MARVELL™

Migrating from the GT6426x Devices to the MV6436x Devices

Doc. No. MV-S100648-00, Rev. A

February 6, 2002

MOVING FORWARD
FASTER™

Document Status	
Advanced Information	This datasheet contains design specifications for initial product development. Specifications may change without notice. Contact Marvell Field Application Engineers for more information.
Preliminary Information	This datasheet 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 datasheet 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:	
Final Information	Technical Publication: Rev. A

Preliminary or Advanced Information

This document provides preliminary or advanced information about the products described. All specifications described herein are based on design goals only. **Do not use for final design.** Visit Marvell's web site at www.marvell.com or call 1-866-674-7253 for the latest information on Marvell products.

Disclaimer

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

Marvell Semiconductor, Inc.
2350 Zanker Road, San Jose, CA 95131
Phone: (408) 367-1400, Fax: (408) 367-140

Table of Contents

List of Tables

List of Figures

Section 1. Introduction

This document describes the system software implications of the new/updated features of the GT6426x devices that differ from previous GT6426x devices.

The MV6436x datasheets and related documents discuss the detailed changes themselves.

Note

In any conflict in information between the datasheet and this document, the datasheet's information is correct.

This document identifies and discusses some possible strategies for migrating systems from previous environments to the new MV6436x family devices.

Section 2. Overview

Changes requiring a thorough treatment will be fully discussed in subsequent sections. The differences are organized and described according to the GT6426x interfaces or internal units.

- CPU interface
- SDRAM interface
- Device interface
- PCI interface
- Communication interfaces
- IDMA unit
- Interrupt controller
- Integrated SRAM
- General Purpose Port

Figure 1 describes the GT6426x interfaces.

Figure 1: GT6426x Interfaces

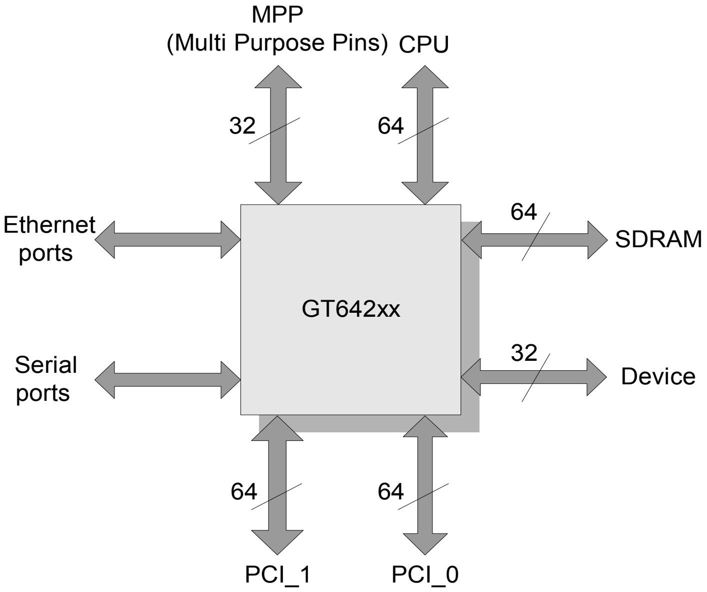
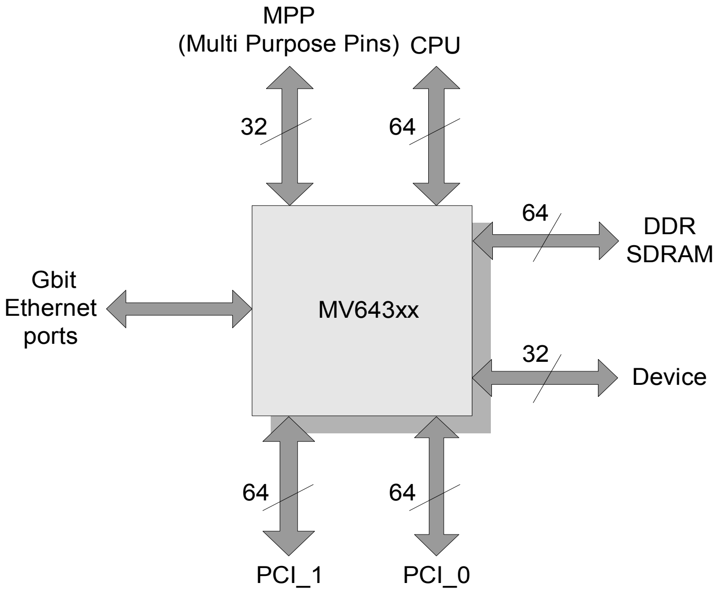


Figure 2 describes the MV6436x interfaces.

Figure 2: MV6436x Interfaces



Note

Unless cited otherwise within this document, a reference to the GT6426x devices is a reference to the GT6426x/A/B.

Section 3. CPU Interface

3.1 Address Decoding

The MV6436x CPU interface address decoding is NOT software compatible with GT6426x CPU interface address decoding scheme. Base and Top registers no longer define the address windows. Instead, the address windows in MV6436x are defined by Base and Size registers, similar to the PCI address-decoding scheme. Moreover the MV6436x support 36-bits addressing.

In the GT6426x devices, two registers define each address window - Low and High. The CPU address is compared with the values in the various CPU Low and High Decode registers.

The GT6426x CPU address windows are defined in 1 Mbyte granularity.

- The Base Address window is 12-bit wide, corresponding to CPU address bits [0:11].
- The High window is 12-bit wide, corresponds to address bits[0:11].

The MV6436x CPU address windows are defined in 64 KB granularity and can be configured up to 4 GB (except of the integrated SRAM which is fixed 256 KB).

- The Base Address is 20-bits wide, corresponding to CPU address bits [0:19]
- The Size is 16-bit wide, corresponding to address bits] [4:19].

The Size register must be programmed as a set of ‘1’s (staring from the LSB) followed by a set of ‘0’s. The set of ‘1’s define the window size. For example, if Size [15:0] is set to 0x03ff, the size is defined as 64Mbyte.

For example, configure SCS[1] to address 0x0.0A00.0000 – 0x0.0BFF.FFFF (32 MB size)

In the GT6426xthe address appears as:

31	30	29	28	27	26	25	24	23	22	21	20	CPU Address Bits
0	0	0	0	1	0	1	0	0	0	0	0	Low Decode Register
0	0	0	0	1	0	1	1	1	1	1	1	High Decode Register

- SCS[1]* Low Decode Address 0x208 = 0xA0
- SCS[1]* High Decode Address 0x210 = 0xBF

In MV6436xthe address appears as:.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	CPU Address Bits
0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	Base Address Register
				0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	Size Register

- CS[1]* Base Address 0x208 = 0xA00
- CS[1]* Size 0x210 = 0x1FF

Also, the default memory map in the MV6436x devices was changed to support an additional address window for the Integrated SRAM (see “Integrated SRAM” on page 41). This additional default internal

SRAM address window (0x4200.0000 to 0x4203.FFFF) was not implemented the GT6426x devices. SW porting must make sure this window is not being used by any other resource or close/move this window.

The GT6426x devices implement two address windows (CPU 0/1) to map the IDMA and communication units to the CPU interface. These windows were implemented to enable the IDMA and communication units to access additional slaves on the CPU interface in multi-GT (multi-slave) modes.

The MV6436x does not implement the CPU 0/1 address decode windows. Instead, each of the Gigabit Ethernet, MPSCs, and the IDMA units interfaces implements its own address decoding mechanism (see “Communication Interfaces” on page 32 and “Interrupt Controller” on page 39). This mechanism enables the IDMA and communication units to access additional slaves on the CPU interface in multi-GT (multi-slave) mode and to any other interface.

3.2 CPU Synchronization Barrier

3.2.1 GT6426x Synchronization Barrier Implementation

The GT6426x devices implement two CPU synchronization barrier registers (offsets 0xC0 and 0xC8). One barrier register for each PCI interface. A CPU read from on of these registers creates a synchronization barrier cycle. When there is no posted write data in the buffers, the CPU read returns random data that should be ignored. The CPU interface treats PCI I/O reads and configuration reads as synchronization barrier cycles. These reads receive a response once no posted data remains within the PCI slave write buffer. To disable these sync barrier, set the CPU Configuration register’s `ConfSBDis` and `IOSBDis` bits [29:28] to ‘1’.

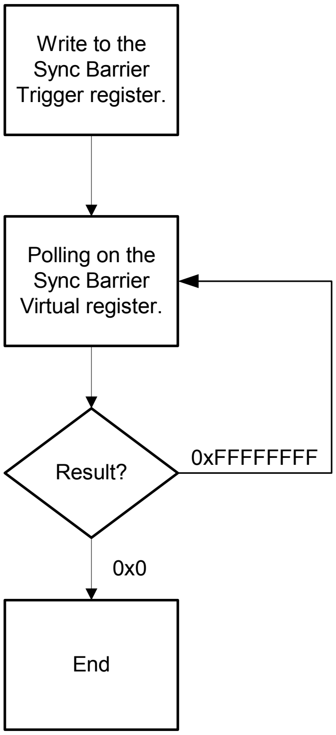
3.2.2 MV6436x Synchronization Barrier Implementation

The MV6436x synchronization barrier implementation is different than the GT6426x implementation.

The MV6436x implementation is register polling based, rather than a single read waiting for resolution. The MV6436x implements two sets of CPU sync barrier register. Each set contains a CPU Sync Barrier Trigger register (offsets 0xC0 and 0xD0), and a CPU Sync Barrier Virtual register (offsets 0xC8 and 0xD8). Two CPUs (one per each CPU) can use the two sets in a multi-CPU configuration.

