



GATEWAYS

Zealand Ltd.
12109742

Link Street™ 88E6208/88E6218 Datasheet

SOHO Gateway SOC with
ARM9E™ CPU, 10/100 Switch and PHYs

Part 2 of 3: CPU & Peripherals

Doc. No. MV-S101300-02, Rev. –
April 3, 2003

MOVING FORWARD
FASTER®

12qru965pi-fg1egpso
MARVELL CONFIDENTIAL





Document Status

Advanced Information	This document contains design specifications for initial product development. Specifications may change without notice. Contact Marvell Field Application Engineers for more information.
Preliminary Information	This document contains preliminary data, and a revision of this document will be published at a later date. Specifications may change without notice. Contact Marvell Field Application Engineers for more information.
Final Information	This document contains specifications on a product that is in final release. Specifications may change without notice. Contact Marvell Field Application Engineers for more information.
Revision Code:	Rev. –
Preliminary	Technical Publication: 0.36

Disclaimer

This document provides Preliminary information about the products described, and such information should not be used for purpose of final design. Visit the Marvell® web site at www.marvell.com for the latest information on Marvell products.

No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose, without the express written permission of Marvell. Marvell retains the right to make changes to this document at any time, without notice. Marvell makes no warranty of any kind, expressed or implied, with regard to any information contained in this document, including, but not limited to, the implied warranties of merchantability or fitness for any particular purpose. Further, Marvell does not warrant the accuracy or completeness of the information, text, graphics, or other items contained within this document. Marvell makes no commitment either to update or to keep current the information contained in this document. Marvell products are not designed for use in life-support equipment or applications that would cause a life-threatening situation if any such products failed. Do not use Marvell products in these types of equipment or applications. The user should contact Marvell to obtain the latest specifications before finalizing a product design. Marvell assumes no responsibility, either for use of these products or for any infringements of patents and trademarks, or other rights of third parties resulting from its use. No license is granted under any patents, patent rights, or trademarks of Marvell. These products may include one or more optional functions. The user has the choice of implementing any particular optional function. Should the user choose to implement any of these optional functions, it is possible that the use could be subject to third party intellectual property rights. Marvell recommends that the user investigate whether third party intellectual property rights are relevant to the intended use of these products and obtain licenses as appropriate under relevant intellectual property rights.

Marvell comprises Marvell Technology Group Ltd. (MTGL) and its subsidiaries, Marvell International Ltd. (MIL), Marvell Semiconductor, Inc. (MSI), Marvell Asia Pte Ltd. (MAPL), Marvell Japan K.K. (MJKK), Marvell Semiconductor Israel Ltd. (MSIL), SysKonnect GmbH, and Radlan Computer Communications, Ltd.

Export Controls. With respect to any of Marvell's Information, the user or recipient, in the absence of appropriate U.S. government authorization, agrees: 1) not to re-export or release any such information consisting of technology, software or source code controlled for national security reasons by the U.S. Export Control Regulations ("EAR"), to a national of EAR Country Groups D:1 or E:2; 2) not to export the direct product of such technology or such software, to EAR Country Groups D:1 or E:2, if such technology or software and direct products thereof are controlled for national security reasons by the EAR; and, 3) in the case of technology controlled for national security reasons under the EAR where the direct product of the technology is a complete plant or component of a plant, not to export to EAR Country Groups D:1 or E:2 the direct product of the plant or major component thereof, if such direct product is controlled for national security reasons by the EAR, or is subject to controls under the U.S. Munitions List ("USML"). At all times hereunder, the recipient of any such information agrees that they shall be deemed to have manually signed this document in connection with their receipt of any such information.

Copyright 2000. Marvell. All rights reserved. Marvell, the Marvell logo, Moving Forward Faster, Alaska, and GalNet are registered trademarks of Marvell. Discovery, FastWriter, GalTis, Horizon, Libertas, Link Street, NetGX, PHY Advantage, Prestera, Raising The Technology Bar, UniMAC, Virtual Cable Tester, and Yukon are trademarks of Marvell. All other trademarks are the property of their respective owners.

Marvell

Table of Contents

PREFACE	9
About this Document	9
Related Documentation	9
Document Conventions	9
 SECTION 1. OVERVIEW	 11
1.1 Link Street SOHO Gateway Router Features	11
 SECTION 2. DIFFERENCES IN FEATURES BETWEEN THE 88E6208 AND THE 88E6218 DEVICES	 13
 SECTION 3. MEMORY MAP	 14
 SECTION 4. INTERCONNECT BUS	 15
4.1 Bus Arbitration	15
 SECTION 5. ARM946E-S™ EMBEDDED PROCESSOR	 17
5.1 ARM946E-S™ Processor Features	17
5.2 CPU Timers	18
5.2.1 Timer Operation	19
5.2.2 Timer Programming Sequence	19
5.3 Interrupts	20
 SECTION 6. MEMORY CONTROLLER	 22
6.1 Functional Description	22
6.2 Address Space	22
6.3 SDRAM	23
6.3.1 X32 Data Bus (for the 88E6218 Only)	23
6.3.2 X16 Data Bus	24
6.3.3 Page Open Mode for SDRAM Device Banks	24
6.4 Flash and ROM Device Banks	25
6.4.1 Burst Access for Asynchronous Device Bank	25
6.4.2 Packing	25
6.5 Boot Device Bank	28
6.5.1 Default Configuration During Power-Up	28



SECTION 7. UNIMAC™ UNIFIED FAST ETHERNET MEDIA ACCESS CONTROL	29
7.1 Functional Description.....	29
7.2 Features.....	30
7.3 Media Access Controller (MAC).....	31
7.4 802.3x Flow Control (Non Marvell Header Mode) — 88E6218 Only	31
7.5 Zero Padding for Short Packets (Non Marvell Header Mode)	32
7.6 CRC Generation.....	32
7.7 Marvell Header Mode.....	32
7.8 Address Filtering (88E6218 Only).....	33
7.8.1 Hash Table Structure.....	33
7.8.2 Hash Modes	35
7.8.3 Hash Entry	35
7.8.4 Table Filling	36
7.8.5 Address Filtering Process (88E6218 Only)	37
7.9 Rx Queuing	39
7.9.1 88E6208 Rx Queuing	39
7.9.2 88E6218 Rx Priority Queuing — Marvell Header Mode Enabled.....	40
7.9.3 88E6218 Rx Priority Queuing — Marvell Header Mode Disabled	40
7.10 DMA Transfer	43
7.11 Receive Operation	44
7.11.1 RX DMA Descriptors	45
7.11.2 RX DMA Pointer Registers	47
7.12 Transmit Operation	47
7.12.1 TX DMA Descriptors	50
7.12.2 TX DMA Pointer Registers	52
7.12.3 TX DMA Notes.....	52
7.13 MII Serial Management Interface (SMI)	53
SECTION 8. INDEPENDENT DMA (88E6218 ONLY).....	54
8.1 Functional Description.....	54
8.1.1 DMA Transfers	54
8.2 DMA Channel Registers.....	54
8.2.1 DMA Channel Descriptor Registers.....	54
8.2.2 DMA Channel Control Registers	55
8.3 Chain and Non-Chain Modes.....	55
8.3.1 Chain Mode	55
8.3.2 Non-Chain Mode	57
8.4 Interrupts.....	57
8.5 DMA Operations	58
8.5.1 Restarting a Disabled Channel.....	58
8.5.2 Reprogramming an Active Channel.....	58
8.5.3 Current Descriptor Pointer Registers.....	58

SECTION 9. TWO-WIRE SERIAL INTERFACE (TWSI)	59
9.1 Functional Description	59
9.2 TWSI Bus Operation	59
9.3 Soft Reset	60
9.4 Peripheral Clock Speeds	61
9.5 TWSI Registers	61
9.5.1 TWSI Clock Prescaler Register	61
9.5.2 TWSI Global Control Register	61
9.5.3 TWSI Control Register	62
9.5.4 TWSI Status Register	62
9.5.5 TWSI Interrupt Mask Register	62
9.5.6 TWSI Data Register	62
9.5.7 TWSI Interrupt Source Register	62
9.6 TWSI Operation	62
9.6.1 Master Write Access	63
9.6.2 Master Read Access	63
SECTION 10. GENERAL PURPOSE I/O PORT (GPIO)	64
10.1 Functional Description	64
10.2 GPIO Control Registers	64
10.3 GPIO LED Support	64
10.3.1 Pulse Stretching for Status LEDs	65
10.3.2 Blink Rate for Link/Activity LED	65
10.4 GPIO Slew Rate Control	65
10.5 GPIO Interrupts	66
10.6 GPIO Port Pin Assignments	66
SECTION 11. UART	68
11.1 Functional Description	68
11.2 Interface Description	68
11.3 Using the UART as a Test Port	68
SECTION 12. WATCHDOG TIMER	69



APPENDIX A 88E6208 AND 88E6218 CPU AND PERIPHERALS REGISTER SET	70
A.1 Register Overview	76
A.1.1 Register Description	76
A.1.2 Register Types	76
A.2 CPU Interface Registers.....	77
A.2.1 CPU Register Map.....	77
A.2.2 CPU Control Registers	78
A.2.3 CPU Interrupt Registers	79
A.2.4 CPU Timer Registers.....	85
A.3 Memory Controller Registers	89
A.3.1 Memory Controller Register Map.....	89
A.3.2 Control and Status Register	90
A.3.3 Configuration Register for SDRAM/Device.....	90
A.3.4 Timing Parameter Registers.....	92
A.4 UniMAC™ Unified Fast Ethernet Media Access Control Registers.....	96
A.4.1 Register Map	96
A.4.2 UniMAC™ Registers	97
A.4.3 IP Differentiated Service Registers.....	109
A.4.4 UniMAC Descriptor Pointer Registers	110
A.5 Independent DMA Registers (88E6218 Only).....	113
A.5.1 Independent DMA Register Map	113
A.5.2 Independent DMA Channel Descriptor Registers.....	114
A.5.3 DMA Channel Control Registers	116
A.6 Two-Wire Serial Interface Registers	122
A.6.1 Two-Wire Serial Interface Register Map.....	122
A.6.2 Two-Wire Serial Interface Registers.....	122
A.7 General Purpose I/O Port (GPIO) Registers	126
A.7.1 GPIO Register Map	126
A.7.2 GPIO Registers	127
A.8 UART Registers	139
A.8.1 UART Register Map	139
A.8.2 UART Registers.....	139
A.9 Watchdog Register Description	145
APPENDIX B. MEMORY CONTROLLER APPLICATION EXAMPLES	147
B.1 Application Examples	147
B.1.1 Memory Controller with SDRAM.....	147
B.1.2 Memory Controller with FLASH/ROM.....	149
APPENDIX C. LIST OF ABBREVIATIONS	150
APPENDIX D. REVISION HISTORY.....	151

List of Tables

Table 1:	Document Conventions	9
Table 2:	Differences Between 88E6208 and 88E6218	13
Table 3:	Memory Map	14
Table 4:	IRQ Interrupt Sources	20
Table 5:	X32 Data Bus Configuration (88E6218 Only)	23
Table 6:	X16 Data Bus Configuration	24
Table 7:	Asynchronous Device Bank Access with Byte Packing	25
Table 8:	Asynchronous Device Bank Access Without Byte Packing	27
Table 9:	Default Configuration	28
Table 10:	Hash Table Entry Fields	34
Table 11:	Terms Used in Address Filtering	37
Table 12:	Packet Filtering Status	39
Table 13:	Receive Logic — 88E6208 only	39
Table 14:	Receive Priority Logic — 88E6218 only	40
Table 15:	Writing IP DSCP Priority Example	42
Table 16:	Writing VLAN Priority Example	42
Table 17:	Writing IP DSCP and VLAN Priority Example	42
Table 18:	Writing IP DSCP and VLAN Priority Register Mapping Example	43
Table 19:	UniMAC RX Descriptor — Command/Status Word	45
Table 20:	UniMAC RX Descriptor — Buffer Size / Byte Count	47
Table 21:	UniMAC RX Descriptor — Buffer Pointer	47
Table 22:	UniMAC RX Descriptor — Next Descriptor Pointer	47
Table 23:	UniMAC TX Descriptor — Command/Status Word	50
Table 24:	UniMAC TX Descriptor — Byte Count	51
Table 25:	UniMAC TX Descriptor — Buffer Pointer	52
Table 26:	UniMAC TX Descriptor — Next Descriptor Pointer	52
Table 27:	DMA Channel Descriptor Registers	55
Table 28:	Uclk Frequencies Based on SYSCLK and CLK_DIV	61
Table 29:	GPIO Port Pin Assignments	66
Table 30:	UART Pin Definitions	68
Table 31:	CPU Register Map Table	77
Table 51:	Memory Controller Register Map	89
Table 57:	UniMAC Register Map Table	96
Table 86:	IDMA Register Map	113
Table 106:	TWSI Register Map	122
Table 116:	GPIO Register Map	126
Table 138:	UART Register Map Table	139
Table 151:	Watchdog Register Map	145
Table 155:	Revision History	151



List of Figures

Figure 1:	Configurable Weights Arbiter	16
Figure 2:	88E6218 CPU Sub-section Block Diagram	18
Figure 3:	Timer Control Register Configuration	19
Figure 4:	88E6218 UniMAC Unit Simplified Block Diagram	29
Figure 5:	88E6208 UniMAC Unit Simplified Block Diagram	30
Figure 6:	UniMAC Hash Table Entry	34
Figure 7:	Address Chain	36
Figure 8:	Address Filtering Process	38
Figure 9:	Type of Service Receive Queuing Algorithm (88E6218 Only)	41
Figure 10:	UniMAC Descriptors and Buffers	43
Figure 11:	UniMAC™ RX DMA Descriptor	45
Figure 12:	UniMAC Packet Transmission Example	49
Figure 13:	UniMAC TX Descriptor	50
Figure 14:	UniMAC TX Buffer Alignment Restrictions (2 byte payload)	50
Figure 15:	Chain Mode	56
Figure 16:	TWSI Examples	60
Figure 17:	1X32 SDRAM (88E6218 Only)	147
Figure 18:	2X16 SDRAM (88E6218 Only)	148
Figure 19:	1X16 SDRAM	148
Figure 20:	1X16 Flash/ROM	149
Figure 21:	1X8 Flash/ROM	149
Figure 22:	1X32 Flash/ROM (88E6218 Only)	149

Preface

About this Document

88E6208/88E6218 Datasheet, Part 2 of 3: CPU & Peripherals provides a description of the ARM946E-S™ CPU and each of the peripheral interfaces of the Link Street™ 88E620888E6218 SOHO Gateway SOC with CPU, Switch, and PHYs and includes the related register tables. This volume is part of a three-volume set that describes the hardware features, the ARM946E-S™ CPU and peripheral interfaces and the switch core and PHYs of the Link Street™ 88E620888E6218 SOHO Gateway SOC with CPU, Switch, and PHYs. The other two manuals in this set are:

- *88E6208/88E6218 Datasheet, Part 1 of 3: Overview, Pinout, & Electrical Specifications* which provides a features list and overview describing the features in the Link Street™ 88E620888E6218 SOHO Gateway SOC with CPU, Switch, and PHYs, and also provides the pin description and pin map and the electrical specifications. In addition, Part 1 provides an overview to the complete three-volume set.
- *88E6208/88E6218 Datasheet, Part 3 of 3: Switch Core, PHYs, & Related Registers* which provides a description of the switch core and PHYs of the Link Street™ 88E620888E6218 SOHO Gateway SOC with CPU, Switch, and PHYs and includes the related register tables.

Related Documentation

The following Link Street™ 88E620888E6218 SOHO Gateway SOC with CPU, Switch, and PHYs documents relate to this manual.

- *88E6208/88E6218 Datasheet, Part 1 of 3: Overview, Pinout, & Electrical Specifications*, Document Number MV-S101300-01
- *88E6208/88E6218 Datasheet, Part 3 of 3: Switch Core, PHYs, & Related Registers*, Document Number MV-S101300-03

Document Conventions

Table 1: Document Conventions

Document Conventions	
The following name and usage conventions are used in this document:	
Signal Range	A signal name followed by a range enclosed in brackets represents a range of logically related signals. The first number in the range indicates the most significant bit (MSb) and the last number indicates the least significant bit (LSb). Example: DB_Addr[12:0]
Active Low Signals n	A lower-case n symbol at the end of a signal name indicates that the signal's active state occurs when voltage is low. Example: INTn



Table 1: Document Conventions (Continued)

State Names	State names are indicated in <i>italic</i> font. Example: <i>linkfail</i>
Register Naming Conventions	Register field names are indicated in a Ariel font. Example: RegInit OR in bracketed as shown: Example: Global_Control<DevEn>, Where Global Control represents the register name, and <DevEn> represents the register field name. 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.

Section 1. Overview



Note

This document applies to both the Marvell Link Street™ 88E6218 and 88E6208 devices. Any reference to the 88E6218 also applies to the 88E6208 devices unless specified otherwise. A summary of the differences between the two devices appears in [Section 2. "Differences in Features Between the 88E6208 and the 88E6218" on page 13.](#)

The 88E6218 and 88E6208 devices offer single-chip integrated SOHO gateway router solutions to reach full-wire-speed 100 Mbps WAN/LAN routing, which passes the FCC Class B EMI conformance test. Marvell is the first to implement this highly integrated gateway router design on a 2-layer PCB, achieving the most cost-effective form factor ready for manufacturing. The Marvell Link Street™ 88E6208 and 88E6218 SOHO gateway router devices allow OEM customers to expand their product offerings, addressing a worldwide home networking and residential gateway market.

The Marvell Link Street 88E6218 gateway router provides single chip integration, integrating the Company's Link Street 7-port Fast Ethernet (FE) switch to enhance network reliability, an ARM946E-S™ CPU to increase processing speed, and a software architecture for enhanced routing performance. In the 88E6208 gateway router the Fast Ethernet (FE) switch has 6 ports. The Marvell Link Street devices also leverage the Company's newest generation 0.15-micron PHY transceiver technology, which interfaces the chip to external cables. The Marvell PHYs incorporate Virtual Cable Tester™ (VCT) technology, which provides built-in cable diagnostic testing capability, quickly resolving cable problems and decreasing downtime.

The highly integrated Marvell Link Street gateway routers achieve the highest level of performance available — 100 Mbps full-wire-speed WAN/LAN routing. The devices utilize the Company's UniMAC™ architecture to accelerate routing performance with pre-packet processing, speeding IP routing and increasing CPU performance. UniMAC also enhances security for systems connected to WAN, LAN and De-Militarized Zone (DMZ) ports. System products integrating the Marvell Link Street 88E6218 device add quality-of-service (QoS) for converged voice/video/data networks, allowing IP services such as IP telephony and video-over-the-Internet.

The Marvell Link Street 88E6208 and 88E6218 gateway routers provide full-wire-speed routing and the lowest system cost implementation — enabling OEMs fast entry into the rapidly-growing SOHO networking market. In addition, the Link Street gateway routers achieve extremely low power consumption, providing OEMs with a complete gateway router design that runs much cooler and achieves a longer life span, resulting in higher reliability.

1.1 Link Street SOHO Gateway Router Features

The Link Street 88E6208 and 88E6218 SOHO gateway routers include a high-speed ARM946E-S™ processor, the Marvell Link Street FE 6-port/7-port switches, and five Marvell 0.15-micron FE transceivers, providing OEMs with a high-performance, cost-effective, low power solution.



Link Street 88E6208 SOHO Gateway Router

The cost-effective Link Street 88E6208 SOHO gateway router includes the following features:

- Full-wire-speed 100 Mbps WAN/LAN routing
- ARM946E-S™ CPU (with DSP Processor instruction extensions) @ 133 MHz
- 2-layer PCB reference design which passes FCC Class B EMI conformance testing, achieving lower system-cost router design
- 16 bit Memory Controller interface including 3 memory device banks (one boot Flash/ROM and two others) with 8 MB of memory per device bank
- Enhanced 6-port FE switch with UniMAC™ IP routing acceleration
- One UART interface for debugging and for backup phone line WAN connection
- Pin-compatible with the 88E6218 device

Link Street 88E6218 SOHO Gateway Router

The Link Street 88E6218 SOHO gateway router offers these additional features:

- Full-wire-speed 100 Mbps WAN/LAN routing, plus a faster CPU providing bandwidth for software VPN services
- ARM946E-S™ CPU (with DSP Processor instruction extensions) @ up to 150 MHz
- High-performance cache memory architecture with 8 KB instruction cache, 8 KB data cache, and 8 KB tightly coupled memory
- 32 bit Memory Controller interface including 4 memory device banks (one boot Flash/ROM and three others) with 64 MB of memory per device bank
- Quality-of-Service (QoS) enhanced 7-port FE switch including UniMAC™ IP routing acceleration, for video-over-the-Internet and audio-over-the-Internet delivery
- External MII port, which can run at up to 200 Mbps full-duplex, providing network expansion capability to an additional multi-port switch or wireless LAN

Section 2. Differences in Features Between the 88E6208 and the 88E6218 Devices

Table 2 summarizes the major lists the differences in feature sets between the 88E6208 and the 88E6218 SOHO gateway routers. Unless mentioned elsewhere in this document, all other functionality and features are the same for both parts.

Table 2: Differences Between 88E6208 and 88E6218

Feature	88E6208	88E6218
CPU		
Maximum CPU Speed	133 MHz	150 MHz
8K Data Tightly-Coupled Memory SRAM	No	Yes
Internal DMA	No	Yes
External Memory Width	16-bits	32-bits
Maximum Flash Size	8 MB	64 MB
External Chip Selects	3 - BootCSn, M_CSn[1:0]	4 - BootCSn, M_CSn[2:0]
M_STATUS pin for Flash Memory	No ¹	Yes
Address Bus pins	M_A[21:0]	M_A[23:0]
UniMAC™ QoS	No	Yes - 4 Queues
UniMAC DA Filter	No	Yes
SWITCH		
# of Switch Ports	6	7
External MII interface	No	Yes
Switch QoS	No	Yes - 4 Queues
Ingress Rate Limiting	No	Yes
Egress Rate Shaping	No	Yes
Max. UniMAC Speed	100 Mbps FD	200 Mbps FD
Embedded Memory	0.5 Mb	1 Mb

¹ A GPIO pin can be used instead.



Section 3. Memory Map

The Memory map for the 88E6208 and 88E6218 are hardwired as shown in Table 3. The same Memory map is used for the ARM946E-S™ and the DMA engines.

Table 3: Memory Map

Start Address	End Address	Size	Description	Note No.
0xC000_0000	0xFFFF_FFFF	1 GB	Memory space	1
0xA000_0000	0xBFFF_FFFF	512 MB	Reserved	2
0x9000_0000	0x9FFF_7FFF	128 MB	CPU Bus Only (Interrupts and Timers)	2
0x9800_0000	0x9800_1FFF	8 KB	D-TCM (Data Tightly Coupled Memory) for the 88E6218 device only. These addresses are reserved in the 88E6208 device.	3
0x9800_2000	0x9FFF_FFFF	~128 MB	Reserved	2
0x8000_E000	0x8000_FFFF	8 KB	Configuration IDMA registers	2 and 4
0x8000_C000	0x8000_DFFF	8 KB	Configuration Peripheral registers	2
0x8000_8000	0x8000_9FFF	8 KB	Configuration UniMAC™ registers	2
0x8000_6000	0x8000_7FFF	8 KB	Configuration Memory Controller registers	2
0x8000_2000	0x8000_3FFF	8 KB	Configuration CPU registers	2
0x0000_0000	0x7FFF_FFFF	2 GB	Memory space	1

- Addresses starting with 0x0XXX_XXXX, 0x4XXX_XXXX and 0xCXXX_XXXX will hit the same Base Address Register (BAR) in the Memory Controller unit.

Addresses starting with 0x1XXX_XXXX, 0x5XXX_XXXX and 0xDXXX_XXXX will hit the same BAR in the Memory Controller unit.

Addresses starting with 0x2XXX_XXXX, 0x6XXX_XXXX and 0ExXXX_XXXX will hit the same BAR in the Memory Controller unit.
- Accessing addresses that are not assigned specific registers in the registers tables or Memory space, may result in unpredictable behavior.
- The D-TCM location in the CPU's memory space is configurable through the ARM® Core CP15 Register 9. Refer to the ARM946E-S technical reference manual. The address shown in this table is the default setting used by the Marvell software package supplied with this product.
- The IDMA registers are only applicable to the 88E6218 device.

Section 4. Interconnect Bus

The interconnect bus of the 88E6208/88E6218 is the primary interconnect within the device's CPU subsystem. It is a multiplexed 32-bit Address/data bus, which carries read/write transactions between various units connected to it. Bus transaction can vary from one up to eight 32-bit word transfers.

The Shared bus interconnects the following units:

- ARM946E-S™ CPU core
- Memory interface unit
- UniMAC™ unit
- IDMA unit (88E6218 only)
- Peripheral unit (UART, TWSI)

The bus supports split read transactions, where a read transaction is composed of two independent phases: Read_request generated by the initiating unit and Read_response generated by the target unit.

The bus also supports transaction pipelining. For example, unit A can pipeline a new read transaction, before receiving back read data of previous read. More over, unit B can pipe a new read or write transaction, before unit A received read data back.

Bus arbitration is controlled by a programmable arbiter. See below.

4.1 Bus Arbitration

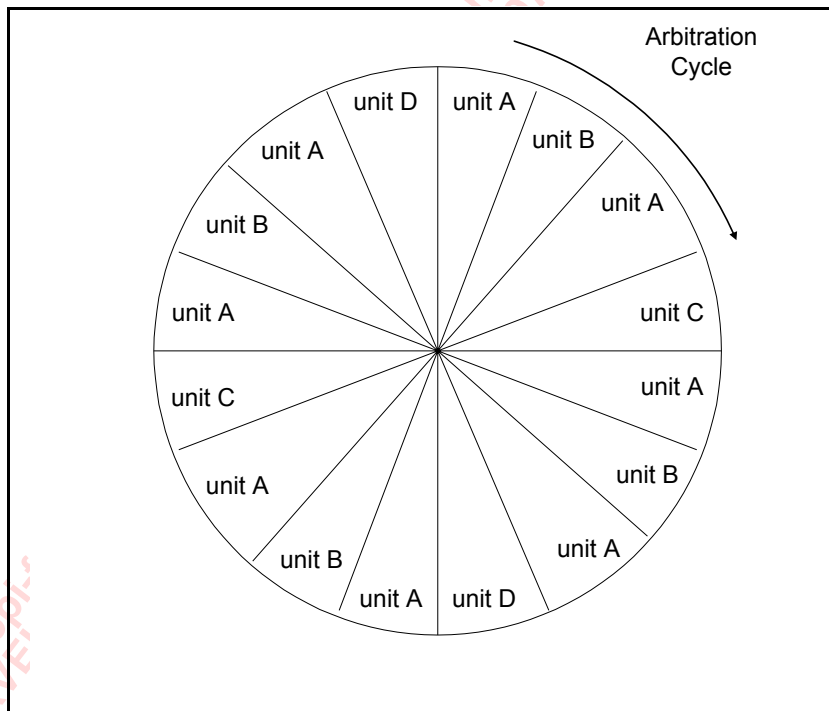
The arbiter is used to allocate a fair bandwidth of the shared bus to the different units. It has 16 slices, each of which can be assigned to any of the units on the bus. Refer to ABU registers description for details about the pizza arbiter control registers.

The arbiter follows a round robin arbitration scheme, where at each cycle it calculates which of the pending requests is to be served next.

The arbiter works on transaction basis, meaning, it would enter a new arbitration cycle, only when the current transaction has ended. This means that setting should be related to the typical transaction size generated by the different units. If for example, the typical transaction size of unit A is twice as large as the transaction size of unit B, unit B should get twice as many slices as unit A, to have the same bandwidth allocation for both units.

An example is shown in [Figure 1](#).

Figure 1: Configurable Weights Arbiter



In this example, unit A gets 50% of the bandwidth, unit B 25%, unit C and unit D 12.5% each. This “pizza” configuration allows the user also to guarantee minimum latency. Even if unit A does not require 50% bandwidth, the above configuration guarantees that unit A shall wait no more than one maximum size transaction, before being served.

There are two types of requests the arbiter serves: initiator request or target read response. The configurable pizza slices are only used for the initiator requests. Read response requests always have higher priority, and are served first. The arbiter maintains fixed round-robin arbitration between all simultaneous read responses.

At each clock cycle the arbiter samples all requests and gives the bus to the next agent according to the “pizza” setting. It is parked on last served unit. If next request is from this unit, there is no arbitration cycle penalty. In other cases, request is served only in next cycle.

An arbiter slice can also be marked as NULL. If marked as NULL, the arbiter works as if the NULL slice does not exist. For example, if only two requests are used, and they need to get the same bandwidth, the user can specify first slice per one request, second slice per the other request, and all the rest slices as NULL. This is equivalent to specifying half of the slices for one request and the other half for the other request.



Note

Once a unit is removed from the Arbiter Control register, this unit has no access to the bus. If for example, the UniMAC™ unit is removed from the arbiter, the UniMAC unit no longer has access to the bus. If it attempts to access the bus, the unit would get stuck.

Section 5. ARM946E-S™ Embedded Processor

5.1 ARM946E-S™ Processor Features

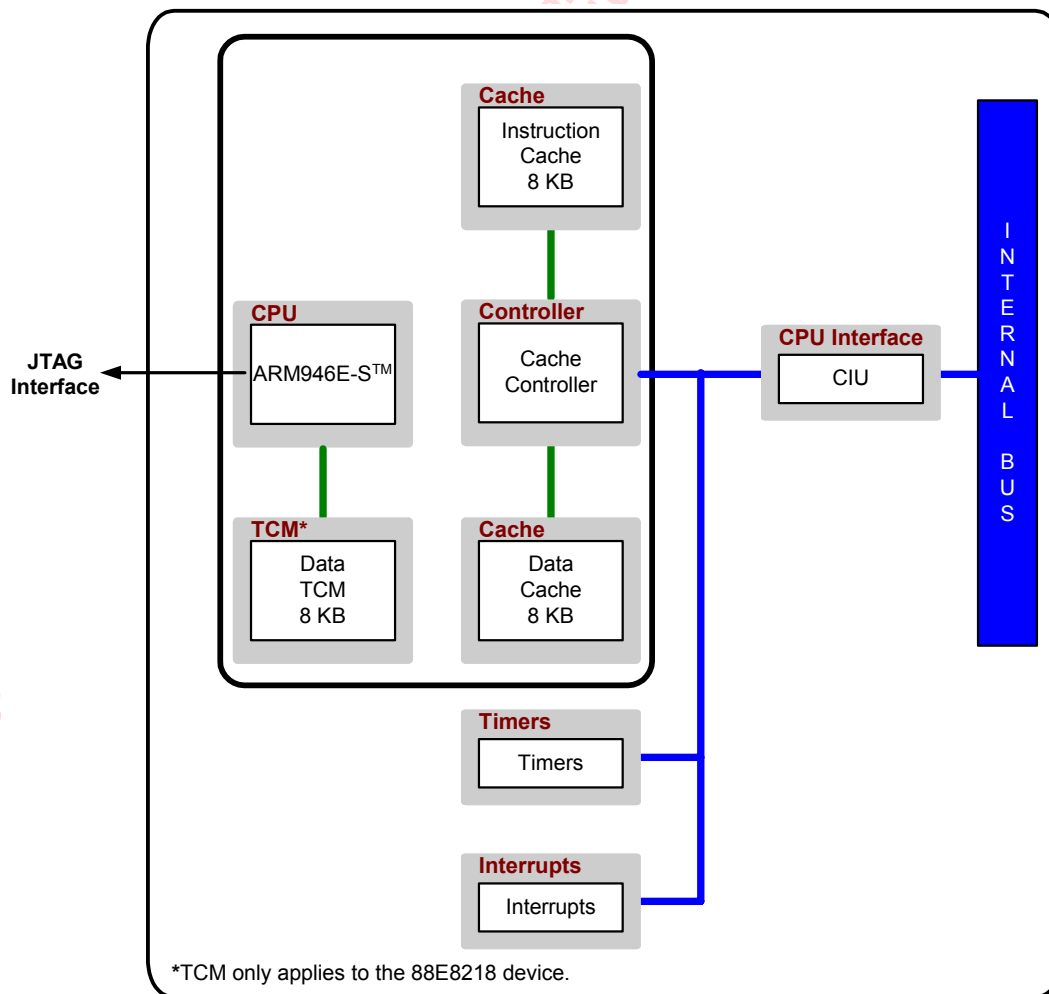
The 88E6218 and 88E6208 incorporate an embedded high performance ARM946E-S™ processor core, referred to as ARM in this document.

The CPU provides the following features:

- Core processor implementing the Instruction Set Architecture (ISA) Version ARMv5TE
- 8 KB Tightly Coupled Memory (TCM) for Data (88E6218 only)
 - Single cycle access
 - 32-bit bus
- 8 KB Data Cache Memory, 8 KB Instruction Cache Memory
 - Four-way set associative
 - Controllable write-back/write-through policy
 - 32-bit bus
- 32-bit ARM instruction set for maximum performance and flexibility
- 16-bit Thumb instruction set for increased code density
- AMBA - AHB (Advanced High performance Bus) for interfacing the other system components
 - Interrupts Unit
 - Timers Unit
 - CPU Interface Unit (CIU) to the 88E6218 CPU's Internal Bus
- Hardware extensions for advanced debug features accessible by the chip JTAG (IEEE standard 1149.1) interface

Figure 2 illustrates the 88E6218 CPU architecture.

Figure 2: 88E6218 CPU Sub-section Block Diagram



5.2 CPU Timers

Four timers are defined within the 88E6218 device, which are controlled by configuration registers (see [Section A.2.4 "CPU Timer Registers" on page 85](#)). Any of the timers can be an independent interrupt source to the ARM CPU.

Each timer is a 16-bit wide down counter with an enable signal derived from either a 1 KHz clock or 1 MHz clock, giving a resolution of 1 mS and 1 uS, respectively.

- Timer 1 is the slow timer with 1 KHz input.
- Timer 2, Timer 3, and Timer 4 are the fast timers with 1 MHz input.

Two modes of operation are available: periodic and one-shot.

- In periodic timer mode, the timer generates an interrupt at a constant interval.
- In one-shot mode, the timer only generates one interrupt when it reaches zero.

