



Centrality
COMMUNICATIONS

Atlas™-II AT460A-BI Application Processor

Developer's Manual

Revision 2.4, Aug , 2005

Information in this document is provided in connection with Centrality products. No license, express or implied, by estoppels or otherwise, to any intellectual property rights is granted by this document. Except as provided in Centrality's Terms and Conditions of Sale for such products, Centrality assumes no liability whatsoever, and Centrality disclaims any express or implied warranty, relating to sale and/or use of Centrality products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Centrality products are not intended for use in medical, life saving, or life sustaining applications.

Centrality may make changes to specifications and product descriptions at any time, without notice.

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Centrality reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

The Atlas-II may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Centrality sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document, or other Centrality literature may be obtained by calling 1-408-982-1818 or by visiting Centrality's website at <http://www.centralitycomm.com>.

Copyright © Centrality Communications Inc, 2004

*Third-party brands and names are the property of their respective owners.

ARM and the ARM Powered logo are trademarks of ARM Limited.

Table of Contents

1	Introduction	10
1.1	Overview.....	10
1.2	Example System.....	10
2	RISC Subsystem	10
2.1	Overview.....	10
2.2	Functional Description	10
2.2.1	ARM926EJ Processor.....	10
2.2.2	Address Mapping	10
2.2.3	Memory Bus Interface	10
2.2.4	I/O Bus Interface	10
2.3	RISC Subsystem Registers.....	10
3	DSP Subsystem.....	10
3.1	Overview.....	10
3.2	Functional Description.....	10
3.2.1	DSP Memory	10
3.2.2	DSP Host Interface.....	10
3.2.3	DSP IO Interface	10
3.3	DSP Subsystem Registers	10
4	System Memory Interface.....	10
4.1	Overview.....	10
4.2	System Arbiter	10
4.2.1	Overview	10
4.3	DRAM Controller	10
4.3.2	DDR Connection Examples	10
4.3.3	SDRAM Connection Examples	10
4.3.4	DRAM Controller Registers	10
5	System Control Modules.....	10
5.1	Mode Configuration Pins	10
5.1.1	Overview	10
5.1.2	Configuration Setup.....	10
5.2	Clock and PLL	10
5.2.3	Overview	10
5.2.4	Oscillator	10
5.2.5	PLL	10
5.2.6	Multiplexer & Divider	10
5.2.7	Clock Switching.....	10
5.2.8	Real Time Clock Registers.....	10
5.3	Power Manager	10
5.3.9	Overview	10
5.3.10	Power On/Off Sequence	10
5.3.11	Sleep/Wakeup Sequence.....	10
5.3.12	Power Manager Registers.....	10
5.4	Interrupt Controller.....	10
5.4.1	Overview	10
5.4.2	Interrupt Controller Registers	10
5.5	OS Timer	10
5.5.3	Overview	10
5.5.4	OS Timer Registers.....	10
5.6	Reset Controller.....	10
5.6.5	Overview	10
5.6.6	Reset Scheme.....	10
5.6.7	Power On Reset.....	10
5.6.8	Reset Controller Registers	10
5.7	Resource Sharing Controller	10

5.7.9	DMA Sharing	10
5.7.10	Pin Sharing	10
5.7.11	Resource Sharing Controller Registers	10
5.8	GPIO	10
5.8.12	Overview	10
5.8.13	GPIO Registers	10
5.9	PWM	10
5.9.14	Overview	10
5.9.15	PWM Registers	10
6	Graphic Display Subsystem	10
6.1	Overview	10
6.2	LCD Controller	10
6.2.1	Overview	10
6.2.2	Pin Description	10
6.2.3	Functional Description	10
6.2.4	LCD Controller Registers	10
6.3	2D/BitBLT Engine	10
6.3.5	Overview	10
6.3.6	Functional Description	10
6.3.7	2D/BitBLT Engine Registers	10
7	PCI Subsystem	10
7.1	Overview	10
7.2	System to PCI Bridge	10
7.2.1	PCI Bridge RISC IO Registers	10
7.2.2	PCI Bridge DSP IO Registers	10
7.3	ROM/SRAM Controller	10
7.3.1	Overview	10
7.3.2	Pin Description	10
7.3.3	Address Mapping	10
7.3.4	ROM/SRAM Controller Registers	10
7.4	SDIO Host Controller	10
7.4.1	Overview	10
7.4.2	Pin Description	10
7.4.3	Functional Description	10
7.4.4	SDIO Host Controller Registers	10
7.5	USB-OTG Interface	10
7.5.1	Overview	10
7.5.2	Pin Description	10
7.5.3	Functional Description	10
7.5.4	USB-OTG Interface Registers	10
7.6	IDE Interface	10
7.6.1	Overview	10
7.6.2	Pin Description	10
7.6.3	IDE Interface Registers	10
7.7	PCMCIA/CF Interface	10
7.7.4	Overview	10
7.7.5	PCMCIA Signal Descriptions	10
7.7.6	CF Card Connection Example	10
7.7.7	PCMCIA Memory Mapping	10
7.7.8	PCMCIA Interface Registers	10
8	Peripheral Subsystem	10
8.1	Overview	10
8.2	I/O Bridge	10
8.2.1	Overview	10
8.2.2	I/O Bridge Registers	10
8.3	DMA Controller	10

8.3.1	Overview	10
8.3.2	Functional Description	10
8.3.3	Basic DMA Operations	10
8.3.4	DMA Controller Registers	10
8.4	NAND Flash Interface	10
8.4.5	Overview	10
8.4.6	Pin Description	10
8.4.7	NAND Flash Interface Registers	10
8.5	Video Input Port	10
8.5.1	Overview	10
8.5.2	Functional Description	10
8.5.3	Video Input Port Registers	10
8.6	Audio CODEC	10
8.6.4	Overview	10
8.6.5	Signal Description	10
8.6.6	Functional Description	10
8.6.7	Audio CODEC Registers	10
8.7	UART	10
8.7.1	Overview	10
8.7.2	Pin Description	10
8.7.3	Functional Description	10
8.7.4	UART Registers	10
8.8	Universal Serial Port	10
8.8.1	Overview	10
8.8.2	Supported Protocols and Devices	10
8.8.3	Pin Description	10
8.8.4	Functional Description	10
8.8.5	USP Registers	10
8.9	CAN Bus Controller	10
8.9.1	Overview	10
8.9.2	Pin Description	10
8.9.3	CAN Bus Registers	10
8.10	PWM	10
8.10.1	Overview	10
8.10.2	PWM Registers	10
8.11	GPS Baseband	10
8.11.1	Overview	10
8.11.2	Pin Description	10
9	Physical Information	10
9.1	Pinout	10
9.2	Package	10
9.3	DC Electrical Characteristics	10
9.3.1	Absolute Maximum Ratings	10
9.3.2	Recommended Operation Conditions	10
9.3.3	DC Electrical Specifications	10
9.3.4	Electrostatic Discharge	10
9.3.5	Power Dissipation	10
9.3.6	Thermal Characteristics	10
9.4	AC Electrical Characteristics	10
9.4.7	AC Operation Frequency Data	10
9.4.8	Clock AC Specifications	10
9.4.9	Resets	10
9.4.10	External Interrupts	10
9.4.11	SDRAM (sdr32)	10
9.4.12	DDR SDRAM	10
9.4.13	Camera Specifications	10

9.4.14	Lcd Specifications.....	10
9.4.15	Nand rom Specifications.....	10
9.4.16	Nans cam Specifications	10
9.4.17	Ide Specifications.....	10
9.4.18	GPIO Specifications.....	10
9.4.19	Serial Port Specifications.....	10
9.4.20	CANBUS AC Specifications	10
9.4.21	PCMCIA AC Specifications	10
9.4.22	USBOTG AC Specifications	10
9.4.23	SDIO AC Specifications.....	10
9.4.24	JTAG AC Specifications	10
10	Revision History	10

List of Figures

Figure 1.	Atlas™-II Block Diagram.....	10
Figure 2.	Atlas™-II as an Infotainment and Navigation Product.....	10
Figure 3.	RISC Subsystem Block Diagram.....	10
Figure 4.	DSP Subsystem Block Diagram.....	10
Figure 5.	DDR Connection in 16-bit Mode.....	10
Figure 6.	DDR Connection in 32-bit Mode.....	10
Figure 7.	SDRAM Interface in 16-bit Mode.....	10
Figure 8.	SDRAM Interface in 32-bit Mode.....	10
Figure 9.	An Example of Mode Configuration Setup.....	10
Figure 10.	Atlas-2 Clock Circuit Block Diagram.....	10
Figure 11.	Oscillator Tank Circuit.....	10
Figure 12.	PG13A1G3 Block Diagram.....	10
Figure 13.	PG13E3G Block Diagram.....	10
Figure 14.	Power on/off Sequence.....	10
Figure 15.	Sleep/Wakeup Sequence.....	10
Figure 16.	Interrupt Controller Block Diagram.....	10
Figure 17.	Block Diagram of Layer Mixer.....	10
Figure 18.	8bits/pixel data in a DWORD (Big Endian).....	10
Figure 19.	12bits/pixel data in a DWORD (Little Endian).....	10
Figure 20.	2bits/pixel data in a DWORD (Big Endian in a byte, Little Endian for byte in a DWORD).....	10
Figure 21.	2bits/pixel data in a DWORD (Big Endian in a byte, Little Endian for byte in a DWORD).....	10
Figure 22.	Screen and OSD Active Region.....	10
Figure 23.	LCD Sync Signals Waveform.....	10
Figure 24.	LCD Alteration Signals Waveform.....	10
Figure 25.	Screen and OSD FIFOs Request Level.....	10
Figure 26.	FIFO Suppress.....	10
Figure 27.	LCD Controller DMA Memory Layout.....	10
Figure 28.	LCD Controller DMA Memory 2-D Layout.....	10
Figure 29.	Pixel Offset.....	10
Figure 30.	Block Diagram of BitBLT Engine.....	10
Figure 31.	BitBLT 2D window dimension.....	10
Figure 32.	ROP2/3 Code Definitions.....	10
Figure 33.	Block Diagram of PCI Bridge.....	10
Figure 34.	32-bit Data Fix-latency Read (BURST_READ=0, DWORD_ACCESS=1, BUS_WIDTH=1, VARI_ACC=0).....	10
Figure 35.	16-bit Data Fix-latency Read (BURST_READ=0, DWORD_ACCESS=0, BUS_WIDTH=1, VARI_ACC=0).....	10
Figure 36.	32-bit Data Fix-latency Read (BURST_READ=1, DWORD_ACCESS=1, BUS_WIDTH=0/1/2, VARI_ACC=0).....	10
Figure 37.	32-bit Data Fix latency Write (BURST_WRITE=1, DWORD_ACCESS=1, BUS_WIDTH=0/1/2, VARI_ACC=0).....	10
Figure 38.	32-bit Data Fix-latency Write (BURST_WRITE=0, DWORD_ACCESS=1, BUS_WIDTH=1, VARI_ACC=0).....	10
Figure 39.	16-bit Data Variable-latency Read (BURST_READ=0, DWORD_ACCESS=0, BUS_WIDTH=1, VARI_ACC=1).....	10
Figure 40.	16-bit Data Variable-latency Write (BURST_WRITE=0, DWORD_ACCESS=0, BUS_WIDTH=1, VARI_ACC=1).....	10
Figure 41.	SD Host Block Diagram.....	10
Figure 42.	USB-OTG Controller Block Diagram.....	10
Figure 43.	Block Diagram of USB-OTG transceiver.....	10
Figure 44.	PCMCIA Block Diagram.....	10
Figure 45.	Connect to Compact Flash Card.....	10
Figure 46.	PCMCIA Memory Mapping.....	10
Figure 47.	Peripheral Subsystem Diagram.....	10
Figure 48.	DMA Controller Diagram.....	10

Figure 49.	DMA Controller Block Diagram	10
Figure 50.	FIFO Request Level Checkpoints	10
Figure 51.	DMA Configuration Register-file.....	10
Figure 52.	2-D DMA.....	10
Figure 53.	2-D DMA Wrap Around (X-length > DMA Width).....	10
Figure 54.	DMA address change if X_LEN=0	10
Figure 55.	NAND Flash Controller Block Diagram	10
Figure 56.	Block Diagram of Video Input Port.....	10
Figure 57.	16-bit RGB Data Format.....	10
Figure 58.	Contraction in Bayer RGB Mode with 1:2 Ratio in Row and Column	10
Figure 59.	Contraction in Bayer RGB Mode with 1:4 Ratio in Row and Column	10
Figure 60.	Contraction in YUV/YCrCb Mode (YUVRGB=1) with 1:2 ratio in Row and Column.....	10
Figure 61.	Contraction in YUV/YCrCb Mode (YUVRGB=1) with 1:4 ratio in Row and Column.....	10
Figure 62.	Active Region	10
Figure 63.	Block Diagram of Audio CODEC Interface.....	10
Figure 64.	AC-link Output Frame Read Command Diagram	10
Figure 65.	AC-link Output Frame Write Command Diagram.....	10
Figure 66.	AC-link Input Frame Command Diagram.....	10
Figure 67.	AC97 PCM Record Functional Block Diagram	10
Figure 68.	AC97 PCM Record Flowchart	10
Figure 69.	Stereo Mode Flowchart	10
Figure 70.	Mono Mode (Left/Right) Flowcharts	10
Figure 71.	AC97 PCM Playback Functional Diagram	10
Figure 72.	UART0 Functional Block Diagram	10
Figure 73.	CAN Bus Interface Connection	10
Figure 74.	CAN Bus Timing.....	10
Figure 75.	Diagram of GPS Baseband.....	10
Figure 76.	Timing Diagram—X_XIN	10
Figure 77.	Timing Diagram—power and reset Timing.....	10
Figure 78.	Timing Diagram—Standard SDRAM Memory Read Timing	10
Figure 79.	Timing Diagram—Standard SDRAM Memory Write Timing	10
Figure 80.	Timing Diagram—DDR SDRAM Memory Read Timing.....	10
Figure 81.	DDR SDRAM Memory Write Timing	10
Figure 82.	Camera data Timing.....	10
Figure 83.	Lcd data Timing.....	10
Figure 84.	Nand rom read data Timing	10
Figure 85.	Nand rom write data Timing	10
Figure 86.	GPIO data Timing.....	10
Figure 87.	Serial Port AC Timing.....	10
Figure 88.	Timing Diagram—USB Output Line	10
Figure 89.	Timing Diagram—JTAG Input/Output Line	10

List of Tables

Table 1.	System Memory Mapping	10
Table 2.	Internal Register Mapping.....	10
Table 3.	RISC Subsystem Register Mapping	10
Table 4.	DSP Memory Address Mapping	10
Table 5.	DSP ISA-IDMA Control Registers.....	10
Table 6.	DSP IO Device Address Mapping.....	10
Table 7.	DSP Subsystem Register Mapping	10
Table 8.	System Bus Master Allocation	10
Table 9.	DDR/SDRAM Configurations for a 16-bit Data Bus.....	10
Table 10.	DDR/SDRAM Configurations of 32-bit Data Bus	10
Table 11.	RISC Interface Register Address Mapping	10
Table 12.	Mode Configuration Pins.....	10
Table 13.	Decoding of JTAG_MODE<1:0>	10
Table 14.	Selection Guide of Crystal for PDXOE3DG	10
Table 15.	Supported Clock Ratios in Atlas-2 (unit: MHz).....	10
Table 16.	RTC Register Mapping.....	10
Table 17.	PWR_MGR Register Mapping	10
Table 18.	Interrupt Controller Register Mapping	10
Table 19.	OS Timer Register Mapping.....	10
Table 20.	Reset Scheme of Each Module.....	10
Table 21.	Reset Controller Register Mapping	10
Table 22.	DMA Channels Allocation.....	10
Table 23.	RSC Register Mapping.....	10
Table 24.	GPIO Pin Mux	10
Table 25.	GPIO Register Mapping	10
Table 26.	PWM Interface Register Mapping	10
Table 27.	Atlas™-II LCD Controller Pin Descriptions.....	10
Table 28.	Atlas™-II LCD Controller Data Pins for passive displays	10
Table 29.	LCD Controller Register Mapping	10
Table 30.	BitBLT Engine Register Mapping	10
Table 31.	PCI Device Registers Mapping	10
Table 32.	PCI Bridge RISC IO Register Mapping	10
Table 33.	PCI Bridge DSP IO Register Mapping	10
Table 34.	ROM/SRAM Interface Pin Description	10
Table 35.	RISC IO access/PCI IO access address Mapping.....	10
Table 36.	RISC IO access/PCI IO access width translation	10
Table 37.	ROM RISC IO Register Mapping	10
Table 38.	SDIO Host Interface Pin Description.....	10
Table 39.	SDIO Host Controller Register Mapping	10
Table 40.	Definition of Transfer Type.....	10
Table 41.	Definition of Response Type.....	10
Table 42.	Response Bit Definition for Each Response Type.....	10
Table 43.	Relation between Transfer Complete and Data Timeout Error.....	10
Table 44.	Relation between Command Complete and Command Timeout Error	10
Table 45.	Relation between Command CRC Error and Command Timeout Error	10
Table 46.	Relation between Auto CMD12 CRC Error and Auto CMD12 Timeout Error	10
Table 47.	Relation between Auto CMD12 CRC Error and Auto CMD12 Timeout Error	10
Table 48.	Maximum Current Value Definition	10
Table 49.	USB OTG Interface Pin Description.....	10
Table 50.	USB-OTG Interface Register Mapping.....	10
Table 51.	ROM/SRAM Interface Pin Description	10
Table 52.	IDE Configuration Register Mapping.....	10
Table 53.	PIO Timing Programming.....	10
Table 54.	IDE Bus Master Control Register Mapping	10

Table 55.	IDE Device Port Map.....	10
Table 56.	PCMCIA Interface Signal Description	10
Table 57.	PCMCIA Configuration Register (PCMCIA_CONFIG) – 0x57C00000	10
Table 58.	M6730 Device Control Register Mapping	10
Table 59.	M6730 Power Control Register (1).....	10
Table 60.	M6730 Power Control Register (2).....	10
Table 61.	Extended Indexed Registers	10
Table 62.	I/O Memory Bus Masters.....	10
Table 63.	I/O Bridge Register Mapping	10
Table 64.	FIFO Requests Level	10
Table 65.	DMA Controller Register Mapping	10
Table 66.	NAND Flash Interface Signal Descriptions	10
Table 67.	NAND Flash Interface Register Mapping	10
Table 68.	YUV/YCrCb 4:2:2 16 bit format	10
Table 69.	YUV/YCrCb 4:2:2 8 bit format	10
Table 70.	Shift example.....	10
Table 71.	YUV2RGB Conversion Coefficient.....	10
Table 72.	Video Input Port Register Mapping	10
Table 73.	AC97 CODEC Interface Pin Description	10
Table 74.	AC-link Output Slots	10
Table 75.	AC-link Input Slots.....	10
Table 76.	Audio CODEC Register Mapping	10
Table 77.	AC97_WRITE_TAGH Description.....	10
Table 78.	UART0 Pin Descriptions.....	10
Table 79.	UART Register Mapping	10
Table 80.	USP Pin Descriptions	10
Table 81.	USP Register Mapping.....	10
Table 82.	CAN Bus Pin Descriptions	10
Table 83.	CAN Bus Address Mapping	10
Table 84.	CAN Bus DMA Register Mapping	10
Table 85.	Transmit Buffer Layout.....	10
Table 86.	Descriptor Field of Transmit Buffer	10
Table 87.	Receive Buffer Layout.....	10
Table 88.	Descriptor Field of Receive Buffer	10
Table 89.	Filter Bit Patterns.....	10
Table 90.	CAN Bus DMA Register Mapping	10
Table 91.	PWM Interface Register Mapping	10
Table 92.	GPS Pin Descriptions.....	10
Table 93.	AT460A-BI Pinout.....	10
Table 95.	Recommended Operating Conditions1	10
Table 96.	DC Electrical Specifications	10
Table 97.	Drive Capability of AtlasII Output Pins	10
Table 98.	ESD Characteristics	10
Table 99.	Power Dissipation.....	10
Table 100.	Thermal Resistance Data.....	10
Table 101.	Clock Frequencies.....	10
Table 102.	X_XIN Timing	10
Table 103.	Minimum pulse width for external interrupts to be recognized.....	10
Table 104.	Standard SDRAM Memory Read Timing	10
Table 105.	Standard SDRAM Write Timing	10
Table 106.	DDR SDRAM Memory Read Timing	10
Table 107.	DDR SDRAM Memory Write Timing	10
Table 108.	Camera data Timing.....	10
Table 109.	Lcd data Timing.....	10
Table 110.	Nand rom read data Timing	10
Table 111.	Nand rom write data Timing	10

Table 112.	GPIO data Timing.....	10
Table 113.	Serial Port AC Timing.....	10
Table 114.	Timing Specifications—USB Output Line.....	10
Table 115.	JTAG Timing Specification.....	10

1 Introduction

1.1 Overview

Centrality® Atlas™-II AT460A-BI application processor is a highly integrated SoC that incorporates a 32-bit ARM® RISC processor core, a 16-bit DSP core, a GPS baseband, an LCD controller, a SDRAM/DDR memory controller, and multiple peripheral interfaces.

Compared with Atlas™-I application processor family, the AT460A-BI provides higher performance, lower power consumption, and lower cost.

The following diagram shows the block diagram of AT460A-BI. The different colors show the difference between Atlas™-II and Atlas™-I.

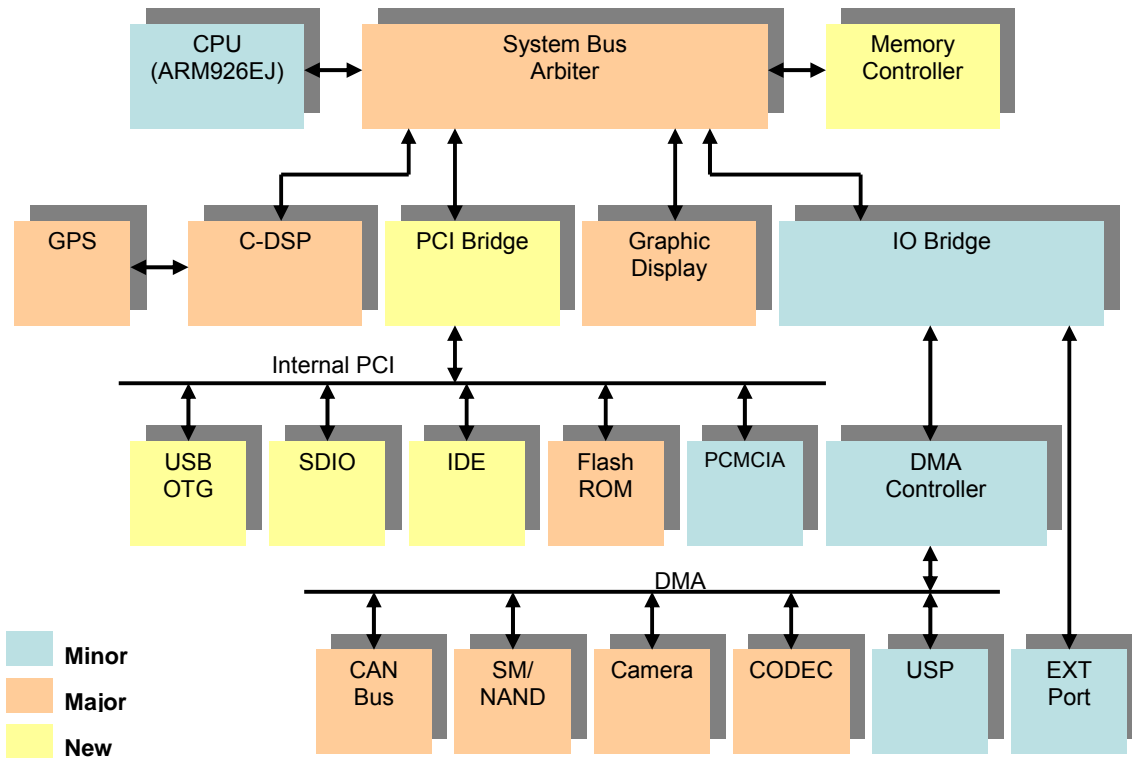


Figure 1. Atlas™-II Block Diagram

Features of Atlas™-II Application Processor

- 276MHz 32-bit RISC (ARM926EJ)
 - 16KB I-Cache
 - 16KB D-Cache
 - Jazelle™ Java Acceleration
 - ETM9 Embedded Trace Module
- 200MHz 16-bit DSP enhanced with hardware acceleration

- 5Kx24bit Program Memory
 - 10Kx16bit Data memory
- 200MHz DDR or 138MHz SDRAM interface
 - 3.3V SDRAM
 - 2.5V DDR
- Enhanced 16 Channel GPS baseband
- Color-TFT LCD panel interface
- 8/16-bit NAND Flash interface
- CMOS sensor interface
- USB OTG 2.0 full-speed interface
- ATA interface
- SDIO interface
- PCMCIA/CF card interface
- 2 CANBus ports
- 5 Universal Serial Ports and 3 UARTs
- PWM interface
- 6 OS timers
- 190 programmable GPIO
- Advanced power management
 - Normal mode
 - Idle mode
 - Sleep mode
- Internal 1GHz and 400MHz PLL
- Internal 12MHz and 32.768KHz oscillator
- TSMC 0.13G process with Dual-Vt
- 324-ball 19x19mm lead-free uBGA package

1.2 Example System

Following figure shows how the Atlas™-II can be used in Telematics Infotainment and Navigation system.

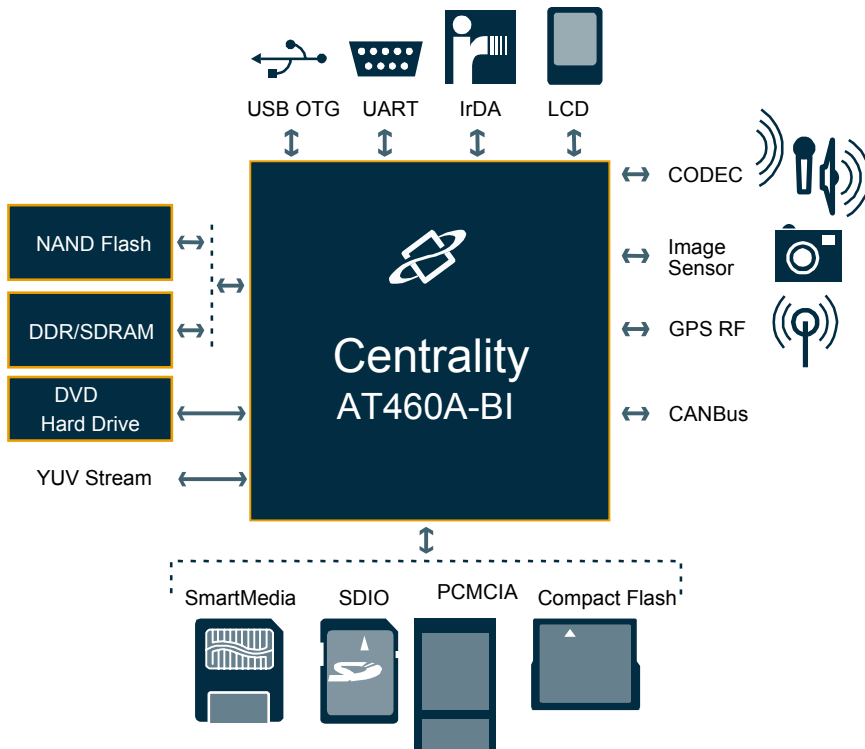


Figure 2. Atlas™-II as an Infotainment and Navigation Product

2 RISC Subsystem

2.1 Overview

The RISC subsystem contains the ARM926EJ core, the instruction and data caches, and an RISC interface block.

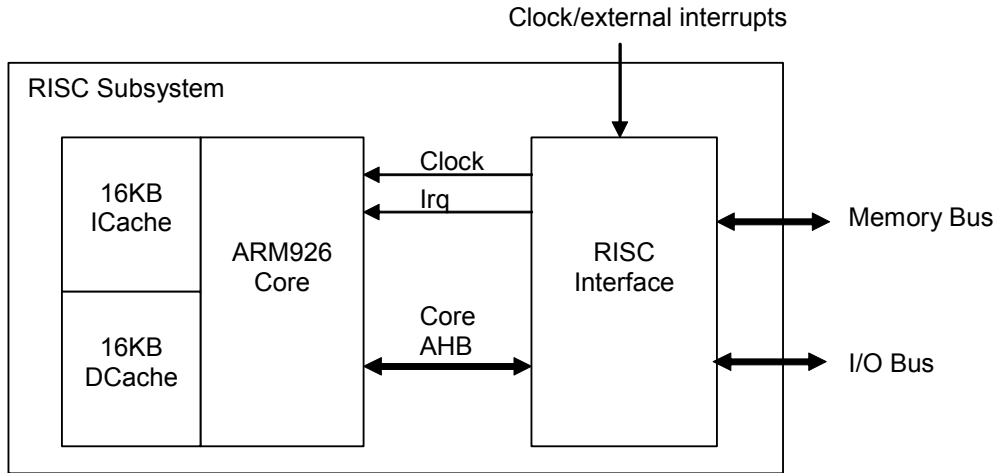


Figure 3. RISC Subsystem Block Diagram

2.2 Functional Description

2.2.1 ARM926EJ Processor

The ARM926EJ RISC Core in Atlas™-II is a hardened version of ARM926EJ-S processor, which is a member of the ARM9 family of general-purpose microprocessors. The ARM926EJ-S processor is targeted at multi-tasking applications where full memory management, high performance, small die size, and low power are all important.

The ARM926EJ-S processor supports the 32-bit ARM and 16-bit THUMB instruction sets, enabling the user to trade off between high performance and high code density. The ARM926EJ-S processor includes features of efficient cod execution of Java byte codes, providing Java performance close to JIT, but without the associated code overhead.

The ARM926EJ-S processor has a Harvard cached architecture (version 5TEJ) and provides a complete high-performance processor subsystem, including the MMU and AHB bus unit. The ARM926EJ-S is a synthesizable core that allows user to configure the cache size between 4KB and 128KB. But ARM926EJ_1616 is the hardened version with fixed 16KB I-Cache and 16KB D-Cache.

2.2.2 Address Mapping

The ARM926EJ has a 32-bit address bus, which will be translated by the RISC interface into either an access to cacheable data memory, non-cacheable data memory, or memory-mapped registers. Bits <26:0> of the address are used as the physical address bus.

All address mapping registers must be inside the RISC interface. The RISC interface will be responsible for all address decoding before it issues the command cycles to the system. The boot ROM should have instructions about how to initialize these address-mapped registers. The programmer needs to provide the system with the initialization routine.

- **ROM & PCMCIA**

The address mapping is defined starting from CPU reset vector (0x0000-0000). There is 512MB set aside for ROM. However, not all of this space can be used. For security reasons, this memory must be mirrored. Thus, the maximum size that can be set for the ROM cannot exceed 256MB. Also, because the mirrored memory is laid out onto two sequential 256MB segments, accessing 0x0000-0000 and 0x1000-0000 directly will yield the same result¹.

The memory space from 512MB to 1GB (0x2000-0000~0x3FFF-FFFF) is allocated to two PCMCIA sockets. Each socket takes 256MB of memory.

- **DSP Shared Memory**

The DSP shared memory takes 128MB address space (0x4800-0000~0x4FFF-FFFF).

- **PCI Address Space**

The address between 0x5000-0000 and 0x5FFF-FFFF (256MB) is assigned to all the internal PCI devices. And there is another 256MB space above it (0x6000-0000~0x6FFF-FFFF) assigned to the PCI ROM/Flash².

- **Reserved Space**

The 256MB address space between 0x7000-0000~0x7FFF-FFFF is reserved. If the CPU reads from the reserved address space, a data abort operation will result. Writes to the reserved address space have no effects.

- **Internal Registers**

Every peripheral device occupies 64K-byte space starting from 2GB to 3GB (0x8000-0000~0xBFFF-FFFF).

- **System Memory**

The system memory is between 3GB and 4GB (0xC000-0000~0xFFFF-FFFF). But the maximum memory size is 512MB. The top 512MB (0xE000-0000~0xFFFF-FFFF) is Zero Bank (reading from this address range will return zero; writing to this address range has no effect.) The actual memory size is also defined in the boot ROM or by the memory auto-sizing program.

The following table shows the memory address mapping of Atlas™-II.

Table 1. System Memory Mapping

Address Range	Usage	Resource Size
0xE000_0000~FFFF_FFFF	Zero Bank	512MB
0xC000_0000~DFFF_FFFF	System Memory	512MB
0x8000_0000~BFFF_FFFF	Internal Registers	1GB
0x7000_0000~7FFF_FFFF	Reserved	256MB
0x6000_0000~6FFF_FFFF	PCI Flash/ROM	256MB
0x5000_0000~5FFF_FFFF	Other PCI Devices	256MB
0x4800_0000~4FFF_FFFF	DSP Shared Memory	128MB
0x4000_0000~47FF_FFFF	Reserved	128MB

¹ After boot-up, the address space from 0x1000-0000~0x0FFF-FFFF can be re-directed to the system memory (DDR/SDRAM); and then user can only access the ROM/Flash through 0x1000-0000~0x1FFF-FFFF.

² The ROM/Flash interface can be accessed as a PCI device too; in this case, user can access the ROM/Flash through PCI address space: 0x6000-0000~0x6FFF-FFFF.

0x3000_0000~3FFF_FFFF	PCMCIA Socket 1	256MB
0x2000_0000~2FFF_FFFF	PCMCIA Socket 0	256MB
0x0000_0000~1FFF_FFFF	Flash/ROM	512MB

2.2.3 Memory Bus Interface

The Memory Bus Interface is responsible for translating the ARM926EJ AHB bus protocol to the system bus protocol.

2.2.4 I/O Bus Interface

If the address the RISC issues is to a memory-mapped register, or an I/O device, or the FIFO storage, the read/write cycle will be redirected to the RISC I/O Bus Interface.