For a CPU to activate a sync barrier, it first needs to write to the Trigger register and then perform read polling on the Virtual register. The trigger register is 4-bit wide. Each bit defines which buffers must be flushed. A write to the CPU Sync Barrier Trigger register triggers the sync barrier state machine. A subsequent read polling on the CPU Sync Barrier Virtual register results in value of 0xffff.ffff, until the sync action completes. As soon as the buffers are flushed, a read of the Virtual register results in value of 0x0. The user can set more then one bit in the trigger register, in this case the synchronization barrier mechanism will wait for the last resource to be flushed before returning data of 0x0on reads from the Sync Barrier Virtual register.

Figure 3: MV6436x Synchronization Barrier Implementation


Write to the Sync Barrier Trigger register.
Polling on the Sync Barrier Virtual register.
Result?
0xFFFFFFFF
0x0
End

Moreover, the GT6426x CPU synchronization barrier is not applicable when using cache coherency. This issue was solved in the MV6436x and the CPU synchronization barrier guarantees that the snoop queue is empty as well as the PCI slave buffer.

3.3 Multi-CPU and SMP (Symmetric Multi Processing) Support

3.3.1 SMP Boot-sequence

The GT-6426x Rev0 does not support SMP Boot-up sequence. To prevent a race between multiple CPU boot code executions, an external logic must be implemented.

The GT-6426x RevA and GT-6436x support SMP Boot-up sequence. After reset, the CPU1 arbitration is disabled, meaning that the arbiter will not grant the 60x bus CPU1. This allows CPU0 to boot first and initialize the system. Upon completing this operation, CPU0 enables the CPU1 arbitration by clearing the CPU Master Control register's MaskBR1 bit [9] (Offset 0x160).

Note

The SMP Boot sequence feature is only applicable when using the internal arbiter. To enable the internal arbiter, the CPU Master Control register's `IntArb` bit [8] to '1' must be set (by sampling at reset or serial ROM initialization).

3.3.2 Interrupts

The GT-6426x devices support only one interrupt signal to the CPU interface that can be masked. The GT-6436x has two interrupt pins, one per each CPU. Each interrupt has its own mask register.

Note

For more information, see “Interrupt Controller” on page 39.

Moreover, the GT-6436x has two 8-bit wide Doorbell registers (offsets 0x214 and 0x224), one per each CPU. The Doorbell registers can be used for CPU-to-CPU interrupt generation or for external PCI device to CPU interrupt generation, or even for the CPU to interrupt itself. When the corresponding bit in a Doorbell Mask registers (offsets 0x234 and 0x23C) is set to ‘1’, set the corresponding bit in the CPU doorbell register (offsets 0x214 and 0x224). Additionally, there is a separate pair of Doorbell Clear registers (offsets 0x21C and 0x22C), one per each CPU, that are used by the CPUs to clear doorbell interrupts. A CPU writes a value of ‘1’ to the Clear Doorbell register bit which clears the corresponding interrupt.

Notes

- Each CPU can only clear it’s own Clear Doorbell register.
- The Doorbell Interrupt feature is applicable only when using the internal arbiter.

3.3.3 Semaphores

The GT-6436x supports eight Semaphore registers (offsets 0x244 – 0x27C). These semaphores can be used as a lock mechanism between the two CPUs and even between the CPUs and an external PCI agent. The first owner to read a Semaphore register receives it’s own ID. This owner now controls this semaphore. Once locked by one of the three possible owners, a read of the Semaphore register by any of the other two possible owners returns the owner ID assigned to the current owner. To unlock a semaphore, the owner writes back a value of 0xff.

For example, communication interface in SMP system can only be accessed by only one CPU at a certain time. before any access to the descriprots list (ring) the CPU must read one of the semaphore registers that is reserved for this use. If this read’s returned data is equals to its own ID (the CPU ID can be read from the "who am I" register, see section below), the CPU can access the descriptors list. If the data returned for the semaphore register is not equal to the CPU ID, it should poll the semaphore register until it gets it own ID from this register.

Note

The semaphore feature is only applicable when using the internal arbiter. To enable the internal arbiter, the CPU Master Control register’s `IntArb` bit [8] to ’1’ must be set (by sampling at reset or serial ROM initialization).

3.3.4 “Who am I" register

The GT-6436x includes an “Who Am I” register (offset 0x200). This is a read only register and it may help the software running on the CPU to identify itself.

- A CPU0 read from this register results in a value of 0x0.
- A CPU1 read results in a value of 0x1.
- An external PCI agent read results in a value of 0x2.

Note

The “Who am I” feature is only applicable when using the internal arbiter. To enable the internal arbiter, the CPU Master Control register’s `IntArb` bit [8] to ’1’.

3.4 Cache Coherency Support

The GT-6426x supports up to four SDRAM address windows, not correlated to specific chip selects, in which cache coherency is maintained. A pair of base/top registers (Snoop Control Register offsets 0x380 to 0x3B8) defines each window. The IDMA unit uses these regions to maintain cache coherency between SDRAM and Cache memories (see “Cache Coherency Support” on page 34).

Note

The cache coherency mechanism in the GT-6426x is restricted for the communication interface.

Note

The PCI cache mechanism uses different address windows (See “cache coherency support” subsection in the PCI interface section).

The GT-6436x does not implement these address windows in the CPU interface. Each unit (IDMA, Gigabit Ethernet, MPSCs, and PCI) has its own address windows that can be configured to cache coherency window.

Note

All the communication interfaces (Gigabit Ethernet and MPSCs) in the GT-6436x support cache coherency.

3.5 CPU Interface Register Settings

Use the following table to determine any differences in the GT6426x and MV6436x devices CPU registers.

Table 1:GT6426x Vs. MV6436x CPU Registers

GT6426x	MV6436x	Comments
CPU Configuration 0x000	CPU Configuration 0x000	Some changes.
CPU Mode 0x120	CPU Mode 0x120	Some changes.
CPU Master Control 0x160	CPU Master Control 0x160	Some changes.
CPU Interface Cross Bar Control (Low) 0x150	CPU Interface Cross Bar Control (Low) 0x150	Some changes.
CPU Interface Crossbar Control (High) 0x158	CPU Interface Crossbar Control (High) 0x158	Some changes.
CPU Interface Crossbar Timeout 0x168	CPU Interface Crossbar Timeout 0x168	Some changes.
CPU Read Response Crossbar Control (Low) 0x170	N/A	Not supported in MV6436x.
CPU Read Response Crossbar Control (High) 0x178	N/A	Not supported in MV6436x.
N/A	Who Am I 0x200	New register in GT-6436x.
N/A	CPU0 Doorbell 0x214	New register in GT-6436x.
N/A	CPU0 Doorbell Clear 0x21c	New register in GT-6436x.
N/A	CPU1 Doorbell 0x224	New register in GT-6436x.
N/A	CPU1 Doorbell Clear 0x22c	New register in GT-6436x.
N/A	CPU0 Doorbell Mask 0x234	New register in GT-6436x.
N/A	CPU1 Doorbell Mask 0x23c	New register in GT-6436x.
N/A	Semaphor0 0x244	New register in GT-6436x.
N/A	Semaphor1 0x24c	New register in GT-6436x.
N/A	Semaphor2 0x254	New register in GT-6436x.
N/A	Semaphor3 0x25c	New register in GT-6436x.
N/A	Semaphor4 0x264	New register in GT-6436x.
N/A	Semaphor5 0x26c	New register in GT-6436x.
N/A	Semaphor6 0x274	New register in GT-6436x.
N/A	Semaphor7 0x27c	New register in GT-6436x.
PCI_0 Sync Barrier Virtual Register 0x0c0	N/A	Not supported in MV6436x.

Table 1:GT6426x Vs. MV6436x CPU Registers (Continued)

GT6426x	MV6436x	Comments
PCI_1 Sync Barrier Virtual Register 0x0c8	N/A	Not supported in MV6436x.
N/A	CPU0 Sync Barrier Trigger Register0 0x0c0	New register in MV6436x.
N/A	CPU0 Sync Barrier Virtual Register0 0x0c8	New register in MV6436x.
N/A	CPU1 Sync Barrier Trigger Register10x0d0	New register in MV6436x.
N/A	CPU1 Sync Barrier Virtual Register10x0d8	New register in MV6436x.
Snoop Base Address 0 0x380	N/A	Not supported in MV6436x.
Snoop Top Address 0 0x388	N/A	Not supported in MV6436x.
Snoop Base Address 1 0x390	N/A	Not supported in MV6436x.
Snoop Top Address 1 0x398	N/A	Not supported in MV6436x.
Snoop Base Address 2 0x3a0	N/A	Not supported in MV6436x.
Snoop Top Address 2 0x3a8	N/A	Not supported in MV6436x.
Snoop Base Address 3 0x3b0	N/A	Not supported in MV6436x.
Snoop Top Address 3 0x3b8	N/A	Not supported in MV6436x.
Protect Low Address 0 0x180	Protect base Address 0 0x180	Bit 31 functions as Window enable.
Protect High Address 0 0x188	Protect size 0 0x188	No changes.
Protect Low Address 10x190	Protect base Address 10x190	Bit 31 functions as Window enable.
Protect High Address 1 0x198	Protect size 10x198	No changes.
Protect Low Address 2 0x1A0	Protect base Address 2 0x1A0	Bit 31 functions as Window enable.
Protect High Address 2 0x1A8	Protect size 2 0x1A8	No changes.
Protect Low Address 3 0x1B0	Protect base Address 3 0x1B0	Bit 31 functions as Window enable.
Protect High Address 3 0x1B8	Protect size 3 0x1B8	No changes.
Protect Low Address 4 0x1D0	N/A	Not supported in MV6436x.
Protect High Address 4 0x1D8	N/A	Not supported in MV6436x.
Protect Low Address 5 0x1D0	N/A	Not supported in MV6436x.
Protect High Address 5 0x1D8	N/A	Not supported in MV6436x.

