table 1201, p. 1392
I2C Data Register (t=0–1)	I2C Port0: 0x00011004 I2C Port1: 0x00011104	Table 1202, p. 1392
I2C Control Register (t=0–1)	I2C Port0: 0x00011008 I2C Port1: 0x00011108	Table 1203, p. 1392
I2C Baud Rate Register (t=0–1)	I2C Port0: 0x0001100C I2C Port1: 0x0001110C	Table 1204, p. 1393
I2C Status Register (t=0–1)	I2C Port0: 0x0001100C I2C Port1: 0x0001110C	Table 1205, p. 1394
I2C Extended Slave Address Register (t=0–1)	I2C Port0: 0x00011010 I2C Port1: 0x00011110	Table 1206, p. 1396
I2C Soft Reset Register (t=0–1)	I2C Port0: 0x0001101C I2C Port1: 0x0001111C	Table 1207, p. 1396
I2C Initialization Last Data Register	0x00011098	Table 1208, p. 1396
I2C Bridge		
I2C Bridge Transmit Data Low Register (t=0–1)	I2C Port0: 0x000110C0 I2C Port1: 0x000111C0	Table 1209, p. 1396
I2C Bridge Transmit Data High Register (t=0–1)	I2C Port0: 0x000110C4 I2C Port1: 0x000111C4	Table 1210, p. 1397
I2C Bridge Receive Data Low Register (t=0–1)	I2C Port0: 0x000110C8 I2C Port1: 0x000111C8	Table 1211, p. 1397
I2C Bridge Receive Data High Register (t=0–1)	I2C Port0: 0x000110CC I2C Port1: 0x000111CC	Table 1212, p. 1397
I2C Bridge Control Register (t=0–1)	I2C Port0: 0x000110D0 I2C Port1: 0x000111D0	Table 1213, p. 1397
I2C Bridge Status Register (t=0–1)	I2C Port0: 0x000110D4 I2C Port1: 0x000111D4	Table 1214, p. 1398
I2C Bridge Interrupt Cause Register (t=0–1)	I2C Port0: 0x000110D8 I2C Port1: 0x000111D8	Table 1215, p. 1399
I2C Bridge Interrupt Mask Register (t=0–1)	I2C Port0: 0x000110DC I2C Port1: 0x000111DC	Table 1216, p. 1399
I2C Bridge Timing gaps Register (t=0–1)	I2C Port0: 0x000110E0 I2C Port1: 0x000111E0	Table 1217, p. 1400

A.15.1 I2C Port

Table 1201:I2C Slave Address Register (t=0–1)

Offset: I2C Port0: 0x00011000, I2C Port1: 0x00011100

Offset Formula: 0x00011000+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:1	SAddr	RW 0x0	Slave address For a 7-bit slave address, bits [7:1] are the slave address. For a 10-bit address, SAddr[7:3] must be set to 11110 and SAddr[2:1] stands for the two MSB (bits [9:8]) of the 10-bit address.
0	GCE	RW 0x0	General Call Enable If set to 1, the I2C slave interface responds to general call accesses.

Table 1202:I2C Data Register (t=0–1)

Offset: I2C Port0: 0x00011004, I2C Port1: 0x00011104

Offset Formula: 0x00011004+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7:0	Data	RW 0x0	Data/Address byte to be transmitted by the I2C master or slave, or data byte received In the case of the Address byte, bit [0] is the Read/Write Command bit.

Table 1203:I2C Control Register (t=0–1)

Offset: I2C Port0: 0x00011008, I2C Port1: 0x00011108

Offset Formula: 0x00011008+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7	IntEn	RW 0x0	Interrupt Enable When set to 1, an interrupt is generated each time the interrupt flag is set.
6	I2CEn	RW 0x0	I2C Enable 0 = Ignore: I2C_SDA and I2C_SCK inputs are ignored. The I2C slave does not respond to any address on the bus. 1 = Respond: The I2C slave responds to calls to its slave address, and to general calls if enabled.

Table 1203:I2C Control Register (t=0–1) (Continued)

Offset: I2C Port0: 0x00011008, I2C Port1: 0x00011108

Offset Formula: 0x00011008+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
5	Start	RW 0x0	Start When set to 1, the I2C master initiates a start condition on the bus, when the bus is free, or a repeated start condition, if the master already drives the bus. The bit is set only. It is cleared by I2C hardware after a start condition is driven on the bus.
4	Stop	RW 0x0	Stop When set to 1, the I2C master initiates a stop condition on the bus. The bit is set only. It is cleared by I2C hardware after a stop condition is driven on the bus.
3	IFlg	RW 0x0	Interrupt Flag If any of the status codes other than 0xF8 are set, the I2C hardware sets the bit to 1. If set to 1 and I2C interrupts are enabled through bit [7], an interrupt is asserted. Cleared by a CPU core write of 0.
2	ACK	RW 0x0	Acknowledge When set to 1, the I2C drives an acknowledge bit on the bus in response to a received address (slave mode), or in response to a data received (read data in master mode, write data in slave mode). For a master to signal a I2C target with a read of last data, the CPU core must clear this bit (generating no acknowledge bit on the bus). For the slave to respond, this bit must always be set back to 1.
1:0	Reserved	RSVD 0x0	Reserved

Table 1204:I2C Baud Rate Register (t=0–1)

Offset: I2C Port0: 0x0001100C, I2C Port1: 0x0001110C

Offset Formula: 0x0001100C+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:7	Reserved	RSVD 0x0	Reserved
6:3	M	WO 0x4	See exact frequency calculation in the I2C section. Write only.
2:0	N	WO 0x5	See exact frequency calculation in the I2C section. Write only.

Table 1205:I2C Status Register (t=0–1)

Offset: I2C Port0: 0x0001100C, I2C Port1: 0x0001110C

Offset Formula: 0x0001100C+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved

Table 1205:I2C Status Register (t=0–1) (Continued)

Offset: I2C Port0: 0x0001100C, I2C Port1: 0x0001110C

Offset Formula: 0x0001100C+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
7:0	Stat	RO 0x0	I2C Status 0 = BusError 8 = StartTransmit: Start condition transmitted. 16 = StartTransmitRepeat: Repeated start condition transmitted. 24 = AddWrAck: Address plus write bit transmitted, acknowledge received. 32 = AddWrNoAck: Address plus write bit transmitted, acknowledge not received. 40 = MstrTxAck: Master transmitted data byte, acknowledge received. 48 = MstrTxNoAck: Master transmitted data byte, acknowledge not received. 56 = MstrLostArb: Master lost arbitration during address or data transfer. 64 = AddRdAck: Address plus read bit transmitted, acknowledge received. 72 = AddRdNoAck: Address +read bit transmitted, acknowledge not received. 80 = MstrRxAck: Master received read data, acknowledge transmitted. 88 = MstrRxNoAck: Master received read data, acknowledge not transmitted. 96 = SlRcdAddAck: Slave received slave address, acknowledge transmitted. 104 = MstrArbLost: Master lost arbitration during address transmit, address is targeted to the slave (write access), acknowledge transmitted. 112 = GnrRxAck: General call received, acknowledge transmitted. 120 = MstrArbLost: Master lost arbitration during address transmit, general call address received, acknowledge transmitted. 128 = SlRxWrDataAck: Slave received write data after receiving slave address, acknowledge transmitted. 136 = SlRxWrDataSlaveNoAck: Slave received write data after receiving slave address, acknowledge not transmitted. 144 = SlRxWrDataAck: Slave received write data after receiving general call, acknowledge transmitted. 152 = SlRxWrDataNoAck: Slave received write data after receiving general call, acknowledge not transmitted. 160 = SLRxStopStart: Slave received stop or repeated start condition. 168 = SlRxAddAck: Slave received address plus read bit, acknowledge transmitted. 176 = MstrArbLost: Master lost arbitration during address transmit, address is targeted to the slave (read access), acknowledge transmitted. 184 = SlTxRdDataAck: Slave transmitted read data, acknowledge received. 192 = SlTxRdDataNoAck: Slave transmitted read data, acknowledge not received. 200 = SlTxRdByteAck: Slave transmitted last read byte, acknowledge received. 208 = SecondAddWrBitTxAck: Second address plus write bit transmitted, acknowledge received. 216 = SecondAddWrBitTxNoAck: Second address plus write bit transmitted, acknowledge not received. 224 = SecondAddRdBitTxAck: Second address plus read bit transmitted, acknowledge received. 232 = SecondAddRdBitTxNoAck: Second address + read bit transmitted, acknowledge not received. 248 = NotRelevant: No relevant status. Interrupt flag is kept 0.

Table 1206:I2C Extended Slave Address Register (t=0–1)

Offset: I2C Port0: 0x00011010, I2C Port1: 0x00011110

Offset Formula: 0x00011010+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7:0	SAddr	RW 0x0	Bits [7:0] of the 10-bit slave address

Table 1207:I2C Soft Reset Register (t=0–1)

Offset: I2C Port0: 0x0001101C, I2C Port1: 0x0001111C

Offset Formula: 0x0001101C+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:0	Rst	WO 0x0	Write Only Write to this register resets the I2C logic and sets all I2C registers to their reset values.

Table 1208:I2C Initialization Last Data Register

Offset: 0x00011098

Bit	Field	Type/InitVal	Description
31:0	Last	RW 0xFFFFFFFF	Serial initialization last data indication. Can be changed during the serial initialization sequence to terminate the initialization with a different pattern.

A.15.2 I2C Bridge

Table 1209:I2C Bridge Transmit Data Low Register (t=0–1)

Offset: I2C Port0: 0x000110C0, I2C Port1: 0x000111C0

Offset Formula: 0x000110C0+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:0	TxDataLow	RW 0x0	4 Low data bytes to be transmitted by the I2C master.

Table 1210:I2C Bridge Transmit Data High Register (t=0–1)

Offset: I2C Port0: 0x000110C4, I2C Port1: 0x000111C4

Offset Formula: 0x000110C4+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:0	TxDataHigh	RW 0x0	4 High data bytes to be transmitted by the I2C master.

Table 1211:I2C Bridge Receive Data Low Register (t=0–1)

Offset: I2C Port0: 0x000110C8, I2C Port1: 0x000111C8

Offset Formula: 0x000110C8+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:0	RxDataLow	RO 0x0	4 Low data bytes received from I2C slave.

Table 1212:I2C Bridge Receive Data High Register (t=0–1)

Offset: I2C Port0: 0x000110CC, I2C Port1: 0x000111CC

Offset Formula: 0x000110CC+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:0	RxDataHigh	RO 0x0	4 High data bytes received from I2C slave.

Table 1213:I2C Bridge Control Register (t=0–1)

Offset: I2C Port0: 0x000110D0, I2C Port1: 0x000111D0

Offset Formula: 0x000110D0+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:21	Reserved	RSVD 0x0	Reserved
20	RepeatedStart	RW 0x0	If set and <Wr> and <Rd> are set, preform a read transaction after the write without a stop bit between them.
19	BridgeEn	RW 0x0	When set and the <Rd>/<Wr> bits are set, starts an I2C bridge transaction.

Table 1213:I2C Bridge Control Register (t=0–1) (Continued)

Offset: I2C Port0: 0x000110D0, I2C Port1: 0x000111D0

Offset Formula: 0x000110D0+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
18:16	RxSize	RW 0x0	This field indicates the number of bytes that needs to be received. 0 = 1 byte 1 = 2 bytes 2 = 3 bytes 3 = 4 bytes 4 = 5 bytes 5 = 6 bytes 6 = 7 bytes 7 = 8 bytes
15:13	TxSize	RW 0x0	This field indicates the number of bytes that needs to be transmitted. 0 = 1 byte 1 = 2 bytes 2 = 3 bytes 3 = 4 bytes 4 = 5 bytes 5 = 6 bytes 6 = 7 bytes 7 = 8 bytes
12	ExtendAddr	RW 0x0	Extended Slave Address Set to 0x1, if the Slave Address is 10 bits. 0 = 7-bit Address 1 = 10-bit Address
11:2	Addr	RW 0x0	I2C Slave Address
1	Rd	RW 0x0	Read Transaction When set to 0x1, perform a read transaction. If the <Wr> bit is set, also perform a read after the write.
0	Wr	RW 0x0	Write Transaction When set to 0x1, perform a write transaction. If the <Rd> bit is set, also perform a read after the write.

Table 1214:I2C Bridge Status Register (t=0–1)

Offset: I2C Port0: 0x000110D4, I2C Port1: 0x000111D4

Offset Formula: 0x000110D4+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:14	Reserved	RSVD 0x0	Reserved

Table 1214:I2C Bridge Status Register (t=0–1) (Continued)

Offset: I2C Port0: 0x000110D4, I2C Port1: 0x000111D4

Offset Formula: 0x000110D4+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
13:11	ErrorState	RO 0x0	Indicates the reason for the error. **NOTE:** Other values are reserved. 0 = No Error 1 = Start Error: Status after <START> bit set indicates an error. 2 = Addr Error: Status after address phase indicates an error. 3 = Extended Addr Error: Status after extended address phase indicates an error. 4 = Tx Error: Status after transmit data indicates an error. 5 = Rx Error: Status after received data indicates an error.
10:8	BytesLeft	RO 0x0	Indicate the number of bytes left to transmit/receive.
7:1	Reserved	RSVD 0x0	Reserved
0	Error	RO 0x0	Set to 1 when an error is detected during the transaction. 0 = No error 1 = Error

Table 1215:I2C Bridge Interrupt Cause Register (t=0–1)

Offset: I2C Port0: 0x000110D8, I2C Port1: 0x000111D8

Offset Formula: 0x000110D8+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	Interrupt	RW 0x0	The interrupt sets when a transaction ends or an error was detected.

Table 1216:I2C Bridge Interrupt Mask Register (t=0–1)

Offset: I2C Port0: 0x000110DC, I2C Port1: 0x000111DC

Offset Formula: 0x000110DC+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	InterruptMask	RW 0x0	When set the interrupt is masked.

Table 1217:I2C Bridge Timing gaps Register (t=0–1)

Offset: I2C Port0: 0x000110E0, I2C Port1: 0x000111E0

Offset Formula: 0x000110E0+t*0x100 : where t (0-1) represents I2C Port

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24:20	StopToStartGap	RW 0x0	A delay of 2^Value in terms of core_clk between the stop bit and the start bit.
19:15	AckToStopGap	RW 0x0	A delay of 2^Value in terms of core_clk between the ACK received and the stop bit.
14:10	AckToNextByteGap	RW 0x0	A delay of 2^Value in terms of core_clk between the ACK received and the next byte transmitted/received.
9:5	StartToAddrGap	RW 0x0	A delay of 2^Value in terms of core_clk between the start bit and the address phase.
4:0	AckToStartGap	RW 0x0	A delay of 2^Value in terms of core_clk between the ACK received and the start bit.

A.16 UART Registers

The following table provides a summarized list of all registers that belong to the UART, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1218:Summary Map Table for the UART Registers

Register Name	Offset	Table and Page
UART External Control Register	0x00010700	Table 1219, p. 1402
Transmit Holding (THR) Register (n=0–3)	UART0: 0x00012000, UART1: 0x00012100, UART2: 0x00012200, UART3: 0x00012300	Table 1220, p. 1403
Divisor Latch Low (DLL) Register (n=0–3)	UART0: 0x00012000, UART1: 0x00012100, UART2: 0x00012200, UART3: 0x00012300	Table 1221, p. 1403
Receive Buffer (RBR) Register (n=0–3)	UART0: 0x00012000, UART1: 0x00012100, UART2: 0x00012200, UART3: 0x00012300	Table 1222, p. 1403
Divisor Latch High (DLH) Register (n=0–3)	UART0: 0x00012004, UART1: 0x00012104, UART2: 0x00012204, UART3: 0x00012304	Table 1223, p. 1404
Interrupt Enable (IER) Register (n=0–3)	UART0: 0x00012004, UART1: 0x00012104, UART2: 0x00012204, UART3: 0x00012304	Table 1224, p. 1404
FIFO Control (FCR) Register (n=0–3)	UART0: 0x00012008, UART1: 0x00012108, UART2: 0x00012208, UART3: 0x00012308	Table 1225, p. 1405
Interrupt Identity (IIR) Register (n=0–3)	UART0: 0x00012008, UART1: 0x00012108, UART2: 0x00012208, UART3: 0x00012308	Table 1226, p. 1406
Line Control (LCR) Register (n=0–3)	UART0: 0x0001200C, UART1: 0x0001210C, UART2: 0x0001220C, UART3: 0x0001230C	Table 1227, p. 1407
Modem Control (MCR) Register (n=0–3)	UART0: 0x00012010, UART1: 0x00012110, UART2: 0x00012210, UART3: 0x00012310	Table 1228, p. 1408
Line Status (LSR) Register (n=0–3)	UART0: 0x00012014, UART1: 0x00012114, UART2: 0x00012214, UART3: 0x00012314	Table 1229, p. 1409

Table 1218:Summary Map Table for the UART Registers (Continued)

Register Name	Offset	Table and Page
Modem Status (MSR) Register (n=0–3)	UART0: 0x00012018, UART1: 0x00012118, UART2: 0x00012218, UART3: 0x00012318	Table 1230, p. 1410
Scratch Pad (SCR) Register (n=0–3)	UART0: 0x0001201C, UART1: 0x0001211C, UART2: 0x0001221C, UART3: 0x0001231C	Table 1231, p. 1411
FIFO Access (FAR) Register (n=0–3)	UART0: 0x00012070, UART1: 0x00012170, UART2: 0x00012270, UART3: 0x00012370	Table 1232, p. 1411
UART Status (USR) Register (n=0–3)	UART0: 0x0001207C, UART1: 0x0001217C, UART2: 0x0001227C, UART3: 0x0001237C	Table 1233, p. 1412
Halt TX (HTX) Register (n=0–3)	UART0: 0x000120A4, UART1: 0x000121A4, UART2: 0x000122A4, UART3: 0x000123A4	Table 1234, p. 1413

Table 1219:UART External Control Register

Offset: 0x00010700

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved

15:8	DMAMode	RW
		0xff
			For each UART this field contains a bit that defines if the UART port is configured to DMA mode.
			If configured to DMA mode, set the relevant bit to 1:
			bit[8] = UART0
			bit[9] = UART1
			bit[10] = UART2
			bit[11] = UART3 (Reserved for the MV76100)

7:1	Reserved	RSVD 0x0	Reserved
0	External_Writes_ Enable	RW 0x0	Enables DMA based transmit over UART. When activating DMA towards the UART, this bit must be asserted. 0 = Normal: An interrupt is sent to the CPU after every UART transmit session (after each stop bit assertion). 1 = DMA Based: An interrupt is sent to the CPU only upon completion of a data chunk transmission over the UART interface.

Table 1220:Transmit Holding (THR) Register (n=0–3)

Offset: UART0: 0x00012000, UART1: 0x00012100, UART2: 0x00012200, UART3: 0x00012300

Offset Formula: 0x00012000+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	TxHold	WO 0x0	The THR is a write-only register that contains data to be transmitted from the serial port. Any time that the Transmit Holding Register Empty <THRE> bit of the Line Status Register (LSR) is set, data can be written to the LSR <TxEmpty> to be transmitted from the serial port. If FIFOs are not enabled and THRE is set, writing a single word to the THR resets the THRE and any additional writes to the THR before the <THRE> is set again causes the THR data to be overwritten. If FIFOs are enabled and <THRE> is set, up to 16 words of data may be written to the THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.

Table 1221:Divisor Latch Low (DLL) Register (n=0–3)

Offset: UART0: 0x00012000, UART1: 0x00012100, UART2: 0x00012200, UART3: 0x00012300

Offset Formula: 0x00012000+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	DivLatchLow	WO 0x0	Lower 8 bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. This register may only be accessed when the DivLatchRdWrt bit (LCR[7]) is set and the UART is not busy (USR[0] is zero). The output baud rate is equal to the serial clock frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (serial clock freq) / (16 * divisor). Note that when the Divisor Latch Registers (DLL and DLH) are set to zero, the baud clock is disabled and no serial communications occur.

Table 1222:Receive Buffer (RBR) Register (n=0–3)

Offset: UART0: 0x00012000, UART1: 0x00012100, UART2: 0x00012200, UART3: 0x00012300

Offset Formula: 0x00012000+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved

Table 1222:Receive Buffer (RBR) Register (n=0–3) (Continued)

Offset: UART0: 0x00012000, UART1: 0x00012100, UART2: 0x00012200, UART3: 0x00012300

Offset Formula: 0x00012000+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
7:0	RxBuf	RO 0x0	The RBR is a read-only register that contains the data byte transmitted to the serial port.The data in this register is valid only if the LSR <DataRxStat> bit in the Line Status Register (LSR) is set. In the non-FIFO mode (fifo_mode = 0), the data in the RBR must be read before the next data arrives; otherwise it will be overwritten, resulting in an overrun error. In the FIFO mode (fifo_mode = 1), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data word arrives, then the data already in the FIFO will be preserved but any incoming data will be lost.

Table 1223:Divisor Latch High (DLH) Register (n=0–3)

Offset: UART0: 0x00012004, UART1: 0x00012104, UART2: 0x00012204, UART3: 0x00012304

Offset Formula: 0x00012004+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	DivLatchHigh	WO 0x0	Upper 8-bits of a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART.This register may only be accessed when the DivLatchRdWrt bit (LCR[7]) is set and the UART is not busy (USR[0] is zero). The output baud rate is equal to the core_clk frequency divided by sixteen times the value of the baud rate divisor, as follows: baud rate = (core clock freq) / (16 * divisor). Note that when the Divisor Latch Registers (DLL and DLH) are set to zero, the baud clock is disabled and no serial communications occur.

Table 1224:Interrupt Enable (IER) Register (n=0–3)

Offset: UART0: 0x00012004, UART1: 0x00012104, UART2: 0x00012204, UART3: 0x00012304

Offset Formula: 0x00012004+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7	PTIME	RW 0x0	Programmable THRE Interrupt Mode Enable that can be written to only when THRE_MODE_USER == Enabled, always readable. This is used to enable/disable the generation of THRE Interrupt. 0 = Disable 1 = Enable

Table 1224:Interrupt Enable (IER) Register (n=0–3) (Continued)

Offset: UART0: 0x00012004, UART1: 0x00012104, UART2: 0x00012204, UART3: 0x00012304

Offset Formula: 0x00012004+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
6:4	Reserved	RSVD 0x0	Reserved
3	ModStatIntEn	RW 0x0	Enable Modem Status Interrupt (EDSSI) This is used to enable/disable the generation of Modem Status Interrupt. This is the fourth highest priority interrupt 0 = Disable: Disable interrupt 1 = Enable: Enable interrupt
2	RxLineStatIntEn	RW 0x0	Enable Receiver Line Status Interrupt (ELSI) This is used to enable/disable the generation of Receiver Line Status Interrupt. This is the highest priority interrupt. 0 = Disable: Disable interrupt 1 = Enable: Enable interrupt
1	TxHoldIntEn	RW 0x0	Enable Transmitter Holding Register Empty Interrupt (ETBEI) This is used to enable/disable the generation of Transmitter Holding Register Empty Interrupt. This is the third highest priority interrupt. 0 = Disable: Disable interrupt 1 = Enable: Enable interrupt
0	RxDataIntEn	RW 0x0	Enable Received Data Available Interrupt (ERBFI). This is used to enable/disable the generation of Received Data Available Interrupt and the Character Timeout Interrupt (if in FIFO mode and FIFOs enabled). These are the second highest priority interrupts. 0 = Disable: Disable interrupt 1 = Enable: Enable interrupt

Table 1225:FIFO Control (FCR) Register (n=0–3)

Offset: UART0: 0x00012008, UART1: 0x00012108, UART2: 0x00012208, UART3: 0x00012308

Offset Formula: 0x00012008+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved

Table 1225:FIFO Control (FCR) Register (n=0–3) (Continued)

Offset: UART0: 0x00012008, UART1: 0x00012108, UART2: 0x00012208, UART3: 0x00012308

Offset Formula: 0x00012008+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
7:6	RxTrigger	WO 0x0	Receive Trigger This is used to select the trigger level in the receiver FIFOat which the Received Data Available Interrupt is generated. In auto flow control mode it is used to determine when the RTSn signal is de-asserted. 0 = 1Byte: 1 byte in Rx FIFO 1 = 4Byte: 4 bytes in Rx FIFO 2 = 8Byte: 8 bytes in Rx FIFO 3 = 14Byte: 14 bytes Rx FIFO
5:4	Tx Empty Trigger	RW 0x0	TX Empty Trigger (TET). This is used to select the empty threshold level at which the THRE Interrupts are generated. 0 = Empty: All Tx Fifo is not occupied 1 = 2Bytes: 14 bytes in Tx Fifo are not occupied 2 = 4Byes: 12 bytes in Tx Fifo are not occupied 3 = 8Bytes: 8 bytes in Tx Fifo are not occupied
3	Reserved	RSVD 0x0	Reserved
2	TxFIFOReset	WO 0x0	Transmit FIFO reset This resets the control portion of the Tx FIFO and treats the FIFO as empty. 0 = NoFlushTx: Do not flush data from the transmit FIFO 1 = FlushTx: Flush data from the transmit FIFO
1	RxFIFOReset	WO 0x0	Receive FIFO reset This resets the control portion of the Rx FIFO and treats the FIFO as empty 0 = NoFlushRx: Do not flush data from the receive FIFO. 1 = FlushRx: Flush data from the receive FIFO.
0	FIFOEn	WO 0x0	Enable transmit and receive FIFOs. This register controls the read and write data FIFO operation. 0 = Disable: Disable FIFOs 1 = Enable: Enable FIFOs

Table 1226:Interrupt Identity (IIR) Register (n=0–3)

Offset: UART0: 0x00012008, UART1: 0x00012108, UART2: 0x00012208, UART3: 0x00012308

Offset Formula: 0x00012008+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved

Table 1226:Interrupt Identity (IIR) Register (n=0–3) (Continued)

Offset: UART0: 0x00012008, UART1: 0x00012108, UART2: 0x00012208, UART3: 0x00012308

Offset Formula: 0x00012008+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
7:6	FIFOEn	RO 0x0	FIFO Enable 0 = Disable: FIFOs are disabled (default in FIFO mode) 3 = Enable: FIFOs are enabled
5:4	Reserved	RSVD 0x0	Reserved
3:0	InterruptID	RO 0x0	Interrupt ID (IID) This indicates the highest priority pending interrupt. The interrupt priorities are split into four levels. Bit 3 indicates an interrupt can only occur when the FIFOs are enabled and used to distinguish a Character Timeout condition interrupt. 0 = Modem: Modem Status Changed 1 = NoInterruptPen: No interrupt pending 2 = THREmpty 4 = RxData: Received Data available 6 = RxStatus: Receiver Status 7 = BusyDetect 12 = TO: Character Time Out

Table 1227:Line Control (LCR) Register (n=0–3)

Offset: UART0: 0x0001200C, UART1: 0x0001210C, UART2: 0x0001220C, UART3: 0x0001230C

Offset Formula: 0x0001200C+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7	DivLatchRdWrt	RW 0x0	This bit must be set to address (reading and writing) of the Divisor Latch Low (DLL) Register and Divisor Latch High (DLH) Register) to set the baud rate of the UART. This bit must be cleared to address the Receive Buffer Register (RBR), Transmit Holding Register (THR) and the Interrupt Enable Register (IER). The bit is writeable only when UART is not busy (USR[0] is zero)
6	Break	RW 0x0	The <Break> bit sends a break signal by holding the SOUT line low until the <Break> bit is reset. 0 = No Signal: Do not send a break signal 1 = Signal: Send a break signal
5	Reserved	RSVD 0x0	Reserved

Table 1227:Line Control (LCR) Register (n=0–3) (Continued)

Offset: UART0: 0x0001200C, UART1: 0x0001210C, UART2: 0x0001220C, UART3: 0x0001230C

Offset Formula: 0x0001200C+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
4	EPS	RW 0x0	Even or Odd Parity Select The bit is writeable only when UART is not busy (USR[0] is zero) 0 = Odd: Odd parity 1 = Even: Even parity
3	PEN	RW 0x0	Parity Enable The bit is writeable only when UART is not busy (USR[0] is zero) 0 = Disable 1 = Enable
2	Stop	RW 0x0	Stop Bits Transmitted This is used to select the number of stop bits per character that the peripheral transmits and receives. If set to zero, one stop bit is transmitted in the serial data. If set to one and the data bits are set to 5 (LCR[1:0] set to zero) one and a half stop bits is transmitted. Otherwise, two stop bits are transmitted. Note that regardless of the number of stop bits selected, the receiver checks only the first stop bit. The bit is writeable only when UART is not busy (USR[0] is zero) 0 = 1Bits 1 = 2Bits
1:0	WLS	RW 0x0	This is used to select the number of data bits per character that the peripheral transmits and receives The bit is writeable only when UART is not busy (USR[0] is zero) 0 = 5Bits 1 = 6Bits 2 = 7Bits 3 = 8Bits

Table 1228:Modem Control (MCR) Register (n=0–3)

Offset: UART0: 0x00012010, UART1: 0x00012110, UART2: 0x00012210, UART3: 0x00012310

Offset Formula: 0x00012010+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:6	Reserved	RW 0x0	Reserved
5	AFCE	RW 0x0	Auto Flow Control Enable. 0 = False: Auto Flow Control Mode disabled 1 = True: Auto Flow Control Mode enabled

Table 1228:Modem Control (MCR) Register (n=0–3) (Continued)

Offset: UART0: 0x00012010, UART1: 0x00012110, UART2: 0x00012210, UART3: 0x00012310

Offset Formula: 0x00012010+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
4	Loopback	RW 0x0	Loopback The <Loopback> bit loops the data on the sout line back to the sin line. In this mode all the interrupts are fully functional. This feature is used for diagnostic purposes. 0 = NoLoopback 1 = Loopback
3:2	Reserved	RSVD 0x0	Reserved
1	RTS	RW 0x0	Request To Send The <RTS> bit is inverted and then drives the corresponding UA_RTSn output.
0	Reserved	RSVD 0x0	Reserved

Table 1229:Line Status (LSR) Register (n=0–3)

Offset: UART0: 0x00012014, UART1: 0x00012114, UART2: 0x00012214, UART3: 0x00012314

Offset Formula: 0x00012014+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7	RxFIFOErr	ROC 0x0	This bit is only active when FIFOs are enabled. It is set when there is at least one parity error, framing error, or break indication in the FIFO. This bit is cleared when the Line Control Register (LCR) is read.
6	TxEmpty	RO 0x0	Transmitter Empty bit In FIFO mode, this bit is set whenever the Transmit Holding Register (THR), the Transmitter Shift Register, and the FIFO are all empty.
5	THRE	RO 0x0	Transmit Holding If the <THRE> bit is set, the device can accept a new character for transmission. If interrupts are enabled, it can cause an interrupt to occur when data from the Transmit Holding Register (THR) is transmitted to the transmit shift register. 0 = Closed: Do not accept a new character for transmission 1 = Open: Accept a new character for transmission
4	BI	ROC 0x0	The <BI> bit is set whenever the serial input (sin) is held in a logic 0 state for longer than the sum of start time + data bits + parity + stop bits. In the FIFO mode, the BI indication is carried through the FIFO and is revealed when the character is at the top of the FIFO. Reading the Line Control Register (LCR) clears the <BI>.

