88F5182

Feroceon® Storage Networking SoC

User Manual

Doc. No. MV-S103345-01, Rev. C
April 29, 2008, Preliminary

Document Conventions

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

Document Status

Doc Status: Preliminary	Technical Publication: 0.x

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

PRODUCT OVERVIEW

The Marvell® 88F5182 device is a high-performance, highly integrated, Storage Networking System Engine. The 88F5182 integrates the Marvell Feroceon® CPU core, compliant with the ARMv5TE architecture.

FEATURES

The 88F5182 device has the following features:

- **High-performance integrated controller**
  - High-performance Feroceon CPU core with integrated 32/32 KB I/D L1 cache, running at up to 500 MHz
  - High bandwidth dual-port memory controller (16-/32-bit DDR1/DDR2 SDRAM)
  - Single PCI Express (x1) port with integrated PHY
  - Single 32-bit PCI2.2 66 MHz port
  - Two SATA 2.0 ports with integrated 3 Gbps SATA II PHYs
  - Single Gigabit Ethernet MAC (10/100/1000 Mbps)
  - Two USB 2.0 ports with integrated PHY
  - Security Cryptographic Engine
  - Two-Wire Serial Interface (TWSI)
  - Two UART ports
  - 16-bit device bus with up to four chip selects
  - NAND Flash support
  - Integrated DMA engine (four channels)
  - XOR engine for RAID applications
  - 26 multi-purpose pins
  - Interrupt controller
  - Timers
- **Marvell® Feroceon® CPU core**
  - 500 MHz with DDR1/DDR2 at 166 MHz
  - 400 MHz with DDR2 at 200 MHz
  - 32-bit and 16-bit RISC architecture
  - Compliant with v5TE architecture as published in the ARM Architect Reference Manual, Second Edition
  - Includes MMU to support virtual memory features
  - MPU can be used instead when not using MMU
  - 32-KB I-Cache and 32-KB D-Cache
  - 64-bit internal data bus
  - Out-of-order execution for increased performance
  - In-order retire via a Reordering Buffer (ROB)
  - Branch Prediction Unit
  - Supports JTAG/ARM Multi-ICE
  - Supports both Big and Little Endian modes
- **DDR1/DDR2 SDRAM controller**
  - DDR SDRAM with a clock ratio of 1:1, 1:2, 1:3, or 1:4 between the DDR SDRAM and the Feroceon CPU core, respectively
  - 16-/32-bit interface
  - DDR1 at up to 333 MHz
  - DDR2 at up to 400 MHz
  - Supports up to two dual-sided DIMMs
  - Supports DDR components of x8 and x16
  - Dual channel memory controller
  - Reduced CPU to DDR SDRAM latency
  - SSTL 2.5V I/Os in DDR1, 1.8V I/Os in DDR2
  - Supports four DDR SDRAM banks (CSs)
  - DDR1 supports device densities of 128, 256, 512 Mbits
  - DDR2 supports device densities of 256, 512 Mbits
  - Up to 1 GB (32-bit interface) and 0.5 GB (16-bit interface) total memory space
  - Supports DDR SDRAM bank interleaving between all DDR SDRAM banks (both the physical banks, and the four internal banks of the DDR SDRAM devices)
  - Supports up to 16 open pages (page per bank)
  - Supports configurable DDR SDRAM timing parameters
  - Supports up to 32-byte burst per single DDR SDRAM access
  - Single ended DQS in DDR2
  - DDR1/DDR2 pad auto calibration
  - Support DDR2 On Die Termination (ODT)
- **PCI Express interface (x1)**
  - PCI Express Base 1.0a compatible
  - Integrated low power SERDES PHY, based on proven Marvell SERDES technology
  - Root Complex port
  - Can be configured also as an Endpoint port

- x1 link width
- 2.5 GHz/s signalling
- Lane polarity reversal support
- Maximum payload size of 128 bytes
- Single Virtual Channel (VC-0)
- Replay buffer support
- Extended PCI Express configuration space
- Advanced Error Reporting (AER) support
- Power management: L0s and software L1 support
- Interrupt emulation message support
- Error message support

■ **PCI Express master specific features**
- Single outstanding read transaction
- Maximum read request of up to 128 bytes
- Maximum write request of up to 128 bytes
- Up to four outstanding read transactions in Endpoint mode

■ **PCI Express target specific features**
- Supports up to eight read request transactions
- Maximum read request size of 4 KB
- Maximum write request of 128 bytes
- Supports PCI Express access to all of the device's internal registers

■ **32-bit PCI interface**
- 66 MHz PCI 2.2 compliant interface
- 3.3V I/Os, 5V tolerant
- Supports 64-bit addressing via DAC transactions
- Configurable PCI arbiter for up to six masters

■ **PCI master specific features**
- Supports all PCI cycles
- Host to PCI bridge—translates CPU cycles to PCI memory, I/O, or configuration cycles
- Supports DMA bursts between PCI and memory
- Supports transaction combining to unlimited PCI burst

■ **PCI target specific features**
- Supports all PCI cycles
- Supports programmable aggressive read prefetch
- Supports unlimited burst write with zero wait states
- Supports up to four delayed reads
- Supports PCI access to all of the device's internal registers
- PCI address remapping to local memory

■ **PICMG Compact PCI Hot-Swap ready**

■ **PCI "Plug and Play" support**
- Plug and Play compatible configuration registers
- PCI configuration registers that are accessible from both the Feroceon CPU core and PCI
- Vital Product Data (VPD) support
- PCI Power Management (PMG) support
- Message Signal Interrupts (MSI) support

■ **SATA II interface (2 ports)**
- Integrates Marvell 3 Gbps (Gen2i) SATA PHY
- 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 OOB sequence to select the active host port
- Supports device 48-bit addressing
- Supports ATA Tag Command Queuing

■ **SATA II host controller**
- Enhanced-DMA [EDMA] per SATA port
  - Automatic command execution without host intervention
  - Command queuing support, for 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
- Target mode operation—Two 88F5182 devices can be attached through SATA ports, enabling data communication between different 88F5182 devices.
- Advanced drive diagnostics via the ATA SMART command

■ **Integrated single GbE (10/100/1000) MAC**
- Supports 10/100/1000 Mbps
- MII, GMII, or RGMII Interface
- Proprietary 200 Mbps Marvell MII (MMII) interface
- Dedicated DMA for data movement between memory and port
- Priority queuing on receive based on DA, VLAN Tag, and IP TOS
- Layer 2/3/4 frame encapsulation detection
- TCP/IP checksum on receive and transmit
- DA address filtering

■ **USB 2.0 ports (2 ports)**
- Each port can serve as a peripheral or host
- USB 2.0 compliant
- Integrated USB 2.0 PHY
- EHCI compatible as a host
- As a host, supports direct connection to all peripheral types (LS, FS, HS)

- As a peripheral, connects to all host types (HS, FS) and hubs
- Up to 4 independent endpoints supporting control, interrupt, bulk, and isochronous data transfers
- Dedicated DMA for data movement between memory and port

- **Two-Wire Serial Interface (TWSI)**
  - Master/slave operation
  - Serial ROM initialization
- **Two UART interfaces**
  - 16550 UART compatible
  - Two pins for transmit and receive operations
  - Two pins for modem control functions
- **Device bus controller**
  - 8-/16-bit width
  - 166 MHz clock frequency
  - 3.3V I/Os
  - Supports many types of standard memory devices such as FLASH and ROM
  - Four chip selects with programmable timing
  - Optional external wait-state support
  - Boot ROM support
- **NAND Flash support**
  - Glueless interface to CE don't care NAND Flash through the device bus interface
  - Glueless interface to CE care NAND Flash through the device bus and MPP interfaces
  - Boot from NAND Flash when the first block, placed on 00h block address, is guaranteed to be a valid block with no errors
  - Supports read bursts of up to 128 bytes
- **Four channel Independent DMA controller**
  - Chaining via linked-lists of descriptors
  - Moves data from any interface to any other interface
  - Supports increment or hold on both source and destination address
- **Two XOR DMAs**
  - Useful for RAID application
  - Supports XOR operation on up to eight source blocks
  - Supports CRC-32 calculation
- **Cryptographic engine**
  - Hardware implementation on encryption and Authentication engines to boost packet processing speed
  - Dedicated DMA to feed the hardware engines with data from internal SRAM memory
  - Implements AES, DES and 3DES encryption algorithms
  - Implements SHA1 and MD5 authentication algorithms
- **26 multi-purpose pins dedicated for peripheral functions and general purpose I/O**
  - Each pin can be configured independently
  - GPIO inputs can be used to register interrupts from external devices and to generate maskable interrupts
- **Interrupt controller**
  - Maskable interrupts to Feroceon CPU core
  - In endpoint mode, maskable interrupts to the PCI/PCI Express interfaces
- **Timers**
  - Two general purpose 32-bit timer/counters
  - One 32-bit Watchdog timer
- **Internal Architecture**
  - AHB bus for high-performance, low latency Feroceon CPU core to DDR SDRAM connectivity
  - Advanced Mbus architecture with any to any concurrent I/O connectivity
  - Dual port DDR SDRAM controller connectivity to both AHB and Mbus
- **Bootable from**
  - Device interface
  - PCI interface
  - DDR interface
- **HSBGA, 23x23 mm, 388L package, 1 mm ball pitch**

Table of Contents

List of Tables

List of Figures

Preface

About This Document

This document, provides a features list, product overview and interface description for the 88F5182 Feroceon® Storage Networking SoC. It also provides detailed information and definitions of the device register set.

In this document the 88F5182 is also referred to as “the device”.

Related Documents

- *88F5182 Feroceon® Storage Networking SoC Datasheet*, Doc. No. MV-S103345-00
- *Orion SoC Hardware Design Guide*, Doc. No. MV-S103315-00[1]
- *88F5182 Feroceon Storage Networking SoC Functional Errata, Guidelines, and Restrictions*, Doc. No. MV-S500802-00
- *Feroceon® 88FR531-vd CPU Datasheet*, Doc. No. MV-S104989-00[1]
- *ARM Architecture Reference Manual*, Second Edition
- *AMBA™ Specification*, Rev 2.0
- *PCI Local Bus Specification*, Revision 2.2
- *PCI-X Addendum to the PCI Local Bus Specification*, Revision 1.0b
- *PCI Express Base Specification*, Revision 1.1
- Serial-ATA II Phase 1.0 Specification (Extension to SATA I Specification)
- *Universal Serial Bus Specification, Revision 2.0*, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips http://www.usb.org
- *Enhanced Host Controller Interface Specification for Universal Serial Bus*, Revision 0.95, November 2000, Intel Corporation http://www.intel.com
- *USB-HS High-Speed Controller Core* reference[1]
- *RFC 1321* (The MD5 Message-Digest Algorithm)
- *FIPS 180-1* (Secure Hash Standard)
- *FIPS 46-2* (Data Encryption Standard)
- *FIPS 81* (DES Modes of Operation)
- *RFC 2104* (HMAC: Keyed-Hashing for Message Authentication).
- *RFC 2405* – The ESP DES-CBC Cipher Algorithm With Explicit IV
- *RFC 1851* – The ESP Triple DES Transform
- *FIPS draft* - Advanced Encryption Standard (Rijndeal)
- AN-123 Power Sequencing for Marvell Devices, Rev. A (Doc. No. MV-S300427-00)[1]

See the Marvell Extranet website for the latest product documentation.

1. This document is a Marvell proprietary confidential document requiring an NDA and can be downloaded from the Marvell Extranet.

Document Conventions

Document Conventions

This document has the following name and usage conventions:

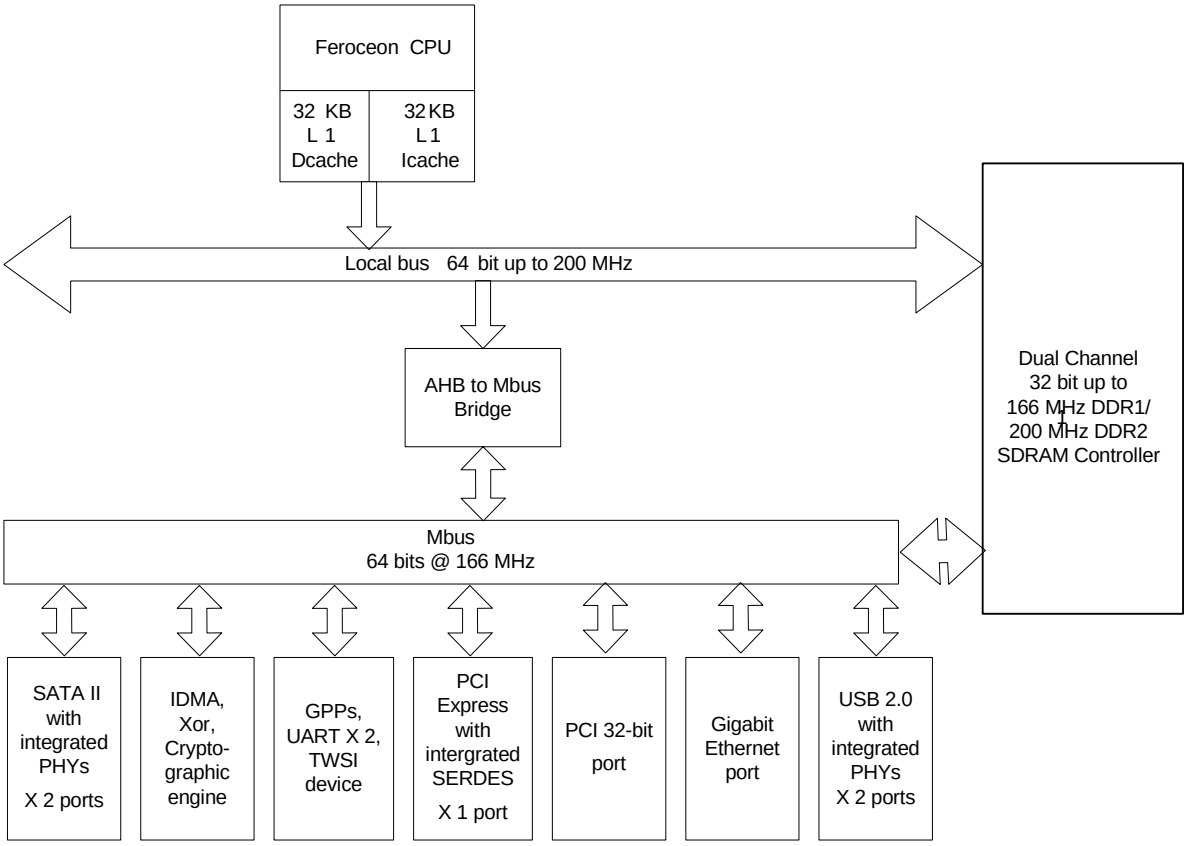
Signal Range	A signal name followed by a range enclosed in brackets represents a range of logically related signals. The first number in the range indicates the most significant bit (MSb) and the last number indicates the least significant bit (LSb). Example: DB_AD[31:0]
Active Low Signals n	A ***n*** 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 enclosed in angle brackets: Example: <SAddrOvr> Register field bits are enclosed in brackets. Example: Bits[31:0] 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.
Abbreviations	Gb: gigabit GB: gigabyte Kb: kilobit KB: kilobyte Mb: megabit MB: megabyte

1 Overview

The Marvell® 88F5182 is a high-performance, highly integrated, storage networking system engine. It integrates the Marvell Feroceon® CPU core, which is compliant with the ARMv5TE architecture.

Figure 1 is a block diagram of the 88F5182 interfaces.

Figure 1: 88F5182 Interface Block Diagram



The 88F5182 incorporates the following:

Feroceon CPU Core	Compliant with V5TE Architecture, as published in the *ARM Architect Reference Manual*, Second Edition. The Feroceon CPU core incorporates an integrated 32/32 KB I/D L1 cache.
DDR SDRAM	Includes a 16-/32-bit DDR1/DDR2 SDRAM controller.
PCI Express	Includes a single PCI Express (x1) host port, with an integrated low power SERDES. The PCI Express port can also be configured as an Endpoint port.
PCI	32-bit conventional PCI interface, operating at a maximum frequency of 66 MHz.
SATA II	Two SATA II ports, fully compliant with SATA II Phase 1.0 specification (Extension to SATA I specification), supporting: • SATA II Native command queuing • Backwards compatibility to SATA I 1.5-Gbps speed and devices • In addition to full support of SATA II Phase 1.0 specification (Extension to SATA I specification), the 88F5182 supports the following advanced SATA II Phase 2.0 specification features: • 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 OOB sequence to select the active host of the SATA II Port Selector.
Gigabit Ethernet	Consists of a single 10/100/1000-Mbps full-duplex Gigabit Ethernet (GbE) port. It can be configured to a 10/100-Mbps MII interface, or a 10/100/1000-Mbps RGMII/GMII interface. When configured as an MII interface, the Gigabit Ethernet MAC can run not only at 10/100 Mbps, but also at 200 Mbps. This is useful for higher throughput interfacing to the Marvell® Fast Ethernet switches.
USB2	Two USB 2.0 high-speed ports each with an embedded PHY. They can be configured to either host ports or peripheral ports.
Cryptographic Engine and Security Accelerator	Supports data encryption and authentication. It also contains a dedicated DMA to feed data from the local 8-KB SRAM into the arithmetic hardware.
Two-Wire Serial Interface (TWSI)	Two-Wire Serial Interface (TWSI) port.
UART	The UART Interface consists of two UART ports.
Device Bus	Includes a 32-bit Device interface.
IDMA Engines	Four IDMA engines, each with the ability to transfer data between any interfaces.
XOR Engine	Two additional XOR DMA engines, useful for Redundant Array of Independent Disks (RAID) applications. 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 writes the result to a destination. The sources and destination can reside in any of the 88F5182 interfaces.
General Purpose I/O Port	26-bit general purpose I/Os

Interrupt Controller	Handles interrupts from all of the various sources and forwards them to the Feroceon CPU core. When working in Endpoint mode, the interrupts can also be forwarded to the Endpoint PCI-Express or the PCI/PCI_X interface.
Timers	Two general purpose 32-bit-wide timers and a single 32-bit-wide watchdog timer.
Internal Architecture	The 88F5182 internal architecture is optimized for high-performance applications. It includes an AHB bus for CPU to DDR SDRAM connectivity and a proprietary full-mesh Mbus architecture for I/O connectivity.

2 Address Map

The 88F5182 has a fully programmable address map. There is a separate address map for each of the device master interfaces. Each interface includes programmable address windows that allow it to access any of the 88F5182 resources:

- Feroceon® CPU core address map
- PCI Express address map
- PCI address map
- SATA address map
- Ethernet Controller address map
- USB address map
- IDMAs address map
- XOR address map

Note Although each master has independent address windows, when a resource is used by multiple masters, all masters must use the same address map for this resource. This means that all masters use the identical address window for each resource.

2.1 Feroceon® CPU Core Address Map

The Feroceon CPU core interface address map consists of eight programmable address windows for the different interfaces and additional four dedicated windows for the DDR interface. See Appendix A.4.1, CPU Address Map Registers, on page 243 and Appendix A.5.1, DDR SDRAM Controller Address Decode Registers, on page 261.

For default address map see Table 1, “88F5182 Default Address Map,” on page 22.

2.2 PCI Express Address Map

The PCI Express interface address map consists of three BARs that map the device’s address space. One BAR is dedicated for the device’s internal registers while the other two are further sub-decoded by six programmable address windows to the different interfaces of the device. See Appendix A.6.1, PCI Express BAR Control Registers, on page 279.

For the default address map, see Table 1, with following exceptions.

- By default, access from the PCI Express interface to PCI interface is disabled.
- By default, access from the PCI Express to Device CS0 and Device CS1 is disabled.

2.3 PCI Address Map

The PCI interface address map consists of 12 BARS address windows for the different interfaces.

For the default address map, see Table 1, with following exceptions.

- By default, access from the PCI interface to the PCI Express interface is disabled.
- By default, access from the PCI interface to Device CS0 and Device CS1 is disabled.
- By default, I/O access from the PCI interface to the device’s internal registers is disabled.

2.4 SATA Address Map

The SATAHC interface address map consists of four programmable address windows for the different interfaces. See Section A.8.4, SATAHC Arbiter Registers, on page 362. By default the SATAHC address map is enabled and addressed to the DRAM as specified in Table 1, 88F5182 Default Address Map, on page 22.

2.5 Gigabit Ethernet Address Map

The Gigabit Ethernet interface address map consists of six programmable address windows for the different interfaces. See Section A.9.1, Gigabit Ethernet Unit Global Registers, on page 410. By default the Gigabit Ethernet MAC address map is disabled.

2.6 USB0 Address Map

The USB0 interface address map consists of four programmable address windows for the different interfaces. See Section A.10.3, USB 2.0 Bridge Address Decoding Registers, on page 446. By default the USB0 address map is disabled.

2.7 USB1 Address Map

The USB1 interface address map consists of four programmable address windows for the different interfaces. See Section A.10.3, USB 2.0 Bridge Address Decoding Registers, on page 446. By default the USB1 address map is disabled.

2.8 IDMA Address Map

The IDMA interface address map consists of eight programmable address windows for the different interfaces. See Section A.15.2, IDMA Address Decoding Registers, on page 490. By default the IDMA address map is disabled.

2.9 XOR Address Map

The XOR interface address map consists of eight programmable address windows for the different interfaces. See Section A.16.4, XOR Engine Address Decoding Registers, on page 505. By default the XOR address map is disabled.

Note

Windows base addresses of the 88F5182 must be aligned to their size (for example, a 128 KB address window must be aligned to 128 KB).

2.10 Default Address Map

Table 1: 88F5182 Default Address Map

Target Interface	Target Interface ID[1]	Target Interface Attribute[2]	Address Space Size	Address Range in Hexadecimal
DDR SDRAM CS0	0	0x0E	256 MB	0000.0000–0FFF.FFFF
DDR SDRAM CS1	0	0x0D	256 MB	1000.0000–1FFF.FFFF
DDR SDRAM CS2	0	0x0B	256 MB	2000.0000–2FFF.FFFF
DDR SDRAM CS3	0	0x07	256 MB	3000.0000–3FFF.FFFF
Reserved	-	-	1 GB	4000.0000–7FFF.FFFF
PCI Express Memory	4	0x59	512 MB	8000.0000–9FFF.FFFF
PCI Memory	3	0x59	512 MB	A000.0000–BFFF.FFFF
PCI Express I/O	4	0x51	64 KB	C000.0000–C000.FFFF
Reserved	-	-	-	C001.0000–C7FF.FFFF
PCI I/O	3	0x51	64 KB	C800.0000–C800.FFFF
Security Accelerator Internal SRAM Memory **NOTE:** There is no access to Security Accelerator Internal SRAM Memory from the PCI interface.	9	0x00	64 KB	C801.0000–C801.FFFF **NOTE:** Only 8 KB SRAM is implemented.
Reserved		-	-	C802.0000–CFFF.FFFF
Internal Address Space[3]		-	1 MB	D000.0000–D00F.FFFF
Reserved	-	-	-	D010.0000–DFFF.FFFF
Device CS0	1	0x1E	128 MB	E000.0000–E7FF.FFFF
Device CS1	1	0x1D	128 MB	E800.0000–EFFF.FFFF
Device CS2	1	0x1B	128 MB	F000.0000–F7FF.FFFF
Flash Boot CS	1	0x0F	128 MB	F800.0000–FFFF.FFFF

1. Defines field <Target> in the window control registers. See Appendix A.4.1, CPU Address Map Registers, on page 243 and Appendix A.6.4, PCI Express Address Window Control Registers, on page 283.
2. Defines field <Attr> in the window control registers. See Appendix A.4.1, CPU Address Map Registers, on page 243 and Appendix A.6.4, PCI Express Address Window Control Registers, on page 283.
3. For the 88F5182 Internal Address Map, see Table 70 on page 241.

3 Feroceon® CPU Core

The 88F5182 uses a Feroceon® CPU core.

4 DDR SDRAM Controller Interface

The DDR SDRAM (Double Data Rate-Synchronous DRAM) controller supports:

- Both 16-bit and 32-bit DDR SDRAM interfaces
- DDR1 SDRAM at up to 166 MHz and DDR2 SDRAM at up to 200 MHz.
- Up to two dual-sided DIMMs (four physical banks)
- A variety of DDR SDRAM components—x8 and x16 devices, at densities of 128 Mb, 256 Mb, and 512 Mb
- Up to 1 GB (32-bit interface) and 0.5 GB (16-bit interface) total memory space

The DDR SDRAM controller is optimized for maximum DDR SDRAM bus utilization. It supports bank interleaving between DDR SDRAM internal banks and physical banks. It also supports up to 16 open pages.

The DDR SDRAM controller supports DDR2 and ODT (On Die Termination).

4.1 Functional Description

The DDR SDRAM controller receives read and write requests from any of the other interfaces through the 88F5182 Mbus, and the DDR SDRAM controller receives Feroceon® CPU core requests from the direct AHB interface. The DDR SDRAM controller translates these requests to DDR SDRAM transactions.

Note The DDR SDRAM controller splits long Mbus transactions into multiple 32-byte DDR SDRAM transactions. This split enables a Feroceon CPU core request to be served in between the 32-byte DDR SDRAM accesses (The CPU request does not need to wait for the completion of the entire 128-Byte access).

The DDR SDRAM controller contains a transaction queue and read and write buffers. It can absorb up to four read transactions and up to two write transactions of 128 bytes each.

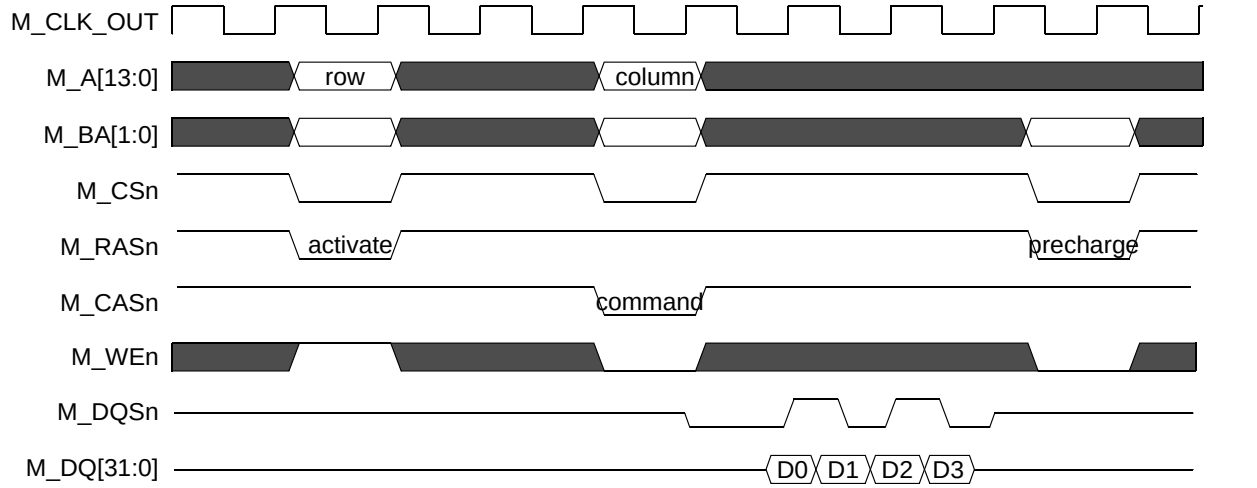
Transactions from the Mbus are pushed into the transaction queue. The DDR SDRAM controller arbitrates between the transaction from the top of the queue and transactions received from the AHB path. The DDR SDRAM controller drives some of the address bits of the selected transaction on M_A[13:0] and M_BA[1:0] during the activate cycle (M_RASn), and it drives the remaining bits during the command cycle (M_CASn).

For a write transaction, write data coming from the requesting unit is placed in the write buffer. The SDRAM write buffer is necessary to compensate on the data rate differences between the received write data rate (single data rate—SDR) and the rate it is driven to the SDRAM (double data rate—DDR).

For a read transaction, after the command cycle (M_CASn), the DDR SDRAM controller samples read data driven by the DDR SDRAM (The sample window depends on the CL parameter.), pushes the data into the read buffer, and drives it back to the requesting unit.

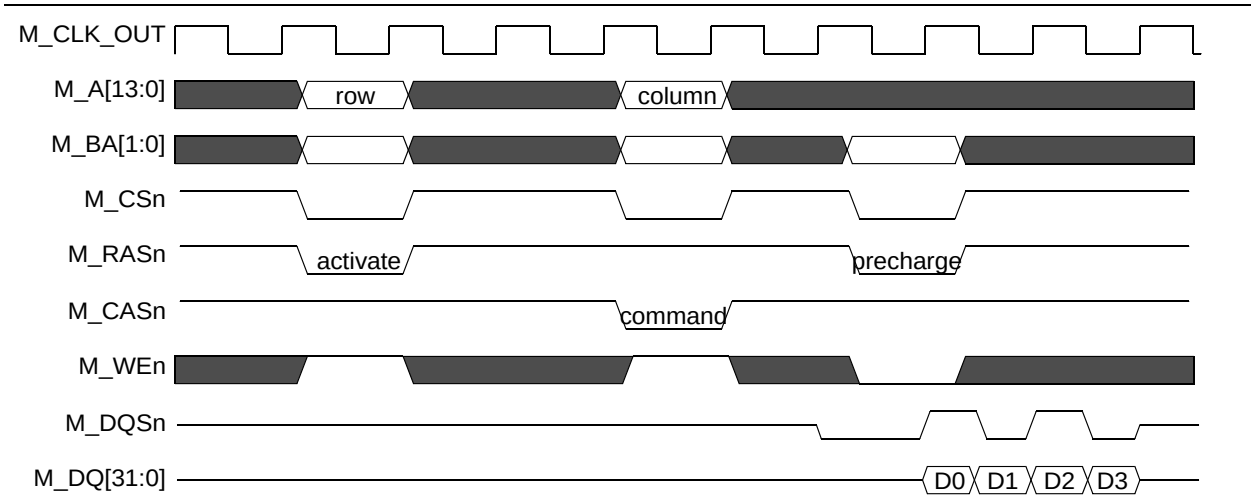
An example of a write transaction is shown in Figure 2. The basic DDR SDRAM controller access to DDR SDRAM consists of an activate cycle (row address), a command (column address), and a precharge at the end of the transaction. Write data is driven with each of the rising and falling edges of the clock, along with DQS. The DDR SDRAM controller also inserts the required preamble and post-amble.

Figure 2: DDR Burst Write Example



An example of a read transaction is shown in Figure 3. The DDR SDRAM controller latches the incoming data with each rise and fall of DQS input.

Figure 3: DDR Burst Read Example (CL = 2)



4.1.1 Address Decoding

The DDR SDRAM controller interfaces the Feroceon CPU core directly via the AHB bus, and has its own address decoding registers. It has an address window defined by the Base and Size registers per each of the DDR SDRAM chip selects.

Every transaction initiated by the Feroceon CPU core on the AHB bus is decoded against these four windows. If there is a hit, the transaction is forwarded to the appropriate DDR SDRAM chip select.

4.1.2 Arbitration and Ordering

Transactions coming from the 88F5182 Mbus are pushed into a transaction queue.

The DDR SDRAM controller arbitrates between the transaction at the top of the queue and transactions coming from AHB bus. Whenever there is a pending Feroceon CPU core transaction, it will get a priority over a pending Mbus transaction. While the DDR SDRAM controller is serving one transaction, the arbiter selects the next transaction to be served from the AHB bus or from the top of the transaction queue. In this manner, the DDR SDRAM controller pipelines transactions to the DDR SDRAM, resulting in maximum DDR SDRAM bus utilization.

The DDR SDRAM controller transaction queue maintains transaction ordering between the source unit over the 88F5182 Mbus and the DDR SDRAM. In addition, when an Feroceon CPU core transaction is received, the DDR SDRAM controller performs a lookup against the pending transactions in the queue. If the newly received address matches one of the addresses already in the queue, the Feroceon CPU core transaction is postponed until the transactions in queue are flushed to DDR SDRAM. This mechanism is required to maintain ordering between Feroceon CPU core and I/O devices.

4.1.3 Lock Transactions

The Feroceon CPU core can generate locked transactions. This is typically used for Read-Modify-Write accesses (such as semaphore update). When DDR SDRAM controller receives a locked transaction from the Feroceon CPU core, it no longer serves any further I/O transactions until CPU lock de-assertion.

Note If the <LockEn> bit[18] in the DDR SDRAM Control Register (Table 124 p. 264) is cleared, The DDR SDRAM controller ignores the lock indication from the Feroceon CPU core.

4.1.4 Error Handling

The only error case that the DDR SDRAM controller is required to handle, is a write access with erroneous data indication from the initiator unit. If the <Perr> bit[18] in the DDR SDRAM Configuration Register is set to 1, the DDR SDRAM controller does not write the erroneous data to DDR SDRAM.

Note Upon a write to the DDR SDRAM Control Register (Table 124 p. 264) file with an erroneous data indication, the data from the write transaction is discarded.

4.1.5 Clock Domains

The 88F5182 Mbus runs at TCLK clock domain. The AHB bus and DDR SDRAM run at the SYSCLK clock domain.

The I/O traffic from Mbus to DDR SDRAM goes through TCLK/SYSCLK synchronizers. However, the AHB and DDR SDRAM share the same clock tree, and do not require any synchronization. This is important for low latency CPU to DDR SDRAM access.

4.2 DDR SDRAM Addressing

The 88F5182 supports 128-, 256-, and 512-Mb DDR1 SDRAM devices as well as 256- and 512-Mb DDR2 devices, in x8 and x16 configurations. The DDR SDRAM devices differ in the usage of M_A[13:0] and M_BA[1:0] lines, as described in Table 2.

Table 2: DDR SDRAM Addressing

SDRAM Type		Internal Bank Address	Row Address	Column Address	Auto Precharge
128 Mb DDR1	16 Mb x 8	M_BA[1:0]	M_A[11:0]	M_A[9:0]	M_A[10]
	8 Mb x 16	M_BA[1:0]	M_A[11:0]	M_A[8:0]	M_A[10]
256 Mb	32 Mb x 8	M_BA[1:0]	M_A[12:0]	M_A[9:0]	M_A[10]
	16 Mb x16	M_BA[1:0]	M_A[12:0]	M_A[8:0]	M_A[10]
512 Mb	64 Mb x 8 DDR1	M_BA[1:0]	M_A[12:0]	M_A[11], M_A[9:0]	M_A[10]
	64 Mb x 8 DDR2	M_BA[1:0]	M_A[13:0]	M_A[9:0]	M_A[10]
	32 Mb x 16	M_BA[1:0]	M_A[12:0]	M_A[9:0]	M_A[10]

The DDR SDRAM controller supports up to four DDR SDRAM physical banks (DDR SDRAM chip selects). The total DDR SDRAM bank address space is determined by the nature of the DDR SDRAM devices. For example, a bank using a build up of four 256 Mb x8 devices (32Mx8) has a 128-MB address space.

The DDR SDRAM controller multiplexes the address bits of the received transaction (from the Mbus or the AHB bus) to the row, column, and internal bank address. It always place the MSB bits of the received address onto the DDR SDRAM BA[1:0] bits as shown in the Table 3 and Table 4.

Table 3: Address Multiplex for 32-bit DDR SDRAM Interface

SDRAM Type		M_BA [1:0]	Row M_A[13:0]	Column M_A[11:0]
128 Mb	16 Mb x 8	25–24	26, 24, 22–11	25, “0”, 23, 10–2
	8 Mb x 16	24–23	26, 24, 22–11	25, “0”, 23, 10–2
256 Mb	32Mb x 8	26–25	26, 24, 22–11	25, “0”, 23, 10–2
	8 Mb x 32	25–24	26, 23, 22–11	25, “0”, 23, 10–2
512 Mb	64 Mb x 8 DDR1	27–26	26, 24, 22–11	25, “0”, 23, 10–2
	64 Mb x 8 DDR2	27–26	25, 24, 22–11	25, “0”, 23, 10–2
	32 Mb x 16	26–25	26, 24, 22–11	25, “0”, 23, 10–2

Table 4: Address Multiplex for 16-bit DDR SDRAM Interface

SDRAM Type		M_BA [1:0]	Row M_A[13:0]	Column M_A[11:0]
128 Mb	16 Mb x 8	24–23	25, 23, 21–10	24, “0”, 22, 9–1
	8 Mb x 16	23–22	25, 23, 21–10	24, “0”, 22, 9–1
256 Mb	32 Mb x 8	25–24	25, 23, 21–10	24, “0”, 22, 9–1
	16 Mb x16	24–23	25, 22, 21–10	24, “0”, 22, 9–1
512 Mb	64 Mb x 8 DDR1	26–25	25, 23, 21–10	24, “0”, 22, 9–1
	64 Mb x 8 DDR2	26–25	24, 23, 21–10	24, “0”, 22, 9–1
	32 Mb x 16	25–24	25, 23, 21–10	24, “0”, 22, 9–1

4.3 DDR SDRAM Timing Parameters

The DDR SDRAM controller supports a wide range of DDR SDRAM timing parameters (see Table 5). These parameters can be configured through the DDR SDRAM Mode Register (Table 133 p. 269) and the DDR SDRAM Timing (Low) Register (Table 125 p. 265) and DDR SDRAM Timing (High) Register (Table 126 p. 266).

Note

DDR SDRAM timing parameters are the same to all DDR SDRAM physical banks.

The DDR SDRAM Trc timing parameter is the sum of Tras and Trp.

Table 5: DDR SDRAM Timing Parameters

DDR SDRAM Timing Parameters	Description
CAS Latency (CL)	The number of cycles from M_CASn assertion to the sampling of the first read data. The DDR SDRAM controller supports CL of 1.5, 2, 2.5, 3, 4, or 5 cycles.
RAS Precharge (Trp)	The minimum number of cycles from precharge to a new activate cycle, to the same DDR SDRAM bank.
M_RASn to M_CASn (Trcd)	The minimum number of cycles between an activate cycle and a command cycle, to the same DDR SDRAM bank.
Row Active Time (Tras)	The minimum number of cycles between an activate cycle and a precharge cycle, to the same DDR SDRAM bank.
Write to Precharge (Twr)	The minimum number of cycles between a write command and a precharge cycle, to the same DDR SDRAM bank.

Table 5: DDR SDRAM Timing Parameters (Continued)

DDR SDRAM Timing Parameters	Description
Write to Read (Twtr)	The minimum number of cycles between a write command and a read command, to the same DDR SDRAM device.
Active to Active (Trrd)	The minimum number of cycles between the activation of bank A and the activation of bank B, in the same DDR SDRAM device.
Refresh Command (Trfc)	The minimum number of cycles between a refresh command and a new activate command.
Read to Read (Tr2r)	The minimum number of cycles between consecutive read commands to different devices. It is not part of the JEDEC spec. It is used for preventing contention between consecutive reads to different DDR SDRAM devices (different chip selects). The DDR SDRAM controller supports Tr2r of 1 or 2 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 spec. It is used for preventing contention between consecutive read after write or write after read. The DDR SDRAM controller supports Tr2w_w2r of 1 or 2 cycles.

4.4 DDR SDRAM Burst

A DDR SDRAM device can be configured to different burst lengths and burst ordering. The 88F5182 DDR SDRAM controller supports only a Burst Length(BL) setting of four. It only supports linear wrap around burst type. (The <BT> bit of the DDR SDRAM Mode Register (Table 133 p. 269) must be set to 0.)

A single DDR SDRAM access can vary from a single byte up to a 32B burst, which turns out to be a burst of four cycles (8 beats) on a 32-bit DDR SDRAM interface, or 8 cycles (16 beats) on a 16-bit DDR SDRAM interface.

When the required DDR SDRAM access is not a full multiple of the DDR SDRAM burst lengths (BL), the burst needs to be terminated. The DDR SDRAM controller terminates the burst by driving a precharge cycle and asserting the DM signals.

Note

- In the case of open pages, the burst is not terminated with a pre-charge but with burst terminate command (BST). This is done to keep the page open (see Section 4.5, DDR SDRAM Open Pages, on page 30).
- In DDR2, burst cannot be terminated before reaching the BL boundary. (There is no BST command.)

Since the DDR SDRAM controller effectively accesses 64-bits for each cycle (In the case of 32-bit DDR SDRAM interface), it always accesses the DDR SDRAM to 64-bit aligned addresses and uses DM to mask non-desired writes. For example, a three 32-bit word burst write to offset 0x1 is executed as a write of four 32-bit words to offset 0x0, with DM masking the first 32-bit word.

4.4.1 DDR SDRAM Bank Interleaving

The 88F5182 supports both physical banks (M_CSn[3:0]) interleaving and internal banks (M_BA[1:0]) interleaving.

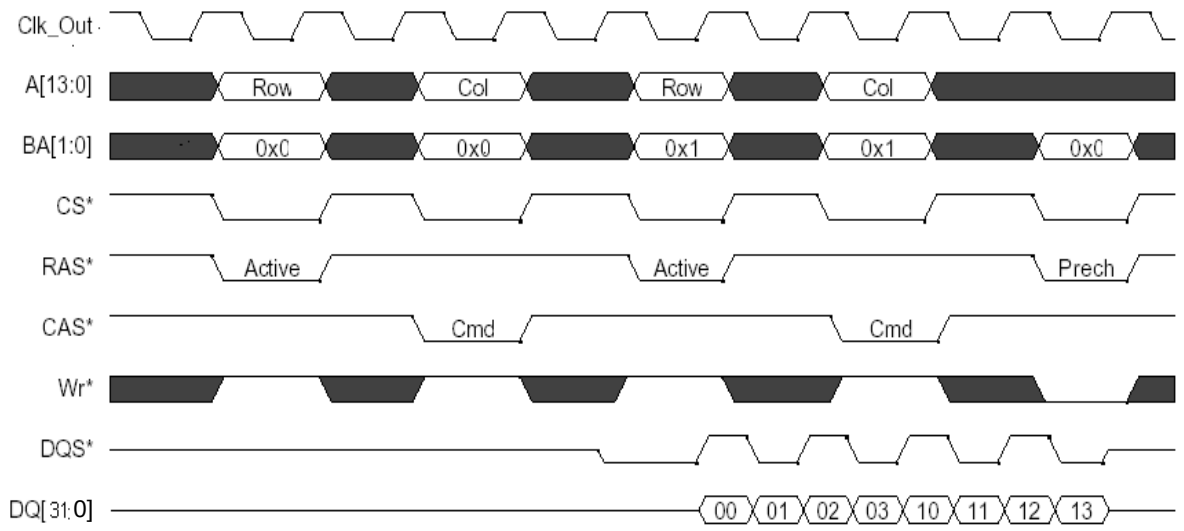
Interleaving provides higher system performance by hiding the active cycles of a new transaction during the data cycles of a previous transaction. This technique gains maximum utilization of the DDR SDRAM bus bandwidth.

Interleaving occurs when there are multiple pending accesses to different DDR SDRAM banks, whether the accesses are by internal banks distinguished by different M_BA[1:0] values or physical banks distinguished by different M_CSn[3:0].

The DDR SDRAM controller performs bank interleaving between the current active transaction and the next transaction to be executed. It does not matter to the controller whether the next transaction comes from the 88F5182 Mbus or from AHB bus.

Figure 4 shows an example of interleaving between two reads to different internal banks.

Figure 4: DDR SDRAM Bank Interleaving



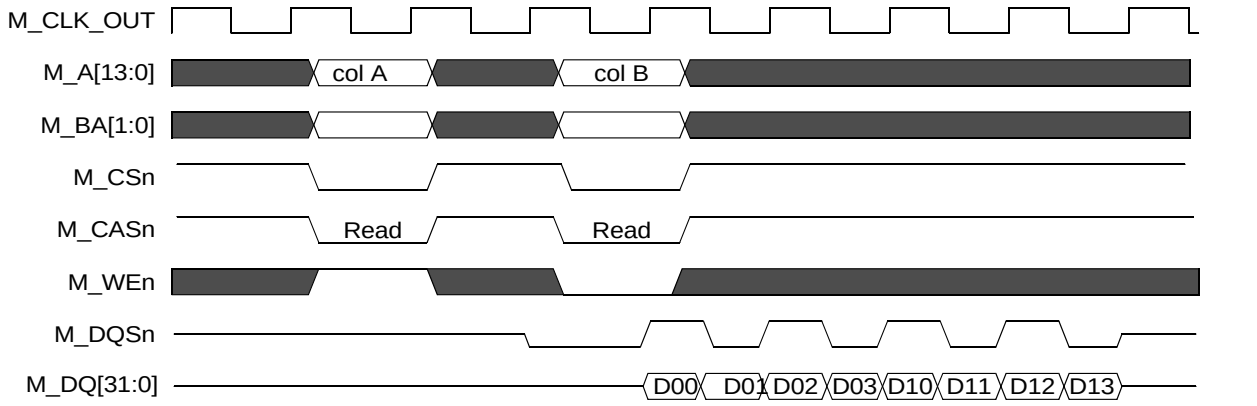
Since the two accesses are targeted to different internal banks (BA[1:0]), interleaving is enabled. Activate and command cycles of the second transaction are issued while the first transaction is receiving read data.

4.5 DDR SDRAM Open Pages

It is possible to configure the 88F5182 DDR SDRAM controller to keep DDR SDRAM pages open. It supports up to 16 open pages simultaneously—one page per each internal bank.

When a page is kept open at the end of a burst (no precharge cycle) and if the next cycle to the same internal bank hits the same page (same row address), there is no need for a new activate cycle. Figure 5 shows an example of access to an open page.

Figure 5: Consecutive Reads to the Same Page



Once a page is open, it remains open until one of the following events occurs:

- There is an access to the same bank but to a different row address. The DDR SDRAM controller performs a precharge (closes the page) and opens a new one (the new row address).
- The refresh counter expires. The DDR SDRAM controller closes all open pages and performs a refresh to all banks.

4.6 DDR SDRAM Refresh

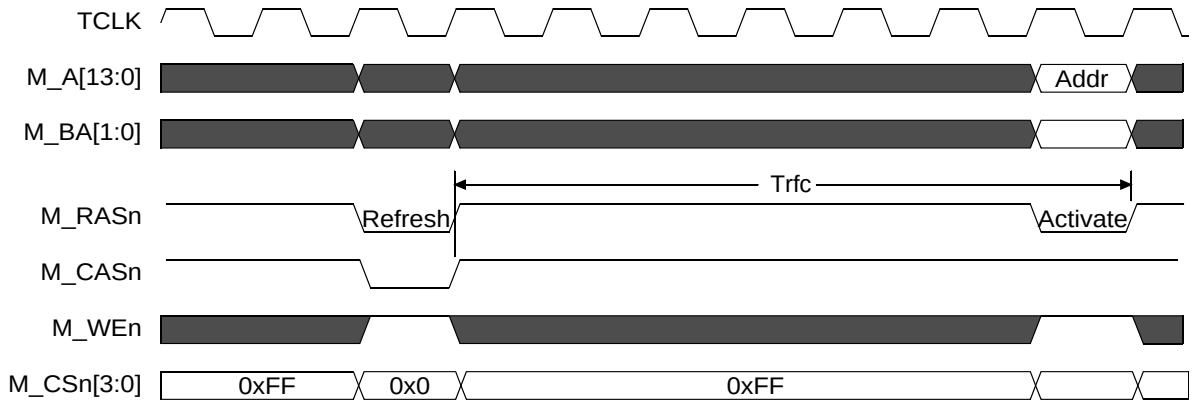
The 88F5182 implements standard CAS before RAS refreshing.

The refresh rate for all banks is determined according to the 14-bit Refresh value in DDR SDRAM Configuration Register (Table 123 p. 263). For example, the default value of Refresh is 0x200. If the M_CLK_OUT frequency is 166 MHz (6 ns cycle), a refresh sequence occurs every 3.072 µs.

Every time the refresh counter reaches its terminal count, a refresh request is sent to the DDR SDRAM controller. It has a higher priority over any other DDR SDRAM access request. As soon as the current outstanding DDR SDRAM transactions complete, the DDR SDRAM controller precharges all banks (both the ones that are open, and the ones that are not open), and performs an automatic refresh command to all DDR SDRAM banks.

Figure 6 shows a refresh cycle example.

Figure 6: DDR SDRAM Refresh



4.7 DDR SDRAM Initialization

The DDR SDRAM controller automatically starts the DDR SDRAM initialization sequence as soon as the Feroceon CPU core sets field <InitEn> in the DDR SDRAM Initialization Control Register (Table 135 p. 271) to 1.

The software must initialize the DDR SDRAM Control registers prior to setting the <InitEn> bit.

Note

- The <InitEn> bit in the DDR SDRAM Initialization Control Register (Table 135 p. 271) can be set only once by the CPU. To change the SDRAM mode and SDRAM extended mode values after initialization has finished, see Section 4.8, DDR SDRAM Operation Register.
- The DDR SDRAM controller postpones any attempt to access DDR SDRAM before the initialization sequence completes. The software needs to poll the <InitEn> field until it resets to 0 before issuing the first access to the DDR SDRAM.

DDR1 Initialization Sequence

After the Feroceon CPU core sets field <InitEn> in the DDR SDRAM Initialization Control Register (Table 135 p. 271), the DDR SDRAM control unit automatically performs the following steps:

1. Precharge all DDR SDRAM banks (all four physical banks).
2. Issue EMRS command based on the Extended DDR SDRAM Mode Register value, to enable the DDR SDRAM DLL.
3. Issue MRS command based on the DDR SDRAM Mode Register (Table 133 p. 269) values, with the reset <DLL> bit [8] activated.
4. Wait 200 cycles.
5. Precharge all banks.
6. Generate two auto-refresh cycles.
7. Load the DDR SDRAM Mode Register with the default DDR SDRAM parameters and with the reset <DLL> bit [8] de-activated.

DDR2 Initialization Sequence

DDR2 initialization sequence consists of the following steps:

1. Precharge all DDR SDRAM banks (all four physical banks).
2. Issue command EMRS(2).
3. Issue command EMRS(3).

Note Extended Mode registers 2 and 3 are reserved registers for future use. Still, the DDR2 initialization sequence requires setting of these registers. The DDR SDRAM controller sets these registers to a value of 0x0.

4. Issue the EMRS command based on the Extended DDR SDRAM Mode Register (Table 134 p. 270) value to enable the DDR SDRAM DLL.
5. Issue MRS command based on the DDR SDRAM Mode Register (Table 133 p. 269) values, with the reset <DLL> bit [8] activated.
6. Wait 200 cycles.
7. Precharge all banks.
8. Generate two auto-refresh cycles.
9. Issue the MRS command based on the DDR SDRAM Mode Register values, with the reset <DLL> bit [8] de-activated.

The DDR2 DDR SDRAM Mode Register (Table 133 p. 269) and Extended DDR SDRAM Mode Register (Table 134 p. 270) contain fields that do not exist in DDR1 SDRAM:

- WR (Write Recovery for Auto-Precharge): Not relevant; the 88F5182 does not use auto-precharge.
- PD (Active Power Down Exit Time): Not relevant; the 88F5182 does not support power down mode.
- DIC (Output Driver Impedance Control): Selects normal or weak output impedance.
- Additive Latency: Not supported.
- Rtt (Termination Control): Selects DDR SDRAM termination value: 75 ohm, 150 ohm, or none (see Section 4.12, DDR2 On Die Termination (ODT)).
- RDQS and DQS: Controls DQS configuration. The 88F5182 does not support RDQS nor differential DQS signaling.

4.8 DDR SDRAM Operation Register

In addition to the normal DDR SDRAM operation mode, DDR SDRAM controller also supports special DDR SDRAM commands through the DDR SDRAM Operation Register (Table 131 p. 268). These operations include:

- Normal DDR SDRAM mode (default mode)
- NOP commands
- Precharge all banks
- Load the DDR SDRAM Mode Register (Table 133 p. 269).
- Load the Extended DDR SDRAM Mode Register (Table 134 p. 270).
- Force a refresh cycle

The register contains 3 bits of command type. Once the Feroceon CPU core changes the register default to one of the command types, the DDR SDRAM controller executes the required command, resets the register back to the default value, and returns to normal operation. The Feroceon CPU core must poll on this register to identify when the DDR SDRAM controller is back in normal operation mode.

Note

When using DDR SDRAM DIMMs, the DDR SDRAM parameters are recorded in the DIMM Serial Presence Detect (SPD) serial ROM. The Feroceon CPU core can read the SPD via the 88F5182 TWSI interface and program the DDR SDRAM parameters accordingly, using the Load Mode register command.

The Feroceon CPU core must not attempt to change the DDR SDRAM Mode Register (Table 133 p. 269) setting prior to DDR SDRAM controller completion of the DDR SDRAM initialization sequence. To guarantee this restriction, it is recommended that the Feroceon CPU core sets the DDR SDRAM Operation Register to NOP command, performs read polling until the register is back in the Normal operation value, and then set the DDR SDRAM Mode Register to its new value.

4.9 DDR SDRAM Self Refresh Mode

The DDR SDRAM controller also supports DDR SDRAM Self Refresh mode.

The DDR SDRAM controller puts DDR SDRAM in Self Refresh mode by generating a refresh cycle with M_CKEx driven low. When exiting self refresh, it drives M_CKEx high, and waits for 200 cycles, before generating any new read transaction to DDR SDRAM.

Note

- There are two CKE signals (M_CKE[1:0]) for load balancing purposes. The 88F5182 drives these signals in the same way. If board topology does not require use of all of these signals, it is possible to only use one of the signals.
- The 88F5182 continues driving M_CLK_OUT and M_CLK_OUTn when DDR SDRAM is in Self Refresh mode.

4.9.1 Power Saving Mode

The DDR SDRAM Self Refresh mode is useful for power saving purposes when the system is in Standby mode. To enter this mode, the following procedure must be followed.

1. The Feroceon CPU core sets the <Cmd> field [2:0] in the DDR SDRAM Operation Register (Table 131 p. 268) to 0x7.
2. The DDR SDRAM controller waits for 256 cycles and then generates a self refresh command to DDR SDRAM and clears the DDR SDRAM Operation Register (Table 131 p. 268).
3. If there are new pending transactions to DDR SDRAM, the DDR SDRAM controller sets M_CKE0/1 and waits 200 cycles before generating a new transaction to DDR SDRAM.

Note

In this mode, all of the DDR SDRAM signals (excluding M_CLK_OUT, M_CKE0/1, and M_STARTBURST) are floated, significantly reducing the power consumption.

It is the responsibility of the system software to trigger the DDR SDRAM Self Refresh mode, only when the system is in Standby mode and is unlikely to receive new DDR SDRAM access requests,. It is recommended that the software, after requesting Self Refresh mode, wait 256 SYS_CLK cycles, and then reads DDR SDRAM Operation Register to confirm that the DDR SDRAM is in Self Refresh mode.

When the DDR SDRAM is in Self Refresh mode, the DDR SDRAM can only be accessed with a read/write transaction.

Caution Attempts to access the DDR SDRAM with one of the operation commands (e.g., the MRS command) results in a system hang.

4.10 DDR SDRAM Address/Data Drive

The DDR SDRAM clock is driven by the M_CLK_OUT/M_CLK_OUTn differential pair in the 88F5182. All DDR SDRAM address and control signals driven by the 88F5182 (single data rate signals) are coupled to the rising or to the falling edge of this clock, according to the setting of the <CtrlPos> field [6] of the DDR SDRAM Control Register (Table 124 p. 264).

Note Typically, address and control signals are driven with the rising edge of M_CLK_OUT. However, under certain board topology and DDR SDRAM load, there may be a hold time problem on these signals. In this case, use the falling edge setting (0).

The front-end logic of the DDR SDRAM controller is responsible for correctly driving of the double data rate data with the M_DQSn signals, as well as for unpacking of the data from 64-bit SDR to the 32-bit DDR.

During a write transaction, 64-bit wide data is pulled out of the write buffer and driven as 32-bit DDR on the bus. The first 32-bit is driven with rising edge of M_CLK_OUT and the second 32-bit with falling edge of M_CLK_OUT. The DDR SDRAM controller drives DQS (data strobe) along with the data. The DDR SDRAM specification requires very accurate DQS timing in respect to the DDR SDRAM clock. The DDR SDRAM controller uses an accurate DLL + delay line to “put the data in place” (shift DQ by 1/4 cycle).

4.11 DDR SDRAM Read Data Sample

The front-end logic of the DDR SDRAM controller is responsible for the correct sampling of the DDR data with M_DQSn signals, as well as the packing of data from 32-bit DDR to 64-bit SDR. The 32-bit DDR read data is latched via the received DQS (shifted by 1/4 cycle, via delay line). The first 32-bits are sampled with the rising of DQS, and the next 32-bit data with the falling of DQS.

To meet the DDR SDRAM AC specification, packed 64-bit read data cannot simply be sampled with the internal DDR SDRAM controller clock. The exact sample point depends on CL (CAS latency) and board topology. The controller drives a M_STARTBURST_OUTn signal. This signal is routed on board all the way to the DDR SDRAM, and back to the controller as M_STARTBURST_INn feedback. This signal is used as a reference for proper data sampling.

For a proper sample of read data, set the <StBurstDel> field [27:24] in the DDR SDRAM Control Register (Table 124 p. 264) according to Table 6.

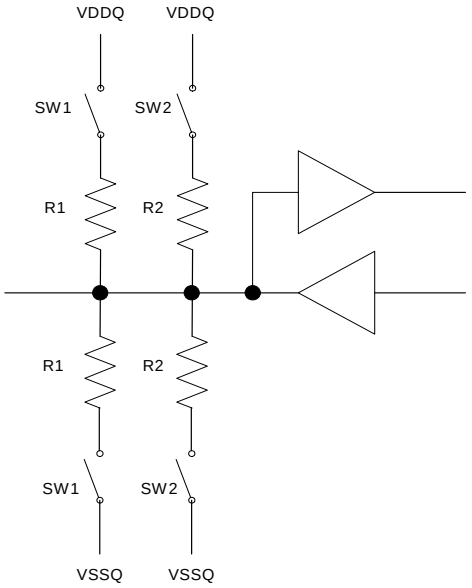
Table 6: Read Data Sampling Window Configuration

	CL=1.5	CL=2	CL=2.5	CL=3	CL=4
No register	0011	0011	0100	0100	0101
Registered DDR SDRAM	0100	0100	0101	0101	0110

4.12 DDR2 On Die Termination (ODT)

The DDR2 supports dynamic turn ON and OFF of termination resistors within the DDR SDRAM I/O buffers, as well as the DDR SDRAM controller I/O buffers. Figure 7 shows the DDR2 I/O buffer.

Figure 7: DDR2 I/O Buffer



The R1 nominal value is 300 ohm, and R2 nominal value is 150 ohm, resulting in Rtt (R/2) of 150 ohm or 75 ohm, respectively.

The DDR SDRAM controller has four ODT signals (M_ODT[3:0]) per the four DDR SDRAM physical banks. There is a fifth ODT signal internal to the 88F5182 to control the termination inside the device's I/O buffers. The ODT signals can dynamically turn the DDR SDRAM termination ON and OFF. This is useful for maintaining proper signal integrity with minimum reflections on the lines, without adding any external termination resistors.

The Extended DDR SDRAM Mode Register (Table 134 p. 270) <Rtt> fields control whether to use R1 or R2 termination during ODT operation, or disable termination.

Typically, when driving a signal and eliminating reflections, place a termination resistor at the end of the line. When the 88F5182 drives data on the DQ lines (write transactions), it is desired to turn ON the termination on the DDR SDRAM. On the other hand, when the DDR SDRAM is driving DQ signals (read transactions), it is required to turn ON the termination on the 88F5182 I/O buffer.

In a multiple DDR SDRAM physical banks environment, termination topology is more complex. It requires some board simulation. The 88F5182 DDR SDRAM controller gives the full flexibility to select which of the four DDR SDRAM physical bank terminations to turn ON or OFF per any read or write transaction to any of the four banks. The turn ON and turn OFF timing is also configurable.

Note Since DDR SDRAM termination turn ON timing is two cycles, when using CL = 3 (i.e., a write latency of 2), the DDR SDRAM controller needs to delay the write command by one cycle. To assert ODT one cycle before the command, activate termination during the write preamble.

The DDR SDRAM controller can also be configured to a static termination configuration—rather than a dynamic termination configuration—on a per transaction basis.

4.13 DDR SDRAM Interface I/O Buffers

The DDR SDRAM interface supports the standard SSTL_2 and SSTL_18 specs. For DDR1 operation, use parallel termination on all DDR SDRAM interface signals (including CLK and CLKn). For DDR2 operation, there is no need for external termination (see Section 4.12, DDR2 On Die Termination (ODT), on page 36).

The DDR SDRAM interface I/O buffers have a calibration mechanism to control buffer output impedance. Calibration is automatically performed after reset, against an external resistor tied to the M_CAL pin. The auto calibration setting can later be overridden by a software setting of the DDR SDRAM Address/Control Pads Calibration Register (Table 136 p. 272) and DDR SDRAM Data Pads Calibration Register (Table 137 p. 272).

5 PCI Express Interface

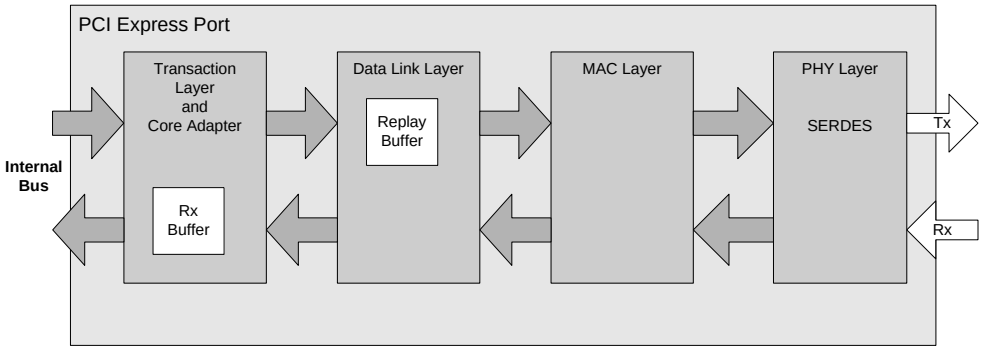
The PCI Express interface is a x1 Root Complex port, allowing 250 MB of bandwidth in ingress and egress directions simultaneously, with a total bandwidth of 500 MB. This interface has the following features:

- *PCI Express Base 1.0a* compatible
- Root Complex port
- Can be configured also as an Endpoint port
- Embedded PCI Express PHY based on Marvell® SERDES technology
- x1 link width
- 2.5 GHz signalling
- Lane polarity inversion 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: L0s-Rx and SW L1 support
- Advanced Error Reporting (AER) capability support
- Single function device configuration header.
- Message Signaled Interrupts (MSI) capability support, as an Endpoint.
- Power Management (PM) capability support, as an Endpoint.
- Expansion ROM support
- Programmable address map.

5.1 Functional Description

The PCI Express (PCIe) 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 PCIe TLPs (Transaction Layer Packets) to the internal bus.

Figure 8: High-level Block Diagram



5.1.1 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 8b10b encoding/decoding. The PHY is responsible for compensating for the skew between the different lanes. This is called lane-to-lane deskew. Such differences can be caused by factors such as board routing. The PHY forwards the aligned symbol to the MAC layer.

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).

5.1.2 MAC Layer

The MAC layer's is responsible for establishment and maintenance of the PCI Express link, packet framing and data packing according to the link width. The PCI Express LTSSM (Link Training and Status State Machine) is located in this layer. It contains all the functionality for link configuration in terms of the link width, lane polarity, and lane numbering. Additionally, the MAC layer performs the scrambling and 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 (Ordered Sets).

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.

5.1.3 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 PCIe 1.0a specifications.

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.

5.1.4 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 internal bus and schedules them for transmission according to 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 to determine if they belong to a valid NP request sent by the device. TLPs are then passed over to the internal bus.

5.2 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:

- Single outstanding NP (Non-Posted) request. Either memory read, I/O or Configuration requests.
- In Endpoint mode, up to four outstanding NP (Non-Posted) request
- Maximum memory read request of 128 bytes
- Maximum memory write request of 128 bytes
- 64-bit addressing

5.3 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:

- Single outstanding NP (Non-Posted) request. Either memory read, I/O or Configuration requests.
- In Endpoint mode, up to four outstanding NP (Non-Posted) request
- Maximum I/O read request of 4 bytes
- Maximum I/O write request of 4 bytes
- 32-bit addressing

Note

- Both I/O read requests and I/O write requests are Non-Posted requests.
- The user must not initiate I/O requests that are larger than 4 bytes and cross 4 bytes address boundary. Such requests are illegal according to the PCI Express Base 1.0a specifications.
- Only partial I/O transactions are supported.
- In Endpoint mode, only memory request generation is allowed by the PCI Express Base 1.0a Specification. The user must not initiate I/O requests when working in Endpoint mode.

5.4 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:

- Single outstanding NP (Non-Posted) request. Either memory read, I/O or Configuration requests.
- In Endpoint mode, up to four outstanding NP (Non-Posted) request
- 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

- Both Configuration read requests and Configuration write requests are Non-Posted requests.
- In Endpoint mode, only memory request generation is allowed by the PCI Express Base 1.0a Specification. User must not initiate Configuration requests when working in Endpoint mode.

5.4.1 Configuration Requests Generation

Configuration requests are generated by using the following *PCI Express Configuration Requests Generation Registers*:

1. PCI Express Configuration Address Register (Table 149 p. 280)
2. PCI Express Configuration Data Register (Table 150 p. 280)

The following procedure is used for generating configuration cycles:

1. PCI Express Configuration Address Register (Table 149 p. 280)—Write the Target Bus, Device, Function, Register and Extended Register Numbers fields, using <ConfigEn> bit[31] to enable this mechanism.
2. PCI Express Configuration Data Register (Table 150 p. 280)—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 (Table 149 p. 280).

The Configuration request generation is enabled only when the <ConfigEn> bit is set.

5.5 Target Memory Transactions

Target Memory transactions are Memory space read and write requests (MRd, MWr TLPs) received over the PCI Express link, and the respective completion TLPs generated and transmitted in return.

The following features are supported as a target Memory completer:

- Reception of up to eight Memory read requests
- Maximum received read request size of 4 KB
- Maximum received write request of 128 bytes
- Support of 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

5.5.1 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.

5.6 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 and must not be generated by downstream device.

5.7 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.

Target Configuration transactions are not supported and must 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 eight NP (Non-Posted) requests—either memory read or Configuration requests
- Maximum received Configuration read request size of 4 bytes
- Maximum received Configuration write request of 4 bytes

5.8 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 is used to enable new capabilities such as active power management, Slot Power Limit and others. The following message groups are supported as a root-complex port.

Table 7: Supported Message Groups

Message Group	Supported	Action
Interrupt Signaling	Yes	A received *Assert_INTx* message is forwarded as an interrupt to the CPU. 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	No	
Error Signaling	Yes	A received Error message is forwarded as an interrupt to the CPU. Both Correctable, Non-fatal and Fatal error messages are supported.
Hot Plug Signaling	No	
Locked Transaction Support	No	
Slot Power Limit Support	No	
Vendor Specific Messages	No	

5.8.1 Messages in Endpoint Mode

The following message groups are supported in Endpoint mode::

Table 8: 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 Deassert 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	

Table 8: Supported Message Groups: Endpoint Mode (Continued)

Message Group	Supported	Action
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	

5.9 Locked Transactions

Locked transaction semantics are not supported. That includes MRdLk, CplLk and Unlock message.

5.10 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. A simple round-robin arbitration is performed on TLPs can be forwarded according the ordering rules.

5.11 PCI Express Register Access

The following section described the PCI Express port internal registers access via the different interfaces (PCI Express interface or internal interface) and via the different address spaces (Memory or Configuration space).

5.11.1 Read Only Register Type

The Read Only (RO) registers have the following access permissions:

- RO registers can be only read via the PCI Express port. They cannot be written via it.
- All RO registers that are not hardwired or driven by the design are read/write (RW) from the internal bus.
- If enabled, all RO registers are RW through the direct memory mapping, see Section 5.11.2, Configuration Header Mapping to Internal Address Space, on page 44.

5.11.2 Configuration Header Mapping to Internal Address Space

The configuration header is mapped to the internal address space as follows:

- All configuration header registers are mapped to the internal memory space.
- Direct memory access is controlled by the <CfgMapTo MemEn> bit in the PCI Express Control Register (Table 174 p. 291). When enabled, access can be done either from the Mbus or from the PCIe port.

5.12 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 <ConfMstr HotReset> bit[24] in the PCI Express Control Register (Table 174 p. 291).
2. To check that Hot Reset has been completed, poll <DL_Down> bit[0] in the PCI Express Status Register (Table 175 p. 292). When this bit is set, DL is down and Hot Reset has been completed.
3. Clear the <conf_mstr_hot_reset> bit[24] in the PCI Express Control Register.

Note Root Complex registers are not reset by Hot Reset.

5.13 Error Handling

5.13.1 Physical Layer Errors

Table 9 list the conditions that may cause a PHY layer Receive error.

Table 9: Physical Layer Error List

Error Name	Conditions
Receiver Error	PHY Overflow PHY Underrun PHY 8B/10B decode error PHY Disparity Error Severity: Correctable.

5.13.2 Data Link Layer Errors

Table 10 lists the Data Link layer errors.

Table 10: 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.

5.13.3 Transaction Layer Errors

Table 11 lists the Transaction layer errors.

Table 11: Transaction Layer Error List

Error Name	Description
Flow Control Protocol Error	DLLP receive timer expiration. 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.
ECRC Check Error	ECRC Error detected in received TLP. 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.

Table 11: Transaction Layer Error List (Continued)

Error Name	Description
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.

Note Peer-to-peer traffic between the PCI Express port and the PCI port is supported only in one direction at a time. Simultaneously forwarding of transactions from both ports is not supported.

6 PCI Interface

6.1 Functional Description

The PCI interface runs up to 66 MHz. It supports a 32-bit bus operation. It also supports 64-bit addressing.

It can act as host bridge, translating CPU transactions to PCI memory, I/O, and configuration cycles. It can also act as PCI Endpoint, responding to host configuration cycles, and having access to all of the device's internal registers.

It also integrates a PCI bus arbiter, to support up to six masters.

6.2 PCI Master Operation

The 88F5182 PCI master supports the following transactions:

- Memory Read
- Memory Write
- Memory Read Line
- Memory Read Multiple
- Memory Write & Invalidate
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- Interrupt Acknowledge
- Special Cycle
- Dual Address Cycle

Note Only partial I/O transactions are supported.

The master generates a Memory Write and Invalidate transaction if:

- The transaction accessing the PCI memory space requests a data transfer size equal to multiples of the PCI cache line size, with all byte enables active.
- The transaction address is cache aligned.
- Memory Write and Invalidate Enable bit in the Configuration Command register is set

The master generates a Memory Read Line transaction if:

- The transaction accessing the PCI memory space requests a data transfer size equal to multiples of the PCI cache line size.
- The transaction address is cache aligned.

A Memory Read Multiple transaction is carried out when the transaction accessing the PCI memory space requests a data transfer that crosses the PCI cache line size boundary.

Note The 88F5182 supports four cache line size values—4 words (16 bytes), 8 words (32 bytes), 16 words (64 bytes), and 32 words (128 bytes). Setting the cache line size to any other value is treated as if cache line size is set to 0.

A Dual Address Cycle (DAC) transaction is carried out if the requested address is beyond 4 GB (address bits [63:32] are not 0).

The 88F5182 PCI master performs configuration read/write cycles, Interrupt Acknowledge cycles, or Special cycles using the Config Address and Config Data registers. For full details on generating these transactions, see Section 6.4, PCI Master Configuration Cycles, on page 52.

The master contains 512 bytes of posted write data buffer and 512 bytes of read buffer. It can absorb up to four 128 byte write transactions plus four 128 byte read transactions. The PCI master posted write buffer in the 88F5182 permits the initiator to complete the write even if the PCI bus is busy. The posted data is written to the target PCI device when the PCI bus becomes available. The read buffer absorbs the incoming data from PCI. Read and Write buffers implementation guarantees that there are no wait states inserted by the master.

Note PCI_IRDYn is never de-asserted in the middle of a transaction.

6.2.1 PCI Master Write Operation

On a write transaction, data from the initiator unit is first written to the master write buffer and then driven on the PCI bus. The master does not need to wait for the write buffer to be full. It starts driving data on the bus when the first data is written into the write buffer.

On consecutive write transactions, the transactions are placed into the queue. When the first transaction is done, the master initiates the transaction for the next transaction in the queue.

The master supports combining memory writes. If combining is enabled through the PCI Command register's <MWrCom> bit [4], the master combines consecutive burst write transactions, if possible.

For combining memory writes to occur, the following conditions must exist:

- Combining is enabled through the PCI Command register's <MWrCom> bit.
- The start address of the second transaction matches the address of data n+1 of the first transaction.
- While the first transaction is still in progress, the request for the new transaction occurs.

Note PCI write combining is not supported with cache line size of 4.

- Fast back-to-back is enabled when bit [9], the <FastBTBEn> field in the PCI Status and Command (Table 282 p. 345), is set to 1.
- The first transaction is a write.
- While the first transaction is still in progress, the new transaction request occurs.

6.2.2 PCI Master Read Operation

On a read transaction, when the initiator requests a PCI read access, the PCI master drives the transaction on the bus (after gaining bus mastership). The returned data is written into the read buffer. The PCI master drives the read data to the initiating unit as soon as the first data arrives from the PCI bus. It can also be configured to only drive the data when the whole burst read is placed in the read buffer via PCI Command register's <MRdTrig> bit [6] (see Table 246 on page 329).

The master also supports combining read transactions. If combining is enabled through PCI Command register's <MRdCom> bit [5], the master combines consecutive burst read transactions. For combining read transactions to occur, the following conditions must exist:

- Combining is enabled.
- The start address of the second transaction matches the address of data n+1 of the first transaction.
- While the first transaction is still in progress, the request for the new transaction occurs.

Note If the target cannot handle a long burst without wait states, combining read transactions is not recommended since the 88F5182 holds the PCI bus for a long time without using it.

6.2.3 PCI Master Transaction Termination

If there is no target response to the initiated transaction within four clock cycles (five clocks In the case of a DAC transaction), the master issues a Master Abort event. The master de-asserts PCI_FRAMEn and on the next cycle de-asserts PCI_IRDYn. Also, the Interrupt Cause register's <MMAbort> bit is set and an interrupt is generated, if not masked.

The master supports several types of target termination:

- Retry
- Disconnect
- Target Abort

If a target terminated a transaction with Retry, the 88F5182 master re-issues the transaction. In default, the master retries a transaction until it is being served. To limit the number of retry attempts, set the Retry Counter register to a desired count value. When the master reaches this count value, it stops the retries, and a bit is set in the Interrupt Cause register.

Note The 88F5182 master keeps retrying a transaction until it is served (or until Retry Counter expires). If it has multiple pending transactions in it's queue, it will not start serving a new transaction before the current retried transaction get to a completion.

If a target terminates a transaction with Disconnect, the master re-issues the transaction from the point it was disconnected. For example, if the master attempts to burst eight 32-bit dwords starting at address 0x18, and the target disconnects the transaction after the fifth data transfer, the master re-issues the transaction with address 0x2C to burst the remaining three DWORDs.

If a target abnormally terminates a transaction with a Target Abort, the master does not attempt to re-issue the transaction. A bit in the Interrupt Cause register is set and an interrupt is generated, if not masked.

6.3 PCI Bus Arbitration

The 88F5182 supports both external arbiter or internal arbiter configuration through the PCI Arbiter Control register's <EN> bit [31] (see Table 252 on page 333). If the bit is set to 1, the internal PCI bus arbiter is enabled.

Note The internal PCI arbiter is by default disabled. If using the arbiter, pull-ups must be set on all PCI_GNTn signals.

6.3.1 PCI Master Bus Arbitration

Whenever there is a pending request for a PCI access, the PCI master requests bus ownership through the PCI_REQn pin. As soon as the PCI master gains bus ownership (PCI_GNTn asserted), it issues the transaction. If no additional pending transactions exist, it de-asserts PCI_REQn the same cycle it asserts PCI_FRAMEn.

The 88F5182 implements the Latency Timer Configuration register, as defined in PCI specification. The timer defines number of clock cycles starting from PCI_FRAMEn assertion that the master is allowed to keep bus ownership, if not granted any more. If the Latency Timer is expired, and the master is not granted (PCI_GNTn not asserted), the master terminates the transaction properly on the next data transfer (PCI_TRDYn assertion). The master re-issues the transaction from the point it was stopped, similar to the case of disconnect.

One exception is Memory Write and Invalidate command. In this case, the master quits the bus only after next cache line boundary, as defined in PCI specification.

6.3.2 Internal PCI Arbiter

The 88F5182 internal PCI arbiter can handle up to six PCI masters.

All PCI_REQn inputs are sampled and all PCI_GNTn outputs are registered, thus the earliest PCI_GNTn to a non parked master, is two cycles after PCI_REQn is asserted.

The PCI arbiters implement a fixed Round Robin (RR) arbitration mechanism. The PCI arbiter performs a default parking on the last agent granted. To overcome problems that happen with some PCI devices that do not handle parking properly, use the PCI Arbiter Control register's <PD> bits [20:14] as an option to disable parking on a per PCI master basis (see Table 252 on page 333).

Note In addition to disabling parking to avoid issues with some problematic devices, disable parking on any unused request/grant pair. This avoids possible parking on non-existent PCI masters. For example, if only three masters are connected to the arbiter, then <PD> [6:4] must be set to 1.

The PCI arbiter also implement broken master detection. A master that requests the bus must initiate a transaction (assert PCI_FRAMEn) as soon as its PCI_GNTn is asserted. If the master is broken and does not initiate a PCI transaction, the PCI bus hangs permanently. To avoid this condition, the internal PCI arbiter implements a programmable broken value counter. If the master granted on the bus does not issue a transaction within the number of cycles specified in PCI Arbiter Control register <BV> bits [6:3], the arbiter de-asserts PCI_GNTn from the broken master and grants the bus to some other master.

6.4 PCI Master Configuration Cycles

The 88F5182 translates CPU read and write cycles into configuration cycles using the PCI configuration mechanism #1 (per the PCI interface).

The 88F5182 contains two registers to support configuration accesses: PCI Configuration Address (Table 272 p. 340) and PCI Configuration Data (Table 273 p. 340). The mechanism for accessing configuration space is to write a value into the Configuration Address register that specifies the:

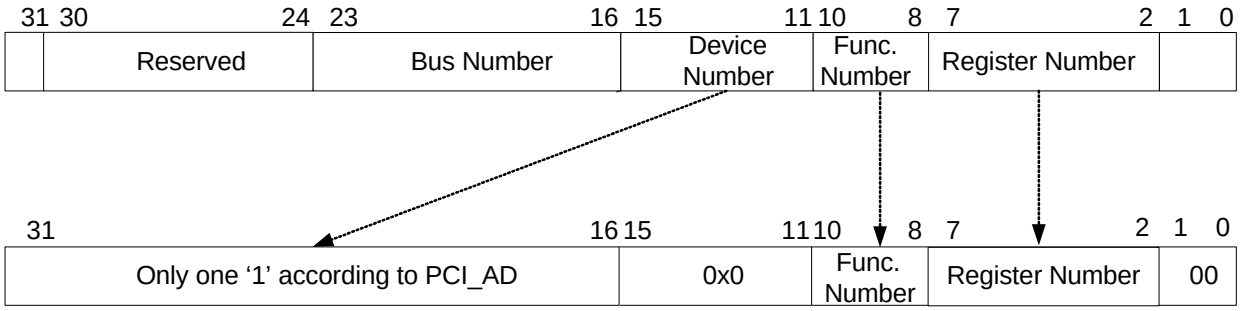
- PCI bus number
- Device number on the bus
- Function number within the 88F5182
- Configuration register within the device/function being accessed

A subsequent read or write to the Configuration Data register causes the 88F5182 to translate that Configuration Address value to the requested cycle on the PCI bus. A CPU access to the Configuration Data register blocks the CPU until the access on the PCI bus has been completed. This to make sure that the software reads the configuration data after the PCI transaction has been completed and to verify that the data is valid.

The PCI P2P Configuration register's <BusNumber> bits [23:16] and <DevNumber> bits [28:24] define the bus number to which the PCI interface of the 88F5182 is connected and the device number on this bus. These fields affect the type of configuration access the PCI master generates.

If the <BusNumber> bits [23:16] in the PCI Configuration Address (Table 272 p. 340) register equals the PCI P2P Configuration register <BusNumber> field, but the <DevNumber> fields do not match, a Type0 access is performed. This type of access addresses a device attached to the local PCI bus. The Configuration Address translation to the value driven on the PCI_AD bus during address phase is shown in Figure 9.

Figure 9: PCI Type 0 Configuration Transaction Address Translation



Driving only one 1 on PCI_AD[31:16] allows easy generation of the PCI_IDSEL signal on board, by connecting each of these AD lines to the appropriate PCI slot PCI_IDSEL signal through a resistor.

Table 12 shows Device Number to IDSEL mapping.

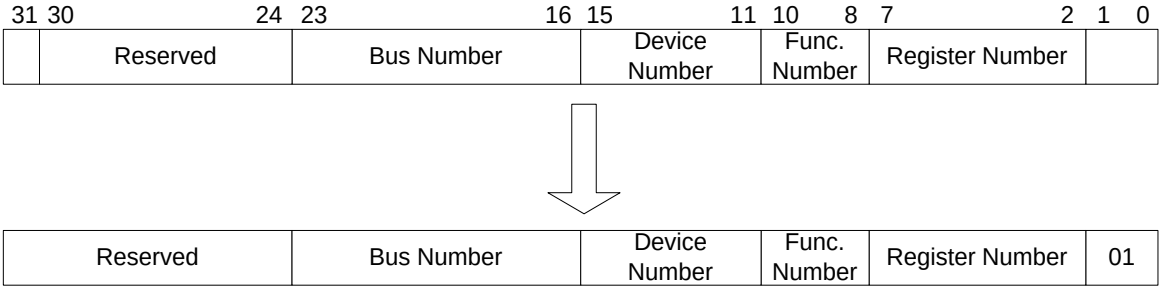
Table 12: Device Number to IDSEL Mapping

Dev #	PCI_AD[31:16]
0x0	0000.0000.0000.0001
0x1	0000.0000.0000.0010
0x2	0000.0000.0000.0100
0x3	0000.0000.0000.1000
0x4	0000.0000.0001.0000
0x5	0000.0000.0010.0000
0x6	0000.0000.0100.0000
0x7	0000.0000.1000.0000
0x8	0000.0001.0000.0000
0x9	0000.0010.0000.0000
0XA	0000.0100.0000.0000
0XB	0000.1000.0000.0000
0XC	0001.0000.0000.0000
0XD	0010.0000.0000.0000
0XE	0100.0000.0000.0000
0XF	1000.0000.0000.0000
0x10–0X1F	0000.0000.0000.0000

Note The 88F5182 performs address stepping for the PCI configuration cycles. Once granted on the bus, it drives a valid address and command on PCI_AD and PCI_CBEn respectively one cycle before asserting PCI_FRAMEn.

If the PCI Configuration Address register's <BusNum> field does not match the PCI P2P Configuration register's <BusNum> field, a Type1 access is performed. This access type addresses a device attached to a remote PCI bus. In this case, the Register Number, Function Number, Device Number, and Bus Number are copied directly from the Configuration Address register to the PCI_AD bus, as shown in Figure 10.

Figure 10: PCI Type 1 Configuration Transaction



A special cycle is generated if all of the following apply:

- The <BusNum> field in the Configuration Address register equals the PCI P2P Configuration register's <BusNum> field.
- The <DevNum> field is 0x1F.
- The function number is 0x7.
- The register offset is 0x0.

The CPU accesses the 88F5182's internal configuration registers when the <DevNum> and <BusNum> fields in the Configuration Address register match the corresponding fields in the PCI P2P Configuration register.

Note The configuration enable bit (<ConfigEn>) in the Configuration Address register must be set before the Configuration Data register is read or written. If this bit is not set, no transaction is driven on the PCI bus. In the case of a write transaction, the data is lost. In the case of a read transaction, a non-deterministic value is returned to the CPU.

6.5 PCI Target Address Decoding

The 88F5182 PCI target interface supports multiple address windows, each defined by the base and size registers. Each window (except of the internal register windows) can decode up to 4 GB space. Each window cannot cross this 4 GB space boundary.

All memory mapped BARs (Base Address Register) are 64-bit registers, supporting 64-bit addressing. If the upper 32-bit of the BAR is set to 0, the BAR acts as a 32-bit BAR and the 88F5182 PCI slave only responds to a SAC transaction. If the BAR's upper 32-bit value is other than 0 (i.e., it allocates a window above the 4 GB address space), the slave responds only to DAC transactions.

The PCI slave responds to an address hit in the I/O BARs only if the configuration Command register's bit [0] (Target I/O Enable) is set to 1. It responds to an address hit in any of the other BARs only if bit [1] of configuration Command register (Target Memory Enable) is set to 1.

To disable a specific BAR space, the 88F5182 includes a BAR Enable register—bit per BAR. Setting a bit to 1 disables the corresponding BAR. A disabled BAR is treated as a reserved register (read only 0). PCI access match to a disabled BAR is ignored and no PCI1_DEVSELn is asserted.

The PCI target interface also supports address remapping to any of the resources. This is especially useful when one needs to reallocate some PCI address window to a different location in memory.

Note There are no size registers for the internal space BARs. They are fixed 1 MB in size.

By default, the four DRAM BARs correspond to the four DRAM chip selects (CS0n BAR maps DRAM M_CSn[0] bank, CS1n BAR maps DRAM M_CSn[1] bank and so on). This default setting can be overridden via the DRAM BAR Bank Select register, allowing two different BARs to map the same physical DRAM bank. For example, setting DRAM BAR Bank Select register's <DB2> field to 0x0, forcing CS2nBAR to map DRAM M_CSn[0] (in addition to CS0n BAR that also maps DRAM M_CSn[0] bank).

This feature is especially useful when working with two PCI agents transferring data to each other via the 88F5182 DRAM, with one device in Little Endian mode and the other in Big Endian mode. The PCI Access Control registers enable different byte swapping per each PCI address space, while the transactions eventually reach the same address in the target DRAM.

6.6 PCI Access Protection

The PCI slave interface supports configurable access control. It is possible to define up to six address ranges to different configurations. Each region can be configured to control PCI slave:

- Write and access protection
- Byte swapping
- Maximum burst size
- Read prefetch

An address received from the PCI is compared against the Access Control registers. If an address matches one of the access windows, the 88F5182 handles the transaction according to transaction type and the attributes programmed in the Access Control Base Address registers.

Each region contains two protection bits:

Access protection Any PCI access to this region (read or write) is forbidden.

Write protection Any PCI write access to this region is forbidden.

This feature is useful for protection from software errors (PCI access to memory space that it is not expected to access). If an access violation occurs:

- The PCI slave interface terminates the transaction with Target Abort.
- The transaction address is latched in the PCI Error Address (Low) register (see Table 278 on page 343).
- The PCI <STAbort> bit in the interrupt cause register is set.

Note The 88F5182 internal register space is not protected, even if the access protection windows contain this space.

The other attributes of the Access Control registers are discussed in Section 6.7, PCI Target Operation, on page 56.

6.7 PCI Target Operation

The 88F5182 responds to the following PCI cycles as a target device:

- Memory Read
- Memory Write
- Memory Read Line
- Memory Read Multiple
- Memory Write and Invalidate
- I/O Read
- I/O Write
- Configuration Read
- Configuration Write
- DAC Cycles

The 88F5182 does not act as a target for Interrupt Acknowledge and Special cycles (these cycles are ignored). The 88F5182 does not support Exclusive Accesses. It treats Locked transactions as regular transactions (it does not support LOCKn pin).

The slave consists of 512 bytes of posted write data buffer that can absorb up to four 128 byte write transactions (or a long burst write of 512 bytes), and four read prefetch buffers, 256 bytes each, to support up to four delayed reads.

6.7.1 PCI Posted Write Operation

Except for configuration writes, all PCI writes are posted. Data is first written into the posted write buffer and later written to the target device.

The slave supports unlimited burst writes. The write logic separates the long PCI bursts to fixed length bursts towards the target device. Program the internal burst length to four, eight, or 16 QWORDs through PCI Access Control Base 0 (Low) register's <WrMBurst> bits [9:8]. Whenever this burst limit is reached, the slave generates a write transaction toward the target device, while continuing to absorb incoming data from the PCI. The PCI burst writes have no wait states (PCI_TRDYn is never de-asserted). If the slave transaction queue is full, a new write transaction is retried (or if it becomes full during a long burst write, the burst is disconnected).

The slave posting writes logic also aligns bursts that do not start on a 32-/64-/128-byte boundary, depending on the burst setting, for more efficient processing by the target units. For example, if <WrMBurst> is set to maximum bursts of eight QWORDS, and a PCI long burst write transaction starts at address 0x18, the slave issues a write transaction of five QWORDS to the target unit and continues with a new transaction to address 0x40.

Note

- If the PCI address does not match any of the PCI Access Control registers address windows, the PCI slave acts as if <WrMBurst> is programmed to 32-bytes.
- The PCI specification defines that I/O writes must not be treated as posted transactions. The slave does not meet this requirement.

6.7.2 PCI Non-Posted Writes

PCI configuration writes are non-posted. The slave asserts PCI_TRDYn only when data is actually written to the configuration register. This implementation guarantees that there is never a race condition between the PCI transaction changing address mapping (Base Address registers) and the following transactions.

Note

A PCI write to any of the 88F5182 internal registers is treated as a posted write, i.e., there may be a race condition, if these registers are set from PCI (e.g. changing a size register and then accessing the window defined by this window). To prevent race conditions such as these, it is recommended to follow the write transaction with a read transaction to the same register, to guarantee that it is updated.

6.7.3 PCI Read Operation

PCI read access suffers from the following limitations:

- There is no way for the target device to know in advance the amount of data required, thus it needs to speculate how much data to prefetch from the target interface.
- Read accesses from high latency interface typically result in many dead cycles on the PCI bus. The PCI master is waiting for the read data to return. This results in low bus utilization.

The 88F5182 PCI slave design, is targeted to solve these basic limitations by:

- User defined typical burst read size per address window. This allows the slave to prefetch the amount of data required by the initiating master.
- All reads are handled as delayed transactions. This enables better bus utilization, in the case of multiple masters.

There are two fields in the PCI Access Control Base 0 (Low) register that affect the slave read behavior—<RdMBurst> bits [9:8] and <RdSize> bits [11:10]. <RdMBurst> defines the maximum burst size the slave requests from the target interface. This parameter is mainly used for bandwidth and latency considerations on the target interface. <RdSize> defines the typical amount of read data required. The slave prefetch data from the target interface is based on the <RdSize> setting. If for example, <RdMBurst> defines maximum burst of 64 bytes and <RdSize> defines typical burst of 256 bytes, the slave generates four 64-byte read transactions toward the target interface, to prefetch the 256 bytes.

There is one exception to the above prefetch scheme. This exception is a read with PCI_FRAMEn assertion for a single cycle. This kind of transaction implies that the master initiating the transaction requires only a single data. In this case, the slave requests only a single data from the target interface.

Note

If the PCI address does not match any of PCI Access Control register's address windows, the slave acts as if both <RdMBurst> and <RdSize> are set to 32 bytes.

The slave supports up to four pending delayed reads. Upon receiving a read transaction, the slave issues a PCI_STOPn immediately (retry termination) but, internally, continues the transaction towards the target interface, prefetching the required amount of data as previously explained. When the data is received from the target, it is written to one of the four read buffers. When the data is ready in the read buffers, a retry of the original transaction results in data driven immediately on the PCI bus. Any attempt to retry the original transaction before the entire data amount is placed in the slave read buffers results in a PCI_STOPn issued by the slave.

If a PCI read transaction is still alive (implying that a longer burst is required) by the time all the burst data is driven on the PCI bus, the slave terminates the transaction with a disconnect.

The slave handles a queue of available free read buffers. With each incoming read transaction, the slave allocates a new read buffer. The read buffer is used storing the read data coming from the target interface. If all four read buffers are full when the slave receives a new read transaction, the incoming read transaction is terminated with RETRY.

To prevent dead locks due to “stuck” buffers (delayed reads that are never completed), the 88F5182 supports a programmable PCI Discard Timer register (see Table 250 on page 333). Each read buffer has its timer initialized to the Discard Timer value. Once a buffer is valid, the buffer timer starts counting down. If the buffer timer reaches 0 before being accessed (no delayed read completion), the buffer is invalidated. Setting the Discard Timer register to 0 prevents the slave from invalidating read buffers.

6.7.4 Aggressive Prefetch

The read operation described above is efficient for transfers of small data payloads (up to 256 bytes). It also efficient for a system with multiple PCI masters interfacing the 88F5182 PCI slave. In such configurations, the multiple delayed reads support can achieve high PCI bus bandwidth.

However, when a single PCI master is interfacing the 88F5182 PCI slave, and this master typically reads big data payloads, a more aggressive read prefetch mechanism is required. Each of the PCI access windows can be marked as aggressive read prefetch via PCI Access Control Base 0 (Low) register’s <Aggr> bit [10], see Table 254 on page 334. If set to 1, the <RdSize> field is ignored, and the PCI slave treats any read that hits that window with the aggressive prefetch policy.

Aggressive prefetch works as follows:

1. The PCI slave terminates the transaction with RETRY.
2. It then prefetches between 132 bytes and 256 bytes, depending on the address alignment from the target unit.
3. The PCI slave starts driving data on the bus depending on the PCI Access Control Base 0 (Low) register’s <AggrWM1> bit [4] setting (see Table 254 on page 334).
4. When the PCI slave starts driving the first 256 bytes on the bus, it prefetches an additional 256 bytes from the target. When the second 256 bytes begins to be driven on the bus, the PCI slave prefetches another 256 bytes and so on. With each 256 bytes it drives on the bus, it prefetches an additional 256 bytes from the target.
5. The slave continues this prefetch loop as long as the requesting PCI master keeps requesting data (PCI_FRAMEn signal is kept asserted). Once the master is satisfied and de-asserts PCI_FRAMEn, the slave no longer prefetches data. It discards redundant data that was already prefetched, but not consumed by the master.

With this aggressive prefetch mechanism, the 88F5182 can deliver a high throughput stream of read data to the PCI masters, with no disconnects in the middle. However, if the target unit is heavily loaded, and cannot meet the PCI bandwidth, the PCI slave terminates the burst read with DISCONNECT. While internally, it keeps prefetching an additional 256 byte of data, so by the time the master re-issues the transaction, the slave can serve it again.

Note In the case of a DISCONNECT, the PCI master is expected to re-issue the transaction from the point it was stopped. If the master will not meet this requirement, the PCI slave will hang, and is only released when the Discard Timer expires.

To reduce the chances for DISCONNECT, configure the PCI slave to drive read data on the PCI when it has 512 bytes available, instead of 256 bytes. Set Access Control Size register’s <AggrWM1> bit [4] to 1, to make this change.

Aggressive prefetch by its nature, introduces some performance penalty on the target unit. This mode is likely to prefetch more data than really needed, thus wasting target unit bandwidth. This is especially true, when the requesting PCI master has variable sized requests. It sometimes requests big data payloads (e.g., data buffers) and sometimes small (e.g., descriptors). If the aggressive prefetch is constantly active, it implies that there is a waste of target unit bandwidth whenever the master request a small data payload.

To solve this conflict, place the small data structures and the big data structures in different locations, corresponding to different access windows. This way it is possible to set one access window to aggressive prefetch enabled, and the other window to non-aggressive prefetch policy.

If this solution is not possible, the 88F5182 PCI slave offers an alternative via the <AggrWM2> bits [7:5] of the Access Control Size register. This field determines the watermark upon which the PCI slave accesses the target unit and prefetches the next buffer. By default, it prefetches the next buffer with the first data it drives on the bus. However, it can be configured to any other value. For example, if the requesting PCI master reads from time to time 32-byte data payloads, it is advised to set <AggrWM2> to 0x4. This way the slave will not prefetch the next buffer from the target unit upon this descriptor read, since the read will be completed on the PCI bus, before this watermark is exceeded.

Note

Aggressive prefetch is useful when interfacing with a single PCI master. If activating this mode while having multiple masters on the bus (meaning, multiple outstanding read requests), the slave will serve only one master at a time. Meaning, once serving one aggressive prefetch request, any other read request is terminated with RETRY, and discarded. Also, new write requests are terminated with RETRY until the aggressive prefetch completes.

6.7.5 Non-Prefetchable Reads

The PCI specification allows a BAR space to be defined as non-prefetchable.

The PCI target device must guarantee that a read access to a non-prefetchable address space is not destructive (or as the PCI specification defines a prefetch from this memory space might cause “side effects”). An example of such memory space could be a FIFO device in which speculative reads are destructive.

If a PCI read access matches a non-prefetchable BAR (bit [3] of the BAR is 0), the 88F5182 PCI slave treats this read access as a non-prefetchable read, regardless of the attributes defined in the PCI access registers. It treats it as a delayed read of a single data.

During non-prefetchable read transactions, the PCI slave requests a single 64-bit word from the target interface with the required byte enables, thus guarantees there is no destructive read. This is in contrast to prefetchable reads in which the 88F5182 prefetches at least 4/8/16 QWORDS from the target interface, ignoring byte enables.

Upon the delayed read completion, if the initiating master requests more than a single data, the 88F5182 PCI slave disconnects this burst attempt.

The PCI reads from the internal and configuration registers of the 88F5182 are also treated as non-prefetchable reads.

6.7.6 PCI Target Transaction Termination

The 88F5182 PCI slave supports the three types of target termination events described in the PCI specification—Target Abort, Retry, and Disconnect.

Target Abort is activated in the following cases:

- I/O transaction with address bits [1:0] is not consistent with byte enables.
- Address parity error.
- Violation of PCI access protection setting.
- Slave accessed and the address match two or more BARs (result of BAD programming of the BAR registers).

In any of these cases:

- The PCI slave interface terminates the transaction with Target Abort.
- The transaction address is latched in PCI Error Address register.
- The PCI STAbort bit in the interrupt cause register is set.

The slave generates a RETRY termination in the following cases:

- Delayed reads (first attempt, or completion attempt while read data is not ready yet).
- A new write transaction while write buffer is full.
- A new read transaction while read buffer is full.
- Retry enable feature is active (see the retry initialization section in the *88F5182 Feroceon*® Storage Networking SoC *Datasheet*
- Non posted write (configuration write) while the write buffer is not empty.

The slave generates a DISCONNECT termination in the following cases:

- Burst access with start address bits [1:0] different than ‘00.
- Burst access that reaches BAR boundary.
- Write buffer becomes full during a burst write.
- Burst access to internal registers.
- Delayed read completion is not satisfied with the amount of data prefetched by the PCI slave

6.8 64-bit Addressing

The PCI master and slave support 64-bit addressing cycles.

If the PCI master is accessed with an address higher than 4 GB (i.e., the upper 32-bit address is not 0), the master initiates a DAC transaction. This means the transaction address phase takes two clock cycles.

On the first cycle, the master drives a ‘1101’ value on PCI_CBEn[3:0] and the lower 32-bit address on PCI_AD[31:0]. On the next cycle it drives the required command on PCI_CBEn[3:0] and the upper 32-bit address on PCI_AD[31:0].

On a DAC transaction, target address decode time is one cycle longer than in SAC transactions. Thus, the master issues a master abort on a DAC transaction only after five clock cycles, rather than four clocks in the case of SAC.

As a target, the 88F5182 responds to DAC transactions if the address matches one of its 64-bit BARs. In this case, the slave starts address decoding only after the second clock cycle (when the whole 64-bit address is available). This implies that PCI_DEVSELn is asserted three clock cycles after PCI_FRAMEn rather than after two clock cycles as in SAC transactions.

6.9 PCI Parity and Error Support

The 88F5182 implements all parity features required by the PCI specification, including PCI_PAR, PCI_PERRn, and PCI_SERRn generation and checking.

The PCI interface also supports other error conditions indications, such as access violation and illegal PCI bus behavior, see Section 6.6, PCI Access Protection, on page 55 and Section 6.7.6, PCI Target Transaction Termination, on page 59 for more details.

6.10 Configuration Space

The 88F5182 PCI interface supports Type 00 configuration space header as defined in PCI specification. The 88F5182 is a multi-function device. It supports functions 0 to 4 and the header is implemented in all of these five functions as shown in Figure 11 and Figure 12. The configuration space is accessible from the CPU or PCI buses.

The 88F5182 PCI slave responds to a type 0 PCI configuration transactions, if IDSEL is active and if the function number is between 0–4. The slave does not respond to configuration access to functions 5–7.

Many of functions 1–4 registers are aliased to function 0 registers. For example, access to Vendor ID register in function 1 actually accesses Vendor ID register of function 0.