Table 1:GT6426x Vs. MV6436x CPU Registers (Continued)

GT6426x	MV6436x	Comments
Protect Low Address 6 0x1E0	N/A	Not supported in MV6436x.
Protect High Address 6 0x1E8	N/A	Not supported in MV6436x.
Protect Low Address 7 0x1F0	N/A	Not supported in MV6436x.
Protect High Address 7 0x1F8	N/A	Not supported in MV6436x.

Section 4. DDR SDRAM Interface

The GT6426x SDRAM controller has a 15-bit address bus (DAdr[12:0] and BankSel[1:0]) and a 64-bit data bus (SData[63:0]). The SDRAM controller supports 16, 64, 128, 256 or 512Mb SDRAMs.Up to 1 GB can be addressed by each SCS for a total SDRAM address space of 4 GB.

The MV6436x DRAM controller has a 16-bit address bus (DAdr[13:0] and BankSel[1:0]) and a 64-bit data bus (SData[63:0]). The SDRAM controller supports 16, 64, 128, 256, 512Mb and 1Gb SDRAMs. Up to 2 GB can be addressed by each SCS for a total SDRAM address space of 8 GB.

4.1 Operation Mode Register Setting

4.1.1 GT6426x SDRAM Operation Mode Register

In the GT6426x the SDRAM Operation Mode register is used to execute commands other than standard memory reads and writes to the SDRAM. These operations include:

- Normal SDRAM Mode
- NOP Commands
- Pre-charge All Banks
- Load DRAM Mode Register
- Force a Refresh Cycle

To execute one of the above commands on the SDRAM, the following procedure must occur:

1. Write to the SDRAM Operation Mode register the required command.
2. Read the SDRAM Operation Mode register. This read guarantees that the following step is executed after the register value is updated.
3. Dummy word (32-bit) writes to an SDRAM bank. This causes that the required cycle is driven to the selected DRAM bank.
4. Polling on SDRAM Operation Mode register until the activate bit is sampled ‘1’.
5. Write a 0x0 value to the SDRAM Operation Mode Register.
6. Read the SDRAM Operation Mode register. This read guarantees the execution of the following access to the DRAM, after the register value is updated.

Note

No access to the SDRAM other then dummy write in the sequence above, is allowed in the middle of the above sequence execution. This means that the above sequence must not be executed from SDRAM since no instruction fetch is allowed.

4.1.2 MV6436x SDRAM Operation Mode Register

In the MV6436x, the SDRAM Operation Mode register is used to execute commands other than standard memory reads and writes to the SDRAM. These operations include:

- Normal SDRAM Mode (default mode)
- NOP Commands
- Pre-charge All Banks
- Load DRAM Mode Register

- Load DRAM Extended Mode Register
- Force a Refresh Cycle

Once the CPU changes the register default to one of the command types, The SDRAM controller:

- Executes the required command.
- Resets the register back to the default value.
- Returns to normal operation.

The CPU must poll on this register to identify when the DRAM controller is back in normal operation mode.

Note

No access to the SDRAM is allowed in the middle of the above sequence execution. This means that the above sequence must not be executed from SDRAM since no instruction fetch is allowed

4.2 SDRAM Timing Parameters

The following table describes the differences between the GT6426x and MV6436x timing parameters:

Table 2:GT6426x Vs. MV6436x SDRAM Timing Parameters

SDRAM Timing Parameter	Description	GT6426x	MV6436x
SCAS* Latency (CL)	The number of cycles from CAS* assertion to the sampling of the first read data.	This parameter can be programmed for two or three TClks cycles. SDRAM Timing Parameter's register CL bits [1:0] (Offset:0x4b4).	The SDRAM controller supports a CL setting of 1.5, 2, 2.5, 3, 3.5 or 4 cycles. SDRAM Mode register's CL bits [6:4] (Offset 0x141c).
SRAS to SCAS Delay (Trcd)	specifies the number of TClk cycles that the DRAM controller inserts between the assertion of SRAS* with a valid row address to the assertion of SCAS* with a valid column address	This parameter can be programmed for two or three TClks cycles. SDRAM Timing Parameter's register Trcd bits [5:4] (Offset:0x4b4).	The SDRAM controller supports Trcd of 2, 3 or 4 cycles. SDRAM Timing Parameter's register Trcd bits [7:4] (Offset:0x1408).
SRAS* Pre-charge (Trp)	Following a possible pre-charge cycle, the minimum number of cycles within which a new activate cycle can occur.	This parameter can be programmed for two or three TClks cycles. SDRAM Timing Parameters register's Trp bits [3:2] (Offset: 0x4b4).	The SDRAM controller supports a Trp setting of 2, 3 or 4 cycles. SDRAM Timing (Low) register's Trp bits [11:8] (Offset: 0x1408).
Row Active Time (Tras)	The minimum number of cycles between activate cycle to pre-charge cycle.	This parameter can be programmed for two or three TClks cycles. SDRAM Timing Parameters register Tras bits [11:8] (Offset: 0x4b4).	The SDRAM controller supports a Tras setting of 5, 6, 7, 8, or 9 cycles. SDRAM Timing (Low) register's Tras bits [23:20] (Offset: 0x1408).

Table 2:GT6426x Vs. MV6436x SDRAM Timing Parameters (Continued)

SDRAM Timing Parameter	Description	GT6426x	MV6436x
Write to DQS (Tdqss) **NOTE:** MV6436x only.	The minimum number of cycles between write commands.	N/A	Value ‘0’ means one cycle; value of ‘1’ means two cycles; and so on. SDRAM Timing (Low) register’s Tdqss bits [3:0] (Offset: 0x1408). **NOTE:** Must be ’0’.
Write to Pre-charge (Twr) **NOTE:** MV6436x only.	The minimum number of cycles between write command and pre-charge.	N/A	Value ‘0’ means one cycle; value of ‘1’ means two cycles; and so on. SDRAM Timing (Low) register’s Twr bits [15:12] (Offset: 0x1408). **NOTE:** Only the values of ’1’, ’2’, or ’3’ are allowed.
Write to Read (Twtr) **NOTE:** MV6436x only.	The minimum numbers of cycles between write command and read command.	N/A	Value ‘0’ means one cycle; value of ‘1’ means two cycles; and so on. SDRAM Timing (Low) register’s Twtr bits [19:16] (Offset: 0x1408). **NOTE:** Only the values of ’1’, ’2’, or ’3’ are allowed.
Active to Active (Trrd) **NOTE:** MV6436x only.	The minimum numbers of cycles between activate bank A to activate bank B.	N/A	Value ‘0’ means one cycle; value of ‘1’ means two cycles; and so on.SDRAM Timing (Low) register’s Trrd bits [27:24] (Offset: 0x1408). **NOTE:** Only the values of ’1’, ’2’, or ’3’ are allowed.
Load Mode Register () **NOTE:** MV6436x only.	The minimum number of cycles between LMR command and the next LMR command or pre-charge.	N/A	SDRAM Timing (Low) register’s Tmrd bits [31:28] (Offset: 0x1408).
Refresh Command (Trfc) **NOTE:** MV6436x only.	The minimum numbers of cycles between refresh command and the new activate command.	N/A	Value ‘0’ means one cycle; value of ‘1’ means two cycles; and so on. SDRAM Timing (High) register’s Trfc bits [3:0] (Offset: 0x140c).

Table 2:GT6426x Vs. MV6436x SDRAM Timing Parameters (Continued)

SDRAM Timing Parameter	Description	GT6426x	MV6436x
Read to Read (Trd2rd) **NOTE:** MV6436x only.	The minimum number of cycles between consecutive read commands.	N/A	Value ‘0’ means one cycle; value of ‘1’ means two cycles; and so on. SDRAM Timing (High) register’s Trd2rd bits [5:4] (Offset: 0x140c). **NOTE:** Only values of ’0’ or ’1’ are allowed.
Read to Write (Trdtwr) **NOTE:** MV6436x only.	The minimum number of cycles between read command to write command.	N/A	Value ‘0’ means one cycle; value of ‘1’ means two cycles; and so on. SDRAM Timing (High) register’s Trdtwrn bits [7:6] (Offset: 0x140c). **NOTE:** Only values of ’0’ or ’1’ are allowed.

Note

In the GT6426x the SDRAM timing parameters CAS latency, RAS to CAS, and RAS Precharge must be set to the same value (2 or 3).

4.3 DDR SDRAM Interface Register Settings

Use the following table to determine any differences between the GT6426x and MV6436x devices SDRAM registers.