Table 1229:Line Status (LSR) Register (n=0–3) (Continued)

Offset: UART0: 0x00012014, UART1: 0x00012114, UART2: 0x00012214, UART3: 0x00012314

Offset Formula: 0x00012014+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
3	FrameErr	ROC 0x0	Frame Error The FE bit flags a framing error in the receiver. A framing error occurs when the receiver does not detect a valid STOP bit in the received data. In the FIFO mode, since the framing error is associated with a character received, it is revealed when the character with the framing error comes to the head of the FIFO. The OE, PE and FE bits are reset when a read on the Line Control Register (LCR) is performed.
2	ParErr	ROC 0x0	Parity Error This bit indicates a parity error in the receiver if the <PEN> bit in the Line Control Register (LCR) is set. In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the bad parity comes to the head of the FIFO.
1	OverRunErr	ROC 0x0	Overrun Error 0 = NoError: No overrun 1 = Error: Overrun error has occurred
0	DataRxStat	RO 0x0	Receive buffer status This bit is cleared when the Receive Buffer Register (RBR) is read. 0 = Empty: No characters in the receive buffer or FIFO. 1 = NotEmpty: Receive buffer or FIFO contains at least one character. A read operation of the Receiver Buffer Register clears this bit.

Table 1230:Modem Status (MSR) Register (n=0–3)

Offset: UART0: 0x00012018, UART1: 0x00012118, UART2: 0x00012218, UART3: 0x00012318

Offset Formula: 0x00012018+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4	CTS	RO 0x0	The CTS Modem Status bit--<CTS> --contains information on the current state of the modem control line. <CTS> is the compliment of UA_CTSn. In Loopback Mode, <CTS> is the same as Modem Control Register (MCR) bit[1] <RTS>.
3:1	Reserved	RSVD 0x0	Reserved

Table 1230:Modem Status (MSR) Register (n=0–3) (Continued)

Offset: UART0: 0x00012018, UART1: 0x00012118, UART2: 0x00012218, UART3: 0x00012318

Offset Formula: 0x00012018+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
0	DCTS	RO 0x0	The <DCTS> bit records whether the modem control line UA_CTSn has changed since the last time the CPU core read the MSR. In Loopback mode, <DCTS> reflects changes on Modem Control Register (MCR) bit[1] <RTS>.

Table 1231:Scratch Pad (SCR) Register (n=0–3)

Offset: UART0: 0x0001201C, UART1: 0x0001211C, UART2: 0x0001221C, UART3: 0x0001231C

Offset Formula: 0x0001201C+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:0	Scratch	RW 0x0	The SCR register is an 8-bit read/write register for programmers to use as a temporary storage space.

Table 1232:FIFO Access (FAR) Register (n=0–3)

Offset: UART0: 0x00012070, UART1: 0x00012170, UART2: 0x00012270, UART3: 0x00012370

Offset Formula: 0x00012070+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	FAR	RW 0x0	Fifo Access Register. This register is use to enable a FIFO access mode for testing, so that the receive FIFO can be written by the master and the transmit FIFO can be read by the master when FIFOs are enabled. When FIFOs are not enabled it allows the RBR to be written by the master and the THR to be read by the master. 0 = FIFO access mode disabled 1 = FIFO access mode enabled Note, that when the FIFO access mode is enabled/disabled, the control portion of the receive FIFO and transmit FIFO is reset and the FIFOs are treated as empty.

Table 1233:UART Status (USR) Register (n=0–3)

Offset: UART0: 0x0001207C, UART1: 0x0001217C, UART2: 0x0001227C, UART3: 0x0001237C

Offset Formula: 0x0001207C+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4	RFF	RO 0x0	Receive FIFO Full. This bit is used to indicate that the receive FIFO is completely full. 0 = Receive FIFO not full 1 = Receive FIFO Full This bit is cleared when the RX FIFO is no longer full.
3	RFNE	RO 0x0	Receive FIFO Not Empty. This bit is used to indicate that the receive FIFO contains one or more entries. 0 = Receive FIFO is empty 1 = Receive FIFO is not empty This bit is cleared when the RX FIFO is empty.
2	TFE	RO 0x1	Transmit FIFO Empty. This bit is used to indicate that the transmit FIFO is completely empty. 0 = Transmit FIFO is not empty 1 = Transmit FIFO is empty This bit is cleared when the TX FIFO is no longer empty.
1	TFNF	RO 0x1	Transmit FIFO Not Full. This is used to indicate that the transmit FIFO in not full. 0 = Transmit FIFO is full 1 = Transmit FIFO is not full This bit is cleared when the TX FIFO is full.
0	BUSY	RO 0x0	UART Busy. This bit indicates that a serial transfer is in progress. When cleared, indicates that the UART unit is idle or inactive. **NOTE:** It is possible for the UART Busy bit to be cleared even though a new character may have been sent from another device. That is, if the UART unit has no data in THR and RBR and there is no transmission in progress and a start bit of a new character has just reached the unit. This is due to the fact that a valid start is not seen until the middle of the bit period and this duration is dependent on the baud divisor that has been programmed. 0 = Idle: Idle or Inactive 1 = Busy: Actively transferring data

Table 1234:Halt TX (HTX) Register (n=0–3)

Offset: UART0: 0x000120A4, UART1: 0x000121A4, UART2: 0x000122A4, UART3: 0x000123A4

Offset Formula: 0x000120A4+n*0x100 : where n (0-3) represents UART

Bit	Field	Type/InitVal	Description
31:1	Reserved	RSVD 0x0	Reserved
0	Halt TX	RW 0x0	This register is use to halt transmissions for testing, so that the transmit FIFO can be filled by the master when FIFOs are enabled. 0 = Halt TX disabled 1 = Halt TX enabled Note, if FIFOs are not enabled, the setting of the halt TX register has no effect on operation.

A.17 Real Time Clock (RTC) Registers

The following table provides a summarized list of all registers that belong to the RTC, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1235:Summary Map Table for the RTC Registers

Register Name	Offset	Table and Page
RTC Time Register	0x00010300	Table 1236, p. 1414
RTC Date Register	0x00010304	Table 1237, p. 1415
RTC Alarm Time Configuration Register	0x00010308	Table 1238, p. 1416
RTC Alarm Date Configuration Register	0x0001030C	Table 1239, p. 1416
RTC Interrupt Mask Register	0x00010310	Table 1240, p. 1417
RTC Interrupt Cause Register	0x00010314	Table 1241, p. 1418
RTC External Alarm Control Register	0x00010320	Table 1242, p. 1418
RTC External Alarm Config Register	0x00010324	Table 1243, p. 1419
RTC External Alarm Status Register	0x00010328	Table 1244, p. 1419
RTC Count Status Register	0x0001032C	Table 1245, p. 1419

Table 1236:RTC Time Register

Offset: 0x00010300

Bit	Field	Type/InitVal	Description
This register provides time information.Note that the values in this register are maintained between power up and power down (with consideration for the real-time passing). Thus, The init value specified is meaningless in these fields.			
31:27	Reserved	RSVD 0x0	Reserved
26:24	WeekDay	RW 0x1	Day within the week (1..7)
23	Reserved	RSVD 0x0	Reserved
22	HourFomat	RW 0x0	12- or 24-hour format configuration: 0 = 24 1 = 12

Table 1236:RTC Time Register (Continued)

Offset: 0x00010300

Bit	Field	Type/InitVal	Description
21:16	Hour	RW 0x0	Hour within the day (0..23 or 1..12) In 12-hour format: Bit[21] is an AM/PM indication, where: 1 = PM 0 = AM Bit[20] sets the decimal digit. Bits[19:16] set the unit digit. In 24-hour format: Bits[19:16] sets the unit digit. Bits[21:20] set the decimal digit.
15	Reserved	RSVD 0x0	Reserved
14:8	Minute	RW 0x0	Minute within the hour (0..59) Bits[14:12] set the decimal digit. Bits[11:8] set the unit digit.
7	Reserved	RSVD 0x0	Reserved
6:0	Second	RW 0x0	Second within the minute (0..59). Bits[6:4] set the decimal digit. Bits[3:0] set the unit digit.

Table 1237:RTC Date Register

Offset: 0x00010304

Bit	Field	Type/InitVal	Description
This register provides date information.Note that the values in this register are maintained between power up and power down (with consideration for the real-time passing). Thus, The init value specified is meaningless in these fields.			
31:24	Reserved	RSVD 0x0	Reserved
23:16	Year	RW 0x0	Year within the 21st century (0..99)
15:13	Reserved	RSVD 0x0	Reserved
12:8	Month	RW 0x1	Month within the year (1..12)
7:6	Reserved	RSVD 0x0	Reserved

Table 1237:RTC Date Register (Continued)

Offset: 0x00010304

Bit	Field	Type/InitVal	Description
5:0	Day	RW 0x1	Day within the month (1..31)

Table 1238:RTC Alarm Time Configuration Register

Offset: 0x00010308

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved
23	AlarmHourValid	RW 0x0	When this field is set to 1, the value in the <AlarmHour> field must match the value in the <Hour> field in the RTC Time register for an alram interrupt to occur.
22	Reserved	RSVD 0x0	Reserved
21:16	AlarmHour	RW 0x0	When the <AlarmHourValid> field is set to 1, an alarm interrupt will occur only when the value in this field matches the value in the <Hour> field in the RTC Time register.
15	AlarmMinuteValid	RW 0x0	When this field is set to 1, the <AlarmMinute> field must match <Minute> field in the RTC Time register for an alram interrupt to occur.
14:8	AlarmMinutes	RW 0x0	When the <AlarmMinueValid> field is set to 1, an alarm interrupt will occur only when the value in this field matches the value in <Minute> field in the RTC Time register.
7	AlarmSecondValid	RW 0x0	When this field is set to 1, the <AlarmSecond> field must match the <Second> field in the RTC Time register for an alram interrupt to occur.
6:0	AlarmSeconds	RW 0x0	When the <AlarmSecondValid> field is set to 1, an alarm interrupt will occur only when the value in this field matches the value in the <Second> field in the RTC Time register.

Table 1239:RTC Alarm Date Configuration Register

Offset: 0x0001030C

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved

Table 1239:RTC Alarm Date Configuration Register (Continued)

Offset: 0x0001030C

Bit	Field	Type/InitVal	Description
24	AlarmYearValid	RW 0x0	When this field is set to 1, the <AlarmYear> field must match the <Year> field in the RTC Date register for an alram interrupt to occur.
23:16	AlarmYear	RW 0x0	When the <AlarmYearValid> field is set to 1, an alarm interrupt will occur only when the value in this field matches the value in the <Year> field in the RTC Date register.
15	AlarmMonthValid	RW 0x0	When this field is set to 1, the <AlarmMonth> field must match the <Month> field in the RTC Date register for an alram interrupt to occur.
14:13	Reserved	RSVD 0x0	Reserved
12:8	AlarmMonth	RW 0x1	When the <AlarmMonthValid> field is set to 1, an alarm interrupt will occur only when the value in this field matches the value in the <Month> field in the RTC Date register.
7	AlarmDayValid	RW 0x0	When this field is set to 1, the <AlarmDay> field must match the <Day> field in the RTC Date register for an alram interrupt to occur.
6	Reserved	RSVD 0x0	Reserved
5:0	AlarmDay	RW 0x1	When the <AlarmDayValid> field is set to 1, an alarm interrupt will occur only when the value in this field matches the value in the <Day> field in the RTC Date register.

Table 1240:RTC Interrupt Mask Register

Offset: 0x00010310

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	BatteryAlarmInterruptEnable	RW 0x0	Mask bit for the alarm battery interrupt Mask only effects the assertion of interrupt pin. It does not effect the setting of bits in the RTC Interrupt Cause register. 0 = Masked: Interrupt masked. 1 = Enabled: Interrupt enabled.

Table 1240:RTC Interrupt Mask Register (Continued)

Offset: 0x00010310

Bit	Field	Type/InitVal	Description
0	AlarmInterruptEnable	RW 0x0	Mask bit for the alarm interrupt Mask only effects the assertion of interrupt pin. It does not effect the setting of bits in the RTC Interrupt Cause register. 0 = Masked: Interrupt masked. 1 = Enabled: Interrupt enabled.

Table 1241:RTC Interrupt Cause Register

Offset: 0x00010314

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	BatteryAlarmInterrupt	RW0C 0x0	Battery alarm interrupt
0	AlarmInterrupt	RW0C 0x0	Set when any of the alarm valid bits is active and all the alarm fields that have an alarm valid bit set match the corresponding real time field. Cleared when written 0.

Table 1242:RTC External Alarm Control Register

Offset: 0x00010320

Bit	Field	Type/InitVal	Description
31:17	Reserved	RSVD 0x0	Reserved
16	AlarmStatus	RO 0x0	RTC External Alarm Status
15:2	Reserved	RSVD 0x0	Reserved
1	AlarmConfigSet	RW 0x0	RTC External Alarm Configuration Set
0	AlarmClr	RW 0x0	RTC External Alarm Clear

Table 1243:RTC External Alarm Config Register

Offset: 0x00010324

Bit	Field	Type/InitVal	Description
31:0	ExternalAlarmCntConfig	RW 0x0	RTC External Alarm Count Configuration This field defines the count value that will trigger the RTC external alarm interrupt. For a precise definition of caculation methods of this field, please refer to the Functionla specification.

Table 1244:RTC External Alarm Status Register

Offset: 0x00010328

Bit	Field	Type/InitVal	Description
31:0	ExternalAlarmStatus	RO 0x0	RTC External Alarm current status Reflects the external alarm count value is currentlly configured in the RTC

Table 1245:RTC Count Status Register

Offset: 0x0001032C

Bit	Field	Type/InitVal	Description
31:0	CountStatus	RO 0x0	RTC Count Current Status Reflects the current value of the RTC counter.

A.18 General Purpose Input/Output (GPIO) Registers

The following table provides a summarized list of all registers that belong to the General Purpose Input/Output (GPIO), including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1246:Summary Map Table for the General Purpose Input/Output (GPIO) Registers

Register Name	Offset	Table and Page
GPIO		
GPIO_<32*n>_<32*n+31>_DataOut Register (n=0–1)	GPIO_Num0: 0x00018100, GPIO_Num1: 0x00018140	Table 1247, p. 1422
GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1)	GPIO_Num0: 0x00018104, GPIO_Num1: 0x00018144	Table 1248, p. 1422
GPIO_<32*n>_<32*n+31>_Blink Enable Register (n=0–1)	GPIO_Num0: 0x00018108, GPIO_Num1: 0x00018148	Table 1249, p. 1422
GPIO_<32*n>_<32*n+31>_Data In Polarity Register (n=0–1)	GPIO_Num0: 0x0001810C, GPIO_Num1: 0x0001814C	Table 1250, p. 1423
GPIO_<32*n>_<32*n+31>_Data In Register (n=0–1)	GPIO_Num0: 0x00018110, GPIO_Num1: 0x00018150	Table 1251, p. 1423
GPIO_<32*n>_<32*n+31>_Interrupt Cause Register (n=0–1)	GPIO_Num0: 0x00018114, GPIO_Num1: 0x00018154	Table 1252, p. 1423
GPIO_<32*n>_<32*n+31>_Interrupt Mask Register (n=0–1)	GPIO_Num0: 0x00018118, GPIO_Num1: 0x00018158	Table 1253, p. 1423
GPIO_<32*n>_<32*n+31>_Interrupt Level Mask Register (n=0–1)	GPIO_Num0: 0x0001811C, GPIO_Num1: 0x0001815C	Table 1254, p. 1424
GPIO_<32*n>_<32*n+31>_Blink_Counter_Select Register (n=0–1)	GPIO_Num0: 0x00018120, GPIO_Num1: 0x00018160	Table 1255, p. 1424
GPIO_<32*n>_<32*n+31>_Control_Set Register (n=0–1)	GPIO_Num0: 0x00018128, GPIO_Num1: 0x00018168	Table 1256, p. 1424
GPIO_<32*n>_<32*n+31>_Control_Clear Register (n=0–1)	GPIO_Num0: 0x0001812C, GPIO_Num1: 0x0001816C	Table 1257, p. 1424
GPIO_<32*n>_<32*n+31>_ Data_Out_Set Register (n=0–1)	GPIO_Num0: 0x00018130, GPIO_Num1: 0x00018170	Table 1258, p. 1425
GPIO_<32n>_<32*n+31>_Data_Out_Clear Register (n=0–1)	GPIO_Num0: 0x00018134, GPIO_Num1: 0x00018174	Table 1259, p. 1425
GPIO_64_66_DataOut Register	0x00018180	Table 1260, p. 1425
GPIO_64_66_Data Out Enable Control Register	0x00018184	Table 1261, p. 1425
GPIO_64_66_Blink Enable Register	0x00018188	Table 1262, p. 1425
GPIO_64_66_Data In Polarity Register	0x0001818C	Table 1263, p. 1426
GPIO_64_66_Data In Register	0x00018190	Table 1264, p. 1426
GPIO_64_66_Interrupt Cause Register	0x00018194	Table 1265, p. 1426
GPIO_64_66_Interrupt Mask Register	0x00018198	Table 1266, p. 1427
GPIO_64_66_Interrupt Level Mask Register	0x0001819C	Table 1267, p. 1427
GPIO_64_66_Blink_Counter_Select Register	0x000181A0	Table 1268, p. 1427
GPIO_64_66_Control_Set Register	0x000181A4	Table 1269, p. 1428
GPIO_64_66_Control_Clear Register	0x000181A8	Table 1270, p. 1428

Table 1246:Summary Map Table for the General Purpose Input/Output (GPIO) Registers

Register Name	Offset	Table and Page
GPIO_64_66_Data_Out_Set Register	0x000181AC	Table 1271, p. 1428
GPIO_64_66 Data Out Clear Register	0x000181B0	Table 1272, p. 1428
GPIO Blink Counter A is ON Duration Register	0x000181C0	Table 1273, p. 1428
GPIO Blink Counter A is OFF Duration Register	0x000181C4	Table 1274, p. 1429
GPIO Blink Counter B is ON Duration Register	0x000181C8	Table 1275, p. 1429
GPIO Blink Counter B is OFF Duration Register	0x000181CC	Table 1276, p. 1429
GPIO per CPU		
CPU<m>_GPIO_<32*n>_<32*n+31>_Interrupt Cause Register (m=0–3, n=0–1)	GPIO_Num0CPU_Num0: 0x00018800, GPIO_Num0CPU_Num1: 0x00018804, GPIO_Num0CPU_Num2: 0x00018808, GPIO_Num0CPU_Num3: 0x0001880C, GPIO_Num1CPU_Num0: 0x00018840, GPIO_Num1CPU_Num1: 0x00018844, GPIO_Num1CPU_Num2: 0x00018848, GPIO_Num1CPU_Num3: 0x0001884C	Table 1277, p. 1429
CPU<m>_GPIO_<32*n>_<32*n+31>_Interrupt Mask Register (m=0–3, n=0–1)	GPIO_Num0CPU_Num0: 0x00018810, GPIO_Num0CPU_Num1: 0x00018814, GPIO_Num0CPU_Num2: 0x00018818, GPIO_Num0CPU_Num3: 0x0001881C, GPIO_Num1CPU_Num0: 0x00018850, GPIO_Num1CPU_Num1: 0x00018854, GPIO_Num1CPU_Num2: 0x00018858, GPIO_Num1CPU_Num3: 0x0001885C	Table 1278, p. 1430
CPU<m>_GPIO_<32*n>_<32*n+31>_Interrupt Level Mask Register (m=0–3, n=0–1)	GPIO_Num0CPU_Num0: 0x00018820, GPIO_Num0CPU_Num1: 0x00018824, GPIO_Num0CPU_Num2: 0x00018828, GPIO_Num0CPU_Num3: 0x0001882C, GPIO_Num1CPU_Num0: 0x00018860, GPIO_Num1CPU_Num1: 0x00018864, GPIO_Num1CPU_Num2: 0x00018868, GPIO_Num1CPU_Num3: 0x0001886C	Table 1279, p. 1430
CPU<m>_GPIO_64_66_Interrupt Cause Register (m=0–3)	CPU_Num0: 0x00018870, CPU_Num1: 0x00018874, CPU_Num2: 0x00018878, CPU_Num3: 0x0001887C	Table 1280, p. 1430

Table 1246:Summary Map Table for the General Purpose Input/Output (GPIO) Registers

Register Name	Offset	Table and Page
CPU<m>_GPIO_64_66_Interrupt Mask Register (m=0–3)	CPU_Num0: 0x00018880, CPU_Num1: 0x00018884, CPU_Num2: 0x00018888, CPU_Num3: 0x0001888C	Table 1281, p. 1431
CPU<m>_GPIO_64_66_Interrupt Level Mask Register (m=0–3)	CPU_Num0: 0x00018890, CPU_Num1: 0x00018894, CPU_Num2: 0x00018898, CPU_Num3: 0x0001889C	Table 1282, p. 1431

A.18.1 GPIO

Table 1247:GPIO_<32*n>_<32*n+31>_DataOut Register (n=0–1)

Offset: GPIO_Num0: 0x00018100, GPIO_Num1: 0x00018140

Offset Formula: 0x00018100+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32*n>_<32*n+31>_DataOut	RW 0x0	GPIO Output Pins Value One bit for each GPIO pin.

Table 1248:GPIO_<32*n>_<32*n+31>_Data Out Enable Control Register (n=0–1)

Offset: GPIO_Num0: 0x00018104, GPIO_Num1: 0x00018144

Offset Formula: 0x00018104+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32*n>_<32*n+31>_DOutEn	RW 0xFFFFFFFF	GPIO Port Output Enable This field is active low. Data is driven when the corresponding bit value is 0.

Table 1249:GPIO_<32*n>_<32*n+31>_Blink Enable Register (n=0–1)

Offset: GPIO_Num0: 0x00018108, GPIO_Num1: 0x00018148

Offset Formula: 0x00018108+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32*n>_<32*n+31>_DBlink	RW 0x0	GPIO Data Blink When set and the corresponding bit in GPIO Data Out Enable Control register is enabled, the GPIO pin blinks every ~100 ms (a period of 2^24 TCLK clocks).

Table 1250:GPIO_<32*n>_<32*n+31>_Data In Polarity Register (n=0–1)

Offset: GPIO_Num0: 0x0001810C, GPIO_Num1: 0x0001814C

Offset Formula: 0x0001810C+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32*n>_ <32*n+31>_ DataInActLow	RW 0x0	GPIO Data in Active Low When set to 1, the GPIO Data In register reflects the inverted value of the corresponding pin.

Table 1251:GPIO_<32*n>_<32*n+31>_Data In Register (n=0–1)

Offset: GPIO_Num0: 0x00018110, GPIO_Num1: 0x00018150

Offset Formula: 0x00018110+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32*n>_ <32*n+31>_DIn	RO 0x0	Each bit in this field reflects the value of the corresponding GPIO pin. If the corresponding bit in the GPIO Data In Polarity register is cleared to 0, the bit reflects the value of the pin with no change. If the corresponding bit in the GPIO Data In Polarity register is set to 1, the bit reflects the pin's inverted value.

Table 1252:GPIO_<32*n>_<32*n+31>_Interrupt Cause Register (n=0–1)

Offset: GPIO_Num0: 0x00018114, GPIO_Num1: 0x00018154

Offset Formula: 0x00018114+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<n*32>_ <n*32+31>_Int	RW0C 0x0	A bit in this field is set on transition of the corresponding bit in the <GPIODIn> field in the GPIO Data In register, from 0 to 1.

Table 1253:GPIO_<32*n>_<32*n+31>_Interrupt Mask Register (n=0–1)

Offset: GPIO_Num0: 0x00018118, GPIO_Num1: 0x00018158

Offset Formula: 0x00018118+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32*n>_ <32*n+31>_ IntEdgeMask	RW 0x0	GPIO Interrupt Edge Sensitive Mask The mask bit for each cause bit in the <GPIOInt> field of the GPIO Interrupt Cause register. The mask only affects assertion of the interrupt bits in Main Interrupt Cause register. It does not affect the setting of bits in the GPIO Interrupt Cause register. 0=Interrupt mask; 1=Interrupt enable;

Table 1254:GPIO_<32*n>_<32*n+31>_Interrupt Level Mask Register (n=0–1)

Offset: GPIO_Num0: 0x0001811C, GPIO_Num1: 0x0001815C

Offset Formula: 0x0001811C+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
To set an edge sensitive interrupt, set the corresponding bit in the GPIO Interrupt Mask Register. To set a level sensitive interrupt, set the corresponding bit in the GPIO Interrupt Level Mask Register.			
31:0	GPIO_<32*n>_<32*n+31>_IntLevelMask	RW 0x0	GPIO Interrupt Level Sensitive Mask The mask bit for each bit in the <GPIODIn> field of the GPIO Data In register. The mask only affects assertion of the Interrupt bit in Main Interrupt Cause register. It does not affect the value of bits in the GPIO Data In register. 0=Interrupt mask; 1=Interrupt enable;

Table 1255:GPIO_<32*n>_<32*n+31>_Blink_Counter_Select Register (n=0–1)

Offset: GPIO_Num0: 0x00018120, GPIO_Num1: 0x00018160

Offset Formula: 0x00018120+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32*n>_<32*n+31>_Blink_Counter_Select	RW 0x0	For each GPIO, this bit selects according to which counter it will blink (if the Blink Enable bit is set to 1'b1): 0 = Blink Counter A 1 = Blink Counter B

Table 1256:GPIO_<32*n>_<32*n+31>_Control_Set Register (n=0–1)

Offset: GPIO_Num0: 0x00018128, GPIO_Num1: 0x00018168

Offset Formula: 0x00018128+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIOD_<32*n>_<32*n+31>_OutEnSet	WO 0x0	Write 1 to a bit, sets the corresponding bit in GPIO Data Out Enable register. Write 0 has no affect. Read from this register returns 0.

Table 1257:GPIO_<32*n>_<32*n+31>_Control_Clear Register (n=0–1)

Offset: GPIO_Num0: 0x0001812C, GPIO_Num1: 0x0001816C

Offset Formula: 0x0001812C+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIODOutEn_<32*n>_<32*n+31>_Clear	WO 0x0	Write 1 to a bit, clears the corresponding bit in GPIO Data Out Enable register. Write 0 has no affect. Read from this register returns 0.

Table 1258:GPIO_<32*n>_<32*n+31>_ Data_Out_Set Register (n=0–1)

Offset: GPIO_Num0: 0x00018130, GPIO_Num1: 0x00018170

Offset Formula: 0x00018130+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO<32*n>_ <32*n+31>_ DOutEnSet	WO 0x0	Write 1 to a bit, sets the corresponding bit in GPIO Data Out register. Write 0 has no affect. Read from this register returns 0.

Table 1259:GPIO_<32n>_<32*n+31>_Data_Out_Clear Register (n=0–1)

Offset: GPIO_Num0: 0x00018134, GPIO_Num1: 0x00018174

Offset Formula: 0x00018134+0x40*n: where n (0-1) represents GPIO_Num

Bit	Field	Type/InitVal	Description
31:0	GPIO_<32n>_ <32*n+31>_DOut Clear	WO 0x0	Write 1 to a bit, clears the corresponding bit in GPIO Data Out register. Write 0 has no affect. Read from this register returns 0.

Table 1260:GPIO_64_66_DataOut Register

Offset: 0x00018180

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_ DataOut	RW 0x0	GPIO Output Pins Value One bit for each GPIO pin.

Table 1261:GPIO_64_66_Data Out Enable Control Register

Offset: 0x00018184

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_ DOutEn	RW 0x7	GPIO Port Output Enable This field is active low. Data is driven when the corresponding bit value is 0.

Table 1262:GPIO_64_66_Blink Enable Register

Offset: 0x00018188

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved

Table 1262:GPIO_64_66_Blink Enable Register (Continued)

Offset: 0x00018188

Bit	Field	Type/InitVal	Description
2:0	GPIO_64_66_ DBlink	RW 0x0	GPIO Data Blink When set and the corresponding bit in GPIO Data Out Enable Control register is enabled, the GPIO pin blinks every ~100 ms (a period of 2^24 TCLK clocks).

Table 1263:GPIO_64_66_Data In Polarity Register

Offset: 0x0001818C

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_ DataInActLow	RW 0x0	GPIO Data in Active Low When set to 1, the GPIO Data In register reflects the inverted value of the corresponding pin.

Table 1264:GPIO_64_66_Data In Register

Offset: 0x00018190

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_DIn	RO 0x0	Each bit in this field reflects the value of the corresponding GPIO pin. If the corresponding bit in the GPIO Data In Polarity register is cleared to 0, the bit reflects the value of the pin with no change. If the corresponding bit in the GPIO Data In Polarity register is set to 1, the bit reflects the pin's inverted value.

Table 1265:GPIO_64_66_Interrupt Cause Register

Offset: 0x00018194

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_Int	RW0C 0x0	A bit in this field is set on transition of the corresponding bit in the <GPIODIn> field in the GPIO Data In register, from 0 to 1.

Table 1266:GPIO_64_66_Interrupt Mask Register

Offset: 0x00018198

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_ IntEdgeMask	RW 0x0	GPIO Interrupt Edge Sensitive Mask The mask bit for each cause bit in the <GPIOInt> field of the GPIO Interrupt Cause register. The mask only affects assertion of the interrupt bits in Main Interrupt Cause register. It does not affect the setting of bits in the GPIO Interrupt Cause register. 0=Interrupt mask; 1=Interrupt enable;

Table 1267:GPIO_64_66_Interrupt Level Mask Register

Offset: 0x0001819C

Bit	Field	Type/InitVal	Description
To set an edge sensitive interrupt, set the corresponding bit in the GPIO Interrupt Mask Register. To set a level sensitive interrupt, set the corresponding bit in the GPIO Interrupt Level Mask Register.			
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_ IntLevelMask	RW 0x0	GPIO Interrupt Level Sensitive Mask The mask bit for each bit in the <GPIODIn> field of the GPIO Data In register. The mask only affects assertion of the Interrupt bit in Main Interrupt Cause register. It does not affect the value of bits in the GPIO Data In register. 0=Interrupt mask; 1=Interrupt enable;

Table 1268:GPIO_64_66_Blink_Counter_Select Register

Offset: 0x000181A0

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_Blink_ Counter_Select	RW 0x0	For each GPIO, this bit selects according to which counter it will blink (if the Blink Enable bit is set to 1’b1): 0 = Blink Counter A 1 = Blink Counter B

Table 1269:GPIO_64_66_Control_Set Register

Offset: 0x000181A4

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIOD_64_66_ OutEnSet	WO 0x0	Write 1 to a bit, sets the corresponding bit in GPIO Data Out Enable register. Write 0 has no affect. Read from this register returns 0.

Table 1270:GPIO_64_66_Control_Clear Register

Offset: 0x000181A8

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIODOutEn_64_ 66_Clear	WO 0x0	Write 1 to a bit, clears the corresponding bit in GPIO Data Out Enable register. Write 0 has no affect. Read from this register returns 0.

Table 1271:GPIO_64_66_Data_Out_Set Register

Offset: 0x000181AC

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_ DOutSet	WO 0x0	Write 1 to a bit, sets the corresponding bit in GPIO Data Out register. Write 0 has no affect. Read from this register returns 0.

Table 1272:GPIO_64_66 Data Out Clear Register

Offset: 0x000181B0

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	GPIO_64_66_DOut Clear	WO 0x0	Write 1 to a bit, clears the corresponding bit in GPIO Data Out register. Write 0 has no affect. Read from this register returns 0.

Table 1273:GPIO Blink Counter A is ON Duration Register

Offset: 0x000181C0

Bit	Field	Type/InitVal	Description
31:0	CounterAOnDur	RW 0x1000000	Counter A ON duration in terms of core clock cycles.

Table 1274:GPIO Blink Counter A is OFF Duration Register

Offset: 0x000181C4

Bit	Field	Type/InitVal	Description
31:0	CounterAOffDur	RW 0x1000000	Blink Counter A OFF duration in terms of core clock cycles.

Table 1275:GPIO Blink Counter B is ON Duration Register

Offset: 0x000181C8

Bit	Field	Type/InitVal	Description
31:0	CounterBOnDur	RW 0x1000000	Blink Counter B ON duration in terms of core clock cycles.

Table 1276:GPIO Blink Counter B is OFF Duration Register

Offset: 0x000181CC

Bit	Field	Type/InitVal	Description
31:0	CounterBOffDur	RW 0x1000000	Blink Counter B OFF duration in terms of core clock cycles.