6.10.1 Plug and Play Base Address Registers Sizing

Systems adhering to the plug and play configuration standard determine the size of a Base Address register's decode range by first writing 0xFFFF.FFFF to the BAR, then reading back the value contained in the BAR. Any bits that were unchanged (i.e., read back a zero) indicate that they cannot be set and are not part of the address comparison. With this information the size of the decode region can be determined.

The 88F5182 responds to BAR sizing requests based on the values programmed into the Bank Size Registers. Whenever a BAR is being read, the returned data is the BAR's value masked by its corresponding size register.

Figure 11: PCI Configuration Space Header

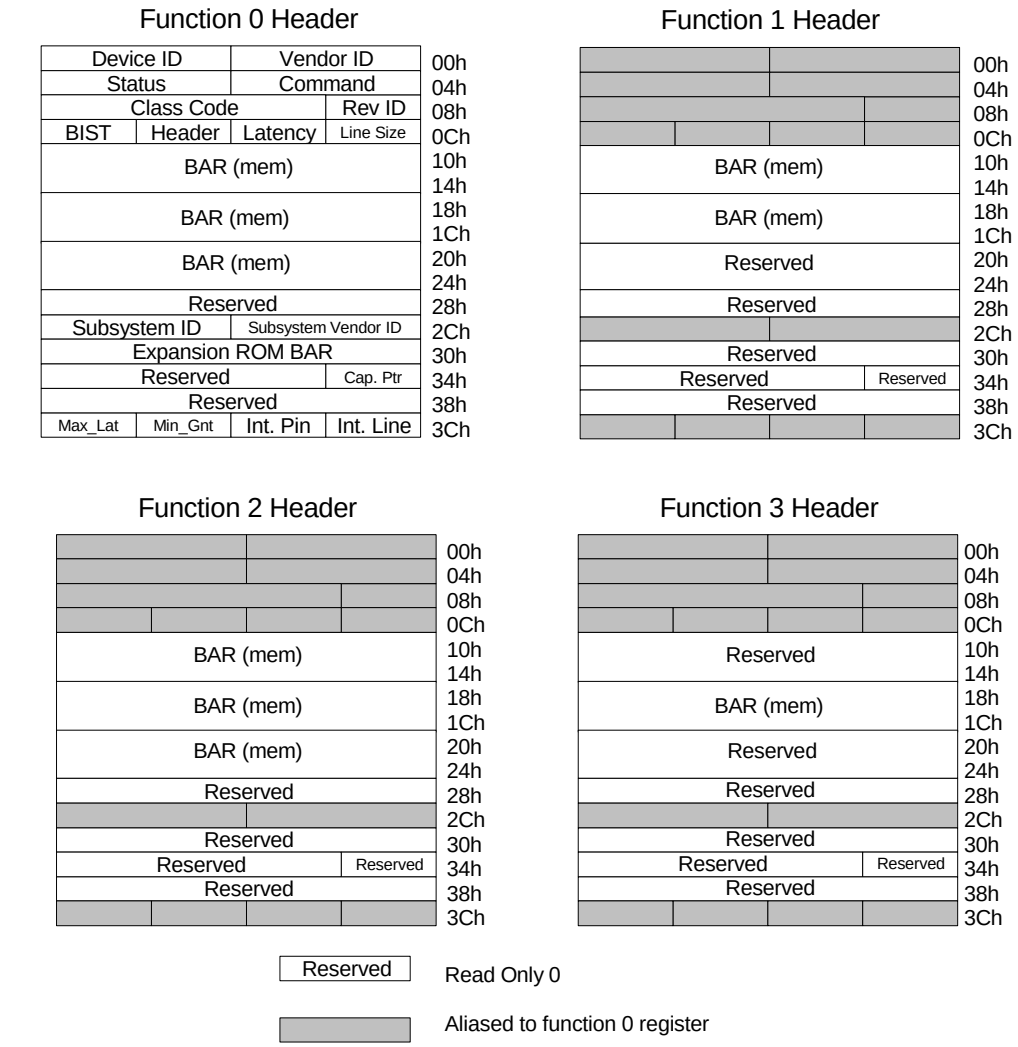
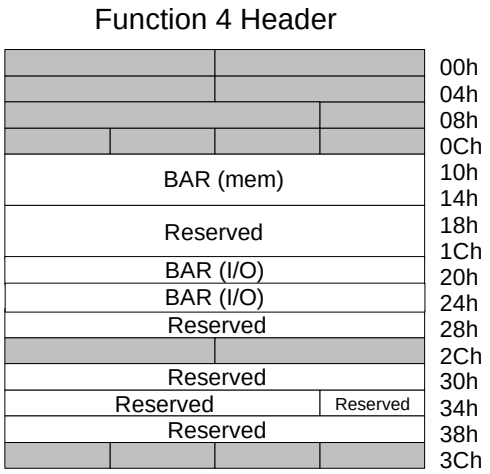


Figure 12: PCI Configuration Space Header (Continued)



6.11 PCI Add-In Card (Endpoint) Special Features

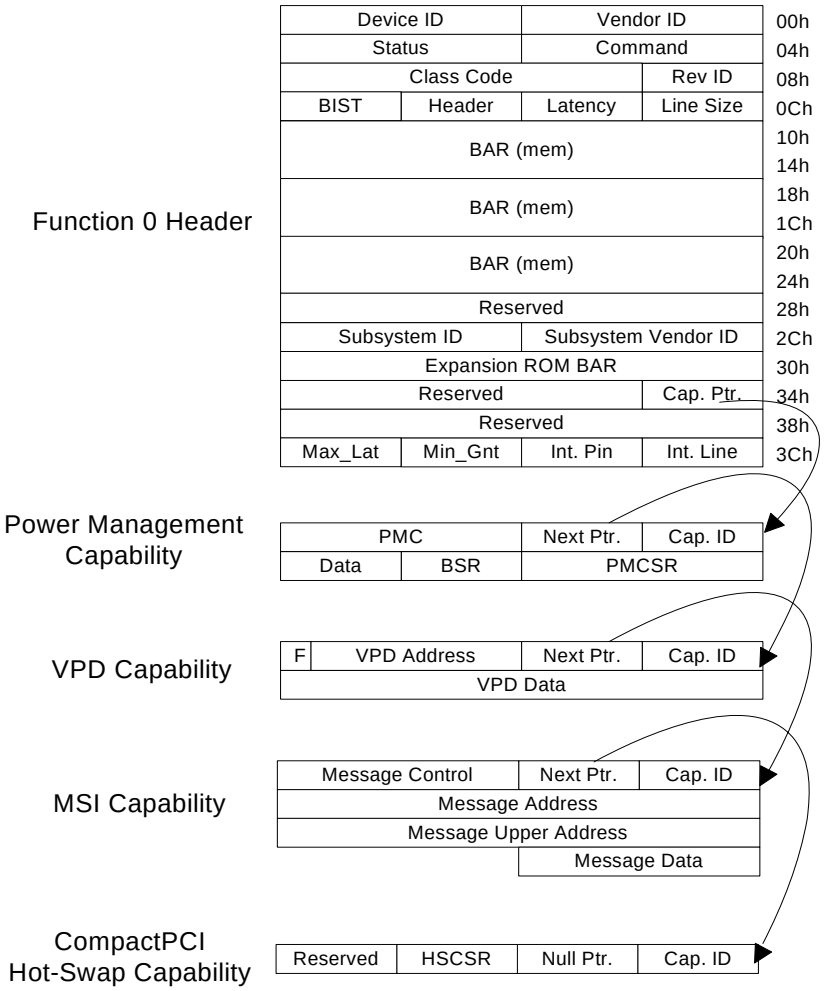
The 88F5182 supports the following special PCI features:

- Built-In Self Test (BIST)
- Vital Product Data (VPD)
- Message Signaled Interrupt (MSI)
- Power Management (PMG)
- Compact PCI Hot Swap

Note These features are useful when using the 88F5182 on a PCI add-in card. They are not relevant when using the 88F5182 as the system host.

The VPD, MSI, PMG, and Hot Swap features are configured through the Capability List, as shown in Figure 13.

Figure 13: 88F5182 Capability List



6.11.1 Power Management

The 88F5182 implements the required configuration registers defined by the PCI specification for supporting system Power Management as well as the PCI_PMEn pin. This implementation is fully compliant with the specification.

Note For full details on system Power Management implementation, see the PCI specification.

The Power Management capability structure consists of the following fields:

- Capability structure ID. The ID of PMG capability is 0x1.
- Pointer to next capability structure.
- Power Management Capability.
- Power Management Status and Control.

Power management registers are accessible from the CPU or PCI. Whenever PCI configuration cycle updates Power State bits (<PState>bits [1:0] of the PCI Power Management Control and Status (Table 296 p. 350), the PM interrupt bit of the PCI Interrupt Cause (Table 276 p. 342) is set and an interrupt to the CPU is generated, if not masked by interrupt mask registers.

PCI_PMEn is an open drain output. When the CPU sets <PME_Status> bit to 1 in the PMCSR register, the 88F5182 asserts PCI_PMEn. It keeps asserting PCI_PMEn as long as the bit is set, and the <PME_En> bit is set to 1 in the PMCSR register. The PCI clears the <PME_Status> by writing 1, causing the de-assertion of PCI_PMEn.

PME0n and PME1n pins are multiplexed on the MPP pins. If PCI_PMEn support is required, first program the MPP pins to the appropriate configuration. (See the Multi Purpose Pins Multiplexing section in the 88F5182 *Feroceon*® Storage Networking SoC *Datasheet*).

Note

The 88F5182 does not support its own power down. It only supports a software capability to power down the CPU or other on board devices.

The PCI Power Management specification requires that a device configured to D1, D2, or D3 state, meet the following:

- The device does not respond to any PCI access other than configuration transactions.
- The device does not initiate transactions as a master. The 88F5182 software drivers must disable all internal DMAs so it can meet this requirement.
- Will not generate interrupts. The 88F5182 software drivers must mask PCI interrupts to meet this requirement.

When the state is changed from D3 back to D0, the device changes to the uninitialized state:

- Configuration registers are set to their initial values.
- Output buffers are not driven.

6.11.2 Vital Product Data (VPD)

VPD is information that uniquely identifies hardware elements of a system. VPD provides the system with information such as part number, serial number or any other information.

The PCI specification defines a method of accessing VPD. The 88F5182 VPD implementation is fully compliant with the specification. For full details on the VPD's structure, see the PCI specification.

The VPD's capability structure consists of the following fields:

- Capability structure ID. The ID of VPD capability is 0x3.
- Pointer to next capability structure.
- VPD Address. The 15-bit address of the accessed VPD structure.
- Flag. Used to indicate data transfer between VPD Data register and memory.
- VPD Data. The 32-bit VPD data written to memory or read from memory.

The 88F5182 supports a VPD located in function the 2 BAR0 space. PCI access to this VPD results in access to this space. Although the PCI specification defines the address to be accessed, as the VPD Address field in the VPD capability list item (15-bit address), the 88F5182 supports remapping

of the 17 high bits by setting the <VPDHighAddr> bits [24:8] in the PCI Address Decode Control (Table 243 p. 326) register to the required address.

For PCI VPD write, the PCI write VPD data first, then writes the VPD address with Flag bit set to 1. As a response, the slave writes the VPD data to the VPD device to the required address and clears the Flag bit as soon as the write is done.

For a PCI VPD read, the PCI writes VPD address with the Flag bit set to 0. As a response, the slave reads the VPD device from the required address, places the data in the VPD data field, and sets the Flag bit to 1. The VPD read is treated as a non-prefetched nor delayed read transaction.

6.11.3 Message Signaled Interrupt (MSI)

Note The MSI feature is useful for the 88F5182 to generate interrupt messages to the system host. There is no special logic for detecting MSI messages driven by external PCI devices to the local CPU.

The MSI feature enables a device to request an interrupt service without using interrupts. The device requests a service by writing a system specified message to a system specified address. The system software initializes the message destination and message during device configuration. The 88F5182 MSI implementation is fully compliant with the PCI specification. It supports a single interrupt message.

The MSI capability structure consists of the following fields:

- Capability structure ID: ID of MSI capability is 0x5
- Pointer to next capability structure
- Message Control
- Message Address: 32-bit message low address
- Message Upper Address: 32-bit message high address
- Message data: 15-bit of message data

Message Control word consists of the following fields in the PCI MSI Message Control (Table 299 p. 352):

- Bit [0]—MSI Enable. If set to 1, MSI is enabled, and the 88F5182 drives interrupt messages rather than asserting the PCI PCI_INTn pin.
- Bits [3:1]—Multiple Message Capable. Defines the number of DIFFERENT MSI messages the 88F5182 can drive.
- Bits [6:4]—Multiple Message Enable. Defines the number of DIFFERENT MSI messages the system allocates for the 88F5182.
- Bit [7]—64-bit address capable. Enables 64-bit addressing messages.

As soon as the PCI enables MSI (sets the <MSIEn> bit [16]), 88F5182 no longer asserts interrupts on the PCI bus. Instead, the PCI master drives a memory write transaction on the PCI bus, with address as specified in Message Address field and data as specified in the Message Data field.

If the Message Upper Address field is set to 0, the master drives a DWORD write, else it drives a DAC DWORD write.

Unlike the PCI PCI_INTn, a level sensitive interrupt that is active as long as there are active non-masked interrupts bits set, MSI is an edge like interrupt. However, to prevent the PCI interrupt handler from missing any new interrupt events, the 88F5182 continues to drive new MSI messages as long as pending, non-masked interrupts exist.

The <Timer>bit [15:0] of the MSI Trigger Timer (Table 251 p. 333) register defines the time gap (TCLK cycles) between sequential MSI requests. A timer starts counting with each new MSI request.

If it reaches 0 and there is still a pending non-masked interrupt, a new MSI request is triggered. If the PCI interrupt handler clears one of the Interrupt Cause register bits, and there is still a pending interrupt, the 88F5182 immediately issues a new MSI without waiting for the timeout to expire.

Setting the MSI Trigger Timer register to 0 disables the timer functionality (as if it was programmed to infinity). In this case, the PCI interrupt handler must confirm that there are no interrupt event is missed.

6.11.4 CompactPCI Hot Swap

The 88F5182 is CompactPCI Hot Swap ready compliant. It implements the required configuration registers defined by CompactPCI Hot Swap specification as well as three required pins.

The CompactPCI Hot Swap capability structure consists of the following fields:

- Capability structure ID. The ID of HS capability is 0x6.
- Pointer to next capability structure.
- Hot Swap Status and Control.

Hot Swap Status and Control register (HS_CSR) is accessible from both CPU and PCI. This register's bits give status of board insertion/extraction as defined in the specification.

HS_CSR bits are:

- EIM: PCI_ENUMn Interrupt Mask. If set to 1, the 88F5182 does not assert a PCI_ENUMn interrupt.
- LOO: LED On/Off. If set to 1 LED is on.
- REM: Removal. Indicates board is about to be extracted.
- INS: Insertion. Indicates board has been inserted.

The 88F5182 device supports three Hot Swap ready required pins.

- PCI_HS: Handle Switch input pin. Indicates insertion or extraction of board. A 0 value indicates the handle is open.
- PCI_LED: LED control output pin. A 1 value activates the on board LED.
- PCI_ENUMn: Open drain output. Asserted upon board insertion or extraction (if not masked by EIM bit).

Board extraction consists of the following steps:

1. The operator opens board ejector handle. As a result, PCI_HS goes LOW, indicating board is about to be extracted.
2. As a result, the REM bit is set and the PCI_ENUMn pin is asserted, if not masked by EIM bit.
3. The System Hot Swap software detects PCI_ENUMn assertion. Checks the REM bits in all Hot Swap compliant boards. Identifies the board about to be extracted and clears the REM bit (by writing a 1 value).
4. The 88F5182 acknowledges the system software by stop asserting the PCI_ENUMn pin.
5. The Hot Swap software might re-configure the rest of the boards, and when ready, it sets the LOO bit, indicating board is allowed to be removed.
6. As a result, 88F5182 drive PCI_LED pin to 1, the on board LED is turned on indicating that the operator may remove the board.

Board insertion consists of the following steps:

Board is inserted. It is powered from Early Power and its reset is asserted from Local PCI PCI_RSTn. The on board LED is turned on by the hardware (not as a result of LOO bit state).

1. Local PCI PCI_RSTn is de-asserted, causing the LED to turn off, indicating that the operator may lock the ejector handle.
2. The operator locks the handle. As a result, PCI_HS goes HIGH, indicating the board is inserted and locked.

3. As a result, the INS bit is set and PCI_ENUMn is asserted, notifying the Hot Swap software that a board has been inserted.
4. System Hot Swap software detects PCI_ENUMn assertion, checks the INS bits in all Hot Swap compliant boards, identifies the inserted board and clears the INS bit (by writing a value of 1).
5. 88F5182 acknowledges system software by stopping to assert the PCI_ENUMn pin. Now the software may re-configure all the boards.

Note

For full details about the Hot Swap process and board requirements, see the CompactPCI Hot Swap specification.

In addition, the 88F5182 supports the following Hot Swap device requirements:

- All PCI outputs floats when PCI_RSTn is asserted.
- All 88F5182 PCI state machines are kept in their idle state while PCI_RSTn is asserted.
- The 88F5182 PCI interface maintains its idle state until the PCI bus is in an IDLE state. If reset is de-asserted in the middle of a PCI transaction, the PCI interface stays in its idle state until the PCI bus is back in idle.
- The 88F5182 has no assumptions on clock behavior prior to its setup to the rising edge of PCI_RSTn.
- The 88F5182 is tolerant of the 1V pre-charge voltage during insertion.
- The 88F5182 can be powered from Early VDD.

6.11.5 Built-In Self Test (BIST)

The 88F5182 supports BIST functionality as defined by the PCI specification. It does not run its own self test. Instead, it enables the PCI to trigger CPU software self test.

The PCI BIST, Header Type/Initial Value, Latency Timer, and Cache Line (Table 284 p. 346) register is located at offset 0x0C of Function 0 configuration header. It consists of the following fields:

- <BISTCap> bit (bit [31]). If BIST is enabled through reset initialization, it is set to 1. This bit is read only from the PCI.
- <BISTAct> bit (bit [30]). Set to 1 by the PCI to trigger CPU software self test. Cleared by the CPU upon test finish.
- <BISTComp> (bits [27:24]). Written by the self test software upon test finish. Any value other than 0 stands for test fail.

Upon PCI triggering of BIST (writing 1 to bit [31]), the CPU interrupt is asserted (if not masked) and the CPU must run the system self test. When the test is completed, the CPU must clear bit [6] and write the completion code.

The PCI specification requires that BIST be completed in two seconds. It is the BIST software responsibility to meet this requirement. If bit [31] is not cleared within two seconds, the PCI BIOS may treat it as a BIST failure.

Note

The 88F5182 does not runs its own self test. The BIST register implementation is just a software hook for the CPU to run a system self test.

6.11.6 Expansion ROM

With the Expansion ROM enabled through PCI Mode (Table 247 p. 331), the 88F5182 configuration space includes an expansion ROM BAR at offset 0x30 of function0 configuration space as specified in the PCI specification. Like the other BARs, there are expansion ROM size and remap registers. Address decoding is done the same way as for the other devices. A hit in the expansion ROM BAR results in an access to function 2 BAR0 space.

Note Expansion ROM size must not exceed function 2 BAR0 size.

With the Expansion ROM disabled, the 88F5182 does not support expansion ROM BAR; offset 0x30 in the configuration space is reserved.

If using expansion ROM, the <ExpROMEn> bit [0] of the PCI Expansion ROM Base Address Register (Table 292 p. 349) must be set to 1 (via the local processor or serial ROM initialization), prior to BIOS access to the device. For the PCI slave to respond to a PCI address hit in the expansion ROM space, the system software must set bit [1], the <MEMEn> field in the PCI Status and Command (Table 282 p. 344) to 1 and <ExpROMEn> bit [0] of PCI Expansion ROM Base Address Register to 1, as defined in PCI specification.

6.12 PCI Clocking

The 66 MHz PCI AC specification requires a 6 ns output delay for all PCI signals. To meet this requirement, the 88F5182 PCI interface uses an internal DLL.

The PCI specification permits using a DLL only when interfacing a 66 MHz PCI bus. The 88F5182 samples the PCI_M66EN pin on reset de-assertion to determine if it is connected to a 66 MHz bus. If the PCI_M66EN pin is sampled low, it means that it is interfacing with a 33 MHz bus. The PCI interface DLL is bypassed, and the 88F5182 meets the AC requirements of a 33 MHz PCI bus.

7 SATA II Interface

This section provides technical information about the Serial-ATA (SATA) II interface.

Based on the Marvell® SATA host controllers (SATAHC) and SATA proven technology, the 88F5182 is fully compatible with SATA II phase 1.0 specification (Extension to SATA I specification).

The 88F5182 employs the latest SATA II PHY technology, with 3.0 Gbps (Gen2i) and is backwards compatible with 1.5 Gbps (Gen1i) SATA I.

The Marvell 88F5182 SATA II PHY has the following features:

- SATA II 3 Gb/s speed
- Backwards compatible with SATA I PHYs and devices
- Support Spread Spectrum Clocking (SSC)
- Programmable PHY for industry leading backplane drive capability
- SATA II power management compliant
- SATA II Device Hot-Swap compliant
- Low power consumption — Less than 200 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

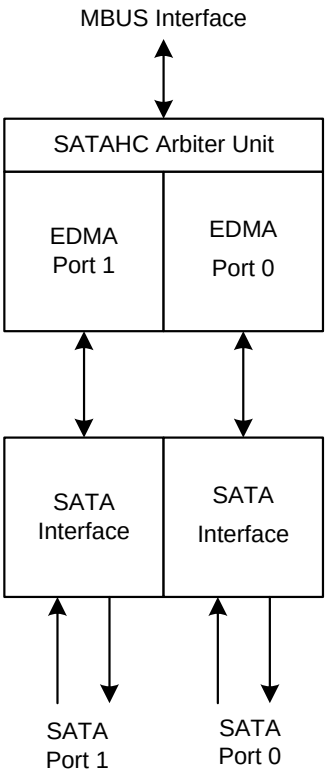
8 Serial-ATA II Host Controller (SATAHC)

The 88F5182 incorporates a Serial-ATA (SATA) host controller (SATAHC) integrating two independent SATA ports. A dedicated Enhanced DMA (EDMA) controls each port.

8.1 SATAHC Block Diagram

The 88F5182 SATAHC consists of an arbiter, two EDMAs, and two SATA ports. Both EDMAs are independent and may work concurrently (see Figure 14).

Figure 14: SATAHC Block Diagram



8.1.1 SATAHC Arbiter

The SATAHC arbiter:

- Performs arbitration between the two EDMAs and the Crossbar interface.
- Performs address decoding of the transactions from the crossbar to the EDMAs.
- Contains the logic common to all channels.
- Contains the registers common to all channels.

8.1.2 SATAHC EDMA

Each SATAHC EDMA:

- Controls the ATA transactions associated with its port.
- Contains a 1-KB buffer for posted write and prefetch read transactions.
- Contains the registers that control the EDMA operation.

8.1.3 SATA Interface

The SATA interface is compliant with the Serial-ATA II Phase 1.0 specification (Extension to SATA I specification). See Section 7, SATA II Interface, on page 69.

8.1.4 Unused SATA Ports

To save power the unused SATA ports can be shut down by setting the appropriate <PhyShutdown> field in the Serial-ATA Interface Configuration Register (Table 379 p. 397) to 1.

8.2 Host Direct Control Over the Hard Disk Drive

When the EDMA is disabled, the <eEnEDMA> field in the EDMA Command Register (Table 351 p. 379) is cleared. The host has direct control over the device through the ATA task registers (see Table 322, Shadow Register Block Registers Map, on page 362).

When the EDMA is enabled, the <eEnEDMA> field 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.

8.3 LED Indications

For each SATA port there are two LED indications:

- Disk present indication
- Disk active indication

These LED indications are selected through the MPP interface.

Optionally, by setting the GPIO Blink Enable Register (Table 594 p. 513), the LED indications for both the SATA and GPIO LEDs may be set to blink.

8.4 EDMA Operation

Although the SATAHC contains two EDMAs, this document describes the operation of a single EDMA within the SATAHC. See Figure 14, SATAHC Block Diagram, on page 70 and refer to Appendix A.8, Serial-ATA Host Controller (SATAHC) Registers, on page 358.

The interface between the host CPU and each EDMA consists of two queues—the request queue and the response queue. The request queue is the interface used by the host CPU to queue ATA DMA commands as a request between the system memory and the device. The response queue is the interface used by the EDMA to notify the host CPU that a data transaction between the system memory and the device has been completed. Each entry in the request queue consists of an ATA DMA command and the EDMA parameters and descriptors used 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 has been completed. This all occurs without CPU intervention. Direct access to the device is also supported for device initialization and error handling.

8.4.1 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 or 128 entries, circular queues (FIFO) whose location is configured by the Queue In-Pointer and the Queue Out-Pointer entries. The entry length is set using <eEDMAQueLen> field in the EDMA Configuration Register (Table 341 p. 372)—for 32 entries <eEDMAQueLen>=0, for 128 entries field <eEDMAQueLen>=1. 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 used by the CPU software 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 used to initiate the device and to perform the data transaction.

The response queue is the interface used by the EDMA to notify the CPU software that a data transaction between the system memory and the device has been completed. Each entry in the response queue is 8 bytes in length, consisting of the command tag and the response flags.

Figure 15: Command Request Queue—32 Entries

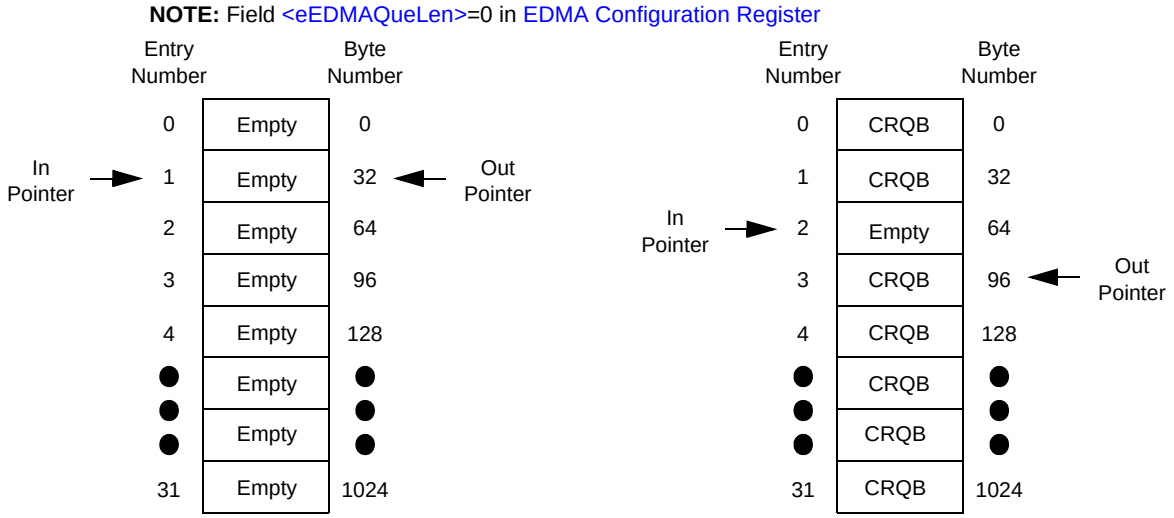


Figure 16: Command Response Queue—32 Entries

NOTE: Field <eEDMAQueLen>=0 in EDMA Configuration Register

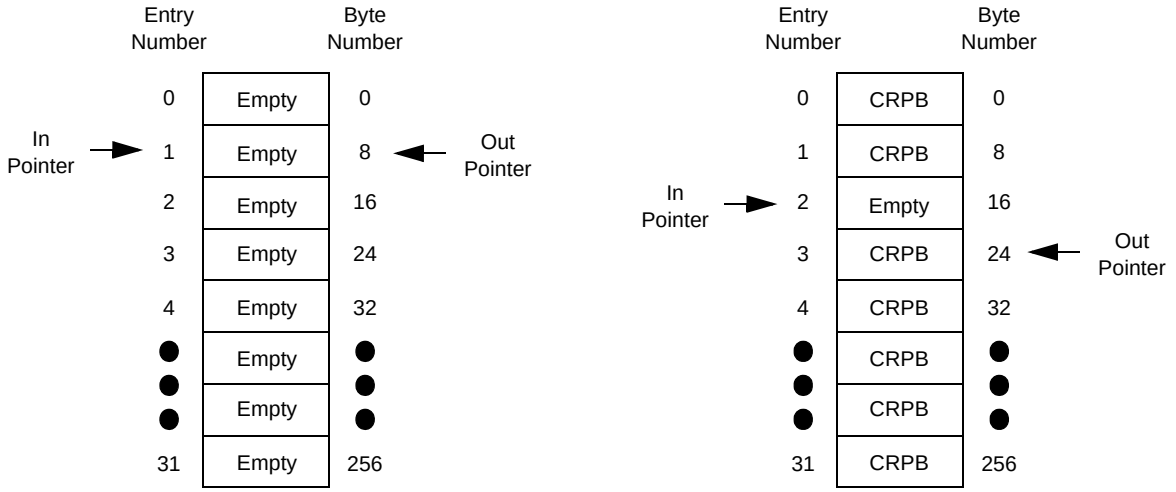


Figure 17: Command Request Queue—128 Entries

NOTE: Field <eEDMAQueLen>=1 in EDMA Configuration Register

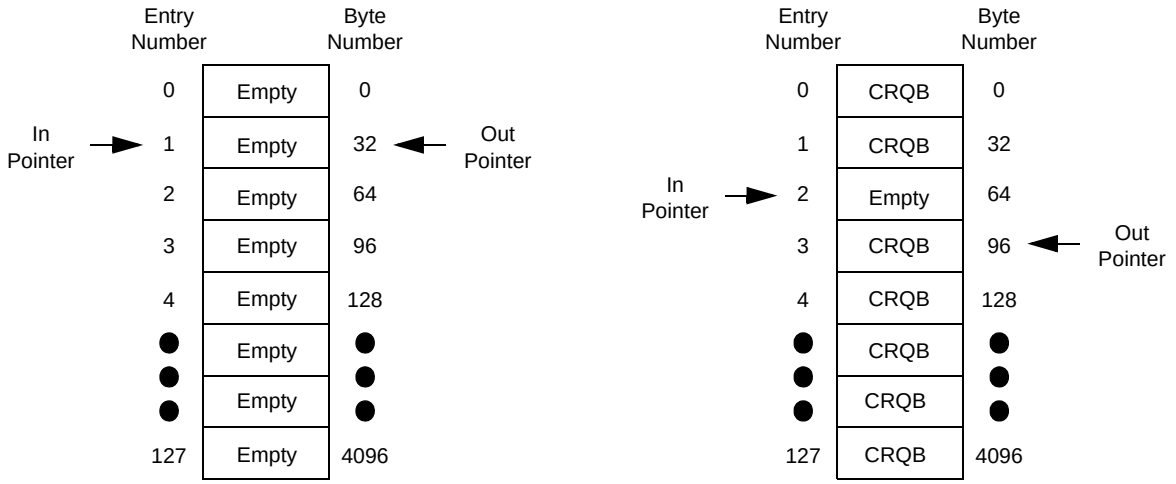
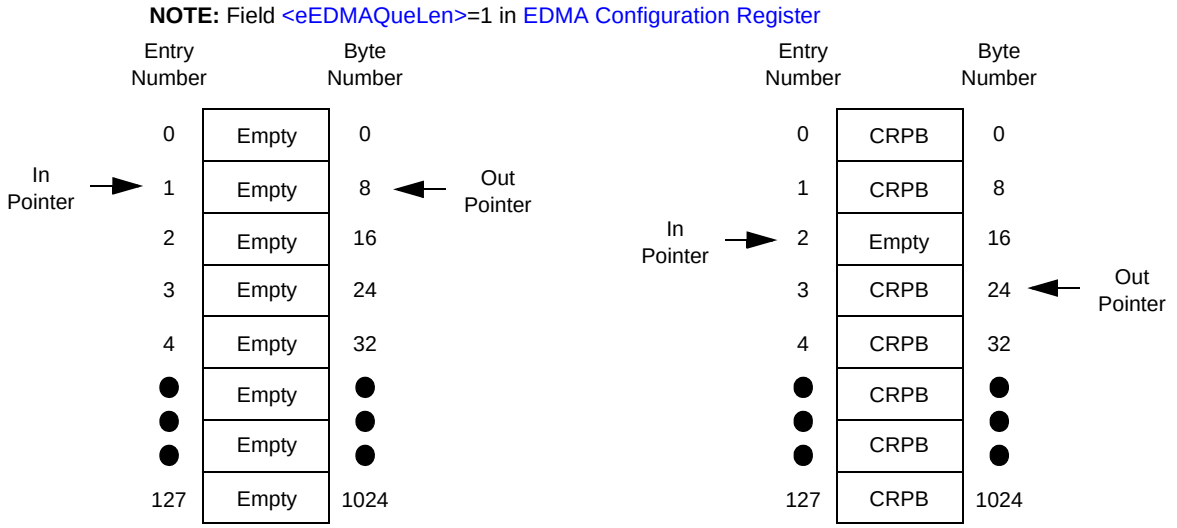


Figure 18: Command Response Queue—128 Entries



8.4.2 EDMA Configuration

The EDMA configuration is determined according to EDMA Configuration Register (Table 341 p. 371). The following registers may be changed only when the <eEnEDMA> field in the EDMA Command Register (Table 351 p. 379), is cleared, and the EDMA is disabled. These registers must not be changed when <eEnEDMA> is set.

- In the SATAHC Address Space (Table 320 p. 358):SATAHC Configuration Register
- In Section A.8.5, EDMA Registers
  - EDMA Configuration Register
  - EDMA Command Delay Threshold Register
- All registers in Section A.8.3, Shadow Registers Block Register Map, on page 362, except that the host is allowed to change the <HOB> bit (bit [7]) in the ATA Device Control register (offset 0x82120) while the EDMA is active.
- All registers in Section A.8.6, Basic DMA Registers
- All registers in Serial-ATA Interface Registers (Table A.8.7 p. 388)
  - <FIS Interrupt Cause Register>
  - <FIS Interrupt Mask Register>

8.4.3 EDMA Mode of Operation

8.4.3.1 Basic DMA Operation

When <eEnEDMA> 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:

- Basic DMA Command Register (Table 361 p. 384)
- Basic DMA Status Register (Table 362 p. 386)
- Descriptor Table Low Base Address Register (Table 363 p. 387)
- Descriptor Table High Base Address Register (Table 364 p. 387)
- SATAHC Interrupt Cause Register (Table 328 p. 365)

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 <DMAxDone> in SATAHC Interrupt Cause Register (Table 328 p. 365) and an interrupt is generated. If an error occurs during execution of the data transfer, the EDMA Interrupt Error Cause Register (Table 343 p. 374) is updated with the error cause, the Basic DMA Status Register (Table 362 p. 386) 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.
4. Initializes the Descriptor Table High Base Address Register.
5. Makes sure the <eEarlyCompletionEn> field in the EDMA Configuration Register (Table 341 p. 372) is cleared to 0.
6. If Port Multiplier is used, initializes the <PMportTx> field in the Serial-ATA Interface Control Register (Table 380 p. 398).
7. Activates the Basic DMA by setting the control bits in the Basic DMA Command Register (Table 361 p. 384).

Basic DMA Read/Write Operation

The Basic DMA performs only the data transaction.

1. The Basic DMA performs the data transaction.
2. It sets field <DMAxDone> 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 (Table 361 p. 384).
2. If the <Start> field is cleared while the Basic DMA is still active, as indicates by the active bit in the Basic DMA Status Register (Table 362 p. 386), the Basic DMA command is aborted, and the data transferred may be discarded before reaching its destination.

8.4.3.2 Target Mode Operation

When the <eEnEDMA> field is cleared to 0 and the <ComChannel> field in the Serial-ATA Interface Configuration Register (Table 379 p. 397) is set to 1, a communication channel is opened with Serial-ATA port of another 88F5182 (or any other Marvell® devices that support target mode). The communication channel is not symmetric: one side should be configured as an initiator (the <TargetMode> field in the Serial-ATA Interface Configuration Register (Table 379 p. 397) is set to 0) while the other side is configured as a target (field <TargetMode> is set to 1).

Full Communication

The two channel should be used as follows:

1. In channel A—The SATA port in the 88F5182 is configured as the initiator, while the companion SATA port in the other 88F5182 is configured as the target.
2. In channel B—The SATA port in the 88F5182 is configured as the target, while the companion SATA port in the other 88F5182 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 8.6, Vendor Unique, on page 94).
3. Target Transport—Update ATA task registers and set the port's <SaDevInterruptx> field in the SATAHC Interrupt Cause Register (Table 328 p. 365), 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 and optionally set the port's <SaDevInterruptx> field in the SATAHC Interrupt Cause Register and generate the interrupt if specified in the Register Device to the Host FIS.
6. Target Host—Activate Basic DMA, set the <eDMAActivate> field in the Serial-ATA Interface Control Register (Table 380 p. 399).
7. Target Transport—Send the data as configured in the target Basic DMA.
8. Initiator Basic DMA—Set the port's <DMAxDone> field and generate the interrupt to the initiator host when data transfer completes.
9. Target Basic DMA—Set the port's <DMAxDone> 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 8.6, Vendor Unique, on page 94).
3. Target Transport—Update ATA task registers and set the port's <SaDevInterruptx> field 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 <SaDevInterruptx> 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 field <eDMAActivate>.
8. Initiator Transport—Send the data as configured in the initiator Basic DMA.
9. Initiator Basic DMA—Set the port's <DMAxDone> field and generate the interrupt to initiator host when the data transfer completes.
10. Target Basic DMA—Set the port's <DMAxDone> 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 322, Shadow Register Block Registers Map, on page 362).
- Link Errors while transmitting Vendor Unique FIS are also reported in the <LinkCtlTxErr> field in the EDMA Interrupt Error Cause Register (Table 343 p. 376).
- For testing, this mode can be used to generate an External Loopback between the Serial-ATA ports of the same 88F5182.

8.4.3.3 Non-queued DMA Commands

When the <eEnEDMA> field in the EDMA Command Register (Table 351 p. 379) is set to 1, the <eSATANatv CmdQue> field in the EDMA Configuration Register (Table 341 p. 371) is clear to 0 and the <eQue> field in that same register 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

8.4.3.4 Queued DMA Commands

When field <eEnEDMA> is set to 1, field <eSATANatv CmdQue> is clear to 0 and field <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 ATA 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

8.4.3.5 SATA Native Command Queuing

When <eEnEDMA> is set to 1 and <eSATANatv CmdQue> 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

8.4.4 EDMA Activation

The CPU activates the EDMA according to the following flow:

1. Verifies that the device is ready to receive data commands, the field <DET> field in the SStatus Register (Table 367 p. 388) = 3, and the fields <Busy> and <DRQ> in the device Status register are cleared.
2. Clears the EDMA Interrupt Error Cause Register (Table 343 p. 374) and clears the appropriate <SaCrpbXDone> field in the SATAHC Interrupt Cause Register (Table 328 p. 365).
3. Initializes the EDMA Configuration Register (Table 341 p. 371).
4. Clears the FIS Interrupt Cause Register (Table 385 p. 404).
5. Initialized the FIS Configuration Register (Table 384 p. 403).
6. Initializes the EDMA Request Queue In-Pointer Register (Table 346 p. 377).
7. Initializes the EDMA Request Queue Out-Pointer Register (Table 347 p. 378).
8. Initializes the EDMA Response Queue In-Pointer Register (Table 349 p. 378).
9. Initializes the EDMA Response Queue Out-Pointer Register (Table 350 p. 379).
10. Activates the EDMA by writing 1 to field <eEnEDMA>.

Note

While the EDMA is enabled, the host should not access the registers listed in Section 8.4.2, EDMA Configuration, on page 74.
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 field <eEnEDMA>—will result in unpredictable behavior.

8.4.5 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, to enable EDMA access to the new CRQBs in the request queue.

8.4.6 Stop EDMA

To stop the EDMA operation, the CPU should set the <eDsEDMA> field in the EDMA Command Register (Table 351 p. 380) to 1. The EDMA stops queue processing, aborts the current command and clears field <eEnEDMA>.

Note

If EDMA is aborted during commands processing, the host must set field <eAtaRst> to recover.

8.4.7 Restart EDMA

To restart the queue, the CPU must follow the EDMA activation flow. See Section 8.4.4, EDMA Activation, on page 78.

8.4.8 Device Link Disconnect

Note Since loss of the link may occur at any time during EDMA programming, when the CPU receives a link-down interrupt, it must wait for reestablishment of the link (see Section 8.4.9, Device Link Connect, on page 79).

See "Device Disconnect" on page 82 for the disconnect procedure.

8.4.9 Device Link Connect

When the link to the device is renewed, the EDMA sets the <eDevCon> field in the EDMA Interrupt Error Cause Register (Table 343 p. 374).

Device hard reset (setting field <eAtaRst>) and device initialization are required before any attempt to access the device.

8.4.10 EDMA Read Burst Limit

The <eRdBSz> field and the <eRdBSzExt> field in the EDMA Configuration Register (Table 341 p. 371), define the maximum burst read transactions SATAHC initiates towards the crossbar. The EDMA supports a maximum read burst size of 128B.

8.4.11 EDMA Write Burst Limit

The EDMA support a maximum write burst size of 128B.

8.4.12 Port Multiplier Support

The 88F5182 supports the Port Multiplier (PM) ingredient in the following modes:

8.4.12.1 Port Multiplier—Command Based Switching

When the <eEDMAFBS> field in the EDMA Configuration Register (Table 341 p. 372) 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 Queued DMA commands (see Section 8.4.3.3, Non-queued DMA Commands, on page 77) are supported.

8.4.12.2 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 8.4.3.3, Non-queued DMA Commands, on page 77)
- Tag Command Queuing (TCQ) commands (see Section 8.4.3.4, Queued DMA Commands, on page 77)
- Native Command Queuing commands (see Section 8.4.3.5, SATA Native Command Queuing, on page 77)

Note

The mode is selected before EDMA is enabled. It must not be changed when bit <eEnEDMA> is set to 1.

8.4.13 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 (Table 384 p. 403):

If bit [1] in the <FISWait4Host RdyEn> field is cleared to 0, the 88F5182 ignores the FIS.

If bit <FISWait4Host RdyEn>[1] is set to 1:

- Bit [1] in the <FISWait4Host RdyEn> field is set. to 1
- The <eTransInt> field in the EDMA Interrupt Error Cause Register (Table 343 p. 375) is also set if the corresponding bit in the FIS Interrupt Mask Register (Table 386 p. 406) is set to 1.

8.4.14 EDMA Interrupts

8.4.14.1 Error indication

The EDMA Interrupt Error Cause Register, provides the various error indications that may occur during DMA operation. For more information, see Section 8.4.15, Error Handling, on page 81.

In addition, each DMA contains a EDMA Interrupt Error Mask Register (Table 344 p. 377). This register may be used to mask the error bits in the EDMA Interrupt Error Cause Register. If one (or more) of the unmasked bits in the EDMA Interrupt Error Cause Register is set, an error indication is propagated to the SATAHC Main Interrupt Cause Register (Table 330 p. 366).

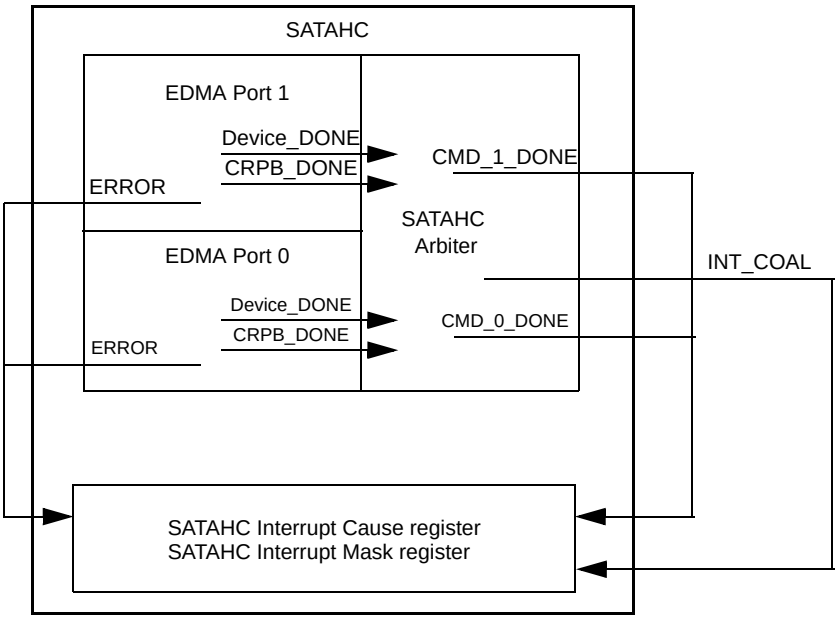
8.4.14.2 Command Completion Indication

The SATAHC Interrupt Cause Register (Table 328 p. 365), 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.

When the EDMA completes an ATA transaction:

- The last data leaves the 88F5182.
- The CRPB is updated.
- The EDMA indicates the appropriate bit in the SATAHC Interrupt Cause Register, and
- An interrupt indication is propagated to SATAHC Main Interrupt Cause Register.

Figure 19: EDMA Interrupt Hierarchy[1]



8.4.14.3 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 88F5182 provides an interrupt coalescing mechanism that sets the interrupt coalescing bit in the SATAHC Interrupt Cause Register (Table 328 p. 365), and propagates an interrupt indication if one of the following is true:

- The number of EDMA commands per the two SATA channels reached the SATAHC interrupt coalescing threshold value (see the SATAHC Interrupt Coalescing Threshold Register (Table 326 p. 364).
- At least one EDMA commands per the two SATA channels 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 327 p. 364)).

8.4.14.4 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 328 p. 365) and a command completion indication is propagated to the SATAHC Main Interrupt Cause Register.

8.4.15 Error Handling

Error indications from all layers are gathered in EDMA Interrupt Error Cause Register (Table 343 p. 374).

8.4.15.1 List Of Unrecoverable Errors

These unrecoverable errors are in the EDMA Interrupt Error Cause Register

- <eDevDis> field in the EDMA Interrupt Error Cause Register (Table 343 p. 374)

1. All interrupt indications from the SATAHC are propagated to the SATAHC Main Interrupt Cause Register (Table 330 p. 366).

- <eIORdyErr>
- Bit [2] of the <LinkCtlRxErr> field
- <LinkDataRxErr>
- <LinkDataTxErr>
- <TransProtErr>

When an unrecoverable error indication is set from the list above, the EMDA is self disabled and the host must set the <eAtaRst> field in the EDMA Command Register (Table 351 p. 380) to recover.

8.4.15.2 PHY Layer Errors

SError Register Errors

For PHY layer errors, see SError Register (Table 368 p. 389).

The <SerrInt> field in the EDMA Interrupt Error Cause Register (Table 343 p. 374) is set when at least one bit in SError Register (Table 368 p. 389) is set to 1, and the corresponding bit in SError Register is enabled.

Device Disconnect

When the device is disconnected, the EDMA halts and:

- Sets the <eDevDis> field in the EDMA Interrupt Error Cause Register (Table 343 p. 374).
- Disables the EDMA operation by clearing <eEnEDMA>.
- Sets the <eSelfDis> field in the EDMA Interrupt Error Cause Register (Table 343 p. 375).

Note

- Host must set <eAtaRst> to recover.
- The CPU is responsible for error recovery.

8.4.15.3 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. Bit [0] in the <LinkCtlRxErr> field in the EDMA Interrupt Error Cause Register (Table 343 p. 375) is set in.
  - Internal FIFO error occurs. Bit [1] in the <LinkCtlRxErr> field is set.
  - Link state errors, coding errors, or running disparity errors occur. Bit [3] in 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 <eAtaRst> to recover.
  - The Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. bit [2] of 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] of the <LinkDataRxErr> field and the <TransProtErr> field are set. The host must set <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] in 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 <LinkCtlTxErr> field is set.
- Internal FIFO error occurs. Bit [1] of the <LinkCtlTxErr> field is set in the EDMA Interrupt Error Cause Register.
- 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] of the <LinkCtlTxErr> field is set.
- Internal FIFO error occurs. Bit [1] of the <LinkCtlTxErr> field is set.
- The Link Layer is reset (to Idle state) by the reception of SYNC primitives from the device. Bit [2] of the <LinkCtlTxErr> field is set.
- Link layer accepts DMAT primitive from the device. Bit [3] of the <LinkCtlTxErr> field is set.
- FIS transmission is aborted due to collision with received traffic. Bit [4] of the <LinkCtlTxErr> field is set.

8.4.15.4 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 field <TransProtErr> and the EDMA is self disabled. Host must set <eAtaRst> to recover. This error state can arise from invalid or poorly formed FISs being received, from invalid state transitions, or from other causes.

Note

- 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

Note

FIS-Based Switching is disabled when field <eEDMAFBS> field in the EDMA Configuration Register (Table 341 p. 372) is cleared.

See Section 8.4.3.3, Non-queued DMA Commands, on page 77 and Section 8.4.3.4, Queued DMA Commands, on page 77.

Bit [2] in the <eHaltMask> field in the EDMA Halt Conditions Register (Table 356 p. 383) field 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", see page 362
  - "Serial-ATA Interface Status Register", see page 400
  - "EDMA Status Register", see page 381
  - "EDMA Interrupt Error Cause Register", see page 374
- Bit [2] in the EDMA Interrupt Error Cause Register is set.
- The EDMA halts.

Device Error Indication in Serial-ATA Native Command Queuing

See Section 8.4.3.5, SATA Native Command Queuing, on page 77 and Section 8.4.12.2, Port Multiplier—FIS-Based Switching, on page 79

Bit [2] in the <eHaltMask> field should be set to 1 in the EDMA Halt Conditions Register (see Table 341 on page 371):

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", see page 400
- "EDMA Status Register", see page 381
- "EDMA Interrupt Error Cause Register", see page 374
- The host identifies which drive caused the error via the <PortNumDevErr> field in the Serial-ATA Interface Test Control Register (Table 381 p. 400)

EDMA does NOT update CRPB with the error indication.

The host must:

- Wait for completion of all outstanding commands associate to other devices (that did not experience a device error).
- Check the <eCacheEmpty> field in the EDMA Status Register (Table 353 p. 381). If cleared, then wait for another Device error interrupt.
- Wait for clear of the <EDMAIdle> field in the EDMA Status Register (Table 353 p. 382). If set, then wait for another Device error interrupt. Good CRPBs may be received.
- Set the <eDsEDMA> field in the EDMA Command Register (Table 351 p. 380) to disable EDMA operation.
- Wait for clearing of the <eEnEDMA> field.
- 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 8.4.3.3, Non-queued DMA Commands, on page 77, Section 8.4.3.4, Queued DMA Commands, on page 77 and Section 8.4.12.2, Port Multiplier—FIS-Based Switching, on page 79

Bit [2] in the <eHaltMask> field should be cleared to 0:

Bits <FISWait4Host RdyEn>[0] should be set to 1 in the FIS Configuration Register (Table 384 p. 403).

When bit <Error> 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] in the <FISWait4HostRdy> field in the FIS Interrupt Cause Register (Table 385 p. 405) field 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", see page 374.
- The EDMA updates the CRPB with the error indication.

The host must:

- Get detailed error information from the Shadow Register Block, see page 158 (see Table 322, Shadow Register Block Registers Map, on page 362).
- Detect the aborted commands for the disk that experience error according to EDMA NCQ0 Done/TCQ0 Outstanding Status Register, EDMA NCQ1 Done/TCQ1 Outstanding Status Register, EDMA NCQ2 Done/TCQ2 Outstanding Status Register and EDMA NCQ3 Done/TCQ3 Outstanding Status Register.
- Clear the <eDevErr> field in the EDMA Interrupt Error Cause Register (Table 343 p. 374).
- Clear bit [0] in the <FISWait4HostRdy> field.
- Wait for completion of all outstanding commands (that is, any commands to the EDMA that did not complete successfully and did not abort or fail).
- Set the <eDsEDMA> field to disable the EDMA operation.
- Wait for clear of the <eEnEDMA> [1].

8.4.15.5 DMA Errors

Internal Parity Error

The 88F5182 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.

8.4.16 EDMA Data Structures

8.4.16.1 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 <eEDMAQueLen> field in the EDMA Configuration Register (Table 341 p. 372) = 0) or 128 entries (field <eEDMAQueLen>=1). The request queue is a circular queue (FIFO) whose location is configured by the EDMA Request Queue In-Pointer Register (Table 346 p. 377), and the EDMA Request Queue Out-Pointer Register (Table 347 p. 378).

- 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 128/32 entries (as configured in field <eEDMAQueLen>).

- 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 15, Command Request Queue—32 Entries, on page 72 and Figure 17, Command Request Queue—128 Entries, on page 73.

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 13, p.86 provides a map of the CRQB data structure registers.

8.4.16.2 EDMA Command Request Block (CRQB) Data

Table 13: EDMA CRQB Data Structure Map

Register	Offset	Page
CRQB DW0—cPRD Descriptor Table Base Low Address	Offset: 0x00	Table 14, p. 86
CRQB DW1—cPRD Descriptor Table Base High Address	Offset: 0x04	Table 15, p. 87
CRQB DW2—Control Flags	Offset: 0x08	Table 16, p. 87
CRQB DW3—Data Region Byte Count	Offset: 0x0C	Table 17, p. 88
CRQB DW4—ATA Command	Offset: 0x10	Table 18, p. 88
CRQB DW5—ATA Command	Offset: 0x14	Table 19, p. 88
CRQB DW6—ATA Command	Offset: 0x18	Table 20, p. 89
CRQB DW7—ATA Command	Offset: 0x1C	Table 21, p. 89

Table 14: 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 15: CRQB DW1—cPRD Descriptor Table Base High Address
Offset: 0x04

Bits	Field	Description
31:0	cPRD[63:32]	CRQB ePRD. When <cPRDMode> is cleared to 0: This DWORD contains bits [63:32] of the physical starting address of a PRD table in system memory. When <cPRDMode> is set to 1: This DWORD contains bits [63:32] of the physical starting address of a data region in system memory. Must be set to 0.

Table 16: 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.
23:17	cHostQueTag	CRQB Host Queue Tag This 7-bit field contains the host identification of the command.
31:24	Reserved	Reserved

Table 17: 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

Note 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 18: 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 322 on page 362).
31:24	Features	This field contains the contents of the Features (Features Current) register of the Shadow Register Block.

Table 19: 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 322 on page 362).
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 20: 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 322 on page 362).
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 21: 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 322 on page 362).
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

Non-Queued Mode

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

Queued Mode

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

Caution Other commands cause unpredictable results!

8.4.16.3 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 22: 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 23: 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 24: 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 set to 0x0.

Table 25: ePRD DWORD 3

Bits	Field	Description
31:0	Reserved	Reserved

8.4.16.4 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 128/32 entry, circular queue (FIFO) whose location is configured by the EDMA Response Queue In-Pointer Register (Table 349 p. 378) and the EDMA Response Queue Out-Pointer Register (Table 350 p. 379).

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 128/32 entries (as configured in the <eEDMAQueLen> field in the EDMA Configuration Register (Table 341 p. 372)).
- 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 16, Command Response Queue—32 Entries, on page 73 and Figure 18, Command Response Queue—128 Entries, on page 74.

Each 8-byte command response entry consists of command ID, response flags, and a timestamp, see Table 26, “EDMA CRPB Data Structure Map,” on page 92. The CRPB data structure, described in Table 13, “EDMA CRQB Data Structure Map,” on page 86, is written by the EDMA.

8.4.16.5 EDMA Command Response Block (CRPB) Data

Table 26 provides a map of the EDMA command response block data structure tables.

Table 26: EDMA CRPB Data Structure Map

Register	Offset	Table, Page
CRPB ID Register	Offset: 0x00	Table 27, p. 92
CRPB Response Flags Register	Offset: 0x02	Table 28, p. 92
CRPB Time Stamp Register	Offset: 0x04	Table 29, p. 93

Table 27: CRPB ID Register
Offset: 0x00

Bits	Field	Description
6:0	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 16, CRQB DW2—Control Flags, on page 87.
15:7	Reserved	Reserved

Table 28: 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 (see Table 343 on page 374) 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 29: CRPB Time Stamp Register
Offset: 0x04

Bits	Field	Description
31:0	cTS	CRPB TS When the command is completed, the content of the EDMA Timer Register (see Table 342 on page 374) is written into this field. This data may be used to estimate the command execution time.

8.5 BIST

8.5.1 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

8.5.2 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 8.6, Vendor Unique, on page 94.
- Set the <BISTMode> field in the BIST Control Register (Table 376 p. 395) to 1 to determine FIS direction.
- Initiate the <BISTPattern> field according to the transmitted BIST Activate FIS.
- Initiate the BIST-DW1 Register (Table 377 p. 395) according to the transmitted BIST Activate FIS.
- Initiate the BIST-DW2 Register (Table 378 p. 396) according to the transmitted BIST Activate FIS.
- Set the <BISTEn> field in the BIST Control Register (Table 376 p. 395) in the BIST Control Register (Table 376 p. 395) to activate the pattern comparator operation.
- Read <BISTResult> status to determine BIST test passes or not.
- Set the <eAtaRst> field in the EDMA Command Register (Table 351 p. 380) to exit both sides of the link from BIST mode.

8.5.3 BIST as the Target Side

The following flow should be performed, when the Serial-ATA port receives BIST Activate FIS:

- Host must set bit [1] of the <FISWait4Rdy> field in the FIS Interrupt Cause Register (Table 385 p. 404) to 1.
- FIS content is updated in FIS DW0 Register through FIS DW6 Register:
- Bit [3] in field <FISWait4HostRdy> in is set.
- Bit [8], the <eTransInt> field in the EDMA Interrupt Error Cause Register (Table 343 p. 375) is also set if the corresponding bit in FIS Interrupt Mask Register is set to 1.
- Host CPU sets field <BISTMode> to 0 in the to determine FIS direction.
- Host CPU initiates field <BISTPattern> according to the received BIST Activate FIS.
- Host CPU initiates the BIST-DW1 Register with the contents matching the received BIST Activate FIS.

- Host CPU initiates the BIST-DW2 Register with the contents matching the received BIST Activate FIS.
- Host CPU sets field <BISTEn> to activate the internal pattern generators to send the data stream onto the Serial-ATA link.

8.6 Vendor Unique

8.6.1 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 <eDsEDMA>.
3. Verify the EDMA is disable, <eEnEDMA> is cleared.
4. Verify the Transport Layer is in idle, the <TransFsmSts> field in the Serial-ATA Interface Status Register (Table 382 p. 402) is cleared.
5. Set Vendor Unique Mode. Write 1 to the <VendorUqMd> field in the Serial-ATA Interface Control Register (Table 380 p. 398).
6. Insert data into the Vendor Unique Register (Table 383 p. 402).
7. Repeat steps 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 the <VendorUqSend> field in the Serial-ATA Interface Control Register (Table 380 p. 398).
9. Write last DWORD in the FIS to Complete FIS transmission.
10. Wait for transmission completion. The <VendorUqDn> field or the <VendorUqErr> field in the Serial-ATA Interface Status Register (Table 382 p. 401) is set to 1.
11. Verify successful transmission of the FIS. Field <VendorUqErr> is cleared.
12. Clear Vendor Unique Mode. Write 0 to field <VendorUqMd>.

8.7 Protocol Based Port Select

The EDMA supports the Port Selector (PS) protocol based ingredient—When the host CPU sets the <PortSelector> field in the PHY Mode 4 Register (Table 373 p. 393), the Serial-ATA II PHY issues the protocol based OOB sequence to select the active host port.

9 Gigabit Ethernet Controller Interface

The Gigabit Ethernet controller operates at 10, 100, and 1000 Mbps. It interfaces with the PHY via a MII, GMII, or RGMII interface. The interface is also configurable as a proprietary 200-Mbps Marvell® MII (MMII) interface. For details on the pinout configurations, see the applicable Gigabit Ethernet pin multiplexing sections and the Reset Configuration section in the *88F5182 Feroceon®* Storage Networking SoC*, Datasheet*.

9.1 Functional Description

The Gigabit Ethernet port includes an IEEE 802.3 compliant 10-/100-/1000-Mbps MAC that supports GMII, MII, and RGMII interfaces with an external PHY/SERDES device. The port speed, duplex and IEEE 802.3 Flow Control can be auto-negotiated, according to IEEE 802.3 standards.

Backpressure is supported for half-duplex mode when operating at 10-/100-Mb speeds. Each port supports MIB counters.

The receive port includes a dedicated MAC-DA (Destination Address) with address filtering of up to 16-Unicast MAC addresses, 256 IP Multicast addresses, and 256 Multicast/Broadcast address. The receive port may also detect Layer2 frame-type encapsulation, as well as common Layer3 and Layer4 protocols.

IP checksum, Transmission Control Protocol (TCP) checksum, and User Datagram Protocol (UDP) checksum are always checked on received traffic, and may be generated for transmitted traffic. This capability increases performance significantly by off-loading these operations from the CPU. Jumbo-frames are also supported.

Each port includes eight dedicated receive DMA queues and one dedicated transmit DMA queue, plus two dedicated DMA engines (one for receive and one for transmit) that operate concurrently. Each queue is managed by buffer-descriptors that are chained together and managed by the software. Memory space may be mapped using configurable address windows to fetch/write buffer data and descriptors to any of the other interfaces of the device.

Queue classification on received traffic is assigned to the DMA queue, based upon a highly configurable analysis that evaluates the DA-MAC, IP, ToS (Type of Service), IEEE 802.1q priority tag, and protocol (ARP, TCP, or UDP). An example for use of this feature is the implementation of differentiated services in a router interface or for 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 service by the CPU.

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 10-/100-/1000-Mbps Gigabit Ethernet unit handles all functionality associated with moving packet data between local memory and the Ethernet ports.

The port's speed (10, 100, or 1000 Mbps) is auto-negotiated through the PHY and does not require user intervention. Auto-Negotiation for MII and GMIII is according to IEEE 802.3, draft 5.0, using the SMI interface. The 1000-Mbps unit operates only in full-duplex mode. The 100- and 10-Mbps units operate either in half- or full-duplex mode, with the selection of the duplex mode auto-negotiated through the PHY without user intervention. GMII only supports symmetric Flow Control.

Note When Auto-Negotiation is disabled, the link must be forced down when changing port speed.

There are eight receive queues and a single transmit queue. Receive/Transmit buffer management is by buffer-descriptor linked lists. Buffers and descriptors can reside throughout the entire device memory space. A Transmit buffer of any byte alignment and any size, above 8 bytes, is supported.The Receive Buffers must be 64-bit aligned. The core frequency assumption is a minimum of 83 MHz in gigabit operation.

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 (according to Ethertype)

Layer 4 (only over IPv4) for:
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)

Frame enqueueing is according to DA, VLAN-802.1q, IP-ToS, using the *highest* priority counts. Frame enqueueing is captured according to the protocol type for TCP, UDP, ARP, or BPDU. Frames smaller than the programmable minimum frame size are automatically discarded. Reception and transmission of long frames, up to 9700 bytes, 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 2 bytes 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.

The Ethernet port provides a great amount of flexibility with many programmable features. The TCP, UDP, and IP checksums are generated on any frame size. This is programmable per frame by command settings in the first descriptor of the frame. In addition, Cyclic Redundancy Check (CRC) generation is programmable for each frame. There are separate, programmable transmit and receive interrupt coalescing mechanisms to aggregate several interrupts (on a time based masking window) before sending an indication to the CPU. The unit provides programmable zero padding of short frames—frames less that 64 bytes.

A transmit buffer of any byte alignment and any size (greater than 8 bytes) is supported. Minimum packet size is 32 bytes.

In the event of collision, frames are retransmitted automatically without additional fetch. An Error and Collision report is provided in the last buffer descriptor.

9.2 Port Features

The 10-/100-/1000-Mbps Gigabit Ethernet port provide the following features:

- IEEE 802.3 compliant MAC layer function.
- IEEE 802.3 compliant MII interface.
- 1000-Mbps operation—full duplex.
- 10-/100-Mbps operation—half and full duplex.
- GMII symmetric Flow Control: IEEE 802.3 Flow Control for full-duplex operation mode.
- MII symmetric Flow Control: Backpressure for half-duplex operation mode.

- RGMII mode (non delay).
- Transmit functions:
  - Zero padding for short frames (less than 64 Bytes).
  - Long frames transmission (limited only by external memory size).
  - Checksum on transmit frames for frames up to 1.5 KB.
  - Programmable values for Inter Packet Gap and Blinder timers.
  - CRC generation (programmable per frame).
  - Backoff algorithm execution.
  - Error reporting.
- Receive functions:
  - Address filtering modes:
  - 16 Unicast
  - Unicast promiscuous mode reception (receptions of Unicast frames, even those not matched in the DA filter).
  - 256 IP Multicast
  - 256 Multicast
  - Broadcast
  - Broadcast reject mode.
  - Automatic discard of error frames, smaller than the programmable minimum frame size.
  - Reception of long frames (Programmable legal frame size is up to 9700 bytes).
    Note: Frames larger than the limit are actually received, however, they are mark in the descriptor as Oversize errors.
  - CRC checking.
  - Error reporting.

9.3 Gigabit Ethernet Unit External Interface

The Gigabit Ethernet port has an external interface that can operate as a GMII, MII, or RGMII port. The PHY serial management port is done via the GE_MDC and GE_MDIO pins.

For the Gigabit Ethernet port GMII interface and the Gigabit Ethernet management interface pin assignments, see the Pin Information section in the *88F5182 Feroceon®* Storage Networking SoC*, Datasheet*.

When configured to RGMII interface, the port pinout is reduced to 12 pins—six Tx pins (GE_TXC, GE_TXD[3:0], GE_TXCTL) and six Rx pins (GE_RXC, GE_RXD[3:0], GE_RXCTL). The RGMII interface is similar to a GMII interface running at double data rate (DDR). This means that data driven on GE_RXD[3:0], GE_RXCTL toggles on both rising and falling edges of GE_RXC; data driven on, GE_TXD[3:0], GE_TXCTL toggles on both rising and falling edges of GE_TXC.

For further information see Section 9.7, Network Interface (10/100/1000 Mbps), on page 117.

9.4 DMA Functionality

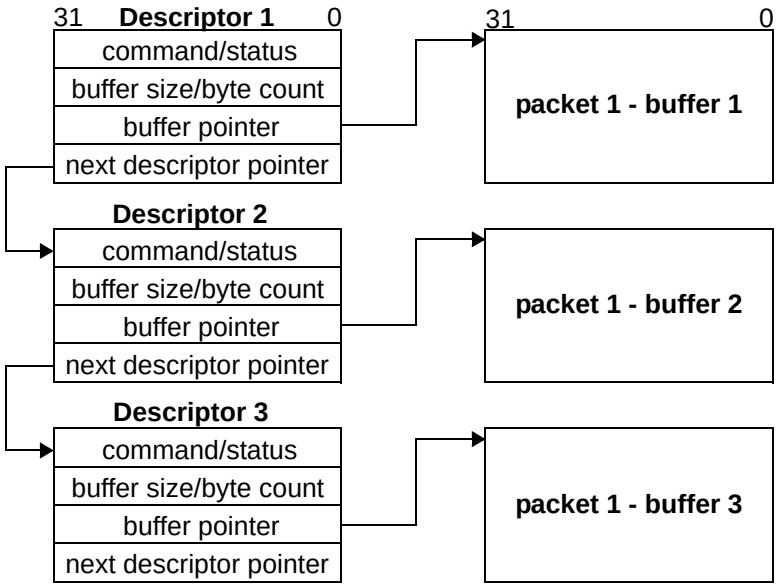
The port interfaces with an Ethernet PHY, on its serial side, and manages packet data transfer between the memory and PHY. The data is stored in memory buffers, with any single packet spanning multiple buffers if necessary. Upon completion of packet transmission or reception, a status report including error indications, is (optionally) written by the Ethernet unit to the first descriptor (for receive ports) or to the last descriptor (for transmit ports) associated with this packet.

The buffers are allocated by the CPU and are managed through chained descriptor lists. Each descriptor points to a single memory buffer and contains all the relevant information relating to that

buffer (that is buffer size, buffer pointer, etc.) and a pointer to the next descriptor. Data is read from the buffer or written to the buffer according to information contained in the descriptor. Whenever a new buffer is needed (end of buffer or end of packet), a new descriptor is automatically fetched, and the data movement operation is continued using the new buffer.

Figure 20 shows an example of memory arrangement for a single packet using three buffers.

Figure 20: Ethernet Descriptors and Buffers



The following sections provide detailed information about the operation and user interface of the Ethernet unit and its logic subsections.

Tx and Rx buffers are managed via link list of descriptors. Descriptors and buffers can be placed in any of the different device interfaces. However, the buffers and descriptors are typically placed in DRAM. Buffers and descriptors are read/write from/to memory by the port Rx and Tx DMAs.

9.4.1 Address Decoding

This section explains how the Gigabit Ethernet unit determines where to access memory for reading/writing descriptor and packets data, in the device's architecture.

The Gigabit Ethernet unit has six address windows. Each address window can be individually configured.

With each of the ports' DMA transactions (buffer read/write, descriptor read/write), the address is compared against the address decoding registers. Each window can be configured to different target interface. Address comparison is done to select the correct target interface.

Four of the six address windows have an upper 32-bit address register. 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 SDMA (if it hits the relevant address window) concatenated with the High Remap register.

Note The Gigabit Ethernet SDMA address decoder can map total of up to 4 GB of address space.

For the port DMA to avoid accessing a forbidden address space (due to a programming bug), the 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 and ID registers (see Section A.9.1, Gigabit Ethernet Unit Global Registers, on page 410).

9.4.2 Endianess and Swap Modes

Each DMA channel has configurable behavior on Little or Big Endian support, per DMA channel data receive and data transmission. See the <BLMT> and <BLMR> fields in the SDMA Configuration (SDC) register (see Table 418 on page 421).

For every DMA channel, the descriptor accesses may be swapped or not. See <Swap-mode> field in SDMA Configuration (SDC).

9.4.3 Transmit DMA Descriptors

9.4.3.1 Transmit Operation

To initialize a transmit operation, the CPU must do the following:

1. Prepare a chained list of descriptors and packet buffers.
2. Write the pointer to the first descriptor to the DMA's current descriptor registers (TxCDP).
3. Initialize and enable the Ethernet port by writing to the port's configuration and command registers.
4. Initialize and enable the DMA by writing to the DMA's configuration and command registers (Triggering the DMA is accomplished by setting the ENQ bit in the Tx Command register).

After completing these steps, the DMA fetches the first descriptor from the queue, and starts transferring data from the memory buffer to the Tx-FIFO. When the entire packet is in the FIFO (during which it may potentially calculate and update IP checksum, TCP, or UDP checksum), the port initiates transmission of the packet across the MII/GMII. While data is read from the FIFO, new data is written into the FIFO by the DMA.

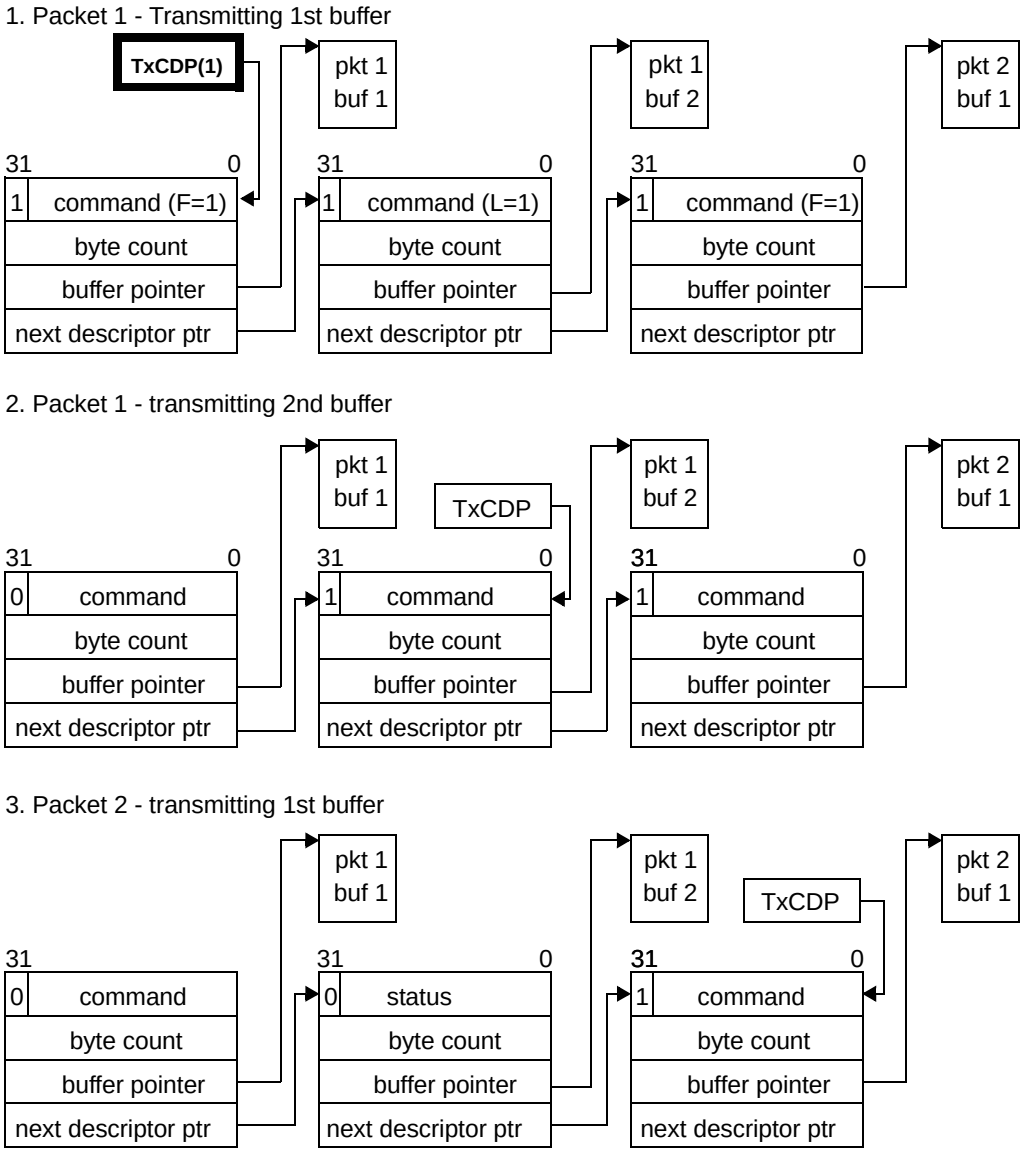
For packets that span more than one buffer in memory, the DMA will fetch new descriptors and buffers as necessary.

When transmission is completed, status is (optionally) written to the first long word of the last descriptor. The Next Descriptor's address, belonging to the next packet in the queue, is written to the current descriptor pointer register.

This process is repeated as long as there are packets pending in the transmit queue. When the DMA encounters a descriptor whose next descriptor pointer field is null, the DMA resets the ENQ bit in the Tx command register and reports the queue end via a TxEnd maskable interrupt in the ICRE register.

Figure 21 shows how the transmit descriptors are managed when a two buffers packet is transmitted.

Figure 21: Ethernet Packet Transmission Example



1. TxCDP = Transmit Current Descriptor Pointer.
Key: pkt = packet, buf = buffer, ptr = pointer.

Ownership of any descriptor other than the last is returned to the CPU upon completion of data transfer from the buffer pointed by that descriptor. The Last descriptor, however, is returned to CPU ownership only after the actual transmission of the packet is completed. While changing the

ownership bit of the Last descriptor, the DMA also writes status information, indicating any errors that might have happened during transmission of this packet. There are two relevant modes:

- AM (Auto Mode): When this mode is set, the DMA will not close descriptors that are not last descriptors (since the only change in non-last descriptors is their ownership).
- <AMNoTxES> programmable bit in the Port Configuration (GEC) (Table 411 p. 418): When this mode is set, the Last descriptors also are not closed.

Both modes save time for crossbar access to DRAM for descriptor closing.

The transmit buffer supports any byte alignment at any size (> 8 bytes) with a minimum packet size of 32 bytes.

9.4.3.2 Retransmission (Collision)

Full 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.

In half-duplex operation mode, 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, conforming 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 sets the maximum number of transmit retries for a single packet. It is defined by the IEEE 802.3 standard as 16. The event of a single packet colliding 16 times is known as *excessive collision*.

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 outside 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.

9.4.3.3 Zero Padding of Short Frames

Zero Padding is a term used to denote the operation of adding zero bytes to a frame. This feature is an option for CPU off-loading.

The Ethernet port offers a per frame padding request bit in the transmit descriptor. This causes the port to enlarge frames shorter than 64 bytes by appending zero bytes. When this feature is used, only frames equal or larger than 64 bytes are transmitted as is. Frames smaller than 64 bytes are zero padded and transmitted as 64-byte packets.

9.4.3.4 CRC Generation

Ethernet CRC denotes four bytes of Frame-Check-Sequence appended to each packet.

CRC logic is integrated into the port and can be used to automatically generate and append CRC to a transmitted packet. One bit in the transmit descriptor is used for specifying if CRC generation is required for a specific packet.

Error handling: If data was fetched with an unrecoverable error (for example, a data integrity error or a non-correctable ECC error from memory), CRC is not generated.

9.4.3.5 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, encapsulated in Ethernet-v2 format, with or without VLAN tag (This must be specified in the descriptor). IPv4 checksum is similarly supported for LLC/SNAP packets (The CPU must set the LLC/SNAP-bit in the descriptor for such packets).

One bit in the transmit descriptor is used for specifying if the IPv4 checksum generation is required for a specific packet.

9.4.3.6 TCP Checksum Generation

The TCP checksum may be enabled per frame. When TCP checksum is enabled, it is calculated during the packet DMA from memory and replaced in the checksum field before transmission begins.

This is supported for TCP over IPv4 over Ethernet-v2, with or without VLAN tag (This must be specified in the descriptor). It is similarly supported for LLC/SNAP packets, including Jumbo frames per the Alteon definition. The CPU must set the LLC/SNAP-bit in the descriptor for such packets.

Since 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 bit [10] in the Tx descriptor):

- Calculate the checksum on the *first* packet in the segment: In that case the 16 bit checksum field in the descriptor must be zero. The checksum is done fully by the port, and will include parsing the header according to the descriptor fields, calculate 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 Transmit Descriptor—Byte Count (Table 31 p. 106) (also see Figure 22, Transmit Descriptor Description, on page 103). The DMA uses this initial checksum value in calculating the TCP checksum over the TCP payload in the packet and place it in the TCP checksum field of the packet before transmission.

Note that the CPU may choose to always calculate the checksum over the pseudo header, and let the hardware take care of the payload checksum.

9.4.3.7 UDP Checksum Generation

The UDP checksum generation is the same as the TCP checksum generation support, with both first and non-first modes (see above).

9.4.3.8 VLAN Bit

The CPU is required to specify whether the packet is VLAN tagged or not. This is needed to facilitate the packet parsing during fetching from DRAM. It is used to correctly locate the IP header when the IP checksum, TCP checksum, or UDP checksum generation is required.

9.4.3.9 LLC/SNAP Bit

The Layer3 and Layer4 checksum generation is supported for Ethernet-v2 frames, or for LLC/SNAP frames (including jumbo frames). This bit must be set in case the checksum generation features is required for LLC/SNP frames or frames that comply with Alteon Jumbo Frame definition.

9.4.3.10 Transmit Descriptor Structure

- Descriptor length is 4 long words (4LW), and it must be 4LW aligned (that is, Descriptor_Address[3:0]==0000).
- Descriptors may reside anywhere in the address space except for a null address (0x00000000), used to indicate the end of the descriptor chain. Descriptor may not be placed on a Device-bus. Descriptors are fetched always in burst of 4LW.
- The last descriptor in the linked chain must have a null value in the <NextDescriptor Pointer> field[31:0] of the Transmit Descriptor—Next Descriptor Pointer (Table 33 p. 106). Alternatively, the last descriptor may be not owned. Having a not owned descriptor is useful for performance optimization, by using a dummy pointer for adding descriptors to a chain without reprogramming the First Descriptor Pointer (FDP) register (see also Section 15.4.3, Chain Mode, on page 182 and Section 15.4.6, Descriptor Ownership, on page 184).
- For packets that span multiple descriptors, the CPU must provide ownership on all the packet's descriptors before giving ownership on the first descriptor of the packet, to avoid underrun situations.
- Tx buffers associated with Tx descriptors are limited to 64 KB and can reside anywhere in memory. However, buffers with a payload of one to eight bytes must be aligned to a 64-bit boundary. Zero size buffers are illegal.

Figure 22: Transmit Descriptor Description

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 09 08	07 06 05 04 03 02 01 00	Offset
Byte 3	**Byte2**	**Byte1**	**Byte0**	
Command / Status				+0
Byte Count[15:0]		<L4iChk>/Reserved		+4
Buffer Pointer[31:0]				+8
Next Descriptor Pointer[31:4]				+C

9.4.3.11 Tx Descriptor Command/Status

Table 30: Transmit Descriptor—Command/Status

Bits	Field	Description
0	ES	Error Summary of MAC level errors on frame transmission. 0 = No Error 1 = Error occurred (Late Collision - LC, or Retransmit Limit - RL, or Underrun Error - UR) **NOTE:** This field is only valid only if <L> bit[20] is set. If <AM> bit[30] is set and the Port Configuration (GEC) <AMNoTxES> bit[12] is set, this field, as well as <EC> bits[2:1], are not updated.
2:1	EC	Error Coding 00 = LC 01 = UR 10 = RL reached (excessive collision) 11 = Reserved **NOTE:** Valid only if <L> bit[20] is set and <ES> bit[0] is set.
8:3	Reserved	Reserved
9	LLC/SNAP	When set, this bit signifies that the packet has an LLC/SNAP format. 0 = Not LLC/SNAP 1 = LLC/SNAP **NOTE:** Valid only if F is set, and if GL4chk or GIPchk is set. IP and TCP/UDP checksum is supported for LLC/SNAP frames or for Ethernetv2 frames This bit must be set for jumbo-frame formatted, according to Alteon specification as described in IEEE 802.3 LLC/SNAP.
10	L4Chk_Mode	Provides the TCP/UDP frame type for checksum calculation mode when GL4chk=1. 0 = Frame is IP fragmented. (The CPU must provide the initial checksum value calculated over the pseudo-header in the <L4iChk> field.) 1 = Frame is *not* IP fragmented. (The CPU must provide zero value in the <L4iChk> field.) **NOTE:** The payload length over which the checksum is calculated is determined by the Layer4 <LENGTH> field in the packet, and therefore, it must NOT include any pad bytes. Valid only if <F> is set, and if <GL4chk> = 1 and <L4type> = TCP or UDP.
14:11	IPv4HdLen	Provides the length in long words (4 bytes) of the IPv4 header. **NOTE:** This is only valid if <GL4chk> bit[17] and <F>[21] are set.
15	VLAN	When <GL4chk> bit[17] is set, VLAN signifies if the Ethernet-v2 frame is VLAN tagged or not. Only if <GIPchk> bit[18] or <GL4chk> are set, this field must have a correct value. 0 = Frame is *not* VLAN tagged. 1 = Frame is VLAN tagged. **NOTE:** This is only valid if <F> bit[21] is set.
16	L4type	When <GL4chk> is set, signifies which Layer4 protocol is carried in the frame. 0 = TCP 1 = UDP **NOTE:** This is only valid if the <F> bit[21] is set.

Table 30: Transmit Descriptor—Command/Status (Continued)

Bits	Field	Description
17	GL4chk	Generate TCP/UDP Checksum 0 = No operation 1 = Generate TCP/UDP checksum. **NOTE:** This may only be set to TCP or to UDP over IPv4 over Ethernetv2 frames (tagged or untagged). The CPU must provide the initial checksum value calculated over the pseudo-header in the Transmit Descriptor register's <L4iChk> bits[15:0]. The payload length over which the checksum is calculated is determined by the ayer4 Length field in the packet, and therefore, it must NOT include any pad bytes. This is only valid if <F> bit[21] is set.
18	GIPchk	Generate IPv4 checksum. This is supported for Ethernetv2 and LLC/SNAP frames (tagged or untagged), with a valid IPv4 Header (`IPHL>=5, IPHL*4<=IPTL`). **NOTE:** This is only valid if the <F> bit[21] is set.
19	P	Padding When this bit is set and the packet is smaller than 60 bytes, zero-value bytes are appended to the packet. Use this feature to prevent transmission of fragments. **NOTE:** This is only valid if <L> bit[20] is set. If set, the <GC> bit[22] is regarded as also set.
20	L	Last Indicates the last buffer of frame.
21	F	First Indicates the first buffer of a frame.
22	GC	Generate Ethernet CRC 0 = Do not generate. 1 = Generate. **NOTE:** If <GIPchk> or <GL4chk> are set, this bit is regarded as set. Only valid if the <F> bit[21] is set.
23	EI	Enable Interrupt When set, a maskable interrupt will be generated upon the closing descriptor. **NOTE:** To limit the number of interrupts and prevent an interrupt per buffer situation, set this bit only in descriptors associated with Last buffers. This way the TxBuffer interrupt is only set when transmission of a frame is completed. Interrupts may be further delayed by the Interrupt coalescing mechanism (see Section 9.6.1, Interrupt Coalescing, on page 116).
29:24	Reserved	Reserved
30	AM	Auto Mode When set, the DMA will not clear the Ownership bit <O> at the end of the buffer process. If the Port Configuration (GEC) <AMNoTxES> field[12] is set, no status is reported in the last descriptor (See <ES> field[0] and <EC>[2:1] field).
31	O	Ownership Bit 0 = Buffer owned by the CPU. 1 = Buffer owned by the DMA.

Table 31: Transmit Descriptor—Byte Count

Bits	Name	Description
15:0	L4iChk	The CPU provides the initial checksum value calculated on the pseudo header when: The Transmit Descriptor's <GL4chk> bit[17] is set The Transmit Descriptor's <L4Chk_Mode> bit[10] is cleared. Otherwise these bits are reserved. **NOTE:** Only valid if the <F> bit[21] is set.
31:16	Byte Count	Number of bytes to be transmitted from the associated buffer. This is the payload size in bytes.

Table 32: Transmit Descriptor—Buffer Pointer

Bits	Name	Description
31:0	Buffer Pointer	A 32-bit pointer to the beginning of the buffer associated with this descriptor. **NOTE:** There is a 64-bit alignment requirement for buffers that have a setting in the Transmit Descriptor register's Byte Count bits[31:16] of 1–8 bytes.

Table 33: Transmit Descriptor—Next Descriptor Pointer

Bits	Name	Description
31:0	NextDescriptor Pointer	A 32-bit pointer that points to the beginning of the next descriptor. **NOTE:** Bits[3:0] must be set to 0. A DMA operation is stopped when a null (all zeros) value is encountered in this field.

9.4.3.12 Transmit DMA Pointer Registers

The Tx DMA employs a single 32-bit pointer register, the TX DMA Current Descriptor Pointer (TxCDP) register.

The TxCDP register is a 32-bit register used to point to the current descriptor of a transmit packet. The CPU must initialize this register before enabling DMA operation. The value used for initialization is the address of the first descriptor to use.

9.4.3.13 Transmit DMA Notes

The transmit DMA process is packet oriented. The transmit DMA does not close the last descriptor of a packet, until the packet has been fully transmitted. When closing the last descriptor, the DMA writes packet transmission status to the Command/Status word and resets the ownership bit. A TxBuffer maskable interrupt is generated in the ICRE register for each queue, if the <EI> bit in the last descriptor is set.

Updating the status in the descriptor is programmable per the <AM> bit in the Tx descriptor. When set, the DMA will not clear the Ownership bit at the end of buffer process. If, in addition <AMNoTxES> bit is set in the Port Configuration register, no status will be reported in last descriptor. The advantage of this is that it reduces memory write access per descriptor This versus the trade-off of not getting error indications per packet, like late collisions, and not relying on the ownership bit for each descriptor.

Transmit DMA stops processing a Tx queue whenever a descriptor with a null value in the Next Descriptor Pointer field is reached or when a CPU owned descriptor is fetched. When that happens, a TxEnd maskable interrupt is generated in the ICRE register (per queue) and the ENQ bit is reset. To restart the queue, the CPU issues an Enable-queue command by writing 1 to the ENQ bit in the Tx command register.[1]

The transmit DMA does not expect a null Next Descriptor Pointer or a CPU owned descriptor in the middle of a packet. Also the transmit DMA does not expect a data integrity error on descriptors. When any of these events occurs, the DMA aborts transmission and stops queue processing (that is, it resets the ENQ bit). A TxError maskable interrupt is generated. To restart the queue the CPU issues an Enable_Queue command.

A transmit underrun occurs when the DMA cannot access the memory fast enough and packet data is not transferred to the FIFO before the FIFO becomes empty. In this case, the DMA aborts transmission and closes the last descriptor with a UR bit set in the status word. Also, a Tx_Underrun maskable interrupt is generated. The transmit process continues with the next packet. In the port Tx DMA, transmitting packets less than 10 KB long, such an error *cannot* happen, as the packet is fully buffered in the FIFO before transmission begins.

To stop DMA operation before the DMA reaches the end of descriptor chain, the CPU issues a Disable-Queue command by writing 1 to the DISQ bit in the DMA command register. The DMA stops queue processing as soon as the current packet transmission is completed and its last descriptor returned to CPU ownership, and then resets the ENQ bit. In addition, a TxEnd maskable interrupt is generated. To restart this queue, the CPU must issue a Enable-Queue command.

When the Ethernet link was lost during normal operation, the DMA will disable the queue by resetting the ENQ bits. Since loosing link may happen anytime during DMA programming by the CPU (for example, a disconnected cable or a far end disconnect) the following precaution must be taken: If the CPU gets a link-down interrupt, then the CPU must wait for the DMA to reset the ENQ bits of the DMA channels for Tx, after the link down event, before re-enabling the DMA channel.

The CPU must never modify the DMA configuration register or the TxCDP register while the DMA ENQ bit is set. Modifying the TxCDP registers is allowed only when the respective DMA ENQ bit is reset. Modifying the DMA configuration registers may be done only when *all* the DMA channel ENQ bits are reset.

The DMA ENQ bit cannot be reset by the CPU. Only the hardware resets it as a response to the DISQ command, or an end-condition, error condition, or link down.

9.4.4 Receive DMA Descriptors

9.4.4.1 Receive Operation

To initialize a receive operation, the CPU must do the following:

1. Prepare a chained list of descriptors and packet buffers.

NOTE: The RxDMA supports eight priority queues. If the user wants to take advantage of this capability, a separate list of descriptors and buffers must be prepared for each of the priority queues.

2. Write the pointer to the first descriptor to the DMA's current receive descriptor registers (RxCDP) associated with the priority queue to be started. If multiple priority queues are needed, the user has to initialize RxCDP for each queue.
3. Initialize and enable the DMA channel by writing to the DMA's configuration and command registers.
4. Initialize the Ethernet port by writing to the port's configuration registers (among them PSCR, Address Filter Tables, and MII/GMII Serial Parameter registers, if necessary) for the desired operational modes. Enable the port by writing to the <PortEn> bit in the register Port Serial Control (PSC) (Table 423 p. 424).

1. When the DMA stops due to a null descriptor pointer, the CPU has to write TxCDP before issuing an Enable_Queue command. Otherwise, TxCDP remains null and the DMA cannot restart the queue processing.

After completing these steps, the port starts waiting for a receive frame to arrive at the MII or GMII interface. When this occurs, receive data is packed and transferred to the RxFIFO. At the same time, address filtering test is done to decide if the packet is destined to this port. If the packet passes the address filtering check, a decision is made regarding the destination queue to which this packet is transferred. When this is done, actual data transfer to memory takes place. For detailed address filtering and priority queue assignment decisions, refer to Section 9.5, Receive Frame Processing, on page 114.

Note Packets that fail address filtering are dropped and not transferred to memory.

For packets that span more than one buffer in memory, the DMA will fetch new descriptors as necessary. However, the first descriptor pointer will not be changed until packet reception is completed.

When reception is completed, status is written to the first long word of the first descriptor, and the Next Descriptor's address is written to the current descriptor pointer register. This process is repeated for each received packet.

Note

- Only after the packet had been fully received and status information was written to the first LW of the first descriptor, will the ownership bit be reset (that is, the descriptor is returned to CPU ownership).
- Ownership of any descriptor other than the first is returned to the CPU upon completion of the data transfer to the buffer pointed by that descriptor. This means that, for each packet, the first descriptor of a packet is the last descriptor to return to CPU ownership.

9.4.4.2 Receive DMA Pointer Register

The Rx DMA employs one 32-bit pointer register per queue: RxCDP.

RxCDP is a 32-bit register used to point to the first descriptor of a receive packet. The CPU must initialize this register before enabling DMA operation. The value used for initialization is the address of the first descriptor to use. CPU must not write to this register while the DMA is enabled. Reading from this register could be used to assess the DMA progress, as well as to monitor the queue status.

9.4.4.3 Receive DMA Notes

The Receive DMA process is packet oriented. The DMA does not close the first descriptor of a packet, until the last descriptor of the packet is closed. When closing the first descriptor, the DMA writes the status to the Command/Status word and resets the ownership bit. A RxBuffer maskable interrupt is generated if the <EI> bit in the first descriptor is set.

When the DMA encounters a null next descriptor pointer or a CPU owned descriptor during normal operation (both are the only legal queue end conditions), the current received frame may be closed with error status in the descriptor, if there is insufficient space to store it in memory. The RxDMA engine will assert a maskable RxErrorQueue interrupt.

If the end-condition was a null next descriptor pointer, the DMA disables the queue by resetting the ENQ bit once it tries to prefetch the next descriptor. If the RxDMA requests a new descriptor before the CPU re-enables the queue, the DMA increments the Discarded Frames Counter (DFC). Any new frame to this queue will be discarded. If the ending condition was a unowned descriptor, then the DMA does not disable itself, but rather continues to try to read the descriptor, every time a new frame arrives to this queue.

The latter case optimizes for high speed descriptor-buffer receive allocation, as it allows the CPU to avoid re-enabling the queue, every time it adds new descriptors to the queue.

Before the CPU may enable the queue again, it must write the correct descriptor pointer to the RxCDP register. Alternatively, in case the queue end was a result of an unowned descriptor, the CPU may simply provide ownership of it to the DMA and re-enable it.

When a frame is received while the Ethernet link was lost (link down), the last frame received is cut-off and closed as a bad CRC in the first descriptor.

The CPU must never modify the DMA configuration register or the RxCDP register while the DMA ENQ bit is set. Modifying the RxCDP registers is allowed only when the respective DMA ENQ bit is reset.

DMA ENQ bits are reset after the CPU writes to the DISQ bits, and the DMA completes the current transaction on the disabled Queue (if working with the specific disabled Queue).If the CPU gets a NULL of not owned descriptor in the middle of a chain and the CPU does not solve the problem in time, the frame will be discarded, The last closed descriptor will be reclosed as a last descriptor, and the first descriptor will be closed with a resource error.

Note

The RX DMA does not reset the enable bits under link down. To reprogram, disable the queue by writing to the DISQ bits.

9.4.4.4 Frame Type Indications

The receive processing of the frame (See Section 9.5, Receive Frame Processing, on page 114) allows passing various useful indications about each individual packet in the Rx descriptor to convey MAC level errors (like Ethernet CRC check fail) and to facilitate CPU processing overhead in packet header processing and in Layer3 and Layer4 checksum calculations.

See the descriptor description for details, and for a definition of the indications see Section 9.5, Receive Frame Processing.

9.4.4.5 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 TCP header and some of the IPv4 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 off loading most of the overhead from the software. The Rx descriptor below, provides frame type indications such as: IPv4, validity of IP header with correct IPHL, IPTL, and IP checksum checked OK, Layer2 encapsulation information (VLAN, Ethernetv2 or LLC/SNAP) and TCP or UDP type detection.

TCP checksum check results are generated in the Rx descriptor in the following way, where two cases are identified:

1. For frames that have in the IP Header Flags<MF> = 0 and Offset = 0x0: This means that the IP is not fragmented (The IPv4Frg bit is reset in the descriptor) and the <RxCS> bit[25] is set to 1 in the register Port Configuration (GEC) (Table 411 p. 418). Therefore, take into account that the L4 has the L4 header in this frame and that the L4 payload can be calculated (see note below for the calculation).

NOTE: The length field for the pseudo header is taken from the following operation: `IPTL - IPHL * 4`. For the checksum calculation, the value 16'h00 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 16'h00.

The descriptor will be closed with an indication that frame is not fragmented, and the L4 checksum compare result will be valid.

2. For frames that are not from the type of #1 (either MF!= 0 or Offset!= 0)—This means that the IP is fragmented (The IPV4Frg bit is set in the descriptor) or the Port Configuration (GEC) register <RxCS> bit[25] is set to 0 (calculation without pseudo header). Therefore, the pseudo header is not calculated in the checksum.

The checksum is calculated only on the L4 payload and places the result in the F descriptor of each frame. Therefore, the checksum compare bit (L4ChkOK bit) is not valid.

Note For this type of frame, the checksum calculation does *not* put zero in the checksum field, and therefore, in frames that have Offset =0x0 and MF!= 0 (first fragment of IP), the checksum including the checksum field of the TCP header may be calculated. This is corrected by the software driver.

For this type of frame, the software adds up all the checksum calculations for the complete IP frame and subtracts the actual checksum field received in the frame and then does the comparison by itself.

9.4.4.6 UDP Checksum Checking

UDP frames include a 16-bit checksum that protects the entire segment payload (usually spanning over a number of packets) as well as UDP header and some of the IPv4 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 off loading most of the overhead from the software. The Rx descriptor below, provides frame type indications such as: IPv4, validity of IP header with correct IPHL, IPTL, and IP checksum checked OK, Layer2 encapsulation info (VLAN, Ethernetv2 or LLC/SNAP), and TCP or UDP type detection.

UDP checksum check results are generated in the Rx descriptor in the following way, where two cases are identified:

1. For frames that have in the IP Header Flags<MF> = 0 and Offset = 0x0: This means that the IP is not fragmented (IPv4Frg bit is reset in the descriptor) and the Port Configuration (GEC) register <RxCS> bit[25] is set to 1 (see Table 411 on page 418). Therefore, take into account that the L4 has the L4 header in this frame as a result:

The checksum is calculated with the corresponding pseudo header and compares the results to the frame L4 checksum. *(If the frame checksum is 0x0, then the checksum function does not compare and close the descriptor as checksum OK, since this is the indication that checksum check was disabled, according to the standard)*.

NOTE: The length field for the pseudo header is taken from the following operation: `IPTL - IPHL * 4`. For the checksum calculation, the value 16'h00 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 16'h00.

The descriptor will be closed with an indication that frame is not fragmented and the L4 checksum compare result will be valid.

2. For frames that are not from the type of #1 (either MF!= 0 or Offset!= 0)—This means that the IP is fragmented (The IPv4Frg bit is set in the descriptor) or the Port Configuration (GEC) <RxCS> bit[25] is set to 0 (calculation without pseudo header). Therefore, the pseudo header is not calculated in the checksum.

The checksum is calculated only on the L4 payload and places the result in the F descriptor of each frame. Therefore, the checksum compare bit (L4ChkOK bit) is not valid.

NOTE: For this type of frame do NOT put zero in the checksum field, and therefore, in frames that have Offset =0x0 and MF!= 0 (first fragment of IP), calculate the checksum including the checksum field of the UDP header. This is corrected by the software driver.

For this type of frame, the software adds up all the checksum calculations for the complete IP frame and subtracts the actual checksum field received in the frame and then does the comparison by itself.

9.4.4.7 BPDU Indication

If a frame is detected as BPDU, and BPDU detection is enabled then the BPDU bit is set (see also Section 9.5, Receive Frame Processing, on page 114). The rest of the L3/4 fields are still provided, but the user may want to ignore them, as they will likely not be relevant for most BPDU protocols.

9.4.4.8 Receive Descriptor Structure

- Descriptor length is 4LW, and it must be 4LW aligned (i.e. Descriptor_Address[3:0]==0000).
- Descriptors may reside anywhere in the address space except for the null address (0x00000000), used to indicate the end of a descriptor chain. Descriptors cannot be placed on a Device-bus as they are fetched always in a burst of 4LW.
- The last descriptor in the linked chain must have a null value in the <NextDescriptor Pointer> field[31:0] of the Transmit Descriptor—Next Descriptor Pointer (Table 33 p. 106). Alternatively, the last descriptor may be not owned. The latter option is useful for performance optimization, by using a dummy pointer for adding descriptors to a chain without reprogramming the RxCDP register (see also Section 15.4.3, Chain Mode, on page 182 and Section 15.4.6, Descriptor Ownership, on page 184).
- Receive buffers associated with Receive descriptors are limited to 64 KB and must be 64-bit aligned (i.e., Buffer_Address[2:0]==000).
- The minimum buffer size for the Receive buffer is eight bytes.