Timer interrupts are latched by the [Table 35, "Interrupt Request Status Register," on page 79](#) and the [Table 36, "Interrupt Source Status Register," on page 80](#), which can be read by the CPU. The interrupt signals also feed into the interrupt controller to produce FIQ and IRQ interrupts. See [Section 5.3 "Interrupts" on page 20](#) for an explanation of the FIQ (Fast Interrupt Request) and IRQ (Interrupt Request) to the ARM CPU.



Note

In addition to the CPU timers described here, also see [Section 12. "Watchdog Timer" on page 69](#).

5.2.1 Timer Operation

Timer operation is described in the following steps.

1. The Timer is loaded, by writing to the Timer x Length Register (see [Table 46 on page 85](#)), and then, enabled. The timer counts down to zero.
2. Upon reaching a zero, an interrupt is generated.
3. After reaching a zero count, if the timer is in Periodic mode, it reloads its initial value from the Timer x Length Register and continues to decrement; otherwise, it remains at zero.

5.2.2 Timer Programming Sequence

There is a separate Timer x Length Register (see [Table 46 on page 85](#)) for each of the four timers (1–4). The four Timer x Length Registers contain the initial value for each of the four timers. These registers are also used to reload the timers when in Periodic mode. When writing to the Timer x Length Registers, the upper 16 bits should be written as 0. When reading from the Timer x Length Registers, the upper 16 bits will be undefined.

The Timer Control Register (see [Table 47 on page 85](#)) controls the operation of the four timer counters as indicated by the format in [Figure 3](#).

Figure 3: Timer Control Register Configuration¹

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Function	Timer 4				Timer 3				Timer 2				Timer 1			
	LD	INT	A/D	AA	LD	INT	A/D	AA	LD	INT	A/D	AA	LD	INT	A/D	AA
Default	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Setting bits 15, 11, 7, and 3 of the Timer Control Register to 1 enables the counter to reload its value from the corresponding Timer x Length Register (see [Table 46 on page 85](#)).

Bits 14, 10, 6, and 2 of the Timer Control Register determine whether each timer is enabled or disabled. A value of 1 enables a timer interrupt to the CPU. A value of 0 disables a timer interrupt to the CPU.

Bits 13, 9, 5, and 1 of the Timer Control Register determine whether each timer is activated (starts) or deactivated (stops). A value of 1 starts a timer. A value of 0 stops a timer. A timer stops after reaching zero count and generat-

1. LD = Load, INT = Interrupt, A/D = Activate and Deactivate, AA = Automatic Reset.



ing an interrupt, if the Timer4AA bit is set to 0. A timer remains activated, reloads, and continues to decrement if the Timer4AA bit is set to 1.

Bits 12, 8, 4, and 0 of the Timer Control Register determine whether each timer automatically resets itself and continues to decrement after reaching zero count and generating an interrupt or stops a zero and waits for software to enable it. Each of these bits set bits in an Interrupt Enable Register (see [Table 38 on page 82](#)). When writing to these bits a '1' sets the corresponding bit in the Interrupt Enable Register. Writing a '0' has no effect on the Interrupt Enable Register.

There is a separate Timer x Value Register (see [Table 48 on page 87](#)) for each of the four timers (1–4). The four Timer x Value Registers contain the current value for each of the four timers. When reading from the Timer x Value Registers, the upper 16 bits will be undefined.

5.3 Interrupts

The interrupt controller provides a simple software interface to the interrupt system. In an ARM system, two levels of interrupt are available.

- Fast Interrupt Request (FIQ) for fast, low-latency interrupt handling
- Interrupt Request (IRQ) for more general interrupts

The interrupt registers appear in [Section A.2.3 "CPU Interrupt Registers" on page 79](#).

In an ARM system, only a single FIQ source would be in use at any particular time. This single source provides a true low-latency interrupt, because it ensures that the interrupt service routine can be executed directly without the need to determine the source of the interrupt. The FIQ is hardwired to the Timer 2 interrupt.

Separate interrupt controllers are used for FIQ and IRQ. The difference between them is that only a single bit position is defined for FIQ (which is intended for use by a single interrupt source), while 32 bits are available in the IRQ controller.

The IRQ interrupt controller uses a bit position for each different interrupt source (see [Table 4, "IRQ Interrupt Sources," on page 20](#)).

Table 4: IRQ Interrupt Sources

Interrupt	Bit	Interrupt Source
FIQ Interrupt	0	Hardwired to Timer 1 interrupt.
SW interrupt	1	Software programmed interrupt
Reserved	2	Reserved, must be masked
Reserved	3	Reserved, must be masked
Timer1 interrupt	4	Timer 1
Timer2 interrupt	5	Timer 2
Timer3 interrupt	6	Timer 3
Timer4 interrupt	7	Timer 4
Reserved	8	Reserved, must be masked
UniMAC™	9	UniMAC
Switch INTn interrupt	10	Switch

Table 4: IRQ Interrupt Sources (Continued)

Interrupt	Bit	Interrupt Source
UART interrupt	11	UART
GPIO interrupt	12	GPIO
TWSI interrupt	13	TWSI Interface Unit
Reserved	14	Reserved, must be masked
DMA Ch0 interrupt	15	DMA Unit Channel 0 NOTE: This interrupt only applies to the 88E6218. When using the 88E6208 device, this interrupt must be masked.
DMA Ch1 interrupt	16	DMA Unit Channel 1 NOTE: This interrupt only applies to the 88E6218. When using the 88E6208 device, this interrupt must be masked.
Reserved	17	Reserved, must be masked

All interrupt source inputs are active high and level sensitive.

No hardware priority scheme is provided, nor any form of interrupt vectoring, since these functions can be provided in software.

Bit 1 supports a software programmed interrupt, which allows the user to generate an interrupt under software control.

The following registers are provided for both FIQ and IRQ interrupt controllers:

- Interrupt Source Status Register (see [Table 36 on page 80](#))—This register indicates whether the appropriate interrupt source is active prior to masking. The interrupt source shall be active high. Therefore, a logic high in the source status register indicates that the interrupt source is active.
- Interrupt Request Status Register (see [Table 35 on page 79](#))—This register indicates whether the interrupt source will generate an interrupt request to the processor, meaning the interrupt source is high and the interrupt is not masked.
- Interrupt Enable Register (see [Table 38 on page 82](#))—This register is used to determine whether an active interrupt source should generate an interrupt request to the processor. The Interrupt Enable Register has a dual mechanism for setting and clearing the enable bits. This mechanism allows enable bits to be set or cleared independently with no knowledge of the other bits in the Interrupt Enable Register. The mechanism consists of an independent set register and an independent clear register (see [Table 37, "Interrupt Enable Set Register," on page 81](#) and [Table 39, "Interrupt Enable Clear Register," on page 83](#)).

When writing to the enable set location, each data bit that is high will set the corresponding bit in the Interrupt Enable Register and all other bits of the enable register will be unaffected. The enable clear location is conversely used to clear bits in the Interrupt Enable Register while leaving other bits unaffected.



Section 6. Memory Controller

6.1 Functional Description

The 88E6218 and 88E6208 memory controllers are designed to create a bridge between the internal bus and the external memory devices. It can be configured to support two types of memory devices — SDRAM and asynchronous chip select devices like Flash or ROM. Several devices of both types can be supported simultaneously.

The bus width for each memory device bank¹ can be selected by register settings inside the memory controller. The timing parameters for each memory device bank are also programmable, so that different speeds of asynchronous memory device banks can be supported. The main features of the memory controller include:

- Support for up to 4 memory device banks, one boot Flash/ROM and 3 others in any combination of SDRAM or asynchronous device banks in the 88E6218, and support for 3 memory device banks, one boot Flash/ROM and two others in the 88E6208.
- Each of the external device banks can be selectively programmed for configuration and timing parameters.
- Each SDRAM device bank can support up to 64 MB space in the 88E6218 and 32 MB space in the 88E6208.
- Each Flash/ROM device bank can support up to 64 MB in the 88E6218 and 8 MB in the 88E6208.
- Supports 16 Mbit, 64 Mbit, and 128 Mbit SDRAM sizes, with widths of 8, 16, or 32 bits in the 88E6218 device and widths of 8 or 16 in the 88E6208 device.
- For SDRAM,
 - the 88E6218 can support X16 and X32 bit SDRAM device banks.
 - the 88E6208 can support X16 bit SDRAM device banks.
- For asynchronous chip selected device banks,
 - the 88E6218 can support X8, X16, and X32 bit banks.
 - the 88E6208 can support X8, and X16 bit banks.
- SDRAM supports page open mode to improve performance.
- Supports burst transfer with linear order (no wrap around) and wrap around mode to SDRAM.
- Supports up to 32-byte burst in 16- and 32-bit width SDRAM configurations (in the 88E6218 device only).

6.2 Address Space

The address window for each memory device bank is determined by an internal Base Address Register and Base Address Mask Register pair². The Base Address Register defines the start address for the memory device bank, while the Base Address Mask Register defines the maximum size of the memory that can be addressed.

The Base Address Mask Register is ANDed with the internal hardware address Addr bits 29:22 and compared to the Base Address Register for the corresponding memory transaction to select the correct memory Chip Select. The internal hardware address is generated by the CPU or any DMA access.

This feature means the maximum memory size can go from 4 MB to 64 MB depending on mask bits setting. The Base Address Register and Base Address Mask Register are both 8 bits wide.

1. A memory device bank refers to a group of devices sharing a single 88E6218 device chip select.
2. The 88E6218 provides a pair of Base Address and Base Address Mask registers for each chip select.

The address distribution from MSB to LSB is Bank Address, Row Address, and Column Address in order (see Table 5, "X32 Data Bus Configuration (88E6218 Only)," on page 23 and Table 6, "X16 Data Bus Configuration," on page 24).

6.3 SDRAM

The memory controller supports 16 Mbit, 64 Mbit, and 128 Mbit SDRAM devices, with SDRAM device widths of 8, 16, and 32 bits in the 88E6218, and 8 and 16 bits in the 88E6208.

6.3.1 X32 Data Bus (for the 88E6218 Only)

The SDRAM controller can support SDRAM implementations in which the data bus width between the 88E6218 and the SDRAM is 32 bits as shown in Table 5, "X32 Data Bus Configuration (88E6218 Only)," on page 23. In these implementations, the number of SDRAM chips used is a variable dependent upon the bit width of the particular SDRAM device. For example, if 8-bit SDRAM devices are used, four of them are placed in parallel to achieve a 32-bit data width.

The SDRAM controller can also support implementations in which the data width between the 88E6218 and the SDRAM device bank is less than 32 bits as shown in Table 6, "X16 Data Bus Configuration," on page 24. In these implementations, multiple SDRAM accesses are needed per internal bus access to translate between the internal 32-bit bus width and the SDRAM data bus width.

The Table 5 shows the detail configuration for building a 32-bit data bus using different size and bit width of SDRAM device. Also detailed in this table are the Bank, Row and Column Addresses as parts of the address bus of the 88E6218.

Table 5: X32 Data Bus Configuration (88E6218 Only)

SDRAM Type	Total Chips	SDRAM Width	Total Size	Row Address	Column Address	Bank Address
16 Mbits	4	X8	8 MB	A0~A10	A0~A8	1
	2	X16	4 MB	A0~A10	A0~A7	1
	1	X32	2 MB	A0~A10	A0~A6	1
64 Mbits	4	X8	32 MB	A0~A11	A0~A8	2
	2	X16	16 MB	A0~A11	A0~A7	2
	1	X32	8 MB	A0~A10	A0~A7	2

Table 5: X32 Data Bus Configuration (88E6218 Only) (Continued)

SDRAM Type	Total Chips	SDRAM Width	Total Size	Row Address	Column Address	Bank Address
128 Mbits	4	X8	64 MB	A0~A11	A0~A9	2
	2	X16	32 MB	A0~A11	A0~A8	2
	1	X32	16 MB	A0~A11	A0~A7	2

6.3.2 X16 Data Bus

The SDRAM controller also supports X8, and X16 of SDRAM device banks for a total 16-bit 88E6218/88E6208 SDRAM data width implementation. The SDRAM controller performs multiple access to SDRAM to assemble 32 bits of data bus. The following table shows the detail configuration for a 16-bit data bus using different size and bit width of SDRAM device.

Table 6: X16 Data Bus Configuration

SDRAM Type	Total Chips	SDRAM Width	Total Size	Row Address	Column Address	Bank Address
16 Mbits	2	X8	4 MB	A0~A10	A0~A8	1
	1	X16	2 MB	A0~A10	A0~A7	1
64 Mbits	2	X8	16 MB	A0~A11	A0~A8	2
	1	X16	8 MB	A0~A11	A0~A7	2
128 Mbits	2	X8	32 MB	A0~A11	A0~A9	2
	1	X16	16 MB	A0~A11	A0~A8	2

6.3.3 Page Open Mode for SDRAM Device Banks

The memory controller provides an option for Page Open mode by programming KeepRwOpen bit 12 in the Configuration Register 0.

When KeepRwOpen is cleared to 0, the row is closed after every read/write operation. When KeepRwOpen is set to 1, the row is kept open for each read/write operation until a new row is accessed. In this configuration, subsequent accesses to same row will have a lower latency than when the bit is not set, but an access to a different row will have an increased latency. This feature is available for SDRAM device banks only.

6.4 Flash and ROM Device Banks

The memory controller supports asynchronous chip select device banks like Flash and ROM. It can support device banks with a bus width of X8, X16, and X32 in the 88E6218 and with a bus width of X8 and X16 in the 88E6208. The supported device bank configuration is programmable. No byte write enable signal is supported for these memory device banks.

6.4.1 Burst Access for Asynchronous Device Bank

For asynchronous memory device banks like Flash or ROM, the memory controller performs multiple single accesses to mimic the burst access on the internal bus.

6.4.2 Packing

There are two modes of access to the device banks:

- Byte packing
- No byte packing

All accesses follows the Little Endian rule.

6.4.2.1 Byte Packing

In the case of 32 bits of internal bus access, when using the X8 device bank, the memory controller performs four accesses to the X8 device bank to assemble a 32 bits of data. When using a X16 device bank, the memory controller performs two accesses to the X16 device bank to assemble the 32 bits of data. If the bus width for byte enable signal on the internal bus side is less than the bus width of the device bank type, the memory controller will write the whole byte or word whose bus width match the device bank bus width to the device bank. [Table 7, "Asynchronous Device Bank Access with Byte Packing," on page 25](#) shows some examples for device bank access using the packing option.

Table 7: Asynchronous Device Bank Access with Byte Packing

Access Start Byte Address	Internal Bus Byte En[3:0]	Access Type	Device Bank Bus Width	Action to Device Bank
0X000A	4'b1111	Read	X8	4 reads 0X000A - 0X000D Data[31:0]
0X000A	4'b0000	Write	X8	4 writes 0X000A - 0X000D Data[31:0]
0X000A	4'b1111	Read	X16	2 reads 0X000A - 0X000D Data[31:0]
0X000A	4'b0000	Write	X16	2 writes 0X000A - 0X000D Data[31:0]

Table 7: Asynchronous Device Bank Access with Byte Packing (Continued)

Access Start Byte Address	Internal Bus Byte En[3:0]	Access Type	Device Bank Bus Width	Action to Device Bank
0X000A	4'b1101	Write	X8	1 write 0X000B Data[15:8]
0X000A	4'b1100	Write	X8	2 writes 0X000A - 0X000B Data[15:0]
0X000A	4'b1001	Write	X8	2 writes 0X000B - 0X000C Data[23:8]
0X000A	4'b0001	Write	X8	3 writes 0X000B - 0X000D Data[31:8]
0X000A	4'b1100	Write	X16	1 write 0X000A - 0X000B Data[15:0]
0X000A	4'b0011	Write	X16	1 write 0X000C - 0X000D Data[31:16]
0X000A	4'b0110 (user error)	Write	X16	1 write 0X000A - 0X000B Data[31:24], Data[7:0]
0X000A	4'b0111 (user error)	Write	X16	1 write 0X000C - 0X000D Data[31:16]
0X000A	4'b0001 (user error)	Write	X16	2 write 0X000A - 0X000D Data[31:0]

6.4.2.2 No Byte Packing

When an asynchronous device bank is accessed by the memory without using the packing option, the memory controller will not perform multiple accesses to the device bank to pack 32 bits. Only one access will be performed according to the bus width of the device bank, and access priority will be given to the lower bytes first. [Table 8](#) shows some examples for device bank access without using the packing option.

Table 8: Asynchronous Device Bank Access Without Byte Packing

Access Start Byte Address	Internal Bus Byte En[3:0]	Access Type	Device Bank Bus Width	Action to Device Bank
0X000A	4'b1111	Read	X8	1 read 0X000A Data[7:0]
0X000A	4'b0000	Write	X8	1 write 0X000A Data[7:0]
0X000A	4'b1111	Read	X16	1 read 0X000A - 0X000B Data[15:0]
0X000A	4'b0000	Write	X16	1 write 0X000A - 0X000B Data[15:0]
0X000A	4'b1101	Write	X8	1 write 0X000B Data[15:8]
0X000A	4'b1100	Write	X8	1 write 0X000A Data[7:0]
0X000A	4'b1001	Write	X8	1 write 0X000B Data[15:8]
0X000A	4'b0001	Write	X8	1 write 0X000B Data[15:8]
0X000A	4'b1100	Write	X16	1 write 0X000A - 0X000B Data[15:0]
0X000A	4'b0011	Write	X16	1 write 0X000C - 0X000D Data[31:16]
0X000A	4'b0110 (user error)	Write	X16	1 write 0X000A - 0X000B Data[31:24], Data[7:0]
0X000A	4'b0111 (user error)	Write	X16	1 write 0X000C - 0X000D Data[15:0]
0X000A	4'b0001 (user error)	Write	X16	1 write 0X000A - 0X000B Data[23:8]



6.5 Boot Device Bank

The memory controller has a dedicated chip select pin, BOOTCSn, for the boot device bank selection. The boot device bank must be Flash/ROM.

6.5.1 Default Configuration During Power-Up

Configuration and timing parameters for boot device banks are fixed according to the default configuration shown in Table 9.



Note

For correct operation, make sure the configuration and timing specifications for boot device banks meet the default configuration

Table 9: Default Configuration

Parameter	Default Value
Boot Device Bank Base Address	32'hFFC0_0000
Boot Device Bank Size	4 MB
External Boot Flash	Configurable to either X8 or X16 bit device. Default is X8. X16 configuration is possible through reset strapping (See Reset Initialization in Part 1 of the three part 88E6208/88E6218 datasheet set, Document Number MV-S101300-01). Read access delay tAA = 28 * SystemClock Output hold time from CEn tOD = 2 * SystemClock

Section 7. UniMAC™ Unified Fast Ethernet Media Access Control

The Marvell Unified Media Access Controller (UniMAC™) architecture provides an efficient networking interface between the CPU and the 88E6208/88E6218 multi-port Fast Ethernet switch fabric. The UniMAC architecture enhances networking performance in the Link Street SOHO gateway/router application.

The UniMAC is a Fast Ethernet MAC compatible with the IEEE 802.3 standard and contains a superset of features including extra speed and functionality. The UniMAC can operate up to double the standard clock rate in the 88E6218, so the UniMAC runs up to 200 Mbps full-duplex (that is, 200 megabits per second in both transmit and receive directions, simultaneously). In the 88E6208, the UniMAC runs up to 100 Mbps full-duplex (that is, 100 megabits per second in both transmit and receive directions, simultaneously). In addition, the UniMAC manages multiple receive queues and multiple transmit queues, which provides a higher level of routing performance by allowing packet pre-queuing based upon its source and/or destination. The UniMAC also accelerates Internet Protocol routing by aligning the incoming Internet Protocol data packet to a 32-bit boundary.

7.1 Functional Description

The UniMAC handles all functionality associated with moving packet data between host memory and Port 5 of the Switch. The UniMAC provides one port capable of running at either 10, 100, or 200 Mbps for the 88E6218 device and at either 10 or 100 Mbps for the 88E6208 device. See [Figure 5](#) and [Figure 6](#).

Figure 4: 88E6218 UniMAC Unit Simplified Block Diagram

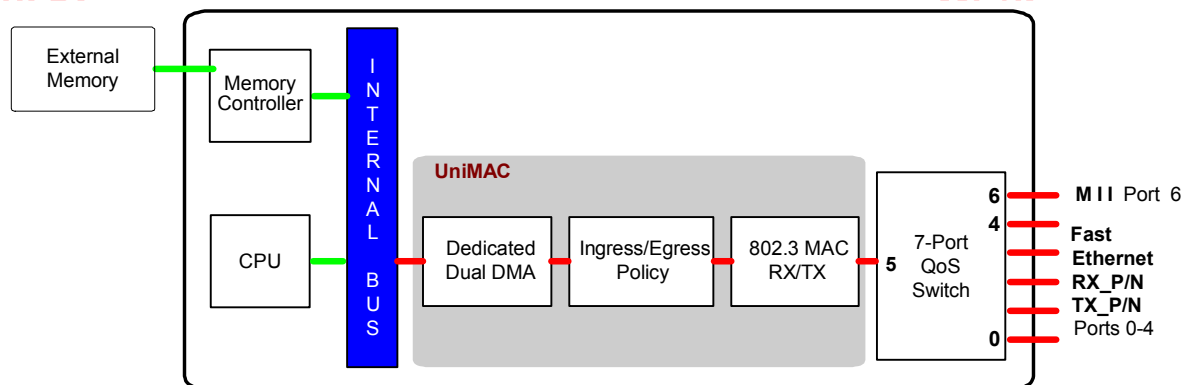
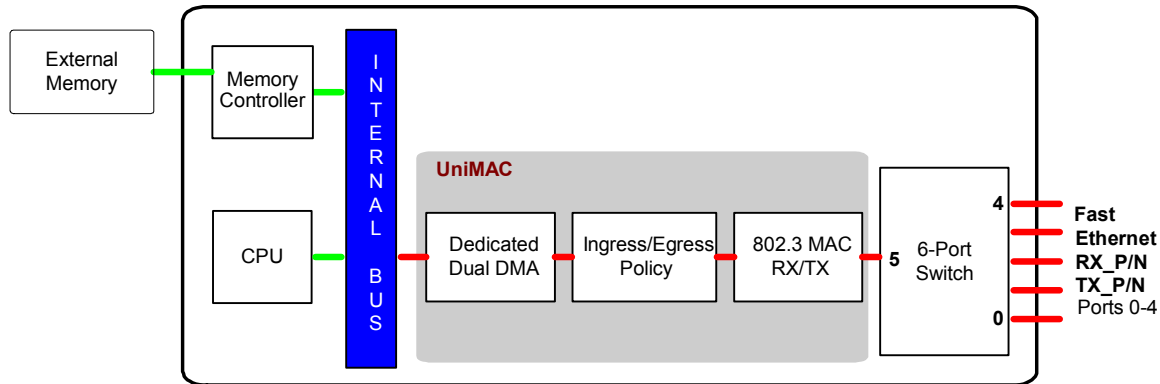


Figure 5: 88E6208 UniMAC Unit Simplified Block Diagram



7.2 Features

802.3 Compliant Fast Ethernet MAC

- Integrated Media Access Controller (MAC) fully compliant with IEEE 802.3, 802.3u, 802.x standards
- IEEE 802.3x flow-control to/from Switch (88E6218 only)
- Receive functions
 - In both the 88E6218 and 88E6208 devices:*
 - Promiscuous mode
 - Only in the 88E6218 device:*
 - 1/2k or 8k address filtering capability
 - Perfect filtering
 - Reverse filtering
 - Broadcast reject mode
 - Proprietary hash function for address table management
 - Supports Multicast and Unicast address entries
 - On-the-fly IGMP packet trapping (layer 3 analysis in hardware)¹
 - CRC checking
- Transmit functions:
 - In both the 88E6218 and 88E6208 devices:*
 - Short frame (less than 64 bytes) zero padding¹
 - CRC generation (programmable per packet)
- Wire-speed operation at 200 Mbps full-duplex in the 88E6218 and at 100 Mbps full-duplex in the 88E6208

¹ This feature only functions when the Marvell® Header mode is disabled.

Dedicated DMA

- Supports Marvell Header¹
- Two DMA engines, one each for receive and transmit operations
- Multiple queues
 - Four receive priority queues for the 88E6218 and two for the 88E6208
 - Two transmit priority queues for the 88E6218 and one for the 88E6208
- Scatter and gather operation
- Error reporting (per packet)

7.3 Media Access Controller (MAC)

The MAC logic performs all of the functions of the 802.3 protocol such as frame formatting, frame stripping, etc. It also ensures that any outgoing packet complies with the 802.3 specification in terms of preamble structure — 56 preamble bits are transmitted before Start of Frame Delimiter (SFD).

The MAC operates in full duplex mode only, which enables it to transmit and receive frames simultaneously.



Note

Half Duplex operation is not supported by the 88E6208 and 88E6218 UniMAC.

7.4 802.3x Flow Control (Non Marvell Header Mode) — 88E6218 Only

When Marvell® Header mode is disabled the UniMAC supports Flow Control.

IEEE 802.3x flow control is enabled by setting the Port_Configuration_Extend <FCTL> bit 12 (see [Table 60 on page 99](#)).

it is operating in full-duplex and if When the port receives a PAUSE packet, it does not transmit a new packet for a period of time specified in this PAUSE packet.

A received packet is recognized as flow control PAUSE, if it was received without errors and is either of the following:

- DA = 01-80-C2-00-00-01 and type=88-08 and MAC_Control_Opcode=01
- DA = (The port address) and type=88-08 and MAC_Control_Opcode=01. The 48-bit port address is in the registers Source_Address_Low, Source_Address_High. This address is also used as source address for PAUSE packets that the port generates (to DA=01-80-C2-00-00-01)

PAUSE packets are sent by the port when instructed to do so by the CPU. This is done by setting Port_Command<FC> bit.

¹ The Marvell Header is 2 bytes long and is prepended to the packets by the switch core.

7.5 Zero Padding for Short Packets (Non Marvell Header Mode)

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

The UniMAC port offers a per packet padding request bit in the TX descriptor. This causes the port logic to enlarge packets shorter than 64 bytes by appending zero bytes. When this feature is used, only packets equal or larger than 64 bytes are transmitted as is. Packets smaller than 64 bytes are zero padded and transmitted as 64 byte packets.



Note

When Marvell Header mode is enabled, the software is responsible for padding short packets and making sure that the packet length is a minimum of 66 bytes (including the header and the CRC).

7.6 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 TX descriptor is used for specifying if CRC generation is required for a specific packet.



Note

When Marvell Header mode is enabled in the Switch, the Switch will strip the Marvell Header and then recalculate the new CRC before forwarding the frame.

7.7 Marvell Header Mode

Switch Port 5 can be set to prepend 2 bytes of data in front of each frame sent to the CPU and to remove these 2 bytes of data from frames coming back from the CPU. This Marvell Header Mode is optional, but it can increase CPU routing performance greatly by aligning the IP data portion of the frames in the CPU memory on 32-bit boundaries. On ingress to the CPU, the Marvell Header contains information that the CPU needs, like the switch's source port of the frame, its priority (88E6218 only), etc. On egress from the CPU, the CPU uses the Header to indicate the port based VLAN information of the frame so that frames sent to the LAN ports only go to the LAN ports and not out the WAN port and vice versa.

The Marvell Header supports the following features:

- Higher routing performance due to 32-bit IP data alignment
- The WAN port can be any port on the switch
- More than one WAN port can be used, either as a hot backup or with link aggregation
- Any switch port can be a DMZ port (or any type of port that is not WAN or LAN)

Refer to Part 3 of the three part 88E6208/88E6218 datasheet set (Document Number MV-S101300-03) for more information about the format of the Marvell Header when routing to and from the CPU.

The UniMAC can independently be set to process the Marvell Header or ignore it. If Switch Port 5 does not generate/use the Marvell Header, then the UniMAC must not try to process it. If Switch Port 5 generates/uses Marvell Headers, the UniMAC can process or ignore the Header.

When the Header mode is enabled on Switch Port 5:

- Two extra bytes are added to the beginning of the frame, just before the received frame's DA, and a new CRC is calculated for the frame by the switch before the frame is forwarded to the UniMAC.
- The Header is used for priority and queue steering (if the UniMAC Header mode is enabled). In the 88E6218, when both the Switch's and the UniMAC Header mode are enabled, DA filtering is still supported by the UniMAC.



Notes

- If the Switch Port 5 is set to Marvell Header mode and the UniMAC is not set for Header mode, then the UniMAC must be in Promiscuous mode.
- There is only one HeaderMode bit per switch port. When enabled, it effects both the Ingress and Egress of this switch port. Prepending the header for UniMAC transmit packets must be done by software.

The Marvell Header contains the following fields that are updated by the Switch and examined by the UniMAC to determine the frame receive priority/target queue:

- DBNum[3:0] - Database Number.
- PRI[2:0] - Frame priority (applicable to the 88E6218 only).
- SPID[2:0] - Source Port ID.

7.8 Address Filtering (88E6218 Only)

The UniMAC implements DA filtering by searching each DA in an address data base. The search utilizes Hashing algorithm and Hash table mechanism.

This section describes the Hash algorithm and Hash table data structure. The CPU must build this table for the 88E6218 before enabling the UniMAC port.

7.8.1 Hash Table Structure

The 88E6218 Hash table is a data structure prepared by the CPU and resides in the system memory. Its location is identified by a 32-bit pointer stored in the Hash Table Pointer register (HTPR) (see [Table 63 on page 103](#)). The Hash table must be octet-byte aligned. The lowest three bits of the Hash Table Pointer register (HTPR) are hard-wired to '0'.

There are two possible sizes for the Hash table. Table size is defined by the HS bit in the Port Configuration Register (PCR) (see [Table 59 on page 98](#)).

- 8K address table. 256 KB of memory required (4 x 64-KB banks)
- 1/2K address table. 16 KB of memory required (4 x 4-KB banks)

A multiple of four banks are used to reduce the number of addresses that are mapped to the same table entry.

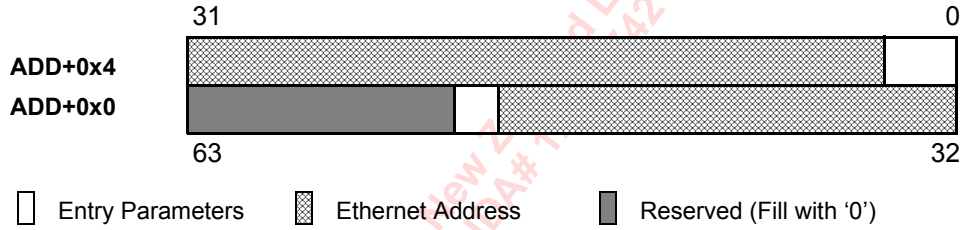


Note

The user must initialize the Hash table before enabling the UniMAC™.

Each Address entry is a two-word data field (64 bits) as shown below.

Figure 6: UniMAC Hash Table Entry



The following table describes the Hash table entry fields.

Table 10: Hash Table Entry Fields

Bit	Command	Usage
63:53	Reserved	Fill With '0'
52:51	Priority	The priority queue uses this Ethernet address, in case there are no other priority signals (i.e., ToS or VLAN). NOTE: The Priority command is applicable only when the Marvell® Header mode is inactive.
50:47	Ethernet Address[47:44]	Mapped to Ethernet address[7:4].
46:43	Ethernet Address[43:40]	Mapped to Ethernet address[3:0].
42:39	Ethernet Address[39:36]	Mapped to Ethernet address[15:12].
38:35	Ethernet Address[35:32]	Mapped to Ethernet address[11:8].
34:31	Ethernet Address[31:28]	Mapped to Ethernet address[23:20].
30:27	Ethernet Address[27:24]	Mapped to Ethernet address[19:16].
26:23	Ethernet Address[23:20]	Mapped to Ethernet address[31:28].
22:19	Ethernet Address[19:16]	Mapped to Ethernet address[27:24].
18:15	Ethernet Address[15:12]	Mapped to Ethernet address[39:36].
14:11	Ethernet Address[11:8]	Mapped to Ethernet address[35:32].
11:7	Ethernet Address[7:4]	Mapped to Ethernet address[47:44].
6:3	Ethernet Address[3:0]	Mapped to Ethernet address[43:40].
2	Receive/Discard (RD)	0 - Discard packet upon match 1 - Receive packet upon match NOTE: The Marvell Header overrides the Receive/Discard command.
1	Skip	Skip empty entry in a chain
0	Valid	Indicates Valid Entry

7.8.2 Hash Modes

There are two Hash functions in the 88E6218; Hash Mode 1 and Hash Mode 0 that selected by Port Configuration Register (PCR) (see [Table 59 on page 98](#)).

Hash Mode 0

In Hash mode 0, the Hash entry address is calculated in the following manner:

$\text{hashResult}[14:0] = \text{hashFunc0}(\text{ethernetADD}[47:0])$

- hashResult is the 15-bit Hash entry address.
- ethernetADD is a 48-bit number, which is derived from the Ethernet Destination Address, by nibble swapping in every byte; i.e., Ethernet address of 0x123456789ABC translates to ethernetADD of 0x21436587A9CB.
- inverse every nibble; i.e. ethernetADD of 0x21436587A9CB translates to 0x482C6A1E593D

hashFunc0 calculates the hashResult in the following manner:

- $\text{hashResult}[14:9] = \text{ethernetADD}[7:2]$
- $\text{hashResult}[8:0] = \text{ethernetADD}[14:8, 1, 0] \text{ XOR } \text{ethernetADD}[23:15] \text{ XOR } \text{ethernetADD}[32:24]$

Hash Mode 1

In Hash mode 1, the Hash entry address is calculated in the following manner:

$\text{hashResult}[14:0] = \text{hashFunc1}(\text{ethernetADD}[47:0])$

- hashResult is the 15-bit Hash entry address.
- ethernetADD is a 48-bit number, which is derived from the Ethernet Destination Address, by nibble swapping in every byte (i.e Ethernet address of 0x123456789ABC translates to ethernetADD of 0x21436587A9CB).
- bit swap within every nibble; i.e. ethernetADD of 0x21436587A9CB translates to 0x482c6A1E593D

hashFunc1 calculates the hashResult in the following manner:

- $\text{hashResult}[14:9] = \text{ethernetADD}[0:5]$
- $\text{hashResult}[8:0] = \text{ethernetADD}[6:14] \text{ XOR } \text{ethernetADD}[15:23] \text{ XOR } \text{ethernetADD}[24:32]$

7.8.3 Hash Entry

For each Ethernet address, the Hash table entry address is the lower 15 bits of the hashResult for the 8K address table, or the lower 11 bits for the 0.5K address table. The entry is an offset from the address base and is octet-byte aligned. The address entry is therefore:

- 8K Address Table: $\text{tblEntryAdd} = \text{HTP field in the HTPR register} + \{\text{hashResult}[14:0], 000\}$
- 1/2K Address Table: $\text{tblEntryAdd} = \text{HTP field in the HTPR register} + \{\text{hashResult}[10:0], 000\}$

7.8.4 Table Filling

When preparing the Hash table data structure, the CPU must first (typically at boot time) initialize the Hash table memory to '0'.