A.18.2 GPIO per CPU

Table 1277:CPU<m>_GPIO_<32*n>_<32*n+31>_Interrupt Cause Register (m=0–3, n=0–1)

Offset: GPIO_Num0CPU_Num0: 0x00018800, GPIO_Num0CPU_Num1: 0x00018804, GPIO_Num0CPU_Num2: 0x00018808, GPIO_Num0CPU_Num3: 0x0001880C, GPIO_Num1CPU_Num0: 0x00018840, GPIO_Num1CPU_Num1: 0x00018844, GPIO_Num1CPU_Num2: 0x00018848, GPIO_Num1CPU_Num3: 0x0001884C

Offset Formula: 0x00018800+0x40*n+0x4*m: where n (0-1) represents GPIO_Num, where m (0-3) represents CPU_Num

Bit	Field	Type/InitVal	Description
31:0	CPU<m>_GPIO_<n*32>_<n*32+31>_Int	RW0C 0x0	A bit in this field is set on transition of the corresponding bit in the <GPIODIn> field in the GPIO Data In register, from 0 to 1.

Table 1278:CPU<m>_GPIO_<32*n>_<32*n+31>_Interrupt Mask Register (m=0–3, n=0–1)

Offset: **GPIO_Num0CPU_Num0: 0x00018810, GPIO_Num0CPU_Num1: 0x00018814, GPIO_Num0CPU_Num2: 0x00018818, GPIO_Num0CPU_Num3: 0x0001881C, GPIO_Num1CPU_Num0: 0x00018850, GPIO_Num1CPU_Num1: 0x00018854, GPIO_Num1CPU_Num2: 0x00018858, GPIO_Num1CPU_Num3: 0x0001885C**

Offset Formula: **0x00018810+0x40*n+0x4*m: where n (0-1) represents GPIO_Num, where m (0-3) represents CPU_Num**

Bit	Field	Type/InitVal	Description
31:0	CPU<m>_GPIO_<32*n>_<32*n+31>_IntEdgeMask	RW 0x0	GPIO Interrupt Edge Sensitive Mask The mask bit for each cause bit in the <GPIOInt> field of the GPIO Interrupt Cause register. The mask only affects assertion of the interrupt bits in Main Interrupt Cause register. It does not affect the setting of bits in the GPIO Interrupt Cause register. 0=Interrupt mask; 1=Interrupt enable;

Table 1279:CPU<m>_GPIO_<32*n>_<32*n+31>_Interrupt Level Mask Register (m=0–3, n=0–1)

Offset: **GPIO_Num0CPU_Num0: 0x00018820, GPIO_Num0CPU_Num1: 0x00018824, GPIO_Num0CPU_Num2: 0x00018828, GPIO_Num0CPU_Num3: 0x0001882C, GPIO_Num1CPU_Num0: 0x00018860, GPIO_Num1CPU_Num1: 0x00018864, GPIO_Num1CPU_Num2: 0x00018868, GPIO_Num1CPU_Num3: 0x0001886C**

Offset Formula: **0x00018820+0x40*n+0x4*m: where n (0-1) represents GPIO_Num, where m (0-3) represents CPU_Num**

Bit	Field	Type/InitVal	Description
To set an edge sensitive interrupt, set the corresponding bit in the GPIO Interrupt Mask Register. To set a level sensitive interrupt, set the corresponding bit in the GPIO Interrupt Level Mask Register.			
31:0	CPU<m>_GPIO_<32*n>_<32*n+31>_IntLevelMask	RW 0x0	GPIO Interrupt Level Sensitive Mask The mask bit for each bit in the <GPIODIn> field of the GPIO Data In register. The mask only affects assertion of the Interrupt bit in Main Interrupt Cause register. It does not affect the value of bits in the GPIO Data In register. 0=Interrupt mask; 1=Interrupt enable;

Table 1280:CPU<m>_GPIO_64_66_Interrupt Cause Register (m=0–3)

Offset: **CPU_Num0: 0x00018870, CPU_Num1: 0x00018874, CPU_Num2: 0x00018878, CPU_Num3: 0x0001887C**

Offset Formula: **0x00018870+0x4*m: where m (0-3) represents CPU_Num**

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	CPU<m>_GPIO_64_66_Int	RW0C 0x0	A bit in this field is set on transition of the corresponding bit in the <GPIODIn> field in the GPIO Data In register, from 0 to 1.

Table 1281:CPU<m>_GPIO_64_66_Interrupt Mask Register (m=0–3)

Offset: CPU_Num0: 0x00018880, CPU_Num1: 0x00018884, CPU_Num2: 0x00018888, CPU_Num3: 0x0001888C

Offset Formula: 0x00018880+0x4*m: where m (0-3) represents CPU_Num

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2:0	CPU<m>_GPIO_64_66_IntEdgeMask	RW 0x0	GPIO Interrupt Edge Sensitive Mask The mask bit for each cause bit in the <GPIOInt> field of the GPIO Interrupt Cause register. The mask only affects assertion of the interrupt bits in Main Interrupt Cause register. It does not affect the setting of bits in the GPIO Interrupt Cause register. 0=Interrupt mask; 1=Interrupt enable;

Table 1282:CPU<m>_GPIO_64_66_Interrupt Level Mask Register (m=0–3)

Offset: CPU_Num0: 0x00018890, CPU_Num1: 0x00018894, CPU_Num2: 0x00018898, CPU_Num3: 0x0001889C

Offset Formula: 0x00018890+0x4*m: where m (0-3) represents CPU_Num

Bit	Field	Type/InitVal	Description
To set an edge sensitive interrupt, set the corresponding bit in the GPIO Interrupt Mask Register. To set a level sensitive interrupt, set the corresponding bit in the GPIO Interrupt Level Mask Register.			
31:3	Reserved	RSVD 0x0	Reserved
2:0	CPU<m>_GPIO_64_66_IntLevelMask	RW 0x0	GPIO Interrupt Level Sensitive Mask The mask bit for each bit in the <GPIODIn> field of the GPIO Data In register. The mask only affects assertion of the Interrupt bit in Main Interrupt Cause register. It does not affect the value of bits in the GPIO Data In register. 0=Interrupt mask; 1=Interrupt enable;

A.19 Multi-Purpose Ports (MPP) Registers

The following table provides a summarized list of all registers that belong to the Multi-Purpose Ports (MPP), including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1283:Summary Map Table for the Multi-Purpose Ports (MPP) Registers

Register Name	Offset	Table and Page
MPP_Control_<n*8>_<n*8+7> Register (n=0–7)	MPP_Num0: 0x00018000, MPP_Num1: 0x00018004, MPP_Num2: 0x00018008, MPP_Num3: 0x0001800C, MPP_Num4: 0x00018010, MPP_Num5: 0x00018014, MPP_Num6: 0x00018018, MPP_Num7: 0x0001801C	Table 1284, p. 1432
MPP_Control_64_66 Register	0x00018020	Table 1285, p. 1433
Sample at Reset Register	0x00018230	Table 1286, p. 1434
Sample at Reset High Register	0x00018234	Table 1287, p. 1434

Table 1284:MPP_Control_<n*8>_<n*8+7> Register (n=0–7)

Offset: MPP_Num0: 0x00018000, MPP_Num1: 0x00018004, MPP_Num2: 0x00018008, MPP_Num3: 0x0001800C, MPP_Num4: 0x00018010, MPP_Num5: 0x00018014, MPP_Num6: 0x00018018, MPP_Num7: 0x0001801C

Offset Formula: 0x00018000+n*4: where n (0-7) represents MPP_Num

Bit	Field	Type/InitVal	Description
31:28	MPPSel<n*8+7>	RW 0x0	MPP<n*8+7> Select See the MPP Function Summary table in Hardware Specifications for this device.
27:24	MPPSel<n*8+6>	RW 0x0	MPP<n*8+6> Select See the MPP Function Summary table in Hardware Specifications for this device.
23:20	MPPSel<n*8+5>	RW 0x0	MPP<n*8+5> Select See the MPP Function Summary table in Hardware Specifications for this device.
19:16	MPPSel<n*8+4>	RW 0x0	MPP<n*8+4> Select See the MPP Function Summary table in Hardware Specifications for this device.

Table 1284:MPP_Control_<n*8>_<n*8+7> Register (n=0–7) (Continued)

Offset: MPP_Num0: 0x00018000, MPP_Num1: 0x00018004, MPP_Num2: 0x00018008, MPP_Num3: 0x0001800C, MPP_Num4: 0x00018010, MPP_Num5: 0x00018014, MPP_Num6: 0x00018018, MPP_Num7: 0x0001801C

Offset Formula: 0x00018000+n*4: where n (0-7) represents MPP_Num

Bit	Field	Type/InitVal	Description
15:12	MPPSel<n*8+3>	RW 0x0	MPP<n*8+3> Select See the MPP Function Summary table in Hardware Specifications for this device.
11:8	MPPSel<n*8+2>	RW 0x0	MPP<n*8+2> Select See the MPP Function Summary table in Hardware Specifications for this device.
7:4	MPPSel<n*8+1>	RW 0x0	MPP<n*8+1> Select See the MPP Function Summary table in Hardware Specifications for this device.
3:0	MPPSel<n*8>	RW 0x0	MPP<n*8> Select See the MPP Function Summary table in Hardware Specifications for this device.

Table 1285:MPP_Control_64_66 Register

Offset: 0x00018020

Bit	Field	Type/InitVal	Description
31:12	MPP_Reserved	RW 0x0	Reserved
11:8	MPPSel66	RW 0x0	MPP66 Select See the MPP Function Summary table in Hardware Specifications for this device.
7:4	MPPSel65	RW 0x0	MPP65 Select See the MPP Function Summary table in Hardware Specifications for this device.
3:0	MPPSel64	RW 0x0	MPP64 Select See the MPP Function Summary table in Hardware Specifications for this device.

Table 1286:Sample at Reset Register

Offset: 0x00018230

Bit	Field	Type/InitVal	Description
31:0	SampleAtReset [31:0]	RW SAR	**NOTE:** Initial value is sampled at reset. See the Hardware Specification for more details, including bit locations and field descriptions.

Table 1287:Sample at Reset High Register

Offset: 0x00018234

Bit	Field	Type/InitVal	Description
31:0	Sample at Reset [63:32]	RW SAR	**NOTE:** Initial value is sampled at reset. See the Hardware Specification for more details, including bit locations and field descriptions.

A.20 Power Management Unit (PMU) Registers

The following table provides a summarized list of all registers that belong to the Power Management Unit, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1288:Summary Map Table for the Power Management Unit Registers

Register Name	Offset	Table and Page
PMU Power Domain Delay Select Register	0x0001C010	Table 1289, p. 1435
PMU Power Up Delay 0 Register	0x0001C014	Table 1290, p. 1436
PMU Power Up Delay 1 Register	0x0001C018	Table 1291, p. 1436
PMU DFS Control 1 Register	0x0001C054	Table 1292, p. 1437
PMU DFS Control 2 Register	0x0001C058	Table 1293, p. 1437
PMU DFS Control 3 Register	0x0001C05C	Table 1294, p. 1437
PMU DFS Control 4 Register	0x0001C060	Table 1295, p. 1438
PMU Interrupt Cause Register	0x0001C120	Table 1296, p. 1438
PMU Interrupt Mask Register	0x0001C124	Table 1297, p. 1438

Table 1289:PMU Power Domain Delay Select Register

Offset: 0x0001C010

Bit	Field	Type/InitVal	Description
31:5	Reserved	RSVD 0x0	Reserved
4	PmuPdDelaySel4	RW 0x0	Controls which delay value will be defined as the delay between the power down control signal de-assertion, and the actual power stabilization. 0 = Delay0: Use the delay value specified by PMU Power Up Delay 0 register. 1 = Delay1: Use the delay value specified by PMU Power Up Delay 1 register.
3	PmuPdDelaySel3	RW 0x0	Controls which delay value will be defined as the delay between the power down control signal de-assertion, and the actual power stabilization. 0 = Delay0: Use the delay value specified by PMU Power Up Delay 0 register. 1 = Delay1: Use the delay value specified by PMU Power Up Delay 1 register.

Copyright © 2014 Marvell

Table 1289:PMU Power Domain Delay Select Register (Continued)

Offset: 0x0001C010

Bit	Field	Type/InitVal	Description
2	PmuPdDelaySel2	RW 0x0	Controls which delay value will be defined as the delay between the power down control signal de-assertion, and the actual power stabilization. 0 = Delay0: Use the delay value specified by PMU Power Up Delay 0 register. 1 = Delay1: Use the delay value specified by PMU Power Up Delay 1 register.
1	PmuPdDelaySel1	RW 0x0	Controls which delay value will be defined as the delay between the power down control signal de-assertion, and the actual power stabilization. 0 = Delay0: Use the delay value specified by PMU Power Up Delay 0 register. 1 = Delay1: Use the delay value specified by PMU Power Up Delay 1 register.
0	PmuPdDelaySel0	RW 0x0	Controls which delay value will be defined as the delay between the power down control signal de-assertion, and the actual power stabilization. 0 = Delay0: Use the delay value specified by PMU Power Up Delay 0 register. 1 = Delay1: Use the delay value specified by PMU Power Up Delay 1 register.

Table 1290:PMU Power Up Delay 0 Register

Offset: 0x0001C014

Bit	Field	Type/InitVal	Description
31:0	PmuPowerUpDelay 0	RW 0x0100	Controls the delay between the power down signal de-assertion and the actual power stabilization. The delay value is counted in core clock cycles. For each power domain, the user can specify whether to use the Delay0 or Delay1 values.

Table 1291:PMU Power Up Delay 1 Register

Offset: 0x0001C018

Bit	Field	Type/InitVal	Description
31:0	PmuPowerUpDelay 1	RW 0xa	Controls the delay between the power down signal de-assertion and the actual power stabilization. The delay value is counted in core clock cycles. For each power domain, the user can specify whether to use the Delay0 or Delay1 values.

Copyright © 2014 Marvell

Table 1292:PMU DFS Control 1 Register

Offset: 0x0001C054

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:24	DfsInitRatio1	RW SAR	Saves the Reset-Strap divider ratio. (Which is a function of "CPU Clock Frequency Select" and "Fabric Frequency Options" reset configurations)
23:16	DfsRatio1	RW 0x0	Defines the required ratio the DFS flow will change the divider to.
15:0	Reserved	RSVD 0x406	Reserved

Table 1293:PMU DFS Control 2 Register

Offset: 0x0001C058

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:24	DfsInitRatio2	RW SAR	Saves the Reset-Strap divider ratio. (Which is a function of "CPU Clock Frequency Select" and "Fabric Frequency Options" reset configurations)
23:16	DfsRatio2	RW 0x0	Defines the required ratio the DFS flow will change the divider to.
15:0	Reserved	RSVD 0x507	Reserved

Table 1294:PMU DFS Control 3 Register

Offset: 0x0001C05C

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:24	DfsInitRatio3	RW SAR	Saves the Reset-Strap divider ratio. (Which is a function of "CPU Clock Frequency Select" and "Fabric Frequency Options" reset configurations)
23:16	DfsRatio3	RW 0x0	Defines the required ratio the DFS flow will change the divider to.
15:0	Reserved	RSVD 0x608	Reserved

Table 1295:PMU DFS Control 4 Register

Offset: 0x0001C060

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:24	DfsInitRatio4	RW SAR	Saves the Reset-Strap divider ratio. (Which is a function of "CPU Clock Frequency Select" and "Fabric Frequency Options" reset configurations)
23:16	DfsRatio4	RW 0x0	Defines the required ratio the DFS flow will change the divider to.
15:0	Reserved	RSVD 0x709	Reserved

Table 1296:PMU Interrupt Cause Register

Offset: 0x0001C120

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	ThermOverheatInt	RW0C 0x0	Thermal Overheat interrupt
0	ThermCoolingInt	RW0C 0x0	Thermal Cooling Interrupt

Table 1297:PMU Interrupt Mask Register

Offset: 0x0001C124

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1:0	PmuIntrMask	RW 0x0	Mask resigter, for each of the interrupt fields in the PMU Interrupt Cause

A.21 Cryptographic Engine and Security Accelerator (CESA) Registers

The following table provides a summarized list of all registers that belong to the Cryptographic Engine and Security Accelerator (CESA), including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1298:Summary Map Table for the Cryptographic Engine and Security Accelerator (CESA) Registers

Register Name	Offset	Table and Page
AES Encryption Interface		
AES Encryption Key Column 7 Register (p=0–1)	Tunit_port0: 0x0009DD80, Tunit_port1: 0x0009FD80	Table 1299, p. 1443
AES Encryption Key Column 6 Register (p=0–1)	Tunit_port0: 0x0009DD84, Tunit_port1: 0x0009FD84	Table 1300, p. 1443
AES Encryption Key Column 5 Register (p=0–1)	Tunit_port0: 0x0009DD88, Tunit_port1: 0x0009FD88	Table 1301, p. 1443
AES Encryption Key Column 4 Register (p=0–1)	Tunit_port0: 0x0009DD8C, Tunit_port1: 0x0009FD8C	Table 1302, p. 1443
AES Encryption Key Column 3 Register (p=0–1)	Tunit_port0: 0x0009DD90, Tunit_port1: 0x0009FD90	Table 1303, p. 1443
AES Encryption Key Column 2 Register (p=0–1)	Tunit_port0: 0x0009DD94, Tunit_port1: 0x0009FD94	Table 1304, p. 1444
AES Encryption Key Column 1 Register (p=0–1)	Tunit_port0: 0x0009DD98, Tunit_port1: 0x0009FD98	Table 1305, p. 1444
AES Encryption Key Column 0 Register (p=0–1)	Tunit_port0: 0x0009DD9C, Tunit_port1: 0x0009FD9C	Table 1306, p. 1444
AES Encryption Data In/Out Column 3 Register (p=0–1)	Tunit_port0: 0x0009DDA0, Tunit_port1: 0x0009FDA0	Table 1307, p. 1444
AES Encryption Data In/Out Column 2 Register (p=0–1)	Tunit_port0: 0x0009DDA4, Tunit_port1: 0x0009FDA4	Table 1308, p. 1444
AES Encryption Data In/Out Column 1 Register (p=0–1)	Tunit_port0: 0x0009DDA8, Tunit_port1: 0x0009FDA8	Table 1309, p. 1445
AES Encryption Data In/Out Column 0 Register (p=0–1)	Tunit_port0: 0x0009DDAC, Tunit_port1: 0x0009FDAC	Table 1310, p. 1445
AES Encryption Command Register (p=0–1)	Tunit_port0: 0x0009DDB0, Tunit_port1: 0x0009FDB0	Table 1311, p. 1445
AES Decryption Interface		
AES Decryption Key Column 7 Register (p=0–1)	Tunit_port0: 0x0009DDC0, Tunit_port1: 0x0009FDC0	Table 1312, p. 1446
AES Decryption Key Column 6 Register (p=0–1)	Tunit_port0: 0x0009DDC4, Tunit_port1: 0x0009FDC4	Table 1313, p. 1446
AES Decryption Key Column 5 Register (p=0–1)	Tunit_port0: 0x0009DDC8, Tunit_port1: 0x0009FDC8	Table 1314, p. 1446
AES Decryption Key Column 4 Register (p=0–1)	Tunit_port0: 0x0009DDCC, Tunit_port1: 0x0009FDCC	Table 1315, p. 1446

Copyright © 2014 Marvell

Table 1298:Summary Map Table for the Cryptographic Engine and Security Accelerator (CESA) Registers (Continued)

Register Name	Offset	Table and Page
AES Decryption Key Column 3 Register (p=0–1)	Tunit_port0: 0x0009DDD0, Tunit_port1: 0x0009FDD0	Table 1316, p. 1447
AES Decryption Key Column 2 Register (p=0–1)	Tunit_port0: 0x0009DDD4, Tunit_port1: 0x0009FDD4	Table 1317, p. 1447
AES Decryption Key Column 1 Register (p=0–1)	Tunit_port0: 0x0009DDD8, Tunit_port1: 0x0009FDD8	Table 1318, p. 1447
AES Decryption Key Column 0 Register (p=0–1)	Tunit_port0: 0x0009DDDC, Tunit_port1: 0x0009FDDC	Table 1319, p. 1447
AES Decryption Data In/Out Column 3 Register (p=0–1)	Tunit_port0: 0x0009DDE0, Tunit_port1: 0x0009FDE0	Table 1320, p. 1447
AES Decryption Data In/Out Column 2 Register (p=0–1)	Tunit_port0: 0x0009DDE4, Tunit_port1: 0x0009FDE4	Table 1321, p. 1448
AES Decryption Data In/Out Column 1 Register (p=0–1)	Tunit_port0: 0x0009DDE8, Tunit_port1: 0x0009FDE8	Table 1322, p. 1448
AES Decryption Data In/Out Column 0 Register (p=0–1)	Tunit_port0: 0x0009DDEC, Tunit_port1: 0x0009FDEC	Table 1323, p. 1448
AES Decryption Command Register (p=0–1)	Tunit_port0: 0x0009DDF0, Tunit_port1: 0x0009FDF0	Table 1324, p. 1448
TDMA Interrupt		
TDMA Error Cause Register (p=0–1)	Tunit_port0: 0x000908C8, Tunit_port1: 0x000928C8	Table 1325, p. 1449
TDMA Error Mask Register (p=0–1)	Tunit_port0: 0x000908CC, Tunit_port1: 0x000928CC	Table 1326, p. 1450
Interrupt Cause		
Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1)	Tunit_port0: 0x0009DE20, Tunit_port1: 0x0009FE20	Table 1327, p. 1450
Cryptographic Engine/Security Accelerator/TDMA Interrupt Mask Register (p=0–1)	Tunit_port0: 0x0009DE24, Tunit_port1: 0x0009FE24	Table 1328, p. 1452
Cryptographic Interrupt Coalescing Threshold Register (p=0–1)	Tunit_port0: 0x0009DE30, Tunit_port1: 0x0009FE30	Table 1329, p. 1452
Cryptographic Interrupt Time Threshold Register (p=0–1)	Tunit_port0: 0x0009DE34, Tunit_port1: 0x0009FE34	Table 1330, p. 1453
TDMA Address Decoding		
Base Address 0 Register (p=0–1)	Tunit_port0: 0x00090A00, Tunit_port1: 0x00092A00	Table 1331, p. 1454
Window Control 0 Register (p=0–1)	Tunit_port0: 0x00090A04, Tunit_port1: 0x00092A04	Table 1332, p. 1454
Base Address 1 Register (p=0–1)	Tunit_port0: 0x00090A08, Tunit_port1: 0x00092A08	Table 1333, p. 1454
Window Control 1 Register (p=0–1)	Tunit_port0: 0x00090A0C, Tunit_port1: 0x00092A0C	Table 1334, p. 1455
Base Address 2 Register (p=0–1)	Tunit_port0: 0x00090A10, Tunit_port1: 0x00092A10	Table 1335, p. 1455
Window Control 2 Register (p=0–1)	Tunit_port0: 0x00090A14, Tunit_port1: 0x00092A14	Table 1336, p. 1455

Table 1298:Summary Map Table for the Cryptographic Engine and Security Accelerator (CESA) Registers (Continued)

Register Name	Offset	Table and Page
Base Address 3 Register (p=0–1)	Tunit_port0: 0x00090A18, Tunit_port1: 0x00092A18	Table 1337, p. 1456
Window Control 3 Register (p=0–1)	Tunit_port0: 0x00090A1C, Tunit_port1: 0x00092A1C	Table 1338, p. 1456
TDMA Control		
Control Register (p=0–1)	Tunit_port0: 0x00090840, Tunit_port1: 0x00092840	Table 1339, p. 1456
TDMA Descriptor		
TDMA Byte Count Register (p=0–1)	Tunit_port0: 0x00090800, Tunit_port1: 0x00092800	Table 1340, p. 1458
TDMA Source Address Register (p=0–1)	Tunit_port0: 0x00090810, Tunit_port1: 0x00092810	Table 1341, p. 1458
TDMA Destination Address Register (p=0–1)	Tunit_port0: 0x00090820, Tunit_port1: 0x00092820	Table 1342, p. 1459
Next Descriptor Pointer Register (p=0–1)	Tunit_port0: 0x00090830, Tunit_port1: 0x00092830	Table 1343, p. 1459
Current Descriptor Pointer Register (p=0–1)	Tunit_port0: 0x00090870, Tunit_port1: 0x00092870	Table 1344, p. 1459
DES Engine		
DES Initial Value Low Register (p=0–1)	Tunit_port0: 0x0009DD40, Tunit_port1: 0x0009FD40	Table 1345, p. 1459
DES Initial Value High Register (p=0–1)	Tunit_port0: 0x0009DD44, Tunit_port1: 0x0009FD44	Table 1346, p. 1459
DES Key0 Low Register (p=0–1)	Tunit_port0: 0x0009DD48, Tunit_port1: 0x0009FD48	Table 1347, p. 1460
DES Key0 High Register (p=0–1)	Tunit_port0: 0x0009DD4C, Tunit_port1: 0x0009FD4C	Table 1348, p. 1460
DES Key1 Low Register (p=0–1)	Tunit_port0: 0x0009DD50, Tunit_port1: 0x0009FD50	Table 1349, p. 1460
DES Key1 High Register (p=0–1)	Tunit_port0: 0x0009DD54, Tunit_port1: 0x0009FD54	Table 1350, p. 1460
DES Command Register (p=0–1)	Tunit_port0: 0x0009DD58, Tunit_port1: 0x0009FD58	Table 1351, p. 1460
DES Key2 Low Register (p=0–1)	Tunit_port0: 0x0009DD60, Tunit_port1: 0x0009FD60	Table 1352, p. 1462
DES Key2 High Register (p=0–1)	Tunit_port0: 0x0009DD64, Tunit_port1: 0x0009FD64	Table 1353, p. 1462
DES Data Buffer Low Register (p=0–1)	Tunit_port0: 0x0009DD70, Tunit_port1: 0x0009FD70	Table 1354, p. 1462
DES Data Buffer High Register (p=0–1)	Tunit_port0: 0x0009DD74, Tunit_port1: 0x0009FD74	Table 1355, p. 1462
DES Data Out Low Register (p=0–1)	Tunit_port0: 0x0009DD78, Tunit_port1: 0x0009FD78	Table 1356, p. 1462
DES Data Out High Register (p=0–1)	Tunit_port0: 0x0009DD7C, Tunit_port1: 0x0009FD7C	Table 1357, p. 1463

Table 1298:Summary Map Table for the Cryptographic Engine and Security Accelerator (CESA) Registers (Continued)

Register Name	Offset	Table and Page
Security Accelerator		
Security Accelerator Command Register (p=0–1)	Tunit_port0: 0x0009DE00, Tunit_port1: 0x0009FE00	Table 1358, p. 1463
Security Accelerator Descriptor Pointer Register (p=0–1)	Tunit_port0: 0x0009DE04, Tunit_port1: 0x0009FE04	Table 1359, p. 1463
Security Accelerator Configuration Register (p=0–1)	Tunit_port0: 0x0009DE08, Tunit_port1: 0x0009FE08	Table 1360, p. 1464
Security Accelerator Status Register (p=0–1)	Tunit_port0: 0x0009DE0C, Tunit_port1: 0x0009FE0C	Table 1361, p. 1465
SHA-1, SHA-2, and MD5 Interface		
SHA-1/SHA-2/MD5 Initial Value/Digest A Register (p=0–1)	Tunit_port0: 0x0009DD00, Tunit_port1: 0x0009FD00	Table 1362, p. 1465
SHA-1/SHA-2/MD5 Initial Value/Digest B Register (p=0–1)	Tunit_port0: 0x0009DD04, Tunit_port1: 0x0009FD04	Table 1363, p. 1466
SHA-1/SHA-2/MD5 Initial Value/Digest C Register (p=0–1)	Tunit_port0: 0x0009DD08, Tunit_port1: 0x0009FD08	Table 1364, p. 1466
SHA-1/SHA-2/MD5 Initial Value/Digest D Register (p=0–1)	Tunit_port0: 0x0009DD0C, Tunit_port1: 0x0009FD0C	Table 1365, p. 1466
SHA-1/SHA-2 Initial Value/Digest E Register (p=0–1)	Tunit_port0: 0x0009DD10, Tunit_port1: 0x0009FD10	Table 1366, p. 1466
SHA-1/SHA-2/MD5 Authentication Command Register (p=0–1)	Tunit_port0: 0x0009DD18, Tunit_port1: 0x0009FD18	Table 1367, p. 1467
SHA-1/SHA-2/MD5 Bit Count Low Register (p=0–1)	Tunit_port0: 0x0009DD20, Tunit_port1: 0x0009FD20	Table 1368, p. 1469
SHA-1/SHA-2/MD5 Bit Count High Register (p=0–1)	Tunit_port0: 0x0009DD24, Tunit_port1: 0x0009FD24	Table 1369, p. 1469
SHA-2 Initial Value/Digest F Register (p=0–1)	Tunit_port0: 0x0009DD28, Tunit_port1: 0x0009FD28	Table 1370, p. 1469
SHA-2 Initial Value/Digest G Register (p=0–1)	Tunit_port0: 0x0009DD2C, Tunit_port1: 0x0009FD2C	Table 1371, p. 1469
SHA-2 Initial Value/Digest H Register (p=0–1)	Tunit_port0: 0x0009DD30, Tunit_port1: 0x0009FD30	Table 1372, p. 1470
SHA-1/SHA-2/MD5 Data In Register (p=0–1)	Tunit_port0: 0x0009DD38, Tunit_port1: 0x0009FD38	Table 1373, p. 1470

A.21.1 AES Encryption Interface

Table 1299:AES Encryption Key Column 7 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD80, Tunit_port1: 0x0009FD80

Offset Formula: 0x0009DD80+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol7	RW 0x0	Contains Column 7 of the AES encryption key, or Column 7 of the decryption key when AES Key Read Mode is set.

Table 1300:AES Encryption Key Column 6 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD84, Tunit_port1: 0x0009FD84

Offset Formula: 0x0009DD84+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol6	RW 0x0	Contains Column 6 of the AES encryption key, or Column 6 of the decryption key when AES Key Read Mode is set.

Table 1301:AES Encryption Key Column 5 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD88, Tunit_port1: 0x0009FD88

Offset Formula: 0x0009DD88+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol5	RW 0x0	Contains Column 5 of the AES encryption key, or Column 5 of the decryption key when AES Key Read Mode is set.

Table 1302:AES Encryption Key Column 4 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD8C, Tunit_port1: 0x0009FD8C

Offset Formula: 0x0009DD8C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol4	RW 0x0	Contains Column 4 of the AES encryption key, or Column 4 of the decryption key when AES Key Read Mode is set.

Table 1303:AES Encryption Key Column 3 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD90, Tunit_port1: 0x0009FD90

Offset Formula: 0x0009DD90+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol3	RW 0x0	Contains Column 3 of the AES encryption key, or Column 3 of the decryption key when AES Key Read Mode is set.

Table 1304:AES Encryption Key Column 2 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD94, Tunit_port1: 0x0009FD94

Offset Formula: 0x0009DD94+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol2	RW 0x0	Contains Column 2 of the AES encryption key, or Column 2 of the decryption key when AES Key Read Mode is set.

Table 1305:AES Encryption Key Column 1 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD98, Tunit_port1: 0x0009FD98

Offset Formula: 0x0009DD98+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol1	RW 0x0	Contains Column 1 of the AES encryption key, or Column 1 of the decryption key when AES Key Read Mode is set.

Table 1306:AES Encryption Key Column 0 Register (p=0–1)

Offset: Tunit_port0: 0x0009DD9C, Tunit_port1: 0x0009FD9C

Offset Formula: 0x0009DD9C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncKeyCol0	RW 0x0	Contains Column 0 of the AES encryption key, or Column 0 of the decryption key when AES Key Read Mode is set.

Table 1307:AES Encryption Data In/Out Column 3 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDA0, Tunit_port1: 0x0009FDA0

Offset Formula: 0x0009DDA0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncDatCol3	RW 0x0	At first, this field contains Column 3 of the input data block to be encrypted. Once the AES completes the calculation, this field will contain Column 3 of the AES result.