Table 3:GT6426x Vs. MV6436x SDRAM Registers

GT6426x	MV6436x	Comments
SDRAM Configuration 0x448	SDRAM Configuration 0x1400	Some changes in MV6436x.
SDRAM Operation Mode 0x474	SDRAM Operation 0x1418	Some changes in MV6436x.
SDRAM Address Control 0x47c	SDRAM Address Control 0x1410	Some changes in MV6436x.
SDRAM Timing Parameters 0x4b4	SDRAM Timing Control (LOW and HIGH) 0x1408 and 0x140C	See “SDRAM Timing Parameters” on page 19
SDRAM UMA Control 0x4a4	N/A	Not supported.
SDRAM Interface Crossbar Control (Low) 0x4a8	SDRAM Interface Cross Bar Control (Low) 0x1430	No changes.
SDRAM Interface Crossbar Control (High) 0x4ac	SDRAM Interface Cross Bar Control (High) 0x1434	No changes.
SDRAM Interface Crossbar Timeout 0x4b0	SDRAM Interface Cross Bar Timeout 0x1438	No changes.

Table 3:GT6426x Vs. MV6436x SDRAM Registers (Continued)

GT6426x	MV6436x	Comments
SDRAM Bank0 Parameters 0x44c	N/A	Not supported in MV6436x.
SDRAM Bank1 Parameters 0x450	N/A	Not supported in MV6436x.
SDRAM Bank2 Parameters 0x454	N/A	Not supported in MV6436x.
SDRAM Bank3 Parameters 0x458	N/A	Not supported in MV6436x.
N/A	SDRAM Open Pages Control 0x1414	New register in MV6436x.
N/A	SDRAM Mode 0x141c	New register in MV6436x.
N/A	SDRAM Extended Mode 0x1420	New register in MV6436x.
N/A	Dunit Control (Low) 0x1404	New register in MV6436x.
N/A	Dunit Control (High) 0x1424	New register in MV6436x.
N/A	SDRAM Timing Control (Low) 0x1408	New register in MV6436x.
N/A	SDRAM Timing Control (High) 0x140c	New register in MV6436x.
SDRAM Error Data (Low) 0x484	SDRAM Error Data (Low) 0x1444	No changes.
SDRAM Error Data (High) 0x480	SDRAM Error Data (High) 0x1440	No changes.
SDRAM Error Address 0x490	SDRAM and Device Error Address 0x1450	No changes.
SDRAM Received ECC 0x488	SDRAM Received ECC 0x1448	No changes.
SDRAM Calculated ECC 0x48c	SDRAM Calculated ECC 0x144c	No changes.
SDRAM ECC Control 0x494 SDRAM	ECC Control 0x1454	No changes.
SDRAM ECC Error Counter 0x498	SDRAM ECC Error Counter 0x1458	No changes.
N/A	MMASK 0x1B40	New register in MV6436x. **NOTE:** Do not use this register. It is reserved for future use.

Section 5. Device Interface

5.1 Parity Support for Devices

There is no dedicated logic in the GT6426x devices to support parity on the device bus. If device parity checking is required, connect an external logic to the GPP inputs to generate an interrupt when a bad parity occurs.

The MV6436x device controller supports generating and checking of data parity on the device interface via the DevDP[3:0] pins. Enable/disable parity on a device chip select basis via the Device Bank Parameters registers' `DPEn` bit [30] (offsets 0x45C to 0x46C). Even or Odd parity is selectable via the Device Interface Control register's `ParSel` bit [20].

5.2 Device Interface Register Settings

Use the following table to determine any differences between the GT6426x and MV6436x devices device controller registers.

Table 4:GT6426x Vs. MV6436x SDRAM Registers

GT6426x	MV6436x	Comments
Device Bank0 Parameters 0x45c		Add data parity support in MV6436x.
Device Bank1 Parameters 0x460		Add data parity support in MV6436x.
Device Bank2 Parameters 0x464		Add data parity support in MV6436x.
Device Bank3 Parameters 0x468		Add data parity support in MV6436x.
Boot Device Parameters 0x46c		Add data parity support in MV6436x.
Device Interface Control 0x4c0		Add data parity support in MV6436x.
Device Interface Crossbar Control (Low) 0x4c8		Some changes in MV6436x.
Device Interface Crossbar Control (High) 0x4cc		Some changes in MV6436x.
Device Interface Crossbar Timeout 0x4c4		No changes.
Device Interrupt Cause 0x4d0		Some changes in MV6436x.
Device Interrupt Mask 0x4d4		Some changes in MV6436x.
Device Error Address 0x4d8		No changes.
N/A	Device Error Data 0x4dc	New register in MV6436x.

Table 4:GT6426x Vs. MV6436x SDRAM Registers (Continued)

GT6426x	MV6436x	Comments
N/A	Device Error Parity 0x4e0	New register in MV6436x.
N/A	MMASK 0x4F0	New register in MV6436x. **NOTE:** Do not use this register. It is reserved for future use.

Section 6. PCI Interface

The GT6426x implements PCI 2.2 compliant interfaces.

The MV6436x implements PCI-X compliant interfaces. This section will not discuss the PCI-X mode. For more information on the PCI-X mode see the MV6436x data sheet.

6.1 PCI Configuration Space Header

The GT6426x implements separate SAC BAR and DAC BAR for the same memory space.

Alternatively, the MV6436x BARs are 64-bit wide and support 32 or 64-bit addressing (SAC or DAC), depending on the setting of the upper 32-bit of the BAR.

Note

For more information see the respective “PCI Configuration Space Header’ figures in the GT6426x or MV6436x datasheet.

Figure 4: GT6426xPCI Configuration Space Header

Function 0 Header

Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Rev ID	08h
BIST	Header	Latency	Line Size	0Ch
SCS[0] BAR				10h
SCS[1] BAR				14h
SCS[2] BAR				18h
SCS[3] BAR				1Ch
Mem Mapped Internal BAR				20h
IO Mapped Internal BAR				24h
Reserved				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM BAR				30h
Reserved			Cap. Ptr	34h
Reserved				38h
Max_Lat	Min_Gnt	Int. Pin	Int. Line	3Ch

Function 1 Header

				00h
				04h
				08h
				0Ch
CS[0] BAR				10h
CS[1] BAR				14h
CS[2] BAR				18h
CS[3] BAR				1Ch
BootCS BAR				20h
Reserved				24h
Reserved				28h
				2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
				3Ch

Function 2 Header

				00h
				04h
				08h
				0Ch
P2P Mem0 BAR				10h
P2P Mem1 BAR				14h
P2P IO BAR				18h
CPU BAR				1Ch
Reserved				20h
Reserved				24h
Reserved				28h
				2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
				3Ch

Function 3 Header

				00h
				04h
				08h
				0Ch
Reserved				10h
Reserved				14h
Reserved				18h
Reserved				1Ch
Reserved				20h
Reserved				24h
Reserved				28h
				2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
				3Ch

Function 4 Header

				00h
				04h
				08h
				0Ch
SCS[0] 64-bit BAR				10h
				14h
SCS[1] 64-bit BAR				18h
				1Ch
P2P Mem0 64-bit BAR				20h
				24h
Reserved				28h
				2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
				3Ch

Function 5 Header

				00h
				04h
				08h
				0Ch
SCS[2] 64-bit BAR				10h
				14h
SCS[3] 64-bit BAR				18h
				1Ch
P2P Mem1 64-bit BAR				20h
				24h
Reserved				28h
				2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
				3Ch

Function 6 Header

				00h
				04h
				08h
				0Ch
CS[0] 64-bit BAR				10h
				14h
CS[1] 64-bit BAR				18h
				1Ch
CS[2] 64-bit BAR				20h
				24h
Reserved				28h
				2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
				3Ch

Function 7 Header

				00h
				04h
				08h
				0Ch
CS[3] 64-bit BAR				10h
				14h
BootCS 64-bit BAR				18h
				1Ch
CPU 64-bit BAR				20h
				24h
Reserved				28h
				2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
				3Ch

Figure 5: MV6436xPCI Configuration Space Header

Function 0 Header

Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Rev ID	08h
BIST	Header	Latency	Line Size	0Ch
SCS[0] BAR				10h
				14h
SCS[1] BAR				18h
				1Ch
Mem Mapped Internal BAR				20h
				24h
Reserved				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM BAR				30h
Reserved			Cap. Ptr	34h
Reserved				38h
Max_Lat	Min_Gnt	Int. Pin	Int. Line	3Ch

Function 1 Header

▨		▨		00h
▨		▨		04h
▨			▨	08h
▨	▨	▨	▨	0Ch
SCS[2] BAR				10h
				14h
SCS[3] BAR				18h
				1Ch
Int SRAM BAR				20h
				24h
Reserved				28h
▨		▨		2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
▨	▨	▨	▨	3Ch

Function 2 Header

▨		▨		00h
▨		▨		04h
▨			▨	08h
▨	▨	▨	▨	0Ch
CS[0] BAR				10h
				14h
CS[1] BAR				18h
				1Ch
CS[2] BAR				20h
				24h
Reserved				28h
▨		▨		2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
▨	▨	▨	▨	3Ch

Function 3 Header

▨		▨		00h
▨		▨		04h
▨			▨	08h
▨	▨	▨	▨	0Ch
CS[3] BAR				10h
				14h
BootCS BAR				18h
				1Ch
CPU BAR				20h
				24h
Reserved				28h
▨		▨		2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
▨	▨	▨	▨	3Ch

Function 4 Header

▨		▨		00h
▨		▨		04h
▨			▨	08h
▨	▨	▨	▨	0Ch
P2P Mem0 BAR				10h
				14h
P2P Mem1 BAR				18h
				1Ch
P2P I/O BAR				20h
I/O Mapped Internal BAR				24h
Reserved				28h
▨		▨		2Ch
Reserved				30h
Reserved			Reserved	34h
Reserved				38h
▨	▨	▨	▨	3Ch

6.2 PCI Synchronization Barrier

Within the GT6426x devices, the PCI slave “synchronization barrier” cycles are Configuration Reads. If there is no posted data within the CPU interface write buffer and PCI master write buffer, the cycle ends normally. If after a timeout0 period there is still posted data in the buffers, the cycle is terminated with the Retry signal. Until the original cycle ends, any new “synchronization barrier” cycles are terminated with Retry. The PCI slave only handles a single pending sync barrier transaction at a time.