The table filling algorithm is described below.

1. Calculate tblEntryAdd according to mode of operation (Hash Mode 1 or Hash Mode 0).
2. Check that tblEntry is empty (Valid bit is "0").
3. If the tblEntry is empty, write the hashEntry (Valid, Skip and RD bits and Ethernet address).
4. If the tblEntry is occupied (i.e. Valid bit is 1 and Skip bit is 0), move to tblEntry+1.
5. If less than hopNumber of attempts, Repeat to Step 3.

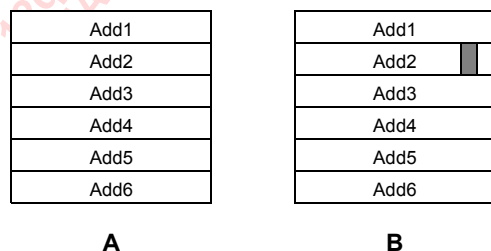
The hopNumber should be selected and initialized before entering this routine. The Hash table hopNumber (Number of Hops) is 12. After 12 attempts to identify an address, the 88E6218 passes the address to the CPU and sets the HE (Hash Expired) bit in the descriptor status field. Therefore, the hopNumber is the number of times the CPU will attempt to write a new Ethernet address into the Hash table.

If after the hopNumber of attempts the CPU has been unable to locate a free table entry, the CPU can:


- Defragment the table.
- Create a new Hash table using the alternate Hash Mode, which may redistribute the addresses more evenly in the table.

In cases where more than one address is mapped to the same table entry, an address chain is created. In this case, when the CPU needs to erase an address that is part of an address chain, it cannot clear its Valid bit since this would cut the chain. Instead, the CPU should set the Skip bit to '1'. This is shown in [Figure 7](#).

Figure 7: Address Chain



In case A where Add1-6 has the same Hash function, and thus start with the same tblEntry, the CPU allocates them in the table by increasing tblEntry by one entry each time. Add1 is the first address to be written into the table and Add6 is the last.

When the CPU is required to remove Add2 from the table, it cannot clear its valid bit since that would break the chain from Add1 to Add3. Instead, it sets Add2's Skip bit to '1' (denoted as ). It is also recommended that the CPU defragments the table from time to time.

7.8.5 Address Filtering Process (88E6218 Only)

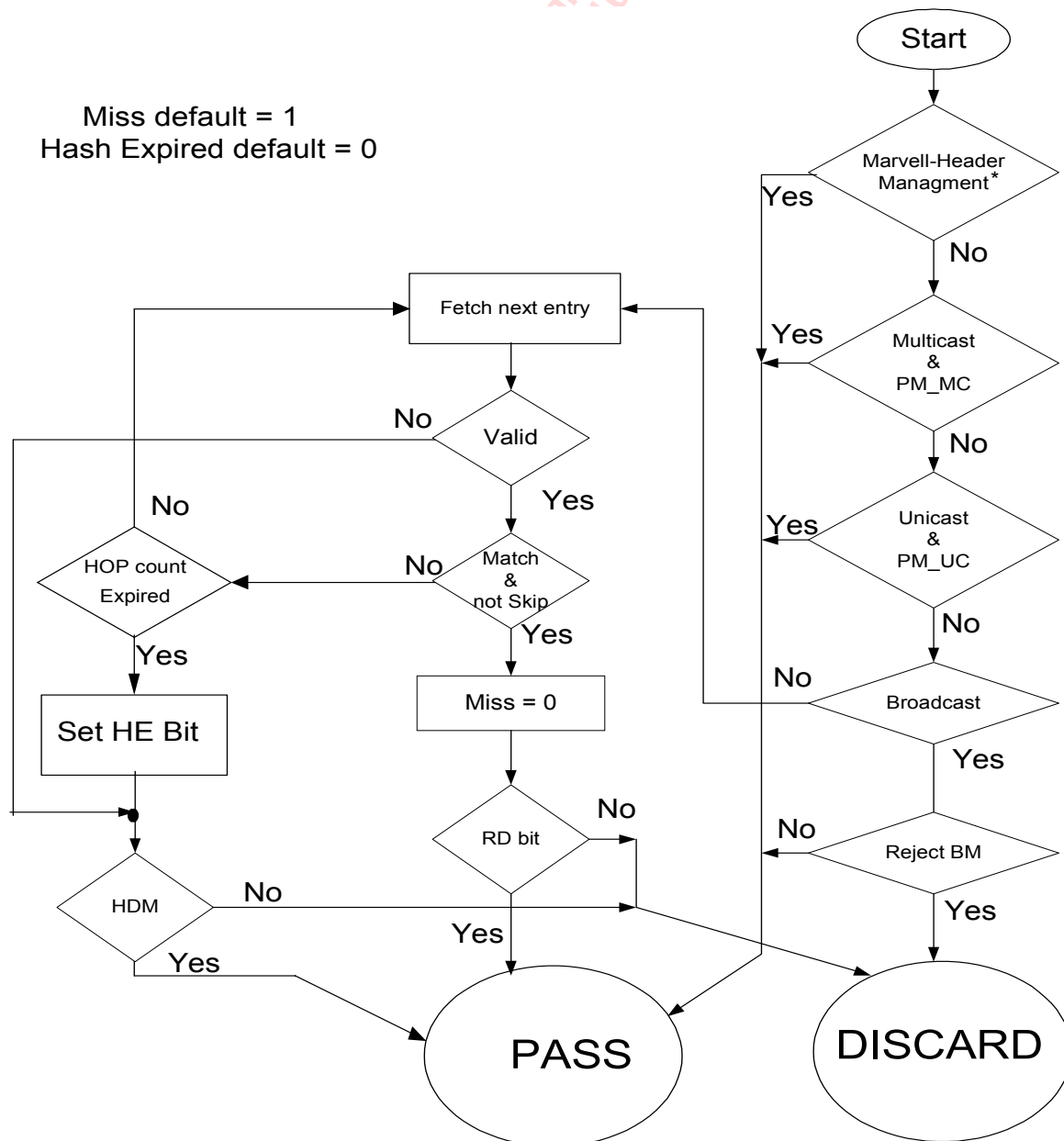
The following terms are used when referring to the address filtering process.

Table 11: Terms Used in Address Filtering

Term	Definition
BPDU	The Frame is Bridge Protocol Data Unit (BPDU). To enable capture, see the BPDU bit 1 in the Port Configuration Extend Register (PCXR) (Table 60 on page 99). NOTE: BPDU trapping can only be used in Non Marvell Header mode.
Broadcast	The Frame DA is a broadcast address.
DA	Ethernet Destination Address
HDM	See the Port Configuration register's HDM bit 14.
HE	Hash expired indication copied to receive descriptor HE bit 13.
HOP Expired	The number of Hash table entries fetched exceed 12.
IGMP	The Frame is Internet Group Management Protocol (IGMP) packet. To enable capture, see the IGMP bit [0] in the Port Configuration Extend Register (PCXR). NOTE: IGMP trapping can only be used in Non Marvell Header mode.
Management	Marvell Header management bit field. Received frames marked by the Switch as Management are always passed to the CPU without Hash lookup. Management is applicable to header mode only.
Match	The frame DA equals the Hash table entry DA.
Miss	Address is not found in the table. Indication copied to receive descriptor M bit 12.
PM	Promiscuous Mode: When enabled, all packets are passed to the CPU. The 88E6218 still executes the Hash process reporting to the CPU, regardless whether the address is in the Hash table or not. Determined by the configuration in the Port Configuration register's PM_UC bit 0.
PM_MC	Promiscuous Multicast: When enabled, all Multicast packets are passed to the CPU. No Hash address lookup is performed. Determined by the configuration in the Port Configuration register's PM_MC bit.
PM_UC	Promiscuous Unicast: When enabled, all Unicast packets are passed to the CPU. No Hash address lookup is performed. Determined by the configuration in the Port Configuration register's PM_UC bit.
RD Bit	Receive/Discard frame bit from the hash table entry.
Reject BM	Reject Broadcast: Determined by the configuration in the Port Configuration register's RBM bit 1.
Valid/Skip	Valid hash table entry has its valid bit set or the temporary removed hash table entry has its skip bit set.

The 88E6218 address filtering process is illustrated by Figure 8. See Table 11 for the definition of the terms used in this figure.

Figure 8: Address Filtering Process



*Managment frame indicated by the managment bit in the Marvell Header. Applicable in Marvell Header mode only.

The 88E6218 uses the HE (Hash Expired) and M (Miss/Match) bits in the descriptor for reporting the packet filtering status. Table 12 describes the various reports and summarizes their meaning.

Table 12: Packet Filtering Status

HE	M	Condition
0	0	Hash Table No Hit The address was not found in the Hash table, but Promiscuous Mode is enabled.
0	1	Hash Table Hit Either an address found in the Hash table and RD bit set OR an address that was not found in the Hash table, in case that HDM bit is set.
1	0	Hash Table Expired The hopNumber expired before the address was found in the Hash table.
1	1	UnUsed

7.9 Rx Queuing

The 88E6218 supports four receive priority queues. The 88E6208 supports two receive queues that are *not* priority based but are based on the Database Number and SPID. For additional details about queuing, see:

- [Section 7.9.1 "88E6208 Rx Queuing" on page 39](#)
- [Section 7.9.2 "88E6218 Rx Priority Queuing — Marvell Header Mode Enabled" on page 40](#)
- [Section 7.9.3 "88E6218 Rx Priority Queuing — Marvell Header Mode Disabled" on page 40](#)

The following terms are used when referring to the receive queues:

- SPID_Setting — Configuration in the Port Configuration Extend Register (PCXR) (see [Table 60 on page 99](#)).
- DBNum[3:0] — Marvell Header Database Number as marked by the Switch.
- PRI[2:0] — Marvell Header Frame priority as marked by the Switch.
- SPID[2:0] — Marvell Header Source Port ID as marked by the Switch.

7.9.1 88E6208 Rx Queuing

The 88E6208 does not support priority queuing. However, it does support queuing based on the Marvell Header mode setting. Refer to [Table 13](#).

Table 13: Receive Logic — 88E6208 only

Mode	Rx Queue ¹
00	Marvell Header mode in the UniMAC is disabled and only one queue is used. The default queue is 3.
01	Marvell Header mode in the UniMAC is enabled. Rx Queue = 3.
10	Marvell Header mode is enabled in the UniMAC. Rx queue set according to {DBNum[0], 1}.
11	Marvell Header mode is enabled in the CPU. Rx queue set according to {SPID[3:0]==SPID_Setting, 1}.

¹ DBNum, SPID and SPID_Setting are defined in [Section 7.9 "Rx Queuing" on page 39](#).

7.9.2 88E6218 Rx Priority Queuing — Marvell Header Mode Enabled

The received frame priority is determined by the Switch Core and conveyed in the Marvell Header.

For a detailed definition of Marvell Header refer to Part 3 of the three part 88E6208/88E6218 datasheet set (Document Number MV-S101300-03).

The receive queue priority logic is shown in [Table 14](#).

Table 14: Receive Priority Logic — 88E6218 only

NOTE: Priority Mode — Configured by the DA_PREFIX field in the Port Configuration Extend Register (PCXR) (see [Table 60 on page 99](#)).

Priority Mode (DA_PREFIX)	Rx Queue ¹
00	Marvell Header mode in the CPU is disabled as described in Section 7.9.3 "88E6218 Rx Priority Queuing — Marvell Header Mode Disabled" on page 40 .
01	Marvell Header mode Enabled in the UniMAC: Rx queue set according to PRI[2:1]
10	Marvell Header mode Enabled in the UniMAC: Rx queue set according to {DBNum[0], PRI[2]}
11	Marvell Header mode Enabled in the UniMAC: Rx queue set according to {(SPID[3:0]==SPID_Setting), PRI[2]}

¹ PRI, DBNum, SPID and SPID_Setting are defined in [Section 7.9 "Rx Queuing" on page 39](#).

The BPDU and IGMP detection are not supported by the UniMAC when Marvell Header mode is enabled (see [Section 7.8.5 "Address Filtering Process \(88E6218 Only\)" on page 37](#)).

7.9.3 88E6218 Rx Priority Queuing — Marvell Header Mode Disabled

The UniMAC supports four receive priority queues. When Marvell Header Mode is disabled, the UniMAC assigns priority to each packet according to the following algorithm:

- The Type of Service (ToS) queuing algorithm is based on the decoding of the DSCP field from the IP header. The DSCP field is located in the 6-most significant bits of the second byte in the IP header. This field is used as an index to the 64 IPT Table entries, residing in the 88E6218 UniMAC register space. This table's 2-bit priority output is referred to in the algorithm as `tos_priority`.



Note

The `tos_priority` is only valid if the UniMAC Port Configuration Extend Register (PCXR) DSCPen, bit 21, referred to in the ToS algorithm as `tos2prio_en`, is set.

- If a VLAN tag exists in the packet, the VLAN priority tag is decoded from the 3-most significant bits of the second word in the VLAN tag. This field is used as an index to the eight entries in the VPT Table, residing in the 88E6218 UniMAC™ register space. This table's 2-bit priority output is referred to in the algorithm as `vlan_priority`.

- The UniMAC can decode BPDU and IGMP protocol packets, referred to in the algorithm as frame_bpdu and frame_igmp. This protocol detection is controlled by the UniMAC Port Configuration Extend Register (PCXR) BPDU and IGMP bits, referred to in the algorithm as bpdu_capture and igmp_capture respectively. BPDU and IGMP packets are always transferred to the high priority queue (Queue 3).



Notes

- IGMP packets are trapped only if Port_Configuration_Extend<IGMP> bit is set.
 - BPDU packets are trapped only if Port_Configuration_Extend<BPDU> bit is set.
- The UniMAC Port Configuration Extend register's PRIOrx_Override bit 8, referred to in the ToS algorithm as override_priority, takes precedence over tos_priority and vlan_priority. If this bit is set, all packets (except frame_bpdu and frame_igmp) are sent to the default_priority queue. The algorithm notation for the UniMAC Port Configuration Extend register's PRIOrx 2-bit field is default_priority.
 - The packet type is checked after checking the source address, VLAN tag (if it exists) and LLC-SNAP header (if it exists). The packet's EtherType is compared to 0x8100, referred to in the algorithm as vlan_type, or to 0x800, referred to in the algorithm as ip_type. If either vlan_type or (ip_type AND tos2prio_en) is true, the packet is referred to in the algorithm as frame_tagged.
If the packet is marked as frame_tagged, the UniMAC sends the packet to the tos_priority queue or vlan_priority queue. If both tos_priority and vlan_priority are extracted from the packet, the 88E6218 sends the packet to the higher value queue.
 - If tos_priority and vlan_priority are missing from the packet, the 88E6218 uses the priority value found in the matched Hash Table entry. The Hash Table entry match, referred to in the algorithm as da_found, occurs when the Destination Address matches the entry's address and the entry is valid.
The 2-bit priority value extracted from the hash table, referred to in the algorithm as ht_priority, is located on bits 52:51 of the Hash Table entry. The address to be compared is located on bits 50:3 of the Hash Table entry. The validity of the entry is stated in bits 2:0.
 - In all other cases, priority is determined by Port_Configuration_Extend<PRIOrx> bits. This includes untagged Broadcast packets.

Figure 9 shows the Type of Service receive queuing algorithm.

Figure 9: Type of Service Receive Queuing Algorithm (88E6218 Only)

```

if ((frame_bpdu & bpdu_capture) |
    frame_igmp & igmp_capture))
    queue = 3; // highest
else if (override_priority ||
    frame_broadcast & !tagged_frame))
    queue = default_priority;
else if (tagged_frame)
    queue = (tos_priority > vlan_priority)?
            tos_priority : vlan_priority;
else if (da_found)
    queue = ht_priority;
else queue = default_priority;

```

7.9.3.1 Mapping IP DSCP or VLAN Tag to the Rx Priority Queue

To define a priority queue to the IP DSCP or VLAN entry, the entry's priority 0 and priority 1 bits must be defined (see the registers in [Section A.4.3 "IP Differentiated Service Registers" on page 109](#)). [Table 15](#) and [Table 16](#) describe the writing of IP DSCP and VLAN entries respectively for a few predefined examples. [Table 17](#) and [Table 18](#) describe three example cases for mixed priority queuing.

Table 15: Writing IP DSCP Priority Example

IP DSCP Value	Priority MSB Bit	Priority LSB Bit
0	DSC1L[0]	DSC0L[0]
16	DSC1L[16]	DSC0L[16]
31	DSC1L[31]	DSC0L[31]
32	DSC1H[0]	DSC0H[0]
48	DSC1H[16]	DSC0H[16]
63	DSC1H[31]	DSC0H[31]

Table 16: Writing VLAN Priority Example

VLAN Priority Value	Priority MSB Bit	Priority LSB Bit
0	VPT2P[8]	VPT2P[0]
4	VPT2P[12]	VPT2P[4]
7	VPT2P[15]	VPT2P[7]

Table 17: Writing IP DSCP and VLAN Priority Example

Case	IPDSCP	All Others	VLAN Tag Packets
A	0x0 and 0x3f (63) directed to priority queue 3	directed to priority queue 0	priority 0
B	<0x20 (32) directed to priority queue 2	directed to priority queue 1	directed to priority queue 3
C	0x1f (31) and 0x20 (32) directed to priority queue 3 0x0 and 0x3f (63) directed to priority queue 0 <0x1f (31) directed to priority queue 1	directed to priority queue 2	>3 and IP DSCP ≠ (0x1F or 0x20) directed to priority queue 2 other tags are ignored

Table 18: Writing IP DSCP and VLAN Priority Register Mapping Example

Register	Case A	Case B	Case C
DSC0L	0x00000001	0x00000000	0xFFFFFFFFE
DSC0H	0x80000000	0xFFFFFFFFF	0x00000001
DSC1L	0x00000001	0xFFFFFFFFF	0x80000000
DSC1H	0x80000000	0x00000000	0x7FFFFFFF
VPT2P	0x00000000	0x0000FFFF	0x0000F000

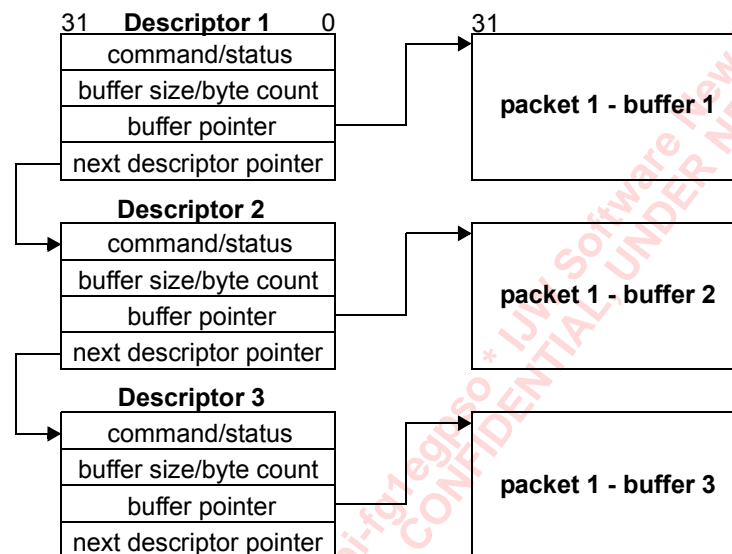
7.10 DMA Transfer

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, which includes error indications, is written by the UniMAC to the first or last descriptor 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 (i.e. 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 10 shows memory arrangement for a single packet using three buffers.

Figure 10: UniMAC Descriptors and Buffers



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

7.11 Receive Operation

To initialize a receive 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 first and current descriptor registers (RxFDP, RxCDP) (see [UniMAC First Rx Descriptor Pointer 0 Register \(88E6218 Only\)](#) and [UniMAC Current Rx Descriptor Pointer 0 Register \(88E6218 Only\)](#)) associated with the queue to be started. If multiple queues are needed, the user has to initialize RxFDP and RxCDP for each queue. See [Table 76 on page 110](#) and [Table 80 on page 111](#).
3. Initialize and enable the UniMAC port by writing to the port's configuration and command registers.
4. Initialize and enable the DMA channel by writing to the DMA's configuration and command registers.



Note

The RxDMA supports four queues in the 88E6218 and two in the 88E6208. If the user wants to take advantage of this capability, a separate list of descriptors and buffers should be prepared for each of the queues.

After completing these steps, the port starts waiting for a receive frame to arrive at the Switch interface. When this occurs, receive data is packed and transferred to the internal RxFIFO. At the same time, address filtering test is done to decide if the packet is destined to this port.¹ If the packet passes address filtering check, a decision is made regarding the destination queue to which this packet should be transferred. When this is done, actual data transfer to memory takes place.



Note

In the 88E6218, packets which 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 done.

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



Notes

- The RxCDP and RxFDP point to the same descriptor whenever the DMA is ready for receiving a new packet. RxFDP is not modified during packet reception and points to the first descriptor. 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 (i.e. descriptor returned to CPU ownership).
- Ownership of any descriptor other than the first is returned to CPU upon completion of data transfer to the buffer pointed by that descriptor. This means that the first descriptor of a packet is the last descriptor to return to CPU ownership.

¹ Address filtering is relevant to the 88E6218 only.

Bits	Name	Description
31	O	Ownership bit. When set to '1', the buffer is "owned" by the UniMAC. When set to '0', the buffer is owned by CPU.
30	AM	Auto Mode When set, the DMA does not clear the Ownership bit at the end of buffer processing.
29:24	RES	Reserved.
23	EI	Enable Interrupt The UniMAC generates a maskable interrupt upon closing the descriptor. To limit the number of interrupts and prevent an interrupt per buffer situation, the user should set the EI bits in all the Rx descriptors and set RIFB bit in the DMA Configuration register. Once RIFB bit is set, the RxBuffer interrupt is set only on frame (rather than buffer) boundaries.
22:18	RES	Reserved.
17	F	First Indicates first buffer of a packet.
16	L	Last Indicates last buffer of a packet.

Table 19: UniMAC RX Descriptor — Command/Status Word (Continued)

Bits	Name	Description
15	ES	Error Summary Valid only if F (bit 17) is set. <i>In the 88E6218 only:</i> ES = CE or OR or SF <i>In the 88E6208:</i> The Error Summary is equal to OR.
14	RES	Reserved
13	HE	Hash Table Expired Set to indicate that hash process was not completed in time. This means there is no definite answer as to whether this packet's address is in the hash table or not. Also, set when there is no room in the table for this address. Valid only if F (bit 17) is set. NOTE: This bit applies to the 88E6218 only.
12	M	Missed Frame 0 - Match 1 - Miss Set to indicate that this packet's Destination Address is not found in the address table. This bit may be set if HDM, PM_MC or PM_UC are set in the Port Configuration register (see Table 59 on page 98). Also, set to receive broadcast packets regardless of the HDM or PM settings in the Port Configuration register. This bit is valid only if F (bit 17) is set. NOTE: This bit will be set to 0 only if the DMA searched and found an address, refer to Figure 8, "Address Filtering Process," on page 38 . This bit applies to the 88E6218 only.
11:9	RES	Reserved.
8	SF	Short Frame Error Indicates that a frame shorter than 64 bytes was received. In normal operation mode short packets are automatically discarded by the port. Short packets are accepted only when PBF is set in the Port Configuration register. Valid only if F (bit 17) is set. NOTE: This bit applies to the 88E6218 only.
7	RES	Reserved
6	OR	Overrun Error Indicates that the RX DMA was unable to transfer data from Rx FIFO to memory fast enough, causing data overrun in the FIFO. Valid only if F (bit 17) is set.
5:1	RES	Reserved.
0	CE	CRC Error Received CRC does not match calculated CRC for the received packet. Valid only if F (bit 17) is set. NOTE: The Switch Core never forwards frames with bad CRC. This bit applies to the 88E6218 only.

Table 20: UniMAC RX Descriptor — Buffer Size / Byte Count

Bits	Name	Description
31:16	Buffer Size	Buffer Size in Bytes When number of bytes written to this buffer is equal to Buffer Size value, the DMA closes the descriptor and moves to the next descriptor. Bits 18:16 must be set to 0.
15:0	Byte Count	When the descriptor is closed this field is written by the device with a value indicating number of bytes actually written by the DMA into the buffer.

Table 21: UniMAC RX Descriptor — Buffer Pointer

Bits	Name	Description
31:0	Buffer Pointer	32-bit Pointer to the Beginning of the Buffer Associated with the Descriptor RX buffers have to be 64-bit aligned, so bits 2:0 must be set to 0.

Table 22: UniMAC RX Descriptor — Next Descriptor Pointer

Bits	Name	Description
31:0	Next Descriptor Pointer	32-bit Next Descriptor Pointer to the Beginning of Next Descriptor Bits 3:0 must be set to 0. DMA operation is stopped when a NULL value in the Next Descriptor Pointer field is encountered.

7.11.2 RX DMA Pointer Registers

The RX DMA employs two 32-bit pointer registers per queue: Rx FDP and Rx CDP.

- Rx FDP - RX DMA First Descriptor Pointer see [Table 79: "UniMAC First Rx Descriptor Pointer 3 Register" on page 111](#).
Rx FDP 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 should be the address of the first descriptor to use.
- Rx CDP - RX DMA Current Descriptor Pointer see [Table 83: "UniMAC Current Rx Descriptor Pointer 3 Register" on page 112](#).
Rx CDP is a 32-bit register used to point to the current descriptor of a receive packet. The CPU must initialize this register before enabling DMA operation. The value used for initialization should be the same as the value used for initializing Rx FDP (i.e. address of first descriptor to use).

7.12 Transmit Operation

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

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



Note

The TxDMA supports two transmit queues — high and low — in the 88E6218 and one transmit queue in the 88E6208. 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 descriptor registers (TxCDP) associated with the queue to be started. If both queues are needed, initialize TxCDP for each queue.¹
3. Initialize and enable the UniMAC 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.

After completing these steps, the DMA starts and performs arbitration between the transmit queues according to the value programmed in Port_Configuration_Extend<PRIOtx>. The DMA then fetches the first descriptor from the specific queue it decided to serve, and starts transferring data from the memory buffer to the TX-FIFO.



Note

The 88E6208 supports only a single UniMAC Tx queue and, therefore, no queue arbitration is implemented.

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

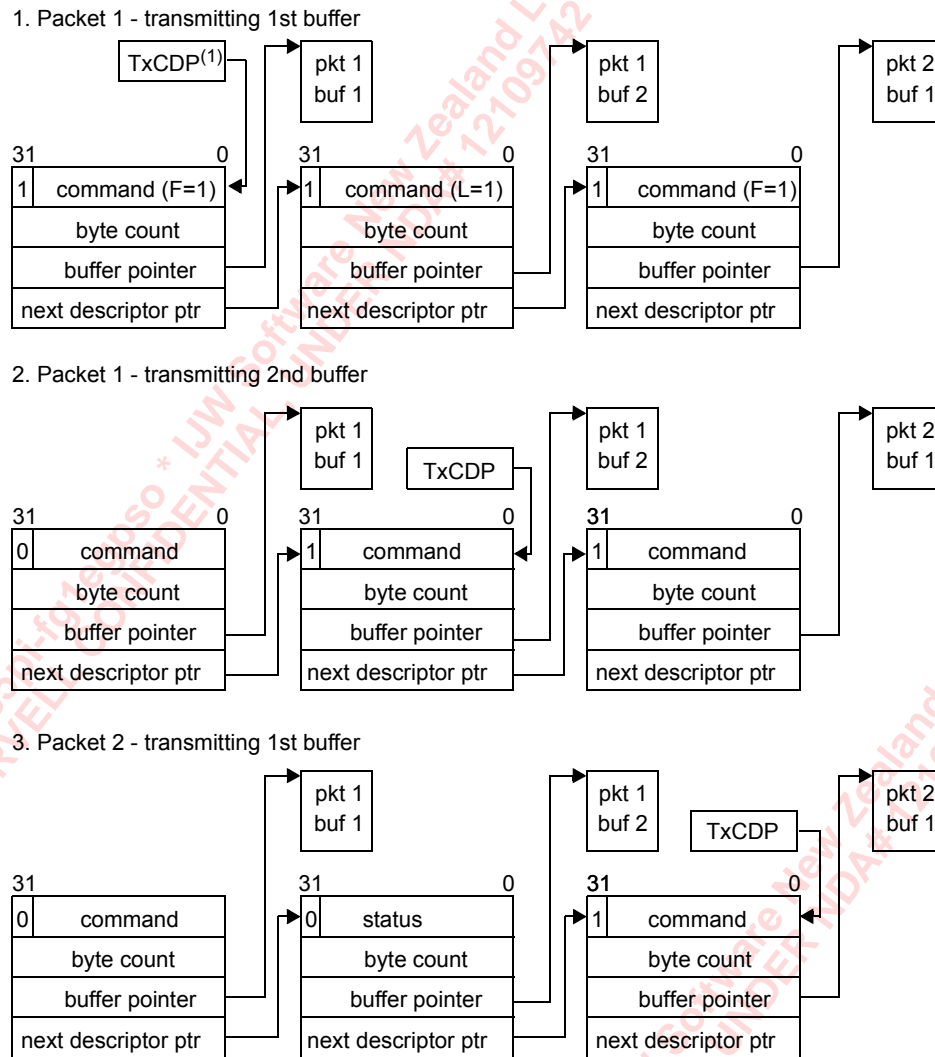
When transmission is completed, status is written to the first longword of the last descriptor. The Next Descriptor's address, which belongs to the next packet in the queue, is written to the current descriptor pointer register.

This process (starting with DMA arbitration) is repeated as long as there are packets pending in the transmit queues.

Figure 12 shows how the TX descriptors are managed when a two buffers packet is transmitted.

¹ Multiple Tx Queues are supported in the 88E6218 only.

Figure 12: UniMAC Packet Transmission Example



(1) TxCDP = TX Current Descriptor Pointer

Ownership of any descriptor other than the last is returned to 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 frame was forwarded to Switch Core. While changing the ownership bit of the last descriptor, the DMA also writes status information, which indicates any errors that might have happened during transmission of this packet.

7.12.1 TX DMA Descriptors

Figure 13 depicts the format of TX DMA descriptors. The following set of restrictions apply to TX descriptors (also see Figure 14).

- Descriptor length is 4LW and it must be 4LW aligned (i.e., Descriptor_Address[3:0]=0000).
- Descriptors may reside anywhere in the CPU accessible address space except for NULL address (0x00000000), which is used to indicate the end of a descriptor chain.
- Last descriptor in the linked chain must have a NULL value in its NextDescriptorPointer field.
- TX buffers associated with TX descriptors are limited to 64 KB and can reside anywhere in memory. However, buffers with a payload smaller than 4 bytes must be aligned to 32-bit boundary or spread across two consecutive 32-bit words. Figure 14 illustrates possible alignments for 2 byte payload.

Figure 13: UniMAC TX Descriptor

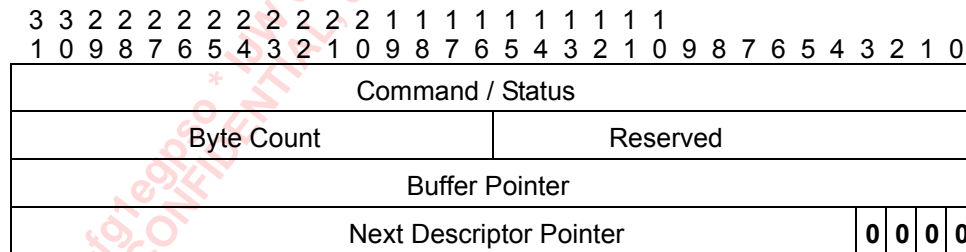


Figure 14: UniMAC TX Buffer Alignment Restrictions (2 byte payload)

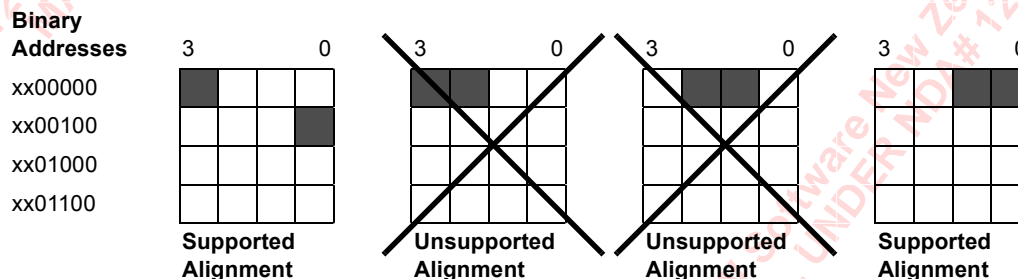


Table 23 through Table 26 provide detailed information about the TX descriptor.

Table 23: UniMAC TX Descriptor — Command/Status Word

Bits	Name	Description
31	O	Ownership bit When set to '1', the buffer is "owned" by the UniMAC. When set to '0', the buffer is owned by the CPU. Buffers owned by the CPU are not processed by the DMA.
30	AM	Auto Mode When set, the DMA does not clear the Ownership bit at the end of buffer processing.