Table 1308:AES Encryption Data In/Out Column 2 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDA4, Tunit_port1: 0x0009FDA4

Offset Formula: 0x0009DDA4+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncDatCol2	RW 0x0	At first, this field contains Column 2 of the input data block to be encrypted. Once the AES completes the calculation, this field will contain Column 2 of the AES result.

Table 1309:AES Encryption Data In/Out Column 1 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDA8, Tunit_port1: 0x0009FDA8

Offset Formula: 0x0009DDA8+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncDatCol1	RW 0x0	At first, this field contains Column 1 of the input data block to be encrypted. Once the AES completes the calculation, this field will contain Column 1 of the AES result.

Table 1310:AES Encryption Data In/Out Column 0 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDAC, Tunit_port1: 0x0009FDAC

Offset Formula: 0x0009DDAC+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesEncDatCol0	RW 0x0	At first, this field contains Column 0 of the input data block to be encrypted. Once the AES completes the calculation, this field will contain Column 0 of the AES result.

Table 1311:AES Encryption Command Register (p=0–1)

Offset: Tunit_port0: 0x0009DDB0, Tunit_port1: 0x0009FDB0

Offset Formula: 0x0009DDB0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31	Termination	RO 0x1	This bit is set by the engine to indicate completion of an AES calculation process. Any write operation to the encryption engine will clear this bit.
30:9	Reserved	RSVD 0x0	Reserved
8	OutByteSwap	RW 0x0	This bit controls whether byte swap is activated for output. 0 = No byte swap 1 = Byte swap
7:5	Reserved	RSVD 0x0	Reserved
4	DataByteSwap	RW 0x0	This bit controls whether data byte swap is activated upon input. 0 = No byte swap 1 = Byte swap
3:2	Reserved	RSVD 0x0	Reserved

Table 1311:AES Encryption Command Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x0009DDB0, Tunit_port1: 0x0009FDB0

Offset Formula: 0x0009DDB0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
1:0	AesEncKeyMode	RW 0x0	This field specifies the AES128 key size used. 0 = 128-bit key 1 = 192-bit key 2 = 256-bit key 3 = Reserved

A.21.2 AES Decryption Interface

Table 1312:AES Decryption Key Column 7 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDC0, Tunit_port1: 0x0009FDC0

Offset Formula: 0x0009DDC0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol7	RW 0x0	Contains Column 7 of the AES Decryption Key.

Table 1313:AES Decryption Key Column 6 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDC4, Tunit_port1: 0x0009FDC4

Offset Formula: 0x0009DDC4+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol6	RW 0x0	Contains Column 6 of the AES Decryption Key.

Table 1314:AES Decryption Key Column 5 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDC8, Tunit_port1: 0x0009FDC8

Offset Formula: 0x0009DDC8+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol5	RW 0x0	Contains Column 5 of the AES Decryption Key.

Table 1315:AES Decryption Key Column 4 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDCC, Tunit_port1: 0x0009FDCC

Offset Formula: 0x0009DDCC+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol4	RW 0x0	Contains Column 4 of the AES Decryption Key.

Table 1316:AES Decryption Key Column 3 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDD0, Tunit_port1: 0x0009FDD0

Offset Formula: 0x0009DDD0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol3	RW 0x0	Contains Column 3 of the AES Decryption Key.

Table 1317:AES Decryption Key Column 2 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDD4, Tunit_port1: 0x0009FDD4

Offset Formula: 0x0009DDD4+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol2	RW 0x0	Contains Column 2 of the AES Decryption Key.

Table 1318:AES Decryption Key Column 1 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDD8, Tunit_port1: 0x0009FDD8

Offset Formula: 0x0009DDD8+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol1	RW 0x0	Contains Column 1 of the AES Decryption Key.

Table 1319:AES Decryption Key Column 0 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDDC, Tunit_port1: 0x0009FDDC

Offset Formula: 0x0009DDDC+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecKeyCol0	RW 0x0	Contains Column 0 of the AES Decryption Key.

Table 1320:AES Decryption Data In/Out Column 3 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDE0, Tunit_port1: 0x0009FDE0

Offset Formula: 0x0009DDE0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecDatCol3	RW 0x0	At first, this field contains Column 3 of the input data block to be decrypted. Once the AES completes the calculation, this field will contain Column 3 of the AES result.

Table 1321:AES Decryption Data In/Out Column 2 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDE4, Tunit_port1: 0x0009FDE4

Offset Formula: 0x0009DDE4+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecDatCol2	RW 0x0	At first, this field contains Column 2 of the input data block to be decrypted. Once the AES completes the calculation, this field will contain Column 2 of the AES result.

Table 1322:AES Decryption Data In/Out Column 1 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDE8, Tunit_port1: 0x0009FDE8

Offset Formula: 0x0009DDE8+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecDatCol1	RW 0x0	At first, this field contains Column 1 of the input data block to be decrypted. Once the AES completes the calculation, this field will contain Column 1 of the AES result.

Table 1323:AES Decryption Data In/Out Column 0 Register (p=0–1)

Offset: Tunit_port0: 0x0009DDEC, Tunit_port1: 0x0009FDEC

Offset Formula: 0x0009DDEC+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	AesDecDatCol0	RW 0x0	At first, this field contains column 0 of the input data block to be decrypted. Once the AES completes the calculation, this field will contain Column 0 of the AES result.

Table 1324:AES Decryption Command Register (p=0–1)

Offset: Tunit_port0: 0x0009DDF0, Tunit_port1: 0x0009FDF0

Offset Formula: 0x0009DDF0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31	Termination	RO 0x1	This bit is set by the engine to indicate completion of an AES calculation process. Any write operation to the decryption engine will clear this bit.
30	AesDecKeyReady	RO 0x0	This bit is set to 1 whenever the key generation process is done. This bit is cleared to 0 whenever any of the AES Dec Key registers are written.
29:9	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 1324:AES Decryption Command Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x0009DDF0, Tunit_port1: 0x0009FDF0

Offset Formula: 0x0009DDF0+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
8	OutByteSwap	RW 0x0	This bit controls whether byte swap is activated for output. 0 = No byte swap 1 = Byte swap
7:5	Reserved	RSVD 0x0	Reserved
4	DataByteSwap	RW 0x0	This bit controls whether data byte swap is activated upon input. 0 = No byte swap 1 = Byte swap
3	Reserved	RSVD 0x0	Reserved
2	AesDecMakeKey	RW 0x0	This bits controls whether the decryption key is calculated in the engine prior to data decryption. 0 = NoDecrypt: No decryption key calculation 1 = Decrypt: Decryption key calculation
1:0	AesDecKeyMode	RW 0x0	These bits specify what AES128 key size is used. 0 = 128-bit key 1 = 192-bit key 2 = 256-bit key 3 = Reserved

A.21.3 TDMA Interrupt

Table 1325:TDMA Error Cause Register (p=0–1)

Offset: Tunit_port0: 0x000908C8, Tunit_port1: 0x000928C8

Offset Formula: 0x000908C8+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:7	Reserved	RSVD 0x0	Reserved
6	IPSecAuthDigestErr	RW0C 0x0	Sets when Authentication Decode Digest Error (comparison of original digest and calculated digest yields error indication) occurs.
5	IPSecParErrorPortB	RW0C 0x0	Sets when parity error occur on Z-domain of Accelerator Internal SRAM.
4	IPSecParErrorPortA	RW0C 0x0	Sets when parity error occur on T-domain of Accelerator Internal SRAM.
3	DataError	RW0C 0x0	Sets when MBUS parity error occur

Table 1325:TDMA Error Cause Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x000908C8, Tunit_port1: 0x000928C8

Offset Formula: 0x000908C8+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
2	BothHit	RW0C 0x0	Sets when the TDMA is enabled and configured so that both source and destination addresses hit any of the BARs.
1	DoubleHit	RW0C 0x0	Sets when the TDMA is enabled and configured so that the source or destination address hit is in multiple BARs.
0	Miss	RW0C 0x0	Sets when the TDMA is enabled and configured so that both source and destination addresses miss all BARs.

Table 1326:TDMA Error Mask Register (p=0–1)

Offset: Tunit_port0: 0x000908CC, Tunit_port1: 0x000928CC

Offset Formula: 0x000908CC+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:7	Reserved	RSVD 0x0	Reserved
6:0	Mask	RW 0x0	Mask Bit Per Err Cause Bit Mask only affects the assertion of an error pin. It does not affect the setting of bits in the Error Cause register. 0 = Masked: Error is masked. 1 = Enabled: Error is enabled

A.21.4 Interrupt Cause

Table 1327:Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1)

Offset: Tunit_port0: 0x0009DE20, Tunit_port1: 0x0009FE20

Offset Formula: 0x0009DE20+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
NOTE: The cryptographic engine has a dedicated Interrupt Cause register. This register is set by events occurring in the engine. Clearing this register's bits is done by writing 0 to the cause bits. Writing 1 to a bit has no effect. This register is shared by the DES and the Authentication engine.			
31:15	Reserved	RW0C 0x0	Reserved

Table 1327:Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x0009DE20, Tunit_port1: 0x0009FE20

Bit	Field	Type/InitVal	Description
14	EopCoalInt	RW0C 0x0	Cryptographic Interrupt Coalescing This bit is set: When the accumulated number of <AccAndTDMAInt_CM> indications since the last <EopCoalInt> negation - reaches the value set in the <Cryptographic Interrupt Coalescing Threshold> Register, or When the time from first <AccAndTDMAInt_CM> assertion after <EopCoalInt> negation reaches the value set in the <Cryptographic Interrupt Time Threshold> Register.
13:12	Reserved	RW0C 0x0	Reserved
11	AccAndTDMAInt_CM	RW0C 0x0	If the security accelerator is configured to accelerate continuous mode, this interrupt is set when processing of previous packet is completed (fetch first TDMA descriptor of next packet).
10	TDMAOwnIinterrupt	RW0C 0x0	Indicates an ownership error in the TDMA.
9	TDMACompIinterrupt	RW0C 0x0	Indicates completion of the TDMA operation.
8	Reserved	RW0C 0x0	Reserved
7	AccAndTDMAInt	RW0C 0x0	Acceleration and TDMA Interrupt This bit is set to 1 when the entire security accelerator process is completed, including both encryption/authentication process and its associated TDMA operation. When the TDMA is configured to copy the outcome of the security accelerator process back to the main memory (that is, when the <ActivateTDMA> field in the Security Accelerator Configuration Register, <AccAndTDMAInt> is set to 1) after the TDMA completes copying the data back to the main memory. When the TDMA is NOT configured to copy the outcome of the security accelerator process back to the DDR (that is, when the <ActivateTDMA> field in the Security Accelerator Configuration Register is set to 0), <AccAndTDMAInt> is set after the security accelerator completes the process and data is valid in the Internal SRAM. **NOTE:** If ZDMA is configured to continuous mode, AccAndTDMAInt is asserted only with the completion of the last packet in the chain. AccAndTDMAInt_CM interrupt is set with the completion of each packet.
6	Reserved	RSVD 0x0	Reserved
5	AccInt0	RW0C 0x0	This bit is the Security accelerator session 0 termination clear indication. The interrupt is set when the Security accelerator session 0 completes its operation.

Table 1327:Cryptographic Engine/Security Accelerator/TDMA Interrupt Cause Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x0009DE20, Tunit_port1: 0x0009FE20

Bit	Field	Type/InitVal	Description
4	ZInt4	RW0C 0x0	This bit is the encryption termination clear indication. The interrupt is set when the encryption engine finishes the calculation process.
3	ZInt3	RW0C 0x0	This bit is the AES decryption termination clear indication. The interrupt is set when the AES decryption engine finishes the calculation process.
2	ZInt2	RW0C 0x0	This bit is the AES encryption termination clear indication. The interrupt is set when the AES encryption engine finishes the calculation process.
1	ZInt1	RW0C 0x0	This bit is the DES encryption all termination clear indication. The interrupt is set when the encryption engine finishes the calculation process.
0	ZInt0	RW0C 0x0	This bit is the authentication termination clear indication. The interrupt is set when the authentication engine finishes the calculation process.

Table 1328:Cryptographic Engine/Security Accelerator/TDMA Interrupt Mask Register (p=0–1)

Offset: Tunit_port0: 0x0009DE24, Tunit_port1: 0x0009FE24

Offset Formula: 0x0009DE24+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	Mask bit for each cause bit Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register. 0 = Masked: Interrupt masked. 1 = Enabled: Interrupt enabled.

Table 1329:Cryptographic Interrupt Coalescing Threshold Register (p=0–1)

Offset: Tunit_port0: 0x0009DE30, Tunit_port1: 0x0009FE30

Offset Formula: 0x0009DE30+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved

Table 1329:Cryptographic Interrupt Coalescing Threshold Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x0009DE30, Tunit_port1: 0x0009FE30

Offset Formula: 0x0009DE30+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
7:0	EopPacketCoalTh	RW 0x0	Cryptographic Interrupt Coalescing Threshold This field provides a way to minimize the number of interrupts to off load the CPU. It defines the number of <AccAndTDMAInt_CM> indications before asserting the <EopCoalInt> bit in the Cryptographic interrupt Cause Register. Once the accumulated number of <AccAndTDMAInt_CM> indications reaches the EopPacketCoalTh value, a <EopCoalInt> interrupt is asserted. When <EopCoalInt> is negated or when the <Cryptographic Interrupt Coalescing Threshold> Register is written, the interrupts counter is cleared. 0x0 => Assertion of <AccAndTDMAInt_CM> causes an immediate assertion of <EopCoalInt>. 0x1-0xFF => <EopCoalInt> assertion is provided for every ‚ÄòEopPacketCoalTh’ times of <AccAndTDMAInt_CM> assertion.

Table 1330:Cryptographic Interrupt Time Threshold Register (p=0–1)

Offset: Tunit_port0: 0x0009DE34, Tunit_port1: 0x0009FE34

Offset Formula: 0x0009DE34+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:24	Reserved	RSVD 0x0	Reserved
23:0	EopTimeTh	RW 0x0	Cryptographic Interrupt Time Threshold This field provides a way to ensure maximum delay between <AccAndTDMAInt_CM> assertion and assertion bit <EopCoalInt> in Cryptographic Interrupt Cause Register (even if the number of <AccAndTDMAInt_CM> indications did not reach the <EopCoalPacketTh> value). When <EopCoalInt> is negated or when the <Cryptographic Interrupt Time Threshold> Register is written, the down counter is cleared. A new count is enabled in the assertion of the next <AccAndTDMAInt_CM> indication. When set to 0 = Assertion of <AccAndTDMAInt_CM> causes an immediate assertion of bit <EopCoalInt>. When set from 1-23 = Up to <n> internal clocks between assertion of <AccAndTDMAInt_CM> and assertion of <EopCoalInt>.

A.21.5 TDMA Address Decoding

Table 1331:Base Address 0 Register (p=0–1)

Offset: Tunit_port0: 0x00090A00, Tunit_port1: 0x00092A00

Offset Formula: 0x00090A00+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Window Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 1332:Window Control 0 Register (p=0–1)

Offset: Tunit_port0: 0x00090A04, Tunit_port1: 0x00092A04

Offset Formula: 0x00090A04+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size Specifies the size of the window in 64 KB granularity.
15:8	Attr	RW 0xE	Attribute See the Units IDs and Attributes table in Section Address Decoding.
7:4	TargetID	RW 0x0	Target Unit ID **NOTE:** Must be configured to the SDRAM Controller. See the Units IDs and Attributes table in Section Address Decoding.
3:1	Reserved	RSVD 0x0	Reserved
0	Enable	RW 0x1	Window Enable

Table 1333:Base Address 1 Register (p=0–1)

Offset: Tunit_port0: 0x00090A08, Tunit_port1: 0x00092A08

Offset Formula: 0x00090A08+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x1000	Window Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 1334:Window Control 1 Register (p=0–1)

Offset: Tunit_port0: 0x00090A0C, Tunit_port1: 0x00092A0C

Offset Formula: 0x00090A0C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0FFF	Window Size Specifies the size of the window in 64 KB granularity.
15:8	Attr	RW 0xD	Attribute See the Units IDs and Attributes table in Section Address Decoding.
7:4	TargetID	RW 0x0	Target Unit ID **NOTE:** Must be configured to the SDRAM Controller. See the Units IDs and Attributes table in Section Address Decoding.
3:1	Reserved	RSVD 0x0	Reserved
0	Enable	RW 0x1	Window Enable

Table 1335:Base Address 2 Register (p=0–1)

Offset: Tunit_port0: 0x00090A10, Tunit_port1: 0x00092A10

Offset Formula: 0x00090A10+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Window Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 1336:Window Control 2 Register (p=0–1)

Offset: Tunit_port0: 0x00090A14, Tunit_port1: 0x00092A14

Offset Formula: 0x00090A14+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size Specifies the size of the window in 64 KB granularity.
15:8	Attr	RW 0x0	Attribute See the Units IDs and Attributes table in Section Address Decoding.
7:4	TargetID	RW 0x0	Target Unit ID **NOTE:** Must be configured to the SDRAM Controller. See the Units IDs and Attributes table in Section Address Decoding.
3:1	Reserved	RSVD 0x0	Reserved

Table 1336:Window Control 2 Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x00090A14, Tunit_port1: 0x00092A14

Offset Formula: 0x00090A14+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
0	Enable	RW 0x0	Window Enable

Table 1337:Base Address 3 Register (p=0–1)

Offset: Tunit_port0: 0x00090A18, Tunit_port1: 0x00092A18

Offset Formula: 0x00090A18+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Window Base Address
15:0	Reserved	RSVD 0x0	Reserved

Table 1338:Window Control 3 Register (p=0–1)

Offset: Tunit_port0: 0x00090A1C, Tunit_port1: 0x00092A1C

Offset Formula: 0x00090A1C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size Specifies the size of the window in 64 KB granularity.
15:8	Attr	RW 0x0	Attribute See the Units IDs and Attributes table in Section Address Decoding.
7:4	TargetID	RW 0x0	Target Unit ID **NOTE:** Must be configured to the SDRAM Controller. See the Units IDs and Attributes table in Section Address Decoding.
3:1	Reserved	RSVD 0x0	Reserved
0	Enable	RW 0x0	Window Enable

A.21.6 TDMA Control

Table 1339:Control Register (p=0–1)

Offset: Tunit_port0: 0x00090840, Tunit_port1: 0x00092840

Offset Formula: 0x00090840+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:18	Reserved	RSVD 0x0	Reserved

Table 1339:Control Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x00090840, Tunit_port1: 0x00092840

Offset Formula: 0x00090840+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
17:16	NumOfOutstand	RW 0x0	Configure number of oustanding reads. Valid when <OustandingRdEn> set, otherwise, bits must be cleared. 3 = New Mode: Out of order - Maximum 4 outstanding read transactions
15	Reserved	RSVD 0x0	Reserved
14	TDMA Act	RO 0x0	TDMA's activation status. 0 = Idle: TDMA idle 1 = Active: TDMA active
13	FetchND	RW0C 0x0	Fetch Next Descriptor If set to 1, forces a fetch of the next descriptor. Cleared by hardware after the fetch is completed. FetchND is only relevant in chain mode. **NOTE:** This bit may be set by writing to this register only when TDMA is IDLE. Trying to set this bit is When <TDMA Act> bit is asserted or when <Own> bit is set to CPU is illegal. **NOTE:**
12	TDMAEn	RW 0x0	TDMA Enable 0 = Disabled 1 = Enabled
11	BS	RW 0x1	Byte swap on TDMA access to Mbus 0 = Byte swap 1 = No byte swap
10	Reserved	RSVD 0x0	Reserved
9	TdmaChainMode	RW 0x0	Chained Mode 0 = Chained mode 1 = Non-Chained mode
8:6	SrcBurstLimit	RW 0x3	Source Burst Limit in Each TDMA Access. The Source Burst Limit used if the TDMA Destination Address matches to the address space of the Security Accelerator Internal SRAM. 3 = 32 Bytes 4 = 128 Bytes
5	Reserved	RSVD 0x0	Reserved

Table 1339:Control Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x00090840, Tunit_port1: 0x00092840

Offset Formula: 0x00090840+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
4	OutstandingRdEn	RW 0x0	Outstanding Read Enable 0 = Disable: Outstanding read is disabled 1 = Enable: TDMA may assert outstanding reads. The field <NumOfOutstand> controls the number of outstanding reads.
3	Reserved	RSVD 0x0	Reserved
2:0	DstBurstLimit	RW 0x3	Destination Burst Limit in Each TDMA Access. The Destination Burst Limit used if the TDMA Source address matches to the address space of the Security Accelerator Internal SRAM. 3 = 32 Bytes 4 = 128 Bytes

A.21.7 TDMA Descriptor

Table 1340:TDMA Byte Count Register (p=0–1)

Offset: Tunit_port0: 0x00090800, Tunit_port1: 0x00092800

Offset Formula: 0x00090800+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31	Own	RO 0x1	Ownership Bit This bit indicates whether the descriptor is owned by the CPU (0) or the TDMA engine (1). This bit is relevant only when the descriptor is fetched. 0 = CPU: CPU owned 1 = TDMA: TDMA engine owned
30:16	Reserved	RSVD 0x0	Reserved
15:0	ByteCnt	RW 0x0	Number of bytes left for the TDMA to transfer. When <Own> bit cleared (owned by CPU) this field should be configured to zero.

Table 1341:TDMA Source Address Register (p=0–1)

Offset: Tunit_port0: 0x00090810, Tunit_port1: 0x00092810

Offset Formula: 0x00090810+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	SrcAdd	RW 0x0	Bits [31:0] of the TDMA source address

Table 1342:TDMA Destination Address Register (p=0–1)

Offset: Tunit_port0: 0x00090820, Tunit_port1: 0x00092820

Offset Formula: 0x00090820+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DestAdd	RW 0x0	Bits [31:0] of the TDMA destination address

Table 1343:Next Descriptor Pointer Register (p=0–1)

Offset: Tunit_port0: 0x00090830, Tunit_port1: 0x00092830

Offset Formula: 0x00090830+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	NextDescPtr	RW 0x0	Bits [31:0] of the TDMA next descriptor address The address must be 32-byte aligned (bits [3:0] must be 0x0)

Table 1344:Current Descriptor Pointer Register (p=0–1)

Offset: Tunit_port0: 0x00090870, Tunit_port1: 0x00092870

Offset Formula: 0x00090870+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	CDPTR0	RO 0x0	Bits [31:0] of the address from which the current descriptor was fetched

A.21.8 DES Engine

Table 1345:DES Initial Value Low Register (p=0–1)

Offset: Tunit_port0: 0x0009DD40, Tunit_port1: 0x0009FD40

Offset Formula: 0x0009DD40+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESIVLo	RW 0x0	Contains low bits of the Initial Value in CBC mode. (This register is ignored in ECB mode).

Table 1346:DES Initial Value High Register (p=0–1)

Offset: Tunit_port0: 0x0009DD44, Tunit_port1: 0x0009FD44

Offset Formula: 0x0009DD44+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESIVHi	RW 0x0	Contains high bits of the Initial Value in CBC mode. (This register is ignored in ECB mode.)

Table 1347:DES Key0 Low Register (p=0–1)

Offset: Tunit_port0: 0x0009DD48, Tunit_port1: 0x0009FD48

Offset Formula: 0x0009DD48+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESKey0Lo	RW 0x0	Contains the low bits of the DES key, or of the first key of the Triple DES keys.

Table 1348:DES Key0 High Register (p=0–1)

Offset: Tunit_port0: 0x0009DD4C, Tunit_port1: 0x0009FD4C

Offset Formula: 0x0009DD4C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESKey0Hi	RW 0x0	Contains the high bits of the DES key, or of the first key of the Triple DES keys.

Table 1349:DES Key1 Low Register (p=0–1)

Offset: Tunit_port0: 0x0009DD50, Tunit_port1: 0x0009FD50

Offset Formula: 0x0009DD50+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESKey1Lo	RW 0x0	Contains the low bits of the second key of the Triple DES keys. (This register is ignored in DES mode.)

Table 1350:DES Key1 High Register (p=0–1)

Offset: Tunit_port0: 0x0009DD54, Tunit_port1: 0x0009FD54

Offset Formula: 0x0009DD54+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESKey1Hi	RW 0x0	Contains the high bits of the second key of the Triple DES keys. (This register is ignored in DES mode.)

Table 1351:DES Command Register (p=0–1)

Offset: Tunit_port0: 0x0009DD58, Tunit_port1: 0x0009FD58

Offset Formula: 0x0009DD58+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31	Termination	RO 0x1	This bit is set by the engine to indicate completion of a DES calculation process. Any write operation to the encryption engine will clear this bit.
30	AllTermination	RO 0x1	This bit indicates to the host that the encryption calculation has been completed, the encryption parameters can be updated, and data can be written.

Table 1351:DES Command Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x0009DD58, Tunit_port1: 0x0009FD58

Offset Formula: 0x0009DD58+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
29	WriteAllow	RO 0x1	This bit indicates that the host can write data to the engine. Read operation from DES Data Out Low will clear this bit. 0 = NotAllowed: Write data to engine not allowed 1 = Allowed: Write data to engine allowed
28:9	Reserved	RSVD 0x0	Reserved
8	OutByteSwap	RSVD 0x0	This bit controls whether byte swap is activated for output. 0 = No byte swap 1 = Byte swap
7	Reserved	RSVD 0x0	Reserved
6	IVByteSwap	RW 0x0	This bit controls whether initial value byte swap is activated. 0 = No byte swap 1 = Byte swap
5	Reserved	RSVD 0x0	Reserved
4	DataByteSwap	RW 0x0	This bit controls whether data byte swap is activated upon input. 0 = No byte swap 1 = Byte swap
3	DESMode	RW 0x0	This bit controls the DEC encryption/decryption mode. 0 = ECB 1 = CBC
2	TripleDESMode	RW 0x0	This bit controls the Triple DES encryption/decryption mode. 0 = EEE 1 = EDE
1	Algorithm	RW 0x0	This bit controls whether the DES or Triple DES algorithm is used. 0 = DES 1 = 3DES: Triple DES
0	Direction	RW 0x0	This bit controls the direction of the operation: Encryption or Decryption. 0 = Encryption 1 = Decryption

Table 1352:DES Key2 Low Register (p=0–1)

Offset: Tunit_port0: 0x0009DD60, Tunit_port1: 0x0009FD60

Offset Formula: 0x0009DD60+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESKey2Lo	RW 0x0	Contains the low bits of the third key of the Triple DES keys. (This register is ignored in DES mode.)

Table 1353:DES Key2 High Register (p=0–1)

Offset: Tunit_port0: 0x0009DD64, Tunit_port1: 0x0009FD64

Offset Formula: 0x0009DD64+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DESKey2Hi	RW 0x0	Contains the high bits of the third key of the Triple DES keys. (This register is ignored in DES mode.)

Table 1354:DES Data Buffer Low Register (p=0–1)

Offset: Tunit_port0: 0x0009DD70, Tunit_port1: 0x0009FD70

Offset Formula: 0x0009DD70+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DataBufLo	WO 0x0	The host writes data blocks of low words to be encrypted/decrypted to this register.

Table 1355:DES Data Buffer High Register (p=0–1)

Offset: Tunit_port0: 0x0009DD74, Tunit_port1: 0x0009FD74

Offset Formula: 0x0009DD74+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DataBufHi	WO 0x0	The host writes data blocks of high words to be encrypted/decrypted to this register.

Table 1356:DES Data Out Low Register (p=0–1)

Offset: Tunit_port0: 0x0009DD78, Tunit_port1: 0x0009FD78

Offset Formula: 0x0009DD78+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DataOutLo	RO 0x0	When the DES (or the Triple DES) completes the calculation, this field will contain the low bits of the DES result.

Table 1357:DES Data Out High Register (p=0–1)

Offset: Tunit_port0: 0x0009DD7C, Tunit_port1: 0x0009FD7C

Offset Formula: 0x0009DD7C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DataOutHi	RO 0x0	When the DES (or the Triple DES) completes the calculation, this field will contain the high bits of the DES result.

A.21.9 Security Accelerator

Table 1358:Security Accelerator Command Register (p=0–1)

Offset: Tunit_port0: 0x0009DE00, Tunit_port1: 0x0009FE00

Offset Formula: 0x0009DE00+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x0	Reserved
2	DsSecurityAccl	SC 0x0	Disable accelerator This bit is self negated. When this bit is set to 1, the accelerator aborts the current command, and then clears bits AccInt0, AccInt1. **NOTE:** ARZ: Auto-Reset to Zero after the AccInt0, AccInt1bits are cleared. **NOTE:**
1	Reserved	RSVD 0x0	Reserved.
0	EnSecurityAccl	SC 0x0	Setting this bit activates an accelerator session. After operation completion, this bit is cleared to zero by the hardware. Writing zero to this bit has no effect. 0 = Session is idle. 1 = Session is set to active. 0 = Idle: Session 0 is idle. 1 = Active: Session 0 is set to active.

Table 1359:Security Accelerator Descriptor Pointer Register (p=0–1)

Offset: Tunit_port0: 0x0009DE04, Tunit_port1: 0x0009FE04

Offset Formula: 0x0009DE04+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:0	SecurityAcclDescPtr0	RW 0x0	Security accelerator descriptor pointer for session 0 (DWORD aligned) Bits [0], [1], [2], [13], [14] and [15] are reserved, and are assumed to be read as 0 regardless of programming.

Table 1360:Security Accelerator Configuration Register (p=0–1)

Offset: Tunit_port0: 0x0009DE08, Tunit_port1: 0x0009FE08

Offset Formula: 0x0009DE08+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:16	Reserved	RSVD 0x0	Reserved
15:14	Reserved	RSVD 0x0	Reserved
13	EngineParallelDis	RW 0x0	Control whether the Security Accelerator will do processing of Authentication and Encryption parallely or serially. 0 = Parallel Enable: Parallel process 1 = Parallel Disable: Serial process
12	Reserved	RSVD 0x0	Reserved
11	MultiPacketChainMode	RW 0x0	Security Acceleration Multi-Packet Chain Mode **NOTE:** The Multi-Packet Chain mode can only be enabled when using Security Accelerator Enhanced Mode (<WaitForTDMA> and <ActivateTDMA> bits are set). 0 = Single packet: Security accelerator halts after processing a single packet. 1 = Multi-Packet: Security accelerator continues to process packets until the TDMA reaches a NULL pointer (no more packets to process).
10	Reserved	RSVD 0x0	Reserved
9	ActivateTDMA	RW 0x0	Activation for TDMA When set to 1, the Security accelerator completes the current process.
8	Reserved	RSVD 0x0	Reserved
7	WaitForTDMA	RW 0x0	Wait for TDMA When set to 1, the security accelerator is activated only a when a TDMA ownership error occurs. This bit also configures the TDMA to start working when the Security accelerator is enabled.
6:2	Reserved	RSVD 0x0	Reserved
1:0	StopOnDecodeDigestErr	RW 0x1	Controls whether the engine stops or skip operations when there is a digest error in decode. 0 = Continue: Do not stop encryption operation on digest decode error. 1 = Skip: Skip encryption operation on digest decode error of current packet. 3 = Stop: Stop on digest decode error. Note: In multi packet chain mode TDMA will stop after it copies data back to main memory.