The MV6436xPCI slave implements two registers:

- Sync Barrier Trigger register (0x1d18 or 0x1d98)
- Sync Barrier Virtual register (0x1d10 or 0x1d90)

A write to the Sync Barrier Trigger register, triggers the sync barrier’s state machine. A subsequent read polling on the Sync Barrier Virtual register results in a value of 0xffff.ffff until the sync action completes. As soon as the PCI master write buffers are flushed, a read of the Virtual register results in value of 0x0.

The MV6436x also supports nesting of sync barriers. The PCI agent that triggers the sync barrier may decide to trigger a new sync barrier prior to the result of the previous one (new write to Sync Barrier Trigger register). The MV6436x sync barrier implementation guarantees that a subsequent read polling on the Sync Barrier Virtual register will result in a value of 0x0, only after the second sync barrier is resolved.

6.3 Cache Coherency Support

The GT-6426x and GT-6436x support PowerPC cache coherency. Any PCI access to SDRAM may generate a snoop transaction on the CPU bus, to maintain coherency between the CPU cache and SDRAM.

In the GT-6426x, it is possible to configure up to four address ranges in which cache coherency is maintained. PCI Snoop Base and Top Address registers define the address windows. Any PCI access that hits one of these address windows results in snoop transaction.

The GT-6436x does not implement separate cache coherency windows settings. Instead, the cache coherency windows are defined as part of the PCI Access Control windows (PCI Access Control Base register bits 3:2). See section “PCI Access Control Windows” on page 28.

6.4 PCI Access Control Windows

The GT6426x PCI slave interface supports configurable access control. It is possible to define up to eight address ranges to different configurations. Each region can be configured to:

- Read prefetch and aggressive prefetch
- PCI unit max burst to other units
- Delayed read
- Write protection
- Access protection
- Byte swapping

Three registers define each address window - Base (low and high) and Top.
The MV6436x PCI slave interface also supports configurable access control. Yet, It is only possible to define up to six address ranges to different configurations. Each region can be configured to:

- Write and access protection
- Force REQ64
- Byte swapping
- Cache coherency

- Typical PCI unit to other units burst size
- Read prefetch

Three registers define each address window - Base (low and high) and Size.

6.5 PCI Interface Register Settings

Since the PCI configuration space header in the MV6436x implements fewer BARs (see section “PCI Configuration Space Header”), there are fewer BAR size and remap registers in the MV6436x PCI interface.

Table 5:GT6426x Vs. MV6436x PCI Registers

GT6426x	MV6436x	Comments
N/A	Integrated SRAM BAR Size • PCI0: 0xE00 • PCI1: 0xE80	New registers in MV6436x.
N/A	Integrated SRAM Base Address Remap • PCI0: 0xE00 • PCI1: 0xE80	New registers in MV6436x.
N/A	PCI Headers Retarget Control • PCI0: 0xF40 • PCI1: 0xFC0	New registers in MV6436x.
N/A	PCI Headers Retarget Base • PCI0: 0xF44 • PCI1: 0xFC4	New registers in MV6436x.
N/A	PCI Headers Retarget (High) • PCI0: 0xF48 • PCI1: 0xFC8	New registers in MV6436x.
N/A	Status and Command • DLL PCI0: 0x1d20 • PCI1: 0x1da0	New registers in MV6436x.
N/A	PCI/MPP Pads Drive Control • PCI_0/MPP[31:16]: 0x1d1c • PCI_1/MPP[15:0]: 0x1d9c	New registers in MV6436x.
PCI Command • PCI_0: 0xc00 • PCI_1: 0xc80		Some changes in MV6436x.
PCI Mode • PCI_0: 0xd00 • PCI_1: 0xd80		Some changes in MV6436x.
PCI Timeout and Retry • PCI_0: 0xc04 • PCI_1: 0xc84		Timeout0 and Timeout1 bits are not implemented in the MV6436x.

Table 5:GT6426x Vs. MV6436x PCI Registers (Continued)

GT6426x	MV6436x	Comments
PCI Read Buffer Discard Timer • PCI_0: 0xd04 • PCI_1: 0xd84		`RdBuffEn` bits are not implemented in MV6436x.
PCI Arbiter Control • PCI_0: 0x1d00 • PCI_1: 0x1d80		No changes.
PCI Interface Crossbar Control (Low) • PCI_0 Offset: 0x1d08 • PCI_1 Offset: 0x1d88		Some changes in MV6436x.
PCI Interface Cross Bar Control (High) • PCI_0: 0x1d0c • PCI_1: 0x1d8c		Some changes in MV6436x.
PCI Interface Cross Bar Timeout • PCI_0: 0x1d04 • PCI_1: 0x1d84		No changes.
N/A	PCI Sync Barrier Trigger Register • PCI_0: 0x1d18 • PCI_1: 0x1d98	New registers in MV6436x.
N/A	PCI Sync Barrier Virtual Register • PCI_0: 0x1d10 • PCI_1: 0x1d90	New registers in MV6436x.
PCI P2P Configuration • PCI_0: 0x1d14 • PCI_1: 0x1d94		Some changes for PCI-X mode in MV6436x.
PCI Access Control Base 0-7 (Low)	PCI Access Control Base 0-5 (Low)	Only six windows with some change in the bits settings in MV6436x.
PCI P2P Swap Control • PCI_0: 0x1d54 • PCI_1: 0x1dd4	N/A	Not supported in MV6436x.
PCI Snoop Control Base 0-3 (Low)	N/A	Not supported in MV6436x.
PCI Snoop Control Base 0-3 (High)	N/A	Not supported in MV6436x.
PCI Snoop Control Top 0-3	N/A	Not supported in MV6436x.
PCI SERR* Mask • PCI_0: 0xc28 • PCI_1: 0xca8		Not supported in MV6436x.

Table 5:GT6426x Vs. MV6436x PCI Registers (Continued)

GT6426x	MV6436x	Comments
PCI Interrupt Cause • PCI_0: 0x1d58 • PCI_1: 0x1dd8		Some changes in MV6436x
PCI Interrupt Mask • PCI_0 Offset: 0x1d5c • PCI_1 Offset: 0x1ddc		Some changes in MV6436x
N/A	PCI Error Attribute • PCI_0 Offset: 0x1d48 • PCI_1 Offset: 0x1dc8	New registers in MV6436x.
N/A	PCI MMask • PCI_0 Offset: 0x1d24 • PCI_1 Offset: 0x1da4	New registers in MV6436x. **NOTE:** The software must not use this register. It is reserved for future use, only.

Section 7. Communication Interfaces

The MV6436x implements a complete new Gigabit Ethernet unit that replaced the previous 10/100Mb Ethernet unit in the GT6426x. The Gigabit Ethernet unit main features are:

- Three ports support:
  - GMII (Full Duplex only)
  - MII (Full Duplex and Half Duplex)
  - PCS (Full Duplex only)
- 802.3x flow control
- Back pressure for MII Half Duplex
- Weighted Round Robin and Maximal BW distribution (byte based) for transmit
- Transmit short frames zero padding
- Long frames support (up to 9K)
- IP and TCP/UPD checksum generation

Note

This is a partial list, for full feature list see the data sheet.

7.1 Address Decoding

In the GT6426x devices, the communication unit uses the address mapping of the CPU interface. In the MV6436x, the three Gb Ethernet (GbE) ports share a single address decoding logic consisting of six address windows. The two MPSCs share a single address decoding logic consisting of four address windows. Whenever one of the ports generates a read or a write transaction (e.g. fetch descriptor), the address is compared against these address windows, to determine which interface must be accessed.

The address decoding scheme in the GbE ports is the same as the IDMA logic, with one exception. If the address does not match any of the address windows, or if it violates the access protection settings, an interrupt is generated. The transaction is executed but not to the original address. Instead, the transaction is executed to a default address and the target as specified in the Default Address and ID registers (Offset: 0x2008 and 0x200C). This prevents the ports from becoming stuck and helps the SW to deal with the problem since the data is placed in a known place.

If the address does not match any of the address windows in the two MPSCs address decode mechanisms, an interrupt is generated and the port halts.

The PCI interface supports 64-bit addressing. Four of the six address windows in the GbE ports and two of the four address windows in the MPSCs ports have an upper 32-bit address register. The address generated on the PCI bus is composed of the window base address and the High Remap register.

Note

The address decoding scheme in the communication interfaces (MPSCs and GbE) is the same as the IDMA logic.

7.2 Gigabit Ethernet Unit Priority Queuing

In the GT6426x there are four receive priority queues and two transmit priority queues per 10/100Mb Ethernet port. Priority information for received packets is either extracted from the packet tag (if the packet is VLAN tagged) or from the destination-address entry in the address table (if the packet is not tagged). The transmitted packets can be transmitted from two separate queues, High priority and Low priority. The arbitration between the two queues is defined by the ratio programmed in Port Configuration Extend register's (offsets 0x2408, 0x2808, 0x2c08) `PRIOtx` bits [5:3].