Table 34: Receive Descriptor Description

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	Offset
Byte 3								Byte2								Byte1								Byte0								
Command / Status																																+0
Byte Count[15:0]																Buffer Size[15:3]													Frg	0	0	+4
Buffer Pointer[31:3]																													0	0	0	+8
Next Descriptor Pointer[31:4]																												0	0	0	0	+C

9.4.4.9 Receive Descriptor Command/Status

Table 35: Receive Descriptor—Command/Status

Bits	Name	Description
0	ES	Error Summary 0 = No Error 1 = Error Occurred (RE or MF or OR or CE), **NOTE:** This is only valid if <F> bit[27] is set.
2:1	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[0] are set. If multiple errors occurred, then the reporting priority is Resource Error, Maximum Frame Length Error, Overrun Error, and CRC Error.
18:3	L4Chk	Calculated TCP/UDP Checksum **NOTE:** This is only valid if <Layer4> field[22:21] are set to 00 or 01. This is only valid if the <F> bit[27] is set and no MAC errors occurred. This is only valid if the <IPHeadOK> bit[25] and the <L3IP> bit[24] are set. The calculation does not include the pseudo header if the Receive Descriptor—Byte Count register's <IPv4Frg> bit[2] is set or the Port Configuration (GEC) register <RxCS> bit[25] is set to 0 (calculation without pseudo header).
19	VLAN	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[0] is set to 0.
20	BPDU	Bridge Protocol Data Unit Set when the frame is BPDU. **NOTE:** Only valid if the <F> bit[27] is set and the <ES> bit[0] is set to 0.
22:21	Layer4	Frame encapsulation and protocol. 00 = Frame is TCP over IPv4 over Ethernetv2 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 over Ethernetv2 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:** This is only valid if the <F> bit[27] is set, and the <ES> bit[0] is set to 0. This is only valid if the <IPHeadOK> bit[25] and the <L3IP><L3IP> bit[24] are set.
23	Layer2Ev2	Set if Layer2 is Ethernetv2.
24	L3IP	Frame type is IPv4. This is only set if Ethertype-0x800 over Ethernetv2, or over LLC/SNAP (with or without VLAN tag). Otherwise, reset. **NOTE:** This is only valid if the <F> bit[27] is set, and the <ES> bit[0] is set to 0.

Table 35: Receive Descriptor—Command/Status (Continued)

Bits	Name	Description
25	IPHeadOK	IP header is “ok” if: • Frame type is IPv4 • `IPHL >=5` • `IPHL *4<= IPTL` • IPheader Checksum is OK. 0 = Check failed 1 = Check passed **NOTE:** This is only valid if <L3IP> bit[24] is set. This is only valid if the <F> bit[27] is set, and the <ES> bit[0] is set to 0.
26	L	Last Indicates the last buffer of a frame.
27	F	First Indicates the first buffer of a frame.
28	U	Unknown Destination Address The frame is Unicast and was not matched to the MAC Address Base (DA[47:6]). **NOTE:** This is only set if working in promiscuous mode. See the Port Configuration (GEC) register <UPM>bit[0]. This is only valid if the <F> bit[27] is set and the <ES> bit[0] is set to 0.
29	EI	Enable Interrupt When set, a maskable interrupt is generated upon the closing descriptor. **NOTE:** To limit the number of interrupts and prevent an interrupt per buffer situation, set the <RIFB> (Receive Interrupt on Frame Boundary) bit[0] in the SDMA Configuration (SDC) register. Interrupts may be further delayed by the Interrupt coalescing mechanism (see Section 9.6.1, Interrupt Coalescing, on page 116).
30	L4ChkOK	Layer4 Checksum OK 1 = OK (passed) 0 = Check failed **NOTE:** If <Layer4> field[22:21] is 01 and the received frame checksum field was 16'h00, then the bit will indicate passed. This is only valid if the <IPHeadOK> bit[25] and the <L3IP> bit[24] are set. This is only valid if <Layer4> is 00 or 01. This is only valid if the <F> bit[27] is set, and the <ES> bit[0] is set to 0. This is only valid if the Receive Descriptor - Byte Count register’s <IPv4Frg> bit[2] is cleared and the Port Configuration (GEC) register’s <RxCS> bit[25] is set to 1.
31	O	Ownership 0 = Buffer owned by the CPU. 1 = Buffer owned by the DMA.

Table 36: Receive Descriptor—Byte Count

Bits	Name	Description
1:0	Reserved	Reserved
2	IPv4Frg	IPv4 is fragmented. 1 = Fragmented 0 = Not fragmented **NOTE:** 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's <F> bit[27] is set, and <ES> bit[0] is set to 0.
15:3	Buffer Size	Buffer Size in Bytes When the number of bytes written to this buffer is equal to the field's value, the DMA closes the descriptor and moves to the next descriptor. **NOTE:** The buffer size is represented by 16 bits, where the lower 3 bits are fixed to zero. Buffer Size = Bits[15:3] | | '000'.
31:16	Byte Count	When a descriptor is closed, this field is written by the port with a value indicating the number of bytes actually written by the DMA into the buffer. **NOTE:** This is only valid if the <F> bit[27] is set.

Table 37: Receive Descriptor—Buffer Pointer

Bits	Name	Description
31:0	Buffer Pointer	A 32-bit pointer to the beginning of the buffer associated with this descriptor. **NOTE:** This field must be 64-bit aligned; therefore, bits[2:0] must be set to 0.

Table 38: Receive Descriptor—Next Descriptor Pointer

Bits	Name	Description
31:0	Next Descriptor Pointer	A 32-bit pointer that points to the beginning of the next descriptor. **NOTE:** This field must be 4LW aligned; therefore, bits[3:0] must be set to 0. The DMA operation is stopped when a null (all zeros) value in the Next Descriptor Pointer field is encountered.

9.5 Receive Frame Processing

Once a frame is received by the port, the frame is parsed to go through the following processing:

- MAC errors checking.
- Accept or reject decision.
- Select the Receive queue (0 through 7).
- MIB counter increments.
- Extract Layer2/3/4 protocols and perform IP and/or TCP/UDP checksum.

Some MAC level errors, like fragments, are normally filtered from reception of frames and are only counted in MIB counters. Other MAC level errors (like CRC error frames and frames beyond the maximum allowed size) are reported in the first descriptor and in the MIB counters.

Note

- The frames maximum size is defined in the Port Serial Control (PSC) register's <MRU> bits [19:17]. The receiver can also accept jumbo frames, where the IEEE 802.3 Type/Length field is set to 0x8870 (with or without IEEE 802.1q VLAN tag).
- In this datasheet, the MAC Destination address bit [47] is the Multicast/Unicast bit. 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]).

9.5.1 Parsing the Frames

The frame goes through the following stages to decide whether or not to accept the frame and to determine which queue receives it:

1. 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) and the Port Configuration Extend (GECX) register's Span bit[1] is set, the frame is received in queue 7, the highest priority queue.
2. If the frame is Unicast then the MAC DA bits[47:4] are compared with MAC[47:4] (see the MAC Address Low (MACAL) and MAC Address High (MACAH) registers). If they do not match, then according to the Port Configuration (GEC) register's <UPM> (Unicast Promiscuous Mode) bit[0], the frame is rejected or accepted to the queue in the register's <RXQ> bits[3:1]. If matched, then the MAC DA[3:0] bits are used as a pointer to the Unicast Table entries in the DA-Filter table. The Destination Address Filter Unicast Table (DFUT) register's <Pass> and <Queue> bits (3:0) determines whether to filter the frame and its queue number.
3. Or, if DA=0xFFFFFFFF and the protocol is 0x806 Address Resolution Protocol (ARP), in Ethernet-v2, tagged or not, the frame is accepted or rejected according to the Port Configuration register's <RBArp> bit[9] setting and the queue is handled by the <RXQArp> bits[6:4].
4. Or, if DA=0xFFFFFFFF and the protocol is 0x800 Internet Protocol (IP), in Ethernet-v2 or LLC/SNAP, tagged or not, the frame is accepted or rejected according to the Port Configuration register's <RBIP> bit[8] and the queue is handled by the <RXQ> bits[3:1].
5. Or, if DA=0xFFFFFFFF and the frame is accepted or rejected according to the Port Configuration register's <RB> bit[7] and the queue is handled by the <RXQ> bits[3:1].
6. Or, if DA=0x01-00-5E-00-00-XX (where 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. The <Pass> and <Queue> bits determine the frames' filter and queue number.
7. Or, if (the frame is a Multicast of another type): A CRC-8bit (Polynomial: 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 <Pass> and <Queue> bits determine the frames' filter and queue number.

In stages 2 and 4–7, If:

- The frame is *not* discarded,
- The frame is not BPDU and the Port Configuration Extend register's Span bit[1] is not enabled,
- The frame is not a Unicast frame that is accepted because of the Port Configuration (GEC) register's <UPM> bit[0] setting or the frame is ARP protocol broadcast,

the resulting queue is still not the final one. The final queue is determined by one of the following scenarios:

- If the frame is a Transmission Control Protocol (TCP) frame and the Port Configuration (GEC) register's <TCP_CapEn>[14] is set, the queue number is determined by the Port Configuration register's <TCPQ>bits[18:16].
- If the frame is a User Datagram Protocol (UDP) frame and the Port Configuration register's <UDP_CapEn> bit[15] is set, the queue number is determined by the Port Configuration register's <UDPQ> bits[21:19].

- If the frame is VLAN tagged or is an IPv4 frame (over Ethernet v2 or LLC/SNAP), the queue number is determined as the *highest* queue from the one determined by the three DA filters, as described in stages 2–7.
  - IEEE 802.1q if the frame is VLAN tagged (see the VLAN Priority Tag to Priority (VPT2P) register)
  - Differentiated Services-Code-Point (see the IP Differentiated Services CodePoint 0 to Priority (DSCP0) register) if frame is IPv4 over EthernetV2 or LLC/Snap.

Note

- To not use IEEE 802.1q mapping or the Differentiated Services CodePoint (DSCP), program these registers to their default value— all zeros.
- Before enabling the port, the DA-Filter tables must be programmed in full, as their initial values are undefined.

9.6 Ethernet Interrupts

The GbE unit activates the following interrupt bits in the Main Interrupt Cause Register (Table 99 p. 253):

<GbEErr> This is a unit level Interrupt activated by the <EtherIntSum> field in the Ethernet Unit Interrupt Cause (EUIC).

<GbEMisc> Activated by bits [16] and [23:20] of the Port Interrupt Cause Extend (ICE) (Table 429 p. 431).

<GbERx> Activated by bits [18:2] of the Port Interrupt Cause (IC) (Table 428 p. 430) and bits [18:17] of the Port Interrupt Cause Extend (ICE) (Table 429 p. 431).

<GbETx> Activated by bits [30:19] of the Port Interrupt Cause (IC) (Table 428 p. 430) and bits [19] and [15:0] of the Port Interrupt Cause Extend (ICE) (Table 429 p. 431).

<GbESum> A port interrupt summary bit activated by bit [31] in the Port Interrupt Cause Extend (ICE) (Table 429 p. 431).

9.6.1 Interrupt Coalescing

Since the Gigabit Ethernet line rate provides a high packet rate, it is important to reduce the amount of interrupts that the Ethernet DMA may generate.

For this purpose, the DMA Receive and Transmit have several modes that provide the option of choosing the type of events that initiate issuing interrupts. (See also Port Interrupt Cause Extend (ICE) and Ethernet Unit Interrupt Cause (EUIC)).

The most intensive interrupts are the packet-level interrupts on receive and transmit. In addition to the CPU's ability to specify, in the Receive and Transmit descriptor, which descriptor close may cause an interrupt, the port provides a programmable mechanism that allows coalescing these types of interrupts.

On a per port basis, and for Rx and Tx transactions, the device has a programmable timer in the SDMA Configuration (SDC) register's <IPGIntRx> bits[21:8] for receive and <IPGIntTx> (transmit) to force a minimum time between interrupts associated with the Port Interrupt Cause (IC) register's <RxBufferQueue> bits[9:2] and the Port Tx FIFO Urgent Threshold (PTFUT) register <IPGIntTx> bits[17:4]. This minimum time is programmable and may be changed dynamically during normal operation.

The flow for packet-level interrupts on receive and transmit is as follows:

Note

The following example describes the Receive flow, however, the Transmit flow is implemented in an identical fashion.

1. A non-masked <RxBufferQueue> interrupt cause is asserted. As a result, an interrupt is raised (propagated in the interrupt hierarchy etc.) and the relevant interrupt coalescing counter begins to count. From this point (after CPU read from the interrupt register) until the count-down finishes, *no new interrupts can be raised* due to new packet reception (transmission) from any of the eight queues.
2. During the countdown time, and before the next CPU read of the Port Interrupt Cause (IC) (Table 428 p. 430) register (or Port Interrupt Cause Extend (ICE) (Table 429 p. 431) for Tx), the <RxBufferQueue> events would still cause loading 1 to the appropriate cause bit, but would not cause raising an interrupt at the unit, port, or port-Rx (Tx) level. Before reading the register, it is assumed that the CPU do not reset the <RxBufferQueue> cause bits.
3. Once the CPU reads the IC register (or ICE register for Tx), the value of all <RxBufferQueue> interrupts that are recorded later on, accumulate in a shadow register, actually two separate registers—one for IC register <RxBufferQueue> and one for ICE register TxBuffer additional events, both of which are invisible to the software.
4. When the countdown timer expires, the <RxBufferQueue> in the shadow register is loaded into the IC register (or ICE register for Tx). This may cause raising an interrupt, again.

This mechanism prevents loss of interrupt indications in the time frame between the time that the CPU reads the interrupt register and the time that it starts switching off interrupt bits. During this interval, new interrupts that arrive for the same receive or transmit queue would have been lost and buffers might have gotten stuck indefinitely. The shadow registers, just described, prevents this from happening.

9.7 Network Interface (10/100/1000 Mbps)

The port can be connected to a Gigabit Ethernet network using the GMII PHY (to 1000BASET copper or 1000BASEX), to the RGMII PHY, or the MII PHY.

To support all speeds, the port 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 MII mode only).
- Flow Control option in full-duplex.

Auto-Negotiation is supported for all interface modes:

MII and GMIII according to IEEE 802.3 draft 5.0 using MDC/MDIO pins.

9.7.1 Gigabit Ethernet MAC

The port Gigabit MAC supports the following:

- Connection to GMII PHY (for 1000BASEX or 1000BASET), or GMII PHY.
- 1000 Mbps full duplex.
- Standard IEEE 802.3 Flow Control in full duplex.

The port MAC performs all of the functions of the IEEE 802.3 such as frame formatting, frame stripping, collision handling, deferral to link traffic, etc. The port ensures that any outgoing packet complies with the IEEE 802.3 specification in terms of preamble structure. The port transmits 56 preamble bits before the Start-of-Frame Delimiter.

9.7.2 GMII Interface

The transmit and receive operations are done in full duplex and implement the standard.

9.7.2.1 GMII Transmission in Full Duplex

When the port has a frame ready for transmission and the IPG counter has expired, frame transmission begins.

Note The Carrier Sense (CRS) and Collision Detect (COL) input pins are ignored in this mode.

The data is transmitted via pins GE_TXD[7:0] of the transmitting port and clocked on the rising edge of GE_TXCLK_OUT. At the same time, signal GE_TXEN is asserted. The GE_TXER signal is always driven LOW as there is no carrier-extension required.

9.7.2.2 GMII Reception

Frame reception starts with the PHY's assertion of GE_RXDV or GE_RXER (while the port is not transmitting). Once GE_RXDV or GE_RXER is asserted, the port begins sampling incoming data on pins GE_RXD[7:0] on the rising edge of the GE_RXClk.

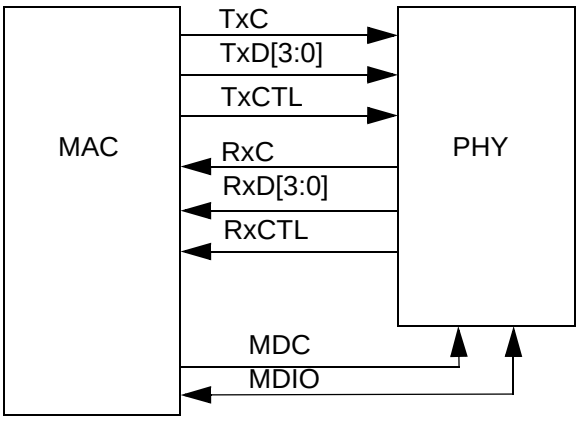
Note The GE_RXDV signal is high during reception of packet-data.

9.7.3 RGMII Interface

The port supports RGMII/Modified MII interface. The RGMII specification proposed by Hewlett Packard is intended to be an alternative to the IEEE 802.3 (MII) and the IEEE 802.3 (GMII).

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 23). 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 23: RGMII Pin Interconnection Between MAC and PHY



9.7.3.1 Signal Definition

The RGMII interface uses a 125 MHz DDR clock with a 4-bit 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.

Table 39: RGMII /Modified MII Signals

Port Pins	RGMII	Description
GE_TXCLK_OUT	TXC	125 MHz, 25 MHz, or 2.5 MHz transmit clock derived from the CLK_125 input.
GE_TXCTL	TX_CTL	Transmit Control Signals. GE_TXEN is encoded on the rising edge of GE_TXCLK, and GE_TXERR XOR GE_TXEN is encoded on the falling edge of GE_TXCLK.
GE_TXD[3:0]	TD[3:0]	Transmit Data. In 1000BASE-T and 1000BASE-X modes, TxD[3:0] is presented on both edges of TxC. In 100BASE-TX and 10BASE-T modes, GE_TXD[3:0] is presented on the rising edge of GE_TXCLK.
GE_RXCLK	RXC	125 MHz, 25 MHz, or 2.5 MHz receive clock with a +/- 50 ppm tolerance derived from the received data stream and based on the selected speed.
GE_RXCTL	RX_CTL	Receive Control Signals. GE_RXDV is encoded on the rising edge of GE_RXCLK, and GE_RXERR XOR GE_RXDV is encoded on the falling edge of GE_RXCLK.
GE_RXD[3:0]	RD[3:0]	Receive Data. In 1000BASE-T and 1000BASE-X modes, RXD[3:0] is presented on both edges of GE_RXCLK. In 100BASE-TX and 10BASE-T modes, GE_RXD[3:0] is presented on the rising edge of GE_RXCLK.

9.7.3.2 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 V MHz for 10 Mbps. The MAC always generates the SYS_CLK signal and the PHY always generates the GE_RXCLK signal.

During packet reception, GE_RXCLK may 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 GE_TXCTL (TX_CTL) low until the MAC has ensured that GE_TXCTL (TX_CTL) is operating at the same speed as the PHY.

9.7.3.3 Signals Encoding

The RGMII interface is basically a GMII interface running at double data rate:

- GMII TxD[3:0] is driven on GEx_TXD[3:0] on the rising edge of GEx_TXCLKOUT; GMII TxD[7:4] is driven on GEx_TXD[3:0] on the falling edge of GEx_TXCLKOUT.
- GMII TxEN is driven on GEx_TXCTL on the rising edge of GEx_TXCLKOUT.
- A logical value of GMII TxEN XOR GMII TxERR is driven on GEx_TXCTL on the falling edge of GEx_TXCLKOUT.
- GMII RxD[3:0] is sampled on GEx_RXD[3:0] on the rising edge of GEx_RXCLK; GMII RxD[7:4] is sampled on GEx_RXD[3:0] on the falling edge of GEx_RXCLK.
- GMII RxDV is sampled on GEx_RXCTL on the rising edge of GEx_RXCLK.
- A logical value of GMII RxDV XOR GMII RxERR is sampled on GEx_RXCTL on the falling edge of GEx_RXCLK.

9.7.3.4 In-Band Status

To ease detection of the link status, speed, and duplex mode of the PHY, inter-frame signals are placed onto the GE_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 GE_RXD[3:0] bits.

Collision is determined at the MAC when `TX_EN = True`, while either GE_CRS or P0_RXDV are true. The PHY does not assert GE_CRS as a result of GE_TXEN being true.

9.7.4 10-/100-/200-MII Interface

The port MAC allows it to be connected to a 10-Mbps, 100-Mbps or 200-Mbps network. The port interfaces to 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 port can switch automatically between 10- or 100-Mbps operation depending on the speed of the network. Data transfers are clocked by the 50-MHz transmit and receive clocks in 200-Mbps operation, or by 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.

Note The 200-Mbps operation uses the proprietary Marvell MII (MMII) interface.

9.7.5 Interface Mode Selection

Every Gigabit Ethernet interface mode is individually configured to RGMII/GMII/MII by its reset configuration pins mode.

If Auto-Negotiation is enabled for GMII or MII interface mode (by the Port Serial Control (PSC) register's <AN_Duplex> bit[2] and <AN_FC> bit[2]), the MDC/MDIO Auto-Negotiation takes place.

9.8 Auto-Negotiation Modes

9.8.1 Auto-Negotiation in MII/GMII Modes

The port 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 Port Serial Control (PSC) register <ANSpeed> bit [13] to 0, to switch between GMII and MII modes.

Note

The registers and bits referred to in this sub-Section (for example, registers 4, 5, and 15 and bit 1.8) are PHY Device registers.

The port continuously reads the PHY register 1 to establish 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 port advertises its Flow Control ability (if Auto-Negotiation for Flow Control is enabled by the Port Serial Control (PSC) register's <AN_FC> bit.

The port 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) and register 15 exists. The port continues to read register 15 to determine whether the PHY is 1000BASEX-capable or 1000BASET capable. If it is 1000BASEX, then the port regards the multiplexed speed as 1000 Mbps only and follows the IEEE 802.3 clause 37 rules (for register 4 and 5 format) for duplex and Flow Control Auto-Negotiation. If it is 1000BASET capable, the port follows the IEEE 802.3 rules to resolve the speed (1000 Mbps (using GMII interface) or 10/100Mbps (using MII interface)), duplex mode, or Flow Control.

After Auto-Negotiation is complete, the port resolves negotiated modes of operation. These values update the Port Status register fields and affect the Network port operation.

9.8.2 Auto-Negotiation Bypass Mode

The IEEE standard Auto-Negotiation state machine, per the IEEE 802.3 Clause 37, requires that both sides support Auto-Negotiation before the link can be established. If one side implements the Auto-Negotiation function and the other does not, two-way communication is not established, unless Auto-Negotiation is manually disabled and both sides are configured to work in the same operational modes.

When the bypass mode is enabled, the port Auto-Negotiation state machine changes from the type specified in clause 37:

When entering the "Ability_Detect" state, a timer is started to count down with an initial value of 20 times the link-timer value. Since the link-timer value is ~10 ms, the "Ability_Detect" associated timer is ~200 ms.

If the timer expires and, during this period, the receive synchronization machine stayed in synchronization and did not report RUDI(INVALID) and the state-machine is still in the Ability_Detect state, this is interpreted as a sign that the other side is "alive" but cannot send configuration codes to perform Auto-Negotiation. Therefore, the state-machine moves to a new "Bypass_Link_Up" state. In

this state, the port assumes a link up and the operational mode is set to the Port Serial Control (PSC) register's <AN_Duplex> and <AN_FC> values are at the time.

Note Once the other device is replaced by a device that can perform Auto-Negotiation, the Auto-Negotiation is automatically restarted.

If the other device only transmits idles during this extended timer period, the bypass is performed. In this instance, configuration codes are not idles. Therefore, a regular Auto-Negotiation device does not allow the bypass to take place.

If the bypass was performed, the port reports this via the Port Status register's <Bypass_Activated> bit, together with the Interrupt stating link change. Therefore, management can recognize whether the link was resumed due to standard Auto-Negotiation or bypass.

9.9 Data Blinder

The MII Serial Parameters (Table 413 p. 419) register's <DataBlind> bits[21:17] set the time period during which the port does not look at the wire to decide to defer a pending transmission, due to receive activity.

9.10 Inter-Packet Gap

The Inter-packet Gap (IPG) is the idle time between the CRC from the first packet to the preamble of the next packet from the same port. The default (from the standard) is 96 bit times.

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 MII Serial Parameters (Table 413 p. 419) and GMII Serial Parameters (Table 414 p. 420) to set the IPG size.

9.11 Illegal Frames

For undersized frames (with or without good CRC), the port discards all illegal frames. The 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 MRU) with or without bad CRC (bad checksum) are forwarded to the DMA queue with an error summary report in the Rx descriptor.

9.12 Backpressure Mode

Only when the network port is operating in half-duplex, MII mode will the port implement a Backpressure algorithm.

The Backpressure algorithm is enabled by setting the Port Serial Control (PSC) register's <ForceBPMode> bits[8:7].

For a port in Backpressure mode, the port waits until the medium is idle and then transmits a JAM pattern for a programmable value of time. To program the period that the JAM pattern is transmitted, set the <JAM LENGTH> field[1:0]. The IPG between two consecutive JAM patterns (or between the

last transmit and the first JAM) is programmed using the <JAM-IPG> bits[6:2]. <JAM LENGTH> and <JAM-IPG> are set in the MII Serial Parameters register.

When a port in Backpressure mode has a pending packet for transmission, it halts the transmission of the JAM pattern. The JAM pattern is halted for an IPG value set through the MII Serial Parameters registers' <IPG-JAM_TO_ DATA> bits[11:7]. After the IPG is completed, the port transmits the packet. If the port remains in Backpressure mode, it resumes the JAM pattern transmission after an IPG set by the <JAM-IPG> bits, following the packet transmission.

9.13 Flow Control

The port 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 SMI interface (MII or GMII PHYs)

The behavior of the port is determined by the value in Ethernet Port Status (PS) (Table 425 p. 427) <EnFC> bit and Port Serial Control (PSC) (Table 423 p. 424) <ForceFCMode> bit.

The CPU may write to the PSCR <ForceFCMode> bit when Flow Control operation is enabled in Port Status register <EnFC> bit (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 Ethernet Port Status (PS) <EnFC> bit).

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 (see the more detailed description in Section 9.13.2, Pause Transmit Operation, on page 124. The CPU response time to congestion cases would determine if and how many packets may be lost on receive.

The value in Ethernet Port Status (PS) <EnFC> bit can be set from the following:

- CPU programming.
- Result of Auto-Negotiation for Flow Control according to IEEE 802.3 standard in all modes – MII, and GMII.

When in MII or GMII modes, with Auto-Negotiation for Flow Control enabled, the port 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 port supports Symmetric Flow Control only according to the IEEE 802.3 standard (section 37.2.3.2).

The advertised ability of Pause support depends on the setting of the Port Serial Control (PSC) register's <Pause_Adv> bit[4] as follows:

- When set, the port advertises symmetric capability for Pause.
- When reset, the port advertises No Pause capability.

9.13.1 Pause Receive Operation

When the port receives a Pause packet, it avoids transmitting a new packet to the port for the period of time specified in the received Pause packet.

The pause quantum is 512 bits regardless of the operation speed or the duration of the slot time.

A received packet is recognized as Flow Control if it was received without errors and is one of the following:

- DA = 01-80-C2-00-00-01 and type=88-08 and MAC_Control_Opcode=01.

A packet received that is identified as a Pause packet is always discarded, even if the Pause function is disabled.

9.13.2 Pause Transmit Operation

For enabling Pause Transmit operation or either enabling or disabling the Ethernet Port Status (PS) <EnFC> bit must be in the active state.

It is the CPU responsibility in sensing 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, which is allocated to the port via the crossbar Mbus SDRAM arbiter to avoid bandwidth shortage for the gigabit port.

Note

This mechanism does not provide hardware guarantee of zero frame-loss, as it 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 01 value to the PSCR <ForceFCMode> bits [6:5]. The transmit port will schedule transmission of a pause-disable frame (timer=0xFFFF) at the next possible frame boundary and will automatically retransmit it at least every 4.2 msec (GMII), 42 msec (MII at 100 MB), or 420 msec (MII at 10 MB) as long as the value in the <ForceFCMode> field remains 01.

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 the PSCR <ForceFCMode> bits[6:5]. This will trigger transmission of a single pause enable packet (timer = 0x0000) that enables 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 PSCR <ForceFCMode> bits[6:5] are always reset to 00 (No Pause disable frames are sent).

9.14 MII/GMII Serial Management Interface (SMI)

The port MAC contains a Serial Management Interface (SMI) for MII or GMII compliant PHYs.

This allows control and status parameters to be passed between the port 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 port 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.

The CPU can write/read to/from all PHY addresses/registers. The SMI allows the CPU to have direct control over an MII or GMII compatible PHY device via the SMI register. This control allows the

driver software to place the PHY in specific modes such as Full-Duplex, Loopback, 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 port 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.

9.14.1 SMI Cycles

The SMI protocol consists of a bit stream that is driven or sampled by the port on each rising edge of the MDC clock. The SMI frame, bit-stream format starts with PRE and ends with IDLE. Its various steps are described in Table 40.

Table 40: SMI Bit Stream Format

	PRE	ST	OP	PHYAd	RegAd	TA	Data	IDLE
READ	1...1	01	10	AAAAA	RRRRR	Z0	D.D(16)	Z
WRITE	1...1	01	01	AAAAA	RRRRR	10	D.D(16)	Z

The column headings in Table 40 are defined below:

PRE Preamble — At the beginning of each transaction the port sends a sequence of 32 contiguous logic 1 bits on the MDIO, with 32 corresponding cycles on the MDC, to provide the PHY with a pattern it can use to establish synchronization.

ST Start of Frame — Start-of-Frame pattern of 01.

OP Operation Code — 10 - Read
01 - Write

PhyAd PHY Address — 5-bit address of the PHY device (32 possible addresses). The first PHY Address bit transmitted by the port is the MSB of the address.

RegAd Register Address — 5-bit address of the PHY register (32 possible registers in the PHY). The first register address bit transmitted by the port is the MSB of the address. The port always queries the PHY device for status of the link by reading register 1, bit 2.

TA Turn Around — Turnaround time is a 2-bit time spacing between the <RegAd> field and the <Data> field of the SMI frame, to avoid contention during a read transaction. During a read transaction the PHY must not drive MDIO in the first bit time and drive 0 in the second bit time. During a write transaction the port drives a 10 pattern, to fill the TA time.

Data Data — Data field is 16 bits long. The PHY drives the data field during read transactions. The port drives the data field during write transactions. The first data bit transmitted and received is bit 15 of the PHY register being addressed.

IDLE Idle — IDLE condition on MDIO is a high impedance state. The MDIO driver is disabled and the PHY must pull-up the MDIO line to a logic 1.

9.14.2 SMI Accelerated Modes

The 88F8x5x support faster clock speed modes. When connecting the device to other Marvell devices, these modes enable a higher utilization of the SMI bus. In these modes, the MDC clock output rate can be set to 10.3 MHz or 20.7 MHz, by dividing the TCLD by 16 or by 8, respectively. Changing the MDC clock rate is done by updating Ethernet Unit Reserved (EU) (Table 399 p. 412) at offset 0x72014.

9.14.3 SMI Timing Requirements

When the MDIO signal is driven by the PHY, it is sampled by the port synchronously with respect to the rising edge of MDC. Per the IEEE 802.3 specification, the clock to output delay from the PHY, as measured on the device pads, shall be a minimum of 0 ns and a maximum of 300 ns, as shown in Figure 24. Further, when the MDIO signal is driven by the device, there is a minimum of 10 ns of setup time and minimum of 10 ns of hold time as shown in Figure 25.

Figure 24: MDIO Sourced by PHY

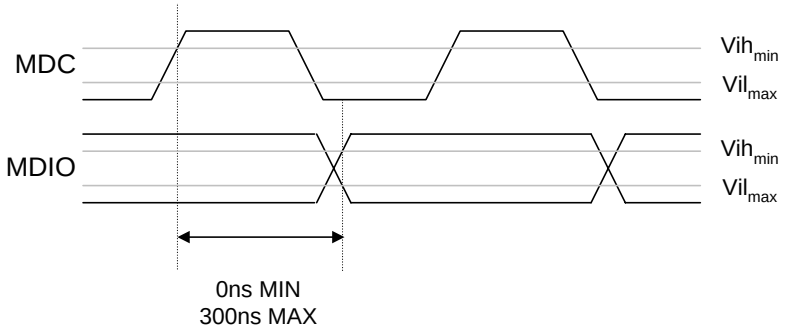
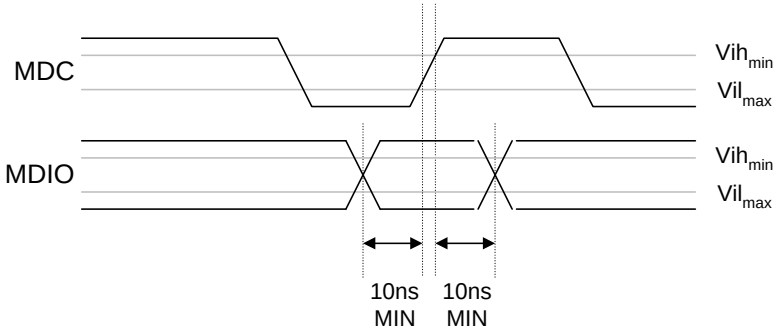


Figure 25: MDIO Sourced by Device



9.15 Link Detection and Link Detection Bypass (ForceLinkPass)

Typically, the port 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 (out of a possible 32 addresses). The port 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 port 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).

9.15.1 Force_Link_Fail

The PSCR register's <ForceLinkFail> bit (bit 10) has the default value of forcing the link detection on each port to link down. The user *must* set this bit, to get the true link status of the port and to enable the port link indication to go up.

The user must not program the <ForceLinkFail> bit and the <Force_Link_ Pass> bit to be set at the same time.

9.16 Network Management Interface Counters

The port incorporates a set of management counters.

For a complete description refer to Appendix A.9.3, Port MIB Counter Register, on page 441.

9.17 Port MIB Counters

The MAC MIB Counters 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 space. 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.

9.17.1 Definitions

The following table summarizes the terms used in the definition of the counters.

Table 41: Definitions for MAC MIB Counters

Term	Definition
Collision Event	A collision has been detected before 576-bit times into the transmitted packet after GE_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 GE_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 specification. 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.

Table 41: Definitions for MAC MIB Counters (Continued)

Term	Definition
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.
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

The following figures illustrate the terms defined above:

Figure 26: Ethernet Frame Classification

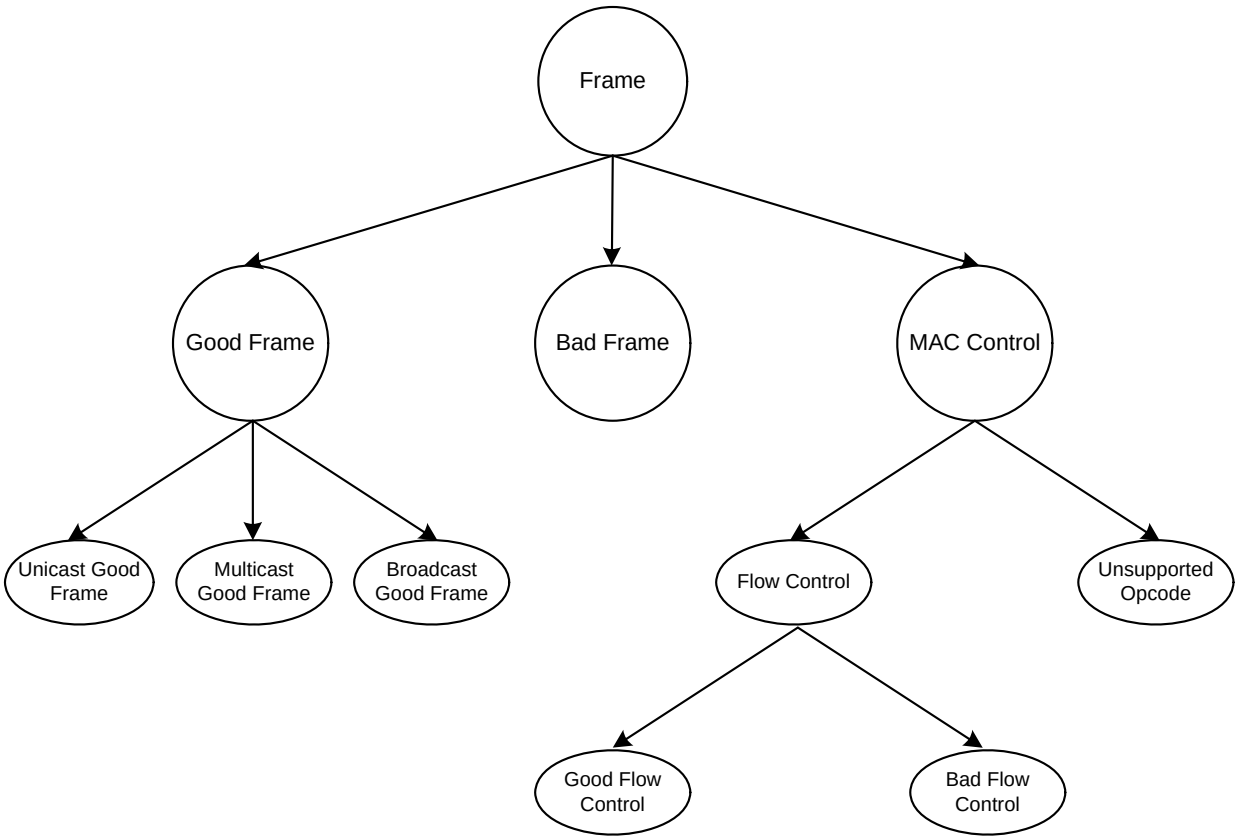
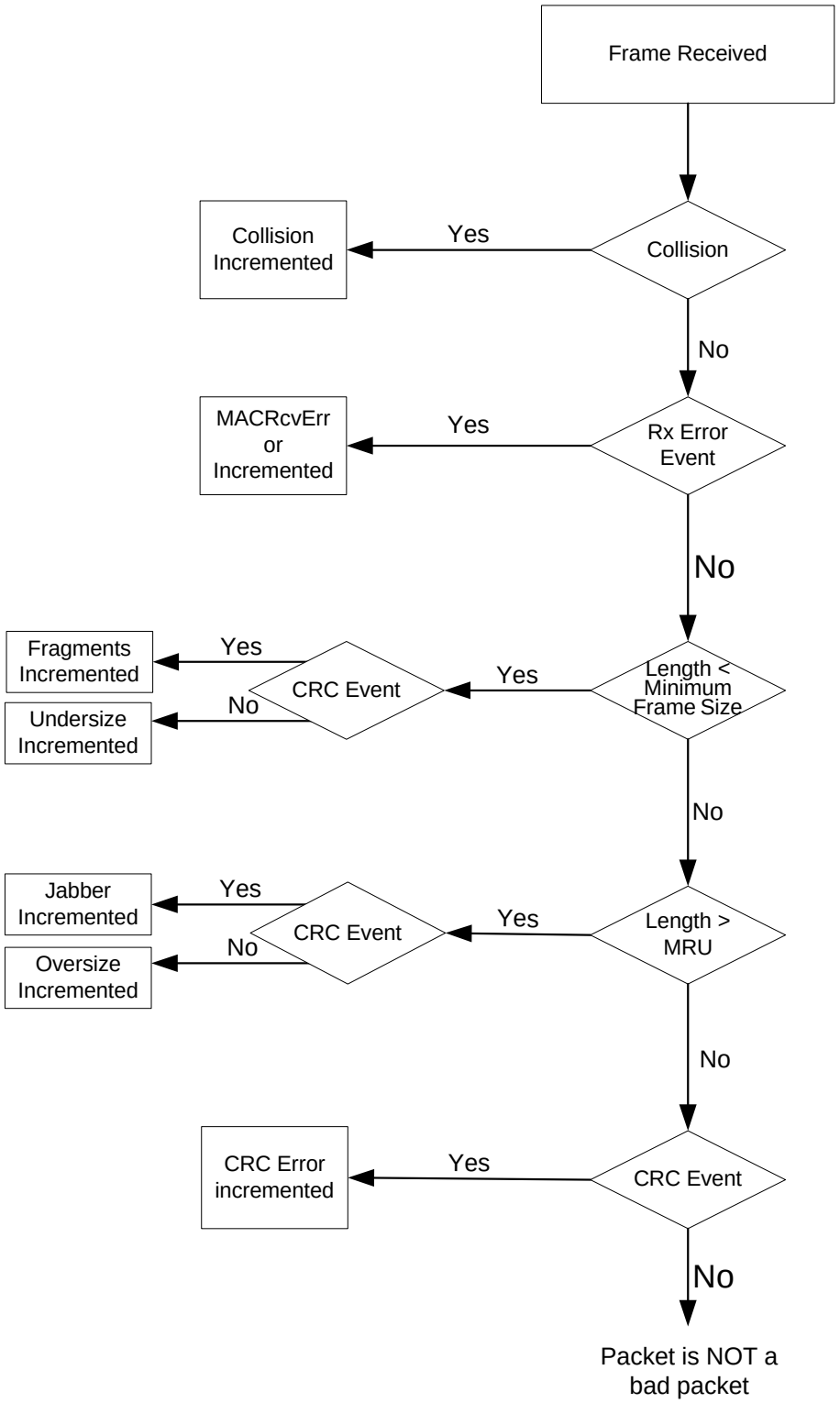


Figure 27: Bad Frame Procedure



9.17.2 Per Port Counters

The counters initialize to 0 after reset. The counters are read only. Upon counter read, it is reset again to 0.

Most counters are 32 bits. The Good Bytes received and Good Bytes Sent are 64-bit counters that are read in two accesses— low address first and high address next (counter is reset upon read from high address).

Refer to PHY Address (Table 395 p. 410) for the address offsets of all the counters for the port. Within the counters block, the counter offset is in the address bits[6:0].

The following table lists the counters maintained per port, with the address field providing the address offset of each counter from the base address of the counters for that port.

In addition to the per port counters, in the MAC MIB Counters (Table 448 p. 441) register, there are some additional counters that count filtered frames for reasons like MAC address lookup results, called Port Overrun Frame Counter (POFC) (Table 436 p. 434) and Port Rx Discard Frame Counter (GEDFC) (Table 435 p. 434) registers. In conjunction with the counters block they provide total frames received information.

10 USB 2.0 Interface

The 88F5182 supports two USB 2.0 ports each with an embedded USB 2.0 PHY.

The USB 2.0 interface can act either as a USB high-speed peripheral (device) or as a USB host controller. It is fully compliant with the *Universal Serial Bus Specification, Revision 2.0* (USB 2.0).

The USB 2.0 interface contains a single dual-role controller (aka *USB controller*) and a bridge, connecting the controller to the internal Crossbar interface (aka *USB bridge*).

The USB 2.0 port contains an embedded USB 2.0 PHY (aka *USB PHY*), supporting both host and peripheral modes.

10.1 Functional Description

The USB 2.0 interface supports the following features:

Host Controller
EHCI compliant as a host.
As a host, supports direct connection to all peripheral device types—Low Speed (LS), Full Speed (FS), High Speed (HS)

Peripheral
USB 2.0 compliant peripheral controller.
As a peripheral, connecting to all host types (HS, FS) and hubs.
Four independent endpoints support control, interrupt, bulk and isochronous data transfers.

Embedded PHY
480 Mbps High Speed (HS)/ 12 Mbps FS, FS only and LS only 1.5 Mbps serial data transmission rates.
SYNC/EOP generation and checking.
Data and clock recovery from serial stream on the USB.
NRZI encoding/decoding with bit stuffing/unstuffing.
Bit stuffing/unstuffing; bit stuff error detection.
Holding registers to stage transmit and receive data.
Supports USB 2.0 Test Modes.
Ability to switch between FS and HS terminations/signaling.

11 Cryptographic Engines and Security Accelerator

The 88F5182 integrates hardware-based cryptographic engines and a security accelerator. These engines were designed to perform time-consuming cryptographic operations such as AES/DES/3DES encryption and MD5/SHA1 authentication, to reduce CPU packet processing overhead.

There are four cryptographic engines, each operating independently. They 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 and MD5

The following acronyms, abbreviations, and definitions are used in this section.

Table 42: Acronyms, Abbreviations, and Definitions

Acronym	Definition
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
Block/data	Block of 512 bits in the Authentication engine
CBC	Cipher Block Chain
CFB	Cipher Feedback
DES	Data Encryption Standard
3DES	Triple Data Encryption Standard
ECB	Electronic Code Book
EDE	Encryption Decryption Encryption
EEE	Encryption Encryption Encryption
IV	Initial Vector / Initial Value
MD5	Message Digest 5
OFB	Output Feedback
SHA-1	Secure Hash Algorithm 1

Table 42: Acronyms, Abbreviations, and Definitions (Continued)

Acronym	Definition
W0…W15	Designates the 16 words in an authentication input data block; W0 is the first word and W15 is the last word.
WORD	32-bit

11.0.1 Cryptographic Engine Features

The cryptographic engines support the following features:

- Authentication in the MD5 or SHA 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 and 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. (Saves write cycles.)
- Authentication and encryption can be performed 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.

11.0.2 Security Accelerator Features

The security accelerator supports the following features:

- Performs a complete over-the-packet operation with no software intervention.
- Supports two consecutive sessions, allowing pipelining in packets processing.
- Supports four types of operation:
  - Authentication only (MD5 / SHA-1 / HMAC-MD5 / HMAC-SHA1)
  - Encryption/Decryption only (DES / 3DES / AES - both ECB and CBC)
  - Authentication followed by Decryption/Encryption
  - Decryption/Encryption followed by Authentication

11.0.3 Using the Cryptographic Engines

The cryptographic engines can be accessed by the security accelerator or by the host[1], by writing and reading to specified addresses in the engine.

11.0.3.1 Commands and Control

The engines' modes of operation (AES, DES, 3DES, and SHA) and the endianness of the input data are controlled by the host via four specified command registers:

- SHA-1/MD5 Authentication Command Register
- DES Command Register

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.

- AES Encryption Command Register
- AES Decryption Command Register

(see Appendix A.11, Cryptographic Engine and Security Accelerator Registers, on page 451).

These registers also contain flags that are used as status indicators to the host.

The engines provide interrupts that are set when an operation is completed (see Table 519, Cryptographic Engines and Security Accelerator Interrupt Cause Register, on page 468).

11.0.3.2 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. These are provided by writing to specific registers.

Both the Encryption engine and the Authentication engine support byte swap of input data.

11.0.3.3 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 or five words that are the hash signature of the input data block.

The output data is accessed through specific registers in the engines.

11.0.3.4 Principle of Operation

When a host wants to perform cryptographic operations, it writes the cryptographic parameters (IV, keys, etc.) and the command to be carried out in the registers. Then it 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 has been 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. The host can then read the result from the engine's registers.

11.0.4 Using the Security Accelerator

The accelerator is activated by the CPU. It works on top of the cryptographic engines, setting them to work in the required manner. The entire operation of the accelerator is performed in the local SRAM of the security accelerator. The CPU copies the packet to be processed into the local SRAM of the security accelerator. (The IDMA engine can be used by the CPU for this purpose.)The CPU then prepares a descriptor (see Section 11.2.4 "Security Accelerator Descriptor Data Structure") stating the required operation and activates the accelerator. The accelerator reads the descriptor, sets up the engines, and starts feeding the packet data into the engine, one data block at a time. It waits for completion, reads the data from the engine, and stores it in the local SRAM of the security accelerator. This continues until the entire packet has been processed.

11.1 Cryptographic Engines Operation

The unit combines four separate engines:

- DES encryption/Decryption

- AES128 Encryption
- AES128 Decryption
- Authentication MD5/SHA

Each of these engines has separate registers for data, control, and operation modes. Address allocation is specified in Table 465, Cryptographic Engine and Security Accelerator Register Map, on page 451.

11.1.1 Authentication

To activate the authentication process, the host performs the following, steps:

1. Verify the termination bit in the Authentication Command 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 performed by the host to the Authentication engine resets the termination bit.
2. Write operational mode and endianness fields of the SHA-1/MD5 Authentication Command Register (see Table 487 on page 458). (Optional if no change is needed.)
3. Write initial values:

   Required only if multiple packets are processed simultaneously.

   Externally written initial values are valid only if the <Mode> field in the SHA-1/MD5 Authentication Command Register (Table 487 p. 458) is selected.

   In SHA, the initial value is 5 words long and in MD5 it is 4 words long. The addresses of the IV/digest registers are specified in Table 482, SHA-1/MD5 Initial Value/Digest A Register, on page 456.

   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 when it is necessary to swap only part of the data.

SHA/MD5 algorithms work in 512-bit chunks, equal to 16 words. There are two ways to write the data words to the engine—Write cycles to Data In register, or Write cycles to the Data In register and to Byte Count registers.

11.1.1.1 Write Cycles to Data In Register

The host must perform 16 write cycles—first to W0, then to W1 through W15.

11.1.1.2 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 zero padding is skipped, thus less than 16 write accesses activate the engine.

This access is performed as follows:

1. The host writes between 0 and 13 words to the Data In register. The write to the Data In register is performed until the word containing 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. The host writes the lower part (MD5) or the higher part (SHA) of the bit count value to the SHA-1/MD5 Bit Count Low Register (Table 480 p. 456)—word 14 (see Table 43).
3. The host writes the higher part (MD5) or the lower part (SHA) of the bit count value to the SHA-1/MD5 Bit Count High Register (Table 481 p. 456)—word 15 (see Table 43). 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.

Table 43:Authentication of a Data Chunk

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 Authentication Command register or wait for the interrupt.

   After the engine has been loaded with the data chunk or after a write to the two Bit Count registers, the engine starts working automatically.

   When the <Termination> bit—bit[31] in the SHA-1/MD5 Authentication Command Register (Table 487 p. 458) is 1, this is an indication for the host that the engine has finished the calculation process and the digest is ready.

   The host can then poll the <Termination> bit or wait for the ZInt0 interrupt—bit 0 in the Cryptographic Engines and Security Accelerator Interrupt Cause Register (Table 519 p. 468). This interrupt is activated on the rising edge of the <Termination> bit. To clear the interrupt, the host must write a 0 to it. Writing a 1 has no effect.

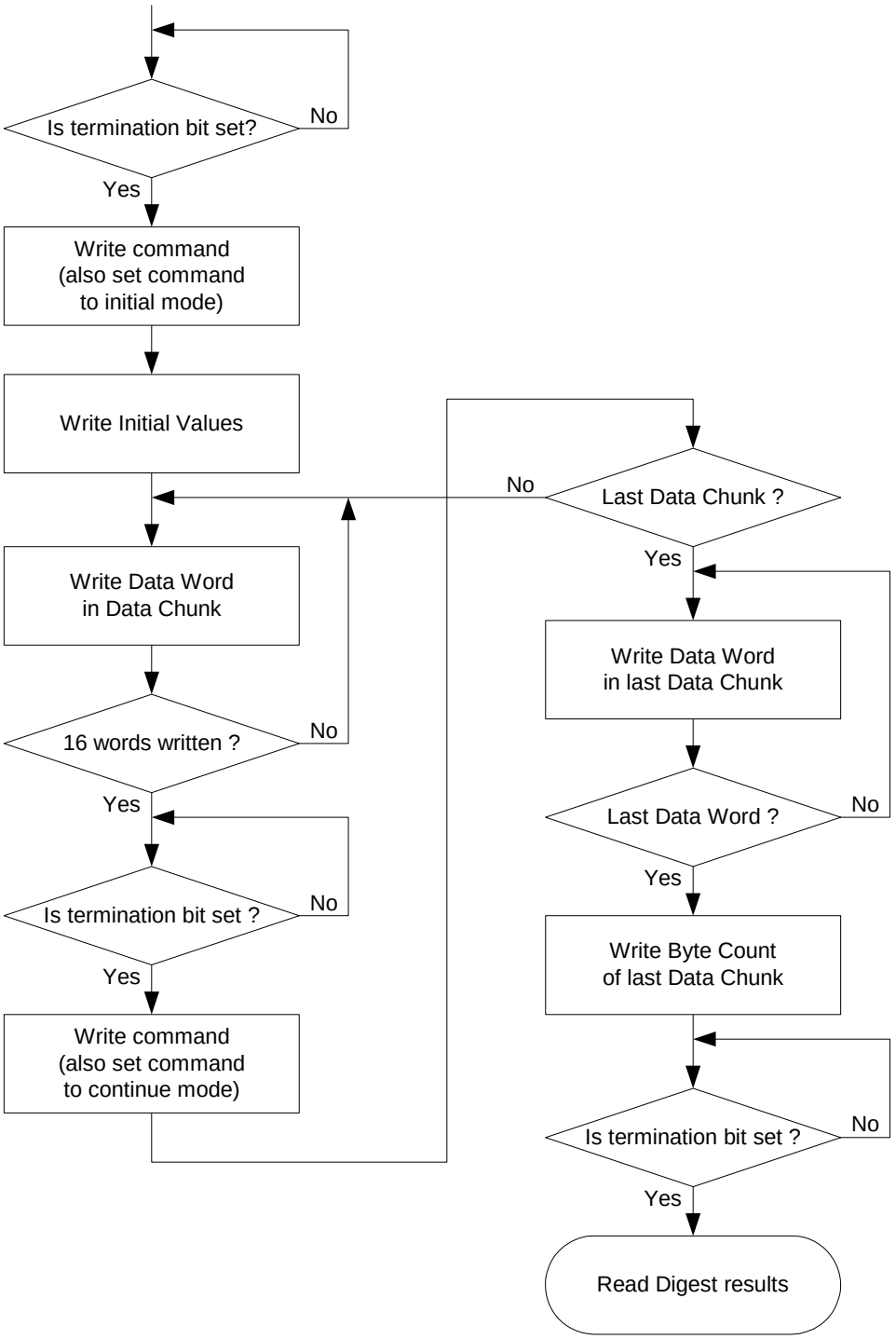
5. Read result

   Once the <Termination> bit has been asserted, the host can read the digest. The digest length is 4 words for SHA-1 or 5 words for MD5. These words are stored in the IV/Digest registers (see Table 482 on page 456 through Table 486 on page 457).

   After reading the result, the host can immediately start the write command again for initial values and data or just for data.

Figure 28 shows a typical authentication flow for a packet.

Figure 28: Typical Authentication Flow for a Packet



11.1.2 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 A 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 Table 470 on page 453 and Table 471 on page 453).

Two other modes—CFB and OFB—can 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 can work with three different keys for high security and can be combined with ECB or CBC modes.

All the modes are reciprocal, i.e., they decipher or cipher data.

Encryption/Decryption calculation time is nine cycles in DES mode and 25 cycles in 3DES mode. This is without taking into consideration the read and write cycles associated with writing the input data/key and reading the result.

To activate the Encryption engine, the following steps are required:

1. Verify termination bits in the DES Command Register.

   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 (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>—bit [30] in the DES Command Register (Table 478 p. 455)—must be set.

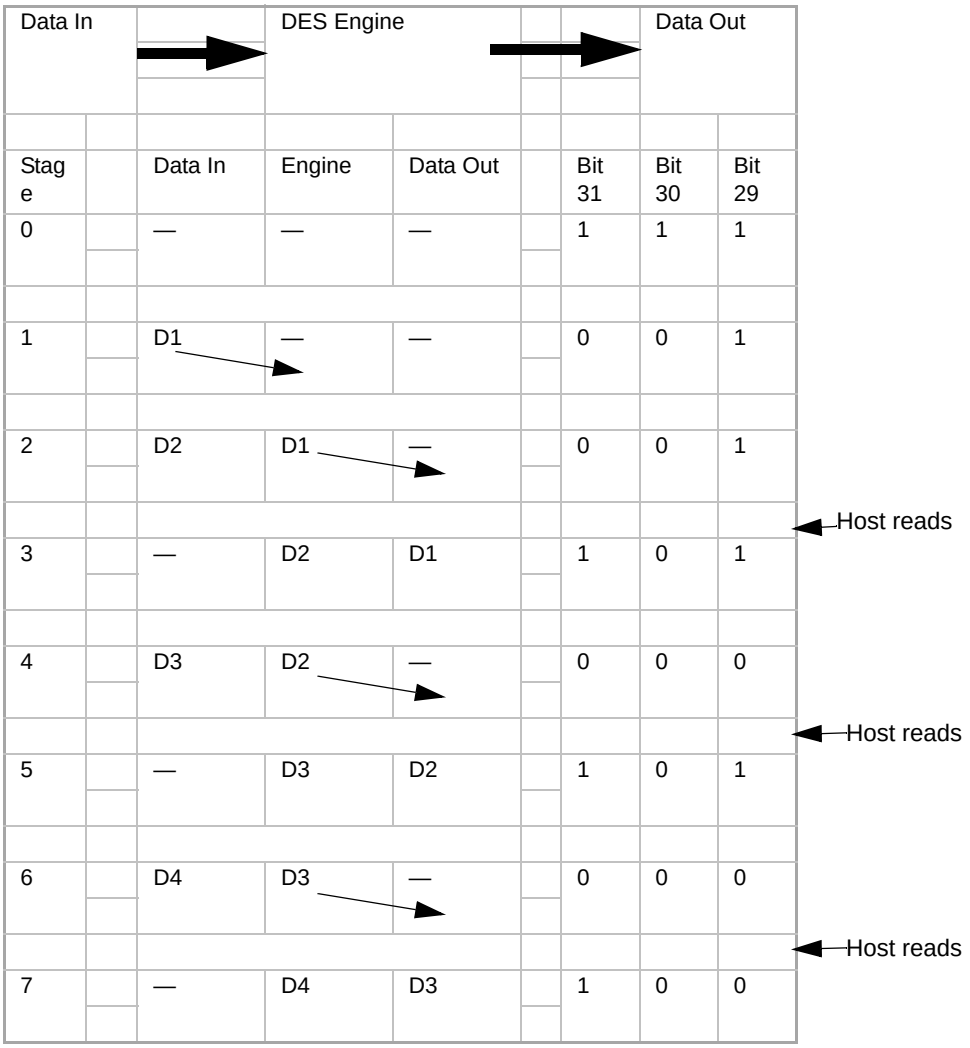
   When the <Termination> bit (bit [31] in the DES Command Register) is set, a 64-bit DES data block can be written to the engine, but this does not necessarily mean that parameters can be updated. Writing to the engine de-asserts this bit.

   However, when the <AllTermination> bit [30] 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 29.)

   The host writes to the engine when bit [29] is set. When a result is ready, the host must read it, to enable the engine to process the next data in the pipeline. When bit [29] is set, the host can write one data block to the engine.

Figure 29:DES Engine Pipeline



Note: A read of Dataout Low resets bit[29].

Meaning of bits [31:30] and their possible states.:

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 endianness for fields in the DES Command register:

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.
IV byte swap	Swapping of data written to the DES IV register. Similar to 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.

   In single DES mode, a write to a single key must be performed.The key used in DES mode is the KEY0 register.

   In 3DES modes, it is recommended to write to three different keys—KEY0, KEY1, and KEY2 registers.

   Since keys are only changed occasionally, this step is not always necessary.

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 IV_LOW and the other to IV_HIGH. 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, loaded by writing to one of the DES Data Buffer registers (see Table 468, p. 453 or Table 469, p. 453). One or two data blocks can be loaded, according to the engine status (see example below).

   If a data block is shorter than 64 bits, the specification requires zero padding to 64 bits.

   NOTE: 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 468, p. 453 and Table 469, p. 453). This is useful when it is necessary to encrypt a message consisting of multiple 64-bit blocks. In this case it is efficient to use the DES pipeline option and to write two blocks at once, as specified in the example below.