All the registers, interface to I/O devices, and FIFO storage in Atlas™-II will be based on 32-bit addresses (except for some registers in SDIO and IDE interface). Some of the registers are in System Clock Domain and others are in I/O Clock Domain. For those registers in I/O Clock Domain, RISC may insert wait states while accessing those devices. The users can change the number of wait states that can be programmed by setting the wait state register. There are only three exceptions:

1. When the RISC accesses the variable latency IO device, the I/O cycle maybe variant in length.
2. When RISC access the DSP shared memory through IDMA port.
3. When RISC read from the Flash/ROM controller at boot-up, the latency is also variable.

The following table shows the internal register mapping:

Table 2. Internal Register Mapping

Address Range	Device Mapped	Clock Domain
0x8000_0000~0x8000_FFFF	Serial Port 0	IOCLK
0x8001_0000~0x8001_FFFF	Serial Port 1	IOCLK
0x8002_0000~0x8002_FFFF	Serial Port 2	IOCLK
0x8003_0000~0x8003_FFFF	Serial Port 3	IOCLK
0x8004_0000~0x8004_FFFF	Serial Port 4	IOCLK
0x8005_0000~0x8005_FFFF	Serial Port 5	IOCLK
0x8006_0000~0x8006_FFFF	CODEC Interface	IOCLK
0x8007_0000~0x8007_FFFF	SmartMedia ® Interface	IOCLK
0x8008_0000~0x8008_FFFF	Video Input Port	IOCLK
0x8009_0000~0x8009_FFFF	GPIO	IOCLK
0x800A_0000~0x800A_FFFF	GPS Interface	IOCLK
0x800B_0000~0x800B_FFFF	Reserved	-
0x800C_0000~0x800C_FFFF	PCI Bridge	IOCLK
0x800D_0000~0x800D_FFFF	CANBus0 Interface	IOCLK
0x800E_0000~0x800E_FFFF	CANBus1 Interface	IOCLK
0x800F_0000~0x800F_FFFF	PWM	IOCLK
0x8010_0000~0x8010_FFFF	Serial Port 6	IOCLK
0x8011_0000~0x8011_FFFF	Serial Port 7	IOCLK
0x8012_0000~0x8FFF_FFFF	Reserved	-
0x9000_0000~0x9000_FFFF	RISC Interface	SYCLK
0x9001_0000~0x9001_FFFF	DSP Interface	SYCLK
0x9002_0000~0x9002_FFFF	Interrupt Controller Interface	SYCLK
0x9003_0000~0x9003_FFFF	Resource Sharing Controller	SYCLK

0x9004_0000~0x9004_FFFF	Real Time Controller	SYSCLK
0x9005_0000~0x9005_FFFF	OS Timer	SYSCLK
0x9006_0000~0x9006_FFFF	Power Manager	SYSCLK
0x9007_0000~0x9007_FFFF	Reset Controller	SYSCLK
0x9008_0000~0x9FFF_FFFF	Reserved	-
0xA000_0000~0xA000_FFFF	Memory Controller & Arbiter	SYSCLK
0xA001_0000~0xA001_FFFF	ROM Controller	IOCLK
0xA002_0000~0xAFFF_FFFF	Reserved	-
0xB000_0000~0xB000_FFFF	DMA Controller & I/O Bridge	IOCLK
0xB001_0000~0xB00F_FFFF	Reserved	-
0xB800_0000~0xB800_FFFF	LCD Controller	SYSCLK
0xB011_0000~0xBFFF_FFFF	Reserved	-

2.3 RISC Subsystem Registers

Table 3. RISC Subsystem Register Mapping

RISC Address <11:0>	Register	Description
0x0000	RISCINT_FIFO_FLUSH	FIFO Flush Register
0x0004	RISCINT_MEM_SIZE	External memory size Register
0x0008	RISCINT_ROM_SIZE	ROM size Register
0x000C	RISCINT_BOOT_UP	ROM boot-up Register
0x0010	RISCINT_WAIT1	Wait states Register
0x0014	RISCINT_WAIT2	Wait states Register
0x0018	RISCINT_WIDTH	Width control Register
0x001C	RISCINT_TIMEOUT	Timeout Register
0x0020	RISCINT_TIMEOUT_INT	Timeout Interrupt Register
0x0024	RISCINT_PREFETCH_EN	Prefetch Enable Register
Others	-	Reserved

- **RISC Interface FIFO Flush Register (RISCINT_FIFO_FLUSH) – 0x0000**

The RISC interface has three post-write FIFO: one for cacheable memory accesses, one for non-cacheable memory accesses, and the other for I/O accesses. The FIFO will be flushed in the following cases:

- The FIFO is full and there are more data written.
- Interrupt or Abort exception happen.
- User writes a "1" to the control bit of this register.

After the FIFO is flushed, the flush control bit will be cleared automatically.

Bit	Name	Default	Description
0 (R/W)	FIFO_FLUSH	1'b0	FIFO flush-control bit. Set to 1 will flush the FIFO.
1 (R/W)	AUTO_FIFO	1'b1	FIFO auto-flush bit. Set to 1 will enable the automatic flush when exceptions happen.
2 (R)	FIFO_EMPTY	1'b0	FIFO empty bit.
31:3	-	29'h0	Reserved.

- **RISC Interface Cacheable Memory Size Register (RISCINT_MEM_SIZE) – 0x0004**

During boot-up, the boot program needs to do memory auto-sizing and then writes the cacheable/non-cacheable memory size into the corresponding registers in RISC interface.

Bit	Name	Default	Description
24:0 (R/W)	MS<24:0>	25'h1FFFFFF	Cacheable memory size in 32-bit D-word.
31:25	-	7'h0	Reserved.

- **RISC Interface ROM Size Register (RISCINT_ROM_SIZE) – 0x0008**

Bit	Name	Default	Description
28:0 (R/W)	RS<28:0>	29'h1FFFFFFF	ROM size in byte.
31:29	-	3'h0	Reserved.

- **RISC Interface Boot-up Register (RISCINT_BOOT_UP) – 0x000C**

After boot-up, the boot program needs to do re-direct the ROM access to the shadowed SDRAM memory space. To make sure the setting taking effect as soon as possible, user can issue a Flush IO_FIFO instruction right after set this Boot-up register.

Due to the pipeline nature of the RISC core, after the Boot-up register been set, user CANNOT access the ROM address space at once. It needs to insert at least one NOP between them.

Bit	Name	Default	Description
0 (R/W)	COLD_BOOT	1'b0	Set to 1 when system is cold boot-up.
1 (R/W)	WARM_BOOT	1'b0	Set to 1 when system is warm boot-up.
31:2	-	30'h0	Reserved.

- **RISC Interface Wait States Register (RISCINT_WAIT1) – 0x0010**

RISC I/O read cycle can be inserted with up to 16 wait states. There are 4 different wait values to be set at the same time. Each value is for some group of internal devices.

Bit	Name	Default	Description
7:0 (R/W)	WS0<7:0>	8'h0	Wait states number used by RISC Interface and Interrupt Controller.
15:8 (R/W)	WS1<7:0>	8'h1	Wait states number used by DSP interface, GPS interface, Graphic controller and memory controller.
23:16 (R/W)	WS2<7:0>	8'h2	Wait states number used by OS Timer.
31:24 (R/W)	WS3<7:0>	8'h3	Wait states number used by Power Manager and Resource Sharing Controller, Reset Controller, and Real-time Clock.

- **RISC Interface Wait States Register (RISCINT_WAIT2) – 0x0014**

RISC I/O read cycle can be inserted with up to 16 wait states. There are 4 different wait values to be set at the same time. Each value is for some group of internal devices.

Bit	Name	Default	Description
-----	------	---------	-------------

7:0 (R/W)	WS4<7:0>	8'h4	Wait states number used by SM/DF interface, Video Input Port, Audio CODEC, UART6, and UART7.
15:8 (R/W)	WS5<7:0>	8'h5	Wait states number used by PCI Bridge, DMA Controller, and USP5.
23:16 (R/W)	WS6<7:0>	8'h6	Wait states number used by ROM Controller, PWM, and USP0~USP4.
31:24 (R/W)	WS7<7:0>	8'h7	Wait states number used by CAN Bus interface and GPIO.

- **RISC Interface Width Control Register (RISCINT_WIDTH) – 0x0018**

RISC I/O Bus can access two different clock domains, one is SYS_CLK and the other is IO_CLK. SYS_CLK can be 1X, 2X, or 4X of the IO_CLK. So the write pulse width should be able to be programmable.

Bit	Name	Default	Description
3:0 (R/W)	PW0<3:0>	4'h0	Write pulse width for System clock domain.
7:4 (R/W)	PW1<3:0>	4'h1	Write pulse width for IO clock domain. When SYS_CLK:IO_CLK = 1, PW1 = 0; When SYS_CLK:IO_CLK = 2, PW1 = 1; When SYS_CLK:IO_CLK = 4, PW1 = 3.
31:8	-	24'h0	Reserved.

- **RISC Interface Timeout Register (RISCINT_TIMEOUT) – 0x001C**

When RISC accesses I/O device and the devices has no response in a certain period, the RISC interface will be timeout and generate an interrupt.

Bit	Name	Default	Description
15:0 (R/W)	TO<15:0>	16'hFFFF	Timeout value.
30:16	-	15'h0	Reserved.
31 (R/W)	TO_EN	1'b0	Timeout enable. 1 – Enable timeout check. 0 – Disable timeout check (default).

- **RISC Interface Timeout Interrupt Register (RISCINT_TIMEOUT_INT) – 0x0020**

When RISC accesses I/O device and the devices has no response in a certain period, the RISC interface will be timeout and generate an interrupt.

Bit	Name	Default	Description
0 (R/W)	TO_INT	1'b0	Timeout interrupt status. 1 – There is timeout interrupt. 0 – No timeout interrupt. Write a 1 to this bit will clear the interrupt.
31:1	-	31'h0	Reserved.

- **RISC Interface Pre-fetch Enable Register (RISCINT_PREFETCH_EN) – 0x0024**

There is a pre-fetch FIFO inside of the RISC interface. When Atlas™-II uses DDR or 32-bit SDRAM as system memory, this FIFO can be enabled to speed up the CPU read.

Bit	Name	Default	Description
-----	------	---------	-------------

0 (R/W)	PREFETCH_EN	1'b1	Pre-fetch enable bit. 1 – Enabled 0 – Disabled
31:1	-	31'h0	Reserved.

NOTE: This register bit has to be disabled in 16-bit SDRAM mode.

3 DSP Subsystem

3.1 Overview

The DSP Subsystem in Atlas™-II is similar to the Atlas™-I: it includes a 16-bit general-purpose DSP core, Program & Data memory, System bus interface, and Host Interface.

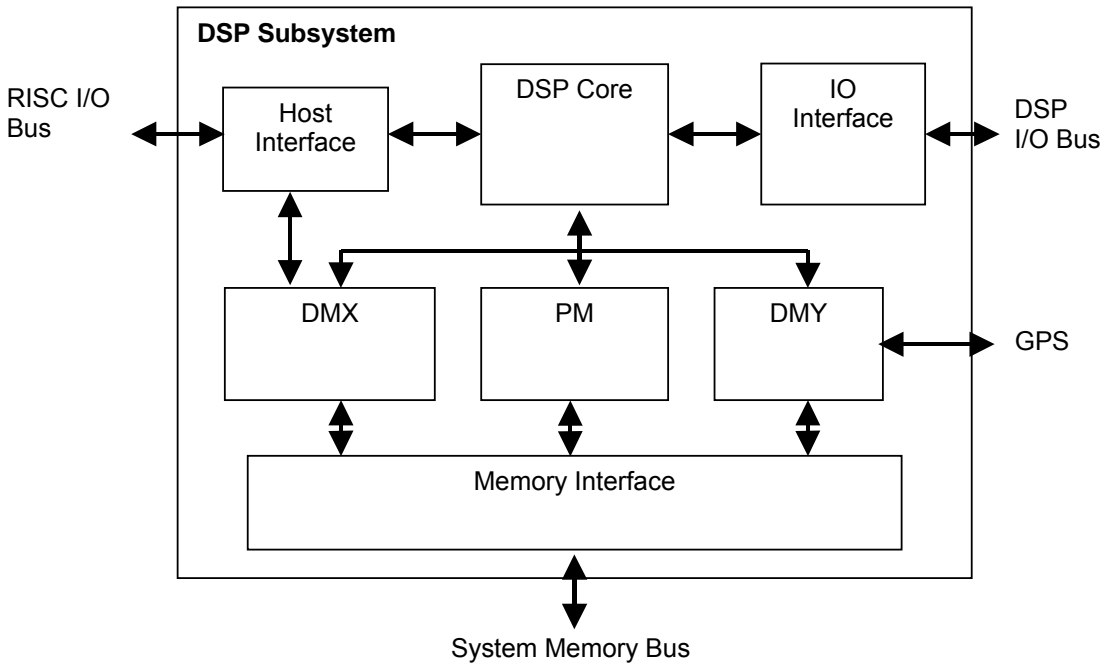


Figure 4. DSP Subsystem Block Diagram

But the DSP Subsystem in Atlas™-II provides higher performance and more features as following:

- Faster speed (200MHz)
- 4x larger Memory than Atlas™-1
- Support 1X and 2X1 Clock modes
- Idle/Power down mode

The flexible architecture and comprehensive instruction set allow the DSP to perform multiple operations simultaneously. In one processor cycle, the processor can:

- Generate the next program address
- Fetch the next instruction
- Perform one or two data moves
- Update one or two data address pointers
- Perform a computational operation

3.2 Functional Description

3.2.1 DSP Memory

¹ 2X means the DSP runs at twice clock frequency as the system bus.

There're three memories in Atlas™-II DSP Subsystem:

- Program Memory (PM)
- Data Memory (DMX)
- Coefficient Memory (DMY)

Three-memory architecture allows the DSP core to fetch two operands in a single cycle, one from coefficient memory and one from data memory. In the same cycle, the DSP core can also fetch the next instruction from program memory.

DM and PM are further split to two parts, one part is accessed by DSP only (named DMX-in and PM-in respectively), and another part can either be configured as accessed by DSP or accessed by DMA controller (named DMX-swap and PM-swap respectively). This gives flexibility to change program or move data when the DSP is running and is the compensation to small memory space. The DSP can change memory settings by configure memory control registers and do DMA transfer by configure the DMA control registers.

DMX-swap and PM-swap can be configured as single buffer or double buffer. When configured as double buffer, the buffer is spitted into two parts: one part is accessed by the DSP and another part is accessed by the DMA controller side. When configured as single buffer, the buffer is either accessed by the DSP or accessed by the DMA controller.

The size of DMX is 5K 16-bit WORD; PM is 5K 24-bit WORD; and DMY is 5K 16-bit WORD. The following table is the summary of the DSP memory spaces.

Table 4. DSP Memory Address Mapping

Memory	Starting Address	Size(WORD)	Word Length	Accessed By
DMX-in	0x0000 (DM)	1024	16	DSP only
DMX-swap (single)	0x0400 (DM)	4096	16	DSP/SDRAM
DMX-swap (double)	0x0400 (DM)	2048	16	DSP/SDRAM
DMY (DSP interface)	0x2800 (PM)	2048	16	DSP/SDRAM
DMY (GPS baseband)	0x3000 (PM)	3072	16	DSP/SDRAM
PM-in	0x0000 (PM)	1024	24	DSP only
PM-swap (single)	0x0400 (PM)	4096	24	DSP/SDRAM
PM-swap (double)	0x0400 (PM)	2048	24	DSP/SDRAM

3.2.2 DSP Host Interface

In Atlas™-II the DSP Host Interface is called ISA-IDMA Port. The ISA-IDMA Port provides an efficient method of communication between a host system and the DSP core. The port can be used to access the on-chip program memory, coefficient memory and data memory of the DSP with only on DSP cycle per word overhead. The ISA-IDMA port can not, however, be used to write to the DSP's memory-mapped control registers. A typical ISA-IDMA transfer process is described as follows:

The start address and destination memory type should be set before data is transferred. Destination address will be incremented automatically after each data transfer for DMX/DMY data or every two data transfer for PM data. **(Note that the DSP continues program execution during the transfer operation, except during the “stolen” cycle used to access memory.)**

```
//For PM data
Set starting address
Set memory type to PM
While (Data to be transferred is not empty)
{
```

```

    transfer the low part,
    transfer the high part, fill the high byte to zero.
}

//For DMX/DMY data
Set starting address
Set memory type to DMX/DMY
While (Data to be transfer is not empty)
{
    transfer the data
}

//rebooting
Write 1 to REBOOT address

```

The ISA-IDMA control registers are located in the DSP Shared Memory (started from 0x4800_0000 in RISC address space).

Table 5. DSP ISA-IDMA Control Registers

Names	RISC Address	Width	Description
MADDR	<i>DSP_SHARED_MEMORY+0</i>	16	Starting address to internal memory
MTYPE	<i>DSP_SHARED_MEMORY+8</i>	16	Destination memory, 0x0: PM, 0x1: DMX, 0x2: DMY, 0x3: reserved
MDATA	<i>DSP_SHARED_MEMORY+16</i>	16	For PM: One 24-bits instruction takes two cycles to transfer. At the first cycle, transfer the lower bytes of code [15:0], at the second cycle transfer the high byte of code {8'h0, [23:16]}. For DMX and DMY: transfer 16-bit data in one cycle
REBOOT	<i>DSP_SHARED_MEMORY+24</i>	16	A write will reboot the processor core and force DSP starting execution at PM (0x0). (Note: BDMA and ISA part won't be rebooted)

3.2.3 DSP IO Interface

It's different with Atlas™-I that the I/O space no longer occupies the overlaid DMX space. Instead, it uses the independent IO/BDMA port.

The DSP core supports an additional external memory space called I/O space. This space is designed to support simple connections to peripherals (such as data converters and external registers) or to bus interface ASIC data registers. I/O space supports 2048 locations of 16-bit-wide data. The lower eleven bits of the external address bus are used; the upper three bits are undefined. The I/O space also has four dedicated four-bit wait-state registers, which specify up to 15 wait-states to be automatically generated for each of four regions.

The Block memory DMA (BDMA) transfer allows loading and storing of program instructions and data. The BDMA circuit is able to access the external memory space while the processor is operating normally.

In Atlas™-II DSP subsystem there are two types of peripheral devices that can be accessed by DSP core. One is slow device that can run at slower clock frequency than DSP core; the other is fast device that always run at the same clock frequency as the DSP core. There are many slow devices that can be

accessed by DSP core, but fast device is only GPS, which occupies the DMX memory space (0x2000~0x2FFF).

Either I/O or BDMA can be used to access slow devices, but only the 16bit mode is supported. The following table shows the IO space mapping of the DSP in Atlas™-II. Every slow device occupies 128 16-bit WORD I/O space.

Table 6. DSP IO Device Address Mapping

DSP Address	Address space	Device Mapped
0x000~0x07F	IO	Interrupt controller
0x080~0x0FF	IO	Video Input Port
0x100~0x17F	IO	Audio CODEC
0x180~0x1FF	IO	DMA
0x200~0x27F	IO	GPIO
0x280~0x2FF	IO	USP0
0x300~0x37F	IO	USP1
0x380~0x3FF	IO	USP2
0x400~0x47F	IO	USP3
0x480~0x4FF	IO	USP4
0x500~0x57F	IO	USP5
0x580~0x3FF	IO	PCI
0x600~0x67F	IO	Reserved
0x680~0x6FF	IO	Reserved
0x700~0x77F	IO	Reserved
0x780~0x7FF	IO	Reserved
0x2000~0x2FFF	DM	GPS Control Registers
0x2800~0x2FFF	PM	DSP Interface
0x3000~0x3BFF	PM	GPS RF FIFO

3.3 DSP Subsystem Registers

Table 7. DSP Subsystem Register Mapping

RISC Address <11:0>	DSP DMX Address <12:0>	Register	Description
0x0000	-	DSP_REG_MODE	DSP register mode
0x0004	0x1C01	DSP_DMA_MODE	DSP DMA mode
0x0008	0x1C02~0x1C03	DSP_DMX_START_ADD	DSP DMA DMX start address
0x000C	0x1C04~0x1C05	DSP_PM_START_ADD	DSP DMA PM start address
0x0010	0x1C06~0x1C07	DSP_DMA_LENGTH	DSP DMA length
0x0014	0x1C08~0x1C09	DSP_MEM_START_ADD	DSP SDRAM start address
0x0018	0x1C0A~0x1C0B	DSP_DMA_PITCH	DSP DMA pitch
0x001C	0x1C0C	DSP_BYTE_MODE	DSP DMA byte mode
0x0020	0x1C00	DSP_MEM_MODE	DSP memory mode
0x0024	0x1C0E~0x1C0F	DSP_DMY_START_ADD	DSP DMA DMY start address
0x0028~0x40	-	-	Reserved
0x0044	0x1C41	DSP_FSM_RST	DSP FSM reset
0x0048	0x1C42	DSP_DIV_CLK	DSP/System Clock ratio
0x004c	0x1C43	DSP_IO_MODE	DSP IO Mode

0x0050	0x1C44	DSP_BDMA_SA	DSP BDMA Start Address
0x0054~0x5C			Reserved
0x0060	0x1C60	DSP_INT_FROM_RISC	DSP interrupt from RISC
0x0064	0x1C61	DSP_INT_TO_RISC	DSP interrupt to RISC
0x006C~0x7C	-	-	Reserved
0x0080	0x1C80~0x1C81	DSP_GEN_REG0	DSP general register 0
0x0084	0x1C82~0x1C83	DSP_GEN_REG1	DSP general register 1
0x0088	0x1C84~0x1C85	DSP_GEN_REG2	DSP general register 2
0x008C	0x1C86~0x1C87	DSP_GEN_REG3	DSP general register 3
Others		-	Reserved

- **DSP Register Mode Register (DSP_REG_MODE) – RISC: 0x0**

Bit	Name	Default	Description
0 (R/W)	RISC_DSP	1'b0	0: DSP take the DMA control 1: RISC take the DMA control
15:1	-	15'h0	Reserved
16 (R/W)	STANDBY	1'b0	DSP core standby control 1: put DSP core standby. 0: put DSP in normal mode
31:17	-	15'h0	Reserved

- **DSP DMA Mode Register (DSP_DMA_MODE) – RISC: 0x4, DSP: 0x1C01**

Bit	Name	Default	Description
0 (R/W)	STATUS	1'b0	0: DMA done 1: DMA busy Write 1 to this register will start the DMA automatically
2:1 (R/W)	TYPE	2'h0	00: No DMA 01: DMA for PM 10: DMA for DMX 11: DMA for DMY
3 (R/W)	DIR	1'b0	0: DMA from SDRAM to SRAM 1: DMA from SRAM to SDRAM
15:4	-	12'h0	Reserved
16 (R/W)	ENDIAN	1'b1	DSP Endian value.
17 (R/W)	SRAM_OE	1'b0	SRAM OE value.
31:18	-	14'h0	Reserved

- **DSP DMX Start Address Register (DSP_DMX_START_ADD) – RISC: 0x8, DSP: 0x1C02~0x1C03**

Bit	Name	Default	Description
31:0 (R/W)	START_ADD	32'h0	DSP DMA DMX start address

- **DSP PM Start Address Register (DSP_PM_START_ADD) – RISC: 0xC, DSP: 0x1C04~0x1C05**

Bit	Name	Default	Description
-----	------	---------	-------------

31:0 (R/W)	START_ADD	32'h0	DSP DMA PM start address.
---------------	------------------	-------	---------------------------

- **DSP DMA Length Register (DSP_DMA_LENGTH) – RISC: 0x10, DSP: 0x1C06~0x1C07**

Bit	Name	Default	Description
15:0 (R/W)	X_LEN	16'h0	DSP DMA X Length.
31:16 (R/W)	Y_LEN	16'h0	DSP DMA Y Length.

- **DSP Memory Start Address Register (DSP_MEM_START_ADD) – RISC: 0x14, DSP: 0x1C08~0x1C09**

Bit	Name	Default	Description
31:0 (R/W)	START_ADD	32'h0	DSP DMA SDRAM start address

- **DSP DMA Pitch Register (DSP_DMA_PITCH) – RISC: 0x18, DSP: 0x1C0A~0x1C0B**

Bit	Name	Default	Description
31:0 (R/W)	PITCH	32'h0	2-D DMA Pitch, i.e. the length between the end of last line to the beginning of next line. (In DWORD unit)

- **DSP DMA Byte Mode Register (DSP_BYTE_MODE) – RISC: 0x1C, DSP: 0x1C0C**

Bit	Name	Default	Description
0 (R/W)	BYTE_MODE	1'b0	1: Byte mode enabled. 0: Not enabled
1 (R/W)	DUMMY_WRITE	1'b0	1: the first DWORD write to DMX is a dummy write 0: The first DWORD write to DMX is not a dummy write
31:2	-	30'h0	Reserved

- **DSP DMA Memory Mode Register (DSP_MEM_MODE) – RISC: 0x20, DSP: 0x1C00**

Bit	Name	Default	Description
0 (R/W)	PM_MODE	1'b0	0: PM is organized as single buffer 1: PM is organized as double buffer
1 (R/W)	PM_SWAP	1'b0	If single buffer: 0: DSP takes it 1: System bus takes it. If double buffer 0: PM is organized as double buffer mode 1 (DSP takes buffer 2, System bus takes buffer 3) 1: PM is organized as double buffer mode 2 (DSP takes buffer 3, System bus takes buffer 2)
2 (R/W)	DMX_MODE	1'b0	0: DMX is organized as single buffer 1: DMX is organized as double buffer
3 (R/W)	DMX_SWAP	1'b0	If single buffer: 0: DSP takes it 1: System bus takes it. If double buffer: 0: DMX is organized as double buffer mode 1 (DSP

			takes buffer 2, System bus takes buffer 3) 1: DMX is organized as double buffer mode 2 (DSP takes buffer 3, System bus takes buffer 2)
4 (R/W)	DMA_ENDIAN	1'b0	0: Inverted endian 1: Normal endian
6:5 (R/W)	DMY_MODE	2'b0	00: DSP takes both DMY SRAM and GPS SRAM 01: System bus takes DMY SRAM, DSP takes the GPS SRAM 10: DSP takes DMY SRAM, system bus takes GPS SRAM 11: System bus takes both DMY and GPS SRAM.
31:7	-	25'h0	Reserved

- **DSP DMY Start Address Register (DSP_DMY_START_ADD) – RISC: 0x24, DSP: 0x1C0E~0x1C0F**

Bit	Name	Default	Description
31:0 (R/W)	START_ADD	32'h0	DSP DMA DMY start address

- **DSP FSM Reset Register (DSP_FSM_RST) – RISC: 0x44, DSP: 0x1C41**

Bit	Name	Default	Description
0 (R/W)	IC_FSM_RST	1'b1	Force reset IDMA Main Control State Machine
1		1'b0	Reserved
2 (R/W)	DMA_FSM_RST	1'b1	Force reset System Bus DMA State Machine
3 (R/W)	SRAM_FSM_RST	1'b1	Force reset SRAM Interface State Machine
31:4	-	28'h0	Reserved

- **DSP/System Clock Ratio Register(DSP_DIV_CLK) – RISC: 0x48, DSP: 0x1C42**

Bit	Name	Default	Description
1:0 (R/W)	CLK_RATIO	2'b0	0: DSP/System clock ratio is 1 1: DSP/System clock ratio is 2 2: DSP/System clock ratio is 4 3: DSP/System clock ratio is 8
31:2	-	30'h0	Reserved

- **DSP IO Mode Register(DSP_IO_MODE) – RISC: 0x4C, DSP: 0x1C43**

Bit	Name	Default	Description
0(R/W)	IO_BDMA	1'b0	0: Normal IO Mode or BDMA Mode 1: Fixed Address BDMA Mode
31:1	-	31'h0	Reserved

- **DSP BDMA Start Address(DSP_BDMA_SA) – RISC: 0x50, DSP: 0x1C44**

Bit	Name	Default	Description
11:0 (R/W)	BDMA_SA	12'b0	Start Address of Fixed Address BDMA Mode
31:12	-	20'b0	

R			
---	--	--	--

- **DSP Interrupt from RISC Register (DSP_INT_FROM_RISC) – RISC: 0x60, DSP: 0x1C60**

Bit	Name	Default	Description
14:0 (R/W)	MSG_TO_DSP	15'h0	Message bit to DSP with the interrupt operation. RISC writes to this register will interrupt the DSP.
15	INT_DSP	1'h0	The interrupt status bit. DSP writes 1 to this bit will clear the interrupt.
31:16	-	16'h0	Reserved

- **DSP Interrupt to RISC Register (DSP_INT_TO_RISC) – RISC: 0x64, DSP: 0x1C61**

Bit	Name	Default	Description
0 (R/W)	INT_RISC	1'b0	The interrupt status bit. RISC writes 1 to this bit will clear the interrupt.
15:1	-	15'h0	Reserved
31:16 (R)	MSG_TO_RISC	16'h0	Message bit to RISC with the interrupt operation. DSP writes to this register will interrupt the RISC.

- **DSP General Data Register (DSP_GEN_REG0) – RISC: 0x80, DSP: 0x1C80-0x1C81**

Bit	Name	Default	Description
15:0 (R/W)	DATA_L	16'h0	The general registers used to exchange the information between RISC and DSP. RISC had the priority to write this register
31:16 (R/W)	DATA_H	16'h0	Same as above.

- **DSP General Data Register (DSP_GEN_REG1) – RISC: 0x84, DSP: 0x1C82-0x1C83**

Bit	Name	Default	Description
15:0 (R/W)	DATA_L	16'h0	The general registers used to exchange the information between RISC and DSP. RISC had the priority to write this register
31:16 (R/W)	DATA_H	16'h0	Same as above.

- **DSP General Data Register (DSP_GEN_REG2) – RISC: 0x88, DSP: 0x1C84-0x1C85**

Bit	Name	Default	Description
15:0 (R/W)	DATA_L	16'h0	The general registers used to exchange the information between RISC and DSP. RISC had the priority to write this register
31:16 (R/W)	DATA_H	16'h0	Same as above.

- **DSP General Data Register (DSP_GEN_REG3) – RISC: 0x8C, DSP: 0x1C86-0x1C87**

Bit	Name	Default	Description
15:0	DATA_L	16'h0	The general registers used to exchange the

(R/W)			information between RISC and DSP. RISC had the priority to write this register
31:16 (R/W)	DATA_H	16'h0	Same as above.

4 System Memory Interface

4.1 Overview

The Atlas™-II implements a high-speed internal system memory bus that can run up to 200MHz. There are 5 bus masters on this bus: RISC Subsystem, DSP Subsystem, Graphic Display, PCI Bridge, and the IO Bridge. And there are 2 bus slaves (target): the DRAM Controller and/or SRAM Controller¹. Each bus master can choose its bus slave (target) between the DRAM Controller and SRAM Controller separately. There is a System Arbiter to arbitrate the memory access of all 5 bus masters.

In most cases when users want to achieve high performance and high bandwidth, then the DRAM Controller should be used as the bus slave. In some other cases when users want to achieve low power and reasonable bandwidth, then the SRAM Controller may be used as the bus slave.

4.2 System Arbiter

4.2.1 Overview

As specified above, there are a total of 5 bus masters on the system memory bus. The System Arbiter arbitrates the requests from the 5 bus masters and passes the appropriate accesses to the active bus slave (either DRAM Controller or SRAM Controller).

Table 8. System Bus Master Allocation

Bus Masters	Block Name
System Bus Master0	RISC
System Bus Master1	DSP
System Bus Master2	I/O Bridge
System Bus Master3	Graphic
System Bus Master4	PCI Bridge

When the bus master wants to access the memory bus, it must own the bus first. To own the bus, the master must assert the request signal; the System Arbiter can then grant the bus to the master with the grant signal.

4.3 DRAM Controller

4.3.1.1 Overview

The Atlas™-II DRAM Controller supports the following configurations:

- 200MHz DDR-SDRAM at 2.5V
- 138MHz SDRAM at 3.3V

The DRAM Controller supports both 16-bit and 32-bit data bus. In 16-bit mode, two consecutive pieces of data, each 16-bits wide, is fetched by the memory interface and returned to system memory as a 32-bit wide data. The memory mapping from DRAM to system memory is done automatically through hardware logic.

¹ The SRAM Controller is in PCI Subsystem. Please refer to the corresponding section in "PCI Subsystem" for more details.

The following tables show some examples of DDR/SDRAM configurations:

Table 9. DDR/SDRAM Configurations for a 16-bit Data Bus

DDR/SDRAM Type	DDR/SDRAM count	Total Size (byte)
1Mx16	1/2	2M/4M
8Mx16	1/2	16M/32M
16Mx16	1/2	32M/64M
32Mx16	1/2	64M/128M

Table 10. DDR/SDRAM Configurations of 32-bit Data Bus

DDR/SDRAM type	DDR/SDRAM count	Total Size (byte)
1Mx16	2/4	4M/8M
8Mx16	2/4	32M/64M
16Mx16	2/4	64M/128M
32Mx16	2/4	128M/256M

The DRAM Controller IO pins use the multi-voltage/multi-functional programmable IO. This kind of IO is special designed so that it can be configured to different modes as following:

- 1.8V LVCMOS mode
- 2.5V SSTL mode
- 2.5V LVCMOS mode
- 3.3V LVCMOS mode

For details about how to configure these modes, please refer to the section of “Mode Configuration Pins”.

4.3.2 DDR Connection Examples

The following figures show the examples of configuration of 16-bit and 32-bit memory interface, with 16Mx16 DDR or SDRAM.