The MV6436x implements a Weighted Round Robin priority queuing mechanism for up to eight transmit queues. In addition, it supports up to eight receive queues.

In the MV6436x, each transmit queue can be configured in two modes:

- Fixed priority mode
- Weighted Round Robin (WRR) priority mode

For more information on the MV6436x transmit arbitration, see the section "Transmit Weighted Round Robin Arbitration" in the data sheet.

For more information on the MV6436x receive arbitration, see the section "Parsing the Frames" in the data sheet.

7.3 Interrupt Coalescing

The interrupt coalescing capability is only available to the MV6436x devices.

Since the GbE line rate provides a high packet rate, it is important to reduce the amount of interrupts that may be generated. The most intensive interrupts are the packet level interrupts on receive and transmit transactions.

The interrupt coalescing capability prevents the loss of interrupt indications during the period from when the CPU reads the interrupt register to the time that it starts switching off interrupt bits. During this interval, new interrupts that arrive for the same receive or transmit queue would have been lost and buffers may become indefinitely stuck.

7.4 Rx Address Recognition

In the MV6436x devices, there is per port dedicated MAC-DA address filtering of up to 16 unicast MAC addresses, 256 IP Multicast addresses, and 256 Multicast/broadcast addresses. There is no longer any hash table lookup in DRAM.

7.5 TCP/UDP Checksum Generation and Checking

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

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

7.6 Support for Jumbo Packets (9 KB)

The support for jumbo frames corresponds to the Alteon jumbo-spec rev 2.1.

Section 8. IDMA Unit

The IDMA engines in the MV6436x devices are the same as those in the GT6426x devices. The only difference is in the number of IDMA engines.

- GT6426x incorporates eight DMA engines.
- MV6436x incorporates four IDMA engines.

8.1 Address Decoding

Unlike the GT6426x devices, in which the IDMA shared the CPU address decoding windows, the MV6436x IDMAs have their own dedicated address decoding windows. It is possible to have the same IDMA and CPU address map by programming the IDMA address decoding registers to the same values of the CPU address decoders. Each window can be configured to a different target interface. Address comparison is done to select the correct target interface (DRAM, PCI etc.). If the address does not match any of the address windows, an interrupt is generated and the IDMA engine is stopped.

Note

Since the IDMAs address decoding is not coupled with CPU address decoding, the PCI override feature that was implemented in the GT6426x devices is no longer supported in MV6436x.

8.2 Cache Coherency Support

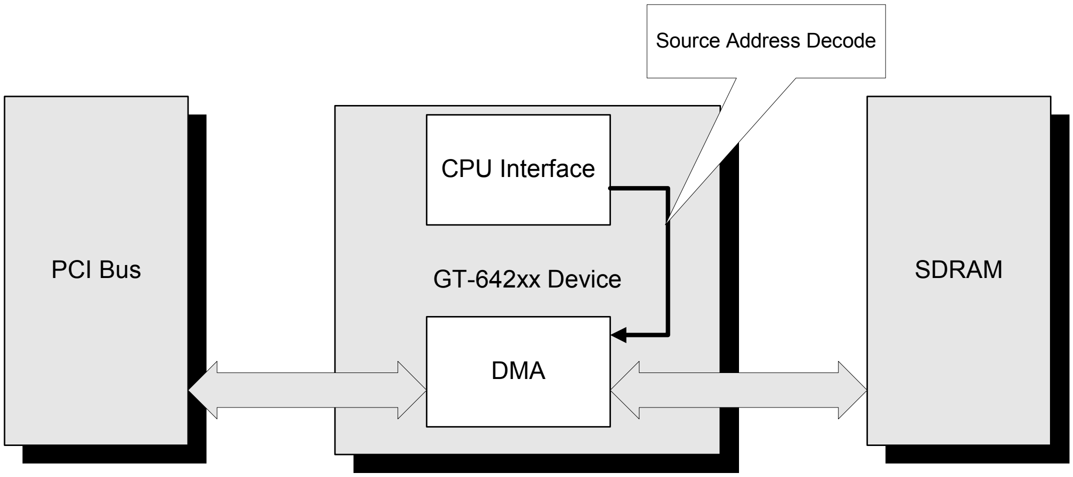
In the GT-6426x devices, each IDMA engine transaction address is compared against the four cache coherency regions. If an address hits one of these regions, the DRAM access results a snoop action. Each channel has a programmable bit per source, destination, and next descriptor pointer to enable/disable snoops.

In the GT-6436x, the IDMAs have their own dedicated address decoding windows that can be configured to cache coherency.

8.3 Address Override

In the GT6426x devices, the source, destination, and next descriptor addresses for each channel can be marked as PCI override. This means the IDMA engine accesses the PCI interface directly without executing any address decoding. The GT6426x only supports override to PCI0 and PCI1. For example, when the IDMA engine destination address override to PCI0 is enabled, data can be transferred from address 0x0 in SDRAM to address 0x0 in PCI0. In this configuration, the source address uses the CPU address decode mechanism to map address 0x0 and the destination is transferred directly to PCI0.

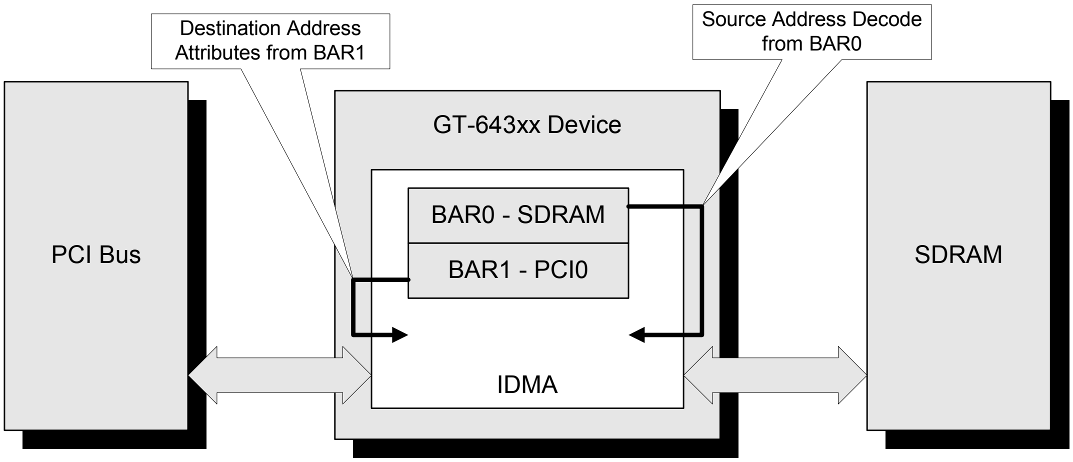
Figure 6: GT6426x Devices Address Override



In the MV6436x devices, the IDMA also supports an address override feature. Each of the source, destination, or next descriptor addresses can be configured to use the override feature via the Channel Control register's `SAddrOvr`, `DAddrOvr` and `NAddrOvr` fields, respectively. When the field is set to 0x1, the transaction target interface, attributes, and upper 32-bit address are taken from Base Address register 1. When set to 0x2, they are taken from BAR 2 and when set to 0x3, they are taken from BAR 3.

This feature enables additional address decoupling. For example, to transfer data from address 0x0 in the SDRAM to address 0x0 in PCI0, BAR0 is configured to address 0x0 of SDRAM and BAR1 is configured to other addresses in PCI0 (i.e., 0x12000000). Program the source and destination addresses to 0x0. When the `DaddrOvr` is set to 0x1, the destination attributes are taken from BAR1 as shown in the following figure.

Figure 7: MV6436x Devices Address Override



8.4 PCI 64-bit Address

In the GT6426x devices, each IDMA channel generates a 64-bit address to the PCI interface via source, destination, and next descriptor PCI High Address register. If the PCI High Address register value is ‘0’, the PCI master issues a SAC transaction. If the register value is other than ’0’ (meaning the address is beyond 4Gbyte space), the PCI master generates a DAC transaction.

In the MV6436x devices, four of the eight address windows have an upper 32-bit address register (see section “PCI Configuration Space Header” on page 25"). To access the PCI bus with 64-bit addressing cycles (DAC cycles), assign one (or more) of these four windows to target the PCI bus. The address generated on the PCI bus is composed of the window base address and the High Remap register.

8.5 IDMA Unit Register Settings

Table 6:GT6426x Vs. MV6436x PCI Registers

GT6426x	MV6436x	Comments
Channel 3-0 DMA Byte Count 0x800, 0x804, 0x808, 0x80c		No changes.
Channel 7-4 DMA Byte Count 0x900, 0x904, 0x908, 0x90c	N/A	Not supported in MV6436x.
Channel 3-0 DMA Source Address 0x810, 0x814, 0x818, 0x81c		No changes.
Channel 7-4 DMA Source Address 0x910, 0x914, 0x918, 0x91c	N/A	Not supported in MV6436x.
Channel 3-0 DMA Destination Address 0x820, 0x824, 0x828, 0x82c		No changes.
Channel 7-4 DMA Destination Address 0x920, 0x924, 0x928, 0x92c	N/A	Not supported in MV6436x.
Channel 3-0 Next Descriptor Pointer 0x830, 0x834, 0x838, 0x83c		No changes.
Channel 7-4 Next Descriptor Pointer 0x930, 0x934, 0x938, 0x93c	N/A	Not supported in MV6436x.
Channel 3-0 Current Descriptor Pointer 0x870, 0x874, 0x878, 0x87c		No changes.
Channel 7-4 Current Descriptor Pointer 0x970, 0x974, 0x978, 0x97c	N/A	Not supported in MV6436x.
N/A	Base Address Register 7-0 0xa00, 0xa08, 0xa10, 0xa18, 0xa20, 0xa28, 0xa30, 0xa38	New registers in MV6436x.
N/A	Size Register 7-0 0xa04, 0xa04, 0xa14, 0xa1c, 0xa24, 0xa2c, 0xa34, 0xa3c	New registers in MV6436x.
N/A	High Address Remap 0xa60, 0xa64, 0xa68, 0xa6c	New registers in MV6436x.