Table 23: UniMAC TX Descriptor — Command/Status Word (Continued)

Bits	Name	Description
29:24		Reserved.
23	EI	Enable Interrupt The device generates a maskable TxBuffer interrupt upon closing the descriptor. To limit the number of interrupts and prevent an interrupt per buffer situation, the user should set this bit only in descriptors associated with LAST buffers. If this is done, TxBuffer interrupt will be set only when transmission of a frame is completed.
22	GC	Generate CRC When set, CRC is generated and appended to this packet. Valid only if L (bit 16) is set. NOTE: In Marvell Header mode the CRC includes the header. The header is stripped by the Switch and CRC is recalculated before being forwarded by the switch. A frame with bad/no CRC will be discarded by Switch Core.
21:19		Reserved.
18	P	Padding When this bit is set, zero bytes are appended to the packet if the packet is smaller than 60 bytes. Use this feature to prevent transmission of fragments. Valid only if L (bit 16) is set. NOTE: In Marvell Header mode the Padding should be done by the SW driver.
17	F	First Indicates first buffer of a packet.
16	L	Last Indicates last buffer of a packet.
15	ES	Error Summary ES = UR Set by the device to indicate an error event that occurred during packet the packet. NOTE: Valid only if L (bit 16) is set.
14:7	RES	Reserved.
6	UR	Under-Run error Indicates that part of the packet's data was not available while transmission was in progress, probably due to memory access delays). The Under-run frame will have no CRC. Valid only if L (bit 16) is set. NOTE: The bad/no CRC frame will be discarded by Switch Core.
5:0		Reserved

Table 24: UniMAC TX Descriptor — Byte Count

Bits	Name	Description
31:16	Byte Count	Number of bytes to be transmitted from associated buffer. This is the payload size in bytes.
15:0		Reserved.



Table 25: UniMAC TX Descriptor — Buffer Pointer

Bits	Name	Description
31:0	Buffer Pointer	32-bit pointer to the beginning of the buffer associated with this descriptor. Buffers with a payload smaller than 4 bytes must be aligned to 32-bit boundary or spread across two consecutive 32-bit words. (see Figure 14, “UniMAC TX Buffer Alignment Restrictions (2 byte payload),” on page 50).

Table 26: UniMAC TX Descriptor — Next Descriptor Pointer

Bits	Name	Description
31:0	Next Descriptor Pointer	32-bit pointer that points to the beginning of next descriptor. Bits 3:0 must be set to 0. DMA operation is stopped when a NULL (all zero) value in the Next Descriptor Pointer field is encountered.

7.12.2 TX DMA Pointer Registers

The TX DMA employs a single 32-bit pointer register per queue: TxCDP - TX DMA Current Descriptor Pointer.

TxCDP 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 should be the address of the first descriptor to use.

7.12.3 TX DMA Notes

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 if the EI bit in the last descriptor is set.

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 Tx_End maskable interrupt is generated. In order to restart the queue, the CPU should issue a Start_Tx command by writing '1' to the Start_Tx bit in the DMA 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. When that happens, the DMA aborts transmission and stops queue processing. A TX_Resource_Error maskable interrupt is generated. To restart the queue, the CPU should issue a Start_Tx command.

A transmit underrun occurs when the DMA can not access the memory fast enough and packet data is not transferred to the UniMAC TxFifo before the FIFO gets 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. Transmit process continues with the next packet.

To stop DMA operation before the DMA reaches the end of descriptor chain, the CPU should issue a STOP command by writing '1' to the Stop_Tx 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. In addition, a Tx_End maskable interrupt is generated. To restart this queue, the CPU should issue a Start_Tx command.

¹ When the DMA stops due to NULL descriptor pointer, the CPU has to write TxCDP before issuing a Start_Tx command. Otherwise, TxCDP remains NULL and the DMA can not restart queue processing.



Note

Most of the terms used to denote either DMA commands (Start_Tx and Stop_Tx) or interrupts (TxBuffer, Tx_End, TX_Resource_Error) actually reflect multiple terms (one per queue). For example, the 88E6218 provides two Start_Tx commands. There is a separate Start_Tx_High command, associated with the high priority queue, and a Start_Tx_low command that is related to the low priority queue. The same applies to the other commands and interrupts listed above.

7.13 MII Serial Management Interface (SMI)

The SMI interface controls the Switch Core and PHYs as well as external devices.

The UniMAC has an integrated MII Serial Management Interface (SMI) logic. This interface consists of two signals: serial data (MDIO) and, clock (MDC).

These signals enable control and status parameters to be passed between PHY devices¹ and the CPU. Multiple PHY devices can be controlled using this simple 2-pin interface.

A CPU can write/read to/from all PHY devices addresses/registers by writing and reading to/from the SMI control register. The SMI allows the CPU to directly control an MII compatible PHY device via the SMI control register. This enables the driver software to program the PHY devices into specific operation mode such as Full Duplex, Loopback, Power Down, 10/100 speed selection as well as control of the PHY device's Auto-Negotiation function, if it exists. The SMI also allows the CPU to configure all the registers contained in the 88E6218/88E6208's switch since the switch core uses the SMI interface protocol. The CPU writes commands to the SMI register and the SMI unit performs the actual data transfer via MDIO, which is a bi-directional data pin. These serial data transfers are clocked by the MDC clock output.

Refer to Part 3 of the three part 88E6208/88E6218 datasheet set (Document Number MV-S101300-03) for more information about how to program the device's internal switch and PHY registers using this SMI interface.

¹ The term PHY devices denotes any device that can be accessed through MDC/MDIO. This includes, for example, physical layer devices (either internal or external) and Marvell SOHO switches (internal and/or external).



Section 8. Independent DMA (88E6218 Only)

8.1 Functional Description

The DMA unit is a dual channel independent DMA. Each of the DMA channels arbitrates internal to the DMA controller for access to the shared bus.

The dual independent DMA channels share a 36-byte FIFO. The DMA unit has one independent DMA controller.

The DMA controller optimizes system performance by moving large amounts of data without significant CPU intervention. Rather than having the CPU read data from one source and write it to another, the DMA controller can be programmed to automatically transfer data independent of the CPU. This feature frees the CPU to continue executing other instructions simultaneously to the movement of data. Each DMA channel can move data within the SDRAM/device memory area.

8.1.1 DMA Transfers

Data is transferred from the source device into an internal FIFO and from the internal FIFO to the destination device. The DMA controller can be programmed to move up to 64 Kbytes of data per transaction. The burst length of each transfer of DMA can be set to 4, 8, 16 or 32 bytes.

Accesses can be non-aligned both in the source and the destination. There is one 36-byte FIFO available for the utilization of the DMA engine. Although the maximum DMA burst size is 32 bytes, the extra 4 bytes in the FIFO is required for non-word aligned transfers. The DMA controller supports chained mode of operation. The descriptors are stored in memory, and the DMA controller moves the data until the Null Pointer is reached.

8.2 DMA Channel Registers

8.2.1 DMA Channel Descriptor Registers

Each DMA Channel descriptor consists of the following registers that can be written by the CPU or DMA controller in the process of fetching a new descriptor from memory.

Table 27: DMA Channel Descriptor Registers

Register	Definition
Byte Count	This register is programmed with a 16-bit number which contains the number of bytes of data that this channel must DMA. The maximum number of bytes which the DMA controller can be configured to transfer is 64 Kbytes. This register will decrement at the end of every burst of transmitted data from source to destination. When the byte count register is 0, the DMA transaction is terminated.
Source Address	This register is programmed with a 32-bit address. This is the source address for data and can be from the external memories. This register will either increment, decrement, or hold the same value according to bits 3:2 of the Channel Control Register.
Destination Address	This register is programmed with a 32-bit address. This is the destination address for data and can be for the external memories. This register will either increment, decrement, or hold the same value according to bits 5:4 of the Channel Control Register.
Next Descriptor Pointer	This register contains a 32-bit address where the values for the next DMA Channel descriptor can be loaded for chained operation. This can be from the external memories. The byte count, source address, and destination address must be located at sequential addresses. NOTE: The next descriptor pointer must be 16 bytes aligned. This means bits 3:0 must be set to 0. This register is only used when the channel is configured for Chained Mode. A value of 0 (null) for this register indicates that this is the last descriptor in the chain. Refer to the DMA Control register ChainMod bit 9 for more information about Chained DMA mode.
Current Descriptor Pointer	This register contains the 32-bit address of the DMA Channel descriptor currently being processed.

8.2.2 DMA Channel Control Registers

Each DMA channel has its own unique Control Register where certain DMA modes can be programmed. The bit descriptions for each field in the control register appear in [Table 97, "Channel 0 Control Register," on page 116](#) and [Table 98, "Channel 1 Control Register," on page 118](#). The offset for Channel 0 is 0x8000E840, and the offset for Channel 1 is 0x8000E844.

8.3 Chain and Non-Chain Modes

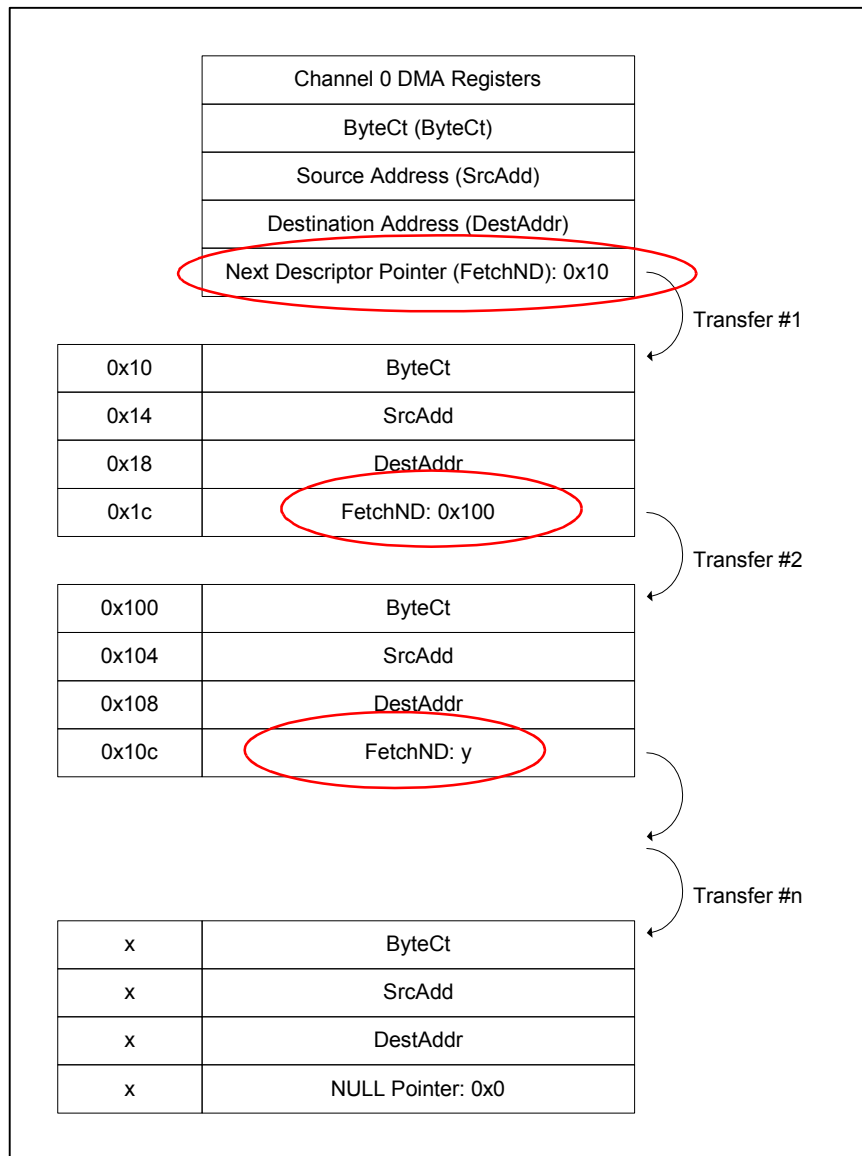
8.3.1 Chain Mode

In Chain mode, the DMA channel parameters (Byte Count, Source Address, Destination Address, and Next Descriptor Pointer, see [Table 87, p. 114](#) through [Table 94, p. 115](#)) are read from the descriptor's located in the external memory. The address of the first descriptor should be initialized by writing it to the Next Descriptor Pointer Register of the channel. Enable the channel by setting ChanEn to 1 and setting FetchND to 1. Setting the ChanEn and FetchND bits can be done during the same write or set FetchND to 1 on the first write and then initiate the DMA by setting ChanEn to 1 on another write.

The DMA channel stays in the active state until Next Descriptor Pointer is null or the byte count reaches the terminal count. In this mode, an interrupt can be asserted every time the byte count reaches the terminal count or when both the Byte Count reaches the terminal count and the Next Descriptor Pointer is null.

Figure 15 illustrates the Next Descriptor Pointer chained to the next channel descriptor for Channel 0 DMA registers.

Figure 15: Chain Mode



8.3.1.1 Dynamic DMA Chaining

Dynamic chaining is when DMA descriptors are added to a chain that a DMA controller is actively working on. The main issue is to synchronize between when the DMA controller reads the last chain descriptor (the Null pointer) to the time when the CPU changes the current last DMA descriptor. Following is an algorithm which provides this synchronization mechanism.

1. Prepare the new descriptor.
2. Change the last descriptor's Next Descriptor Pointer to point to the new descriptor.
3. Read the DMA control register.

If the DMAActSt bit is 0 (not active) {

Update the Pointer to Next Descriptor Pointer in the device and assert the FetchND bit:

}

else {

read the Pointer to Next Descriptor Pointer of the DMA unit. If it's equal to NULL {

Mark (by using a flag or something) that in the next DMA chain complete interrupt you'll need to -

[[[{

Update the Next Descriptor Pointer register in the DMA unit and write the FetchND

}]}}

}

}

8.3.2 Non-Chain Mode

In Non-Chain mode, the CPU master initiates the DMA channel parameters (Source, Destination Byte Count and Command registers). The DMA will start to transfer data after the enable bit in the Command register is set to 1. The DMA remains in an active state until the Byte Count reaches a terminal count or until the channel is disabled.



Note

In non-chained mode the Byte Count, Source, and Destination registers should be initialized prior to enabling the channel. See [Table 87, p. 114](#) through [Table 92, p. 115](#).

8.4 Interrupts

Each IDMA channel has one DMA completion signaling interrupt. For each interrupt the following apply:

- Interrupt mask
- Interrupt reset
- Interrupt status

See [Table 100, "Channel 0 Interrupt Mask Register," on page 120](#) through [Table 105, "Channel 1 Interrupt Status Register," on page 121](#).

Both channels interrupts are connected to the 88E6218 Interrupt Request Controller (see [Section 5.3 "Interrupts" on page 20](#) and [Table 4, "IRQ Interrupt Sources," on page 20](#)).



8.5 DMA Operations

8.5.1 Restarting a Disabled Channel

In Non-Chained mode, ChanEn must be set to 1.

In Chained mode, the software must determine if the first fetch took place. If it did, only ChanEn should be set to 1. If the first fetch did not take place, the FetchND must also be set to 1.

8.5.2 Reprogramming an Active Channel

To reprogram an active channel, set ChanEn to 0 to disable the channel.

If CDE is set, then ABR bit 20 in the Channel Control register must also be set. Confirm that the channel is no longer active by polling the DMAActSt of the channel, and so on. Program new DMA parameters prior to re-enabling the channel by setting ChanEn to 1.

8.5.3 Current Descriptor Pointer Registers

Each DMA channel has a current Descriptor Pointer Register (CDPTR) associated with it. They are located at offsets 0x8000E874 (see [Table 96 on page 116](#)). They are read/write registers. However, the CPU should not write them.

When the Next Pointer (NPTR) is written by the CPU, then the CDPTR reloads itself with the same value written to the NPTR. After processing a descriptor, the DMA channel updates the current descriptor using CDPTR, saves NPTR into CDPTR, and fetches a new descriptor. This register is used for closing the current descriptor before fetching the next descriptor.

Section 9. Two-Wire Serial Interface (TWSI)

9.1 Functional Description

The Two-Wire Serial Interface is a simple and efficient serial bus, which is commonly used in the industry to connect peripheral devices like serial EPROMs, small NVRAM devices, voltage/temperature monitors, etc. The 88E6218 provides full TWSI master support, which makes it capable of generating read/write requests towards TWSI slave devices.



Note

88E6218 does not support TWSI slave mode, nor does it support a multiple TWSI masters environment.

9.2 TWSI Bus Operation

The TWSI port consists of two open drain signals that are internally pulled high:

- SCK (Serial Clock)
- SDA (Serial Data/Address)



Note

The 88E6218 does not have dedicated TWSI pins. The SCK and SDA signals are multiplexed on GPIO pins [15] and [14], respectively. Refer to [Section 10.6 "GPIO Port Pin Assignments" on page 66](#) for details about setting these GPIO pins for TWSI operation.

The TWSI master starts a transaction by driving a start condition followed by a 7- or 10-bit slave ID and a read/write bit indication. The target TWSI slave responds with acknowledge.

In case of write access (R/\overline{W} bit is '0'), following the acknowledge, the master drives 8-bit address and the slave responds with acknowledge. This write access (8-bit data followed by acknowledge) continues until the TWSI master ends the transaction with stop condition.

In case of read access, following the 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.

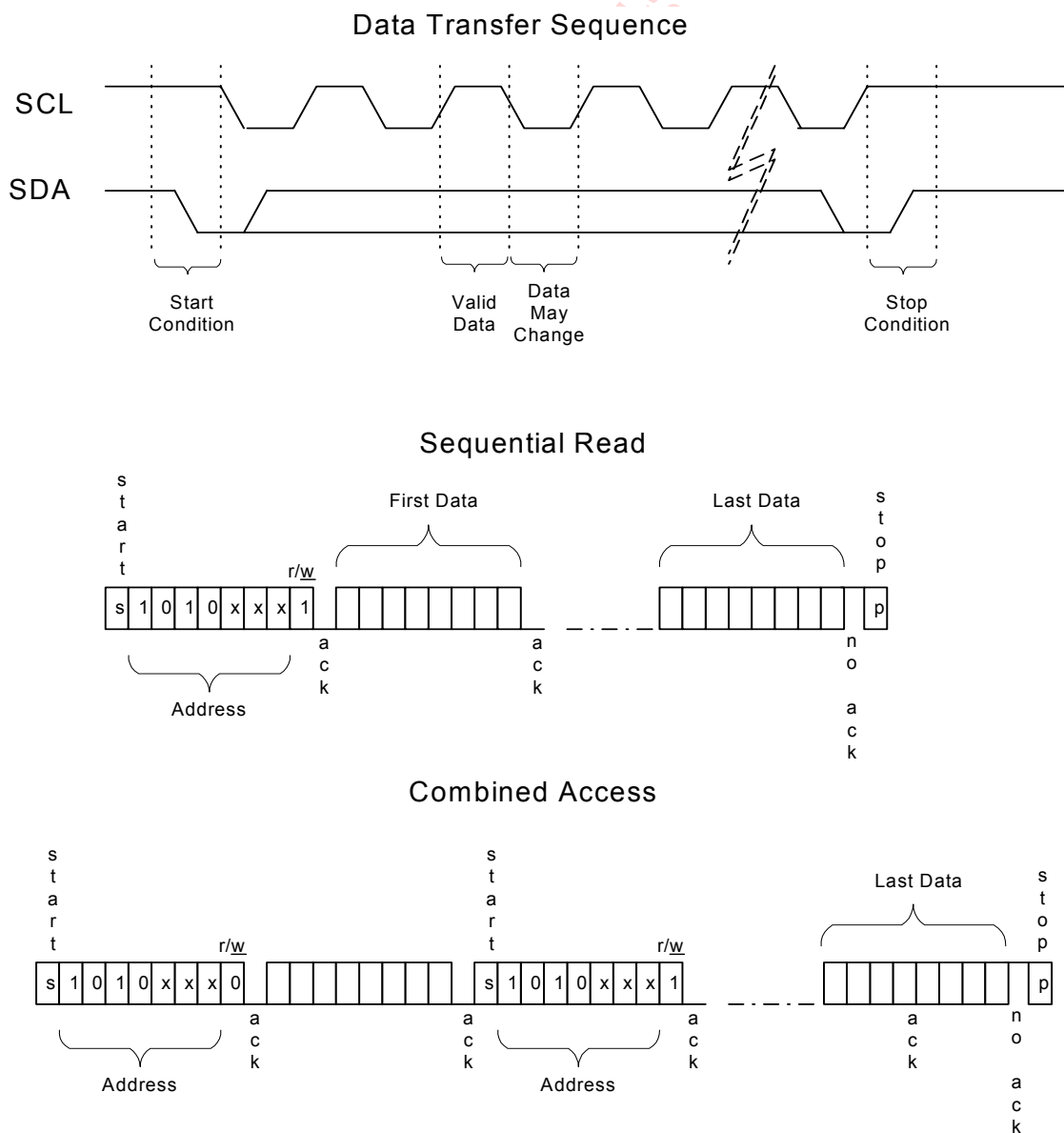
A target slave that cannot drive valid read data right after it received the address, can insert "wait states" by forcing SCK low until it has valid data to drive on the SDA line.

A master is allowed to combine two transactions. After the last data transfer, it can drive a new start condition followed by new slave address, rather than drive stop condition. Combining transactions guarantees that the master does not loose arbitration to some other TWSI master.¹

TWSI examples are shown in [Figure 16](#). For a full TWSI protocol description refer to the Philips Semiconductor TWSI specification.

1. This is a general characteristic of the TWSI bus. However, the 88E6218 does not support multiple masters on the TWSI bus.

Figure 16: TWSI Examples



9.3 Soft Reset

Soft reset is programmed through Peripheral Reset Register, offset 0x8000C000 (see [Table 114 on page 124](#)). Set bit Prst to 1 to reset the peripherals. This signal generates a 4-cycle length of active low pulse after the rising edge of Uclk to initialize peripheral devices to their default states. The register value is cleared to zero at the end of reset.

9.4 Peripheral Clock Speeds

Peripheral clock speeds are computed by the following equation:

$$Uclk = SYSCLK(1/(CLK_DIV+1))$$

where SYSCLK is equal to 83.33, 100, 125, 133.33, 143, 150, or 166 MHz.

Uclk frequency is programmed through the Peripheral Clock Divider Register, offset 0x8000C004 (see [Table 115, page 124](#)). The table shows the resulting Uclk frequencies given the selected clock divider (CLK_DIV) setting.

Table 28: Uclk Frequencies Based on SYSCLK and CLK_DIV

Clock Divider			SYSCLK Setting in MHz						
Value	Description	Divider	83.33	100	125	133.33	143	150	166.67
00	SYSCLK/1	1	83.33	100	125	133.33	143	150	166.67
01	SYSCLK/2	2	41.67	50	62.5	66.67	71.5	75	83.34
10	SYSCLK/4	4	20.83	25	31.25	33.33	35.75	37.5	41.67
11	SYSCLK/8	8	10.42	12.5	15.62	16.67	17.88	18.75	20.83

9.5 TWSI Registers

The TWSI unit provides a simple register based software interface, providing the CPU with full control of the TWSI interface activities. The following sections describe each of these registers.

9.5.1 TWSI Clock Prescaler Register

TWSI specification defines maximum SCK frequency of 100 KHz (400 KHz in fast mode). The TWSI module contains a clock divider that divides $Uclk^1$ to generate the SCK clock. SCK frequency is defined as follows:

$$FREQ_{SCK} = \frac{FREQ_{Uclk}}{5 \times PRER}$$

Where PRER is the value represented by bits 15:0 of the TWSI Clock Prescaler Register (see [Table 107 on page 122](#)).

9.5.2 TWSI Global Control Register

This register must be written 0x2 for proper operation of the TWSI unit. See [Table 108, "TWSI Global Control Register," on page 122](#).

1. Uclk is the clock used within the Low Speed Bus unit.

9.5.3 TWSI Control Register

This register is used to set start or stop conditions on the next TWSI transaction. When the TWSIStart bit is set, the next TWSI transaction will begin with a START signal. When the TWSIStop bit is set, the next TWSI transaction will end with a STOP condition. See [Table 109, "TWSI Control Register," on page 123](#).

9.5.4 TWSI Status Register

This is a read-only register, which contains status information about the TWSI interface. The TWSIBusy bit indicates that the bus is currently busy. The RxAck bit is used to indicate if the last TWSI write access has been acknowledged or not. See [Table 110, "TWSI Status Register," on page 123](#).

9.5.5 TWSI Interrupt Mask Register

The ARM CPU writes this register to mask the TWSI interrupt. Only one TWSI interrupt event has been defined — TWSI write-slave_nonACK interrupt. See [Table 111, "TWSI Interrupt Mask Register," on page 123](#).

9.5.6 TWSI Data Register

This is an 8-bit register into which the CPU must place the slave address or write data to be transmitted. In case of read access, this register holds the received data (which must be read by the CPU). See [Table 112, "TWSI Data Register," on page 124](#).



Note

Data register MSB contains the first bit transmitted or received on the TWSI bus.

9.5.7 TWSI Interrupt Source Register

ARM CPU reads this register to check the post masked interrupt sources. The register only holds its value for 1 cycle after read access and then clears itself. The ARM CPU then needs to do a follow up write to enable it to latch future interrupt events. See [Table 113, "TWSI Interrupt Source Register," on page 124](#).

9.6 TWSI Operation

The CPU can initiate TWSI master read and write transactions via TWSI registers, as described in the following sections.



Note

Before initiating TWSI transactions, the CPU must program the TWSI Clock Prescaler Register (see [Table 107 on page 122](#)) and the TWSI Global Control Register (see [Table 108 on page 122](#)).

9.6.1 Master Write Access

Master write access consists of the following steps:

1. Read the TWSI Status register. Check the TWSIBusy bit to verify that the bus is IDLE.
2. Write the TWSI Control register to set the TWSIStart bit.
3. Write the 7-bit TWSI slave ID and R/W bit to the TWSI Data register.
4. Write the 8-bit address to the TWSI Data register.
5. Write the 8-bit data to the TWSI Data register.
6. Continue writing 8-bit data bytes to the TWSI Data register if using page write mode.
7. Before writing the last data byte, write the TWSI Control register with the TWSIStop bit set.
8. Write the last byte of data to the TWSI Data register.



Note

This sequence describes a normal operation. There are also abnormal cases, such as a slave not responding with acknowledge. These cases are reported in the TWSI Status register and needs to be handled by the CPU.

9.6.2 Master Read Access

Master read access consists of the following steps:

1. Read the TWSI Status register. Check the TWSIBusy bit to verify that the bus is IDLE.
2. Write the TWSI Control register to set the TWSIStart bit.
3. Write the 7-bit TWSI slave ID and R/\overline{W} bit to the TWSI Data register.
4. Write the 8-bit address to the TWSI Data register.
5. Write the TWSI Control register to set the $TWSIStart$ bit.
6. Write the 7-bit TWSI slave address and R/\overline{W} bit to the TWSI Data register.
7. Read the 8-bit data from the TWSI Data register.
8. Continue reading 8-bit data bytes from the TWSI Data register if using sequential read.
9. Before reading the last data byte, write the TWSI Control register with the TWSIStop bit set.
10. Read the last byte of data to the TWSI Data register.

Section 10. General Purpose I/O Port (GPIO)

10.1 Functional Description

The 88E6218 and 88E6208 contain a 16-bit General Purpose I/O Port (GPIO).

Each of the GPIO pins can be assigned to act as a general purpose input or output pin and can be used to register external interrupts (when assigned as input pin). Moreover, the GPIO port also supports additional functions (like TWSI and software/status LEDs).



Note

External interrupts are triggered only on GPIO[15:4].

10.2 GPIO Control Registers

The GPIO port has associated control registers, which allow a per pin control of the functionality.

The I/O Control registers determine the direction for each GPIO pin, either input or output. When a GPIO pin is configured as output, it drives a value written into the corresponding GPIO output register, while when configured as input it allows reading the pin state through the GPIO input register.

The GPIO port has an additional control register, namely the GPIO Select register. This register¹ is used to select between multiple functional options associated with each pin. Using this register the user can program a specific GPIO pin as either a general purpose input/output, software controlled LED, status LED² or alternate function (e.g., TWSI).

10.3 GPIO LED Support

As mentioned above, the GPIO port provides software controlled LED functionality. Almost all GPIO pins can be configured as a software controlled LED output pins. The GPIO Select registers control the setting of each GPIO pin as either LED output or other functions.

For each pin configured as a software controlled LED output, the 88E6218 and 88E6208 provides full control of the LED cycle (blink rate) as well as the duty cycle. This is done by programming registers SW_LED_CYCLE[15:0] and SW_LED_DUTY_CYCLE[15:0].

Slew rate control for the LEDs is also supported. See [Section 10.4 "GPIO Slew Rate Control" on page 65](#) for more details.

1. The GPIO_Select is actually comprised of two registers, GPIO_Select[15:8] and GPIO_Select[7:0] (see [Table 117, p. 127](#) and [Table 118, p. 128](#)).

2. This option exist only for GPIO[3:0].



Notes

- The software LED control registers are not designed to provision a constant ON setting. The maximum duty cycle supported is 1/2. To get a software LED with constant ON capability, the user should use the GPIO output rather than LED output setting. Configuring a pin as a GPIO output enables full software control of its state through the GPIO Output register.
- Also, if a GPIO pin is configured as GPIO output and is used to drive a LED, it is recommended to set the corresponding Software LED Output Enable bit to '1'.

GPIO pins [3:0] can also be configured as status LEDs, to provide indication for internal events associated with the UniMAC™-to-switch-core link. These events are: TxEn (Transmit Enable signal driven by the internal UniMAC), RxDV (Receive Data Valid driven from the internal switch core to UniMAC), Act (OR of TxEn and RxDV) and Link/Activity¹. For these status LEDs, the 88E6218 and 88E6208 support programmable pulse stretching. Programmable blink rate is supported for the Link/Activity LED only.

Slew rate control for status LEDs is also supported. See [Section 10.4 "GPIO Slew Rate Control" on page 65](#).

10.3.1 Pulse Stretching for Status LEDs

LED status signals driven on GPIO pins [3:0] can be pulse stretched. Pulse stretching is necessary because the duration of these status events may be too short to be observable on the LEDs. The pulse-stretched duration is programmable via register (LED[3:0] Stretch Duration register).

10.3.2 Blink Rate for Link/Activity LED

88E6218 and 88E6208 support programmable blink rate for the Link/Activity status LED. Use the Link/Activity Cycle register to program the blink rate for this LED. The duty cycle of this LED is 1/2.

10.4 GPIO Slew Rate Control

88E6218 and 88E6208 support slew rate control for the GPIO pins. Slew rate and drive capability control are provided as part of the LED support described above. The slew rate for each GPIO pin is controlled by the register Software LED Output Enable. When a GPIO pin is configured as software controlled LED, the user should set the Software LED Output Enable bit associated with that pin. Otherwise, this bit must be reset.



Note

If a GPIO pin is not used to drive a LED, the corresponding bit in the Software LED Output Enable register must be reset. Otherwise, this pin may not function properly.

Controlling the slew rate for the status LEDs is done through another register — the Status LED Output Enable register.



Note

Once a GPIO pin is configured as a status LED, its slew rate is controlled only by the corresponding Status LED Output Enable register bit. The setting of the Software LED Output Enable register is not relevant for GPIO pins which function as status LEDs.

1. The Link/Activity shows the status of the internal link between the UniMAC™ and the switch. It is constantly on when the link is up and there is no activity. It blinks when the link is up and there is activity.

10.5 GPIO Interrupts

The GPIO input pins can be used to register external interrupts.

Assertion of a GPIO input pin (toggle from '0' to '1' in case of active high pin, from high to low in case of active low pin) results in setting the corresponding bit in the GPIO Interrupt Status register.



Note

The GPIO pin must be kept active for at least one internal SysClk cycle to guarantee that the interrupt is properly registered.

If not masked, the GPIO interrupt may cause a CPU interrupt. The interrupt signal¹ from GPIO is de-asserted as soon as software clears the corresponding bit in the GPIO Interrupt Status register. Multiple assertions of a GPIO input pin, while the interrupt cause bit is already set, cannot be registered.



Note

GPIO [3:0] cannot be interrupt sources.

10.6 GPIO Port Pin Assignments

The following table summarizes the various functions associated with each of the GPIO port pins as a function of the corresponding GPIO Select register bits setting (see also [Table 117 on page 127](#) and [Table 118 on page 128](#)).

Table 29: GPIO Port Pin Assignments

GPIO Pin	GPIOSelect =			
	00	(Default value) 01	10	11
GPIO[0]	LED for status signal UniMAC™ Rx	GPIO output register GPIO_OUT[0]	Reserved	Reserved
GPIO[1]	LED for status signal UniMAC Tx	GPIO output register GPIO_OUT[1]	Reserved	Reserved
GPIO[2]	LED for status signal Act (logic OR of RxEn, TxDV)	GPIO output register GPIO_OUT[2]	Reserved	Reserved
GPIO[3]	UniMAC LED for status signal Link/Activity	GPIO output register GPIO_OUT[3]	Reserved	Reserved
GPIO[4]	Software controlled LED output SW_LED_OUT[4]	GPIO output register GPIO_OUT[4]	Reserved	Reserved
GPIO[5]	Software controlled LED output SW_LED_OUT[5]	GPIO output register GPIO_OUT[5]	Reserved	Reserved
GPIO[6]	Software controlled LED output SW_LED_OUT[6]	GPIO output register GPIO_OUT[6]	Reserved	Reserved
GPIO[7]	Software controlled LED output SW_LED_OUT[7]	GPIO output register GPIO_OUT[7]	Reserved	Reserved

1. This is an internal signal.

Table 29: GPIO Port Pin Assignments (Continued)

GPIO Pin	GPIOSelect =			
	00	(Default value) 01	10	11
GPIO[8]	Software controlled LED output SW_LED_OUT[8]	GPIO output register GPIO_OUT[8]	Reserved	Reserved
GPIO[9]	Software controlled LED output SW_LED_OUT[9]	GPIO output register GPIO_OUT[9]	Reserved	Reserved
GPIO[10]	Software controlled LED output SW_LED_OUT[10]	GPIO output register GPIO_OUT[10]	Reserved	Reserved
GPIO[11]	Software controlled LED output SW_LED_OUT[11]	GPIO output register GPIO_OUT[11]	Reserved	Reserved
GPIO[12] ¹	Reserved	GPIO output register GPIO_OUT[12]	Reserved	Reserved
GPIO[13]	Reserved	GPIO output register GPIO_OUT[13]	Software controlled LED output SW_LED_OUT[12]	Reserved
GPIO[14]	TWSI Data	GPIO output register GPIO_OUT[14]	Software controlled LED output SW_LED_OUT[13]	Reserved
GPIO[15]	TWSI Clock	GPIO output register GPIO_OUT[15]	Software controlled LED output SW_LED_OUT[14]	Reserved

1. When the Watchdog Enable Register bit 0 WdEn is set to 1 (see Table 152 on page 145), the watchdog will use GPIO[12] pin no matter what the value set in GPIOSelect[12].