   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.
The host writes 0x11223344 to address 0xDD74.
```

Results:

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 (Table 478 p. 455) or wait for the interrupt.

   After the engine is loaded with one 64-bit block, it starts working automatically. The host must not write anything to the engine until the <ReadAllow> bit[29] is set.

   The host must poll the <Termination> bit[31] in the DES Command Register. When the bit is set to 1, this is an indication to the host that the engine has finished the calculation process and the result is ready.

   The host may write one data block each time, or it may perform consecutive writes of two data blocks:

   If the host wrote one data block each time:

   The result of the data block is ready and no more data is in the engine pipeline. Bit [30] <AllTermination> is set and engine parameters can be updated.

   If the host performed consecutive writes of two data blocks:

   When the first data block result was ready, the <Termination> bit[31] and the <ReadAllow> bit[29] were set and the engine had the second data in the pipeline. However the calculation could not start until the host had read the result of the first data via Data Out register (read the Data High before the Data Low). Here bit [30], <AllTermination>, was not set.

   Once the first data result was read, the <ReadAllow> bit[29] was reset and the engine started calculation of the second data. The host could then write one data block (the third block) to the engine. This was indicated by <ReadAllow> bit[29] (see Figure 29).

   When the second data block result was ready, <Termination> bit[31] was set. The engine had completed the second data calculation, the pipeline was empty, and the <AllTermination> bit [30] was also set. Once the data result was read, the host could alter the DES parameter and then write two data blocks.

   The ZInt1 interrupt is set every time the <AllTermination> bit—bit[30] in the DES Command Register (Table 478 p. 455)—changes from 0 to 1.

   The ZInt4 interrupt is set every time the <Termination> bit—(bit[31] in the DES Command Register)—changes from 0 to 1.

   After the interrupt occurs, the host writes 0 to the relevant interrupt bit in the Cryptographic Engines and Security Accelerator Interrupt Cause Register (Table 519 p. 468) 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 register.

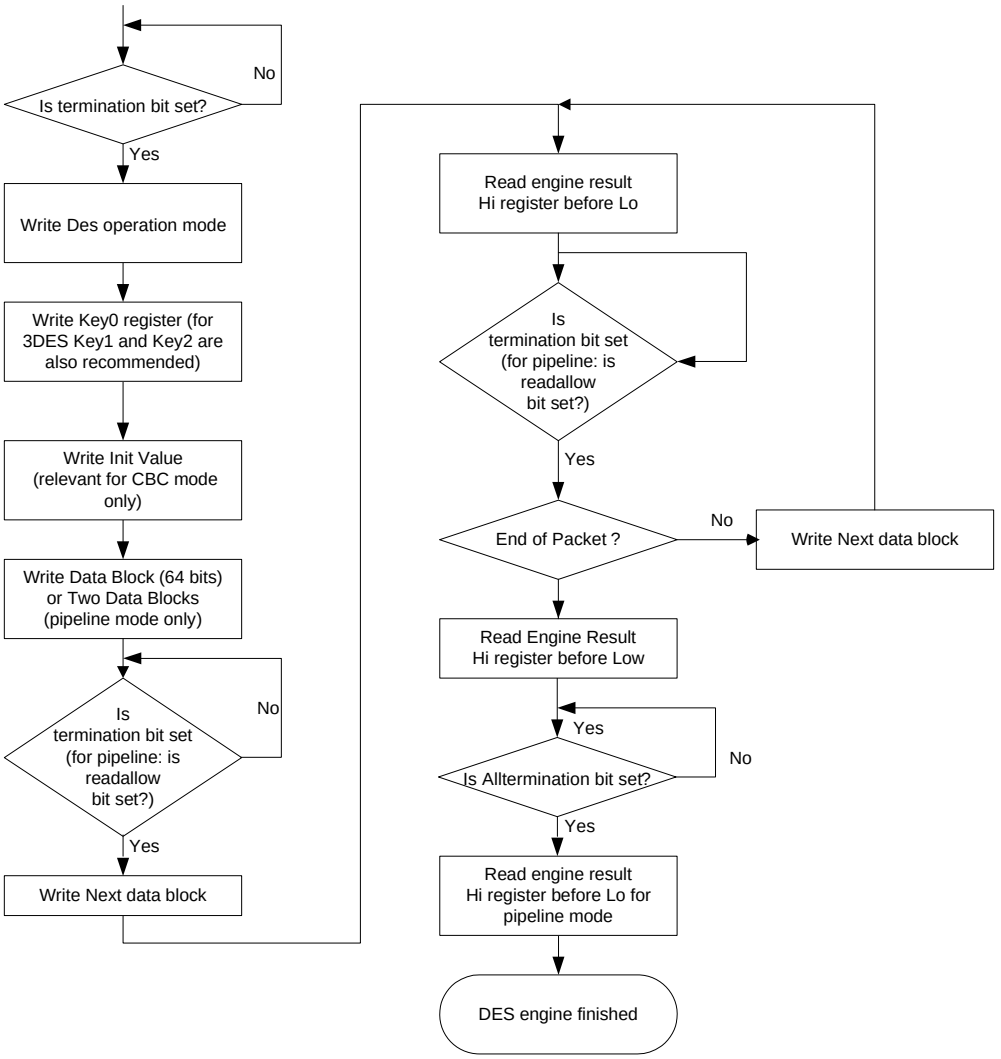
   Two read operations are required to read the result—the first from address 0xDD7C and the second from address 0xDD78.

   Byte swap bits have no effect on reads, i.e., the engine does not perform any endianness change on the result. To perform an endianness change, use the <OutByteSwap> field in the DES Command Register (Table 478 p. 455)).

8. Read 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 Table 466 on page 452 and Table 467 on page 453).

Figure 30 illustrates the typical DES/3DES packet encryption flow.

Figure 30: Typical DES/3DES Packet Encryption Flow



11.1.3 AES128 Encryption

The AES128 Encryption engine complies with the AES standard, as described in the FIPS standard. This engine performs encryption only. Decryption is performed by the AES128 Decryption engine.

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

Encryption calculation time is 20 cycles, without taking into consideration the read and write cycles associated with writing the input data/key and reading the result.

To activate the AES Encryption engine, the following steps are required.

1. Verify the termination bit in the AES Encryption Command Register (Table 500 p. 461).

   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 will result in erroneous data. Unless the host made a write to the engine before, the <Termination> bit (bit [31] in the AES Encryption Command Register 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 endianness for fields in the AES Encryption Command Register.
   - Key Length mode—128, 192 or 256 bit
   - Data byte swap. This is similar to the swapping performed in the Authentication engine. See Section 11.0.2 for further details.
   - Data out byte swap: These bits control byte swap of the cipher output result.

3. Write key.

   The AES key is pliable, according 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 containing 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 only changed occasionally, this step is not always necessary.

4. Write block for the 128-bit data.

   AES encryption requires a 128-bit block of input data, 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.

   NOTE: If the next block to be processed uses the same key; the host writes only the AES Data In/out registers (seeTable 488 on page 459 through Table 491 on page 460). This is useful for encrypting a message consisting of multiple 128-bit blocks.

   The AES machine starts working automatically when a 128-bit data block is written to it (i.e., operation starts after the host writes to all the AES data in/out addresses).

   This data is a 128-bit block, 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 (order does not matter), the AES cipher machine automatically starts working, as shown in the following example:

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 one word in width.

Results of this write operation:

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 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.

   The host must poll the <Termination> bit[31] in the AES Encryption Command Register (Table 500 p. 461). When the bit is set to 1, this is an indication to the host that the engine has finished the calculation process and the result is ready.

   Authentication calculation termination activates the ZInt2 interrupt (see the Cryptographic Engines and Security Accelerator Interrupt Cause Register (Table 519 p. 468)). It can serve as an alternative to host polling. This interrupt is set every time the <Termination> bit (bit [31] in the AES Encryption Command Register) changes from 0 to 1.

   After the interrupt occurs, the host writes 0 to bit [1] in the Cryptographic Engines and Security Accelerator Interrupt Cause Register 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 488 on page 459 through Table 491 on page 460).

   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 endianness change on the result. To perform an endianness change, use the AES Encryption Command Register (Table 500 p. 461) <OutByteSwap> bit [8].

7. Read keys.

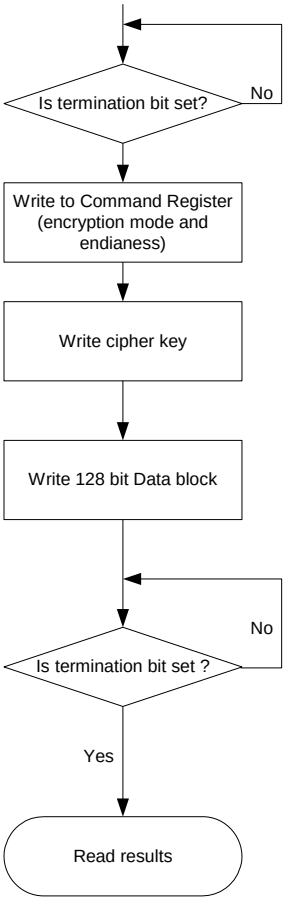
   It is possible to read the key values at any time, by accessing the appropriate registers. To perform an endianness change on key reads, use the AES Encryption Command Register <OutByteSwap> bit [8].

   The AES Encryption Command Register 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 11.1.4)

Figure 31 shows a typical AES encryption flow for a data block.

Figure 31: Typical AES Encryption Flow for a Data Block



11.1.4 AES128 Decryption

The AES128 Decryption engine complies with the AES standard, as described in the FIPS draft.

This engine only performs decryption. The modes of operation and activation are similar to AES encryption. The difference is that before loading the key, the host must calculate a decryption key. This process is performed by loading a key into the AES encryption unit, performing a “dummy” encryption cycle, then reading the resolved key from that engine (i.e., 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 an AES algorithm on a 128-bit data block size in three possible mode sizes:

- 128-bit mode
- 192-bit mode
- 256-bit mode

Encryption calculation time is 20 cycles, without taking into consideration the read and write cycles associated with writing the input data/key and reading the result.

To activate the AES Decryption engine, the following steps are required:

1. Calculate decryption key:

   This step is unique to the AES Decryption engine and is only necessary when a new key (i.e., one that 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 Encryption engine, then start the encryption process with any “dummy” data. At the end of the encryption process the host reads the key registers from the Encryption engine. This decryption 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 <AesKeyRdMode> bit to 1 prior to reading the AES encryption key registers. Setting this bit enables reading of the internal key in the AES Encryption engine. At the end of an encryption process this 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.

2. Verify the termination bit in the AES Decryption/Encryption Command register[1].
3. Write operational mode and endianness for fields in the AES Decryption Command Register (Table 513 p. 465).
4. Write the decryption key1.
5. Write the block for the 128-bit data1.
6. Poll the AES Decryption/Encryption Command register or wait for the interrupt.
7. Read AES result1.
8. Read keys:

   It is possible to read the key values at any time, by accessing the appropriate registers. To perform an endianness change on key reads, use bit [8] <OutByteSwap> of the AES Encryption Command Register (Table 500 p. 461) or AES Decryption Command Register (Table 513 p. 465).

   The AES Decryption Key registers (see Table 505 on page 463 through Table 512 on page 464) 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 in Step 1 above).

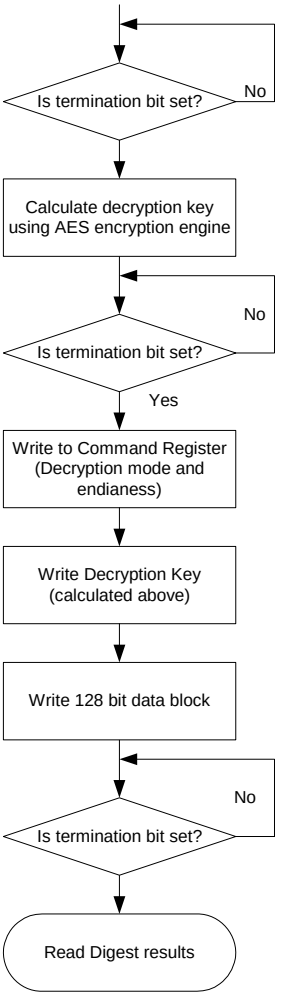
   NOTE: Directly loading an encryption key to these registers returns incorrect results!

   The AES Decryption Command Register controls the AES decryption operation. It also flags when the processing is complete.

1. Same as for AES encryption (see Section 11.1.3).

Figure 32 shows a typical AES decryption flow for a data block.

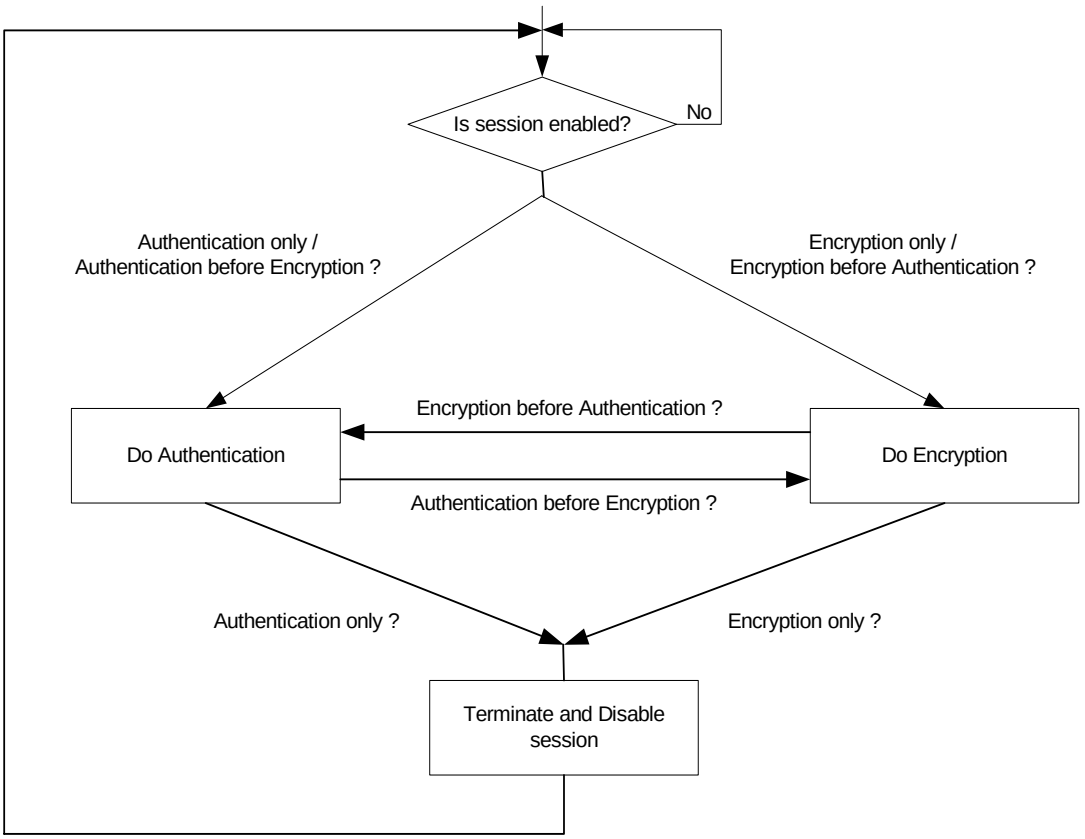
Figure 32: Typical AES Decryption Flow for a Data Block



11.2 Security Accelerator Operation

Figure 33 shows the main accelerator decision flow.

Figure 33: Security Accelerator Main Decision Flow



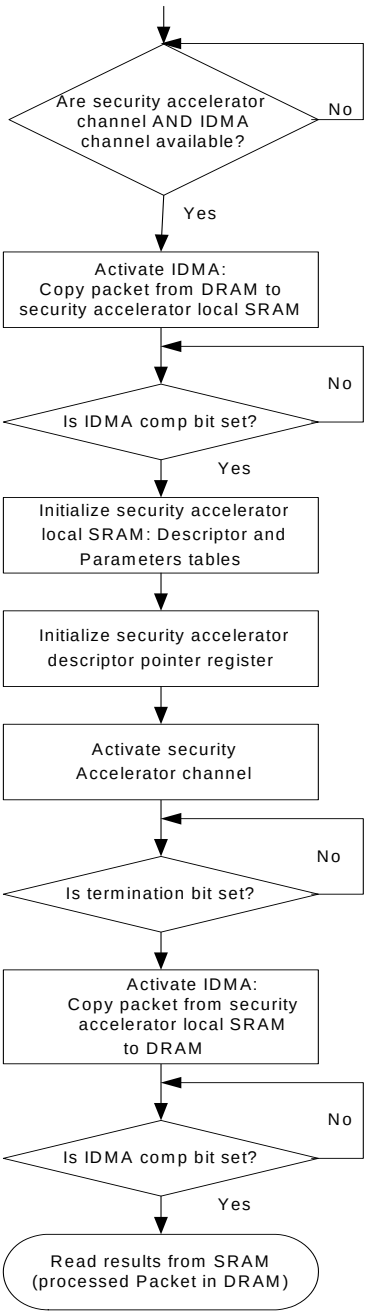
The managing software performs the following:

1. Uses the IDMA to copy the packet from main memory into the local SRAM of the security accelerator. Since the local SRAM of the security accelerator is only 8 KB, it cannot contain large packets.
2. Receives the cryptographic parameters relevant for the security accelerator operation to be undergone by the packet.
3. Prepares a descriptor in the local SRAM of the security accelerator, stating the required operation and parameters for the accelerator. The descriptor, 8 DWords long, is described in Section 11.2.4.
4. Writes the pointer to this descriptor into the selected session, the <SecurityAccl DescPtr0> field in the Security Accelerator Descriptor Pointer Session 0 Register (Table 515 p. 466) or the <SecurityAccl DescPtr1> field in the Security Accelerator Descriptor Pointer Session 1 Register (Table 516 p. 466).
5. Activates the programmed session, by setting the appropriate bit in the Security Accelerator Command Register (Table 514 p. 465).

6. Waits for the session completion indication either by polling the Security Accelerator Status Register (Table 518 p. 467) or by interrupt. In the meantime, a new session can be prepared and activated. The accelerator will start processing the second session as soon as the first has been completed.
7. Uses the IDMA to copy the processed packet back into main memory.

Figure 34 illustrates the security acceleration flow for packet processing and Figure 35 illustrates the enhanced mode.

Figure 34: Security Acceleration Flow for Packet Processing



11.2.1 Attach IDMA to Security Accelerator—Enhanced Software Flowchart

To perform the following flow, set bits <Ch0WaitFor IDMA>, <Ch1WaitFor IDMA>, <Ch0Activate IDMA>, and <Ch1Activate IDMA> in the Security Accelerator Configuration Register (Table 517 p. 467).

When the security accelerator operates in conjunction with IDMA, it can operate only with the external DRAM. It can activate the IDMA, sense its status, and provide a single completion interrupt (see Figure 35). The first five steps in the flow are performed by software (SW) and the remaining steps by hardware (HW).

Figure 35: Security Acceleration Flow for Packet Processing—Enhanced Mode

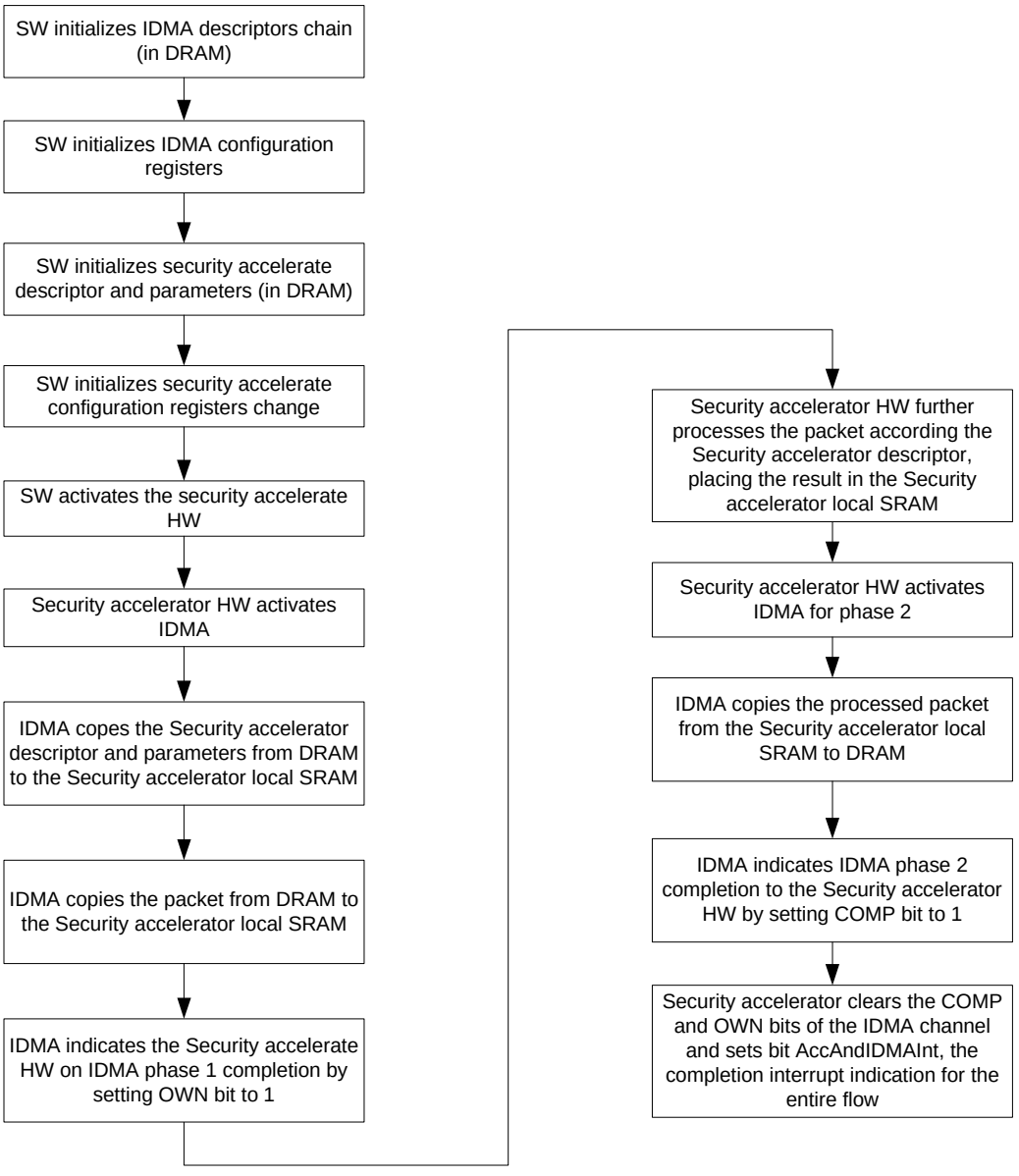
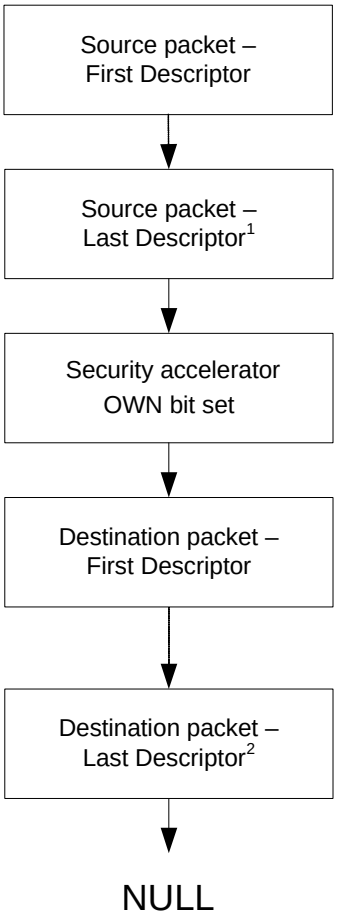


Figure 36: IDMA Channel Descriptors Structure for Security Accelerator Packet Processing—Enhanced Mode



[1]This step only applies if there is more that one source packet descriptor.

[2]This step applies if there is more that one destination packet descriptor.

11.2.2 Encryption Operation

Initialization

Initialization is carried out as follows:

1. Encryption mode is read from the Security Accelerator Data Structure DWord 0—Configuration (Table 44 p. 155) and the configuration register of the selected cryptographic engine is written.
2. Source and destination pointers are read from Security Accelerator Data Structure DWord 1—Encryption Pointers (Table 45 p. 156).
3. Number of bytes to be encrypted is read from Security Accelerator Data Structure DWord 2—Encryption Data Length (Table 46 p. 156).
4. Keys for encryption are read from the pointer specified in Security Accelerator Data Structure DWord 3—Encryption Keys Pointer (Table 47 p. 156) and are written to the selected cryptographic engine.

5. When the mode selected is CBC, initial values are read from the pointer specified in Security Accelerator Data Structure DWord 4—Encryption Initial Values Pointer (Table 48 p. 157). For DES/3DES, initial values are written to the engine.

Data Processing

Data processing is carried out as follows:

1. Data is read from the source pointer, block by block (8 bytes for DES/3DES, 16 bytes for AES) for the specified data size. If the size is not a multiple of block size, the last block is padded with zeros.
2. Each block is fed 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, the first block data is XORed with the initial values before it is written 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 of AES CBC decryption, the result of the first block processing is XORed with the initial values before it is written to the destination. After that, 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, the last block of the destination data is written 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, the last block of source data is written to memory, according to the pointer specified in the Security Accelerator Data Structure DWord 4—Encryption Initial Values Pointer.

11.2.3 Authentication Operation

Initialization

Initialization is carried out as follows:

1. Authentication mode is read from the Security Accelerator Data Structure DWord 0—Configuration (Table 44 p. 155) and the configuration register of the Authentication engine is written.
2. The source pointer is read from the Security Accelerator Data Structure DWord 5—MAC Source Pointer (Table 49 p. 157).
3. The Digest location pointer and the number of bytes in the message are read from the Security Accelerator Data Structure DWord 6—MAC Digest (Table 50 p. 157).
4. When the direction is “Decode”, the original digest is first read and stored internally, then it is overwritten with zeros.
5. When using the HMAC operation, the inner initial values pointer is read from the Security Accelerator Data Structure DWord 7—MAC Initial Values Pointers (Table 51 p. 158), and the values are written to the IV registers of the Authentication engine.

Data Processing

Data processing is carried out as follows:

1. Data is read from the source pointer block by block (64 bytes per block).
2. The last chunk of data is padded 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, the outer initial values pointer is read from the Security Accelerator Data Structure DWord 7—MAC Initial Values Pointers (Table 51 p. 158), and the values are written 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 (now reflecting the opad string length).

Termination

Termination is carried out as follows:

1. The resulting digest is read from the engine, and stored at the location that has been read from Security Accelerator Data Structure DWord 6—MAC Digest.
2. When the direction is “Decode”, the digest is also compared to the copy extracted from the original message. If the digests are not identical, an indication bit is set in the Security Accelerator Status Register (Table 518 p. 467).
   If the <StopOnDecodeDigestErr> field in the Security Accelerator Configuration Register (Table 517 p. 467) was set, the session is immediately aborted.

11.2.3.1 Security Accelerator in Continuous Mode

The Security Accelerator supports Continuous mode.

In this mode, the Security Accelerator can process fragmented packets one at a time. Only single packets need to reside in the Security Accelerator local SRAM, while the total size of the packet is actually unlimited.

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.

Note Both sessions may be used, as long as the fragment order is maintained.

1. Processing the First Fragment:
   Operation starts the same as in Non-fragmented mode. Finalization is not performed. In encryption, IV is not stored. In authentication, data is not padded, the outer operation in HMAC is not performed, and the digest is not written or compared.
2. Processing the Middle Fragment (Not First And Not Last):
   Initializations are not performed. In encryption, keys and IVs are not loaded into the engines. In authentication, the digest is not read or cleared. Inner operation of HMAC is not performed. Finalization is not performed, as for the first fragment.
3. Processing the Last Fragment:
   Initializations are not performed, as for a middle fragment. Finalization is performed as in Non-fragmented mode.

11.2.4 Security Accelerator Descriptor Data Structure

The descriptor data structure is described in Table 44 through Table 51.

Table 44: 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
6:4	MacMode	100 = MD5 101 = SHA1 110 = HMAC-MD5 111 = HMAC-SHA1 All other combinations are reserved (no operation).
7	AuthResultLen	Authentication result length 0 = Full size (128 bit in MD-5, 160b in SHA-1) 1 = 96b
9:8	EncryptMode	00 = Reserved (no operation) 01 = DES 10 = 3DES 11 = AES
11:10	Reserved	Reserved
12	Direction	0 = Encode 1 = Decode
15:13	Reserved	Reserved
16	EncryptConfidentialityMode	0 = ECB 1 = CBC
19:17	Reserved	Reserved
20	3DESMode	0 = EEE 1 = EDE Relevant only in 3DES encryption mode.
23:21	Reserved	Reserved
25:24	AESKeyLength	00 = 128 bit key 01 = 192 bit key 10 = 256 bit key 11 = Reserved Relevant only in AES encryption mode.
29:26	Reserved	Reserved
31:30	FragMode	Fragmentation mode 00 = Not fragmented 01 = First Fragment in packet 10 = Last Fragment in packet 11 = Middle Fragment in packet

Table 45: Security Accelerator Data Structure DWord 1—Encryption Pointers

Bits	Field	Function
12:0	EncrypSourceDataPtr	Pointer to the first DWord of data to encrypt (DWord aligned) Bits [2:0] must be 0.
15:13	Reserved	Reserved
28:16	EncrypDesDataPtr	Pointer to the first DWord of encrypted data (DWord aligned) Bits [18:16] must be 0.
31:29	Reserved	Reserved

Table 46: Security Accelerator Data Structure DWord 2— Encryption Data Length

Bits	Field	Function
12:0	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.
31:13	Reserved	Reserved

Table 47: Security Accelerator Data Structure DWord 3—Encryption Keys Pointer

Bits	Field	Function
12:0	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.
31:13	Reserved	Reserved

Table 48: Security Accelerator Data Structure DWord 4—Encryption Initial Values Pointer

Bits	Field	Function
12:0	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
15:13	Reserved	Reserved
28:16	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.
31:29	Reserved	Reserved

Table 49: Security Accelerator Data Structure DWord 5—MAC Source Pointer

Bits	Field	Function
12:0	MACSourceDataPointer	Pointer to the first DWord of data to MAC (DWord aligned) Bits [2:0] must be 0.
15:13	Reserved	Reserved
31:16	TotalMacDataLength	In MAC Non Fragment mode, this field should be equal to MacDataLength (see Table 50). In the last MAC fragment, it should be equal to the total data lengths of all the packet fragments.

Table 50: Security Accelerator Data Structure DWord 6—MAC Digest

Bits	Field	Function
12:0	MACDigestPointer	Byte location in which digest is stored during encoding or should be stored during decoding Bits [2:0] must be 0.
15:13	Reserved	Reserved
28:16	MACDataLength	Numbers of bytes to MAC
31:29	Reserved	Reserved

Table 51: Security Accelerator Data Structure DWord 7—MAC Initial Values Pointers

Bits	Field	Function
12:0	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-SHA1 MIIV[1] = Inner IV 1 of HMAC-MD5/HMAC-SHA1 MIIV[2] = Inner IV 2 of HMAC-MD5/HMAC-SHA1 MIIV[3] = Inner IV 3 of HMAC-MD5/HMAC-SHA1 MIIV[4] = Inner IV 4 of HMAC-SHA1 Bits [2:0] must be 0.
15:13	Reserved	Reserved
28:16	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-SHA1 MOIV[1] = Outer IV 1 of HMAC-MD5/HMAC-SHA1 MOIV[2] = Outer IV 2 of HMAC-MD5/HMAC-SHA1 MOIV[3] = Outer IV 3 of HMAC-MD5/HMAC-SHA1 MOIV[4] = Outer IV 4 of HMAC-SHA1 Bits [18:16] must be 0.
31:29	Reserved	Reserved

12 Two-Wire Serial Interface (TWSI)

12.1 Functional Description

The 88F5182 provides full Two-Wire Serial Interface (TWSI) support. It can act as master generating read/write requests and as a slave responding to read/write requests. It fully supports a multiple TWSI-masters environment (clock synchronization, bus arbitration).

The TWSI interface can be used for various applications. It can be used to control other TWSI on board devices, to read DIMM SPD ROM, and for serial ROM initialization. For more details, see the Reset Pins and Configuration section in the *88F5182 Feroceon*® Storage Networking SoC*, Datasheet*.

The TWSI port consists of two open drain signals:

- TW_SCK (Serial Clock)
- TW_SDA (Serial address/data)

The TWSI 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 TWSI slave responds with acknowledge.

In the case of a write access (R/W bit is 0, following the TWSI slave acknowledge), the master drives 8-bit data and the slave responds with acknowledge. This write access (8-bit data followed by acknowledge) continues until the TWSI master ends the transaction with a stop condition.

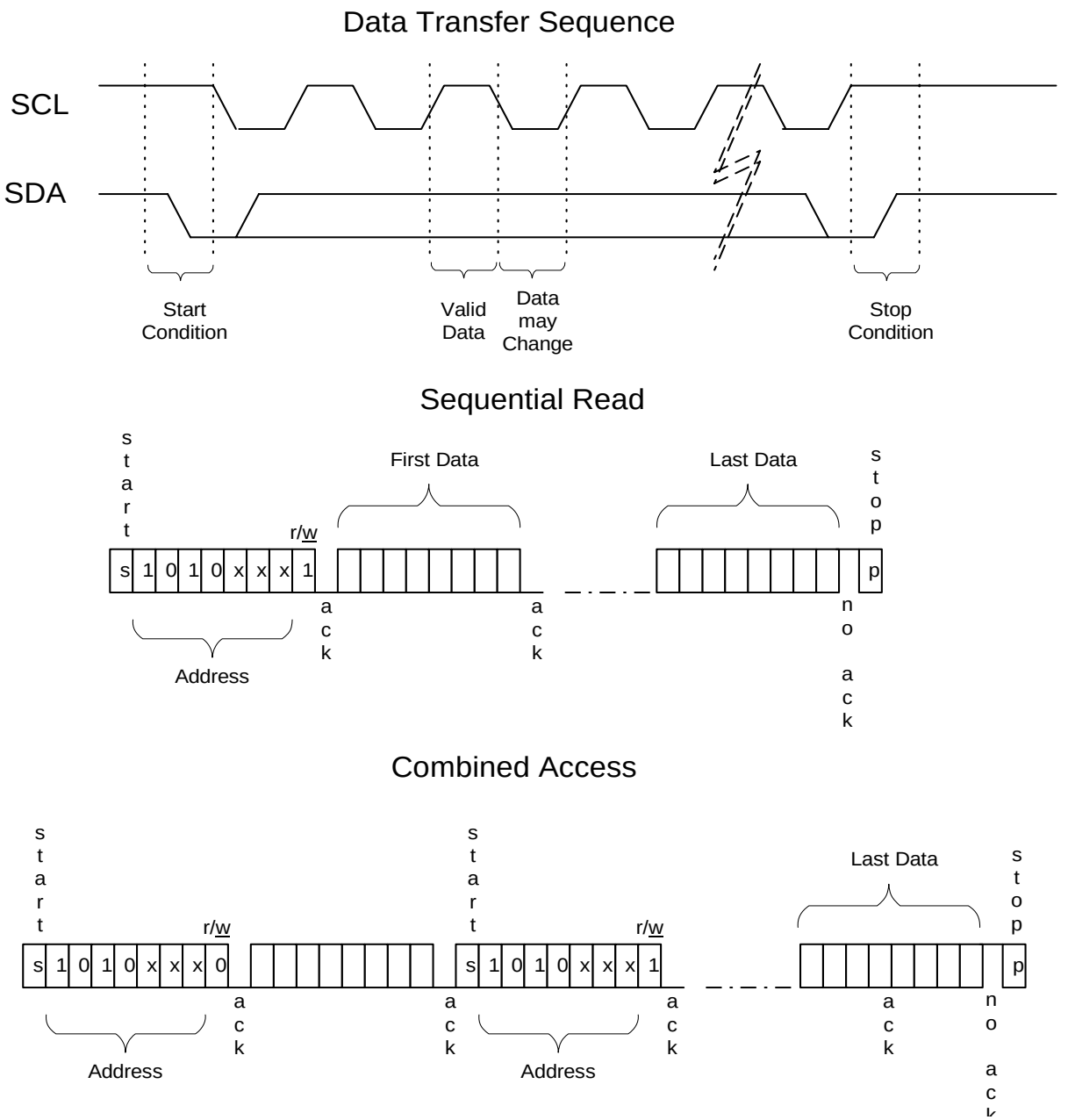
In the case of read access following the TWSI slave address acknowledge, the TWSI slave drives 8-bit data and the master responds with acknowledge. This read access (8-bit data followed by acknowledge) continues until the TWSI master ends the transaction by responding with no acknowledge to the last 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 TW_SCK low until it has valid data to drive on the TW_SDA line.

A master is allowed to combine two transactions. After the last data transfer, it can drive a new start condition followed by a new slave address, rather than driving a stop condition. Combining transactions guarantees that the master does not loose arbitration to some other TWSI master. The TWSI interface master and slave activities are handled by Feroceon® CPU core access to internal registers, plus the interrupt interface.

TWSI examples are shown in Figure 37.

Figure 37: TWSI Examples



12.1.1 TWSI Slave Addressing

The TWSI slave interface supports both 7-bit and 10-bit addressing. The slave address is programmed by the TWSI Slave Address register (see Table 522 on page 470) and TWSI Extended Slave Address register (see Table 523 on page 470).

When the TWSI receives a 7-bit address after a start condition, it compares it against the value programmed in the Slave Address register, and if it matches, it responds with acknowledge.

If the received 7 address bits are 11110xx, meaning that it is an 10-bit slave address, the TWSI compares the received 10-bit address with the 10-bit value programmed in the TWSI Slave Address (Table 522 p. 470) and TWSI Extended Slave Address (Table 523 p. 470) registers, and if it matches, it responds with acknowledge.

The TWSI interface also support slave response to general call transactions. If the <GCE> bit in the TWSI Slave Address register is set to 1, the TWSI also responds to the general call address (0x0).

12.1.2 TWSI Data

An 8-bit TWSI Data (Table 524 p. 471) register is used both in master and slave modes.

In master mode, the Feroceon CPU core must place the slave address or write data to be transmitted. In the case of read access, it contains received data (need to be read by the Feroceon CPU core).

In slave mode, the Data register contains data received from master on write access, or data to be transmitted (written by the Feroceon CPU core) on read access. TWSI Data register MSB contains the first bit to be transmitted or being received.

12.1.3 TWSI Control

An 8-bit TWSI Control (Table 525 p. 471) register is used both in master and slave modes.

12.1.4 TWSI Status

An 8-bit TWSI Status (Table 526 p. 473) register is used both in master and slave modes.

This 8-bit register contains the current status of the TWSI interface. Bits [7:3] are the status code, bits [2:0] are Reserved (read only 0).

12.1.5 Baud Rate Register

TWSI specification defines TW_SCK frequency of 100 KHz (400 KHz in fast mode). The TWSI module contains a clock divider to generate the TW_SCK clock. Setting bits[6:0] of TWSI Baud Rate (Table 527 p. 474) register (offset 0x1100C) defines TW_SCK frequency as follows:

Table 52: Setting the Baud Rate Register

TClk	N	M	TWSI Frequency (in kHz)
166 MHz	3	10	94.3
	3	13	74.1
	4	9	51.8
	6	12	99.7

$$F_{TWSCK} = \frac{F_{TClk}}{10 \cdot (M + 1) \cdot 2^{(N+1)}}$$

Note Where M is the value represented by bits [6:3] and N the value represented by bits [2:0]. If for example *M=N=4* (the default values), running *TCLK* at 100 MHz results in *SCL* frequency of 62.5 kHz.

As defined in the TWSI specification, the maximum supported TW_SCK frequency is 100 kHz. Fast mode (where TW_SCK frequency is 400 kHz) is not supported.

12.2 TWSI Master Operation

The Feroceon CPU core can initiate TWSI master read and write transactions via TWSI registers, as described in the following sections.

12.2.1 Master Write Access

A master write access consists of the following steps:

1. The Feroceon CPU core sets the <Start> bit in the TWSI Control register (see Table 525 on page 471) to 1. The TWSI 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 Feroceon CPU core writes 7-bit address plus a write bit to the TWSI Data register (see Table 524 on page 471) and clears Interrupt flag for the TWSI 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 TWSI device has an 10-bit address, the Feroceon CPU core needs to write the remainder 8-bit address bits to the Data register. The Feroceon CPU core 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 TWSI Status register (see Table 526 on page 473).
3. The Feroceon CPU core writes data byte to the TWSI Data register, and then clears Interrupt flag for the TWSI master interface to drive the data on the bus. The target slave responds with acknowledge, causing Interrupt flag to be set, and status code of 0x28 be registered in the Status register. The Feroceon CPU core continues this loop of writing new data to the Data register and clear Interrupt flag, as long as it needs to transmit write data to the target.
4. After the last data transmit, the Feroceon CPU core may terminate the transaction or restart a new transaction. To terminate the transaction, the Feroceon CPU core sets the Control register's <Stop> bit and then clears the Interrupt flag, causing the TWSI master to generate a stop condition on the bus, and go back to idle state. To restart a new transaction, the Feroceon CPU core sets the TWSI Control register's <Start> bit and clears the Interrupt flag, causing TWSI master to generate a new start condition.

Note This 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 TWSI Status register and needs to be handled by the Feroceon CPU core.

12.2.2 Master Read Access

A master read access consists of the following steps:

1. Generate, a start condition, exactly the same as in the case of write access, see Section 12.2.1 Master Write Access.
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 the case of a 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, an interrupt flag is set, and status code of 0x50 is registered in the Status register. The Feroceon CPU core 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 Feroceon CPU core 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 Feroceon CPU core clears the TWSI Control register's Acknowledge bit (when clearing the Interrupt bit), causing the TWSI 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 Feroceon CPU core can issue a stop condition or a new start condition.

Note

The above 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 Feroceon CPU core.

12.3 TWSI Slave Operation

The TWSI 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.

The two cases are described in the following sections.

12.3.1 Slave Read Access

Upon detecting a new address driven on the bus with read bit indication, the TWSI 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 status code to 0xA8.

Note

If the TWSI slave address is 10-bit, the Interrupt flag is set and status code changes only after receiving and identify address match also on the 2nd address byte).

The Feroceon CPU core now must write new read data to the Data register and clears the Interrupt flag, causing TWSI 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, status code 0f 0xC0 is registered, and TWSI slave interface returns back to idle state.

If the master generates a stop condition after driving an acknowledge bit, the TWSI slave interface returns back to idle state.

12.3.2 Slave Write Access

Upon detecting a new address driven on the bus with write bit indication, the TWSI slave interface compares the address against the address programmed in the Slave Address register and, if it matches, responds with acknowledge. It also sets an Interrupt flag, and sets status code to 0x60 (0x70 in the case of general call address, if general call is enabled).

Following each write byte received, the TWSI slave interface responds with acknowledge, sets an Interrupt flag, and sets status code to 0x80 (0x90 in the case of general call access). The Feroceon CPU core 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.

13 UART Interface

13.1 Functional Description

The 88F5182 supports two Universal Asynchronous Receiver/Transmitter (UART) ports. One of the UART ports is multiplexed on the MPP port.

The UART is integrated into the device to support data input/output operations for peripheral devices connected through a standard UART interface.

The UART includes the following features:

- FIFO mode permanently selected for transmit and receive operations
- Modem control functions (CTSn, RSTn)

13.2 UART Interface Pin Assignment

The 88F5182 supports the UART interface through the UA0_TXD and UA0_RXD pins and provides modem control functions through the UA0_CTSn and UA0_RTSn pins.

Table 53 shows the signal names on the 88F5182 and the description of the pins.

Table 53: UART Pin Assignments

Pin Name	Type	Description
UA0_TX UA1_TX	O	The UA0_TX signals are the serial data output to the modem, data set, or peripheral device. The UA0_TX signals are set high when the reset is applied.
UA0_RX UA1_RX	I	The UA0_RX signals are the serial data input from the modem, data set, or peripheral device.
UA0_CTSn UA1_CTSn	I	CLEAR TO SEND: When low, these pins indicate that the receiving UART is ready to receive data. When the receiving UART de-asserts UA0_CTSn high, the transmitting UART stops transmission to prevent overflow of the receiving UART buffer.
UA0_RTSn UA1_RTSn	O	REQUEST TO SEND: When low, these pins informs the remote device that the UART is ready to receive data.

Note

For further information see Section A.13, UART Interface Registers, on page 475.

14 Device Controller Interface

14.1 Functional Description

The 88F5182 Device controller interface supports up to four banks of devices. Each bank supports up to 512 MB of address space. Each bank has its own timing parameters register. Bank width can be programmed to 8- or 16-bits. Bank timing parameters can be programmed to support different device types (e.g., sync burst SRAM, Flash, ROM, I/O Controllers).

The four chip selects are typically separated into three individual chip selects and one chip select for a boot device. The boot device bank is the same as any of the other banks except that the core boots from the boot device and its default width is sampled at reset.

The device controller multiplexes the address and data buses. The interface latches the address into latches to support up to 512 MB of address space. The interface supports any size access up to 128 bytes. See Table 54 for device controller pin assignment.

Note All output signals are driven with the rising edge of TCLK and all inputs are sampled with the rising edge of TCLK.

14.2 Device Interface Pin Assignment

Table 54 provides a list of the Device interface pins and describes their function.

Table 54: Device Controller Pin Assignments

Pin Name	Type	Description
DEV_CEn[2:0]	O	Device Bus Chip Enable correspond to bank [2:0]
DEV_BootCEn	O	Device Bus Chip Enable correspond to Boot Bank
DEV_OEn	O	Device Bus Output Enable
DEV_WEn[1:0]	O	Device Bus Write Enable
DEV_ALE[1:0]	O	Device Bus Address Latch Enable correspond to ALE[0].
DEV_D[8:0]	t/s I/O	Device Bus Multiplexed Address/Data bus
DEV_D[15:9]	t/s I/O	Device Bus Data bus
DEV_A[2:0]	O	Device Bus Address
DEV_READY	I	Device Ready
DEV_BURSTn/DEV_LASTn	O	Device Burst/Device last

14.3 Device Interface Block Diagram

Figure 38 provides a diagram of the Device block.

Figure 38: Device Block Diagram Example

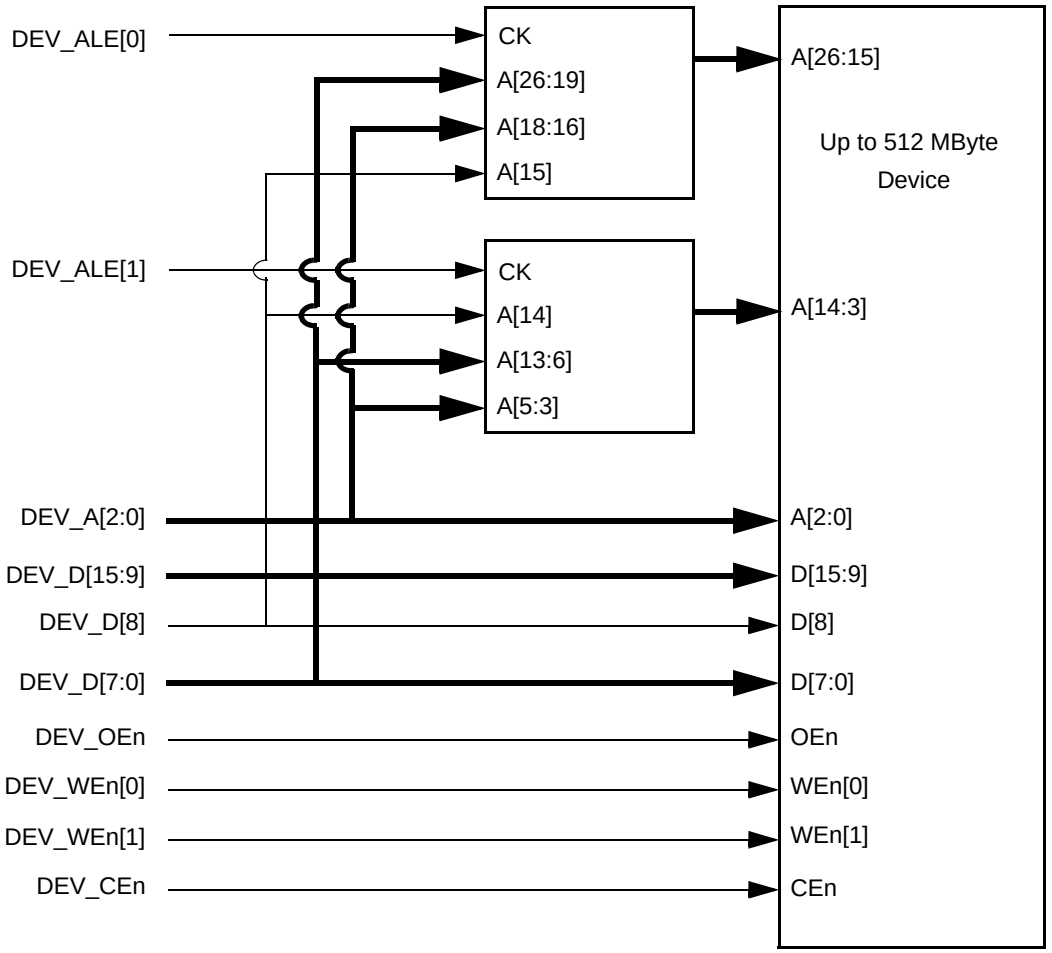
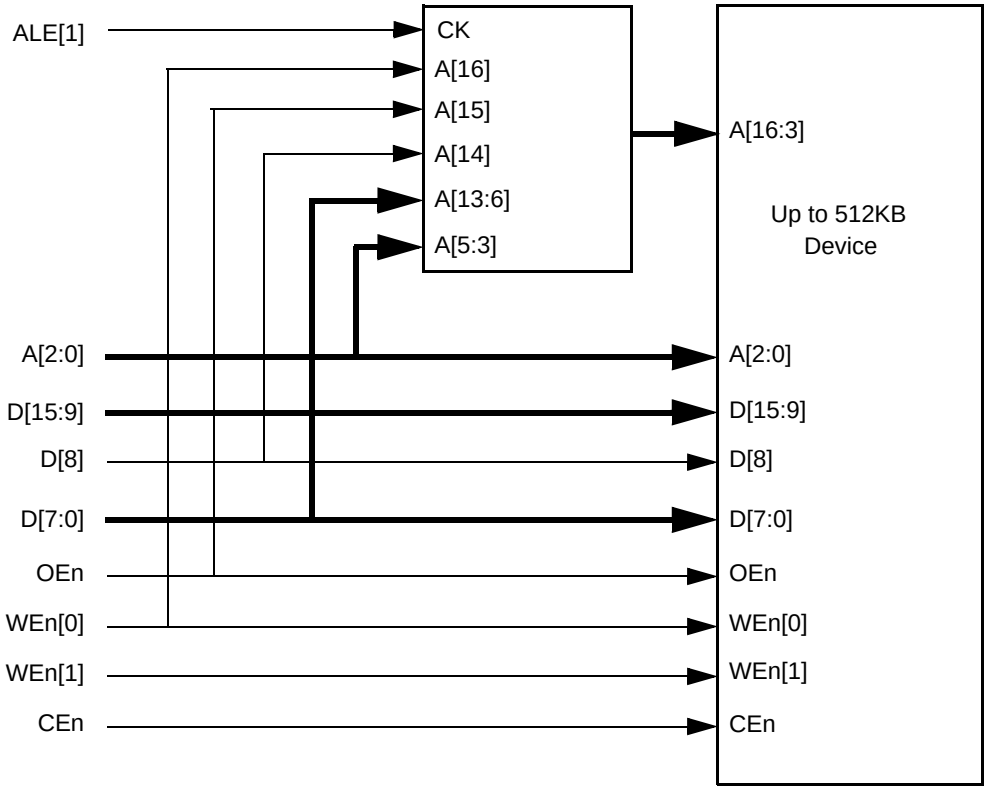


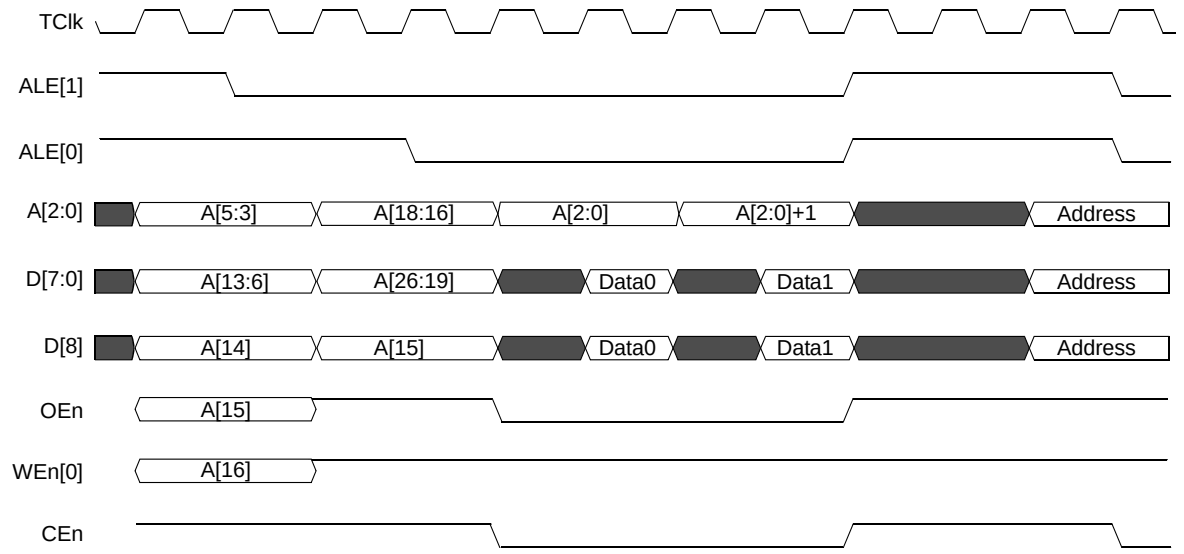
Figure 39: Up to 512-KB Device with Single Latch Block Diagram Example



14.4 Address Multiplexing

Figure 40 provides a diagram of the address multiplexing.

Figure 40: Address Multiplexing



14.5 Device Interface Read Timing Parameters

To allow flexible interfacing to slow and fast devices, the interface can be programmed with different timing parameters.

14.5.1 TurnOff

The TurnOff parameter defines the number of TCLK cycles that the 88F5182 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 minimum setting of this parameter is 0x2.

14.5.2 Acc2First

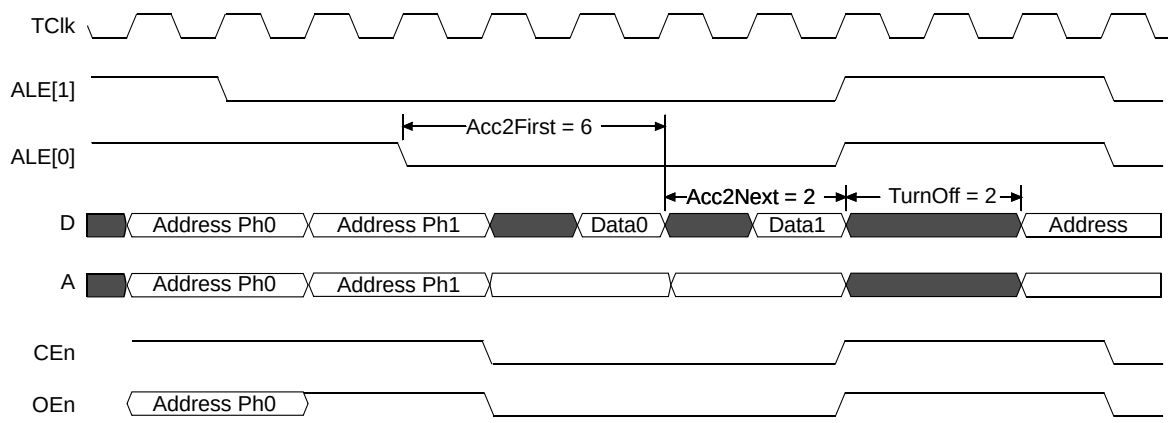
The Acc2First parameter defines the number of TCLK cycles from the negation of ALE[0] to the cycle that the first read data is sampled by 88F5182. Extend this parameter by extending the READYn pin, see Section 14.8, READYn Support, on page 172. The number of cycles = <Acc2First> - 3. The minimum setting of this parameter is 0x6.

14.5.3 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). Extend this parameter by extending the READYn pin, see Section 14.8, READYn Support, on page 172. The minimum setting of this parameter is 0x2.

Figure 41 shows a device read timing parameters example.

Figure 41: 8-bit Flash Read Parameters Example



14.6 Device Interface Write Timing Parameters

To allow flexible interfacing to slow and fast devices, the interface can be programmed with different timing parameters.

14.6.1 ALE2Wr

The ALE2Wr parameter defines the number of TCLK cycles from the ALE[0] negation cycle to the WEn assertion. The number of cycles = <ALE2Wr> - 3. The minimum setting of this parameter is 0x4.

14.6.2 WrLow

The WrLow parameter defines the number of TCLKs that WEn is active (low). Extend this parameter by extending the READYn pin (see Section 14.8, READYn Support, on page 172). A[2:0] and Data are kept valid as long as WEn is active. This parameter defines the setup time of address and data to WEn rise. The minimum setting of this parameter is 0x1.

14.6.3 WrHigh

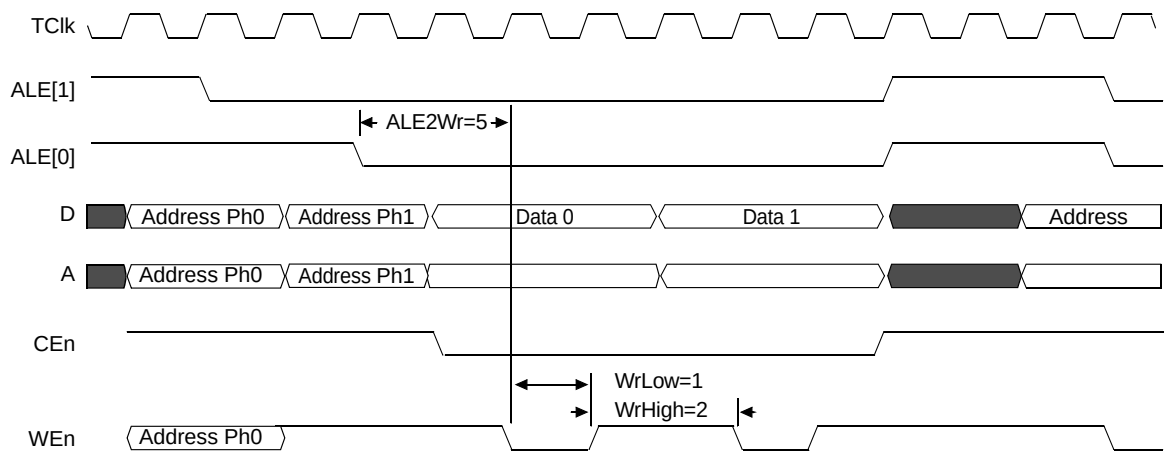
The <WrHigh> parameter defines the number of TCLK cycles that WEn is kept inactive (high) between data beats of a burst write. A[2:0] and Data are kept valid (do not toggle) for <WrHigh>-1 periods. This parameter defines the hold time of address and data after WEn rise. The minimum setting of this parameter is 0x1.

Note

- Programming <WrHigh> to 0 is only used for zero wait states burst access (e.g., sync burst SRAM access). It is only allowed when WrLow is set to 1.
- If setting WrHigh to 0, A[2:0] and write data toggle every cycle.
- If setting WrHigh to 1, A[2:0] and write data toggle the same cycle Wr toggles from High to Low.

Figure 42 shows a Flash write timing parameters example.

Figure 42: 8-bit Flash Write Parameters Example



14.7 Data Pack/Unpack and Burst Support

The Device interface is an 8-/16-bit wide interface and supports up to a 128-byte burst. The Device controller contains a single 128-byte buffer. When a 8b or 16b device is used, Multiple 8-byte 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 (not like the latched address on the multiplexed AD bus). The Device controller supports packing/unpacking of data between the device and the initiator (for example, the core).

As previously described, the Device controller needs to pack read data from the 8-/16-bit wide Device to the 88F5182 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.

Note

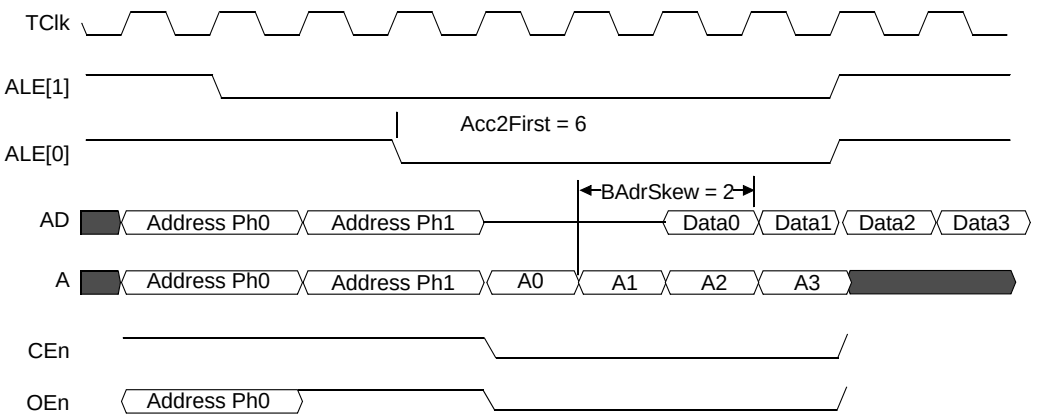
The Device controller does not support non-sequential byte enables to 8- or 16-bit wide devices (e.g., a write of a 32-bit word to an 8-bit wide device with byte enable 1'b1010).

14.7.1 BAdrSkew

The 88F5182 device also supports early toggle of burst address during read access. BAdrSkew parameter defines the number of TCLK cycles from A[2:0] toggle, to read data sample. This parameter is useful for SyncBurst SRAM type of devices, where the address precedes the read data by one (Flow Through SRAM) or two (Pipelined SRAM) cycles.

Figure 43 shows a BAdrSkew usage example.

Figure 43: Pipeline Sync Burst SRAM Read Example



14.8 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).

Ready can extend the following timing parameters—<Acc2First>, <Acc2Next> and <WrLow>. 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 44, Figure 45, and Figure 46 show examples of the READYn operation.

Note

- If the WrLow or WrHigh timing parameter is set to 0, READYn is not supported during a write access.
- When interfacing a device with a non-deterministic access time, the timing parameters are set to their minimum values and the actual access time is controlled via READYn pin.
- The Device controller samples read data two cycles after DEV_READYn assertion on a read access, and de-asserts DEV_WRn two cycles after DEV_READYn assertion on a write access.

Figure 44: READYn Extending Acc2First Example

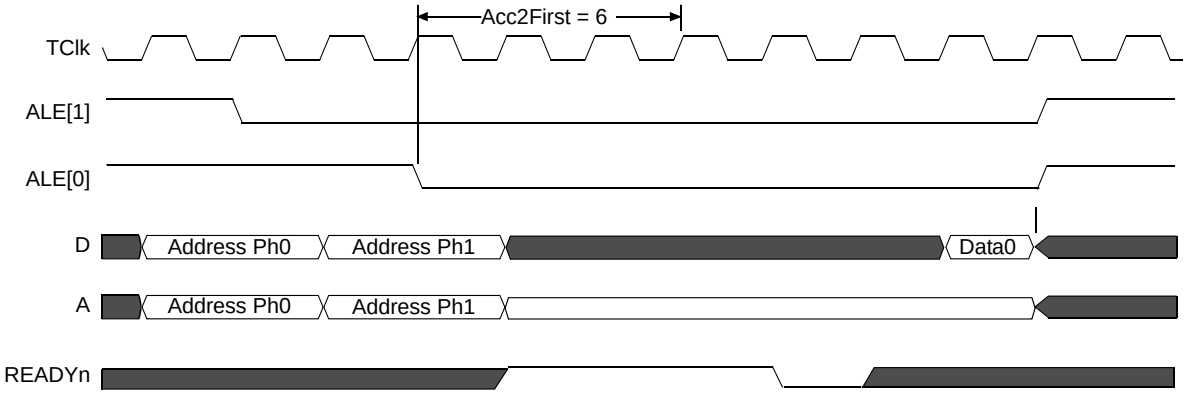


Figure 45: READYn Extending Acc2Next Example

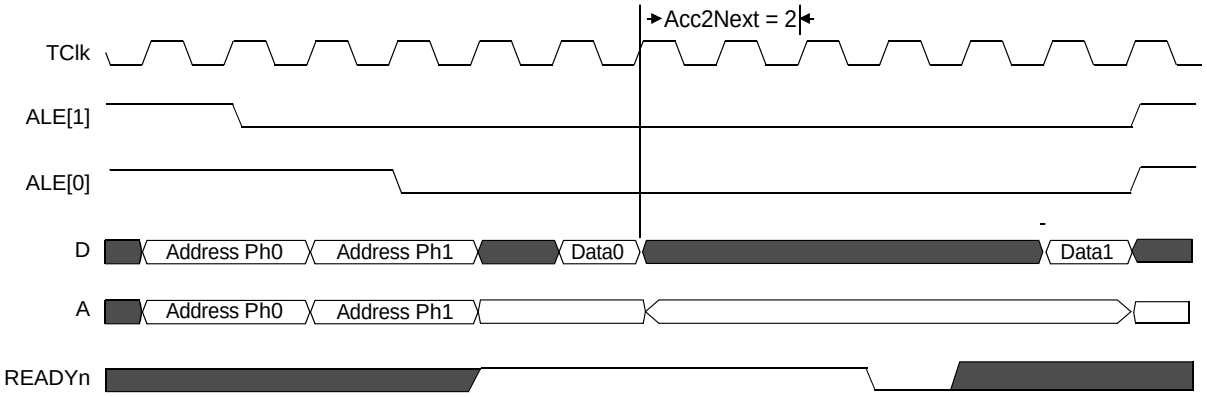
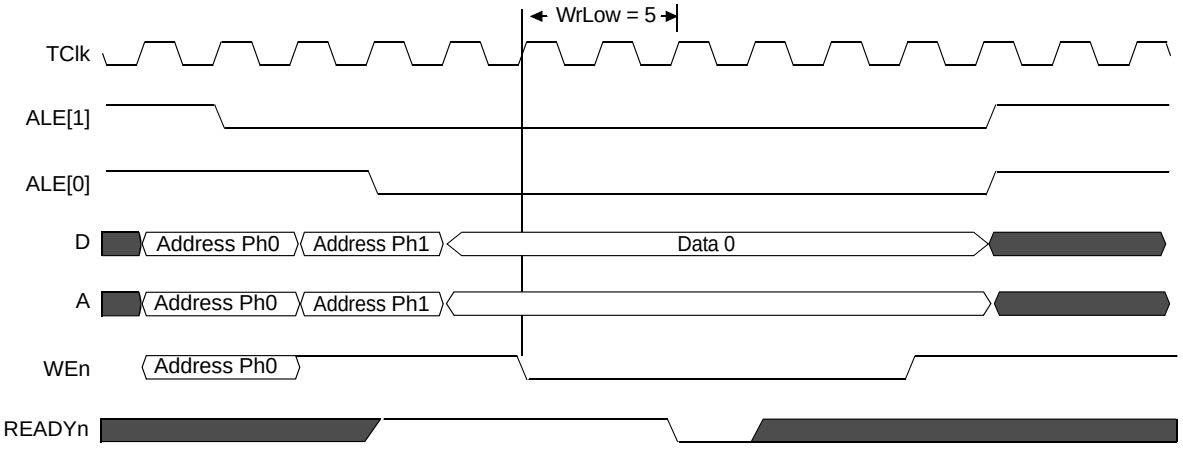


Figure 46: READYn Extending WrLow Example



To prevent system hang due to a lack of READYn assertion, the 88F5182 implements a programmable timer that allows 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

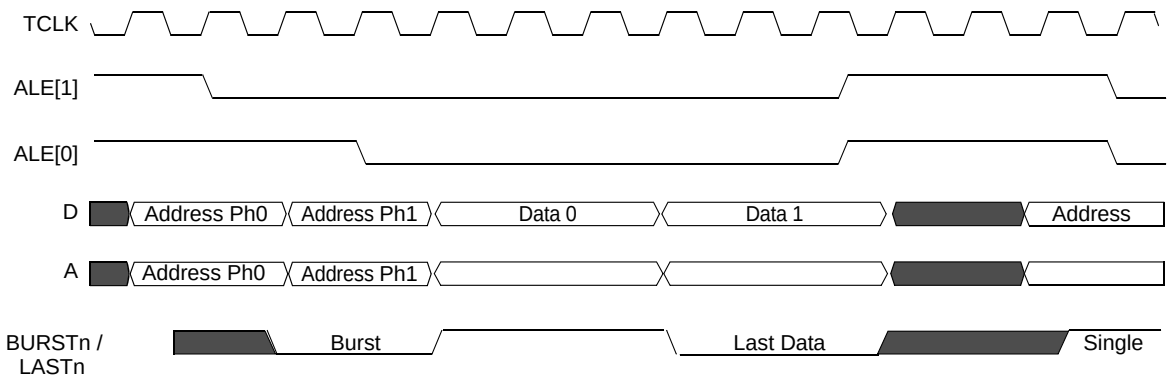
if READYn was asserted and generates an interrupt. This might result in bad data read/write from/to the device. Setting the timer to 0x0 disables the timer, and the device controller waits for READYn forever. The timer must be programmed to a number that must never be exceeded in normal operation.

14.9 Additional Device Interface Signaling

To make it easy to glue external logic on the device bus, the 88F5182 supports burst and last indication via MPP lines. BURSTn/LASTn is driven low on the address phase (It needs to be latched via ALE[0]) to indicate a burst access and is driven low on the last data phase to indicate the last data transfer.

Figure 47 shows an example of these additional signals.

Figure 47: BURSTn/DEV_LASTn Example



14.10 NAND Flash Support

The 88F5182 device bus supports both chip enable (CE) care NAND Flash and CE don't care NAND Flash.

The difference between CE care NAND Flash and CE don't care NAND Flash is that chip enable does not need to be continuously asserted low during the entire NAND Flash transaction. The removal of this restriction allows chip enable to be deasserted between individual read or write cycles, to enable sharing the read enable (RE) and write enable (WE) with other devices. CE care NAND Flash devices require dedicated RE and WE signals provided through the MPP[1:0] pins.

14.10.1 NAND Flash Features

The device bus provides interface to NAND Flash with following features:

- Glueless interface to CE don't care NAND Flash through the device bus interface
- Glueless interface to CE care NAND Flash through the device bus and MPP interfaces
- Boot from NAND Flash when the first block—placed on the 00h block address—is guaranteed to be a valid block with no errors
- Read bursts of up to 128 bytes, splits the transaction to multiple bursts—eight beats each burst.
  - 8 bytes in 8b device
  - 16 bytes in 16b device

14.10.2 Software Responsibilities

Software is responsible on the following:

- Following the appropriate guidelines, as specified in Section 14.10.3, Guidelines for Access to NAND Flash, on page 175
- Generating ECC
- Checking ECC
- Error recognition
- Error correction
- Error handling

14.10.3 Guidelines for Access to NAND Flash

Note

These guidelines apply to both CE care NAND Flash and CE don't care NAND Flash.

For both CE care and CE don't care NAND Flash devices, make sure bit <NF> is set to 1 in the appropriate bank of the NAND Flash Control Register (Table 547 p. 485).

For CE care NAND Flash device, make sure bit <NFActCEn> is also set to 1 in the appropriate bank of the NAND Flash Control Register. This bit may be cleared to 0 after the NAND Flash transaction is completed.

- Command phase: write to NAND Flash with A[1:0] = 01[1]
  - Transaction length must be 1 byte in 8-bit NAND Flash.
  - Transaction length must be 2 byte in 16-bit NAND Flash.
- Address phase: write to NAND Flash with A[1:0] = 10
  - Transaction length must be 1 byte in 8-bit NAND Flash.
  - Transaction length must be 2 byte in 16-bit NAND Flash.
- Write Data phase: Single write transaction with A[1:0] = 00. Bursts are not allowed.
- Read Data phase: Single or burst read transactions are allowed.
- Polling (reading) the NAND Flash device status via the NAND Flash device Status Register, which may be read to determine whether the program or erase operation is completed, and whether the operation has completed successfully. Alternatively the ready busy (R/B) signal can be connected to the 88F5182 MPP interface.

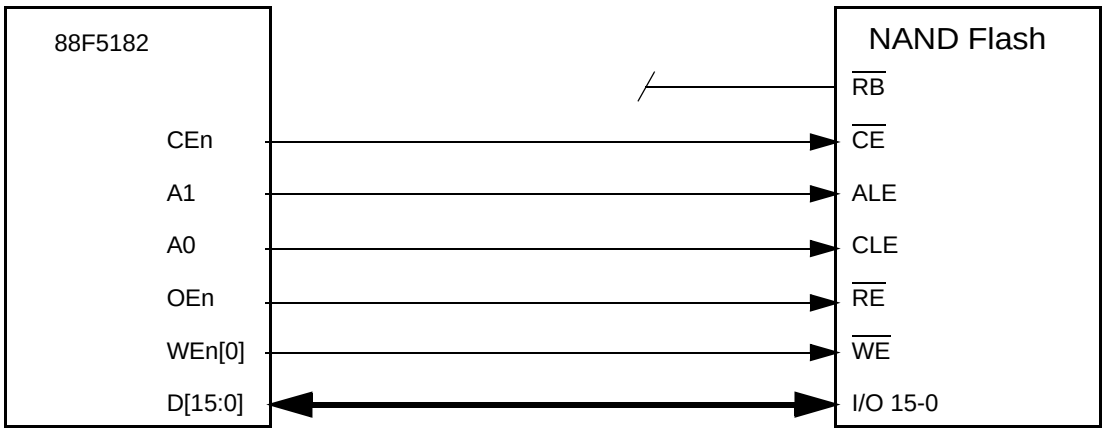
14.10.4 CE Don't Care NAND Flash

When the 88F5182 is connected to CE don't care NAND Flash devices, bit <NF> (NAND Flash) is set to 1 and bit <NFActCEn> (CE Care NAND Flash) is cleared to 0 in the appropriate bank of the NAND Flash Control Register.

The 88F5182 is connected to CE don't care NAND Flash devices using the 88F5182 device bus interface as shown in Figure 48.

1. The device address is effected by the device width. In the 16-bit device, the address refers to 16-bit words, meaning the address is shifted left in the device bus controller.

Figure 48: Chip Enable Don't Care NAND Flash

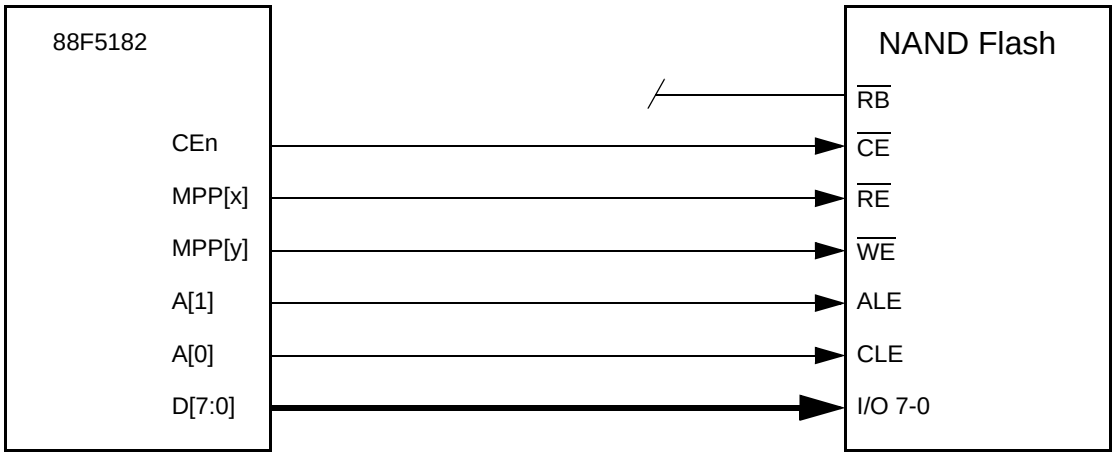


14.10.5 CE Care NAND Flash

The 88F5182 is connected to CE care NAND Flash via the device bus and the MPP interfaces as shown in Figure 49.

When the 88F5182 is connected to CE care NAND Flash devices, bit <NF> (NAND Flash) is set to 1 in the appropriate bank of the NAND Flash Control Register (Table 547 p. 485) and bit <NFActCEn> (in the same register) *must be set to* 1 before accessing the NAND Flash device. This bit may be cleared to 0 after the NAND Flash transaction is completed.

Figure 49: CE Care NAND Flash Using MPPs



Note For the CE Care NAND Flash Pin Multiplexing on MPP Interface, see the multiplexing section of the *88F5182 Datasheet*.

14.11 NAND Flash Controller Implementation

The NAND Flash controller implementation is shown in detailed in Figure 50 through Figure 52. It is controlled by NAND Flash Control Register (Table 547 p. 485). Burst transactions are supported by toggling the RE signal every read cycle as described in Figure 50.

Figure 50: Mask ALE during NAND Flash Read Data Phase

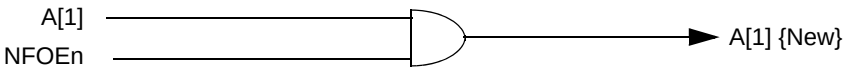


Figure 51: Generate Dedicated NAND Flash $\overline{WE}$ Signal

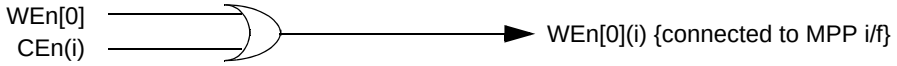
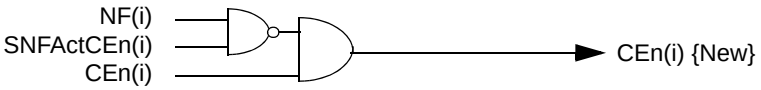


Figure 52: Generate CE Covers All NAND Flash Transaction



14.12 Boot from NAND Flash

The 88F5182 supports booting directly from NAND Flash, when the first block—placed on 00h block address—is guaranteed to be a valid block with no errors.

If the boot chip select is configured to NAND Flash—bit <NFBoot> is set to 1 in the NAND Flash Control Register (Table 547 p. 485)—and bit <NFISD> is cleared to 0 in the NAND Flash Control Register, an internal mechanism performs one of the sequences described in Section 14.12.1, Boot from 8b NAND Flash or in Section 14.12.2, Boot from 16b NAND Flash. These sequences are performed immediately after reset to issue the read command to the NAND Flash device. This mechanism allows the CPU to perform sequential reads to the NAND Flash, starting from address 0x0, right after reset.

14.12.1 Boot from 8b NAND Flash

The Boot device width is defined by bootstrap.

1. Set CPU internal reset.
2. Write 8b to the NAND Flash to Address = 0x1 with data =0x0. Set NAND FLASH command to Read.
3. Write 8b to the NAND Flash to Address = 0x2 with data =0x0. Set Address to 0x0.
4. Write 8b to the NAND Flash to Address = 0x2 with data =0x0. Set Address to 0x0.
5. Write 8b to the NAND Flash to Address = 0x2 with data =0x0. Set Address to 0x0.
6. Wait until NAND FLASH is ready with valid data, as define by the <NFTr> field in the NAND Flash Control Register.
7. Clear CPU internal reset.

14.12.2 Boot from 16b NAND Flash

The Boot device width is defined by bootstrap.

1. Set CPU internal reset.
2. Write 16b to the NAND Flash to Address = 0x2 with data =0x0. Set NAND FLASH command to Read.
3. Write 16b to the NAND Flash to Address = 0x4 with data =0x0. Set Address to 0x0.
4. Write 16b to the NAND Flash to Address = 0x4 with data =0x0. Set Address to 0x0.
5. Write 16b to the NAND Flash to Address = 0x4 with data =0x0. Set Address to 0x0.
6. Wait until NAND FLASH is ready with valid data as define by the <NFTr> field in the NAND Flash Control Register.
7. Clear CPU internal reset.

14.12.3 Boot from NAND Flash—Pins Sample Configuration

Sample at reset pins define if and how the 88F5182 boots from NAND Flash.

- Reset configuration defines the default value of bit <NFBoot> in the NAND Flash Control Register. This bit is internal pulled down to 0.
- Reset configuration defines the default value of bit <NFActCEnBoot> in the NAND Flash Control Register. This bit is internal pulled down to 0.
- When the reset configuration of both <NFBoot> and <NFActCEnBoot> are set to 1, fields <MPPSel4> and <MPPSel5> in MPP Control 0 Register (Table 601 p. 515) are set to 0x4. Otherwise, these fields are cleared to 0x0.

15 IDMA Controller

The 88F5182 has four independent IDMA engines. The IDMA engines optimize system performance by moving large amounts of data without significant CPU intervention.

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 that mode each buffer has its own descriptor.

15.1 Functional Description

IDMA unit contains four 512-byte buffers, one buffer per IDMA channel.

When a channel is activated, data is read from the source into the buffer, and then written to the destination. Read and write transactions are handled independently. The IDMA engine transfers the buffer in chunks of from 8 up to 128 bytes. The IDMA engine 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 IDMA interface.

The four channels share the same resources. They use fixed round-robin arbitration.

15.2 IDMA Descriptors

Each IDMA Channel Descriptor consists of four 32-bit registers. Each channel can be configured to work in 64-KB Descriptor mode, or in 16-MB Descriptor mode, as shown in Figure 53.

Figure 53: IDMA Descriptors

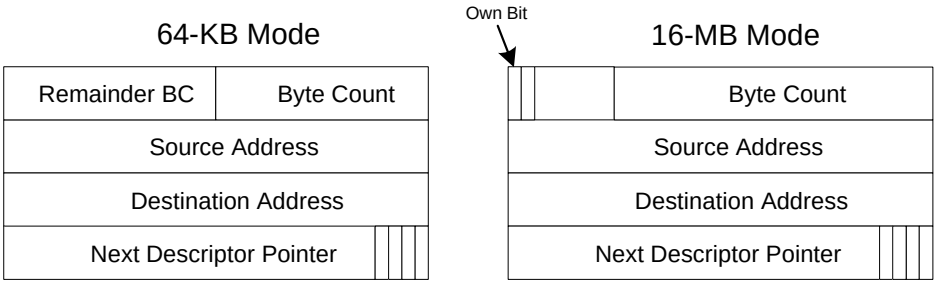


Table 55: IDMA Descriptor Definitions

IDMA Descriptor	Definition
Byte Count	Number of bytes of data to transfer. The maximum number of bytes to which the IDMA controller can be configured 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, the IDMA transaction is finished or terminated.
Own Bit	When running in 16M descriptor mode, this bit indicates whether the descriptor is owned by the CPU (0) or the IDMA engine (1).
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:** This descriptor is used only used when the channel is configured to Chained mode.

15.3 IDMA Address Decoding

The four IDMA 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 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, an interrupt is generated and the IDMA engine halts.

For the IDMA engine to avoid accessing a forbidden address space (due to a programming bug), each channel uses access protection logic that prevents it from having read/write access to specific address windows. In the case of an 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 Channel Control (Low) Register (Table 562 p. 492) register's <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, and upper 32-bit address are taken from The Base Address register (BAR) 1. When set to 0x2, these items are taken from BAR 2. And when set to 0x3, they are taken from BAR 3. See the Base Address Register x (Table 557 p. 490).

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.

15.4 IDMA Channel Control

Each IDMA Channel has its own unique control register where certain IDMA modes are programmed. The following are the bit descriptions for each field in the control registers.

15.4.1 Address Increment/Hold

The IDMA engine supports both increment and hold modes, on both source and destination addresses.

If the <SrcHold> bit [3] in the Channel Control (Low) Register (Table 562 p. 492) is set to 0, the IDMA automatically increments the source address with each transfer. If this bit is set to 1, the source address remains constant throughout the IDMA burst.

Similarly, If the <DestHold> bit [5] in the same register is set to 0, the IDMA automatically increments the destination address. If this bit is set to 1, the destination address remains constant throughout the IDMA burst.

Setting the <SrcHold> or <DestHold> bits 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 Channel Control (Low) Register <SrcBurstLimit>bits [8:6] and <DstBurstLimit>bits [2:0].

15.4.2 Burst Limit

The IDMA byte count is chopped into small bursts.

The burst limit can vary from 8 to 128 bytes in modulo-2 steps (i.e. 8, 16..., 128). It determines the burst length of IDMA transaction against the source and destination. There are separate Burst Limit 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.

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.

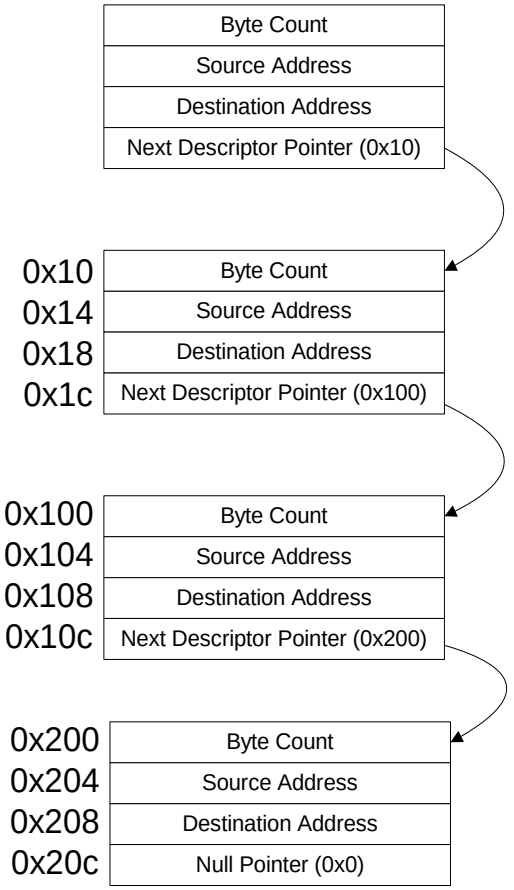
15.4.3 Chain Mode

When <ChainMode> bit [9] in the Channel Control (Low) Register (Table 562 p. 492) is set 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 IDMA engine fetches the new descriptor, and starts transferring the new buffer.

Figure 54 shows an example of an IDMA descriptors chain.

Figure 54: Chained Mode IDMA



Fetch next descriptor can be forced by <FetchND> bit [13] in the Channel Control (Low) Register.

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 transaction 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 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 IDMA transaction is in progress, the <Abr> bit [20] in the Channel Control (Low) Register must be set.

The first descriptor of a chain can be set directly by programming 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 <ChanEn> bit [12] in the Channel Control (Low) Register to 1 (see Section 15.4.4, Channel Activation, on page 183) and setting <FetchND> to 1.

When the IDMA transfer is done, an IDMA completion interrupt is set. When running in chain mode, <IntMode>, bit [10] of the in the Channel Control (Low) Register, controls whether to assert an interrupt on the completion of every byte count transfer or only upon last descriptor byte count completion. If set to 0, <Comp> bit [0] in the Interrupt Mask Register (Table 565 p. 495) is set every time the IDMA byte count reaches 0. If set to 1, the <Comp> bit is asserted when both the Pointer to Next Descriptor Register has a NULL value and byte count is 0.

If <ChainMode> in the Channel Control (Low) Register 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.

Note

- In non-chained mode, the Byte Count, Source, and Destination registers must be initialized prior to enabling the channel.
- If upon reading a new descriptor, the read data is marked by the target unit as erroneous, and the channel halts.
- If using <FetchND> to fetch the first descriptor and the <IntMode> bit is set to 0, a dummy IDMA completion interrupt is asserted with this first fetch. Although, no data has been transferred.

15.4.4 Channel Activation

Software channel activation is done via the Channel Control (Low) register's <ChanEn> bit [12].

When set 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> bit and then the active channel can be continued from the point it was stopped by setting <ChanEn> bit back to 1.

Clearing the <ChanEn> bit during 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> bit [14].

To restart a suspended channel in non-chained mode, the <ChanEn> bit 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> bit is set to 1. If the fetch did not take place, the <FetchND> bit must also be set to 1.

The <ChanAct> bit is read only. If set 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 the pointer to the next descriptor is NULL and byte count reaches zero.

If the <ChanEn> bit is set to 0 during IDMA transfer, the <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> bit to 1. Setting this bit has a similar effect to clearing <ChanEn> bit. 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 is cleared, allowing the software to re-program the channel.

Note

- When a channel is stopped, it completes the last read from the source, and then writes all of the remaining data in the FIFO to the destination.
- If the byte count is smaller than 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 bit after first clearing the <ChanEn> bit and then clearing the <ChanAct> bit.

15.4.5 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. It first reads 4 bytes from address 0x4 and then reads 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 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.

15.4.6 Descriptor Ownership

A typical application of chain mode IDMA involves the CPU preparing a chain of descriptors in memory and then preparing buffers to move from source to destination. Buffers may be dynamically prepared, i.e., once a buffer was transferred the CPU can prepare a new buffer in the same location to be sent. This application requires some handshake between the IDMA engine and the CPU.

When working with the 16 MB descriptor mode, Channel IDMA Byte Count Register (Table 552 p. 489) <Own> bit [31] acts as an ownership bit. If set to 1, the descriptor is owned by the 88F5182 IDMA. If set 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 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 set to 0, indicating to the CPU that the buffer was transferred. When the CPU 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 CPU (i.e., the CPU did not yet prepare a new buffer), 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 the 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 data 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 CPU 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 CPU must not write to it. When the NPTR (Next pointer) is first programmed, 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.

15.5 IDMA Interrupts

The IDMA interrupts are registered in the Interrupt Cause Register (Table 564 p. 495) 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 control whether an interrupt event causes an interrupt assertion. The setting of the mask register only affects the interrupt assertion, it has no effect on the cause register bits setting.

The following interrupt events are supported per each channel:

- IDMA completion
- IDMA address out of range
- IDMA access protect violation
- IDMA write protect violation
- IDMA descriptor ownership violation

In the 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 is read.

16 XOR Engine

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 off-loading in various RAID implementations. In addition, the XOR engine provides iSCSI CRC32C calculation, DMA operation, memory initialization, and memory ECC errors cleanup operation support.

The XOR engine enables PC/Server manufactures (ROM), Internal RAID Controllers and External RAID systems to speed up overall system performance.

XOR engine features:

- Two separate channels for enabling concurrent operation (e.g., concurrent XOR and iSCSI CRC32C calculations).
- 1KB temporary result store queue per channel. Arranged as 128 X 8B buffer.
- Support packing/unpacking of 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.

16.1 Theory of Operation

XOR engine unit (XEunit) has five 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 XOR engine has two independent channels. Each channel can be configured to one of the operation modes at a time. The operation mode is defined through the <OperationMode> field in the XOR Engine [0..1] Configuration (XExCR) (Table 570 p. 500)(bits[2:0]). In the XOR, CRC and DMA operation modes, the XOR engine is controlled by chain descriptors and responds to similar activation scheme. These modes differ only in the interpretation of the chain descriptor fields. In ECC and 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 two XOR engine channels are in an inactive state and can be configured to any operation mode (XOR, CRC, DMA, MemInit or ECC). 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.

Note

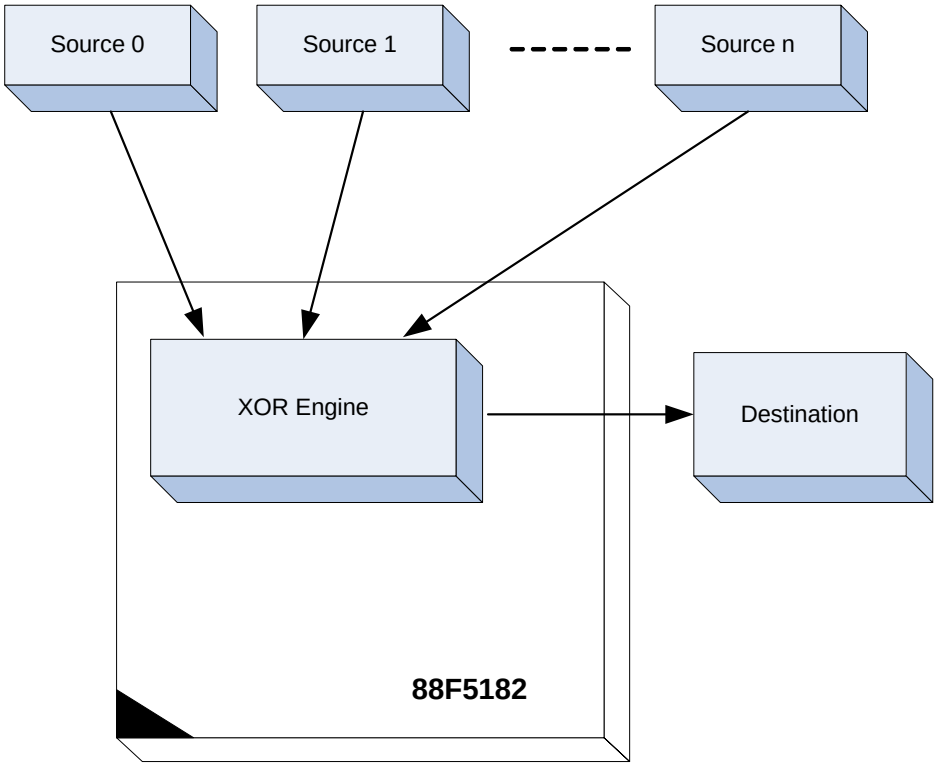
- The two XOR engine channels are independent in their operation modes. The only exception is that both engines must not be configured to ECC or MemInit operation modes. These modes share hardware resources.
- Attempting to change the channels operation mode during a pause will result in unexpected behavior.

16.1.1 XOR 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 55 shows how the XOR operation works with multiple blocks of source (incoming) data and stores the result back in a destination block.

Figure 55: XOR Operation with Multiple Incoming Data Blocks



Destination = (source 0) xor (source 1) ... xor (source n)
Number of sources can be up to 8 (n = 7)

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.

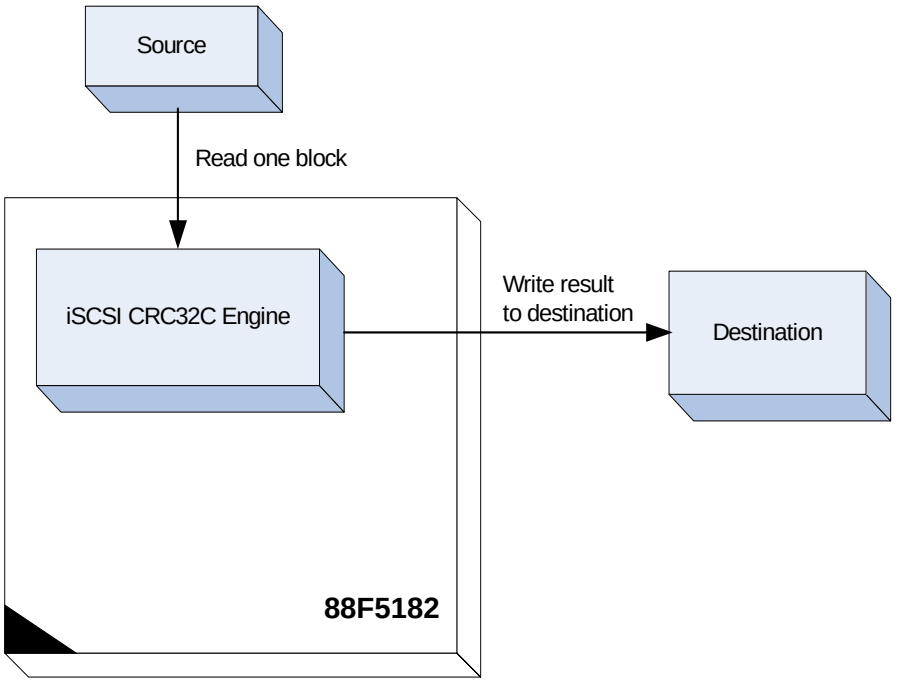
The Basic XOR operation algorithm is as follows:

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 steps 1–4 until the descriptor byte count in is reached.

16.1.2 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 56.

Figure 56: 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 (<OperationMode> field of the channel Configuration Register - XE0CR or XE1CR - 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 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.

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.

16.1.3 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.

16.1.4 Memory Initialization

The XOR engine provides memory initialization capabilities. It performs writes of pre-defined value 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 unexpected.

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) (Table 589 p. 511) and XOR Engine Initial Value High (XEIVRH) (Table 590 p. 511) Registers, in a cyclical method to the destination block. Upon completion of the memory initialization operation, the XOR engine channel asserts the EOC interrupt.

16.1.5 Memory ECC Errors Cleanup (Scrubbing)

The XOR engine provides a 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 memory controller.

The memory ECC errors cleanup operation is based on the 88F5182 DRAM 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 two bit 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 size of destination block.

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, which can interfere with the normal system operation. Moreover, ECC errors cleanup can run in the background, without CPU involvement. Once the CPU configures the parameters for the ECC operation, the ECC errors 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) (Table 586 p. 510) (bits[12:8]).

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, the 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 to its register. This means that read results are in the counter's real time value. The timer is only enabled 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, and asserts EOD interrupt. It 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 <XEstop> field in the XOR Engine [0..1] Activation (XExACTR) (Table 571 p. 501)(bit[1]).

If activated in non-timer mode, the whole target block will be processed at one time and the XOR Unit becomes inactive.

Note If the destination block is not 8 byte aligned, the XOR engine initiates write requests to the smallest 8 byte aligned block that contain the original block

16.2 Descriptor Chain

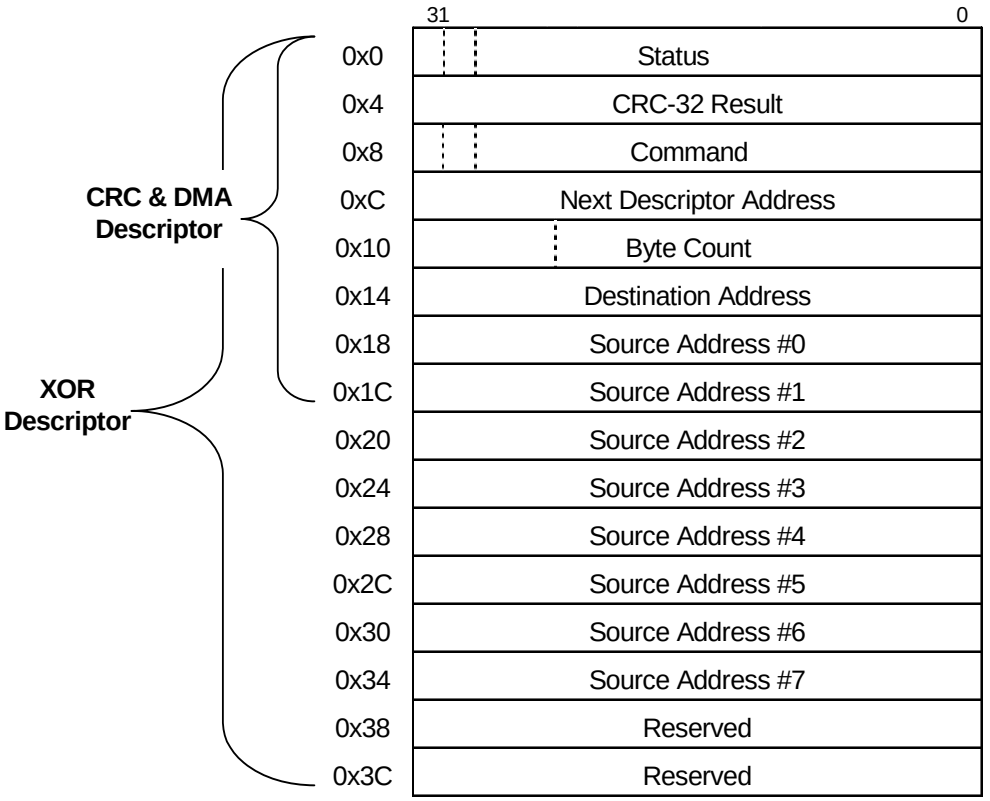
16.2.1 Descriptor Format

The XOR engine descriptor format supports 32-bit addressing. In XOR mode, the descriptor consists of sixteen 32-bit words totalling 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 57.

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 Chain descriptor operation is valid only in XOR, CRC and DMA operation modes. In ECC and MemInit modes, the XOR engine gets the operation data directly from its internal registers.

Figure 57: XOR Descriptor Format



Note

The XOR descriptor must be 64 Bytes 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.

Table 56: 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 57: 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 58: 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). 0x0 = Null Command - Data from Source will be disregarded in the current descriptor operation. 0x1 = XOR Command - Data from source will be transferred and will be significant in the XOR calculation. **NOTE:** Relevant only on XOR operation mode. disregarded in all other operation modes.
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). **NOTE:** Relevant only on XOR operation mode. Disregard in all other operation modes.
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). **NOTE:** Relevant only on XOR operation mode. Disregard in all other operation modes.
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). **NOTE:** Relevant only on XOR operation mode. Disregard in all other operation modes.
4	Src4Cmd	Specifies the type of operation to be carried out on the data pointed by SA#4 (Source Address #4 word of the descriptor). **NOTE:** Relevant only on XOR operation mode. Disregard in all other operation modes.

Table 58: Descriptor Command Word Definition (Continued)

Bit	Field	Description
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). **NOTE:** Relevant only on XOR operation mode. Disregard in all other operation modes.
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). **NOTE:** Relevant only on XOR operation mode. disregarded in all other operation modes.
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). **NOTE:** Relevant only on XOR operation mode. Disregard in all other operation modes.
29:8	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 59: 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 60: 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 61: 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 62: 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.

16.3 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.

16.3.1 Target Interface

Source data blocks, destination data block, and descriptors can be targeted to any of the device's Interfaces. The unique attributes of each interface are configured per address window through the XOR engine BARs (Base Address Registers), see Appendix A.16.4, XOR Engine Address Decoding Registers, on page 505.

16.3.2 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 DMA, if it hits the relevant address window, concatenated with the High Remap register.

Note

The XOR DMA address decoder can map total of up to 4 GB address space.

16.3.3 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 [0..1] Address Override Control (XExAOCR) (Table 583 p. 507). When override is enabled and the respective pointer field is set 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.

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.

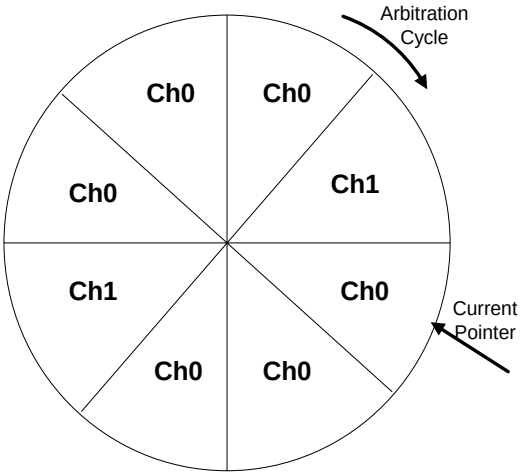
16.4 Arbitration

The two XOR engine channels and the four IDMA channels share the same crossbar port. The arbitration is performed in two steps. First, an arbitration between the two XOR engine channels and in parallel arbitration between the four IDMA channels. Second, an arbitration between the chosen XOR engine channel and the chosen IDMA channel is performed.

16.4.1 Arbitration Between XOR Engine Channels

The two XOR engine channel use the same crossbar port. A programmable weighted round robin arbiter controls the bandwidth allocation for each channel on the crossbar port. Each channel can be configured to have a different bandwidth allocation. Figure 58 shows an example of the arbitration cycle.

Figure 58: Programmable Channel Pizza Arbiter



The pizza arbiter has 8 slices, each slice can be configured to serve a different channel. In Figure 58, 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 two ports demand crossbar port service at the same time (congestion conditions). If only one channel

demands crossbar bandwidth, the channel receives 100% of the bandwidth. For example, in Figure 58, only Channel0 is active and so it gets 100% of the crossbar port bandwidth.

16.4.2 Arbitration between XOR Engine and IDMA

A fixed round robin arbitration is performed between the XOR engine and the IDMA. If both are active, the effective bandwidth allocation in congestion conditions is approximately 50% for each unit.

16.5 XOR Engine Programming

16.5.1 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.

16.5.1.1 Activation On Startup

To activate an XOR engine for the first time after Reset de-assertion, the software must perform the following sequence:

1. Confirm that the relevant XOR engine channel is inactive (<XEstatus> field in the XOR Engine [0..1] Activation (XExACTR) (Table 571 p. 502).
2. Initialize the relevant XOR engine channel configuration (XOR Engine [0..1] Configuration (XExCR) (Table 570 p. 500)).
3. Prepare the descriptor (or chain of descriptors) in memory.
4. Update the relevant XOR Engine [0..1] Next Descriptor Pointer (XExNDPR) (Table 576 p. 504).
5. Set <XEstatus> in the relevant XOR Engine [0..1] Activation (XExACTR) register.
6. When the XOR engine activates the relevant channel (<XEstatus> field in XE0ACTR or XE1ACTR is set).

When activated (<XEstatus> is set), the XOR engine fetches the descriptor pointed by the XOR Engine [0..1] Next Descriptor Pointer (XExNDPR), 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.

16.5.1.2 Update Descriptor Chain

A new descriptor can be added to the chain even 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

Pay attention that the ownership mechanism is violated in that case - CPU will write to a XOR engine owned descriptor (the former last one), but that does not affect the XOR engine operation.

16.5.1.3 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 crossbar, 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. 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 (<XEstatus> bit in the Activation Register - XE0ACTR or XE1ACTR- is set). If it is not active, the pause operation is not necessary.
2. Set <XEpause> in the relevant XOR Engine [0..1] Activation (XExACTR) register (see Table 571 on page 501).
3. Check the relevant <XEstatus> field in the XOR Engine [0..1] Activation (XExACTR) (Table 571 p. 502). 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.

Note Receiving 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 and that the channel is in stop mode and not in paused mode. The software must act accordingly and reactivate the channel according to Section , Re-Activation After Stop.

Re-Activation After Pause

To re-activate the channel, the software must set the <XErestart> bit in the relevant XOR Engine [0..1] Activation (XExACTR) register (XE0ACTR or XE1ACTR). When <XEstatus> field is set, the XOR engine has resumed operation.

Note After pausing a channel, it is not allowed stopping it. In order to stop it, software must first perform re-activation after pause, and only after the channel becomes active again, it can now stop it.

16.5.1.4 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 [0..1] Activation (XExACTR) (Table 571 p. 502)—XE0ACTR or XE1ACTR—is set). If it is not active, the stop operation is not necessary.
2. Set <XEstop> in the relevant XOR Engine [0..1] Activation (XExACTR) register.
3. Check relevant <XEstatus> bit. 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> bit 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

Similar to activation on startup, see Section 16.5.1.1, Activation On Startup, on page 197.

The software must perform the following steps:

1. Confirm that the relevant XOR engine channel is inactive. The <XEstatus> field in the XOR Engine [0..1] Activation (XExACTR) (Table 571 p. 502) is set to 0.
2. Initialize the relevant XOR engine channel through the XOR Engine [0..1] Configuration (XExCR) (Table 570 p. 500).
3. Prepare a descriptor (or chain of descriptors) in memory.
4. Update the XOR Engine [0..1] Next Descriptor Pointer (XExNDPR) (Table 576 p. 504).
5. Set the <XEStart> field in the XOR Engine [0..1] Activation (XExACTR) (Table 571 p. 501).
6. When the XOR engine activates the relevant channel (the <XEstatus> field in XE0ACTR or XE1ACTR is set), the activation sequence is complete.

When activated (<XEstatus> is set), the XOR engine fetches the descriptor pointed by the XOR Engine [0..1] Next Descriptor Pointer (XExNDPR), 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.

16.5.1.5 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>).

Upon receiving an EOC interrupt, two options must be examined by the software:

True EOC The XOR engine reaches the end of a descriptor chain.

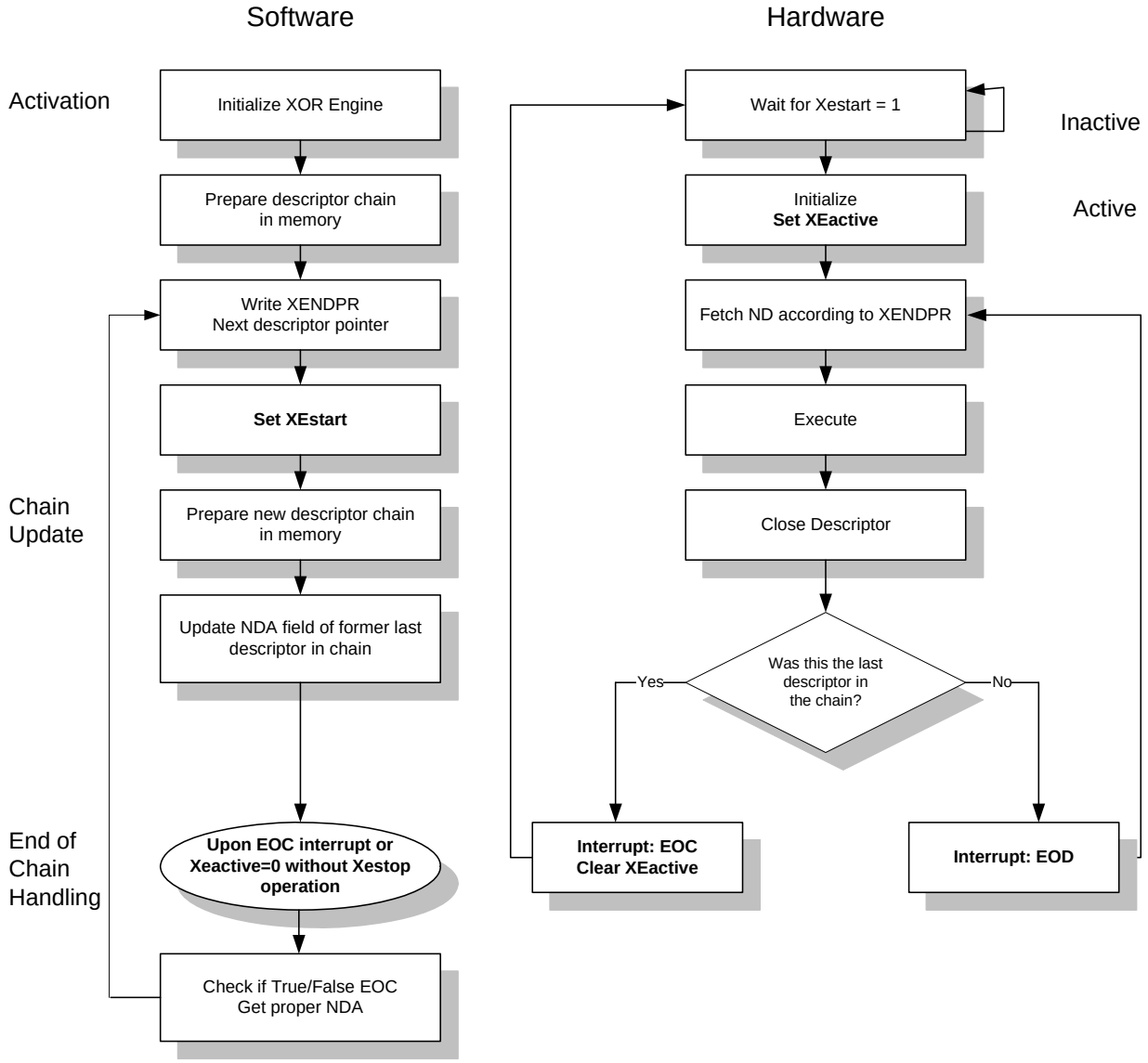
False EOC The chain was updated and the XOR engine is not in the current EOC. This can occur when software updates the descriptor chain while the XOR engine is processing the former last descriptor in the chain. In this case, the XOR Engine [0..1] Next Descriptor Pointer (XExNDPR) has a NULL value. Although 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 [0..1] Current Descriptor Pointer (XExCDPR) (Table 577 p. 505) 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> field in the XOR Engine [0..1] Activation (XExACTR) (Table 571 p. 501).

16.5.1.6 Synchronizing Software and Hardware



16.5.2 Programming in ECC and MemInit Modes

ECC and MemInit modes are programmed and controlled directly through internal registers (without using descriptor chains).

16.5.2.1 Activation

To activate the XOR engine, the software must perform the following sequence.

1. Confirm that XOR engine relevant channel is inactive. The <XEstatus>bit is set to 0 in the XOR Engine [0..1] Activation (XExACTR) (Table 571 p. 501).
2. Initialize the relevant XOR engine channel configuration through the XOR Engine [0..1] Configuration (XExCR) register.

3. Program the relevant internal registers (XOR engine ECC/MemInit Registers).
4. Set <XEStart> in the relevant XOR Engine [0..1] Activation (XExACTR) register.

16.5.2.2 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> bit in the XOR Engine [0..1] Activation (XExACTR) must be set. If it is not active, the stop operation is not necessary.
2. Set the <XEstop> bit in the relevant activation register.
3. Check the relevant <XEstatus> bit. 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> bit in the activation register and asserting the stopped interrupt. The inactive state is similar to initial state of the XOR engine on startup.

16.5.3 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 configuration register's <RegAccProtect> field (XECR0 or XECR1).

Note If at least one of the channels enables register write access protection, write accesses to internal registers shared by channels (e.g, 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).

16.6 Burst Limit

The maximum burst sizes of different transaction types on the internal crossbar can be configured through the XOR Engine [0..1] Configuration (XExCR) (Table 570 p. 500).

- 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).

Note If an XOR engine writes to a cache coherent DRAM region or accesses (read/write) cache coherent SRAM, the burst limit must not exceed 32 bytes.

16.7 Endianness

The XOR engine supports byte swapping on an 8 byte granularity – byte 0 swapped with byte 7, byte 1 swapped with byte 6, and so on. This is useful for Big/Little Endian conversions.

To byte swap data being read from the source block, set the <DrdResSwp> field in the XOR Engine [0..1] Configuration (XExCR) (Table 570 p. 500). Byte swap of data being written to destination block can be set via <DrdResSwp> bit the same register.

The 88F5182 internal registers are kept in the Little Endian convention. Also, the descriptors (that are being loaded from memory into internal registers) are treated in Little Endian convention. During descriptor fetch and descriptor close, the XOR engine must be configured to perform byte swap on descriptors via the <DesSwp> bit in the Configuration Register (XE0CR or XE1CR).

16.8 Errors and Interrupts

The XOR engine interrupts are registered in the XOR Engine Interrupt Cause (XEICR) (Table 572 p. 502). 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) (Table 573 p. 503) controls whether an interrupt event causes an interrupt assertion. The setting of the mask register only affects the interrupt assertion. This setting has no affect on the cause register bits setting.

The XOR engine interrupts can be divided to two 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.

A summary of each interrupts group is registered in the Main Interrupt Cause Register (Table 99 p. 253).

Table 63 summarizes the interpretation of EOD and EOC interrupts for each operation mode.

Table 63: 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, 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 that matches more than one address window.
- Access Protect Error: Accessing an access-protected address.
- 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) (Table 575 p. 504). Once an address is latched, no new addresses (due to additional errors) can be latched until the current address being read.

17 General Purpose I/O Port Interface

The 88F5182 contains a 26-bit General Purpose Port I/O (GPIO). The GPIO interface provides the following features:

- Each of the GPIO pins can be assigned to act as a general purpose input or output pin.
- A dedicated register provides the GPIO input value.
- Each of the GPIO input pins can be programmed to generate an Edge sensitive or a Level sensitive maskable interrupt.
- A dedicated register provides the GPIO output value.
- Each of the GPIO outputs can be programmed for the LED to blink every ~100 ms.

Note The GPIO interface is multiplexed on the external pins as described in Section A.18.1, MPP Registers, on page 515.

For the 88F5182 GPIO registers, see Table 591, “GPIO Registers Map,” on page 512.

18 Interrupt Controller

18.1 Functional Description

The 88F5182 includes an interrupt controller that routes internal interrupt requests as well as external interrupt requests (GPIOs) to the Feroceon® CPU core.

The 88F5182 interrupt controller drives two interrupt signals to the Feroceon CPU core—FIQ (high priority) and IRQ (regular priority). All interrupts are level sensitive. The interrupt is kept active as long as there is at least one non-masked cause bit set in the Interrupt Cause register.

The 88F5182 can also be used as the interrupt controller for external devices generating interrupts to the Feroceon CPU core via GPIO inputs. The interrupt controller can also receive interrupt messages from an external PCI Express device.

The 88F5182 can also act as a PCI or PCI Express Endpoint. As such, it can generate the PCI Express INTA emulation message or the INTAn signal.

18.2 Local Interrupt Cause and Mask Registers

The 88F5182 handles interrupts in two stages. The first stage is specific unit cause and mask registers, that distinguish between a specific interrupt events within the unit.

Once an interrupt event occurs, its corresponding bit in the unit cause register is set to 1. If the interrupt is not masked by the unit mask register, it is marked in the Main Interrupt Cause register. The unit local mask register has no effect on the setting of interrupt bits in the unit local cause register. It only affects the setting of the interrupt bit in the Main Interrupt Cause register

When working in level mode, the GPIO Data In register must be used. Do not use the GPIO Interrupt Cause register.

The different units cause registers are:

- AHB to Mbus Bridge Interrupt Cause register
- PCI Express Interrupt Cause registers
- PCI Interrupt Cause register
- SATAHC Main Interrupt Cause register
- GbE Port Interrupt Cause register
- USB0/1 Interrupt Cause register
- Cryptographic Engine Security Accelerator Cause register
- TWSI Interrupt Cause register
- UART0/1 Interrupt Cause registers
- Device Interrupt Cause register
- GPIO Interrupt Cause register
- IDMA Interrupt Cause register
- XOR Engine Interrupt Cause register

18.3 Main Interrupt Cause and Mask Registers

The second stage includes the Main Interrupt Cause register and Main Interrupt mask registers that summarize the interrupts generated by each unit. The interrupt handler first reads the main cause register, and then reads the specific unit cause register.

Note

The Main Interrupt Cause register bits are Read Only. To clear an interrupt cause, the software needs to clear the active bit(s) in the specific unit cause register.

There are two mask registers corresponding to the two CPU interrupt lines—IRQ and FIQ. Setting these registers allows the reporting of different interrupt events on the different interrupt lines. If a bit in the mask register is set to 1, the corresponding interrupt event is enabled. The setting of the mask bits has no effect on the value registered in the Main Interrupt Cause register, it only affects the assertion of the interrupt pin. An interrupt is asserted if at least one of the non masked bits in the cause register is set to 1.

When the 88F5182 functions as Endpoint, a third mask register corresponding to PCI Express/PCI interrupt is used to generate an interrupt towards the host. The host is connected to 88F5182 through the interface defined by bit <EndPointIF> in the CPU Configuration Register (Table 93 p. 250). The INTA interrupt or MSI is routed to host according to this bit value.

Note

- See Table 71, CPU Register Map, on page 242.
- See Table 18.5, 88F5182 Interrupt Controller Scheme, on page 207.

18.4 Doorbell Interrupt

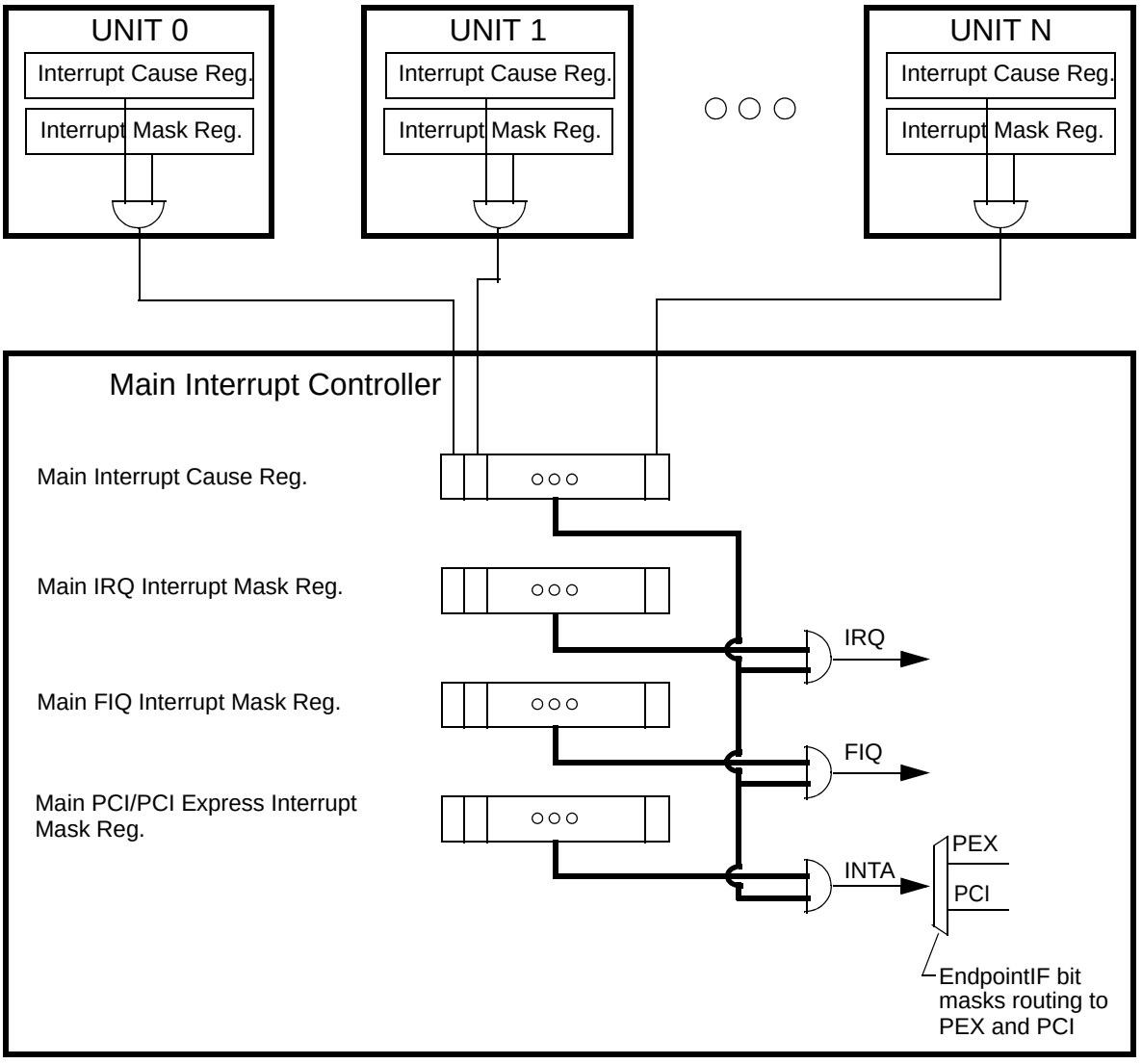
When 88F5182 functions as Endpoint, a doorbell mechanism is provided to communicate between Feroceon CPU core and the external host.

The 88F5182 supports 32-bit doorbell interrupt register from host to Feroceon CPU core. See Table 110, Host-to-CPU Doorbell Register, on page 258 and Table 111, Host-to-CPU Doorbell Mask Register, on page 258.

The 88F5182 supports 32-bit doorbell interrupt register from Feroceon CPU core to host. See Table 112, CPU-to-Host Doorbell Register, on page 259 and Table 113, CPU-to-Host Doorbell Mask Register, on page 259.

18.5 88F5182 Interrupt Controller Scheme

Figure 59: 88F5182 Interrupt Controller Scheme



19 Timers

19.1 Functional Description

The 88F5182 provides two general purpose timers and one watchdog timer.

19.2 32-bit-wide Timers

The 88F5182 provides two 32-bit-wide timers. Each timer decrements with every TCLK rising edge if the corresponding enabled bit is enabled. Reads and write from/to the timer are done to the counter itself.

The timers provide auto mode:

- When the timers are set to auto mode disabled and the timers reach to 0, the timers stop counting.
- When the timers are set to auto mode enabled and the timers reach to 0, the timers preload and continue counting.

Regardless of whether auto mode is enabled or disabled, when the timers reach 0, a maskable interrupt is generated.

19.3 Watchdog Timer

The 88F5182 internal watchdog timer is a 32-bit count down counter that can be used to generate a maskable interrupt or reset the system in the event of unpredictable software behavior.

After the watchdog is enabled, it is a free running counter that needs to be serviced periodically to prevent its expiration. After reset, the watchdog is enabled or disabled according to sample at reset pin value.

When the watchdog timer expires and bit <WDRstOutEn> is set to 1 in the RSTOUTn Mask Register (Table 95 p. 252), the SYSRST_OUTn output signal is set.

See Section A.4.4, CPU Timers Registers, on page 256.

20 Internal Architecture

20.1 AHB—Feroceon® CPU Core Local Bus

The 88F5182 Feroceon CPU core local bus is compatible with the AHB bus. See the AMBA Specification, Rev 2.0.

This local bus provides the following features to reduce cache read latency:

- Direct connection to the DDR controller
- Synchronous interface to the DDR controller (The AHB bus runs at DDR clock.)
- Extension to 64 bits for cache reads

20.2 Mbus—Internal Bus

The Mbus is a 64-bit internal full-mesh bus used for data transfer between the different units, except for the Feroceon CPU core access to DDR SDRAM. The different units can act as masters on the bus generating requests or as targets driving read response. Table 64 shows the units connected through the Mbus and the functions implemented in each unit. The Mbus runs at TCLK.

Table 64: Mbus Units

Unit	Unit ID	Function
DDR SDRAM controller	0x0	Target
Device bus, UART, and TWSI	0x1	Master/Target
AHB to Mbus bridge	0x2	Master/Target
PCI Express port (PEX0)	0x4	Master/Target
PCI port	0x3	Master/Target
USB 2.0 port0	0x5	Master/Target
IDMA port and XOR port	0x6	Master/Target
Gigabit Ethernet port	0x7	Master/Target
SATA ports	0x8	Master/Target
Cryptographic engines and Security accelerator	0x9	Target
USB 2.0 port1	0xA	Master/Target

Note

The Device bus, UART, and TWSI interfaces behave as targets. When using TWSI serial ROM initialization, the TWSI behaves as a master only after reset.

The Mbus uses a proprietary protocol. All read transactions are split transactions. This ensures that the bus is not tied up while a slow memory unit is accessing the requested data. The bus supports up to 128B transfer per a single transaction.

20.2.1 Mbus Arbitration

The DDR SDRAM interface Mbus port is using a programmable arbitration scheme to optimize the 88F5182 performance, according to the system requirements. The arbitration priorities for the initiators can be adjusted through the registers in Section A.5, DDR SDRAM Controller Registers, on page 260.

The DDR SDRAM controller further arbitrates between the winning Mbus transaction and AHB requests from the Feroceon CPU core. For more details, see Section 4.1.2, Arbitration and Ordering, on page 25.

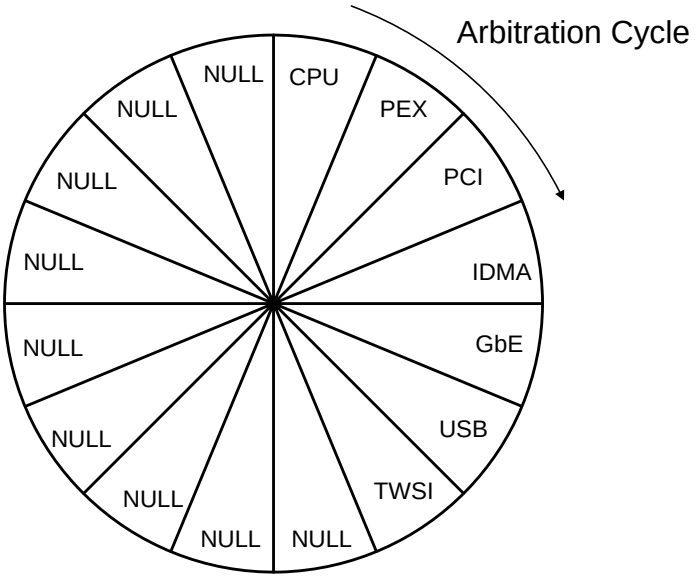
Note

Each of the rest of the Mbus target units also has a dedicated arbiter. Those arbiters used a fixed arbitration scheme that cannot be programmed. A two-level arbitration scheme is used for those arbiters. All read responses target units are serviced first, followed by all initiator requests from master units. The arbitration between the initiator units operates in a fixed round-robin fashion. The arbitration between target units also operates in a fixed round-robin fashion.

Figure 60 shows the arbitration in the form of a wheel. Each slice of a wheel represents a transaction on the Mbus. The arbitration scheme works as follows:

- The arbiter gives access to the initiator unit that has priority in the current transaction slice.
- If the initiator unit that has priority in the current slice has not reasserted a request, the arbiter will give the transaction to the initiator unit that has priority in the next slice. This will continue until an initiator unit is found that has asserted a request. This will occur in the same clock cycle.

Figure 60: Masters Request Default Arbitration Cycle



The Mbus SDRAM arbiter priority scheme can be used to allocate a fair bandwidth to the different master units. The arbiter has 16 slices, each of which can be assigned to any unit. The arbiter works in a round-robin fashion, calculating at each available cycle which of the pending requests is the next to be served. The arbiter works on a transaction basis, meaning, it enters a new arbitration cycle, only when a transaction ends. This means that priority settings are affected by the typical transaction size for each unit. If for example, the typical transaction size of unit A is twice the typical transaction size of unit B, unit B receives twice as many priority slices as unit A, so as to have the same bandwidth allocation.

20.3 AHB to Mbus Bridge

The Feroceon CPU core interfaces with the 88F5182 units over the AHB to Mbus bridge. This bridge forwards Feroceon CPU core transactions to the 88F5182 units over the Mbus, and forwards read responses back from the units to the Feroceon CPU core. This bridge also contains the 88F5182 control and status registers related to the CPU.

The AHB to Mbus Bridge incorporates the following features:

- Unidirectional bridge—Only transactions from AHB to Mbus are supported.
- A single read transaction at a time: The AHB bus is available for the next transaction only when data from the current transaction was returned on the AHB bus.
- A single write transaction at a time: The write transaction is completed when data is stored in the bridge internal buffer. The AHB bus is then available for next transaction.
- Responds to write transaction only when its buffer is empty.
- Contains a 32B single read/write buffer.

The AHB to Mbus bridge contains the registers listed in following sections:

- Section A.4.1, CPU Address Map Registers, on page 243.
- Section A.4.2, CPU Control and Status Registers, on page 250.
- Section A.4.3, Main Interrupt Controller Registers, on page 253.
- Section A.4.4, CPU Timers Registers, on page 256.
- Section A.4.5, CPU Doorbell Registers, on page 258.

20.4 Transaction Ordering

The 88F5182 architecture obeys the rules of PCI Express transactions ordering. In addition, transactions from the Feroceon CPU core to the DDR interface are snooped on the Mbus transaction queue within the dual port DDR controller unit.

Note

When an interrupt is received from the PCI Express port, the Feroceon CPU core must perform a read of the corresponding cause register, to ensure that all previous transactions have been completed (Producer-Consumer model).

21 System Considerations

21.1 Endianness

The 88F5182 supports both Little Endian and Big Endian modes.

The Endian mode is set using bit [7] B in Register R1 in the CPU-CP15 registers.

The value of B bit is reflected by bit[15] <BigEndian> field in the CPU Control and Status Register (Table 94 p. 251).

The initial value of B bit is set by bit <EndianInit> of CPU Configuration Register (Table 93 p. 250).

The initial value of bit <EndianInit> of CPU Configuration Register (Table 93 p. 250) is defined by boot strap.

Regardless of the endianness mode, the 88F5182 internal registers always operate in Little Endian mode.

21.1.1 Little Endian Mode (Default)

When the 88F5182 is in Little Endian mode, bit [7] B in Register R1 within CPU-CP15 registers is cleared to 0 (default). The 88F5182 transfers all transactions without any change.

21.1.2 Big Endian Mode

When the 88F5182 is in Big Endian mode, bit [7] B in Register R1 in the CPU-CP15 registers is set to 1. The Feroceon® CPU core implements Big Endian mode in compliance with the *ARM Architecture Reference Manual*, Second Edition. In addition, a 32b bytes swap is done on the Feroceon CPU core local bus.

The swapping is done in both directions:

- A value of 0xAABBCCDD–EEFFGGHH leaving the CPU is swapped to 0xDDCCBBAA_HHGGFFEE before arriving at the DDR controller and the MBUS bridge.
- A value of 0xAABBCCDD–EEFFGGHH leaving the DDR controller and the MBUS bridge is swapped to 0xDDCCBBAA_HHGGFFEE before arriving at the CPU.

No further swapping is done in the rest of the device.

21.2 Boot Sequence

The 88F5182 supports the following boot sequences:

- Boot from Flash (Default)
- Boot from PCI Express/PCI interface
- Boot from DDR—The external PCI Express/PCI master downloads the boot code into DDR memory.

21.2.1 Boot from Flash/NAND Flash

When the Feroceon CPU core is booted from the Flash, the following steps are performed:

- The location of exception vectors and the first PC generated from the Feroceon CPU core is to address
  0xFFFF–0000, as configured in bit <VecInitLoc> of the CPU Configuration Register (Table 93 p. 250).

- The Feroceon CPU core starts to boot from the Flash boot device as configured in the Window7 Control Register (Table 90 p. 249) and the Window7 Base Register (Table 91 p. 249). See also Section 2.10, Default Address Map, on page 22.
- At the end of the boot sequence, an MCR instruction is used to modify the location of exception vectors to 0x0000–0000 (Mapped to DDR). See Section 2.10, Default Address Map, on page 22.

21.2.2 Boot from PCI Express/PCI Interface

When the Feroceon CPU core is booted from the PCI Express/PCI interface, the following steps are performed:

- Serial ROM initialization is enabled and used to configure the following registers:
  - Bit <PCIDs> field in the CPU Control and Status Register (Table 94 p. 251) is cleared to 0, to enable the PCI Express/PCI interface.
  - Window7 Control Register (Table 90 p. 249) is disabled.
  - Window0 Base Register (Table 73 p. 244) is configured to match address F800.0000–FFFF.FFFF.
  - Window0 Base Register is configured to the appropriate PCI Express/PCI interface.
  - Window0 Remap Registers (Table 74 on page 244 and Table 75 on page 244) can be used to remap the transaction to the appropriate location in the PCI Express/PCI interface.

Note

The Feroceon CPU core remains at reset until the Serial initialization sequence is completed and until bit <CPUReset> field in the CPU Control and Status Register (Table 94 p. 251) is cleared to 0.

- The location of exception vectors and the first PC generated from the Feroceon CPU core is to address
  0xFFFF–0000, as configured in bit <VecInitLoc> of the CPU Configuration Register (Table 93 p. 250).
- The Feroceon CPU core starts to boot from the PCI Express/PCI interface as configured in the Window0 Control Register (Table 72 p. 243), the Window0 Base Register (Table 73 p. 244) and the Window0 Remap Registers (Table 74 on page 244 and Table 75 on page 244).