Table 6:GT6426x Vs. MV6436x PCI Registers (Continued)

GT6426x	MV6436x	Comments
N/A	Base Address Enable Register 0xa80	New registers in MV6436x.
N/A	Access Protection Register 0xa70, 0xa74, 0xa78, 0xa7c	New registers in MV6436x.
N/A	IDMA Headers Retarget Control 0xa84	New registers in MV6436x.
N/A	IDMA Headers Retarget Base 0xa88	New registers in MV6436x.
Channel 3-0 Control (Low) 0x840, 0x844, 0x848, 0x84c		Bits 8:6 (`SrcBurstLimit`) always acts as source DTL.
Channel 7-4 Control (Low) 0x940, 0x944, 0x948, 0x94c	N/A	Not supported in MV6436x.
Channel 3-0 Control (High) 0x880, 0x884, 0x888, 0x88c		Some changes in MV6436x. New bit settings.
Channel 7-4 Control (High) 0x980, 0x984, 0x988, 0x98c	N/A	Not supported in MV6436x.
Channels 0-3 Interrupt Cause 0x8c0	Interrupt Cause 0x8c0	No changes.
Channels 0-3 Interrupt Mask 0x8c4	Interrupt Mask 0x8c4	No changes.
Channels 0-3 Error Address 0x8c8	Error Address 0x8c8	No changes.
Channels 0-3 Error Select 0x8cc	Error Select 0x8cc	No changes.
Channels 7-4 Interrupt Cause 0x9c0	N/A	Not supported in MV6436x.
Channels 7-4 Interrupt Mask 0x9c4	N/A	Not supported in MV6436x.
Channels 7-4 Error Address 0x9c8	N/A	Not supported in MV6436x.
Channels 7-4 Error Select 0x9cc	N/A	Not supported in MV6436x.
N/A	IDMA Spare 0xa8c	New register in MV6436x. The software must not use this register. It is reserved for future use, only.
Channels 0-3 Arbiter Control 0x860	Arbiter Control 0x860	Some changes in MV6436x. New bit settings.
Channels 7-4 Arbiter Control 0x960	N/A	Not supported in MV6436x.

Table 6:GT6426x Vs. MV6436x PCI Registers (Continued)

GT6426x	MV6436x	Comments
Channels 0-3 Crossbar Timeout 0x8d0	Cross Bar Timeout 0x8d0	No changes.
Channels 7-4 Crossbar Timeout 0x9d0	N/A	Not supported in MV6436x.

Section 9. Interrupt Controller

9.1 Interrupt Controller Register Settings

The MV6436x interrupt controller registers are implemented as part of the CPU interface unit. This implementation minimizes read latency from the CPU interrupt handler.

This is not backward compatible with the GT6426x implementation (meaning the registers are placed in different offsets). The following table describes the interrupt cause and mask registers for each interrupt signal.

In addition, the cause registers in all of the units were changed, as were bit locations within the registers.

Table 7:Interrupt Cause and Mask Registers

MV6436x Interrupt Controller Registers	GT6426x Interrupt Controller Registers
Main Interrupt Cause (Low), offset 0x004	Main Interrupt Cause (Low), offset 0xc18
Main Interrupt Cause (High), offset 0x00c	Main Interrupt Cause (High), offset 0xc68
CPUInt[0]* Mask (Low), offset 0x014	CPU Interrupt Mask (Low), offset 0xc1c
CPUInt[0]* Mask (High), offset 0x01c	CPU Interrupt Mask (High), offset 0xc6c
CPUInt[0]* Select Cause, offset 0x024	CPU Select Cause, offset 0xc70
CPUInt[1]* Mask (Low), offset 0x034	CPUInt[1]* output signal is not available in theGT6426x.
CPUInt[1]* Mask (High), offset 0x03c	CPUInt[1]* output signal is not available in theGT6426x.
CPUInt[1]* Select Cause, offset 0x044	CPUInt[1]* output signal is not available in theGT6426x.
INT0* Mask (Low), offset 0x054	PCI_0 Interrupt Mask (Low), offset 0xc24
INT0* Mask (High), offset 0x05c	PCI_0 Interrupt Mask (High), offset 0xc64
INT0* Select Cause, offset 0x064	PCI_0 Select Cause, offset 0xc74
INT1* Interrupt Mask (Low), offset 0x074	PCI_1 Interrupt Mask (Low), offset 0xca4
INT1* Interrupt Mask (High), offset 0x07c	PCI_1 Interrupt Mask (High), offset 0xce4
INT1* Select Cause, offset 0x084	PCI_1 Select Cause, offset 0xcf4
CPU Int[0]* signal is not multiplexed on MPP in the MV6436x.	CPU Int[0]* Mask, offset 0xe60 (Multiplexed on MPP signals)
CPU Int[1]* signal is not multiplexed on MPP in the MV6436x.	CPU Int[1]* Mask, offset 0xe64 (Multiplexed on MPP signals)

Table 7:Interrupt Cause and Mask Registers (Continued)

MV6436x Interrupt Controller Registers	GT6426x Interrupt Controller Registers
CPU Int[2]* signal is not multiplexed on MPP in the MV6436x.	CPU Int[2]* Mask, offset 0xe68 (Multiplexed on MPP signals)
CPU Int[3]* signal is not multiplexed on MPP in the MV6436x.	CPU Int[3]* Mask, offset 0xe6c (Multiplexed on MPP signals)

Section 10. Integrated SRAM

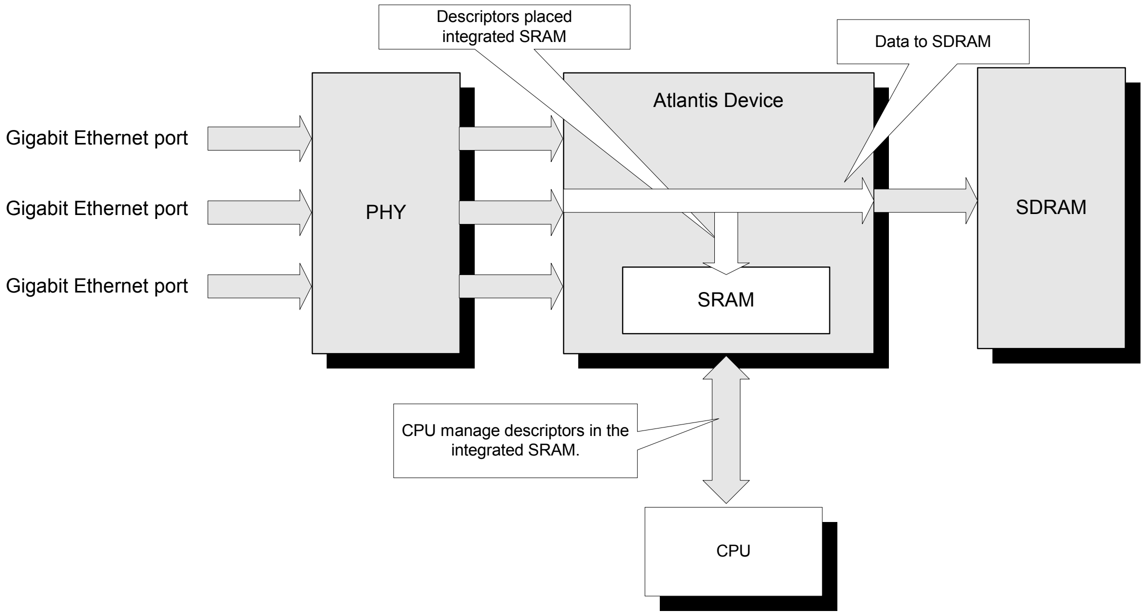
The MV6436x devices integrate 2MSRAM that is a general-purpose memory and accessible from any of the MV6436x units. Since it is an internal resource, its typical access time is much faster than access to external resources, such as SDRAM. Moreover, to achieve low CPU read latency, the SRAM is integrated as part of the CPU interface unit. This results in a CPU read latency that is as fast as six cycles.

The SRAM is accessible through the regular MV6436x address decoding logic. The CPU and PCI interfaces have a dedicated fixed size (256Kbyte) address windows per the integrated SRAM. All other units can assign one of their addresses decoding windows to be targeted to the integrated SRAM.

The integrated SRAM also supports parity generation and checking. It also supports cache coherency per cache coherent regions basis.

Figure 8 shows a typical application that takes advantage of the integrated SRAM. The application places the communication descriptors in the SRAM, this reduces the latency for descriptors access to the communication unit and to the CPU.