Section 11. UART

11.1 Functional Description

The 88E6218 and 88E6208 devices support a simple, 2-pin and Universal Asynchronous Receiver/Transmitter (UART).

For complete information regarding the UART, refer to the Synopsis DW_16550 specification.

The UART is integrated into the device to support the following operations:

- Diagnostic tests
- Data input/output operations for peripheral devices connected through a standard UART interface

The UART includes the following features:

- Synchronous interface
- FIFO mode permanently selected for transmit and receive operations
- Two pins for transmit and receive operations

11.2 Interface Description

The 88E6218 supports the UART interface through the UART_TX (53) and UART_RX (52) pins.

Table 30 shows the signal name on the 88E6218 device, the corresponding pin name in the 16550 specification, and the description of the pin.

Table 30: UART Pin Definitions

88E6218 Pin Name	16550 Standard Pin Name	Description
UART_TX	sout	The UART_TX signal is the serial data output to the modem, data set, or peripheral device. The UART_TX signal is set high when the reset is applied.
UART_RX	sin	The UART_RX signal is the serial data input from the modem, data set, or peripheral device.

11.3 Using the UART as a Test Port

The user can upload a test diagnostic program to the CPU through the Flash, EEPROM, or UART interface. During execution, the diagnostic program can transmit performance and status information through the UART.

Section 12. Watchdog Timer

The watchdog timer is a hardware protection against malfunction. It is a programmable timer that is reset by the software at regular intervals. Failure will cause the 88E6208/88E6218 to reset.¹ When the system is operating properly, the software that is executing will clear the timer on a regular basis. Setting and clearing the watchdog timer is performed by the software.

When using the watchdog, GPIO[12] must be connected to the reset module on the board, and should cause the reset module to assert a system reset upon receiving a GPIO[12] signal that is LOW.

After the watchdog timer is enabled, if it is not cleared within the time period specified in the Watchdog Timer Interval Register (see [Table 153 on page 145](#)), the 88E6208/88E6218 forces the GPIO[12] pin to LOW, and GPIO[12] remains LOW until system reset is done. After the 88E6208/88E6218 resets, GPIO12 will be pulled HIGH since the watchdog circuit is cleared at reset and should be re-enabled by the software.²



Note

When enabling the watchdog, GPIO12 operates as an open drain output, which is pulled-up by the internal pull-up in the 88E6208/88E6218. This configuration allows connecting the signal without having to add additional logic to the hardware reset module on the board.

1. The 88E6208/88E6218 does not have internal logic to automatically reset the device upon a watchdog event. The system designed is expected to handle that.

2. The Watchdog timer and relating registers do not operate in 88E6208/88E6218 silicon rev. A.



Appendix A

88E6208 and 88E6218 CPU and Peripherals Register Set

List of Registers

A.2	CPU Interface Registers	77
Table 32:	System Configuration Registers Offset: 0x80002000	78
Table 33:	Shared Bus Arbiter Configuration Low Offset: 0x80002004	78
Table 34:	Shared Bus Arbiter Configuration High Offset: 0x80002008	79
Table 35:	Interrupt Request Status Register Offset: 0x90008000	79
Table 36:	Interrupt Source Status Register Offset: 0x90008004	80
Table 37:	Interrupt Enable Set Register Offset: 0x90008008	81
Table 38:	Interrupt Enable Register Offset: 0x90008008	82
Table 39:	Interrupt Enable Clear Register Offset: 0x9000800C	83
Table 40:	Software Programmed Interrupt Register Offset: 0x90008010	84
Table 41:	Fast Interrupt Request Status Register Offset: 0x90008100	84
Table 42:	Fast Interrupt Source Status Register Offset: 0x90008104	84
Table 43:	Fast Interrupt Enable Set Register Offset: 0x90008108	84
Table 44:	Fast Interrupt Enable Register Offset: 0x90008108	84
Table 45:	Fast Interrupt Enable Clear Register Offset: 0x9000810C	85
Table 46:	Timer x Length Register Offset: Timer 1 0x90009000, Timer 2 0x90009004, Timer 3 0x90009008, Timer 4 0x9000900C	85
Table 47:	Timer Control Register Offset: 0x90009010	85
Table 48:	Timer x Value Register Offset: Timer 1 0x90009014, Timer 2 0x90009018, Timer 3 0x9000901C, Timer 4 0x90009020	87
Table 49:	Interrupt Request Register Offset: 0x90009024	88
Table 50:	Interrupt Mask Register Offset: 0x90009028	88
A.3	Memory Controller Registers	89
Table 52:	Control and Status Register Offset: 0x80006000	90
Table 53:	Configuration Register x (CFGRx) Offset: CFGR0: 0x80006004, CFGR1: 0x80006010, CFGR2: 0x8000601C, CFGR3: 0x80006028	90
Table 54:	Timing Parameter (SDRAM) Register 0 Offset: M_CS0n: 0x80006014, M_CS1n: 0x80006020, M_CS2n: 0x8000602C	92



Table 55:	Timing Parameter (SDRAM) Register 1 Offset: M_SC0n: 0x80006018, M_SC1n: 0x80006024, M_SC2n: 0x80006030	93
Table 56:	Timing Parameter (Asynchronous Device) Register 0 Offset: BootCSn: 0x80006008, M_CS0n: 0x80006014, M_CS1n: 0x80006020, M_CS2n: 0x8000602C .	94
A.4	UniMAC™ Unified Fast Ethernet Media Access Control Registers	96
Table 58:	SMI Register (SMIR) Offset: 0x80008010	97
Table 59:	Port Configuration Register (PCR) Offset: 0x80008400	98
Table 60:	Port Configuration Extend Register (PCXR) Offset: 0x80008408	99
Table 61:	Port Command Register (PCMR) Offset: 0x80008410	102
Table 62:	Port Status Register (PSR) Offset: 0x80008418	102
Table 63:	Hash Table Pointer Register (HTPR) — 88E6218 Only Offset: 0x80008428	103
Table 64:	Flow Control Source Address Low Register (FCSAL) — 88E6218 Only Offset: 0x80008430	103
Table 65:	Flow Control Source Address High Register (FCSAH) — 88E6218 Only Offset: 0x80008438	103
Table 66:	SDMA Configuration Register (SDCR) Offset: 0x80008440	104
Table 67:	SDMA Command Register (SDCMR) Offset: 0x80008448	104
Table 68:	Interrupt Cause Register (ICR) Offset: 0x80008450	106
Table 69:	Interrupt Reset Select Register (IRSR) Offset: 0x80008454	108
Table 70:	Interrupt Mask Register (IMR) Offset: 0x80008458	109
Table 71:	IP Differentiated Services CodePoint to Priority0 Low Register (DSC0L) Offset: 0x80008460	109
Table 72:	IP Differentiated Services CodePoint to Priority0 High Register (DSC0H) Offset: 0x80008464	109
Table 73:	IP Differentiated Services CodePoint to Priority1 Low Register (DSC1L) Offset: 0x80008468	109
Table 74:	IP Differentiated Services CodePoint to Priority1 High Register (DSC1H) Offset: 0x8000846C	110
Table 75:	VLAN Priority Tag to Priority Register (VPT2P) Offset: 0x80008470	110
Table 76:	UniMAC First Rx Descriptor Pointer 0 Register (88E6218 Only) Offset: 0x80008480	110
Table 77:	UniMAC First Rx Descriptor Pointer 1 Register (88E6218 Only) Offset: 0x80008484	110
Table 78:	UniMAC First Rx Descriptor Pointer 2 Register Offset: 0x80008488	111
Table 79:	UniMAC First Rx Descriptor Pointer 3 Register Offset: 0x8000848C	111
Table 80:	UniMAC Current Rx Descriptor Pointer 0 Register (88E6218 Only) Offset: 0x800084A0	111

Table 81:	UniMAC Current Rx Descriptor Pointer 1 Register (88E6218 Only)	
	Offset: 0x800084A4	111
Table 82:	UniMAC Current Rx Descriptor Pointer 2 Register	
	Offset: 0x800084A8	111
Table 83:	UniMAC Current Rx Descriptor Pointer 3 Register	
	Offset: 0x800084AC	112
Table 84:	UniMAC Current Tx Descriptor Pointer 0 Register	
	Offset: 0x800084E0	112
Table 85:	UniMAC Current Tx Descriptor Pointer 1 Register (88E6218 Only)	
	Offset: 0x800084E4	112
A.5	Independent DMA Registers (88E6218 Only).....	113
Table 87:	Channel 0 DMA Byte Count Register	
	Offset: 0x8000E800	114
Table 88:	Channel 1 DMA Byte Count Register	
	Offset: 0x8000E804	114
Table 89:	Channel 0 DMA Source Address Register	
	Offset: 0x8000E810	114
Table 90:	Channel 1 DMA Source Address Register	
	Offset: 0x8000E814	114
Table 91:	Channel 0 DMA Destination Address Register	
	Offset: 0x8000E820	115
Table 92:	Channel 1 DMA Destination Address Register	
	Offset: 0x8000E824	115
Table 93:	Channel 0 Next Descriptor Pointer Register	
	Offset: 0x8000E830	115
Table 94:	Channel 1 Next Descriptor Pointer Register	
	Offset: 0x8000E834	115
Table 95:	Channel 0 Current Descriptor Pointer Register	
	Offset: 0x8000E870	115
Table 96:	Channel 1 Current Descriptor Pointer Register	
	Offset: 0x8000E874	116
Table 97:	Channel 0 Control Register	
	Offset: 0x8000E840	116
Table 98:	Channel 1 Control Register	
	Offset: 0x8000E844	118
Table 99:	Arbiter Control Register	
	Offset: 0x8000E860	120
Table 100:	Channel 0 Interrupt Mask Register	
	Offset: 0x8000E880	120
Table 101:	Channel 1 Interrupt Mask Register	
	Offset: 0x8000E884	120
Table 102:	Channel 0 Interrupt Reset Select Register	
	Offset: 0x8000E890	121
Table 103:	Channel 1 Interrupt Reset Select Register	
	Offset: 0x8000E894	121
Table 104:	Channel 0 Interrupt Status Register	
	Offset: 0x8000E8A0	121
Table 105:	Channel 1 Interrupt Status Register	
	Offset: 0x8000E8A4	121



A.6	Two-Wire Serial Interface Registers	122
Table 107:	TWSI Clock Prescaler Register Offset: 0x8000C100	122
Table 108:	TWSI Global Control Register Offset: 0x8000C104	122
Table 109:	TWSI Control Register Offset: 0x8000C110	123
Table 110:	TWSI Status Register Offset: 0x8000C11C	123
Table 111:	TWSI Interrupt Mask Register Offset: 0x8000C120	123
Table 112:	TWSI Data Register Offset: 0x8000C124	124
Table 113:	TWSI Interrupt Source Register Offset: 0x8000C12C	124
Table 114:	Peripheral Reset Register Offset: 0x8000C000	124
Table 115:	Peripheral Clock Divider Register Offset: 0x8000C004	124
A.7	General Purpose I/O Port (GPIO) Registers	126
Table 117:	GPIO Select[7:0] Register Offset: 0x8000D000	127
Table 118:	GPIO Select[15:8] Register Offset: 0x8000D004	128
Table 119:	Control Bidirectional GPIO Output Enable Pins Register Offset: 0x8000D008	129
Table 120:	GPIO Output Registers Offset: 0x8000D00C	129
Table 121:	GPIO Input Registers Offset: 0x8000D010	130
Table 122:	GPIO Interrupt Edge Trigger Select Register (IER) Offset: 0x8000D014	130
Table 123:	GPIO Input Interrupt Mask Register (IMR) Offset: 0x8000D018	130
Table 124:	GPIO Reset Select Register (RSR) Offset: 0x8000D01C	130
Table 125:	GPIO Interrupt Status Register (ISR) Offset: 0x8000D020	131
Table 126:	SW LED Output Enable Register Offset: 0x8000D024	131
Table 127:	SW LED Cycle[3:0] Register Offset: 0x8000D028	131
Table 128:	SW LED Cycle[7:4] Register Offset: 0x8000D02C	131
Table 129:	SW LED Cycle[11:8] Register Offset: 0x8000D030	132
Table 130:	SW LED Cycle[15:12] Register Offset: 0x8000D034	133
Table 131:	SW LED Duty Cycle[3:0] Register Offset: 0x8000D038	134
Table 132:	SW LED Duty Cycle[7:4] Register Offset: 0x8000D03C	134

Table 133:	SW LED Duty Cycle[11:8] Register Offset: 0x8000D040	135
Table 134:	SW LED Duty Cycle[15:12] Register Offset: 0x8000D044	136
Table 135:	LED Output Enable for Status Signals Register Offset: 0x8000D048	136
Table 136:	LED[3:0] Stretch Duration for Status Signals Register Offset: 0x8000D04C	137
Table 137:	Link/Active Cycle Register Offset: 0x8000D050	138
A.8	UART Registers	139
Table 139:	UART Base Clock Select Register Offset: 0x8000C008	139
Table 140:	Receive Buffer Register (RBR) Offset: 0x8000C840	140
Table 141:	Transmit Holding Register (THR) Offset: 0x8000C840	140
Table 142:	Divisor Latch Low (DLL) Register Offset: 0x8000C840	140
Table 143:	Interrupt Enable Register (IER) Offset: 0x8000C844	141
Table 144:	Divisor Latch High (DLH) Register Offset: 0x8000C844	141
Table 145:	Interrupt Identity Register (IIR) Offset: 0x8000C848	141
Table 146:	FIFO Control Register (FCR) Offset: 0x8000C848	142
Table 147:	Line Control Register (LCR) Offset: 0x8000C84C	143
Table 148:	Modem Control Register (MCR) Offset: 0x8000C850	143
Table 149:	Line Status Register (LSR) Offset: 0x8000C854	144
Table 150:	Scratch Pad Register (SCR) Offset: 0x8000C85C	144
A.9	Watchdog Register Description	145
Table 152:	Watchdog Enable Register Offset: 8000D800	145
Table 153:	Watchdog Timer Interval Register Offset: 8000D802	145
Table 154:	Watchdog Timer Counter Clear Register Offset: 8000D804	145



A.1 Register Overview

A.1.1 Register Description

All registers are 32-bits wide [31:0], except the UART registers, which are 8-bits wide. The 88E6218 and 88E6208 registers use Little Endian byte ordering 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 31 is the Most Significant Bit (MSb) and bit 0 is the Least Significant Bit (LSb).

A.1.2 Register Types

The 88E6208 and 88E6218 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:

Type	Description
RES	Reserved for future use. All reserved bits are read as zero unless otherwise noted.
RO	Read Only. Writing to this type of field may cause unpredictable results.
RW	Read and Write with initial value indicated.
RWR	Read/Write Reset. All bits are readable and writable. After reset 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. Reads to this type of register field return undefined data.

A.2 CPU Interface Registers

The CPU registers are 32 bits in length.

A.2.1 CPU Register Map

Table 31: CPU Register Map Table

Register Name	Offset	Table and Page
CPU Control		
System Configuration Registers	0x80002000	Table 32, p. 78
Shared Bus Arbiter Configuration Low	0x80002004	Table 33, p. 78
Shared Bus Arbiter Configuration High	0x80002008	Table 34, p. 79
CPU Interrupt		
Interrupt Request Status Register	0x90008000	Table 35, p. 79
Interrupt Source Status Register	0x90008004	Table 36, p. 80
Interrupt Enable Set Register	0x90008008	Table 37, p. 81
Interrupt Enable Register	0x90008008	Table 38, p. 82
Interrupt Enable Clear Register	0x9000800C	Table 39, p. 83
Software Programmed Interrupt Register	0x90008010	Table 40, p. 84
Fast Interrupt Request Status Register	0x90008100	Table 41, p. 84
Fast Interrupt Source Status Register	0x90008104	Table 42, p. 84
Fast Interrupt Enable Set Register	0x90008108	Table 43, p. 84
Fast Interrupt Enable Register	0x90008108	Table 44, p. 84
Fast Interrupt Enable Clear Register	0x9000810C	Table 45, p. 85
CPU Timer		
Timer x Length Register	Timer 1 0x90009000, Timer 2 0x90009004, Timer 3 0x90009008, Timer 4 0x9000900C	Table 46, p. 85
Timer Control Register	0x90009010	Table 47, p. 85
Timer x Value Register	Timer 1 0x90009014, Timer 2 0x90009018, Timer 3 0x9000901C, Timer 4 0x90009020	Table 48, p. 87
Interrupt Request Register	0x90009024	Table 49, p. 88
Interrupt Mask Register	0x90009028	Table 50, p. 88

A.2.2 CPU Control Registers

Table 32: System Configuration Registers
Offset: 0x80002000

Bits	Field	Type/ InitVal	Description
31:7	Reserved	RES	Reserved
6	PcbOp[1]	RO	This bit shows the reset sample status of pin M_A[7].
5	PcbOp[0]	RO	This bit shows the reset sample status of pin M_A[6].
4	Reserved	RES	Reserved
3	BootROMWidth	RO	This bit shows the width of the boot ROM as sample at reset pin M_DP0/ M_A18.
2:0	Reserved	RES	Reserved

Table 33: Shared Bus Arbiter Configuration Low
Offset: 0x80002004

Bits	Field	Type/ InitVal	Description
31:28	Arb7	RW 0x7	Slice 7 owner of the arbiter.
27:24	Arb6	RW 0x6	Slice 6 owner of the arbiter.
23:20	Arb5	RW 0x5	Slice 5 owner of the arbiter.
19:16	Arb4	RW 0x4	Slice 4 owner of the arbiter.
15:12	Arb3	RW 0x3	Slice 3 owner of the arbiter.
11:8	Arb2	RW 0x2	Slice 2 owner of the arbiter.
7:4	Arb1	RW 0x1	Slice 1 owner of the arbiter.
3:0	Arb0	RW 0x0	Slice 0 owner of the arbiter 0 = Host interface unit 1 = CPU interface unit 2 = Reserved 3 = Memory interface unit 4 = Fast ethernet unit 5 = Reserved 6 = Slow bus unit 7 = DMA control unit

Table 34: Shared Bus Arbiter Configuration High
Offset: 0x80002008

Bits	Field	Type/ InitVal	Description
31:28	Arb15	RW 0x7	Slice 15 owner of the arbiter.
27:24	Arb14	RW 0x6	Slice 14 owner of the arbiter.
23:20	Arb13	RW 0x5	Slice 13 owner of the arbiter.
19:16	Arb12	RW 0x4	Slice 12 owner of the arbiter.
15:12	Arb11	RW 0x3	Slice 11 owner of the arbiter.
11:8	Arb10	RW 0x2	Slice 10 owner of the arbiter.
7:4	Arb9	RW 0x1	Slice 9 owner of the arbiter.
3:0	Arb8	RW 0x0	Slice 8 owner of the arbiter.

A.2.3 CPU Interrupt Registers

Table 35: Interrupt Request Status Register
Offset: 0x90008000

Bits	Field Name	Type/ InitVal	Description
31:17	Reserved	RES	Reserved
16	Dma1Int	RO 0x0	1 = DMA Unit Channel 1 Interrupt request active. 0 = DMA Unit Channel 1 Interrupt request inactive. NOTE: This bit only applies to the 88E6218.
15	Dma0Int	RO 0x0	1 = DMA Unit Channel 0 Interrupt request active. 0 = DMA Unit Channel 0 Interrupt request inactive. NOTE: This bit only applies to the 88E6218.
14:13	Reserved	RES	Reserved
12	GpioIntRq	RO 0x0	1 = GPIO Interrupt request active. 0 = GPIO Interrupt request inactive.
11	UartIntRq	RO 0x0	1 = UART Interrupt request active. 0 = UART Interrupt request inactive.

Table 35: Interrupt Request Status Register (Continued)
Offset: 0x90008000

Bits	Field Name	Type/ InitVal	Description
10	SwitGlob	RO 0x0	1 = Switch Global Interrupt request active. 0 = Switch Global Interrupt request inactive.
9	FEMAC	RO 0x0	1 = UniMAC™ Interrupt request active. 0 = UniMAC Interrupt request inactive.
8	Reserved	RES	Reserved
7	Time4Int	RO 0x0	1 = Timer 4 Interrupt request active. 0 = Timer 4 Interrupt request inactive.
6	Time3Int	RO 0x0	1 = Timer 3 Interrupt request active. 0 = Timer 3 Interrupt request inactive.
5	Time2Int	RO 0x0	1 = Timer 2 Interrupt request active. 0 = Timer 2 Interrupt request inactive.
4	Time1Int	RO 0x0	1 = Timer 1 Interrupt request active. 0 = Timer 1 Interrupt request inactive.
3:2	Reserved	RES	Reserved
1	Proglnt	RO 0x0	1 = Programmed Interrupt request active. 0 = Programmed Interrupt request inactive.
0	FiqInt	RO 0x0	1 = FIQ Interrupt request active. 0 = FIQ Interrupt request inactive.

Table 36: Interrupt Source Status Register
Offset: 0x90008004

Bits	Field Name	Type/ InitVal	Description
31:17	Reserved	RES	Reserved
16	Dma1Int	RO 0x0	1 = DMA Unit Channel 1 Interrupt source active. 0 = DMA Unit Channel 1 Interrupt source inactive. NOTE: This bit only applies to the 88E6218.
15	Dma0Int	RO 0x0	1 = DMA Unit Channel 0 Interrupt source active. 0 = DMA Unit Channel 0 Interrupt source inactive. NOTE: This bit only applies to the 88E6218.
14:13	Reserved	RES	Reserved
12	GpioIntRq	RO 0x0	1 = GPIO Interrupt source active. 0 = GPIO Interrupt source inactive.
11	UartIntRq	RO 0x0	1 = UART Interrupt source active. 0 = UART Interrupt source inactive.
10	SwitGlob	RO 0x0	1 = Switch Global Interrupt source active. 0 = Switch Global Interrupt source inactive.
9	FEMAC	RO 0x0	1 = UniMAC™ Interrupt source active. 0 = UniMAC Interrupt source inactive.
8	Reserved	RES	Reserved

Table 36: Interrupt Source Status Register (Continued)
Offset: 0x90008004

Bits	Field Name	Type/ InitVal	Description
7	Time4Int	RO 0x0	1 = Timer 4 Interrupt source active. 0 = Timer 4 Interrupt source inactive.
6	Time3Int	RO 0x0	1 = Timer 3 Interrupt source active. 0 = Timer 3 Interrupt source inactive.
5	Time2Int	RO 0x0	1 = Timer 2 Interrupt source active. 0 = Timer 2 Interrupt source inactive.
4	Time1Int	RO 0x0	1 = Timer 1 Interrupt source active. 0 = Timer 1 Interrupt source inactive.
3:2	Reserved	RES	Reserved
1	ProgInt	RO 0x0	1 = Programmed Interrupt source active. 0 = Programmed Interrupt source inactive.
0	FiqInt	RO 0x0	1 = FIQ Interrupt source active. 0 = FIQ Interrupt source inactive.

Table 37: Interrupt Enable Set Register
Offset: 0x90008008

Bits	Field Name	Type/ InitVal	Description
31:17	Reserved	RES	Reserved
16	Dma1Int	WO 0x0	1 = DMA Unit Channel 1 Interrupt Enable Register will be set to 1. 0 = DMA Unit Channel 1 Interrupt Enable Register will not be changed. NOTE: This bit only applies to the 88E6218.
15	Dma0Int	WO 0x0	1 = DMA Unit Channel 0 Interrupt Enable Register will be set to 1. 0 = DMA Unit Channel 0 Interrupt Enable Register will not be changed. NOTE: This bit only applies to the 88E6218.
14:13	Reserved	RES	Reserved
12	GpioIntRq	WO 0x0	1 = GPIO Interrupt Enable Register will be set to 1. 0 = GPIO Interrupt Enable Register will not be changed.
11	UartIntRq	WO 0x0	1 = UART Interrupt Enable Register will be set to 1. 0 = UART Interrupt Enable Register will not be changed.
10	SwitGlob	WO 0x0	1 = Switch Global Interrupt Enable Register will be set to 1. 0 = Switch Global Interrupt Enable Register will not be changed.
9	FEMAC	WO 0x0	1 = UniMAC™ Interrupt Enable Register will be set to 1. 0 = UniMAC Interrupt Enable Register will not be changed.
8	Reserved	RES	Reserved
7	Time4Int	WO 0x0	1 = Timer 4 Interrupt Enable Register will be set to 1. 0 = Timer 4 Interrupt Enable Register will not be changed.
6	Time3Int	WO 0x0	1 = Timer 3 Interrupt Enable Register will be set to 1. 0 = Timer 3 Interrupt Enable Register will not be changed.
5	Time2Int	WO 0x0	1 = Timer 2 Interrupt Enable Register will be set to 1. 0 = Timer 2 Interrupt Enable Register will not be changed.

Table 37: Interrupt Enable Set Register (Continued)
Offset: 0x90008008

Bits	Field Name	Type/ InitVal	Description
4	Time1Int	WO 0x0	1 = Timer 1 Interrupt Enable Register will be set to 1. 0 = Timer 1 Interrupt Enable Register will not be changed.
3:2	Reserved	RES	Reserved
1	ProgInt	WO 0x0	1 = Programmed Interrupt Enable Register will be set to 1. 0 = Programmed Interrupt Enable Register will not be changed.
0	FiqInt	WO 0x0	1 = FIQ Interrupt Enable Register will be set to 1. 0 = FIQ Interrupt Enable Register will not be changed.

Table 38: Interrupt Enable Register
Offset: 0x90008008

Bits	Field Name	Type/ InitVal *	Description
31:17	Reserved	RES	Reserved
16	Dma1Int	RO 0x0	1 = DMA Unit Channel 1 Interrupt is enabled. 0 = DMA Unit Channel 1 Interrupt is disabled. NOTE: This bit only applies to the 88E6218.
15	Dma0Int	RO 0x0	1 = DMA Unit Channel 0 Interrupt is enabled. 0 = DMA Unit Channel 0 Interrupt is disabled. NOTE: This bit only applies to the 88E6218.
14:13	Reserved	RES	Reserved
12	GpioIntRq	RO 0x0	1 = GPIO Interrupt is enabled. 0 = GPIO Interrupt is disabled.
11	UartIntRq	RO 0x0	1 = UART Interrupt is enabled. 0 = UART Interrupt is disabled.
10	SwitGlob	RO 0x0	1 = Switch Global Interrupt is enabled. 0 = Switch Global Interrupt is disabled.
9	FEMAC	RO 0x0	1 = UniMAC Interrupt is enabled. 0 = UniMAC Interrupt is disabled.
8	Reserved	RES	Reserved
7	Time4Int	RO 0x0	1 = Timer 4 Interrupt is enabled. 0 = Timer 4 Interrupt is disabled.
6	Time3Int	RO 0x0	1 = Timer 3 Interrupt is enabled. 0 = Timer 3 Interrupt is disabled.
5	Time2Int	RO 0x0	1 = Timer 2 Interrupt is enabled. 0 = Timer 2 Interrupt is disabled.
4	Time1Int	RO 0x0	1 = Timer 1 Interrupt is enabled. 0 = Timer 1 Interrupt is disabled.
3:2	Reserved	RES	Reserved
1	ProgInt	RO 0x0	1 = Programmed Interrupt is enabled. 0 = Programmed Interrupt is disabled.

Table 38: Interrupt Enable Register (Continued)
Offset: 0x90008008

Bits	Field Name	Type/ InitVal	Description
0	FiqInt	RO 0x0	1 = FIQ Interrupt is enabled. 0 = FIQ Interrupt is disabled.

Table 39: Interrupt Enable Clear Register
Offset: 0x9000800C

Bits	Field Name	Type/ InitVal	Description
31:17	Reserved	RES	Reserved
16	Dma1Int	WO 0x0	1 = DMA Unit Channel 1 Interrupt Enable Register will be set to 0. 0 = DMA Unit Channel 1 Interrupt Enable Register will not be changed. NOTE: This bit only applies to the 88E6218.
15	Dma0Int	WO 0x0	1 = DMA Unit Channel 0 Interrupt Enable Register will be set to 0. 0 = DMA Unit Channel 0 Interrupt Enable Register will not be changed. NOTE: This bit only applies to the 88E6218.
14:13	Reserved	RES	Reserved
12	GpioIntRq	WO 0x0	1 = GPIO Interrupt Enable Register will be set to 0. 0 = GPIO Interrupt Enable Register will not be changed.
11	UartIntRq	WO 0x0	1 = UART Interrupt Enable Register will be set to 0. 0 = UART Interrupt Enable Register will not be changed.
10	SwitGlob	WO 0x0	1 = Switch Global Interrupt Enable Register will be set to 0. 0 = Switch Global Interrupt Enable Register will not be changed.
9	FEMAC	WO 0x0	1 = UniMAC™ Interrupt Enable Register will be set to 0. 0 = UniMAC Interrupt Enable Register will not be changed.
8	Reserved	RES	Reserved
7	Time4Int	WO 0x0	1 = Timer 4 Interrupt Enable Register will be set to 0. 0 = Timer 4 Interrupt Enable Register will not be changed.
6	Time3Int	WO 0x0	1 = Timer 3 Interrupt Enable Register will be set to 0. 0 = Timer 3 Interrupt Enable Register will not be changed.
5	Time2Int	WO 0x0	1 = Timer 2 Interrupt Enable Register will be set to 0. 0 = Timer 2 Interrupt Enable Register will not be changed.
4	Time1Int	WO 0x0	1 = Timer 1 Interrupt Enable Register will be set to 0. 0 = Timer 1 Interrupt Enable Register will not be changed.
3:2	Reserved	RES	Reserved
1	ProgInt	WO 0x0	1 = Programmed Interrupt Enable Register will be set to 0. 0 = Programmed Interrupt Enable Register will not be changed.
0	FiqInt	WO 0x0	1 = FIQ Interrupt Enable Register will be set to 0. 0 = FIQ Interrupt Enable Register will not be changed.



Table 40: Software Programmed Interrupt Register
Offset: 0x90008010

Bits	Field Name	Type/ InitVal	Description
31:2	RES	RES	Reserved
1	SWProgInt	WO 0x0	1 = Software Programmed Interrupt request active. 0 = Software Programmed Interrupt request inactive.
0	RES	RES	Reserved

Table 41: Fast Interrupt Request Status Register
Offset: 0x90008100

Bits	Field Name	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	FiqRqSrc	RO 0x0	1 = FIQ Interrupt request active. 0 = FIQ Interrupt request inactive.

Table 42: Fast Interrupt Source Status Register
Offset: 0x90008104

Bits	Field Name	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	FiqSrcAct	RO 0x0	1 = FIQ Interrupt source active. 0 = FIQ Interrupt source inactive.

Table 43: Fast Interrupt Enable Set Register
Offset: 0x90008108

Bits	Field Name	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	FiqIntEn	WO 0x0	1 = FIQ Interrupt Enable Register will be set to 1. 0 = FIQ Interrupt Enable Register will not be changed.

Table 44: Fast Interrupt Enable Register
Offset: 0x90008108

Bits	Field Name	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	FiqFastEn	RO 0x0	1 = FIQ Interrupt is enabled. 0 = FIQ Interrupt is disabled.

Table 45: Fast Interrupt Enable Clear Register
Offset: 0x9000810C

Bits	Field Name	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	FiqClrInEn	WO 0x0	1 = FIQ Interrupt Enable Register will be set to 0. 0 = FIQ Interrupt Enable Register will not be changed.

A.2.4 CPU Timer Registers

Table 46: Timer x Length Register
Offset: Timer 1 0x90009000, Timer 2 0x90009004, Timer 3 0x90009008, Timer 4 0x9000900C

Bits	Field Name	Type/ InitVal	Description
31:16	Reserved	RES	Reserved When writing to this register, write 0 to this field. When reading this register, this field will be undefined.
15:0	TimeXLen	RW 0x0	Sets the length of Timer x.

Table 47: Timer Control Register
Offset: 0x90009010

Bits	Field Name	Type/ InitVal	Description
31:16	Reserved	RES	Reserved When writing to this register, write 0 to this field. When reading this field will be undefined.
15	Timer4Ld	RW 0x0	Timer 4 Load Enables the counter to reload its value from the Timer 4 Length register. 0 = Disabled 1 = Enabled
14	Timer4Int	RW 0x0	Enables the Timer 4 interrupt to the CPU. 0 = Disabled 1 = Enabled
13	Timer4AD	RW 0x0	Activates and Deactivates Timer 4. 0 = Deactivates (stops) Timer 4 1 = Activates (starts) Timer 4 The Timer 4 stops after reaching zero count and generating an interrupt, if the Timer4AA bit is set to '0'. The Timer 4 remains activated, reloads, and continues to decrement if the Timer4AA bit is set to '1'.



Table 47: Timer Control Register (Continued)
Offset: 0x90009010

Bits	Field Name	Type/ InitVal	Description
12	Timer4AA	RW 0x0	Automatic Reset When this bit is set Timer 4 automatically resets itself and continues to decrement after reaching zero count and generating an interrupt or stops a zero and waits for software to enable it. This bit sets bits in an interrupt enable register. When writing to this bit a '1' sets the corresponding bit in the enable register. Writing a '0' has no effect on the enable register.
11	Timer3Ld	RW 0x0	Timer 3 Load Enables the counter to reload its value from the Timer 3 Length register. 0 = Disabled 1 = Enabled
10	Timer3Int	RW 0x0	Enables the Timer 3 interrupt to the CPU. 0 = Disabled 1 = Enabled
9	Timer3AD	RW 0x0	Activates and Deactivates Timer 3. 0 = Deactivates (stops) Timer 3 1 = Activates (starts) Timer 3 The Timer 3 stops after reaching zero count and generating an interrupt, if the Timer3AA bit is set to '0'. The Timer 3 remains activated, reloads, and continues to decrement if the Timer3AA bit is set to '1'.
8	Timer3AA	RW 0x0	Automatic Reset When this bit is set Timer 3 automatically resets itself and continues to decrement after reaching zero count and generating an interrupt or stops a zero and waits for software to enable it. This bit sets bits in an interrupt enable register. When writing to this bit a '1' sets the corresponding bit in the enable register. Writing a '0' has no effect on the enable register.
7	Timer2Ld	RW 0x0	Timer 2 Load Enables the counter to reload its value from the Timer 2 Length register. 0 = Disabled 1 = Enabled
6	Timer2Int	RW 0x0	Enables the Timer 2 interrupt to the CPU. 0 = Disabled 1 = Enabled
5	Timer2AD	RW 0x0	Activates and Deactivates Timer 2. 0 = Deactivates (stops) Timer 2 1 = Activates (starts) Timer 2 The Timer 2 stops after reaching zero count and generating an interrupt, if the Timer2AA bit is set to '0'. The Timer 2 remains activated, reloads, and continues to decrement if the Timer2AA bit is set to '1'.