Table 1361:Security Accelerator Status Register (p=0–1)

Offset: Tunit_port0: 0x0009DE0C, Tunit_port1: 0x0009FE0C

Offset Formula: 0x0009DE0C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:12	AcclState	RO 0x0	Internal State of the accelerator
11:10	Reserved	RSVD 0x0	Reserved
9	Reserved	RO 0x0	Reserved
8	DecodeDigestErr	RO 0x0	Signals a decode digest error during the session. This bit is cleared when the session is activated.
7:2	Reserved	RSVD 0x0	Reserved
1	Reserved	RO 0x0	Reserved
0	AccActive	RO 0x0	Session State This bit equals <AccInt>. 0 = Idle: Session 0 is idle 1 = Active: Session 0 is active

A.21.10 SHA-1, SHA-2, and MD5 Interface

Table 1362:SHA-1/SHA-2/MD5 Initial Value/Digest A Register (p=0–1)

Offset: Tunit_port0: 0x0009DD00, Tunit_port1: 0x0009FD00

Offset Formula: 0x0009DD00+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	IVDigA	RW 0x67452301	IV A contains the first word of the Initial Value, and Digest A contains the first word of the digest. When the SHA-1/MD5 algorithm is enabled, the initial value is: 0x67452301. When the SHA-2 algorithm is enabled, the initial value is: 0x6a09e667.

Table 1363:SHA-1/SHA-2/MD5 Initial Value/Digest B Register (p=0–1)

Offset: Tunit_port0: 0x0009DD04, Tunit_port1: 0x0009FD04

Offset Formula: 0x0009DD04+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	IVDigB	RW 0xEFCDAB89	IV B contains the second word of the Initial Value, and Digest B contains the second word of the digest. When the SHA-1/MD5 algorithm is enabled, the initial value is: 0xefcdab89. When the SHA-2 algorithm is enabled, the initial value is: 0xbb67ae85.

Table 1364:SHA-1/SHA-2/MD5 Initial Value/Digest C Register (p=0–1)

Offset: Tunit_port0: 0x0009DD08, Tunit_port1: 0x0009FD08

Offset Formula: 0x0009DD08+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	IVDigC	RW 0x98BADCFE	IV C contains the third word of the Initial Value, and Digest C contains the third word of the digest. When the SHA-1/MD5 algorithm is enabled, the initial value is: 0x98badcfe. When the SHA-2 algorithm is enabled, the initial value is: 0x3c6ef372.

Table 1365:SHA-1/SHA-2/MD5 Initial Value/Digest D Register (p=0–1)

Offset: Tunit_port0: 0x0009DD0C, Tunit_port1: 0x0009FD0C

Offset Formula: 0x0009DD0C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	IVDigD	RW 0x10325476	IV D contains the fourth word of the Initial Value, and Digest D contains the fourth word of the digest. When the SHA-1/MD5 algorithm is enabled, the initial value is: 0x10325476. When the SHA-2 algorithm is enabled, the initial value is: 0xa54ff53a.

Table 1366:SHA-1/SHA-2 Initial Value/Digest E Register (p=0–1)

Offset: Tunit_port0: 0x0009DD10, Tunit_port1: 0x0009FD10

Offset Formula: 0x0009DD10+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	IVDigE	RW 0xC3D2E1F0	IV E contains the fifth word of the Initial Value, and Digest E contains the fifth word of the digest. **NOTE:** This register is only used in SHA-1 and SHA-2, since SHA mode requires a 5-word (SHA-1) or 8-word (SHA-2) initial value to produce the 5-word or 8-word SHA signature. When the SHA-1 algorithm is enabled, the initial value is: 0xc3d2e1f0. When the SHA-2 algorithm is enabled, the initial value is: 0xa54ff53a.

Table 1367:SHA-1/SHA-2/MD5 Authentication Command Register (p=0–1)

Offset: Tunit_port0: 0x0009DD18, Tunit_port1: 0x0009FD18

Offset Formula: 0x0009DD18+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31	Termination	RO 0x1	This bit is set by the engine to indicate completion of a hash calculation process. Any write to the Authentication engine will clear this bit.
30:7	Reserved	RSVD 0x0	Reserved
6	SHAMode	RW 0x0	This bit controls the mode of operation: SHA-2 or SHA-1. These are two different algorithms for calculating the authentication signature. They are described in the references. The SHA-1 calculation takes 85 clock cycles; the SHA-2 takes 65 clock cycles. SHA-1 mode results in a 5-word signature (and a 5-word initial value is required); SHA-2 results in an 8-word signature (and an 8-word initial value is required). These algorithms differ in their complexity and security levels, and it is left to the user to choose the algorithm. When MD5 is used, this bit must be configured to SHA1_Enable. 0 = SHA1 Enable: SHA-1 Mode 1 = SHA2 Enable: SHA-256 Mode
5	Reserved	RSVD 0x0	Reserved
4	IVByteSwap	RW 0x0	This bit controls whether initial value byte swap is activated. This is the same as the data swap, but only for initial values written to the IV/Digest registers. 0 = No byte swap 1 = Byte swap
3	Reserved	RSVD 0x0	Reserved
2	DataByteSwap	RW 0x0	This bit controls whether data-byte swap is activated. Packet data written to the engine can be used as is, or swapped by the engine before processing. The main purpose of this field is for processing different notations of packet data--data may be annotated as Big Endian or Little Endian. 0 = No byte swap: Data to engine W0...W15 -- 0x01234567. 1 = Byte swap: Data to engine W0...W15 -- 0x67452301.

Table 1367:SHA-1/SHA-2/MD5 Authentication Command Register (p=0–1) (Continued)

Offset: Tunit_port0: 0x0009DD18, Tunit_port1: 0x0009FD18

Offset Formula: 0x0009DD18+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
1	Mode	RW 0x0	This bit controls whether the initial value is used or the operation continues from the last value. Both SHA and MD5 algorithms do a computational process on chunks of 512 bits where the last 64 bits in the last chunk are reserved for packet size. When a packet length is less then 448 bits, the host must add one bit of 1 to the end of the packet and pad it to 448-bit size with zeros. Then, the host adds a double word (64 bits) that contains the length. Then, the chunk is ready for processing by the engine. Packets may be of arbitrary length (up to 2^64 bits). They are broken into 512-bit chunks. The last chunk of the packet is padded to 448 bits as described above, and 64 bits representing packet length are added to make a 512-bit block. Prior to writing the first chunk of a packet, the host must select the Initial mode (0) in the command register. After the first chunk is processed, all the proceeding chunks of the packet must be processed using Continue mode (1). In Initial mode, the engine starts processing the data block, using the initial values of the algorithm. In Continue mode, the results of the previous calculation are used. The user may wish to share the engine for multiple packet signature calculations. This is done by calculating a chunk or chunks of a specific packet, reading the intermediate digest, and saving the digest in a memory. Then, it is possible to start to process another packet. To continue processing the first packet, the host must write the intermediate digest that was saved in the memory, to the initial values registers and continue packet processing in Continue mode. 0 = Initial value: Use initial value 1 = Last value: Continue from the last value
0	Algorithm	RW 0x0	This bit controls the mode of operation: SHA-1 or MD5. These are two different algorithms for calculating the authentication signature. They are described in the references. The SHA-1 calculation takes 85 clock cycles; the MD5 takes 65 clock cycles. SHA-1 mode results in a 5-word signature (and a 5-word initial value is required); MD5 results in a 4-word signature (and a 4-word initial value is required). The MD5 is byte swapped compared to the SHA-1. These algorithms differ in their complexity and security levels, and it is left to the user to choose the algorithm. When MD5 is used, the SHAMode bit must be configured to SHA1_ Enable. 0 = MD5 1 = SHA1

Table 1368:SHA-1/SHA-2/MD5 Bit Count Low Register (p=0–1)

Offset: Tunit_port0: 0x0009DD20, Tunit_port1: 0x0009FD20

Offset Formula: 0x0009DD20+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	BitCntLo	WO 0x0	Fourteenth word of data in the array This register is accessed only when automatic padding is needed.

Table 1369:SHA-1/SHA-2/MD5 Bit Count High Register (p=0–1)

Offset: Tunit_port0: 0x0009DD24, Tunit_port1: 0x0009FD24

Offset Formula: 0x0009DD24+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	BitCntHi	WO 0x0	Fifteenth word of data in the array This register is accessed only when automatic padding is needed.

Table 1370:SHA-2 Initial Value/Digest F Register (p=0–1)

Offset: Tunit_port0: 0x0009DD28, Tunit_port1: 0x0009FD28

Offset Formula: 0x0009DD28+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
IV E contains the fifth word of the Initial Value, and Digest E contains the fifth word of the digest. **NOTE:** This register is only used in SHA-1, since SHA mode requires a 5-word initial value to produce the 5-word SHA signature.			
31:0	IVDigF	RW 0x9b05688c	IV F contains the sixth word of the Initial Value, and Digest F contains the sixth word of the digest. **NOTE:** This register is only used in SHA-2, since SHA-2 mode requires an 8-word initial value to produce the 8-word SHA-2 signature. **NOTE:**

Table 1371:SHA-2 Initial Value/Digest G Register (p=0–1)

Offset: Tunit_port0: 0x0009DD2C, Tunit_port1: 0x0009FD2C

Offset Formula: 0x0009DD2C+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	IVDigG	RW 0x1f83d9ab	IV G contains the seventh word of the Initial Value, and Digest G contains the seventh word of the digest. **NOTE:** This register is only used in SHA-2, since SHA-2 mode requires an 8-word initial value to produce the 8-word SHA-2 signature. **NOTE:**

Table 1372:SHA-2 Initial Value/Digest H Register (p=0–1)

Offset: Tunit_port0: 0x0009DD30, Tunit_port1: 0x0009FD30

Offset Formula: 0x0009DD30+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	IVDigH	RW 0x5be0cd19	IV H contains the eight word of the Initial Value, and Digest H contains the eight word of the digest. **NOTE:** This register is only used in SHA-2, since SHA-2 mode requires an 8-word initial value to produce the 8-word SHA-2 signature. **NOTE:**

Table 1373:SHA-1/SHA-2/MD5 Data In Register (p=0–1)

Offset: Tunit_port0: 0x0009DD38, Tunit_port1: 0x0009FD38

Offset Formula: 0x0009DD38+p*0x2000: where p (0-1) represents Tunit_port

Bit	Field	Type/InitVal	Description
31:0	DataIn	WO 0x0	Words of the 512-bit hash block should be written to this register. With each write, the data in this field is pushed into the authentication engine’s 16-word FIFO.

A.22 XOR Registers

The following table provides a summarized list of all registers that belong to the XOR, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1374:Summary Map Table for the XOR Registers

Register Name	Offset	Table and Page
XOR Engine Address Decoding Registers		
XOR Engine Window Control (XEnWCR)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060B40, unit0XOR1: 0x00060B44, unit1XOR0: 0x000F0B40, unit1XOR1: 0x000F0B44	Table 1375, p. 1473
XOR Engine Base Address (XEBARx)<n> Register (m=0–1, n=0–7)	unit0XEBAR0: 0x00060B50, unit0XEBAR1: 0x00060B54... unit1XEBAR7: 0x000F0B6C	Table 1376, p. 1474
XOR Engine Size Mask (XESMRx)<n> Register (m=0–1, n=0–7)	unit0XESMR0: 0x00060B70, unit0XESMR1: 0x00060B74... unit1XESMR7: 0x000F0B8C	Table 1377, p. 1474
XOR Engine High Address Remap (XEHARRx)<n> Register (m=0–1, n=0–3)	unit0XEHARR0: 0x00060B90, unit0XEHARR1: 0x00060B94, unit0XEHARR2: 0x00060B98, unit0XEHARR3: 0x00060B9C, unit1XEHARR0: 0x000F0B90, unit1XEHARR1: 0x000F0B94, unit1XEHARR2: 0x000F0B98, unit1XEHARR3: 0x000F0B9C	Table 1378, p. 1475
XOR Engine Address Override Control (XEAOCR)<n> Register (m=0–1, n=0–1)	unit0XEAOCR0: 0x00060BA0, unit0XEAOCR1: 0x00060BA4, unit1XEAOCR0: 0x000F0BA0, unit1XEAOCR1: 0x000F0BA4	Table 1379, p. 1475
XOR Engine Control Registers		
XOR Engine Channel Arbiter (XECHAR) Register (m=0–1)	unit0: 0x00060900, unit1: 0x000F0900	Table 1380, p. 1478
XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060910, unit0XOR1: 0x00060914, unit1XOR0: 0x000F0910, unit1XOR1: 0x000F0914	Table 1381, p. 1478
XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060920, unit0XOR1: 0x00060924, unit1XOR0: 0x000F0920, unit1XOR1: 0x000F0924	Table 1382, p. 1480
XOR Engine Descriptor Registers		
XOR Engine Next Descriptor Pointer (XEnNDPR)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060B00, unit0XOR1: 0x00060B04, unit1XOR0: 0x000F0B00, unit1XOR1: 0x000F0B04	Table 1383, p. 1481

Table 1374:Summary Map Table for the XOR Registers (Continued)

Register Name	Offset	Table and Page
XOR Engine Current Descriptor Pointer (XEnCDPR)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060B10, unit0XOR1: 0x00060B14, unit1XOR0: 0x000F0B10, unit1XOR1: 0x000F0B14	Table 1384, p. 1481
XOR Engine Byte Count (XEnBCR)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060B20, unit0XOR1: 0x00060B24, unit1XOR0: 0x000F0B20, unit1XOR1: 0x000F0B24	Table 1385, p. 1482
XOR Engine Memory Initialization		
XOR Engine Destination Pointer (XEnDPR0)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060BB0, unit0XOR1: 0x00060BB4, unit1XOR0: 0x000F0BB0, unit1XOR1: 0x000F0BB4	Table 1386, p. 1482
XOR Engine Block Size (XEnBSR)<n> Register (m=0–1, n=0–1)	unit0XOR0: 0x00060BC0, unit0XOR1: 0x00060BC4, unit1XOR0: 0x000F0BC0, unit1XOR1: 0x000F0BC4	Table 1387, p. 1482
XOR Engine Timer Mode Control (XETMCR) Register (m=0–1)	unit0: 0x00060BD0, unit1: 0x000F0BD0	Table 1388, p. 1483
XOR Engine Timer Mode Initial Value (XETMIVR) Register (m=0–1)	unit0: 0x00060BD4, unit1: 0x000F0BD4	Table 1389, p. 1484
XOR Engine Timer Mode Current Value (XETMCVR) Register (m=0–1)	unit0: 0x00060BD8, unit1: 0x000F0BD8	Table 1390, p. 1484
XOR Engine Initial Value Low (XEIVRL) Register (m=0–1)	unit0: 0x00060BE0, unit1: 0x000F0BE0	Table 1391, p. 1484
XOR Engine Initial Value High (XEIVRH) Register (m=0–1)	unit0: 0x00060BE4, unit1: 0x000F0BE4	Table 1392, p. 1485
XOR Engine Interrupt Registers		
XOR Engine Interrupt Cause (XEICR1) Register (m=0–1)	unit0: 0x00060930, unit1: 0x000F0930	Table 1393, p. 1485
XOR Engine Interrupt Mask (XEIMR) Register (m=0–1)	unit0: 0x00060940, unit1: 0x000F0940	Table 1394, p. 1487
XOR Engine Error Cause (XEECR) Register (m=0–1)	unit0: 0x00060950, unit1: 0x000F0950	Table 1395, p. 1488
XOR Engine Error Address (XEEAR) Register (m=0–1)	unit0: 0x00060960, unit1: 0x000F0960	Table 1396, p. 1488

A.22.1 XOR Engine Address Decoding Registers

Table 1375:XOR Engine Window Control (XEnWCR)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060B40, unit0XOR1: 0x00060B44, unit1XOR0: 0x000F0B40, unit1XOR1: 0x000F0B44

Offset Formula: 0x00060B40+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:30	Win7acc	RW 0x3	Window7 access control - see Window0 access control
29:28	Win6acc	RW 0x3	Window6 access control - see Window0 access control
27:26	Win5acc	RW 0x3	Window5 access control - see Window0 access control
25:24	Win4acc	RW 0x3	Window4 access control - see Window0 access control
23:22	Win3acc	RW 0x3	Window3 access control - see Window0 access control
21:20	Win2acc	RW 0x3	Window2 access control - see Window0 access control
19:18	Win1acc	RW 0x3	Window1 access control - see Window0 access control
17:16	Win0acc	RW 0x3	Window0 Access control In case of write protect violation (e.g. write data to a read only region), an interrupt is set, and the transaction is not driven to the target interface 0 = No access allowed 1 = Read Only 2 = Reserved 3 = Full access: Read or Write
15:8	Reserved	RO 0x0	Reserved
7	Win7en	RW 0x0	Window7 Enable - see Window0 Enable
6	Win6en	RW 0x0	Window6 Enable - see Window0 Enable
5	Win5en	RW 0x0	Window5 Enable - see Window0 Enable
4	Win4en	RW 0x0	Window4 Enable - see Window0 Enable
3	Win3en	RW 0x0	Window3 Enable - see Window0 Enable
2	Win2en	RW 0x0	Window2 Enable - see Window0 Enable

Table 1375:XOR Engine Window Control (XEnWCR)<n> Register (m=0–1, n=0–1) (Continued)

Offset: unit0XOR0: 0x00060B40, unit0XOR1: 0x00060B44, unit1XOR0: 0x000F0B40, unit1XOR1: 0x000F0B44

Offset Formula: 0x00060B40+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
1	Win1en	RW 0x0	Window1 Enable - see Window0 Enable
0	Win0en	RW 0x0	Window0 Enable 0 = Disable 1 = Enable

Table 1376:XOR Engine Base Address (XEBARx)<n> Register (m=0–1, n=0–7)

Offset: unit0XEBAR0: 0x00060B50, unit0XEBAR1: 0x00060B54...unit1XEBAR7: 0x000F0B6C

Offset Formula: 0x00060B50+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-7) represents XEBAR

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Used with the corresponding XOR Engine Size Mask register to set the address window size and location within the range of 4 GB space. The base address granularity is 64KB
15:8	Attr	RW 0x0	Specifies target specific attributes depending on the target interface.
7:4	Reserved	RO 0x0	Reserved
3:0	Target	RW 0x0	Specifies the target interface associated with this window: See Address Decoding chapter for full details

Table 1377:XOR Engine Size Mask (XESMRx)<n> Register (m=0–1, n=0–7)

Offset: unit0XESMR0: 0x00060B70, unit0XESMR1: 0x00060B74...unit1XESMR7: 0x000F0B8C

Offset Formula: 0x00060B70+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-7) represents XESMR

Bit	Field	Type/InitVal	Description
31:16	SizeMask	RW 0x0	Window Size Used with the corresponding XOR Engine Base Address register to set the address window size and location within the range of 4 GB space. Must be programed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window in 64 KB granularity (e.g. a value of 0x00ff specifies 256x64k = 16 MB).

Table 1377:XOR Engine Size Mask (XESMRx)<n> Register (m=0–1, n=0–7) (Continued)

Offset: unit0XESMR0: 0x00060B70, unit0XESMR1: 0x00060B74...unit1XESMR7: 0x000F0B8C

Offset Formula: 0x00060B70+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-7) represents XESMR

Bit	Field	Type/InitVal	Description
15:0	Reserved	RO 0x0	Reserved

Table 1378:XOR Engine High Address Remap (XEHARRx)<n> Register (m=0–1, n=0–3)

Offset: unit0XEHARR0: 0x00060B90, unit0XEHARR1: 0x00060B94, unit0XEHARR2: 0x00060B98, unit0XEHARR3: 0x00060B9C, unit1XEHARR0: 0x000F0B90, unit1XEHARR1: 0x000F0B94, unit1XEHARR2: 0x000F0B98, unit1XEHARR3: 0x000F0B9C

Offset Formula: 0x00060B90+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-3) represents XEHARR

Bit	Field	Type/InitVal	Description
NOTE: High Address Remap Register #N corresponds to Base Address register #N, respectively.			
31:0	Remap	RW 0x0	Remap Address Specifies address bits[63:32] to be driven to the target interface for the corresponding window (window0 to winndow3). Only relevant for target interfaces that supports more than 4 GB address space. When using target interface that do not support more than 4 GB address space, this register must be cleared. Note: For window4 to window7, 0x0 will be generated to address bits [63:32]

Table 1379:XOR Engine Address Override Control (XEAOCR)<n> Register (m=0–1, n=0–1)

Offset: unit0XEAOCR0: 0x00060BA0, unit0XEAOCR1: 0x00060BA4, unit1XEAOCR0: 0x000F0BA0, unit1XEAOCR1: 0x000F0BA4

Offset Formula: 0x00060BA0+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XEAOCR

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved

Table 1379:XOR Engine Address Override Control (XEAOCR)<n> Register (m=0–1, n=0–1) (Continued)

Offset: unit0XEAOCR0: 0x00060BA0, unit0XEAOCR1: 0x00060BA4, unit1XEAOCR0: 0x000F0BA0, unit1XEAOCR1: 0x000F0BA4

Offset Formula: 0x00060BA0+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1)

Bit	Field	Type/InitVal	Description
29:28	NDAOvrPtr	RW 0x0	Override Next Descriptor Address Pointer Specifies the register from which the override parameters will be taken. **NOTE:** Valid only if NDAOvrEn is set. 0 = XEHARR0: Target and attributes are taken from XEBAR0. Address[63:32] taken from XEHARR0. 1 = XEHARR1: Target and attributes are taken from XEBAR1. Address[63:32] taken from XEHARR1. 2 = XEHARR2: Target and attributes are taken from XEBAR2. Address[63:32] taken from XEHARR2. 3 = XEHARR3: Target and attributes are taken from XEBAR3. Address[63:32] taken from XEHARR3.
27	NDAOvrEn	RW 0x0	Override Next Descriptor Address Control 0 = No Override 1 = Override enable
26:25	DAOvrPtr	RW 0x0	Override Destination Address Pointer Specifies the register from which the override parameters will be taken. **NOTE:** Valid only if DAOvrEn is set. 0 = XEHARR0: Target and attributes are taken from XEBAR0. Address[63:32] taken from XEHARR0. 1 = XEHARR1: Target and attributes are taken from XEBAR1. Address[63:32] taken from XEHARR1. 2 = XEHARR2: Target and attributes are taken from XEBAR2. Address[63:32] taken from XEHARR2. 3 = XEHARR3: Target and attributes are taken from XEBAR3. Address[63:32] taken from XEHARR3.
24	DAOvrEn	RW 0x0	Override Destination Address Control 0 = No override 1 = Override enable
23:22	SA7OvrPtr	RW 0x0	Override Source Address #7 Pointer **NOTE:** Valid only if SA7OvrEn is set. **NOTE:**
21	SA7OvrEn	RW 0x0	Override Source Address #7 Control
20:19	SA6OvrPtr	RW 0x0	Override Source Address #6 Pointer **NOTE:** Valid only if SA6OvrEn is set. **NOTE:**
18	SA6OvrEn	RW 0x0	Override Source Address #6 Control

Table 1379:XOR Engine Address Override Control (XEAOCR)<n> Register (m=0–1, n=0–1) (Continued)

Offset: unit0XEAOCR0: 0x00060BA0, unit0XEAOCR1: 0x00060BA4, unit1XEAOCR0: 0x000F0BA0, unit1XEAOCR1: 0x000F0BA4

Offset Formula: 0x00060BA0+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1)

Bit	Field	Type/InitVal	Description
17:16	SA5OvrPtr	RW 0x0	Override Source Address #5 Pointer **NOTE:** Valid only if SA5OvrEn is set. **NOTE:**
15	SA5OvrEn	RW 0x0	Override Source Address #5 Control
14:13	SA4OvrPtr	RW 0x0	Override Source Address #4 Pointer **NOTE:** Valid only if SA4OvrEn is set. **NOTE:**
12	SA4OvrEn	RW 0x0	Override Source Address #4 Control
11:10	SA3OvrPtr	RW 0x0	Override Source Address #3 Pointer **NOTE:** Valid only if SA3OvrEn is set. **NOTE:**
9	SA3OvrEn	RW 0x0	Override Source Address #3 Control
8:7	SA2OvrPtr	RW 0x0	Override Source Address #2 Pointer **NOTE:** Valid only if SA2OvrEn is set. **NOTE:**
6	SA2OvrEn	RW 0x0	Override Source Address #2 Control
5:4	SA1OvrPtr	RW 0x0	Override Source Address #1 Pointer **NOTE:** Valid only if SA1OvrEn is set. **NOTE:**
3	SA1OvrEn	RW 0x0	Override Source Address #1 Control
2:1	SA0OvrPtr	RW 0x0	Override Source Address #0 Pointer Specifies the register from which the override parameters will be taken. **NOTE:** Valid only if SA0OvrEn is set. 0 = XEHARR0: Target and attributes are taken from XEBAR0. Address[63:32] taken from XEHARR0. 1 = XEHARR1: Target and attributes are taken from XEBAR1. Address[63:32] taken from XEHARR1. 2 = XEHARR2: Target and attributes are taken from XEBAR2. Address[63:32] taken from XEHARR2. 3 = XEHARR3: Target and attributes are taken from XEBAR3. Address[63:32] taken from XEHARR3.
0	SA0OvrEn	RW 0x0	Override Source Address #0 Control

A.22.2 XOR Engine Control Registers

Table 1380:XOR Engine Channel Arbiter (XECHAR) Register (m=0–1)

Offset: unit0: 0x00060900, unit1: 0x000F0900

Offset Formula: 0x00060900+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved
7	Slice7	RW 0x1	Slice #7 of the channel pizza arbiter
6	Slice6	RW 0x0	Slice #6 of the channel pizza arbiter
5	Slice5	RW 0x1	Slice #5 of the channel pizza arbiter
4	Slice4	RW 0x0	Slice #4 of the channel pizza arbiter
3	Slice3	RW 0x1	Slice #3 of the channel pizza arbiter
2	Slice2	RW 0x0	Slice #2 of the channel pizza arbiter
1	Slice1	RW 0x1	Slice #1 of the channel pizza arbiter
0	Slice0	RW 0x0	Slice #0 of the channel pizza arbiter. 0 = Channel0: Slice is owned by channel 0 1 = Channel1: Slice is owned by channel 1

Table 1381:XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060910, unit0XOR1: 0x00060914, unit1XOR0: 0x000F0910, unit1XOR1: 0x000F0914

Offset Formula: 0x00060910+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Reserved
15	RegAccProtect	RW 0x1	Internal Register Access protection Enable 0 = Disable 1 = Enable

Table 1381:XOR Engine Configuration (XEnCR)<n> Register (m=0–1, n=0–1) (Continued)

Offset: unit0XOR0: 0x00060910, unit0XOR1: 0x00060914, unit1XOR0: 0x000F0910, unit1XOR1: 0x000F0914

Offset Formula: 0x00060910+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
14	DesSwp	RW 0x0	Descriptor read/write Endianess Swap control If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc. 0 = No swap 1 = Swap
13	DwrReqSwp	RW 0x0	Data Write request Endianess Swap control If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc. 0 = No swap 1 = Swap
12	DrdResSwp	RW 0x0	Data Read Response Endianess Swap control If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc. 0 = No Swap 1 = Swap
11	Reserved	RW 0x0	Reserved
10:8	DstBurstLimit	RW 0x4	Burst Limit in each destination write request access over the Internal Mbus 2 = 32 Bytes 3 = 64 Bytes 4 = 128 Bytes
7	Reserved	RW 0x0	Reserved
6:4	SrcBurstLimit	RW 0x4	Burst Limit in each source read request access over the Internal Mbus 2 = 32 Bytes 3 = 64 Bytes 4 = 128 Bytes
3	Reserved	RW 0x0	Reserved
2:0	OperationMode	RW 0x0	Specifies the type of operation to be carried out by XOR Engine. 0 = XOR calculate operation 1 = CRC-32 calculate operation 2 = DMA operation 3 = ECC_cleanup_operation 4 = Memory Initialization operation

Table 1382:XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060920, unit0XOR1: 0x00060924, unit1XOR0: 0x000F0920, unit1XOR1: 0x000F0924

Offset Formula: 0x00060920+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:6	Reserved	RO 0x0	Reserved
5:4	XEstatus	RO 0x0	XOR Engine Status indication 0 = Channel not active 1 = Channel active 2 = Channel paused 3 = Reserved
3	XErestart	WO 0x0	XOR Engine Restart after Pause Control 0 = No meaning: Clearing this bit has no meaning and will be disregarded by the XOR Engine. 1 = Pause state: The XOR Engine restart after pause. Setting this bit after the XOR Engine channel enters the pause state re-activates the channel and resumes the suspended operation execution.
2	XEpause	WO 0x0	XOR Engine Pause Control 0 = No meaning: Clearing this bit has no meaning and will be disregarded by XOR Engine. 1 = Pause state: When the software sets this bit, it pauses the relevant XOR Engine channel. XOR Engine will suspend at the earliest opportunity (refer to Pause Operation section of the Functional Specification). After entering paused state, the XOR Engine will signal the software by setting the <XEstatus> field to Channel paused (0x2) and asserting the paused interrupt.
1	XEstop	WO 0x0	**NOTE:** Setting < XEstop> when XOR Engine is inactive or paused will be disregarded. 0 = No meaning: Clearing this bit has no meaning and will be disregarded by XOR Engine. 1 = Stop: When the software sets this bit, it de-activates the relevant XOR Engine channel. XOR Engine will stop the current operation at the earliest opportunity (refer to Stop Operation section of the Functional Specification). After entering de-active state XOR Engine will signal the software by setting the <XEstatus> field to Channel not active (0x0) and asserting the stopped interrupt.

Copyright © 2014 Marvell

Table 1382:XOR Engine Activation (XEnACTR)<n> Register (m=0–1, n=0–1) (Continued)

Offset: unit0XOR0: 0x00060920, unit0XOR1: 0x00060924, unit1XOR0: 0x000F0920, unit1XOR1: 0x000F0924

Offset Formula: 0x00060920+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
0	XEStart	WO 0x0	**NOTE:** Software must confirm that the <XEstatus> field is in the Channel not active (0x0) state. Setting <XEstart> when XOR Engine is active will be disregarded. 0 = No meaning: Clearing this bit has no meaning and will be disregarded by XOR Engine. 1 = Start: When the software sets this bit, it activates the relevant XOR Engine channel. After entering active state, XOR Engine will signal the software by setting the <XEstatus> field to Channel active (0x1).

A.22.3 XOR Engine Descriptor Registers

Table 1383:XOR Engine Next Descriptor Pointer (XEnNDPR)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060B00, unit0XOR1: 0x00060B04, unit1XOR0: 0x000F0B00, unit1XOR1: 0x000F0B04

Offset Formula: 0x00060B00+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:0	NextDescPtr	RW 0x0	XOR Engine’s next descriptor address pointer In XOR mode, bits[5:0] must be zero In CRC/DMA mode, bits[4:0] must be zero **NOTE:** The value 0x0 is reserved for end of chain NULL indication. Descriptors must not be placed at address 0x0. The XOR Engine will ignore attempts to read a descriptor from that address. **NOTE:**

Table 1384:XOR Engine Current Descriptor Pointer (XEnCDPR)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060B10, unit0XOR1: 0x00060B14, unit1XOR0: 0x000F0B10, unit1XOR1: 0x000F0B14

Offset Formula: 0x00060B10+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:0	CurrentDescPtr	RO 0x0	XOR Engine current descriptor address pointer. Points to the last descriptor that was fetched.

Table 1385:XOR Engine Byte Count (XEnBCR)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060B20, unit0XOR1: 0x00060B24, unit1XOR0: 0x000F0B20, unit1XOR1: 0x000F0B24

Offset Formula: 0x00060B20+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:0	ByteCnt	RO 0x0	Number of bytes left for the XOR Engine to execute the current descriptor operation. In XOR, DMA, and Memory Initialization modes: updated after every write action. In CRC mode: updated after every calculation operation.