- At the end of the boot sequence, an MCR instruction is used to modify the location of exception vectors to 0x0000–0000 (Mapped to DDR). See Section 2.10, Default Address Map, on page 22.

21.2.3 Boot from DDR

When the Feroceon CPU core is booted from the DDR interface, the external PCI Express/PCI master downloads the boot code to the DDR memory.

The following steps are performed.

- Serial ROM initialization is enabled and used to configure the following registers/bits.
  - <PCIDs> field in the CPU Control and Status Register (Table 94 p. 251) is cleared to 0, to enable the PCI Express/PCI interface.
  - Bit <VecInitLoc> is cleared to 0 in the CPU Configuration Register.
  - <CPUReset> field in the CPU Control and Status Register (Table 94 p. 251) is set to 1.

Note

- Any transaction from the PCI Express/PCI is rejected until the serial initialization sequence is completed.
- The Feroceon CPU core remains at reset until the serial initialization sequence is completed and the <CPUReset> field in the CPU Control and Status Register (Table 94 p. 251) clears to 0.

- External device writes the code to DDR.
- External device clears the <CPUReset> field in the CPU Control and Status Register (Table 94 p. 251) to 0.
- The Feroceon CPU core starts booting from the DDR. The location of exception vectors and first PC generated from the Feroceon CPU core is to address 0x0000–0000, as configured in bit <VecInitLoc> of the CPU Configuration Register.
- The Feroceon CPU core transaction is propagated to the DDR interface, as configured in the Window2 Control Register (Table 80 p. 245) and the Window2 Base Register (Table 81 p. 246).

Caution

From each of the boot sequences listed above (Flash, PCI Express/PCI, DDR), the user must not attempt to issue a transaction to the PCI Express or PCI ports, before the <PCIDs> bit is cleared. Transactions to the PCI Express or PCI port internal registers are allowed.

21.3 Power Management

The 88F5182includes various power management (PM) features that enable fine-tuning of the device's power consumption according to the desired usage scheme.

21.3.1 DDR Controller Power Management Features

DDR SDRAM Self Refresh mode is supported (see Section 4.9, DDR SDRAM Self Refresh Mode, on page 34).

21.3.2 PCI/PCI-X Interface Power Management Features

PCI Power Management Capability is supported, including PM_PME support (see Section 6.11.1, Power Management, on page 63).

21.3.3 PCI Express Interface Power Management Features

The following PCI Express Root Complex and Endpoint power management features are supported.

Root Complex features

- SW Power management: D0, D1, D2, D3 device states and L0, L1 link states supported.
- ASPM L0s-Rx supported.
- Turn-off mechanism is not supported.

Endpoint features:

- SW Power management: D0, D1, D2, D3 device states and L0, L1 link states supported.
- ASPM L0s-Rx supported.
- Turn-off mechanism supported.

21.3.4 SATA Interface Power Management Features

The unused SATA ports are shut down to save power by clearing the appropriate <PhyShutdown> field in the Serial-ATA Interface Configuration Register (Table 379 p. 397).

21.4 Error Handling

The 88F5182 provides error handling for the following types of errors:

- Feroceon CPU core address decoding errors
- PCI Express errors
- PCI/ errors
- USB errors

Note As the DDR controller does not support ECC, no errors are generated by the DDR controller.

21.4.1 CPU Address Decoding Errors

Table 65 lists the CPU address decoding errors and describes how they are handled.

Table 65: CPU Address Decoding Error Handling

Error Type	Error Handling
Access to unmapped window	Accesses are completed according to the setting of bit <AHBErrorProp> in Table 93: CPU Configuration Register. 0 = Error indications are not propagate to AHB bus. The transactions are completed normally. 1 = Error indications are propagate to AHB bus.
Write Access to write-protected window	
Other errors	Unpredictable behavior.

21.4.2 PCI Express Errors

Table 66 lists the PCI Express errors and describes how they are handled.

Table 66: PCI Express Error Handling

Flow	Error Type	Error Handling
PCI Express Master Write	Error indication from initiator unit.	Forward the transactions with data poisoning indications to the PCI Express interface.
PCI Express Master Read completion	1. Data Poisoning from PCI Express interface. 2. Completion timeout. 3. Received completion with unsuccessful completion status.	Forward the transactions with error indications to the initiator.[1]
PCI Express Slave Write	Data Poisoning from PCI Express interface.	Drop the data. Close the transactions normally.
PCI Express Slave Read	Error indication from target unit.	Forward the transactions with data poisoning indications to the PCI Express interface.

Table 66: PCI Express Error Handling (Continued)

Flow	Error Type	Error Handling
Link Fail		1. Reset link state machine. 2. Generate maskable interrupt. 3. Activate system reset if enabled.
Hot reset received (Endpoint mode)		
PHY/Link/Transport Unrecoverable error		Set maskable interrupt.
PHY/Link/Transport Recoverable error		
Address Decoding errors		Unpredictable behavior.

1. The CPU is the initiator. The transaction is propagated on the AHB bus if bit <AHBErrorProp> in Table 93: CPU Configuration Register is set to 1.

21.4.3 PCI Errors

Table 67 lists the PCI/PCI-X errors and describes how they are handled.

Table 67: PCI Error Handling

Flow	Error Type	Error Handling
PCI Master Write	Error indication from initiator unit.	Forward the transactions with bad parity to the PCI interface.
PCI Slave Write	Bad parity detected on received data.	Drop the data. Close the transactions normally.
PCI Slave Read	Error indication from target unit.	Forward the transactions with bad parity to the PCI interface.

21.4.4 USB Errors

Table 68 lists the USB errors and describes how they are handled.

Table 68: USB Error Handling

Flow	Error Type	Error Handling
Address Decoding errors	No hit or multiple hit on address windows.	Transactions forwarded according to window0. Generate maskable interrupts.
USB Controller errors		Handled by the USB controller core according to USB 2.0 spec.

88F5182

Feroceon® Storage Networking SoC Register Set

This page is intentionally left blank.

List of Registers

A 88F5182 Register Set

This appendix provides full definitions for the 88F5182 registers.

A.1 Register Description

All registers are 32-bits wide [31:0]. The 88F5182 registers use the PCI Byte Ordering (Little Endian) in which the Most Significant Byte (MSB) of a multi-byte expression is located in the highest address. 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).

A.2 Register Types

The 88F5182 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 69: Standard Register Field Type Codes

Type	Description
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.
RES	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.
RWC	Read/Write Clear on Read. All bits are readable and writable. After reset or after the register is read, the register field is cleared to zero.
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.
WO	Write Only. A write to the register field will trigger an internal function and a read will return an undefined value.

A.3 Internal Registers Address Map

Table 70 lists the 88F5182 internal registers.These registers reside in a 1-MB address space divided into 64-KB segments.

Table 70: 88F5182 Internal Registers Address Map

Unit ID	Unit Name	Address Space Size	Address Range	Address Range in Hexadecimal
0	DDR registers	64 KB	0–64K	0x00000–0x0FFFF
1	Device Bus registers	64 KB	64K–128K	0x10000–0x1FFFF
2	AHB to Mbus Bridge registers	64 KB	128K–192K	0x20000–0x2FFFF
3	PCI register	64 KB	192K–256K	0x30000–0x3FFFF
4	PCI Express registers	64 KB	256K–320K	0x40000–0x4FFFF
5	USB registers Port 0	64 KB	320K-384K	0x50000–0x5FFFF
6	IDMA registers and XOR registers	64 KB	384K–448K	0x60000–0x6FFFF
7	Gigabit Ethernet registers	64 KB	448K–512K	0x70000–0x7FFFF
8	SATA registers	64 KB	512K–576K	0x80000–0x8FFFF
9	Cryptographic Engine and Security Accelerator registers	64 KB	576K–640K	0x90000–0x9FFFF
A	USB Registers Port 1	64 KB	640K-704K	0xA0000–0xAFFFF
B–F	Reserved	-	704K–1024K	

A.4 AHB to Mbus Bridge Registers

Table 71: CPU Register Map

Register Name	Offset	Table Number and Page Number
CPU Address Map Registers		
Window0 Control Register	0x20000	Table 72, p.243
Window0 Base Register	0x20004	Table 73, p.244
Window0 Remap Low Register	0x20008	Table 74, p.244
Window0 Remap High Register	0x2000C	Table 75, p.244
Window1 Control Register	0x20010	Table 76, p.244
Window1 Base Register	0x20014	Table 77, p.245
Window1 Remap Low Register	0x20018	Table 78, p.245
Window1 Remap High Register	0x2001C	Table 79, p.245
Window2 Control Register	0x20020	Table 80, p.245
Window2 Base Register	0x20024	Table 81, p.246
Window3 Control Register	0x20030	Table 82, p.246
Window3 Base Register	0x20034	Table 83, p.247
Window4 Control Register	0x20040	Table 84, p.247
Window4 Base Register	0x20044	Table 85, p.247
Window5 Control Register	0x20050	Table 86, p.248
Window5 Base Register	0x20054	Table 87, p.248
Window6 Control Register	0x20060	Table 88, p.248
Window6 Base Register	0x20064	Table 89, p.249
Window7 Control Register	0x20070	Table 90, p.249
Window7 Base Register	0x20074	Table 91, p.249
88F5182 Internal Registers Base Address Register	0x20080	Table 92, p.250
CPU Control and Status Registers		
CPU Configuration Register	0x20100	Table 93, p.250
CPU Control and Status Register	0x20104	Table 94, p.251
RSTOUTn Mask Register	0x20108	Table 95, p.252
System Soft Reset Register	0x2010C	Table 96, p.252
System Soft Reset Register	0x20110	Table 97, p.252
AHB to Mbus Bridge Interrupt Mask Register	0x20114	Table 98, p.253
Main Interrupt Controller Registers		
Main Interrupt Cause Register	0x20200	Table 99, p.253
Main IRQ Interrupt Mask Register	0x20204	Table 100, p.255
Main FIQ Interrupt Mask Register	0x20208	Table 101, p.255
Endpoint Interrupt Mask Register	0x2020C	Table 102, p.256
CPU Timers Registers		
CPU Timers Control Register	0x20300	Table 103, p.256

Table 71: CPU Register Map (Continued)

Register Name	Offset	Table Number and Page Number
CPU Timer0 Reload Register	0x20310	Table 104, p.257
CPU Timer 0 Register	0x20314	Table 105, p.257
CPU Timer1 Reload Register	0x20318	Table 106, p.257
CPU Timer 1 Register	0x2031C	Table 107, p.257
CPU Watchdog Timer Reload Register	0x20320	Table 108, p.258
CPU Watchdog Timer Register	0x20324	Table 109, p.258
CPU Doorbell Registers		
Host-to-CPU Doorbell Register	0x20400	Table 110, p.258
Host-to-CPU Doorbell Mask Register	0x20404	Table 111, p.258
CPU-to-Host Doorbell Register	0x20408	Table 112, p.259
CPU-to-Host Doorbell Mask Register	0x2040C	Table 113, p.259

A.4.1 CPU Address Map Registers

Table 72: Window0 Control Register
Offset:0x20000

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x1	Window0 Enable 0 = Disabled: Window is disabled. 1 = Enabled: Window is enabled.
3:1	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x4	Specifies the target interface associated with this window. See Section 2.10 "Default Address Map" on page 22. **NOTE:** Do not configure this field to the SDRAM Controller.
15:8	Attr	RW 0x59	Specifies the target interface attributes associated with this window. See Section 2.10 "Default Address Map" on page 22.
31:16	Size	RW 0x1FFF	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 KB granularity (e.g., a value of 0x00FF specifies 256 = 16 MB). **NOTE:** A value of 0x0 specifies 64-KB size.

Table 73: Window0 Base Register
Offset:0x20004

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!

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0x8000	Base Address Used with the <Size> field to set the address window size and location. Corresponds to transaction address[31:16].

Table 74: Window0 Remap Low Register
Offset:0x20008

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0x8000	Remap Address Used with the <Size> field to specifies address bits[31:0] to be driven to the target interface.

Table 75: Window0 Remap High Register
Offset:0x2000C

Bits	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address Specifies address bits[63:32] to be driven to the target interface.

Table 76: Window1 Control Register
Offset:0x20010

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x1	Window1 Enable. See the Window0 Control Register (Table 72 p. 243).
3:1	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x3	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register (Table 72 p. 243).
15:8	Attr	RW 0x59	Target specific attributes depending on the target interface. See the Window0 Control Register (Table 72 p. 243).

Table 76: Window1 Control Register (Continued)
Offset:0x20010

Bits	Field	Type/InitVal	Description
31:16	Size	RW 0x1FFF	Window Size See the Window0 Control Register (Table 72 p. 243).

Table 77: Window1 Base Register
Offset:0x20014

NOTE: If the remap function for this register is not used, the <Remap> field in the Window1 Remap Low Register must be set to same value as the <Base> field in this register!

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0xA000	Base Address See the Window0 Base Register.

Table 78: Window1 Remap Low Register
Offset:0x20018

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0xA000	Remap Address See the Window0 Remap Low Register.

Table 79: Window1 Remap High Register
Offset:0x2001C

Bits	Field	Type/InitVal	Description
31:0	RemapHigh	RW 0x0	Remap Address See the Window0 Remap High Register.

Table 80: Window2 Control Register
Offset:0x20020

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x1	Window2 Enable See the Window0 Control Register (Table 72 p. 243).

Table 80: Window2 Control Register (Continued)
Offset:0x20020

Bits	Field	Type/InitVal	Description
3:1	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x4	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register (Table 72 p. 243).
15:8	Attr	RW 0x51	Target specific attributes depending on the target interface. See the Window0 Control Register (Table 72 p. 243).
31:16	Size	RW 0x0	Window Size See the Window0 Control Register (Table 72 p. 243).

Table 81: Window2 Base Register
Offset:0x20024

NOTE:

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0xC000	Base Address See the Window0 Base Register.

Table 82: Window3 Control Register
Offset:0x20030

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x1	Window3 Enable See the Window0 Control Register (Table 72 p. 243).
3:1	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x3	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register (Table 72 p. 243).
15:8	Attr	RW 0x51	Target specific attributes depending on the target interface. See the Window0 Control Register (Table 72 p. 243).
31:16	Size	RW 0x0	Window Size See the Window0 Control Register (Table 72 p. 243).

Table 83: Window3 Base Register
Offset:0x20034

NOTE:

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0xC800	Base Address See the Window0 Base Register.

Table 84: Window4 Control Register
Offset:0x20040

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x1	Window4 Enable
3:1	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x9	Specifies the unit ID (target interface) associated with this window.
15:8	Attr	RW 0x0	Target specific attributes depending on the target interface. See the Window0 Control Register (Table 72 p. 243).
31:16	Size	RW 0x0	Window Size See the Window0 Control Register (Table 72 p. 243).

Table 85: Window4 Base Register
Offset:0x20044

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0xC801	s

Table 86: Window5 Control Register
Offset:0x20050

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x0	Window5 Enable See the Window0 Control Register (Table 72 p. 243).
3:1	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register (Table 72 p. 243).
15:8	Attr	RW 0x0	Target specific attributes depending on the target interface. See the Window0 Control Register (Table 72 p. 243).
31:16	Size	RW 0x0	Window Size See the Window0 Control Register (Table 72 p. 243).

Table 87: Window5 Base Register
Offset:0x20054

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0x0	Base Address See the Window0 Base Register.

Table 88: Window6 Control Register
Offset:0x20060

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x1	Window6 Enable See the Window0 Control Register (Table 72 p. 243).
1	WinWrProt	RW 0x0	Window6 Write Protection. When this bit is set to 1, the window is write protected and a write transaction to this window responds to the Feroceon® CPU core with an indication. 0 = Not protected: Not write protected 1 = Protected: Write protected
3:2	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register (Table 72 p. 243).
15:8	Attr	RW 0x1B	Target specific attributes depending on the target interface. See the Window0 Control Register (Table 72 p. 243).

Table 88: Window6 Control Register (Continued)
Offset:0x20060

Bits	Field	Type/InitVal	Description
31:16	Size	RW 0x07FF	Window Size See the Window0 Control Register (Table 72 p. 243).

Table 89: Window6 Base Register
Offset:0x20064

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0xF000	Base Address See the Window0 Base Register.

Table 90: Window7 Control Register
Offset:0x20070

Bits	Field	Type/InitVal	Description
0	win_en	RW 0x1	Window7 Enable See the Window0 Control Register (Table 72 p. 243).
1	WinWrProt	RW 0x0	Window7 Write Protection When this bit is set to 1, the window is write protected and a write transaction to this window responds to the Feroceon CPU core with an error indication. 0 = Not protected: Not write protected 1 = Protected: Write protected
3:2	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register (Table 72 p. 243).
15:8	Attr	RW 0x0F	Target specific attributes depending on the target interface. See the Window0 Control Register (Table 72 p. 243).
31:16	Size	RW 0x07FF	Window Size See the Window0 Control Register (Table 72 p. 243).

Table 91: Window7 Base Register
Offset:0x20074

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved

Table 91: Window7 Base Register (Continued)
Offset:0x20074

Bits	Field	Type/InitVal	Description
31:16	Base	RW 0xF800	Base Address See the Window0 Base Register.

Table 92: 88F5182 Internal Registers Base Address Register
Offset:0x20080

Bits	Field	Type/InitVal	Description
19:0	Reserved	RSVD 0x0	Reserved The size of the 88F5182 Internal registers address space is 1 MB.
31:20	Base	RW 0xD00	Internal registers Base Address

A.4.2 CPU Control and Status Registers

Table 93: CPU Configuration Register
Offset:0x20100

Bits	Field	Type/InitVal	Description
0	EndPointIF	RW Sample at reset	When the 88F5182 functions as Endpoint, this bit defines the interface connected to host. The default value of this bit is 0 when PCI Express0 functions as Endpoint. The default value of this bit is 1 when PCI Express0 functions as Root Complex. 0 = PCI Express: Host is connected to PCI Express interface. 1 = PCI: Host is connected to PCI interface.
1	VecInitLoc	RW 0x1	Determines the reset location of the boot starting address. **NOTE: For 88F5x8x:** At func_mode_ boot from DDR. Func_mode_ determined by sampling DEV_AD[18] pad at reset. 0 = 0x00000000: Boot starting address is 0x00000000. 1 = 0xFFFF0000: Boot starting address is 0xFFFF0000.
2	AHBErrorProp	RW 0x1	AHB Error propogation For a list of errors see Section 21.4, Error Handling, on page 215. 0 = Error not propagated: Error indications are not propagated to AHB bus. The transactions are completed normally. 1 = Error propagated: Error indications are propagated to AHB bus.
3	EndianInit	RW Sample at reset value of DEV_D[17]	Endian Initialization Determines the endianness of the 88F5182 CPU core operation after reset. This bit defines the initial value of bit [7] <B> of the Control Register—R1. 0 = Little Endian: Little Endian mode 1 = Big Endian: Big Endian mode
4	Reserved	RSVD 0x0	Reserved

Table 93: CPU Configuration Register (Continued)
Offset:0x20100

Bits	Field	Type/Init Val	Description
5	MMU_Disabled	RW 0x0	MMU Disabled When set to 1, this bit disables the MMU and activates the MPU instead.
22:6	Reserved	RSVD 0x0	Reserved
23	PexLinkdown ResetCpu	RW 0x0	PCI Express Link Down Reset of CPU 0 = PCI Express link down value does not affect CPU reset. 1 = CPU reset remains asserted when PCI Express link down is asserted.
31:24	Reserved	RSVD 0x0	s

Table 94: CPU Control and Status Register
Offset:0x20104

Bits	Field	Type/InitVal	Description
0	PCIDs	RW 0x1	When this bit is set to 1, PCI transaction towards 88F5182 address space is terminated with retry and PCI Express link negotiation is disabled. This is used to block PCI Express from access 88F5182 while CPU boot is still in progress. 0 = PCI and PCI Express Enable 1 = PCI and PCI Express Disable
1	CPUReset	RO - CPU RW - other 0x0	CPU Reset When this bit is set to 1, the Feroceon CPU core is reset.
2	SelfInt	SC 0x0	When set to 1, the <CPUSelfInt> field in the AHB to Mbus Bridge Interrupt Cause Register (Table 97 p. 252) is also set to 1.
14:3	Reserved	RSVD 0x0	Reserved
15	BigEndian	RO 0x0	Big Endian Reflects the value of the Big Endian field of the Feroceon CPU CP15 register. 0x0 = Little Endian mode 0x1 = Big Endian mode
31:16	Reserved	RSVD 0x0	Reserved

Table 95: RSTOUTn Mask Register
Offset:0x20108

Bits	Field	Type/InitVal	Description
0	PexRstOutEn	RW 0x0	If set to 1, the 88F5182 asserts RSTOUTn upon receiving a PCI Express reset indication from the PCI Express Endpoint interface, as configured in the <EndPointIF> field in the CPU Configuration Register (Table 93 p. 250).
1	WDRstOutEn	RW 0x0	If set to 1, the 88F5182 asserts RSTOUTn upon watchdog timer expiration (See Table 109, CPU Watchdog Timer Register, on page 258)
2	SoftRstOutEn	RW 0x0	If set to 1, the 88F5182 asserts RSTOUTn upon SW reset. (See Table 96, System Soft Reset Register, on page 252)
31:3	Reserved	RO 0x0	

Table 96: System Soft Reset Register
Offset:0x2010C

Bits	Field	Type/InitVal	Description
0	SystemSoftRst	RW 0x0	When SW set this bit to 1 and bit <SoftRstOutEn> is set to 1 in the RSTOUTn Mask Register, the 88F5182 asserts the RSTOUTn pin.
31:1	Reserved	RSVD 0x0	Reserved

Table 97: AHB to Mbus Bridge Interrupt Cause Register
Offset:0x20110

NOTE: A cause bit is set upon an error condition occurrence. Writing a 0 value clears the bit. Writing a 1 value has no affect.

Bits	Field	Type/InitVal	Description
0	CPUSelfInt	RWC 0x0	This bit is set when bit <SelfInt> is set to 1 in CPU Control and Status Register.
1	CPUTimer0IntReq	RWC 0x0	CPU Timer 0 Interrupt This bit is set when field <CPUTimer0> in CPU Timer 0 Register reaches 0.
2	CPUTimer1IntReq	RWC 0x0	CPU Timer 1 Interrupt This bit is set when field <CPUTimer1> in CPU Timer 1 Register reaches 0.
3	CPUWDTimerIntReq	RWC 0x0	CPU Watchdog Timer Interrupt This bit is set when field <CPUWDTimer> in CPU Watchdog Timer Register reaches 0.
31:4	Reserved	RWC 0x0	s

Table 98: AHB to Mbus Bridge Interrupt Mask Register
Offset:0x20114

Bits	Field	Type/InitVal	Description
3:0	Mask	RW 0x0	There is a mask bit per each cause bit. 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register.
31:4	Reserved	RSVD 0x0	Reserved

A.4.3 Main Interrupt Controller Registers

Table 99: Main Interrupt Cause Register
Offset: 0x20200

NOTE: All bits are read only. To clear an interrupt, software must access the Local Interrupt Cause registers.

Bits	Field	Type/InitVal	Description
0	Bridge	RO 0x0	AHB to Mbus Bridge interrupt This bit is set when at least one bit is set in AHB to Mbus Bridge Interrupt Cause Register and is not masked by the corresponding bit in AHB to Mbus Bridge Interrupt Mask Register
1	Host2CPUDoorbell	RO 0x0	Doorbell interrupt This bit is set when at least one bit is set in Host-to-CPU Doorbell Register and is not masked by the corresponding bit in Host-to-CPU Doorbell Mask Register.
2	CPU2HostDoorbell	RO 0x0	Doorbell interrupt This bit is set when at least one bit is set in CPU-to-Host Doorbell Register and is not masked by the corresponding bit in CPU-to-Host Doorbell Mask Register.
3	UART0	RO 0x0	UART0 interrupt
4	UART1	RO 0x0	UART1 Interrupt
5	TWSI	RO 0x0	TWSI interrupt (TWSI port 0)
6	GPIO7_0	RO 0x0	GPIO[7:0] interrupt
7	GPIO15_8	RO 0x0	GPIO[15:8] interrupt
8	GPIO23_16	RO 0x0	GPIO[23:16] interrupt

Table 99: Main Interrupt Cause Register (Continued)
Offset: 0x20200

NOTE: All bits are read only. To clear an interrupt, software must access the Local Interrupt Cause registers.

Bits	Field	Type/InitVal	Description
9	GPIO25_24	RO 0x0	GPIO[25:24] interrupt
10	PEX0Err	RO 0x0	PCI Express error
11	PEX0INT	RO 0x0	PCI Express INTA, B, C, and D message
12	USBCnt1	RO 0x0	USB Port 1 controller interrupt
13	Reserved	RO 0x0	Reserved Read only
14	DEVErr	RO 0x0	Device bus error (DEV_READY timeout)
15	PCIErr	RO 0x0	PCI error
16	USBBr	RO 0x0	USB bridge Port 0 or 1 error
17	USBCnt0	RO 0x0	USB Port 0 controller interrupt
18	GbERx	RO 0x0	GbE receive interrupt
19	GbETx	RO 0x0	GbE transmit interrupt
20	GbEMisc	RO 0x0	GbE miscellaneous interrupt
21	GbESum	RO 0x0	GbE summary
22	GbEErr	RO 0x0	GbE error
23	DMAErr	RO 0x0	DMA or XOR error
24	IDMA0	RO 0x0	IDMA Channel0 completion
25	IDMA1	RO 0x0	IDMA Channel1 completion
26	IDMA2	RO 0x0	IDMA Channel2 completion

Table 99: Main Interrupt Cause Register (Continued)
Offset: 0x20200

NOTE: All bits are read only. To clear an interrupt, software must access the Local Interrupt Cause registers.

Bits	Field	Type/InitVal	Description
27	IDMA3	RO 0x0	IDMA Channel3 completion
28	SecurityInterrupt	RO 0x0	Security accelerator interrupt indication
29	SataInterrupt	RO 0x0	Serial-ATA interrupt indication
30	XOR0	RO 0x0	XOR engine 0 completion interrupt indication
31	XOR1	RO 0x0	s

Table 100: Main IRQ Interrupt Mask Register
Offset:0x20204

Bits	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	Mask bit per each cause bit 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of Feroceon CPU core IRQ interrupt line. It does not affect the setting of bits in the Cause register.

Table 101: Main FIQ Interrupt Mask Register
Offset:0x20208

Bits	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	Mask bit per each cause bit 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of Feroceon CPU core FIQ interrupt line. It does not affect the setting of bits in the Cause register.

Table 102: Endpoint Interrupt Mask Register
Offset:0x2020C

Bits	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	Mask bit per each cause bit 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of the interrupt pin. It does not affect the setting of bits in the Cause register. The interrupt pin is asserted in the appropriate interface as defined by bit <EndPointIF> field in the CPU Configuration Register (Table 93 p. 250).

A.4.4 CPU Timers Registers

Table 103: CPU Timers Control Register
Offset:0x20300

Bits	Field	Type/InitVal	Description
0	CPUTimer0En	RW 0x0	CPU Timer 0 Enable 0 = Timer0 is disabled. 1 = Timer0 is enabled.
1	CPUTimer0Auto	RW 0x0	CPU Timer 0 Auto Mode When this bit is cleared to 0 and <CPUTimer0> has reached zero, <CPUTimer0> stops counting. When this bit is set to 1 and <CPUTimer0> has reached zero, <CPUTimer0Rel> is reload to <CPUTimer0>, then it continues to count.
2	CPUTimer1En	RW 0x0	CPU Timer 1 Enable 0 = Timer1 is disabled. 1 = Timer1 is enabled.
3	CPUTimer1Auto	RW 0x0	CPU Timer 1 Auto Mode When this bit is clear to 0 and <CPUTimer1> has reached to zero, <CPUTimer1> stops counting When this bit is set to 1 and <CPUTimer1> has reached zero, <CPUTimer1Rel> is reload to <CPUTimer1>, then it continues to count.
4	CPUWDTimerEn	RW Sample at reset	CPU Watchdog Timer Enable 0 = Watchdog timer is disabled. 1 = Watchdog timer is enabled.
5	CPUWDTimerAuto	RW 0x0	CPU Watchdog Timer Auto Mode When this bit is clear to 0 and <CPUWDTimer> has reached zero, <CPUWDTimer> stops counting. When this bit is set to 1 and <CPUWDTimer> has reached zero, <CPUTimer0Rel> is reload to <CPUWDTimer>, then it continues to count.
31:6	Reserved	RW 0x0	Reserved

Table 104: CPU Timer0 Reload Register
Offset:0x20310

Bits	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 105: CPU Timer 0 Register
Offset:0x20314

Bits	Field	Type/InitVal	Description
31:0	CPUTimer0	RW 0x0	CPU Timer 0 This 32-bit counter is decremented every 88F5182 internal clock When <CPUTimer0En> = 0, <CPUTimer0> stop. When <CPUTimer0En> = 1 and <CPUTimer0Auto> = 0, CPUTimer0 is decremented until it reaches to 0, then it stops. When <CPUTimer0En> = 1 and <CPUTimer0Auto> = 1, <CPUTimer0> is decremented until it reaches to 0, then <CPUTimer0Rel> is reload to <CPUTimer0> and then <CPUTimer0> continues to count. When <CPUTimer0> reaches 0, the<CPUTimer0IntReq> field in the AHB to Mbus Bridge Interrupt Cause Register (Table 97 p. 252) is set to 1.

Table 106: CPU Timer1 Reload Register
Offset:0x20318

Bits	Field	Type/InitVal	Description
31:0	CPUTimer1Rel	RW 0x0	CPU Timer 1 Reload See the CPU Timer0 Reload Register.

Table 107: CPU Timer 1 Register
Offset:0x2031C

Bits	Field	Type/InitVal	Description
31:0	CPUTimer1	RW 0x0	CPU Timer 0 See the CPU Timer 0 Register.

Table 108: CPU Watchdog Timer Reload Register
Offset:0x20320

Bits	Field	Type/InitVal	Description
31:0	CPUWDTimerLen	RW 0x0	CPU Timer 1 Length See the CPU Timer0 Reload Register.

Table 109: CPU Watchdog Timer Register
Offset:0x20324

Bits	Field	Type/InitVal	Description
31:0	CPUWDTimer	RW 0x7FFF_FFFF	CPU Watchdog Timer See the CPU Timer 0 Register.

A.4.5 CPU Doorbell Registers

Table 110: Host-to-CPU Doorbell Register
Offset:0x20400

Bits	Field	Type/InitVal	Description
31:0	HostIntCs	Host: RW CPU: RW 0x0	Host Command Cause When this bit is not zero and the corresponding bit <HostIntCsMask> in the Host-to-CPU Doorbell Mask Register is set, bit <Host2CPUDoorbell> is set in the Main Interrupt Cause Register. A Host write of 1 sets the bits in this field. A Host write of 0 has no effect. An CPU write of 0 clear the bits in this field. An CPU write of 1 has no effect.

Table 111: Host-to-CPU Doorbell Mask Register
Offset:0x20404

Bits	Field	Type/InitVal	Description
31:0	HostIntCsMask	RW 0x0	Host Interrupt Cause Mask Mask bit per each cause bit in the <HostIntCs> field in the Host-to-CPU Doorbell Register. 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of interrupt bit in Main Interrupt Cause Register. It does not affect the setting of bits in the Host-to-CPU Doorbell Register.

Table 112: CPU-to-Host Doorbell Register
Offset:0x20408

Bits	Field	Type/InitVal	Description
31:0	CPUIntCs	CPU: RW Host: RW 0x0	CPU Interrupt Cause When a bit in this field is set and the corresponding bit in <CPUIntCsMask> in the CPU-to-Host Doorbell Mask Register is also set, bit <Host2CPUDoorbell> is set in the Main Interrupt Cause Register. An CPU write of 1 set the bits in this field.An CPU write of 0 has no effect. A Host write of 0 clear the bits in this field. A Host write of 1 has no effect.

Table 113: CPU-to-Host Doorbell Mask Register
Offset:0x2040C

Bits	Field	Type/InitVal	Description
31:0	CPUIntCsMask	RW 0x0	CPU Interrupt Cause Mask Mask bit per each cause bit in <CPUIntCs> field in the CPU-to-Host Doorbell Register. 0 = Interrupt is masked. 1 = Interrupt is enabled. 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 CPU-to-Host Doorbell Register.

A.5 DDR SDRAM Controller Registers

Table 114: DDR SDRAM Register Map

Register Name	Offset	Table and Page
DDR SDRAM Controller Address Decode Registers		
CS[0]n Base Address Register	0x01500	Table 115, p.261
CS[0]n Size Register	0x01504	Table 116, p.261
CS[1]n Base Address Register	0x01508	Table 117, p.261
CS[1]n Size Register	0x0150C	Table 118, p.262
CS[2]n Base Address Register	0x01510	Table 119, p.262
CS[2]n Size Register	0x01514	Table 120, p.262
CS[3]n Base Address Register	0x01518	Table 121, p.263
CS[3]n Size Register	0x0151C	Table 122, p.263
DDR SDRAM Control Registers		
DDR SDRAM Configuration Register	0x01400	Table 123, p.263
DDR SDRAM Control Register	0x01404	Table 124, p.264
DDR SDRAM Timing (Low) Register	0x01408	Table 125, p.265
DDR SDRAM Timing (High) Register	0x0140C	Table 126, p.266
DDR2 SDRAM Timing (Low) Register	0x01428	Table 127, p.267
DDR2 SDRAM Timing (High) Register	0x0147C	Table 128, p.267
DDR SDRAM Address Control Register	0x01410	Table 129, p.268
DDR SDRAM Open Pages Control Register	0x01414	Table 130, p.268
DDR SDRAM Operation Register	0x01418	Table 131, p.268
DDR SDRAM Operation Control Register	0x0142C	Table 132, p.269
DDR SDRAM Mode Register	0x0141C	Table 133, p.269
Extended DDR SDRAM Mode Register	0x01420	Table 134, p.270
DDR SDRAM Initialization Control Register	0x01480	Table 135, p.271
DDR SDRAM Address/Control Pads Calibration Register	0x014C0	Table 136, p.272
DDR SDRAM Data Pads Calibration Register	0x014C4	Table 137, p.272
DDR2 SDRAM ODT Control (Low) Register	0x01494	Table 138, p.273
DDR2 SDRAM ODT Control (High) Register	0x01498	Table 139, p.273
DDR2 SDRAM ODT Control Register	0x0149C	Table 140, p.274
DDR SDRAM Interface Mbus Control (Low) Register	0x01430	Table 141, p.274
DDR SDRAM Interface Mbus Control (High) Register	0x01434	Table 142, p.275
DDR SDRAM Interface Mbus Timeout Register	0x01438	Table 143, p.276
DDR SDRAM MMask Register	0x014B0	Table 144, p.276

The base and size of a base address register may be changed only when that base address register is disabled.

A.5.1 DDR SDRAM Controller Address Decode Registers

Table 115: CS[0]n Base Address Register
Offset: 0x01500

Bits	Field	Type/InitVal	Description
15:0	Reserved	RO 0x0	Reserved
23:16	Reserved	RW 0x0	Reserved
31:24	Base	RW 0x00	s

Table 116: CS[0]n Size Register
Offset:0x01504

Bits	Field	Type/InitVal	Description
0	En	RW 0x1	Window Enable 0 = Disable 1 = Enable
15:1	Reserved	RO 0x0	Reserved
23:16	Reserved	RW 0xFF	Reserved
31:24	Size	RW 0x0F	CS[0]n Bank Size Corresponds to Base Address bits [31:24]. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's.

Table 117: CS[1]n Base Address Register
Offset:0x01508

Bits	Field	Type/InitVal	Description
15:0	Reserved	RO 0x0	Reserved
23:16	Reserved	RW 0x0	Reserved
31:24	Base	RW 0x10	CS[1] Base Address. Corresponds to Feroceon® CPU core address bits [31:24].

Table 118: CS[1]n Size Register
Offset:0x0150C

Bits	Field	Type/InitVal	Description
0	En	RW 0x1	Window Enable 0 = Disable 1 = Enable
15:1	Reserved	RO 0x0	Reserved
23:16	Reserved	RO 0xFF	Reserved
31:24	Size	RW 0x0F	CS[1]n Bank Size Corresponds to Base Address bits [31:24]. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's.

Table 119: CS[2]n Base Address Register
Offset:0x01510

Bits	Field	Type/InitVal	Description
15:0	Reserved	RO 0x0	Reserved
23:16	Reserved	RW 0xFF	Reserved
31:24	Base	RW 0x20	CS[2] Base Address. Corresponds to Feroceon CPU core address bits [31:24].

Table 120: CS[2]n Size Register
Offset:0x01514

Bits	Field	Type/InitVal	Description
0	En	RW 0x1	Window Enable 0 = Disable 1 = Enable
15:1	Reserved	RO 0x0	Reserved
23:16	Reserved	RW 0xFF	Reserved
31:24	Size	RW 0x0F	CS[2]n Bank Size Corresponds to Base Address bits [31:24]. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's.

Table 121: CS[3]n Base Address Register
Offset:0x01518

Bits	Field	Type/InitVal	Description
15:0	Reserved	RO 0x0	Reserved
23:16	Reserved	RW 0x0	Reserved
31:24	Base	RW 0x30	CS[3] Base Address. Corresponds to Feroceon CPU core address bits [31:24].

Table 122: CS[3]n Size Register
Offset:0x0151C

Bits	Field	Type/InitVal	Description
0	En	RW 0x1	Window Enable 0 = Disable 1 = Enable
15:1	Reserved	RO 0x0	Reserved
23:16	Reserved	RW 0xFF	Reserved
31:24	Size	RW 0x0F	CS[3]n Bank Size Corresponds to Base Address bits [31:24]. Must be programmed from LSB to MSB as a sequence of 1's followed by a sequence of 0's.

A.5.2 DDR SDRAM Control Registers

Table 123: DDR SDRAM Configuration Register
Offset: 0x01400

Bits	Field	Type/InitVal	Description
13:0	Refresh	RW 0x0400	Refresh interval count value
15:14	Dwidth	RW 0x2	this is a new field. 0x0 = Reserved 0x1 = 16-bit DDR SDRAM interface 0x2 = 32-bit DDR SDRAM interface 0x3 = Reserved
16	Dtype	RW Reset Strap	DDR SDRAM Type 0 = DDR1 1 = DDR2

Table 123: DDR SDRAM Configuration Register (Continued)
Offset: 0x01400

Bits	Field	Type/InitVal	Description
17	RegDIMM	RW 0x0	Registered DIMM enable 0 = Non-buffered DIMM 1 = Registered DIMM **NOTE:** Even if using DDR SDRAM devices on board, this register can still be used to buffer DDR SDRAM address/control signals. In that case, set <RegDIMM> bit to 1. There are no registered DIMMs. We might remove this feature if we conclude that no need registers on board.
18	Perr	RW 0x1	Erroneous write data policy 0 = Ignore erroneous data indication. Write data to DDR SDRAM. 1 = Do not write to DDR SDRAM upon erroneous data indication.
19	Reserved	RW 0x0	Reserved bit[18] used to be ECC. bit[19] IErr
21:20	DCfg	RW 0x2	DDR SDRAM devices configuration 0x0 = Reserved used to be x32 0x1 = x16 devices 0x2 = x8 devices 0x3 = Reserved used to be x4
23:22	Reserved	RO 0x0	Reserved
24	SRMode	RW 0x1	Reserved Must be 0x1. Self Refresh Mode 0x0 = Once entered self refresh, exit only upon power on reset. Useful for battery backup applications 0x1 = Exit self refresh when new SDRAM access is requested.
25	SRClk	RW 0x1	Clock drive upon self refresh 0 = Clock is kept driven during self refresh. 1 = Clock is gated during self refresh.
31:26	Reserved	RO 0x0	s

Table 124: DDR SDRAM Control Register
Offset:0x01404

Bits	Field	Type/InitVal	Description
5:0	Reserved	RO 0x0	Reserved
6	CtrlPos	RW 0x1	Address/Control Output Timing 0 = Toggle on falling edge of clock 1 = Toggle on rising edge of clock **NOTE:** Set according to board timing simulation results.

Table 124: DDR SDRAM Control Register (Continued)
Offset:0x01404

Bits	Field	Type/InitVal	Description
11:7	Reserved	RO 0x0	Reserved (bit[7] used to be read data sample control; bit[8] used to be read data path sync mode; bit[9] used to be RMW Path Synchronization; bits[11:10] used to be Prio - now arbiter always give priority to AHB bus)
12	Clk1Drv	RW 0x1	M_CLK_OUT[1] Drive 0 = M_CLK_OUT[1] and M_CLK_OUTn[1] are high-z. 1 = M_CLK_OUT[1] and M_CLK_OUTn[1] are driven normally. **NOTE:** When not using M_CLK_OUT[1], set it to high-z.
13	Reserved	RO 0x0	Reserved
17:14	Reserved	RO 0x0	Reserved
18	LockEn	RW 0x1	CPU Lock Enable 0 = Disable) 1 = Enable
23:19	Reserved	RO 0x0	Reserved
27:24	StBurstDel	RW 0x3	Number of sample stages on StartBurstIn Program StBurstDel based on CL, registered/non-buffered DIMM.
31:28	Reserved	RO 0x0	Reserved

Table 125: DDR SDRAM Timing (Low) Register
Offset:0x01408

NOTE: The default values fit DDR2-400 speed grade. Change these timing parameters according to the DDR SDRAM type and operating frequency.

Bits	Field	Type/InitVal	Description
3:0	Reserved	RO 0x0	Reserved
7:4	t_{RCD}	RW 0x2	Activate to Command Value 0 means one cycle; value of 1 means two cycles; and so on.
11:8	t_{RP}	RW 0x2	Precharge Period (precharge to active) Value 0 means one cycle; value of 1 means two cycles; and so on.
15:12	t_{WR}	RW 0x2	Write Command to Precharge Value 0 means one cycle; value of 1 means two cycles; and so on.
19:16	t_{WTR}	RW 0x1	Write Command to Read Command Value 0 means one cycle; value of 1 means two cycles; and so on.
23:20	t_{RAS}	RW 0x8	Minimum Row Active Time (active to precharge) Value 0 means one cycle; value of 1 means two cycles; and so on.

Table 125: DDR SDRAM Timing (Low) Register (Continued)
Offset:0x01408

NOTE: The default values fit DDR2-400 speed grade. Change these timing parameters according to the DDR SDRAM type and operating frequency.

Bits	Field	Type/InitVal	Description
27:24	t_{RRD}	RW 0x1	Activate Bank A to Activate Bank B Value 0 means one cycle; value of 1 means two cycles; and so on.
31:28	t_{RTP}	RW 0x1	Read Command to Precharge Value 0 means one cycle; value of 1 means two cycles; and so on. **NOTE:** Must be set to 0x1 (two cycles) when using DDR1.

Table 126: DDR SDRAM Timing (High) Register
Offset:0x0140C

Bits	Field	Type/InitVal	Description
3:0	t_{RFC}	RW 0xD	Refresh Command Period Value 0 means one cycle; value of 1 means two cycles; and so on.
5:4	t_{R2R}	RW 0x0	Minimum Gap Between DDR SDRAM Read Accesses This timing parameter is not part of the JEDEC standard. It is used to prevent contention between read data driven from two different DDR SDRAM devices (DIMMs). 0 = One cycle 1 = Two cycles 2, 3 = Reserved
7:6	t_{R2W_W2R}	RW 0x0	Minimum Gap Between DDR SDRAM Read and Write Accesses This timing parameter is not part of the JEDEC standard. It is used to prevent contention between read and write data driven from two different devices (same parameter for read after write and write after read). 0 = One cycle 1 = Two cycles 2, 3 = Reserved
9:8	t_{RFC}	RW 0x0	Extension (MSB) of t_{RFC} (bits [3:0])
11:10	t_{W2W}	RW 0x0	Minimum Gap between Write to Write to different DDR SDRAM devices. Value 0 means no gap; value 1 means one cycle gap; and so on.
31:12	Reserved	RO 0x0	Reserved

Table 127: DDR2 SDRAM Timing (Low) Register
Offset:0x01428

Bits	Field	Type/InitVal	Description
3:0	Reserved	RW 0x5	Reserved
7:4	$t_{ODT\_ON\_RD}$	RW 0x0	The number of cycles from Read command to the assertion of M_ODT signal. Value 0 means same cycle as read command; value of 1 means one cycle later and so on. This value depends on DDR SDRAM CL and t_{AOND} timing parameters. For CL = 3 and t_{AOND} = 2, set t_{ODT_ON} to 0.
11:8	$t_{ODT\_OFF\_RD}$	RW 0x3	The number of cycles from Read command to de-asserting M_ODT signal. Value depends on DDR SDRAM CL and t_{AOfD} timing parameters. For CL = 3 and t_{AOFD} = 2.5, set t_{ODT_OFF} to 0x3.
15:12	$t_{ODT\_ON\_CTL\_RD}$	RW 0x3	The number of cycles from Read command to the assertion of the internal ODT signal to the DDR SDRAM controller I/O buffer. The same as t_{ODT_ON}
19:16	$t_{ODT\_OFF\_CTL\_RD}$	RW 0x6	The number of cycles from Read command to de-asserting of the internal ODT signal to the DDR SDRAM controller I/O buffer. The same as t_{ODT_OFF}.
31:20	Reserved	RO 0x0	Reserved

Table 128: DDR2 SDRAM Timing (High) Register
Offset:0x0147C

Bits	Field	Type/InitVal	Description
3:0	$t_{ODT\_ON\_WR}$	RW 0x0	The number of cycles from Write command to the assertion of M_ODT signal. Value 0 means one cycle before write command; value of 1 means the same cycle as write command, value 2 means one cycle latter and so on. Value depends on DDR SDRAM CL and t_{AOND} timing parameters. For CL = 3 and t_{AOND} = 2, set $t_{ODT\_ON\_WR}$ to 0.
7:4	$t_{ODT\_OFF\_WR}$	RW 0x3	The number of cycles from Write command to de-asserting M_ODT signal. Value depends on DDR SDRAM CL and t_{AOFD} timing parameters. For CL = 3 and t_{AOFD} = 2.5, set $t_{ODT\_OFF\_WR}$ to 0x3.
11:8	$t_{ODT\_ON\_CTL\_WR}$	RW 0x3	The number of cycles from Write command to the assertion of the internal ODT signal to the DDR SDRAM controller I/O buffer. Same as $t_{ODT\_ON\_CTL\_WR}$.
15:12	$t_{ODT\_OFF\_CTL\_WR}$	RW 0x6	The number of cycles from Write command to de-asserting of the internal ODT signal to the DDR SDRAM controller I/O buffer. Same as $t_{ODT\_OFF\_CTL\_WR}$.
31:16	Reserved	RO 0x0	Reserved

Table 129: DDR SDRAM Address Control Register
Offset:0x01410

Bits	Field	Type/InitVal	Description
3:0	Reserved	RO 0x0	Reserved
5:4	DSize	RW 0x1	0x0 = 128 Mb **NOTE:** 128 Mb DDR SDRAM is relevant only to DDR1 0x1 = 256 Mb 0x2 = 512 Mb 0x3 = Reserved
31:6	Reserved	RO 0x0	Reserved

Table 130: DDR SDRAM Open Pages Control Register
Offset:0x01414

Bits	Field	Type/InitVal	Description
0	OPEn	RW 0x0	Open Pages Enable 0 = The DDR Controller keeps the corresponding bank page open whenever it can. 1 = The DDR Controller always closes the page at the end of the transaction.
31:1	Reserved	RO 0x0	Reserved

Table 131: DDR SDRAM Operation Register
Offset:0x01418

Bits	Field	Type/InitVal	Description
3:0	Cmd	SC 0x0	DDR SDRAM Mode Select 0x0 = Normal SDRAM Mode 0x1 = Precharge all banks command 0x2 = Refresh all banks command 0x3 = Mode Register Set (MRS) command 0x4 = Extended Mode Register Set (EMRS) command 0x5 = NOP command 0x6 = Reserved 0x7 = Enter self refresh 0x8 = EMRS2 command (DDR2 only) 0x9 = EMRS3 command (DDR2 only) 0xA–0xF = Reserved Setting this field results in the DDR Controller execution of the required command to the DDR SDRAM. Then, the DDR Controller resets this field to the default 0x0 value.

Table 131: DDR SDRAM Operation Register (Continued)
Offset:0x01418

Bits	Field	Type/InitVal	Description
31:4	Reserved	RO 0x0	Reserved

Table 132: DDR SDRAM Operation Control Register
Offset:0x0142C

Bits	Field	Type/InitVal	Description
1:0	CS	RW 0x0	DDR SDRAM chip select Defines to which DDR SDRAM bank to issue the EMRS1 command. This is useful for setting different ODT values for different DDR SDRAM banks. 0x0 = M_CSn[0] 0x1 = M_CSn[1] 0x2 = M_CSn[2] 0x3 = M_CSn[3]
31:2	Reserved	RO 0x0	Reserved

Table 133: DDR SDRAM Mode Register
Offset:0x0141C

NOTE: If configured to DDR1 SDRAM, bits[11:9] are not relevant, and must be set to 0x0.

Bits	Field	Type/InitVal	Description
2:0	BL	RW 0x2	Burst Length Must be set to 0x2 (burst length of 4).
3	BT	RW 0x0	Burst Type Must be set to 0x0 (sequential burst).
6:4	CL	RW 0x3	CAS Latency For DDR1 SDRAM: 0x2 = CL = 2 0x3 = CL = 3 0x4 = CL = 4 0x5 = CL = 1.5 0x6 = CL = 2.5 0x0, 0x1, 0x7 = Reserved For DDR2 SDRAM: 0x3 = CL = 3 0x4 = CL = 4 0x5 = CL = 5 0x0–0x2, 0x6, 0x7 = Reserved
7	TM	RW 0x0	Test Mode 0x0 = Normal operation 0x1 = Test mode

Table 133: DDR SDRAM Mode Register (Continued)
Offset:0x0141C

NOTE: If configured to DDR1 SDRAM, bits[11:9] are not relevant, and must be set to 0x0.

Bits	Field	Type/InitVal	Description
8	DLL	RW 0x0	Reset DLL 0x0 = Normal operation 0x1 = Reset DLL
11:9	WR	RW **DDR2:** 0x2 **DDR1:** 0x0	**For DDR2 SDRAM:** Write recovery for auto-precharge **NOTE:** Auto-precharge is not supported. **For DDR1 SDRAM:** Reserved must be set to 0x0.
12	PD	RW 0x0	Active power down exit time 0 = Fast exit 1 = Slow exit Must be 0x0. **NOTE:** Active power down is not supported for DDR2 SDRAM.
13	Reserved	RW 0x0	Reserved
31:14	Reserved	RO 0x0	Reserved

Table 134: Extended DDR SDRAM Mode Register
Offset:0x01420

NOTE: If configured to DDR1 SDRAM, bits[13:2] are not relevant, and must be set to 0x0.

Bits	Field	Type/InitVal	Description
0	DLL	RW 0x0	DDR SDRAM DLL Enable 0 = Enable 1 = Disable
1	DS	RW 0x0	DDR SDRAM Drive Strength 0 = Normal 1 = Reduced
2	Rtt[0]	RW **DDR2:** 0x1 **DDR1:** 0x0	**DDR2 SDRAM:** Rtt[1:0] is ODT Control 0x0 = ODT disable 0x1 = 75 ohm termination 0x2 = 150 ohm termination 0x3 = Reserved Refer to the *Design Considerations* for this product. **DDR1 SDRAM:** Reserved must be set to 0x0.
5:3	AL	RW **DDR2:** 0x0 **DDR1:** 0x0	**DDR2 SDRAM:** Additive Latency Must be 0x0. **NOTE:** Additive Latency is not supported. **DDR1 SDRAM:** Reserved must be set to 0x0.

Table 134: Extended DDR SDRAM Mode Register (Continued)
Offset:0x01420

NOTE: If configured to DDR1 SDRAM, bits[13:2] are not relevant, and must be set to 0x0.

Bits	Field	Type/InitVal	Description
6	Rtt[1]	RW **DDR2:** 0x0 **DDR1:** 0x0	**DDR2 SDRAM:** See <Rtt[0]>. Refer to the *Design Considerations* for this product. **DDR1 SDRAM:** Reserved must be set to 0x0.
9:7	Reserved	RW 0x0	Reserved
10	DQS	RW **DDR2:** 0x1 **DDR1:** 0x0	**DDR2 SDRAM:** Single ended DQS must be 0x1. **DDR1 SDRAM:** Reserved must be set to 0x0.
11	RDQS	RW **DDR2:** 0x0 **DDR1:** 0x0	**DDR2 SDRAM:** Read DQS 0 = RDQS disabled 1 = RDQS enabled Must be 0x0. **NOTE:** Read DQS is not supported. **DDR1 SDRAM:** Reserved must be set to 0x0.
12	Qoff	RW **DDR2:** 0x0 **DDR1:** 0x0	**DDR2 SDRAM:** DDR SDRAM output buffer enable 0 = Enabled 1 = Disabled **DDR1 SDRAM:** Reserved must be set to 0x0.
13	Reserved	RW 0x0	Reserved
31:14	Reserved	RO 0x0	Reserved

Table 135: DDR SDRAM Initialization Control Register
Offset:0x01480

Bits	Field	Type/InitVal	Description
0	InitEn	SC 0x0	Initialization enable DDR SDRAM initialization sequence starts upon setting this bit to 1.This bit is cleared when initialization completes.
31:1	Reserved	RO 0x0	Reserved

Table 136: DDR SDRAM Address/Control Pads Calibration Register
Offset:0x014C0

Bits	Field	Type/InitVal	Description
5:0	DrvN	RW 0x0	Pad Driving N Strength Refer to the *Design Considerations* for this product. **NOTE:** Only applicable when auto-calibration is disabled.
11:6	DrvP	RW 0x0	Pad Driving P Strength Refer to the *Design Considerations* for this product. **NOTE:** Only applicable when auto-calibration is disabled.
13:12	DriveStrength	RW 0x0	Drive Strength This field defines the output drive strength. Refer to the *Design Considerations* for this product. For DDR2 SDRAM: The value is 0x3. For DDR1 SDRAM: The value is 0x1.
15:14	Reserved	RO 0x0	Read Only
16	TuneEn	RW 0x1	Set to 1 enables the auto-calibration of pad driving strength.
22:17	LockN	RO 0x0	When auto-calibration is enabled, represents the final N locked value of the driving strength. Read Only Refer to the *Design Considerations* for this product.
28:23	LockP	RO 0x0	When auto-calibration is enabled, represents the final P locked value of the driving strength. Read Only Refer to the *Design Considerations* for this product.
30:29	Reserved	RO 0x0	Read Only
31	WrEn	RW 0x0	Write Enable 0 = Register is read only (except for bit [31]). 1 = Register is writable.

Table 137: DDR SDRAM Data Pads Calibration Register
Offset:0x014C4

Bits	Field	Type/InitVal	Description
31:0	Various	RW RO 0x144	Same as the DDR SDRAM Address/Control Pads Calibration Register register.

Table 138: DDR2 SDRAM ODT Control (Low) Register
Offset:0x01494

Bits	Field	Type/InitVal	Description
3:0	ODT0Rd	RW 0x0	M_ODT[0] control for read transactions Bit[0] = if set to 1, M_ODT[0] is asserted during read from DDR SDRAM bank 0. Bit[1] = if set to 1, M_ODT[0] is asserted during read from DDR SDRAM bank 1. Bit[2] = if set to 1, M_ODT[0] is asserted during read from DDR SDRAM bank 2. Bit[3] = if set to 1, M_ODT[0] is asserted during read from DDR SDRAM bank 3. Refer to the *Design Considerations* for this product.
7:4	ODT1Rd	RW 0x0	M_ODT[1] control for read transactions Same as <ODT0Rd>.
11:8	ODT2Rd	RW 0x0	M_ODT[2] control for read transactions Same as <ODT0Rd>.
15:12	ODT3Rd	RW 0x0	M_ODT[3] control for read transactions Same as <ODT0Rd>.
19:16	ODT0Wr	RW 0x0	M_ODT[0] control for write transactions Bit[0] = if set to 1, M_ODT[0] is asserted during write to DDR SDRAM bank 0. Bit[1] = if set to 1, M_ODT[0] is asserted during write to DDR SDRAM bank 1. Bit[2] = if set to 1, M_ODT[0] is asserted during write to DDR SDRAM bank 2. Bit[3] = if set to 1, M_ODT[0] is asserted during write to DDR SDRAM bank 3. Refer to the *Design Considerations* for this product.
23:20	ODT1Wr	RW 0x0	M_ODT[1] control for write transactions Same as <ODT0Wr>.
27:24	ODT2Wr	RW 0x0	M_ODT[2] control for write transactions Same as <ODT0Wr>.
31:28	ODT3Wr	RW 0x0	M_ODT[3] control for write transactions Same as <ODT0Wr>.

Table 139: DDR2 SDRAM ODT Control (High) Register
Offset:0x01498

Bits	Field	Type/InitVal	Description
1:0	ODT0En	RW 0x0	M_ODT[0] Enable 0x0, 0x2 = M_ODT[0] assertion/de-assertion is controlled by DDR2 SDRAM ODT Control Low register 0x1 = M_ODT[0] is never active 0x3 = M_ODT[0] is always active Refer to the *Design Considerations* for this product.
3:2	ODT1En	RW 0x0	M_ODT[1] Enable Same as <ODT0En>.
5:4	ODT2En	RW 0x0	M_ODT[2] Enable Same as <ODT0En>.

Table 139: DDR2 SDRAM ODT Control (High) Register (Continued)
Offset:0x01498

Bits	Field	Type/InitVal	Description
7:6	ODT3En	RW 0x0	M_ODT[3] Enable Same as <ODT0En>.
31:8	Reserved	RO 0x0	Reserved

Table 140: DDR2 SDRAM ODT Control Register
Offset:0x0149C

Bits	Field	Type/InitVal	Description
3:0	ODTRd	RW 0x0	DDR Controller I/O buffer ODT control for read transactions Bit[0] = if set to 1, internal ODT is asserted during read from DDR SDRAM bank 0. Bit[1] = if set to 1, internal ODT is asserted during read from DDR SDRAM bank 1. Bit[2] = if set to 1, internal ODT is asserted during read from DDR SDRAM bank 2. Bit[3] = if set to 1, internal ODT is asserted during read from DDR SDRAM bank 3. Refer to the *Design Considerations* for this product.
7:4	ODTWr	RW 0x0	DDR Controller I/O buffer ODT control for write transactions Bit[0] = if set to 1, internal ODT is asserted during write to DDR SDRAM bank 0. Bit[1] = if set to 1, internal ODT is asserted during write to DDR SDRAM bank 1. Bit[2] = if set to 1, internal ODT is asserted during write to DDR SDRAM bank 2. Bit[3] = if set to 1, internal ODT is asserted during write to DDR SDRAM bank 3. Refer to the *Design Considerations* for this product.
9:8	ODTEn	RW 0x0	DDR Controller I/O buffer ODT Enable 0x0, 0x2 = internal ODT assertion/de-assertion is controlled by <ODTRd>/<ODTWr> fields 0x1 = Internal ODT is never active. 0x3 = Internal ODT is always active. Refer to the *Design Considerations* for this product.
11:10	ODTSel4DqDqsDm	RW 0x0	DDR Controller I/O Buffer ODT Select for DQ, DQS, and DM 0x0 = Turned off 0x1 = 150 ohm 0x2 = 75 ohm 0x3 = 50 ohm
31:12	Reserved	RO 0x0	Reserved

Table 141: DDR SDRAM Interface Mbus Control (Low) Register
Offset:0x01430

Bits	Field	Type/InitVal	Description
3:0	Arb0	RW 0x0	Slice 0 of device controller Mbus SDRAM arbiter

Table 141: DDR SDRAM Interface Mbus Control (Low) Register (Continued)
Offset:0x01430

Bits	Field	Type/InitVal	Description
7:4	Arb1	RW 0x1	Slice 1 of device controller Mbus SDRAM arbiter
11:8	Arb2	RW 0x2	Slice 2 of device controller Mbus SDRAM arbiter
15:12	Arb3	RW 0x3	Slice 3 of device controller Mbus SDRAM arbiter
19:16	Arb4	RW 0x4	Slice 4 of device controller Mbus SDRAM arbiter
23:20	Arb5	RW 0x5	Slice 5 of device controller Mbus SDRAM arbiter
27:24	Arb6	RW 0x6	Slice 6 of device controller Mbus SDRAM arbiter
31:28	Arb7	RW 0x7	Slice 7 of device controller Mbus SDRAM arbiter

Table 142: DDR SDRAM Interface Mbus Control (High) Register
Offset:0x01434

Bits	Field	Type/InitVal	Description
3:0	Arb8	RW 0x8	Slice 8 of device controller Mbus SDRAM arbiter
7:4	Arb9	RW 0x9	Slice 9 of device controller Mbus SDRAM arbiter
11:8	Arb10	RW 0xA	Slice 10 of device controller Mbus SDRAM arbiter
15:12	Arb11	RW 0xB	Slice 11 of device controller Mbus SDRAM arbiter
19:16	Arb12	RW 0xC	Slice 12 of device controller Mbus SDRAM arbiter
23:20	Arb13	RW 0xD	Slice 13 of device controller Mbus SDRAM arbiter
27:24	Arb14	RW 0xE	Slice 14 of device controller Mbus SDRAM arbiter
31:28	Arb15	RW 0xF	Slice 15 of device controller Mbus SDRAM arbiter

Table 143: DDR SDRAM Interface Mbus Timeout Register
Offset:0x01438

Bits	Field	Type/InitVal	Description
7:0	Timeout	RW 0xFF	Mbus SDRAM Arbiter Timeout Preset Value
15:8	Reserved	RO 0x0	Reserved
16	TimeoutEn	RW 0x1	Mbus SDRAM Arbiter Timer Enable 0 = Enable 1 = Disable
31:17	Reserved	RO 0x0	Reserved

Table 144: DDR SDRAM MMask Register
Offset:0x014B0

Bits	Field	Type/InitVal	Description
1:0	Reserved	RES	Reserved
3:2	ODTSel4Start Burst	RW 0x0o	DDR Controller I/O Buffer ODT Select for StartBurst 0 = Turned off 1 = 150 ohm 2 = 75 ohm 3 = 50 ohm
4	ODTEn4Start Burst	RW 0x0	DDR Controller I/O Buffer ODT Enable 0 = ODT disable 1 = ODT enable
8:5	DDR2PADComp PowerDown	0xF	0 = Power down of DDR1/DDR2 PAD receiver When the receiver is powered down, a simple CMOS receiver is activated. **NOTE:** This mode is only valid when working with a 16-bit DDR interface.
31:9	Reserved	RES	Reserved

A.6 PCI Express Interface Registers

Table 145: PCI Express Register Map Table

Register Name	Offset	Table & Page
PCI Express BAR Control Registers		
PCI Express BAR1 Control Register	0x41804	Table 146, p. 279
PCI Express BAR2 Control Register	0x41808	Table 147, p. 279
PCI Express Expansion ROM BAR Control Register	0x4180C	Table 148, p. 279
PCI Express Configuration Requests Generation Registers		
PCI Express Configuration Address Register	0x418F8	Table 149, p. 280
PCI Express Configuration Data Register	0x418FC	Table 150, p. 280
PCI Express Interrupt Registers		
PCI Express Interrupt Cause	0x41900	Table 151, p. 280
PCI Express Interrupt Mask	0x41910	Table 152, p. 283
PCI Express Address Window Control Registers		
PCI Express Window0 Control Register	0x41820	Table 153, p. 283
PCI Express Window0 Base Register	0x41824	Table 154, p. 284
PCI Express Window0 Remap Register	0x4182C	Table 155, p. 284
PCI Express Window1 Control Register	0x41830	Table 156, p. 285
PCI Express Window1 Base Register	0x41834	Table 157, p. 285
PCI Express Window1 Remap Register	0x4183C	Table 158, p. 285
PCI Express Window2 Control Register	0x41840	Table 159, p. 286
PCI Express Window2 Base Register	0x41844	Table 160, p. 286
PCI Express Window2 Remap Register	0x4184C	Table 161, p. 286
PCI Express Window3 Control Register	0x41850	Table 162, p. 287
PCI Express Window3 Base Register	0x41854	Table 163, p. 287
PCI Express Window3 Remap Register	0x4185C	Table 164, p. 287
PCI Express Window4 Control Register	0x41860	Table 165, p. 288
PCI Express Window4 Base Register	0x41864	Table 166, p. 288
PCI Express Window4 Remap Register	0x4186C	Table 167, p. 288
PCI Express Window5 Control Register	0x41880	Table 168, p. 289
PCI Express Window5 Base Register	0x41884	Table 169, p. 289
PCI Express Window5 Remap Register	0x4188C	Table 170, p. 289
PCI Express Default Window Control Register	0x418B0	Table 171, p. 290
PCI Express Expansion ROM Window Control Register	0x418C0	Table 172, p. 290
PCI Express Expansion ROM Window Remap Register	0x418C4	Table 173, p. 290
PCI Express Control and Status Registers		
PCI Express Control Register	0x41A00	Table 174, p. 291
PCI Express Status Register	0x41A04	Table 175, p. 292
PCI Express Completion Timeout Register	0x41A10	Table 176, p. 293
PCI Express Flow Control Register	0x41A20	Table 177, p. 293

Table 145: PCI Express Register Map Table (Continued)

Register Name	Offset	Table & Page
PCI Express Acknowledge Timers (1X) Register	0x41A40	Table 178, p. 294
PCI Express Debug Control Register	0x41A60	Table 179, p. 294
PCI Express Configuration Header Registers		
PCI Express TL Control Register	0x41AB0	Table 180, p. 295
PCI Express Device and Vendor ID Register	0x40000	Table 181, p. 295
PCI Express Command and Status Register	0x40004	Table 182, p. 295
PCI Express Class Code and Revision ID Register	0x40008	Table 183, p. 298
PCI Express BIST, Header Type and Cache Line Size Register	0x4000C	Table 184, p. 298
PCI Express BAR0 Internal Register	0x40010	Table 185, p. 299
PCI Express BAR0 Internal (High) Register	0x40014	Table 186, p. 299
PCI Express BAR1 Register	0x40018	Table 187, p. 299
PCI Express BAR1 (High) Register	0x4001C	Table 188, p. 300
PCI Express BAR2 Register	0x40020	Table 189, p. 300
PCI Express BAR2 (High) Register	0x40024	Table 190, p. 300
PCI Express Subsystem Device and Vendor ID	0x4002C	Table 191, p. 301
PCI Express Expansion ROM BAR Register	0x40030	Table 192, p. 301
PCI Express Capability List Pointer Register	0x40034	Table 193, p. 302
PCI Express Interrupt Pin and Line Register	0x4003C	Table 194, p. 302
PCI Express Power Management Capability Header Register	0x40040	Table 195, p. 302
PCI Express Power Management Control and Status Register	0x40044	Table 196, p. 303
PCI Express MSI Message Control Register	0x40050	Table 197, p. 304
PCI Express MSI Message Address Register	0x40054	Table 198, p. 304
PCI Express MSI Message Address (High) Register	0x40058	Table 199, p. 305
PCI Express MSI Message Data Register	0x4005C	Table 200, p. 305
PCI Express Capability Register	0x40060	Table 201, p. 305
PCI Express Device Capabilities Register	0x40064	Table 202, p. 306
PCI Express Device Control Status Register	0x40068	Table 203, p. 307
PCI Express Link Capabilities Register	0x4006C	Table 204, p. 309
PCI Express Link Control Status Register	0x40070	Table 205, p. 310
PCI Express Advanced Error Report Header Register	0x40100	Table 206, p. 312
PCI Express Uncorrectable Error Status Register	0x40104	Table 207, p. 312
PCI Express Uncorrectable Error Mask Register	0x40108	Table 208, p. 313
PCI Express Uncorrectable Error Severity Register	0x4010C	Table 209, p. 313
PCI Express Correctable Error Status Register	0x40110	Table 210, p. 314
PCI Express Correctable Error Mask Register	0x40114	Table 211, p. 314
PCI Express Advanced Error Capability and Control Register	0x40118	Table 212, p. 315
PCI Express Header Log First DWORD Register	0x4011C	Table 213, p. 315
PCI Express Header Log Second DWORD Register	0x40120	Table 214, p. 316
PCI Express Header Log Third DWORD Register	0x40124	Table 215, p. 316
PCI Express Header Log Fourth DWORD Register	0x40128	Table 216, p. 316

A.6.1 PCI Express BAR Control Registers

Table 146: PCI Express BAR1 Control Register
Offset:0x41804

Bits	Field	Type/InitVal	Description
0	Bar1En	RW 0x1	BAR1 Enable
15:1	Reserved	RSVD 0x0	Reserved
31:16	Bar1Size	RW 0x3FFF	BAR1 Size BAR sizes range from 64 KB to 4 GB in powers of 2. default value: 03FFF = 1 GB

Table 147: PCI Express BAR2 Control Register
Offset:0x41808

Bits	Field	Type/InitVal	Description
0	Bar2En	RW 0x1	BAR2 Enable
15:1	Reserved	RSVD 0x0	Reserved
31:16	Bar2Size	RW 0x0FFF	BAR2 Size BAR sizes range from 64 KB to 4 GB in powers of 2. default value: 0x0FFF = 256 MB

Table 148: PCI Express Expansion ROM BAR Control Register
Offset:0x4180C

Bits	Field	Type/InitVal	Description
0	ExpROMEn	RW 0x0	Expansion ROM BAR Enable
18:1	Reserved	RSVD 0x0	Reserved
21:19	ExpROMSz	RW 0x0	Expansion ROM Address Bank Size 0 = 512: 512 KB 1 = 1024:1024 KB 2 = 2048: 2048 KB 3 = 4093: 4096 KB Only 2 bits are strapped in Her-II (GPIO[4:3]). Runit maps them to the 3-bit value. Default value (without external pull-up/down) = 0x0 (3'b000)
31:22	Reserved	RSVD 0x0	s

A.6.2 PCI Express Configuration Cycles Generation Registers

Table 149: PCI Express Configuration Address Register
Offset:0x418F8

NOTE:

Bits	Field	Type/InitVal	Description
1:0	Reserved	RSVD 0x0	Reserved
7:2	RegNum	RW 0x0	Register Number
10:8	FunctNum	RW 0x0	Function Number
15:11	DevNum	RW 0x0	Device Number
23:16	BusNum	RW 0x0	Bus Number
27:24	ExtRegNum	RW 0x0	Extended Register Number
30:28	Reserved	RSVD 0x0	Reserved
31	ConfigEn	RW 0x0	Configuration Enable Bit

Table 150: PCI Express Configuration Data Register
Offset:0x418FC

NOTE:

Bits	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.6.3 PCI Express Interrupt Registers

Table 151: PCI Express Interrupt Cause
Offset:0x41900

NOTE: All bits except bits[27:24] are Read/Write Clear 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.

Bits	Field	Type/InitVal	Description
0	Reserved	RSVD 0x0	Reserved

Table 151: PCI Express Interrupt Cause (Continued)
Offset:0x41900

NOTE: All bits except bits[27:24] are Read/Write Clear 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.

Bits	Field	Type/InitVal	Description
1	MDis	RW0C 0x0	Attempt to generate a PCI transaction while the master is disabled.
2	Reserved	RW0C 0x0	Reserved
3	ErrWrToReg	RW0C 0x0	Erroneous write attempt to PCI Express 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).
4	HitDfltWinErr	RW0C 0x0	Hit Default Window Error
7:5	Reserved	RSVD 0x0	Reserved
8	CorErrDet	RW0C 0x0	Correctable Error Detected Indicates status of correctable errors detected by 88F5182.
9	NFErrDet	RW0C 0x0	Non-Fatal Error Detected Indicates status of Non-Fatal errors detected by 88F5182.
10	FErrDet	RW0C 0x0	Fatal Error Detected Indicates status of Fatal errors detected by 88F5182.
11	DstateChange	RW0C 0x0	Dstate Change Indication Any change in the Dstate asserts this bit. **NOTE:** This bit is relevant only for Endpoint.
12	BIST	RW0C 0x0	PCI Express BIST activated BIST is not supported.
15:13	Reserved	RW0C 0x0	Reserved
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.
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.
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.

Table 151: PCI Express Interrupt Cause (Continued)
Offset:0x41900

NOTE: All bits except bits[27:24] are Read/Write Clear 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.

Bits	Field	Type/InitVal	Description
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.
20	PexSlvHot 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.
21	PexSlvDisLink	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.
22	PexSlvLb	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.
23	PexLinkFail	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.
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 some 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 some other bits in this register, since it is cleared by the interrupting device downstream. Relevant for Root Complex only.

Table 151: PCI Express Interrupt Cause (Continued)
Offset:0x41900

NOTE: All bits except bits[27:24] are Read/Write Clear 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.

Bits	Field	Type/InitVal	Description
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 some other bits in this register, since it is cleared by the interrupting device downstream. Relevant for Root Complex only.
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 some others bit in this register, since it is cleared by the interrupting device downstream. Relevant for Root Complex only.
31:28	Reserved	RSVD 0x0	s

Table 152: PCI Express Interrupt Mask
Offset:0x41910

Bits	Field	Type/InitVal	Description
31:0	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. **NOTE:** Bits [23:20] are sticky bits—not initialized by reset.

A.6.4 PCI Express Address Window Control Registers

Table 153: PCI Express Window0 Control Register
Offset:0x41820

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window0 Enable 0 = Disabled: Window is disabled. 1 = Enabled: Window is enabled.
1	BarMap	RW 0x0	Mapping to BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.

Table 153: PCI Express Window0 Control Register (Continued)
Offset:0x41820

Bits	Field	Type/InitVal	Description
3:2	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See Section 2.2, PCI Express Address Map, on page 20
15:8	Attr	RW 0x0E	Target specific attributes depending on the target interface. See Section 2.2, PCI Express Address Map, on page 20
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 KB granularity. (A value of 0x0FFF specifies 256 MB).

Table 154: PCI Express Window0 Base Register
Offset:0x41824

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
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].

Table 155: PCI Express Window0 Remap Register
Offset:0x4182C

Bits	Field	Type/InitVal	Description
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disabled: Remap disabled. 1 = Enabled: Remap enabled.
15:1	Reserved	RSVD 0x0	Reserved
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.

Table 156: PCI Express Window1 Control Register
Offset:0x41830

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window Enable.
1	BarMap	RW 0x0	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
3:2	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window:
15:8	Attr	RW 0x0D	Target specific attributes depending on the target interface.
31:16	Size	RW 0x0FFF	Window Size (A value of 0x0FFF specifies 256 MB).

Table 157: PCI Express Window1 Base Register
Offset:0x41834

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0x1000	Base Address

Table 158: PCI Express Window1 Remap Register
Offset:0x4183C

Bits	Field	Type/InitVal	Description
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disabled: Remap disabled. 1 = Enabled: Remap enabled.
15:1	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0x0	Remap Address

Table 159: PCI Express Window2 Control Register
Offset:0x41840

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x0	Window Enable
1	BarMap	RW 0x0	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
3:2	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window.
15:8	Attr	RW 0x0B	Target specific attributes depending on the target interface.
31:16	Size	RW 0x0FFF	Window Size

Table 160: PCI Express Window2 Base Register
Offset:0x41844

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0x2000	Base Address

Table 161: PCI Express Window2 Remap Register
Offset:0x4184C

Bits	Field	Type/InitVal	Description
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disabled: Remap disabled. 1 = Enabled: Remap enabled.
15:1	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0x0	Remap Address

Table 162: PCI Express Window3 Control Register
Offset:0x41850

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x0	Window Enable. Window is disabled by default.
1	BarMap	RW 0x0	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
3:2	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window.
15:8	Attr	RW 0x07	Target specific attributes depending on the target interface.
31:16	Size	RW 0x0FFF	Window Size

Table 163: PCI Express Window3 Base Register
Offset:0x41854

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0x3000	Base Address

Table 164: PCI Express Window3 Remap Register
Offset:0x4185C

Bits	Field	Type/InitVal	Description
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disabled: Remap disabled. 1 = Enabled: Remap enabled.
15:1	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0x0	Remap Address

Table 165: PCI Express Window4 Control Register
Offset:0x41860

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window Enable
1	BarMap	RW 0x1	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
3:2	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window.
15:8	Attr	RW 0x1B	Target specific attributes depending on the target interface.
31:16	Size	RW 0x07FF	Window Size (A value of 0x07FF specifies 128 MB).

Table 166: PCI Express Window4 Base Register
Offset:0x41864

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0xF000	Base Address

Table 167: PCI Express Window4 Remap Register
Offset:0x4186C

Bits	Field	Type/InitVal	Description
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disabled: Remap disabled. 1 = Enabled: Remap enabled.
15:1	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0x0	Remap Address

Table 168: PCI Express Window5 Control Register
Offset:0x41880

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window Enable
1	BarMap	RW 0x1	Mapping To BAR 0 = BAR1: Window is mapped to BAR1. 1 = BAR2: Window is mapped to BAR2.
3:2	Reserved	RSVD 0x0	s
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window.
15:8	Attr	RW 0x0F	Target specific attributes depending on the target interface.
31:16	Size	RW 0x07FF	Window Size (A value of 0x07FF specifies 128 MB).

Table 169: PCI Express Window5 Base Register
Offset:0x41884

Bits	Field	Type/InitVal	Description
15:0	Reserved	RSVD 0x0	Reserved
31:16	Base	RW 0xF800	Base Address

Table 170: PCI Express Window5 Remap Register
Offset:0x4188C

Bits	Field	Type/InitVal	Description
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disabled: Remap disabled. 1 = Enabled: Remap enabled.
15:1	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0x0	Remap Address

Table 171: PCI Express Default Window Control Register
Offset:0x418B0

Bits	Field	Type/InitVal	Description
3:0	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window:
15:8	Attr	RW 0x0	Target specific attributes depending on the target interface.
31:16	Reserved	RSVD 0x0	Reserved

Table 172: PCI Express Expansion ROM Window Control Register
Offset:0x418C0

Bits	Field	Type/InitVal	Description
3:0	Reserved	RSVD 0x0	Reserved
7:4	Target	RW 0x1	Specifies the unit ID (target interface) associated with this window.
15:8	Attr	RW 0x0F	Target specific attributes depending on the target interface.
31:16	Reserved	RSVD 0x0	Reserved

Table 173: PCI Express Expansion ROM Window Remap Register
Offset:0x418C4

Bits	Field	Type/InitVal	Description
0	RemapEn	RW 0x0	Remap Enable Bit 0 = Disabled: Remap disabled. 1 = Enabled: Remap enabled.
15:1	Reserved	RSVD 0x0	Reserved
31:16	Remap	RW 0x0	Remap Address

A.6.5 PCI Express Control and Status Registers

Table 174: PCI Express Control Register
Offset:0x41A00

Bits	Field	Type/InitVal	Description
0	Reserved	RSVD 0x1	Reserved Must write 1.
1	ConfRoot Complex	RO 0x0	PCI Express Device Type Control 0 = Endpoint: Endpoint device (downstream device / upstream link). 1 = Root: Root Complex device. (upstream device / downstream link). **NOTE:** Cannot be configured during operation, must be configured only by reset strapping or from TWSI during auto_loader.
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. 0x0 = Disabled: Mapping disabled. 0x1 = Enabled: Mapping enabled.
7:3	Reserved	RSVD 0x1	Reserved
9:8	Reserved	RSVD 0x3	Reserved Must write 0x3.
15:10	Reserved	RSVD 0x0	Reserved
23:16	Reserved	RSVD 0x14	Reserved Must write 0x14.
24	ConfMstr HotReset	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. **NOTE:** This bit is used only when working in Root Complex mode.
25	Reserved	RSVD 0x0	Reserved
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.

Table 174: PCI Express Control Register (Continued)
Offset:0x41A00

Bits	Field	Type/InitVal	Description
27	ConfMstrDisScrmb	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 is programmed before the start of link initialization.
28	Reserved	RSVD 0x0	Reserved Must write 0.
31:29	Reserved	RSVD 0x0	Reserved

Table 175: PCI Express Status Register
Offset:0x41A04

Bits	Field	Type/InitVal	Description
0	DLDown	RO	DL_Down indication 0 = Active: DL is up 1 = Inactive: DL is down
4:1	Reserved	RSVD 0x0	Reserved
7:5	Reserved	RSVD 0x0	Reserved
15:8	PexBusNum	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.
20:16	PexDevNum	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.
23:21	Reserved	RSVD 0x0	Reserved
24	PexSlvHotReset	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 hot-reset indication is received.
25	PexSlvDisLink	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 a link disabled indication is received.
26	PexSlvLb	RO 0x0	Slave Loopback Indication This field sets when the opposite device on the PCI Express port acts as a loopback master, and a loopback indication is received.

Table 175: PCI Express Status Register (Continued)
Offset:0x41A04

Bits	Field	Type/InitVal	Description
27	PexSlvDis 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 a scrambling disabled indication is received.
31:28	Reserved	RSVD 0x0	Reserved

Table 176: PCI Express Completion Timeout Register
Offset:0x41A10

Bits	Field	Type/InitVal	Description
15:0	ConfCmpToThrshld	RW 0x2710	Completion Timeout Threshold This field controls the size of the completion timeout interval. The NP request is cleared from MCT within -0% +100% of the timeout value. **NOTE:** Timescale: 256*symbol_time = 1 µs Initial Value (0x2710) represents a 10 ms value (10,000 decimal) 0x0 = Disabled. No timeout mechanism on N.P TLPs. Minimum Value: 40 (40 µs) Maximum Value: 25K (25 ms)
31:16	Reserved	RSVD 0x0	Reserved

Table 177: PCI Express Flow Control Register
Offset:0x41A20

Bits	Field	Type/InitVal	Description
7:0	ConfPhInitFc	RW 0x0	Posted Headers Flow Control Credit Initial Value
15:8	ConfNphInitFc	RW 0x8	Non-Posted Headers Flow Control Credit Initial Value
23:16	ConfChInitFc	RW 0x0	Completion Headers Flow Control Credit Initial Value Infinite.

Table 177: PCI Express Flow Control Register (Continued)
Offset:0x41A20

Bits	Field	Type/InitVal	Description
31:24	ConfFc UpdateTo	RW 0x78	Flow Control Update Timeout This field controls the Flow Control update interval period. **NOTE:** Timescale: 64*symbol_time = 256 ns 0x0 = Disabled. No timeout mechanism on update FC. Minimum Value: 120 (30 µs) Maximum Value: 180 (45 µs) When extended sync is enabled the check threshold is configured to 120 µs. 120 = 30 µs **NOTE:**

Table 178: PCI Express Acknowledge Timers (1X) Register
Offset:0x41A40

Bits	Field	Type/InitVal	Description
15:0	AckLatTOX1	RW 0x4	Acknowledge Latency Timer Timeout Value for 1X Link Used when the PHY link width Auto-Negotiation result is 1X. **NOTE:** Timescale: symbol_time = 4 ns Minimum Value: 4 (4 symbol times) Maximum Value: 237 (237symbol times
31:16	AckRplyTOX1	RW 0x320	Acknowledge Replay Timer Timeout Value for 1X **NOTE:** Timescale: symbol_time = 4 ns Initial value (0x320) represents a 800 symbol times timeout. Minimum Value: 711 (711 symbol times) Maximum Value: 64K-1 (64K-1 symbol times)

Table 179: PCI Express Debug Control Register
Offset:0x41A60

Bits	Field	Type/InitVal	Description
15:0	Reserved	RES 0x0	Reserved
16	ConfMskLinkFail	RW 0x1	Mask Link fail from PEX reset output. **NOTE:** Sticky bit — not initialized by reset.
17	ConfMskHotReset	RW 0x1	Mask Hot Reset from PEX reset output. **NOTE:** Sticky bit — not initialized by reset.
18	Reserved	RES 0x0	Reserved
19	ConfDisLinkFailRegRst	RW 0x1	Disable PEX Register File reset upon link failure. **NOTE:** Sticky bit — not initialized by reset.

Table 179: PCI Express Debug Control Register (Continued)
Offset:0x41A60

Bits	Field	Type/InitVal	Description
31:20	Reserved	RES 0x0	Reserved

Table 180: PCI Express TL Control Register
Offset:0x41AB0

Bits	Field	Type/InitVal	Description
31:0	Reserved	RSVD 0x0	Reserved

A.6.6 PCI Express Configuration Header Registers

Table 181: PCI Express Device and Vendor ID Register
Offset:0x40000
Configuration: 0x0

Bits	Field	Type/InitVal	Description
15:0	VenID	RO 0x11AB	Vendor ID This field identifies Marvell® as the Vendor of the device.
31:16	DevID	RO	Device ID
31:16	DevID	RO 0x5182	Device ID

Table 182: PCI Express Command and Status Register
Offset:0x40004
Configuration: 0x4

Bits	Field	Type/InitVal	Description
0	IOEn	RW 0x0	I/O Space Enable. The I/O space is not enabled; therefore, this bit is not functional.
1	MemEn	RW 0x0	Memory Space Enable Controls 88F5182 response to PCI Express memory accesses. 0 = Disable: All Memory accesses from PCI Express are completed as Unsupported Requests. 1 = Enable:

Table 182: PCI Express Command and Status Register (Continued)
Offset:0x40004
Configuration: 0x4

Bits	Field	Type/InitVal	Description
2	MasEn	RW 0x0	Master Enable This bit controls the ability of the 88F5182 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. 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. **NOTE:** Message Signal Interrupts (MSI) are memory write requests, and as such are disabled in accordance to this bit. Messages and completions are transmitted to the PCI Express regardless of the setting of this bit.
5:3	Reserved	RSVD 0x0	Does not apply to PCI Express devices. This bit is hardwired to 0.
6	PErrEn	RW 0x0	Parity Error Enable This bit controls the ability of the 88F5182 to respond to poisoned data errors as a requestor (master) on the PCI Express port. Controls MasDataPerr status bit assertion in PCMDSTT register. 0 = Disabled: MasDataPerr assertion is disabled. 1 = Enabled: MasDataPerr assertion is enabled. **NOTE:** The setting of this bit does not affect the DetectedPErr status bit.
7	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 88F5182 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. 0 = Disabled: SSysErr assertion is disabled. 1 = Enabled: SSysErr assertion is enabled. In addition uncorrectable error messages generation is enabled. **NOTE:** PCI Express uncorrectable error messages are reported if enabled either through this bit or through bits NFErrRepEn or FErrRepEn in Table 203, "PCI Express Device Control Status Register".
9	Reserved	RSVD 0x0	Does not apply to PCI Express. This bit is hardwired to 0.
10	IntDis	RW 0x0	Interrupt Disable This bit controls the ability of the 88F5182 to generate interrupt emulation messages. When set, interrupt messages are not generated. 0 = Enabled: Interrupt messages enabled. 1 = Disabled: Interrupt messages disabled. Root Complex mode: this bit has no effect, received interrupt messages are still forwarded to the internal interface.
18:11	Reserved	RSVD 0x0	This bit is hardwired to 0.

Table 182: PCI Express Command and Status Register (Continued)
Offset:0x40004
Configuration: 0x4

Bits	Field	Type/InitVal	Description
19	IntStat	RO 0x0	Interrupt Status When set, this bit indicates that an interrupt message is pending internally in the device. 0 = Disabled: No interrupt asserted. 1 = Enabled: Interrupt asserted.
20	CapList	RO 0x1	Capability List Support This bit indicates that the 88F5182 configuration header includes capability list. This bit is hardwired to 1, since this is always supported in PCI Express.
23:21	Reserved	RSVD 0x0	This bit is hardwired to 0.
24	MasDataPerr	SC 0x0	Master Data Parity Error Set by the 88F5182 when poisoned data is detected as a requestor (master). Set when PErrEn bit[6] is set and either: • Poisoned completion is received for the PCIe port or • Poisoned TLP is transmitted to the PCIe port. Write 1 to clear.
26:25	Reserved	RSVD 0x0	Does not apply to PCI Express. These bits are hardwired to 0.
27	STarAbort	SC 0x0	Signaled Target Abort This bit is set when the 88F5182, as a completer (target), completes a transaction as a Completer Abort. Write 1 to clear.
28	RTAbort	SC 0x0	Received Target Abort This bit is set when the 88F5182, as a requester (master), receives a completion with the status Completer Abort. Write 1 to clear.
29	RMAbort	SC 0x0	Received Master Abort. This bit is set when the 88F5182, as a requester (master), receives a completion with the status Unsupported Request. Write 1 to clear.
30	SSysErr	SC 0x0	Signalled System Error This bit is set when the 88F5182 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.
31	DetParErr	SC 0x0	Detected Parity Error This bit is set when the 88F5182 receives a poisoned TLP. **NOTE:** The bit is set regardless of the state of the <PErrEn> bit in this register. Write 1 to clear.

Table 183: PCI Express Class Code and Revision ID Register
Offset:0x40008
Configuration: 0x8

Bits	Field	Type/InitVal	Description
7:0	RevID	RO 0x0	88F5182 Revision Number
15:8	ProgIF	RO 0x0	Register Level Programming Interface init value - 0 for Her-II-E
23:16	SubClass	RO 0x80	88F5182 Sub class—Other Memory Controller
31:24	BaseClass	RO 0x05	88F5182 Base Class—Memory Controller

Table 184: PCI Express BIST, Header Type and Cache Line Size Register
Offset:0x4000C
Configuration: 0xC

Bits	Field	Type/InitVal	Description
7:0	CacheLine	RW 0x00	88F5182 Cache Line Size **NOTE:**
15:8	Reserved MasLatTimer	RSVD 0x0	Does not apply to PCI Express. This bit is hardwired to 0.
23:16	HeadType	RO 0x0	88F5182 Configuration Header Type Type 0 single-function configuration header.
27:24	BISTComp	RO 0x0	BIST Completion Code 0x0 = BIST passed. Other = BIST failed.
29:28	Reserved	RSVD 0x0	Reserved
30	BISTAct	RO 0x0	BIST Activate bit BIST is not supported.
31	BISTCap	RO 0x0	BIST Capable Bit BIST is not supported. This bit must be set to 0.

Table 185: PCI Express BAR0 Internal Register
Offset:0x40010
Configuration: 0x10

Bits	Field	Type/InitVal	Description
0	Space	RO 0x0	Memory Space Indicator.
2:1	Type	RO 0x2	BAR Type. BAR can be located in 64-bit memory address space.
3	Prefetch	RO 0x1	Prefetch Enable Indicates that pre-fetching is enabled.
19:4	Reserved	RSVD 0x0	Reserved
31:20	Base	RW 0xD00	Internal register memory Base Address Indicates a 1 MB address space. Corresponds to address bits[31:20].

Table 186: PCI Express BAR0 Internal (High) Register
Offset:0x40014
Configuration: 0x14

Bits	Field	Type/InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits[63:32].

Table 187: PCI Express BAR1 Register
Offset:0x40018
Configuration: 0x18

Bits	Field	Type/InitVal	Description
0	Space	RO 0x0	Memory Space Indicator
2:1	Type	RO 0x2	BAR Type BAR can be located in 64-bit memory address space.
3	Prefetch	RO 0x1	Prefetch Enable Indicates that pre-fetching is enabled.
15:4	Reserved	RSVD 0x0	Reserved

Table 187: PCI Express BAR1 Register (Continued)
Offset:0x40018
Configuration: 0x18

Bits	Field	Type/InitVal	Description
31:16	Base/Reserved	RSVD 0x0000	Base address Defined according to Table 146, “PCI Express BAR1 Control Register”. Indicates a 64-KB up to 4-GB address space. Corresponds to address bits[31:16].

Table 188: PCI Express BAR1 (High) Register
Offset:0x4001C
Configuration: 0x1C

Bits	Field	Type/InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits[63:32].

Table 189: PCI Express BAR2 Register
Offset:0x40020
Configuration: 0x20

Bits	Field	Type/InitVal	Description
0	Space	RO 0x0	Memory Space Indicator
2:1	Type	RO 0x2	BAR Type BAR can be located in 64-bit memory address space.
3	Prefetch	RO 0x1	Prefetch Enable Indicates that pre-fetching is enabled.
15:4	Reserved	RSVD 0x0	Reserved
31:16	Base/Reserved	RSVD 0xF000	Base address Defined according to Table 147, “PCI Express BAR2 Control Register”. Indicates 64-KB up to 4-GB address space. Corresponds to address bits[31:16].

Table 190: PCI Express BAR2 (High) Register
Offset:0x40024
Configuration: 0x24

Bits	Field	Type/InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits[63:32].

Table 191: PCI Express Subsystem Device and Vendor ID
Offset:0x4002C
Configuration: 0x2C

Bits	Field	Type/InitVal	Description
15:0	SSVenID	RO 0x11AB	Subsystem Manufacturer ID Number The default is the Marvell® Vendor ID.
31:16	SSDevID	RO 0x11AB	Subsystem Device ID Number The default is the Marvell Vendor ID.

Table 192: PCI Express Expansion ROM BAR Register
Offset:0x40030
Configuration: 0x30

NOTE: If expansion ROM is not enabled via the ExpROMEn bit[0] in Table 148, “PCI Express Expansion ROM BAR Control Register”, this register is Reserved.

Bits	Field	Type/InitVal	Description
0	RomEn	RW 0x0	Expansion ROM Enable 0 = Disabled: 1 = Enabled: **NOTE:** 88F5182 expansion ROM address space is enabled only if both this bit (RomEn) and <MemEn> in the Table 182, “PCI Express Command and Status Register” are set.
18:1	Reserved	RSVD 0x0	Reserved
21:19	RomBase/ Reserved	RSVD 0x0	These bits are defined according to field <ExpROMSz> in the PCI Express Expansion ROM BAR Control Register (Table 148 p. 279). When ROM Size is: 0 = 512KBSpace: Bits[21:19] are used as Expansion ROM Base Address[21:19]. 1 = 1MBSpace: Bits[21:20] are used as Expansion ROM Base Address[21:20], and bit[19] is reserved. 2 = 2MBSpace: Bits[21] used as Expansion ROM Base Address[21], and bits[20:19] are reserved. 3 = 4MBSpace: Bits[21:19] are reserved. .
31:22	RomBase	RW 0x0	Expansion ROM Base Address[31:22]

Table 193: PCI Express Capability List Pointer Register
Offset:0x40034
Configuration: 0x34

Bits	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 Table 195, “PCI Express Power Management Capability Header Register” at offset 0x40.
31:8	Reserved	RSVD 0x0	Reserved

Table 194: PCI Express Interrupt Pin and Line Register
Offset:0x4003C
Configuration: 0x3C

Bits	Field	Type/InitVal	Description
7:0	IntLine	RW 0x00	Provides interrupt line routing information.
15:8	IntPin	RO 0x01	Indicates that 88F5182 is using INTA in the interrupt emulation messages.
31:16	Reserved	RSVD 0x0	Does not apply to PCI Express This field is hardwired to 0.

Table 195: PCI Express Power Management Capability Header Register
Offset:0x40040
Configuration: 0x40

Bits	Field	Type/InitVal	Description
7:0	CapID	RO 0x01	Capability ID Current value identifies the PCI Power Management capability.
15:8	NextPtr	RO 0x50	Next Item Pointer Current value points to MSI capability.
18:16	PMCVer	RO 0x2	PCI Power Management Capability Version.
20:19	Reserved	RSVD 0x0	Does not apply to PCI Express This field must be hardwired to 0.
21	DSI	RO 0x0	Device Specific Initialization 88F5182 does not requires device specific initialization.
24:22	AuxCur	RO 0x0	Auxiliary Current Requirements

Table 195: PCI Express Power Management Capability Header Register (Continued)
Offset:0x40040
Configuration: 0x40

Bits	Field	Type/InitVal	Description
25	D1Sup	RO 0x0	This bit indicates whether the device supports D1 Power Management state. The 88F5182 does not support D1 state.
26	D2Sup	RO 0x0	This bit indicates whether the device supports D2 Power Management state. The 88F5182 does not support D2 state.
31:27	PMESup	RO 0x00	This field indicates whether the device supports PM Event generation. The 88F5182 does not support the PMEn pin.

Table 196: PCI Express Power Management Control and Status Register
Offset:0x40044
Configuration: 0x44

Bits	Field	Type/InitVal	Description
1:0	PMState	RW 0x0	Power State This field controls the Power Management state of the 88F5182. The device supports all Power Management states. 0 = D0: 1 = D1: 2 = D2: 3 = D3: .
7:2	Reserved	RSVD 0x0	Reserved
8	PMEEn	RW 0x0	PM_PME Message Generation Enable 0 = Disabled 1 = Enabled **NOTE:** Sticky bit—not initialized by hot reset. **NOTE:**
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.
14:13	PMDataScale	RO 0x0	Data Scale Indicated 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.
15	PMEStat	RW 0x0	PME Status Write 1 to clear. **NOTE:** Sticky bit—not initialized by hot reset.
23:16	Reserved	RSVD 0x0	Does not apply to PCI Express. This field must be hardwired to 0.

Table 196: PCI Express Power Management Control and Status Register (Continued)
Offset:0x40044
Configuration: 0x44

Bits	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.

Table 197: PCI Express MSI Message Control Register
Offset:0x40050
Configuration: 0x50

Bits	Field	Type/InitVal	Description
7:0	CapID	RO 0x5	Capability ID The current value of this field identifies the PCI MSI capability.
15:8	NextPtr	RO 0x60	Next Item Pointer The current value of this field points to PCI Express capability.
16	MSIEn	RW 0x0	MSI Enable This bit controls the 88F5182 interrupt signaling mechanism. 0 = Disabled: Interrupts are signalled through the interrupt emulation messages. 1 = Enabled: Interrupts are signaled through MSI mechanism.
19:17	MultiCap	RO 0x0	Multiple Message Capable The 88F5182 is capable of driving a single message.
22:20	MultiEn	RW 0x0	Multiple Messages Enable The number of messages the system allocates to the 88F5182 (This number must be smaller or equal to what is in the <MultiCap> field).
23	Addr64	RO 0x1	64-bit Addressing Capable This field indicates whether the 88F5182 is capable of generating a 64-bit message address. 0 = Not Capable: 1 = Capable:
31:24	Reserved	RSVD 0x0	Reserved

Table 198: PCI Express MSI Message Address Register
Offset:0x40054
Configuration: 0x54

Bits	Field	Type/InitVal	Description
1:0	Reserved	RSVD 0x0	Reserved

Table 198: PCI Express MSI Message Address Register (Continued)
Offset:0x40054
Configuration: 0x54

Bits	Field	Type/InitVal	Description
31:2	MSIAddr	RW 0x0	Message Address This field corresponds to Address[31:2] of the MSI MWr TLP.

Table 199: PCI Express MSI Message Address (High) Register
Offset:0x40058
Configuration: 0x58

Bits	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 200: PCI Express MSI Message Data Register
Offset:0x4005C
Configuration: 0x5C

Bits	Field	Type/InitVal	Description
15:0	MSIData	RW 0x0	Message Data
31:16	Reserved	RSVD 0x0	s

Table 201: PCI Express Capability Register
Offset:0x40060
Configuration: 0x60

Bits	Field	Type/InitVal	Description
7:0	CapID	RO 0x10	Capability ID The current value of this field identifies the PCI PE capability.
15:8	NextPtr	RO 0x0	Next Item Pointer The current value of this field points to the end of the capability list (NULL).
19:16	CapVer	RO 0x1	Capability Version This field indicates the PCI Express Base spec 1.0 version of the PCI-Express capability.
23:20	DevType	RO 0x0	Device/Port Type

Table 201: PCI Express Capability Register (Continued)
Offset:0x40060
Configuration: 0x60

Bits	Field	Type/InitVal	Description
24	SlotImp	RO 0x0	Slot Implemented hardwired to 0.
29:25	IntMsgNum	RO 0x0	Interrupt Message Number This bit is hardwired to 0.
31:30	Reserved	RSVD 0x0	Reserved

Table 202: PCI Express Device Capabilities Register
Offset:0x40064
Configuration: 0x64

Bits	Field	Type/InitVal	Description
2:0	MaxPldSizeSup	RO 0x0	Maximum Payload Size Supported 128B MPS support.
5:3	Reserved	RO 0x0	Reserved These bits are hardwired to 0.
8:6	EPL0sAccLat	RO 0x2	Endpoint L0s Acceptable Latency This field indicates the amount of latency that can be absorbed by the 88F5182 due to the transition 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 = 512nsto1us: 512 ns–1 µs. 5 = 1to2us: 1 µs–2 µs. 6 = 2to4us: 2 µs–4 µs. 7 = Above4us: more than 4 µs.
11:9	EPL1AccLat	RO 0x0	Endpoint L1 Acceptable Latency This field indicates the amount of latency that can be absorbed by the 88F5182 due to the transition from L1 to L0. The current value specifies less than 1 µs.
12	AttButPrs	RO 0x0	Attention Button Present 0x0 = Not Implemented: Attention button is not implemented on the card/module. 0x1 = Implemented: Attention button is implemented on the card/module.
13	AttIndPrs	RO 0x0	Attention Indicator Present 0 = Not Implemented: Attention indicator is not implemented on the card/module. 1 = Implemented: Attention indicator is implemented on the card/module.

Table 202: PCI Express Device Capabilities Register (Continued)
Offset:0x40064
Configuration: 0x64

Bits	Field	Type/InitVal	Description