Table 47: Timer Control Register (Continued)
Offset: 0x90009010

Bits	Field Name	Type/ InitVal	Description
4	Timer2AA	RW 0x0	Automatic Reset When this bit is set Timer 2 automatically resets itself and continues to decrement after reaching zero count and generating an interrupt or stops a zero and waits for software to enabled it. This bit sets bits in an interrupt enable register. When writing to this bit a '1' sets the corresponding bit in the enable register. Writing a '0' has no effect on the enable register.
3	Timer1Ld	RW 0x0	Timer 1 Load Enables the counter to reload its value from the Timer 1 Length register. 0 = Disabled 1 = Enabled
2	Timer1Int	RW 0x0	Enables the Timer 1 interrupt to the CPU. 0 = Disabled 1 = Enabled
1	Timer1AD	RW 0x0	Activates and Deactivates Timer 1. 0 = Deactivates (stops) Timer 1 1 = Activates (starts) Timer 1 The Timer 1 stops after reaching zero count and generating an interrupt, if the Timer1AA bit is set to '0'. The Timer 1 remains activated, reloads, and continues to decrement if the Timer1AA bit is set to '1'.
0	Timer1AA	RW 0x0	Automatic Reset When this bit is set Timer 1 automatically resets itself and continues to decrement after reaching zero count and generating an interrupt or stops a zero and waits for software to enabled it. This bit sets bits in an interrupt enable register. When writing to this bit a '1' sets the corresponding bit in the enable register. Writing a '0' has no effect on the enable register.

Table 48: Timer x Value Register
Offset: Timer 1 0x90009014, Timer 2 0x90009018, Timer 3 0x9000901C, Timer 4 0x90009020

Bits	Field Name	Type/ InitVal	Description
31:16	Reserved	RO	Reserved When reading this register, this field will be undefined.
15:0	TimeXLen	RO 0xFFFF	Gives the current value of the timer.

Table 49: Interrupt Request Register
Offset: 0x90009024

Bits	Field Name	Type/ InitVal	Description
31:4	Reserved	RW	Reserved
3	IntStatAf4	RW 0x0	The field provides the status of the interrupt sources for Timer 4 after masking. 0 = Interrupt for Timer 4 is inactive. 1 = Interrupt for Timer 4 is active and will generate an interrupt to the interrupt controller. NOTE: A write of either 0 or 1 clears this bit.
2	IntStatAf3	RW 0x0	The field provides the status of the interrupt sources for Timer 3 after masking. 0 = Interrupt for Timer 3 is inactive. 1 = Interrupt for Timer 3 is active and will generate an interrupt to the interrupt controller. NOTE: A write of either 0 or 1 clears this bit.
1	IntStatAf2	RW 0x0	The field provides the status of the interrupt sources for Timer 2 after masking. 0 = Interrupt for Timer 2 is inactive. 1 = Interrupt for Timer 2 is active and will generate an interrupt to the interrupt controller. NOTE: A write of either 0 or 1 clears this bit.
0	IntStatAf1	RW 0x0	The field provides the status of the interrupt sources for Timer 1 after masking. 0 = Interrupt for Timer 1 is inactive. 1 = Interrupt for Timer 1 is active and will generate an interrupt to the interrupt controller. NOTE: A write of either 0 or 1 clears this bit.

Table 50: Interrupt Mask Register
Offset: 0x90009028

Bits	Field Name	Type/ InitVal	Description
31:4	Reserved	RW	Reserved
3	InStatBfT4	RW 0x0	The field provides masking for the interrupt sources for Timer 4. 0 = Interrupt for Timer 4 is masked. 1 = Interrupt for Timer 4 is not masked.
2	InStatBfT3	RW 0x0	The field provides masking for the interrupt sources for Timer 3. 0 = Interrupt for Timer 3 is masked. 1 = Interrupt for Timer 3 is not masked.
1	InStatBfT2	RW 0x0	The field provides masking for the interrupt Sources for Timer 2. 0 = Interrupt for Timer 2 is masked. 1 = Interrupt for Timer 2 is not masked.
0	InStatBfT1	RW 0x0	The field provides masking for the interrupt Sources for Timer 1. 0 = Interrupt for Timer 1 is masked. 1 = Interrupt for Timer 1 is not masked.

A.3 Memory Controller Registers

A.3.1 Memory Controller Register Map

The 88E6218 and 88E6208 device supports three memory devices besides the Boot device. Via the Configuration Register x (CFGRx), the user has the option to select the types of memory devices. Both SDRAM and asynchronous chip select devices are supported.

Each chip select references a Configuration register and two Timing Parameter registers (see [Table 53, p. 90](#) through [Table 56, p. 94](#)).

[Table 51](#) provides the Memory Controller register map.

Refer to the SDRAM Timing Parameter registers or the asynchronous chip select (Flash/ROM) Timing Parameter registers based on the type of memory connected to the bus.

Table 51: Memory Controller Register Map

Register Name	Offset	Table and Page
Control and Status Register	0x80006000	Table 52, p. 90
Configuration Register x (CFGRx)	CFGR0: 0x80006004, CFGR1: 0x80006010, CFGR2: 0x8000601C, CFGR3: 0x80006028	Table 53, p. 90
Timing Parameter (SDRAM) Register 0	M_CS0n: 0x80006014, M_CS1n: 0x80006020, M_CS2n: 0x8000602C	Table 54, p. 92
Timing Parameter (SDRAM) Register 1	M_SC0n: 0x80006018, M_SC1n: 0x80006024, M_SC2n: 0x80006030	Table 55, p. 93
Timing Parameter (Asynchronous Device) Register 0	BootCSn: 0x80006008, M_CS0n: 0x80006014, M_CS1n: 0x80006020, M_CS2n: 0x8000602C	Table 56, p. 94

A.3.2 Control and Status Register

Table 52: Control and Status Register
Offset: 0x80006000

Bits	Field	Type/ InitVal	Description
31:18	RefInt	RW 0x0157	Refresh Interval (SDRAM only) The number of clock cycles from one refresh command to the next refresh command. The number is obtained by dividing the SDRAM refresh period (Tref) by the total number of rows to be refreshed and then dividing the result by the clock cycle time. The default value = 14'd781 (0b00001100001101) For example: 8,192/64 ms using a 100 MHz clock $\frac{64 \text{ ms} * 100 \text{ MHz}}{8192} = 781.25$
17:16	Reserved	RES	Reserved
15	CS	RO 0x1	Chip Select. This register indicates if any of the 4 memory devices is select. 0 = No memory device is selected 1 = Any memory device is selected
14:1	Reserved	RES	These bits must be set to 0.
0	MStatus	RO 0x0	This bit returns the current value of the M_STATUS pin. Software can use this register to determine that the last Flash device written to is available. Note: This bit is not applicable to the 88E6208.

A.3.3 Configuration Register for SDRAM/Device

Table 53: Configuration Register x (CFGRx)
Offset: CFGR0: 0x80006004, CFGR1: 0x80006010, CFGR2: 0x8000601C, CFGR3: 0x80006028

NOTE: Writes to this register cause an SDRAM initialization to occur. This register should not be written unless the Timing Parameter (SDRAM) 0/1 registers in Table 54 on page 92 and Table 55 on page 93 have been written first.

Bits	Field	Type/InitVal	Description
31:24	BAR	RW CFGR0: 0xFF CFGR1: 0xE0 CFGR2: 0xD0 CFGR3: 0xC0	Base Address for the Memory Device Bank This field is compared with address bits 29:22. NOTE: Address bits 31:30 are not compared.

Table 53: Configuration Register x (CFGRx) (Continued)**Offset: CFGR0: 0x80006004, CFGR1: 0x80006010, CFGR2: 0x8000601C, CFGR3: 0x80006028****NOTE:** Writes to this register cause an SDRAM initialization to occur. This register should not be written unless the Timing Parameter (SDRAM) 0/1 registers in [Table 54 on page 92](#) and [Table 55 on page 93](#) have been written first.

Bits	Field	Type/InitVal	Description
23:16	BARMsk	RW 0xFF	Base Address Mask for the Memory Device Bank Writing a 0 to this register's bits masks the corresponding bits' position. It is ANDed with Addr bits 29:22 and the result is compared with the base address for the memory controller to select the corresponding memory device bank. (from 4 MB to 1 GB) NOTE: The default is 0xFF (4 MB).
15	Reserved	RES	Must be 0.
14:13	WordSize	CFGR0 8-bit: 0x01 16-bit: 0x10 ¹ CFGR1,2,3: 0x10	Configured word size for the device bank 00 = Reserved 01 = X8 word size 10 = X16 word size 11 = X32 word size (88E6218 only)
12	KeepRwOpen	RW 0x0	Keep Row Open (SDRAM only) 0 = Keep row closed after read/write. 1 = Keep row opened after read/write.
11	Reserved	RES	Must be 0.
10	WrtProtect	RW 0x0	Write Protect (Flash only, depends on Flash) This bit's value goes out the CASn/WPn pin if this Chip Select is configured for Flash and this Chip Select's address is being accessed. Each Chip Select has its own WrtProtect register bit that can be independently set for that Chip Select. 0 = Write protect is disabled. 1 = Write protect is enabled.
9	ByteMode	CFGR0 8-bit: 0x0 16-bit: 0x1 ¹ CFGR1,2,3: 0x1	Byte mode or Word mode (Flash only) This bit's value goes out the DQMn[0]/BYTEn pin if this Chip Select is configured for Flash and this Chip Select's address is being accessed. Each Chip Select has its own ByteMode register bit that can be independently set for that Chip Select. ByteMode for Configuration Register 0 is configured at reset by the inversion of the M_A[18] configuration pin (Flash boot width). 0 = Byte mode (X8 bit) 1 = Word mode (X16 bit)
8	Reserved	RW 0x0	Must be 0.
7	PackEn	RW 0x1	For Asynchronous chip device: 0 = Disable Packing the data to 32 bits 1 = Enable Packing the data to 32 bits For SDRAM: 0 = Supports only X32 bus SDRAM 1 = Supports X16, or X32 word size (88E6218 only)

Table 53: Configuration Register x (CFGRx) (Continued)

Offset: CFGR0: 0x80006004, CFGR1: 0x80006010, CFGR2: 0x8000601C, CFGR3: 0x80006028

NOTE: Writes to this register cause an SDRAM initialization to occur. This register should not be written unless the Timing Parameter (SDRAM) 0/1 registers in [Table 54 on page 92](#) and [Table 55 on page 93](#) have been written first.

Bits	Field	Type/InitVal	Description
6:5	MemSize	RW 0x00	Memory Size for <i>Single</i> SDRAM (SDRAM only) 00 = 16 Mb 01 = 64 Mb 10 = 128 Mb 11 = Reserved
4:3	MemType	RW 0x00	Memory Type 00 = Flash/ROM 01 = Reserved 10 = SDRAM 11 = Reserved
2:1	BusWidth	RW CFGR0 8-bit: 0x01 16-bit: 0x10 ¹ CFGR1,2,3: 0x10	Bus Width For SDRAM it indicated the data bus width for single SDRAM device, total bus width for the SDRAM can be configured to 16 bits or 32 bit (see CFGRn[14:13]). For asynchronous chip select device, it indicated the data bus width for the attached device. 00 = Reserved 01 = 8 bits bus 10 = 16 bits bus 11 = 32 bits bus (88E6218 only)
0	ChipSelectEn	RW 0x1	Chip select enable 0 = Chip select is disabled. 1 = Chip select is enabled.

¹ CFGR0 Initial value is dependent on the boot device width setting at reset as implemented on pin M_A[18].

A.3.4 Timing Parameter Registers

Table 54: Timing Parameter (SDRAM) Register 0

Offset: M_CS0n: 0x80006014, M_CS1n: 0x80006020, M_CS2n: 0x8000602C

NOTE: This description of the register is used when configuring an SDRAM device on the relevant chip select. M_CS2n, at offset 0x8000602C, does not apply to the 88E6208.

Bits	Field	Type/InitVal	Description
31:15	Reserved	RES	Reserved
14:13	BALMR	RW 0x00	These bits drive the BA[1:0] pins during the SDRAM Load command that occurs during SDRAM initialization. 00 = Default value

Table 54: Timing Parameter (SDRAM) Register 0 (Continued)**Offset: M_CS0n: 0x80006014, M_CS1n: 0x80006020, M_CS2n: 0x8000602C****NOTE:** This description of the register is used when configuring an SDRAM device on the relevant chip select.
M_CS2n, at offset 0x8000602C, does not apply to the 88E6208.

Bits	Field	Type/ InitVal	Description
12:10	Reserved	RES	Reserved
9	WrtBurstMode	RW 0x0	Write Burst Mode 0 = Programmed Burst Length 1 = Single Location Access
8:7	OpMode	RW 0x00	Operation Mode 00 = Standard mode 01 - 11 = Reserved
6:4	CASLat	RW 010	CAS Latency 000 = Reserved 001 = Reserved 010 = 2 cycles 011 = 3 cycles 100 - 111 = Reserved
3	Reserved	RES	Must be 0.
2:0	BurstLen	RW 0x010	Burst Length 000 = 1 001 = 2 010 = 4 011 = 8 100-110 = Reserved 111 = Full Page

Table 55: Timing Parameter (SDRAM) Register 1**Offset: M_SC0n: 0x80006018, M_SC1n: 0x80006024, M_SC2n: 0x80006030****NOTE:** This description of the register is used when configuring an SDRAM device on the relevant chip select.

Bits	Field	Type/ InitVal	Description
31:28	Reserved	RES	Reserved
27:24	RefActCmdDly	RW 0x1000	Refresh to Active Command Delay. The number of clock cycles needed for SDRAM to recover from a refresh command to be ready to take the next command. It is also the number of clock cycles needed for SDRAM to recover from executing one active command and be ready to accept the next active command. The value is obtained by dividing the Trc by the clock cycles.
23:20	PreCmdDly	RW 0x0011	Precharge to next Command delay.
19:16	Ref2CmdDly	RW 0x1000	Exit Self Refresh to next command delay.



Table 55: Timing Parameter (SDRAM) Register 1 (Continued)

Offset: M_SC0n: 0x80006018, M_SC1n: 0x80006024, M_SC2n: 0x80006030

NOTE: This description of the register is used when configuring an SDRAM device on the relevant chip select.

Bits	Field	Type/ InitVal	Description
15:12	Ld2ActRefCmd Dly	RW 0x0011	Load Mode Registers command to Active or Refresh command delay
11:8	ActCmdPreDly	RW 0x0110	Active Command to Precharge Delay
7:4	ActRWDly	RW 0x0011	Active Command to Read or Write Delay (RASn to CASn delay)
3:0	WrtRecTime	RW 0x0010	Write Recovery Time The number of clock cycles needed for SDRAM to recover from a write com- mand and be able to accept a precharge command

Table 56: Timing Parameter (Asynchronous Device) Register 0

Offset: BootCSn: 0x80006008, M_CS0n: 0x80006014, M_CS1n: 0x80006020,
M_CS2n: 0x8000602C

NOTE: This description of the register is used when connecting an asynchronous device to the relevant chip select.
M_CS2n, at offset 0x8000602C, does not apply to the 88E6208.

Bits	Field	Type/ InitVal	Description
31:24	TWB	RW 0x0000 1000	WEn high pulse width if the write is the last write in a multi-word transfer. Refer to the Flash/ROM Write Access diagram, in the electrical section of Part 1 of the three part 88E6208/88E6218 datasheet set (Document Num- ber MV-S101300-01).
23:20	TWPH	RW 0x0110	CEn high pulse width. Refer to the Flash/ROM Write Access diagram, in the electrical section of Part 1 of the three part 88E6208/88E6218 datasheet set (Document Num- ber MV-S101300-01).
19:16	TWH	RW 0x0010	CEn hold time from WEn high. TWH is the hold time the device drives the write data, Address and CEn, after WEn goes high. Refer to the Flash/ROM Write Access diagram, in the electrical section of Part 1 of the three part 88E6208/88E6218 datasheet set (Document Num- ber MV-S101300-01).
15:12	TWP	RW 0x1100	WEn pulse width Flash write cycle is TWP+TWH+TWPH except for the last write where it is TWP+TWH+TWB. Write data out of the device is driven at the same time the Address, CEn, and WEn are driven (OEn is high for at least TOD-1 prior to the start of a write). The data out, Address and CEn stay driven for TWP+TWH. Refer to the Flash/ROM Write Access diagram, in the electrical section of Part 1 of the three part 88E6208/88E6218 datasheet set (Document Num- ber MV-S101300-01).

Table 56: Timing Parameter (Asynchronous Device) Register 0 (Continued)
Offset: BootCSn: 0x80006008, M_CS0n: 0x80006014, M_CS1n: 0x80006020,
M_CS2n: 0x8000602C

NOTE: This description of the register is used when connecting an asynchronous device to the relevant chip select. M_CS2n, at offset 0x8000602C, does not apply to the 88E6208.

Bits	Field	Type/ InitVal	Description
11:8	TOD	RW 0x0010	OEn or CEn high to output High-z The TOD value-1 is the time the memory controller keeps CEn and OEn high after a Flash read before it will start the next memory controller operation (be it either an SDRAM or a Flash access). This parameter prevents data bus contention by allowing the Flash to enter tri-state in case the next cycle is a write cycle. This value needs to be set to the tri-state turn off time of the Flash that is installed in the system plus 1. The frequency of the CPU needs to be taken into account. Refer to the Flash/ROM Read Access diagram, in the electrical section of Part 1 of the three part 88E6208/88E6218 datasheet set (Document Number MV-S101300-01).
7:0	TAA	RW 0x00011 100	Read address to valid data delay (also CEn/OEn insert to valid data delay) Default setting: 8'b00011100 This value must be less than 251 since MDU internally adds 4 to it to compensate for the Flops pipeline latency; to make sure it is not overflowed, TAA should be programmed to less than 251. The TAA value+2 is the time from Address, CEn, and OEn low to data being latched inside the device. This parameter needs to be set to the TAA value of the Flash installed. The two cycle increase is needed for Address valid delay on the pins and the data setup time back inside this device. The frequency of the CPU needs to be taken into account. The Flash read cycle is TAA+2+TOD. Refer to the Flash/ROM Read Access diagram, in the electrical section of Part 1 of the three part 88E6208/88E6218 datasheet set (Document Number MV-S101300-01).



A.4 UniMAC™ Unified Fast Ethernet Media Access Control Registers

A.4.1 Register Map

Table 57: UniMAC Register Map Table

Register Name	Offset	Table and Page
UniMAC™ Registers		
SMI Register (SMIR)	0x80008010	Table 58, p. 97
Port Configuration Register (PCR)	0x80008400	Table 59, p. 98
Port Configuration Extend Register (PCXR)	0x80008408	Table 60, p. 99
Port Command Register (PCMR)	0x80008410	Table 61, p. 102
Port Status Register (PSR)	0x80008418	Table 62, p. 102
Hash Table Pointer Register (HTPR) — 88E6218 Only	0x80008428	Table 63, p. 103
Flow Control Source Address Low Register (FCSAL) — 88E6218 Only	0x80008430	Table 64, p. 103
Flow Control Source Address High Register (FCSAH) — 88E6218 Only	0x80008438	Table 65, p. 103
SDMA Configuration Register (SDCR)	0x80008440	Table 66, p. 104
SDMA Command Register (SDCMR)	0x80008448	Table 67, p. 104
Interrupt Cause Register (ICR)	0x80008450	Table 68, p. 106
Interrupt Reset Select Register (IRSR)	0x80008454	Table 69, p. 108
Interrupt Mask Register (IMR)	0x80008458	Table 70, p. 109
IP Differentiated Service Registers		
IP Differentiated Services CodePoint to Priority0 Low Register (DSC0L)	0x80008460	Figure 71, p. 109
IP Differentiated Services CodePoint to Priority0 High Register (DSC0H)	0x80008464	Figure 72, p. 109
IP Differentiated Services CodePoint to Priority1 Low Register (DSC1L)	0x80008468	Figure 73, p. 109
IP Differentiated Services CodePoint to Priority1 High Register (DSC1H)	0x8000846C	Figure 74, p. 110
VLAN Priority Tag to Priority Register (VPT2P)	0x80008470	Figure 75, p. 110

Table 57: UniMAC Register Map Table (Continued)

Register Name	Offset	Table and Page
UniMAC Descriptor Pointer Registers		
UniMAC First Rx Descriptor Pointer 0 Register (88E6218 Only)	0x80008480	Table 76, p. 110
UniMAC First Rx Descriptor Pointer 1 Register (88E6218 Only)	0x80008484	Table 77, p. 110
UniMAC First Rx Descriptor Pointer 2 Register	0x80008488	Table 78, p. 111
UniMAC First Rx Descriptor Pointer 3 Register	0x8000848C	Table 79, p. 111
UniMAC Current Rx Descriptor Pointer 0 Register (88E6218 Only)	0x800084A0	Table 80, p. 111
UniMAC Current Rx Descriptor Pointer 1 Register (88E6218 Only)	0x800084A4	Table 81, p. 111
UniMAC Current Rx Descriptor Pointer 2 Register	0x800084A8	Table 82, p. 111
UniMAC Current Rx Descriptor Pointer 3 Register	0x800084AC	Table 83, p. 112
UniMAC Current Tx Descriptor Pointer 0 Register	0x800084E0	Table 84, p. 112
UniMAC Current Tx Descriptor Pointer 1 Register (88E6218 Only)	0x800084E4	Table 85, p. 112

A.4.2 UniMAC™ Registers

Table 58: SMI Register (SMIR)
Offset: 0x80008010

Bits	Field	Type/ InitVal	Description
31:29	N/A	RW 0x0	Must be written as '0' for any write to the SMI register.
28	Busy	RW 0x0	1 = Indicates that an operation is in progress and that CPU must not write to the SMI register at this time.
27	ReadValid	RW 0x0	1 = Indicates that the Read operation has been completed for the addressed RegAd register and that the data is valid on the Data field.
26	OpCode	RW 0x0	0 = Write 1 = Read
25:21	RegAd	RW 0x0	Device Register Address
20:16	PhyAd	RW 0x0	Device Address

Table 58: SMI Register (SMIR) (Continued)
Offset: 0x80008010

Bits	Field	Type/ InitVal	Description
15:0	Data	RW 0x0	<p>SMI Read Operation Two transactions are required: (1) write to the SMI register with OpCode = 1, PhyAd, RegAd with the Data being any value; (2) 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'.</p> <p>SMI Write Operation One transaction is required. Write to the SMI register with OpCode = 0, PhyAd, RegAd with the Data to be written to the addressed PHY register.</p>

Table 59: Port Configuration Register (PCR)
Offset: 0x80008400

Bits	Field	Type/ InitVal	Description
31	Reserved	RES	Must be 0.
30:16	Reserved	RES 0x0	Reserved.
15	Reserved	RW 0x0	NOTE: Must be written as 1.
14	HDM	RW 0x0	<p>Hash Default Mode <i>In the 88E6218:</i> 0 = Discard addresses not found in address table. 1 = Pass addresses not found in address table. NOTE:In the 88E6208, this bit is reserved.</p>
13	HM	RW 0x0	<p>Hash Mode <i>In the 88E6218:</i> 0 = Hash Function 0 1 = Hash Function 1 NOTE: In the 88E6208, this bit is reserved.</p>
12	HS	RW 0x0	<p>Hash Size <i>In the 88E6218:</i> 0 = 8K address filtering (256KB of memory space required). 1 = 1/2K address filtering (16KB of memory space required). NOTE:In the 88E6208, this bit is reserved.</p>
11:8	Reserved	RES 0x0	Reserved and must be written 0.
7	EN	RW 0x0	<p>Enable 0 = Disabled 1 = Enable When enabled, the UniMAC™ port is ready to transmit/receive.</p>

Table 59: Port Configuration Register (PCR) (Continued)
Offset: 0x80008400

Bits	Field	Type/ InitVal	Description
6:4	Reserved	RES 0x0	Reserved and must be written 0.
3	PM_MC	RW 0x0	Promiscuous Multicast mode <i>In the 88E6218:</i> 0 = Normal mode Multicast Frames are only received if the destination address is found in the hash table. 1 = Promiscuous Multicast mode Multicast Frames are received regardless of their destination address. NOTE: In the 88E6208, this bit must be set to 1.
2	Reserved	RES	Must be 0.
1	RBM	RW 0x0	<i>In the 88E6218:</i> Reject Broadcast Mode 0 = Receive broadcast address 1 = Reject frames with broadcast address Overridden by the PM_MC mode. NOTE: In the 88E6208, this bit is reserved.
0	PM_UC	RW 0x0	Promiscuous Unicast mode <i>In the 88E6218:</i> 0 = Normal mode Unicast frames are only received if the destination address is found in the hash table. 1 = Promiscuous Unicast mode Unicast frames are received regardless of their destination address. NOTE: In the 88E6208, this bit must be set to 1.

Table 60: Port Configuration Extend Register (PCXR)
Offset: 0x80008408

Bits	Field	Type/ InitVal	Description
31:28	Reserved	RES 0x0	Reserved.
27:24	SPID_Setting	RW 0x0	Source Port ID Setting to match the Switch SPID reported in Marvell Header for frame receive queuing.



Table 60: Port Configuration Extend Register (PCXR) (Continued)
Offset: 0x80008408

Bits	Field	Type/ InitVal	Description
23:22	DA_PREFIX	RW 0x0	<i>In the 88E6218:</i> Marvell Header mode setting: 00 = Marvell Header mode in the UniMAC is disabled. Rx priority queuing is done as described in Section 7.9.3 "88E6218 Rx Priority Queuing — Marvell Header Mode Disabled" on page 40. 01 = Marvell Header mode in the UniMAC is enabled. Rx priority queue is set to PRI[2:1]. 10 = Marvell Header mode in the v is enabled. Rx priority queue is set to (DBNum[0], PRI[2]). 11 = Marvell Header mode in the UniMAC is enabled. Rx priority queue set to (SPID[3:0]==SPID_Setting,PRI[2]). NOTE: For a description of the Rx priority queuing, refer to Section 7.9 "Rx Queuing" on page 39. <i>In the 88E6208:</i> Marvell Header mode setting: 00 = Marvell Header mode in the UniMAC is disabled. Only one Rx queue is used. The default queue is 3. 01 = Marvell Header mode in the UniMAC is enabled. Rx Queue = 3. 10 = Marvell Header mode is enabled in the UniMAC. Rx queue set to {DBNum[0], 1}. 11 = Marvell Header mode is enabled in the UniMAC. Rx queue set to {SPID[3:0]==SPID_Setting, 1}
21	DSCPen	0x0	DSCP enable (only when Marvell Header mode is disabled) <i>In the 88E6218:</i> 0 = IP DSCP field decoding is disabled. 1 = IP DSCP field decoding is enabled. NOTE: In the 88E6208, this bit is reserved and must be set to 0.
20:19	Reserved	RES 0x0	Reserved. NOTE: Bit 19 <i>must</i> be 1.
18	Reserved	RW 0x0	Reserved NOTE: Must be set to 1.
17:13	Reserved	RES 0x0	Reserved. Bits 15:14 <i>must</i> be 01.
12	FCTL	RW 0x0	<i>In the 88E6218:</i> Flow-Control Mode (only when Marvell Header mode is disabled) 0 = Enable IEEE 802.3x flow-control 1 = Disable IEEE 802.3x flow-control NOTE: In the 88E6208, this bit is reserved and must be set to 1.
11:9	Reserved	RES 0x2	Reserved. This field <i>must</i> be written 011.
8	PRIOrx_ Override	0x0	<i>In the 88E6218:</i> Override Priority for Packets Received on this Port This bit is effective only if Marvell Header mode is disabled. 0 = Do not override 1 = Override with <PRIOrx> field NOTE: In the 88E6208, this bit is reserved and must be set to 1.

Table 60: Port Configuration Extend Register (PCXR) (Continued)
Offset: 0x80008408

Bits	Field	Type/ InitVal	Description
7:6	PRIOrx	RW 0x0	<p><i>In the 88E6218:</i> Default Priority for Packets Received on this Port 00 = Queue 0 Lowest priority 01 = Queue 1 10 = Queue 2 11 = Queue 3 Highest priority NOTE:In the 88E6208, these bits are reserved and must be set to 1.</p>
5:3	PRIOTx	RW 0x0	<p><i>In the 88E6218:</i> Priority weight in the round-robin between high and low priority Tx queues. 000 = 1 packet transmitted from HIGH, 1 packet transmitted from LOW. 001 = 2 packets transmitted from HIGH, 1 packet transmitted from LOW. 010 = 4 packets transmitted from HIGH, 1 packet transmitted from LOW. 011 = 6 packets transmitted from HIGH, 1 packet transmitted from LOW. 100 = 8 packets transmitted from HIGH, 1 packet transmitted from LOW. 101 = 10 packets transmitted from HIGH, 1 packet transmitted from LOW. 110 = 12 packets transmitted from HIGH, 1 packet transmitted from LOW. 111 = All packets transmitted from HIGH, 0 packets transmitted from LOW. LOW is served only if HIGH is empty. If the HIGH queue is emptied before finishing the count, the count is reset until the next first HIGH comes in. NOTE:In the 88E6208, these bits are reserved.</p>
2	Reserved	RES 0x0	Reserved.
1	BPDU	0x0	<p><i>In the 88E6218:</i> BPDU Tree Packets Capture Enable (only when Marvell Header mode is disabled) 0 = BPDU (Bridge Protocol Data Unit) packets are treated as normal Multicast packets. 1 = BPDU packets are trapped and sent to high priority RX queue. NOTE:In the 88E6208, this bit is reserved and must be set to 0.</p>
0	IGMP	0x0	<p><i>In the 88E6218:</i> IGMP Packets Capture Enable (only when Marvell Header mode is disabled) 0 = IGMP packets are treated as normal Multicast packets. 1 = IGMP packets on IPv4/IPv6 over Ethernet/802.3 are trapped and sent to high priority RX queue. NOTE:In the 88E6208, this bit is reserved and must be set to 0.</p>

Table 61: Port Command Register (PCMR)
Offset: 0x80008410

Bits	Field	Type/ InitVal	Description
31:16	Reserved	RES 0x0	Reserved.
15	FC	RW 0x1	<i>In the 88E6218:</i> Flow Control (only when Marvell Header mode is disabled) When the CPU recognizes that it is going to run out of receive buffers, Setting this bit causes the port's transmitter to send flow-control PAUSE packets. The CPU must reset this bit when more resources are available. NOTE: In the 88E6208, this bit is reserved and must be set to 0.
14:0	Reserved	RES 0x0	Reserved.

Table 62: Port Status Register (PSR)
Offset: 0x80008418

Bits	Field	Type/ InitVal	Description
31:8	Reserved	R0 0x0	Reserved.
7	TXinProg	R0 0x0	TX in Progress Indicates that the port's transmitter is in an active transmission state. This bit is read only.
6	TxStatusHigh	R0 0x0	<i>In the 88E6218:</i> Tx High Priority Status Indicates the status of the high priority transmit queue: 0 = Stopped 1 = Running This bit is read only. NOTE: In the 88E6208, this bit is reserved.
5	TxStatus	R0 0x0	<i>In the 88E6218:</i> Tx Low Priority Status Indicates the status of the low priority transmit queue: <i>In the 88E6208:</i> Tx Status Indicates the status of the transmit queue: <i>In both the 88E6208 and 88E6218:</i> 0 = Stopped 1 = Running This bit is read only.

Table 62: Port Status Register (PSR) (Continued)
Offset: 0x80008418

Bits	Field	Type/ InitVal	Description
4	Pause	R0 0x0	<i>In the 88E6218:</i> Indicates that the port is in flow-control disabled state. This bit is set when an IEEE 802.3x flow-control PAUSE (XOFF) packet is received (assuming that flow-control is enabled). Reset when XON is received, or when the XOFF timer has expired. This bit is read only. NOTE: When Marvell Header is enabled the UniMAC ignores the received PAUSE packet. In the 88E6208, this bit is reserved.
3:0	Reserved	R0 0x8	Reserved.

Table 63: Hash Table Pointer Register (HTPR) — 88E6218 Only
Offset: 0x80008428

Bits	Field	Type/ InitVal	Description
31:0	HTP	RW 0x0	32-bit pointer to the address table. Bits 2:0 must be set to zero.

Table 64: Flow Control Source Address Low Register (FCSAL) — 88E6218 Only
Offset: 0x80008430

Bits	Field	Type/ InitVal	Description
31:16	Reserved	0x0	Reserved
15:0	SA[15:0]	0x0	Source Address The least significant bits of the source address for the port. This address is used for Flow Control.

Table 65: Flow Control Source Address High Register (FCSAH) — 88E6218 Only
Offset: 0x80008438

Bits	Field	Type/ InitVal	Description
31:0	SA[47:16]	0x0	Source Address The most significant bits of the source address for the port. This address is used for Flow Control.



Table 66: SDMA Configuration Register (SDCR)
Offset: 0x80008440

Bits	Field	Type/ InitVal	Description
31:14	Reserved	RES 0x0	Reserved.
13:12	BSZ	RW 0x0	Burst Size Sets the maximum burst size for SDMA transactions: 00 = Burst is limited to 1 32-bit words. 01 = Burst is limited to 2 32-bit words. 10 = Burst is limited to 4 32-bit words. 11 = Burst is limited to 8 32-bit words.
11:10	Reserved	RES 0x0	Reserved.
9	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).
8	Reserved	RES 0x0	Reserved.
7	BLMT	RW 0x1	Big/Little Endian Transmit Mode The DMA supports Big or Little Endian configurations. The BLMT bit only affects data transfer from memory. When set to 0 (Big Endian), the UniMAC Tx DMA performs byte swap on each 32-bit word fetched from memory before transmitting the packet. 0 = Big Endian 1 = Little Endian
6	BLMR	RW 0x1	Big/Little Endian Receive Mode The DMA supports Big or Little Endian configurations on a per channel basis. The BLMR bit only affects data transfer to memory. When set to 0 (Big Endian), the UniMAC Rx DMA performs byte swap on each 32-bit data packet before transferring it to memory. 0 = Big Endian 1 = Little Endian
5:0	Reserved	RES 0x0	Reserved.