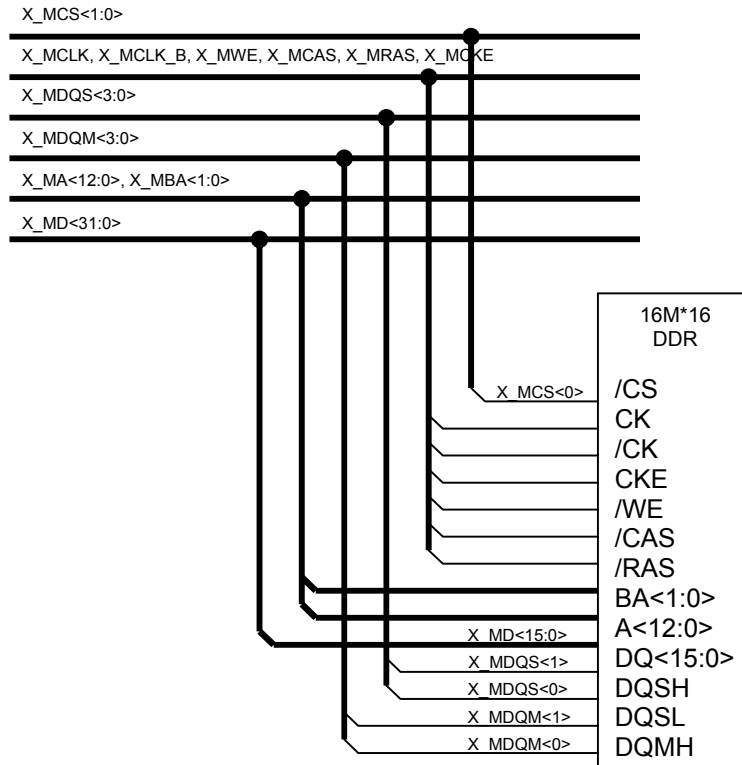


Figure 5. DDR Connection in 16-bit Mode

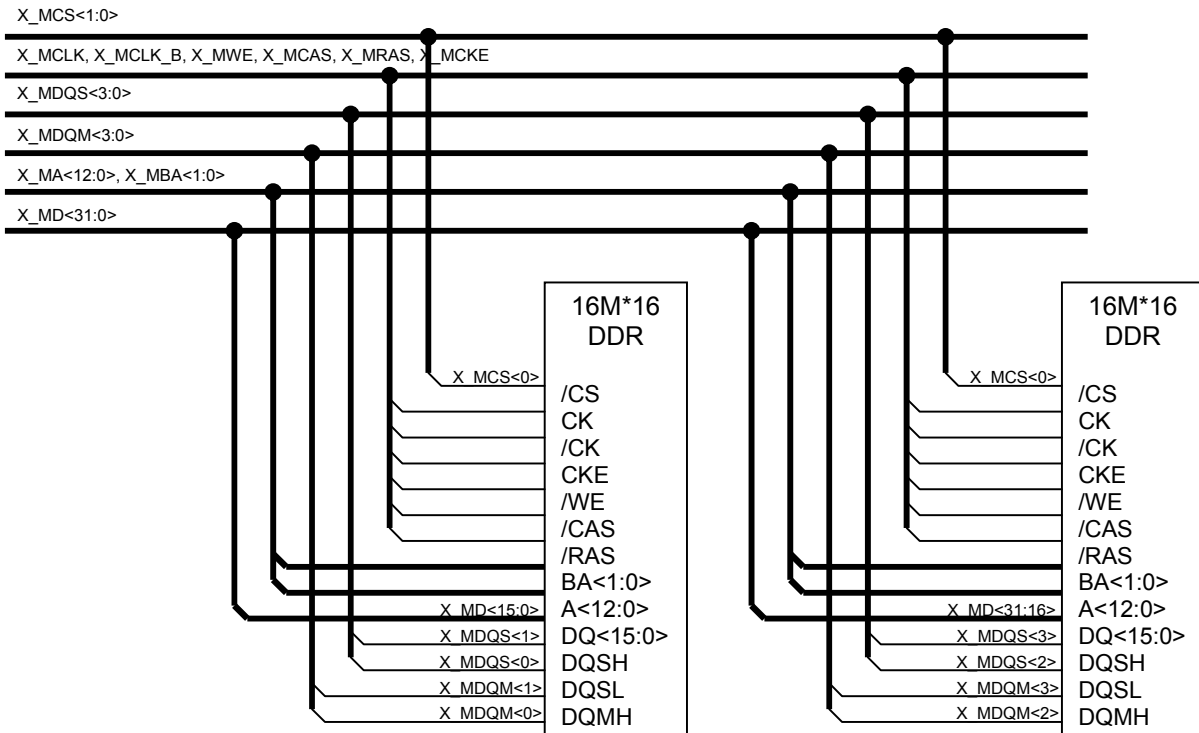


Figure 6. DDR Connection in 32-bit Mode

4.3.3 SDRAM Connection Examples

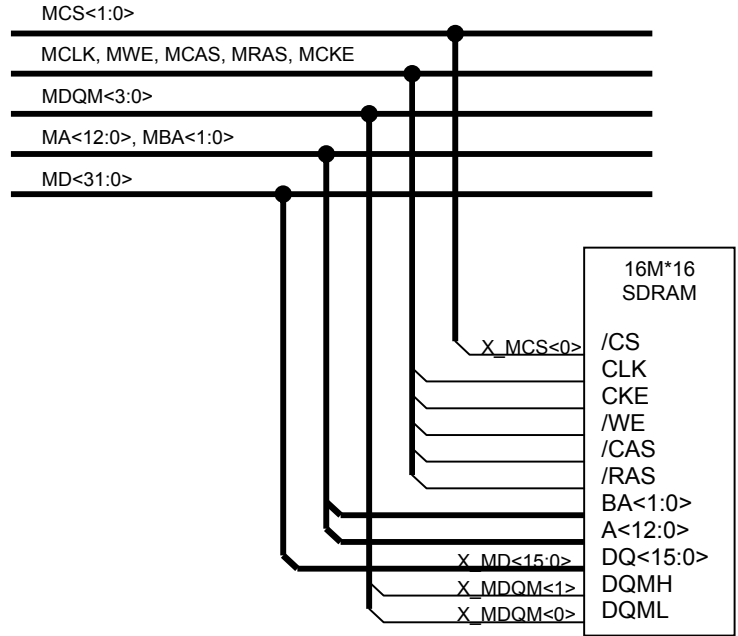


Figure 7. SDRAM Interface in 16-bit Mode

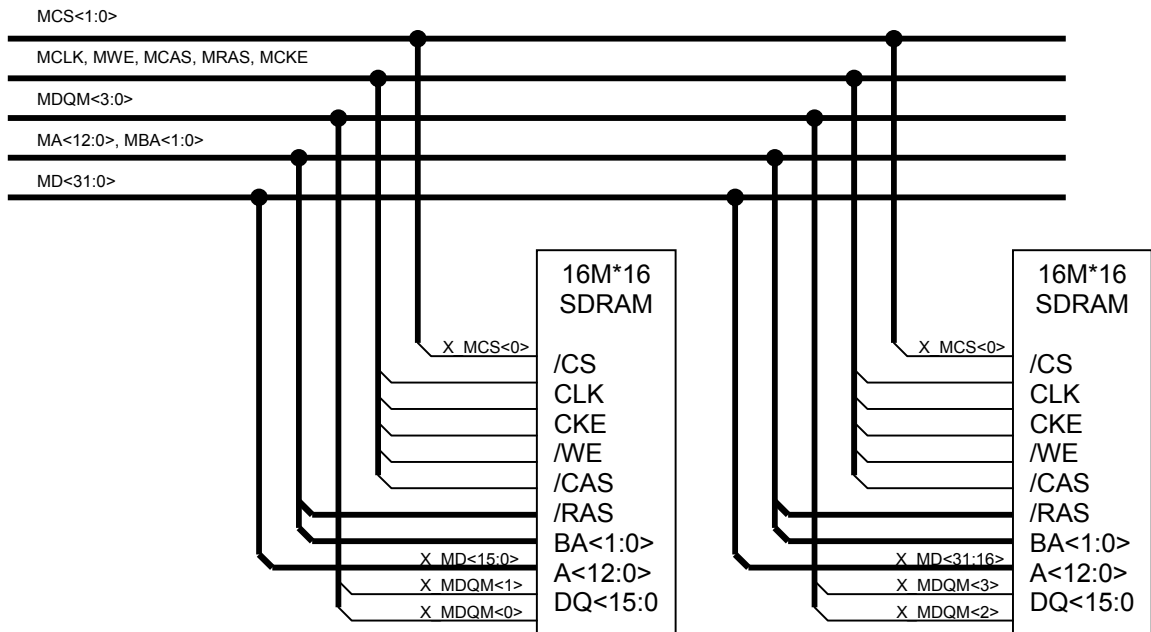


Figure 8. SDRAM Interface in 32-bit Mode

4.3.4 DRAM Controller Registers

There are 6 registers in System Memory Controller. 4 of 6 registers are only used during the memory initialization.

NOTE: After the memory is initialized, the register content can not be changed.

The DRAM needs a series of initialization steps before the DRAM is ready to be accessed. One must program the first 5 registers (offset: 0x0~0x10) in order. Then 200 us after programming MEM_EXTEND (0x10) register, one can program MEM_START (0x14) to start the DRAM initialization bus sequence. Another 200 us settling time is required by DRAM to stabilize its internal clocks. When DRAM is ready, the bit 0 of MEM_START will be set to '1'. One can read this bit to check whether DRAM is ready.

Table 11. RISC Interface Register Address Mapping

RISC Address <11:0>	Register	Description
0x0000	MEM_TYPE	Memory type and configuration register
0x0004	MEM_TIMING	Memory timing register
0x0008	MEM_POWER	Memory power management
0x000C	MEM_MODE	DRAM mode register
0x0010	MEM_EXTEND	DRAM extended mode register
0x0014	MEM_START	DRAM initialization register
Others	-	Reserved

- **Memory Type and Configuration Register (MEM_TYPE) – RISC: 0x0**

Bit	Name	Default	Description
7:0 (R/W)	DRAM_TYPE	8'h01	00000001 -> 1Mx16 00000010 -> 8Mx8 00000100 -> 16Mx8 00001000 -> 32Mx8 00010000 -> 64Mx8 00100000 -> 8Mx16 01000000 -> 16Mx16 10000000 -> 32Mx16 others: reserved
8 (R/W)	DRAM_MODULE	1'b0	0 -> 1 module 1 -> 2 module
9 (R/W)	EN_32BIT	1'b1	0 -> 16 bit data wide 1 -> 32 bit data wide
10 (R/W)	EN_LARGE_PAGE	1'b1	0 -> Disable large page (BA map to highest address bits) 1 -> Enable large page (BA map to the middle of Row and Column address bits)
11 (R/W)	EN_DDR	1'b0	0 -> SDRAM 1 -> DDR
12	EN_MOBILE	1'b0	0 -> standard DRAM 1 -> mobile DRAM

- **Memory Timing Register (MEM_TIMING) – RISC: 0x4**

Bit	Name	Default	Description
1:0 (R/w)	REFRESH_MODE	2'b01	00 -> one row refresh per refresh request 01 -> two row refresh per refresh request 10-> three row refresh per refresh request 11 -> four row refresh per refresh request
12:2 (R/W)	REFRESH_TIME	11'h60	192 clocks of CLK_12M for default. The base timing unit for refresh period is based on 12MHz clock.
13 (R/W)	SF_EN	0	0 -> disable self refresh 1 -> enable self refresh
17:14 (R/W)	T_SREX	4'h7	Number of memory clock is required to return from self-refresh to normal refresh. Default is 7 memory clocks. For DDR this is the interval from CKE rising high to beginning DLL reset delay counter
21:18 (R/W)	T_RAS	4'h7	T_RAS is number of memory clocks required for clock being stable before self refresh exit. Default is 7 memory clocks.
25:22 (R/W)	T_RFC	4'h7	T_RFC is the number of memory clocks required for refresh cycle. Default is 7 memory clocks.
27:26 (R/W)	T_RP	2'b11	T_RP is the number of memory clocks required for row precharge time (interval between Precharge to RAS). Default is 3 memory clocks.
29:28 (R/W)	T_RCD	2'b11	T_RCD is the number of memory clocks required for CAS after RAS (interval between Precharge to RAS). Default is 3 memory clocks.
31:30	-	2'b00	Reserved

This register can be change during DRAM operation if the memory clock period is changed. Notice that one can only program these timing parameters when DRAM is idle.

- **Memory Power Management Register (MEM_POWER) – RISC: 0x8**

Bit	Name	Default	Description
10:0 (R/W)	-	11'h0	Reserved
11 (R/W)	D_PDN_EN	1'b0	0 -> disable dynamic power down 1 -> enable dynamic power down
17:12 (R/W)	T_PDN	6'h0	T_PDN is the number of system idle clocks. When the system is idle more than T_PDN clocks, then DRAM will enter low power down mode. CKE to DRAM will be asserted to low.
18 (R)	PDN_STATUS	1'b0	1 -> DRAM is in power down
19 (R)	SF_STATUS	1'b0	1 -> DRAM is in self-refresh
20 (R)	DRAM_RDY	1'b0	1 -> DRAM is in ready to be accessed.
31:18	-	14'h0	Reserved

NOTE: If user tries to write the read only bits (PDN_STATUS, SF_STATUS, and DRAM_RDY), there might be unexpected results.

- **Memory Mode Register (MEM_MODE) – RISC: 0xC**

Please refer user manual of DRAM from DRAM vendor.

- **Memory Extend Register (MEM_EXTEND) – RISC: 0x10**

Please refer user manual of DRAM from DRAM vendor

- **Memory Initialization Register (MEM_START) – RISC: 0x14**

Bit	Name	Default	Description
0 (R)	DRAM_RDY	0	0 -> DRAM is not ready to be accessed. 1-> DRAM is ready to be accessed.
2:1 (R/W)	DRAM_INIT	00	11 -> Start DRAM initialization bus sequence if DRAM_RDY is 0. If DRAM_RDY is 1, the initialization is locked. No programming can change DRAM internal registers. Others -> no action.
3 (R/W)	SOFT_RESET	0	Write 1 to reset DRAM controller Write 0 to clear the reset

Notes: DRAM clock will be controlled by GPIO2_CTRL19 register. On reset the DRAM clock is always enabled, user can choose to disable DRAM clock output by software programming. See DRAM application notes for detail description.

5 System Control Modules

5.1 Mode Configuration Pins

5.1.1 Overview

There are several mode configuration pins that control the different modes of Atlas™-II. These pins are:

Table 12. Mode Configuration Pins

Pin Name	Function Name	Description
X_FA<14:13>	PAD_MODE<1:0>	Configure the SSTL IO mode: 2'b00: 1.8V LVCMOS 2'b01: 2.5V SSTL 2'b10: 2.5V LVCMOS 2'b11: 3.3V LVCMOS
X_FA<15>	NAND_MODE	Configure the NAND Flash mode: 1'b0: Disable the support of 2K page 1'b1: Enable the support of 2K page
X_FA<16>	NAND_BOOT	Configure the Boot mode: 1'b0: Boot from ROM/NOR Flash 1'b1: Boot from NAND Flash
X_FA<17>	BOOT_IS16	Configure Boot Flash bit width: 1'b0: 8-bit wide 1'b1: 16-bit wide
X_FA<19:18>	TEST_MODE<1:0>	Configure Test Mode: 2'b00: Normal mode 2'b01: ATPG mode 2'b10: JTAG mode 2'b11: BIST mode
X_FA<21:20>	JTAG_MODE<1:0>	Configure JTAG Mode (see following table for details)
X_DF_RY_BY	NAND_SEL	Configure NAND Flash pin sharing mode: 1'b0: Pin share with Video Input Port 1'b1: Pin share with ROM I/F

NOTE: Please also refer to the section of Resource Sharing Controller for more details about the NAND Flash pin multiplex with Video Input Port and ROM I/F.

The following table shows the decoding of JTAG_MODE<1:0>:

Table 13. Decoding of JTAG_MODE<1:0>

Test Mode	JTAG_MODE<1:0>	Description
ATPG mode	2'b00	ARM926/ETM9 INTEST ¹ mode
	2'b01	ATPG mode
	2'b1X	Reserved
Normal mode	2'b01	RISC JTAG mode

5.1.2 Configuration Setup

¹ Please refer to ARM926 Reference Manual.

The mode configuration pins have no on-chip pull-up/down resistors. So when Atlas™-II is powered up, there is no default value. User needs to use external pull-up/down resistors to set up the configurations.

Because these mode configuration pins are shared with normal functional pins, the configuration process has to be completed before Atlas™-II enters the normal functional mode. This is done by an internal strobing logic which latches the configuration valued on the rising edge of the power on reset. So the functional toggling after the reset is finished will not change the configuration mode anymore.

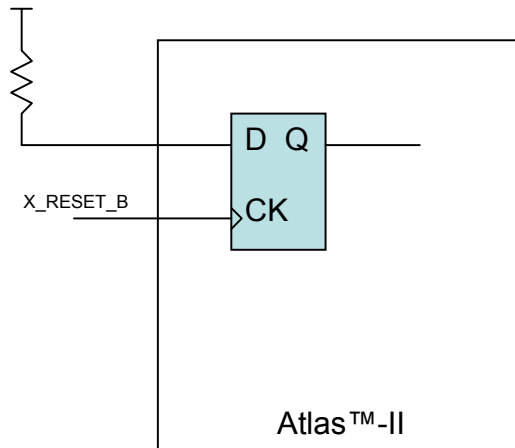


Figure 9. An Example of Mode Configuration Setup

5.2 Clock and PLL

5.2.3 Overview

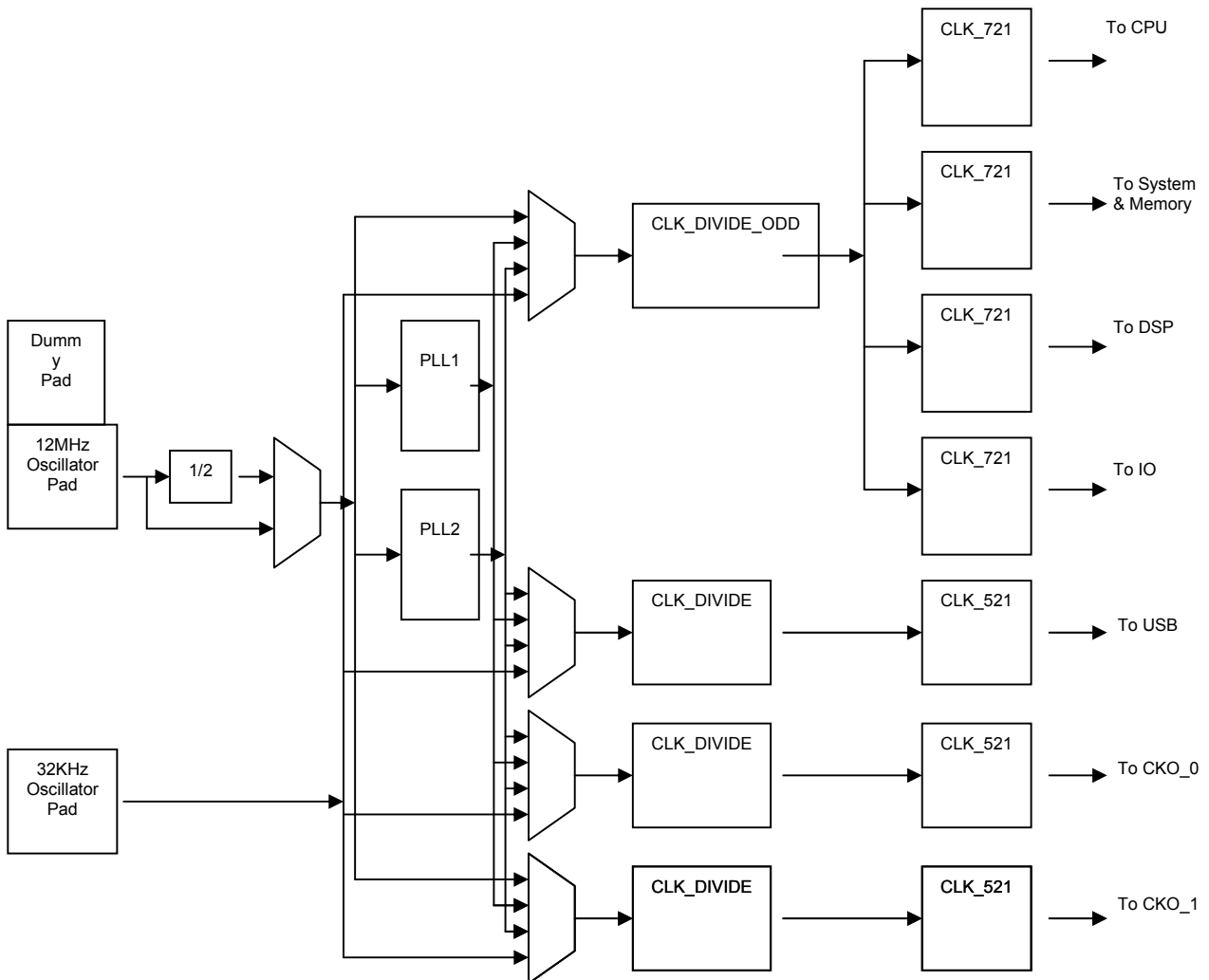


Figure 10. Atlas-2 Clock Circuit Block Diagram

The above figure shows the clock circuit architecture in Atlas-II™. The whole circuit can be divided into 3 parts:

- Oscillator
- PLL
- Multiplexers & Dividers

These stages will be explained in detail in the following sections.

5.2.4 Oscillator

There are two oscillator pads in Atlas-2: one is a high-speed oscillator pad (TSMC: PDXOE3DG); the other is a real-time oscillator pad (TSMC: PDXOE4DG).

The PDXOE3DG is designed to oscillate for crystal samples from 2MHz to 30MHz. For applications out of 2~30MHz range, please contact with Centrality IC Design Group for further suggestions.

The PDXOE4DG is a low power 32.768 KHz crystal oscillator.

To ensure the oscillation start up, the tank circuit must provide a negative resistance (-Re) at least **5 times** greater than the equivalent series resistance (ESR) of the crystal sample. The greater the negative resistance provided, the faster the crystal starts up.

To select the proper crystal is crucial to make the oscillator work stably. The key parameters of the crystal lie in CL (load capacitance) and the maximum ESR at the target frequency. Reducing the CL can help increase the negative resistance of the tank circuit, but if CL is too small, the deviation from the target frequency increases because of the growing percentage of the capacitance variation. Therefore, there is a trade-off between the short start-up time and the small frequency deviation in deciding the CL value.

The smaller ESR (with a higher price) also helps to reduce the start-up time. Please refer to the following table for the CL & ESR selection guidance. This table is for reference only and applicable for typical conditions.

Table 14. Selection Guide of Crystal for PDXOE3DG

Target Freq. (Hz)	2M~6M	6M~10M	10M~20M	20M~30M
CL (pF)	16	12	8	6
Max. ESR (Ohm)	100	50	40	20

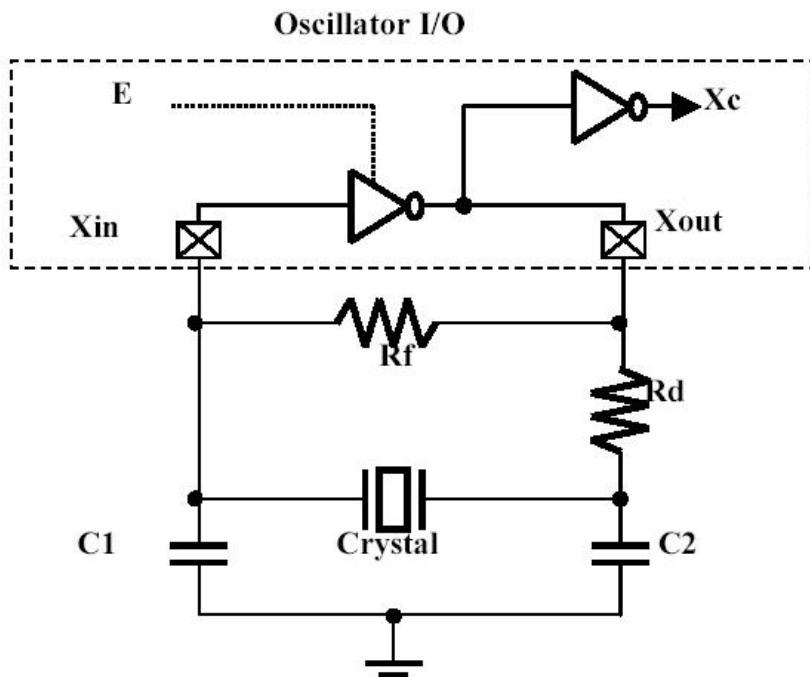


Figure 11. Oscillator Tank Circuit

The above figure shows a reference tank circuit for a crystal oscillating in the fundamental mode. It shows several components that decide the behavior of the oscillation; and they affect the loop in many aspects:

- **Rf** represents the feedback resistor to bias the inverter in the high gain region. Rf cannot be too low, or the loop may fail to oscillate. Normally, and Rf of 1Mohm is sufficient for MHz band applications.
- **Rd** represents the damping resistor that helps increase stability, save power, and suppresses the gain in the high frequency area. The trade-off for inserting Rd is the reduction of negative resistance. Therefore, Rd cannot be too large, or the loop could fail to oscillate. Sometimes users may drop Rd in high frequency applications to reduce the production costs.
- **C1** and **C2** are decided according to the crystal CL specification. In the steady state of oscillating, CL is defined as the $(C1 \times C2) / (C1 + C2)$. In fact, the I/O ports, the bond pad, and package pin all contribute the parasitic capacitance to C1 and C2. Therefore, we can rewrite CL to be $(C1 \times C2^*) / (C1^* + C2^*)$, where $C1^* = (C1 + C_{in, stray})$ and $C2^* = (C2 + C_{out, stray})$. In this example, the required C1 and C2 would be reduced.

This tank circuit is for parallel resonance but not for series resonance. Since C1, C2, Rd, and Rf vary with the crystal specifications, there is no single set of component specifications for all applications. For reference values, please contact the Centrality IC Design Group with your crystal specifications.

For the PDXOE3DG oscillator, there are two extra design considerations:

1. There is a divide-by-2 circuit on the output of PDXOE3DG. By default this divide-by-2 circuit is disabled. If user wants to enable it, please contact the Centrality IC Design Group first.
2. There is a dummy pad sitting besides of PDXOE3DG. This is for test purpose only: on ATE tester, PDXOE3DG will not allow a high-speed clock to feed in through it. So if we want to do high-speed test on ATE tester, we need to use a dummy IO pad¹. This dummy XIN is enabled when:

TEST_MODE<1:0>	JTAG_MODE<1:0>	Dummy XIN
2'b00	2'bX1	Enabled
2'b00	2'bX0	Disabled ²

5.2.5 PLL

There are two different PLL's in Atlas-2: one is 1GHz PLL (TSMC: PG13A1G3); the other is 400MHz PLL (TSMC: PG13E3G). Both PLL's use the same reference clock input (from PDXOE3DG).

5.2.5.1 PG13A1G3 1GHz PLL

¹ The dummy IO pad is muxed with X_SCLK_3.

² By default the Dummy XIN is disabled. If user wants to enable it, please contact the Centrality IC Design Group.

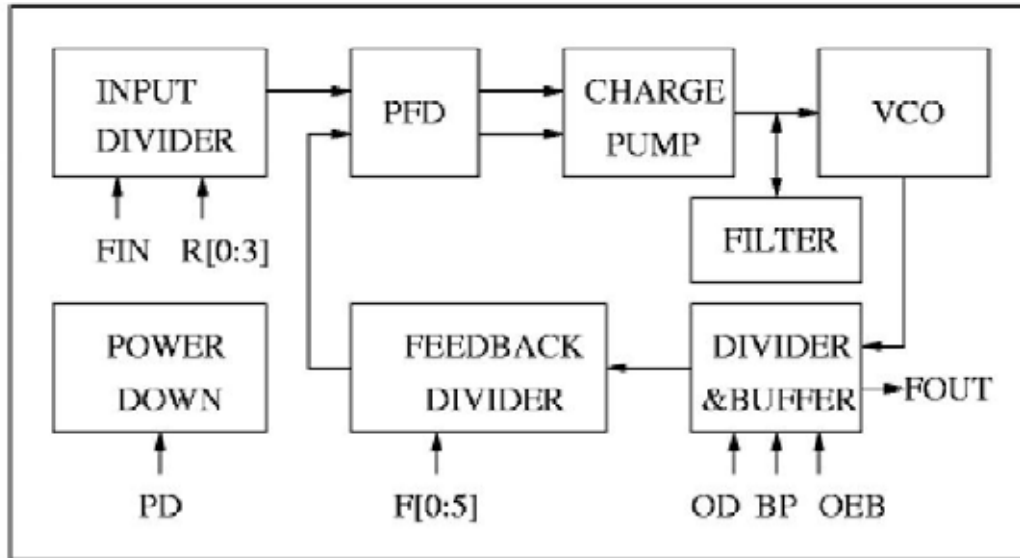


Figure 12. PG13A1G3 Block Diagram

PG13A1G3 is designed to output a clock frequency range between 250MHz~1GHz¹. There are 3 operation modes in PG13A1G3: **Normal mode**, **Power Down mode**, and **Bypass mode**. **Normal mode** synthesizes suitable F_{OUT} value by programming divider values. **Power Down mode** forces PLL in low power consuming state². **Bypass mode** provides F_{OUT} with the same frequency as F_{IN}.

It needs a **Tready**³ time (Pull_in Time + Locking Time) to relock the F_{IN} clock when PLL wakes up from Power Down mode to Normal mode. In general, it should be reserved a **Tready** time for re-locking when PLL is changed to Normal mode from Power Down or Bypass mode, or when any divider setting is changed.

In Normal mode operation, it is necessary to set suitable divider values to make sure PLL functional:

- PLL Divider Value Setting

There are 3 divider values (NR, NF, NO) to set the PLL output clock frequency F_{OUT} :

- Input Divider Value **NR**
NR = R<3:0> + 1
- Feedback Divider Value **NF**
NF = (3-NO)*(F<5:0> + 2)
- Output Divider Value **NO (See Note⁴)**
NO = OD + 1

- PLL Output Frequency Setting

$$F_{OUT} = NF/NR \times F_{IN}$$

Meanwhile the following constraints must be followed:

¹ Due to the speed limitation of our clock circuit, the PG13A1G3 should not be programmed to be over 500MHz in worst case or 600MHz in best case.

² F_{OUT} is uncertain in Power Down mode.

³ Tready = 0.5ms

⁴ When OD=1, F_{OUT} would have around 50% duty cycle. So it's suggested to set OD=1 if it's possible.

- $10\text{MHz} < F_{\text{REF}} < 50\text{MHz}$, where $F_{\text{REF}} = F_{\text{IN}}^*/\text{NR}$ (**See Note¹**)
- $500\text{MHz} < F_{\text{VCO}} < 1000\text{MHz}$, where $F_{\text{VCO}} = F_{\text{IN}}^*\text{NF}*\text{NO}/\text{NR}$
- $\text{NF} \geq 6*(3 - \text{NO})$

For example, if we want to get a 252MHz output frequency clock (when using 12MHz crystal oscillator as input):

NR = 1, NF = 21, NO = 2, $F_{\text{VCO}} = 504\text{MHz}$, $F_{\text{OUT}} = 252\text{MHz}$
F<5:0> = 19 (0x13)
R<3:0> = 0
OD = 1

PG13A1G3 requires 3 pairs of power supply in two different voltages (1.2V/3.3V) as following:

- AHVDD, AHVSS: Analog Power (3.3V) & Ground
- AHVDDG, AHVSSG: Analog Power (3.3V) & Ground
- DVDD, DVSS: Digital Power (1.2V) & Ground

In system design, all power (AHVDD, AHVDDG, and DVDD) and ground (AHVSS, AHVSSG, and DVSS) should be considered as analog power. DO NOT share these power/ground with any other power/ground planes in the system.

5.2.5.2 PG13E3G 400MHz PLL

PG13E3G is designed to output a clock frequency range between 50~400MHz. The working modes are the same as PG13A1G3: **Normal mode**, **Power Down mode**, and **Bypass mode**.

¹ When F_{IN} is 12MHz, NR has to be 1.

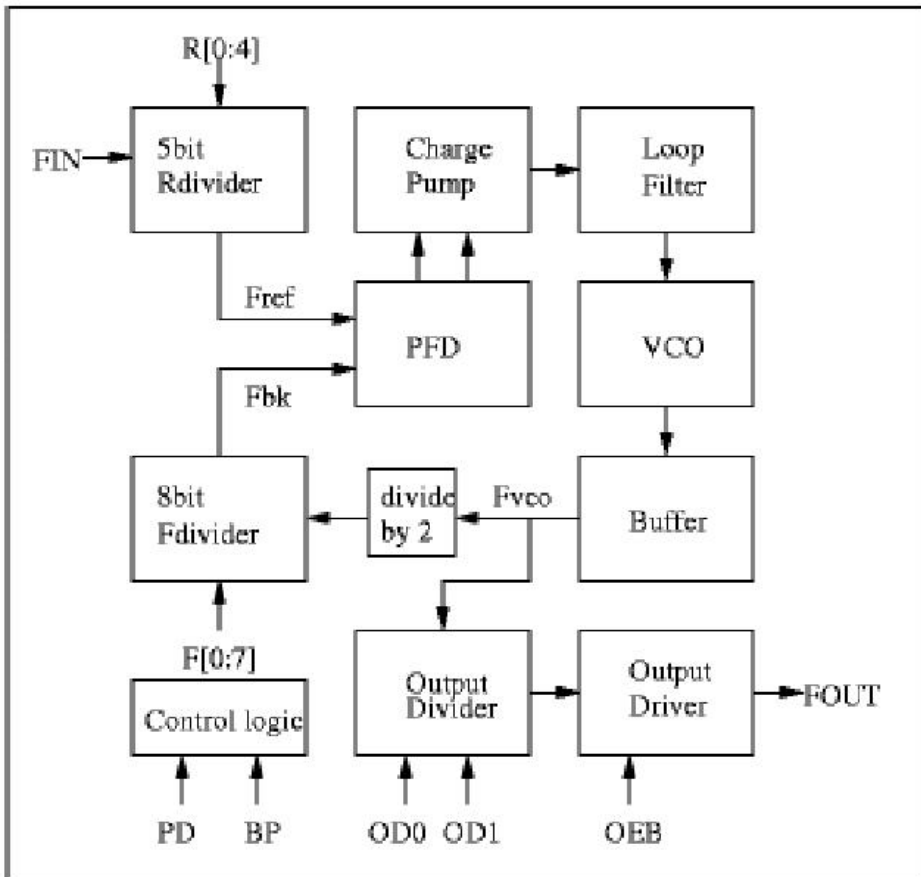


Figure 13. PG13E3G Block Diagram

In Normal mode operation, it is necessary to set suitable divider settings to make sure PLL functional:

- PLL Divider Value Setting

There are 3 divider values (NR, NF, NO) to set the PLL output clock frequency F_{OUT} :

- Input Divider Value **NR**
NR = R<3:0>
- Feedback Divider Value **NF**
NF = 2*F<7:0>
- Output Divider Value **NO (See Note¹)**

OD<1:0>	NO
2'b00	Not allowed
2'b01	1
2'b10	2
2'b11	4

- PLL Output Frequency Setting

$$F_{OUT} = NF/(NR*NO) \times F_{IN}$$

¹ When OD=2 or 3, FOUT would have around 50% duty cycle. So it's suggested to set OD=2 or 3 if it's possible.