A.22.4 XOR Engine Memory Initialization

Table 1386:XOR Engine Destination Pointer (XEnDPR0)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060BB0, unit0XOR1: 0x00060BB4, unit1XOR0: 0x000F0BB0, unit1XOR1: 0x000F0BB4

Offset Formula: 0x00060BB0+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:0	DstPtr	RW 0x0	Points to the target block of the Memory Initialization operations. **NOTE:** Valid only for Memory Initialization mode. **NOTE:**

Table 1387:XOR Engine Block Size (XEnBSR)<n> Register (m=0–1, n=0–1)

Offset: unit0XOR0: 0x00060BC0, unit0XOR1: 0x00060BC4, unit1XOR0: 0x000F0BC0, unit1XOR1: 0x000F0BC4

Offset Formula: 0x00060BC0+n*4 + 0x90000*m : where m (0-1) represents unit, where n (0-1) represents XOR

Bit	Field	Type/InitVal	Description
31:0	BlockSize	RW 0x0	Size of block in bytes for Memory Initialization operation. This field along with XE0DPR or XE1DPR (Destination pointer registers), defines the target block for these operations. Minimum value: 128B Maximum Value: 4GB The value 0x00000000 stands for 4-GB block size. The block must not cross 4-GB boundary. **NOTE:** Valid only for Memory Initialization mode. **NOTE:**

Table 1388:XOR Engine Timer Mode Control (XETMCR) Register (m=0–1)

Offset: unit0: 0x00060BD0, unit1: 0x000F0BD0

Offset Formula: 0x00060BD0+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:13	Reserved	RO 0x0	Reserved
12:8	SectionSizeCtrl	RW 0x0	Section Size Control Specifies the section size for ECC timer mode operation. Actual section size (in bytes) = 2^<SectionSizeCtrl> Minimum Value: 7 (section size of 128B). Maximum Value: 31 (section size of 2 GB). Must be less than Block Size (XEBSR) Reserved values: 0-6 **NOTE:** Valid only in ECC Timer mode.
7:1	Reserved	RO 0x0	Reserved
0	TimerEn	RW 0x0	Enable Timer Mode Enables triggering ECC operation with timer. If enabled the target block will be divided into sections according to the <SectionSizeCtrl> field value and the ECC timer will be enabled. Upon expiration of the timer, one section will be processed. When the timer expires the second time, the next section will be processed, and so on, until all the target blocks are processed. The XOR Engine will then again start cleaning the target block. To stop the operation, <XEstop> must be set. The ECC timer will be activated upon setting <XEstart> of a channel that is in ECC Timer mode. **NOTE:** Valid only in ECC mode. 0 = Disable 1 = Enable

Table 1389:XOR Engine Timer Mode Initial Value (XETMIVR) Register (m=0–1)

Offset: unit0: 0x00060BD4, unit1: 0x000F0BD4

Offset Formula: 0x00060BD4+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:0	TimerInitVal	RW 0x0	ECC Timer Mode Initial Value for Count-Down Controls the time period between executions of subsequent sections ECC cleanup. Specifies the number of TCLK cycles between subsequent sections cleanup. If timer expires before current section cleanup has ended, it will be disregarded. **NOTE:** Valid only if one of the XOR Engine channels is in ECC timer mode.

Table 1390:XOR Engine Timer Mode Current Value (XETMCVR) Register (m=0–1)

Offset: unit0: 0x00060BD8, unit1: 0x000F0BD8

Offset Formula: 0x00060BD8+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:0	TimerCrntVal	RO 0x0	ECC Timer Mode Current Value **NOTE:** Valid only if one of the XOR Engine channels is in ECC Timer mode.

Table 1391:XOR Engine Initial Value Low (XEIVRL) Register (m=0–1)

Offset: unit0: 0x00060BE0, unit1: 0x000F0BE0

Offset Formula: 0x00060BE0+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:0	InitValL	RW 0x0	LSB of Initial Value to be written cyclically to target block in MemInit mode. Mapped to bits[31:0] of initial value. This register is shared between the two XOR Engine channels. The XOR Engine will compose a 64 bit Initial Value out of InitValL and InitValH registers and write it cyclically to the target block. Target block can be of any alignment. **NOTE:** Valid only on MemInit modes. **NOTE:**

Table 1392:XOR Engine Initial Value High (XEIVRH) Register (m=0–1)

Offset: unit0: 0x00060BE4, unit1: 0x000F0BE4

Offset Formula: 0x00060BE4+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:0	InitValH	RW 0x0	MSB of Initial Value to be written cyclically to target block in MemInit mode. Mapped to bits[63:32] of initial value. This register is shared between the two XOR Engine channels. The XOR Engine will compose a 64 bit Initial Value out of InitValL and InitValH registers and write it cyclically to the target block. Target block can be of any alignment. **NOTE:** Valid only on MemInit modes. **NOTE:**

A.22.5 XOR Engine Interrupt Registers

Table 1393:XOR Engine Interrupt Cause (XEICR1) Register (m=0–1)

Offset: unit0: 0x00060930, unit1: 0x000F0930

Offset Formula: 0x00060930+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
NOTE: All cause bits are clear only. They are set to 1 upon an interrupt event and cleared when the software writes a value of 0. Writing 1 has no effect (don’t care). The XOR Engine disregards such write attempts.			
31:26	Reserved	RO 0x0	Reserved
25	XbarErr1	RW0C 0x0	Channel#1 Mbus Parity Error Caused by erroneous read response from the Mbus.
24	IntParityErr1	RW0C 0x0	Channel#1 Parity Error Caused by erroneous internal buffer read.
23	OwnErr1	RW0C 0x0	Channel#1 Descriptor Ownership Violation Attempt to access the descriptor owned by the CPU.
22	WrProt1	RW0C 0x0	Channel#1 Write Protect Violation Trying to write to a window that is write protected.
21	AccProt1	RW0C 0x0	Channel#1 Access Protect Violation Trying to access an address in a window in which access is not allowed.
20	AddrDecode1	RW0C 0x0	Channel#1 Failed address decoding Address is not in any window or matches more than one window.

Table 1393:XOR Engine Interrupt Cause (XEICR1) Register (m=0–1) (Continued)

Offset: unit0: 0x00060930, unit1: 0x000F0930

Offset Formula: 0x00060930+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
19	Paused1	RW0C 0x0	Channel#1 Paused XOR Engine completed Pausing routine, after receiving pause command (setting <XEpause>). It has entered Pause state.
18	Stopped1	RW0C 0x0	Channel#1 Stopped XOR Engine completed stopping routine, after receiving stop command (setting <XEstop>). It has entered Inactive state.
17	EOC1	RW0C 0x0	Channel#1 End of Chain Asserted when the XOR Engine finished transfer of current descriptor operation, and it is currently the last in the descriptor chain (byteCount==0) and (XENDP=NULL). Also asserted upon end of chain processing due to error condition.
16	EOD1	RW0C 0x0	Channel#1 End of Descriptor Asserted when the XOR Engine finished the transfer of the current descriptor operation (byteCount==0). Software can control EOD interrupt assertion per descriptor through <EODIntEn> bit in the Command field of the descriptor.
15:10	Reserved	RO 0x0	Reserved
9	XbarErr0	RW0C 0x0	Channel#0 Mbus Parity Error Caused by erroneous read response from the Mbus.
8	IntParityErr0	RW0C 0x0	Channel#0 Parity Error Caused by erroneous internal buffer read.
7	OwnErr0	RW0C 0x0	Channel#0 Descriptor Ownership Violation Attempt to access the descriptor owned by the CPU.
6	WrProt0	RW0C 0x0	Channel#0 Write Protect Violation Trying to write to a window that is write protected.
5	AccProt0	RW0C 0x0	Channel#0 Access Protect Violation Trying to access an address in a window in which access is not allowed.
4	AddrDecode0	RW0C 0x0	Channel#0 Failed address decoding Address is not in any window or matches more than one window.

Table 1393:XOR Engine Interrupt Cause (XEICR1) Register (m=0–1) (Continued)

Offset: unit0: 0x00060930, unit1: 0x000F0930

Offset Formula: 0x00060930+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
3	Paused0	RW0C 0x0	Channel#0 Paused XOR Engine completed Pausing routine, after receiving pause command (setting <XEpause>). It has entered Pause state.
2	Stopped0	RW0C 0x0	Channel#0 Stopped XOR Engine completed stopping routine, after receiving stop command (setting <XEstop>). It has entered Inactive state.
1	EOC0	RW0C 0x0	Channel#0 End of Chain Asserted when the XOR Engine finished transfer of current descriptor operation, and it is currently the last in the descriptor chain (byteCount==0) and (XENDP=NULL). Also asserted upon end of chain processing due to error condition.
0	EOD0	RW0C 0x0	Channel#0 End of Descriptor Asserted when the XOR Engine finished the transfer of the current descriptor operation (byteCount==0). Software can control EOD interrupt assertion per descriptor through <EODIntEn> bit in the Command field of the descriptor.

Table 1394:XOR Engine Interrupt Mask (XEIMR) Register (m=0–1)

Offset: unit0: 0x00060940, unit1: 0x000F0940

Offset Formula: 0x00060940+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:26	Reserved	RO 0x0	Reserved
25:16	Channel1IntrptMask	RW 0x0	Channel1 Interrupt Mask Each bit in this field mask the interrupt generation due to the corresponding bit in the XOR Engine Interrupt Cause Register. 0 = Disabled: Interrupt generation is disabled 1 = Enabled: Interrupt generation is enabled
15:10	Reserved	RW 0x0	Reserved
9:0	Channel0IntrptMask	RW 0x0	Channel0 Interrupt Mask Each bit in this field mask the interrupt generation due to the corresponding bit in the XOR Engine Interrupt Cause Register. 0 = Disabled: Interrupt generation is disabled 1 = Enabled: Interrupt generation is enabled

Table 1395:XOR Engine Error Cause (XEECR) Register (m=0–1)

Offset: unit0: 0x00060950, unit1: 0x000F0950

Offset Formula: 0x00060950+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:5	Reserved	RO 0x0	Reserved
4:0	ErrorType	ROC 0x0	Specifies the error event currently reported in the Error Address register: 0x0 = Null 0x1 - 0x3 = Reserved. 0x4 = AddrDecode0 0x5 = AccProt0 0x6 = WrProt0 0x7- 0x13 = Reserved. 0x14 = AddrDecode1 0x15 = AccProt1 0x16 = WrProt1 0x17-0x1F = Reserved. This field is self cleared by reading the Error Address Register (XEEAR). Once the error cause is latched, no new error cause or address is latched to XEECR or XEEAR, until SW reads XEEAR. The Software should read XEECR first, and than XEEAR.

Table 1396:XOR Engine Error Address (XEEAR) Register (m=0–1)

Offset: unit0: 0x00060960, unit1: 0x000F0960

Offset Formula: 0x00060960+0x90000*m: where m (0-1) represents unit

Bit	Field	Type/InitVal	Description
31:0	ErrAddr	RO 0x0	Bits[31:0] of Error Address Latched upon any of the address windows violation event (address miss, multiple hit, access protect, write protect). Once the address is latched, no new address is latched until SW reads it (Read access to XEEAR). SW should read XEECR first, and then XEEAR.

A.23 IDMA Registers

The following table provides a summarized list of all registers that belong to the IDMA, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1397:Summary Map Table for the IDMA Registers

Register Name	Offset	Table and Page
IDMA Address Decoding		
Basen Address Register (n=0–7)	BAR0: 0x00060A00, BAR1: 0x00060A08, BAR2: 0x00060A10, BAR3: 0x00060A18, BAR4: 0x00060A20, BAR5: 0x00060A28, BAR6: 0x00060A30, BAR7: 0x00060A38	Table 1398, p. 1490
Size Register_n (n=0–7)	SR0: 0x00060A04, SR1: 0x00060A0C, SR2: 0x00060A14, SR3: 0x00060A1C, SR4: 0x00060A24, SR5: 0x00060A2C, SR6: 0x00060A34, SR7: 0x00060A3C	Table 1399, p. 1491
High Address Remapn Register (n=0–3)	Register0: 0x00060A60, Register1: 0x00060A64, Register2: 0x00060A68, Register3: 0x00060A6C	Table 1400, p. 1491
Channeln Access Protect Register (n=0–3)	Channel0: 0x00060A70, Channel1: 0x00060A74, Channel2: 0x00060A78, Channel3: 0x00060A7C	Table 1401, p. 1491
Base Address Enable Register	0x00060A80	Table 1402, p. 1492
IDMA Channel Control		
Channeln Control (Low) Register (n=0–3)	Channel0: 0x00060840, Channel1: 0x00060844, Channel2: 0x00060848, Channel3: 0x0006084C	Table 1403, p. 1492
Channeln Control (High) Register (n=0–3)	Channel0: 0x00060880, Channel1: 0x00060884, Channel2: 0x00060888, Channel3: 0x0006088C	Table 1404, p. 1495
Mbus Timeout Register (Internal)	0x000608D0	Table 1405, p. 1496
IDMA Descriptor		
Channel IDMA Byte Count Register (n=0–3)	Channel0: 0x00060800, Channel1: 0x00060804, Channel2: 0x00060808, Channel3: 0x0006080C	Table 1406, p. 1496

Table 1397:Summary Map Table for the IDMA Registers (Continued)

Register Name	Offset	Table and Page
Channel IDMA Source Address Register (n=0–3)	Channel0: 0x00060810, Channel1: 0x00060814, Channel2: 0x00060818, Channel3: 0x0006081C	Table 1407, p. 1497
Channel IDMA Destination Address Register (n=0–3)	Channel0: 0x00060820, Channel1: 0x00060824, Channel2: 0x00060828, Channel3: 0x0006082C	Table 1408, p. 1497
Channel Next Descriptor Pointer Register (n=0–3)	Channel0: 0x00060830, Channel1: 0x00060834, Channel2: 0x00060838, Channel3: 0x0006083C	Table 1409, p. 1497
Channel Current Descriptor Pointer Register (n=0–3)	Channel0: 0x00060870, Channel1: 0x00060874, Channel2: 0x00060878, Channel3: 0x0006087C	Table 1410, p. 1498
IDMA Interrupt		
Interrupt Cause Register	0x000608C0	Table 1411, p. 1498
Interrupt Mask Register	0x000608C4	Table 1412, p. 1499
Error Address Register	0x000608C8	Table 1413, p. 1500
Error Select Register	0x000608CC	Table 1414, p. 1500

A.23.1 IDMA Address Decoding

Table 1398:Basen Address Register (n=0–7)

Offset: BAR0: 0x00060A00, BAR1: 0x00060A08, BAR2: 0x00060A10, BAR3: 0x00060A18, BAR4: 0x00060A20, BAR5: 0x00060A28, BAR6: 0x00060A30, BAR7: 0x00060A38

Offset Formula: 0x00060A00+n*8: where n (0-7) represents BAR

Bit	Field	Type/InitVal	Description
31:16	Base	RW 0x0	Base Address Used with the size register to set the address window size and location within the range of 4 GB space. An address driven by one of the IDMAs is considered as window hit if: (address | size) == (base | size).
15:8	Attr	RW 0x0	Specifies target unit specific attributes depending on the target interface. **NOTE:** See the "Address Map" section. **NOTE:**
7:4	Reserved	RW 0x0	Reserved
3:0	Target	RW 0x0	Specifies the target interface associated with this window. **NOTE:** See the "Address Map" section. **NOTE:**

Table 1399:Size Register_n (n=0–7)

Offset: SR0: 0x00060A04, SR1: 0x00060A0C, SR2: 0x00060A14, SR3: 0x00060A1C, SR4: 0x00060A24, SR5: 0x00060A2C, SR6: 0x00060A34, SR7: 0x00060A3C

Offset Formula: 0x00060A04+n*8: where n (0-7) represents SR

Bit	Field	Type/InitVal	Description
31:16	Size	RW 0x0	Window Size Used with the size register to set the address window size and location within the range of 4 GB space. Must be programed from LSB to MSB as sequence of 1s followed by sequence of 0s. The number of 1s specifies the size of the window in 64 KB granularity (e.g. a value of 0x00ff specifies 256x64k = 16 MB). An address driven by one of the IDMAs is considered as window hit if: (address | size) == (base | size).
15:0	Reserved	RO 0x0	Reserved

Table 1400:High Address Remapn Register (n=0–3)

Offset: Register0: 0x00060A60, Register1: 0x00060A64, Register2: 0x00060A68, Register3: 0x00060A6C

Offset Formula: 0x00060A60+n*4: where n (0-3) represents Register

Bit	Field	Type/InitVal	Description
Remap 0 corresponds to Base Address register 0. Remap 1 to Base Address register 1, Remap 2 to Base Address register 2 Remap 3 to Base Address register 3.			
31:0	Remap	RW 0x0	Remap Address Specifies address bits [63:32] to be driven to the target interface. Only relevant for target interfaces that supports more than 4 GB address space.

Table 1401:Channeln Access Protect Register (n=0–3)

Offset: Channel0: 0x00060A70, Channel1: 0x00060A74, Channel2: 0x00060A78, Channel3: 0x00060A7C

Offset Formula: 0x00060A70+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
31:16	Reserved	RO 0x0	Read only.
15:14	Win7	RW 0x3	Window7 access control
13:12	Win6	RW 0x0	Window6 access control
11:10	Win5	RW 0x0	Window5 access control

Table 1401:ChannelIn Access Protect Register (n=0–3) (Continued)

Offset: Channel0: 0x00060A70, Channel1: 0x00060A74, Channel2: 0x00060A78, Channel3: 0x00060A7C

Offset Formula: 0x00060A70+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
9:8	Win4	RW 0x3	Window4 access control
7:6	Win3	RW 0x3	Window3 access control
5:4	Win2	RW 0x3	Window2 access control
3:2	Win1	RW 0x0	Window1 access control
1:0	Win0	RW 0x0	Window0 Access control In case of access violation (e.g. write data to a read only region), an interrupt is set, and the transaction is not driven to the target interface. 0 = No access allowed 1 = Read Only 2 = Reserved 3 = Full access (read or write)

Table 1402:Base Address Enable Register

Offset: 0x00060A80

Bit	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Read only.
7:0	En	RW 0xFF	Address Window Enable Bit per window. If set to 0, the corresponding address window is enabled.

A.23.2 IDMA Channel Control

Table 1403:ChannelIn Control (Low) Register (n=0–3)

Offset: Channel0: 0x00060840, Channel1: 0x00060844, Channel2: 0x00060848, Channel3: 0x0006084C

Offset Formula: 0x00060840+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
31	DescMode	RW 0x0	Descriptor Mode 0 = 64K descriptor 1 = 16M descriptor
30:27	Reserved	RSVD 0x0	Reserved

Table 1403:Channeln Control (Low) Register (n=0–3) (Continued)

Offset: Channel0: 0x00060840, Channel1: 0x00060844, Channel2: 0x00060848, Channel3: 0x0006084C

Offset Formula: 0x00060840+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
26:25	NAddrOvr	RW 0x0	Override Next Descriptor Address 0 = No override 1 = BAR 1: Next descriptor interface and attributes are taken from BAR 1. 2 = BAR 2: Next descriptor interface and attributes are taken from BAR 2. 3 = BAR 3: Next descriptor interface and attributes are taken from BAR 3.
24:23	DAddrOvr	RW 0x0	Override Destination Address 0 = No override 1 = BAR 1: Destination interface and attributes are taken from BAR 1. 2 = BAR 2: Destination interface and attributes are taken from BAR 2. 3 = BAR 3: Destination interface and attributes are taken from BAR 3.
22:21	SAddrOvr	RW 0x0	Override Source Address 0 = No override 1 = BAR 1: Source interface and attributes are taken from BAR 1. 2 = BAR 2: Source interface and attributes are taken from BAR 2. 3 = BAR 3: Source interface and attributes are taken from BAR 3.
20	Abr	RW 0x0	Channel Abort When the software sets this bit to 1, the IDMA aborts in the middle. The bit is cleared by the IDMA hardware.
19:18	Reserved	RSVD 0x0	Reserved
17	CDEn	RW 0x0	Close Descriptor Enable If enabled, the IDMA writes the upper byte(s) of the byte count field back to memory. In 64K descriptor mode, it writes the remainder byte count into bits [31:16] of the byte count field. In n16M descriptor mode, it writes the ownership and status bits into bits [31:24] of byte count field. **NOTE:** Enable in chain mode only. Disable when a new chain is begun by directly programming the first descriptor of the chain into the channel registers instead of fetching the descriptor from memory using the <FetchND> bit [13]. 0 = Disable 1 = Enable
16	DMAReqMode	RW 0x0	DMAReqn Mode 0 = Level: DMAReqn is level input. 1 = Edge: DMAReqn is edge triggered input.

Table 1403:ChannelIn Control (Low) Register (n=0–3) (Continued)

Offset: Channel0: 0x00060840, Channel1: 0x00060844, Channel2: 0x00060848, Channel3: 0x0006084C

Offset Formula: 0x00060840+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
15	DMAReqDir	RW 0x0	DMAReq Direction Relevant for channel 0 only and for DMA -based UART only . See Section UART Interface in the device Functional Specifications. 0 = Source: DMAReqn generated by the source. 1 = Destination: DMAReqn generated by the destination.
14	ChanAct	RW 0x0	IDMA Channel Active Read only. 0 = Inactive 1 = Active
13	FetchND	RW 0x0	Fetch Next Descriptor If set to 1, forces a fetch of the next descriptor. Cleared after the fetch is completed. **NOTE:** FetchND is only relevant in chain mode. **NOTE:**
12	ChanEn	RW 0x0	Channel Enable Re-setting the bit to 1, allows the channel to continue the IDMA transfer. 0 = Suspended 1 = Activated
11	Reserved	RSVD 0x0	Reserved
10	IntMode	RW 0x0	Interrupt Mode **NOTE:** IntMode is only relevant in chain mode. 0 = 0 byte count: Interrupt asserted every time the IDMA byte count reaches 0. 1 = 0 byte count and Null: Interrupt asserted when the Next Descriptor pointer is NULL and the IDMA byte count reaches 0.
9	ChainMode	RW 0x0	Chained Mode 0 = Chained mode 1 = Non-Chained mode
8:6	SrcBurstLimit	RW 0x0	Burst Limit in Each IDMA Access 000 = 8 Bytes 001 = 16 Bytes 010 = Reserved 011 = 32 Bytes 100 = 128 Bytes 101 = Reserved 110 = Reserved 111 = 64 Bytes

Table 1403:ChannelIn Control (Low) Register (n=0–3) (Continued)

Offset: Channel0: 0x00060840, Channel1: 0x00060844, Channel2: 0x00060848, Channel3: 0x0006084C

Offset Formula: 0x00060840+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
5	DestHold	RW 0x0	Destination Hold 0 = Increment: Increment destination address. 1 = Hold: Hold in the same value.
4	Reserved	RSVD 0x0	Reserved
3	SrcHold	RW 0x0	Source Hold 0 = Increment: Increment source address. 1 = Hold: Hold in the same value.
2:0	DstBurstLimit	RW 0x0	000 = 8 Bytes 001 = 16 Bytes 010 = Reserved 011 = 32 Bytes 100 = 128 Bytes 101 = Reserved 110 = Reserved 111 = 64 Bytes

Table 1404:ChannelIn Control (High) Register (n=0–3)

Offset: Channel0: 0x00060880, Channel1: 0x00060884, Channel2: 0x00060888, Channel3: 0x0006088C

Offset Formula: 0x00060880+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
31:8	Reserved_31_8	RW 0x0	Read Only.
7	DPPar	RW 0x0	IDMA data path parity select: **NOTE:** Should be set to even parity for normal operation. Odd parity is for debug only. Only DPPar bit of Channel 0 Control register is used to select even/odd parity. DPPar bits of the other channels are non-applicable. 0 = Even parity 1 = Odd parity
6:2	Reserved	RSVD 0x0	Reserved
1	DescBS	RW 0x0	Descriptors Byte Swap If enabled, DMA swap the bytes of 64-bit dword during descriptor fetch and close. 0 = Enable 1 = Disable

Table 1404:ChannelIn Control (High) Register (n=0–3) (Continued)

Offset: Channel0: 0x00060880, Channel1: 0x00060884, Channel2: 0x00060888, Channel3: 0x0006088C

Offset Formula: 0x00060880+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
0	Reserved	RSVD 0x0	Reserved

Table 1405:Mbus Timeout Register (Internal)

Offset: 0x000608D0

Bit	Field	Type/InitVal	Description
31:17	Reserved	RO 0x0	Read only.
16	TimeoutEn	RW 0x1	Mbus Arbiter Timer Enable 0 = Enable 1 = Disable
15:8	Reserved	RO 0x0	Read only.
7:0	Timeout	RW 0xFF	Mbus Arbiter Timeout Preset Value **NOTE:** Must keep value of 0xFF as long as Timeout counter is not enabled **NOTE:**

A.23.3 IDMA Descriptor

Table 1406:Channel IDMA Byte Count Register (n=0–3)

Offset: Channel0: 0x00060800, Channel1: 0x00060804, Channel2: 0x00060808, Channel3: 0x0006080C

Offset Formula: 0x00060800+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
When running in 64K descriptor mode and when closing the descriptor, the DMA writes to bits31:16] the left byte count to be transferred.			
31	Own	RW 0x0	Ownership Bit When running in 16M descriptor mode, this bit indicates whether the descriptor is owned by the CPU (0) or the IDMA engine (1). 0 = CPU owned 1 = IDMA engine owned
30	BCLeft	RW 0x0	Left Byte Count When running in 16M descriptor mode and when closing a descriptor, indicates whether the whole byte count was completely transferred. 0 = Whole byte count: The whole byte count transferred. 1 = Terminated: Transfer terminated before the whole byte count was transferred.

Table 1406:Channel IDMA Byte Count Register (n=0–3) (Continued)

Offset: Channel0: 0x00060800, Channel1: 0x00060804, Channel2: 0x00060808, Channel3: 0x0006080C

Offset Formula: 0x00060800+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
29:24	Reserved	RSVD 0x0	Reserved
23:0	ByteCnt	RW 0x0	Number of bytes left for the IDMA to transfer When running in 64K descriptor mode, the byte count is 16-bit only (bits [15:0]).

Table 1407:Channel IDMA Source Address Register (n=0–3)

Offset: Channel0: 0x00060810, Channel1: 0x00060814, Channel2: 0x00060818, Channel3: 0x0006081C

Offset Formula: 0x00060810+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
31:0	SrcAdd	RW 0x0	Bits [31:0] of the IDMA source address.

Table 1408:Channel IDMA Destination Address Register (n=0–3)

Offset: Channel0: 0x00060820, Channel1: 0x00060824, Channel2: 0x00060828, Channel3: 0x0006082C

Offset Formula: 0x00060820+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
31:0	DestAdd	RW 0x0	Bits [31:0] of the IDMA destination address.

Table 1409:Channel Next Descriptor Pointer Register (n=0–3)

Offset: Channel0: 0x00060830, Channel1: 0x00060834, Channel2: 0x00060838, Channel3: 0x0006083C

Offset Formula: 0x00060830+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
31:0	NextDescPtr	RW 0x0	Bits [31:0] of the IDMA next descriptor address. The address must be 32-byte aligned (bits [3:0] must be 0x0).

Table 1410:Channel Current Descriptor Pointer Register (n=0–3)

Offset: Channel0: 0x00060870, Channel1: 0x00060874, Channel2: 0x00060878, Channel3: 0x0006087C

Offset Formula: 0x00060870+n*4: where n (0-3) represents Channel

Bit	Field	Type/InitVal	Description
31:0	CDPTR0/1/2/3	RW 0x0	Bits [31:0] of the address from which the current descriptor was fetched.

A.23.4 IDMA Interrupt

Table 1411:Interrupt Cause Register

Offset: 0x000608C0

Bit	Field	Type/InitVal	Description
All cause bits are clear only. They are set to 1 upon an interrupt event and cleared when the software writes a value of 0. Writing 1 has no affect.			
31:29	Reserved	RSVD 0x0	Reserved
28:24	Various	RW 0x0	Same as Channel3 cause bits
23:21	Reserved	RSVD 0x0	Reserved
20:16	Various	RW 0x0	Same as Channel2 cause bits
15:13	Reserved	RSVD 0x0	Reserved
12:8	Various	RW 0x0	Same as Channel1 cause bits
7	DPErr	RW 0x0	IDMA internal data path parity error detected.
6	Reserved	RSVD 0x0	Reserved
5	Reserved	RSVD 0x0	Reserved
4	Own	RW 0x0	Channel0 Descriptor Ownership Violation Attempt to access the descriptor owned by the CPU.
3	WrProt	RW 0x0	Channel0 Write Protect
2	AccProt	RW 0x0	Channel0 Access Protect Violation
1	AddrMiss	RW 0x0	Channel0 Address Miss Failed address decoding.

Table 1411:Interrupt Cause Register (Continued)

Offset: 0x000608C0

Bit	Field	Type/InitVal	Description
0	Comp	RW 0x0	Channel0 IDMA Completion

Table 1412:Interrupt Mask Register

Offset: 0x000608C4

Bit	Field	Type/InitVal	Description
31:29	Reserved	RSVD 0x0	Reserved
28:24	Various	RW 0x0	Same as Channel3 mask bits
23:21	Reserved	RSVD 0x0	Reserved
20:16	Various	RW 0x0	Same as Channel2 mask bits
15:13	Reserved	RSVD 0x0	Reserved
12:8	Various	RW 0x0	Same as Channel1 mask bits
7:5	Reserved	RSVD 0x0	Reserved
4	Own	RW 0x0	Ownership Violation Interrupt 0 = Disable 1 = Enable
3	WrProt	RW 0x0	Write Protection Interrupt 0 = Disable 1 = Enable
2	AccProt	RW 0x0	Access Protection Interrupt
1	AddrMiss	RW 0x0	Address Miss Interrupt 0 = Disable 1 = Enable
0	Comp	RW 0x0	Comp Interrupt 0 = Disable 1 = Enable

Table 1413:Error Address Register

Offset: 0x000608C8

Bit	Field	Type/InitVal	Description
31:0	ErrAddr	RW 0x0	Bits [31:0] of Error Address Latched upon any of the error events interrupts (address miss, access protection, write protection, ownership violation). Once the address is latched, no new address is latched until the register is read. **NOTE:** If the erroneous transaction was retargeted, the address reported in this field is the retargeted address, not the original address. **NOTE:**

Table 1414:Error Select Register

Offset: 0x000608CC

Bit	Field	Type/InitVal	Description
31:5	Reserved	RO 0x0	Read only.
4:0	Sel	RW 0x0	Specifies the error event currently reported in the Error Address register: 0x0 = Reserved 0x1 = AddrMiss0 0x2 = AccProt0 0x3 = WrProt0 0x4 = Own0 0x5 - 0x7 = Reserved 0x8 = Reserved 0x9 = AddrMiss1 0xA = AccProt1 0xB = WrProt1 0xC = Own1 0xD - 0xF = Reserved 0x10 = Reserved 0x11 = AddrMiss2 0x12 = AccProt2 0x13 = WrProt2 0x14 = Own2 0x15 - 0x17 = Reserved 0x18 = Reserved 0x19 = AddrMiss3 0x1A = AccProt3 0x1B = WrProt3 0x1C = Own3 0x1D - 0x1F = Reserved Read Only.