Figure 8: Application with Integrated SRAM

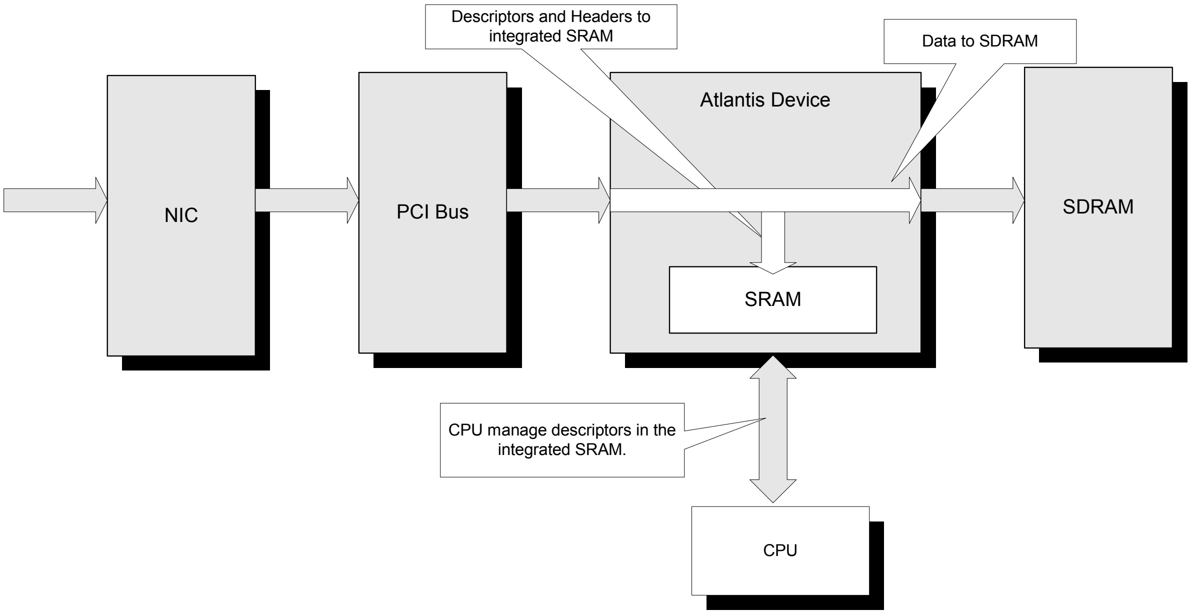


Note

For more information and register settings, see the MV6436x datasheet's Interrupt Controller section.

10.1 Headers Retarget

Since the SDRAM read latency is many times a performance bottleneck, it would be useful to place the packet headers in the low latency integrated SRAM and place the rest of the packet in DRAM. In a manner that is transparent to the application software, the Headers Retarget scheme enables re-directing of the headers to the integrated SRAM. From the software point of view, the whole packet goes to SDRAM.


Descriptors and Headers to integrated SRAM
Data to SDRAM
Atlantis Device
NIC
PCI Bus
SDRAM
SRAM
CPU manage descriptors in the integrated SRAM.
CPU

The headers retarget scheme supports a fixed size header of 64 bytes. This means the first 64 bytes of the buffer are retargeted to the integrated SRAM

Headers retarget control is supported on PCI0, PCI1, GbE, IDMA, and CPU Interface units. Each of the PCI0, PCI1, GbE, and IDMA units use two registers to identify the header space.

Table 8:Headers Retarget Control

Bits	Field	Type	Description
0	En	0x0	Headers Retarget Enable 0 = Disable 1 = Enable
3:1	BSize	0x0	Buffer Size 0 = 256 bytes 1 = 512 bytes 2 = 1 KB 3 = 2 KB 4 = 4 KB 5 = 8 KB 6 - 7= Reserved
15:4	Reserved	0x0	Reserved.

Table 8:Headers Retarget Control

Bits	Field	Type	Description
31:16	Mask1	0x0	In 64 KB granularity, defines the total space of the buffers to be manipulated. Size must be set from LSB to MSB as a sequence of 1's, followed by sequence of 0's. For example, to retarget the headers of 1K buffers of 1 KB size, which means 1MB of buffers space, Mask1 must be set to 0x000f **NOTE:** The total address space of retargeted headers must not exceed the integrated SRAM size (256 KB). The minimum buffers space to be manipulated is 64 KB.

Table 9:Header Retarget Base

Bits	Field	Type	Description
15:0	Reserved	0x0	Reserved.
31:16	Base	0x0	Base Address Retarget is executed if the address bits, which are not masked by Mask1, match the corresponding bits of the base. For example, if Mask1 is set to 0x000f (1MB of buffers), retarget is executed if address bits[31:20] are equal to Base[31:20].

In the CPU interface unit, there are four pairs of registers. Each register pair corresponds to one of the PCI0, PCI1, GbE, or IDMA units. These registers are used by the CPU interface to identify the headers space for transactions coming from the CPU, but also used by the integrated SRAM controller for remapping of the addresses (placing the headers consecutively).

Table 10:CPU Header Retarget Base and Remap

Bits	Field	Type	Description
3:0	Remap	0x0	Remap Address Defines the upper bits of headers space in SRAM.
7:4	Mask2	0x0	Defines how many of the integrated SRAM address upper bits must be remapped. If for example, the headers space is 64 KB (which is one quarter of the total SRAM space), Mask2 can be set to b'0011. This means that the two MSB bits of SRAM address (address bits[17:16]) are replaced with the corresponding Remap bits.
15:12	BaseH	0x0	High Base Address Corresponds to address bits[35:32].
31:16	Base	0x0	Base Address Retarget is executed if the address bits, unmasked by Mask1, matches the corresponding Base bits. Corresponds to CPU address bits[31:16]

For example, to retarget the headers of 2K buffers will require 128 KB of space in SRAM (header size is fixed to 64 Bytes), one half of the total SRAM space. If each buffer size is 8 KB, this means that 16 MB of space is manipulated and the base address is 0x1000000 (16 MB).

The following tables show the header retarget registers settings for the IDMA unit. It is the same for all other interfaces.

Note

The PCI interface has an additional register for 64-bits of address support.

Table 11:Headers Retarget Control
Offset:0xa84

Bits	Field	Type	Function
0	En	0x0	Headers retarget enable bit 0x0 = Disable 0x1 = Enable
3:1	BSize	0x0	Buffer Size 0x0 = 256 bytes 0x1 = 512 bytes 0x2 = 1 KB 0x3 = 2 KB 0x4 = 4 KB 0x5 = 8 KB 0x6 - 0x7= Reserved
15:4	Reserved	0x0	Read only
31:16	Mask1	0x0	Defines the total space of the buffers to be manipulated, in 64 KB granularity. Size must be set from LSB to MSB as a sequence of of 1’s, followed by sequence of 0’s. For example, in order to retarget the headers of 1K buffers of 1 KB size, which means 1 MB of buffers space, Mask1 should be set to 0x000f **NOTE:** The total address space of retargeted headers must not exceed integrated SRAM size (256 KB). The minimum buffers space to be manipulated is 64 KB

Table 12:Header Retarget Base
Offset:0xa88

Bits	Field	Type	Function
15:0	Reserved	0x0	Read only.
31:16	Base	0x0	Base address. Retarget is executed if address matches Base

Note

For proper operation, the CPU/IDMA Headers Retarget Control settings must be the same as IDMA Headers Retarget Control register.

Table 13:CPU/PCI0 Header Retarget Base
Offset:0x3b8

Bits	Field	Type	Description
3:0	Remap	0x0	Remap address. Defines the upper bits of headers space in SRAM.
7:4	Mask2	0x0	Defines how many of the integrated SRAM address upper bits should be remapped. If for example, the headers space is 64 KB (which is one quarter of the total SRAM space), Mask2 can be set to b'0011, which means that the two MSB bits of SRAM address (address bits[17:16]) will be replaced with the corresponding Remap bits).
15:12	BaseH	0x0	High Base address. Corresponds to address bits[35:32]
31:16	Base	0x0	Base address. Retarget is executed if address matches Base. Corresponds to CPU address bits[31:16]

Notes

- For the retarget scheme to work properly, the programing of the CPU interface registers must be the same as the programing in the corresponding unit register. Both registers must have same BSize and Mask1 settings, and should map the same DRAM space.
- It is the user responsibility to have the same cache coherency policy for DRAM and SRAM. If for example, the packets are placed in a cache coherent WB DRAM region, also configure the integrated SRAM to WB cache coherency policy
- If the integrated SRAM is also used for other tasks (such as descriptors memory), the retargeting scheme must not cause headers to override other data in SRAM.

Section 11. General Purpose Port

The MV6436x GPP implements a few additional features for SMP support that are not supported by the GT6426x. In SMP applications, there might be cases where each of the two CPUs requires handling a different set of GPP interrupts. For these cases, MV6436x implements two GPP Mask registers, one per CPU. Each CPU can select, via it’s mask register (GPP Interrupt Mask0 0xf10c and GPP Interrupt Mask1 0xf114), which GPP interrupts to handle.

Additionally, in SMP applications in which the GPP is configured as an output, there might be cases where each of the two CPUs requires handling a different set of GPP outputs. In these cases, both CPUs might access the value register concurrently and create a race condition. To prevent this condition occurring, two additional registers were implemented in the MV6436x - GPP Value Set (GPP Value Set 0xf118) and GPP Value Clear registers (GPP Value Clear 0xf11c). Writing a value of ’1’ to these registers bits sets/clears the corresponding bit in the GPP Value register.

Note

Writing a value of ’0’ has no affect on the corresponding bit in the GPP Value register.

The GT6426x implements two dedicated MPSCs interfaces. in the GT6426x, the MPSCs ports are multiplexed on the MPP interface.