Meanwhile the following constraints must be followed:

- $2\text{MHz} < F_{\text{REF}} < 8\text{MHz}$, where $F_{\text{REF}} = F_{\text{IN}} \cdot \text{NR}$ (**See Note¹**)
- $200\text{MHz} < F_{\text{VCO}} < 400\text{MHz}$, where $F_{\text{VCO}} = F_{\text{IN}} \cdot \text{NF} / \text{NR}$

For example, if we want to get a 192MHz output frequency clock (when using 12MHz crystal oscillator as input):

NR = 2, NF = 64, NO = 2, $F_{\text{VCO}} = 384\text{MHz}$, $F_{\text{OUT}} = 192\text{MHz}$
F<7:0> = 32 (0x20)
R<3:0> = 2
OD = 2

PG13E3G requires 2 pairs of power supply in single voltage (1.2V) as following:

- AVDD, AVSS: Analog Power & Ground
- DVDD, DVSS: Digital Power & Ground

In system design, all power (AVDD, DVDD) and ground (AVSS, DVSS) should be considered as analog power. DO NOT share these power/ground with any other power/ground planes in the system.

5.2.6 Multiplexer & Divider

There are 3 stages of multiplexer & divider:

1. 4-to-1 MUX
2. Clock Divider
3. 8-to-1 or 5-to-1 MUX

5.2.6.1 4-to-1 MUX

There are 4 of such 4-to-1 MUX's; one for each of the following clock domains: system clock, USB clock, CKO_0, and CKO_1.

Each MUX have 4 input clock sources:

- High-speed oscillator (PDXOE3DG)
- Real-time oscillator (PDXOE4DG)
- PLL1 (PG13A1G3)
- PLL2 (PG13E3G)

There is a 2-bit select signal for each of the MUX to choose one of the 4 input sources. These signals are controlled by the PWR_CLK_SWITCH register. Please refer to the Atlas-2 Developer's Manual for more details.

5.2.6.2 Clock Divider

Each 4-to-1 MUX is followed by a clock divider. The purpose of the clock divider is to divide the clock output of the 4-to-1 MUX to a lower frequency clock so that it can be used by the internal logic of Atlas-2.

There are two types of clock dividers though. First type is only for the system clock domain. And the second type is for USB, CKO_0, and CKO_1 domains. The difference is: the first type of divider can divide the input by 1, 2, 3, 4, 6, 8, 12, and 16; while the second type can only divide the input by 1, 2, 4, 8, and 16. So there are 8 clock outputs for the first type; and only 5 clock outputs for the second type.

¹ when F_{IN} is 12MHz, NR has to be 2~6.

5.2.6.3 8-to-1 & 5-to-1 MUX

Because the clock divider for system clock domain has 8 different clock outputs (1x, 1/2x, 1/3x, 1/4x, 1/6x, 1/8x, 1/12x, and 1/16x), the MUX follows it will be an 8-to-1 MUX. While the MUX follows the clock dividers in USB, CKO_0, and CKO_1 domains will be 5-to-1 MUX.

In the system clock domain, there are 4 8-to-1 MUX's in total. They are used to generate the clocks for:

- CPU
- System bus (including Memory)
- DSP
- IO

In other clock domains (USB, CKO_0, and CKO_1), there is only 1 5-to-1 MUX in each domain.

The reason to have multiple 8-to-1 MUX in system clock domain is that the clocks to different parts of internal logic (CPU, Bus/Memory, DSP, and IO) can be different, although they must be generated from a single source.

There is a register (PWR_CLK_RATIO) that controls the different ratios of those clocks in system domain. And software programmer needs to be aware of that there are certain limitations of the different configurations. All the configurations supported in Atlas-II™ are listed in the following table:

Table 15. Supported Clock Ratios in Atlas-2 (unit: MHz)

PLL #1	CPU	DSP	System Bus	Memory				IO	PWR_PLL1_C ONFIG (0x9006_0028)	PWR_CLK_RA TIO (0x9006_0040)
				DDR 32bit	DDR 16bit (1x, 2x)	SDR 32bit	SDR 16bit			
552	276	184	92			92		46	0x0000_0015	0x0XXX_C632
552	276	138	138			69		34.5	0x0000_0014	0x0XXX_8442
528	264	132	132			132		66	0x0000_0014	0x0XXX_8442
504	252	126	126			126		63	0x0000_0013	0x0XXX_8442
396	198	198	198	198		-		99	0x0000_041F	0x0XXX_4222
252	252	126	126			126		63	0x0000_0413	0x0XXX_4221
252	126	126	126			126		63	0x0000_0413	0x0XXX_4222
252	84	84	84		-	84		42	0x0000_0413	0x0XXX_6333
252	63	63	63	-	-	63		31	0x0000_0413	0x0XXX_8444
252	42	42	42	-	-	42		21	0x0000_0413	0x0XXX_C666
252	31.5	31.5	31.5	-	-	31.5		15.75	0x0000_0413	0x0XXX_F888

The following constraints must be followed when programming the clock ratios:

- The frequency of DSPCLK can only be 1X or 2X of SYSCLK
- The frequency of IOCLK can only be 1X, 1/2X or 1/4X of SYSCLK
- The maximum clock output of PLL1 should not exceed 600MHz in best case (500MHz in worst case)
- The maximum memory clock frequency should not exceed 138MHz in SDRAM mode (worst case)
- The maximum memory clock frequency should not exceed 200MHz in DDR-32bit mode (worst case)
- The maximum memory clock frequency should not exceed 138MHz in DDR-16bit mode (worst case)

- After the PWR_CLK_RATIO register is changed, software shall not access DSP or IO modules in **Tready** time (= 0.1ms).

NOTE: As you may notice, some configurations can be achieved by multiple approaches such as the two configurations highlighted in gray. But to get better clock duty cycle, we suggest using the configuration with higher PLL VCO frequency.

NOTE: All the other clock configurations not shown in the above table are not guaranteed to be working.

5.2.7 Clock Switching

The clock configurations can be changed on the fly. According to the clock circuit described as above, there are two ways to change the clocks:

1. Change the configuration of the PLL's;
2. Change the clock sources;
2. Change the clock ratios.

But certain constraints must be followed when switching between different clock configurations:

- When changing the clock ratios, the clock source has to be the external 12 MHz crystal input.
- If the clock switching affects the DDR memory clock, the DDR memory has to be put into self-refresh mode during the switch.

5.2.8 Real Time Clock Registers

There is a programmable 16-bit divider (RTC_DIV) to divide the input 32.768 KHz clock to the frequency that users need (E.g. 1-Hz). The divided real time clock will be used to driven a 32-bit counter (RTC_COUNTER) that provides user the real-time.

In each cycle of the divided-real-time clock, there is a Hertz interrupt generated to the RISC. User can also configure an alarm (RTC_ALARM). When the RTC_COUNTER matches the alarm, there is an Alarm interrupt generated to the RISC.

Table 16. RTC Register Mapping

RISC Address <11:0>	Register	Description
0x0000	RTC_COUNTER	RTC Counter Register
0x0004	RTC_ALARM	RTC Alarm Register
0x0008	RTC_STATUS	RTC Status Register
0x000C	RTC_DIV	RTC Division Register
Others	-	Reserved

- **RTC Counter Register (RTC_COUNTER) – 0x0000**

The RTC counter register (RTC_COUNTER) is a read/write register and is not cleared by any reset source except the hardware reset (RESET_B). The counter may be written by the RISC at any time. It is recommended that the RTC_COUNTER to be protected by the MMU protection mechanisms.

Bit	Name	Default	Description
31:0 (R/W)	CN<31:0>	32'h0	Real-time Counter value.

- **RTC Alarm Register (RTC_ALARM) – 0x0004**

The real-time clock alarm register is a 32-bit register that is accessible by the RISC. In each cycle of the divided real time clock, this register is compared to the RTC_COUNTER. If the two matches and the enable bit (ALE) is set, then the alarm bit in the RTC status register is set.

Bit	Name	Default	Description
31:0 (R/W)	AL<31:0>	32'h0	Alarm value.

- **RTC Status Register (RTC_STATUS) – 0x0008**

Writing ones to AL and HZ will clear them.

Bit	Name	Default	Description
0 (R/W)	AL	1'b0	RTC alarm detected. 0 – No alarm has been detected. 1 – An alarm has been detected (RTNR matches RTC_ALARM).
1 (R/W)	HZ	1'b0	1 Hz rising-edge detected. 0 – No rising edge has been detected. 1 – A rising edge has been detected.
2 (R/W)	ALE	1'b0	RTC alarm interrupt enable. 0 – RTC alarm interrupt is not enabled. 1 – RTC alarm interrupt is enabled.
3 (R/W)	HZE	1'b0	1 Hz interrupt enable. 0 – The 1-Hz interrupt is not enabled. 1 – The 1-Hz interrupt is enabled.
31:4	-	28'h0	Reserved

- **RTC Division Register (RTC_DIV) – 0x000C**

To generate a 1-Hz divided real time clock, it needs to divide the clock input from the external real time crystal (RTC_CLK=32.768 KHz):

$$\text{RTC_DIV} = \text{RTC_CLK}/2 - 1$$

To generate real time clock in frequency other than 1Hz, for example, X-Hz, then it needs to setup the register as the following:

$$\text{RTC_DIV} = (\text{RTC_CLK}/\text{X})/2 - 1$$

Bit	Name	Default	Description
15:0 (R/W)	DIV<15:0>	16'h0	Division value.
31:16	-	16'h0	Reserved

5.3 Power Manager

5.3.9 Overview

Just as Atlas-I™, Atlas-II™ has multiple power management modes, such as Normal mode, Idle Mode, and Sleep mode. (Please refer to the Atlas-I™ Datasheet for the description of these modes).

Besides of that, Atlas-II™ has two new power-down modes: Standby mode and Deep Sleep mode:

- In Deep Sleep mode, the major part of the silicon will be powered off except the Real Time Clock, Power Manager, Reset Controller, Resource Sharing Controller, and GPIO; all clocks are stopped except the RTC.

Atlas-II™ has multiple power domains as following:

- IO power domains
 - SSTL IO power domain (1.8/2.5/3.3V)
 - General IO power domain (2.5/3.3V)
- Core power domains
 - Power-down domain (1.2V)
 - Non-powerdown domain (1.2V)
- PLL power domains
 - 1.2 or 3.3V (Please refer to the PLL in previous sections)

5.3.10 Power On/Off Sequence

When power on, always power up the higher voltage domain first. So if we have 1.8, 2.5, 3.3V power supply, please follow the power on/off sequence as shown in the following figure.

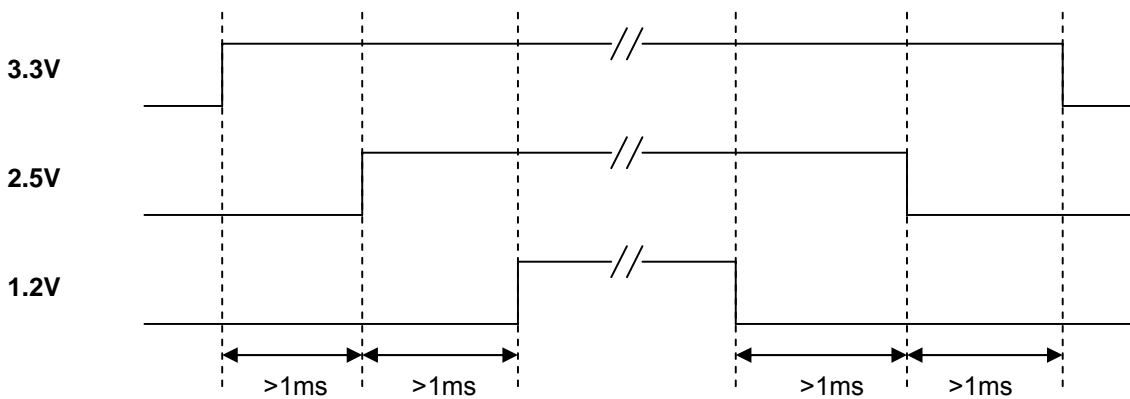


Figure 14. Power on/off Sequence

5.3.11 Sleep/Wakeup Sequence

The sleep/wakeup sequence is controlled by the 32.768 KHz (RTC) clock. The following figure shows the major events in the sleep/wakeup sequence.

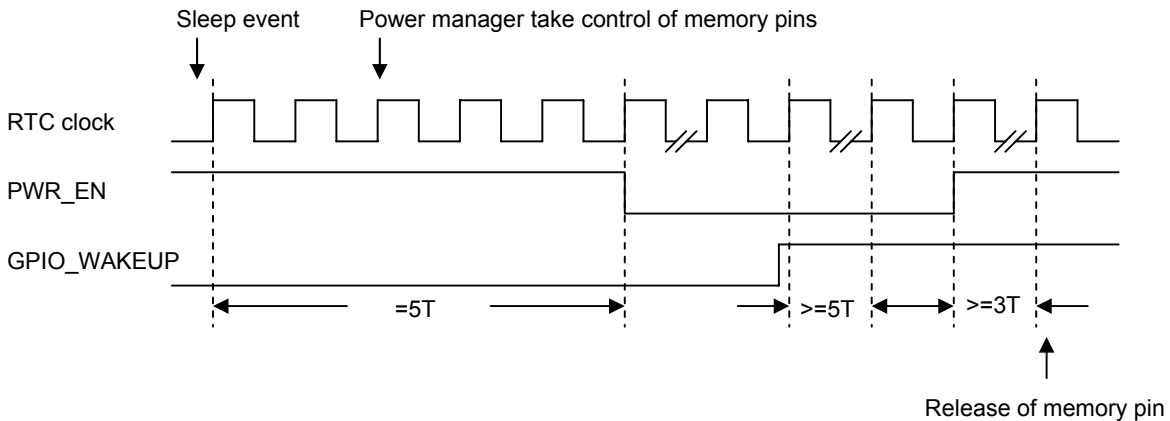


Figure 15. Sleep/Wakeup Sequence

First of all, there is a sleep event that triggers the sleep process. Normally this event is generated by writing a SLEEP register bit in software. Although by asserting the VDD_FAULT/BATT_FAULT pin can also generate a sleep event, it's not very applicable because the related clock switching may not be allowed (to happen)¹.

NOTE: It's suggested to use the software sleep even in case that VDD_FAULT/BATT_FAULT happens.

After sleep event is triggered, it takes 5 RTC cycle (~160us) that the PWR_EN signal will be disasserted. Then the power supply to the power-down domain can be turned off. In 2 RTC cycles after sleep event triggered, the memory pins will be taken over by the power manager.

And for wakeup event, it's normally triggered by the GPIO. After wakeup event triggered, it takes at least 5 RTC cycles (maybe more²) to raise the PWR_EN signal again. And after that, it takes some other RTC cycles (At least 3, but generally it will be much more so that the power supply and crystal can be stabilized.) to exit the sleep mode. In the exit of the sleep mode, the memory pins will be released.

5.3.12 Power Manager Registers

Table 17. PWR_MGR Register Mapping

RISC Address <11:0>	Register	Description
0x0000	PWR_CTRL	Power Manager Control Register
0x0004	PWR_CONFIG	Power Manager Configuration Register
0x0008	PWR_WAKEUP_EN	Power Manager Wake-up Enable Register
0x000C	PWR_SLEEP_STATUS	Power Manager Sleep Status Register
0x0010	PWR_SCRATCH_PAD	Power Manager Scratch-pad Register
0x0014	-	Reserved
0x0018	-	Reserved
0x001C	PWR_OSC_STATUS	Power Manager Oscillator Status Register
0x0020	PWR_CLK_SWITCH	Power Manager Clock Switch Register
0x0024	-	Reserved
0x0028	PWR_PLL1_CONFIG	Power Manager PLL1 Configuration Register

¹ Please refer to the previous sections about clock switching. To switch the clock ratio when clock source is not 12 MHz may incur the unexpected result.

² Please refer to the related document for more details about the wakeup.

0x002C	PWR_PLL2_CONFIG	Power Manager PLL2 Configuration Register
0x0030	PWR_CLK_EN	Power Manager Clock Enable Register
0x0034	PWR_WAIT_TIME	Power Manager Oscillator Wait Register
0x0038	PWR_STOP_LEVEL	Power Manager Clock Stop Level Register
0x003C	-	Reserved
0x0040	PWR_CLK_RATIO	Power Manager Clock Ratio Register
0x0044	PWR_XIN_RATIO	Power Manager XIN Ratio Register
0x0048	PWR_PAD_CTRL	Power Manager Pad Control Register
0x004C	PWR_DELAY_CTRL_0	Power Manager Delay Control Register 0
0x0050	PWR_DELAY_CTRL_1	Power Manager Delay Control Register 1
Others	-	Reserved

- **Power Manager Control Register (PWR_CTRL) – 0x0000**

Bit	Name	Default	Description
0 (W)	SLEEP¹	1'b0	1 – Force the system enter sleep mode.
1 (W)	MEM_STOP	1'b0	1 – Force the DRAM controller to stop. It needs to hold long enough (at least 2 refresh cycles) to make sure the DRAM controller is stopped.
31:1	-	31'h0	Reserved

- **Power Manager Configuration Register (PWR_CONFIG) – 0x0004**

Bit	Name	Default	Description
0 (R/W)	OPD_EN	1'b0	12-MHz oscillator power-down enable. 0 – Do not turn-off the oscillator during sleep mode (reset condition). 1 – Turn-off the 12-MHz oscillator during sleep mode.
1 (R/W)	OSC_FD	1'b0	Force 12-MHz oscillator power-down This bit is used to force the 12-MHz oscillator power down during normal operation mode. Before enable this bit, user needs to switch the clock source to the 32-KHz oscillator.
2 (R/W)	OSC_FO	1'b0	Force 12-MHz oscillator on. This bit is used to allow software to force the Atlas™ to use the 12-MHz oscillator instead of waiting for it to stabilize in the normal way.
31:3	-	29'h0	Reserved

- **Power Manager Wake-up Enable Register (PWR_WAKEUP_EN) – 0x0008**

Bit	Name	Default	Description
0 (R/W)	GPIO_WE	1'b1	Sleep wake-up enable by GPIO. 0 – Wake-up by GPIO disabled. 1 – Wake-up by GPIO enabled.
15:1	-	15'h0	Reserved.
30:16 (R/W)	WAIT<14:0>	15'h3ff	GPIO wake-up de-bounce wait cycle number (real time clock). To ensure a success GPIO wake-up, the GPIO must perform a 2-stage (de-active->active) toggle. Each stage should be longer than the period defined by this register.

¹ **Note:** This bit can be and only be cleared on wake-up or hardware reset.

			Besides of the WE<15:0>, WAIT<14:0> must be greater than 0 to enable the GPIO wakeup function.
31 (R/W)	ALARM_WE	1'b0	Sleep wake-up enable by RTC alarm. 0 – Wake-up by RTC alarm disabled. 1 – Wake-up by RTC alarm enabled.

NOTE: Only GPIO in Group 0 and Group 5 can be configured as wakeup source. Please refer to the GPIO section for more details.

- **Power Manager Sleep Status Register (PWR_SLEEP_STATUS) – 0x000C**

Bit	Name	Default	Description
0 (R/W)	SW_SLEEP	1'b0	Software sleep status 0 – Chip has not been placed in sleep mode by setting the force sleep (FS) control bit since it was last cleared by reset or by the CPU. 1 – Chip was placed in sleep mode by setting the force sleep (FS) control bit.
1 (R/W)	BATT_SLEEP	1'b0	Battery fault status. 0 – BATT_FAULT pin has not been asserted since it was last cleared by a hardware reset or by the CPU. 1 – BATT_FAULT pin has been asserted.
2 (R/W)	VDD_SLEEP	1'b0	VDD fault status. 0 – VDD_FAULT pin has not been asserted since it was last cleared by a hardware reset or by the CPU. 1 – VDD_FAULT pin has been asserted.
3 (R/W)	DRAM_HOLD	1'b0	SDRAM hold control. This bit is set upon exit from sleep mode and indicates that the MWE_B, MCKE_B<1:0>, MRAS_B, MCAS_B and MCS_B<1:0> continue to be held invalid so that the DRAM is still in self-refresh mode. The CPU can clear this bit after the DRAM interface has been configured but before any DRAM access is attempted. The CPU can also enable this bit if necessary. This bit is cleared on hardware reset.
4 (R/W)	FLASH_HOLD	1'b0	Flash hold control. This bit is set upon exit from sleep mode and indicates that the Flash interface (NAND & NOR) gets the hold of pins (otherwise the pins will be held by GPIO) so that the chip can boot from it after sleep. The CPU can clear this bit by writing a one to it after the boot is done and then the pins will be released to GPIO again. The CPU can also enable this bit if necessary. This bit is cleared on hardware reset.
31:5	-	27'h0	Reserved

- **Power Manager Scratch Pad Register (PWR_SCRATCH_PAD) – 0x0010**

Bit	Name	Default	Description
31:0	SP<31:0>	32'h0	Scratch Pad value.

- **Power Manager Oscillator Status Register (PWR_OSC_STATUS) – 0x001C**

Bit	Name	Default	Description
-----	------	---------	-------------

0 (R)	OSC_OK	1'b0	Oscillator OK. This bit is cleared on a hardware reset and set after the 32KHz oscillator has stabilized.
1 (R)	OSC_PD	1'b0	12MHz Crystal power down. 1 – Crystal is in power down mode. 0 – Crystal is activated.
31:2	-	30'h0	Reserved

- **Power Manager Clocks Switch Register (PWR_CLK_SWITCH) – 0x0020**

Bit	Name	Default	Description
1:0 (R/W)	SYS_CS	2'b00	System Clock Source Select. 00-> 12MHz crystal 01-> Select PLL1 as the system clock source 10-> Select PLL2 as the system clock source 11-> 32KHz.
3:2 (R/W)	USB_CS	2'b00	USB Clock Source Select. Definition is the same as above.
5:4 (R/W)	CKO_0_CS	2'b00	CKO_0 Clock Source Select. Definition is the same as above.
7:6 (R/W)	CKO_1_CS	2'b00	CKO_1 Source Select. Definition is the same as above.
31:8	-	24'h0	Reserved

- **Power Manager PLL1 Configuration Register (PWR_PLL1_CONFIG) – 0x28**

Both PLL1 and PLL2 can be programmed to different frequencies. User needs to make sure that when programming one PLL the Atlas™-2 clock source has been switched to another PLL. And only after the PLL is stable, the clock source can be switched back.

NF<5:0> and NR<3:0> determine the clock frequency by the following equation:

$$F_{OUT} = NF/NR \times F_{IN}$$

NOTE: NF = (3 – NO) * (F<5:0> + 2), NR = R<3:0> + 1, NO = OD + 1.

Meanwhile the following constraints must be followed:

- $10\text{MHz} < F_{REF} < 50\text{MHz}$, where $F_{REF} = F_{IN}*/NR$ (when F_{IN} is 12MHz, NR has to be 1)
- $500\text{MHz} < F_{VCO} < 1000\text{MHz}$, where $F_{VCO} = F_{IN}*NF*NO/NR$
- $NF \geq 6*(3 - NO)$

For example, if $F_{IN} = 12\text{MHz}$, and by default $NR = 1$, $NO = 2$, $NF = 21$, so the PLL output will be 252MHz.

Bit	Name	Default	Description
5:0 (R/W)	F<5:0>	6'h13	Feedback divider (0~63)
9:6 (R/W)	R<3:0>	4'h0	Input divider (0~15) Note: R has to be 0 when input frequency is 12MHz.
10 (R/W)	OD	1'b1	Output divider 1'b0 – NO = 1 1'b1 – NO = 2 Note: When OD is 1'b1, FOUT would have around

			50% duty cycle.
11 (R/W)	BP	1'b1	1 – Bypass the PLL 0 – Not bypass the PLL
12 (R/W)	OE_B	1'b1	FOUT enable 1 – Disabled 0 – Enabled
13 (R/W)	PD	1'b1	Power down mode 1 – Power down 0 – Power up
31:14	-	18'h0	Reserved

- **Power Manager PLL2 Configuration Register (PWR_PLL2_CONFIG) – 0x002C**

NF<7:0> and NR<4:0> determine the clock frequency by the following equation:

$$F_{OUT} = NF/(NR*NO) \times F_{IN}$$

NOTE: NF = 2*F<7:0>, NR = R<3:0>, NO please refer to the following table.

Meanwhile the following constraints must be followed:

- 2MHz < F_{REF} < 8MHz, where F_{REF} = F_{IN}*NR (when F_{IN} is 12MHz, NR has to be 2~6)
- 200MHz < F_{VCO} < 400MHz, where F_{VCO} = F_{IN}*NF/NR

For example, if F_{IN} = 12MHz, and by default NR = 2, NO = 2, NF = 64, so the PLL output will be 192MHz.

Bit	Name	Default	Description
7:0 (R/W)	F<7:0>	8'h20	Feedback divider (1~255) Note: F<7:0> cannot be 0.
12:8 (R/W)	R<4:0>	5'h2	Input divider (1~31) Note: R<4:0> cannot be 0.
14:13 (R/W)	OD<1:0>	2'b10	Output divider 2'b00 – Not Allowed 2'b01 – NO = 1 2'b10 – NO = 2 2'b11 – NO = 4 Note: When OD is 2'b10 or 2'b11, FOUT would have around 50% duty cycle.
15 (R/W)	BP	1'b1	1 – Bypass the PLL 0 – Not bypass the PLL
16 (R/W)	OE_B	1'b1	FOUT enable 1 – Disabled 0 – Enabled
17 (R/W)	PD	1'b1	Power down mode 1 – Power down 0 – Power up
31:18	-	14'h0	Reserved

- **Power Manager Clocks Enable Register (PWR_CLK_EN) – 0x0030**

User can enable/disable the clock of majority logic in Atlas™, except the RISC Core, Memory Controller, and some other system control modules. The clock to these modules can also be disabled when the chip enters Sleep mode because user can power down the PLL and 12-MHz oscillator.

Bit	Name	Default	Description
0 (R/W)	DSP_EN	1'b0	DSP Core clock enable. 1 – Enable (The following are the same.) 0 – Disable
1 (R/W)	ROM_EN	1'b1	Flash Controller clock enable.
2 (R/W)	DMA_EN	1'b0	DMA Controller clock enable.
3 (R/W)	LCD_EN	1'b0	LCD Controller clock enable.
4 (R/W)	GPS_EN	1'b0	GPS clock enable.
5 (R/W)	USB_EN	1'b0	USB OTG clock enable.
6 (R/W)	PCMCIA_EN	1'b0	PCMCIA Core clock enable.
7 (R/W)	CAM_EN	1'b0	Video Input Port clock enable.
8 (R/W)	CODEC_EN	1'b0	CODEC clock enable
9 (R/W)	UART_EN	1'b0	UART (SP0, 6, & 7) clock enable
10 (R/W)	SP1_EN	1'b0	Serial Port 1 clock enable
11 (R/W)	SP2_EN	1'b0	Serial Port 2 clock enable
12 (R/W)	SP3_EN	1'b0	Serial Port 3 clock enable
13 (R/W)	SP4_EN	1'b0	Serial Port 4 clock enable
14 (R/W)	SP5_EN	1'b0	Serial Port 5 clock enable
15 (R/W)	IPOLATE_EN	1'b0	Interpolation block clock enable
16 (R/W)	-	1'b0	Reserved
17 (R/W)	SDIO_EN	1'b0	SDIO interface clock enable
18 (R/W)	SM_EN	1'b1	SmartMedia interface clock enable
19 (R/W)	COPY_EN	1'b0	PCI copy block clock enable
20 (R/W)	-	1'b0	Reserved
21 (R/W)	IO_EN	1'b0	I/O clock enable
22 (R/W)	PCI_EN	1'b0	PCI clock enable
23 (R/W)	CKO0_EN	1'b0	Output clock_0 enable
24 (R/W)	CKO1_EN	1'b0	Output clock_1 enable
25 (R/W)	CAN0_EN	1'b0	CAN Bus 0 clock enable
26 (R/W)	CAN1_EN	1'b0	CAN Bus 1 clock enable
27 (R/W)	PADARB_EN	1'b1	Pad arbitration clock enable
28 (R/W)	XIN_EN	1'b1	XIN clock enable
29 (R/W)	XINW_EN	1'b1	XINW clock enable
30 (R/W)	IDE_EN	1'b0	IDE interface clock enable
31	-	1'b0	Reserved

NOTE: UART0, 6, 7 share the same clock enable bit.

- **Power Manager Oscillator Wait Time Register (PWR_WAIT_TIME) – 0x0034**

During wake-up sequence from sleep mode, there is a two-step procedure: In the first step, an internal timer begins to time the power ramp. This timer waits for approximately 25ms by default. In the second step another internal timer begins to time the oscillator as it begins to ramp up to speed. This timer waits for 150ms by default. If the oscillator is not powered off during sleep mode, the second step is not needed.

Bit	Name	Default	Description
15:0 (R/W)	OOK_WAIT_TIME	16'h327	Oscillator-OK wait time
31:16 (R/W)	OPU_WAIT_TIME	16'h1333	Oscillator-Power-up wait time

- **Power Manager Clock Stop Level Register (PWR_STOP_LEVEL) – 0x0038**

Bit	Name	Default	Description
0 (R/W)	STOP_AT_LOW	1'b1	Clock stop level. 1 – Clock stops at low (default). 0 – Clock stops at high.
31:1	-	30'h0	Reserved

- **Power Manager CLK Ratio Register (PWR_CLK_RATIO) – 0x0040**

This register is used to configure the frequency of the clock by programming the divide parameter from the clock source.

Bit	Name	Default	Description
3:0 (R/W)	CPU_RATIO<3:0>	4'h1	The clock ratio PLL_OUT ¹ :CPUCLK (default value is 1) It can be 1, 2, 3, 4, 6, 8, 12, or 16. Any other value written to it will be equivalent to the next value bigger than that (e.g. 5 is equivalent to 6; 9 is equivalent to 12; etc.). It's the same as all the following ratios.
7:4 (R/W)	DSP_RATIO<3:0>	4'h1	PLL_OUT:DSPCLK ratio (same as above)
11:8 (R/W)	SYS_RATIO<3:0>	4'h1	PLL_OUT:SYSCLK ratio (same as above)
15:12 (R/W)	IO_RATIO<3:0>	4'h2	PLL_OUT:IOCLK ratio (same as above)
19:16 (R/W)	USB_RATIO<3:0>	4'h1	PLL_OUT:USBCLK ratio It can be 1, 2, 4, 8, or 16. Any other value written to it will be equivalent to the next value bigger than that (e.g. 5 is equivalent to 8; 9 is equivalent to 16; etc.). It's the same as all the following ratios.
23:20 (R/W)	CKO_0_RATIO<3:0>	4'h1	PLL_OUT:CKO_0 ratio (same as above)
27:24 (R/W)	CKO_1_RATIO<3:0>	4'h1	PLL_OUT:CKO_1 ratio (same as above)
31:28	-	4'h0	Reserved

The following constraints must be followed:

- The frequency of DSPCLK can only be 1X or 2X of SYSCLK
- The frequency of IOCLK can only be 1/2X of SYSCLK

NOTE: Any configurations violated the above constraints will incur unexpected result.

- **Power Manager XIN Ratio Register (PWR_XIN_RATIO) – 0x0044**

Bit	Name	Default	Description
0 (R/W)	XIN_RATIO	1'b0	1'b0 – XIN is not divided by 2 1'b1 – XIN is divided by 2

¹ PLL_OUT is determined by PWR_CLK_SWITCH register: it can be 12MHz crystal, 32KHz crystal, PLL1, or PLL2.

31:1	-	31'h0	Reserved
------	---	-------	----------

- **Power Manager Pad Control Register (PWR_PAD_CTRL) – 0x0048**

Atlas™-II memory interface can be configured to 4 different modes: LVCMOS18, SSTL2, LVCMOS5, and LVCMOS33. This register is used to control these different modes.