A.24 Electronic Fuse (eFuse) Registers

The following table provides a summarized list of all registers that belong to the Electronic Fuse (eFuse), including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1415:Summary Map Table for the Electronic Fuse (eFuse) Registers

Register Name	Offset	Table and Page
Secured BootROM		
Fuse Secured Boot Control Register	0x00010010	Table 1416, p. 1502
Fuse Secured Boot Box ID Register	0x00010014	Table 1417, p. 1502
Fuse Secured Boot Public Key N Register (N=0–7)	Public Key Word number0: 0x00010020, Public Key Word number1: 0x00010024, Public Key Word number2: 0x00010028, Public Key Word number3: 0x0001002C, Public Key Word number4: 0x00010030, Public Key Word number5: 0x00010034, Public Key Word number6: 0x00010038, Public Key Word number7: 0x0001003C	Table 1418, p. 1502
Fuse Secured Boot Symmetric Key N Register (N=0–3)	Symetric Key Word Number0: 0x00010080, Symetric Key Word Number1: 0x00010084, Symetric Key Word Number2: 0x00010088, Symetric Key Word Number3: 0x0001008C	Table 1419, p. 1503
Secured Boot Visibility Control Register	0x000100B0	Table 1420, p. 1503
ULT		
Unique Device ID 0 Register	0x00010000	Table 1421, p. 1503
Unique Device ID 1 Register	0x00010004	Table 1422, p. 1503
Fuse General Purpose		
Fuse General Purpose N Register (N=0–11)	Fuse GP Word0: 0x000100C0, Fuse GP Word1: 0x000100C4...Fuse GP Word11: 0x000100EC	Table 1423, p. 1504
Fuse Software Control		
Fuse Control Register	0x00010008	Table 1424, p. 1504
Fuse Protection FSB Register	0x0001000C	Table 1425, p. 1505

Copyright © 2014 Marvell

A.24.1 Secured BootROM

Table 1416:Fuse Secured Boot Control Register

Offset: 0x00010010

Bit	Field	Type/InitVal	Description
31:16	Flash ID	RW 0x0	Used fo anti-cloning of official boot image across platforms
15:8	Reserved	RSVD 0x0	Reserved
7:4	Boot Device	RW 0x0	Defines the location of the boot device. Can be one of: SPI Flash, NAND/NOR Flash, SATA.
3:2	Reserved	RSVD 0x0	Reserved
1	Secure Debug Disable	RW 0x0	A write-once register. Must be written by the boot F/W to either 0 (retain secured state) or 1 (override secured state). Any write to it locks it from further write (unlock only at sys reset). Defines rather JTAG I/F can accessed the CPU debugger.
0	Secure Boot Mode	RW 0x0	When enabled (1), direct boot from SPI, I2C and NandFlash are blocked. Used by Boot F/W.

Table 1417:Fuse Secured Boot Box ID Register

Offset: 0x00010014

Bit	Field	Type/InitVal	Description
31:0	Secure Boot Box ID	RW 0x0	Box ID For trusted debug

Table 1418:Fuse Secured Boot Public Key N Register (N=0–7)

Offset: Public Key Word number0: 0x00010020, Public Key Word number1: 0x00010024, Public Key Word number2: 0x00010028, Public Key Word number3: 0x0001002C, Public Key Word number4: 0x00010030, Public Key Word number5: 0x00010034, Public Key Word number6: 0x00010038, Public Key Word number7: 0x0001003C

Offset Formula: 0x00010020+4 * N: where N (0-7) represents Public Key Word number

Bit	Field	Type/InitVal	Description
31:0	Public Key WordN	RW 0x0	Holds Word N of the Public Key value.

Table 1419:Fuse Secured Boot Symmetric Key N Register (N=0–3)

Offset: Symetric Key Word Number0: 0x00010080, Symetric Key Word Number1: 0x00010084, Symetric Key Word Number2: 0x00010088, Symetric Key Word Number3: 0x0001008C

Offset Formula: 0x00010080+4 * N: where N (0-3) represents Symetric Key Word Number

Bit	Field	Type/InitVal	Description
31:0	Symmetric Key N	RW 0x0	Holds the value of Word N of the Symmetric Key.

Table 1420:Secured Boot Visibility Control Register

Offset: 0x000100B0

Bit	Field	Type/InitVal	Description
31:9	Reserved	RSVD 0x0	Reserved
8	Secure Boot Symetric Key Visible	RW 0x0	When this field is active, read accesses from <Fuse Secure Boot Symmetric Key> will be responded with an all 0’s word. The control bit is a write-once register and is written by boot firmware . Any write to it locks it from further writes.
7:1	Reserved	RSVD 0x0	Reserved
0	Secure Boot Trusted DBG Enable	RW 0x0	When this bit is active (‘1’) the eFuse control bit JTAG debug mode is overriden, and the debug by JTAG-ICE is allowable. When the bit non-active, the JTAG debug is dependant on the eFuse. The control bit is a write-once register and is written by boot firmware . Any write to it locks it from further writes.

A.24.2 ULT

Table 1421:Unique Device ID 0 Register

Offset: 0x00010000

Bit	Field	Type/InitVal	Description
31:0	UniqeID_val0	RO 0x0	Unique Device ID[31:0] read only, programmed by Marvell.

Table 1422:Unique Device ID 1 Register

Offset: 0x00010004

Bit	Field	Type/InitVal	Description
31:0	UniqeID_val1	RO 0x0	Unique Device ID[63:32] read only, programmed by Marvell.

A.24.3 Fuse General Purpose

Table 1423:Fuse General Purpose N Register (N=0–11)

Offset: Fuse GP Word0: 0x000100C0, Fuse GP Word1: 0x000100C4...Fuse GP Word11: 0x000100EC

Offset Formula: 0x000100C0+4*N: where N (0-11) represents Fuse GP Word

Bit	Field	Type/InitVal	Description
31:0	FuseN Low	RW 0x0	eFuse General Purpose N This register is writeable only when the corresponding <FSB> is cleared. Upon reset de-assertion, data from the eFuse is loaded into this register. If the protection bit of this register is not set, software has the ability to write any value to burn to the eFuse. **NOTE:** Per bit, clearing a value (that is, writing 0x0 to a bit that holds the value of 0x1) is not permitted. Since a fuse bit cannot be erased, clearing a bit will cause the data stored in the fuse after the trigger command to be different than the data in this register.

A.24.4 Fuse Software Control

Table 1424:Fuse Control Register

Offset: 0x00010008

Bit	Field	Type/InitVal	Description
31	Fuse Access Done	RO 0x0	Cleared when one of the Write Triggers is written and when an upload trigger is written Set upon completion of eFuse access process.
30	Fuse Burn Success	RO 0x0	When Burn_Done=1, this field specifies if the burn has been done successfully.
29:16	Reserved	RSVD 0x0	Reserved
15:8	Fuse Burn Address	RW 0x0	Upon a write of 0x1 to <BurnTrigger>, this field defines which Fuse Register will be burned. The actual fuse that will be burned is the one located at: 0x10000 + <BurnAddr>. The address is treated as word-aligned, i.e., bits[1:0] are ignored and treated as 0. If the address points to a register that does not contain Fuse Data, the burn trigger is ignored.
7:3	Reserved	RSVD 0x0	Reserved

Copyright © 2014 Marvell

Table 1424:Fuse Control Register (Continued)

Offset: 0x00010008

Bit	Field	Type/InitVal	Description
2	Fuse Upload Trigger	WO 0x0	When <Burn_Mode> is active and 0x1 is written into this field, the Fuse corresponding to the register pointed by <Fuse_Burn_Addr> is re-read and its data is uploaded into the corresponding register.
1	Fuse Burn Trigger	WO 0x0	When Burn mode is active and 0x1 is written to this field, the data in the register pointed by <Fuse_Burn_Addr> is burned into its corresponding Fuse. The FSB of the corresponding Fuse will be written also according to the FSB field of its group.
0	Fuse Access Mode	RW 0x0	eFuse Access Mode Set on burning eFuse or on uploading eFuse values to registers. Setting this bit is possible only when one of the FSB bits is cleared. Any writes to Fuse data registers (<Fuse_Secure_Boot_Control>, <Fuse_Scure_Boot_Public_Key> <Fuse_Scure_Boot_Symetric_Key>, <fuse_General_Purpose>), Fuse Protection or to Fuse Burn Trigger registers is ignored when this bit is cleared

Table 1425:Fuse Protection FSB Register

Offset: 0x0001000C

Bit	Field	Type/InitVal	Description
31:20	Reserved	RSVD 0x0	Reserved
19	FSB19	RW 0x0	Fuse Security Bit 19
18	FSB18	RW 0x0	Fuse Security Bit 18
17	FSB17	RW 0x0	Fuse Security Bit 17
16	FSB16	RW 0x0	Fuse Security Bit 16
15	FSB15	RW 0x0	Fuse Security Bit 15
14	FSB14	RW 0x0	Fuse Security Bit 14
13	FSB13	RW 0x0	Fuse Security Bit 13
12	FSB12	RW 0x0	Fuse Security Bit 12
11	FSB11	RW 0x0	Fuse Security Bit 11

Table 1425:Fuse Protection FSB Register (Continued)

Offset: 0x0001000C

Bit	Field	Type/InitVal	Description
10	FSB10	RW 0x0	Fuse Security Bit 10
9	FSB9	RW 0x0	Fuse Security Bit 9
8	FSB8	RW 0x0	Fuse Security Bit 8
7	FSB7	RW 0x0	Fuse Security Bit 7
6	FSB6	RW 0x0	Fuse Security Bit 6
5	FSB5	RW 0x0	Fuse Security Bit 5
4	FSB4	RW 0x0	Fuse Security Bit 4
3	FSB3	RW 0x0	Fuse Security Bit 3
2	FSB2	RW 0x0	Fuse Security Bit 2
1	FSB1	RW 0x0	Fuse Security Bit 1
0	FSB0	RW 0x0	Fuse Security Bit 0

A.25 Core and CPU AVS Registers

The following table provides a summarized list of all registers that belong to the Core and CPU AVS, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1426:Summary Map Table for the Core and CPU AVS Registers

Register Name	Offset	Table and Page
Core AVS		
Core AVS Control 0 Register	0x00018300	Table 1427, p. 1507
Core AVS Control 2 Register	0x00018308	Table 1428, p. 1508
CPU AVS		
AVS Control 0 Register	0x00020860	Table 1429, p. 1508
AVS Control 2 Register	0x00020868	Table 1430, p. 1509

A.25.1 Core AVS

Table 1427:Core AVS Control 0 Register

Offset: 0x00018300

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x40380	Reserved
7:4	AvsHighVDDLimit	RW 0x7	Set the upper limit of VDD 0 = 0.9125V 1 = 0.925V 2 = 0.9375V 3 = 0.95V 4 = 0.9625V 5 = 0.975V 6 = 0.9875V 7 = 1V
3:0	AvsLowVDDLimit	RW 0xf	Set the lower limit of VDD 5 = 0.7875V 6 = 0.8V 7 = 0.8125V 8 = 0.825V 9 = 0.8375V 10 = 0.85V 11 = 0.8625V 12 = 0.875V 13 = 0.8875V 14 = 0.9V 15 = 0.9125V

Table 1428:Core AVS Control 2 Register

Offset: 0x00018308

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x4080	Reserved
9	AvsEnable	RW 0x0	Enable the AVS Mechanism 0 = Disable: Value of vddfb = vdd. 1 = Enable: AVS is on.
8:0	Reserved	RSVD 0x0	Reserved

A.25.2 CPU AVS

Table 1429:AVS Control 0 Register

Offset: 0x00020860

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x40380	Reserved
7:4	AvsHighVDDLimit	RW SAR	AVS VDD High Limit 0 = 0.9125v 1 = 0.925v 2 = 0.9375v 3 = 0.95v 4 = 0.9625v 5 = 0.975v 6 = 0.9875v 7 = 1v 8 = 1.0125v 9 = 1.025v 10 = 1.0375v 11 = 1.05v 12 = 1.0625v 13 = 1.075v 14 = 1.0875v 15 = 1.1v

Table 1429:AVS Control 0 Register (Continued)

Offset: 0x00020860

Bit	Field	Type/InitVal	Description
3:0	AvsLowVDDLimit	RW 0xF	AVS VDD Low Limit 5 = 0.7875v 6 = 0.8v 7 = 0.8125v 8 = 0.825v 9 = 0.8375v 10 = 0.85v 11 = 0.8625v 12 = 0.875v 13 = 0.8875v 14 = 0.9v 15 = 0.9125v

Table 1430:AVS Control 2 Register

Offset: 0x00020868

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24	AvsSwRst	RW 0x1	Clearing this bit asserts port rst_b of MISL_0040G_AVS.
23:10	Reserved	RSVD 0x88	Reserved
9	AvsEnable	RW 0x1	AVS enable control 0 = OFF: AVS is off, vddfb=vdd. 1 = ON: AVS is on.
8:0	Reserved	RSVD 0x0	Reserved

A.26 System Registers

The following table provides a summarized list of all registers that belong to the System, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1431:Summary Map Table for the System Registers

Register Name	Offset	Table and Page
Configuration and Control		
SoC Control Register	0x00018204	Table 1432, p. 1511
SoC Device Multiplex Register	0x00018208	Table 1433, p. 1512
System Pre-Load Register	0x0001822C	Table 1434, p. 1513
Device ID Register	0x00018238	Table 1435, p. 1514
CPU SoC ID Register	0x0001823C	Table 1436, p. 1514
SYSRSTn Length Counter Register	0x00018250	Table 1437, p. 1514
RSTOUTn Mask Register	0x00018260	Table 1438, p. 1514
System Soft Reset Register	0x00018264	Table 1439, p. 1516
Thermal Sensor Status Register	0x000182B0	Table 1440, p. 1516
PCI Express Boot Address Register	0x000182D4	Table 1441, p. 1516
General Purpose 0 Register	0x000182E0	Table 1442, p. 1517
General Purpose 1 Register	0x000182E4	Table 1443, p. 1517
Lvds Pads Control Register	0x000182F0	Table 1444, p. 1518
Communication PHY Reference Clock Alignment Register	0x000182F8	Table 1445, p. 1519
SSCG Configuration Register	0x000184D8	Table 1446, p. 1521
IO Configuration 0 Register	0x000184E0	Table 1447, p. 1521
IO Configuration 5 Register	0x000184F0	Table 1448, p. 1522
QSGMII Control and Status		
QSGMII Control 0 Register	0x00018400	Table 1449, p. 1522
QSGMII Control 1 Register	0x00018404	Table 1450, p. 1523
QSGMII Status Register	0x00018408	Table 1451, p. 1524
SERDES		
Shared SERDES 0-7 Selectors Register	0x00018270	Table 1452, p. 1525
Shared SERDES 8-15 Selectors Register	0x00018274	Table 1453, p. 1526
Power Management and Memory Power Down		
Power Management Memory Power Down 1 Register	0x0001820C	Table 1454, p. 1527
Power Management Memory Power Down 2 Register	0x00018210	Table 1455, p. 1528
Power Management Memory Power Down 3 Register	0x00018214	Table 1456, p. 1529
Power Management Memory Power Down 4 Register	0x00018218	Table 1457, p. 1530
Power Management Memory Power Down 5 Register	0x0001821C	Table 1458, p. 1531
Power Management Clock Gating Control Register	0x00018220	Table 1459, p. 1533
Thermal Manager		
Thermal Manager Control and Status Register	0x000184C4	Table 1460, p. 1535
Thermal Manager Cooling Delay Register	0x000184C8	Table 1461, p. 1536

Table 1431:Summary Map Table for the System Registers (Continued)

Register Name	Offset	Table and Page
Thermal Manager Overheat Delay Register	0x000184CC	Table 1462, p. 1536
Thermal Sensor Configuration0 Register	0x000184D0	Table 1463, p. 1536
BootROM		
BootROM Routine and Error Code Register	0x000182D0	Table 1464, p. 1537
Interrupt		
Reserved Interrupt Cause Register	0x000182A0	Table 1465, p. 1539
Reserved Interrupt Mask Register	0x000182A4	Table 1466, p. 1539
Reserved Error Interrupt Cause Register	0x000182A8	Table 1467, p. 1539
Reserved Error Interrupt Mask Register	0x000182AC	Table 1468, p. 1540
Design for Testablity (DFT)		
PUP Enable Register	0x0001864C	Table 1469, p. 1540

A.26.1 Configuration and Control

Table 1432:SoC Control Register

Offset: 0x00018204

Bit	Field	Type/InitVal	Description
31	Reserved_32	RW 0x0	Reserved Must be 0.
30:10	Reserved	RO 0x0	
9	DMASharedTrans	RW 0x1	Must be 1 0 = NonShared: Non-Shared transaction on xbar 1 = Shared: Shared transaction on xbar
8	PCIe1QuadBy1En	RW 0x1	PCI Express ports configuration. Changing this field must be done at system initialization and before the assertion of PCIe0En. To access PCIe1.1, PCIe1.2, and PCIe1.3 from any agent the value of this field must be Enable.
7	PCIe0QuadBy1En	RW 0x0	PCI Express ports configuration. Changing this field must be done at system initialization and before the assertion of PCIe0En. To access PCIe0.1, PCIe0.2, and PCIe0.3 from any agent the value of this field must be Enable.
6	Reserved_6	RO 0x0	

Table 1432:SoC Control Register (Continued)

Offset: 0x00018204

Bit	Field	Type/InitVal	Description
5	PCIe1_CLKOUT_EN	RW 0x1	PCIe1 CLKOUT Enable 0 = Clock Out Disable 1 = Clock Out Enable
4	PCIe0_CLKOUT_EN	RW 0x1	PCIe0 CLKOUT Enable 0 = Clock Out Disable 1 = Clock Out Enable
3	PCIe3En	RW 0x0	PCI Express port3 Enable 0 = Disable 1 = Enable
2	PCIe2En	RW 0x0	PCI Express port2 Enable 0 = Disable 1 = Enable
1	PCIe1En	RW 0x0	PCI Express port1 Enable. **NOTE:** If PCIe1QuadBy1En is QuadX1 then this field is relevant to all the 4*PCIe_X1. 0 = Disable 1 = Enable
0	PCIe0En	RW 0x0	PCI Express port0.X Enable **NOTE:** If PCIe0QuadBy1En is QuadX1 then this field is relevant to all the 4*PCIe_X1. 0 = Disable 1 = Enable

Table 1433:SoC Device Multiplex Register

Offset: 0x00018208

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27	NfArbiterEn	RW 0x0	Enables arbitration between device and NAND Flash
26	NfExpandTrrTtimingEn	RW 0x0	Controls minimal tRR delay time (NF_REn de-assertion to NF_RDYn assertion). 0 = Minimal: Delay (4-5 Clock cycles) 1 = Expand Minimal Delay: (4-5 Clock cycles) by additional 5 Clock cycles.

Table 1433:SoC Device Multiplex Register (Continued)

Offset: 0x00018208

Bit	Field	Type/InitVal	Description
25	NfCtrlIntMask	RW 0x0	NAND Controller Interrupt Mask Register 0 = Interrupt mask 1 = Interrupt enable
24	NfCtrlIntCause	RW0C 0x0	NAND Controller Interrupt Cause Register
23	Reserved	RSVD 0x0	Reserved
22	NfClkInvert	RW 0x0	NAND clock polarity 0 = Normal 1 = Inverted
21	NfCoreSwRst_	RW 0x1	NAND Core clock software reset
20	NfEccSwRst_	RW 0x1	NAND ECC clock software reset
19	NfAddressMaskEn	RW 0x0	NAND External Access Enable 0 = Disable: Internal access only (for 4B PIO) 1 = Enable: Enables external access (for 8B PIO).
18	NfCsExpansionEn	RW 0x0	Chip Select Expansion Enable
17	NfWrErrPropEn	RW 0x0	Write Response Error Propagate Enable
16:1	Reserved	RSVD 0x0	Reserved
0	NandFlashEnabled	RW SAR	Configure NAND Flash Enable 0 = Disable: NAND flash controller is disabled. 1 = Enable: NAND flash controller is enabled.

Table 1434:System Pre-Load Register

Offset: 0x0001822C

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

Table 1435:Device ID Register

Offset: 0x00018238

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

Table 1436:CPU SoC ID Register

Offset: 0x0001823C

Bit	Field	Type/InitVal	Description
31:24	SOC ID	RW 0x0	See the Hardware Specification for more details.
23:16	Reserved	RSVD 0x0	Reserved
15:0	DEVICE ID	RO 0x0	See the Hardware Specification for more details.

Table 1437:SYSRSTn Length Counter Register

Offset: 0x00018250

Bit	Field	Type/InitVal	Description
31	Clr	WO 0x0	Writing 0x1 to this field clears the <Count> field.
30:29	Reserved	RSVD 0x0	Reserved
28:0	Count	RO 0x0	SYSRST Length Count. The value in this field specifies the number of REF_CLK cycles that the SYSRSTn was asserted. Software may clear this field by writing 0x1 to <Clr>.

Table 1438:RSTOUTn Mask Register

Offset: 0x00018260

Bit	Field	Type/InitVal	Description
31:9	Reserved	RSVD 0x0	Reserved
8	PCIe3RstOutMaskTrst	RW 0x1	Mask PCIe3_Rst_Out_ from influencing the Trst_Tree. 0 = Do Not Mask: Do not mask PCIe3_Rst_Out from influencing the Trst_Tree. 1 = Mask: Mask PCIe3_Rst_Out from influencing the Trst_Tree.

Table 1438:RSTOUTn Mask Register (Continued)

Offset: 0x00018260

Bit	Field	Type/InitVal	Description
7	PCIe2RstOutMaskTrst	RW 0x1	Mask PCIe2_Rst_Out_ from influencing the Trst_Tree. 0 = Do Not Mask: Do not mask PCIe2_Rst_Out from influencing the Trst_Tree. 1 = Mask: Mask PCIe2_Rst_Out from influencing the Trst_Tree.
6	PCIe1RstOutMaskTrst	RW 0x1	Mask PCIe1_Rst_Out_ from influencing the Trst_Tree. 0 = Do Not Mask: Do not mask PCIe1_Rst_Out from influencing the Trst_Tree. 1 = Mask: Mask PCIe1_Rst_Out from influencing the Trst_Tree.
5	PCIe0RstOutMaskTrst	RW 0x1	Mask PCIe0_Rst_Out_ from influencing the Trst_Tree. 0 = Do Not Mask: Do not mask PCIe0_Rst_Out from influencing the Trst_Tree. 1 = Mask: Mask PCIe0_Rst_Out from influencing the Trst_Tree.
4	PCIe3RstOutMaskSysRstOut	RW 0x1	Mask PCIe3_Rst_Out from influencing the SYSRST_OUTn 0 = Do Not Mask: Do not mask PCIe3_Rst_Out from influencing the SYSRST_OUTn. 1 = Mask: Mask PCIe3_Rst_Out from influencing the SYSRST_OUTn.
3	PCIe2RstOutMaskSysRstOut	RW 0x1	Mask PCIe2_Rst_Out from influencing the SYSRST_OUTn 0 = Do Not Mask: Do not mask PCIe2_Rst_Out from influencing the SYSRST_OUTn. 1 = Mask: Mask PCIe2_Rst_Out from influencing the SYSRST_OUTn.
2	PCIe1RstOutMaskSysRstOut	RW 0x1	Mask PCIe1_Rst_Out from influencing the SYSRST_OUTn 0 = Do Not Mask: Do not mask PCIe1_Rst_Out from influencing the SYSRST_OUTn. 1 = Mask: Mask PCIe1_Rst_Out from influencing the SYSRST_OUTn.
1	PCIe0RstOutMaskSysRstOut	RW 0x1	Mask PCIe0_Rst_Out from influencing the SYSRST_OUTn 0 = Do Not Mask: Do not mask PCIe0_Rst_Out from influencing the SYSRST_OUTn. 1 = Mask: Mask PCIe0_Rst_Out from influencing the SYSRST_OUTn.
0	GlobalSoftRstOutEn	RW 0x0	If set to 1, the device asserts RSTOUTn upon SW reset.

Table 1439:System Soft Reset Register

Offset: 0x00018264

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	CommSwRstn	RW 0x1	0x0 - Reset is asserted. 0x1 - Reset is de-asserted.
0	GlobalSoftRst	RW 0x0	When SW set this bit to 1 and bit <SoftRstOutEn> is set to 1 in the RSTOUTn Mask Register, the device asserts the RSTOUTn pin.

Table 1440:Thermal Sensor Status Register

Offset: 0x000182B0

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27:19	tsen_temp_out	RO 0x0	
18:10	tsen_temp_out_int	RO 0x0	
9:2	tsen_cal_value	RO 0x0	
1	tsen_cal	RO 0x0	
0	tsen_t_valid	RO 0x0	

Table 1441:PCI Express Boot Address Register

Offset: 0x000182D4

Bit	Field	Type/InitVal	Description
31:0	BootAddr	RW 0xFFFF0000	The 32-bit address of the bootROM header. After initializing the PCI Express to operate as an Endpoint, the bootROM sets this register to 0xFFFFFFFF and waits for the register to be updated by the Root Complex.

Table 1442:General Purpose 0 Register

Offset: 0x000182E0

Bit	Field	Type/InitVal	Description
31:3	Reserved	RSVD 0x1FFFE000	Reserved
2	TDMDelayTuningEn	RW 0x0	0 = False 1 = True
1	Reserved	RSVD 0x0	Reserved
0	AvsCoreAvddDetEn	RW 0x0	AVS_AVDD Detect Enable If enabled, the AVS will be disabled once AVDD drops below a preset value. 0 = False: AVDD power down detection is disabled 1 = True: AVDD power down detection is enabled

Table 1443:General Purpose 1 Register

Offset: 0x000182E4

Bit	Field	Type/InitVal	Description
31:6	Reserved	RSVD 0x3FFFC00	Reserved
5	RdHoldADEn	RW 0x1	When set, DEV_A[2:0] remains stable between OEn de-assertion and CS de-assertion. 0 = False 1 = True
4	DevBootCSAssertDelayEn	RW 0x0	Defines if DEV ADDR is guaranteed to be stable 4 core clock cycles before BootCS assertion. Setting this bit requires changing RdSetup of the corresponding CS to be at least 0x4. And, Ale2Wr of the corresponding CS to be at least 2*(Addr2ALE+1) + 6. 0 = False 1 = True
3	CS3Setup	RW 0x0	Defines if DEV ADDR is guaranteed to be stable 4 core clock cycles before CS3 assertion. Setting this bit requires changing RdSetup of the corresponding CS to be at least 0x4. And, Ale2Wr of the corresponding CS to be at least 2*(Addr2ALE+1) + 6 . 0 = False: DEV_A[2:0] is set together with CS assertion 1 = True: DEV_A[2:0] is set four cycles prio to CS assertion

Table 1443:General Purpose 1 Register (Continued)

Offset: 0x000182E4

Bit	Field	Type/InitVal	Description
2	CS2Setup	RW 0x0	Defines if DEV ADDR is guaranteed to be stable 4 core clock cycles before CS2 assertion. Setting this bit requires changing RdSetup of the corresponding CS to be at least 0x4. And, Ale2Wr of the corresponding CS to be at least 2*(Addr2ALE+1) + 6 . 0 = False: DEV_A[2:0] is set together with CS assertion 1 = True: DEV_A[2:0] is set four cycles prio to CS assertion
1	CS1Setup	RW 0x0	Defines if DEV ADDR is guaranteed to be stable 4 core clock cycles before CS1 assertion. Setting this bit requires changing RdSetup of the corresponding CS to be at least 0x4. And, Ale2Wr of the corresponding CS to be at least 2*(Addr2ALE+1) + 6 . 0 = False: DEV_A[2:0] is set together with CS assertion 1 = True: DEV_A[2:0] is set four cycles prio to CS assertion
0	CS0Setup	RW 0x0	Defines if DEV ADDR is guaranteed to be stable 4 core clock cycles before CS0 assertion. Setting this bit requires changing RdSetup of the corresponding CS to be at least 0x4. And, Ale2Wr of the corresponding CS to be at least 2*(Addr2ALE+1) + 6 . 0 = False: DEV_A[2:0] is set together with CS assertion 1 = True: DEV_A[2:0] is set four cycles prio to CS assertion

Table 1444:Lvds Pads Control Register

Offset: 0x000182F0

Bit	Field	Type/InitVal	Description
31:22	Reserved	RSVD 0x3FF	Reserved
21	Reserved	RW 0x1	Reserved Must be 0x1.
20	Conf Lvds PD 4	RW 0x1	Power mode input for LVDS_Data_Out_4-Pad. 0 = Pad is active. 1 = Pad is in Powered Down mode.
19	Conf Lvds PD 3	RW 0x1	Power mode input for LVDS_Data_Out_3-PAD. 0 = Pad is active. 1 = Pad is in Powered Down mode.
18	Conf Lvds PD 2	RW 0x1	Power mode input for LVDS_Data_Out_2-PAD. 0 = Pad is active. 1 = Pad is in Powered Down mode.

Table 1444:Lvds Pads Control Register (Continued)

Offset: 0x000182F0

Bit	Field	Type/InitVal	Description
17	Conf Lvds PD 1	RW 0x1	Power mode input for LVDS_Data_Out_1-PAD. 0 = Pad is active. 1 = Pad is in Powered Down mode.
16	Conf Lvds PD 0	RW 0x1	Power mode input for LVDS_Data_Out_0-PAD. 0 = Pad is active. 1 = Pad is in Powered Down mode.
15:0	Reserved	RSVD 0x0	Reserved

Table 1445:Communication PHY Reference Clock Alignment Register

Offset: 0x000182F8

Bit	Field	Type/InitVal	Description
COMPHY_RefClk_Alignment			
31:16	Reserved	RSVD 0xFFFF	Reserved
15	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
14	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
13	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
12	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
11	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
10	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align

Table 1445:Communication PHY Reference Clock Alignment Register (Continued)

Offset: 0x000182F8

Bit	Field	Type/InitVal	Description
9	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
8	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
7	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
6	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
5	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
4	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
3	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
2	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
1	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align
0	Comphy%PhyPortPCIeX4Mode	RW 0x0	0 = Normal_Mode: ComPhy[<%phy_port>]-RefClk_Align = 0x0 1 = PCIe_X4_Mode: RefClk_Align_out of ComPhy0 is connected to ComPhy[x]-RefClk_Align

Table 1446:SSCG Configuration Register

Offset: 0x000184D8

Bit	Field	Type/InitVal	Description
31:18	Reserved	RSVD 0x0	Reserved
17:16	SSCG Mode	RW 0x0	SSCG Configuration 0 = Spread down 1 = Spread up 2 = Central spread 3 = Reserved
15:8	SSCG Low Boundary	RW 0x3e	See <SSCG_High_Boundary> for a description. Note: SSCG_High_Boundary must be greater than SSCG_Low_ Boundary.
7:0	SSCG High Boundary	RW 0x80	Notation: - H = SSCG_High_Boundary - L = SSCG_Low_Boundary Modulation frequency: - For up spread or down spread: Modulation frequency = 550 MHz/ [(H-L)*H*2] - For center spread: Modulation frequency = 550 MHz/ [(H-L)*H*4] Spread percentage: - For up spread or down spread: (1/96)*(H-L)/H pk to pk - For center spread: (1/96)*(H-L)/H pk to pk For example, to set ~33 KHz modulation frequency and ~0.5% down spread: H = 128 and L = 64.