Table 67: SDMA Command Register (SDCMR)
Offset: 0x80008448

Bits	Field	Type/ InitVal	Description
31:25	Reserved	RES 0x0	Reserved.

Table 67: SDMA Command Register (SDCMR) (Continued)
Offset: 0x80008448

Bits	Field	Type/ InitVal	Description
24	TXDL	RW 0x0	<p><i>In the 88E6218:</i> Start Tx Low Set to '1' by the CPU to cause the SDMA to fetch the first descriptor and start a transmit process from the low priority Tx queue.</p> <p><i>In the 88E6208:</i> Start Tx Set to '1' by the CPU to cause the SDMA to fetch the first descriptor and start a transmit process from the Tx queue.</p> <p><i>In both the 88E6208 and 88E6218:</i> Writing '1' to TXDL resets STDL bit. Writing '0' to this bit has no effect.</p>
23	TXDH	RW 0x0	<p><i>In the 88E6218:</i> Start Tx High Set to '1' by the CPU in order to cause the SDMA to fetch the first descriptor and start a transmit process from the high priority Tx queue. Writing '1' to TXDH resets STDH bit. Writing '0' to this bit has no effect. NOTE: In the 88E6208, this bit is reserved.</p>
22:18	Reserved	RES 0x0	Reserved.
17	STDL	RW 0x0	<p><i>In the 88E6218:</i> Stop TX Low Set to '1' by the CPU to stop the transmission process from the low priority Tx queue at the end of the current frame. An interrupt is generated when the stop command has been executed.</p> <p><i>In the 88E6208:</i> Stop TX Set to '1' by the CPU to stop the transmission process from the TX queue at the end of the current frame. An interrupt is generated when the stop command has been executed.</p> <p><i>In both the 88E6208 and 88E6218:</i> Writing '1' to STDL resets TXDL bit. Writing '0' to this bit has no effect.</p>
16	STDH	RW 0x0	<p>Stop TX High Set to '1' by the CPU to stop the transmission process from the high priority queue at the end of the current frame. An interrupt is generated when the stop command has been executed. Writing '1' to STDH resets TXDH bit. Writing '0' to this bit has no effect. NOTE: This bit is applicable to the 88E6218. In the 88E6208, this bit is reserved.</p>
15	AR	RW 0x0	<p>Abort Receive Set to '1' by the CPU to abort a receive SDMA operation. When the AR bit is set, the SDMA aborts its current operation and moves to IDLE. No descriptor is closed. The AR bit is cleared upon entering IDLE. After setting the AR bit, the CPU must poll the bit to verify that the abort sequence is completed.</p>

Table 67: SDMA Command Register (SDCMR) (Continued)
Offset: 0x80008448

Bits	Field	Type/ InitVal	Description
14:8	Reserved	RES 0x0	Reserved.
7	ERD	RW 0x0	Enable RX DMA. Set to '1' by the CPU to cause the SDMA to start a receive process. Cleared when the CPU issues an Abort Receive command.
6:0	Reserved	RES 0x0	Reserved.

Table 68: Interrupt Cause Register (ICR)
Offset: 0x80008450

Bits	Field	Type/ InitVal	Description
31:30	Reserved	RES 0x0	Reserved.
29	SMIdone	RO 0x0	SMI Command Done Indicates that the SMI completed a MII management command (either read or write) that was initiated by the CPU writing to the SMI register.
28:24	Reserved	RES 0x0	Reserved.
23	RxError-Queue[3]	RO 0x0	Rx Resource Error in Priority Queue[3] Indicates a Rx resource error event in receive priority queue[3].
22	RxError-Queue[2]	RO 0x0	Rx Resource Error in Priority Queue[2] Indicates a Rx resource error event in receive priority queue[2].
21	RxError-Queue[1]	RO 0x0	<i>In the 88E6218:</i> Rx Resource Error in Priority Queue[1] Indicates a Rx resource error event in receive priority queue[1]. NOTE: In the 88E6208, this bit is reserved.
20	RxError-Queue[0]	RO 0x0	<i>In the 88E6218:</i> Rx Resource Error in Priority Queue[0] Indicates a Rx resource error event in receive priority queue[0]. NOTE: In the 88E6208, this bit is reserved.
19	RxBuffer-Queue[3]	RO 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 a receive priority queue[3].
18	RxBuffer-Queue[2]	RO 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 a receive priority queue[2].

Table 68: Interrupt Cause Register (ICR) (Continued)
Offset: 0x80008450

Bits	Field	Type/ InitVal	Description
17	RxBuffer-Queue[1]	RO 0x0	<i>In the 88E6218:</i> 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 a receive priority queue[1]. NOTE: In the 88E6208, this bit is reserved.
16	RxBuffer-Queue[0]	RO 0x0	<i>In the 88E6218:</i> 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 a receive priority queue[0]. NOTE: In the 88E6208, this bit is reserved.
15:14	Reserved	RES 0x0	Reserved.
13	TxUdr	RO 0x0	Tx Underrun Indicates an underrun event that occurred during transmission of packets.
12	RxOVR	RO 0x0	Rx Overrun Indicates an overrun event that occurred during reception of a packet.
11	TxError	RO 0x0	<i>In the 88E6218:</i> Tx Resource Error for Low Priority Tx Queue Indicates a Tx resource error event during packet transmission from the low priority Tx queue. <i>In the 88E6208:</i> Tx Resource Error for Tx Queue Indicates a Tx resource error event during packet transmission from the Tx queue.
10	TxErrorHigh	RO 0x0	Tx Resource Error for High Priority Queue Indicates a Tx resource error event during packet transmission from the high priority queue. NOTE: This bit is applicable to the 88E6218. In the 88E6208, this bit is reserved.
9	Reserved	RES 0x0	Reserved.
8	RxError	RO 0x0	Rx Resource Error <i>In the 88E6218:</i> Indicates a Rx resource error event in any priority queue. To get a Rx Resource Error Indication per priority queue, use bit 23:20. <i>In the 88E6208:</i> Indicates a Rx resource error event in either queue. To get a Rx Resource Error Indication per queue, use bit 23:22.
7	TxEnd	RO 0x0	<i>In the 88E6218:</i> Tx End for Low Priority Tx Queue Indicates that the Tx DMA stopped processing the low priority queue after stop command, or that it reached the end of the low priority descriptor chain. <i>In the 88E6208:</i> Tx End for Tx Queue Indicates that the Tx DMA stopped processing the Tx queue after stop command, or that it reached the end of the descriptor chain.



Table 68: Interrupt Cause Register (ICR) (Continued)
Offset: 0x80008450

Bits	Field	Type/ InitVal	Description
6	TxEndHigh	RO 0x0	<i>In the 88E6218:</i> Tx End for High Priority Tx Queue Indicates that the Tx DMA stopped processing the high priority queue after stop command, or that it reached the end of the high priority descriptor chain. NOTE: In the 88E6208, this bit is reserved.
5:4	Reserved	RES 0x0	Reserved.
3	TxBuffer	RO 0x0	<i>In the 88E6218:</i> Tx Buffer for Low Priority Tx Queue <i>In the 88E6208:</i> Tx Buffer for Tx Queue <i>In both the 88E6208 and 88E6218:</i> 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 which has its EI bit set. To limit the interrupts to frame (rather than buffer) boundaries, the user must set EI only in the last descriptor.
2	TxBufferHigh	RO 0x0	Tx Buffer for High Priority Tx Queue Indicates a Tx buffer returned to CPU ownership or that the port finished transmission of a Tx frame. NOTE: In the 88E6218 this bit is set upon closing any Tx descriptor which has its EI bit set. To limit the interrupts to frame (rather than buffer) boundaries, the user must set EI only in the last descriptor. In the 88E6208, this bit is reserved.
1	Reserved	RES 0x0	Reserved.
0	RxBuffer	RO 0x0	Rx Buffer Return <i>In the 88E6218:</i> Indicates an Rx buffer returned to CPU ownership or that the port finished reception of a Rx frame in any priority queue. This bit is set upon closing any Rx descriptor which has its EI bit set. To get a Rx Buffer return per queue, use bits 19:16. <i>In the 88E6208:</i> Indicates an Rx buffer returned to CPU ownership or that the port finished reception of a Rx frame in either queue. This bit is set upon closing any Rx descriptor which has its EI bit set. To get a Rx Buffer return per queue, use bits 19:18.

Table 69: Interrupt Reset Select Register (IRSR)
Offset: 0x80008454

Bits	Field	Type/ InitVal	Description
31:0	Various	0x0	Reset bits of the Interrupt Cause register. 0 = Write to clear the Interrupt Cause register. 1 = Read will clear the Interrupt Cause register.

Table 70: Interrupt Mask Register (IMR)
Offset: 0x80008458

Bits	Field	Type/ InitVal	Description
31:0	Various	0x0	Mask bits for the Interrupt Cause register. 0 = Mask 1 = Enable

A.4.3 IP Differentiated Service Registers

Table 71: IP Differentiated Services CodePoint to Priority0 Low Register (DSC0L)
Offset: 0x80008460

Bits	Field	Type/ InitVal	Description
31:0	Priority0 low	RW 0x0	<i>In the 88E6218:</i> The LSB priority bits for DSCP[31:0] entries. NOTE: In the 88E6208, these bits are reserved and must be set to 0xFFFF.FFFF.

Table 72: IP Differentiated Services CodePoint to Priority0 High Register (DSC0H)
Offset: 0x80008464

Bits	Field	Type/ InitVal	Description
31:0	Priority0 high	RW 0x0	<i>In the 88E6218:</i> The LSB priority bits for DSCP[63:32] entries. NOTE: In the 88E6208, these bits are reserved and must be set to 0xFFFF.FFFF.

Table 73: IP Differentiated Services CodePoint to Priority1 Low Register (DSC1L)
Offset: 0x80008468

Bits	Field	Type/ InitVal	Description
31:0	Priority1 low	RW 0x0	<i>In the 88E6218:</i> The MSB priority bits for DSCP[31:0] entries. NOTE: In the 88E6208, these bits are reserved and must be set to 0xFFFF.FFFF.



Table 74: IP Differentiated Services CodePoint to Priority1 High Register (DSC1H)
Offset: 0x8000846C

Bits	Field	Type/ InitVal	Description
31:0	Priority1 high	RW 0x0	<i>In the 88E6218:</i> The MSB priority bit for DSCP[63:32] entries. NOTE: In the 88E6208, these bits are reserved and must be set to 0xFFFF.FFFF.

Table 75: VLAN Priority Tag to Priority Register (VPT2P)
Offset: 0x80008470

Bits	Field	Type/ InitVal	Description
31:16	Reserved	RES 0x0	Reserved.
15:8	Priority1	RW 0xF0	<i>In the 88E6218:</i> The MSB priority bits for VLAN Priority[7:0] entries. NOTE: In the 88E6208, these bits are reserved and must be set to 0xFF.
7:0	Priority0	RW 0xCC	<i>In the 88E6218:</i> The LSB priority bits for VLAN Priority[7:0] entries. NOTE: In the 88E6208, these bits are reserved and must be set to 0xFF.

A.4.4 UniMAC Descriptor Pointer Registers

Table 76: UniMAC First Rx Descriptor Pointer 0 Register (88E6218 Only)
Offset: 0x80008480

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	First Rx Descriptor Pointer 0 for receive priority queue 0.

Table 77: UniMAC First Rx Descriptor Pointer 1 Register (88E6218 Only)
Offset: 0x80008484

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	First Rx Descriptor Pointer 1 for receive priority queue 1.

Table 78: UniMAC First Rx Descriptor Pointer 2 Register
Offset: 0x80008488

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	First Rx Descriptor Pointer 2 for receive priority queue 2.

Table 79: UniMAC First Rx Descriptor Pointer 3 Register
Offset: 0x8000848C

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	First Rx Descriptor Pointer 3 for receive priority queue 3.

Table 80: UniMAC Current Rx Descriptor Pointer 0 Register (88E6218 Only)
Offset: 0x800084A0

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	Current Rx Descriptor Pointer 0 for receive priority queue 0.

Table 81: UniMAC Current Rx Descriptor Pointer 1 Register (88E6218 Only)
Offset: 0x800084A4

Bits	Field	Type/ InitVal	Description
31:0	Descriptor	RW 0x0	Current Rx Descriptor Pointer 1 for receive priority queue 1.

Table 82: UniMAC Current Rx Descriptor Pointer 2 Register
Offset: 0x800084A8

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	Current Rx Descriptor Pointer 2 for receive priority queue 2.



Table 83: UniMAC Current Rx Descriptor Pointer 3 Register
Offset: 0x800084AC

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	Current Rx Descriptor Pointer 3 for receive priority queue 3.

Table 84: UniMAC Current Tx Descriptor Pointer 0 Register
Offset: 0x800084E0

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	<i>In the 88E6218:</i> Current Tx Descriptor Pointer 0 for transmit priority queue 0. <i>In the 88E6208:</i> Current Tx Descriptor Pointer 0 for transmit queue 0.

Table 85: UniMAC Current Tx Descriptor Pointer 1 Register (88E6218 Only)
Offset: 0x800084E4

Bits	Field	Type/ InitVal	Description
31:0	Descriptor Pointer	RW 0x0	Current Tx Descriptor Pointer 1 for transmit priority queue 1.

A.5 Independent DMA Registers (88E6218 Only)

A.5.1 Independent DMA Register Map

The following table provides a summary of all of the DMA registers, including the register names, their type, offset, and a reference to the corresponding table and page.

Table 86: IDMA Register Map

Register Name	Offset	Table and Page
Channel Descriptor		
Channel 0 DMA Byte Count Register	0x8000E800	Table 87, p. 114
Channel 1 DMA Byte Count Register	0x8000E804	Table 88, p. 114
Channel 0 DMA Source Address Register	0x8000E810	Table 89, p. 114
Channel 1 DMA Source Address Register	0x8000E814	Table 90, p. 114
Channel 0 DMA Destination Address Register	0x8000E820	Table 91, p. 115
Channel 1 DMA Destination Address Register	0x8000E824	Table 92, p. 115
Channel 0 Next Descriptor Pointer Register	0x8000E830	Table 93, p. 115
Channel 1 Next Descriptor Pointer Register	0x8000E834	Table 94, p. 115
Channel 0 Current Descriptor Pointer Register	0x8000E870	Table 95, p. 115
Channel 1 Current Descriptor Pointer Register	0x8000E874	Table 96, p. 116
DMA Channel Control		
Channel 0 Control Register	0x8000E840	Table 97, p. 116
Channel 1 Control Register	0x8000E844	Table 98, p. 118
DMA Arbiter		
Arbiter Control Register	0x8000E860	Table 99, p. 120
Channel Interrupt		
Channel 0 Interrupt Mask Register	0x8000E880	Table 100, p. 120
Channel 1 Interrupt Mask Register	0x8000E884	Table 101, p. 120
Channel 0 Interrupt Reset Select Register	0x8000E890	Table 102, p. 121
Channel 1 Interrupt Reset Select Register	0x8000E894	Table 103, p. 121
Channel 0 Interrupt Status Register	0x8000E8A0	Table 104, p. 121
Channel 1 Interrupt Status Register	0x8000E8A4	Table 105, p. 121



A.5.2 Independent DMA Channel Descriptor Registers

Table 87: Channel 0 DMA Byte Count Register
Offset: 0x8000E800

Bits	Field	Type/ InitVal	Description
31:16	Reserved	RES	Reserved
15:0	ByteCt0	RWR 0x0	The number of bytes that are left to transfer.

Table 88: Channel 1 DMA Byte Count Register
Offset: 0x8000E804

Bits	Field	Type/ InitVal	Description
31:16	Reserved	RES	Reserved
15:0	ByteCt1	RWR 0x0	The number of bytes that are left to transfer.

Table 89: Channel 0 DMA Source Address Register
Offset: 0x8000E810

Bits	Field	Type/ InitVal	Description
31:0	SrcAdd0	RWR 0x0	The address from which the DMA controller reads the data.

Table 90: Channel 1 DMA Source Address Register
Offset: 0x8000E814

Bits	Field	Type/ InitVal	Description
31:0	SrcAdd1	RWR 0x0	The address from which the DMA controller reads the data.

Table 91: Channel 0 DMA Destination Address Register
Offset: 0x8000E820

Bits	Field	Type/ InitVal	Description
31:0	DestAdd0	RWR 0x0	The address that the DMA controller writes the data to.

Table 92: Channel 1 DMA Destination Address Register
Offset: 0x8000E824

Bits	Field	Type/ InitVal	Description
31:0	DestAdd1	RWR 0x0	The address that the DMA controller writes the data to.

Table 93: Channel 0 Next Descriptor Pointer Register
Offset: 0x8000E830

Bits	Field	Type/ InitVal	Description
31:0	NDPTR0	RWR 0x0	The address for the next descriptor of DMA. A '0' value means a Null pointer (end of the chained list). The next descriptor pointer must be 16 byte aligned. This means bits 3:0 must be set to '0'.

Table 94: Channel 1 Next Descriptor Pointer Register
Offset: 0x8000E834

Bits	Field	Type/ InitVal	Description
31:0	NDPTR1	RWR 0x0	The address for the next descriptor of DMA. A '0' value means a Null pointer (end of the chained list). The next descriptor pointer must be 16 bytes aligned. This means bits 3:0 must be set to '0'.

Table 95: Channel 0 Current Descriptor Pointer Register
Offset: 0x8000E870

Bits	Field	Type/ InitVal	Description
31:0	CDPTR0	RO 0x0	The address for the current descriptor that the DMA is working on.

Table 96: Channel 1 Current Descriptor Pointer Register
Offset: 0x8000E874

Bits	Field	Type/ InitVal	Description
31:0	CDPTR1	RO 0x0	The address for the current descriptor that the DMA is working on.

A.5.3 DMA Channel Control Registers

Table 97: Channel 0 Control Register
Offset: 0x8000E840

Bits	Field	Type/ InitVal	Description
31:21	Reserved	RES	Reserved
20	ABR	RWR 0x0	Abort DMA transfer Only set this if the CPU wants to stop and re-program DMA, and CDE (bit 17) is set to 1. When the CPU issues this command, it should also set ChanEn (bit 12) to 0. 0 = No influence on channel behavior. 1 = Abort DMA.
19:18	Reserved	RES	Reserved
17	CDE	RWR 0x0	Close Descriptor Enable. If enabled, DMA writes the remaining of byte count to bits 31:16 of the Byte-Count field in the current descriptor. 0 = Disable 1 = Enable
16	Reserved	RES	Reserved
15	SDA	RWR 0x0	Source Destination Alignment 0 = Alignment is done towards source address 1 = Alignment is done towards destination address Destination alignment is not supported when the burst limit is 1-, 2-, or 4-bytes. NOTE: If both the DMA Source and Destination addresses are aligned, the meaning of this bit is irrelevant.
14	DMAActSt	RO 0x0	DMA Activity Status (read only) Assertion of this bit results from asserting ChanEn (bit 12). In non-chain mode, this bit is deasserted when Byte_Count reaches zero. In chain-mode, this bit is deasserted when the next descriptor pointer is null and Byte_Count reaches zero. This bit will be reset if the CPU sets ChanEn to 0 during DMA transfer. 0 = Channel is not active 1 = Channel is active

Table 97: Channel 0 Control Register (Continued)
Offset: 0x8000E840

Bits	Field	Type/ InitVal	Description
13	FetchND	SC 0x0	Fetch Next Descriptor 0 = Do not force. 1 = Forces a fetch of the next Descriptor (Even if the current DMA has not ended.) This bit is reset after fetch is completed. This only applies in Chained mode.
12	ChanEn	SC 0x0	Channel Enable 0 = Disable 1 = Enable
11	Reserved	RES	Must be '1'.
10	IntMode	RWR 0x0	Interrupt Mode 0 = Interrupt asserted every time the DMA byte count reaches 0. 1 = Interrupt asserted when the DMA byte count reaches 0 and the next descriptor pointer value is null. NOTE: If chained mode is disabled, the setting of IntMode is irrelevant, and the DMAComp Interrupt will be asserted every time the Byte Count reaches 0.
9	ChainMod	RWR 0x0	Chained Mode 0 = Chained mode When a DMA access is completed, the parameters of the next DMA access comes from a descriptor in memory to which a Next Descriptor Counter register points. 1 = Non-Chained mode Only the values programmed by the CPU directly into the Byte Count, Source Address, and Destination Address registers are used. See Section 8.3.1 "Chain Mode" on page 55 , Figure 15, "Chain Mode," on page 56 and Section 8.3.2 "Non-Chain Mode" on page 57 .
8:6	BurstLimit	RWR 0x0	Burst Limit in each DMA access. 101 = Reserved 110 = Reserved 000 = 4 Bytes 001 = 8 Bytes 011 = 16 Bytes 111 = 32 Bytes
5:4	DestDir	RWR 0x00	Destination Direction 00 = Increment destination address 01 = Reserved 10 = Hold in the same value 11 = Reserved
3:2	SrcDir	RWR 0x00	Source Direction 00 = Increment source address 01 = Reserved 10 = Hold in the same value 11 = Reserved



Table 97: Channel 0 Control Register (Continued)
Offset: 0x8000E840

Bits	Field	Type/ InitVal	Description
1:0	Reserved	RES	Reserved

Table 98: Channel 1 Control Register
Offset: 0x8000E844

Bits	Field	Type/ InitVal	Description
31:21	Reserved	RES	Reserved
20	ABR	RWR 0x0	Abort DMA transfer Only set this if the CPU wants to stop and re-program DMA, and CDE (bit 17) is set to 1. When the CPU issues this command, it should also set ChanEn (bit 12) to 0. 0 = No influence on channel behavior. 1 = Abort DMA.
19:18	Reserved	RES	Reserved
17	CDE	RWR 0x0	Close Descriptor Enable. If enabled, DMA writes the remaining of byte count to bits 31:16 of the Byte-Count field in the current descriptor. 0 = Disable 1 = Enable
16	Reserved	RES	Reserved
15	SDA	RWR 0x0	Source Destination Alignment 0 = Alignment is done towards source address 1 = Alignment is done towards destination address Destination alignment is not supported when the burst limit is 1-, 2-, or 4-bytes. NOTE: If both the DMA Source and Destination addresses are aligned, the meaning of this bit is irrelevant.
14	DMAActSt	RO 0x0	DMA Activity Status (read only) Assertion of this bit results from asserting ChanEn (bit 12). In non-chain mode, this bit is deasserted when Byte_Count reaches zero. In chain-mode, this bit is deasserted when the next descriptor pointer is null and Byte_Count reaches zero. This bit will be reset if the CPU sets ChanEn to 0 during DMA transfer. 0 = Channel is not active 1 = Channel is active

Table 98: Channel 1 Control Register (Continued)
Offset: 0x8000E844

Bits	Field	Type/ InitVal	Description
13	FetND	SC 0x0	Fetch Next Descriptor 0 = Do not force. 1 = Forces a fetch of the next Descriptor (Even if the current DMA has not ended.) This bit is reset after fetch is completed. This only applies in Chained mode.
12	ChanEn	SC 0x0	Channel Enable 0 = Disable 1 = Enable
11	Reserved	RES	Must be '1'.
10	IntMode	RWR 0x0	Interrupt Mode 0 = Interrupt asserted every time the DMA byte count reaches 0. 1 = Interrupt asserted when the DMA byte count reaches 0 and the next descriptor pointer value is null. NOTE: If chained mode is disabled, the setting of IntMode is irrelevant, and the DMAComp Interrupt will be asserted every time the Byte Count reaches 0.
9	ChainMod	RWR 0x0	Chained Mode 0 = Chained mode When a DMA access is completed, the parameters of the next DMA access comes from a descriptor in memory to which a Next Descriptor Counter register points. 1 = Non-Chained mode Only the values programmed by the CPU directly into the Byte Count, Source Address, and Destination Address registers are used. See Section 8.3.1 "Chain Mode" on page 55 , Figure 15, "Chain Mode," on page 56 and Section 8.3.2 "Non-Chain Mode" on page 57 .
8:6	BurstLimit	RWR 0x0	Burst Limit in each DMA access. 101 = Reserved 110 = Reserved 000 = 4 Bytes 001 = 8 Bytes 011 = 16 Bytes 111 = 32 Bytes
5:4	DestDir	RWR 0x00	Destination Direction 00 = Increment destination address 01 = Reserved 10 = Hold in the same value 11 = Reserved
3:2	SrcDir	RWR 0x00	Source Direction 00 = Increment source address 01 = Reserved 10 = Hold in the same value 11 = Reserved

Table 98: Channel 1 Control Register (Continued)
Offset: 0x8000E844

Bits	Field	Type/ InitVal	Description
1:0	Reserved	RES	Reserved

A.5.3.1 DMA Arbiter Registers

Table 99: Arbiter Control Register
Offset: 0x8000E860

Bits	Field	Type/ InitVal	Description
31:2	Reserved	RES	Reserved
1:0	PrioChan1/0	RWR 0x0	Priority between Channel 0 and Channel 1. 00 = Round Robin 01 = Priority to channel 1 over channel 0 10 = Priority to channel 0 over channel 1 11 = Reserved

A.5.3.2 DMA Channel Interrupt Registers

Table 100: Channel 0 Interrupt Mask Register
Offset: 0x8000E880

Bits	Field	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	Comp0	RWR 0x0	If set to '1', Channel 0 DMA Completion interrupt is enabled.

Table 101: Channel 1 Interrupt Mask Register
Offset: 0x8000E884

Bits	Field	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	Comp1	RWR 0x0	If set to '1', Channel 1 DMA Completion interrupt is enabled.

Table 102: Channel 0 Interrupt Reset Select Register
Offset: 0x8000E890

Bits	Field	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	Comp0	RWR 0x0	0 = Channel 0 DMA Completion interrupt is cleared by writing a '0' to the corresponding bit of interrupt status register. 1 = Channel 0 DMA Completion interrupt is cleared by reading the corresponding interrupt status register.

Table 103: Channel 1 Interrupt Reset Select Register
Offset: 0x8000E894

Bits	Field	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	Comp1	RWR 0x0	0 = Channel 1 DMA Completion interrupt is cleared by writing a '0' to the corresponding bit of interrupt status register. 1 = Channel 1 DMA Completion interrupt is cleared by reading the corresponding interrupt status register.

Table 104: Channel 0 Interrupt Status Register
Offset: 0x8000E8A0

Bits	Field	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	Comp0	RO 0x0	Channel 0 DMA Completion.

Table 105: Channel 1 Interrupt Status Register
Offset: 0x8000E8A4

Bits	Field	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	Comp1	RO 0x0	Channel 1 DMA Completion.



A.6 Two-Wire Serial Interface Registers

A.6.1 Two-Wire Serial Interface Register Map

Table 106: TWSI Register Map

Register Name	Offset	Table and Page
TWSI Clock Prescaler Register	0x8000C100	Table 107, p. 122
TWSI Global Control Register	0x8000C104	Table 108, p. 122
TWSI Control Register	0x8000C110	Table 109, p. 123
TWSI Status Register	0x8000C11C	Table 110, p. 123
TWSI Interrupt Mask Register	0x8000C120	Table 111, p. 123
TWSI Data Register	0x8000C124	Table 112, p. 124
TWSI Interrupt Source Register	0x8000C12C	Table 113, p. 124
Peripheral Reset Register	0x8000C000	Table 114, p. 124
Peripheral Clock Divider Register	0x8000C004	Table 115, p. 124

A.6.2 Two-Wire Serial Interface Registers

Table 107: TWSI Clock Prescaler Register
Offset: 0x8000C100

Bits	Field	Type/ InitVal	Description
15:0	PRER	RW 0x0	This register is used to scale down the UART clock (Uclk) frequency for the TWSI unit (SCK). Make sure the generated SCK clock is less than the maximum clock rate specified in the data sheet. SCK is calculated based on the equation below. $SCK = (Uclk)/(5 \cdot PRER)$

Table 108: TWSI Global Control Register
Offset: 0x8000C104

Bits	Field	Type/ InitVal	Description
7:0	Reserved	RO 0x00	Must be 0x00.

Table 109: TWSI Control Register
Offset: 0x8000C110

Bits	Field	Type/ InitVal	Description
7:2	Reserved	RES	Reserved
1	TWSIStart	RW 0x0	Sets the start of TWSI transmission. 1 = Start 0 = No effect
0	TWSIStop	RW 0x0	Sets the end of TWSI transmission. 1 = Stop 0 = No effect

Table 110: TWSI Status Register
Offset: 0x8000C11C

Bits	Field	Type/ InitVal	Description
7	MicroWireTip	RO 0x0	Indicates MicroWire command execution in progress.
6	MicroWireBusy	RO 0x0	Indicates MicroWire serial bus busy.
5	SPIWrEn	RO 0x0	SPI write enable. 0 = The serial EEPROM is not write enabled. 1 = The serial EEPROM is write enabled.
4	SPIRdy	RO 0x0	SPI Ready signal. 0 = The serial EEPROM is ready. 1 = The internal write cycle is in progress.
3	RxAck	RO 0x0	TWSI receive Acknowledge 0 = Receiving ACK bit from memory slave during latest TWSI write. 1 = Non-ACK bit is received from slave at latest write.
2	TWSIBusy	RO 0x0	Indicates TWSI bus is busy.
1	3W4WTip	RO 0x0	Indicates 3W4W command execution in progress.
0	3W4WBusy	RO 0x0	Indicates 3W4W bus is busy.

Table 111: TWSI Interrupt Mask Register
Offset: 0x8000C120

Bits	Field	Type/ InitVal	Description
7:1	Reserved	RES	Reserved



Table 111: TWSI Interrupt Mask Register (Continued)
Offset: 0x8000C120

Bits	Field	Type/ InitVal	Description
0	IMR	RW 0x0	ARM writes this register to mask TWSI interrupt. 0 = Disabled. 1 = Enabled. Default state after reset is disabled. TWSI Interrupt Events: Only one interrupt event has been defined for the 88E6218: TWSI write-slave_nonACK interrupt.

Table 112: TWSI Data Register
Offset: 0x8000C124

Bits	Field	Type/ InitVal	Description
15:0	Data	RW 0x0	TWSI Data register.

Table 113: TWSI Interrupt Source Register
Offset: 0x8000C12C

Bits	Field	Type/ InitVal	Description
7:1	Reserved	RES	Reserved
0	ISR	RW 0x0	The ARM CPU reads this register to check the post masked interrupt sources. The register only holds its value for 1 cycle after read access and clears itself after that 1 cycle. The ARM CPU then needs to do a follow up write to enable it to latch future interrupt events.

Table 114: Peripheral Reset Register
Offset: 0x8000C000

Bits	Field	Type/ InitVal	Description
31:1	Reserved		Reserved
0	Prst	WO 0x0	Peripheral reset, active low. Asserted by the TWSI by writing 1 to this register. This register activates a 4-cycle length of active low pulse on the rising edge of Uclk to initialize peripheral devices to their default states. The register is automatically cleared to 0 at the end of reset.

Table 115: Peripheral Clock Divider Register
Offset: 0x8000C004

Bits	Field	Type/ InitVal	Description
31:2	Reserved		Reserved

Table 115: Peripheral Clock Divider Register (Continued)
Offset: 0x8000C004

Bits	Field	Type/ InitVal	Description
1:0	CLK_DIV	RW 0x0	Peripheral clock select. Uclk is generated based on the following equation: $Uclk = SYSCLK/(CLK_DIV+1)$ 00 = SYSCLK/1 01 = SYSCLK/2 10 = SYSCLK/4 11 = SYSCLK/8 Also see Table 28 , "Uclk Frequencies Based on SYSCLK and CLK_DIV," on page 61 .



A.7 General Purpose I/O Port (GPIO) Registers

A.7.1 GPIO Register Map

Table 116: GPIO Register Map

Register Name	Offset	Table and Page
GPIO Select[7:0] Register	0x8000D000	Table 117, p. 127
GPIO Select[15:8] Register	0x8000D004	Table 118, p. 128
Control Bidirectional GPIO Output Enable Pins Register	0x8000D008	Table 119, p. 129
GPIO Output Registers	0x8000D00C	Table 120, p. 129
GPIO Input Registers	0x8000D010	Table 121, p. 130
GPIO Interrupt Edge Trigger Select Register (IER)	0x8000D014	Table 122, p. 130
GPIO Input Interrupt Mask Register (IMR)	0x8000D018	Table 123, p. 130
GPIO Reset Select Register (RSR)	0x8000D01C	Table 124, p. 130
GPIO Interrupt Status Register (ISR)	0x8000D020	Table 125, p. 131
SW LED Output Enable Register	0x8000D024	Table 126, p. 131
SW LED Cycle[3:0] Register	0x8000D028	Table 127, p. 131
SW LED Cycle[7:4] Register	0x8000D02C	Table 128, p. 131
SW LED Cycle[11:8] Register	0x8000D030	Table 129, p. 132
SW LED Cycle[15:12] Register	0x8000D034	Table 130, p. 133
SW LED Duty Cycle[3:0] Register	0x8000D038	Table 131, p. 134
SW LED Duty Cycle[7:4] Register	0x8000D03C	Table 132, p. 134
SW LED Duty Cycle[11:8] Register	0x8000D040	Table 133, p. 135
SW LED Duty Cycle[15:12] Register	0x8000D044	Table 134, p. 136
LED Output Enable for Status Signals Register	0x8000D048	Table 135, p. 136
LED[3:0] Stretch Duration for Status Signals Register	0x8000D04C	Table 136, p. 137
Link/Active Cycle Register	0x8000D050	Table 137, p. 138

A.7.2 GPIO Registers

Table 29, "GPIO Port Pin Assignments," on page 66 provides a summary of the bit settings presented in Table 117 and Table 118.