14	PwrIndPrs	RO 0x0	Power Indicator Present 0 = Not Implemented: Power indicator is not implemented on the card/module. 1 = Implemented: Power indicator is implemented on the card/module.
17:15	Reserved	RSVD 0x0	Reserved
25:18	CapSPLVal	RO 0x0	Captured Slot Power Limit Value
27:26	CapSPLScl	RO 0x0	Captured Slot Power Limit Scale
31:28	Reserved	RSVD 0x0	Reserved

Table 203: PCI Express Device Control Status Register
Offset:0x40068
Configuration: 0x68

Bits	Field	Type/InitVal	Description
0	CorErrRepEn	RW 0x0	Correctable Error Reporting Enable 0 = Disabled: ERR_COR error messages are masked. Status bit is not masked. 1 = Enabled: ERR_COR error messages enabled. 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 Table 182, “PCI Express Command and Status Register” is set.
1	NFErrRepEn	RW 0x0	Non-Fatal Error Reporting Enable 0 = Disabled: ERR_NONFATAL error messages are masked. Status bit is not masked. 1 = Enabled: ERR_NONFATAL error messages enabled. 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 Table 182, “PCI Express Command and Status Register” is set.

Table 203: PCI Express Device Control Status Register (Continued)
Offset:0x40068
Configuration: 0x68

Bits	Field	Type/InitVal	Description
2	FErrRepEn	RW 0x0	Fatal Error Reporting Enable 0 = Disabled: ERR_FATAL error messages are masked. Status bit is still affected. 1 = Enabled: ERR_FATAL error messages enabled. 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 Table 182, "PCI Express Command and Status Register" is set.
3	URRepEn	RW 0x0	Unsupported Request (UR) Reporting Enable 0 = Disabled: UR related error messages are masked. Status bit is not masked. 1 = Enabled: UR related error messages enabled. 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 Table 182, "PCI Express Command and Status Register" is set.
4	EnRO	RO 0x0	Enable Relaxed Ordering 88F5182 never sets the Relaxed Ordering attribute in transactions it initiates as a requester. Hardwired to 0x0.
7:5	MaxPldSz	RW 0x0	Maximum Payload Size The maximum payload size supported is 128B (refer to bit <MaxPldSizeSup> in the Table 202, "PCI Express Device Capabilities Register"). 0x0 = 128B Other = Reserved
9:8	Reserved	RO 0x0	Reserved These bits are hardwired to 0.
10	Reserved	RSVD 0x0	Reserved
11	EnNS	RO 0x0	Enable No Snoop 88F5182 never sets the No Snoop attribute in transactions it initiates as a requester. Hardwired to 0x0.
14:12	MaxRdRqSz	RW 0x2	Maximum Read Request Size This field limits the 88F5182 maximum read request size as a requestor (master). 0 = 128B: 1 = 256B: 2 = 512B: 3 = 1 KB: 4 = 2 KB: 5 = 4 KB: Other = Reserved

Table 203: PCI Express Device Control Status Register (Continued)
Offset:0x40068
Configuration: 0x68

Bits	Field	Type/InitVal	Description
15	Reserved	RSVD 0x0	Reserved
16	CorErrDet	SC 0x0	Correctable Error Detected This bit indicates the status of the correctable errors detected by the 88F5182. Write 1 to clear.
17	NFErrDet	SC 0x0	Non-Fatal Error Detected This bit indicates the status of the Non-Fatal errors detected by the 88F5182. Write 1 to clear.
18	FErrDet	SC 0x0	Fatal Error Detected This bit indicates the status of the Fatal errors detected by the 88F5182. Write 1 to clear.
19	URDet	SC 0x0	Unsupported Request Detected This bit indicates that the 88F5182 received an unsupported request. Write 1 to clear.
20	Reserved	RSVD 0x0	Reserved
21	TransPend	RO 0x0	Transactions Pending This bit indicates that the 88F5182 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.
31:22	Reserved	RSVD 0x0	Reserved

Table 204: PCI Express Link Capabilities Register
Offset:0x4006C
Configuration: 0x6C

Bits	Field	Type/InitVal	Description
3:0	MaxLinkSpd	RO 0x1	Maximum Link Speed The current value identifies the 2.5 Gbps link.
9:4	MaxLnkWdth	RO 0x1	Maximum Link Width The current value identifies a X1 link.
11:10	AspmSup	RO 0x1	Active State Link PM Support The current value identifies L0s Entry Support.

Table 204: PCI Express Link Capabilities Register (Continued)
Offset:0x4006C
Configuration: 0x6C

Bits	Field	Type/Init Val	Description
14:12	L0sExtLat	RO 0x2	L0s Exit Latency The time required by the 88F5182 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 = 512nsto1us: 512 ns–1 µs. 5 = 1to2us: 1 µs–2 µs. 6 = 2to4us: 2 µs–4 µs. 7 = Reserved:
17:15	Reserved	RSVD 0x7	Reserved
23:18	Reserved	RSVD 0x0	Reserved
31:24	PortNum	RO 0x0	Port Number Controls the PCI Express port number as advertised in the link training process. In Endpoint mode, indicates the PCI Express port number as was advertised by the Root Complex during the link training process.

Table 205: PCI Express Link Control Status Register
Offset:0x40070
Configuration: 0x70

Bits	Field	Type/InitVal	Description
1:0	AspmCnt	RW 0x0	Active State Link PM Control This field controls the level of active state PM supported on the link. 0 = Disabled: Disabled. 1 = L0Support: L0s entry supported. 2 = Reserved: 3 = L0L1Support: L0s and L1 entry supported.
2	Reserved	RSVD 0x0	Reserved
3	RCB	RW 0x1	Read Completion Boundary 0 = 64b: 1 = 128B: **NOTE:** This bit has no effect on the device behavior. Completions are always returned with 128B read completion boundary.

Table 205: PCI Express Link Control Status Register (Continued)
Offset:0x40070
Configuration: 0x70

Bits	Field	Type/InitVal	Description
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 (Table 175, "PCI Express Status Register", bit 0) ensure the link is disabled. 3. Clear the bit to exit to detect and enable the link again.
5	RetrnLnk	RW 0x0	Retrain Link This bit forces the device to initiate link retraining. Always returns 0 when read. **NOTE:** If configured as an Endpoint, this field is reserved and has no effect.
6	CmnClkCfg	RW 0x0	Common Clock Configuration When set by SW, this bit indicates that both devices on the link use a distributed common reference clock.
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.
15:8	Reserved	RSVD 0x0	Reserved
19:16	LnkSpd	RO 0x1	Link Speed This field indicates the negotiated link speed. The current value indicates 2.5 Gbps.
25:20	NegLnkWdth	RO	Negotiated Link Width This field indicates the negotiated link width. 1 = X1 Other = Reserved This field is initialized by the hardware. **NOTE:** This field is only valid once the link is up.
26	Reserved	RO 0x0	Reserved
27	LnkTrn	RO 0x0	Link Training This bit indicates that link training is in progress or that 1b was written to the Retrain Link bit, but Link training has not yet begun. This bit is cleared once link training is complete.
28	SltClkCfg	RO 0x1	Slot Clock Configuration 0 = Independent: The 88F5182 uses an independent clock, irrespective of the presence of a reference clock on the connector. 1 = Reference: The 88F5182 uses the reference clock that the platform provides.
31:29	Reserved	RSVD 0x0	Reserved

Table 206: PCI Express Advanced Error Report Header Register
Offset:0x40100
Configuration: 0x100

Bits	Field	Type/InitVal	Description
15:0	PECapID	RO 0x1	Extended Capability ID The current value of this field identifies the Advanced Error Reporting capability.
19:16	CapVer	RO 0x1	Capability Version
31:20	NextPtr	RO 0x0	Next Item Pointer This field indicates the last item in the extended capabilities linked list.

Table 207: PCI Express Uncorrectable Error Status Register
Offset:0x40104
Configuration: 0x104

NOTE: 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.

Bits	Field	Type/InitVal	Description
3:0	Reserved	RSVD 0x0	Reserved
4	DLPrtErr	SC 0x0	Data Link Protocol Error Status **NOTE:** Hot sticky bit—not initialized by hot reset.
11:5	Reserved	RSVD 0x0	Reserved
12	RPsnTlpErr	SC 0x0	Poisoned TLP Status **NOTE:** Hot sticky bit—not initialized by hot reset.
13	Reserved	SC 0x0	Reserved **NOTE:** Sticky bit—not initialized by hot reset.
14	CmpTOErr	SC 0x0	Completion Timeout Status **NOTE:** Sticky bit—not initialized by hot reset.
15	CAErr	SC 0x0	Completer Abort Status **NOTE:**Sticky bit—not initialized by hot reset.
16	UnexpCmpErr	SC 0x0	Unexpected Completion Status **NOTE:** Sticky bit—not initialized by hot reset.
17	Reserved	SC 0x0	Reserved **NOTE:** Sticky bit—not initialized by hot reset.
18	MalfTlpErr	SC 0x0	Malformed TLP Status **NOTE:** Sticky bit—not initialized by hot reset.
19	Reserved	RSVD 0x0	Reserved.

Table 207: PCI Express Uncorrectable Error Status Register (Continued)
Offset:0x40104
Configuration: 0x104

NOTE: 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.

Bits	Field	Type/InitVal	Description
20	URErr	SC 0x0	Unsupported Request Error Status **NOTE:** Sticky bit—not initialized by hot reset.
31:21	Reserved	RSVD 0x0	Reserved

Table 208: PCI Express Uncorrectable Error Mask Register
Offset:0x40108
Configuration: 0x108

NOTE: All fields in this register are sticky—not initialized or modified by hot reset.

Bits	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	When an error is indicated in the PCI Express Uncorrectable Error Status Register (Table 207 p. 312) 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. 0x0 = Not masked 0x1 = Masked.

Table 209: PCI Express Uncorrectable Error Severity Register
Offset:0x4010C
Configuration: 0x10C

NOTE: All fields in this register are hot sticky—not initialized or modified by reset.

Bits	Field	Type/InitVal	Description
31:0	Severity	RW 0x00060010	Uncorrectable Error Severity Control 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 (Table 207 p. 312). 0 = Error type is Non-Fatal. 1 = Error type is Fatal.

Table 210: PCI Express Correctable Error Status Register
Offset:0x40110
Configuration: 0x110

NOTE: 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.

Bits	Field	Type/InitVal	Description
0	RcvErr	SC 0x0	Receiver Error Status **NOTE:** When set, this bit indicates that a Receiver error has occurred.
5:1	Reserved	RSVD 0x0	Reserved
6	BadTlpErr	SC 0x0	Bad TLP Status
7	BadDllpErr	SC 0x0	Bad DLLP Status
8	RplyRllovrErr	SC 0x0	Replay Number Rollover Status
11:9	Reserved	RSVD 0x0	Reserved
12	RplyTOErr	SC 0x0	Replay Timer Timeout status
31:13	Reserved	RSVD 0x0	Reserved

Table 211: PCI Express Correctable Error Mask Register
Offset:0x40114
Configuration: 0x114

NOTE: All fields in this register are sticky—not initialized or modified by hot reset.

Bits	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	If set, an error message is not generated upon occurrence of a Receiver error. 0 = Not masked 1 = Masked

Table 212: PCI Express Advanced Error Capability and Control Register
Offset:0x40118
Configuration: 0x118

Bits	Field	Type/InitVal	Description
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 (Table 207 p. 312). 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
31:5	Reserved	RSVD 0x0	Reserved Bits [5] and [7] are Read Only and are hardwired to 0.

Table 213: PCI Express Header Log First DWORD Register
Offset:0x4011C
Configuration: 0x11C

NOTE: All fields in this register are sticky—not initialized or modified by hot reset.

Bits	Field	Type/InitVal	Description
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 (Table 207 p. 312). 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.

Table 214: PCI Express Header Log Second DWORD Register
Offset:0x40120
Configuration: 0x120

NOTE: All fields in this register are sticky—not initialized or modified by hot reset.

Bits	Field	Type/InitVal	Description
31:0	Hdrlog2DW	RO 0x0	Header Log Second DWORD

Table 215: PCI Express Header Log Third DWORD Register
Offset:0x40124
Configuration: 0x124

NOTE: All fields in this register are sticky—not initialized or modified by hot reset.

Bits	Field	Type/InitVal	Description
31:0	Hdrlog3DW	RO 0x0	Header Log Third DWORD

Table 216: PCI Express Header Log Fourth DWORD Register
Offset:0x40128
Configuration: 0x128

NOTE: All fields in this register are sticky—not initialized or modified by hot reset.

Bits	Field	Type/InitVal	Description
31:0	Hdrlog4DW	RO 0x0	Header Log Fourth DWORD

A.7 PCI Interface Registers

The PCI interface registers consist of configuration registers (PCI configuration header) and internal registers.

The internal registers are part of the entire device's internal registers map. They can be accessed from the local CPU and also from external PCI host, via Memory mapped (or I/O mapped) Internal registers BAR space. The specified register offset is, in respect to the Internal registers, the window base address.

The configuration registers are located at their standard offset, as defined in the PCI specification. They can be accessed from the local CPU, using Configuration Address and Configuration Data registers. They can also be accessed by an external PCI host, using type 0 configuration cycle. The specified register offset is the address within the PCI configuration header of the specific function number.

Table 217: PCI Interface Register Map

Register	Offsets	Page
PCI Slave Address Decoding Registers		
CSn[0] BAR Size	0x30C08	Table 218, p.320
CSn[1] BAR Size	0x30D08	Table 219, p.320
CSn[2] BAR Size	0x30C0C	Table 220, p.320
CSn[3] BAR Size	0x30D0C	Table 221, p.321
DevCSn[0] BAR Size	0x30C10	Table 222, p.321
DevCSn[1] BAR Size	0x30D10	Table 223, p.321
DevCSn[2] BAR Size	0x30D18	Table 224, p.321
Boot CSn BAR Size	0x30D14	Table 225, p.321
P2P Mem0 BAR Size	0x30D1C	Table 226, p.322
P2P I/O BAR Size	0x30D24	Table 227, p.322
Expansion ROM BAR Size	0x30D2C	Table 228, p.322
Base Address Registers Enable	0x30C3C	Table 229, p.322
CSn[0] Base Address Remap	0x30C48	Table 230, p.323
CSn[1] Base Address Remap	0x30D48	Table 231, p.324
CSn[2] Base Address Remap	0x30C4C	Table 232, p.324
CSn[3] Base Address Remap	0x30D4C	Table 233, p.324
DevCSn[0] Base Address Remap	0x30C50	Table 234, p.324
DevCSn[1] Base Address Remap	0x30D50	Table 235, p.325
DevCSn[2] Base Address Remap	0x30D58	Table 236, p.325
BootCSn Base Address Remap	0x30D54	Table 237, p.325
P2P Mem0 Base Address Remap (Low)	0x30D5C	Table 238, p.325
P2P Mem0 Base Address Remap (High)	0x30D60	Table 239, p.325
P2P I/O Base Address Remap	0x30D6C	Table 240, p.326
Expansion ROM Base Address Remap	0x30F38	Table 241, p.326
DRAM BAR Bank Select	0x30C1C	Table 242, p.326
PCI Address Decode Control	0x30D3C	Table 243, p.326
PCI Control Registers		

Table 217: PCI Interface Register Map (Continued)

Register	Offsets	Page
PCI DLL Control	0x31D20	Table 244, p.327
PCI/MPP Pads Calibration	0x31D1C	Table 245, p.328
PCI Command	0x30C00	Table 246, p.329
PCI Mode	0x30D00	Table 248, p.332
PCI Retry	0x30C04	Table 249, p.332
PCI Discard Timer	0x30D04	Table 250, p.333
MSI Trigger Timer	0x30C38	Table 251, p.333
PCI Arbiter Control	0x31D00	Table 252, p.333
PCI P2P Configuration	0x31D14	Table 253, p.334
PCI Access Control Base 0 (Low)	0x31E00	Table 254, p.334
PCI Access Control Base 0 (High)	0x31E04	Table 255, p.335
PCI Access Control Size 0	0x31E08	Table 256, p.336
PCI Access Control Base 1 (Low)	0x31E10	Table 257, p.336
PCI Access Control Base 1 (High)	0x31E14	Table 258, p.337
PCI Access Control Size 1	0x31E18	Table 259, p.337
PCI Access Control Base 2 (Low)	0x31E20	Table 260, p.337
PCI Access Control Base 2 (High)	0x31E24	Table 261, p.337
PCI Access Control Size 2	0x31E28	Table 262, p.337
PCI Access Control Base 3 (Low)	0x31E30	Table 263, p.338
PCI Access Control Base 3 (High)	0x31E34	Table 264, p.338
PCI Access Control Size 3	0x31E38	Table 265, p.338
PCI Access Control Base 4 (Low)	0x31E40	Table 266, p.338
PCI Access Control Base 4 (High)	0x31E44	Table 267, p.338
PCI Access Control Size 4	0x31E48	Table 268, p.339
PCI Access Control Base 5 (Low)	0x31E50	Table 269, p.339
PCI Access Control Base 5 (High)	0x31E54	Table 270, p.339
PCI Access Control Size 5	0x31E58	Table 271, p.339
PCI Configuration Access Registers		
PCI Configuration Address	0x30C78	Table 272, p.340
PCI Configuration Data	0x30C7C	Table 273, p.340
PCI Interrupt Acknowledge	0x30C34	Table 274, p.340
PCI Error Report Registers		
PCI SERRn Mask	0x30C28	Table 275, p.341
PCI Interrupt Cause	0x31D58	Table 276, p.342
PCI Interrupt Mask	0x31D5C	Table 277, p.343
PCI Error Address (Low)	0x31D40	Table 278, p.343
PCI Error Address (High)	0x31D44	Table 279, p.343
PCI Error Command	0x31D50	Table 280, p.344
PCI Configuration, Function 0, Registers		
PCI Device and Vendor ID	0x00	Table 281, p.344

Table 217: PCI Interface Register Map (Continued)

Register	Offsets	Page
PCI Status and Command	0x04	Table 282, p.344
PCI Class Code and Revision ID	0x08	Table 283, p.346
PCI BIST, Header Type/Initial Value, Latency Timer, and Cache Line	0x0C	Table 284, p.346
PCI CSn[0] Base Address (Low)	0x10	Table 285, p.347
PCI CSn[0] Base Address (High)	0x14	Table 286, p.347
PCI CSn[1] Base Address (Low)	0x18	Table 287, p.347
PCI CSn[1] Base Address (High)	0x1C	Table 288, p.348
PCI Internal Registers Memory Mapped Base Address (Low)	0x20	Table 289, p.348
PCI Internal Registers Memory Mapped Base Address (High)	0x24	Table 290, p.348
PCI Subsystem Device and Vendor ID	0x2C	Table 291, p.348
PCI Expansion ROM Base Address Register	0x30	Table 292, p.349
PCI Capability List Pointer Register	0x34	Table 293, p.349
PCI Interrupt Pin and Line	0x3C	Table 294, p.349
PCI Power Management	0x40	Table 295, p.349
PCI Power Management Control and Status	0x44	Table 296, p.350
PCI VPD Address	0x48	Table 297, p.351
PCI VPD Data	0x4C	Table 298, p.351
PCI MSI Message Control	0x50	Table 299, p.352
PCI MSI Message Address	0x54	Table 300, p.352
PCI MSI Message Upper Address	0x58	Table 301, p.352
PCI Message Data	0x5C	Table 302, p.353
CompactPCI HotSwap	0x68	Table 303, p.353
PCI Configuration, Function 1, Registers		
PCI CSn[2] Base Address (Low)	0x10	Table 304, p.354
PCI CSn[2] Base Address (High)	0x14	Table 305, p.354
PCI CSn[3] Base Address (Low)	0x18	Table 306, p.354
PCI CSn[3] Base Address (High)	0x1C	Table 307, p.354
PCI Configuration, Function 2, Registers		
PCI DevCS[0] Base Address (Low)	0x10	Table 308, p.354
PCI DevCSn[0] Base Address (High)	0x14	Table 309, p.355
PCI DevCSn[1] Base Address (Low)	0x18	Table 310, p.355
PCI DevCSn[1] Base Address (High)	0x1C	Table 311, p.355
PCI DevCSn[2] Base Address (Low)	0x20	Table 312, p.355
PCI DevCSn[2] Base Address (High)	0x24	Table 313, p.356
PCI Configuration, Function 3, Registers		
PCI BootCS Base Address (Low)	0x18	Table 314, p.356
PCI BootCSn Base Address (High)	0x1C	Table 315, p.356
PCI Configuration, Function 4, Registers		

Table 217: PCI Interface Register Map (Continued)

Register	Offsets	Page
PCI P2P Mem0 Base Address (Low)	0x10	Table 316, p.356
PCI P2P Mem0 Base Address (High)	0x14	Table 317, p.356
PCI P2P I/O Base Address	0x20	Table 318, p.357
PCI Internal Registers I/O Mapped Base Address	0x24	Table 319, p.357

A.7.1 PCI Slave Address Decoding Registers

Table 218: CSn[0] BAR Size
Offset:0x30C08

Bits	Field	Type/ InitVal	Description
11:0	Reserved	RO 0x0	Read only.
31:12	BARSize	RW 0x0FFFF	CSn[0] BAR Address Bank Size Must be programmed from LSB to MSB as sequence of '1s' followed by sequence of '0s'. For example, a 0x0FFF.F000 size register value represents a BAR size of 256 MB.

Table 219: CSn[1] BAR Size
Offset:0x30D08

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x0FFFF000	Same as CSn[0] BAR Size

Table 220: CSn[2] BAR Size
Offset:0x30C0C

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x0FFFF000	Same as CSn[0] BAR Size

Table 221: CSn[3] BAR Size
Offset:0x30D0C

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x0FFFF000	Same as CSn[0] BAR Size

Table 222: DevCSn[0] BAR Size
Offset:0x30C10

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x07FFF000	Same as CSn[0] BAR Size

Table 223: DevCSn[1] BAR Size
Offset:0x30D10

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x07FFF000	Same as CSn[0] BAR Size

Table 224: DevCSn[2] BAR Size
Offset:0x30D18

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x07FFF000	Same as CSn[0] BAR Size

Table 225: Boot CSn BAR Size
Offset:0x30D14

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x07FFF000	Same as CSn[0] BAR Size

Table 226: P2P Mem0 BAR Size
Offset:0x30D1C

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x1FFFF000	s

Table 227: P2P I/O BAR Size
Offset:0x30D24

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x0000F000	Same as CSn[0] BAR Size

Table 228: Expansion ROM BAR Size
Offset:0x30D2C

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x0	Same as CSn[0] BAR Size

Table 229: Base Address Registers Enable
Offset:0x30C3C

Bits	Field	Type/ InitVal	Description
0	CS0En	RW 0x0	CSn[0] BAR Enable 0 = Enable 1 = Disable
1	CS1En	RW 0x0	CSn[1] BAR Enable 0 = Enable 1 = Disable
2	CS2En	RW 0x0	CSn[2] BAR Enable 0 = Enable 1 = Disable
3	CS3En	RW 0x0	CSn[3] BAR Enable 0 = Enable 1 = Disable

Table 229: Base Address Registers Enable (Continued)
Offset:0x30C3C

Bits	Field	Type/ InitVal	Description
4	DevCS0En	RW 0x1	DevCSn[0] BAR Enable 0 = Enable 1 = Disable
5	DevCS1En	RW 0x1	DevCSn[1] BAR Enable 0 = Enable 1 = Disable
6	DevCS2En	RW 0x0	DevCSn[2] BAR Enable 0 = Enable 1 = Disable
7	Reserved	RW 0x1	Must be 1.
8	BootCSEn	RW 0x0	BootCSn BAR Enable 0 = Enable 1 = Disable
9	IntMemEn	RW 0x0	Memory Mapped Internal Registers BAR Enable 0 = Enable 1 = Disable
10	IntIOEn	RW 0x1	I/O Mapped Internal Registers BAR Enable 0 = Enable 1 = Disable
NOTE: The 88F5182 prevents disabling both memory mapped and I/O mapped BARs (bits [9] and [10] cannot simultaneously be set to 1).			
11	P2PMem0En	RW 0x1	P2P Mem0 BAR Enable 0 = Enable 1 = Disable
12	Reserved	RW 0x1	Reserved
13	P2PIOEn	RW 0x1	P2P I/O BAR Enable 0 = Enable 1 = Disable
15:14	Reserved	RW 0x3	Reserved Must be 0x3.
31:16	Reserved	RES 0xFFFF	Reserved

Table 230: CSn[0] Base Address Remap
Offset:0x30C48

Bits	Field	Type/ InitVal	Description
11:0	Reserved	RO 0x0	Read only.

Table 230: CSn[0] Base Address Remap (Continued)
Offset:0x30C48

Bits	Field	Type/ InitVal	Description
31:12	CS0Remap	RW 0x00000	CSn[0] BAR Remap Address

Table 231: CSn[1] Base Address Remap
Offset:0x30D48

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x10000000	Same as CSn[0] Base Address Remap

Table 232: CSn[2] Base Address Remap
Offset:0x30C4C

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x20000000	Same as CSn[0] Base Address Remap

Table 233: CSn[3] Base Address Remap
Offset:0x30D4C

Bits	Field	Type/ InitVal	Description
31:0	Various	RO RW 0x30000000	Same as CSn[0] Base Address Remap

Table 234: DevCSn[0] Base Address Remap
Offset:0x30C50

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0xE0000000	Same as CSn[0] Base Address Remap

Table 235: DevCSn[1] Base Address Remap
Offset:0x30D50

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0xE8000000	Same as CSn[0] Base Address Remap

Table 236: DevCSn[2] Base Address Remap
Offset:0x30D58

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0xF0000000	Same as CSn[0] Base Address Remap

Table 237: BootCSn Base Address Remap
Offset:0x30D54

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0xF8000000	Same as CSn[0] Base Address Remap

Table 238: P2P Mem0 Base Address Remap (Low)
Offset:0x30D5C

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x80000000	Same as CSn[0] Base Address Remap

Table 239: P2P Mem0 Base Address Remap (High)
Offset:0x30D60

Bits	Field	Type/ InitVal	Description
31:0	P2P0Remap	RW 0x0	P2P Mem0 BAR Remap Address

Table 240: P2P I/O Base Address Remap
Offset:0x30D6C

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0xC0000000	Same as CSn[0] Base Address Remap

Table 241: Expansion ROM Base Address Remap
Offset:0x30F38

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0xE0000000	Same as CSn[0] Base Address Remap

Table 242: DRAM BAR Bank Select
Offset:0x30C1C

Bits	Field	Type/ InitVal	Description
1:0	DB0	RW 0x0	DRAM CS0n BAR select: 0x0 = CS0n 0x1 = CS1n 0x2 = CS2n 0x3 = CS3n
3:2	DB1	RW 0x1	Same as <DB0>
5:4	DB2	RW 0x2	Same as <DB0>
7:6	DB3	RW 0x3	Same as <DB0>
31:8	Reserved	RO 0x0	Reserved

Table 243: PCI Address Decode Control
Offset:0x30D3C

Bits	Field	Type/ InitVal	Description
0	RemapWrDis	RW 0x0	Address Remap Registers Write Disable 0 = Write to a BAR result sin updating the corresponding remap register with the new value of the BAR. 1 = Write to a BAR has no effect on the corresponding Remap register value.

Table 243: PCI Address Decode Control (Continued)
Offset:0x30D3C

Bits	Field	Type/ InitVal	Description
2:1	Reserved	RES 0x0	
3	Reserved	RW 0x1	Reserved
7:4	Reserved	RES 0x0	Reserved
24:8	VPDHighAddr	RW 0x0	Bits [31:15] of the VPD address
31:25	Reserved	RES 0x0	Reserved

A.7.2 PCI Control Registers

Table 244: PCI DLL Control
Offset:0x31D20

Bits	Field	Type/ InitVal	Description
0	En	RW 0x0	DLL Enable 0 = Disabled 1 = Enabled
2:1	Mode	RW 0x1	DLL Mode Only relevant if <En> is enabled. 0x0 = Use PCLK as reference clock. 0x1 = Use inverted PCLK as reference clock. 0x2 = Reserved 0x3 = Reserved
4:3	Err	RO 0x0	DLL Error indication—DLL reach delay line boundary 0x0 = No error 0x1 = Delay line pointer at start, and down count. 0x2 = Delay line pointer at end, and up count. 0x3 = Reserved
8:5	RCnt	RO 0x0	Row Counter. Current row number of the delay line. Calculate the TAPs number, using row and column numbers. Read only.
12:9	CCnt	RO 0x0	Column Counter. Current column number of the delay line. Calculate the TAPs number, using row and column numbers. Read only.
14:13	Init	RW 0x0	Delay line initial state 0x0 = 4 Taps from the beginning of delay line. 0x1 = 8 Taps from the beginning of delay line. 0x2 = 16 Taps from the beginning of delay line. 0x3 = 50 Taps from the beginning of delay line.

Table 244: PCI DLL Control (Continued)
Offset:0x31D20

Bits	Field	Type/ InitVal	Description
15	Reserved	RES 0x0	Reserved
16	CompEn	RW 0x0	Enable Pad Delay Compensation 0 = Disable 1 = Enable
19:17	Reserved	RW 0x0	Reserved
21:20	DLLDelicateTune	RW 0x1	DLL Delicate Tune Control
30:22	Reserved	RES 0x0	Reserved
31	WrEn	RW 0x0	DLL Status and Control Register Write Enable 0 = The register becomes read only (except for bit [31]). 1 = The register can be written to.

Only applicable when MPPSetEN = 0.

Table 245: PCI/MPP Pads Calibration
Offset:0x31D1C

Bits	Field	Type/ InitVal	Description
3:0	DrvN	RW 0xF	PCI Pad Driving N Strength **NOTE:** Only applicable when auto-calibration is disabled.
7:4	MppDrvP0	RES 0xF	MPP Pad Driving P Strength **NOTE:** Only applicable when <MPPSetEn> = 1.
11:8	MppDrvN1	RW 0xF	MPP Pad Driving N Strength Useful for changing MPP drive N strength after the automatic tuning completes. **NOTE:** Only applicable when <MPPSetEn> = 0.
15:12	MppDrvP1	RES 0xF	MPP Pad Driving P Strength Useful for changing MPP drive P strength after the automatic tuning completes. **NOTE:** Only applicable when <MPPSetEn> = 0.
16	TuneEn	RW 0x1	Auto-calibration of Pad Driving Strength 0 = Disabled 1 = Enabled
17	MPPSetEn	RW 0x1	MPP Drive Strength Enable 0 = Enables changing the drive strength of the MPPs via <MppDrvN1> and <MppDrvP1>. 1 = The drive strength of the MPPs is equal to <DrvN> and <MppDrvP0>.
21:18	Lock	RO 0x0	When auto-calibration is enabled, this field represent the final locked value of the driving strength. Read Only.

Table 245: PCI/MPP Pads Calibration (Continued)
Offset:0x31D1C

Bits	Field	Type/ InitVal	Description
30:22	Reserved	RES 0x0	Reserved
31	WrEn	RW 0x0	PCI/MPP Pads Calibration Register Write Enable 0 = The register becomes read only (except for bit [31]). 1 = The register can be written to. If set to 1, this register is writable. If set to 0, the register becomes read only (except for bit [31])

Table 246: PCI Command
Offset:0x30C00

Bits	Field	Type/ InitVal	Description
0	MByteSwap	RW 0x1	PCI Master Byte Swap When set to 0, the 88F5182 PCI master swaps the bytes of the incoming and outgoing PCI data (swap the 8 bytes of a long-word).
3:1	Reserved	RES 0x0	Read Only.
4	MWrCom	RW 0x1	PCI Master Write Combine Enable When set to 1, write combining is enabled. **NOTE:** Write combining must not be enabled when using cache line size of 4.
5	MRdCom	RW 0x1	PCI Master Read Combine Enable When set to 1, read combining is enabled.
6	MWrTrig	RW 0x1	PCI Master Write Trigger 0 = Accesses the PCI bus only when the whole burst is written into the master write buffer. 1 = Accesses the PCI bus when the first data is written into the master write buffer.
7	MRdTrig	RW 0x0	PCI Master Read Trigger 0 = Returns read data to the initiating unit only when the whole burst is written into master read buffer. 1 = Returns read data to the initiating unit when the first read data is written into master read buffer.
8	MRdLine	RW 0x1	PCI Master Memory Read Line Enable 0 = Disable 1 = Enable
9	MRdMul	RW 0x1	PCI Master Memory Read Multiple Enable 0 = Disable 1 = Enable
10	MWordSwap	RW 0x0	PCI Master Word Swap When set to 1, the 88F5182 PCI master swaps the 32-bit words of the incoming and outgoing PCI data.

Table 246: PCI Command (Continued)
Offset:0x30C00

Bits	Field	Type/ InitVal	Description
11	SWordSwap	RW 0x0	PCI Slave Word Swap When set to 1, the 88F5182 PCI slave swaps the bytes of the incoming and outgoing PCI data (swaps the two words of a long-word).
12	Reserved	RES 0x0	Reserved
13	Reserved	RES 0x1	Reserved
14	Reserved	RW 0x1	Reserved
15	Reserved	RES 0x0	Reserved
16	SByteSwap	RW 0x1	PCI Slave Byte Swap When set to 0, the 88F5182 PCI slave swaps the bytes of the incoming and outgoing PCI data (swap the 8 bytes of a long-word).
17	MDACEn	RW 0x1	PCI Master DAC Enable 0 = The PCI master never drives the DAC cycle. 1 = The PCI master drives the DAC cycle, if the upper 32-bit address is not 0.
18	Reserved	RES 0x0	Reserved
19	PErrProp	RW 0x0	Parity/ECC Errors Propagation Enable 0 = The PCI interface always drives correct parity on the PAR signal. 1 = In case of erroneous data indication, the PCI interface drives bad parity on the PAR signal (PCI slave read data or PCI master write data).
20	SSwapEn	RW 0x0	PCI Slave Swap Enable 0 = PCI slave data swapping is determined via <SByteSwap> and <SWordSwap> bits (bits [16] and [11]). 1 = PCI slave data swapping is determined via <PCISwap> bits [7:6] in the PCI Access Control registers. **NOTE:** If PCI address does not match any of the Access Control windows, the PCI slave data swapping works according to <SByteSwap> and <SWordSwap> bits, even if the <SSwapEn> bit is set to 1.
21	MSwapEn	RW 0x0	PCI Master Swap Enable 0 = PCI master data swapping is determined via <MByteSwap> and <MWordSwap> bits (bits 0 and 10). 1 = PCI master data swapping is determined via <PCISwap> bits in the different unit Base Address registers.
23:22	Reserved	RW 0x0	Reserved Must be 0x0
25:24	SIntSwap	RW 0x1	PCI Slave data swap control on PCI accesses to the 88F5182 internal and configuration registers. 00 = Byte Swap 01 = No swapping 10 = Both byte and word swap 11 = Word swap

Table 246: PCI Command (Continued)
Offset:0x30C00

Bits	Field	Type/ InitVal	Description
27:26	Reserved	RW 0x0	Reserved Must be 0x0
28	SSBInt	RW 0x0	PCI-to-CPU Ordering enable—internal registers access. If set to 1, PCI to CPU ordering is maintained by hardware upon any PCI read access from any of the 88F5182 internal registers.
31:29	Reserved	RW 0x0	Reserved Must be 0x3.

Table 247: PCI Mode
Offset:0x30D00

Bits	Field	Type/ InitVal	Description
1:0	Reserved	RES 0x0	Reserved
2	Pci64	RO Sampled on reset	64-bit PCI Interface When set to 1, the PCI interface is configured to a 64-bit interface. Read Only.
3	Reserved	RES 0x0	Reserved
5:4	PciMode	RW Sampled at reset.	PCI Interface Mode of Operation 0x0 = Conventional PCI 0x1 = 66 MHz PCI-X 0x2 = 100 MHz PCI-X 0x3 = 133 MHz PCI-X **NOTE:** Must not be changed after reset.
7:6	Reserved	RES 0x0	Reserved
8	ExpRom	RW 0x0	Expansion ROM Active When set to 1, the expansion ROM BAR is supported. Read Only from PCI.
30:9	Reserved	RES 0x0	Reserved
31	PRst	RO 0x1	PCI Interface Reset Indication Set to 0 as long as the PCI_RSTn pin is asserted. Read Only.

Table 248: PCI Mode
Offset:0x30D00

Bits	Field	Type/ InitVal	Description
3:0	Reserved	RES 0x0	Reserved
5:4	PciMode	RW Sampled at reset.	PCI Interface Mode of Operation 0x0 = Conventional PCI 0x1 = Reserved 0x2 = Reserved 0x3 = Reserved **NOTE:** Must not be changed after reset.
7:6	Reserved	RES 0x0	Reserved
8	ExpRom	RW 0x0	Expansion ROM Active When set to 1, the expansion ROM BAR is supported. Read Only from PCI.
30:9	Reserved	RES 0x0	Reserved
31	PRst	RO 0x1	PCI Interface Reset Indication Set to 0 as long as the PCI_RSTn pin is asserted. Read Only.

Table 249: PCI Retry
Offset:0x30C04

Bits	Field	Type/ InitVal	Description
15:0	Reserved	RES 0x0	Reserved
23:16	RetryCtr	RW 0x0	Retry Counter Specifies the number of times the 88F5182 retries a transaction before it quits. Applies to the PCI Master when acting as a requester. A 0x00 value means a “retry forever”.
31:24	Reserved	RES 0x0	Reserved

Table 250: PCI Discard Timer
Offset:0x30D04

NOTE: This register must not be updated while read buffers are not cleared.

Bits	Field	Type/ InitVal	Description
15:0	Timer	RW 0x0	Specifies the number of PCLK cycles the 88F5182 PCI slave keeps a non-accessed read buffers (non-completed delayed read) before invalidating the buffer. Set to 0 to disable the timer. The PCI slave waits for delayed read completion forever. **NOTE:** Must be set to a number greater than 0x7F, unless using the “wait for ever” setting 0x0. Must not be updated while there are pending read requests.
31:16	Reserved	RES 0x0	Reserved

Table 251: MSI Trigger Timer
Offset:0x30C38

Bits	Field	Type/ InitVal	Description
15:0	Timer	RW 0xFFFF	Specifies the number of TCLK cycles between consecutive MSI requests. **NOTE:** If set to 0x0, the timer is disabled.
31:16	Reserved	RES 0x0	Reserved

Table 252: PCI Arbiter Control
Offset:0x31D00

NOTE: Arbiter is set once and must not be changed during operation.

Bits	Field	Type/ InitVal	Description
0	Reserved	RES 0x0	Reserved
1	BDEn	RW 0x0	Broken Detection Enable If set to 1, broken master detection is enabled. A master is said to be broken if it fails to respond to grant assertion within a window specified in BV field.
2	Reserved	RES 0x0	Reserved
6:3	BV	RW 0x6	Broken Value The value sets the maximum number of cycles that the arbiter waits for a PCI master to respond to its grant assertion. If a PCI master fails to assert PCI_FRAMEn within this time, the PCI arbiter aborts the transaction and performs a new arbitration cycle and a maskable interrupt is generated. **NOTE:** Must be greater than 1.

Table 252: PCI Arbiter Control (Continued)
Offset:0x31D00

NOTE: Arbiter is set once and must not be changed during operation.

Bits	Field	Type/ InitVal	Description
13:7	Reserved	RES 0x0	Reserved
20:14	PD[6:0]	RW 0x0	Parking Disable When a PD bit is set to 1, parking on the associated PCI master is disabled. **NOTE:** The arbiter parks on the last master granted unless disabled through the PD bit. Also, if PD bits are all 1, the PCI arbiter parks on the internal PCI master. PD0 corresponds to the internal master. PD1 corresponds to PCI_GNTn.
30:21	Reserved	RES 0x0	Reserved
31	EN	RW 0x0	Enable Internal Arbiter Operation 0 = Disable 1 = Enable

Table 253: PCI P2P Configuration
Offset:0x31D14

Bits	Field	Type/ InitVal	Description
15:0	Reserved	RW 0xFF	Reserved Must be 0xFF
23:16	BusNumber	RW 0x0	The number of the PCI bus to which the PCI interface is connected.
28:24	DevNum	RW 0x0	The Device number of the PCI interface
31:29	Reserved	RES 0x0	Reserved

Table 254: PCI Access Control Base 0 (Low)
Offset:0x31E00

Bits	Field	Type/ InitVal	Description
0	En	RW 0x0	Access control window enable 0 = Disable 1 = Enable
3:1	Reserved	RW 0x0	Reserved

Table 254: PCI Access Control Base 0 (Low) (Continued)
Offset:0x31E00

Bits	Field	Type/ InitVal	Description
4	AccProt	RW 0x0	Access Protect 0 = PCI access to this region is allowed. 1 = PCI access to this region is forbidden.
5	WrProt	RW 0x0	Write Protect 0 = PCI write is to this region is allowed. 1 = PCI write to this region is forbidden.
7:6	PCISwap	RW 0x0	PCI Slave Data Swap Control 00 = Byte Swap 01 = No swapping 10 = Both byte and word swap 11 = Word swap
9:8	RdMBurst	RW 0x0	Read Maximum Burst Specifies the maximum read burst size for a single transaction between a PCI slave and the other interfaces. 00 = 32 bytes 01 = 64 bytes 10 = 128 bytes 11 = Reserved
11:10	RdSize	RW 0x0	Typical PCI Read Transaction Size Defines the amount of data the slave prefetches from the target unit: 00 = 32 bytes 01 = 64 bytes 10 = 128 bytes 11 = 256 bytes **NOTE:** If the transaction address hits a non prefetchable space (prefetch bit of the BAR is cleared), the slave reads a single data, regardless of the setting of this field.
31:12	Base	RW 0x0	Access Control Base Address Corresponds to address bits [31:12].

Table 255: PCI Access Control Base 0 (High)
Offset:0x31E04

Bits	Field	Type/ InitVal	Description
31:0	Base	RW 0x0	Base Address High Corresponds to address bits [63:32].

Table 256: PCI Access Control Size 0
Offset:0x31E08

Bits	Field	Type/ InitVal	Description
3:0	Reserved	RW 0x0	Reserved
4	AggrWM1	RW 0x0	Aggressive Prefetch Water Mark 1 0 = The 88F5182 drives read data on the bus as soon as it has **one** 256-byte read buffer valid 1 = The 88F5182 drives read data on the bus as soon as it has **two** 256-byte read buffers valid.
7:5	AggrWM2	RW 0x0	Aggressive Prefetch Water Mark 2 Defines at which point the 88F5182 PCI slave prefetches the next buffer from memory. 0 = Fetches next buffer after driving one 64-bit word on the bus. 1 = Fetches after driving two 64-bit words on the bus, and so on.
9:8	WrMBurst	RW 0x0	Write Max Burst Specifies the maximum burst size for a single write transaction between a PCI slave and the other interfaces 00 = 32 bytes 01 = 64 bytes 10 = 128 bytes 11 = Reserved
10	Aggr	RW 0x0	Aggressive Prefetch Enable If set to 1, RdSize setting is ignored, and the 88F5182 PCI slave prefetches read data as long as the requester does not DISCONNECT.
11	PciOR	RW 0x0	PCI Ordering required 0 = Hardware does not support PCI ordering. 1 = Hardware enforced PCI ordering
31:12	Size	RW 0x0	PCI access window size Must be programmed from LSB to MSB as sequence of ‘1s’ followed by sequence of ‘0s’. For example, a 0x00FFF size register value represents a window size of 16 MB

Table 257: PCI Access Control Base 1 (Low)
Offset:0x31E10

Bits	Field	Type/ InitVal	Description
31:0	Various	RW RES 0x0	Same as in Access Control Base 0 (Low)

Table 258: PCI Access Control Base 1 (High)
Offset:0x31E14

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as in Access Control Base 0 (High)

Table 259: PCI Access Control Size 1
Offset:0x31E18

Bits	Field	Type/ InitVal	Description
31:0	Various	RES RW 0x0	Same as in PCI Access Control Size 0

Table 260: PCI Access Control Base 2 (Low)
Offset:0x31E20

Bits	Field	Type/ InitVal	Description
31:0	Various	RW RES 0x0	Same as in Access Control Base 0 (Low)

Table 261: PCI Access Control Base 2 (High)
Offset:0x31E24

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as in Access Control Base 0 (High)

Table 262: PCI Access Control Size 2
Offset:0x31E28

Bits	Field	Type/ InitVal	Description
31:0	Various	RES RW 0x0	Same as in PCI Access Control Size 0

Table 263: PCI Access Control Base 3 (Low)
Offset:0x31E30

Bits	Field	Type/ InitVal	Description
31:0	Various	RW RES 0x0	Same as in Access Control Base 0 (Low)

Table 264: PCI Access Control Base 3 (High)
Offset:0x31E34

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as in Access Control Base 0 (High)

Table 265: PCI Access Control Size 3
Offset:0x31E38

Bits	Field	Type/ InitVal	Description
31:0	Various	RES RW 0x0	Same as in PCI Access Control Size 0

Table 266: PCI Access Control Base 4 (Low)
Offset:0x31E40

Bits	Field	Type/ InitVal	Description
31:0	Various	RW RES 0x0	Same as in Access Control Base 0 (Low)

Table 267: PCI Access Control Base 4 (High)
Offset:0x31E44

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as in Access Control Base 0 (High)

Table 268: PCI Access Control Size 4
Offset:0x31E48

Bits	Field	Type/ InitVal	Description
31:0	Various	RES RW 0x0	Same as in PCI Access Control Size 0

Table 269: PCI Access Control Base 5 (Low)
Offset:0x31E50

Bits	Field	Type/ InitVal	Description
31:0	Various	RW RES 0x0	Same as in Access Control Base 0 (Low)

Table 270: PCI Access Control Base 5 (High)
Offset:0x31E54

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as in Access Control Base 0 (High)

Table 271: PCI Access Control Size 5
Offset:0x31E58

Bits	Field	Type/ InitVal	Description
31:0	Various	RES RW 0x0	Same as in PCI Access Control Size 0

A.7.3 PCI Configuration Access Registers

Note

PCI Configuration Address, PCI Configuration Data, and PCI Interrupt Acknowledge registers are only accessible by CPU. They must not be accessed from PCI.

Table 272: PCI Configuration Address
Offset:0x30C78

NOTE: This register is also accessible via the a PCI slave read and write transaction.

Bits	Field	Type/ InitVal	Description
1:0	Reserved	RES 0x0	Read Only.
7:2	RegNum	RW 0x0	Register number
10:8	FunctNum	RW 0x0	Description number
15:11	DevNum	RW 0x0	Device number
23:16	BusNum	RW 0x0	Bus number
30:24	Reserved	RES 0x0	Read Only.
31	ConfigEn	RW 0x0	When set, an access to the Configuration Data register is translated into a Configuration or Special cycle on the PCI bus.

Table 273: PCI Configuration Data
Offset:0x30C7C

Bits	Field	Type/ InitVal	Description
31:0	ConfigData	RW 0x0	The data is transferred to/from the PCI bus when the CPU accesses this register and the ConfigEn bit in the Configuration Address register is set. A CPU access to this register causes the 88F5182 to perform a Configuration or Special cycle on the PCI bus.

Table 274: PCI Interrupt Acknowledge
Offset:0x30C34

Bits	Field	Type/ InitVal	Description
31:0	IntAck	RO 0x0	A CPU read access to this register forces an interrupt acknowledge cycle on the PCI bus.

A.7.4 PCI Error Report Registers

Table 275: PCI SERRn Mask
Offset:0x30C28

NOTE: 88F5182 only asserts PCI_SERRn if the PCI Status and Command register's <SErrEn> bit [8] is enabled, see Table 282 on page 344.

Bits	Field	Type/ InitVal	Description
0	DPErr	RW 0x0	If set to 1, asserts PCI_SERRn upon internal data path error detection in the PCI interface
1	SWrPerr	RW 0x0	If set to 1, asserts PCI_SERRn upon PCI slave detection of bad write data parity.
2	SRdPerr	RW 0x0	If set to 1, asserts PCI_SERRn upon a PCI_PERRn response to read data driven by the PCI slave.
4:3	Reserved	RES 0x0	Reserved
5	MWrPerr	RW 0x0	If set to 1, asserts PCI_SERRn upon a PCI_PERRn response to write data driven by the PCI master.
6	MRdPerr	RW 0x0	If set to 1, asserts PCI_SERRn upon a bad data parity detection during a PCI master read transaction or upon bad parity detection during split completion to a write transaction initiated by the master.
7	Reserved	RES 0x0	Reserved
8	MMabort	RW 0x0	If set to 1, asserts PCI_SERRn upon a PCI master generation of master abort.
9	MTabort	RW 0x0	If set to 1, asserts PCI_SERRn upon a PCI master detection of target abort.
10	MDis	RW 0x0	If set to 1, assert PCI_SERRn upon an attempt to generate a PCI transaction while master is disabled.
11	MRetry	RW 0x0	If set to 1, asserts PCI_SERRn upon a PCI master reaching retry counter limit.
16:12	Reserved	RES 0x0	Reserved
17	STabort	RW 0x0	If set to 1, asserts PCI_SERRn upon a PCI slave generate Target Abort.
19:18	Reserved	RES 0x0	Reserved
20	SRdBuf	RW 0x0	If set to 1, asserts PCI_SERRn if the PCI slave's read buffer, discard timer expires.
21	Arb	RW 0x0	If set to 1, asserts PCI_SERRn upon the internal PCI arbiter detection of a *broken* PCI master.
31:22	Reserved	RES 0x0	Reserved

Note Error Address is not latched with the following interrupt events: <MDis> bit [10], <SRdBuf> bit [20], <Arb> bit [21], <BIST> bit [24], <PMG> bit [25], and <PRST> bit [26] (see Table 275, PCI SERRn Mask, on page 341).

Table 276: PCI Interrupt Cause
Offset:0x31D58

NOTE: All bits are Read/Write Clear only. A cause bit sets upon error event occurrence. A write of 0 clears the bit. A write of 1 has no affect.

Bits	Field	Type/ InitVal	Description
23:0	Cause	RWC 0x0	Cause bit per event, as described in the SERRn Mask register.
24	BIST	RWC 0x0	PCI BIST Interrupt
25	PMG	RWC 0x0	PCI Power Management Interrupt
26	PRST	RWC 0x0	PCI Reset Assert
31:27	Sel	RWC 0x0	Specifies the error event currently being reported in the Error Address registers: 0x0 = Reserved 0x1 = SWrPerr 0x2 = SRdPerr 0x3 = Reserved 0x4 = Reserved 0x5 = MWrPerr 0x6 = MRdPerr 0x7 = Reserved 0x8 = MMabort 0x9 = MTabort 0xA = Reserved 0xB = MRetry 0xC = Reserved 0xD = Reserved 0xE = Reserved 0xF = Reserved 0x10 = Reserved 0x11 = STabort 0x12 = Reserved 0x13 = Reserved 0x14 = Reserved 0x15 = Reserved 0x16 = Reserved 0x17 = Reserved 0x18–0x1F = Reserved

Table 277: PCI Interrupt Mask
Offset:0x31D5C

Bits	Field	Type/ InitVal	Description
26:0	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 interrupt.
31:27	Reserved	RES 0x0	Reserved

Table 278: PCI Error Address (Low)
Offset:0x31D40

Bits	Field	Type/ InitVal	Description
31:0	ErrAddr	RO 0x0	PCI address bits [31:0] are latched as a result of an error latched in the Interrupt Cause Register. Upon address latched, no new address can be registered (due to another error) until the register is being read. An error masked in the Interrupt Mask Register will not set this register. In the case of a split completion by an external completer with any error, the PCI address error samples the address of split completion. **NOTE:** Do not compare the four least significant bits (of the seven address bits) of the completion address to the original crossbar request address. The master always issues transaction on the PCI with address aligned to 32-bits. In case the internal completer initiates a split completion and detects an error, the unit samples and locks the split completion address and not the original address.

Table 279: PCI Error Address (High)
Offset:0x31D44

Bits	Field	Type/ InitVal	Description
31:0	ErrAddr	RO 0x0	PCI address bits [63:32] are latched as a result of an error latched in the Interrupt Cause Register. Upon address latched, no new address can be registered (due to another error) until the Address Low register is being read. An error masked in the Interrupt Mask Register will not set this register. **NOTE:** Applicable only when running DAC cycle.

Table 280: PCI Error Command
Offset:0x31D50

Bits	Field	Type/ InitVal	Description
3:0	ErrCmd	RO 0x0	PCI Command bits [3:0] are latched as a result of an error latched in the Interrupt Cause Register. When latched, no new command can be registered (due to another error) until the Address Low register is being read. An error masked in the Interrupt Mask Register will not set this register.
4	DAC	RO 0x0	If set to 1, indicates that the transaction latched in the error registers is a DAC transaction (meaning the address is composed of Error Address Low and High registers).
31:5	Reserved	RES 0x0	Reserved

A.7.5 Function 0 Configuration Registers

Table 281: PCI Device and Vendor ID
Offset:0x00

Bits	Field	Type/ InitVal	Description
15:0	VenID	RW 0x11AB	Marvell's Vendor ID. Read only from PCI.
31:16	DevID	RW 0x5182	88F5182 Device ID. Read only from PCI.

Table 282: PCI Status and Command
Offset:0x04

Bits	Field	Type/ InitVal	Description
0	IOEn	RW 0x0	Controls the 88F5182's ability to response to PCI I/O accesses. 0 = Disable 1 = Enable
1	MEMEn	RW 0x0	Controls the 88F5182's ability to response to PCI Memory accesses. 0 = Disable 1 = Enable
2	MasEn	RW 0x0	Controls the 88F5182's ability to act as a master on the PCI bus. 0 = Disable 1 = Enable
3	SpecialEn	RO 0x0	Controls the 88F5182's ability to respond to PCI special cycles. Read only 0 (88F5182 PCI slave does not support special cycles).

Table 282: PCI Status and Command (Continued)
Offset:0x04

Bits	Field	Type/ InitVal	Description
4	MemWrInv	RW 0x0	Controls the 88F5182's ability to generate memory write and invalidate commands on the PCI bus. 0 = Disable 1 = Enable
5	VGA	RO 0x0	VGA Palette Snoops Not supported. Read only 0.
6	PErrEn	RW 0x0	Controls the 88F5182's ability to respond to parity errors on the PCI. If this bit is disable the 88F5182 ignores any Parity Errors. 0 = Disable 1 = Enable **NOTE:** The 88F5182 asserts PCI_PERRn only when detects data parity error.
7	AddrStep	RW 0x0	Address Stepping Enable The 88F5182 PCI master performs address stepping only on configuration accesses. **NOTE:** Read only from the PCI.
8	SErrEn	RW 0x0	Controls the 88F5182's ability to assert the PCI_SERRn pin. 0 = Disable 1 = Enable
9	FastBTBEn	RW 0x0	Controls the 88F5182's ability to generate fast back-to-back transactions. 0 = Disable 1 = Enable
19:10	Reserved	RO 0x0	Read only.
20	CapList	RW 0x1	Capability List Support Indicates that the 88F5182 configuration header includes capability list. **NOTE:** Read only from the PCI.
21	66MHzEn	RW 0x1	66 MHz Capable Indicates that the 88F5182 PCI interface is capable of running at 66 MHz **NOTE:** Read only from PCI.
22	Reserved	RES 0x0	Read only.
23	TarFastBB	RW 0x1	Indicates that the 88F5182 is capable of accepting fast back-to-back transactions on the PCI bus. **NOTE:** Read only from the PCI.
24	DataPerr	RWC 0x0	Set by the 88F5182 Master when detects or generates Perr. Clear only by writing 1.
26:25	DevSelTim	RW 0x1	Indicates the 88F5182's DEVSEL timing (medium). **NOTE:** Read only from the PCI.
27	SlaveTabort	RWC 0x0	Set when the 88F5182's slave terminates a transaction with Target Abort. Clear only by writing 1.
28	MasterTabort	RWC 0x0	Set when the 88F5182's master detects a Target Abort termination. Clear only by writing 1.

Table 282: PCI Status and Command (Continued)
Offset:0x04

Bits	Field	Type/ InitVal	Description
29	MAbort	RWC 0x0	Set when the 88F5182's master generates a Master Abort (except of special cycle). Clear only by writing 1.
30	SysErr	RWC 0x0	Set when the 88F5182 asserts PCI_SERRn. Clear only by writing 1.
31	DetParErr	RWC 0x0	Set upon the 88F5182 detection of Parity error (both as master and slave). Clear only by writing 1.

Table 283: PCI Class Code and Revision ID
Offset:0x08

Bits	Field	Type/ InitVal	Description
7:0	RevID	RW *Rev. A1:* 0x1 *Rev. A2:* 0x2	Indicates the Revision number. **NOTE:** Read only from PCI.
15:8	Reserved	RO 0x0	Read only.
23:16	SubClass	RW 0x80	Indicates the Subclass. **NOTE:** Read only from PCI.
31:24	BaseClass	RW 0x05	Indicates the Base Class. **NOTE:** Read only from PCI.

Table 284: PCI BIST, Header Type/Initial Value, Latency Timer, and Cache Line
Offset:0x0C

Bits	Field	Type/ InitVal	Description
7:0	CacheLine	RW 0x00	Specifies the 88F5182's cache line size.
15:8	LatTimer	RW 0x0	Specifies in units of PCI clock cycles the latency timer value of the 88F5182. Used by the PCI master when acting as a requester.
23:16	HeadType/ InitVal	RW 0x80	Specifies Configuration Header Type/ Initial Value **NOTE:** Read only from PCI.
27:24	BISTComp	RW 0x0	BIST Completion Code Written by the CPU upon BIST completion. **NOTE:** Read only from PCI.
29:28	Reserved	RES 0x0	Reserved

Table 284: PCI BIST, Header Type/Initial Value, Latency Timer, and Cache Line (Continued)
Offset:0x0C

Bits	Field	Type/ InitVal	Description
30	BISTAct	RW 0x0	BIST Activate bit Set to 1 by PCI to activate BIST. Cleared by CPU upon BIST completion.
31	BISTCap	RW 0x1	BIST Capable Bit **NOTE:** Read Only from PCI.

Table 285: PCI CSn[0] Base Address (Low)
Offset:0x10

Bits	Field	Type/ InitVal	Description
0	MemSpace	RO 0x0	Memory Space Indicator
2:1	Type/ InitVal	RW 0x2	BAR Type/Initial Value Located anywhere in 64-bit address space. **NOTE:** Read only from PCI.
3	Prefetch	RW 0x1	Prefetch Enable **NOTE:** Read only from PCI.
11:4	Reserved	RES 0x0	Read only.
31:12	Base	RW 0x00000	Base address. Corresponds to address bits [31:12].

Table 286: PCI CSn[0] Base Address (High)
Offset:0x14

Bits	Field	Type/ InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits [63:32].

Table 287: PCI CSn[1] Base Address (Low)
Offset:0x18

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x10000000	Same as CSn[0] Base Address (Low)

Table 288: PCI CSn[1] Base Address (High)
Offset:0x1C

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0] Base Address (High)

Table 289: PCI Internal Registers Memory Mapped Base Address (Low)
Offset:0x20

Bits	Field	Type/ InitVal	Description
0	MemSpace	RO 0x0	Memory Space Indicator
2:1	Type/ InitVal	RW 0x2	BAR Type/Initial Value Located anywhere in 64-bit address space. **NOTE:** Read only from PCI.
3	Prefetch	RW 0x0	Prefetch Enable **NOTE:** Read only from PCI.
15:4	Reserved	RES 0x0	Read only.
31:16	Base	RW 0xD0000	Base Address Corresponds to address bits [31:16]

Table 290: PCI Internal Registers Memory Mapped Base Address (High)
Offset:0x24

Bits	Field	Type/ InitVal	Description
31:0	Base	RW 0x0	Base address Corresponds to address bits [63:32]

Table 291: PCI Subsystem Device and Vendor ID
Offset:0x2C

Bits	Field	Type/ InitVal	Description
15:0	VenID	RW 0x0	Subsystem Manufacturer ID Number **NOTE:** Read only from PCI.
31:16	DevID	RW 0x0	Subsystem Device ID Number **NOTE:** Read only from PCI.

Table 292: PCI Expansion ROM Base Address Register
Offset:0x30

NOTE: If Expansion ROM is not enabled (PCI Mode register), this register is Reserved (Read Only 0)

Bits	Field	Type/ InitVal	Description
0	ExpROMEn	RW 0x0	Expansion ROM Enable 0 = Disable 1 = Enable
11:1	Reserved	RES 0x0	Reserved
31:12	Base	RW 0xE0000	Expansion ROM Base Address

Table 293: PCI Capability List Pointer Register
Offset:0x34

Bits	Field	Type/ InitVal	Description
7:0	CapPtr	RW 0x40	Capability List Pointer **NOTE:** Read only from PCI.
31:8	Reserved	RES 0x0	Reserved

Table 294: PCI Interrupt Pin and Line
Offset:0x3C

Bits	Field	Type/ InitVal	Description
7:0	IntLine	RW 0x0	Provides interrupt line routing information.
15:8	IntPin	RW 0x1	Indicates which interrupt pin is used by the 88F5182. **NOTE:** Read only from PCI.
31:16	Reserved	RES 0x0	Read only.

Table 295: PCI Power Management
Offset:0x40

Bits	Field	Type/ InitVal	Description
7:0	CapID	RW 0x1	Capability ID **NOTE:** Read only from PCI.

Table 295: PCI Power Management (Continued)
Offset:0x40

Bits	Field	Type/ InitVal	Description
15:8	NextPtr	RW 0x48	Next Item Pointer **NOTE:** Read only from PCI.
18:16	PMCVer	RW 0x2	PCI Power Management Spec Revision **NOTE:** Read only from PCI.
19	PMEClk	RW 0x1	PME Clock Indicates that the PCI clock is required for the 88F5182 to assert PCI_PMEn **NOTE:** Read only from PCI.
20	Reserved	RES 0x0	Read only from PCI
21	DSI	RW 0x0	Device Specific Initialization **NOTE:** Read only from PCI.
24:22	AuxCur	RW 0x0	Auxiliary Current Requirements **NOTE:** Read only from PCI.
25	D1Sup	RW 0x1	D1 Power Management State Support 0 = Not supported 1 = Supported **NOTE:** Read only from PCI.
26	D2Sup	RW 0x1	D2 Power Management State Support 0 = Not supported 1 = Supported **NOTE:** Read only from PCI.
31:27	PMESup	RW 0xf	PCI_PMEn Signal Support Indicates in which power states the 88F5182 supports the PCI_PMEn pin. Each bit corresponds to different state (bit [0]—D0, bit [1]—D1, bit [2]—D2, bit [3]—D3-hot, bit [4]—D3-cold). For example, ‘b01001 stands for supporting PCI_PMEn only on D0 and D3-hot states. **NOTE:** Read only from PCI.

Table 296: PCI Power Management Control and Status
Offset:0x44

Bits	Field	Type/ InitVal	Description
1:0	PState	RW 0x0	Power State 00 = D0 01 = D1 10 = D2 11 = D3-hot
7:2	Reserved	RW 0x0	Read only
8	PME_EN	RW 0x0	PCI_PMEn Pin Assertion Enable

Table 296: PCI Power Management Control and Status (Continued)
Offset:0x44

Bits	Field	Type/ InitVal	Description
12:9	DSel	RW 0x0	Data Select
14:13	DScale	RW 0x0	Data Scale **NOTE:** Read only from PCI.
15	PME_Stat	RW 0x0	PCI_PMEn Pin Status CPU set only by writing 1. PCI clear only by writing 1. When set to 1, the 88F5182 asserts PCI_PMEn pin.
23:16	P2P	RW 0x0	Power Management Status and Control for P2P Bridge **NOTE:** Read only from PCI.
31:24	Data	RW 0x0	State Data **NOTE:** Read only from PCI.

Table 297: PCI VPD Address
Offset:0x48

Bits	Field	Type/ InitVal	Description
7:0	CapID	RW 0x3	Capability ID **NOTE:** Read only from PCI.
15:8	NextPtr	RW 0x50	Next Item Pointer **NOTE:** Read only from PCI.
30:16	Addr	RW 0x0	VPD Address Points to the location of the VPD structure in memory. **NOTE:** The 88F5182 also implements remapping of the high address bits through the PCI Address Decoding Control register.
31	Flag	RW 0x0	Flag Flipped by System or 88F5182 during VPD Access On VPD writes, system sets the flag to 1 indicating VPD write is required. The 88F5182 clears the flag to indicate that the VPD write is done (data from the VPD Data register was written to memory). On VPD reads, the system sets the flag to 0, indicating VPD read is required. The 88F5182 sets the flag to 1 when the read is done (data has been read from memory and put in VPD Data register).

Table 298: PCI VPD Data
Offset:0x4C

Bits	Field	Type/ InitVal	Description
31:0	Data	RW 0x0	VPD Data

Table 299: PCI MSI Message Control
Offset:0x50

Bits	Field	Type/ InitVal	Description
7:0	CapID	RW 0x5	Capability ID **NOTE:** Read only from PCI.
15:8	NextPtr	RW 0x68	Next Item Pointer **NOTE:** Read only from PCI.
16	MSIEn	RW 0x0	MSI Enable 0 = Disable The 88F5182 generates a PCI interrupt. 1 = Enabled The 88F5182 generates MSI messages instead of interrupts.
19:17	MultiCap	RW 0x0	Multiple Messages Capable The 88F5182 is capable of driving a single message. **NOTE:** Read only from PCI.
22:20	MultiEn	RW 0x0	Multiple Messages Enable The number of messages the system allocates to the 88F5182 (must be smaller or equal to MultiCap).
23	Addr64	RW 0x1	64-bit Addressing Capable Indicates whether the 88F5182 is capable of generating 64-bit message address. Read only from PCI. 0 = Not capable 1 = Capable
31:24	Reserved	RO 0x0	Read only 0.

Table 300: PCI MSI Message Address
Offset:0x54

Bits	Field	Type/ InitVal	Description
31:0	Addr	RW 0x0	Message Address

Table 301: PCI MSI Message Upper Address
Offset:0x58

Bits	Field	Type/ InitVal	Description
31:0	Addr	RW 0x0	Message Upper Address 32 upper address bits. If set to a value other than 0, the 88F5182 issues MSI message as DAC cycle.

Table 302: PCI Message Data
Offset:0x5C

Bits	Field	Type/ InitVal	Description
15:0	Data	RW 0x0	Message Data Initiated by the system.
31:16	Reserved	RES 0x0	Read only 0.

Table 303: CompactPCI HotSwap
Offset:0x68

Bits	Field	Type/ InitVal	Description
7:0	CapID	RW 0x6	Capability ID **NOTE:** Read only from PCI.
15:8	NextPtr	RO 0x0	Next Item Pointer **NOTE:** Read only from PCI. This is a null pointer
16	Reserved	RES 0x0	Read only 0.
17	EIM	RW 0x0	PCI_ENUMn Interrupt Mask 0 = Enable signal 1 = Mask signal
18	Reserved	RES 0x0	Read only 0.
19	LOO	RW 0x0	LED On/Off 0 = LED off 1 = LED on
21:20	Reserved	RES 0x0	Read only 0.
22	Ext	RWC 0x0	Extraction Indicates that the board is about to be extracted (set to 1). **NOTE:** Write 1 to clear.
23	Ins	RWC 0x0	Insertion Indicates that the board has just been inserted (set to 1). **NOTE:** Write 1 to clear.
31:24	Reserved	RES 0x0	Read only 0.

A.7.6 Function 1 Configuration Registers

Table 304: PCI CSn[2] Base Address (Low)
Offset:0x10

Bits	Field	Type/ InitVal	Description
31:0	Various	RO 0x0	Same as CSn[0] Base Address (Low) See Table 285, PCI CSn[0] Base Address (Low), on page 347.

Table 305: PCI CSn[2] Base Address (High)
Offset:0x14

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0] Base Address (High) See Table 286, PCI CSn[0] Base Address (High), on page 347.

Table 306: PCI CSn[3] Base Address (Low)
Offset:0x18

Bits	Field	Type/ InitVal	Description
31:12	Various	RO 0x0	Same as CSn[0] Base Address (Low)

Table 307: PCI CSn[3] Base Address (High)
Offset:0x1C

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0]Base Address (High)

A.7.7 Function 2 Configuration Registers

Table 308: PCI DevCS[0] Base Address (Low)
Offset:0x10

Bits	Field	Type/ InitVal	Description
0	MemSpace	RO 0x0	Memory Space Indicator
2:1	Type/ InitVal	RW 0x2	BAR Type/Initial Value Located anywhere in 64-bit address space. **NOTE:** Read only from PCI.

Table 308: PCI DevCS[0] Base Address (Low) (Continued)
Offset:0x10

Bits	Field	Type/ InitVal	Description
3	Prefetch	RW 0x0	Prefetch Enable **NOTE:** Read only from PCI.
11:4	Reserved	RES 0x0	Read only.
31:12	Various	RW 0xE0000	Same as CSn[0] Base Address (Low)

Table 309: PCI DevCSn[0] Base Address (High)
Offset:0x14

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0] Base Address (High)

Table 310: PCI DevCSn[1] Base Address (Low)
Offset:0x18

Bits	Field	Type/ InitVal	Description
31:0	Various	0x0	Same as DevCSn[0] Base Address (Low)

Table 311: PCI DevCSn[1] Base Address (High)
Offset:0x1C

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0] Base Address (High)

Table 312: PCI DevCSn[2] Base Address (Low)
Offset:0x20

Bits	Field	Type/ InitVal	Description
31:0	Various	0x0	Same as DevCSn[0] Base Address (Low)

Table 313: PCI DevCSn[2] Base Address (High)
Offset:0x24

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0] Base Address (High)

Table 314: PCI BootCS Base Address (Low)
Offset:0x18

Bits	Field	Type/ InitVal	Description
31:0	Various	0xF8000000	Same as DevCSn[0] Base Address (Low) See Table 308, PCI DevCS[0] Base Address (Low), on page 354.

Table 315: PCI BootCSn Base Address (High)
Offset:0x1C

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0] Base Address (High).

Table 316: PCI P2P Mem0 Base Address (Low)
Offset:0x10

Bits	Field	Type/ InitVal	Description
31:0	Various	0x80000000	Same as CSn[0] Base Address (Low) See Table 285, PCI CSn[0] Base Address (Low), on page 347.

Table 317: PCI P2P Mem0 Base Address (High)
Offset:0x14

Bits	Field	Type/ InitVal	Description
31:0	Various	RW 0x0	Same as CSn[0] Base Address (High)

Table 318: PCI P2P I/O Base Address
Offset:0x20

Bits	Field	Type/ InitVal	Description
0	IOSpace	RO 0x1	I/O Space Indicator
11:1	Reserved	RO 0x0	Read only.
31:12	Various	RW 0xC0000	Same as CSn[0]Base Address (Low)

Table 319: PCI Internal Registers I/O Mapped Base Address
Offset:0x24

Bits	Field	Type/ InitVal	Description
0	IOSpace	RO 0x1	I/O Space Indicator
11:1	Reserved	RO 0x0	Read only.
31:12	Various	RW 0xD0000	Same as CSn[0] Base Address

A.8 Serial-ATA Host Controller (SATAHC) Registers

A.8.1 SATAHC Address Space

The SATAHC contains two SATA ports. In the address space of the SATAHC, three regions are allocated, one region per port and an additional region for the common registers. See Figure 61 and Table 320, p.358. Reading from non existing address space results in reading a data of 0.

Figure 61: SATAHC Address Space

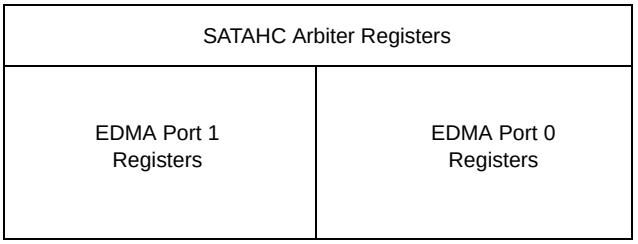


Table 320: SATAHC Address Space

Offset	Target
80000–81FFF	SATAHC Arbiter Registers
82000–83FFF	EDMA Port 0 Registers
84000–85FFF	EDMA Port 1 Registers

A.8.2 SATAHC Arbiter Registers Map

Table 321: Serial-ATA Host Controller (SATAHC) Registers Map

Register	Offset	Table, Page
SATAHC Arbiter Registers		
SATAHC Configuration Register	0x80000	Table 323, p. 362
SATAHC Request Queue Out-Pointer Register	0x80004	Table 324, p. 363
SATAHC Response Queue In-Pointer Register	0x80008	Table 325, p. 363
SATAHC Interrupt Coalescing Threshold Register	0x8000C	Table 326, p. 364
SATAHC Interrupt Time Threshold Register	0x80010	Table 327, p. 364
SATAHC Interrupt Cause Register	0x80014	Table 328, p. 365
Reserved Register	0x80018	Table 329, p. 366
SATAHC Main Interrupt Cause Register	0x80020	Table 330, p. 366
SATAHC Main Interrupt Mask Register	0x80024	Table 331, p. 367
SATAHC LED Configuration Register	0x8002C	Table 332, p. 367
Window0 Control Register	0x80030	Table 333, p. 368
Window0 Base Register	0x80034	Table 334, p. 368

Table 321: Serial-ATA Host Controller (SATAHC) Registers Map (Continued)

Register	Offset	Table, Page
Window1 Control Register	0x80040	Table 335, p. 369
Window1 Base Register	0x80044	Table 336, p. 369
Window2 Control Register	0x80050	Table 337, p. 369
Window2 Base Register	0x80054	Table 338, p. 370
Window3 Control Register	0x80060	Table 339, p. 370
Window3 Base Register	0x80064	Table 340, p. 370
EDMA Registers		
EDMA Configuration Register	Port 0: 0x82000, Port 1: 0x84000	Table 341, p. 371
EDMA Timer Register	Port 0: 0x82000, Port 1: 0x84000	Table 342, p. 374
EDMA Interrupt Error Cause Register	Port 0: 0x82004, Port 1: 0x84004	Table 343, p. 374
EDMA Interrupt Error Mask Register	Port 0: 0x82008, Port 1: 0x84008	Table 344, p. 377
EDMA Request Queue Base Address High Register	Port 0: 0x82010, Port 1: 0x84010	Table 345, p. 377
EDMA Request Queue In-Pointer Register	Port 0: 0x82014, Port 1: 0x84014	Table 346, p. 377
EDMA Request Queue Out-Pointer Register	Port 0: 0x82018, Port 1: 0x84018	Table 347, p. 378
EDMA Response Queue Base Address High Register	Port 0: 0x8201C, Port 1: 0x8401C	Table 348, p. 378
EDMA Response Queue In-Pointer Register	Port 0: 0x82020, Port 1: 0x84020	Table 349, p. 378
EDMA Response Queue Out-Pointer Register	Port 0: 0x82024, Port 1: 0x84024	Table 350, p. 379
EDMA Command Register	Port 0: 0x82028, Port 1: 0x84028	Table 351, p. 379
EDMA Test Control Register	Port 0: 0x8202C, Port 1: 0x8402C	Table 352, p. 381
EDMA Status Register	Port 0: 0x82030, Port 1: 0x84030	Table 353, p. 381
EDMA IORdy Timeout Register	Port 0: 0x82034, Port 1: 0x84034	Table 354, p. 382
EDMA Command Delay Threshold Register	Port 0: 0x82040, Port 1: 0x84040	Table 355, p. 382
EDMA Halt Conditions Register	Port 0: 0x82060, Port 1: 0x84060	Table 356, p. 383
EDMA NCQ0 Done/TCQ0 Outstanding Status Register	Port 0: 0x82094, Port 1: 0x84094	Table 357, p. 383
EDMA NCQ1 Done/TCQ1 Outstanding Status Register	Port 0: 0x82098, Port 1: 0x84098	Table 358, p. 384
EDMA NCQ2 Done/TCQ2 Outstanding Status Register	Port 0: 0x8209C, Port 1: 0x8409C	Table 359, p. 384

Table 321: Serial-ATA Host Controller (SATAHC) Registers Map (Continued)

Register	Offset	Table, Page
EDMA NCQ3 Done/TCQ3 Outstanding Status Register	Port 0: 0x820A0, Port 1: 0x840A0	Table 360, p. 384
Basic DMA Registers		
Basic DMA Command Register	Port 0: 0x82224, Port 1: 0x84224	Table 361, p. 384
Basic DMA Status Register	Port 0: 0x82228, Port 1: 0x84228	Table 362, p. 386
Descriptor Table Low Base Address Register	Port 0: 0x8222C, Port 1: 0x8422C	Table 363, p. 387
Descriptor Table High Base Address Register	Port 0: 0x82230, Port 1: 0x84230	Table 364, p. 387
Data Region Low Address Register	Port 0: 0x82234, Port 1: 0x84234	Table 365, p. 387
Data Region High Address Register	Port 0: 0x82238, Port 1: 0x84238	Table 366, p. 388
Serial-ATA Interface Registers		
SStatus Register	Port 0: 0x82300, Port 1: 0x84300	Table 367, p. 388
SError Register	Port 0: 0x82304, Port 1: 0x84304	Table 368, p. 389
SError Interrupt Mask Register	Port 0: 0x82340, Port 1: 0x84340	Table 369, p. 390
SControl Register	Port 0: 0x82308, Port 1: 0x84308	Table 370, p. 390
LTMode Register	Port 0: 0x8230C, Port 1: 0x8430C	Table 371, p. 391
PHY Mode 3 Register	Port 0: 0x82310, Port 1: 0x84310	Table 372, p. 392
PHY Mode 4 Register	Port 0: 0x82314, Port 1: 0x84314	Table 373, p. 393
PHY Mode 1 Register	Port 0: 0x8232C, Port 1: 0x8432C	Table 374, p. 393
PHY Mode 2 Register	Port 0: 0x82330, Port 1: 0x84330	Table 375, p. 394
BIST Control Register	Port 0: 0x82334, Port 1: 0x84334	Table 376, p. 395
BIST-DW1 Register	Port 0: 0x82338, Port 1: 0x84338	Table 377, p. 395
BIST-DW2 Register	Port 0: 0x8233C, Port 1: 0x8433C	Table 378, p. 396
Serial-ATA Interface Configuration Register	Port 0: 0x82050, Port 1: 0x84050	Table 379, p. 396
Serial-ATA Interface Control Register	Port 0: 0x82344, Port 1: 0x84344,	Table 380, p. 398
Serial-ATA Interface Test Control Register	Port 0: 0x82348, Port 1: 0x84348	Table 381, p. 399

Table 321: Serial-ATA Host Controller (SATAHC) Registers Map (Continued)

Register	Offset	Table, Page
Serial-ATA Interface Status Register	Port 0: 0x8234C, Port 1: 0x8434C	Table 382, p. 400
Vendor Unique Register	Port 0: 0x8235C, Port 1: 0x8435C	Table 383, p. 402
FIS Configuration Register	Port 0: 0x82360, Port 1: 0x84360	Table 384, p. 403
FIS Interrupt Cause Register	Port 0: 0x82364, Port 1: 0x84364	Table 385, p. 404
FIS Interrupt Mask Register	Port 0: 0x82368, Port 1: 0x84368	Table 386, p. 406
FIS DW0 Register	Port 0: 0x82370, Port 1: 0x84370	Table 387, p. 406
FIS DW1 Register	Port 0: 0x82374, Port 1: 0x84374	Table 388, p. 406
FIS DW2 Register	Port 0: 0x82378, Port 1: 0x84378	Table 389, p. 406
FIS DW3 Register	Port 0: 0x8237C, Port 1: 0x8437C	Table 390, p. 407
FIS DW4 Register	Port 0: 0x82380, Port 1: 0x84380	Table 391, p. 407
FIS DW5 Register	Port 0: 0x82384, Port 1: 0x84384	Table 392, p. 407
FIS DW6 Register	Port 0: 0x82388, Port 1: 0x84388	Table 393, p. 407

A.8.3 Shadow Registers Block Register Map

Table 322: Shadow Register Block Registers Map
NOTE: See ATA/ATAPI6 Specification

Bits 31 -24	Bits 23-16	Bits 15-8	Bits 7-0	Offset
Reserved	Reserved	Device PIO Data register		Port 0: 0x82100, Port 1: 0x84100
Reserved	Reserved	Reserved	Device - Features (Features Current)/Error register	Port 0: 0x82104, Port 1: 0x84104
Reserved	Reserved	Reserved	Device - Sector Count (Sector Count Current) register	Port 0: 0x82108, Port 1: 0x84108
Reserved	Reserved	Reserved	Device - LBA Low register	Port 0: 0x8210C, Port 1: 0x8410C
Reserved	Reserved	Reserved	Device - LBA Mid register	Port 0: 0x82110, Port 1: 0x84110
Reserved	Reserved	Reserved	Device - LBA High register	Port 0: 0x82114, Port 1: 0x84114
Reserved	Reserved	Reserved	Device - Device/Head (Device) register	Port 0: 0x82118, Port 1: 0x84118
Reserved	Reserved	Reserved	Device - Command/Status register	Port 0: 0x8211C, Port 1: 0x8411C
Reserved	Reserved	Reserved	Device - Control/ Alternate Status register	Port 0: 0x82120, Port 1: 0x84120

A.8.4 SATAHC Arbiter Registers

Table 323: SATAHC Configuration Register
Offset: 0x80000

Bits	Field	Type/InitVal	Description
7:0	Timeout	RW 0xFF	SATAHC interface Crossbar Arbiter Timeout Preset Value
15:8	Reserved	RES 0x0	Reserved
16	TimeoutEn	RW 0x1	Crossbar Arbiter Timer Enable 0 = Enable 1 = Disable
23:17	Reserved	RES 0x0	Reserved

Table 323: SATAHC Configuration Register
Offset: 0x80000

Bits	Field	Type/InitVal	Description
25:24	CoalDis	RW 0x0	Coalescing Disable When a bit in field <CoalDis> is set to 1, the completing indication of the corresponding port is ignored in the following coalescing counters: 1. Table 326, SATAHC Interrupt Coalescing Threshold Register, on page 364 2. Table 327, SATAHC Interrupt Time Threshold Register, on page 364 CoalDis[0] (Bit 24) 0 = Coalescing enabled for port 0 1 = Coalescing disable for port 0 CoalDis[1] (bit 25) 0 = Coalescing enabled for port 1 1 = Coalescing disable for port 1
31:26	Reserved	RES 0x0	Reserved

Table 324: SATAHC Request Queue Out-Pointer Register
Offset: 0x80004

Bits	Field	Type/InitVal	Description
6:0	eRQQOP0	RO 0x0	EDMA Request Queue Out-Pointer Port 0 This field reflects the value of bits [11:5] in the EDMA Request Queue Out-Pointer Register (see Table 347 on page 378) located in port 0.
7	Reserved	RES 0x0	Reserved
14:8	eRQQOP1	RO 0x0	EDMA Request Queue Out-Pointer Port 1 This field reflects the value of bits [11:5] in the EDMA Request Queue Out-Pointer Register located in port 1.
15	Reserved	RES 0x0	Reserved
31:16	Reserved	RES 0x0	Reserved

Table 325: SATAHC Response Queue In-Pointer Register
Offset: 0x80008

Bits	Field	Type/InitVal	Description
6:0	eRPQIP0	RO 0x0	EDMA Response Queue In-Pointer Register Port 0 This field reflects the value of bits [9:3] in the EDMA Response Queue In-Pointer Register (see Table 349 on page 378) located in port 0.
7	Reserved	RES 0x0	Reserved

Table 325: SATAHC Response Queue In-Pointer Register (Continued)
Offset: 0x80008

Bits	Field	Type/Init Val	Description
14:8	eRPQIP1	RO 0x0	EDMA Response Queue In-Pointer Register Port 1 This field reflects the value of bits [9:3] in the EDMA Response Queue In-Pointer Register located in port 1.
15	Reserved	RES 0x0	Reserved
31:16	Reserved	RES 0x0	Reserved

Table 326: SATAHC Interrupt Coalescing Threshold Register
Offset: 0x8000C

Bits	Field	Type/Init Val	Description
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> field in the SATAHC Interrupt Cause Register (Table 328 p. 365). Once the accumulated number of SaCrpbXDone indications provided by both SATAHC ports reaches the SAICOALT value, <SaIntCoal> interrupt is asserted. When `SaIntCoal` is negated or when SATAHC Interrupt Coalescing Threshold Register is written, the interrupts counter is cleared. 0 = Assertion of SaCrpbXDone causes an immediate assertion of <SaIntCoal>. n = <SaIntCoal> assertion is provided for every n * SaCrpbXDone assertion.
31:8	Reserved	RES 0x0	Reserved

Table 327: SATAHC Interrupt Time Threshold Register
Offset: 0x80010

Bits	Field	Type/Init Val	Description
23:0	SAITMTH	RW 0x0	SATA Interrupt Time Threshold This field provides a way to ensure maximum delay between <SaCrpbXDone> assertion and assertion of <SaIntCoal> (even if the number of <SaCrpbXDone> 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 <SaCrpbXDone> indication. 0 = Assertion of <SaCrpbXDone> causes an immediate assertion of <SaIntCoal>. n = Up to n internal clocks between assertion of <SaCrpbXDone> and assertion of <SaIntCoal>.
31:24	Reserved	RES 0x0	Reserved

Table 328: SATAHC Interrupt Cause Register
Offset: 0x80014

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 affect.

Bits	Field	Type/Init Val	Description
0	SaCrpb0Done/ DMA0Done	RW0 0x0	SATA CRPB 0 Done / Basic DMA 0 Done When EDMA is enable: (The <eEnEDMA> field in the EDMA Command Register (Table 351 p. 379). This field is set when the EDMA in port 0 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: (Field eEnEDMA is cleared.) This field is set when the Basic DMA in port 0 completes the data transfer, clears field <BasicDMAActive>, and move to idle state. 0 = Basic DMA has not completed the data transfer. 1 = Basic DMA completed the data transfer.
1	SaCrpb1Done/ DMA1Done	RW0 0x0	SATA CRPB 1 Done / Basic DMA 1 Done When EDMA is enable: (Field <eEnEDMA> is set.) This field is set when the EDMA in port 1 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: (Field <eEnEDMA> is cleared.) This field is set when the Basic DMA in port 1 completes the data transfer, clears field <BasicDMAActive>, and moves to idle state. 0 = Basic DMA has not completed the data transfer. 1 = Basic DMA completed the data transfer.
3:2	Reserved	RES 0x0	Reserved
4	SaIntCoal	RW0 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 the setting of the <CoalDis> field in the SATAHC Configuration Register (Table 323 p. 363), 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 = Cause for Interrupt Coalescing did not occur. 1 = Cause for Interrupt Coalescing occurs.
7:5	Reserved	RES 0x0	Reserved

Table 328: SATAHC Interrupt Cause Register (Continued)
Offset: 0x80014

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 affect.

Bits	Field	Type/Init Val	Description
8	SaDevInterrupt0	RW0 0x0	SATA Device Interrupt Port 0 This bit is set if field <eEnEDMA> is cleared and the ATA interrupt line in port 0 is active. 0 = The ATA interrupt line in port 0 was not active. 1 = The ATA interrupt line in port 0 was active.
9	SaDevInterrupt1	RW0 0x0	SATA Device Interrupt Port 1 This bit is set if field <eEnEDMA> is cleared and the ATA interrupt line in port 1 is active. 0 = The ATA interrupt line in port 1 was not active. 1 = The ATA interrupt line in port 1 was active.
31:10	Reserved	RES 0x0	Reserved

Table 329: Reserved Register
Offset: 0x80018

Bits	Field	Type/Init Val	Description
31:0	Reserved	RW 0x0	Reserved

Table 330: SATAHC Main Interrupt Cause Register
Offset: 0x80020

NOTE: The bits in this register mirror the interrupt indications coming from the other SATAHC interrupt cause registers.

Bits	Field	Type/Init Val	Description
0	Sata0Err	RO 0x0	SATA port0 error
1	Sata0Done	RO 0x0	SATA port0 command done. This bit is set when one of the following occurs: • The <SaCrpb0Done/ DMA0Done> field in the SATAHC Interrupt Cause Register (Table 328 p. 365) of the SATAHC is set. • The <SaDevInterrupt0> of the same register of the SATAHC is set.
2	Sata1Err	RO 0x0	SATA port1 error
3	Sata1Done	RO 0x0	SATA port1 command done. This bit is set when one of the following occurs: • <SaCrpb1Done/ DMA1Done> of the SATAHC is set. • <SaDevInterrupt1> of the SATAHC is set.

Table 330: SATAHC Main Interrupt Cause Register (Continued)
Offset: 0x80020

NOTE: The bits in this register mirror the interrupt indications coming from the other SATAHC interrupt cause registers.

Bits	Field	Type/Init Val	Description
4	Sata0DmaDone	RO 0x0	SATA port0 DMA done This bit is set when both of the following occur: • <SaCrpb0Done/ DMA0Done> field in Table 328, SATAHC Interrupt Cause Register, on page 365 of SATAHC is set. • <SaDevInterrupt0> in Table 328, SATAHC Interrupt Cause Register, on page 365 of the SATAHC is set.
5	Sata1DmaDone	RO 0x0	SATA port1 DMA done This bit is set when both of the following occur: • <SaCrpb1Done/ DMA1Done> of SATAHC is set. • <SaDevInterrupt1> of the SATAHC is set.
6:7	Reserved	RO 0x0	Reserved
8	SataCoalDone	RO 0x0	SATA ports coalescing done. This bit is set when field <SaIntCoal> of SATAHC is set. **NOTE:** SATA completion fields <Sata0Done> and <Sata1Done> must be masked to use this bit.
31:9	Reserved	RO 0x0	Reserved

Table 331: SATAHC Main Interrupt Mask Register
Offset: 0x80024

Bits	Field	Type/Init Val	Description
31:0	Mask	RW 0x0	Mask bit per each cause bit. 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register.

Table 332: SATAHC LED Configuration Register
Offset: 0x8002C

Bits	Field	Type/Init Val	Description
0	act_led_blink	RW 0x0	Active LED Blink 0 = As is; Use indication as is. 1 = Blinking; Set indication to blinking.
1	Reserved	RSVD 0x0	Reserved

Table 332: SATAHC LED Configuration Register (Continued)
Offset: 0x8002C

Bits	Field	Type/Init Val	Description
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.
3	led_polarity	RW 0x0	LED Polarity 0 = Invert; Invert the active indication. 1 = No change; Do not change the active indication.
31:4	Reserved	RSVD 0x0	Reserved

Table 333: Window0 Control Register
Offset: 0x80030

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window 0 Enable 0x0 = Window is disabled. 0x1 = Window is enabled.
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See Section 2.10 "Default Address Map" on page 22. **NOTE:** Do not configure this field to the SDRAM Controller.
15:8	Attr	RW 0x0E	Specifies the target interface attributes associated with this window. See Section 2.10 "Default Address Map" on page 22.
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 KB granularity (e.g., a value of 0x00FF specifies 256 = 16 MB). **NOTE:** A value of 0x0 specifies 64-KB size.

Table 334: Window0 Base Register
Offset: 0x80034

Bits	Field	Type/InitVal	Description
15:0	Reserved	RES 0x0	Reserved
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].

Table 335: Window1 Control Register
Offset: 0x80040

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window1 Enable. See the Window0 Control Register (Table 333 p. 368).
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0.	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
15:8	Attr	RW 0x0D	Target specific attributes depending on the target interface. See the Window0 Control Register.
31:16	Size	RW 0x0FFF	Window Size See the Window0 Control Register.

Table 336: Window1 Base Register
Offset: 0x80044

Bits	Field	Type/InitVal	Description
15:0	Reserved	RES 0x0	Reserved
31:16	Base	RW 0x1000	Base Address See the Window0 Base Register.

Table 337: Window2 Control Register
Offset: 0x80050

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window2 Enable See the Window0 Control Register (Table 333 p. 368).
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
15:8	Attr	RW 0x0B	Target specific attributes depending on the target interface. See the Window0 Control Register.
31:16	Size	RW 0x0FFF	Window Size See the Window0 Control Register.

Table 338: Window2 Base Register
Offset: 0x80054

Bits	Field	Type/InitVal	Description
15:0	Reserved	RES 0x0	Reserved
31:16	Base	RW 0x2000	Base Address See the Window0 Base Register.

Table 339: Window3 Control Register
Offset: 0x80060

Bits	Field	Type/InitVal	Description
0	WinEn	RW 0x1	Window3 Enable See the Window0 Control Register (Table 333 p. 368).
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0	Specifies the unit ID (target interface) associated with this window. See the Window0 Control Register.
15:8	Attr	RW 0x07	Target specific attributes depending on the target interface. See the Window0 Control Register.
31:16	Size	RW 0x0FFF	Window Size See the Window0 Control Register.

Table 340: Window3 Base Register
Offset: 0x80064

Bits	Field	Type/InitVal	Description
15:0	Reserved	RES 0x0	Reserved
31:16	Base	RW 0x3000	Base Address See the Window0 Base Register.

A.8.5 EDMA Registers

Table 341: EDMA Configuration Register
Offset: Port 0: 0x82000, Port 1: 0x84000

Bits	Field	Type/InitVal	Description
4:0	Reserved	RW 0x1F	Reserved
5	eSATANatv CmdQue	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 = Native Command Queuing is not in use. 1 = Native Command Queuing is in use.
7:6	Reserved	RW 0x0	Reserved
8	eRdBSz	RW 0x0	EDMA Burst Size. This bit sets the maximum burst size initiated by the DMA to the Mbus. This bit is related to *read* operation. 0 = 128B 1 = Reserved **NOTE:** This field must be set to 0.
9	eQue	RW 0x0	EDMA Queued. When EDMA uses queued DMA commands, this bit must be set to 1. 0 = Non-Queued DMA commands are in use. 1 = Only Queued DMA commands are in use.
10	Reserved	RW 0x0	Reserved
11	eRdBSzExt	RO 0x0 RW 0x0	EDMA Burst Size Ext. This bit sets the maximum burst size initiated by the DMA to the Mbus. This bit is related to the *read* operation. 0 = EDMA Burst size is configured according to field <eRdBSz>. 1 = Reserved **NOTE:** This field must be set to 0.
12	Reserved	RW 0x1	Reserved This field must be set to 0x1.
13	eWrBufferLen	RW 0x0	EDMA Write Buffers Length. This bit defines the length of the write buffers. 0 = Write buffer is 256B. 1 = Reserved **NOTE:** This field must be set to 0. **NOTE:** This bit value is ignored and assumed to be 0 if <eRdBSzExt> is cleared to 0.
15:14	Reserved	RW 0x0	Reserved

Table 341: EDMA Configuration Register (Continued)
Offset: Port 0: 0x82000, Port 1: 0x84000

Bits	Field	Type/InitVal	Description
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.
17	eCutThroughEn	RW 0x0	This bit enables Basic DMA cut-though operation when writing data to the drive. When the maximum read burst size initiated by the DMA to the Mbus 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 and forward. 1 = Cut through.
18	eEarlyComple-tionEn	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 drive to the Mbus, and it updates the following bits immediately, instead of waiting for this data to be visible in the system memory: • The <BasicDMAActive> field in the Basic DMA Status Register (Table 362 p. 386) is cleared to 0. • When EDMA is disabled, Bits[3:0] <DMAxDone> in the SATAHC Interrupt Cause Register (Table 328 p. 365) 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 <SaCrpbxDone> in the EDMA Interrupt Error Cause Register (Table 343 p. 374). 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 = Early DMA completion disabled. 1 = Early DMA completion enabled.
19	eEDMAQueLen	RW 0x0	EDMA response queue and request queues length 0 = 32 entries 1 = 128 entries
21:20	Reserved	RW 0x0	Reserved
22	eHostQueue-CacheEn	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 = 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 = Host Queue Cache is enabled.
23	eMaskRxPM	RW 0x0	This bit masks the PM field in the incoming FISs. 0 = Mask the PM field in incoming FISs. 1 = Do not mask.

Table 341: EDMA Configuration Register (Continued)
Offset: Port 0: 0x82000, Port 1: 0x84000

Bits	Field	Type/InitVal	Description
24	ResumeDis	RW 0x0	When set to 1, the EDMA never sets the <ContFromPrev> field in the Basic DMA Command Register (Table 361 p. 385). When cleared to 0, the EDMA sets field <ContFromPrev> whenever possible.
25	Reserved	RW 0x0	Reserved
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 field <BasicDMAActive>, clears field <BasicDMAPaused>, and halts. These two fields are in the Basic DMA Status Register (Table 362 p. 386). When this field 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 (Table 381 p. 400)). 2. Clears <BasicDMAActive>. 3. Sets <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 field <eEarlyCompletionEn> is set it waits for data visible in the system memory. 3. Clears <BasicDMAActive>. 4. Sets <BasicDMAPaused> if command was not completed. 5. Halts. 0 = FIS based Switching mode disabled. Basic DMA stops on a command (PRD) boundary. 1 = FIS Based Switching mode enabled. Basic DMA stops on a FIS boundary. **NOTE:** This bit must be set to 1 when FIS based switching mode is used. See Section 8.4.12.2, Port Multiplier—FIS-Based Switching, on page 79. 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.
31:27	Reserved	RES 0x0	Reserved

Table 342: EDMA Timer Register
Offset: Port 0: 0x82004, Port 1: 0x84004

Bits	Field	Type/Init Val	Description
31:0	eTimer	RW 0x0	EDMA Timer. This 32-bit counter is increment every 16 clocks, when the EDMA is enabled (i.e., the <eEnEDMA> field in the EDMA Command Register (Table 351 p. 379) is set to 1). When EDMA command is completed, the content of this register is written into the command response queue. This data may be used to estimate the command execution time.

Table 343: EDMA Interrupt Error Cause Register
Offset: Port 0: 0x82008, Port 1: 0x84008

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 affect.

Bits	Field	Type/Init Val	Description
1:0	Reserved	RW0 0x0	Reserved
2	eDevErr	RW0 0x0	EDMA Device Error This bit is set to 1 when: 1. The EDMA is enabled (i.e., field <eEnEDMA> 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 Section , Device Error Indications, on page 84.
3	eDevDis	RW0 0x0	EDMA Device Disconnect This bit is set to 1 if the device is disconnected. 0 = Device was not disconnected. 1 = Device was disconnected. **NOTE:** After DevDis interrupt, device hard reset is required. Set the <eAtaRst> field in the EDMA Command Register (Table 351 p. 380) to 1. See Section , Device Disconnect, on page 82.
4	eDevCon	RW0 0x0	EDMA Device Connected This bit is set to 1 if the device was disconnected and is connected again. 0 = Device was not reconnected. 1 = Device was reconnected.
5	SerrInt	RW0 0x0	This bit is set to 1 when at least one bit in SStatus Register (Table 367 p. 388) is set to 1and the corresponding bit in SError Interrupt Mask Register (Table 369 p. 390) is enabled. 0 = A bit in SError Register was not set to 1. 1 = A bit in SError Register was set to 1. See Section , SError Register Errors, on page 82. **NOTE:** This bit should be cleared only after clearing the SError Register.
6	Reserved	RW0 0x0	Reserved

Table 343: EDMA Interrupt Error Cause Register (Continued)
Offset: Port 0: 0x82008, Port 1: 0x84008

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 affect.

Bits	Field	Type/Init Val	Description
7	eSelfDis	RW0 0x0	EDMA Self Disable This bit is set to 1 if the EDMA encounters error and clears <eEnEDMA>. 0 = EDMA was not self disabled. 1 = EDMA was self disabled.
8	eTransInt	RW0 0x0	Transport Layer Interrupt This bit is set when at least one bit in the FIS Interrupt Cause Register (Table 385 p. 404) is set to 1 and the corresponding bit in the FIS Interrupt Mask Register (see Table 386 on page 406) is set to 1 (enabled). 0 = Transport layer does not wait for host. 1 = Transport layer waits for host. **NOTE:** This bit should be cleared only after clearing the FIS Interrupt Cause Register.
11:9	Reserved	RW0 0x0	Reserved
12	eIORdyErr	RW0 0x0	EDMA IORdy Error IORdy timeout occurred. See Table 354, “EDMA IORdy Timeout Register,” on page 382. **NOTE:** This bit is only set when the EDMA is disabled.
16:13	LinkCtlRxErr	RW0 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 Section , Serial-ATA II Link Layer Error During Reception of a Control Frame, on page 82.
20:17	LinkDataRxErr	RW0 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 Section , Serial-ATA II Link Layer Error During Reception of a Data Frame, on page 83.

Table 343: EDMA Interrupt Error Cause Register (Continued)
Offset: Port 0: 0x82008, Port 1: 0x84008

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 affect.

Bits	Field	Type/Init Val	Description
25:21	LinkCtlTxErr	RW0 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, on page 83.
30:26	LinkDataTxErr	RW0 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 Section , Serial-ATA II Link Layer Error During Transmission of a Data Frame, on page 83.
31	TransProtErr	RW0 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 Section , Serial-ATA II Transport Layer Protocol Non Transient Errors, on page 83.

Table 344: EDMA Interrupt Error Mask Register
Offset: Port 0: 0x8200C, Port 1: 0x8400C

Bits	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 (Table 343 p. 374), and if these bits are set (0), they mask the interrupt. 0 = Mask 1 = Do not mask

Table 345: EDMA Request Queue Base Address High Register
Offset: Port 0: 0x82010, Port 1: 0x84010

Bits	Field	Type/InitVal	Description
31:0	eRqQBA[63:32]	RW 0x0	The EDMA Request Queue Base Address corresponds to bits [63:32].

Table 346: EDMA Request Queue In-Pointer Register
Offset: Port 0: 0x82014, Port 1: 0x84014

Bits	Field	Type/InitVal	Description
4:0	Reserved	RES 0x0	Reserved
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.
11:10	eRqQIP/eRqQBA	RW 0x0	Function of this field depends on the <eEDMAQueLen> field in the EDMA Configuration Register (Table 341 p. 372). <eEDMAQueLen>=0 This field serves as the EDMA Request Queue Base Address[11:10]. <eEDMAQueLen>=1 This field serves as the EDMA Request Queue In-Pointer[6:5].
31:12	eRqQBA[31:12]	RW 0x0	EDMA Request Queue Base Address corresponds to bits [31:12].

Table 347: EDMA Request Queue Out-Pointer Register
Offset: Port 0: 0x82018, Port 1: 0x84018

Bits	Field	Type/Init Val	Description
4:0	Reserved	RES 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.
11:10	eRqQOP	RW 0x0	Function of this field depends on <eEDMAQueLen>. <eEDMAQueLen>=0 This field is reserved. <eEDMAQueLen>=1 This field serve as EDMA Request Queue Out Pointer[6:5]
31:12	Reserved	RES 0x0	Reserved

Table 348: EDMA Response Queue Base Address High Register
Offset: Port 0: 0x8201C, Port 1: 0x8401C

Bits	Field	Type/Init Val	Description
31:0	eRpQBA[63:32]	RW 0x0	The EDMA Response Queue Base Address corresponds to bits [63:32].

Table 349: EDMA Response Queue In-Pointer Register
Offset: Port 0: 0x82020, Port 1: 0x84020

Bits	Field	Type/Init Val	Description
2:0	Reserved	RES 0x0	Reserved
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.
9:8	eRpQIP	RW 0x0	Function of this field depends on <eEDMAQueLen>. <eEDMAQueLen>=0 This field is reserved. <eEDMAQueLen>=1 This field serve as EDMA Response Queue In-Pointer[6:5]

Table 349: EDMA Response Queue In-Pointer Register (Continued)
Offset: Port 0: 0x82020, Port 1: 0x84020

Bits	Field	Type/Init Val	Description
31:10	Reserved	RES 0x0	Reserved

Table 350: EDMA Response Queue Out-Pointer Register
Offset: Port 0: 0x82024, Port 1: 0x84024

Bits	Field	Type/Init Val	Description
2:0	Reserved	RES 0x0	Reserved
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.
9:8	eRPQBA/eRpQIP	RW 0x0	The function of this field depends on <eEDMAQueLen>. <eEDMAQueLen>=0 The EDMA Response Queue base address corresponds to bits [9:8]. <eEDMAQueLen>=1 This field serves as the EDMA Response Queue Out-Pointer[6:5].
31:10	eRPQBA[31:10]	RW 0x0	The EDMA Response Queue base address corresponds to bits [31:10].

Table 351: EDMA Command Register
Offset: Port 0: 0x82028, Port 1: 0x84028

Bits	Field	Type/Init Val	Description
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: 1. Bit <eDsEDMA> (bit [1]) is set or 2. An error condition occurs and not masked corresponding bit in EDMA Halt Conditions Register (Table 356 p. 383) or 3. Link is down and not masked corresponding bit in EDMA Halt Conditions Register Writing 0 to this bit is ignored. 0 = The EDMA is idle. 1 = The EDMA is active.

Table 351: EDMA Command Register (Continued)
Offset: Port 0: 0x82028, Port 1: 0x84028

Bits	Field	Type/Init Val	Description
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 <eEnEDMA> (bit [0]). If the EDMA operation is aborted during command execution, the host SW must set <eAtaRst> (bit [2]) to recover.
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 Section A.8.7, Serial-ATA Interface Registers, on page 388) 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 (Table 370 p. 390). 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. 0 = Normal operation. 1 = Device reset. ■ When this bit is set and EDMA is enabled (<eEnEDMA> is set to 1) the EDMA operation must be aborted: set <eDsEDMA> to 1. ■ When this bit is set and the Basic DMA is enabled (the <Start> field in the Basic DMA Command Register (Table 361 p. 384) is set to 1) the Basic DMA operation must be aborted: clear <Start> to 0.