Bit	Name	Default	Description
1:0 (R/W)	SSTL_S<1:0>	2'b11	2'b00 – LVCMOS 1.8V 2'b01 – SSTL 2.5V 2'b10 – LVCMOS 2.5V 2'b11 – LVCMOS 3.3V (default)
2 (R/W)	SSTL_PD	1'b0	This bit is only valid in SSTL 2.5V mode. 1'b1 – Memory pad is power down 1'b0 – Memory pad is power up (default)
31:3	-	29'h0	Reserved

- **Power Manager Delay Control Register (PWR_DELAY_CTRL_0) – 0x004C**

Bit	Name	Default	Description
5:0 (R/W)	HCLKEN_DELAY<5:0>	8'h6	The path delay of HCLKEN will be HCLKEN_DELAY * 0.2ns (Only bit<4:0> is valid, and it's the same for all the followings.)
7:6	Reserved	-	-
13:8 (R/W)	MCKO_DELAY<5:0>	8'ha	The path delay of MCK_O will be MCK_O_DELAY * 0.2ns
15:14	Reserved	-	-
21:16 (R/W)	MCK2X_O_DELAY<5:0>	8'h3	The path delay of MCK2X_O will be MCK2X_O_DELAY * 0.2ns
23:22	Reserved	-	-
24 (R/W)	MCK_I_SEL	1'h0	0: mck_i is selected from pad loop back 1: mck_i is selected from internal mck_o
31:25	-	7'h0	Reserved

- **Power Manager Delay Control Register (PWR_DELAY_CTRL_1) – 0x0050**

Bit	Name	Default	Description
5:0 (R/W)	CKO_0_DELAY<5:0>	8'hf	The path delay of CKO_0 will be CKO_0_DELAY * 0.2ns
7:6	Reserved	-	-
13:8 (R/W)	CKO_1_DELAY<5:0>	8'hf	The path delay of CKO_1 will be CKO_1_DELAY * 0.2ns
15:14	Reserved	-	-
21:16 (R/W)	EXTCLK_DELAY<5:0>	8'hf	The path delay of EXTCLK will be EXTCLK_DELAY * 0.2ns
23:22	Reserved	-	-
29:24 (R/W)	MCK_I_DELAY<5:0>	8'h0	The path delay of MCK_I will be MCK_I_DELAY * 0.2ns
31:30	Reserved	-	-

5.4 Interrupt Controller

5.4.1 Overview

In the Atlas™-II processor, both the RISC Core and the DSP Core can accept external interrupts. In addition, the RISC and DSP can interrupt them each other.

The RISC Core has two external interrupt inputs: FIQ and IRQ. FIQ allows for fast interrupt processing by providing five additional dedicated general-purpose registers.

The DSP Core has up to six external interrupts pins: IRQ2n, IRQ1n, IRQ0n, IRQL1n, IRQL0n, and IRQEn. In Atlas™-II, only IRQL1n and IRQL0n are used for all interrupt sources.

The interrupt hierarchy of the Atlas™-II is a two-level structure. The first level is responsible for the masking/unmasking of all enabled interrupts and sends the interrupt to the processor; the second level is implemented in the source device (the device generates the first level interrupt bit). The second-level interrupt status register gives additional information about the interrupt and is used inside the service routine. In general, multiple second-level interrupts are OR'ed to produce a first-level interrupt bit. The enabling of interrupts is performed inside the the source device.

The following figure shows the interrupt controller block RISC part diagram. For the DSP part it is almost the same, and is not included.

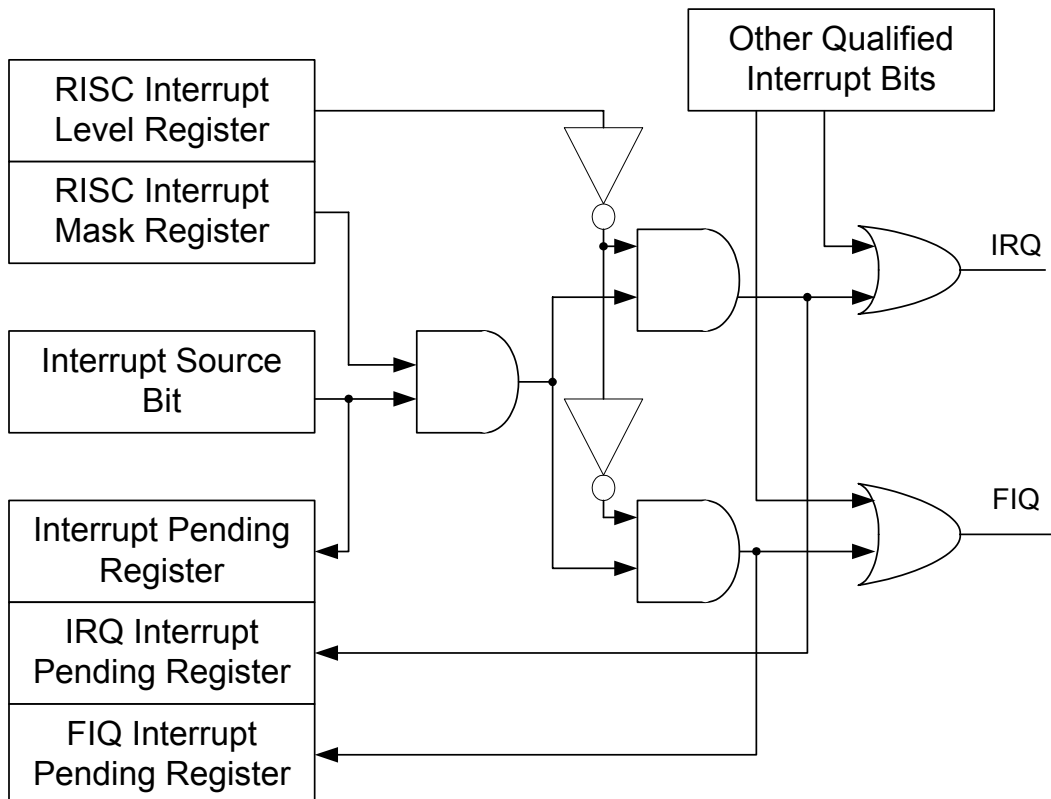


Figure 16. Interrupt Controller Block Diagram

For the RISC, the first level of structure is represented by the interrupt controller IRQ pending register (INT_IRQ_PENDING) and the interrupt controller FIQ pending register (INT_FIQ_PENDING). The INT_IRQ_PENDING register contains the interrupts that are programmed to generate an IRQ interrupt. The INT_FIQ_PENDING register contains all valid interrupts that are programmed to generate a FIQ interrupt. The routing is programmed via the RISC interrupt controller level register (INT_RISC_LEVEL).

NOTE: In most cases, the root source of an interrupt can be determined by reading these two registers, followed by the status register within the device to find the exact function needing service.

For the DSP, we use low level sensitive interrupts of the DSP core, IRQL1n and IRQL0n. The routing of interrupts to these two interrupts is programmed by the DSP interrupt controller level register (INT_DSP_LEVEL). This is a more simple process than with the RISC: all the qualified interrupt bits will be stored in an interrupt controller pending registers (INT_DSP1_PENDING & INT_DSP0_PENDING). The DSP can read the according interrupt pending register, followed by the status register in the device. It will use a different interrupt mask register from the RISC so that the DSP can be interrupted independently by different sources with the RISC.

5.4.2 Interrupt Controller Registers

The interrupt Controller registers are mapped as the following table:

Table 18. Interrupt Controller Register Mapping

RISC Address <11:0>	DSP Address <6:0>	Register	Description
0x0000	0x0~1	INT_PENDING	Interrupt Pending Register
0x0004	-	INT_IRQ_PENDING	IRQ Pending Register
0x0008	-	INT_FIQ_PENDING	Read only FIQ Pending Register Default: 0x00000000
0x000C	0x6~7	INT_DSP0_PENDING	Read only DSP Pending Register for DSP's IRQL0 interrupt Default: 0x00000000
0x0010	-	INT_RISC_MASK	RISC Mask Register Default: 0x00000000
0x0014	0xA~B	INT_DSP_MASK	DSP Mask Register Default: 0x00000000
0x0018	-	INT_RISC_LEVEL	RISC Level Register Default: 0x00000000
0x001C	0xE	INT_DSP_ACCEN	DSP Access enable This register is read only register for DSP. Only RISC can R&W it. Default: 0x00000000
0x0020	0x10	CHIP_ID	This is a read only register. It contains the CHIP ID. Default: Depend on chip version.
0x0024	0x12~13	INT_DSP_LEVEL	DSP Level Register Default: 0x00000000
0x0028	0x14~15	INT_DSP1_PENDING	Read only DSP Pending Register for DSP's IRQL1 interrupt Default: 0x00000000

0x002C		INT_RISC_MASK_EXT	Timer 1- Timer 5 interrupt enable to RISC Default: 0x0000001F
0x0030	0x18~19	INT_DSP_MASK_EXT	Timer 1- Timer 5 interrupt enable to DSP Default: 0x00000000
0x0034	0x1A~1B	INT_PENDING_EXT	Extended Interrupt Pending Register
0x0038	-	INT_IRQ_PENDING_EXT	Extended IRQ Pending Register
0x003C	-	INT_FIQ_PENDING_EXT	Extended FIQ Pending Register
0x0040	0x20~21	INT_DSP0_PENDING_EXT	Extended DSP Pending Register for DSP's IRQL0 interrupt r
0x0044	0x22~23	INT_DSP1_PENDING_EXT	Extended DSP Pending Register for DSP's IRQL1 interrupt r
0x0048	-	INT_ID	Interrupt ID
Others	Others	-	Reserved

- **Interrupt Controller Pending Register (INT_PENDING) – RISC: 0x0000, DSP: 0x000~0x001**

The INT_PENDING is a 32-bit read-only register that shows all active interrupts in the system. These bits are not affected by the state of the mask register. The following table shows the pending interrupt source assigned to each bit position in the INT_PENDING. For more details on the second-level interrupts, see the section describing that unit.

Bits within the INT_PENDING are read only, and represent the logical OR of status bits for a given interrupt within the source unit. Once an interrupt has been serviced, the handler clears the pending interrupt at the *source* by writing a one to the necessary status bit. Clearing the interrupt status bit at the source automatically clears the corresponding INT_IRQ_PENDING and INT_FIQ_PENDING flag provided there are no other interrupt status bits set within the source unit. All interrupt source status bits are cleared by writing a one to them. Writing a zero to an interrupt status bit has no effect.

After power up, the default value is unknown.

Bit/Name	Unit	Source Module	Description
IP31 (R)	System	Real-time Clock	Clock TIC Count
IP30 (R)			RTC Alarm
IP29 (R)		OS Timer	OS Timer equals match register 1-5.
IP28 (R)			OS Timer equals match register 0.
IP27 (R)	Peripheral	CAN Bus 0/1	CAN Bus 0/1 service request.
IP26 (R)		Serial Port 4/5/6/7	Serial Port 4/5/6/7 service request.
IP25 (R)		DMA Controller	DMA finished.
IP24 (R)		LCD Controller /LCD 2D	LCD Controller/LCD 2D service request.
IP23 (R)		Serial Port 3	Serial Port 3 service request.
IP22 (R)		Serial Port 2	Serial Port 2 service request.
IP21 (R)		Serial Port 1	Serial Port 1 service request.
IP20 (R)		Serial Port 0	Serial Port 0 service request.
IP19 (R)		System	System related block interrupt. This interrupt is the bit-or of IOBG, SYS2PCI,PCIARB,CPUIF,SYSARB These interrupt are for test only
IP18 (R)		GPS	GPS service request.
IP17 (R)		USB	USB service request.
IP16 (R)		CODEC	CODEC service request
IP15 (R)		Camera	Camera service request.

IP14 (R)		Smartmedia /NAND Flash	Smartmedia/NAND Flash service request.
IP13 (R)		YUV_CHG	YUV_CHG interrupt request
IP12 (R)		IPOLATE	IPOLATE interrupt request
IP11 (R)		PCMCIA	If PCMCIA is enabled this is PCMCIA interrupt request.
IP10 (R)		DSP -> RISC (RISC access)	When RISC read this bit, it contains DSP interrupt RISC information.
IP10 (R)		RISC -> DSP (DSP access)	When DSP read this bit, it contains RISC interrupt DSP information.
IP9 (R)		IDE	IDE interrupt request
IP8 (R)		SD	SD interrupt request.
IP7 (R)		YUV_RGB	YUV_RGB interrupt request
IP6 (R)		PCI_COPY	PCI_COPY interrupt request
IP5 (R)	GPIO		GPIO interrupt request
IP4 (R)			GPIO interrupt request
IP3 (R)			GPIO interrupt request.
IP2 (R)			GPIO interrupt request.
IP1 (R)			GPIO interrupt request.
IP0 (R)			GPIO interrupt request.

- **Interrupt Controller IRQ/FIQ Pending Register (INT_IRQ_PENDING/INT_FIQ_PENDING) – RISC: 0x0004/0x0008**

The INT_IRQ_PENDING and the INT_FIQ_PENDING contain one flag per interrupt (32 total) that indicates an interrupt request has been made by a unit. Inside the interrupt service routine, the INT_IRQ_PENDING and INT_FIQ_PENDING registers are read by RISC to determine the interrupt source. In general, software then reads status registers within the interrupting device to determine how to service the interrupt.

The following table shows the bit locations corresponding to the 32 separate interrupt pending status flags in the INT_IRQ_PENDING. The next table shows the bit locations corresponding to the 32 separate interrupt pending status flags in the INT_FIQ_PENDING. This is a read-only register.

Bit	Name	Default	Description
31:0 (R)	IP<31:0>	32'h0	IRQ pending bits. 0 – No interrupt pending. 1 – Interrupt pending.

Bit	Name	Default	Description
31:0 (R)	FP<31:0>	32'h0	FIQ pending bits. 0 – No interrupt pending. 1 – Interrupt pending.

- **Interrupt Controller DSP Pending Register (INT_DSP0_PENDING) – RISC: 0x000C, DSP: 0x0006~0x0007**

The INT_DSP0_PENDING contain one flag per interrupt (32 total) that indicates an interrupt request to IRQL0 has been made by a unit. Inside the interrupt service routine, the INT_DSP0_PENDING are read by DSP to determine the interrupt source. In general, software then reads status registers within the interrupting device to determine how to service the interrupt.

The following table shows the bit locations corresponding to the 32 separate interrupt pending status flags in the INT_DSP0_PENDING. This is a read-only register.

Bit	Name	Default	Description
31:0 (R)	DSP0<31:0>	32'h0	DSP Interrupt pending bits. 0 – No interrupt pending. 1 – Interrupt pending.

- **Interrupt Controller RISC Mask Register (INT_RISC_MASK) – RISC: 0x0010**

There are two interrupt mask registers, one for RISC (INT_RISC_MASK) and the other for DSP (INT_DSP_MASK). Each mask register contains one mask bit per pending interrupt bit (totally 32).

Mask bits serve two purposes. First, they allow periodic software polling of interruptible sources while preventing them from actually causing an interrupt. Second, they allow the interrupt handler routine to prevent interrupts of lower priority from occurring while still maintaining a list of pending interrupts that may have occurred previously (or during the servicing of another interrupt).

The INT_RISC_MASK/INT_DSP_MASK is not initialized at reset; a question mark indicates that the values are unknown at reset. The following table shows the bit locations corresponding to the 32 separate interrupt mask bits.

Bit	Name	Default	Description
31:0 (W/R)	RM<31:0>	32'h0	RISC interrupts mask bits. 0 – Pending interrupt is masked from becoming active. 1 – Pending interrupt is allowed to become active.

The IP 11, IP 15, IP 24, IP 26, IP 27, IP 29 for RISC/DSP is MUXED with several interrupt:

IP11: PCMCIA/EXT

IP15: Camera

IP24: LCD Controller /LCD 2D

IP26: Serial Port 4/5/6/7

IP27: CANBUS 0/1

IP29: OS Timer 1-5

So there needs to have some 2nd level mask bits for those interrupts. Please refer to the INT_RISC_MASK_EXT/INT_DSP_MASK_EXT registers for the 2nd level mask bits.

- **Interrupt Controller DSP Mask Register (INT_DSP_MASK) – RISC: 0x0014, DSP: 0x00A~0x00B**

The definition of each bit is the same as above.

Bit	Name	Default	Description
31:0 (W/R)	DM<31:0>	32'h0	DSP interrupts mask bits. 0 – Pending interrupt is masked from becoming active. 1 – Pending interrupt is allowed to become active.

- **Interrupt Controller Level Register (INT_RISC_LEVEL) – RISC: 0x0018**

The RISC interrupt controller level register (INT_RISC_LEVEL) controls whether a pending interrupt generates an FIQ or an IRQ RISC¹ interrupt.

¹ This register can only be accessed by RISC.

If a pending interrupt is unmasked, the corresponding INT_LEVEL bit field is decoded to select which interrupt should be asserted. If the interrupt is masked, then the corresponding bit in the INT_LEVEL has no effect. The following table shows the location of all interrupt level bits in the INT_LEVEL; question marks indicate that the values are unknown at reset.

Bit	Name	Default	Description
31:0 (W/R)	RIL<31:0>	32'h0	RISC Interrupts level bits. 0 – Interrupt routed to RISC IRQ interrupt input. 1 – Interrupt routed to RISC FIQ interrupt input.

- **Interrupt DSP Access Enable Register (INT_DSP_ACCEN) – RISC: 0x001C, DSP: 0x00E**

At default all the register can only be written RISC only. In order for DSP to write INT_DSP_MASK register, user should set INT_DSP_ACCEN register by RISC.

Bit	Name	Default	Description
0 (W/R)	DSP_EN	1'b0	DSP Access enable
31:1	-	31'h0	Reserved

- **Chip ID Register (CHIP_ID) – RISC: 0x0020, DSP: 0x010**

This register is read only, it contain the CHIP ID for different chip version.

Bit	Name	Default	Description
15:0 (W/R)	CHIP_ID<15:0>	16'h0	CHIP ID
31:16	-	-	Reserved

- **Interrupt Controller DSP Pending Register (INT_DSP1_PENDING) – RISC: 0x0024, DSP: 0x012~0x013**

The INT_DSP1_PENDING contain one flag per interrupt (32 total) that indicates an interrupt request to IRQL1 has been made by a unit. Inside the interrupt service routine, the INT_DSP1_PENDING are read by DSP to determine the interrupt source. In general, software then reads status registers within the interrupting device to determine how to service the interrupt.

The following table shows the bit locations corresponding to the 32 separate interrupt pending status flags in the INT_DSP1_PENDING. This is a read-only register.

Bit	Name	Default	Description
31:0 (R)	DSP1<31:0>	32'h0	DSP Interrupt pending bits. 0 – No interrupt pending. 1 – Interrupt pending.

- **Interrupt Controller DSP Level Register (INT_DSP_LEVEL) – RISC: 0x0028, DSP: 0x014~0x015**

The DSP interrupt controller level register (INT_DSP_LEVEL) controls whether a pending interrupt generates an IRQL1 or an IRQL0 DSP interrupt.

If a pending interrupt is unmasked, the corresponding INT_LEVEL bit field is decoded to select which interrupt should be asserted. If the interrupt's mask bit is 0, then the corresponding bit in the INT_LEVEL has no effect. The following table shows the location of all interrupt level bits in the INT_LEVEL; question marks indicate that the values are unknown at reset.

Bit	Name	Default	Description
31:0 (W/R)	DIL<31:0>	32'h0	RISC Interrupts level bits. 0 – Interrupt routed to DSP IRQL0 interrupt input. 1 – Interrupt routed to DSP IRQL1 interrupt input.

- **Interrupt Controller RISC Mask Extended Register (INT_RISC_MASK_EXT) – RISC: 0x002C**

Bit	Name	Default	Description
0 (W/R)	RME<0>	1'b0	Interrupt extended mask register bit for PCMCIA in IP11: 0 – Pending interrupt is masked from becoming active. 1 – Pending interrupt is allowed to become active. (Same as following)
1 (W/R)	RME<1>	1'b0	Interrupt extended mask register bit for EXTPORT in IP11
2 (W/R)	RME<2>	1'b0	Interrupt extended mask register bit for Camera in IP15
3	-	1'b0	Reserved
4 (W/R)	RME<4>	1'b0	Interrupt extended mask register bit for LCD Controller in IP24
5 (W/R)	RME<5>	1'b0	Interrupt extended mask register bit for LCD2D in IP24
6 (W/R)	RME<6>	1'b0	Interrupt extended mask register bit for CANBUS0 in IP27
7 (W/R)	RME<7>	1'b0	Interrupt extended mask register bit for CANBUS1 in IP27
8 (W/R)	RME<8>	1'b0	Interrupt extended mask register bit for USP4 in IP26
9 (W/R)	RME<9>	1'b0	Interrupt extended mask register bit for USP5 in IP26
10 (W/R)	RME<10>	1'b0	Interrupt extended mask register bit for USP6 in IP26
11 (W/R)	RME<11>	1'b0	Interrupt extended mask register bit for USP7 in IP26
12 (W/R)	RME<12>	1'b0	Interrupt extended mask register bit for OS Timer1 in IP29
13(W/R)	RME<13>	1'b0	Interrupt extended mask register bit for OS Timer2 in IP29
14(W/R)	RME<14>	1'b0	Interrupt extended mask register bit for OS Timer3 in IP29
15(W/R)	RME<15>	1'b0	Interrupt extended mask register bit for OS Timer4 in IP29
16(W/R)	RME<16>	1'b0	Interrupt extended mask register bit for OS Timer5 in IP29
31:17	-	15'h0	Reserved

- **Interrupt Controller DSP Mask Extended Register (INT_DSP_MASK_EXT) – RISC: 0x0030, DSP: 0x018~0x019**

The definition of each bit is the same as above.

Bit	Name	Default	Description
16:0 (W/R)	DME<16:0>	17'h0	DSP extended interrupts mask bits. 0 – Pending interrupt is masked from becoming active. 1 – Pending interrupt is allowed to become active.
31:17	-	15'h0	Reserved

- **Interrupt Controller Pending Extended Register (INT_PENDING_EXT) – RISC: 0x0034 DSP: 0x01A~0x1C**

This register contains the extended interrupt pending for MUXED interrupt: IP 11, IP 15, IP 24, IP 26, IP 27, IP 29. Following table details the extended interrupt pending register content.

Bit	Name	Default	Description
0(R)	PE<0>	1'b0	Interrupt extended bit for PCMCIA in IP11
1(R)	PE<1>	1'b0	Interrupt extended bit for EXTPORT in IP11

2(R)	PE<2>	1'b0	Interrupt extended bit for CAMERA in IP15
3	-	1'b0	Reserved
4(R)	PE<4>	1'b0	Interrupt extended bit for LCD Controller in IP24
5(R)	PE<5>	1'b0	Interrupt extended bit for LCD2D in IP24
6(R)	PE<6>	1'b0	Interrupt extended bit for CANBUS0 in IP27
7(R)	PE<7>	1'b0	Interrupt extended bit for CANBUS1 in IP27
8(R)	PE<8>	1'b0	Interrupt extended bit for USP4 in IP26
9(R)	PE<9>	1'b0	Interrupt extended bit for USP5 in IP26
10(R)	PE<10>	1'b0	Interrupt extended bit for USP6 in IP26
11(R)	PE<11>	1'b0	Interrupt extended bit for USP7 in IP26
12(R)	PE<12>	1'b0	Interrupt extended bit for OS Timer1 in IP29
13(R)	PE<13>	1'b0	Interrupt extended bit for OS Timer2 in IP29
14(R)	PE<14>	1'b0	Interrupt extended bit for OS Timer3 in IP29
15(R)	PE<15>	1'b0	Interrupt extended bit for OS Timer4 in IP29
16(R)	PE<16>	1'b0	Interrupt extended bit for OS Timer5 in IP29
31-17	-	15'h0	Reserved

- **Interrupt Controller IRQ/FIQ Extended Pending Register**
(INT_IRQ_PENDING_EXT/INT_FIQ_PENDING_EXT) – RISC: 0x0038/0x003C

The INT_IRQ_PENDING_EXT and the INT_FIQ_PENDING_EXT contain one flag per interrupt for MUXED interrupts. Please refer to INT_PENDING_EXT register for the exact bit definition.

Bit	Name	Default	Description
16:0 (R)	IPE<16:0>	17'h0	IRQ extended pending bits. 0 – No interrupt pending. 1 – Interrupt pending.
31:17	-	15'h0	Reserved

Bit	Name	Default	Description
16:0 (R)	FPE<16:0>	17'h0	FIQ extended pending bits. 0 – No interrupt pending. 1 – Interrupt pending.
31:17	-	15'h0	Reserved

- **Interrupt Controller DSP/DSP1 Extended Pending Register**
(INT_DSP0_PENDING_EXT/INT_DSP1_PENDING_EXT) – RISC: 0x0040/0x0044 DSP:0x20/0x22

The INT_DSP0_PENDING_EXT and the INT_DSP1_PENDING_EXT contain one flag per interrupt for MUXED interrupts. Please refer to INT_PENDING_EXT register for the exact bit definition.

Bit	Name	Default	Description
16:0 (R)	DSP0E<16:0>	17'h0	DSP0 extended pending bits. 0 – No interrupt pending. 1 – Interrupt pending.
31:17	-	15'h0	Reserved

Bit	Name	Default	Description
16:0 (R)	DSP1E<16:0>	17'h0	DSP1 extended pending bits. 0 – No interrupt pending. 1 – Interrupt pending.

31:17	-	15'h0	Reserved
-------	---	-------	----------

- **Interrupt Controller ID Register (INT_ID) – RISC: 0x0048**

The Interrupt Controller ID register returns the interrupt ID from pending interrupts. Each time RISC read this register, it will return current highest priority enabled pending interrupt.

Bit	Name	Default	Description
7:0 (R)	INT_ID<7:0>	8'hff	Interrupt ID
31:8	-	-	Reserved

Following table gives detail information about the priority and description of each ID (the smaller the ID value the higher the interrupt priority).

ID value	Description
0	Interrupt from TIMER0
1	Interrupt from RTC-ALARM
2	Interrupt from RTC-TIC
3	-
4	-
5	-
6	Interrupt from DMA
7	-
8	Interrupt from UPS3
9	Interrupt from USP2
10	Interrupt from USP1
11	Interrupt from USP0
12	Interrupt from SYSTEM
13	Interrupt from GPS
14	Interrupt from USB
15	Interrupt from CODEC
16	-
17	Interrupt from SMARTMEDIA
18	Interrupt from YUVCHG
19	Interrupt from IPOLATE
20	-
21	Interrupt from DSP-IF
22	Interrupt from IDE
23	Interrupt from SDIO
24	Interrupt from YUV2RGB
25	Interrupt from PCICOPY
26	Interrupt from GPIO GROUP 5
27	Interrupt from GPIO GROUP 4
28	Interrupt from GPIO GROUP 3
29	Interrupt from GPIO GROUP 2
30	Interrupt from GPIO GROUP 1
31	Interrupt from GPIO GROUP 0
32	Interrupt from PCMCIA
33	Interrupt from EXTPORT
34	Interrupt from CAMERA
35	-

36	Interrupt from LCD Controller
37	Interrupt from LCD2D
38	Interrupt from CANBUS0
39	Interrupt from CANBUS1
40	Interrupt from USP4
41	Interrupt from USP5
42	Interrupt from USP6
43	Interrupt from USP7
44	Interrupt from TIMER1
45	Interrupt from TIMER2
46	Interrupt from TIMER3
47	Interrupt from TIMER4
48	Interrupt from TIMER5

NOTE:

1. The INT_PENDING, INT_IRQ_PENDING, INT_FIQ_PENDING, INT_DSP0_PENDING, INT_DSP1_PENDING register are all read only. They only perform some combinational logic of interrupt inputs from each internal block. To clear the interrupt, user need to clear the interrupt in the block, this generates the interrupt. These registers will change as soon as the interrupt signal from the internal block changes.
2. The INT_FIQ_PENDING, register are the result from INT_PENDING & INT_RISC_MASK & (INT_RISC_LEVEL)
The INT_DSP1_PENDING, register are the result from INT_PENDING & INT_DSP_MASK & (INT_DSP_LEVEL)
So if user changes the value in other register, the INT_FIQ_PENDING will also change. This needs special care in interrupt routine. If in interrupt routine user write the INT_RISC_MASK register to mask the interrupt from certain block, the INT_FIQ_PENDING will change to 0 accordingly. User can not read this register to determine the interrupt status. Instead user should read INT_PENDING register.
3. The INT_IRQ_PENDING, register are the result from INT_PENDING & INT_RISC_MASK & (~INT_RISC_LEVEL)
The INT_DSP0_PENDING, register are the result from INT_PENDING & INT_DSP_MASK & (~INT_DSP_LEVEL)
So if user changes the value in other register, the INT_IRQ_PENDING will also change. This needs special care in interrupt routine. If in interrupt routine user writes the INT_RISC_MASK register to mask the interrupt from certain block, the INT_IRQ_PENDING will change to 0 accordingly. User can not read this register to determine the interrupt status. Instead user should read INT_PENDING register.
4. The INT_PENDING register contain different information for DSP and RISC. The bit that are different are listed below:

INT_PENDING bit	RISC read	DSP read
IP0	GPIO interrupt0 for RISC	GPIO interrupt0 for DSP
IP1	GPIO interrupt1 for RISC	GPIO interrupt1 for DSP
IP10	DSP interrupt RISC	RISC interrupt DSP
IP29	OS TIMER1-5 interrupt & TIMERMUX_RISC_MASK	OS TIMER1-5 interrupt & TIMERMUX_DSP_MASK

5.5 OS Timer

5.5.3 Overview

There are 6 programmable OS timers in the Atlas™-II. User can program them independently. All 6 OS timers are actually generated from a single 64-bit counter (TIMER_COUNTER). The counter is clocked by a divided clock from system clock. User can configure the frequency of this divided clock by programming TIMER_DIV register. If the value of the TIMER_COUNTER matches either one of the 6 timers, there will be an interrupt generated to the RISC if the corresponding enable bit is set.

The 6th timer can also act as a Watchdog timer when the Watchdog mode is enabled. In this mode, if this timer matches the TIMER_COUNTER (lower 32-bit), a watchdog match event will be generated and it will cause the reset of most of the on-chip modules.

5.5.4 OS Timer Registers

Table 19. OS Timer Register Mapping

RISC Address <11:0>	Register	Description
0x0000	TIMER_COUNTER_LO	OS Timer Counter Low Register
0x0004	TIMER_COUNTER_HI	OS Timer Counter High Register
0x0008	TIMER_MATCH_0	OS Timer Match Register 0
0x000C	TIMER_MATCH_1	OS Timer Match Register 1
0x0010	TIMER_MATCH_2	OS Timer Match Register 2
0x0014	TIMER_MATCH_3	OS Timer Match Register 3
0x0018	TIMER_MATCH_4	OS Timer Match Register 4
0x001C	TIMER_MATCH_5	OS Timer Match Register 5
0x0020	TIMER_STATUS	OS Timer Status Register
0x0024	TIMER_INT_EN	OS Timer Interrupt Enable Register
0x0028	TIMER_WATCHDOG_EN	OS Timer Interrupt Enable Register
0x002C	TIMER_DIV	OS Timer Division Register
0x0030	TIMER_LATCH	OS Timer Latch Register
0x0034	TIMER_LATCHED_LO	OS Timer Latched Low Register
0x0038	TIMER_LATCHED_HI	OS Timer Latched High Register
Others	-	Reserved

- **OS Timer Counter Low Register (TIMER_COUNTER_LO) – 0x0000**

The OS timer count low register is the low 32-bit of the 64-bit-counter that increments on rising edges of the timer clock. This counter can be read or written at any time. It is recommended that this register to be write-protected through the MMU protection mechanisms.

Bit	Name	Default	Description
31:0 (R/W)	CN<31:0>	32'h0	OS Timer Counter low 32-bit value.

- **OS Timer Counter High Register (TIMER_COUNTER_HI) – 0x0004**

The OS timer count high register is the high 32-bit of the 64-bit-counter that increments on rising edges of the timer clock. This counter can be read or written at any time. It is recommended that this register to be write-protected through the MMU protection mechanisms.

Bit	Name	Default	Description
31:0	CN<63:32>	32'h0	OS Timer Counter high 32-bit value.

(R/W)			
-------	--	--	--

- **OS Timer Match Register 0-5 (TIMER_MATCH_x) – 0x0008–0x001C**

These registers are 32 bits wide and are accessible by the RISC. They are compared with the lower 32-bit of TIMER_COUNTER following every timer clock cycle. If any of these registers match the counter at this time, then the corresponding status bit in the TIMER_STATUS is set.

Bit	Name	Default	Description
31:0 (R/W)	MA<31:0>	32'h0	OS Timer match value

- **OS Timer Status Register (TIMER_STATUS) – 0x0020**

These bits are set when the Timer match event occurs and cleared by writing a one to it. Writing zeros to this register has no effect. All reserved bits read as zeros and are unaffected by writes.

Bit	Name	Default	Description
0 (R/W)	M0	1'b0	Match status channel 0. 0 – OS timer match register<0> has not matched the OS timer counter since the last clear. 1 – OS timer match register<0> has matched the OS timer counter.
1 (R/W)	M1	1'b0	Match status channel 1. 0 – OS timer match register<1> has not matched the OS timer counter since the last clear. 1 – OS timer match register<1> has matched the OS timer counter.
2 (R/W)	M2	1'b0	Match status channel 2. 0 – OS timer match register<2> has not matched the OS timer counter since the last clear. 1 – OS timer match register<2> has matched the OS timer counter.
3 (R/W)	M3	1'b0	Match status channel 3. 0 – OS timer match register<3> has not matched the OS timer counter since the last clear. 1 – OS timer match register<3> has matched the OS timer counter.
4 (R/W)	M4	1'b0	Match status channel 4. 0 – OS timer match register<4> has not matched the OS timer counter since the last clear. 1 – OS timer match register<4> has matched the OS timer counter.
5 (R/W)	M5	1'b0	Match status channel 5. 0 – OS timer match register<5> has not matched the OS timer counter since the last clear. 1 – OS timer match register<5> has matched the OS timer counter.
31:6	-	26'h0	Reserved

- **OS Timer Interrupt Enable Register (TIMER_INT_EN) – 0x0024**

Each match register has a corresponding enable bit. Clearing an enable bit will prevent the corresponding interrupt status bit be generated.