Table 1447:IO Configuration 0 Register

Offset: 0x000184E0

Bit	Field	Type/InitVal	Description
31:10	Reserved	RSVD 0x4402A	Reserved
9:8	VDDO Device Pads Voltage	RW SAR	This field set the voltage for pads: DEV* and MPP[66:48]. 0 = 1.8 V 2 = 3.3 V

Table 1447:IO Configuration 0 Register (Continued)

Offset: 0x000184E0

Bit	Field	Type/InitVal	Description
7:6	VDDO D Pads Voltage	RW 0x2	This field set the voltage for pads: MPP[47:36]. 2 = 3.3 V
5:4	VDDO C Pads Voltage	RW SAR	This field set the voltage for pads: MPP[35:24]. 0 = 1.8 V 2 = 3.3 V
3:2	VDDO B Pads Voltage	RW 0x2	This field set the voltage for pads: MPP[23:12]. 0 = 1.8 V 1 = 2.5 V 2 = 3.3 V
1:0	VDDO A Pads Voltage	RW 0x2	This field belongs to pads: GE_MDC, GE_MDIO, and MPP[11:0]. 0 = 1.8 V 1 = 2.5 V 2 = 3.3 V

Table 1448:IO Configuration 5 Register

Offset: 0x000184F0

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

A.26.2 QSGMII Control and Status

Table 1449:QSGMII Control 0 Register

Offset: 0x00018400

Bit	Field	Type/InitVal	Description
31	TestModeEn	RW 0x0	Enable Test mode **NOTE:** This field may be changed only when <PortReset> is HIGH. 0 = Disabled: Test mode disabled (functional mode). 1 = Enabled: Test mode enabled, according to <TestMode>.
30	TestGenEn	RW 0x0	Enable the test configured in <TestMode> generation. 0 = Disabled 1 = Enabled

Table 1449:QSGMII Control 0 Register (Continued)

Offset: 0x00018400

Bit	Field	Type/InitVal	Description
29:20	CyclicData2	RW 0x0	CyclicData2[9:0] See CyclicData0 for details.
19:10	CyclicData1	RW 0x0	CyclicData1[9:0] See CyclicData0 for details.
9:0	CyclicData0	RW 0x0	CyclicData0[9:0] The data to be transmitted when <TestMode> is Cyclic Data. Cyclic Data pattern is: <CyclicData0><CyclicData1><CyclicData2><CyclicData3>. **NOTE:** Data is transmitted from left to right ,i.e., <CyclicData0> =><CyclicData1> etc. **NOTE:** Bit0 (LSB) in each register is transmitted first and Bit9 (MSB) is transmitted last. **NOTE:**

Table 1450:QSGMII Control 1 Register

Offset: 0x00018404

Bit	Field	Type/InitVal	Description
31	QsgmiiRstn	RW 0x1	QSGMII reset, AL 0 = Reset mode 1 = Normal mode
30	QsgmiiActive	RW 0x0	0 = QsgmiiDisconnected: Serdes Connected to the Gunits 1 = QsgmiiConnected: Qsgmii Connected to the Gunits
29:25	Reserved	RSVD 0x1	Reserved
24	QsgmiiResetPrbsCnt	RW 0x0	Reset PRBS error counter
23:17	Reserved	RSVD 0x0	Reserved
16	Reserved	RW 0x0	
15	PrbsCheckEn	RW 0x0	[15] - PrbsCheckEn Enable PRBS checker 0 = Disabled 1 = Enabled

Table 1450:QSGMII Control 1 Register (Continued)

Offset: 0x00018404

Bit	Field	Type/InitVal	Description
14	UseCommaDetectSyncSM	RW 0x0	UseCommaDetectSyncSM Enable second Sync SM in Comma Detect block - non standard mode (compatible with S-Unit mode) **NOTE:** This field may be changed only when <PortReset> is HIGH. 0 = Disable 1 = Enable
13	LoopBackEn	RW 0x0	Enable loopback This is the loopback SERDES from receive to transmit. 0 = Disable 1 = Enable
12	UseTestClk	RW 0x0	Enable use of the external clock. 0 = Use SERDES clocks 1 = Use test clock
11:10	TestMode	RW 0x0	Test mode configuration **NOTE:** This field may be changed only when <PortReset> is HIGH. 0 = Cyclic Data: The data in Cyclic Date Register 0 - 3 is transmitted. 1 = PRBS-7 2 = PRBS-23 3 = Reserved
9:0	CyclicData3	RW 0x0	CyclicData3 The data to be transmitted when <TestMode> is Cyclic Data. Cyclic Data pattern is: <CyclicData0><CyclicData1><CyclicData2><CyclicData3>. **NOTE:** Data is transmitted from left to right ,i.e., <CyclicData0> =><CyclicData1> etc. **NOTE:** Bit0 (LSB) in each register is transmitted first and Bit9 (MSB) is transmitted last. **NOTE:**

Table 1451:QSGMII Status Register

Offset: 0x00018408

Bit	Field	Type/InitVal	Description
31:8	Reserved	RSVD 0x0	Reserved
7:3	Reserved	RO 0x0	

Table 1451:QSGMII Status Register (Continued)

Offset: 0x00018408

Bit	Field	Type/InitVal	Description
2	PrbsCheckLock	RO 0x0	PrbsCheckLock PRBS checker is locked. Indicates that the PRBS checker has locked alignment to the bit-stream received from the SERDES PRBS generator. 0 = Not locked 1 = Locked
1	SyncOK	RO 0x0	SyncOK Rx synchronization SM has reached synchronization. 0 = Not synchronized 1 = Synchronized
0	CommaAligned	RO 0x0	Comma is detected. 0 = Not detected 1 = Detected

A.26.3 SERDES

Table 1452:Shared SERDES 0-7 Selectors Register

Offset: 0x00018270

Bit	Field	Type/InitVal	Description
31:28	SERDES7 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe13 2 = SGMII0 3 = SGMII3 4 = sETM1
27:24	SERDES6 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe12 2 = SATA0 3 = SGMII2 4 = SGMII0
23:20	SERDES5 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe11 2 = SATA1 3 = SGMII1 4 = QSGMII

Table 1452:Shared SERDES 0-7 Selectors Register (Continued)

Offset: 0x00018270

Bit	Field	Type/InitVal	Description
19:16	SERDES4 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe10 2 = SATA0 3 = SGMII2 4 = sETM0
15:12	SERDES3 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe03 2 = SGMII1 3 = QSGMII
11:8	SERDES2 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe02 2 = SGMII0 3 = sETM1
7:4	SERDES1 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe01
3:0	SERDES0 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe00

Table 1453:Shared SERDES 8-15 Selectors Register

Offset: 0x00018274

Bit	Field	Type/InitVal	Description
31:28	SERDES15 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe33
27:24	SERDES14 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe32
23:20	SERDES13 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe31

Table 1453:Shared SERDES 8-15 Selectors Register (Continued)

Offset: 0x00018274

Bit	Field	Type/InitVal	Description
19:16	SERDES12 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe30
15:12	SERDES11 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe23
11:8	SERDES10 Selector	RW 0x0	Shared SERDES vselector. 0 = Unconnected 1 = PCIe22
7:4	SERDES9 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe21
3:0	SERDES8 Selector	RW 0x0	Shared SERDES selector. 0 = Unconnected 1 = PCIe20

A.26.4 Power Management and Memory Power Down

Table 1454:Power Management Memory Power Down 1 Register

Offset: 0x0001820C

Bit	Field	Type/InitVal	Description
Memory power down			
31:7	Reserved	RSVD 0x0	Reserved
6:4	Crypto1 Mem Power Down	RW 0x0	Crypto1 Memory Power Down 0 = Functional 5 = PowerDown
3	Reserved	RSVD 0x0	Reserved
2:0	Communication Mem Power Down	RW 0x0	Communication Unit Memory Power Down 0 = Functional 5 = PowerDown

Table 1455:Power Management Memory Power Down 2 Register

Offset: 0x00018210

Bit	Field	Type/InitVal	Description
Memory power down			
31:30	Reserved	RW 0x0	Reserved
29:27	PCIe30 Mem Power Down	RW 0x0	PCIe30 Memory Power Down 0 = Functional 5 = PowerDown
26:24	PCIe20 Mem Power Down	RW 0x0	PCIe20 Memory Power Down 0 = Functional 5 = PowerDown
23:21	PCIe13 Mem Power Down	RW 0x0	PCIe13 Memory Power Down 0 = Functional 5 = PowerDown
20:18	PCIe12 Mem Power Down	RW 0x0	PCIe12 Memory Power Down 0 = Functional 5 = PowerDown
17:15	PCIe11 Mem Power Down	RW 0x0	PCIe11 Memory Power Down 0 = Functional 5 = PowerDown
14:12	PCIe10 Mem Power Down	RW 0x0	PCIe10 Memory Power Down 0 = Functional 5 = PowerDown
11:9	PCIe03 Mem Power Down	RW 0x0	PCIe03 Memory Power Down 0 = Functional 5 = PowerDown
8:6	PCIe02 Mem Power Down	RW 0x0	PCIe02 Memory Power Down 0 = Functional 5 = PowerDown

Table 1455:Power Management Memory Power Down 2 Register (Continued)

Offset: 0x00018210

Bit	Field	Type/InitVal	Description
5:3	PCIe01 Mem Power Down	RW 0x0	PCIe01 Memory Power Down 0 = Functional 5 = PowerDown
2:0	PCIe00 Mem Power Down	RW 0x0	PCIe00 Memory Power Down 0 = Functional 5 = PowerDown

Table 1456:Power Management Memory Power Down 3 Register

Offset: 0x00018214

Bit	Field	Type/InitVal	Description
Memory power down			
31:27	Reserved	RW 0x0	
26:24	Embedded trace buffer Mem Power Down	RW 0x0	NCSMemory Power Down 0 = Functional 5 = PowerDown
23:21	Coherency FabricMem Power Down	RW 0x0	CFU Memory Power Down 0 = Functional 5 = PowerDown
20:18	L2 Mem Power Down	RW 0x0	L2Memory Power Down 0 = Functional 5 = PowerDown
17:15	Coherency IO Bridge Mem Power Down	RW 0x0	CIB Memory Power Down 0 = Functional 5 = PowerDown
14:12	DRAM Mem Power Down	RW 0x0	DRAM Memory Power Down 0 = Functional 5 = PowerDown

Table 1456:Power Management Memory Power Down 3 Register (Continued)

Offset: 0x00018214

Bit	Field	Type/InitVal	Description
11:9	CPU3 Mem Power Down	RW 0x0	CPU3 Memory Power Down 0 = Functional 5 = PowerDown
8:6	CPU2 Mem Power Down	RW 0x0	CPU2 Memory Power Down 0 = Functional 5 = PowerDown
5:3	CPU1 Mem Power Down	RW 0x0	CPU1 Memory Power Down 0 = Functional 5 = PowerDown
2:0	CPU0 Mem Power Down	RW 0x0	CPU0 Memory Power Down 0 = Functional 5 = PowerDown

Table 1457:Power Management Memory Power Down 4 Register

Offset: 0x00018218

Bit	Field	Type/InitVal	Description
Memory power down			
31:30	Reserved	RW 0x0	Reserved
29:27	NF Mem Power Down	RW 0x0	NF Memory Power Down 0 = Functional 5 = PowerDown
26:24	XOR1 Mem Power Down	RW 0x0	XOR1 Memory Power Down 0 = Functional 5 = PowerDown
23:21	Device Mem Power Down	RW 0x0	Device Memory Power Down 0 = Functional 5 = PowerDown

Table 1457:Power Management Memory Power Down 4 Register (Continued)

Offset: 0x00018218

Bit	Field	Type/InitVal	Description
20:18	Crypto0 Mem Power Down	RW 0x0	Crypto0 Memory Power Down 0 = Functional 5 = PowerDown
17:15	XOR0 Mem Power Down	RW 0x0	XOR0 Memory Power Down 0 = Functional 5 = PowerDown
14:12	IDMA Mem Power Down	RW 0x0	IDMA Memory Power Down 0 = Functional 5 = PowerDown
11:9	USB2 Mem Power Down	RW 0x0	USB2 Memory Power Down 0 = Functional 5 = PowerDown
8:6	USB1 Mem Power Down	RW 0x0	USB1 Memory Power Down 0 = Functional 5 = PowerDown
5:3	USB0 Mem Power Down	RW 0x0	USB0 Memory Power Down 0 = Functional 5 = PowerDown

Table 1458:Power Management Memory Power Down 5 Register

Offset: 0x0001821C

Bit	Field	Type/InitVal	Description
Memory power down			
31:30	Reserved	RW 0x0	Reserved
29:27	Sata1 Core Mem Power Down	RW 0x0	Sata1 Core Memory Power Down 0 = Functional 5 = PowerDown

Table 1458:Power Management Memory Power Down 5 Register (Continued)

Offset: 0x0001821C

Bit	Field	Type/InitVal	Description
26:24	Sata1 Phy Mem Power Down	RW 0x0	Sata1 Phy Memory Power Down 0 = Functional 5 = PowerDown
23:21	Sata0 Core Mem Power Down	RW 0x0	Sata0 Core Memory Power Down 0 = Functional 5 = PowerDown
20:18	Sata0 Phy Mem Power Down	RW 0x0	Sata0 Phy Memory Power Down 0 = Functional 5 = PowerDown
17:15	BM Mem Power Down	RW 0x0	BM Memory Power Down 0 = Functional 5 = PowerDown
14:12	Reserved	RSVD 0x0	Reserved
11:9	GE0 Mem Power Down	RW 0x0	GE0 Memory Power Down 0 = Functional 5 = PowerDown
8:6	GE1 Mem Power Down	RW 0x0	GE1 Memory Power Down 0 = Functional 5 = PowerDown
5:3	GE2 Mem Power Down	RW 0x0	GE2 Memory Power Down 0 = Functional 5 = PowerDown
2:0	GE3 Mem Power Down	RW 0x0	GE3 Memory Power Down 0 = Functional 5 = PowerDown

Table 1459:Power Management Clock Gating Control Register
Offset: 0x00018220

Bit	Field	Type/InitVal	Description
Core clock gating			
31	Reserved_31	RW 0x1	Reserved. Write only 0x0.
30	Sata1 Core Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
29	Sata1 Link Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
28	XOR1 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
27	PCIe30 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
26	PCIe20 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
25	TDM Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
24	Reserved	RSVD 0x1	Reserved
23	Crypto Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
22	XOR0 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
21	IDMA Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
20	USB2 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
19	USB1 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
18	USB0 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable

Table 1459:Power Management Clock Gating Control Register (Continued)

Offset: 0x00018220

Bit	Field	Type/InitVal	Description
17	SDIO Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
15	Sata0 Core Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
14	Sata0 Link Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
13	BP Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
12	PCIe13 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
11	PCIe12 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
10	PCIe11 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
9	PCIe10 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
8	PCIe03 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
7	PCIe02 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
6	PCIe01 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
5	PCIe00 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
4	GE0 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
3	GE1 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable

Table 1459:Power Management Clock Gating Control Register (Continued)

Offset: 0x00018220

Bit	Field	Type/InitVal	Description
2	GE2 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
1	GE3 Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable
0	Audio Clock Gate	RW 0x1	0 = Clock Disable 1 = Clock Enable

A.26.5 Thermal Manager

Table 1460:Thermal Manager Control and Status Register

Offset: 0x000184C4

Bit	Field	Type/InitVal	Description
31:28	Reserved	RSVD 0x0	Reserved
27:19	TMOverHeatThr	RW 0x0	Thermal Over Heat Threshold To calculate the current temperature, user the formula Specifies the thermal overheat boundary. TM OverHeatThr = 315.3 - (1.3825*temperature theshold) **NOTE:** The value for the temperature threshold is user define
18:10	TMCoolThr	RW 0x0	Thermal Cooling Threshold Specifies the thermal cooling boundary. TM OverHeatThr = 315.3 - (1.3825*temperature theshold) **NOTE:** The value for the temperature threshold is user define
9:1	ThermTempOut	RW 0x0	Thermal Manager Current Temperature To calculate the current temperature, user the formula Tj = (315.3 -ThermTempOut)/1.3825
0	TMDis	RW 0x1	Thermal Manager Disable Disables the Thermal Manager engine. 0 = Enable: Thermal Manager engine is enabled. 1 = Disable: Thermal Manager engine is disabled.

Table 1461:Thermal Manager Cooling Delay Register
Offset: 0x000184C8

Bit	Field	Type/InitVal	Description
31:0	TMCoolingAlarmDelay	RW 0x700	Thermal Cooling Alarm Delay Specifies the time duration that the temperature is below the cooling threshold, until the appropriate interrupt assertion. The timer value in this field is measured in clock-cycles of the Core clock. Note: The value cannot be 0. If the value is 0, the interrupt is asserted, even if it should not be asserted.

Table 1462:Thermal Manager Overheat Delay Register
Offset: 0x000184CC

Bit	Field	Type/InitVal	Description
31:0	TMOverHitAlarmDelay	RW 0x700	Thermal Overheat Alarm Delay Specifies the time duration that the temperature is above the overheat threshold, until the appropriate interrupt assertion. The timer value in this field is measured in Core clock cycles. Note: The value cannot be 0. If the value is 0, the interrupt is asserted, even if it should not be asserted.

Table 1463: Thermal Sensor Configuration0 Register
Offset: 0x000184D0

Bit	Field	Type/InitVal	Description
31	Reserved	RSVD 0x0	Reserved
30	OTF Calibration	RW 0x0	On-The-Fly calibration mode. Calibration is initiated upon relatively high temperature changes. 0 = No OTF Calibration 1 = OTF Calibration
29:26	Reserved	RSVD 0x9	Reserved
25	Start Cal	RW 0x1	Initiate calibration sequence. Level sensitive signal (unless already in calibration, every change in this signal initiates calibration). Setting the default value to 1 will initiate calibration automatically after reset. Internally synchronized.
24:20	Reserved	RSVD 0x4	Reserved
19:11	Ref Cal Count	RW 0x0f1	Reference value for calibration.

Table 1463: Thermal Sensor Configuration0 Register (Continued)
Offset: 0x000184D0

Bit	Field	Type/InitVal	Description
10:2	Reserved	RSVD 0x14	Reserved
1	Soft Reset	RW 0x0	Software reset – active high.
0	Reserved	RSVD 0x0	Reserved

A.26.6 BootROM

Table 1464:BootROM Routine and Error Code Register
Offset: 0x000182D0

Bit	Field	Type/InitVal	Description
31:28	Reserved	RW 0xf	Reserved Write only 0xF.
27:24	Reserved	RW 0xf	Reserved Write only 0xF.
23:16	Retry_Count	RW 0xFF	The retry count is incremented in the Error Handler routine.
15	Reserved	RW 0x0	Reserved Write only 0x0.
14:13	Reserved	RW 0x0	Reserved Write only 0x0.
12	Reserved	RW 0x0	Reserved Write only 0x0.
11:8	Error Location	RW 0x0	Error Location 0 = Initialization 4 = SATA: Boot from SATA interface 7 = SPI: Boot from SPI interface 9 = NAND Flash: Boot from NAND Flash interface 11 = Exception: Exception handler 12 = Main: Main execution loop 14 = PCI Express: Boot from PCI Express interface

Table 1464:BootROM Routine and Error Code Register (Continued)

Offset: 0x000182D0

Bit	Field	Type/InitVal	Description
7:0	Error Code	RW 0x0	Error Code: 0x00 = No Error 0x11 = Invalid header ID 0x12 = Invalid header checksum 0x13 = Invalid image checksum 0x15 = Invalid extended header checksum 0x17 = SATA device is busy. 0x18 = SATA device link error 0x19 = SATA DMA transfer error 0x1A = SATA PIO transfer error 0x1C = Invalid header size 0x1D = Unknown boot device 0x1E = Unsupported reset strap option 0x21 = Image size not aligned to 32 bits 0x22 = Source address not aligned to 32 bits 0x23 = Destination address not Alligned to 32 bits 0x30 = Failed to read from NAND flash 0x31 = Invalid bootROM checksum 0x32 = NAND flash image not aligned to 512 bytes 0x33 = NAND flash ECC error 0x34 = NAND flash timeout 0x35 = NAND flash excessive bad blocks 0x36 = NAND flash bad block encountered 0x37 = NAND flash unsupported ECC type 0x51 = Invalid security header 0x52 = Invalid RSA pubic key 0x53 = Invalid RSA public key FMT 0x54 = Invalid RSA header signature 0x55 = Invalid RSA image signature 0x56 = Invalid box ID 0x57 = Invalid flash ID 0x61 = Invalid binary header 0x62 = Invalid image length 0x63 = Invalid header length 0x1B = I2C timeout (Reserved) 0x40 = I2C read error (Reserved) 0x41 = I2C write error (Reserved) 0x42 = I2C start error (Reserved) 0x43 = I2C stop error (Reserved) 0x44 = I2C no header found (Reserved) 0x58 = RSA library error (Reserved)

A.26.7 Interrupt

Table 1465:Reserved Interrupt Cause Register

Offset: 0x000182A0

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	PCIeMacro1AccessToUnusedPorts	RW0C 0x0	PCIe Macro1Access to Unused Ports
0	PCIeMacro0AccessToUnusedPorts	RW0C 0x0	PCIe Macro0Access to Unused Ports

Table 1466:Reserved Interrupt Mask Register

Offset: 0x000182A4

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	PCIeMacro1UnusedPortAccessMask	RW 0x0	PCIe Macro1 Unused Port Access Mask
0	PCIeMacro0UnusedPortAccessMask	RW 0x0	PCIe Macro0 Unused Port Access Mask

Table 1467:Reserved Error Interrupt Cause Register

Offset: 0x000182A8

Bit	Field	Type/InitVal	Description
31:2	Reserved	RSVD 0x0	Reserved
1	PCIeMacro1AccessToUnusedPorts	RW0C 0x0	PCIe Macro1 Access to Unused Ports
0	PCIeMacro0AccessToUnusedPorts	RW0C 0x0	PCIe Macro0 Access to Unused Ports

Table 1468:Reserved Error Interrupt Mask Register

Offset: 0x000182AC

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

A.26.8 Design for Testablity (DFT)

Table 1469:PUP Enable Register

Offset: 0x0001864C

Bit	Field	Type/InitVal	Description
31:6	Reserved	RSVD 0x2003EF0	Reserved
5	Spi PUP Enable	RW SAR	SPI PUP Enable
4	Nand PUP Enable	RW 0x0	NAND PUP Enable
3	Reserved	RSVD 0x0	Reserved
1	GPort1 PUP Enable	RW 0x0	GPort1 PUP Enable
0	GPort0 PUP Enable	RW 0x0	GPort0 PUP Enable

Copyright © 2014 Marvell

A.27 Clock Complex Registers

The following table provides a summarized list of all registers that belong to the Clock Complex, including the names, their type, offset, and a reference to the corresponding table and page for a detailed description of each element and its fields.

Table 1470:Summary Map Table for the Clock Complex Registers

Register Name	Offset	Table and Page
CPU Divider Clock Control0 Register	0x00018700	Table 1471, p. 1541
CPU Divider Clock Control2 Ratio Full0 Register	0x00018708	Table 1472, p. 1542
CPU Divider Clock Control3_Ratio_Full1 Register	0x0001870C	Table 1473, p. 1543
CPU PLL Control1 Register	0x00018720	Table 1474, p. 1543
DCO Divider Clock Control Register	0x00018730	Table 1475, p. 1544
Core Divider Clock Control Register	0x00018740	Table 1476, p. 1544
Core Divider Clock Ratio Full0 Register	0x00018748	Table 1477, p. 1545
TDM PLL Control Register	0x00018770	Table 1478, p. 1546
DDRPHY APLL Control Register	0x00018780	Table 1479, p. 1546
DCO Control Register	0x00018794	Table 1480, p. 1546
SPCR and DCO Status Register	0x0001879C	Table 1481, p. 1547

Table 1471:CPU Divider Clock Control0 Register

Offset: 0x00018700

Bit	Field	Type/InitVal	Description
31:25	Reserved	RSVD 0x0	Reserved
24	CpuClkDivReloadRatio	RW 0x0	Central request to switch ratios. There is no need to assert at “start of day”. It is implied by exiting PLL reset. It is needed for all later requests for ratio modifications. It must be asserted until it is acknowledge by re-assertion of ClkDiv_Ratio_Stable, indicating that the clocks are operating.

Table 1471:CPU Divider Clock Control0 Register (Continued)

Offset: 0x00018700

Bit	Field	Type/InitVal	Description
23:16	CpuClkDivReloadSmooth	RW 0x0	Reload Ratio trigger qualifier. Per clock tree. Request to modify new ratio, at the next convenient point – where effecting clocks that are not modified will remain intact and all associated cross-masks will transit smoothly. If a modified clock effects others (NBCLK on CPU's, or HCLK on DDR and NB), the modification request will infer a restart of the affected clocks as well. For SMP, it is recommended to use the smooth request below. FieldBit-0 -> H-Domain. FieldBit-1 -> DDR-Domain. FieldBit-2 -> NB-Domain. FieldBit-3 -> ATCLK-Domain. FieldBit-4 -> PCLK0-Domain. FieldBit-5 -> PCLK1-Domain. FieldBit-6 -> PCLK2-Domain. FieldBit-7 -> PCLK3-Domain.
15:8	CpuClkDivResetMask	RW 0x0	Per clock tree reset masking – in case a repeated reset is not wanted, but may be asserted due to PLL re-stabilization or other events. These masks must be default set to zero for enabling initial reset. Later they can be selectively asserted as needed. FieldBit-0 -> H-Domain. FieldBit-1 -> DDR-Domain. FieldBit-2 -> NB-Domain. FieldBit-3 -> ATCLK-Domain. FieldBit-4 -> PCLK0-Domain. FieldBit-5 -> PCLK1-Domain. FieldBit-6 -> PCLK2-Domain. FieldBit-7 -> PCLK3-Domain.
7:0	Reserved	RSVD 0xFF	Reserved

Table 1472:CPU Divider Clock Control2 Ratio Full0 Register

Offset: 0x00018708

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:24	cpu_at_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to at-clk (for debug).
23:22	Reserved	RSVD 0x0	Reserved
21:16	cpu_nb_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to nb-clk.

Table 1472:CPU Divider Clock Control2 Ratio Full0 Register (Continued)

Offset: 0x00018708

Bit	Field	Type/InitVal	Description
15:14	Reserved	RSVD 0x0	Reserved
13:8	cpu_ddr_clk_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to ddr_clk.
7:6	Reserved	RSVD 0x0	Reserved
5:0	cpu_h_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to hclk.

Table 1473:CPU Divider Clock Control3_Ratio_Full1 Register

Offset: 0x0001870C

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:24	cpu_cpu3_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to cpu3-clk.
23:22	Reserved	RSVD 0x0	Reserved
21:16	cpu_cpu2_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to cpu2-clk.
15:14	Reserved	RSVD 0x0	Reserved
13:8	cpu_cpu1_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to cpu1-clk.
7:6	Reserved	RSVD 0x0	Reserved
5:0	cpu_cpu0_clkdiv_ratio_full	RW SAR	Full Integer Ratio from PLL-Out frequency to cpu0-clk.

Table 1474:CPU PLL Control1 Register

Offset: 0x00018720

Bit	Field	Type/InitVal	Description
cpu_pll_control1			
31:0	Reserved	RSVD 0x0	Reserved

Table 1475:DCO Divider Clock Control Register

Offset: 0x00018730

Bit	Field	Type/InitVal	Description
31:26	DCOClkDivRatio	RW 0x6	DCO Clock Divider Ratio This field provides the divider ration for the clock from the DCO to the TDM unit.
25	DCOClkDivClockEn	RW 0x1	DCO Divider Clock Enable 0 = Disabled: The clock from DCO clock divider is gated. 1 = Enabled: The clock from DCO clock divider is running.
24:0	Reserved	RSVD 0x0	Reserved

Table 1476:Core Divider Clock Control Register

Offset: 0x00018740

Bit	Field	Type/InitVal	Description
31:24	core_clkdiv_clock_enable	RW 0xff	External enable per clock tree, in case the PMU or other request an instant override to shut off clocks. Must be always asserted otherwise. (may be used in case of shutting off the PLL). FieldBit-0 -> NA-Domain. FieldBit-1 -> NAND-Domain. FieldBit-2 -> clk2-Domain. FieldBit-3 -> Encip-Domain. FieldBit-4 -> SDIO-Domain. FieldBit-5 -> Gbe-Domain. FieldBit-6 -> SETM-Domain. FieldBit-7 -> Usb_phy-Domain.
23:9	Reserved	RSVD 0x0	Reserved
8	core_clkdiv_reload_ratio	RW 0x0	Central request to switch ratios. No need to assert at “start of day” – it is implied by getting out of pll_rst_. Needed for all later requests for ratio modifications. Must be asserted till acknowledge by re-assertion of clkdiv_ratio_stable indicating that clocks are back to life.

Table 1476:Core Divider Clock Control Register (Continued)

Offset: 0x00018740

Bit	Field	Type/InitVal	Description
7:0	core_clkdiv_reload_force	RW 0x0	Reload Ratio trigger qualifier. Per clock tree. Request to force new ratio regardless of relation to other clocks. Applicable to HCLK/ATCLK, or any generic PLL Wrapper usage. In case of SMP, it is recommended to use the Smooth request below. FieldBit-0 -> NA-Domain. FieldBit-1 -> NAND-Domain. FieldBit-2 -> clk2-Domain. FieldBit-3 -> Encip-Domain. FieldBit-4 -> SDIO-Domain. FieldBit-5 -> Gbe-Domain. FieldBit-6 -> SETM-Domain. FieldBit-7 -> Usb_phy-Domain.

Table 1477:Core Divider Clock Ratio Full0 Register

Offset: 0x00018748

Bit	Field	Type/InitVal	Description
31:30	Reserved	RSVD 0x0	Reserved
29:24	EncripClkDivRatioFull	RW SAR	Full Integer Ratio from PLL-Out frequency to encryption engine.
23:22	Reserved	RSVD 0x0	Reserved
21:16	Clk2ClkDivRatioFull	RW SAR	Full Integer Ratio from PLL-Out frequency
15:14	Reserved	RSVD 0x0	Reserved
13:8	NandEccClkDivRatioFull	RW 0x5	Full Integer Ratio from PLL-Out frequency to NAND-ECC block. Ecc_clk = 2000 MHz / NandEccClkDivRatioFull. ND_CLK = Ecc_clk / 2 The default value of the Ecc_clk is 400 MHz. The default value of the ND_CLK is 200 MHz.
7:6	Reserved	RSVD 0x0	Reserved
5:0	CoreClkDivRatioFull	RO SAR	Full Integer Ratio from PLL-Out frequency to core clock

Table 1478:TDM PLL Control Register

Offset: 0x00018770

Bit	Field	Type/InitVal	Description
31:20	Reserved	RSVD 0x0	Reserved
19	Tdm Apll Cfg Vpostst Oe	RW 0x0	When this signal is high, apll_tst_vpos is valid. 0 = Not Valid: apll_tst_vpos_not_valid 1 = Valid: apll_tst_vpos_valid
18:17	Tdm Clk Src Sel	RW 0x3	Selects TDM clock source: 0 = Pll Mode: 2.048 MHz / 4.098 MHz / 8.192 MHz 1 = Reserved 2 = ExtRef Mode: Use the TDM_REF_PCLK reference clock. 3 = DCO Mode: Use 8.192 MHz clock from DCO
16	TDM Clk En	RW 0x1	TDM Clock Enable 0 = Clock Disable 1 = Clock Enable
15:13	Reserved	RSVD 0x0	Reserved
12:0	Tdm Full Div	RW 0xb1	The value of this field divides the dedicate TDM-APLL-clk. In default the TDM-APLL-clk = 1450 MHz and by that the tdm-clk = 1450/177 = 8.192 MHz.

Table 1479:DDRPHY APLL Control Register

Offset: 0x00018780

Bit	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x2000000	Reserved

Table 1480:DCO Control Register

Offset: 0x00018794

Bit	Field	Type/InitVal	Description
This register controls the DCO base frequency and offset. After this register is written, before it can be re-written, it must be read. After the register is read, a delay of 400 ns should be inserted and then a new value can be written.			
31:14	Reserved	RSVD 0x0	Reserved

Table 1480:DCO Control Register (Continued)

Offset: 0x00018794

Bit	Field	Type/InitVal	Description
13:2	dco_ctrloffset	RW 0x800	Offset control in which each step equals to 1.27 ppm. dco_mclk = base frequency*(1+1.27*(dco_ctrloffset-2048)/1e6) Offset values are: 0x020: -2016*1.27 ppm . 0x800: 0 ppm . 0xFD0: +2000*1.27 ppm **NOTE:** Values above 0xFD0 or below 0x020 are reserved and should not be used.
1:0	dco_ctrlfs	RW 0x2	Control FS selects the base frequency of the DCO. 0 = 11.2896 MHz 1 = 12.288 MHz 2 = 24.576 MHz 3 = Reserved

Table 1481:SPCR and DCO Status Register

Offset: 0x0001879C

Bit	Field	Type/InitVal	Description
31:17	Reserved	RSVD 0x0	Reserved
16	dco_lock	RO 0x0	DCO Lock Indication 0 = Unlocked DCO 1 = Locked DCO
15:3	Reserved	RSVD 0x0	Reserved
2:0	spcr_ctrlfs	RO 0x0	Defines the output base frequency FS. 0 = 11.2896 MHz 1 = 12.288 MHz 2 = 24.576 MHz 3 = Lower than 11.2896 MHz 4 = Higher than 24.576 MHz 5 = Reserved 6 = Reserved 7 = Other frequency

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.988.8279

www.marvell.com

Marvell. Moving Forward Faster