3	Reserved	RW 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 new commands from CRQB and insert them into the drive. 1 = Do not Pull new commands from CRQB.
31:5	Reserved	RES 0x0	Reserved

Table 352: EDMA Test Control Register
Offset: Port 0: 0x8202C, Port 1: 0x8402C

Bits	Field	Type/Init Val	Description
0	eOddPry	RW 0x0	EDMA Odd Parity This bit is used for debug of internal parity mechanism. When this bit is set and a write transaction to internal memory is performed, odd parity bit is calculated. When this bit is cleared and a write transaction to internal memory is performed, even parity bit is calculated. During a read transaction from internal memory, the even parity bit is always calculated. 0 = Even bit parity is inserted during the write transaction to internal memory. 1 = Odd bit parity is inserted during the write transaction to internal memory.
1	eLoopBack	RW 0x0	EDMA Loopback Mode When this bit is set to 1, the EDMA loopback mode is enabled and the SATA is reset. When store operation to the device is performed, the data written to the link layer is written into an 8-bit temporary register that receives the last 8 bits of data in the store command. 0 = EDMA Loopback mode disabled. 1 = EDMA Loopback mode enabled.
31:2	Reserved	RES 0x0	Reserved

Table 353: EDMA Status Register
Offset: Port 0: 0x82030, Port 1: 0x84030

Bits	Field	Type/Init Val	Description
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).
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 = System memory to device 1 = Device to system memory
6	eCacheEmpty	RO 0x1	Cache Empty This field indicates if EDMA cache is empty. 0 = Cache not empty 1 = Cache empty

Table 353: EDMA Status Register (Continued)
Offset: Port 0: 0x82030, Port 1: 0x84030

Bits	Field	Type/Init Val	Description
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.
15:8	eSTATE	RO 0x0	EDMA State These bits indicate the current state of the machine.
21:16	eIOId	RO 0x0	EDMA I/O ID This field indicates the I/O 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).
31:22	Reserved	RES 0x0	Reserved

Table 354: EDMA IORdy Timeout Register
Offset: Port 0: 0x82034, Port 1: 0x84034

Bits	Field	Type/Init Val	Description
15:0	eIORdyTimeout	RW 0xBC	EDMA IORdy Signal Timeout Value If SATAHC unit is not ready to complete the transaction for the number of system cycles set in this field, a timeout will occur and the transaction will be completed. If the transaction is terminated as a result of the IORdy timeout, field <eIORdyErr> is set. 0 = eIORdy timeout is ignored; the transaction will not be aborted. x = Timeout will occur after x system clocks.
31:16	Reserved	RES 0x0	Reserved

Table 355: EDMA Command Delay Threshold Register
Offset: Port 0: 0x82040, Port 1: 0x84040

Bits	Field	Type/Init Val	Description
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 355: EDMA Command Delay Threshold Register (Continued)
Offset: Port 0: 0x82040, Port 1: 0x84040

Bits	Field	Type/Init Val	Description
30:16	Reserved	RES 0x0	Reserved
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.

Table 356: EDMA Halt Conditions Register
Offset: Port 0: 0x82060, Port 1: 0x84060

Bits	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 (Table 343 p. 374) and is not masked by the corresponding bit in this field. 0 = Mask 1 = Do not mask

Table 357: EDMA NCQ0 Done/TCQ0 Outstanding Status Register
Offset: Port 0: 0x82094, Port 1: 0x84094
NOTE: When the EDMA is disabled, the fields in this register are Read Only.

Bits	Field	Type/Init Val	Description
31:0	NCQDone/TCQ Outstand[31:0]	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].

Table 358: EDMA NCQ1 Done/TCQ1 Outstanding Status Register
Offset: Port 0: 0x82098, Port 1: 0x84098
NOTE: When the EDMA is disabled, the fields in this register are Read Only.

Bits	Field	Type/InitVal	Description
31:0	NCQDone/TCQ Outstand[63:32]	RW 0x0	See Table 357, EDMA NCQ0 Done/TCQ0 Outstanding Status Register, on page 383.

Table 359: EDMA NCQ2 Done/TCQ2 Outstanding Status Register
Offset: Port 0: 0x8209C, Port 1: 0x8409C
NOTE: When the EDMA is disabled, the fields in this register are Read Only.

Bits	Field	Type/InitVal	Description
31:0	NCQDone[95:64]/ TCQOutstand	RW 0x0	See Table 357, EDMA NCQ0 Done/TCQ0 Outstanding Status Register, on page 383.

Table 360: EDMA NCQ3 Done/TCQ3 Outstanding Status Register
Offset: Port 0: 0x820A0, Port 1: 0x840A0
NOTE: When the EDMA is disabled, the fields in this register are Read Only.

Bits	Field	Type/InitVal	Description
31:0	NCQ-Done[127:96]/ TCQOutstand	RW 0x0	See Table 357, EDMA NCQ0 Done/TCQ0 Outstanding Status Register, on page 383.

A.8.6 Basic DMA Registers

Table 361: Basic DMA Command Register
Offset: Port 0: 0x82224, Port 1: 0x84224

Bits	Field	Type/InitVal	Description
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 <eAtaRst> (bit [2]) to recover.
2:1	Reserved	RW 0x0	Reserved

Table 361: Basic DMA Command Register (Continued)
Offset: Port 0: 0x82224, Port 1: 0x84224

Bits	Field	Type/Init Val	Description
3	Read	RW 0x0	This bit sets the direction of the Basic DMA transfer: 0 = Basic reads are performed: Read from system memory and store in the device. 1 = Basic writes are performed: Load from the device and write to system memory. This bit must not be changed when the Basic DMA is active.
7:4	Reserved	RW 0x0	Reserved
8	DRegionValid	RW 0x0	This bit indicates if the <DataRegion ByteCount> field [31:16] in this register, the Data Region Low Address Register (Table 365 p. 387) and the Data Region High Address Register (Table 366 p. 388) are valid and to be used by the DMA. 0 = Data Region is not valid. 1 = Data Region is valid. This bit must not be changed when the Basic DMA is active.
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 <DataRegion ByteCount> field [31:16] in this register, Data Region Low Address Register and Data Region High Address Register are the last data region to be transferred. 0 = Not last data region. 1 = Last data region to be transferred in the PRD table. This bit must not be changed when the Basic DMA is active.
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> field in the Basic DMA Status Register (Table 362 p. 386) is cleared to 0 and a new PRD table is loaded. 0 = Load new PRD table. 1 = Continue from the PRD table associated with its previous DMA transaction. This bit must not be changed when the Basic DMA is active.
15:11	Reserved	RW 0x0	Reserved
31:16	DataRegion ByteCount	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 the <BasicDMAActive> field in the Basic DMA Status Register (Table 362 p. 386) is cleared to 0. The host must not write to this bit when the Basic DMA is active.

Table 362: Basic DMA Status Register
Offset: Port 0: 0x82228, Port 1: 0x84228

Bits	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 • The <eEDMAFBS> field in the EDMA Configuration Register (Table 341 p. 372) 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 the <eEarlyCompletionEn> field in the EDMA Configuration Register (Table 341 p. 372) is set, this bit is cleared as soon as last data leaves the DMA. When field <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 = The DMA is in idle state. 1 = The DMA is active.
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.
2	BasicDMAPaused	RO 0x0	This bit is valid when bit[0] <BasicDMAActive> in this register is cleared. This bit is set when: • Bit <eEDMAFBS> 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 = The DMA is in idle state. 1 = The DMA is paused.
3	BasicDMALast	RO 0x0	This bit is valid when bit[0] <BasicDMAActive> in this register is cleared. This bit is set when: • Field <eEDMAFBS> is set and • Field <BasicDMAActive> is cleared and • EOT for that region is set in the region descriptor This bit is cleared when: • Field <BasicDMAActive> is cleared and • EOT for that region is cleared in the region descriptor 0 = DMA halts before the last data region. 1 = DMA halts in the last data region.

Table 362: Basic DMA Status Register (Continued)
Offset: Port 0: 0x82228, Port 1: 0x84228

Bits	Field	Type/InitVal	Description
31:4	Reserved	RES 0x0	Reserved

Table 363: Descriptor Table Low Base Address Register
Offset: Port 0: 0x8222C, Port 1: 0x8422C

NOTE: Enhanced Physical Region Descriptors are in use to enable 64-bit memory addressing. See Section 8.4.16.3, EDMA Physical Region Descriptors (ePRD) Table Data Structure, on page 90.
The CPU accesses this register for direct access to the device when the EDMA is disabled.

Bits	Field	Type/InitVal	Description
3:0	Reserved	RES 0x0	Reserved
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.

Table 364: Descriptor Table High Base Address Register
Offset: Port 0: 0x82230, Port 1: 0x84230

NOTE: Enhanced Physical Region Descriptors are in use to enable 64-bit memory addressing. See Section 8.4.16.3, EDMA Physical Region Descriptors (ePRD) Table Data Structure, on page 90.

Bits	Field	Type/InitVal	Description
31:0	Descriptor Table Base Address	RW 0x0	The Descriptor Table Base Address corresponds to address A[63:32].

Table 365: Data Region Low Address Register
Offset: Port 0: 0x82234, Port 1: 0x84234

NOTE: The CPU accesses this register for direct access to the device when the EDMA is disabled. Field <eEnEDMA> in EDMA Command Register (see Table 351 on page 379) is cleared. While the EDMA is enabled, the host must not write this register. If any of these bits are written when field <eEnEDMA> is set, the write transaction will cause unpredictable behavior.

Bits	Field	Type/InitVal	Description
31:0	DataRegion[31:0]	RW 0x0	DataRegion. 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 <BasicDMAActive> is cleared to 0. The host must not write to this bit when the Basic DMA is active.

Table 366: Data Region High Address Register
Offset: Port 0: 0x82238, Port 1: 0x84238

NOTE: The CPU accesses this register for direct access to the device when the EDMA is disabled. Field <eEnEDMA> in EDMA Command Register (see Table 351 on page 379) is cleared. While the EDMA is enabled, the host must not write this register. If any of these bits are written when field <eEnEDMA> is set, the write transaction will cause unpredictable behavior.

Bits	Field	Type/Init Val	Description
31:0	DataRegion [63:32]	RW 0x0	DataRegion. 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 <BasicDMAActive> is cleared to 0. The host must not write to this bit when the Basic DMA is active.

A.8.7 Serial-ATA Interface Registers

Table 367: SStatus Register
Offset: Port 0: 0x82300, Port 1: 0x84300

NOTE: See the Serial-ATA specification for a detailed description.

Bits	Name	Type/Init Val	Description
3:0	DET	RO 0x4	These bits set the interface device detection and PHY state. **0000 =** No device detected, and PHY communication is not established. **0001 =** Device presence detected, but PHY communication is not established. **0011 =** Device presence detected, and PHY communication is established. **0100 =** PHY in offline mode as a result of the interface being disabled or running in a loopback mode. All other values are reserved.
7:4	SPD	RO 0x0	These bits set whether the negotiated interface communication speed is established. **0000 =** No negotiated speed. The device is not present or communication is not established. **0001 =** Generation 1 communication rate negotiated. **0010 =** Generation 2 communication rate negotiated. All other values are reserved.
11:8	IPM	RO 0x0	These bits set the current interface power management state. **0000 =** Device not present, or communication not established. **0001 =** Interface in active state. **0010 =** Interface in PARTIAL power management state. **0110 =** Interface in SLUMBER power management state. All other values are reserved.
31:12	Reserved	RES 0x0	Reserved

Table 368: SError Register
Offset: Port 0: 0x82304, Port 1: 0x84304
NOTE: A write of 1 clears the bits in this register. A write of 0 has no affect.

Bits	Name	Type/ InitVal	Description
0	Reserved	RES 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, or 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.
10:2	Reserved	RES 0x0	Reserved
15:11	Reserved	RES 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.
17	Reserved	RES 0x0	Reserved
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.
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.
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.
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.
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, or other error conditions leading to a negative handshake on a transmitted frame.

Table 368: SError Register (Continued)
Offset: Port 0: 0x82304, Port 1: 0x84304
NOTE: A write of 1 clears the bits in this register. A write of 0 has no affect.

Bits	Name	Type/ InitVal	Description
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.
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.
25	Reserved	RES 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.
31:27	Reserved	RES 0x0	Reserved

Table 369: SError Interrupt Mask Register
Offset: Port 0: 0x82340, Port 1: 0x84340

Bits	Field	Type/Init Val	Description
31:0	eSErrIntMsk	RW 0x019C0000	SError Interrupt Mask Bits Each of these bits checks the corresponding bit in SError Register (Table 368 p. 389), and if these bits are disabled (0), they mask the interrupt. 0 = Mask 1 = Do not mask

Table 370: SControl Register
Offset: Port 0: 0x82308, Port 1: 0x84308
NOTE: See the Serial-ATA specification for a detailed description.

Bits	Name	Type/ InitVal	Description
3:0	DET	RW 0x4	This field controls the host adapter device detection and interface initialization. 0000 = No device detection or initialization action requested. 0001 = Perform interface communication initialization sequence to establish communication. 0100 = Disable the Serial-ATA interface and put the PHY in offline mode. All other values are reserved.

Table 370: SControl Register (Continued)
Offset: Port 0: 0x82308, Port 1: 0x84308
NOTE: See the Serial-ATA specification for a detailed description.

Bits	Name	Type/ InitVal	Description
7:4	SPD	RW 0x0	This field represents the highest allowed communication speed the interface is able to negotiate. 0000 = No speed negotiation restrictions. 0001 = Limit speed negotiation to a rate not greater than Generation 1 communication rate. 0010 = Limit speed negotiation to a rate not greater than Generation 2 communication rate. All other values are reserved.
11:8	IPM	RW 0x0	This field represents the enabled interface power management states that can be invoked via the Serial-ATA interface power management capabilities. 0000 = No interface power management state restrictions. 0001 = Transition to the PARTIAL power management state disabled. 0010 = Transition to the SLUMBER power management state disabled. 0011 = Transition to both the PARTIAL and SLUMBER power management states disabled. All other values are reserved.
15:12	SPM	RO 0x0	This 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. 0000 = No power management state transition requested. 0001 = Transition to the PARTIAL power management state initiated. 0010 = Transition to the SLUMBER power management state initiated. 0100 = Transition to the active power management state initiated. All other values are reserved. This field is read only 0000.
31:16	Reserved	RES 0x0	Reserved

Table 371: LTMode Register
Offset: Port 0: 0x8230C, Port 1: 0x8430C
NOTE:

Bits	Name	Type/ InitVal	Description
5:0	RcvWaterMark	RW 0x30	WaterMark Receiving Flow Control settings (values in DWORDs). When the Rx FIFO's available entry is less than this value, Serial-ATA Flow Control is performed by sending HOLD primitives.
6	Reserved	RES 0x0	Reserved
7	NearEndLBEn	RW 0x1	Near-End Loopback Enable. 0 = Near-end loopback (BIST) is disabled. 1 = Near-end loopback (BIST) is enabled.
13:8	Reserved	RW 0x0	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.

Table 371: LTMode Register (Continued)
Offset: Port 0: 0x8230C, Port 1: 0x8430C
NOTE:

Bits	Name	Type/ InitVal	Description
14	Reserved	RW 0x0	Reserved
16:15	Reserved	RW 0x1	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
18:17	Reserved	RW 0x0	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
20:19	Reserved	RW 0x0	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
23:21	Reserved	RW	Reserved
24	Reserved	RW 0x1	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
31:25	Reserved	RW	Reserved

Table 372: PHY Mode 3 Register
Offset: Port 0: 0x82310, Port 1: 0x84310
NOTE: This register must be fully written on every write access to any of its fields.

Bits	Name	Type/ InitVal	Description
1:0	Reserved	RES 0x2	Reserved
4:2	SQ	RW 0x2	Squelch Detector Threshold. 000 = 50 mV (peak-to-peak) 001 = 100 mV (peak-to-peak) 010 = 150 mV (peak-to-peak) — default 011 = 200 mV (peak-to-peak) 100 = 250 mV (peak-to-peak) 101 = 300 mV (peak-to-peak) 110 = 350 mV (peak-to-peak) 111 = 400 mV (peak-to-peak)
31:5	Reserved	RES 0x0	Reserved **NOTE:** Must write the value 27h'5815601 to this field on every access.

Table 373: PHY Mode 4 Register
Offset: Port 0: 0x82314, Port 1: 0x84314
NOTE: This register must be fully written on every write access to any of its fields.

Bits	Name	Type/ InitVal	Description
1:0	PhyIntConfPara	RW 0x2	PHY Internal Configuration Parameter Must initialize 0x01 to this field.
17:2	Reserved	RES 0x0	Reserved **NOTE:** Must write the value 16h'0001 to this field on every access.
20:18	HotPlugTimer	RW 0x011	Delay to Disconnect Hot Plug 000 = 2 ms 001 = 4 ms 010 = 10 ms 011 = 16 ms—Default 100 = 32 ms 101 = 64 ms 110 = 128 ms 111 = 256 ms
24:21	Reserved	RW 0x0	Reserved **NOTE:** Must write the value 4h'0 to this field on every access.
25	PortSelector	RW 0x0	Port Selector 0 = Port Selector function is off. 1 = A 0-to-1 transition of this bit will cause Port Selector protocol-based OOB sequence to be sent.
28:26	Reserved	RES 0x0	Reserved **NOTE:** Must write the value 0x0 to this field on every access.
29	DisSwap	RW 0x0	Hot Swap detection. 0 = On 1 = Off. When hot swapping, no COMRESET/COMINIT will be sent.
30	Reserved	RES 0x0	Reserved **NOTE:** Must write the value 0x0 to this field on every access.
31	OOBBypass	RW 0x0	Out of band bypass. PHY Ready method selection 0 = Normal 1 = Bypass OOB handshake, and force PhyRdy.

Table 374: PHY Mode 1 Register
Offset: Port 0: 0x8232C, Port 1: 0x8432C
NOTE: This register must be fully written on every write access to any of its fields.

Bits	Name	Type/ InitVal	Description
31:0	Reserved	RES 0x0	Reserved **NOTE:** Must write the value 32h'40550520 to this field on every access.

Table 375: PHY Mode 2 Register
Offset: Port 0: 0x82330, Port 1: 0x84330
NOTE: This register must be fully written on every write access to any of its fields.

Bits	Name	Type/ InitVal	Description
4:0	Reserved		Reserved **NOTE:** Must write the value 5h'0F to this field on every access.
7:5	TxPre	RW 0x2	Transmitter Pre-emphasis. 000 = 1-0.00Z^-1 001 = 1-0.05Z^-1 010 = 1-0.10Z^-1 011 = 1-0.15Z^-1 100 = 1-0.20Z^-1 101 = 1-0.25Z^-1 110 = 1-0.30Z^-1 111 = 1-0.35Z^-1
10:8	TxAmp	RW 0x4	Transmitter Differential Amplitude. 000 = I=4 mA, V=200 mVp-p 001 = I=6 mA, V=300 mVp-p 010 = I=8 mA, V=400 mVp-p 011 = I=10 mA, V=500 mVp-p 100 = I=12 mA, V=600 mVp-p 101 = I=14 mA, V=700 mVp-p 110 = I=16 mA, V=800 mVp-p 111 = I=18 mA, V=900 mVp-p
11	Loopback	RW 0x0	Loopback function as near-end loopback. 0 = Analog PHY is in Normal mode. 1 = Analog PHY is in Loopback mode.
19:12	Reserved	RW 0x0	Reserved **NOTE:** Must write the value 8h'09 to this field on every access.
23:20	Reserved	RW 0x9	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
25:24	Reserved	RW 0x0	Reserved **NOTE:** Must write the value 0x0 to this field on every access.
29:26	Reserved	RW 0x9	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
31:30	Reserved	RW 0x0	Reserved **NOTE:** Must write the value 0x0 to this field on every access.

Table 376: BIST Control Register
Offset: Port 0: 0x82334, Port 1: 0x84334

Bits	Name	Type/ InitVal	Description
7:0	BISTPattern	RW 0x0	BIST Pattern Test pattern, refer to bits [15:8] of the first DWORD of the BIST Activate FIS.
8	BISTMode	RW 0x0	BIST mode Test direction. 0 = BIST Activate FIS Receiver mode. 1 = BIST Activate FIS Transmitter mode.
9	BISTEn	RW 0x0	BIST Test enable. On the assertion of the signal, BIST mode starts. 0 = Disabled. 1 = Enabled.
10	BISTResult	RO 0x0	BIST Test Pass 0 = Passed. 1 = Failed.
15:11	Reserved	RES 0x0	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.
22:16	Reserved	RES 0x0	Reserved
31:23	Reserved	RES 0x1	Reserved **NOTE:** Perform a read-modify-write access to this field to avoid the contents being changed.

Table 377: BIST-DW1 Register
Offset: Port 0: 0x82338, Port 1: 0x84338

Bits	Name	Type/ InitVal	Description
31:0	BistDw1	WO 0x0	In BIST mode: (The <BISTEn> field in the BIST Control Register (Table 376 p. 395) is set to 1). This field is the second DWORD of the BIST Activate FIS. In PHY Loopback mode: (The <LBEnable> field in the Serial-ATA Interface Test Control Register (Table 381 p. 400) is set to 1). This field is the high DWORD [63:32] of the user specified loopback pattern of the BIST Activate FIS.

Table 378: BIST-DW2 Register
Offset: Port 0: 0x8233C, Port 1: 0x8433C

Bits	Name	Type/ InitVal	Description
31:0	BistDw2	WO 0x0	**In BIST mode:** (Field <BISTEn> is set to 1). This field is the third DWORD of the BIST Activate FIS. In PHY Loopback mode: (Field <LBEnable> is set to 1). This field is the low DWORD [31:0] of the user specified loopback pattern of the BIST Activate FIS.

Table 379: Serial-ATA Interface Configuration Register
Offset: Port 0: 0x82050, Port 1: 0x84050

NOTE: After any modification in this register, the host must set bit[2]<eAtaRst> field in the EDMA Command Register (Table 351 p. 380).

Bits	Field	Type/InitVal	Description
1:0	RefClkCnf	RW 0x01	PHY PLL Reference clock frequency. 00 = 20 MHz 01 = 25 MHz 10 = 30 MHz 11 = 40 MHz **NOTE:** Must be set to 01.
3:2	RefClkDiv	RW 0x01	PHY PLL Reference clock divider. 00 = Divided by 1. 01 = Divided by 2. Used when PHY PLL Reference clock is 20 MHz or 25 MHz 10 = Divided by 4. Used when PHY PLL Reference clock is 40 MHz 11 = Divided by 3. Used when PHY PLL Reference clock is 30 MHz **NOTE:** Must be set to 01.
5:4	RefClkFeedDiv	RW 0x01	PHY PLL Reference clock feedback divider. Its setting is configured according to bit <Gen2En>. When Gen2En is set to 0: 00 = Divided by 50. 01 = Divided by 60. Used when PHY PLL Reference clock is 25 MHz 10 = Divided by 75. Used when PHY PLL Reference clock is 20 MHz, 30 MHz, or 40 MHz 11 = Divided by 90. When Gen2En is set to 1: 00 = Divided by 100. 01 = Divided by 120. Used when PHY PLL Reference clock is 25 MHz 10 = Divided by 150. Used when PHY PLL Reference clock is 20 MHz, 30 MHz, or 40 MHz 11 = Divided by 180. **NOTE:** Must be set to 01.
6	PhySSCEn	RW 0x0	SSC enable 0 = SSC disable 1 = SSC enable

Table 379: Serial-ATA Interface Configuration Register (Continued)
Offset: Port 0: 0x82050, Port 1: 0x84050

NOTE: After any modification in this register, the host must set bit[2]<eAtaRst> field in the EDMA Command Register (Table 351 p. 380).

Bits	Field	Type/InitVal	Description
7	Gen2En	RW 0x1	Generation 2 communication speed support. 0 = Disabled 1 = Enabled
8	CommEn	RW 0x0	PHY communication enable signal to override field <DET> field in the SControl Register (Table 370 p. 390) setting. 0 = DET setting is not overridden. 1 = If DET value is 0x4, its value is overridden with a value of 0x0.
9	PhyShutdown	RW 0x0	PHY shutdown. **0 = PHY is** functional. **1 =** PHY is in Shutdown mode.
10	TargetMode	RW 0x0	Target Mode. This bit defines the Serial-ATA port that functions as a target during Target mode operation. This bit may be set only when bit[11] <ComChannel> is also set. 0 = Target 1 = Initiator
11	ComChannel	RW 0x0	Communication Channel Operating Mode. This bit defines if the Serial-ATA port functions in Target mode operation. 0 = Disk controller 1 = 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 Table 322, Shadow Register Block Registers Map, on page 362).
23:12	Reserved	RW 0x9B7	Reserved This field must be written with a value of 0x9B7.
24	IgnoreBsy	RW 0x0	When this bit is set to 1, the ATA task registers are updated when Register Device to Host FIS is received or when the host writes to the ATA Status registers, regardless of the value of the <BSY> bit in the ATA Status register (see Table 322, Shadow Register Block Registers Map, on page 362). When EDMA is enabled the transport layer ignores the value of this bit and assume a value of 1. This bit must be cleared before the host issues any PIO commands.
25	LinkRstEn	RW 0x0	When this bit is set to 1 and when bit <RST> is set to 1 in the ATA status register (see Table 322, Shadow Register Block Registers Map, on page 362) in the middle of data FIS reception, the link layer responds with a SYNC primitive to reception of data FIS. When the transport layer receives SYNC in the back channel in response, it sends the Control FIS with the <SRST> bit set to 1. When this bit is cleared to 0 and when bit <RST> is set to 1 in the ATA status register, in a middle of data FIS reception, the transport layer drops the data of the incoming FIS. When the incoming data FIS completes, it sends the Control FIS with the <SRST> bit set to 1.

Table 379: Serial-ATA Interface Configuration Register (Continued)
Offset: Port 0: 0x82050, Port 1: 0x84050

NOTE: After any modification in this register, the host must set bit[2]<eAtaRst> field in the EDMA Command Register (Table 351 p. 380).

Bits	Field	Type/InitVal	Description
26	CmdRetxDs	RW 0x0	When this bit is cleared to 0 and Register Host to Device FIS transmission was not completed successfully as indicated by the <FISTxDone> field and the <FISTxErr> field in the FIS Interrupt Cause Register (Table 385 p. 405), the transport layer retransmits the Register Host to Device FIS. When this bit is set to 1 and Register Host to Device FIS transmission was not completed successfully, as indicated by fields <FISTxDone> and <FISTxErr>, the transport layer does *not* retransmit the Register Host to Device FIS. When the EDMA is enabled, this bit value is ignored and assumed to be 1. EDMA retransmits the Register Host to Device FIS.
31:27	Reserved	RES 0x0	Reserved

Table 380: Serial-ATA Interface Control Register
Offset: Port 0: 0x82344, Port 1: 0x84344,

NOTE: When field <eEnEDMA> is set, this register must not be written.
If this register is written when field <eEnEDMA> is set, the write transaction will cause unpredictable behavior.

Bits	Field	Type/Init Val	Description
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.
7:4	Reserved	RW 0x0	Reserved
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 (Table 383 p. 402). When this bit is cleared, the <VendorUqDn> field and the <VendorUqErr> field in the Serial-ATA Interface Status Register (Table 382 p. 401) are also cleared. Before setting this bit the Host must verify: • No pending commands are in progress. • The EDMA is disabled, field <eEnEDMA> is cleared.
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. Field <VendorUqDn> is set when the transmission is completed; field <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.
15:10	Reserved	RES 0x0	Reserved

Table 380: Serial-ATA Interface Control Register (Continued)
Offset: Port 0: 0x82344, Port 1: 0x84344,

NOTE: When field <eEnEDMA> is set, this register must not be written.
If this register is written when field <eEnEDMA> is set, the write transaction will cause unpredictable behavior.

Bits	Field	Type/Init Val	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., the <ComChannel> field in the Serial-ATA Interface Configuration Register (Table 379 p. 397) 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 SW 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 = Transport layer does not send DMA data FISs. 1 = Transport layer keeps sending (multiple) DMA data FISs until the data transaction associated with the command is completed.
23:17	Reserved	RES 0x0	Reserved
24	ClearStatus	SC 0x0	Status Self-Clear This bit clears bits [16], [30], and [31] in the Serial-ATA Interface Status Register (Table 382 p. 400)). 0 = Does not clear bits. 1 = Clears the bits.
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.
31:26	Reserved	RES 0x0	Reserved

Table 381: Serial-ATA Interface Test Control Register
Offset: Port 0: 0x82348, Port 1: 0x84348

Bits	Field	Type/Init Val	Description
0	MBistEn	RW 0x0	Memory BIST Enable Start Memory BIST test in MBIST mode. 0 = Memory BIST test disabled. 1 = Memory BIST test enabled.
1	TransFrmSizExt	RW 0x0	Maximum Transmit Frame Size Extended See field <TransFrmSiz> [15:14].
3:2	Reserved	RW 0x0	Reserved

Table 381: Serial-ATA Interface Test Control Register (Continued)
Offset: Port 0: 0x82348, Port 1: 0x84348

Bits	Field	Type/Init Val	Description
5:4	Reserved	RO 0x0	Reserved
7:6	Reserved	RW 0x0	Reserved
8	LBEnable	RW 0x0	PHY Loopback Enable This bit enables SATA near-end PHY loopback. 0 = PHY near-end Loopback disabled. 1 = PHY near-end Loopback enabled.
12:9	LBPattern	RW 0x0	PHY Loopback pattern
13	LBStartRd	RW 0x0	Loopback Start BIST-DW1 Register (Table 377 p. 395) is used as a User-specified test pattern low DWORD. BIST-DW2 Register (Table 378 p. 396) is used as a User-specified test pattern high DWORD. These registers must be initiated before this bit is set. 0 = Disable 1 = Enable Loopback Start.
15:14	TransFrmSiz	RW 0x0	Maximum Transmit Frame Size When<TransFrmSizExt> is 0: 00 = Maximum Transmit Frame Size is 8 KB. 01 = Maximum Transmit Frame Size is 512B. 10 = Maximum Transmit Frame Size is 64B. 11 = Maximum Transmit Frame Size is 128B. When <TransFrmSizExt> is 1: 00 = Maximum Transmit Frame Size is 256B. 01 = Maximum Transmit Frame Size is 1 KB. 10 = Maximum Transmit Frame Size is 2 KB. 11 = Maximum Transmit Frame Size is 4 KB.
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> field is changed from 0 to 1.

Table 382: Serial-ATA Interface Status Register
Offset: Port 0: 0x8234C, Port 1: 0x8434C

Bits	Field	Type/Init Val	Description
7:0	FISTypeRx	RO 0x0	FIS Type Received. This field specifies the FIS Type of the last received FIS.

Table 382: Serial-ATA Interface Status Register (Continued)
Offset: Port 0: 0x8234C, Port 1: 0x8434C

Bits	Field	Type/Init Val	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.
12	VendorUqDn	RO 0x0	Vendor Unique FIS Transmission Done. This bit is set when the Vendor Unique FIS transmission has completed. 0 = Vendor Unique FIS transmission has not completed. 1 = Vendor Unique FIS transmission has 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 field <VendorUqDn> of this register is set. 0 = Vendor Unique FIS transmission has completed successfully. 1 = Vendor Unique FIS transmission has completed with error.
14	MBistRdy	RO 0x1	Memory BIST Ready This bit indicates when the memory BIST test is completed. 0 = Memory BIST test is not completed. 1 = Memory BIST test is completed.
15	MBistFail	RO 0x0	Memory BIST Fail This bit indicates if the memory BIST test passed. It is valid when field <MBistRdy> of this register is set. 0 = Pass 1 = Fail
16	AbortCommand	RO 0x0	Abort Command This bit indicates if the transport has aborted a command as a response to collision with incoming FIS. 0 = Command was not aborted. 1 = Command was aborted. **NOTE:** This bit is cleared when the <ClearStatus> field in the Serial-ATA Interface Control Register (Table 380 p. 399).
17	LBPass	RO 0x0	Near-end Loopback Pass This bit indicates if the Near-end Loopback test passed. 0 = Fail 1 = Pass
18	DMAAct	RO 0x0	DMA Active This bit indicates if the transport DMA 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
20	RxHdAct	RO 0x0	Rx Header Active This bit indicates if the transport Rx Header FSM is active. 0 = Idle 1 = Active

Table 382: Serial-ATA Interface Status Register (Continued)
Offset: Port 0: 0x8234C, Port 1: 0x8434C

Bits	Field	Type/Init Val	Description
21	TxHdAct	RO 0x0	Tx Header Active This bit indicates if the transport Tx Header FSM is active. 0 = Idle 1 = Active
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> field in the SError Register (Table 368 p. 390). 0 = Device presence not detected. 1 = Device presence detected.
23	LinkDown	RO 0x1	SATA communication is not ready. Primitives or FISs are not able to be transmitted or received. 0 = Link is ready. 1 = Link is not ready.
28:24	TransFsmSts	RO 0x0	Transport Layer FSM status 0x00 = Transport layer is idle. 0x00–0x1F = Transport layer is not idle.
29	Reserved	RO 0x0	Reserved
30	RxBIST	RO 0x0	Set to 1 when BIST FIS is received. **NOTE:** This bit is cleared when the <ClearStatus> field in the Serial-ATA Interface Control Register (Table 380 p. 399) is set to 1.
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> field is set to 1.

Table 383: Vendor Unique Register
Offset: Port 0: 0x8235C, Port 1: 0x8435C

Bits	Name	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 384: FIS Configuration Register
Offset: Port 0: 0x82360, Port 1: 0x84360

Bits	Name	Type/Init Val	Description
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 <eEnEDMA> is set to 1, 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. 0 = Do not wait for host ready. 1 = Wait for host ready. 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
15:8	FISWait4Host RdyEn	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. 0 = Do not wait for host ready. 1 = Wait for host ready. The function of each bit in this field is: <FISWait4Host RdyEn>[0]: Register — Device to Host FIS with <ERR> or <DF> bit set to 1. <FISWait4Host RdyEn>[1]: SDB FIS is received with <N> bit set to 1. <FISWait4Host RdyEn>[2]: SDB FIS is received with <ERR> bit set to 1. <FISWait4Host RdyEn>[3]: BIST activate FIS <FISWait4Host RdyEn>[4]: PIO Setup FIS <FISWait4Host RdyEn>[5]: Data FIS with Link error <FISWait4Host RdyEn>[6]: Unrecognized FIS type <FISWait4Host RdyEn>[7]: Any FIS
16	FISDMAActive SyncResp	RW 0x0	This bit identifies whether the transport layer responses with a single SYNC primitive after the DMA activates FIS reception. 0 = Normal response 1 = Response with single SYNC primitive. Must be set to 1 when bit <eEDMAFBS> field in the EDMA Configuration Register (Table 341 p. 372) is set to 1.
17	FISUnrecType Cont	RW 0x0	When this bit is set, the transport layer state machine ignores incoming FIS with unrecognized FIS type. 0 = When an unrecognized FIS type is received, the transport layer goes into error state and asserts a protocol error. 1 = When an unrecognized FIS type is received, the transport layer does not go into error state and does not assert a protocol error.
31:18	Reserved	RES 0x0	Reserved

Table 385: FIS Interrupt Cause Register
Offset: Port 0: 0x82364, Port 1: 0x84364

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 affect.

Bits	Name	Type/Init Val	Description
7:0	FISWait4Rdy	RW0 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 DW <FISWait4Rdy>[5]: Data FIS entire FIS <FISWait4Rdy>[7:6]: Reserved 0 = No interrupt indication. 1 = Corresponding interrupt occurs. 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 DW0 Register through FIS DW6 Register (Table 393 p. 407) 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 DW0 Register through FIS DW6 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 DW0 Register is updated with the content of the FIS. FIS DW1 Register through FIS DW6 Register are not updated. When at least one bit in this field is set and the corresponding bit in the <FISWait4RdyEn> field in the FIS Configuration Register (Table 384 p. 403) is enabled (set to 1), the transport layer prevents assertion of the primitive R_RDY and the reception of the next FIS.

Table 385: FIS Interrupt Cause Register (Continued)
Offset: Port 0: 0x82364, Port 1: 0x84364

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 affect.

Bits	Name	Type/Init Val	Description
15:8	FISWait4HostRdy	RW0 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. 0 = No interrupt indication. 1 = Corresponding interrupt occurs. 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 DW0 Register through FIS DW6 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 DW0 Register is updated with the content of the FIS. FIS DW1 Register through FIS DW6 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 DW0 Register through FIS DW6 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 <FISWait4Host RdyEn> field in the FIS Configuration Register (Table 384 p. 403) is enabled (set to 1), the transport layer prevents assertion of the primitive R_RDY and the reception of the next FIS.
23:16	Reserved	RW0 0x0	Reserved.
24	FISTxDone	RW0 0x0	This bit is set to 1 when the FIS transmission is done, either aborted or completed with R_OK or R_ERR. 0 = Frame transmission continues. 1 = Frame transmission completed, either with R_ERR or R_OK, or frame transmission aborted.
25	FISTxErr	RW0 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 = Frame transmission was completed successful with R_OK. 1 = Frame transmission was not completed successful.
31:26	Reserved	R0 0x0	Reserved

Table 386: FIS Interrupt Mask Register
Offset: Port 0: 0x82368, Port 1: 0x84368

Bits	Name	Type/Init Val	Description
25:0	FISIntMask	RW 0x0000A00	FIS Interrupt Error Mask Bits Each of these bits mask the corresponding bit in the FIS Interrupt Cause Register (Table 385 p. 404). If a bit in the FIS Interrupt Cause Register is set and the corresponding bit in this register is set to 1, the <eTransInt> field in the EDMA Interrupt Error Cause Register (Table 343 p. 375) in EDMA Interrupt Error Cause Register (Table 343 p. 374) is also set to 1. 0 = Mask 1 = Do not mask
31:26	Reserved	RO 0x0	Reserved

Table 387: FIS DW0 Register
Offset: Port 0: 0x82370, Port 1: 0x84370

Bits	Name	Type/Init Val	Description
31:0	RxFISDW0	RO 0x0	This field contains DWORD 0 of the incoming data or non-data FIS.

Table 388: FIS DW1 Register
Offset: Port 0: 0x82374, Port 1: 0x84374

Bits	Name	Type/Init Val	Description
31:0	RxFISDW1	RO 0x0	This field contains DWORD 1 of the incoming non-data FIS.

Table 389: FIS DW2 Register
Offset: Port 0: 0x82378, Port 1: 0x84378

Bits	Name	Type/Init Val	Description
31:0	RxFISDW2	RO 0x0	This field contains DWORD 2 of the incoming non-data FIS.

Table 390: FIS DW3 Register
Offset: Port 0: 0x8237C, Port 1: 0x8437C

Bits	Name	Type/Init Val	Description
31:0	RxFISDW3	RO 0x0	This field contains DWORD 3 of the incoming non-data FIS.

Table 391: FIS DW4 Register
Offset: Port 0: 0x82380, Port 1: 0x84380

Bits	Name	Type/Init Val	Description
31:0	RxFISDW4	RO 0x0	This field contains DWORD 4 of the incoming non-data FIS.

Table 392: FIS DW5 Register
Offset: Port 0: 0x82384, Port 1: 0x84384

Bits	Name	Type/Init Val	Description
31:0	RxFISDW5	RO 0x0	This field contains DWORD 5 of the incoming non-data FIS.

Table 393: FIS DW6 Register
Offset: Port 0: 0x82388, Port 1: 0x84388

Bits	Name	Type/Init Val	Description
31:0	RxFISDW6	RO 0x0	This field contains DWORD 6 of the incoming non-data FIS.

A.9 Gigabit Ethernet Controller Registers

Note

- If the Ethernet Unit Control (EUC) register's <Port0_PW> bits [18] is set to 0 (deactivated), the specific port's registers can not be accessed. An attempt to read from a deactivated port's registers will cause a system hang.
- All interrupt cause registers are write 0 to clear, meaning that writing 1 value has no effect, while writing 0 resets the relevant bit in the register.

Table 394: Ethernet Unit Global Registers Map

Description	Offset	Table, Page
Ethernet Unit Global Registers		
PHY Address	0x72000	Table 395, p.410
SMI	0x72004	Table 396, p.411
Ethernet Unit Default Address (EUDA)	0x72008	Table 397, p.411
Ethernet Unit Default ID (EUDID)	0x7200C	Table 398, p.411
Ethernet Unit Reserved (EU)	0x72014	Table 399, p.412
Ethernet Unit Interrupt Cause (EUIC)	0x72080	Table 400, p.412
Ethernet Unit Interrupt Mask (EUIM)	0x72084	Table 401, p.413
Ethernet Unit Error Address (EUEA)	0x72094	Table 402, p.414
Ethernet Unit Internal Address Error (EUIAE)	0x72098	Table 403, p.414
Ethernet Unit Port Pads Calibration (EUPCR)	0x720A0	Table 404, p.414
Ethernet Unit Control (EUC)	0x720B0	Table 405, p.415
Base Address	BA0 0x72200, BA1 0x72208, BA2 0x72210, BA3 0x72218, BA4 0x72220, BA5 0x72228	Table 406, p.415
Size (S)	SR0 0x72204, SR1 0x7220C, SR2 0x72214, SR3 0x7221C, SR4 0x72224, SR5 0x7222C	Table 407, p.416
High Address Remap (HA)	HARR0 0x72280, HARR1 0x72284, HARR2 0x72288, HARR3 0x7228C	Table 408, p.416
Base Address Enable (BARE)	0x72290	Table 409, p.417
Ethernet Port Access Protect (EPAP)	0x72294	Table 410, p.417
Ethernet Unit Port Registers		
Port Configuration (GEC)	0x72400	Table 411, p.418
Port Configuration Extend (GECX)	0x72404	Table 412, p.419
MII Serial Parameters	0x72408	Table 413, p.419
GMII Serial Parameters	0x7240C	Table 414, p.420

Table 394: Ethernet Unit Global Registers Map (Continued)

Description	Offset	Table, Page
VLAN EtherType (EVLANE)	0x72410	Table 415, p.420
MAC Address Low (MACAL)	0x72414	Table 416, p.421
MAC Address High (MACAH)	0x72418	Table 417, p.421
SDMA Configuration (SDC)	0x7241C	Table 418, p.421
IP Differentiated Services CodePoint 0 to Priority (DSCP0)	0x72420	Table 419, p.423
IP Differentiated Services CodePoint 1 to Priority (DSCP1)	0x72424	Table 420, p.423
IP Differentiated Services CodePoint 2 to Priority (DSCP2, DSCP3, DSCP4, DSCP5)	DSCP2 0x72428, DSCP3 0x7242C, DSCP4 0x72430, DSCP5 0x72434	Table 421, p.423
IP Differentiated Services CodePoint 6 to Priority (DSCP6)	0x72438	Table 422, p.423
Port Serial Control (PSC)	0x7243C	Table 423, p.424
VLAN Priority Tag to Priority (VPT2P)	0x72440	Table 424, p.427
Ethernet Port Status (PS)	0x72444	Table 425, p.427
Transmit Queue Command (TQC)	0x72448	Table 426, p.429
Maximum Transmit Unit (MTU)	0x72458	Table 427, p.429
Port Interrupt Cause (IC)	0x72460	Table 428, p.430
Port Interrupt Cause Extend (ICE)	0x72464	Table 429, p.431
Port Interrupt Mask (PIM)	0x72468	Table 430, p.432
Port Extend Interrupt Mask (PEIM)	0x7246C	Table 431, p.433
Port Rx FIFO Urgent Threshold (PRFUT)	0x72470	Table 432, p.433
Port Tx FIFO Urgent Threshold (PTFUT)	0x72474	Table 433, p.433
Port Rx Minimal Frame Size (PMFS)	0x7247C	Table 434, p.434
Port Rx Discard Frame Counter (GEDFC)	0x72484	Table 435, p.434
Port Overrun Frame Counter (POFC)	0x72488	Table 436, p.434
Port Internal Address Error (EUIAE)	0x72494	Table 437, p.434
Ethernet Current Receive Descriptor Pointers (CRDP)	Q0 0x7260C, Q1 0x7261C, Q2 0x7262C, Q3 0x7263C, Q4 0x7264C, Q5 0x7265C, Q6 0x7266C, Q7 0x7267C	Table 438, p.435
Receive Queue Command (RQC)	0x72680	Table 439, p.435
Transmit Current Served Descriptor Pointer	0x72684 (Read Only)	Table 440, p.436
Transmit Current Queue Descriptor Pointer (TCQDP)	Q0 0x726C0	Table 441, p.436

Table 394: Ethernet Unit Global Registers Map (Continued)

Description	Offset	Table, Page
Transmit Queue Token-Bucket Counter (TQxTBC) **NOTE:** Transmit Queues 1–7 are reserved.	Q0 0x72700, Q1 0x72710, Q2 0x72720, Q3 0x72730, Q4 0x72740, Q5 0x72750, Q6 0x72760, Q7 0x72770	Table 442, p.436
Transmit Queue Token Bucket Configuration (TQxTBC) **NOTE:** Transmit Queues 1–7 are reserved.	Q0 0x72704, Q1 0x72714, Q2 0x72724, Q3 0x72734, Q4 0x72744, Q5 0x72754, Q6 0x72764, Q7 0x72774	Table 443, p.436
Transmit Queue Arbiter Configuration (TQxAC) **NOTE:** Transmit Queues 1–7 are reserved.	Q0 0x72708, Q1 0x72718, Q2 0x72728, Q3 0x72738, Q4 0x72748, Q5 0x72758, Q6 0x72768, Q7 0x72778	Table 444, p.437
Destination Address Filter Special Multicast Table (DFSMT)	0x 73400–0x734FC	Table 445, p.437
Destination Address Filter Other Multicast Table (DFUT)	0x73500–0x735FC	Table 446, p.438
Destination Address Filter Unicast Table (DFUT)	0x73600–0x7360C	Table 447, p.439
MAC MIB Counters	0x73000–0x7307C	Description under Appendix A.9.3, Port MIB Counter Register, on page 441.

A.9.1 Gigabit Ethernet Unit Global Registers

Table 395: PHY Address
Offset:0x72000

Bits	Field	Type/InitVal	Description
4:0	PhyAd_0	RW 0x8	PHY device address
14:5	Reserved	RW 0x0	Reserved
31:15	Reserved	RO 0x0	Reserved

Table 396: SMI
Offset:0x72004

Bits	Field	Type/InitVal	Description
15:0	Data	RW N/A	**Management for SMI READ operation:** Two transactions are required: (1) management write to the SMI register where <Opcode> = 1, <PhyAd>, <RegAd> with the Data having any value. (2) management read from the SMI register. When reading back the SMI register, the Data is the addressed PHY register contents if the ReadValid bit[27] is 1. The Data remains undefined as long as ReadValid is 0. **Management for SMI WRITE operation:** One Management transaction is required: Management write to the SMI register with <Opcode> = 0, <PhyAd>, <RegAd> with the Data to be written to the addressed PHY register.
20:16	PhyAd	RW 0x0	PHY device address
25:21	RegAd	RW 0x0	PHY device register address
26	Opcode	RW 0x1	0 = Write 1 = Read
27	ReadValid	RO 0x0	1 = Indicates that the Read operation for the addressed RegAd register has completed, and that the data is valid on the <Data> field.
28	Busy	RO 0x0	1 = Indicates that an operation is in progress and that the CPU must not write to the SMI register at this time.
31:29	N/A	RW 0x0	These bits are driven 0x0 during any write to the SMI register.

Table 397: Ethernet Unit Default Address (EUDA)
Offset:0x72008

Bits	Field	Type/InitVal	Description
31:0	DAR	RW 0x0	Specifies the Default Address to which the Ethernet unit directs no match, multiple address hits, and address protect violations. Occurrence of this event may be the result of programming errors of the descriptor pointers or buffer pointers.

Table 398: Ethernet Unit Default ID (EUDID)
Offset:0x7200C

Bits	Field	Type/InitVal	Description
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 398: Ethernet Unit Default ID (EUDID) (Continued)
Offset:0x7200C

Bits	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.
31:12	Reserved	RO 0x0	Read Only

Table 399: Ethernet Unit Reserved (EU)
Offset:0x72014

Bits	Field	Type/InitVal	Description
0	Fast MDC	RW 0x0	Use Faster MDC. 0 = Normal mode. 1 = MDC clock will be set to TCLK dividing by 16.
1	ACCS	RW 0x0	Accelerate Slot Time. 0 = Normal mode. 1 = MDC clock will be set to TCLK dividing by 8.
31:2	Reserved	RO 0x0	Read Only

Table 400: Ethernet Unit Interrupt Cause (EUIC)
Offset:0x72080

NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.

Bits	Field	Type/InitVal	Description
0	EtherIntSum	RO 0x0	Ethernet Unit Interrupt Summary This bit is a logical OR of the unmasked bits[12:1] in the register.
1	Parity	RW 0x0	Parity Error Effect on Tx DMA operation: If a parity error occurs on the first descriptor fetch, the DMA stops and disables the queue. If it is on a non-first descriptor, the Tx DMA, in addition, asserts the Tx Error interrupt. If it is 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 on the first descriptor fetch, the DMA stops and disables the queue. If it is on a non-first descriptor, the Rx_DMA, in addition, asserts the Rx Error interrupt.
2	Address Violation	RW 0x0	This bit is set if an Ethernet DMA violates a window access protection
3	Address NoMatch	RW 0x0	This bit is set if an Ethernet DMA address does not match any of the Ethernet address decode windows.

Table 400: Ethernet Unit Interrupt Cause (EUIC) (Continued)
Offset:0x72080

NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.

Bits	Field	Type/InitVal	Description
4	SMIdone	RW 0x0	SMI Command Done Indicates the SMI completed a MII management command (either read or write) that was initiated by the CPU writing to the SMI register.
5	Count_wa	RW 0x0	Counters Wrap Around Indication MIB Counter WrapAround Interrupt is set if one of the MIB counters wrapped around (passed 32 bits).
6	Reserved	RO 0x0	Reserved
7	Internal AddrError	RW 0x0	Internal Address Error is set when there is an access to an illegal offset of the internal registers. When set, the Internal Address Error register locks the address that caused the error.
8	Reserved	RO 0x0	Reserved
9	Port0_DPErr	RW 0x0	Port Internal data path parity error detected.
11:10	Reserved	RW 0x0	Reserved
12	TopDPErr	RW 0x0	G Top Internal data path parity error detected.
31:13	Reserved	RO 0x0	Reserved

Table 401: Ethernet Unit Interrupt Mask (EUIM)
Offset:0x72084

Bits	Field	Type/InitVal	Description
12:0	Various	RW 0x0	Mask bits for Unit Interrupt Cause register 0 = Mask 1 = Do not mask
31:13	Reserved	RO 0x0	Reserved

Table 402: Ethernet Unit Error Address (EUEA)
Offset:0x72094

Bits	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, 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 403: Ethernet Unit Internal Address Error (EUIAE)
Offset:0x72098

Bits	Field	Type/InitVal	Description
8:0	Internal Address	RO 0x0	If there is an address violation of unmapped access to the Gigabit Ethernet unit top registers, locks the relevant internal address bits (bit 12 and bits 9:2). The Address is locked until the register is read. **NOTE:** This field is used for software debugging after an address violation interrupt is raised.
31:9	Reserved	RO 0x0	Reserved

Table 404: Ethernet Unit Port Pads Calibration (EUPCR)
Offset:0x720A0

Bits	Field	Type/InitVal	Description
4:0	DrvN	RW 0xB	Pad Nchannel Driving Strength **NOTE:** Only applicable when auto-calibration is disabled.
15:5	Reserved	RO 0x0	Reserved
16	TuneEn	RW 0x0	Set to 1 enables the auto-calibration of pad driving strength.
21:17	LockN	RO 0x0	When auto-calibration is enabled, represents the final locked value of the Nchannel Driving Strength. Read Only
23:22	Reserved	RO 0x0	Reserved
28:24	Offset	RO 0x0	**NOTE:** Reserved for Marvell® usage.
30:29	Reserved	RO 0x0	Reserved

Table 404: Ethernet Unit Port Pads Calibration (EUPCR) (Continued)
Offset:0x720A0

Bits	Field	Type/InitVal	Description
31	WrEn	RW 0x0	Write Enable CPU Pads Calibration register 0 = Register is read only (except for bit [31]). 1 = Register is writable.

Table 405: Ethernet Unit Control (EUC)
Offset:0x720B0

Bits	Field	Type/InitVal	Description
0	Port0_DPPar	RW 0x0	Gigabit Ethernet port data path parity select: 0 = Even parity 1 = Odd parity **NOTE:** Must be set to even parity for normal operation. Odd parity is for debugging only.
2:1	Reserved	RW 0x0	Reserved
3	Top_DPPar	RW 0x0	Gigabit Ethernet Top data path parity select: 0 = Even parity 1 = Odd parity **NOTE:** Must be set to even parity for normal operation. Odd parity is for debugging only.
15:4	Reserved	RO 0x0	Reserved
16	Port0_PW	RW 0x1	Gigabit Ethernet port power management: 0 = Power Down (port deactivate) 1 = Power Up (normal operation) **NOTE:** Reserved for Marvell usage.
18:17	Reserved	RW 0x0	Reserved
31:19	Reserved	RO 0x0	Reserved

Table 406: Base Address
Offset:BA0 0x72200, BA1 0x72208, BA2 0x72210, BA3 0x72218, BA4 0x72220, BA5 0x72228

Bits	Field	Type/InitVal	Description
3:0	Target	RW 0x0	Specifies the target resource associated with this window. See Address Decoding chapter for full details
7:4	Reserved	RO 0x0	Reserved

Table 406: Base Address (Continued)
Offset:BA0 0x72200, BA1 0x72208, BA2 0x72210, BA3 0x72218, BA4 0x72220, BA5 0x72228

Bits	Field	Type/InitVal	Description
15:8	Attr	RW 0x0	Specifies target specific attributes depending on the target interface. See Address Decoding chapter for full details.
31:16	Base	RW 0x0	Base address Used together 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 Ethernet SDMAs is considered as a window hit if (`address | size) == (base | size`).

Table 407: Size (S)
Offset:SR0 0x72204, SR1 0x7220C, SR2 0x72214, SR3 0x7221C, SR4 0x72224, SR5 0x7222C

Bits	Field	Type/InitVal	Description
15:0	Reserved	RO 0x0	Reserved
31:16	Size	RW 0x0	Window size Used together with the size 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 1's followed by sequence of 0's. The number of 1's specifies the size of the window in 64 KB granularity (for example, a value of 0x00FF specifies 256x64k = 16 MB). An address driven by one of the Ethernet MACs is considered as a window hit if (`address | size) == (base | size`).

Table 408: High Address Remap (HA)[1]
Offset:HARR0 0x72280, HARR1 0x72284, HARR2 0x72288, HARR3 0x7228C

Bits	Field	Type/InitVal	Description
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.

1. 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.

Table 409: Base Address Enable (BARE)
Offset:0x72290

Bits	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 the Window0; bit[1] matches to Window1, etc.) 0 = Enable 1 = Disable
31:6	Reserved	RO 0x0	Reserved

Table 410: Ethernet Port Access Protect (EPAP)
Offset:0x72294

Bits	Field	Type/InitVal	Description
1:0	Win0	RW 0x3	Window0 access control 0x0 = No access allowed. 0x1 = Read Only 0x2 = Reserved 0x3 = Full access (read or write) In the 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.
3:2	Win1	RW 0x3	Window1 access control (the same as Win0 access control)
5:4	Win2	RW 0x3	Window2 access control (the same as Win0 access control)
7:6	Win3	RW 0x3	Window3 access control (the same as Win0 access control)
9:8	Win4	RW 0x3	Window4 access control (the same as Win0 access control)
11:10	Win5	RW 0x3	Window5 access control (the same as Win0 access control)
31:12	Reserved	RO 0x0	Reserved

A.9.2 Port Control Registers

Table 411: Port Configuration (GEC)
Offset:0x72400

Bits	Field	Type/InitVal	Description
0	UPM	RW 0x0	Unicast Promiscuous mode 0 = Normal mode. 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 mode. Unicast unmatched frames are received in the Rx queue.
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.
6:4	RXQArp	RW 0x0	Default Rx Queue for ARP Broadcasts, if receiving ARP Broadcasts is enabled.
7	RB	RW 0x0	Reject mode of MAC Broadcasts that are not IP or ARP Broadcast. 0 = Receive to the RXQ queue 1 = Reject
8	RBIP	RW 0x0	Reject mode of MAC Broadcasts that are IP (Ethertype 0x800). 0 = Receive to the RXQ queue 1 = Reject
9	RBArp	RW 0x0	Reject mode of MAC Broadcasts that are ARP (Ethertype 0x806). 0 = Receive to RXQArp queue 1 = Reject
11:10	Reserved	RW 0x0	Reserved
12	AMNoTxES	RW 0x0	Automatic mode not updating Error Summary in Tx descriptor The advantage of using this bit is that it avoids another write to memory to update the error status.
13	Reserved	RW 0x0	Reserved Must be set to 0.
14	TCP_CapEn	RW 0x0	Capture TCP frames to <TCPQ>. 1 = Enable 0 = Disable
15	UDP_CapEn	RW 0x0	Capture UDP frames to <UDPQ>. 1 = Enable 0 = Disable
18:16	TCPQ	RW 0x0	Captured TCP frames are directed to this Queue number.
21:19	UDPQ	RW 0x0	Captured UDP frames are directed to this Queue number.
24:22	BPDUQ	RW 0x7	Captured BPDU frames (if PCXR<Span> is set) are directed to this Queue number.

Table 411: Port Configuration (GEC) (Continued)
Offset:0x72400

Bits	Field	Type/InitVal	Description
25	RxCS	RW 0x1	Rx TCP checksum mode 0 = Calculate without pseudo header. 1 = Calculation include pseudo header.
31:26	Reserved	RO 0x0	Reserved

Table 412: Port Configuration Extend (GECX)
Offset:0x72404

Bits	Field	Type/InitVal	Description
0	Reserved	RO 0x0	Reserved
1	Span	RW 0x0	Spanning Tree packets capture enable 0 = BPDU packets are treated as normal Multicast packets. 1 = BPDU packets are trapped and sent to the Port Configuration register BPDU queue.
2	Reserved	RO 0x0	Reserved
31:3	Reserved	RO 0x0	Reserved

Table 413: MII Serial Parameters
Offset:0x72408

Bits	Field	Type/InitVal	Description
1:0	JAM LENGTH	RW 0x3	These two bits determine the JAM Length (in Back Pressure) as follows: 00 = 12K bit times 01 = 24K bit times 10 = 32K bit times 11 = 48K bit times **NOTE:** These bits can only be changed when <PortEn> field is set to 0 in the Port Control Register (Port is disabled).
6:2	JAM-IPG	RW 0x8	These five bits determine the JAM IPG. The step is 4-bit times. The <JAM IPG> varies between 4- and 124-bit times. **NOTE:** These bits can only be changed when <PortEn> field is set to 0 in the Port Control Register (Port is disabled). The <JAM IPG> bit cannot be programmed to 0 or 1.

Table 413: MII Serial Parameters (Continued)
Offset:0x72408

Bits	Field	Type/InitVal	Description
11:7	IPG-JAM_TO_DATA	RW 0x10	These five bits determine the IPG JAM to DATA. The step is 4-bit times. The value may vary between 4- and 128-bit times. **NOTE:** These bits can only be changed when <PortEn> field is set to 0 in the Port Control Register (Port is disabled).
16:12	IPG-DATA	RW 0x18	Inter-Packet Gap (IPG) The step is 4-bit times. The value may vary between 12- and 124-bit times. **NOTE:** These bits can only be changed when <PortEn> field is set to 0 in the Port Control Register (Port is disabled).
21:17	DataBlind	RW 0x10	Data Blinder The number of nibbles from the beginning of the IFG, in which the port restarts the IFG counter when detecting a carrier activity. Following this value, the port enters the Data Blinder zone and does not reset the IFG counter. This ensures fair access to the medium. The value must be written in hexadecimal format. The default is 10 hex (64-bit times; 2/3 of the default IPG). The step is 4-bit times. Valid range is 3 to 1F hex nibbles. **NOTE:** These bits can only be changed when <PortEn> field is set to 0 in the Port Control Register (Port is disabled).
31:22	Reserved	RO 0x0	Reserved

Table 414: GMII Serial Parameters
Offset:0x7240C

Bits	Field	Type/InitVal	Description
2:0	IPG-DATA	RW 0x6	Inter-Packet Gap (IPG) The step is 16-bit times. The value may vary between 48- to 112-bit times. GMII is full-duplex only. **NOTE:** These bits may be changed only when <PortEn> field is set to 0 in the Port Control Register (Port is disabled).
31:3	Reserved	RO 0x0	Reserved

Table 415: VLAN EtherType (EVLANE)
Offset:0x72410

Bits	Field	Type/InitVal	Description
15:0	VL_EtherType	RW 0x8100	The Ethertype for packets carrying the VLAN tag, for IEEE 802.1q priority field processing, and for continued parsing of the received frames Layer3/4 headers.
31:16	Reserved	RO 0x0	Reserved

Table 416: MAC Address Low (MACAL)
Offset:0x72414

Bits	Field	Type/InitVal	Description
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.
31:16	Reserved	RO 0x0	Read Only

Table 417: MAC Address High (MACAH)
Offset:0x72418

Bits	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 418: SDMA Configuration (SDC)
Offset:0x7241C

Bits	Field	Type/InitVal	Description
0	RIFB	RW 0x0	Receive Interrupt on Frame Boundaries When set, the SDMA Rx generates interrupts only on frame boundaries (i.e. after writing the frame status to the descriptor). See also the <IPGIntRx> field description (bits[21:8]) for further masking.
3:1	RxBSZ	RW 0x4	Rx Burst Size Sets the maximum burst size for Rx SDMA transactions: 000 = Burst is limited to 1 64-bit words. 001 = Burst is limited to 2 64-bit words. 010 = Burst is limited to 4 64-bit words. 011 = Burst is limited to 8 64-bit words. 100 = Burst is limited to 16 64-bit words. **NOTE:** This field effects only data-transfers. Descriptor fetch is done always with 4LW burst size. Must not be changed (other values degrade performance). However, a larger value is optimal for DDR SDRAM performance.
4	BLMR	RW 0x1	Big/Little Endian Receive Mode The DMA supports Big or Little Endian configurations per channel. The BLMR bit only affects data transfer to memory. 1 = No swap. 0 = Byte swap.

Table 418: SDMA Configuration (SDC) (Continued)
Offset:0x7241C

Bits	Field	Type/InitVal	Description
5	BLMT	RW 0x1	Big/Little Endian Transmit Mode The DMA supports Big or Little Endian configurations per channel. The BLMT bit only affects data transfer from memory. 1 = No swap. 0 = Byte swap.
6	SwapMode	RW 0x0	Swap mode The DMA supports swapping, for descriptors only, for both receive and transmit ports, on every access to memory space. 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.
7	Reserved	RW 0x0	Reserved
21:8	IPGIntRx	RW 0x0	Rx frame IPG between interrupts counter and enable This field provides a way to force a delay from the last ICR[RxBufferQueue] interrupt from any of the queues, to the next RxBufferQueue interrupt from any of the queues. The ICR bits still reflect the new interrupt, but this masking is reflected by potentially not propagating to the device's main interrupt cause register. This provides a way for interrupt coalescing on receive packet events. The time is calculated in multiples of 64 clock cycles. Valid values are 0 (No delay between packets to CPU, the counter is effectively disabled.), through 0x3FFF (1,048,544 clock cycles).
24:22	TxBSZ	RW 0x4	Tx Burst Size Sets the maximum burst size for Tx SDMA transactions: 000 = Burst is limited to 1 64-bit words. 001 = Burst is limited to 2 64-bit words. 010 = Burst is limited to 4 64-bit words. 011 = Burst is limited to 8 64-bit words. 100 = Burst is limited to 16 64-bit words. **NOTE:** This field effects only data-transfers. Descriptor fetch is done always with 4LW burst size. Must not be changed (other values degrade performance). However, a larger value is optimal for DDR SDRAM performance.
31:25	Reserved	RO 0x0	Read Only

Table 419: IP Differentiated Services CodePoint 0 to Priority (DSCP0)
Offset:0x72420

Bits	Field	Type/InitVal	Description
29:0	TOS_Q[29:0]	RW 0x0	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]). **NOTE:** The initial value means that ToS does not effect queue decisions.
31:30	Reserved	RO 0x0	Reserved

Table 420: IP Differentiated Services CodePoint 1 to Priority (DSCP1)
Offset:0x72424

Bits	Field	Type/InitVal	Description
29:0	TOS_Q[59:30]	RW 0x0	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]). **NOTE:** The initial value means that ToS does not effect queue decisions.
31:30	Reserved	RO 0x0	Reserved

Table 421: IP Differentiated Services CodePoint 2 to Priority (DSCP2, DSCP3, DSCP4, DSCP5)
Offset:DSCP2 0x72428, DSCP3 0x7242C, DSCP4 0x72430, DSCP5 0x72434

Bits	Field	Type/InitVal	Description
29:0	TOS_Q[89:60]	RW 0x0	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]). **NOTE:** The initial value means that ToS does not effect queue decisions.
31:30	Reserved	RO 0x0	Reserved

Table 422: IP Differentiated Services CodePoint 6 to Priority (DSCP6)
Offset:0x72438

Bits	Field	Type/InitVal	Description
11:0	TOS_Q[191:180]	RW 0x0	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]). **NOTE:** The initial value means that ToS does not effect queue decisions.
31:12	Reserved	RO 0x0	Reserved

Note The following steps for changing the value of the Port Serial Control register bits **do not** apply to <Force_Link_ Pass> bit[1], <ForceFCMode> bits[6:5], <ForceBPMode> bits[8:7], and <ForceLinkFail> bit [10].

When changing the value of the Port Serial Control register bits, the following steps must be taken:

- If the Tx DMA is enabled before, it must be disabled through its command registers. The CPU must verify that it is disabled by polling the TxQ Command register, then the CPU must poll each port's Port Status register to verify that:
  - There is no transmission in progress (<TxInProg> bit[7] is 0)
  - It's Tx FIFO is empty (<TcFIFOEmp> bit[10] is 1)
- Read the Port Serial Control register.
- Disable the Serial port by writing a 0 to bit[0] (<PortEn>) of the Port Serial Control register.
- Set the desired bits in the Port Serial Control register.
- Enable the Serial port by writing a 1 to bit[0] (<PortEn>) of the Port Serial Control register.
- Re-enable the Tx DMAs, according to their initialization sequence, as necessary.

Table 423: Port Serial Control (PSC)
Offset:0x7243C

Bits	Field	Type/InitVal	Description
0	PortEn	RW 0x0	Serial Port Enable No frames will be received of transmitted while serial port is disabled. 0 = Serial Port is disabled. 1 = Serial Port is enabled. **NOTE:** The port must not be disabled during Tx DMA operation. See guidelines above this table.
Link			
1	Force_Link_ Pass	RW 0x0	Force Link status on port to Link UP state 0 = Do NOT Force Link Pass 1 = Force Link pass
Duplex			
2	AN_Duplex	RW 0x0	Enable Auto-Negotiation for duplex mode 0 = Enable 1 = Disable **NOTE:** Half-Duplex mode is not supported in 1000 Mbps mode.
IEEE 802.3 Flow Control and Backpressure			
3	AN_FC	RW 0x1	Enable Auto-Negotiation for Flow Control 0 = Enable 1 = Disable When enabled, the port can either advertise no Flow Control or symmetric Flow Control according to the Port Serial Control Register's <Pause_Adv> bit. Asymmetric Flow Control advertisement is not supported.

Table 423: Port Serial Control (PSC) (Continued)
Offset:0x7243C

Bits	Field	Type/InitVal	Description
4	Pause_Adv	RW 0x1	Flow control advertise 0 = Advertise no Flow Control. 1 = Advertise symmetric Flow Control support in Auto-Negotiation. The port does not modify this bit as a result of the Auto-Negotiation process (unlike the Port Status Register's <EnFC> bit ,which may be modified based on Auto-Negotiation results).
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 PSR <EnFC> bit being set in the following way: 00 = 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. 01 = When this field is set to 01 value, *and* Flow Control is enabled (The PSR <EnFC> bit is set) then Pause disable frames (timer=0xFFFF) are retransmitted at least every 4.2 msec (1000 Mbps), 42 msec (100 Mbps), or 420 msec (10 Mbps). 10 = Reserved 11 = Reserved **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.
8:7	ForceBPMode	RW 0x0	When this bit is set, the port will start transmitting JAM on the line (Backpressure) in half-duplex (only supported in 10/100 Mbps) according to the following settings: 00 = No JAM (no backpressure) 01 = JAM is transmitted continuously, on next frame boundary. 10 = Reserved 11 = Reserved **NOTE:** When the link falls, this field goes to disabled 0x00 and must be reprogrammed only after the link is up.
9	Reserved	RW 0x1	Reserved Must be set to 1.
10	ForceLinkFail	RW 0x0	Force Link status on port to Link DOWN state 0 = Force Link Fail 1 = Do NOT Force Link Fail
12:11	Reserved	RW 0x0	Reserved
13	ANSpeed	RW 0x0	Enable Auto-Negotiation of interface speed in GMII mode 0 = Enable update 1 = Disable update
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.
16:15	Reserved	RW 0x0	Reserved

Table 423: Port Serial Control (PSC) (Continued)
Offset:0x7243C

Bits	Field	Type/InitVal	Description
19:17	MRU	RW 0x1	The Maximal Receive Packet Size 0 = Accept packets up to 1518 bytes in length 1 = Accept packets up to 1522 bytes in length 2 = Accept packets up to 1552 bytes in length 3 = Accept packets up to 9022 bytes in length 4 = Accept packets up to 9192 bytes in length 5 = Accept packets up to 9700 bytes in length 6–7 = Reserved **NOTE:** Modes 3–5 are supported only when operating in 1000 Mbps mode. Receiving 9700 byte frames is supported *only* during 1000 Mbps operation. *Receiving frames over 2 KB during 100 Mbps operation may result in overrun/underrun in some cases*.
20	Reserved	RW 0x0	Reserved
21	Set_FullDx	RW 0x1	Half/Full Duplex Mode 0 = Port works in Half Duplex mode. 1 = Port works in Full Duplex mode. **NOTE:** This bit is meaningless when the PSCR's <AN_Duplex> bit is set to enable.
22	SetFCEn	RW 0x1	Enable receiving and transmitting of IEEE 802.3 Flow Control frames in full duplex, Or enabling of backpressure in half duplex. 0 = Disabled 1 = Enabled **NOTE:** This bit is meaningless when this PSCR<AN_FC> is set to enable.
23	SetGMIISpeed	RW 0x1	0 = Port works at 10/100 Mbps. 1 = Port works at 1000 Mbps. **NOTE:** This bit is meaningless when <ANSpeed> is set to enable.
24	SetMIISpeed	RW 0x1	If Speed Auto-Negotiation is disabled (<ANSpeed> = 1) and <SetGMIISpeed> = 0, then this bit sets the speed of the MII interface. 0 = Port works in 10 Mbps 1 = Port works in 100 Mbps **NOTE:** This bit is meaningless when <ANSpeed> is enabled.
25	Reserved	RW 0x0	Reserved
27:26	Reserved	RW 0x0	Reserved Must be 0.
31:28	Reserved	RO 0x0	Read Only

Table 424: VLAN Priority Tag to Priority (VPT2P)
Offset:0x72440

Bits	Field	Type/InitVal	Description
23:0	Priority[23:0]	RW 0x0	The Priority queue mapping of received frames with IEEE 802.1q 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.
31:24	Reserved	RO 0x0	Reserved

Table 425: Ethernet Port Status (PS)
Offset:0x72444

Bits	Field	Type/InitVal	Description
0	Reserved	RO Sampled at reset 0x0	Reserved
Link			
1	LinkUp	RO 0x0	The Link Status 0 = Link is down. 1 = Link is up. This bit is set forced to 1 when <Force_Link_ Pass> = 1. This bit is set forced to 0 when <ForceLinkFail> = 0.
Duplex			
2	FullDx	RW Determined by Auto-Negotiation for duplex mode, if the Port Control register's <AN_Duplex> bit is enabled.	Half-/Full-Duplex mode. 0 = Port works in Half-Duplex mode. 1 = Port works in Full-Duplex mode. This bit may change in any time when the <AN_Duplex> bit is set to enable. When <AN_Duplex> in clear (disabled), this bit is set by the management in PSCR <Set_FullDx>. Read Only. **NOTE:** Half-Duplex is not supported in 1000 Mbps.

Table 425: Ethernet Port Status (PS) (Continued)
Offset:0x72444

Bits	Field	Type/InitVal	Description
802.3 Flow Control and Backpressure			
3	EnFC	RO	**NOTE:** Set by Flow_Control Auto-Negotiation if PSCR<AN_FC> is enabled. Enables receiving IEEE 802.3 Flow_Control frames in full-duplex mode: 0 = Disabled 1 = Enabled If Port Control register's <AN_FC>bit is enabled, then each time that Auto-Negotiation is performed on the GMII/MII/RGMII interface, the value in the <EnFC> bit may change.
4	GMIISpeed	RO	**NOTE:** Determined by Auto-Negotiation for speed mode when AN_Speed is enabled. 0 = Port works in 10/100 Mbps mode. 1 = Port works in 1000 Mbps mode. If the <ANSpeed> bit is enabled, then each time that Auto-Negotiation is performed, the value in the <GMIISpeed> field may change. When this bit is 0, the <MIISpeed> defines whether it is 10 Mbps or 100 Mbps. When the <ANSpeed> bit is disabled, this bit is set by the management in the PSCR <SetGMIISpeed>.
5	MIISpeed	RO Auto-Negotiation for duplex mode when AN_Speed is enabled.	MII Speed This field is meaningful only when <GMIISpeed> = 10/100 Mbps. 0 = Port works at 10 Mbps 1 = Port works at 100 Mbps If <ANSpeed> is enabled, then each time that Auto-Negotiation is performed, the value in the <MIISpeed> may change. When <ANSpeed> is disabled, this bit is set by the management in PSCR <SetMIISpeed>.
7	TxInProg	RO 0x0	Transmit in Progress Indicates that the port's transmitter is in an active transmission state.
9:8	Reserved	RO 0x0	Reserved
10	TxFIFOEmp	RO 0x0	Set when the port Transmit FIFO is empty.
31:11	Reserved	RO 0x0	Read Only.

Table 426: Transmit Queue Command (TQC)
Offset: 0x72448

Bits	Field	Type/InitVal	Description
0	ENQ	RW 0x0	Enable Queue Writing 1 enables the queue. The transmit DMA will fetch the first descriptor programmed to the FDP register for the queue and starts the transmit process. Writing 1 to the ENQ bit resets the matching DISQ bit. Writing 1 to the ENQ bit of a DMA that is already in enable state, has not effect. Writing 0 to the ENQ bit has no effect. When transmit DMA encounters queue end either by a null terminated descriptor pointer or a descriptor with a parity error or a CPU owned descriptor, The DMA will clear the ENQ bit for that queue. Thus reading these bits reports the active enable status for each queue. **NOTE:** The ENQ bits will be cleared on link down. After link is up, the CPU has to restart the DMA (set ENQ bits).
7:1	Reserved	RW 0x0	Reserved
8	DISQ	RW 0x0	Disable 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 queue end either by a null terminated descriptor pointer or by a CPU owned descriptor or by 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 disables by the CPU and those stopped by DMA due to null pointer or CPU owned descriptor or parity error on descriptor. **NOTE:** The DISQ bits will be cleared on link down.
31:9	Reserved	RO 0x0	Reserved

Table 427: Maximum Transmit Unit (MTU)
Offset:0x72458

Bits	Field	Type/InitVal	Description
5:0	Reserved	RW 9 KB/256 = 36 (=0x24)	Reserved. **NOTE:** Must be programmed to 0x0.
31:6	Reserved	RO 0x0	Reserved

Table 428: Port Interrupt Cause (IC)
Offset:0x72460

NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.

Bits	Field	Type/InitVal	Description
0	RxBuffer	RW 0x0	Rx Buffer Return Indicates a Rx buffer returned to CPU ownership or that the port finished reception of a Rx frame in either priority queues. **NOTE:** To get a Rx Buffer return per priority queue, use bits[9:2]. To limit the interrupts to frame (rather than buffer) boundaries, set the SDCR<RIFB> bit.
1	Extend	RW 0x0	Interrupt Cause Extend register (ICERx) of this port has a bit set.
2	RxBufferQueue[0]	RW 0x0	Rx Buffer Return in Priority Queue[0] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[0].
3	RxBufferQueue[1]	RW 0x0	Rx Buffer Return in Priority Queue[1] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[1].
4	RxBufferQueue[2]	RW 0x0	Rx Buffer Return in Priority Queue[2] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[2].
5	RxBufferQueue[3]	RW 0x0	Rx Buffer Return in Priority Queue[3] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[3].
6	RxBufferQueue[4]	RW 0x0	Rx Buffer Return in Priority Queue[4] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[4].
7	RxBufferQueue[5]	RW 0x0	Rx Buffer Return in Priority Queue[5] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[5].
8	RxBufferQueue[6]	RW 0x0	Rx Buffer Return in Priority Queue[6] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[6].
9	RxBufferQueue[7]	RW 0x0	Rx Buffer Return in Priority Queue[7] indicates a Rx buffer returned to CPU ownership or that the port completed reception of a Rx frame in receive priority queue[7].
10	RxError	RW 0x0	Rx Resource Error indicates a Rx resource error event in either Rx priority queues. To get a Rx Resource Error Indication per priority queue, use bits[18:11].
11	RxErrorQueue[0]	RW 0x0	Rx Resource Error in Priority Queue[0] indicates a Rx resource error event in receive priority queue[0].
12	RxErrorQueue[1]	RW 0x0	Rx Resource Error in Priority Queue[1] indicates a Rx resource error event in receive priority queue[1].
13	RxErrorQueue[2]	RW 0x0	Rx Resource Error in Priority Queue[2] indicates a Rx resource error event in receive priority queue[2].

Table 428: Port Interrupt Cause (IC) (Continued)
Offset:0x72460

NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.

Bits	Field	Type/InitVal	Description
14	RxErrorQueue[3]	RW 0x0	Rx Resource Error in Priority Queue[3] indicates a Rx resource error event in receive priority queue[3].
15	RxErrorQueue[4]	RW 0x0	Rx Resource Error in Priority Queue[0] indicates a Rx resource error event in receive priority queue[4].
16	RxErrorQueue[5]	RW 0x0	Rx Resource Error in Priority Queue[1] indicates a Rx resource error event in receive priority queue[5].
17	RxErrorQueue[6]	RW 0x0	Rx Resource Error in Priority Queue[2] indicates a Rx resource error event in receive priority queue[6].
18	RxErrorQueue[7]	RW 0x0	Rx Resource Error in Priority Queue[3] indicates a Rx resource error event in receive priority queue[7].
19	TxEnd	RW 0x0	Tx End indicates that the Tx DMA stopped processing the queue after a stop command (DISQ), or that it reached the end of the descriptor chain (through a null pointer or not owned descriptor).
30:20	Reserved	RW 0x0	Reserved
31	EtherIntSum	RO 0x0	Ethernet Interrupt Summary This bit is a logical OR of the (unmasked) bits[30:0] in the Interrupt Cause register of the port.

Table 429: Port Interrupt Cause Extend (ICE)
Offset:0x72464

NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.

Bits	Field	Type/InitVal	Description
0	TxBuffer	RW 0x0	Tx Buffer Indicates a Tx buffer returned to CPU ownership or that the port finished transmission of a Tx frame. **NOTE:** This bit is set upon closing any Tx descriptor that has its EI bit set. To limit the interrupts to frame (rather than buffer) boundaries, set EI only in the last descriptor.
7:1	Reserved	RW 0x0	Reserved
8	TxError	RW 0x0	Tx Resource Error Indicates a Tx resource error event during packet transmission from the queue.
15:9	Reserved	RW 0x0	Reserved

Table 429: Port Interrupt Cause Extend (ICE) (Continued)
Offset:0x72464

NOTE: Write 0 to clear interrupt bits. Writing 1 does not effect the interrupt bits.

Bits	Field	Type/InitVal	Description
16	PhySTC	RW 0x0	PHY Status Change Indicates a status change reported by the PHY connected to this port. If there is any change in the link, speed, duplex mode, or Flow Control capability as it is detected by the MDIO interface with the PHY, this interrupt will be set. This interrupt is set regardless of the actual changes in the status register, since Auto-Negotiation might be set to disabled on some of the parameters.
17	Reserved	RO 0x0	Reserved
18	RxOVR	RW 0x0	Rx Overrun Indicates an overrun event that occurred during reception of a packet.
19	TxUdr	RW 0x0	Tx Underrun Indicates an underrun event that occurred during transmission of packet from either queue.
20	LinkChange	RW 0x0	Link State Change This bit is set by upon a change in the link state (down->up, up->down).
21	Reserved	RW 0x0	Reserved
22	Reserved	RW 0x0	Reserved
23	InternalAddr Error	RW 0x0	Internal Address Error is set when there is an access to an illegal offset of the internal registers. When set, the Internal Address Error register locks the address that caused the error.
30:24	Reserved	RO 0x0	Reserved.
31	EtherIntSum	RO 0x0	Ethernet Interrupt Extend Summary This bit is a logical OR of the (unmasked) bits[30:0] in the Interrupt Cause Extend register.

Table 430: Port Interrupt Mask (PIM)
Offset:0x72468

Bits	Field	Type/InitVal	Description
26:0	Various	RW 0x0	Mask bits for Interrupt Cause register 0 = Mask 1 = Do not mask
31:27	Reserved	RO 0x0	Reserved

Table 431: Port Extend Interrupt Mask (PEIM)
Offset:0x7246C

Bits	Field	Type/InitVal	Description
23:0	Various	RW 0x0	Mask bits for Port Extend Interrupt Cause register 0 = Mask 1 = Do not mask
31:24	Reserved	RO 0x0	Reserved

Table 432: Port Rx FIFO Urgent Threshold (PRFUT)
Offset:0x72470

Bits	Field	Type/InitVal	Description
4:0	RxUThreshold	RW 0x10	Contains the Rx FIFO Threshold to start an Urgent indication. 0x0 = Disable Urgent 0x1–0x1F = Threshold for Urgent is 16–240 entries (128–1920 byte).
31:5	Reserved	RO 0x0	Reserved

Table 433: Port Tx FIFO Urgent Threshold (PTFUT)
Offset:0x72474

Bits	Field	Type/InitVal	Description
3:0	Reserved	RW 0x0	Must be 0.
17:4	IPGIntTx	RW 0x0	Tx frame IPG between interrupt counter and enable This field provides a way to force a delay from the last Port Interrupt Cause Extend (ICE) TxBuffer interrupt (Table 429 on page 431), from any of the queues, to the next TxBuffer interrupt, from any of the queues. The ICE bits still reflect the new interrupt, but this masking is reflected by potentially not propagating to the device's main interrupt cause register. This provides a way for interrupt coalescing on receive packet events. The time is calculated in multiples of 64 clock cycles. Valid values are 0 (no delay between packets to CPU, the counter is effectively disabled), through 0x3FFF (1,048,544 clock cycles).
31:18	Reserved	RO 0x0	Read Only.

Table 434: Port Rx Minimal Frame Size (PMFS)
Offset:0x7247C

Bits	Field	Type/InitVal	Description
1:0	RxMFS[1:0]	RO 0x0	Read Only.
6:2	RxMFS[6:2]	RW 0x10 (Corresponding to 64 bytes)	Contains the Receive Minimal Frame Size in bytes. Valid Range of <RxMFS[6:2]> is 0xA–0x10. (RxMFS = 40,44,48,52,56,60,64 bytes)
31:7	Reserved	RO 0x0	Read Only.

Table 435: Port Rx Discard Frame Counter (GEDFC)
Offset:0x72484

Bits	Field	Type/InitVal	Description
31:0	Rx Discard[31:0]	RO 0x0	Number of frames that were discarded because of a resource error. This register is reset every time the CPU reads from it.

Table 436: Port Overrun Frame Counter (POFC)
Offset:0x72488

Bits	Field	Type/InitVal	Description
31:0	Rx Overrun[31:0]	RO 0x0	Number of frames that were received and overrun This register is reset every time the CPU reads from it.

Table 437: Port Internal Address Error (EUIAE)
Offset:0x72494

Bits	Field	Type/InitVal	Description
8:0	InternalAddress	RO 0x0	Locks the relevant internal address bits (bit 12 and bits 9:2), if there is an address violation of unmapped access to the port registers. The Address is locked until the register is read.
31:9	Reserved	RO 0x0	Reserved

Table 438: Ethernet Current Receive Descriptor Pointers (CRDP)
Offset:Q0 0x7260C, Q1 0x7261C, Q2 0x7262C, Q3 0x7263C, Q4 0x7264C, Q5 0x7265C, Q6 0x7266C, Q7 0x7267C

Bits	Field	Type/InitVal	Description
31:0	RxCDP	RW 0x0	Receive Current Queue Descriptor Pointer

Table 439: Receive Queue Command (RQC)
Offset:0x72680

Bits	Field	Type/InitVal	Description
7:0	ENQ	RW 0x0	Enable Queue[7:0] One bit per each queue. Writing these bits set to 1 enables the queue. The Receive DMA will fetch the first descriptor programmed to the RxCDP register for that queue and start the Receive 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 not effect. Writing 0 to ENQ bit has no effect. When the receive DMA encounters a queue ended by a null terminated descriptor pointer or 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. **NOTE:** Reaching a CPU owned descriptor, a null terminated descriptor or a descriptor that is read with a parity error in the middle of a packet will result in closing the status of the packet with a resource error condition. Reaching a CPU owned descriptor either in the middle or on start of new packet will not result in disabling the DMA, and DMA will continue to read the descriptor it is using, when a new packet arrives to this queue, until it gets ownership of it. For these cases – a not owned descriptor, a null terminated descriptor, or a parity error on descriptor – the DMA will assert the resource RxErrorQueue interrupt.
15:8	DISQ	RW 0x0	Disable Queue[7:0] One bit per each queue.Writing these bits set to 1 disables the queue. The transmit 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 transmit DMA encounters a queue ended either by a null terminated descriptor pointer or 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 disables by CPU and those stopped by DMA due to a null pointer or a parity error on descriptor.
31:16	Reserved	RO 0x0	Read Only.

Table 440: Transmit Current Served Descriptor Pointer
Offset:0x72684

Bits	Field	Type/InitVal	Description
31:0	TxCDP	RO 0x0	Transmit Current Descriptor Pointer

Table 441: Transmit Current Queue Descriptor Pointer (TCQDP)
Offset:Q0 0x726C0

Bits	Field	Type/InitVal	Description
31:0	TxCDP	RW 0x0	Transmit Current Queue Descriptor Pointer

Table 442: Transmit Queue Token-Bucket Counter (TQxTBC)
Offset:Q0 0x72700, Q1 0x72710, Q2 0x72720, Q3 0x72730, Q4 0x72740, Q5 0x72750, Q6 0x72760, Q7 0x72770

NOTE: Transmit Queues 1–7 are reserved.

Bits	Field	Type/InitVal	Description
29:0	Reserved	RW Undefined. Must be initialized.	Reserved **NOTE:** Queue 0 (offset 0x72700) must be programmed to 0x3FFFFFFF. Queue 1 through 7 (offset 0x72710, 0x72720, 0x72730, 0x72740, 0x72750, 0x72760, 0x72770) must be programmed to 0x0.
31:30	Reserved	RO 0x0	Read only.

Table 443: Transmit Queue Token Bucket Configuration (TQxTBC)
Offset:Q0 0x72704, Q1 0x72714, Q2 0x72724, Q3 0x72734, Q4 0x72744, Q5 0x72754, Q6 0x72764, Q7 0x72774

NOTE: Transmit Queues 1–7 are reserved.

Bits	Field	Type/InitVal	Description
25:0	Reserved	RW Undefined Must be initialized.	Reserved **NOTE:** Queue 0 (offset 0x72704) must be programmed to 0x3FFFFFF. Queue 1 through 7 (offset 0x72714, 0x72724, 0x72734, 0x72744, 0x72754, 0x72764, 0x72774) must be programmed to 0x0.
31:26	Reserved	RO 0x0	Read Only

Table 444: Transmit Queue Arbiter Configuration (TQxAC)

Offset:Q0 0x72708, Q1 0x72718, Q2 0x72728, Q3 0x72738, Q4 0x72748, Q5 0x72758, Q6 0x72768, Q7 0x72778

NOTE: Transmit Queues 1–7 are reserved.

Bits	Field	Type/InitVal	Description
25:0	Reserved	RW Undefined Must be initialized.	Reserved **NOTE:** Queue 0 (offset 0x72708) must be programmed to 0xFF. Queue 1 through 7 (offset 0x72718, 0x72728, 0x72738, 0x72748, 0x72758, 0x72768, 0x72778) must be programmed to 0x0.
31:25	Reserved	RO 0x0	Reserved

Table 445: Destination Address Filter Special Multicast Table (DFSMT)

Offset:0x 73400–0x734FC

NOTE: Every register holds four entries. A total of 64 registers appear in the table in consecutive order.

Bits	Field	Type/InitVal	Description
0	Pass[0]	RW N/A	Determines whether to filter or accept, for pointer index 0. 0 = Reject (filter) frame 1 = Accept frame
3:1	Queue[0]	RW N/A	For pointer index 0: Determines the Queue number if Pass[0]=1.
4	Reserved[0]	RW N/A	Reserved Must be set to 0.
7:5	Unused[0]	RO N/A	Reserved
8	Pass[1]	RW N/A	Determines whether to filter or accept, for pointer index 1. 0 = Reject (filter) frame 1 = Accept frame
11:9	Queue[1]	RW N/A	For pointer index 1: Determines the Queue number if Pass[1]=1.
12	Reserved[1]	RW N/A	Reserved Must be set to 0.
15:13	Unused[1]	RO N/A	Reserved
16	Pass[2]	RW N/A	Determines whether to filter or accept, for pointer index 2. 0 = Reject (filter) frame 1 = Accept frame
19:17	Queue[2]	RW N/A	For pointer index 2: Determines the Queue number if Pass[2]=1.
20	Reserved[2]	RW N/A	Reserved Must be set to 0.

Table 445: Destination Address Filter Special Multicast Table (DFSMT) (Continued)
Offset:0x 73400–0x734FC

NOTE: Every register holds four entries. A total of 64 registers appear in the table in consecutive order.

Bits	Field	Type/InitVal	Description
23:21	Unused[2]	RO N/A	Reserved
24	Pass[3]	RW N/A	Determines whether to filter or accept, for pointer index 3. 0 = Reject (filter) frame 1 = Accept frame
27:25	Queue[3]	RW N/A	For pointer index 0: Determines the Queue number if Pass[3]=1.
28	Reserved[3]	RW N/A	Reserved Must be set to 0.
31:29	Unused[3]	RO N/A	Reserved

Table 446: Destination Address Filter Other Multicast Table (DFUT)
Offset:0x73500–0x735FC

NOTE: Every register holds four entries. A total of 64 registers appear in this table in consecutive order.

Bits	Field	Type/InitVal	Description
0	Pass[0]	RW N/A	Determines whether to filter or accept, for pointer index 0. 0 = Reject (filter) frame 1 = Accept frame
3:1	Queue[0]	RW N/A	For pointer index 0: Determines the Queue number if Pass[0]=1.
4	Reserved[0]	RW N/A	Reserved Must be set to 0.
7:5	Unused[0]	RO N/A	Reserved
8	Pass[1]	RW N/A	Determines whether to filter or accept, for pointer index 1. 0 = Reject (filter) frame 1 = Accept frame
11:9	Queue[1]	RW N/A	For pointer index 1: Determines the Queue number if Pass[1]=1.
12	Reserved[1]	RW N/A	Reserved Must be set to 0.
15:13	Unused[1]	RO N/A	Reserved
16	Pass[2]	RW N/A	Determines whether to filter or accept, for pointer index 2. 0 = Reject (filter) frame 1 = Accept frame

Table 446: Destination Address Filter Other Multicast Table (DFUT) (Continued)
Offset:0x73500–0x735FC

NOTE: Every register holds four entries. A total of 64 registers appear in this table in consecutive order.

Bits	Field	Type/InitVal	Description
19:17	Queue[2]	RW N/A	For pointer index 2: Determines the Queue number if Pass[2]=1.
20	Reserved[2]	RW N/A	Reserved Must be set to 0.
23:21	Unused[2]	RO N/A	Reserved
24	Pass[3]	RW N/A	Determines whether to filter or accept, for pointer index 3. 0 = Reject (filter) frame 1 = Accept frame
27:25	Queue[3]	RW N/A	For pointer index 3: Determines the Queue number if Pass[3]=1.
28	Reserved[3]	RW N/A	Reserved Must be set to 0.
31:29	Unused[3]	RO N/A	Reserved

Table 447: Destination Address Filter Unicast Table (DFUT)
Offset:0x73600–0x7360C

NOTE: Every register holds four entries. A total of four registers appear in this table in consecutive order.

Bits	Field	Type/InitVal	Description
0	Pass[0]	RW N/A	Determines whether to filter or accept, for pointer index 0. 0 = Reject (filter) frame 1 = Accept frame
3:1	Queue[0]	RW N/A	For pointer index 0: Determines the Queue number if Pass[0]=1.
4	Reserved[0]	RW N/A	Reserved Must be set to 0.
7:5	Unused[0]	RO N/A	Reserved
8	Pass[1]	RW N/A	Determines whether to filter or accept, for pointer index 1. 0 = Reject (filter) frame 1 = Accept frame
11:9	Queue[1]	RW N/A	For pointer index 1: Determines the Queue number if Pass[1]=1.
12	Reserved[1]	RW N/A	Reserved Must be set to 0.
15:13	Unused[1]	RO N/A	Reserved

Table 447: Destination Address Filter Unicast Table (DFUT) (Continued)
Offset:0x73600–0x7360C

NOTE: Every register holds four entries. A total of four registers appear in this table in consecutive order.

Bits	Field	Type/InitVal	Description
16	Pass[2]	RW N/A	Determines whether to filter or accept, for pointer index 2. 0 = Reject (filter) frame 1 = Accept frame
19:17	Queue[2]	RW N/A	For pointer index 2: Determines the Queue number if Pass[2]=1.
20	Reserved[2]	RW N/A	Reserved Must be set to 0.
23:21	Unused[2]	RO N/A	Reserved
24	Pass[3]	RW N/A	Determines whether to filter or accept, for pointer index 3. 0 = Reject (filter) frame 1 = Accept frame
27:25	Queue[3]	RW N/A	For pointer index 3: Determines the Queue number if Pass[3]=1.
28	Reserved[3]	RW N/A	Reserved Must be set to 0.
31:29	Unused[3]	RO N/A	Reserved

A.9.3 Port MIB Counter Register

Table 448: MAC MIB Counters
Offset:0x73000–0x7307C

NOTE: MIB counters are ROC (Read Only Clear). Read from MIB counter resets the value to 0.

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 802.3 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[6:2]> in the Port Rx Minimal Frame Size (PMFS) (Table 434 p. 434) 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 448: MAC MIB Counters (Continued)
Offset:0x73000–0x7307C

NOTE: MIB counters are ROC (Read Only Clear). Read from MIB counter resets the value to 0.

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.
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
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 IEEE 802.3 Single Collision Frames and IEEE 802.3 Multiple Collision Frames are not implement.

A.10 USB 2.0 Registers

Note **USB 2.0 Controller Registers (0x50000–0x502FF):** refer to *ARC USB-HS OTG High-Speed Controller Core reference V 4.0.1.* The base address for the controller registers is 0x50000. The offsets remain the same as in the above document.

Table 449: USB 2.0 Controller Register Map (Offsets Port0: 0x50000–0x502FF, Port1: 0xA0000–0xA02FF)

Register	Offset
ID	Port0: 0x50000, Port1: 0xA0000
HWGENERAL	Port0: 0x50004, Port1: 0xA0004
HWHOST	Port0: 0x50008, Port1: 0xA0008
HWDEVICE	Port0: 0x5000C, Port1: 0xA000C
HWTXBUF	Port0: 0x50010, Port1: 0xA0010
HWRXBUF	Port0: 0x50014, Port1: 0xA0014
HWTTTXBUF	Port0: 0x50018, Port1: 0xA0018
HWTTRXBUF	Port0: 0x5001C, Port1: 0xA001C
Reserved	Port0: 0x50020–0x500FC, Port1: 0xA0020–0xA00FC
CAPLENGTH	Port0: 0x50100, Port1: 0xA0100
Reserved	Port0: 0x50101, Port1: 0xA0101
HCIVERSION	Port0: 0x50102, Port1: 0xA0102
HCSPARAMS	Port0: 0x50104, Port1: 0xA0104
HCCPARAMS	Port0: 0x50108, Port1: 0xA0108
Reserved	Port0: 0x5010C-0x5011F, Port1: 0xA010C–0xA011F
DCIVERSION	Port0: 0x50120, Port1: 0xA0120
Reserved	Port0: 0x50122, Port1: 0xA0122
DCCPARAMS	Port0: 0x50124, Port1: 0xA0124
Reserved	Port0: 0x50128-0x5013C, Port1: 0xA0128–0xA013C
USBCMD	Port0: 0x50140, Port1: 0xA0140
USBSTS	Port0: 0x50144, Port1: 0xA0144
USBINTR	Port0: 0x50148, Port1: 0xA0148
FRINDEX	Port0: 0x5014C, Port1: 0xA014C
Reserved	Port0: 0x50150, Port1: 0xA0150
PERIODICLISTBASE / Device Addr	Port0: 0x50154, Port1: 0xA0154
ASYNCLISTADDR / Endpointlist Addr	Port0: 0x50158, Port1: 0xA0158
TTCTRL	Port0: 0x5015C, Port1: 0xA015C
BURSTSIZE	Port0: 0x50160, Port1: 0xA0160
TXFILLTUNING	Port0: 0x50164, Port1: 0xA0164
TXTTFILLTUNING	Port0: 0x50168, Port1: 0xA0168
N/A	Port0: 0x5016C, Port1: 0xA016C

Table 449: USB 2.0 Controller Register Map (Offsets Port0: 0x50000–0x502FF, Port1: 0xA0000–0xA02FF) (Continued)

Register	Offset
N/A	Port0: 0x50170-0x5017C, Port1: 0xA0170–0xA017C
CONFIGFLAG	Port0: 0x50180, Port1: 0xA0180
PORTSC1	Port0: 0x50184, Port1: 0xA0184
OTGSC	Port0: 0x501A4, Port1: 0xA01A4
USBMODE	Port0: 0x501A8, Port1: 0xA01A8
ENPDTSETUPSTAT	Port0: 0x501AC, Port1: 0xA01AC
ENDPTPRIME	Port0: 0x501B0, Port1: 0xA01B0
ENDPTFLUSH	Port0: 0x501B4, Port1: 0xA01B4
ENDPTSTATUS	Port0: 0x501B8, Port1: 0xA01B8
ENDPTCOMPLETE	Port0: 0x501BC, Port1: 0xA01BC
ENDPTCTRL0	Port0: 0x501C0, Port1: 0xA01C0
ENDPTCTRL1	Port0: 0x501C4, Port1: 0xA01C4
ENDPTCTRL2	Port0: 0x501C8, Port1: 0xA01C8
ENDPTCTRL3	Port0: 0x501CC, Port1: 0xA01CC

Table 450: USB 2.0 Bridge Register Map (Port0: 0x50300–0x503FF, Port1: 0xA0300–0xA03FF)

Register	Offset	Page
Bridge Control And Status Registers		
USB 2.0 Bridge Control Register	Port0: 0x50300, Port1: 0xA0300	Table 452, p. 445