Bit	Name	Default	Description
0 (R/W)	E0	1'b0	Interrupt enable channel 0. This bit is set by software and allows a match between match register 0 and the OS timer to assert interrupt bit M0 in the TIMER_STATUS.
1 (R/W)	E1	1'b0	Interrupt enable channel 1.
2 (R/W)	E2	1'b0	Interrupt enable channel 2.
3 (R/W)	E3	1'b0	Interrupt enable channel 3.
4 (R/W)	E4	1'b0	Interrupt enable channel 4.
5 (R/W)	E5	1'b0	Interrupt enable channel 5.
31:6	-	26'h0	Reserved

- **OS Timer Watchdog Enable Register (TIMER_WATCHDOG_EN) – 0x0028**

Bit	Name	Default	Description
0 (R/W)	WME¹	1'b0	Watchdog match enable. 0 – OS timer match register<3> matches cause an interrupt request. 1 – OS timer match register<3> matches cause a reset.
31:1	-	31'h0	Reserved

- **OS Timer Division Register (TIMER_DIV) – 0x002C**

To generate the timer clock, it needs to divide the clock input from system clock domain.

$$\text{TIMER_DIV} = (\text{SYS_CLK}/\text{TIMER_CLK})/2 - 1$$

Bit	Name	Default	Description
15:0 (R/W)	DIV<15:0>	16'h0	OS Timer clock division.
31:16	-	16'h0	Reserved

- **OS Timer Latch Register (TIMER_LATCH) – 0x0030**

When user writes a 1'b1 into this register bit, the value of the current OS timer counter will be latched into the TIMER_LATCHED_LO & TIMER_LATCHED_HI registers.

Bit	Name	Default	Description
0 (W)	LATCH	1'b0	OS Timer Counter latch.
31:1 (R/W)	-	31'h0	Reserved

- **OS Timer Latched Low Register (TIMER_LATCHED_LO) – 0x0034**

The OS timer latched low register is the low 32-bit value of the 64-bit-counter that latched when user write a 1'b1 into TIMER_LATCH register.

Bit	Name	Default	Description
31:0 (R/W)	LA<31:0>	32'h0	The low 32-bit latched value.

¹ **Note:** This is a write-once bit that once written, can only be changed after a hardware, software or sleep mode reset.

- **OS Timer Latched High Register (TIMER_LATCHED_HI) – 0x0038**

The OS timer latched high register is the high 32-bit value of the 64-bit-counter that latched when user write a 1'b1 into TIMER_LATCH register.

Bit	Name	Default	Description
31:0 (R/W)	LA<63:32>	32'h0	The high 32-bit latched value.

5.6 Reset Controller

5.6.5 Overview

The reset controller manages the various reset sources within the Atlas-II. There are 4 types of resets:

- **Hardware-reset**

Hardware reset is asserted through the RESET_B pin that is intended to be used for power-on only. It's a full-chip reset so that every registers on-chip will be reset to its default value and the content in the SDRAM will be lost.

- **Software-reset**

There is a RESET_SWR register which can be programmed by user to reset most of the blocks on-chip. User can select to apply reset to each block of the Atlas-II™ separately. The assertion of system-reset bit (bit31: SYS_RST) will reset the majority of Atlas-II™ as well as causing the assertion of the RESET_OUT pin. The Software-reset will not cause the SDRAM data loss.

- **Sleep-reset**

Sleep reset is generated automatically by hardware when the Atlas-II™ enters sleep mode. During sleep mode, the majority of the processor will be reset except the power manager, RTC, Resource Sharing Controller and GPIO controller. Before sleep reset, the SDRAM controller will issue a self-refresh command to the SDRAM so that the data will not be lost during sleep mode.

After boot-up, software can check the reset controller reset status register (RESET_STATUS) to determine which types of reset has just occurred.

5.6.6 Reset Scheme

Each block in Atlas-II™ has its own reset control signal. As described above, the reset has 4 sources: Hardware-reset, Software-reset, Watchdog-reset, and Sleep-reset. The following table shows that how each module is controlled by these 4 types of reset sources.

Table 20. Reset Scheme of Each Module

Module	Hardware reset	System reset ¹	Sleep reset	Block reset ²
RISC Core	Yes	Yes	Yes	No
RISC Interface	Yes	Yes	Yes	No
Interrupt Controller	Yes	Yes	Yes	No
Power Manager	Yes	No	No	No

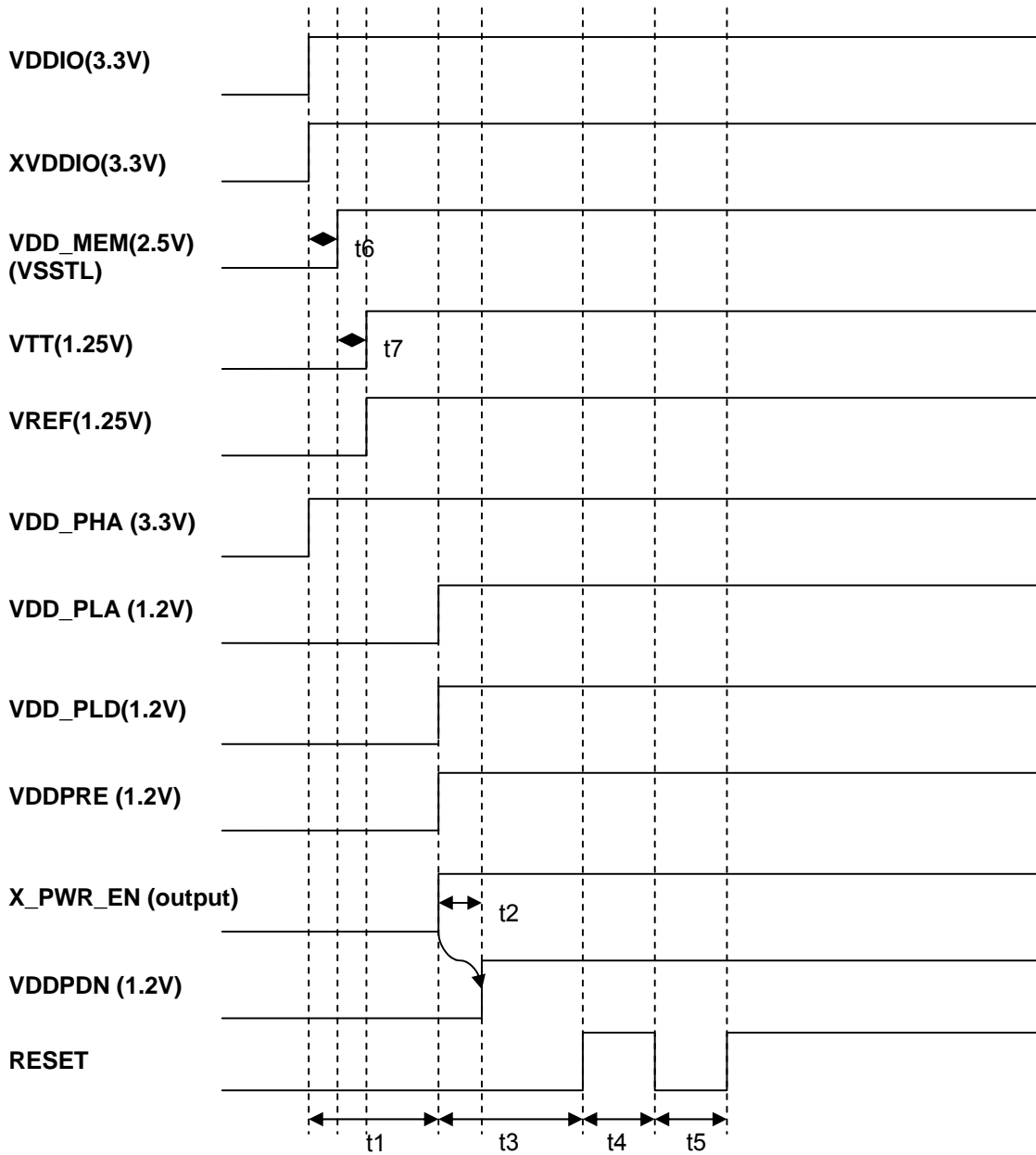
¹ System reset has two sources: first, invoked by programming SYS_RST bit of RESET_SWR register; second, invoked by watchdog timer.

² Block reset is invoked by programming bit<23:0> of RESET_SWR register.

RTC	Yes	No	No	No
Timer	Yes	Yes	Yes	No
Reset Controller	Yes	No	No	No
GPIO	Yes	No	No	No
Resource Sharing Controller	Yes	No	No	No
System Arbiter	Yes	Yes	Yes	No
Others	Yes	Yes	Yes	Yes

5.6.7 Power On Reset

The Atlas-II™ power on reset needs to hold long enough that the SoC can be initialized correctly. And the power on reset should not be disserted until the 12 MHz oscillator is stable.



t1>10ms t2>0s (*) t3>50ms (**) t4>3ms t5 >2ms t6 > 0s t7>0s
 * t2 is depending on the external power control circuit
 ** t3 must be long enough until 12MHz Clock is stable

5.6.8 Reset Controller Registers

Table 21. Reset Controller Register Mapping

RISC Address <11:0>	Register	Description
0x0000	RESET_SR	Reset Controller Software Reset Register
0x0004	RESET_STATUS	Reset Controller Status Register
Others	-	Reserved

- **Reset Controller Software Reset Register (RESET_SWR) – 0x0000**

Writing a one to each bit from bit30 to bit0 causes the corresponding block to be reset. Writing a one to the most-significant bit (SYS_RST) causes all on-chip resources to reset except the Power Manager, RTC, Memory controller, Reset controller, Resource Sharing Controller and GPIO controller. Writing zero to the software-reset bit will clear the corresponding reset. Care should be taken to restrict access to this register by programming MMU permissions. For reserved bits, writes have no effect.

This register will be reset to default value by hardware reset only.

Bit	Name	Default	Description
0 (R/W)	DSP_RST	1'b0	DSP Core software reset. 1 – Reset. (The following are the same.)
1 (R/W)	DIFACE_RST	1'b0	DSP Interface software reset.
2 (R/W)	ROM_RST	1'b0	ROM Controller software reset.
3 (R/W)	PCI_RST	1'b0	PCI software reset.
4 (R/W)	DMA_RST	1'b0	DMA Controller software reset.
5 (R/W)	LCD_RST	1'b0	LCD Controller software reset.
6 (R/W)	IDE_RST	1'b0	IDE software reset.
7 (R/W)	GPS_RST	1'b0	GPS software reset.
8 (R/W)	USB_RST	1'b0	USB-OTG software reset.
9 (R/W)	SM_RST	1'b0	Smartmedia/NAND Flash software reset.
10 (R/W)	PCMCIA_RST	1'b0	PCMCIA Core software reset.
11	-	1'b0	Reserved
12 (R/W)	CAM_RST	1'b0	Video Input Port software reset.
13 (R/W)	CODEC_RST	1'b0	Audio CODEC software reset.
14 (R/W)	SP0_RST	1'b0	Serial Port 0 software reset.
15 (R/W)	SP1_RST	1'b0	Serial Port 1 software reset.
16 (R/W)	SP2_RST	1'b0	Serial Port 2 software reset.
17 (R/W)	SP3_RST	1'b0	Serial Port 3 software reset.
18 (R/W)	SP4_RST	1'b0	Serial Port 4 reset.
19 (R/W)	SP5_RST	1'b0	Serial Port 5 reset.
20	-	1'b0	Reserved
21 (R/W)	SDIO_RST	1'b0	SDIO software reset.
22 (R/W)	PWM_RST	1'b0	PWM software reset.
23 (R/W)	MBUS2PCI_RST	1'b0	MBUS2PCI software reset.
24 (R/W)	USP6_RST	1'b0	USP6 software reset.
25 (R/W)	USP7_RST	1'b0	USP7 software reset.
26 (R/W)	COPY_RST	1'b0	PCI copy blocks software reset.
27 (R/W)	PADARB_RST	1'b0	Pad arbitration software reset.
28 (R/W)	CAN0_RST	1'b0	CAN Bus 0 software reset.

29 (R/W)	CAN1_RST	1'b0	CAN Bus 1 software reset.
30 (R/W)	HWACC_RST	1'b0	Hardware Accelerators (YUV_RGB, YUV_CHG, and IPOLATE) software reset.
31 (R/W)	SYS_RST	1'b0	System software reset (including all blocks except the Power Manager, Resource Sharing Controller, Real Time Clock, and GPIO.)

- **Reset Controller Status Register (RESET_STATUS) – 0x0004**

User can use the reset controller reset status register (RESET_STATUS) to determine the last cause or causes of the reset.

Each RESET_STATUS status bit is set by a different source of reset, and can be cleared by writing a zero back to that bit. After hardware reset, all the status bits are zero. For reserved bits, writes are ignored and reads return zero.

Bit	Name	Default	Description
0 (R/W)	HWR	1'b0	Hardware reset. 0 – Hardware reset has not occurred since the last time the CPU cleared this bit. 1 – Hardware reset has occurred since the last time the CPU cleared this bit.
1 (R/W)	SWR	1'b0	Software reset. 0 – Software reset has not occurred since the last time the CPU cleared this bit. 1 – Software reset has occurred since the last time the CPU cleared this bit.
2 (R/W)	WDR	1'b0	Watchdog reset. 0 – Watchdog reset has not occurred since the last time the CPU cleared this bit. 1 – Watchdog reset has occurred since the last time the CPU cleared this bit.
31:3	-	29'h0	Reserved

5.7 Resource Sharing Controller

5.7.9 DMA Sharing

The Resource Sharing Controller (RSC) manages the two major resources of Atlas-II™: one is the IO pins; the other is the DMA channels. The following table shows the DMA channel sharing:

Table 22. DMA Channels Allocation

DMA Channel #	Shared Functions
0~1	CAN Bus Controller 0 & 1
2	Video Input Port
3	Reserved
4	NAND Flash Controller
5	-
6~7	Audio CODEC Interface
8~11	Audio CODEC Interface, UART0, USP1~5

NOTE: Please refer to the section of “DMA Controller” for more details about the DMA channel sharing.

5.7.10 Pin Sharing

There are following groups of pin sharing:

- **X_BATT_FAULT/ X_VDD_FAULT Pin Mux**

Pin Name (HW Sleep Control)	GPIO
X_BATT_FAULT	GPIO group0 ¹ offset 28
X_VDD_FAULT	GPIO group0 offset 29

These two pins will affect the hardware sleep function. After reset these two pins are not controlled by GPIO and used as input. GPIO can take over the control of these two pins by writing GPIO0_CTRL28, GPIO0_CTRL29. Even in sleep mode GPIO still have the control over these two pins.

NOTE: To use these two pins as GPIO, the connected device can not pull the pin to high after reset, otherwise the chip will detect BATT/VDD fault and enter into sleep mode automatically.

- **GPS Pin Mux**

Pin Name (GPS)	GPIO
X_RF_DATAIN_1	GPIO group2 offset 15
X_RF_DATAIN_0	GPIO group2 offset 16
X_SAMPLE_CLK	GPIO group2 offset 17
X_RF_CLK	GPIO group2 offset 18

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to GPS.

- **LCD Pin Mux**

Pin Name (LCD)	GPIO
X_L_PCLK	GPIO group2 offset 20
X_L_LCK	GPIO group2 offset 21
X_L_FCK	GPIO group2 offset 22
X_L_BIAS	GPIO group2 offset 23
X_LDD_0	
X_LDD_0 ~ X_LDD_7	GPIO group2 offset 24 ~ GPIO group2 offset 31
X_LDD_8 ~ X_LDD_15	GPIO group0 offset 8 ~ GPIO group0 offset 15
X_LDD_16	GPIO group0 offset 24
X_LDD_17	GPIO group0 offset 25

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to GPS. If only 8-bit LCD is used then X_LDD_8~X_LDD_15 can still be used as GPIO.

- **NAND Flash/Video Input Port Pin Mux**

Pin Name	Video Input Port	NAND Flash	GPIO
----------	------------------	------------	------

¹ Please refer to the section of GPIO for detail information about GPIO groups.

X_PXCLK	PXCLK	DF_RY_BY	GPIO group1 offset 18
X_HSYNC	HSYNC	DF_RE_B	GPIO group1 offset 17
X_VSYNC	VSYNC	DF_WE_B	GPIO group1 offset 16
X_PXD_0~ X_PXD_15	PXD_0~ PXD_15	DF_AD<0> ~ DF_AD<7> DF_ALE DF_CLE DF_AD<8> ~ DF_AD<13>	GPIO group1 offset 15~ GPIO group1 offset 0~
X_GPIO_27		DF_AD<14>	GPIO group0 offset 27
X_GPIO_26		DF_AD<15>	GPIO group0 offset 26

After reset NAND Flash will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to GPIO.

The priority of the above shared functions is as following:

- If GPIO has control over these pins, either Video Input Port or NAND Flash can not control the pin.
- If GPIO does not have control over these pins, the **CAM_EN** bit in *RSC_PIN_MUX* register determines whether Video Input Port or NAND Flash has control over these pins.

• **UART0/JTAG Pin Mux**

Pin Name	UART0	JTAG	GPIO
X_GPIO_16	DSR_0	X_TDI	GPIO group0 offset 16
X_GPIO_17	DTR_0	X_TDO	GPIO group0 offset 17
X_GPIO_18	DCD_0	X_TMS	GPIO group0 offset 18
X_GPIO_19		X_NTRST	GPIO group0 offset 19
X_GPIO_20		X_TCK	GPIO group0 offset 20
X_GPIO_21	RI_0	X_RTCK	GPIO group0 offset 21
X_GPIO_22	CTS_0	X_DBGRQ	GPIO group0 offset 22
X_GPIO_23	RTS_0	X_DBGACK	GPIO group0 offset 23
X_TXD_0	TXD_0		GPIO group1 offset 19
X_RXD_0	RXD_0		GPIO group1 offset 20

Depending on the reset status of JTAG_MODE<1:0> (please refer to the section of “Mode Configuration Pins”), if the chip is configured into JTAG_MODE, the JTAG pins will be enabled; GPIO/UART0 functions (except for the last two pins) will be disabled. However, in this mode UART0 can still be used as asynchronous mode, which only needs X_TXD_0, X_RXD_0. Otherwise, after reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to UART0.

• **USP1/CODEC Pin Mux**

Pin Name (USP1)	CODEC	GPIO
X_SCLK_1	BIT_CLK	GPIO group1 offset 21
X_TXD_1	DA_DATA	GPIO group1 offset 22
X_RXD_1	AD_DATA	GPIO group1 offset 23
X_TFS_1	FRAME_SYNC	GPIO group1 offset 24
X_RFS_1		GPIO group1 offset 25

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to USP1/CODEC.

The priority of the above shared functions is as following:

- If GPIO has control over these pins, both USP1/CODEC can not control the pin.
- If GPIO does not have control over these pins, the **CODEC_EN** bit in *RSC_PIN_MUX* register determines whether USP1 or CODEC have control over these pins.

• **USP2/UART6/ETM(Part 1) Pin Mux**

Pin Name (USP2)	UART6	ETM	GPIO
X_SCLK_2		TRACECLK	GPIO group1 offset 26
X_TXD_2		TRACESYNC	GPIO group1 offset 27
X_RXD_2		PIPESTAT<2>	GPIO group1 offset 28
X_TFS_2	X_TXD_6	PIPESTAT<1>	GPIO group1 offset 29
X_RFS_2	X_RXD_6	PIPESTAT<0>	GPIO group1 offset 30

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to USP2/UART6.

The priority of the above shared functions is as following:

- If ETM9 is enabled during debug, ETM9 will have control over these pins no matter which setting it is.
- Else if GPIO has control over these pins, both USP2/UART6 can not control the pin.
- Else if GPIO does not have control over these pins, the **USP_EN<0>** bit in *RSC_PIN_MUX* register determines whether USP2 or UART6 have control over the two last pins (the other pins are controlled by USP2).

• **USP3/UART7 Pin Mux**

Pin Name (USP3)	UART7	GPIO
X_SCLK_3		GPIO group2 offset 0
X_TXD_3		GPIO group2 offset 1
X_RXD_3		GPIO group2 offset 2
X_TFS_3	X_TXD_7	GPIO group2 offset 3
X_RFS_3	X_RXD_7	GPIO group2 offset 4

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to USP3/UART7.

The priority of the above shared functions is as following:

- If GPIO has control over these pins, both USP3/UART7 can not control the pin.
- Else if GPIO does not have control over these pins, the **USP_EN<1>** bit in *RSC_PIN_MUX* register determines whether USP3 or UART7 have control over the two last pins (the other pins are controlled by USP3).

• **USP4, USP5/SD/PCMCIA (Part 1) Pin Mux**

Pin Name (USP4, 5)	SDIO	PCMCIA	GPIO
X_SCLK_4	SD_DAT<3>	PA<23>	GPIO group2 offset 5
X_TXD_4			GPIO group2 offset 6
X_RXD_4			GPIO group2 offset 7
X_TFS_4	SD_DAT<3>	PA<22>	GPIO group2 offset 8
X_RFS_4	SD_DAT<3>	PA<21>	GPIO group2 offset 9
X_SCLK_5	SD_CLK	PA<25>	GPIO group2 offset 10
X_TXD_5			GPIO group2 offset 11

X_RXD_5			GPIO group2 offset 12
X_TFS_5	SD_CMD	PA<24>	GPIO group2 offset 13
X_RFS_5	SD_DAT<3>	PA<20>	GPIO group2 offset 14

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to USP4, USP5/SD/PCMCIA.

The priority of above shared function is as following:

- If GPIO has control over these pins, both USP4, USP5/SD/PCMCIA can not control the pin.
- Else if GPIO does not have control over these pins, the **SD_EN<1:0>** bit in **RSC_PIN_MUX** register determines whether USP4, USP5 or SD or PCMCIA has control over the shared pins (X_TXD_4, X_RXD_4, X_TXD_5, X_RXD_5 are controlled by USP4, 5).

• **ROM(Part 1)/NAND Flash(Part 1)/IDE(Part 1)/PCMCIA(Part 2)/ETM (Part 2) Pin Mux**

Pin Name (ROM)	NAND Flash	IDE	PCMCIA	ETM	GPIO
X_FA_25			PC_OE_B		GPIO group3 offset 9
X_FA_24			PC_WE_B		GPIO group3 offset 8
X_FA_23		IDE_IOW_B	PC_IOWE_B		GPIO group3 offset 7
X_FA_22		IDE_IOR_B	PC_IORD_B		GPIO group3 offset 6
X_FA_21	DF_CLE				GPIO group3 offset 5
X_FA_20	DF_ALE				GPIO group3 offset 4
X_FA_19			PA<19>		GPIO group3 offset 3
X_FA_18			PA<18>		GPIO group3 offset 2
X_FA_17			PA<17>		GPIO group3 offset 1
X_FA_16			PA<16>		GPIO group3 offset 0
X_FA_15			PA<15>		GPIO group4 offset 15
X_FA_14			PA<14>		GPIO group4 offset 14
X_FA_13			PA<13>		GPIO group4 offset 13
X_FA_12			PA<12>		GPIO group4 offset 12
X_FA_11			PA<11>		GPIO group4 offset 11
X_FA_10			PA<10>		GPIO group4 offset 10
X_FA_9			PA<9>		GPIO group4 offset 9
X_FA_8			PA<8>		GPIO group4 offset 8
X_FA_7			PA<7>		GPIO group4 offset 7
X_FA_6			PA<6>		GPIO group4 offset 6
X_FA_5			PA<5>		GPIO group4 offset 5
X_FA_4			PA<4>		GPIO group4 offset 4
X_FA_3			PA<3>		GPIO group4 offset 3
X_FA_2		IDE_A<2>	PA<2>		GPIO group4 offset 2
X_FA_1		IDE_A<1>	PA<1>		GPIO group4 offset 1
X_FA_0		IDE_A<0>	PA<0>		GPIO group4 offset 0
X_FD_31~ X_FD_16				TRACEPKT <15>~ TRACEPKT <0>	GPIO group5 offset 31~ GPIO group5 offset 16
X_FD_15~ X_FD_0	DF_AD<15> ~ DF_AD<0>	IDE_D<15>~ IDE_D<0>	PD<15>~ PD<0>		GPIO group5 offset 15~ GPIO group5 offset 0
X_FOE_B	DF_RD_B				GPIO group3 offset 31
X_FWE_B	DF_WE_B				GPIO group3 offset 30

After reset GPIO will have no control over these pins. User can write corresponding GPIO control registers to give the control of these pins to GPIO.

The priority of the above functions is as following:

- If ETM9 is enabled during debug, ETM9 will have control over X_FD_31~X_FD_16 no matter which setting it is.

- Else if GPIO has control over these pins, only GPIO can control the pin.
- Else the pin is distributed automatically between ROM/NAND Flash/IDE/PCMCIA. Bit 4~0 in *RSC_ROMPAD_MUX* register controls whether that block will be included in the distribution.

Notes: Since PCMCIA port will not co-exist, Bit 0 and 3 in *RSC_ROMPAD_MUX* can not be 1 at the same time.

• **ROM(Part 2)/NAND Flash(Part 2)/IDE(Part 2) Mux**

Pin Name	ROM	NAND Flash	IDE	GPIO
X_FCE_B_3	X_FCE_B_3	DF_CS_B<3>		GPIO group4 offset 21
X_FCE_B_2	X_FCE_B_2	DF_CS_B<2>		GPIO group4 offset 20
X_FCE_B_1	X_FCE_B_1	DF_CS_B<1>		GPIO group4 offset 19
X_FCE_B_0	X_FCE_B_0	DF_CS_B<0>		-
X_FBE_3	X_FBE_3			GPIO group3 offset 29
X_FBE_2	X_FBE_2			GPIO group3 offset 28
X_FBE_1	X_FBE_1			GPIO group3 offset 27
X_FBE_0	X_FBE_0			GPIO group3 offset 26
X_FRDY_B	X_FRDY_B		IDE_IORDY_B	GPIO group3 offset 25
X_DF_RY_BY		DF_RY_BY		GPIO group3 offset 10
X_IDE_CS_B_1			IDE_CS_B_1	GPIO group4 offset 26
X_IDE_CS_B_0			IDE_CS_B_0	GPIO group4 offset 25
X_IDE_DREQ			IDE_DREQ	GPIO group4 offset 24
X_IDE_DACK			IDE_DACK	GPIO group4 offset 23
X_IDE_IRQ			IDE_IRQ	GPIO group4 offset 22

After reset GPIO will have control over pin X_FCE_B_3~0, X_FBE_3~0, and X_FRDY_B. User can write corresponding GPIO control registers to give the control of these pins to GPIO or ROM/NAND Flash/IDE.

The priority of the above functions is as following:

- If GPIO have control over these pins, all others can not control the pin.
- Else
 - For X_FCE_B_3~0, **NAND_CS<3:0>** in *RSC_PIN_MUX* register determines either ROM or NAND Flash control these pins.
 - For X_FRDY_B, since this pin is open-drain in the corresponding device, it can be used with ROM and IDE co-existing.
 - For other pins they will be used by the corresponding device.

• **EXT(Part 2)/PCMCIA (Part 3) Pin Mux**

Pin Name (PCMCIA)	EXT Port	GPIO
X_PC_VS_2	EXT_BE<3>	GPIO group3 offset 24
X_PC_VS_1	EXT_BE<2>	GPIO group3 offset 23
X_PC_CD_B_1		GPIO group3 offset 22
X_PC_CD_B_0		GPIO group3 offset 21
X_PC_SPKR_B	EXT_GNT_B	GPIO group3 offset 20
X_PC_STSCHG		GPIO group3 offset 19
X_PC_IOIS16_B	EXT_REQ	GPIO group3 offset 18
X_PC_WAIT_B	EXT_RDY_B	GPIO group3 offset 17
X_PC_INPACK_B	EXT_CLK	GPIO group3 offset 16
X_PC_REG_B	EXT_SEL_B	GPIO group3 offset 15
X_PC_REG_B	EXT_SEL_B	GPIO group3 offset 15

X_PC_CE_B_1	EXT_BE<1>	GPIO group3 offset 14
X_PC_CE_B_0	EXT_BE<0>	GPIO group3 offset 13
X_PC_IREQ_B	EXT_IRQ_B	GPIO group3 offset 12
X_PC_RESET	EXT_RST_B	GPIO group3 offset 11

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to EXT /PCMCIA.

The priority of the above functions is as following:

- If GPIO has control over these pins, all others can not control the pin.
 - Else if **PCMCIA_EN** is 1 in **RSC_ROMPAD_MUX** register PCMCIA has control over these pins.
 - Else EXT port has control over these pins
- **X_PAD_RQ/X_PAD_GNT Pin Mux**

Pin Name	GPIO
X_PAD_GNT	GPIO group4 offset 16
X_PAD_RQ	GPIO group4 offset 17

After reset GPIO will have control over these pins. User can write corresponding GPIO control registers to give the control of these pins to X_PAD_GNT / X_PAD_RQ.

The priority of above functions is as following:

- If GPIO have control over these pins, all others can not control the pin.
- Else X_PAD_GNT / X_PAD_RQ will take over the pin. (This function is for extension usage for future device).

NOTE: As you may notice, NAND Flash has two options for pin sharing: it can be either shared with Video Input Port; or it can be shared with ROM I/F. It's determined by some of the mode configuration pins when chip is powered on. Please refer to the section of "Mode Configuration Pins" for more details.

Some of the pins are shared statically, which means the pins will be dedicated to a certain function once the configuration bit is set. But some other pins are shared dynamically, which means the pins will be dynamically switched between different functions automatically.

The ROM I/F pins shared with NAND Flash, PCMCIA/CF, and IDE belong to the later: dynamic sharing pins. There is internal arbitration logic to determine which function can take the control of these pins. But these pins can also be programmed to be statically sharing pins, once the user set the corresponding 'PARK' bit in **RSC_ROMPAD_MUX** register.

5.7.11 Resource Sharing Controller Registers

Table 23. RSC Register Mapping

RISC Address <11:0>	Register	Description
0x0000	-	Reserved
0x0004	RSC_PIN_MUX	RSC Pin Multiplex Register
0x0008	RSC_DMA_MUX	RSC DMA Multiplex Register
0x000C	RSC_ROMPAD_MUX	RSC ROM Pad Multiplex Register
Others	-	Reserved

- **Pin Multiplex Register (RSC_PIN_MUX) – 0x4**

Bit	Name	Default	Description
0 (R/W)	CAM_EN	1'b0	Video Input Port enable 1 – Enabled (Video Input Port takes the pins) 0 – Disabled (NAND Flash takes the pins)
1 (R/W)	CODEC_EN	1'b0	CODEC interface enable 1 – Enabled (CODEC takes the pins) 0 – Disabled (USP takes the pins)
3:2 (R/W)	SD_EN<1:0>	1'b0	SD interface enable 2'b00 – USP takes the pins 2'b01 – SDIO takes the pins 2'b1X – PCMCIA (PA<25:20>) takes the pins
5:4	-	2'b00	Reserved
6 (R/W)	USP_EN<0>	1'b0	USP6 enable 1 – Enabled (USP6 takes the pins) 0 – Disabled (USP2 takes the pins)
7 (R/W)	USP_EN<1>	1'b0	USP7 enable 1 – Enabled (USP7 takes the pins) 0 – Disabled (USP3 takes the pins)
11:8 (R/W)	NAND_CS<3:0>	4'b000X ¹	NAND Flash Chip Select enable 1 – Enabled (NAND Flash takes the CS pin) 0 – Disabled (ROM takes the CS Pin)
31:12	-	20'h0	Reserved

- **DMA Multiplex Register (RSC_DMA_MUX) – 0x8**

DMA Channel 8 & 10 are only for DMA read (peripheral to memory); and DMA Channel 9 & 11 are only for DMA write (memory to peripheral).

Bit	Name	Default	Description																		
2:0 (R/W)	CH8_EN<2:0>	3'b000	DMA Channel 8 enable: <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 50%;">CH8_EN<2:0></th> <th style="width: 50%;">DMA channel 8 is used for:</th> </tr> </thead> <tbody> <tr><td>3'b000</td><td>USP0</td></tr> <tr><td>3'b001</td><td>USP1</td></tr> <tr><td>3'b010</td><td>USP2</td></tr> <tr><td>3'b011</td><td>USP3</td></tr> <tr><td>3'b100</td><td>SIB</td></tr> <tr><td>3'b101</td><td>USP4</td></tr> <tr><td>3'b110</td><td>USP5</td></tr> <tr><td>3'b111</td><td>AUX</td></tr> </tbody> </table>	CH8_EN<2:0>	DMA channel 8 is used for:	3'b000	USP0	3'b001	USP1	3'b010	USP2	3'b011	USP3	3'b100	SIB	3'b101	USP4	3'b110	USP5	3'b111	AUX
CH8_EN<2:0>	DMA channel 8 is used for:																				
3'b000	USP0																				
3'b001	USP1																				
3'b010	USP2																				
3'b011	USP3																				
3'b100	SIB																				
3'b101	USP4																				
3'b110	USP5																				
3'b111	AUX																				
5:3 (R/W)	CH9_EN<2:0>	3'b000	DMA Channel 9 enable (same as above)																		
8:6 (R/W)	CH10_EN<2:0>	3'b011	DMA Channel 10 enable (same as above)																		
11:9 (R/W)	CH11_EN<2:0>	3'b011	DMA Channel 11 enable (same as above)																		
31:12	-	20'h0	Reserved																		

- **ROM Pad Multiplex Register (RSC_ROMPAD_MUX) – 0xC**

Bit	Name	Default	Description
0 (R/W)	PCMCIA_EN	1'b0	PCMCIA interface enable

¹ NAND_CS<0>'s default value is defined by sampling input pad during reset. If the pad is pulled low, then its default value is 1'b0.

			1 – Enabled (PCMCIA can access ROM pads) 0 – Disabled (PCMCIA cannot access ROM pads)
1 (R/W)	ROM_EN	1'b1	ROM interface enable 1 – Enabled 0 – Disabled
2 (R/W)	SM_EN	1'b1	SmartMedia/NAND Flash interface enable 1 – Enabled 0 – Disabled
3	-	1'b0	Reserved
4 (R/W)	IDE_EN	1'b0	IDE interface enable 1 – Enabled 0 – Disabled
5 (R/W)	EXTPAD_EN	1'b0	External companion chip enable 1 – Enabled 0 – Disabled
31:6	-	26'h0	Reserved

NOTE: All reserved register bits have to be written with 0. Otherwise, there will be unexpected results.

5.8 GPIO

5.8.12 Overview

In Atlas-II™, there are a total of 188 GPIO. Each of the GPIO pin can be configured as input, output, or open-drain pin independently. When a GPIO is configured as input, it can be also enabled as an interrupt source (edge or level triggered). Most of the GPIO pins have optional internal pull-up resistor (75K ohm); only a few have optional pull-down resistor. And most of the GPIO pins are multiplexed with other functional pins.