Table 117: GPIO Select[7:0] Register
Offset: 0x8000D000

Bits	Field	Type/ InitVal	Description
15:14	GPIOSelect[7]	RW 0x01	Select the GPIO output source for pin GPIO[7]. 00 = Software controlled LED output SW_LED_OUT[7] 01 = GPIO output register GPIO_OUT[7] 10 = Reserved 11 = Reserved
13:12	GPIOSelect[6]	RW 0x01	Select the GPIO output source for pin GPIO[6]. 00 = Software controlled LED output SW_LED_OUT[6] 01 = GPIO output register GPIO_OUT[6] 10 = Reserved 11 = Reserved
11:10	GPIOSelect[5]	RW 0x01	Select the GPIO output source for pin GPIO[5]. 00 = Software controlled LED output SW_LED_OUT[5] 01 = GPIO output register GPIO_OUT[5] 10 = Reserved 11 = Reserved
9:8	GPIOSelect[4]	RW 0x01	Select the GPIO output source for pin GPIO[4]. 00 = Software controlled LED output SW_LED_OUT[4] 01 = GPIO output register GPIO_OUT[4] 10 = Reserved 11 = Reserved
7:6	GPIOSelect[3]	RW 0x01	Select the GPIO output source for pin GPIO[3]. 00 = UniMAC™ LED for status signal Link/Activity 01 = GPIO output register GPIO_OUT[3] 10 = Reserved 11 = Reserved
5:4	GPIOSelect[2]	RW 0x01	Select the GPIO output source for pin GPIO[2]. 00 = LED for status signal Act (logic OR of RxEn, TxDV) 01 = GPIO output register GPIO_OUT[2] 10 = Reserved 11 = Reserved
3:2	GPIOSelect[1]	RW 0x01	Select the GPIO output source for pin GPIO[1]. 00 = LED for status signal UniMAC™ Tx 01 = GPIO output register GPIO_OUT[1] 10 = Reserved 11 = Reserved



Table 117: GPIO Select[7:0] Register (Continued)
Offset: 0x8000D000

Bits	Field	Type/ InitVal	Description
1:0	GPIOSelect[0]	RW 0x01	Select the GPIO output source for pin GPIO[0] 00 = LED for status signal UniMAC™ Rx 01 = GPIO output register GPIO_OUT[0] 10 = Reserved 11 = Reserved

Table 118: GPIO Select[15:8] Register
Offset: 0x8000D004

Bits	Field	Type/ InitVal	Description
15:14	GPIOSelect[15]	RW 0x01	Select the GPIO output source for pin GPIO[15] 00 = TWSI Clock 01 = GPIO output register GPIO_OUT[15] 10 = Software controlled LED output SW_LED_OUT[14] 11 = Reserved
13:12	GPIOSelect[14]	RW 0x01	Select the GPIO output source for pin GPIO[14] 00 = TWSI Data 01 = GPIO output register GPIO_OUT[14] 10 = Software controlled LED output SW_LED_OUT[13] 11 = Reserved
11:10	GPIOSelect[13]	RW 0x01	Select the GPIO output source for pin GPIO[13] 00 = Reserved 01 = GPIO output register GPIO_OUT[13] 10 = Software controlled LED output SW_LED_OUT[12] 11 = Reserved
9:8	GPIOSelect[12]	RW 0x01	Select the GPIO output source for pin GPIO[12] NOTE: When the Watchdog Enable Register bit 0 WdEn is set to 1 (see Table 152 on page 145), the watchdog will use GPIO[12] pin no matter what the value set in GPIOSelect[12]. 00 = Reserved 01 = GPIO output register GPIO_OUT[12] 10 = Reserved 11 = Reserved
7:6	GPIOSelect[11]	RW 0x01	Select the GPIO output source for pin GPIO[11] 00 = Software controlled LED output SW_LED_OUT[11] 01 = GPIO output register GPIO_OUT[11] 10 = Reserved 11 = Reserved

Table 118: GPIO Select[15:8] Register (Continued)
Offset: 0x8000D004

Bits	Field	Type/ InitVal	Description
5:4	GPIOSelect[10]	RW 0x01	Select the GPIO output source for pin GPIO[10] 00 = Software controlled LED output SW_LED_OUT[10] 01 = GPIO output register GPIO_OUT[10] 10 = Reserved 11 = Reserved
3:2	GPIOSelect[9]	RW 0x01	Select the GPIO output source for pin GPIO[9] 00 = Software controlled LED output SW_LED_OUT[9] 01 = GPIO output register GPIO_OUT[9] 10 = Reserved 11 = Reserved
1:0	GPIOSelect[8]	RW 0x01	Select the GPIO output source for pin GPIO[8] 00 = Software controlled LED output SW_LED_OUT[8] 01 = GPIO output register GPIO_OUT[8] 10 = Reserved 11 = Reserved

Table 119: Control Bidirectional GPIO Output Enable Pins Register
Offset: 0x8000D008

Bits	Field	Type/ InitVal	Description
15:0	GPIOOutputEn	RW 0xFFFF	GPIO output enable 0 = GPIO output is enable, output mode 1 = GPIO output is disable, input mode

Table 120: GPIO Output Registers
Offset: 0x8000D00C

Bits	Field	Type/ InitVal	Description
15:0	GPIOOut	RW 0x0000	GPIO output registers. These can be output to GPIO pins by setting the GPIO_SELECT registers.



Table 121: GPIO Input Registers
Offset: 0x8000D010

Bits	Field	Type/ InitVal	Description
15:0	GPIOIn	RW NOTE: The initial value reflects the state of the GPIO[15:0] pins after reset.	GPIO input registers. These can be read through the CPU. NOTE: Bits 15:4 can be used to generate interrupts. Bits 3:0 can <i>not</i> be used as interrupt bits. Bits 3:0 sample the GPIO inputs and can be read by CPU, but they are not interrupt sources.

Table 122: GPIO Interrupt Edge Trigger Select Register (IER)
Offset: 0x8000D014

Bits	Field	Type/ InitVal	Description
15:4	GPIOIer	RW 0x000	GPIO Interrupt edge trigger select. 0 = From 0->1 cause an interrupt event. 1 = From 1->0 cause an interrupt event.
3:0	Reserved	0x0	Reserved

Table 123: GPIO Input Interrupt Mask Register (IMR)
Offset: 0x8000D018

Bits	Field	Type/ InitVal	Description
15:4	GPIOImr	RW 0x000	GPIO Interrupt mask register 0 = Input interrupt for corresponding GPIO is disable. 1 = Input interrupt for corresponding GPIO is enable.
3:0	Reserved	0x0	Reserved

Table 124: GPIO Reset Select Register (RSR)
Offset: 0x8000D01C

Bits	Field	Type/ InitVal	Description
15:4	GPIORsr	RW 0x000	GPIO Interrupt Reset Select register 0 = Write to ISR clears GPIOIsr. 1 = Read to ISR clears GPIOIsr.
3:0	Reserved	0x0	Reserved

Table 125: GPIO Interrupt Status Register (ISR)
Offset: 0x8000D020

Bits	Field	Type/ InitVal	Description
15:4	GPIOIsr	RW 0x000	GPIO Interrupt Status register 0 = Interrupt status is clear. 1 = Interrupt status is set. NOTE: These bits cannot be set from a read or write to this field, only from an interrupt event. They are set according to the GPIO Reset Select Register (RSR) GPIORsr bit, 15:4 (see Table 124 on page 130). When the GPIORsr bit is set, this field is cleared on Read; When the GPIORsr bit is cleared, this field is cleared on Write.
3:0	Reserved	RES 0x0	Reserved

Table 126: SW LED Output Enable Register
Offset: 0x8000D024

Bits	Field	Type/ InitVal	Description
15	Reserved	RES	Reserved
14:4	SWLedEn	RW 0x0000	Software controlled LED enable register 0 = The corresponding software controlled. LED signal is disabled. 1 = The corresponding software controlled. LED signal is enabled. The LED enable signals are used to control the slew rate for LED signals.
3:0	Reserved	RES	Reserved

Table 127: SW LED Cycle[3:0] Register
Offset: 0x8000D028

Bits	Field	Type/ InitVal	Description
15:0	Reserved	RES	Reserved

Table 128: SW LED Cycle[7:4] Register
Offset: 0x8000D02C

Bits	Field	Type/ InitVal	Description
15:12	SWLedCycle[7]	RW 0x0	Control the blink rate for the LED[7] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 4'b1111 = Reserved

Table 128: SW LED Cycle[7:4] Register (Continued)
Offset: 0x8000D02C

Bits	Field	Type/ InitVal	Description
11:8	SWLedCycle[6]	RW 0x0	Control the blink rate for the LED[6] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 4'b1111 = Reserved
7:4	SWLedCycle[5]	RW 0x0	Control the blink rate for the LED[5] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 4'b1111 = Reserved
3:0	SWLedCycle[4]	RW 0x0	Control the blink rate for the LED[4] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 4'b1111 = Reserved

Table 129: SW LED Cycle[11:8] Register
Offset: 0x8000D030

Bits	Field	Type/ InitVal	Description
15:12	SWLedCycle[11]	RW 0x0	Control the blink rate for the LED[11] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved
11:8	SWLedCycle[10]	RW 0x0	Control the blink rate for the LED[10] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved

Table 129: SW LED Cycle[11:8] Register (Continued)
Offset: 0x8000D030

Bits	Field	Type/ InitVal	Description
7:4	SWLedCycle[9]	RW 0x0	Control the blink rate for the LED[9] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved
3:0	SWLedCycle[8]	RW 0x0	Control the blink rate for the LED[8] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved

Table 130: SW LED Cycle[15:12] Register
Offset: 0x8000D034

Bits	Field	Type/ InitVal	Description
15:12	Reserved	RES	Reserved
11:8	SWLedCycle[14]	RW 0x0	Control the blink rate for the LED[14] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved
7:4	SWLedCycle[13]	RW 0x0	Control the blink rate for the LED[13] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved



Table 130: SW LED Cycle[15:12] Register (Continued)
Offset: 0x8000D034

Bits	Field	Type/ InitVal	Description
3:0	SWLedCycle[12]	RW 0x0	Control the blink rate for the LED[12] 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved

Table 131: SW LED Duty Cycle[3:0] Register
Offset: 0x8000D038

Bits	Field	Type/ InitVal	Description
15:0	Reserved	RES	Reserved

Table 132: SW LED Duty Cycle[7:4] Register
Offset: 0x8000D03C

Bits	Field	Type/ InitVal	Description
15:12	SWLedDuty[7]	RW 0x0	Control duty factor for the LED[7] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved
11:8	SWLedDuty[6]	RW 0x0	Control duty factor for the LED[6] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved
7:4	SWLedDuty[5]	RW 0x0	Control duty factor for the LED[5] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved

Table 132: SW LED Duty Cycle[7:4] Register (Continued)
Offset: 0x8000D03C

Bits	Field	Type/ InitVal	Description
3:0	SWLedDuty[4]	RW 0x0	Control duty factor for the LED[4] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved

Table 133: SW LED Duty Cycle[11:8] Register
Offset: 0x8000D040

Bits	Field	Type/ InitVal	Description
15:12	SWLedDuty[11]	RW 0x0	Control duty factor for the LED[11] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved
11:8	SWLedDuty[10]	RW 0x0	Control duty factor for the LED[10] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved
7:4	SWLedDuty[9]	RW 0x0	Control duty factor for the LED[9] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved
3:0	SWLedDuty[8]	RW 0x0	Control duty factor for the LED[8] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved

Table 134: SW LED Duty Cycle[15:12] Register
Offset: 0x8000D044

Bits	Field	Type/ InitVal	Description
15:12	Reserved	RES	Reserved
11:8	SWLedDuty[14]	RW 0x0	Control duty factor for the LED[14] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved
7:4	SWLedDuty[13]	RW 0x0	Control duty factor for the LED[13] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved
3:0	SWLedDuty[12]	RW 0x0	Control duty factor for the LED[12] 0000 = 1/2 0001 = 1/4 0010 = 1/8 0011 = 1/16 0100 = 1/32 0101 to 1111 = Reserved

Table 135: LED Output Enable for Status Signals Register
Offset: 0x8000D048

Bits	Field	Type/ InitVal	Description
15:4	Reserved	0x000	Reserved
3:0	StatLedEn	RW 0x0000	LED enable register for status signals 0 = The corresponding LED driving signal for status signal is disabled. 1 = The corresponding LED driving signal for status signal is enabled. The LED enable signals are used to control the slew rate for LED signals.

Table 136: LED[3:0] Stretch Duration for Status Signals Register
Offset: 0x8000D04C

Bits	Field	Type/ InitVal	Description
15:12	LedStretch[3]	RW 0x0	Pulse stretch duration for the status signal MRXDV to drive LED. 0000 = no pulse stretching 0001 = 17 ms to 19 ms 0010 = 36 ms to 37 ms 0011 = 73 ms to 74 ms 0100 = 148 ms to 149 ms 0101 = 297 ms to 298 ms 0110 = 596 ms 0111 = 1191 ms to 1192 ms 1000 = 2382 ms to 2384 ms 1001 to 4'b1111 = Reserved
11:8	LedStretch[2]	RW 0x0	Pulse stretch duration for the status signal MTXEN to drive LED. 0000 = no pulse stretching 0001 = 17 ms to 19 ms 0010 = 36 ms to 37 ms 0011 = 73 ms to 74 ms 0100 = 148ms to 149 ms 0101 = 297 ms to 298 ms 0110 = 596 ms 0111 = 1191 ms to 1192 ms 1000 = 2382 ms to 2384 ms 1001 to 4'b1111 = Reserved
7:4	LedStretch[1]	RW 0x0	Pulse stretch duration for the status signal RX_RDY to drive LED. 0000 = no pulse stretching 0001 = 17 ms to 19 ms 0010 = 36 ms to 37 ms 0011 = 73 ms to 74 ms 0100 = 148 ms to 149 ms 0101 = 297 ms to 298 ms 0110 = 596 ms 0111 = 1191ms to 1192 ms 1000 = 2382 ms to 2384 ms 1001 to 4'b1111 = Reserved
3:0	LedStretch[0]	RW 0x0	Pulse stretch duration for the status signal TX_PE to drive LED. 0000 = no pulse stretching 0001 = 17 ms to 19 ms 0010 = 36 ms to 37 ms 0011 = 73 ms to 74 ms 0100 = 148 ms to 149 ms 0101 = 297 ms to 298 ms 0110 = 596 ms 0111 = 1191 ms to 1192 ms 1000 = 2382 ms to 2384 ms 1001 to 1111 = Reserved



Table 137: Link/Active Cycle Register
Offset: 0x8000D050

Bits	Field	Type/ InitVal	Description
15:4	Reserved	RES	Reserved.
3:0	LinkActiveCycle	RW 0x0	Link/Active Cycle 0000 = 37 ms 0001 = 74 ms 0010 = 149 ms 0011 = 298 ms 0100 = 596 ms 0101 = 1192 ms 0110 to 1111 = Reserved

A.8 UART Registers

The UART registers are 8 bits in length. The UART registers are aligned to the 32-bit word boundary. Each 8-bit register occupies byte 0 per word.

The UART interface is designed so that offsets 0x8000C840, 0x8000C844, and 0x8000C848 are each used by more than one register.

A.8.1 UART Register Map

Table 138: UART Register Map Table

Register Name	Access	Offset	Table and Page
UART Base Clock Select Register	Read Write	0x8000C008	Table 139, p. 139
Receive Buffer Register (RBR)	Read Only	0x8000C840	Table 140, p. 140
Transmit Holding Register (THR)	Write Only	0x8000C840	Table 141, p. 140
Divisor Latch Low (DLL) Register ¹	Read Write	0x8000C840	Table 142, p. 140
Interrupt Enable Register (IER)	Read Write	0x8000C844	Table 143, p. 141
Divisor Latch High (DLH) Register ¹	Read Write	0x8000C844	Table 144, p. 141
Interrupt Identity Register (IIR)	Read Only	0x8000C848	Table 145, p. 141
FIFO Control Register (FCR)	Write Only	0x8000C848	Table 146, p. 142
Line Control Register (LCR)	Read Write	0x8000C84C	Table 147, p. 143
Modem Control Register (MCR)	Read Write	0x8000C850	Table 148, p. 143
Line Status Register (LSR)	Read Only	0x8000C854	Table 149, p. 144
Scratch Pad Register (SCR)	Read Write	0x8000C85C	Table 150, p. 144

¹ The DivLatchRdWrt bit 7 is the most significant bit of the Line Control Register (LCR) (see Table 147 on page 143). This bit must be set to '1' to address the Divisor Latch High (DLH) and Divisor Latch Low (DLL) registers.

A.8.2 UART Registers

Table 139: UART Base Clock Select Register
Offset: 0x8000C008

Bits	Field	Type/ InitVal	Description
31:1	Reserved	RES	Reserved
0	UARTClkSel	RW 0x0	8-MHz Clock generation source select. 0 = 8 MHz supplied to the UART block. 1 = 50 MHz supplied to the UART block. Also see Table 28, "Uclk Frequencies Based on SYSCLK and CLK_DIV," on page 61.



Table 140: Receive Buffer Register (RBR)
Offset: 0x8000C840

NOTE: DivLatchRdWrt bit 7 of the Line Control Register (LCR), [Table 147 on page 143](#), 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 149, p. 144). In the non-FIFO mode (fifo_mode = 0), the data in the RBR must be read before the next data arrives; otherwise it will be overwritten, resulting in an overrun error. In the FIFO mode (fifo_mode = 1), this register accesses the head of the receive FIFO. If the receive FIFO is full and this register is not read before the next data word arrives, then the data already in the FIFO will be preserved but any incoming data will be lost.

Table 141: Transmit Holding Register (THR)
Offset: 0x8000C840

NOTE: DivLatchRdWrt bit 7 of the Line Control Register (LCR), [Table 147 on page 143](#), 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 149, p. 144), data can be written to the LSR<TxEmpty> to be transmitted from the serial port. If FIFOs are not enabled and THRE is set, writing a single word to the THR resets the THRE and any additional writes to the THR before the THRE is set again causes the THR data to be overwritten. If FIFOs are enabled and THRE is set, up to 16 words of data may be written to the THR before the FIFO is full. Any attempt to write data when the FIFO is full results in the write data being lost.

Table 142: Divisor Latch Low (DLL) Register
Offset: 0x8000C840

NOTE: DivLatchRdWrt bit 7 of the Line Control Register (LCR), [Table 147 on page 143](#), must be set to "1".

Bits	Field	Type/ InitVal	Description
7:0	DivLatchLow	RW 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 Div-LatchRdWrt 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. $\text{baud} = (\text{clock freq}) / (16 * \text{divisor})$

Table 143: Interrupt Enable Register (IER)
Offset: 0x8000C844

NOTE: DivLatchRdWrt bit 7 of the Line Control Register (LCR), Table 147 on page 143, must be set to "0".

Bits	Field	Type/ InitVal	Description
7:4	Reserved	RES 0x0	Reserved
3	ModStatIntEn	RW 0x0	Enable Modem Status Interrupt (EDSSI). 0 = Disable interrupt 1 = Enable interrupt
2	RxLineStatIntEn	RW 0x0	Enable Receiver Line Status Interrupt (ELSI). 0 = Disable interrupt 1 = Enable interrupt
1	TxHoldIntEn	RW 0x0	Enable Transmitter Holding Register Empty Interrupt (ETBEI). 0 = Disable interrupt 1 = Enable interrupt
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.

Table 144: Divisor Latch High (DLH) Register
Offset: 0x8000C844

NOTE: DivLatchRdWrt bit 7 of the Line Control Register (LCR), Table 147 on page 143, must be set to "1".

Bits	Field	Type/ InitVal	Description
7:0	DivLatchHigh	RW 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 Div-LatchRdWrt 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. $\text{baud} = (\text{clock freq}) / (16 * \text{divisor})$

Table 145: Interrupt Identity Register (IIR)
Offset: 0x8000C848

Bits	Field	Type/ InitVal	Description
7:6	FIFOEn	RO 0x0	FIFO Enable. 00 = FIFOs are disabled (default in FIFO mode) 11 = FIFOs are enabled



Table 145: Interrupt Identity Register (IIR) (Continued)
Offset: 0x8000C848

Bits	Field	Type/ InitVal	Description
5:4	Reserved	RO 0x0	Reserved
3:0	InterruptID	RO 0x0	11 FIFO Enabled Interrupt ID 0000 = Modem Status Changed 0001 = No interrupt pending 0010 = THR empty 0100 = Received Data available 0110 = Receiver Status 1100 = Character Time Out

Table 146: FIFO Control Register (FCR)
Offset: 0x8000C848

Bits	Field	Type/ InitVal	Description
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
5:4	Reserved	RES	Reserved
3	DMAMode	WO 0x0	DMA mode. 0 = Single transfer DMA mode 0 1 = Multi transfer DMA mode 1
2	TxFIFOReset	WO 0x0	Transmit FIFO reset. 0 = Do not flush data from the transmit FIFO 1 = Flush data from the transmit FIFO
1	RxFIFOReset	WO 0x0	Receive FIFO reset. 0 = Do not flush data from the receive FIFO 1 = Flush data from the receive FIFO
0	FIFOEn	WO 0x0	Enable transmit and receive FIFOs. This register controls the read and write data FIFO operation and the mode of operation for the DMA signals txrdy_n and rxrdy_n. 0 = Disable FIFOs 1 = Enable FIFOs

Table 147: Line Control Register (LCR)
Offset: 0x8000C84C

Bits	Field	Type/ InitVal	Description
7	DivLatchRdWrt	RW 0x0	This bit enables reading and writing of the Divisor Latch Low and High registers (see Table 142, p. 140 and Table 144, p. 141) to set the baud rate of the UART.
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
5	Reserved	RES	Reserved
4	EPS	RW 0x0	Even or odd parity select. 0 = Odd parity 1 = Even parity
3	PEN	RW 0x0	Parity enable. 0 = Parity disabled 1 = Parity enabled
2	Stop	RW 0x0	Stop bits transmitted. 0 = 1 bit 1 = 2 bits
1:0	WLS	RW 0x0	Number of bits per character. 00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits

Table 148: Modem Control Register (MCR)
Offset: 0x8000C850

Bits	Field	Type/ InitVal	Description
7:5	Reserved	RES	Reserved
4	Loopback	RW 0x0	Loopback. The Loopback bit loops the data on the <i>sout</i> line back to the <i>sin</i> line. In this mode all the interrupts are fully functional. This feature is used for diagnostic purposes. 0 = No loopback 1 = Loopback
3:2	Reserved	RES	Reserved
1:0	Reserved	RES	Reserved



Table 149: Line Status Register (LSR)
Offset: 0x8000C854

Bits	Field	Type/ InitVal	Description
7	RxFIFOErr	RO 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 LSR is read.
6	TxEEmpty	RO 0x1	Transmitter Empty bit. In FIFO mode, this bit is set whenever the Transmitter Holding Register, the Transmitter Shift Register, and the FIFO are all empty.
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) is transmitted to the transmit shift register. 0 = Do not accept a new character for transmission 1 = Accept a new character for transmission
4	BI	RO 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 LSR clears the BI bit.
3	FrameErr	RO 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 LSR is performed.
2	ParErr	RO 0x0	Parity Error. The ParErr bit indicates a parity error in the receiver if the PEN bit in the LCR is set. In the FIFO mode, since the parity error is associated with a character received, it is revealed when the character with the bad parity comes to the head of the FIFO.
1	OverRunErr	RO 0x0	Overrun Error 0 = No overrun 1 = Overrun error has occurred
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.

Table 150: Scratch Pad Register (SCR)
Offset: 0x8000C85C

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.

A.9 Watchdog Register Description



Note

The Watchdog timer and relating registers do not operate in 88E6208/88E6218 silicon rev. A.

The following registers provide configuration and control for the watchdog.

Table 151: Watchdog Register Map

Register Name	Offset	Table and Page
Watchdog Enable Register	8000D800	Table 152, p. 145
Watchdog Timer Interval Register	8000D802	Table 153, p. 145
Watchdog Timer Counter Clear Register	8000D804	Table 154, p. 145

Table 152: Watchdog Enable Register

Offset: 8000D800

Bits	Field	Type/ InitVal	Description
15:6	Reserved	RO 0x0	Read only 0.
5:1	Reserved	RW 0x8	Reserved Bit 4 must be 1.
0	WdEn	RW 0x1	Watchdog timer enable 1 = Enable 0 = Disable

Table 153: Watchdog Timer Interval Register

Offset: 8000D802

Bits	Field	Type/ InitVal	Description
15:2	Reserved	WO 0x0	A "0" is returned after a read.
1:0	WdTime	WO 0x00	Watchdog timer cycle setting 00 = 1 second 01 = 2 second 10 = 3 second 11 = 4 second A "0" is returned after a read.

Table 154: Watchdog Timer Counter Clear Register

Offset: 8000D804

Bits	Field	Type/ InitVal	Description
15:1	Reserved	RO 0x0	Read only 0.



Table 154: Watchdog Timer Counter Clear Register (Continued)
Offset: 8000D804

Bits	Field	Type/ InitVal	Description
0	WdClear	RW 0x0	Watchdog timer counter clear. This bit is used to clear the watchdog at periodic intervals as set in the Watchdog Timer Interval Register (see Table 153). 1 = Clear 0 = Do not clear This bit is automatically cleared upon a write of "1".

Appendix B. Memory Controller Application Examples

This appendix provides Flash and ROM application examples for the SDRAM and Flash/ROM Memory controller.

B.1 Application Examples

B.1.1 Memory Controller with SDRAM

Figure 17 through Figure 19 provide application examples for the Memory controller with SDRAM.

Figure 17: 1X32 SDRAM (88E6218 Only)

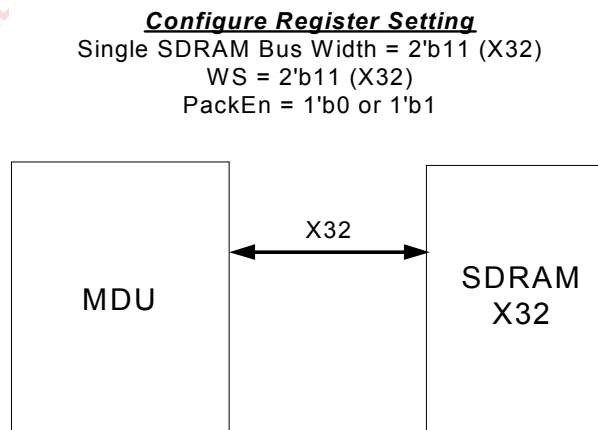


Figure 18: 2X16 SDRAM (88E6218 Only)

Configure Register Setting

Single SDRAM Bus Width = 2'b10 (X16)

WS = 2'b11 (X32)

PackEn = 1'b0 or 1'b1

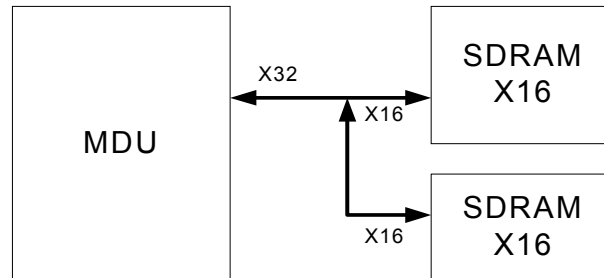


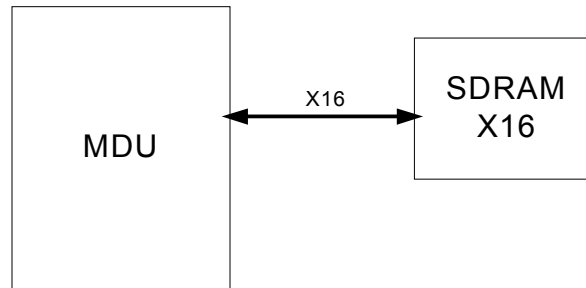
Figure 19: 1X16 SDRAM

Configure Register Setting

Single SDRAM Bus Width = 2'b10 (X16)

WS = 2'b10 (X16)

PackEn = 1'b1



B.1.2 Memory Controller with FLASH/ROM

Figure 20 through Figure 22 provide application examples for the Memory controller with Flash/ROM.

Figure 20: 1X16 Flash/ROM

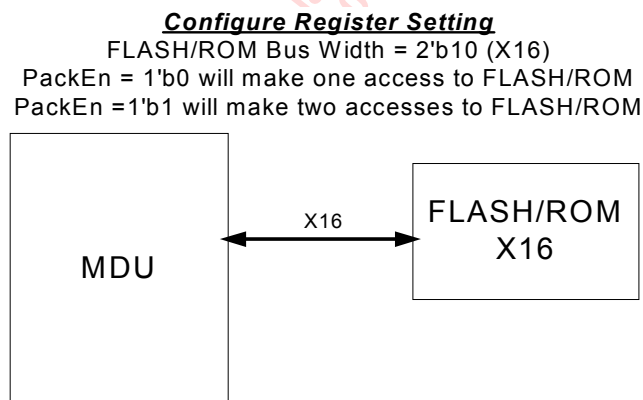


Figure 21: 1X8 Flash/ROM

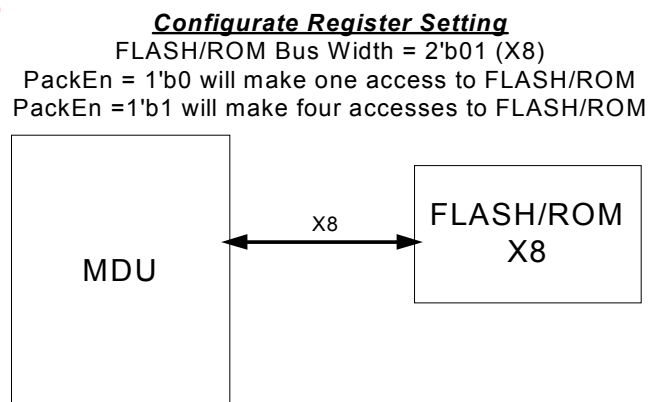
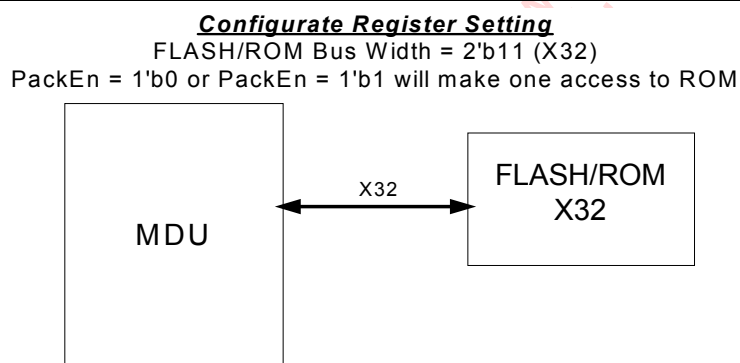


Figure 22: 1X32 Flash/ROM (88E6218 Only)





Appendix C. List of Abbreviations

The following acronyms and terms are used in the three part *Link Street™ 88E6208/88E6218 SOHO Gateway SOC with ARM9E™ CPU, 10/100 Switch and PHYs* datasheet set.

ATU	Address Translation Unit
BAR	Base Address Register
BPDU	Bridge Protocol Data Unit
CRC	Cyclic Redundancy Check
DA	Destination Address
DPV	Destination Port Vector
FCS	Frame Check Sequence
GPIO	General Purpose Input/Output
GPP	General Purpose Port
GPU	General Purpose Unit
IFG	Inter Frame Gap
IGMP	Internet Group Management Protocol
IO	Input/Output
LBU	Low Speed Bus
MAC	Media Access Controller
PAV	Port Association Vector
QoS	Quality of Service
SFD	Start of Frame Delimiter
TWSI	Two-Wire Serial Interface
UART	Universal Asynchronous Receiver/Transmitter
UniMAC™	Unified Media Access Controller
VCT	Virtual Cable Tester™
VID	VLAN ID
VTU	VLAN Translation Unit

Appendix D. Revision History

Table 155: Revision History

Document Type	Rev. #	Date
Preliminary	Rev. –	April 6, 2003
First release of this document		



MOVING FORWARD
FASTER®

Marvell Semiconductor, Inc.

700 First Avenue
Sunnyvale, CA 94089

Phone 408.222.2500
Fax 408.752.9028

www.marvell.com

US and Worldwide Offices

Marvell Semiconductor, Inc.

700 First Avenue
Sunnyvale, CA 94089
Tel: 1.408.222.2500
Fax: 1.408.752.9028

Marvell Asia Pte, Ltd.

151 Lorong Chuan, #02-05
New Tech Park
Singapore 556741
Tel: 65.6756.1600
Fax: 65.6756.7600

Marvell Japan K.K.

Shinjuku Center Bldg. 50F
1-25-1, Nishi-Shinjuku, Shinjuku-ku
Tokyo 163-0650
Tel: 81.(0).3.5324.0355
Fax: 81.(0).3.5324.0354

Marvell Semiconductor Israel, Ltd.

Moshav Manof
D.N. Misgav 20184
Israel
Tel: 972.4.999.9555
Fax: 972.4.999.9574

Worldwide Sales Offices

Western US Sales Office

Marvell
700 First Avenue
Sunnyvale, CA 94089
Tel: 1.408.222.2500
Fax: 1.408.752.9028
Sales Fax: 1.408.752.9029

Central US Sales Office

Marvell
11709 Boulder Lane, Ste. #220
Austin, TX 78726
Tel: 1.512.336.1551
Fax: 1.512.336.1552

Eastern US/Canada Sales Office

Marvell
Knox Trail Office Bldg.
2352 Main Street
Concord, MA 01742
Tel: 1.978.461.0563
Tel: 1.978.461.1406
Fax: 1.978.461.1405

Europe Sales Office

Marvell
3 Clifton Court
Corner Hall
Hemel Hempstead
Hertfordshire, HP3 9XY
United Kingdom
Tel: 44.(0).1442.211668
Fax: 44.(0).1442.211543

Marvell

Fagerstagatan 4
163 08 Spanga
Stockholm, Sweden
Tel: 46.16.146348
Fax: 46.16.482425

Marvell

5 Rue Poincare
56400 Le Bono
France
Tel: 33.297.579697
Fax: 33.297.578933

Israel Sales Office

Marvell
Ofek Center Bldg. 2, Floor 2
Northern Industrial Zone
LOD 71293
Israel
Tel: 972.8.924.7555
Fax: 972.8.924.7554

China Sales Office

Marvell
5J, 1800 Zhong Shan West Road
Shanghai, China 200233
Tel: 86.21.6440.1350
Fax: 86.21.6440.0799

Japan Sales Office

Marvell
Helios Kannai Bldg. 12F
3-21-2 Motohama-cho, Naka-ku
Yokohama, Kanagawa
Japan 231-0004
Tel: 81.45.222.8811
Fax: 81.45.222.8812

Taiwan Sales Office

Marvell
12F-1, 128 Sec. 3
Ming Sheng East Road
Taipei 105
Taiwan, R.O.C.
Tel: 886.2.8712.5700
Fax: 886.2.8712.5707

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