Bridge Interrupt and Error Registers		
USB 2.0 Bridge Interrupt Cause Register	Port0: 0x50310	Table 453, p. 445
USB 2.0 Bridge Interrupt Mask Register	Port0: 0x50314, Port1: 0xA0314	Table 454, p. 446
USB 2.0 Bridge Error Address Register	Port0: 0x5031C, Port1: 0xA031C	Table 455, p. 446
Bridge Address Decoding Registers		
USB 2.0 Window0 Control Register	Port0: 0x50320, Port1: 0xA0320	Table 456, p. 446
USB 2.0 Window0 Base Register	Port0: 0x50324, Port1: 0xA0324	Table 457, p. 447
USB 2.0 Window1 Control Register	Port0: 0x50330, Port1: 0xA0330	Table 458, p. 447
USB 2.0 Window1 Base Register	Port0: 0x50334, Port1: 0xA0334	Table 459, p. 447
USB 2.0 Window2 Control Register	Port0: 0x50340, Port1: 0xA0340	Table 460, p. 448
USB 2.0 Window2 Base Register	Port0: 0x50344, Port1: 0xA0344	Table 461, p. 448

Table 450: USB 2.0 Bridge Register Map (Continued) (Port0: 0x50300–0x503FF, Port1:

Register	Offset	Page
USB 2.0 Window3 Control Register	Port0: 0x50350, Port1: 0xA0350	Table 462, p. 448
USB 2.0 Window3 Base Register	Port0: 0x50354, Port1: 0xA0354	Table 463, p. 449

Table 451: USB 2.0 PHY Register Map (Port0: 0x50400, Port1: 0xA0300)

Register	Offset	Page
USB 2.0 Power Control Register	Port0: 0x50400, Port1: 0xA0400	Table 464, p. 449

A.10.1 USB 2.0 Bridge Control and Status Registers

Table 452: USB 2.0 Bridge Control Register
Offset:Port0: 0x50300, Port1: 0xA0300

Bits	Field	Type/Init Val	Description
3:0	Reserved	RO 0x0	Reserved
7:4	Reserved	RW 0x0	Reserved
31:8	Reserved	RES 0x0	s

A.10.2 USB 2.0 Bridge Interrupt and Error Registers

Table 453: USB 2.0 Bridge Interrupt Cause Register[1]
Offset:Port0: 0x50310

Bit	Field	Type/Init Val	Description
0	AddrDecErr	RWC 0x0	Address Decoding Error Asserted upon address decoding error.
3:1	Reserved	RWC 0x0	Reserved
31:4	Reserved	RO 0x0	s

1. 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.

Table 454: USB 2.0 Bridge Interrupt Mask Register
Offset:Port0: 0x50314, Port1: 0xA0314

Bit	Field	Type/Init Val	Description
0	Mask	RW 0x0	If set to 1, the related interrupt is enabled.
3:1	Reserved	RW 0x0	Reserved
31:4	Reserved	RES 0x0	Reserved

Table 455: USB 2.0 Bridge Error Address Register
Offset:Port0: 0x5031C, Port1: 0xA031C

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.10.3 USB 2.0 Bridge Address Decoding Registers

Table 456: USB 2.0 Window0 Control Register
Offset:Port0: 0x50320, Port1: 0xA0320

Bits	Field	Type/InitVal	Function
0	win_en	RW 0x0	Window0 Enable 0x0 = Window is disabled. 0x1 = Window is enabled.
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See Section 2.10, Default Address Map, on page 22.
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this window. See Section 2.10, Default Address Map, on page 22.
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 1's followed by sequence of 0's. The number of 1's specifies the size of the window (e.g. a value of 0x00FF specifies 256x64k = 16 MB).

Table 457: USB 2.0 Window0 Base Register
Offset:Port0: 0x50324, Port1: 0xA0324

Bits	Field	Type/InitVal	Function
15:0	Reserved	RES 0x0	Reserved
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]

Table 458: USB 2.0 Window1 Control Register
Offset:Port0: 0x50330, Port1: 0xA0330

Bits	Field	Type/InitVal	Function
0	win_en	RW 0x0	Window0 Enable 0x0 = Window is disabled. 0x1 = Window is enabled.
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See Section 2.10, Default Address Map, on page 22.
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this window. See Section 2.10, Default Address Map, on page 22.
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 1's followed by sequence of 0's. The number of 1's specifies the size of the window (e.g. a value of 0x00FF specifies 256x64k = 16 MB).

Table 459: USB 2.0 Window1 Base Register
Offset:Port0: 0x50334, Port1: 0xA0334

Bits	Field	Type/InitVal	Function
15:0	Reserved	RES 0x0	Reserved
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]

Table 460: USB 2.0 Window2 Control Register
Offset:Port0: 0x50340, Port1: 0xA0340

Bits	Field	Type/InitVal	Function
0	win_en	RW 0x0	Window0 Enable 0x0 = Window is disabled. 0x1 = Window is enabled.
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See Section 2.10, Default Address Map, on page 22.
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this window. See Section 2.10, Default Address Map, on page 22.
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 1's followed by sequence of 0's. The number of 1's specifies the size of the window (e.g. a value of 0x00FF specifies 256x64k = 16 MB).

Table 461: USB 2.0 Window2 Base Register
Offset:Port0: 0x50344, Port1: 0xA0344

Bits	Field	Type/InitVal	Function
15:0	Reserved	RES 0x0	Reserved
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]

Table 462: USB 2.0 Window3 Control Register
Offset:Port0: 0x50350, Port1: 0xA0350

Bits	Field	Type/InitVal	Function
0	win_en	RW 0x0	Window0 Enable 0 = Window is disabled. 1 = Window is enabled.
3:1	Reserved	RES 0x0	Reserved
7:4	Target	RW 0x0	Specifies the target interface associated with this window. See Section 2.10, Default Address Map, on page 22.
15:8	Attr	RW 0x0	Specifies the target interface attributes associated with this window. See Section 2.10, Default Address Map, on page 22.

Table 462: USB 2.0 Window3 Control Register (Continued)
Offset:Port0: 0x50350, Port1: 0xA0350

Bits	Field	Type/InitVal	Function
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 1's followed by sequence of 0's. The number of 1's specifies the size of the window (e.g. a value of 0x00ff specifies 256x64k = 16 MB).

Table 463: USB 2.0 Window3 Base Register
Offset:Port0: 0x50354, Port1: 0xA0354

Bits	Field	Type/InitVal	Function
15:0	Reserved	RES 0x0	Reserved
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]

A.10.4 USB 2.0 PHY Registers

Table 464: USB 2.0 Power Control Register
Offset:Port0: 0x50400, Port1: 0xA0400

Bits	Field	Type/InitVal	Function
0	Pu	RW 0x1	Input Power Up
1	PuPll	RW 0x1	Input Power Up PLL
2	SUSPENDM	RW 0x1	Input SUSPENDM
3	VBUS_PWR_FAULT	RW 0x0	Vbus Power Fault Connect to the Core, not to the PHY.
4	PWRCTL_WAKEUP	RW 0x0	USB Power Control Wake Up Connect to the Core, not to the PHY.
5	PuRef	RW 0x1	Power Up Reference Connect to ana_grp.
7:6	BG_VSEL	RW 0x1	BG VSEL Connect to ana_grp.
8	REG_ARC_DPDM_MODE	RW 0x1	0 = Use register programmed pulldown. 1 = Use dp_pulldown and dm_pulldown from controller core.

Table 464: USB 2.0 Power Control Register (Continued)
Offset:Port0: 0x50400, Port1: 0xA0400

Bits	Field	Type/InitVal	Function
9	REG_DP_PULLDOWN	RW 0x0	Register DP Pull 0 = No DP pulldown 1 = Pull down DP.
10	REG_DM_PULLDOWN	RW 0x0	Register DM Pull 0 = No DM pulldown. 1 = Pull down DM.
22:11	Reserved	RW 0x0	Reserved
23	utmi_sessend	RW 0x0	UTMI Session End
24	utmi_vbus_valid	RW 0x1	UTMI Vbus Valid
25	utmi_avalid	RW 0x1	UTMI A Valid
26	utmi_bvalid	RW 0x1	UTMI B Valid
27	TX_BIT_STUFF	RW 0x1	Transmit Bit Stuff
31:28	Reserved	RW 0x1F	Reserved

A.11 Cryptographic Engine and Security Accelerator Registers

Table 465: Cryptographic Engine and Security Accelerator Register Map

Register	Offset	Table, Page
DES Engine Registers		
DES Data Out Low Register	0x9DD78	Table 466, p. 452
DES Data Out High Register	0x9DD7C	Table 467, p. 453
DES Data Buffer Low Register	0x9DD70	Table 468, p. 453
DES Data Buffer High Register	0x9DD74	Table 469, p. 453
DES Initial Value Low Register	0x9DD40	Table 470, p. 453
DES Initial Value High Register	0x9DD44	Table 471, p. 453
DES Key0 Low Register	0x9DD48	Table 472, p. 454
DES Key0 High Register	0x9DD4C	Table 473, p. 454
DES Key1 Low Register	0x9DD50	Table 474, p. 454
DES Key1 High Register	0x9DD54	Table 475, p. 454
DES Key2 Low Register	0x9DD60	Table 476, p. 454
DES Key2 High Register	0x9DD64	Table 477, p. 455
DES Command Register	0x9DD58	Table 478, p. 455
SHA-1 and MD5 Interface Registers		
SHA-1/MD5 Data In Register	0x9DD38	Table 479, p. 456
SHA-1/MD5 Bit Count Low Register	0x9DD20	Table 480, p. 456
SHA-1/MD5 Bit Count High Register	0x9DD24	Table 481, p. 456
SHA-1/MD5 Initial Value/Digest A Register	0x9DD00	Table 482, p. 456
SHA-1/MD5 Initial Value/Digest B Register	0x9DD04	Table 483, p. 457
SHA-1/MD5 Initial Value/Digest C Register	0x9DD08	Table 484, p. 457
SHA-1/MD5 Initial Value/Digest D Register	0x9DD0C	Table 485, p. 457
SHA-1 Initial Value/Digest E Register	0x9DD10	Table 486, p. 457
SHA-1/MD5 Authentication Command Register	0x9DD18	Table 487, p. 458
AES Encryption Interface Registers		
AES Encryption Data In/Out Column 3 Register	0x9DDA0	Table 488, p. 459
AES Encryption Data In/Out Column 2 Register	0x9DDA4	Table 489, p. 459
AES Encryption Data In/Out Column 1 Register	0x9DDA8	Table 490, p. 459
AES Encryption Data In/Out Column 0 RegisterA	0x9DDAC	Table 491, p. 460
AES Encryption Key Column 3 Register	0x9DD90	Table 492, p. 460
AES Encryption Key Column 2 Register	0x9DD94	Table 493, p. 460
AES Encryption Key Column 1 Register	0x9DD98	Table 494, p. 460
AES Encryption Key Column 0 Register	0x9DD9C	Table 495, p. 460
AES Encryption Key Column 7 Register	0x9DD80	Table 496, p. 461
AES Encryption Key Column 6 Register	0x9DD84	Table 497, p. 461
AES Encryption Key Column 5 Register	0x9DD88	Table 498, p. 461

Table 465: Cryptographic Engine and Security Accelerator Register Map (Continued)

Register	Offset	Table, Page
AES Encryption Key Column 4 Register	0x9DD8C	Table 499, p. 461
AES Encryption Command Register	0x9DDB0	Table 500, p. 461
AES Decryption Interface Registers		
AES Decryption Data In/Out Column 3 Register	0x9DDE0	Table 501, p. 462
AES Decryption Data In/Out Column 2 Register	0x9DDE4	Table 502, p. 462
AES Decryption Data In/Out Column 1 Register	0x9DDE8	Table 503, p. 463
AES Decryption Data In/Out Column 0 Register	0x9DDEC	Table 504, p. 463
AES Decryption Key Column 3 Register	0x9DDD0	Table 505, p. 463
AES Decryption Key Column 2 Register	0x9DDD4	Table 506, p. 463
AES Decryption Key Column 1 Register	0x9DDD8	Table 507, p. 463
AES Decryption Key Column 0 Register	0x9DDDC	Table 508, p. 464
AES Decryption Key Column 7 Register	0x9DDC0	Table 509, p. 464
AES Decryption Key Column 6 Register	0x9DDC4	Table 510, p. 464
AES Decryption Key Column 5 Register	0x9DDC8	Table 511, p. 464
AES Decryption Key Column 4 Register	0x9DDCC	Table 512, p. 464
AES Decryption Command Register	0x9DDF0	Table 513, p. 465
Security Accelerator Registers		
Security Accelerator Command Register	0x9DE00	Table 514, p. 465
Security Accelerator Descriptor Pointer Session 0 Register	0x9DE04	Table 515, p. 466
Security Accelerator Descriptor Pointer Session 1 Register	0x9DE14	Table 516, p. 466
Security Accelerator Configuration Register	0x9DE08	Table 517, p. 467
Security Accelerator Status Register	0x9DE0C	Table 518, p. 467
Interrupt Cause Registers		
Cryptographic Engines and Security Accelerator Interrupt Cause Register	0x9DE20	Table 519, p. 468
Cryptographic Engines and Security Accelerator Interrupt Mask Register	0x9DE24	Table 520, p. 469

A.11.1 DES Engine Registers

Table 466: DES Data Out Low Register
Offset: 0x9DD78

Bits	Field	Type/InitVal	Description
31:0	DataOutLo	RO NA	When the DES (or the Triple DES) completes the calculation, this field will contains the low bits of the DES result.

Table 467: DES Data Out High Register
Offset: 0x9DD7C

Bits	Field	Type/InitVal	Description
31:0	DataOutHi	RO NA	When the DES (or the Triple DES) completes the calculation, this field will contains the high bits of the DES result.

Table 468: DES Data Buffer Low Register
Offset: 0x9DD70

Bits	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 469: DES Data Buffer High Register
Offset: 0x9DD74

Bits	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 470: DES Initial Value Low Register
Offset: 0x9DD40

Bits	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 471: DES Initial Value High Register
Offset: 0x9DD44

Bits	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 472: DES Key0 Low Register
Offset: 0x9DD48

Bits	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 473: DES Key0 High Register
Offset: 0x9DD4C

Bits	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 474: DES Key1 Low Register
Offset: 0x9DD50

Bits	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 475: DES Key1 High Register
Offset: 0x9DD54

Bits	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 476: DES Key2 Low Register
Offset: 0x9DD60

Bits	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 477: DES Key2 High Register
Offset: 0x9DD64

Bits	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 478: DES Command Register
Offset: 0x9DD58

Bits	Field	Type/InitVal	Description
0	Direction	RW 0x0	This bit controls the direction of the operation: encryption or decryption. 0 = Encryption 1 = Decryption
1	Algorithm	RW 0x0	This bit controls whether the DES or Triple DES algorithm is used. 0 = DES 1= Triple DES (3DES)
2	TripleDESMode	RW 0x0	This bit controls the Triple DES encryption/decryption mode. 0 = EEE 1 = EDE
3	DESMode	RW 0x0	This bit controls the DEC encryption/decryption mode. 0 = ECB 1 = CBC
4	DataByteSwap	RW 0x0	This bit controls whether data byte swap is activated on input. 0 = No byte swap 1 = Byte swap
5	Reserved	RES 0x0	Reserved
6	IVByteSwap	RW 0x0	This bit controls whether initial value byte swap is activated. 0 = No byte swap 1 = Byte swap
7	Reserved	RES 0x0	Reserved
8	OutByteSwap	RES 0x0	This bit controls whether byte swap is activated for output. 0 = No byte swap 1 = Byte swap
28:9	Reserved	RES 0x0	Reserved
29	WriteAllow	RW 0x1	This bit indicates that the host can write data to the engine. 0 = Write not allowed. 1 = Write allowed.

Table 478: DES Command Register (Continued)
Offset: 0x9DD58

Bits	Field	Type/InitVal	Description
30	AllTermination	RW 0x1	This bit indicates to the host that the encryption calculation has been completed and that the encryption parameters may be updated and data may be written.
31	Termination	RO 0x1	This bit is set by the engine to indicate completion of a DES calculation process. Any write to the Encryption engine will clear this bit.

A.11.2 SHA-1 and MD5 Interface Registers

Table 479: SHA-1/MD5 Data In Register
Offset: 0x9DD38

Bits	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.

Table 480: SHA-1/MD5 Bit Count Low Register
Offset: 0x9DD20

Bits	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 481: SHA-1/MD5 Bit Count High Register
Offset: 0x9DD24

Bits	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 482: SHA-1/MD5 Initial Value/Digest A Register
Offset: 0x9DD00

Bits	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.

Table 483: SHA-1/MD5 Initial Value/Digest B Register
Offset: 0x9DD04

Bits	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.

Table 484: SHA-1/MD5 Initial Value/Digest C Register
Offset: 0x9DD08

Bits	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.

Table 485: SHA-1/MD5 Initial Value/Digest D Register
Offset: 0x9DD0C

Bits	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.

Table 486: SHA-1 Initial Value/Digest E Register
Offset: 0x9DD10

Bits	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 since SHA mode requires a 5-word initial value to produce the 5-word SHA signature.

Table 487: SHA-1/MD5 Authentication Command Register
Offset: 0x9DD18

Bits	Field	Type/InitVal	Description
0	Algorithm	RW 0x0	This bit controls the mode of operation: SHA-1 or MD5. 0 = MD5 1 = SHA1 These are two different algorithms for calculating the authentication signature. They are described in the references. • SHA calculation takes 85 clock cycles; where MD5 takes 65 clock cycles. • SHA mode results in a 5-word signature (and a 5-word initial value is required) where MD5 results in a 4-word signature (and a 4-word initial value is required). • The MD5 is byte swapped compared to the SHA. • These algorithms differ in their complexity and security levels, and it is left to the user to choose the algorithm.
1	Mode	RW 0x0	This bit controls whether the initial value is used or the operation continues from the last value. 0 = Use initial value 1 = Continue 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. After that 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 want to share the engine for multiple packet signature calculations. That 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. **NOTE:** When the host wants to use initial values other than the ones defined by the algorithm, Continue mode *must* be selected.
2	DataByteSwap	RW 0x0	This bit controls whether data-byte swap is activated. 0 = No byte swap (data to engine W0...W15 — 0x01234567) 1 = Byte swap (data to engine W0...W15 — 0x67452301) 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.
3	Reserved	RES 0x0	Reserved

Table 487: SHA-1/MD5 Authentication Command Register (Continued)
Offset: 0x9DD18

Bits	Field	Type/InitVal	Description
4	IVByteSwap	RW 0x0	This bit controls whether initial value byte swap is activated. 0 = No byte swap 1 = Byte swap This is the same as the data swap, but only for initial values written to the IV/Digest registers.
30:5	Reserved	RES 0x0	Reserved
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.

A.11.3 AES Encryption Interface Registers

Table 488: AES Encryption Data In/Out Column 3 Register
Offset: 0x9DDA0

Bits	Field	Type/InitVal	Description
31:0	AesEncDatCol3	RW NA	At first this field contains Column 3 of the input data block to be encrypted. When the AES completes the calculation, this field will contain the Column 3 of the AES result.

Table 489: AES Encryption Data In/Out Column 2 Register
Offset: 0x9DDA4

Bits	Field	Type/InitVal	Description
31:0	AesEncDatCol2	RW NA	At first this field contains Column 2 of the input data block to be encrypted. When the AES completes the calculation, this field will contain the Column 2 of the AES result.

Table 490: AES Encryption Data In/Out Column 1 Register
Offset: 0x9DDA8

Bits	Field	Type/InitVal	Description
31:0	AesEncDatCol1	RW NA	At first this field contains Column 1 of the input data block to be encrypted. When the AES completes the calculation, this field will contain the Column 1 of the AES result.

Table 491: AES Encryption Data In/Out Column 0 Register
Offset: 0x9DDAC

Bits	Field	Type/InitVal	Description
31:0	AesEncDatCol0	RW NA	At first this field contains Column 0 of the input data block to be encrypted. When the AES completes the calculation, this field will contain the Column 0 of the AES result.

Table 492: AES Encryption Key Column 3 Register
Offset: 0x9DD90

Bits	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 493: AES Encryption Key Column 2 Register
Offset: 0x9DD94

Bits	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 494: AES Encryption Key Column 1 Register
Offset: 0x9DD98

Bits	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 495: AES Encryption Key Column 0 Register
Offset: 0x9DD9C

Bits	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 496: AES Encryption Key Column 7 Register
Offset: 0x9DD80

Bits	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 497: AES Encryption Key Column 6 Register
Offset: 0x9DD84

Bits	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 498: AES Encryption Key Column 5 Register
Offset: 0x9DD88

Bits	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 499: AES Encryption Key Column 4 Register
Offset: 0x9DD8C

Bits	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 500: AES Encryption Command Register
Offset: 0x9DDB0

Bits	Field	Type/InitVal	Description
1:0	AesEncKeyMode	RW 0x0	This field specifies the AES128 key size used. 00 = 128-bit key 01 = 192-bit key 10 = 256-bit key 11 = Reserved
3:2	Reserved	RES 0x0	Reserved

Table 500: AES Encryption Command Register (Continued)
Offset: 0x9DDB0

Bits	Field	Type/InitVal	Description
4	DataByteSwap	RW 0x0	This bit controls whether data byte swap is activated on input. 0 = No byte swap 1 = Byte swap
7:5	Reserved	RES 0x0	Reserved
8	OutByteSwap	RW 0x0	This bit controls whether byte swap is activated for output. 0 = No byte swap 1 = Byte swap
30:9	Reserved	RES 0x0	Reserved
31	Termination	RO 0x1	This bit is set by the engine to indicate completion of a AES calculation process. Any write to the Encryption engine will clear this bit.

A.11.4 AES Decryption Interface Registers

Table 501: AES Decryption Data In/Out Column 3 Register
Offset: 0x9DDE0

Bits	Field	Type/InitVal	Description
31:0	AesDecDatCol3	RW NA	At first this field contains Column 3 of the input data block to be decrypted. When the AES completes the calculation, this field will contain the Column 3 of the AES result.

Table 502: AES Decryption Data In/Out Column 2 Register
Offset: 0x9DDE4

Bits	Field	Type/InitVal	Description
31:0	AesDecDatCol2	RW NA	At first this field contains Column 2 of the input data block to be decrypted. When the AES completes the calculation, this field will contain the Column 2 of the AES result.

Table 503: AES Decryption Data In/Out Column 1 Register
Offset: 0x9DDE8

Bits	Field	Type/InitVal	Description
31:0	AesDecDatCol1	RW NA	At first this field contains Column 1 of the input data block to be decrypted. When the AES completes the calculation, this field will contain the Column 1 of the AES result.

Table 504: AES Decryption Data In/Out Column 0 Register
Offset: 0x9DDEC

Bits	Field	Type/InitVal	Description
31:0	AesDecDatCol0	RW NA	At first this field contains column 0 of the input data block to be decrypted. When the AES completes the calculation, this field will contain the Column 0 of the AES result.

Table 505: AES Decryption Key Column 3 Register
Offset: 0x9DDD0

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol3	RW 0x0	Contains Column 3 of the AES decryption key

Table 506: AES Decryption Key Column 2 Register
Offset: 0x9DDD4

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol2	RW 0x0	Contains Column 2 of the AES decryption key

Table 507: AES Decryption Key Column 1 Register
Offset: 0x9DDD8

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol1	RW 0x0	Contains Column 1 of the AES decryption key

Table 508: AES Decryption Key Column 0 Register
Offset: 0x9DDDC

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol0	RW 0x0	Contains Column 0 of the AES decryption key

Table 509: AES Decryption Key Column 7 Register
Offset: 0x9DDC0

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol7	RW 0x0	Contains Column 7 of the AES decryption key

Table 510: AES Decryption Key Column 6 Register
Offset: 0x9DDC4

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol6	RW 0x0	Contains Column 6 of the AES decryption key

Table 511: AES Decryption Key Column 5 Register
Offset: 0x9DDC8

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol5	RW 0x0	Contains Column 5 of the AES decryption key

Table 512: AES Decryption Key Column 4 Register
Offset: 0x9DDCC

Bits	Field	Type/InitVal	Description
31:0	AesDecKeyCol4	RW 0x0	Contains Column 4 of the AES decryption key

Table 513: AES Decryption Command Register
Offset: 0x9DDF0

Bits	Field	Type/InitVal	Description
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
2	AesDecMakeKey	RW 0x0	This bits controls whether the decryption key is calculated in the engine prior to the data decryption. 0 = No decryption key calculation 1 = Decryption key calculation
3	Reserved	RES 0x0	Reserved
4	DataByteSwap	RW 0x0	This bit controls whether data byte swap is activated on input. 0 = No byte swap 1 = Byte swap
7:5	Reserved	RES 0x0	Reserved
8	OutByteSwap	RW 0x0	This bit controls whether byte swap is activated for output. 0 = No byte swap 1 = Byte swap
30:9	Reserved	RES 0x0	Reserved
31	Termination	RO 0x1	This bit is set by the engine to indicate completion of a AES calculation process. Any write to the Decryption engine will clear this bit.

A.11.5 Security Accelerator Registers

Table 514: Security Accelerator Command Register
Offset: 0x9DE00

Bits	Field	Type/InitVal	Description
0	EnSecurityAccl0	RW 0x0	Setting this bit activates session 0 of the accelerator. After operation completion, this bit is cleared to zero by the hardware. Writing zero to this bit has no effect. Security acceleration assures in-order execution and completion. If session 0 is activated before session 1 (by setting this bit before bit <EnSecurityAccl1>), the Security acceleration always execute session 0 before session 1. 0 = Session 0 is idle. 1 = Session 0 is set to active.

Table 514: Security Accelerator Command Register (Continued)
Offset: 0x9DE00

Bits	Field	Type/InitVal	Description
1	EnSecurityAccl1	RW 0x0	Setting this bit activates session 1 of the accelerator. After operation completion this bit is cleared to zero by the hardware. Writing zero to this bit has no effect. Security acceleration assures in-order execution and completion. If session 1 is activated before session 0 (by setting this bit before bit <EnSecurityAccl0>), the Security acceleration always execute session 1 before session 0. 0 = Session 0 is idle. 1 = Session 0 is set to active.
2	DsSecurityAccl	ARZ 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, AccInt1 bits are cleared.
31:3	Reserved	RO 0x0	Reserved

Table 515: Security Accelerator Descriptor Pointer Session 0 Register
Offset: 0x9DE04

Bits	Field	Type/InitVal	Description
15:0	SecurityAccl DescPtr0	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 and are read as 0 regardless of programming.
31:16	Reserved	RW 0x0	Reserved

Table 516: Security Accelerator Descriptor Pointer Session 1 Register
Offset: 0x9DE14

Bits	Field	Type/InitVal	Description
15:0	SecurityAccl DescPtr1	RW 0x0	Security accelerator descriptor pointer for session 1 (DWORD aligned) Bits [16], [17], [18], [29], [30] and [31] are reserved and are assumed to be and are as 0 regardless of programming.
31:16	Reserved	RW 0x0	Reserved

Table 517: Security Accelerator Configuration Register
Offset: 0x9DE08

Bits	Field	Type/InitVal	Description
0	StopOnDecodeDigestErr	RW 0x1	Controls whether the engine stops when digest error in decode. 0 = Do not stop on digest decode error 1 = Stop on digest decode error
1	Reserved	RW 0x0	Must be 0.
6:2	Reserved	RES 0x0	Reserved
7	Ch0WaitForIDMA	RW 0x0	Channel 0 Wait for IDMA When set to 1, Security channel 0 is activated only when bit[4] channel 0 <Own> field in the Interrupt Cause Register (Table 564 p. 495) is set to 1.
8	Ch1WaitForIDMA	RW 0x0	Channel 1 Wait for IDMA When set to 1, Security accelerator channel 1 is activated only when bit[12] channel 1 <Own> field in the Interrupt Cause Register (Table 564 p. 495) is set to 1.
9	Ch0ActivateIDMA	RW 0x0	Channel 0 Activation for IDMA When set to 1, Security accelerator channel 0 activates the IDMA channel 0 when bit[5] in the Cryptographic Engines and Security Accelerator Interrupt Cause Register (Table 519 p. 468) is set to 1.
10	Ch1ActivateIDMA	RW 0x0	Channel 1 Activation for IDMA When set to 1, Security accelerator channel 1 activates the IDMA channel 1 when bit[6] in the Cryptographic Engines and Security Accelerator Interrupt Cause Register (Table 519 p. 468) is set to 1.
31:11	Reserved	RES 0x0	Reserved

Table 518: Security Accelerator Status Register
Offset: 0x9DE0C

Bits	Field	Type/InitVal	Description
0	SecurityActive0	RO 0x0	State of the session 0 This bit equals <AccInt0>. 0 = Session 0 is idle. 1 = Session 0 is active.
1	SecurityActive1	RO 0x0	State of the session 1 This bit equals <AccInt1>. 0 = Session 0 is idle. 1 = Session 0 is active.
7:2	Reserved	RO 0x0	Reserved

Table 518: Security Accelerator Status Register (Continued)
Offset: 0x9DE0C

Bits	Field	Type/InitVal	Description
8	DecodeDigestErr0	RO 0x0	Signals a decode digest error during session 0. This bit is cleared when session 0 is activated.
9	DecodeDigestErr1	RO 0x0	Signals a decode digest error during session 1. This bit is cleared when session 1 is activated.
12:10	Reserved	RO 0x0	Reserved
31:13	AcclState	RO 0x0	Internal State of the accelerator

A.11.6 Interrupt Cause Registers

Table 519: Cryptographic Engines and Security Accelerator Interrupt Cause Register
Offset: 0x9DE20

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.

Bits	Field	Type/InitVal	Description
0	ZInt0	RW0 0x0	This bit is the authentication termination clear indication. The interrupt is set when the Authentication engine finishes the calculation process.
1	ZInt1	RW0 0x0	This bit is the DES encryption all termination clear indication. The interrupt is set when the Encryption engine finishes the calculation process.
2	Zin2	RW0 0x0	This bit is the AES encryption termination clear indication. The interrupt is set when the AES Encryption engine finishes the calculation process.
3	Zint3	RW0 0x0	This bit is the AES decryption termination clear indication. The interrupt is set when the AES Decryption engine finishes the calculation process.
4	ZInt4	RW0 0x0	This bit is the encryption termination clear indication. The interrupt is set when the Encryption engine finishes the calculation process.
5	AccInt0	RW0 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. **NOTE:** Cleared this bit before writing 1 to <EnSecurityAccl0> field in the Security Accelerator Command Register (Table 514 p. 465).
6	AccInt1	RW0 0x0	This bit is the Security accelerator session 1 termination clear indication. The interrupt is set when the Security accelerator session 1 completes its operation. **NOTE:** Cleared this bit before writing 1 to <EnSecurityAccl1> field in the Security Accelerator Command Register (Table 514 p. 466).

Table 519: Cryptographic Engines and Security Accelerator Interrupt Cause Register (Continued)
Offset: 0x9DE20

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.

Bits	Field	Type/InitVal	Description
7	AccAndIDMAInt0	RW0 0x0	Acceleration and IDMA Interrupt 0 This bit is set to 1 when the entire security accelerator process has been completed, including both encryption and authentication processes, as well as the associated IDMA operation. When the IDMA is configured to copy the outcome of the security accelerator process back to DDR, (bit[9] <Ch0Activate IDMA> field in the Security Accelerator Configuration Register (Table 517 p. 467) is set to 1. This bit is set to 1 after the IDMA has finished copying the data back to DDR. When the IDMA is NOT configured to copy the outcome of the security accelerator process back to DDR, (bit[9] <Ch0Activate IDMA> is set to 0. This bit is set after the security accelerator has completed the process and data is valid in the local SRAM. **NOTE:** These bits are relevant only if bit 7 and bit 9 of the Security Accelerator Configuration Register are set to 1.
8	AccAndIDMAInt1	RW0 0x0	Acceleration and IDMA Interrupt 0 This bit is set to 1 when the entire security accelerator process has been completed, including both encryption and authentication processes, as well as the associated IDMA operation. When the IDMA is configured to copy the outcome of the security accelerator process back to DDR, (bit[10] <Ch1Activate IDMA> field in the Security Accelerator Configuration Register (Table 517 p. 467) is set to 1. This bit is set to 1 after the IDMA has finished copying the data back to DDR. When the IDMA is NOT configured to copy the outcome of the security accelerator process back to DDR, (bit[10] <Ch1Activate IDMA> is set to 0. This bit is set after the security accelerator has completed the process and data is valid in the local SRAM. **NOTE:** These bits are relevant only if bit 8 and bit 10 of the Security Accelerator Configuration Register are set to 1.
31:9	Reserved	RES 0x0	Reserved

Table 520: Cryptographic Engines and Security Accelerator Interrupt Mask Register
Offset: 0x9DE24

Bits	Field	Type/InitVal	Description
31:0	Mask	RW 0x0	Mask bit per each cause bit 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register.

A.12 Two-Wire Serial Interface (TWSI) Registers

Table 521: TWSI Interface Register Map

Register	Offset	Page
TWSI Slave Address	0x11000	Table 522, p.470
TWSI Extended Slave Address	0x11010	Table 523, p.470
TWSI Data	0x11004	Table 524, p.471
TWSI Control	0x11008	Table 525, p.471
TWSI Status	0x1100C	Table 526, p.473
TWSI Baud Rate	0x1100C	Table 527, p.474
TWSI Soft Reset	0x1101C	Table 528, p.474

A.12.1 TWSI Registers

Table 522: TWSI Slave Address
Offset:0x11000

Bits	Field	Type/InitVal	Description
0	GCE	RW 0x0	General Call Enable If set to 1, the TWSI slave interface responds to general call accesses.
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.
31:8	Reserved	RO 0x0	Reserved

Table 523: TWSI Extended Slave Address
Offset:0x11010

Bits	Field	Type/InitVal	Description
7:0	SAddr	RW 0x0	Bits [7:0] of the 10-bit slave address
31:8	Reserved	RO 0x0	Reserved

Table 524: TWSI Data
Offset:0x11004

Bits	Field	Type/InitVal	Description
7:0	Data	RW 0x0	Data/Address byte to be transmitted by the TWSI master or slave, or data byte received In the case of the Address byte, bit [0] is the Read/Write Command bit.
31:8	Reserved	RO 0x0	Reserved

Table 525: TWSI Control
Offset:0x11008

Bits	Field	Type/InitVal	Description
1:0	Reserved	RO 0x0	Reserved
2	ACK	RW 0x0	Acknowledge When set to 1, the TWSI 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 TWSI target a read of last data, the Feroceon® 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.
3	IFlg	RW 0x0	Interrupt Flag If any of the status codes other than 0xF8 are set, the TWSI hardware sets the bit to 1. If set to 1 and TWSI interrupts are enabled through bit [7], an interrupt is asserted. Cleared by a Feroceon CPU core write of 0.
4	Stop	RW 0x0	Stop When set to 1, the TWSI master initiates a stop condition on the bus. The bit is set only. It is cleared by TWSI hardware after a stop condition is driven on the bus.
5	Start	RW 0x0	Start When set to 1, the TWSI 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 TWSI hardware after a start condition is driven on the bus.
6	TWSIEn	RW 0x0	TWSI Enable If set to 1, the TWSI slave responds to calls to its slave address, and to general calls if enabled. If set to 0, TW_SDA and TW_SCK inputs are ignored. The TWSI slave does not respond to any address on the bus.
7	IntEn	RW 0x0	Interrupt Enable When set to 1, an interrupt is generated each time the interrupt flag is set.

Table 525: TWSI Control (Continued)
Offset:0x11008

Bits	Field	Type/InitVal	Description
31:8	Reserved	RO 0x0	Reserved

Note

Status and Baud Rate registers share the same offset. When being read, this register functions as Status register. When written, it acts as Baud Rate register.

Table 526: TWSI Status
Offset:0x1100C

Bits	Field	Type/InitVal	Description
7:0	Stat	RO 0xF8	TWSI Status 0x00 = Bus error. 0x08 = Start condition transmitted. 0x10 = Repeated start condition transmitted. 0x18 = Address + write bit transmitted, acknowledge received. 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.
31:8	Reserved	RO 0x0	Reserved

Table 527: TWSI Baud Rate
Offset:0x1100C

Bits	Field	Type/InitVal	Description
2:0	N	WO 0x4	See exact frequency calculation in the TWSI section. Write only.
6:3	M	WO 0x4	See exact frequency calculation in the TWSI section. Write only.
31:7	Reserved	RO 0x0	Reserved

Table 528: TWSI Soft Reset
Offset:0x1101C

Bits	Field	Type/InitVal	Description
31:0	Rst	WO 0x0	Write Only Write to this register resets the TWSI logic and sets all TWSI registers to their reset values.

A.13 UART Interface Registers

Note

A number of UART registers share the same offsets (see Table 529). In addition to writing to these register addresses, <DivLatchRdWrt> bit[7] of the Line Control Register (LCR) (Table 537 p. 479) must be set/cleared as follows:

- Set <DivLatchRdWrt> to address the Divisor Latch Low (DLL) Register and Divisor Latch High (DLH) Register.
- Clear <DivLatchRdWrt> to address the Receive Buffer Register (RBR), Transmit Holding Register (THR), and Interrupt Enable Register (IER).

Table 529: UART Interface Registers Map

Register	Offset	Table, Page	Type	<DivLatchRdWrt> Setting
Receive Buffer Register (RBR)	UART 0: 0x12000, UART 1: 0x12100	Table 530, p. 476	RO	0
Transmit Holding Register (THR)	UART 0: 0x12000, UART 1: 0x12100	Table 531, p. 476	WO	0
Divisor Latch Low (DLL) Register	UART 0: 0x12000, UART 1: 0x12100	Table 532, p. 477	RW	1
Interrupt Enable Register (IER)	UART 0: 0x12004, UART 1: 0x12104	Table 533, p. 477	RW	0
Divisor Latch High (DLH) Register	UART 0: 0x12004, UART 1: 0x12104	Table 534, p. 478	RW	1
Interrupt Identity Register (IIR)	UART 0: 0x12008, UART 1: 0x12108	Table 535, p. 478	RO	NA
FIFO Control Register (FCR)	UART 0: 0x12008, UART 1: 0x12108	Table 536, p. 478	WO	NA
Line Control Register (LCR)	UART 0: 0x1200C, UART 1: 0x1210C	Table 537, p. 479	RW	NA
Modem Control Register (MCR)	UART 0: 0x12010, UART 1: 0x12110	Table 538, p. 480	RW	NA
Line Status Register (LSR)	UART 0: 0x12014, UART 1: 0x12114	Table 539, p. 480	RO	NA
Modem Status Register (MSR)	UART 0: 0x12018, UART 1: 0x12118	Table 540, p. 481	RO	NA
Scratch Pad Register (SCR)	UART 0: 0x1201C, UART 1: 0x1211C	Table 541, p. 482	RW	NA

A.13.1 UART Interface Registers

Table 530: Receive Buffer Register (RBR)
Offset:UART 0: 0x12000, UART 1: 0x12100

NOTE: <DivLatchRdWrt> bit[7] of the Line Control Register (LCR) (Table 537 p. 479) must be set to 0.

Bits	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 (see Table 539 on page 480). 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.
31:8	Reserved	RSVD	Reserved

Table 531: Transmit Holding Register (THR)
Offset:UART 0: 0x12000, UART 1: 0x12100

NOTE: <DivLatchRdWrt> bit[7] of the Line Control Register (LCR) (Table 537 p. 479) must be set to 0.

Bits	Field	Type/InitVal	Description
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 (see Table 539 on page 480), 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.
31:8	Reserved	RSVD	Reserved

Table 532: Divisor Latch Low (DLL) Register
Offset:UART 0: 0x12000, UART 1: 0x12100

NOTE: <DivLatchRdWrt> bit[7] of the Line Control Register (LCR) (Table 537 p. 479) must be set to 1.

Bits	Field	Type/InitVal	Description
7:0	DivLatchLow	WO 0x0	The DLH (Divisor Latch High) register in conjunction with DLL (Divisor Latch Low) register forms a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. It is accessed by first setting the DivLatchRdWrt bit in the Line Control Register (LCR). The output baud rate is equal to the input clock frequency divided by sixteen times the value of the baud rate divisor. baud = (clock frequency) / (16 * divisor)
31:8	Reserved	RSVD	Reserved

Table 533: Interrupt Enable Register (IER)
Offset:UART 0: 0x12004, UART 1: 0x12104

NOTE: <DivLatchRdWrt> bit[7] of the Line Control Register (LCR) (Table 537 p. 479) must be set to 0.

Bits	Field	Type/InitVal	Description
0	RxDataIntEn	RW 0x0	Enable Received Data Available Interrupt (ERBFI) 0 = Disable interrupt 1 = Enable interrupt When the FIFO mode is set in FIFO Control Register, this interrupt provides a character timeout indication.
1	TxHoldIntEn	RW 0x0	Enable Transmitter Holding Register Empty Interrupt (ETBEI) 0 = Disable interrupt 1 = Enable interrupt
2	RxLineStatIntEn	RW 0x0	Enable Receiver Line Status Interrupt (ELSI) 0 = Disable interrupt 1 = Enable interrupt
3	ModStatIntEn	RW 0x0	Enable Modem Status Interrupt (EDSSI) 0 = Disable interrupt 1 = Enable interrupt
31:4	Reserved	RSVD	Reserved

Table 534: Divisor Latch High (DLH) Register
Offset:UART 0: 0x12004, UART 1: 0x12104

NOTE: <DivLatchRdWrt> bit[7] of the Line Control Register (LCR) (Table 537 p. 479) must be set to 1.

Bits	Field	Type/InitVal	Description
7:0	DivLatchHigh	WO 0x0	The DLH (Divisor Latch High) register in conjunction with DLL (Divisor Latch Low) register forms a 16-bit, read/write, Divisor Latch register that contains the baud rate divisor for the UART. It is accessed by first setting the DivLatchRdWrt bit in the Line Control Register (LCR). The output baud rate is equal to the input clock frequency divided by sixteen times the value of the baud rate divisor. baud = (clock frequency) / (16 * divisor)
31:8	Reserved	RSVD	Reserved

Table 535: Interrupt Identity Register (IIR)
Offset:UART 0: 0x12008, UART 1: 0x12108

Bits	Field	Type/InitVal	Description
3:0	InterruptID	RO 0x0	Interrupt ID 0000 = Modem Status Changed 0001 = No interrupt pending 0010 = THR empty 0100 = Received Data available 0110 = Receiver Status 1100 = Character Time Out
5:4	Reserved	RO 0x0	Reserved
7:6	FIFOEn	RO 0x0	FIFO Enable 00 = FIFOs are disabled (default in FIFO mode) 11 = FIFOs are enabled
31:8	Reserved	RSVD	Reserved

Table 536: FIFO Control Register (FCR)
Offset:UART 0: 0x12008, UART 1: 0x12108

Bits	Field	Type/InitVal	Description
0	FIFOEn	WO 0x0	Enable transmit and receive FIFOs. This register controls the read and write data FIFO operation and the mode of operation for the DMA signals UA0_CTSn UA1_CTSn, UA0_RTSn, and UA1_RTSn. 0 = Disable FIFOs 1 = Enable FIFOs
1	RxFIFOReset	WO 0x0	Receive FIFO reset 0 = Do not flush data from the receive FIFO 1 = Flush data from the receive FIFO

Table 536: FIFO Control Register (FCR) (Continued)
Offset:UART 0: 0x12008, UART 1: 0x12108

Bits	Field	Type/InitVal	Description
2	TxFIFOReset	WO 0x0	Transmit FIFO reset 0 = Do not flush data from the transmit FIFO 1 = Flush data from the transmit FIFO
3	DMAMode	WO 0x0	DMA mode. 0 = Single transfer DMA mode 0 1 = Multi transfer DMA mode 1
5:4	Reserved	RSVD 0x0	Reserved
7:6	RxTrigger	WO 0x0	Receive Trigger 00 = 1 byte in FIFO 01 = 4 bytes in FIFO 10 = 8 bytes in FIFO 11 = 14 bytes FIFO
31:8	Reserved	RSVD	Reserved

Table 537: Line Control Register (LCR)
Offset:UART 0: 0x1200C, UART 1: 0x1210C

Bits	Field	Type/InitVal	Description
1:0	WLS	RW 0x0	Number of bits per character 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits
2	Stop	RW 0x0	Stop bits transmitted 0 = 1 bit 1 = 2 bits
3	PEN	RW 0x0	Parity enable 0 = Parity disabled 1 = Parity enabled
4	EPS	RW 0x0	Even or odd parity select 0 = Odd parity 1 = Even parity
5	Reserved	RSVD 0x0	Reserved
6	Break	RW 0x0	The <Break> bit sends a break signal by holding the SOUT line low until the <Break> bit is reset. 0 = Do not send a break signal. 1 = Send a break signal.

Table 537: Line Control Register (LCR) (Continued)
Offset:UART 0: 0x1200C, UART 1: 0x1210C

Bits	Field	Type/InitVal	Description
7	DivLatchRdWrt	RW 0x0	This bit must be set to address (reading and writing) of the Divisor Latch Low (DLL) Register (Table 532 p. 477) and Divisor Latch High (DLH) Register (Table 534 p. 478)) to set the baud rate of the UART. This bit must be cleared to address the Receive Buffer Register (RBR) (Table 530 p. 476), Transmit Holding Register (THR) (Table 531 p. 476) and the Interrupt Enable Register (IER) (Table 533 p. 477).
31:8	Reserved	RSVD	Reserved

Table 538: Modem Control Register (MCR)
Offset:UART 0: 0x12010, UART 1: 0x12110

Bits	Field	Type/InitVal	Description
0	Reserved	RSVD	Reserved
1	RTS	RW 0x0	Request To Send The <RTS> bit is inverted and then drives the corresponding UA0_RTSn and UA1_RTSn output.
3:2	Reserved	RSVD	Reserved
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 = No loopback 1 = Loopback
31:5	Reserved	RSVD	Reserved

Table 539: Line Status Register (LSR)
Offset:UART 0: 0x12014, UART 1: 0x12114

Bits	Field	Type/InitVal	Description
0	DataRxStat	RO 0x0	Receive buffer status 0 = No characters in the receive buffer or FIFO. 1 = Receive buffer or FIFO contains at least one character. A read operation of the Receiver Buffer Register clears this bit. This bit is cleared when the Receive Buffer Register (RBR) (Table 530 p. 476) is read.
1	OverRunErr	ROC 0x0	Overrun Error 0 = No overrun 1 = Overrun error has occurred

Table 539: Line Status Register (LSR) (Continued)
Offset:UART 0: 0x12014, UART 1: 0x12114

Bits	Field	Type/InitVal	Description
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) (Table 537 p. 479) 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.
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) (Table 537 p. 479) is performed.
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) (Table 537 p. 479) clears the <BI>.
5	THRE	RO 0x1	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) (Table 531 p. 476) is transmitted to the transmit shift register. 0 = Do not accept a new character for transmission 1 = Accept a new character for transmission
6	TxEmpty	RO 0x1	Transmitter Empty bit In FIFO mode, this bit is set whenever the Transmit Holding Register (THR) (Table 531 p. 476), the Transmitter Shift Register, and the FIFO are all empty.
7	RxFIFOErr	ROC 0x1	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) (Table 537 p. 479) is read.
31:8	Reserved	RSVD 0x0	Reserved

Table 540: Modem Status Register (MSR)
Offset: UART 0: 0x12018, UART 1: 0x12118

Bits	Field	Type/InitVal	Description
0	DCTS	RW 0x0	The <DCTS> bit records whether the modem control line UA0_CTSn/UA1_CTSn has changed since the last time the Feroceon® CPU core read the MSR. In Loopback mode, <DCTS> reflects changes on Modem Control Register (MCR) (Table 538 p. 480) bit[1] <RTS>.
3:1	Reserved	RSVD	Reserved

Table 540: Modem Status Register (MSR) (Continued)
Offset: UART 0: 0x12018, UART 1: 0x12118

Bits	Field	Type/InitVal	Description
4	CTS	RSVD	The CTS Modem Status bit—<CTS>—contains information on the current state of the modem control line. <CTS> is the compliment of UA0_CTSn/UA1_CTSn. In Loopback Mode, <CTS> is the same as Modem Control Register (MCR) (Table 538 p. 480) bit[1] <RTS>.
31:5	Reserved	RSVD	Reserved

Table 541: Scratch Pad Register (SCR)
Offset:UART 0: 0x1201C, UART 1: 0x1211C

Bits	Field	Type/InitVal	Description
7:0	Scratch	RW 0x0	The SCR register is an 8-bit read/write register for programmers to use as a temporary storage space.
31:8	Reserved	RSVD	Reserved

A.14 Device Controller Registers

Table 542: Device Registers Map

Register	Offset	Table, Page
Device Bank0 Parameters Register	0x1045C	Table 543, p. 483
Device Bank1 Parameters Register	0x10460	Table 544, p. 484
Device Bank2 Parameters Register	0x10464	Table 545, p. 484
Boot Device Parameters Register	0x1046C	Table 546, p. 485
NAND Flash Control Register	0x104E80x104E8	Table 547, p. 485
Device Interface Control	0x104C0	Table 548, p. 486
Device Interrupt Cause	0x104D0	Table 549, p. 486
Device Interrupt Mask Register	0x104D4	Table 550, p. 487

Table 543: Device Bank0 Parameters Register
Offset:0x1045C

Bits	Field	Type/InitVal	Description
2:0	TurnOff	RW 0x7	The number of cycles in a read access between the negation of DEV_CEn to a new Device bus cycle. Minimal value = 0x2 **NOTE:** This field uses an additional bit in field <TurnOffExt> (bit [22]).
6:3	Acc2First	RW 0xF	Defines the number of cycles in a read access between the negation of DEV_ALE[0] and the cycle containing the first data sampled by the 88F5182. Number of cycles = <Acc2First> - 3. Minimal value = 0x **NOTE:** This field uses an additional bit in field <Acc2FirstExt> (bit [23]).
10:7	Acc2Next	RW 0xF	The number of cycles in a burst read access between the cycle containing the first data sampled by the 88F5182 and the cycle containing the next data sampled. Minimal value = 0x2 **NOTE:** This field uses an additional bit in field <Acc2NextExt> (bit [24]).
13:11	ALE2Wr	RW 0x7	Defines the number of cycles in a write access from the DEV_ALE[0] negation to the assertion of DEV_WEn. Number of cycles = <ALE2Wr> - 3. Minimal value = 0x4 **NOTE:** This field uses an additional bit in field <ALE2WrExt> (bit [25]).
16:14	WrLow	RW 0x7	The number of cycles in a write access that the DEV_WEn signal is kept active. **NOTE:** This field uses an additional bit in field <WrLowExt> (bit [26]).
19:17	WrHigh	RW 0x7	The number of cycles in a burst write access that the DEV_WEn signal is kept de-asserted. **NOTE:** This field uses an additional bit in field <WrHighExt> (bit [27]).

Table 543: Device Bank0 Parameters Register (Continued)
Offset:0x1045C

Bits	Field	Type/InitVal	Description
21:20	DevWidth	RW Sampled at reset	Device Width 00 = 8 bits 01 = 16 bits 10 = Reserved 11 = Reserved
22	TurnOffExt	RW 0x1	TurnOff Extension The MSB of the TurnOff parameter.
23	Acc2FirstExt	RW 0x1	Acc2First Extension The MSB of the Acc2First parameter.
24	Acc2NextExt	RW 0x1	Acc2Next Extension The MSB of the Acc2Next parameter.
25	ALE2WrExt	RW 0x1	ALE2Wr Extension The MSB of the ALE2Wr parameter.
26	WrLowExt	RW 0x1	WrLow Extension The MSB of the WrLow parameter.
27	WrHighExt	RW 0x1	WrHigh Extension The MSB of the WrHigh parameter.
29:28	BadrSkew	RW 0x0	Cycles gap between BAdr toggle to read data sample This is useful when interfacing sync burst SRAM. 0x0 = No gap (default setting) 0x1 = One cycle gap 0x2 = Two cycle gaps 0x3 = Reserved
31:30	Reserved	RW	Must be 0x2.

Table 544: Device Bank1 Parameters Register
Offset:0x10460

Bits	Field	Type/InitVal	Description
31:0	Various	RW 0x8FCFFFFF	These fields function as in Device Bank0.

Table 545: Device Bank2 Parameters Register
Offset:0x10464

Bits	Field	Type/InitVal	Description
31:0	Various	RW 0x8FCFFFFF	These fields function as in Device Bank0.

Table 546: Boot Device Parameters Register
Offset:0x1046C

Bits	Field	Type/InitVal	Description
31:0	Various	RW 0x8F?FFFFF[1]	These fields function as in Device Bank0.

1. The boot device width (bits [21:20]) are sampled at reset (see the Pins Sample Configuration in the *88F5182 Feroceon*® Storage Networking SoC *Datasheet* and the initial value for bits [23:22] is '11.

Table 547: NAND Flash Control Register
Offset:0x104E8

Bits	Field	Type/InitVal	Description
0	NFBoot	RW SAR	Defines if DEV_BootCEn is connected to NAND Flash. 0 = Not connected to NAND Flash. 1 = Connected to NAND Flash. Sample at reset.
1	NFActCEnBoot	RW SAR	If both <NFBoot> and <NFActCEnBoot> bits are set to 1, DEV_BootCEn is forced to 0. This bit is used for CE care NAND Flash Sample at reset.
2	NF0	RW 0x0	Defines if CEn[0] is connected to NAND Flash. 0 = Not connected to NAND Flash. 1 = Connected to NAND Flash.
3	NFActCEn0	RW 0x0	If both <NF0> and <NFActCEn0> bits are set to 1, DEV_CEn[0] is forced to 0. This bit is used for CE care NAND Flash. 0 = Regardless of <NF0> value, DEV_CEn[0] is not forced to 0. 1 = If <NF0> is set to 1, DEV_CEn[0] is forced to 0.
4	NF1	RW 0x0	See <NF0> description.
5	NFActCEn1	RW 0x0	See <NFActCEn0> description
6	NF2	RW 0x0	See <NF0> description
7	NFActCEn2	RW 0x0	See <NFActCEn0> description
8	NFISD	RW SAR	NAND Flash Initialization Sequence Disabled 0 = Enabled Initialization Sequence 1 = Disabled Initialization Sequence Sampled at reset.
13:9	NFOEnW	RW 0xC	Defines DEV_OEn high width <(NFOEnHW+1) Core clocks>. Applies to all NAND Flash devices (connected to DEV_BootCEn, DEV_CEn0, DEV_CEn1, and DEV_CEn2). For the default: 0x0E, the calculation is: (15/166 MHz ~90 ns).

Table 547: NAND Flash Control Register (Continued)
Offset:0x104E8

Bits	Field	Type/InitVal	Description
18:14	NFTr	RW 0x1F	NAND Flash Time Ready. Defines the maximum time it takes the boot NAND Flash to transfer the data from the array to the register. <(NFTr+1) x 1024 Core clocks>. The CPU is forced to reset during this time, before it starts the boot procedures. For the default value, 0x1F, the calculation is: (32x1024/166 MHz ~ 197 us).
19	NFOEnDel	RW 0x0	See <NFActCEn0> description 0 = Delay the falling edge of DEV_OEn by one cycle after the DEV_CEn falling edge in access to don't care NAND Flash. 1 = DEV_OEn and DEV_CEn fall on the same edge.
31:20	Reserved	RW 0x0	Reserved

Table 548: Device Interface Control
Offset:0x104C0

Bits	Field	Type/InitVal	Description
15:0	Timeout	RW 0xFFFF	Timeout Timer Preset Value If the device access is not completed within 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.
16	Reserved	RO 0x0	Must be cleared to 0x0.
17	Reserved	RO 0x0	Reserved
19:18	Reserved	RO 0x3	Must be 3.
31:20	Reserved	RO 0x0	Reserved

Table 549: Device Interrupt Cause
Offset:0x104D0

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 effect.

Bits	Field	Type/InitVal	Description
0	Reserved	RW0C 0x0	Reserved
1	DRdyErr	RW0C 0x0	Ready Timer Expired

Table 549: Device Interrupt Cause (Continued)
Offset:0x104D0

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 effect.

Bits	Field	Type/InitVal	Description
31:2	Reserved	RES 0x0	Reserved

Table 550: Device Interrupt Mask Register
Offset:0x104D4

Bits	Field	Type/InitVal	Description
0	Reserved	RO 0x0	Reserved
1	Mask	RW 0x0	Mask bit per each cause bit 0 = Interrupt is masked. 1 = Interrupt is enabled. Mask only affects the assertion of interrupt pins. It does not affect the setting of bits in the Cause register.
31:2	Reserved	RES 0x0	Reserved

A.15 IDMA Controller Interface Registers

Table 551: IDMA Controller Interface Register Map

Register	Offsets	Page
IDAM Descriptor Registers		
Channel IDMA Byte Count Register	Channel 0 0x60800, Channel 1 0x60804, Channel 2 0x60808, Channel 3 0x6080C	Table 552, p.489
Channel IDMA Source Address Register	Channel 0 0x60810, Channel 1 0x60814, Channel 2 0x60818, Channel 3 0x6081C	Table 553, p.489
Channel IDMA Destination Address Register	Channel 0 0x60820, Channel 1 0x60824, Channel 2 0x60828, Channel 3 0x6082C	Table 554, p.489
Channel Next Descriptor Pointer Register	Channel 0 0x60830, Channel 1 0x60834, Channel 2 0x60838, Channel 3 0x6083C	Table 555, p.489
Channel Current Descriptor Pointer Register	Channel 0 0x60870, Channel 1 0x60874, Channel 2 0x60878, Channel 3 0x6087C	Table 556, p.490
IDMA Address Decoding Registers		
Base Address Register x	BAR0 0x60A00, BAR1 0x60A08, BAR2 0x60A10, BAR3 0x60A18, BAR4 0x60A20, BAR5 0x60A28, BAR6 0x60A30, BAR7 0x60A38	Table 557, p.490
Size Register x	SR0 0x60A04, SR1 0x60A0C, SR2 0x60A14, SR3 0x60A1C, SR4 0x60A24, SR5 0x60A2C, SR6 0x60A34, SR7 0x60A3C	Table 558, p.490
High Address Remap x Register	Register 0 0x60A60, Register 1 0x60A64, Register 2 0x60A68, Register 3 0x60A6C	Table 559, p.491
Base Address Enable Register	0x60A80	Table 560, p.491
Channelx Access Protect Register	Channel 0 0x60A70, Channel 1 0x60A74, Channel 2 0x60A78, Channel 3 0x60A7C	Table 561, p.491
IDMA Control Registers		
Channel Control (Low) Register	Channel 0 0x60840, Channel 1 0x60844, Channel 2 0x60848, Channel 3 0x6084C	Table 562, p.492
Channel Control (High) Register	Channel 0 0x60880, Channel 1 0x60884, Channel 2 0x60888, Channel 3 0x6088C	Table 563, p.494
IDMA Interrupt Registers		
Interrupt Cause Register	0x608C0	Table 564, p.495
Interrupt Mask Register	0x608C4	Table 565, p.495
Error Address Register	0x608C8	Table 566, p.496
Error Select Register	0x608CC	Table 567, p.497

A.15.1 IDMA Descriptor Registers

Table 552: Channel IDMA Byte Count Register[1]
Offset:Channel 0 0x60800, Channel 1 0x60804, Channel 2 0x60808, Channel 3 0x6080C

Bits	Field	Type	Description
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]).
29:24	Reserved	RES 0x0	Reserved
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 = The whole byte count transferred. 1 = Transfer terminated before the whole byte count was 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.

1. When running in 64K descriptor mode and when closing the descriptor, the IDMA writes to bits [31:16] the left byte count to be transferred.

Table 553: Channel IDMA Source Address Register
Offset:Channel 0 0x60810, Channel 1 0x60814, Channel 2 0x60818, Channel 3 0x6081C

Bits	Field	Type	Description
31:0	SrcAdd	RW 0x0	Bits [31:0] of the IDMA source address

Table 554: Channel IDMA Destination Address Register
Offset:Channel 0 0x60820, Channel 1 0x60824, Channel 2 0x60828, Channel 3 0x6082C

Bits	Field	Type	Description
31:0	DestAdd	RW 0x0	Bits [31:0] of the IDMA destination address

Table 555: Channel Next Descriptor Pointer Register
Offset:Channel 0 0x60830, Channel 1 0x60834, Channel 2 0x60838, Channel 3 0x6083C

Bits	Field	Type	Description
31:0	NextDescPtr	RW 0x0	Bits [31:0] of the IDMA next descriptor address The address must be 16-byte aligned (bits [3:0] must be 0x0)

Table 556: Channel Current Descriptor Pointer Register
Offset:Channel 0 0x60870, Channel 1 0x60874, Channel 2 0x60878, Channel 3 0x6087C

Bits	Field	Type	Description
31:0	CDPTR0/1/2/3	RW 0x0	Bits [31:0] of the address from which the current descriptor was fetched

A.15.2 IDMA Address Decoding Registers

Table 557: Base Address Register x
Offset:BAR0 0x60A00, BAR1 0x60A08, BAR2 0x60A10, BAR3 0x60A18, BAR4 0x60A20, BAR5 0x60A28, BAR6 0x60A30, BAR7 0x60A38

Bits	Field	Type	Description
3:0	Target	RW 0x0	Target unit ID. Specifies the target interface associated with this window. 0x0 = DRAM 0x1 = Devices 0x2 = Reserved 0x3 = PCI 0x3 = Reserved 0x4 = PCI Express 0x5 = Tunit DMA SRAM Other values = Reserved
7:4	Reserved	RO 0x0	Reserved
15:8	Attr	RW 0x0	Specifies target unit specific attributes
31:16	Base	RW 0x0	Window Base Address

Table 558: Size Register x
Offset:SR0 0x60A04, SR1 0x60A0C, SR2 0x60A14, SR3 0x60A1C, SR4 0x60A24, SR5 0x60A2C, SR6 0x60A34, SR7 0x60A3C

Bits	Field	Type	Description
15:0	Reserved	RO 0x0	Reserved
31:16	Size	RW 0x0	Window Size 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 559: High Address Remap x Register[1]
Offset:Register 0 0x60A60, Register 1 0x60A64, Register 2 0x60A68, Register 3 0x60A6C

Bits	Field	Type	Description
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 32-bit addressing

1. 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.

Table 560: Base Address Enable Register
Offset:0x60A80

Bits	Field	Type	Description
7:0	En	RW 0xFF	Address Window Enable Bit per window. If set to 0, the corresponding address window is enabled.
31:8	Reserved	RO 0x0	Read only.

Table 561: Channelx Access Protect Register
Offset:Channel 0 0x60A70, Channel 1 0x60A74, Channel 2 0x60A78, Channel 3 0x60A7C

Bits	Field	Type	Description
1:0	Win0	RW 0x3	Window0 Access control 0x0 = No access allowed 0x1 = Read Only 0x2 = Reserved 0x3 = Full access (read or write) In the 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
3:2	Win1	RW 0x3	Window1 access control
5:4	Win2	RW 0x3	Window2 access control
7:6	Win3	RW 0x3	Window3 access control
9:8	Win4	RW 0x3	Window4 access control
11:10	Win5	RW 0x3	Window5 access control
13:12	Win6	RW 0x3	Window6 access control
15:14	Win7	RW 0x3	Window7 access control

Table 561: Channelx Access Protect Register (Continued)
Offset:Channel 0 0x60A70, Channel 1 0x60A74, Channel 2 0x60A78, Channel 3 0x60A7C

Bits	Field	Type	Description
31:16	Reserved	RO 0x0	Read only.

A.15.3 IDMA Channel Control Registers

Table 562: Channel Control (Low) Register
Offset:Channel 0 0x60840, Channel 1 0x60844, Channel 2 0x60848, Channel 3 0x6084C

Bits	Field	Type	Description
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
3	SrcHold	RW 0x0	Source Hold 0 = Increment source address. 1 = Hold in the same value.
4	Reserved	RW 0x0	Reserved
5	DestHold	RW 0x0	Destination Hold 0 = Increment destination address. 1 = Hold in the same value.
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
9	ChainMode	RW 0x0	Chained Mode 0 = Chained mode 1 = Non-Chained mode
10	IntMode	RW 0x0	Interrupt Mode 0 = Interrupt asserted every time the IDMA byte count reaches 0. 1 = Interrupt asserted when the Next Descriptor pointer is NULL and the IDMA byte count reaches 0. **NOTE:** IntMode is only relevant in chain mode.
11	Reserved	RW 0x0	Reserved Must be set to 1.

Table 562: Channel Control (Low) Register (Continued)
Offset:Channel 0 0x60840, Channel 1 0x60844, Channel 2 0x60848, Channel 3 0x6084C

Bits	Field	Type	Description
12	ChanEn	RW 0x0	Channel Enable 0 = The channel is suspended. 1 = The channel is activated. Re-setting the bit to 1, allows the channel to continue the IDMA transfer.
13	FetchND	RWC 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.
14	ChanAct	RO 0x0	IDMA Channel Active Read only. 0 = Channel is not active. 1 = Channel is active.
16:15	Reserved	RW 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. 0 = Disable 1 = Enable **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].
19:18	Reserved	RW 0x0	Reserved Must be 0x0
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.
22:21	SAddrOvr	RW 0x0	Override Source Address 00 = No override. 01 = Source interface and attributes are taken from BAR 1 10 = Source interface and attributes are taken from BAR 2 11 = Source interface and attributes are taken from BAR 3
24:23	DAddrOvr	RW 0x0	Override Destination Address 00 = No override. 01 = Destination interface and attributes are taken from BAR 1 10 = Destination interface and attributes are taken from BAR 2 11 = Destination interface and attributes are taken from BAR 3
26:25	NAddrOvr	RW 0x0	Override Next Descriptor Address 00 = No override. 01 = Next descriptor interface and attributes are taken from BAR 1 10 = Next descriptor interface and attributes are taken from BAR 2 11 = Next descriptor interface and attributes are taken from BAR 3

Table 562: Channel Control (Low) Register (Continued)
Offset:Channel 0 0x60840, Channel 1 0x60844, Channel 2 0x60848, Channel 3 0x6084C

Bits	Field	Type	Description
30:27	Reserved	RW 0x0	Reserved
31	DescMode	RW 0x0	Descriptor Mode 0 = 64K descriptor mode 1 = 16M descriptor mode

Table 563: Channel Control (High) Register
Offset:Channel 0 0x60880, Channel 1 0x60884, Channel 2 0x60888, Channel 3 0x6088C

Bits	Field	Type	Description
7:0	Reserved	RW 0x0	Reserved Must be set to 0x3.
31:8	Reserved	RO 0x0	Read Only.

A.15.4 IDMA Interrupt Registers

Table 564: Interrupt Cause Register[1]
Offset:0x608C0

Bits	Field	Type	Description
0	Comp	RW 0x0	Channel0 IDMA Completion
1	AddrMiss	RW 0x0	Channel0 Address Miss Failed address decoding.
2	AccProt	RW 0x0	Channel0 Access Protect Violation
3	WrProt	RW 0x0	Channel0 Write Protect
4	Own	RW 0x0	Channel0 Descriptor Ownership Violation Attempt to access the descriptor owned by the CPU.
7:5	Reserved	RW 0x0	Reserved
12:8	Various	RW 0x0	Same as channel0 cause bits
15:13	Reserved	RES 0x0	Reserved
20:16	Various	RW 0x0	Same as channel0 cause bits
23:21	Reserved	RES 0x0	Reserved
28:24	Various	RW 0x0	Same as channel0 cause bits
31:29	Reserved	RES 0x0	Reserved

1. 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.

Table 565: Interrupt Mask Register
Offset:0x608C4

Bits	Field	Type	Description
0	Comp	RW 0x0	Comp Interrupt 0 = Disable 1 = Enable
1	AddrMiss	RW 0x0	Address Miss Interrupt 0 = Disable 1 = Enable

Table 565: Interrupt Mask Register (Continued)
Offset:0x608C4

Bits	Field	Type	Description
2	AccProt	RW 0x0	Access Protection Interrupt 0 = Disable 1 = Enable
3	WrProt	RW 0x0	Write Protection Interrupt 0 = Disable 1 = Enable
4	Own	RW 0x0	Ownership Violation Interrupt 0 = Disable 1 = Enable
7:5	Reserved	RES 0x0	Reserved
12:8	Various	RW 0x0	Same as channel0 mask bits
15:13	Reserved	RES 0x0	Reserved
20:16	Various	RW 0x0	Same as channel0 mask bits
23:21	Reserved	RES 0x0	Reserved
28:24	Various	RW 0x0	Same as channel0 mask bits
31:29	Reserved	RES 0x0	Reserved

Table 566: Error Address Register
Offset:0x608C8

Bits	Field	Type	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:** No address will be latched where the respective interrupt is masked (disabled). See Table 565, Interrupt Mask Register, on page 495.

Table 567: Error Select Register
Offset:0x608CC

Bits	Field	Type	Description
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.
31:5	Reserved	RO 0x0	Read only.

A.16 XOR Engine Registers

Table 568: XOR Engine Register Map

Register	Offset	Table, Page
XOR Engine Control Registers		
XOR Engine Channel Arbiter (XECHAR)	0x60900	Table 569, p. 499
XOR Engine [0..1] Configuration (XExCR)	XOR0 0x60910, XOR1 0x60914	Table 570, p. 500
XOR Engine [0..1] Activation (XExACTR)	XOR0 0x60920, XOR1 0x60924	Table 571, p. 501
XOR Engine Interrupt Registers		
XOR Engine Interrupt Cause (XEICR)	0x60930	Table 572, p. 502
XOR Engine Interrupt Mask (XEIMR)	0x60940	Table 573, p. 503
XOR Engine Error Cause (XEECR)	0x60950	Table 574, p. 504
XOR Engine Error Address (XEEAR)	0x60960	Table 575, p. 504
XOR Engine Descriptor Registers		
XOR Engine [0..1] Next Descriptor Pointer (XExNDPR)	XOR0 0x60B00, XOR1 0x60B04	Table 576, p. 504
XOR Engine [0..1] Current Descriptor Pointer (XExCDPR)	XOR0 0x60B10, XOR1 0x60B14	Table 577, p. 505
XOR Engine [0..1] Byte Count (XExBCR)	XOR0 0x60B20, XOR1 0x60B24	Table 578, p. 505
XOR Engine Address Decoding Registers		
XOR Engine [0..1] Window Control (XExWCR)	XOR0 0x60B40, XOR1 0x60B44	Table 579, p. 505
XOR Engine Base Address (XEBARx)	XEBAR0 0x60B50, XEBAR1 0x60B54, XEBAR2 0x60B58, XEBAR3 0x60B5C, XEBAR4 0x60B60, XEBAR5 0x60B64, XEBAR6 0x60B68, XEBAR7 0x60B6C	Table 580, p. 506
XOR Engine Size Mask (XESMRx)	XESMR0 0x60B70, XESMR1 0x60B74, XESMR2 0x60B78, XESMR3 0x60B7C, XESMR4 0x60B80, XESMR5 0x60B84, XESMR6 0x60B88, XESMR7 0x60B8C	Table 581, p. 507
XOR Engine High Address Remap (XEHARRx)	XEHARR0 0x60B90, XEHARR1 0x60B94, XEHARR2 0x60B98, XEHARR3 0x60B9C	Table 582, p. 507

Table 568: XOR Engine Register Map (Continued)

Register	Offset	Table, Page
XOR Engine [0..1] Address Override Control (XExAOCR)	XE0AOCR 0x60BA0, XE1AOCR 0x60BA4	Table 583, p. 507
XOR Engine ECC/MemInit Registers		
XOR Engine [0..1] Destination Pointer (XExDPR0)	XOR0 0x60BB0, XOR1 0x60BB4	Table 584, p. 509
XOR Engine[0..1] Block Size (XExBSR)	XOR0 0x60BC0, XOR1 0x60BC4	Table 585, p. 509
XOR Engine Timer Mode Control (XETMCR)	0x60BD0	Table 586, p. 510
XOR Engine Timer Mode Initial Value (XETMIVR)	0x60BD4	Table 587, p. 510
XOR Engine Timer Mode Current Value (XETMCVR)	0x60BD8	Table 588, p. 510
XOR Engine Initial Value Low (XEIVRL)	0x60BE0	Table 589, p. 511
XOR Engine Initial Value High (XEIVRH)	0x60BE4	Table 590, p. 511

A.16.1 XOR Engine Control Registers

Table 569: XOR Engine Channel Arbiter (XECHAR)
Offset: 0x60900

Bits	Field	Type/InitVal	Description
0	Slice0	RW 0x0	Slice #0 of the channel pizza arbiter. 0 - slice is owned by channel 0. 1 - slice is owned by channel 1.
1	Slice1	RW 0x1	Slice #1 of the channel pizza arbiter.
2	Slice2	RW 0x0	Slice #2 of the channel pizza arbiter.
3	Slice3	RW 0x1	Slice #3 of the channel pizza arbiter.
4	Slice4	RW 0x0	Slice #4 of the channel pizza arbiter.
5	Slice5	RW 0x1	Slice #5 of the channel pizza arbiter.
6	Slice6	RW 0x0	Slice #6 of the channel pizza arbiter.
7	Slice7	RW 0x1	Slice #7 of the channel pizza arbiter.
31:8	Reserved	RO 0x0	Reserved.

Table 570: XOR Engine [0..1] Configuration (XExCR)
Offset: XOR0 0x60910, XOR1 0x60914

Bits	Field	Type/InitVal	Description
2:0	OperationMode	RW 0x0	Specifies the type of operation to be carried out by XOR Engine. 0x0 = XOR calculate operation. 0x1 = CRC-32 calculate operation. 0x2 = DMA operation. 0x3 = ECC cleanup operation. 0x4 = Memory Initialization operation. 0x5 = Reserved. 0x6 = Reserved. 0x7 = Reserved.
3	Reserved	RW 0x0	Reserved
6:4	SrcBurstLimit	RW 0x4	Burst Limit in each source read request access over the Internal Crossbar 0x0 = Reserved. 0x1 = Reserved. 0x2 = 32 Bytes 0x3 = 64 Bytes 0x4 = 128 Bytes. 0x5 = Reserved. 0x5 = Reserved. 0x7 = Reserved.
7	Reserved	RW 0x0	Reserved
10:8	DstBurstLimit	RW 0x4	Burst Limit in each destination write request access over the Internal Crossbar 0x0 = Reserved. 0x1 = Reserved. 0x2 = 32 Bytes 0x3 = 64 Bytes 0x4 = 128 Bytes 0x5 = Reserved. 0x5 = Reserved. 0x7 = Reserved. **NOTE:** When using cache coherency, the burst limit must not exceed 32 bytes. If an XOR engine writes to a cache coherent DRAM region or accesses (read/write) cache coherent SRAM, the burst limit must not exceed 32 bytes.
11	Reserved	RW 0x0	Reserved
12	DrdResSwp	RW 0x0	Data Read Response Endianness Swap control. 0 = Do not swap endianness. 1 = Swap endianness. If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc.
13	DwrReqSwp	RW 0x0	Data Write request Endianness Swap control. 0 = Do not swap endianness. 1 = Swap endianness. If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc.

Table 570: XOR Engine [0..1] Configuration (XExCR) (Continued)
Offset: XOR0 0x60910, XOR1 0x60914

Bits	Field	Type/InitVal	Description
14	DesSwp	RW 0x0	Descriptor read/write Endianness Swap control. 0 = Do not swap endianness. 1 = Swap endianness. If swapping is enabled, byte0 is exchanged with byte7, byte1 with byte 6 etc.
15	RegAccProtect	RW 0x1	Internal Register Access protection Enable. 0 = Access protection mechanism is disabled. 1 = Access protection mechanism is enabled.
31:16	Reserved	RO 0x0	Reserved.

Table 571: XOR Engine [0..1] Activation (XExACTR)
Offset: XOR0 0x60920, XOR1 0x60924

Bits	Field	Type/InitVal	Description
0	XEStart	WO 0x0	0 = Clearing this bit has no meaning and will be disregarded by XOR Engine. 1 = XOR Engine Start. When the software sets this bit, it activates the relevant XOR Engine channel. Setting it again after XOR Engine entered inactive state, initiates a new operation.Setting it again after XOR Engine channel entered pause state, re-activates the channel and resumes the suspended operation execution. After entering active state, XOR Engine will signal the software by setting the XEactive bit. **NOTE:** Software must confirm that XOR Engine is inactive before setting XEstart. Setting it when XOR Engine is active will be disregarded.
1	XEstop	WO 0x0	0 = Clearing this bit has no meaning and will be disregarded by XOR Engine. 1 = XOR Engine 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). After entering de-active state XOR Engine will signal the software by clearing XEactive bit and asserting the stopped interrupt. **NOTE:** Setting XEstop when XOR Engine is inactive or paused will be disregarded.
2	XEpause	WO 0x0	0 = Clearing this bit has no meaning and will be disregarded by XOR Engine. 1 = XOR Engine Pause. 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). After entering paused state XOR Engine will signal the software by clearing XEactive bit and asserting the paused interrupt.
3	XErestart	WO 0x0	XOR Engine Restart after Pause Control 0 = Clearing this bit has no meaning and will be disregarded by the XOR Engine. 1 = 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.

Table 571: XOR Engine [0..1] Activation (XExACTR)
Offset: XOR0 0x60920, XOR1 0x60924

Bits	Field	Type/InitVal	Description
5:4	XEstatus	RO 0x0	XOR Engine Status indication. 0 = Channel not active. 1 = Channel active. 2 = Channel paused. 3 = Reserved.
31:6	Reserved	RO 0x0	Reserved.

A.16.2 XOR Engine Interrupt Registers

Table 572: XOR Engine Interrupt Cause (XEICR[1])
Offset: 0x60930

Bit	Field	Type/InitVal	Description
0	EOD0	CO 0x0	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.
1	EOC0	CO 0x0	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.
2	Stopped0	CO 0x0	XOR Engine completed stopping routine, after receiving stop command (setting XEstop). It has entered Inactive state.
3	Paused0	CO 0x0	XOR Engine completed Pausing routine, after receiving pause command (setting XEpause). It has entered paused state.
4	AddrDecode0	CO 0x0	Failed address decoding. Address is not in any window or matches more than one window.
5	AccProt0	CO 0x0	Access Protect Violation. Trying to access an address in a window in which access is not allowed.
6	WrProt0	CO 0x0	Write Protect violation. Trying to write to a write-protected window.
7	OwnErr0	CO 0x0	Descriptor Ownership Violation Attempt to access the descriptor owned by the CPU.
8	IntParityErr0	CO 0x0	Parity error. caused by erroneous internal buffer read.
9	XbarErr0	RW 0x0	Crossbar Parity error Caused by erroneous read response from the Crossbar.
15:10	Reserved	RO 0x0	Reserved.

Table 572: XOR Engine Interrupt Cause (XEICR[1]) (Continued)
Offset: 0x60930

Bit	Field	Type/InitVal	Description
25:16	Channel #1	CO 0x0	Same for XOR Engine channel #1.
31:26	Reserved	RO 0x0	Reserved.

1. 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 (don't care). XOR Engine will disregard such write attempts.

Table 573: XOR Engine Interrupt Mask (XEIMR)
Offset: 0x60940

Bit	Field	Type/InitVal	Description
0	EODMask0	RW 0x0	If set to 1, EOD interrupt is enabled.
1	EOCMask0	RW 0x0	If set to 1, EOC interrupt is enabled.
2	StoppedMask0	RW 0x0	If set to 1, Stopped interrupt is enabled.
3	PauseMask0	RW 0x0	If set to 1, Paused interrupt is enabled.
4	AddrDecodeMask0	RW 0x0	If set to 1, AddrDecode interrupt is enabled.
5	AccProtMask0	RW 0x0	If set to 1, AccProt interrupt is enabled.
6	WrProtMask0	RW 0x0	If set to 1, WrProt interrupt is enabled.
7	OwnMask0	RW 0x0	If set to 1, OwnErr interrupt is enabled.
8	IntParityMask0	RW 0x0	If set to 1, IntParityErr interrupt is enabled.
9	XbarMask0	RW 0x0	If set to 1,XbarErr interrupt is enabled.
15:10	Reserved	RW 0x0	Reserved.
25:16	Channel #1	RO 0x0	Same for XOR Engine Channel #1.
31:26	Reserved	RO 0x0	Reserved.

Table 574: XOR Engine Error Cause (XEECR)
Offset: 0x60950

Bit	Field	Type/InitVal	Description
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.
31:5	Reserved	RO 0x0	Reserved.

Table 575: XOR Engine Error Address (XEEAR)
Offset: 0x60960

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 than XEEAR.

A.16.3 XOR Engine Descriptor Registers

Table 576: XOR Engine [0..1] Next Descriptor Pointer (XExNDPR)
Offset: XOR0 0x60B00, XOR1 0x60B04

Bit	Field	Type/InitVal	Description
31:0	NextDescPtr	[4:0] RO 0x0 [31:5]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.

Table 577: XOR Engine [0..1] Current Descriptor Pointer (XExCDPR)
Offset: XOR0 0x60B10, XOR1 0x60B14

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 578: XOR Engine [0..1] Byte Count (XExBCR)
Offset: XOR0 0x60B20, XOR1 0x60B24

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, ECC and MemInit modes: updated after every write action. in CRC mode: updated after every calculation operation.

A.16.4 XOR Engine Address Decoding Registers

Table 579: XOR Engine [0..1] Window Control (XExWCR)
Offset: XOR0 0x60B40, XOR1 0x60B44

Bits	Field	Type	Description
0	Win0en	RW 0x0	Window0 Enable. 0x0 = Window 0 is disabled. 0x1 = Window 0 is enabled
1	Win1en	RW 0x0	Window1 Enable.
2	Win2en	RW 0x0	Window2 Enable.
3	Win3en	RW 0x0	Window3 Enable.
4	Win4en	RW 0x0	Window4 Enable.
5	Win5en	RW 0x0	Window5 Enable.
6	Win6en	RW 0x0	Window6 Enable.
7	Win7en	RW 0x0	Window7 Enable.
15:8	Reserved	RO 0x0	Reserved.

Table 579: XOR Engine [0..1] Window Control (XExWCR) (Continued)
Offset: XOR0 0x60B40, XOR1 0x60B44

Bits	Field	Type	Description
17:16	Win0acc	RW 0x3	Window0 Access control: 0x0 = No access allowed 0x1 = Read Only 0x2 = Reserved 0x3 = Full access (read or write) 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
19:18	Win1acc	RW 0x3	Window1 access control.
21:20	Win2acc	RW 0x3	Window2 access control.
23:22	Win3acc	RW 0x3	Window3 access control.
25:24	Win4acc	RW 0x3	Window4 access control.
27:26	Win5acc	RW 0x3	Window5 access control.
29:28	Win6acc	RW 0x3	Window6 access control.
31:30	Win7acc	RW 0x3	Window7 access control.

Table 580: XOR Engine Base Address (XEBARx)
Offset: XEBAR0 0x60B50, XEBAR1 0x60B54, XEBAR2 0x60B58, XEBAR3 0x60B5C, XEBAR4 0x60B60, XEBAR5 0x60B64, XEBAR6 0x60B68, XEBAR7 0x60B6C

Bit	Field	Type/InitVal	Description
3:0	Target	RW 0x0	Specifies the target interface associated with this window: See Address Decoding chapter for full details
7:4	Reserved	RO 0x0	Reserved.
15:8	Attr	RW 0x0	Specifies target specific attributes depending on the target interface. See Default Address Map in Section 2, Address Map, on page 20
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.

Table 581: XOR Engine Size Mask (XESMRx)
Offset: XESMR0 0x60B70, XESMR1 0x60B74, XESMR2 0x60B78, XESMR3 0x60B7C, XESMR4 0x60B80, XESMR5 0x60B84, XESMR6 0x60B88, XESMR7 0x60B8C

Bit	Field	Type/InitVal	Description
15:0	Reserved	RO 0x0	Reserved.
31:16	SizeMask	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).

Table 582: XOR Engine High Address Remap (XEHARRx[1])
Offset: XEHARR0 0x60B90, XEHARR1 0x60B94, XEHARR2 0x60B98, XEHARR3 0x60B9C

Bit	Field	Type/InitVal	Description
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. When using target interface that do not support more than 4 GB address space, this register must be cleared.

1. High Address Remap Register #N corresponds to Base Address register #N, respectively.

Table 583: XOR Engine [0..1] Address Override Control (XExAOCR)
Offset: XE0AOCR 0x60BA0, XE1AOCR 0x60BA4

Bit	Field	Type/InitVal	Description
0	SA0OvrEn	RW 0x0	Override Source Address #0 Control 0x0 = No Override. 0x1 = Override is enabled.
2:1	SA0OvrPtr	RW 0x0	Override Source Address #0 Pointer Specifies the register from which the override parameters will be taken. 0x0 = Target and attributes are taken from XEBAR0. Address[63:32] taken from XEHARR0. 0x1 = Target and attributes are taken from XEBAR1. Address[63:32] taken from XEHARR1. 0x2 = Target and attributes are taken from XEBAR2. Address[63:32] taken from XEHARR2. 0x3 = Target and attributes are taken from XEBAR3. Address[63:32] taken from XEHARR3. **NOTE:** Valid only if SA0OvrEn is set.
3	SA1OvrEn	RW 0x0	Override Source Address #1 Control
5:4	SA1OvrPtr	RW 0x0	Override Source Address #1 Pointer. **NOTE:** Valid only if SA1OvrEn is set.
6	SA2OvrEn	RW 0x0	Override Source Address #2 Control

Table 583: XOR Engine [0..1] Address Override Control (XExAOCR) (Continued)
Offset: XE0AOCR 0x60BA0, XE1AOCR 0x60BA4

Bit	Field	Type/InitVal	Description
8:7	SA2OvrPtr	RW 0x0	Override Source Address #2 Pointer **NOTE:** Valid only if SA2OvrEn is set.
9	SA3OvrEn	RW 0x0	Override Source Address #3 Control
11:10	SA3OvrPtr	RW 0x0	Override Source Address #3 Pointer. **NOTE:** Valid only if SA3OvrEn is set.
12	SA4OvrEn	RW 0x0	Override Source Address #4 Control
14:13	SA4OvrPtr	RW 0x0	Override Source Address #4 Pointer **NOTE:** Valid only if SA4OvrEn is set.
15	SA5OvrEn	RW 0x0	Override Source Address #5 Control
17:16	SA5OvrPtr	RW 0x0	Override Source Address #5 Pointer **NOTE:** Valid only if SA5OvrEn is set.
18	SA6OvrEn	RW 0x0	Override Source Address #6 Control.
20:19	SA6OvrPtr	RW 0x0	Override Source Address #6 Pointer **NOTE:** Valid only if SA6OvrEn is set.
21	SA7OvrEn	RW 0x0	Override Source Address #7 Control
23:22	SA7OvrPtr	RW 0x0	Override Source Address #7 Pointer **NOTE:** Valid only if SA7OvrEn is set.
24	DAOvrEn	RW 0x0	Override Destination Address Control 0x0 - No Override. 0x1 - Override is enabled.
26:25	DAOvrPtr	RW 0x0	Override Destination Address Pointer Specifies the register from which the override parameters will be taken. 0x0 = Target and attributes are taken from XEBAR0. Address[63:32] taken from XEHARR0. 0x1 = Target and attributes are taken from XEBAR1. Address[63:32] taken from XEHARR1. 0x2 = Target and attributes are taken from XEBAR2. Address[63:32] taken from XEHARR2. 0x3 = Target and attributes are taken from XEBAR3. Address[63:32] taken from XEHARR3. **NOTE:** Valid only if DAOvrEn is set.
27	NDAOvrEn	RW 0x0	Override Next Descriptor Address Control 0x0 = No Override. 0x1 = Override is enabled.

Table 583: XOR Engine [0..1] Address Override Control (XExAOCR) (Continued)
Offset: XE0AOCR 0x60BA0, XE1AOCR 0x60BA4

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. 0x0 = Target and attributes are taken from XEBAR0. Address[63:32] taken from XEHARR0. 0x1 = Target and attributes are taken from XEBAR1. Address[63:32] taken from XEHARR1. 0x2 = Target and attributes are taken from XEBAR2. Address[63:32] taken from XEHARR2. 0x3 = Target and attributes are taken from XEBAR3. Address[63:32] taken from XEHARR3. **NOTE:** Valid only if NDAOvrEn is set.
31:30	Reserved	RES 0x0	Reserved

A.16.5 XOR Engine ECC/MemInit Registers

Table 584: XOR Engine [0..1] Destination Pointer (XExDPR0)
Offset: XOR0 0x60BB0, XOR1 0x60BB4

Bit	Field	Type/Init Val	Description
31:0	DstPtr	RW 0x0	Points to target block of ECC/MemInit operations. **NOTE:** Valid only on ECC, MemInit modes.

Table 585: XOR Engine[0..1] Block Size (XExBSR)
Offset: XOR0 0x60BC0, XOR1 0x60BC4

Bit	Field	Type/Init Val	Description
31:0	BlockSize	RW 0x0	Size of Block in bytes for ECC or MemInit Operation. Along with XE0DPR or XE1DPR (Destination pointer registers), defines the target block for those operations. Minimum value: 128B. Maximum Value: 4 GB The value 0x00000000 stands for 4 GB block size. Block must not cross 4 GB boundary. **NOTE:** Valid only on ECC, MemInit modes.

Table 586: XOR Engine Timer Mode Control (XETMCR)
Offset: 0x60BD0

Bit	Field	Type/Init Val	Description
0	TimerEn	RW 0x0	Enable Timer Mode. Enables triggering ECC operation with timer. If Enabled the target block will be divided to Sections according to SectionSizeCtrl value and the ECC timer will be enabled. Upon expiration of the timer, one section will be processed. When the timer expires in the second time, the next section will be processed, and so on, until all the target block is processed. The XOR Engine will than start cleaning the target block all over again. In order to stop the operation, XEstop must be set. 1 = Timer Mode Enabled. 0 = Timer Mode Disabled. the ECC timer will be activated upon setting XEstart of a channel in ECC timer operation mode. **NOTE:** Valid only on ECC mode.
7:1	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 2GB). Must be less than Block Size (XEBSR) Reserved values: 0..6 **NOTE:** Valid only on ECC timer mode.
31:13	Reserved	RO 0x0	Reserved

Table 587: XOR Engine Timer Mode Initial Value (XETMIVR)
Offset: 0x60BD4

Bit	Field	Type/Init Val	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. It 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 588: XOR Engine Timer Mode Current Value (XETMCVR)
Offset: 0x60BD8

Bit	Field	Type/Init Val	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 589: XOR Engine Initial Value Low (XEIVRL)
Offset: 0x60BE0

Bit	Field	Type/Init Val	Description
31:0	InitValL	RW 0x0	LSB of Initial Value to be written cyclically to target block in MemInit mode. Mapped to bits[31:00] 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.1

Table 590: XOR Engine Initial Value High (XEIVRH)
Offset: 0x60BE4

Bit	Field	Type/Init Val	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.

A.17 General Purpose Port Registers

Table 591: GPIO Registers Map

Register	Offset	Table, Page
GPIO Data Out Register	0x10100	Table 592, p. 512
GPIO Data Out Enable Control Register	0x10104	Table 593, p. 512
GPIO Blink Enable Register	0x10108	Table 594, p. 513
GPIO Data In Polarity Register	0x1010C	Table 595, p. 513
GPIO Data In Register	0x10110	Table 596, p. 513
GPIO Interrupt Cause Register	0x10114	Table 597, p. 513
GPIO Interrupt Mask Register	0x10118	Table 598, p. 514
GPIO Interrupt Level Mask Register	0x1011C	Table 599, p. 514

A.17.1 GPIO Registers Description

Table 592: GPIO Data Out Register
Offset:0x10100

Bits	Field	Type/InitVal	Description
25:0	GPIODOut	RW 0x0	GPIO Output Pins Value, bit per each GPIO pin
31:26	Reserved	RW 0x0	Reserved

Table 593: GPIO Data Out Enable Control Register
Offset:0x10104

Bits	Field	Type/InitVal	Description
25:0	GPIODOutEn	RW 0xFFFF	GPIO Port Output Enable This field is active low. Data is driven when the corresponding bit value is 0.
31:26	Reserved	RW 0x0	Reserved

Table 594: GPIO Blink Enable Register
Offset:0x10108

Bits	Field	Type/InitVal	Description
25:0	GPIODBlink	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). When set and the corresponding pin on the MPP interface is set for the SATA LED indication, then the LED blinks every ~100 ms (a period of 2^24 TCLK clocks).
31:26	Reserved	RW 0x0	Reserved

Table 595: GPIO Data In Polarity Register
Offset:0x1010C

Bits	Field	Type/InitVal	Description
25:0	GPIODataInActLow	RW 0x0	GPIO Data in Active Low When set to 1 GPIO Data In Register reflects the inverted value of the corresponding pin.
31:26	Reserved	RW 0x0	Reserved

Table 596: GPIO Data In Register
Offset:0x10110

Bits	Field	Type/InitVal	Description
25:0	GPIODIn	RO 0x0	Each bit in this field reflects the value of the corresponding GPIO pin. If corresponding bit in GPIO Data In Polarity Register is cleared to 0, the bit reflects the pin value with no change. If corresponding bit in GPIO Data In Polarity Register is set to 1, the bit reflects the pin inverted value.
31:26	Reserved	RW 0x0	Reserved

Table 597: GPIO Interrupt Cause Register
Offset:0x10114

Bits	Field	Type/InitVal	Description
25:0	GPIOInt	RW0C 0x0	A bit in this field is set on the transition of the corresponding bit in the <GPIODIn> field, in the GPIO Data In Register, from 0 to 1.

Table 597: GPIO Interrupt Cause Register (Continued)
Offset:0x10114

Bits	Field	Type/InitVal	Description
31:26	Reserved	RW 0x0	Reserved

Table 598: GPIO Interrupt Mask Register
Offset:0x10118

Bits	Field	Type/InitVal	Description
25:0	GPIOIntEdge Mask	RW 0x0	GPIO Interrupt Edge Sensitive Mask The mask bit for each cause bit in the <GPIOInt> field of the GPIO Interrupt Cause Register (see Table 597 on page 513). 0 = Interrupt is masked. 1 = Interrupt is enabled. The mask only affects the assertion of the interrupt bits in Main Interrupt Cause Register (Table 99 p. 253). It does not affect the setting of bits in the GPIO Interrupt Cause Register.
31:26	Reserved	RW 0x0	Reserved

Table 599: GPIO Interrupt Level Mask Register
Offset:0x1011C

Bits	Field	Type/InitVal	Description
25:0	GPIOIntLevel Mask	RW 0x0	GPIO Interrupt Level Sensitive Mask The mask bit for each bit in the <GPIODIn> field of the GPIO Data In Register (see Table 596 on page 513). 0 = Interrupt is masked. 1 = Interrupt is enabled The mask only affects the assertion of the interrupt bit in Main Interrupt Cause Register. It does not affect the value of bits in the GPIO Data In Register.
31:26	Reserved	RW 0x0	Reserved

Note

To set an edge sensitive interrupt, set the corresponding bit in GPIO Interrupt Mask Register.
To set a level sensitive interrupt, set the corresponding bit in GPIO Interrupt Level Mask Register.

A.18 Pins Multiplexing Interface Registers

Table 600: MPP Register Map

Register	Offset	Page
MPP Control 0 Register	0x10000	Table 601, p. 515
MPP Control 1 Register	0x10004	Table 602, p. 516
MPP Control 2 Register	0x10050	Table 603, p. 516
Device Multiplex Control Register	0x10008	Table 604, p. 517
Sample at Reset Register	0x10010	Table 605, p. 518

A.18.1 MPP Registers

Table 601: MPP Control 0 Register
Offset:0x10000

Bits	Field	Type/InitVal	Description
3:0	MPPSel0	RW Sample at reset	MPP0 Select See the MPP Function Summary table and the Reset Configuration table in the *88F5182 Feroceon*® Storage Networking SoC*, Datasheet*.
7:4	MPPSel1	RW Sample at reset	MPP1 Select See field MPPSel0.
11:8	MPPSel2	RW Sample at reset	MPP2 Select See field MPPSel0.
15:12	MPPSel3	RW Sample at reset	MPP3 Select See field MPPSel0.
19:16	MPPSel4	RW Sample at reset	MPP4 Select See field MPPSel0.
23:20	MPPSel5	RW Sample at reset	MPP5 Select See field MPPSel0.
27:24	MPPSel6	RW Sample at reset	MPP6 Select See field MPPSel0.
31:28	MPPSel7	RW Sample at reset	MPP7 Select See field MPPSel0.

Table 602: MPP Control 1 Register
Offset:0x10004

Bits	Field	Type/InitVal	Description
3:0	MPPSel8	RW Sample at reset	MPP8 Select See the MPP Function Summary table in the *88F5182 Feroceon*® Storage Networking SoC, *Datasheet*.
7:4	MPPSel9	RW Sample at reset	MPP9 Select See field MPPSel8.
11:8	MPPSel10	RW Sample at reset	MPP10 Select See field MPPSel8.
15:12	MPPSel11	RW Sample at reset	MPP11 Select See field MPPSel8.
19:16	MPPSel12	RW Sample at reset	MPP12 Select See field MPPSel8.
23:20	MPPSel13	RW Sample at reset	MPP13 Select See field MPPSel8.
27:24	MPPSel14	RW Sample at reset	MPP14 Select See field MPPSel8.
31:28	MPPSel15	RW Sample at reset	MPP15 Select See field MPPSel8.

Table 603: MPP Control 2 Register
Offset:0x10050

Bits	Field	Type/InitVal	Description
3:0	MPPSel16	RW Sample at reset	MPP16 Select See the MPP Function Summary table in the *88F5182 Feroceon*® Storage Networking SoC, *Datasheet*.
7:4	MPPSel17	RW Sample at reset	MPP17 Select See field MPPSel16.
11:8	MPPSel18	RW Sample at reset	MPP18 Select See field MPPSel16.

Table 603: MPP Control 2 Register (Continued)
Offset:0x10050

Bits	Field	Type/InitVal	Description
15:12	MPPSel19	RW Sample at reset	MPP19 Select See field MPPSel16.
31:16	Reserved	RES 0x0	Reserved

Table 604: Device Multiplex Control Register
Offset:0x10008

Bits	Field	Type/InitVal	Description
31:0	Reserved	RES 0x03FF0000	Reserved **NOTE:** Must be 0x03FF0000.

Note

For additional information about the reset pins referred to in the Sample at Reset Register, see the Reset Configuration table in the *88F5182 Feroceon*® Storage Networking SoC*, Datasheet* for this device.

Table 605: Sample at Reset Register
Offset:0x10010

NOTE: Writing to this register has no effect on the reset-strapped features. This is a status register only. For further details on the functionality of each bit see the reset section in the *88F5182 Datasheet*.

Bits	Field	Type/InitVal	Description
25:0	SampleAtReset	RW Sample during reset	[0] = DEV_OEn [1] = DEV_WEn[1] [2] = DEV_WEn[0] [3] = DEV_BURSTn [4] = DEV_AD[14] [5] = DEV_AD[13] [6] = Reserved (1'b0) [10:7] = DEV_AD[11:8] [11] = DEV_A[2] [13:12] = DEV_A[1:0] [15:14] = DEV_ALE[1:0] [17:16] = DEV_AD[7:6] [18] = DEV_AD[12] [19] = DEV_AD[15] [20] = DEV_AD[4] [21] = DEV_AD[2] [22] = DEV_AD[5] [23] = DEV_AD[3] [25:24] = DEV_AD[1:0]
31:26	Reserved	RW 0x0	Reserved

B Revision History

Document Type	Revision	Date
Release	A	September 26, 2005
Release	B	January 23, 2008

1. Throughout the document changed Px to Ge, e.g., Px_TXERR and Px_RXERR to GE_TXERR and GE_RXERR.
2. Updated Section , Related Documents, on page 15.
3. Section 3, Feroceon® 1850 CPU Core, on page 23 updated reference to CPU documentation.
4. Updated Sel column in Table 2, Table 3, and Table 4, Address Multiplex for 16-bit DDR SDRAM Interface, on page 28.
5. Table 5, DDR SDRAM Timing Parameters, on page 28 “The DDR SDRAM controller supports CL of 1.5, 2, 2.5, 3, or 4 cycles.“ added “or 5”.
6. Section 6.4, PCI Master Configuration Cycles, on page 52 added sentence after “A subsequent read or write...”
7. Removed Section 7.9.7, Interface I/Os.
8. Updated Figure 9, PCI Type 0 Configuration Transaction Address Translation, on page 52.
9. Added Section 6.11.5, Built-In Self Test (BIST), on page 67.
10. In Section 9, Gigabit Ethernet Controller Interface, on page 95 changed all instances of 802.3u to 802.3, 802.3x to 802.3, 802.1p to 802.1q, 802.3ab to 802.3, 802.3z to 802.3.
11. Section 9.1, Functional Description, on page 95 deleted the following sentence: “There is no Auto-Negotiation for speed on the PCS. “
12. Rewrote Section 9.7.3.3, Signals Encoding, on page 120.
13. Rewrote the first paragraph of Section 9.10, Inter-Packet Gap, on page 122.
14. Deleted the note (For further information see *USB-HS High-Speed Controller Core Reference.) in* Section 10.1, Functional Description, on page 132.
15. Updated Figure 30, Typical DES/3DES Packet Encryption Flow, on page 143
16. Updated Figure 34, Security Acceleration Flow for Packet Processing, on page 150
17. Added formula and note to Section 12.1.5, Baud Rate Register, on page 161.
18. Table 53, UART Pin Assignments, on page 165 deleted the following sentence: “The UA0_CTSn signals are a modem-status input whose condition can be tested by the host processor or by the UART when in Autoflow mode.”.
19. Section 13, UART Interface, on page 165 removed bullet stating that the device supports Synchronous interface.
20. Table 54, Device Controller Pin Assignments, on page 166 changed DEV_READY from O to I.
21. Updated Figure 43, Pipeline Sync Burst SRAM Read Example, on page 172.
22. Section 14.12, Boot from NAND Flash, on page 177 updated
23. Table 55, IDMA Descriptor Definitions, on page 180 added Own bit.
24. Figure 53, IDMA Descriptors, on page 180 added Own bit.
25. Removed Section 16.3.3, Cache Coherency Support, on page 197.

26. Section 21.2.1, Boot from Flash/NAND Flash, on page 212, in the sentence beginning “The Feroceon core starts to boot...” changed the cross-references from Window0 Control Register (Table 72 p. 243) and the Window0 Base Register (Table 73 p. 244) to Window7 Control Register (Table 90 p. 249) and the Window7 Base Register (Table 91 p. 249).
27. Added PCI Mode (Table 247 p. 331)
28. Added PCI Status and Command (Table 282 p. 344).

29. Added PCI Express Debug Control Register (Table 179 p. 294)
30. Added SATAHC LED Configuration Register (Table 332 p. 367)
31. Section A.8, Serial-ATA Host Controller (SATAHC) Registers, on page 358, added SATAHC LED Configuration Register (Table 332 p. 367).

Document Type	Revision	Date
32. In Section A.9, Gigabit Ethernet Controller Registers, on page 408 changed all instances of 802.3u to 802.3, 802.3x to 802.3, 802.1p to 802.1q, 802.3ab to 802.3, 802.3z to 802.3. 33. Cryptographic Engines and Security Accelerator Interrupt Cause Register (Table 519 p. 468) updated descriptions of bits 7 and 8 34. Channel Next Descriptor Pointer Register (Table 555 p. 489) changed “The address must be 32-byte aligned (bits [3:0] must be 0x0).” to “16-byte aligned”.		
Changed document classification.		
Release	C	April 29, 2008
Changed document classification.		

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.752.9028

www.marvell.com

Marvell. Moving Forward Faster