All GPIO pins are grouped into 6 groups. Each group has 32 GPIO's. And there are 32 control registers for each GPIO functional control; 1 register for pad multiplex control; 1 for DSP access control (only valid for group 0); 1 for interrupt status; and 1 for sleep mode status.

The following table shows the Pin Mux of each GPIO:

Table 24. GPIO Pin Mux

Group #	GPIO #	Pin Name
0	27~0	X_GPIO<27:0>
	28	X_BATT_FAULT
	29	X_VDD_FAULT
	30	X_SDA
	31	X_SCL
1	15~0	X_PXD<15:0>
	16	X_VSYNC
	17	X_HSYNC
	18	X_PXCLK
	19	X_TXD_0
	20	X_RXD_0
	21	X_SCLK_1
	22	X_TXD_1

	23	X_RXD_1
	24	X_TFS_1
	25	X_RFS_1
	26	X_SCLK_2
	27	X_TXD_2
	28	X_RXD_2
	29	X_TFS_2
	30	X_RFS_2
2	0	X_SCLK_3
	1	X_TXD_3
	2	X_RXD_3
	3	X_TFS_3
	4	X_RFS_3
	5	X_SCLK_4
	6	X_TXD_4
	7	X_RXD_4
	8	X_TFS_4
	9	X_RFS_4
	10	X_SCLK_5
	11	X_TXD_5
	12	X_RXD_5
	13	X_TFS_5
	14	X_RFS_5
	15	X_RF_DATA_IN1
	16	X_RF_DATA_IN0
	17	X_SAMPLE_CLK
	18	X_RF_CLK
	20	X_L_PCLK
	21	X_L_LCLK
	22	X_L_FCK
	23	X_L_BIAS
	31~24	X_LDD<7:0>
3	9~0	X_FA<25:16>
	10	X_DF_RY_BY
	11	X_PC_RESET
	12	X_PC_IREQ_B
	13	X_PC_CE_B_0
	14	X_PC_CE_B_1
	15	X_PC_REG_B
	16	X_PC_INPACK_B
	17	X_PC_WAIT_B
	18	X_PC_IOIS16_B
	19	X_PC_STSCHG
	20	X_PC_SPKR_B
	21	X_PC_CD_B_0
	22	X_PC_CD_B_1
	23	X_PC_VS_1
	24	X_PC_VS_2
	25	X_FRDY_B

	26	X_FBE_0
	27	X_FBE_1
	28	X_FBE_2
	29	X_FBE_3
	30	X_FWE_B
	31	X_FOE_B
4	15~0	X_FA<15:0>
	16	X_PAD_GNT
	17	X_PAD_RQ
	21~18	X_FCE_B<3:0>
	22	X_IDE_IRQ
	23	X_IDE_DACK
	24	X_IDE_DREQ
	25	X_IDE_CS_B_0
	26	X_IDE_CS_B_1
	27	X_CAN_TXD_0
	28	X_CAN_RXD_0
5	29	X_CAN_TXD_1
	30	X_CAN_RXD_1
	31~0	X_FD<31:0>

5.8.13 GPIO Registers

Use the GPIO Pin Output Enable bit (**OUT_EN**) to set whether the GPIO pins are outputs or inputs. When programmed as an output, the pin can be set high by writing a '1' to the corresponding bit in the GPIO Pin Data output bit (**OUT**) and cleared by writing a '0' to the same register. The GPIO Pin Data output register can be written regardless of its input or output configuration.

If a pin is configured as an input, the programmed output state occurs when the pin is reconfigured to be an output. The user can validate each GPIO pin's state by reading the GPIO Pin data input bit (**DATA_IN**). This register can be read anytime to confirm the state of a pin. The GPIO pull high/low bit (**PULL**) is used to configure the GPIO input pin to pull it high or pull low. Use the GPIO Open Drain bit (**OD**) to set the GPIO into Open Drain mode.

The GPIO supports two types of interrupts: edge or level triggered. The user should set the GPIO interrupt type bit (**INT_TYPE**) to choose. Uses the GPIO interrupt enable bit (**INT_EN**) to enable the interrupt. Use the GPIO high level trigger bit (**INT_HT**) to enable the high level or rising edge interrupt as defined in *GPIO_INT_TYPE*. Uses the GPIO low-level trigger bit (**INT_LT**) to enable the low level or falling edge interrupt as defined in *GPIO_INT_TYPE*. Use the GPIO interrupt status bit (**INT_STATUS**) to read interrupt happened on the GPIO.

For each group GPIO there are 31 or 32 control registers for each GPIO pin, 3-4 registers to control PAD MUX, DSP enable, read interrupt status, and sleep mode.

The registers are mapped as the following table:

Table 25. GPIO Register Mapping

RISC Address <11:0>	DSP Address <6:0>	Register	Description
0x000	0x00	GPIO0_CTRL0	GPIO control for GPIO0

0x004	0x01	GPIO0_CTRL1	GPIO control for GPIO1
0x008	0x02	GPIO0_CTRL2	GPIO control for GPIO2
0x00C	0x03	GPIO0_CTRL3	GPIO control for GPIO3
0x010	0x04	GPIO0_CTRL4	GPIO control for GPIO4
0x014	0x05	GPIO0_CTRL5	GPIO control for GPIO5
0x018	0x06	GPIO0_CTRL6	GPIO control for GPIO6
0x01C	0x07	GPIO0_CTRL7	GPIO control for GPIO7
0x020	0x08	GPIO0_CTRL8	GPIO control for GPIO8
0x024	0x09	GPIO0_CTRL9	GPIO control for GPIO9
0x028	0x0A	GPIO0_CTRL10	GPIO control for GPIO10
0x02C	0x0B	GPIO0_CTRL11	GPIO control for GPIO11
0x030	0x0C	GPIO0_CTRL12	GPIO control for GPIO12
0x034	0x0D	GPIO0_CTRL13	GPIO control for GPIO13
0x038	0x0E	GPIO0_CTRL14	GPIO control for GPIO14
0x03C	0x0F	GPIO0_CTRL15	GPIO control for GPIO15
0x040	0x10	GPIO0_CTRL16	GPIO control for GPIO16
0x044	0x11	GPIO0_CTRL17	GPIO control for GPIO17
0x048	0x12	GPIO0_CTRL18	GPIO control for GPIO18
0x04C	0x13	GPIO0_CTRL19	GPIO control for GPIO19
0x050	0x14	GPIO0_CTRL20	GPIO control for GPIO20
0x054	0x15	GPIO0_CTRL21	GPIO control for GPIO21
0x058	0x16	GPIO0_CTRL22	GPIO control for GPIO22
0x05C	0x17	GPIO0_CTRL23	GPIO control for GPIO23
0x060	0x18	GPIO0_CTRL24	GPIO control for GPIO24
0x064	0x19	GPIO0_CTRL25	GPIO control for GPIO25
0x068	0x1A	GPIO0_CTRL26	GPIO control for GPIO26
0x06C	0x1B	GPIO0_CTRL27	GPIO control for GPIO27
0x070	0x1C	GPIO0_CTRL28	GPIO control for GPIO28
0x074	0x1D	GPIO0_CTRL29	GPIO control for GPIO29
0x078	0x1E	GPIO0_CTRL30	GPIO control for GPIO30
0x07C	0x1F	GPIO0_CTRL31	GPIO control for GPIO31
0x080	-	GPIO0_DSP_EN	GPIO group0 DSP control enable
0x084	-	GPIO0_PAD_EN	GPIO group0 PAD enable
0x088	-	GPIO0_SW_EN	GPIO group0 sleep wakeup enable
0x08C	-	GPIO0_INT_STATUS	GPIO group0 interrupt status for RISC
0x90~0xFC	-	-	Reserved.
0x100	-	GPIO1_CTRL0	GPIO control for GPIO32
0x104	-	GPIO1_CTRL1	GPIO control for GPIO33
0x108	-	GPIO1_CTRL2	GPIO control for GPIO34
0x10C	-	GPIO1_CTRL3	GPIO control for GPIO35
0x110	-	GPIO1_CTRL4	GPIO control for GPIO36
0x114	-	GPIO1_CTRL5	GPIO control for GPIO37
0x118	-	GPIO1_CTRL6	GPIO control for GPIO38
0x11C	-	GPIO1_CTRL7	GPIO control for GPIO39
0x120	-	GPIO1_CTRL8	GPIO control for GPIO40
0x124	-	GPIO1_CTRL9	GPIO control for GPIO41
0x128	-	GPIO1_CTRL10	GPIO control for GPIO42
0x12C	-	GPIO1_CTRL11	GPIO control for GPIO43

0x130	-	GPIO1_CTRL12	GPIO control for GPIO44
0x134	-	GPIO1_CTRL13	GPIO control for GPIO45
0x138	-	GPIO1_CTRL14	GPIO control for GPIO46
0x13C	-	GPIO1_CTRL15	GPIO control for GPIO47
0x140	-	GPIO1_CTRL16	GPIO control for GPIO48
0x144	-	GPIO1_CTRL17	GPIO control for GPIO49
0x148	-	GPIO1_CTRL18	GPIO control for GPIO50
0x14C	-	GPIO1_CTRL19	GPIO control for GPIO51
0x150	-	GPIO1_CTRL20	GPIO control for GPIO52
0x154	-	GPIO1_CTRL21	GPIO control for GPIO53
0x158	-	GPIO1_CTRL22	GPIO control for GPIO54
0x15C	-	GPIO1_CTRL23	GPIO control for GPIO55
0x160	-	GPIO1_CTRL24	GPIO control for GPIO56
0x164	-	GPIO1_CTRL25	GPIO control for GPIO57
0x168	-	GPIO1_CTRL26	GPIO control for GPIO58
0x16C	-	GPIO1_CTRL27	GPIO control for GPIO59
0x170	-	GPIO1_CTRL28	GPIO control for GPIO60
0x174	-	GPIO1_CTRL29	GPIO control for GPIO61
0x178	-	GPIO1_CTRL30	GPIO control for GPIO62
0x17C~0x180	-	-	Reserved
0x184	-	GPIO1_PAD_EN	GPIO group1 pad enable
0x188	-	-	Reserved
0x18C	-	GPIO1_INT_STATUS	GPIO group1 interrupt status for RISC
0x190~0x1FC	-	-	Reserved
0x200	-	GPIO2_CTRL0	GPIO control for GPIO64
0x204	-	GPIO2_CTRL1	GPIO control for GPIO65
0x208	-	GPIO2_CTRL2	GPIO control for GPIO66
0x20C	-	GPIO2_CTRL3	GPIO control for GPIO67
0x210	-	GPIO2_CTRL4	GPIO control for GPIO68
0x214	-	GPIO2_CTRL5	GPIO control for GPIO69
0x218	-	GPIO2_CTRL6	GPIO control for GPIO70
0x21C	-	GPIO2_CTRL7	GPIO control for GPIO71
0x220	-	GPIO2_CTRL8	GPIO control for GPIO72
0x224	-	GPIO2_CTRL9	GPIO control for GPIO73
0x228	-	GPIO2_CTRL10	GPIO control for GPIO74
0x22C	-	GPIO2_CTRL11	GPIO control for GPIO75
0x230	-	GPIO2_CTRL12	GPIO control for GPIO76
0x234	-	GPIO2_CTRL13	GPIO control for GPIO77
0x238	-	GPIO2_CTRL14	GPIO control for GPIO78
0x23C	-	GPIO2_CTRL15	GPIO control for GPIO79
0x240	-	GPIO2_CTRL16	GPIO control for GPIO80
0x244	-	GPIO2_CTRL17	GPIO control for GPIO81
0x248	-	GPIO2_CTRL18	GPIO control for GPIO82
0x24C	-	GPIO2_CTRL19	GPIO control for GPIO83
0x250	-	GPIO2_CTRL20	GPIO control for GPIO84
0x254	-	GPIO2_CTRL21	GPIO control for GPIO85
0x258	-	GPIO2_CTRL22	GPIO control for GPIO86
0x25C	-	GPIO2_CTRL23	GPIO control for GPIO87

0x260	-	GPIO2_CTRL24	GPIO control for GPIO88
0x264	-	GPIO2_CTRL25	GPIO control for GPIO89
0x268	-	GPIO2_CTRL26	GPIO control for GPIO90
0x26C	-	GPIO2_CTRL27	GPIO control for GPIO91
0x270	-	GPIO2_CTRL28	GPIO control for GPIO92
0x274	-	GPIO2_CTRL29	GPIO control for GPIO93
0x278	-	GPIO2_CTRL30	GPIO control for GPIO94
0x27C	-	GPIO2_CTRL31	GPIO control for GPIO95
0x280	-	-	Reserved
0x284	-	GPIO2_PAD_EN	GPIO group2 pad enable
0x288	-	-	Reserved
0x28C	-	GPIO2_INT_STATUS	GPIO group2 interrupt status for RISC
0x290~0x2FC	-	-	Reserved
0x300	-	GPIO3_CTRL0	GPIO control for GPIO96
0x304	-	GPIO3_CTRL1	GPIO control for GPIO97
0x308	-	GPIO3_CTRL2	GPIO control for GPIO98
0x30C	-	GPIO3_CTRL3	GPIO control for GPIO99
0x310	-	GPIO3_CTRL4	GPIO control for GPIO100
0x314	-	GPIO3_CTRL5	GPIO control for GPIO101
0x318	-	GPIO3_CTRL6	GPIO control for GPIO102
0x31C	-	GPIO3_CTRL7	GPIO control for GPIO103
0x320	-	GPIO3_CTRL8	GPIO control for GPIO104
0x324	-	GPIO3_CTRL9	GPIO control for GPIO105
0x328	-	GPIO3_CTRL10	GPIO control for GPIO106
0x32C	-	GPIO3_CTRL11	GPIO control for GPIO107
0x330	-	GPIO3_CTRL12	GPIO control for GPIO108
0x334	-	GPIO3_CTRL13	GPIO control for GPIO109
0x338	-	GPIO3_CTRL14	GPIO control for GPIO110
0x33C	-	GPIO3_CTRL15	GPIO control for GPIO111
0x340	-	GPIO3_CTRL16	GPIO control for GPIO112
0x344	-	GPIO3_CTRL17	GPIO control for GPIO113
0x348	-	GPIO3_CTRL18	GPIO control for GPIO114
0x34C	-	GPIO3_CTRL19	GPIO control for GPIO115
0x350	-	GPIO3_CTRL20	GPIO control for GPIO116
0x354	-	GPIO3_CTRL21	GPIO control for GPIO117
0x358	-	GPIO3_CTRL22	GPIO control for GPIO118
0x35C	-	GPIO3_CTRL23	GPIO control for GPIO119
0x360	-	GPIO3_CTRL24	GPIO control for GPIO120
0x364	-	GPIO3_CTRL25	GPIO control for GPIO121
0x368	-	GPIO3_CTRL26	GPIO control for GPIO122
0x36C	-	GPIO3_CTRL27	GPIO control for GPIO123
0x370	-	GPIO3_CTRL28	GPIO control for GPIO124
0x374	-	GPIO3_CTRL29	GPIO control for GPIO125
0x378	-	GPIO3_CTRL30	GPIO control for GPIO126
0x37C	-	GPIO3_CTRL31	GPIO control for GPIO127
0x380	-	-	Reserved
0x384	-	GPIO3_PAD_EN	GPIO group3 pad enable
0x388	-	-	Reserved
0x38C	-	GPIO3_INT_STATUS	GPIO group3 Interrupt status for

			RISC
0x390~0x3FC	-	-	Reserved
0x400	-	GPIO4_CTRL0	GPIO control for GPIO128
0x404	-	GPIO4_CTRL1	GPIO control for GPIO129
0x408	-	GPIO4_CTRL2	GPIO control for GPIO130
0x40C	-	GPIO4_CTRL3	GPIO control for GPIO131
0x410	-	GPIO4_CTRL4	GPIO control for GPIO132
0x414	-	GPIO4_CTRL5	GPIO control for GPIO133
0x418	-	GPIO4_CTRL6	GPIO control for GPIO134
0x41C	-	GPIO4_CTRL7	GPIO control for GPIO135
0x420	-	GPIO4_CTRL8	GPIO control for GPIO136
0x424	-	GPIO4_CTRL9	GPIO control for GPIO137
0x428	-	GPIO4_CTRL10	GPIO control for GPIO138
0x42C	-	GPIO4_CTRL11	GPIO control for GPIO139
0x430	-	GPIO4_CTRL12	GPIO control for GPIO140
0x434	-	GPIO4_CTRL13	GPIO control for GPIO141
0x438	-	GPIO4_CTRL14	GPIO control for GPIO142
0x43C	-	GPIO4_CTRL15	GPIO control for GPIO143
0x440	-	GPIO4_CTRL16	GPIO control for GPIO144
0x444	-	GPIO4_CTRL17	GPIO control for GPIO145
0x448	-	GPIO4_CTRL18	GPIO control for GPIO146
0x44C	-	GPIO4_CTRL19	GPIO control for GPIO147
0x450	-	GPIO4_CTRL20	GPIO control for GPIO148
0x454	-	GPIO4_CTRL21	GPIO control for GPIO149
0x458	-	GPIO4_CTRL22	GPIO control for GPIO150
0x45C	-	GPIO4_CTRL23	GPIO control for GPIO151
0x460	-	GPIO4_CTRL24	GPIO control for GPIO152
0x464	-	GPIO4_CTRL25	GPIO control for GPIO153
0x468	-	GPIO4_CTRL26	GPIO control for GPIO154
0x46C	-	GPIO4_CTRL27	GPIO control for GPIO155
0x470	-	GPIO4_CTRL28	GPIO control for GPIO156
0x474	-	GPIO4_CTRL29	GPIO control for GPIO157
0x478	-	GPIO4_CTRL30	GPIO control for GPIO158
0x47C~0x480	-	-	Reserved
0x484	-	GPIO4_PAD_EN	GPIO group4pad enable
0x488	-	-	Reserved
0x48C	-	GPIO4_INT_STATUS	GPIO group4 interrupt status for RISC
0x490~0x4FC	-	-	Reserved
0x500	-	GPIO5_CTRL0	GPIO control for GPIO160
0x504	-	GPIO5_CTRL1	GPIO control for GPIO161
0x508	-	GPIO5_CTRL2	GPIO control for GPIO162
0x50C	-	GPIO5_CTRL3	GPIO control for GPIO163
0x510	-	GPIO5_CTRL4	GPIO control for GPIO164
0x514	-	GPIO5_CTRL5	GPIO control for GPIO165
0x518	-	GPIO5_CTRL6	GPIO control for GPIO166
0x51C	-	GPIO5_CTRL7	GPIO control for GPIO167
0x520	-	GPIO5_CTRL8	GPIO control for GPIO168
0x524	-	GPIO5_CTRL9	GPIO control for GPIO169

0x528	-	GPIO5_CTRL10	GPIO control for GPIO170
0x52C	-	GPIO5_CTRL11	GPIO control for GPIO171
0x530	-	GPIO5_CTRL12	GPIO control for GPIO172
0x534	-	GPIO5_CTRL13	GPIO control for GPIO173
0x538	-	GPIO5_CTRL14	GPIO control for GPIO174
0x53C	-	GPIO5_CTRL15	GPIO control for GPIO175
0x540	-	GPIO5_CTRL16	GPIO control for GPIO176
0x544	-	GPIO5_CTRL17	GPIO control for GPIO177
0x548	-	GPIO5_CTRL18	GPIO control for GPIO178
0x54C	-	GPIO5_CTRL19	GPIO control for GPIO179
0x550	-	GPIO5_CTRL20	GPIO control for GPIO180
0x554	-	GPIO5_CTRL21	GPIO control for GPIO181
0x558	-	GPIO5_CTRL22	GPIO control for GPIO182
0x55C	-	GPIO5_CTRL23	GPIO control for GPIO183
0x560	-	GPIO5_CTRL24	GPIO control for GPIO184
0x564	-	GPIO5_CTRL25	GPIO control for GPIO185
0x568	-	GPIO5_CTRL26	GPIO control for GPIO186
0x56C	-	GPIO5_CTRL27	GPIO control for GPIO187
0x570	-	GPIO5_CTRL28	GPIO control for GPIO188
0x574	-	GPIO5_CTRL29	GPIO control for GPIO189
0x578	-	GPIO5_CTRL30	GPIO control for GPIO190
0x57C	-	GPIO5_CTRL31	GPIO control for GPIO191
0x580	-	-	Reserved
0x584	-	GPIO5_PAD_EN	GPIO group5 pad enable
0x588	-	GPIO5_SW_EN	GPIO group5 sleep wakeup enable
0x58C	-	GPIO5_INT_STATUS	GPIO group5 interrupt status for RISC
Others	-	-	Reserved

- **GPIO control register (GPIO_CTRLx)**

The GPIO_CTRLx is the register to control one GPIO pin. There are a total of 192 GPIO control registers. But there are only 188 GPIO pins available. So there are 4 GPIO control registers reserved, user should not access them, and they are: *GPIO1_CTRL31*, *GPIO2_CTRL19*, *GPIO4_CTRL18*, and *GPIO4_CTRL31*.

Bit	Name	Default	Description
0 (R/W)	INT_LT	1'b0	GPIO Interrupt low/falling edge trigger enable. 0 – Either the falling-edge interrupt or low level interrupt is disabled. 1 – If GPIO_INT_TYPE is set, the falling-edge interrupt is enabled; if GPIO_INT_TYPE is de-asserted, the low level interrupt is enabled. In Sleep mode this bit will define whether the low level from this GPIO pin will trigger wake-up of the chip. (Only valid for GPIO group0, 5)
1 (R/W)	INT_HT	1'b0	GPIO Interrupt high/rising edge trigger enable. 0 – either rising-edge interrupt or high level interrupt is disabled. 1 – If GPIO_INT_TYPE is set, the rising-edge interrupt is enabled; if GPIO_INT_TYPE is de-asserted, the high level interrupt is enabled. In Sleep mode this bit will define whether the high level from this

			GPIO pin will trigger wake-up of the chip. (Only valid for GPIO group0, 5)
2 (R/W)	INT_TYPE	1'b0	GPIO Interrupt Type Select bits. 0 – Interrupt is level-triggered. 1 – Interrupt is edge-triggered.
3 (R/W)	RISC_INT_EN	1'b0	Enable the GPIO to generate interrupt to RISC when Interrupt Status is set 0 – Interrupt is disabled. 1 – Interrupt is enabled.
4 (R/W)	INT_STAT US	1'b0	GPIO Interrupt Status. 0 – No interrupt. 1 – There is interrupt request. Write a one will clear the interrupt status bit
5 (R/W)	OUT_EN	1'b0	GPIO Data output enable. 0 – The GPIO is input pin. 1 – The GPIO is output pin.
6 (R/W)	OUT	1'b0	GPIO Data Output.
7 (R/W)	DATA_IN	1'bx	GPIO Data Input.
8 (R/W)	PULL	1'b0	GPIO Pull up/ no pull (see pin description)
9 (R/W)	OD	1'b0	GPIO open-drain mode. 1 – open-drain 0 – normal
10 (R/W)	DSP_INT_EN	1'b0	Exist only for GPIO group0 GPIO will generate different interrupt to DSP and RISC. There is different interrupt enable for DSP and RISC. Enable the GPIO to generate interrupt to DSP when Interrupt Status is set. 0 – Interrupt is disabled. 1 – Interrupt is enabled.
31:11	-	-	Reserved

* notes: *GPIO2_CTRL19* is reserved for DRAM clock control, This register should never be used by GPIO purpose.

- **GPIO DSP control enable Register (GPIOx_DSP_EN)**

This control register only exist for GPIO group0.

Bit	Name	Default	Description
31:0 (R/W)	DSP_EN<31:0>	32'h0	GPIO DSP control enable 1 – the GPIO pin is controlled by DSP 0 – the GPIO pin is controlled by RISC

- **GPIO PAD enable Register (GPIOx_PAD_EN)**

Each bit in *GPIOx_PAD_EN* control the pad mux of one GPIO pin in group x. For example:

GPIO0_PAD_EN<0> control the pad mux of GPIO pin 0 in group 0.

GPIO2_PAD_EN<21> control the pad mux of GPIO pin 85(32*2+21) in group 2.

Each GPIO have different default value for PAD MUX enable as following:

GPIO0_PAD_EN: 32'hF3FF-FFFF

GPIO1_PAD_EN: 32'hFFF8-0000

GPIO2_PAD_EN: 32'hFFFF-FFFF

GPIO3_PAD_EN: 32'h01FF-FC00
 GPIO4_PAD_EN: 32'hFFC3-0000
 GPIO5_PAD_EN: 32'h0000-0000

Please refer to the pin description list for what pins are enabled as GPIO by default. Actually all peripherals except NOR/NAND Flash interface are enabled as GPIO by default.

Bit	Name	Default	Description
31:0 (R/W)	PAD_EN<31:0>	Depends	GPIO PAD MUX control. 1 – the pin is used by GPIO 0 – the pin is used by other block

- **GPIO Wakeup enable Register (GPIOx_WAKEUP_EN)**

x=0,5

Only GPIO group 0,5 can function as wake-up GPIO.

Bit	Name	Default	Description
31:0 (R/W)	WAKEUP_EN<31:0>	32'h0	GPIO wake up enable. 0: in sleep mode, the corresponding GPIO will not wakeup the chip 1: in sleep mode, the corresponding GPIO will wakeup the chip

- **GPIO INTERRUPT STATUS Register (GPIOx_INT_STATUS)**

This register contain the interrupt status of all GPIOs in group x.

This register is read only, write to this register have no effect. To clear the interrupt, write the INT_STATUS bit in GPIO Control register instead.

Bit	Name	Default	Description
31:0 (R/W)	INT_STATUS<31:0>	32'h0	GPIO Interrupt Status. 0 – No interrupt. 1 – There is interrupt request

NOTE: GPIO recommended operation modes:

- To Use as an input without interrupts: set the GPIO as input and read its content
- To Use as output, set the output value first and then set the GPIO as output.

To Use as input which will generate an interrupt: First mask the interrupt with {INT_EN =0}, then set the GPIO as input. Set the INT_TYPE to level or edge. Then set the {INT_HT, INT_LT} for high/low-level or rising/falling-edge interrupt. Clear the GPIO interrupt status register before enabling the GPIO interrupt

5.9 PWM

5.9.14 Overview

PWM (Pulse Wide Modulate) generator is actually part of the GPIO logic. It can generate 4 independent outputs. Each output duty cycle can be adjusted by setting the corresponding wait and hold registers.

The PWM output pins are multiplexed with the GPIO<7:4>. Please refer to the PWM_OE register to find how to enable the PWM outputs.

5.9.15 PWM Registers

Table 26. PWM Interface Register Mapping

RISC Address <11:0>	Register	Description
0x00	PWM_OE	PWM output enable
0x04	PWM_WAIT0	PWM output 0 wait state for high pulse
0x08	PWM_HOLD0	PWM output 0 hold state for low pulse
0x0C	PWM_WAIT1	PWM output 1 wait state for high pulse
0x10	PWM_HOLD1	PWM output 1 hold state for low pulse
0x14	PWM_WAIT2	PWM output 2 wait state for high pulse
0x18	PWM_HOLD2	PWM output 2 hold state for low pulse
0x1C	PWM_WAIT3	PWM output 3 wait state for high pulse
0x20	PWM_HOLD3	PWM output 3 hold state for low pulse

- PWM Output Enable Register (PWM_OE) – 0x00

Bit	Name	Default	Description
0 (R/W)	PWM0 Output Enable	1'b0	Set 1 enable PWM0 output Set 0 disable PWM0 output
1 (R/W)	PWM1 Output Enable	1'b0	Set 1 enable PWM1 output Set 0 disable PWM1 output
2 (R/W)	PWM2 Output Enable	1'b0	Set 1 enable PWM2 output Set 0 disable PWM2 output
3 (R/W)	PWM3 Output Enable	1'b0	Set 1 enable PWM3 output Set 0 disable PWM3 output
31:4		28'h0	Reserved.

- PWM Output Wait State Register (PWM_WAIT<3:0>)

Bit	Name	Default	Description
31:0 (R/W)	HI_WAIT_CNT	32'h0	Number of IO clock cycles that the PWM pin is high = HI_WAIT_CNT + 1

- PWM Hold State Register (PWM_HOLD<3:0>)

Bit	Name	Default	Description
31:0 (R/W)	LO_HOLD_CNT	32'h0	Number of IO clock cycles that the PWM pin is low = LO_WAIT_CNT + 1

6 Graphic Display Subsystem

6.1 Overview

Atlas™-II Graphic Display Subsystem is designed for applications like: GPS Navigation/Telematics systems, PDAs and Smartphones. It can support most of the popular LCD panels as well as the TV encoders. And it has built-in 2D Graphic Acceleration Hardware to improve the 2D Graphic performance.

6.2 LCD Controller

6.2.1 Overview

Atlas™-II LCD Controller supports the following features:

- Resolution up to 1024x1024 pixels¹
- Support pixel depths of 8, 16 bpp for main display layer, and 8, 16 bpp for the two overlay layers
- Support following panels:
 - Maximum 18-bit output data can be used for non-RGB-muxed color TFT panel such as 16-, 18-bit color TFT panel
 - 8-bit RGB muxed TFT color panel
- On-chip RAMs for color palette or FRC sequence table, the FRC supports up to 16 levels of grey scale for monochrome mode, and 32 levels for every R, G, B of color mode
- Integrated 3-channel DMA (three layers mixing support and dual-panel display support)
- On-chip RGB to YUV hardware conversion and interface (8-bit 4:2:2 format) to TV encoder
- Programmable frame and line clock polarity, pulse width and starting locations, and programmable polarity for output enable and programmable toggle frequency of L_BIAS output pin

The LCD controller can be reset by four sources of resets (please refer to section of “Reset Controller” for details). After hardware reset, the LCD controller is disabled, and the output pins are configured as GPIO pins (please refer to the section of “GPIO” for details). And after any type of reset, all LCD controller registers are reset to the default values shown in the LCD register definitions.

If the LCD controller is being enabled for the first time after hardware reset, following registers must be programmed to enable the LCD controller:

- Configure power manager clock enable register (*PWR_CLK_EN*) to enable LCD controller system clock. See section of “Power Manager” for details.
- Configure Timing Control Register (*LCD_TIMCTRL*) to set the LCD controller as master mode or slave mode, that is, set L_PCLK, L_LCLK or L_FCLK as outputs or inputs.
- Configure GPIO pin mux register (*GPIOx_PAD_EN*) to release pins to LCD.
- Write color palette or grey palette (FRC sequence table), if needed.
- Program all of the LCD registers required except the DMA start registers (*SCN_CUR_Y/OSD_CUR_Y* and *SCN_ADDR/OSD_ADDR*) and the Frame Valid bit (bit0 of *DISPLAYMODE* register)
- Write 1 to DMA start registers (*SCN_CUR_Y/OSD_CUR_Y* and *SCN_ADDR/OSD_ADDR*) to start the DMA
- Write 1 to the Frame Valid bit to output the valid image frame to data pins

6.2.2 Pin Description

¹ The actual resolution is limited by the memory bandwidth

When the LCD controller is enabled, LDD<17:0> and L_BIAS are outputs only; and L_PCLK, L_LCLK and L_FCLK can be programmable inputs or outputs (slave or master mode). When the LCD Controller is disabled, LDD<17:8> can be used as general purpose IO (please refer to section of “GPIO” for details).

Table 27. Atlas™-II LCD Controller Pin Descriptions

Pin Name	Function Name	Type	Definition
X_LDD<7:0>	LDD<15:0>	○	LCD Data Bus: Provide four, eight or sixteen bit data at a time to the LCD panel. Either the bottom four pins (LDD<3:0>), the bottom 8 pins (LDD<7:0>) or all 16 pins (LDD<15:0>) will be used. For passive monochrome displays, each pin value represents a single pixel. For passive color displays, every three pins represent a single pixel. For active displays, all the used data pins represent a single pixel. LDD<17> is the R channel's lowest bit and LCD<16> is the B channel's lowest bit when rgb666 output mode is needed. If not used to drive the LCD display, LDD<17:8> can be configured as GPIO pins.
X_GPIO<15:8>	LDD<15:8>	○	
X_GPIO<25:24>	LDD<17:16>	○	
X_L_PCLK	L_PCLK	I/O	LCD Shift / Pixel Clock (programmable polarity): Used by the LCD display to latch the pixel data and control signals.
X_L_LCLK	L_LCLK	I/O	LCD Line Clock (programmable polarity): Used by the LCD display to signal the end or the beginning of a line. Line Pulse for passive (STN) displays. Horizontal sync for active (TFT) displays.
X_L_FCLK	L_FCLK	I/O	LCD Frame Clock (programmable polarity): Used by the LCD display to signal the start of a new frame. First Line marker for passive (STN) displays. Vertical sync for active (TFT) displays.
X_L_BIAS	L_BIAS	○	AC Bias (M signal) / Data Enable: Switch signal of some passive (STN) displays to convert the liquid crystal drive waveform into AC. Data Enable (DE) for active (TFT) displays.

The following table describes the LCD pins required for the various types of passive displays.

Table 28. Atlas™-II LCD Controller Data Pins for passive displays

Color/ Monochrome Panel	Single/ Dual Panel	4bit/ 8bit /16bit data parallel	Screen Portion	Pins	Top left pixel in a frame
Color / Monochrome	Single	4bit	Whole	LDD<3:0>	LDD<3>
Color / Monochrome	Dual	4bit	Upper / Lower	LDD<3:0>	LDD<3>
			Lower / Upper	LDD<7:4>	LDD<7>
Color / Monochrome	Single	8bit	Whole	LDD<7:0>	LDD<7>
Color / Monochrome	Dual	8bit	Upper / Lower	LDD<7:0>	LDD<7>
			Lower / Upper	LDD<15:8>	LDD<15>
Color	Single	16bit	Whole	LDD<15:0>	LDD<15>

NOTE: For passive color mode, one pixel is made of three bits of LDD pins, therefore, the last column of the above table is actually the Red bit of top left pixel in a frame. Following is an example of pixel ordering for passive color 8bit parallel dual panel display, where n is the number of rows of the panel.

Top Left Corner of Screen of the upper half pannel

	Column 0	Column 0	Column 0	Column 1	Column 1	Column 1	Column 2	Column 2	Column 2	
	Red	Green	Blue	Red	Green	Blue	Red	Green	Blue	
Row 0	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>	...
Row 1	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>	...
...										
Row n/2	LDD<7>	LDD<6>	LDD<5>	LDD<4>	LDD<3>	LDD<2>	LDD<1>	LDD<0>	LDD<7>	...

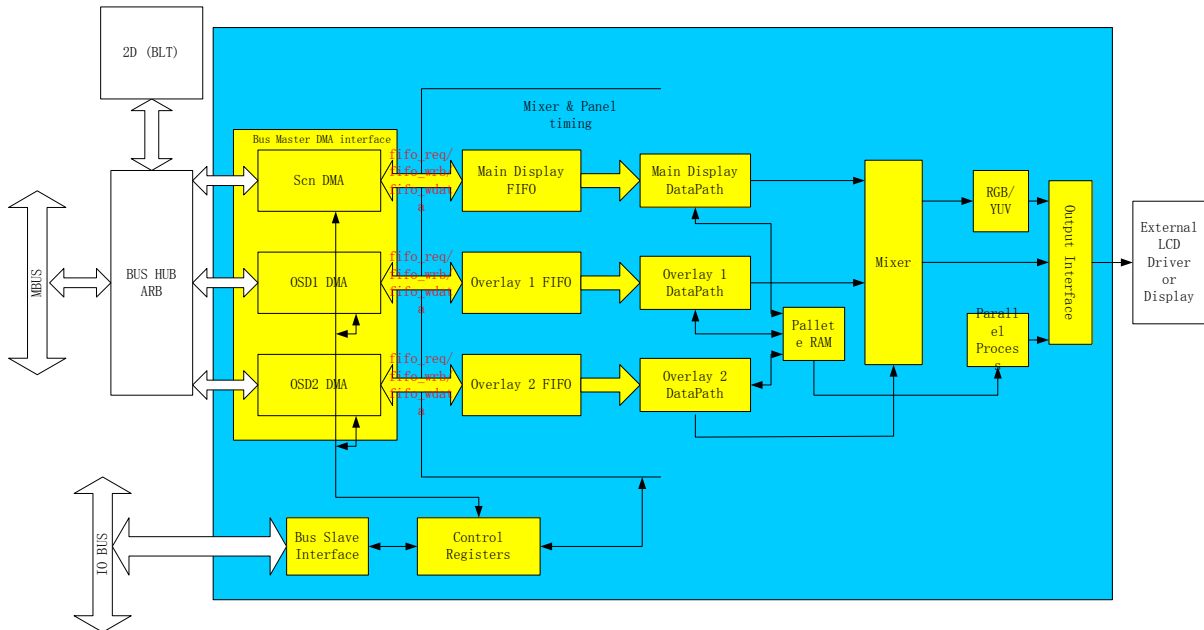
Top Left Corner of Screen of the second half pannel

Row n/2+1	LDD<15>	LDD<14>	LDD<13>	LDD<12>	LDD<11>	LDD<10>	LDD<9>	LDD<8>	LDD<15>	...
-----------	---------	---------	---------	---------	---------	---------	--------	--------	---------	-----

Pixel Ordering of Passive Color 8-bit Parallel Dual Panel Display

For color TFT panels, LDD<17:0> can be used for 12-, 16-, or 18-bit panels, LDD<7:0> are used for 8-bit RGB muxed TFT color panel.

6.2.3 Functional Description



Block Diagram of LCD Controller

The LCD controller block provides a means for Atlas™-II Processor to interface with an external LCD driver or a LCD display. Atlas™-II Processor supports various LCD displays for applications ranging from PDA, GPS, to digital camera. There are three major datapaths – one for the main display data, and one for the second/overlay display data and another for the overlay2 display data to support dual panel LCDs or to mix two layer OSDs on top of the main display.

The LCD controller block connects to the system bus through a BUS_HUB_ARB module, the BUS_HUB_ARB connects to the system bus as a Bus master and it does further arbitration for up to 8 masters. The LCD controller takes 3 master interfaces on the BUS_HUB_ARB. And the LCD controller has one Bus Slave interface for registers reading and writing by the RISC through memory-mapped registers and buffers. The 3 masters are grouped as a Bus Master DMA interface which will request display data from memory by the three DMA controllers independently. On the other side, the LCD controller is connected to the I/O pins to interface with the external LCD driver or display.

The **Bus Master Interface** requests the display data and two overlay data from the DMA controller, and performs the bus protocol to transfer the data to the corresponding FIFO for each datapath. The arbitrator for the 3 DMA is moved into the outside BUS_HUB_ARB to arbitrate the requests from the main display, the second/overlay, the overlay2 along with requests from the 2D block.

Each datapath contains a 128x32bit **FIFO** to temporarily store data bursts from the DMA controller for the datapath to process. The FIFOs send requests to the Bus Master DMA Interface when they are not full. When they are almost empty, they can send a high-level request to the Bus Master Interface. The levels for these requests can be programmed by registers.

The **Main Display Datapath** (also named as SCN datapath) processes the main display data, that is, the first DMA channel data from the FIFO. The main display data can be defined as 1-, 2-, 4-, 8-, 12-, or 16-bit per pixel image. The datapath decomposes the pixel data from the 32-bit FIFO data according to the correct ordering.

The **Second/Overlay Datapath** (also named as OSD datapath) is similar to the Main Display Datapath in function but processes the second/overlay data. The second data, which means the second DMA channel data, is defined for the half panel of the dual panel display. The overlay data can be defined as 2-, 4-, 8-, or 16-bit per pixel image. And it is defined for only a sub-portion (or equal) of the main display.

The **Overlay2 Datapath** (also named as OSD2 datapath) is similar to the Second/Overlay Datapath in function but processes the Overlay2 data. It can be defined as 2-, 4-, 8-, or 16-bit per pixel image. The overlay2 data is defined for only a sub-portion(or equal) of the main display

The **Palette RAM** includes a 256x18bit entry, a 16x18bit entry, and a 32x64bit entry. They can be either a color palette of RGB value or a dithering sequence for FRC. The palette RAMs are written or read directly by RISC Interface to access the specified address. The 2-, 4-, or 8-bit pixel data can be an index of the palette RAMs.

The **Mixer** mixes the values from the Main Display Datapath and the two Overlay Datapaths. It implements the two overlay layers' alpha-blending and/or color-keying on top of the main display layer.

The **RGB/YUV** unit converts the screen output from 24-bit RGB value to 24-bit YUV value using a matrix multiplication with programmable coefficients. **(Note that this unit is after the mixer so both the overlay data and the main screen data will be converted.)**

The **Parallel Process** unit converts the serial data outputted from frame rate control logic to 4-, 8- or 16-bit parallel data. Thus for dual panel passive display, the output data will be 8- or 16- bits. This unit is only valid for STN (S-STN or D-STN) displays.

The **Output Interface** converts the RGB output of the RGB/YUV unit (either RGB values or converted YUV values) and the parallel data of the Parallel Process unit to the format required by the external LCD controller or displays. Because Atlas™ Processor supports different types of LCDs (for different applications), the output interface may output 16-bit RGB values, 8-bit alternate R, G, or B values (8-bit RGB muxed format) for color TFT LCDs, or 4-, 8-, or 16-bit outputs for S-STNs or D-STNs, or 8-bit 4:2:2 mode and 16-bit 4:4:4 mode for YUV output, etc.

The **Timing Control** block controls the timing information for the display. It keeps track of the pixel and line number, and generates pixel clocks and control signals to be used by the datapaths and the Output Interface.

The **Bus Slave Interface** provides the access interface for RISC Interface to read/write LCD controller control registers, palette and FIFOs. The Bus Slave Interface will process the request and either read/write a control register, read/write the palette, or modify the data in one of the FIFOs.

The **Control Registers** block contains all the control registers used by the LCD interface. These registers can be read or written to by the RISC Interface via the Bus Slave Interface.

The following sections describe some functional blocks within the LCD controller such as the DMA to memory interface, palette RAMs and the frame rate control.

6.2.3.1 Display Layers

There are a total of 3 display layers that can be mixed before display to the LCD panel:

- Layer #1: Main Display
- Layer #2: Overlay Display
- Layer #3: Overlay Display

The following diagram shows the block diagram of the mixer that mixes those 3 display layers:

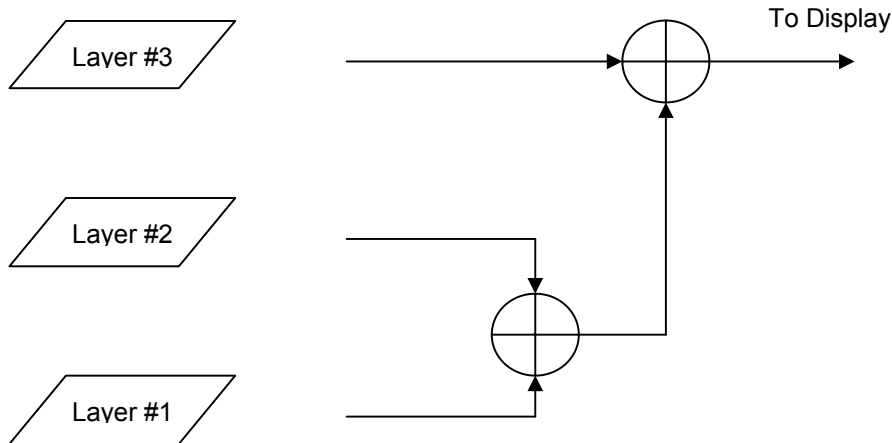


Figure 17. Block Diagram of Layer Mixer

Layer1 is the main layer (SCN layer). It supports 8-bit index (index into RGB 6:6:6 lookup table), 16-bpp (RGB 6:5:5, 5:6:5, 5:5:6) data format. And it also supports 8-bit (with 8-bit index), 18-bit (with 18-bit RGB 6:6:6) color-key range registers¹; if a color matches between the range defined by two registers, the color is transparent and the pixel from the layer2 (OSD layer) will be shown instead.

Layer2 & 3 (OSD & OSD2 layer) are all overlay layers. Each of the overlay layers has the following alpha-blend and/or color-key modes:

- 8-bit index (index into RGB 6:6:6 lookup table)
- RGB 5:6:5
- 8-bit (with 8-bit index), 18-bit (with 18-bit RGB 6:6:6) color-key range registers¹; if a color matches between the range defined by two registers, it will be transparent and the pixel from the main display layer will be shown instead.
- 4-bit planar blending register to blend all pixels on the plane (that are non-transparent) to one planar alpha value.

6.2.3.2 DMA to Memory Interface

Encoded pixel data of frame buffer are stored in off-chip memory (usually DRAM), and are transferred to the LCD controller's 128x32bit FIFOs. As the internal bus is 32bit, while the pixel data can be 1-, 2-, 4-, 8-, 12-, and 16-bit, the user can select how the LCD views the ordering of frame buffer pixel by programming the big/little endian selection bits and the pixel offset bits in DMA SCNBASE, OSDBASE or OSD2BASE registers.

There are two levels of endian selection for a DWORD data from the memory. One is for bytes within a DWORD, and the other is for 1-, 2-, or 4-bit within a byte. Following figures are some examples for these two levels of endian selections in different bit/pixel formats. The pixel offset values are 0x0 in these examples.

¹ Two registers to specify a color key range.

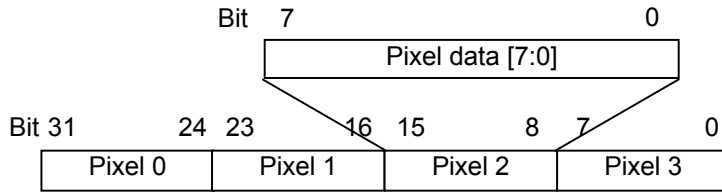


Figure 18. 8bits/pixel data in a DWORD (Big Endian)

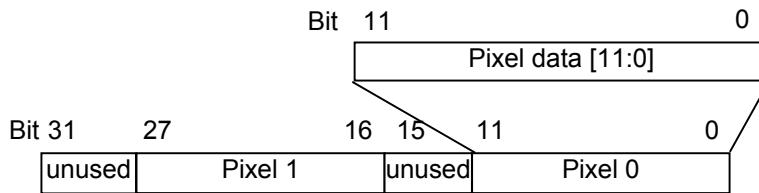


Figure 19. 12bits/pixel data in a DWORD (Little Endian)

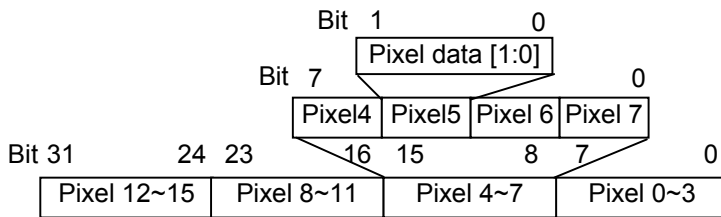


Figure 20. 2bits/pixel data in a DWORD (Big Endian in a byte, Little Endian for byte in a DWORD)

The following example is for non-zero pixel offset value. As pixel width is 2 bits, the valid pixel offset value range is from 0x0 to 0xF. The following figure is the case in which pixel offset is 0x5. And the figure indicates the starting pixels of a frame.

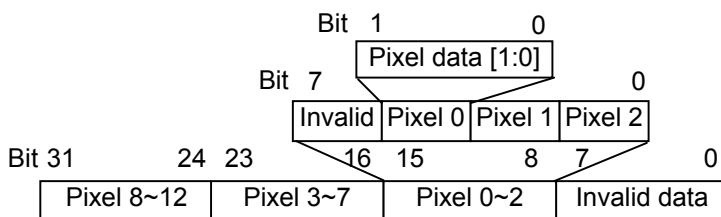


Figure 21. 2bits/pixel data in a DWORD (Big Endian in a byte, Little Endian for byte in a DWORD)

6.2.3.3 Frame Rate Control

The Frame Rate control (FRC) is actually a time-based dithering, which is to turn each pixel on or off in a certain period of time to create the perception of different intensities. It is implemented by looking up the FRC sequence table periodically. For an MxN FRC sequence table, the FRC can support up to M levels, and the sequence repeats itself every N frames for every pixel. Higher 0 to 1 transition frequency within N frames will result in better picture quality.

For passive monochrome mode, the FRC is implemented by looking up the 16 x 32 FRC sequence table (the table is located in part of the 256x18bit palette RAM). The FRC can support up to 16 grey levels.

For passive color mode, the FRC is implemented by looking up the 32x64 FRC sequence table (this table is located in the 64x32 RAM). The FRC can support up to 32 levels for every R, G, B color.

In order to decrease the flickering side effect of FRC, the random control is added to avoid all the pixels in a large group on and off together in synchronization. This random control is implemented by using two linear feedback shift registers (LFSR). The line LFSR which is shifted every HSYNC, and the pixel LFSR which is shifted every pixel clock, are reset every frame. The sum of the lower 5 or 6 bits of the two LFSRs, plus the frame count, gives the index to select one bit from the 32-bit or 64-bit sequence as one pixel ON/OFF value in a frame. The initial values and the algorithms of the two LFSR are changeable to obtain the best quality.

6.2.3.4 Palette RAMs

The palette RAMs are written or read directly by RISC Interface to access the specified address. Three RAMs (256x18bit, 16x18bit and 64x32bit) are used for the palettes and the FRC sequence tables.

For the 256x18bit palette RAM, when as color palette, each address contains RGB value for a color that has an index corresponding to the address. The color component is 6-bits each, specifying the 6 MSB of an 8-bit value. For the 2-bit/pixel lookup, the bottom 4 entries are used. For the 4-bit/pixel lookup, the bottom 16 entries are used. For the 8-bit/pixel lookup, all the 256 entries are used. While for passive monochrome mode, this palette is used for dithering of frame rate control. As the dithering sequence for one gray color is 32 bit, the bottom 32 entries are used for 16 grey levels. For 4bit/pixel mode, bits <15:0> of entry 0 and entry 16 combine as 32 bits for one gray color, and bits <15:0> of entry 1 and entry 17 as another gray color, and so on. And for 2bit/pixel mode, only entry 0~3 and entry 16~19 are used.

The 16x18bit palette RAM is always a color palette, and each address contains RGB value for a color that has an index corresponding to the address. The color component is 6-bits each, specifying the 6 MSB of a 4-bit value. This palette is used for 4-bit color value.

The 64x32bit RAM is used for dithering sequence of frame rate control in passive color mode. It generates a 32 x 64 entry to support 32 grey-scale levels for every R, G, B color. As the dithering sequence for one gray color is 64 bit, the bottom half RAM is used for the first 32 bits of the dithering sequence, and the up half RAM is used for the other 32 bits of the dithering sequence.

6.2.4 LCD Controller Registers

Table 29. LCD Controller Register Map

RISC Address <12:0>	Register	Description
0x0000	LCD_SCN_HSTART	Horizontal start position for active screen
0x0004	LCD_SCN_VSTART	Vertical start position for active screen
0x0008	LCD_SCN_HEND	Horizontal end position for active screen
0x000C	LCD_SCN_VEND	Vertical end position for active screen
0x0010	LCD_OSD_HSTART	Horizontal start position for active OSD
0x0014	LCD_OSD_VSTART	Vertical start position for active OSD
0x0018	LCD_OSD_HEND	Horizontal end position for active OSD
0x001C	LCD_OSD_VEND	Vertical end position for active OSD
0x0020	LCD_OSD2_HSTART	Horizontal start position for active OSD2
0x0024	LCD_OSD2_VSTART	Vertical start position for active OSD2

0x0028	LCD_OSD2_HEND	Horizontal end position for active OSD2
0x002C	LCD_OSD2_VEND	Vertical end position for active OSD2
0x0030	LCD_HSYNC_PERIOD	Period for Master Mode Horizontal Sync
0x0034	LCD_HSYNC_WIDTH	Width for Master Mode Horizontal Sync
0x0038	LCD_VSYNC_PERIOD	Period for Master Mode Vertical Sync
0x003C	LCD_VSYNC_WIDTH	Width for Master Mode Vertical Sync
0x0040	LCD_RGBSEQ	Sequence for RGB outputs
0x0044	LCD_TIMCTRL	Timing Control Register
0x0048	LCD_ALT_HSYNC1	First alternate HSYNC control register
0x004C	LCD_ALT_VSYNC1	First alternate VSYNC control register
0x0050	LCD_OSC_RATIO	OSC PIXCLK divider ratio register
0x0054	LCD_TIM_STATUS	Timing Control Status Register
0x0058	-	Reserved
0x005C	-	Reserved
0x0060	LCD_BLANK	Blanking Register
0x0064	LCD_DISPLAYMODE	Display mode and format register
0x0068	LCD_LINE_LFSR	Line shift register
0x006C	LCD_PIX_LFSR	Pixel shift register
0x0070	LCD_FMOD	Frame Rate control register
0x0074	LCD_RGB_YUV_COEF1	RGB to YUV Coefficient 1
0x0078	LCD_RGB_YUV_COEF2	RGB to YUV Coefficient 2
0x007C	LCD_RGB_YUV_COEF3	RGB to YUV Coefficient 3
0x0080	LCD_YUV_CTRL	RGB to YUV conversion control register
0x0084	LCD_SCN_FIELD	Screen control for external TV encoder register
0x0088	LCD_OSDALPHA	OSD and OSD2 Alpha blending control register
0x008C	LCD_SCNCKEYB	SCN color key big RGB value
0x0090	LCD_SCNCKEYS	SCN color key small RGB value
0x0094	LCD OSDCKEYB	OSD color key big RGB value
0x0098	LCD OSDCKEYS	OSD color key small RGB value
0x009C	LCD OSD2CKEYB	OSD2 color key big RGB value
0x00A0	LCD OSD2CKEYS	OSD2 color key small RGB value
0x00A4	LCD OSDPAL0	OSD Palette Entry 0
0x00A8	LCD OSDPAL1	OSD Palette Entry 1
0x00AC	LCD OSDPAL2	OSD Palette Entry 2
0x00B0	LCD OSDPAL3	OSD Palette Entry 3
0x00C0	LCD_SCNFIFO	Screen FIFO Control Register
0x00C4	LCD OSDFIFO	OSD FIFO Control Register
0x00C8	LCD OSD2FIFO	OSD2 FIFO Control Register
0x00CC	-	Reserved
0x00D0	-	Reserved
0x00D4	-	Reserved
0x00D8	LCD_SCNFIFO_SUPPRESS	Screen FIFO Write Suppress Register
0x00DC	LCD OSDFIFO_SUPPRESS	OSD FIFO Write Suppress Register
0x00E0	LCD OSD2FIFO_SUPPRESS	OSD2 FIFO Write Suppress Register
0x00E4	LCD_FIFOADD_RST	FIFOs Address Reset Register
0x0100	LCD_SCNBASE	Screen Memory Base Register
0x0104	LCD_SCN_XSIZE	X Size for screen DMA

0x0108	LCD_SCN_YSIZE	Y Size for screen DMA
0x010C	LCD_SCN_SKIP	Skip value for screen DMA
0x0110	-	Reserved
0x0114	LCD_SCN_ADDR	Current DMA Address for the Screen Display
0x0118	LCD_OSDBASE	OSD Memory Base Register
0x011C	LCD_OSD_XSIZE	X Size for OSD DMA
0x0120	LCD_OSD_YSIZE	Y Size for OSD DMA
0x0124	LCD_OSD_SKIP	Skip value for OSD DMA
0x0128	-	Reserved
0x012C	LCD_OSD_ADDR	Current DMA Address for the OSD Display
0x0130	LCD_OSD2BASE	OSD2 Memory Base Register
0x0134	LCD_OSD2_XSIZE	X Size for OSD2 DMA
0x0138	LCD_OSD2_YSIZE	Y Size for OSD2 DMA
0x013C	LCD_OSD2_SKIP	Skip value for OSD2 DMA
0x0140	-	Reserved
0x0144	LCD_OSD2_ADDR	Current DMA Address for the OSD2 Display
0x0180	LCD_INT_CTRL	Interrupt Status and Clear register
0x0184	LCD_INT_MASK	Interrupt Mask
0x0188	LCD_SCN_INT_LINE	Interrupt Line Number
0x0400	LCD_SCN_FIFO_PUSH_POP	Push/Pop Screen FIFO
0x0600~0x7FC	LCD_SCN_FIFODATA	Read/Write Screen FIFO
0x0800	LCD_OSD_FIFO_PUSH_POP	Push/Pop OSD FIFO
0x0A00~0xBFC	LCD_OSD_FIFODATA	Read/Write OSD FIFO
0x0C00	LCD_OSD2_FIFO_PUSH_POP	Push/Pop OSD2 FIFO
0x0E00~0xFFC	LCD_OSD2_FIFODATA	Read/Write OSD2 FIFO
0x1000~0x13FC	LCD_PALETTE	Read/Write 256-entry Palette
0x1400~0x143C	LCD_PALETTE16	Read/Write 16-entry Palette
0x1800~0x18FC	LCD_COLORSEQ	Read/Write 32x64 Color Dithering Sequence
Others	-	Reserved

6.2.4.1 Screen and OSD Active Region Definition Registers

The active region of the LCD display is specified by a rectangle defined by two coordinates, the horizontal and vertical start position, and the horizontal and vertical end position, as shown in the diagram below. The LCD display contains internal counters, which will count pixels by pixel clock and lines by horizontal sync. The pixel count is cleared by horizontal sync and line count is cleared by vertical sync.

Note: For display a SCN, OSD or OSD2 active region of M(pixels) x N(lines), following conditions must be satisfied:

1. $M = \text{LCD_SCN_HEND} - \text{LCD_SCN_HSTART} + 1$ or
 $\text{LCD_OSD_HEND} - \text{LCD_OSD_HSTART} + 1$ or
 $\text{LCD_OSD2_HEND} - \text{LCD_OSD2_HSTART} + 1$
2. $N = \text{LCD_SCN_VEND} - \text{LCD_SCN_VSTART}$ or
 $\text{LCD_OSD_VEND} - \text{LCD_OSD_VSTART}$ or
 $\text{LCD_OSD2_VEND} - \text{LCD_OSD2_VSTART}$

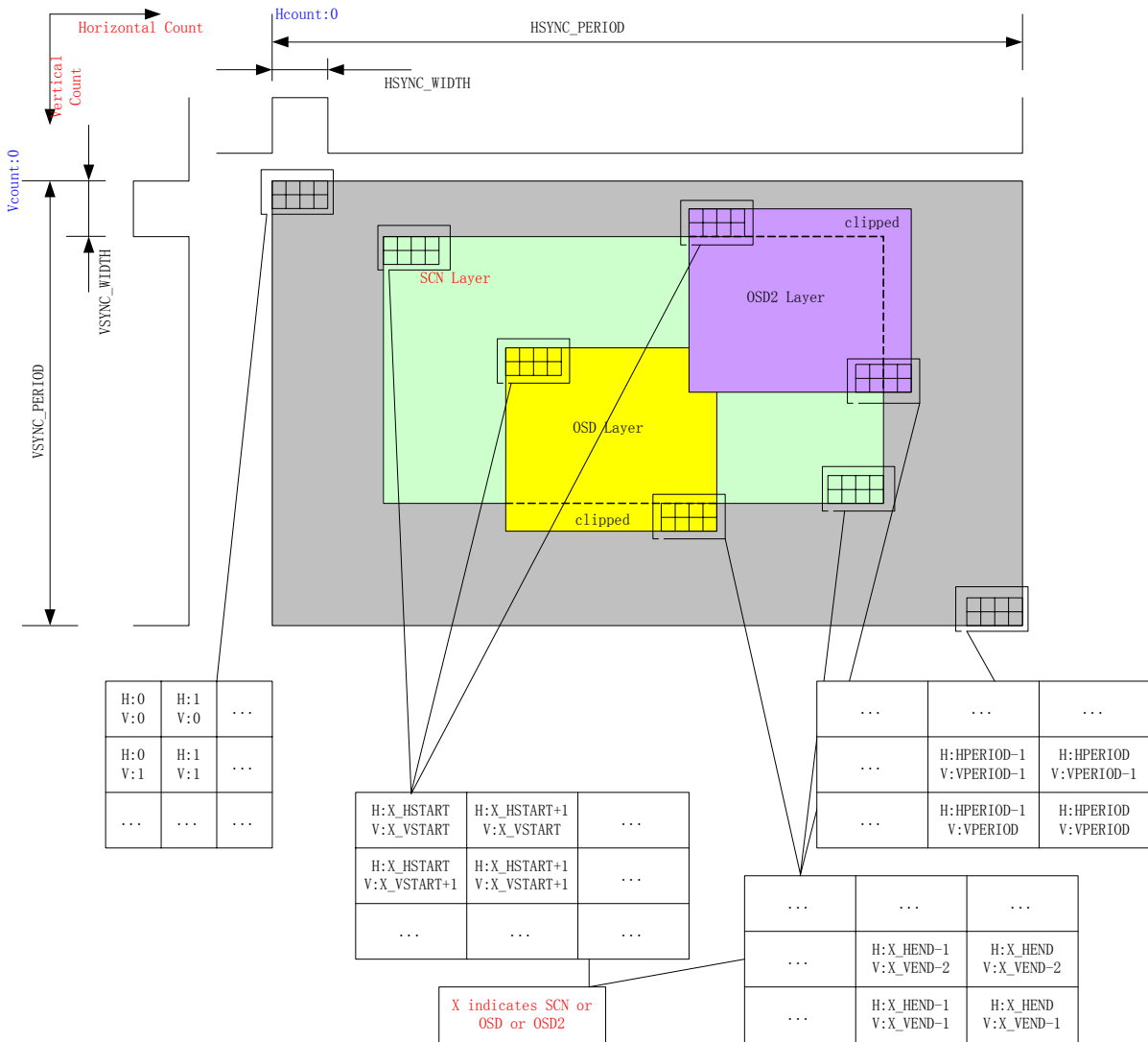


Figure 22. Screen and OSD Active Region

One should understand as show in above figure, only pixels inside the SCN 2D window will be visible, that is OSD and OSD2 layer will be clipped.

- Horizontal start position for active screen (LCD_SCN_HSTART) – 0x0

Bit	Name	Default	Description
10:0 (R/W)	SCN_HSTART	11'h0	Horizontal Start Position (in pixel number) for the active LCD display. This value, along with the horizontal end position and vertical start and end positions, define the rectangle for the active LCD display.
31:11	-	21'h0	Reserved

- Vertical start position for active screen (LCD_SCN_VSTART) – 0x4

Bit	Name	Default	Description
-----	------	---------	-------------

10:0 (R/W)	SCN_VSTART	11'h0	Vertical Start Position (in line number) for the active LCD display. This value, along with the horizontal start and end position and the vertical end positions, define the rectangle for the active LCD display.
31:11	-	21'h0	Reserved

- **Horizontal end position for active screen (LCD_SCN_HEND) – 0x8**

Bit	Name	Default	Description
10:0 (R/W)	SCN_HEND	11'h0	Horizontal End Position (in pixel number) for the active LCD display. This value, along with the horizontal start position and vertical start and end positions, define the rectangle for the active LCD display.
31:11	-	21'h0	Reserved

- **Vertical end position for active screen (LCD_SCN_VEND) – 0xC**

Bit	Name	Default	Description
10:0 (R/W)	SCN_VEND	11'h0	Vertical End Position (in line number) for the active LCD display. This value, along with the horizontal start and end positions and vertical start position, define the rectangle for the active LCD display. This line is not included in the display region as indicated in Figure 8.
31:11	-	21'h0	Reserved

For effects of the SCN region settings, please refer to figure of “Screen and OSC Active Region”. They define the top-left and the bottom-right corner pixel position.

- **Horizontal start position for active OSD screen (LCD_OSD_HSTART) – 0x10**

Bit	Name	Default	Description
10:0 (R/W)	OSD_HSTART	11'h0	Horizontal Start Position (in pixel number) for the active OSD display. This value, along with the horizontal end position and vertical start and end positions, define the rectangle for the active OSD display.
31:11	-	21'h0	Reserved

- **Vertical start position for active OSD screen (LCD_OSD_VSTART) – 0x14**

Bit	Name	Default	Description
10:0 (R/W)	OSD_VSTART	11'h0	Vertical Start Position (in line number) for the active OSD display. This value, along with the horizontal start and end position and the vertical end positions, define the rectangle for the active OSD display.
31:11	-	21'h0	Reserved

- **Horizontal end position for active OSD screen (LCD_OSD_HEND) – 0x18**

Bit	Name	Default	Description
10:0 (R/W)	OSD_HEND	11'h0	Horizontal End Position (in pixel number) for the active OSD display. This value, along with the horizontal start position and vertical start and end positions, define the rectangle for

			the active OSD display.
31:11	-	21'h0	Reserved

- **Vertical end position for active OSD screen (LCD_OSD_VEND) – 0x1C**

Bit	Name	Default	Description
10:0 (R/W)	OSD_VEND	11'h0	Vertical End Position (in line number) for the active OSD display. This value, along with the horizontal start and end positions and vertical start position, define the rectangle for the active OSD display. This line is not included in the display region as indicated in Figure 8 even OSD is not clipped.
11 (R/W)	OSD_SETTING_VALID	1'h0	Confirm the new Region and DMA setting by writing this bit with 1. This bit is self-cleared after being written with 1. Remark: As the layer gets a group of setting such as OSD_HSTART, OSD_HEND, OSD_VSTART, OSD_VEND along with the OSD DMA settings, they should effect synchronously, when one register is changed it should not take effect immediately, so this bit is used as a confirm of the setting-group.
31:12	-	20'h0	Reserved

For effects of the OSD region settings, please refer to figure of “Screen and OSC Active Region”. They define the top-left and the bottom-right corner pixel position.

- **Horizontal start position for the second active OSD screen (LCD_OSD2_HSTART) – 0x20**

Bit	Name	Default	Description
10:0 (R/W)	OSD2_HSTART	11'h0	Horizontal Start Position (in pixel number) for the second active OSD display. This value, along with the horizontal end position and vertical start and end positions, define the rectangle for the second active OSD display.
31:11	-	21'h0	Reserved

- **Vertical start position for the second active OSD screen (LCD_OSD2_VSTART) – 0x24**

Bit	Name	Default	Description
10:0 (R/W)	OSD2_VSTART	11'h0	Vertical Start Position (in line number) for the second active OSD display. This value, along with the horizontal start and end position and the vertical end positions, define the rectangle for the second active OSD display.
31:11	-	21'h0	Reserved

- **Horizontal end position for the second active OSD screen (LCD_OSD2_HEND) – 0x28**

Bit	Name	Default	Description
10:0 (R/W)	OSD2_HEND	11'h0	Horizontal End Position (in pixel number) for the second active OSD display. This value, along with the horizontal start position and vertical start and end positions, define the rectangle for the second active OSD display.
31:11	-	21'h0	Reserved

- Vertical end position for the second active OSD screen (LCD_OSD2_VEND) – 0x2C

Bit	Name	Default	Description
10:0 (R/W)	OSD2_VEND	11'h0	Vertical End Position (in line number) for the second active OSD display. This value, along with the horizontal start and end positions and vertical start position, define the rectangle for the second active OSD display. This line is not included in the display region as indicated in Figure 8 even OSD is not clipped.
11 (R/W)	OSD2_SETTING_VALID	1'h0	Confirm the new Region and DMA setting by writing this bit with 1. This bit is self-cleared after being written with 1 Remark: Refer to the remark of OSD_SETTING_VALID.
31:12	-	20'h0	Reserved

NOTE: SCN layer's setting valid bit is bound with its DMA_START bit.

Please refer to figure of "Screen and OSC Active Region" for the effects of the OSD2 region settings. They define the top-left and the bottom-right corner pixel position.

6.2.4.2 Sync Signal Generation Registers

In the slave mode, the horizontal and vertical sync signals are given by the display. In the master mode, Atlas™-II Processor will generate the sync signals based on a set of parameters, specifically the period and width of each sync pulse.

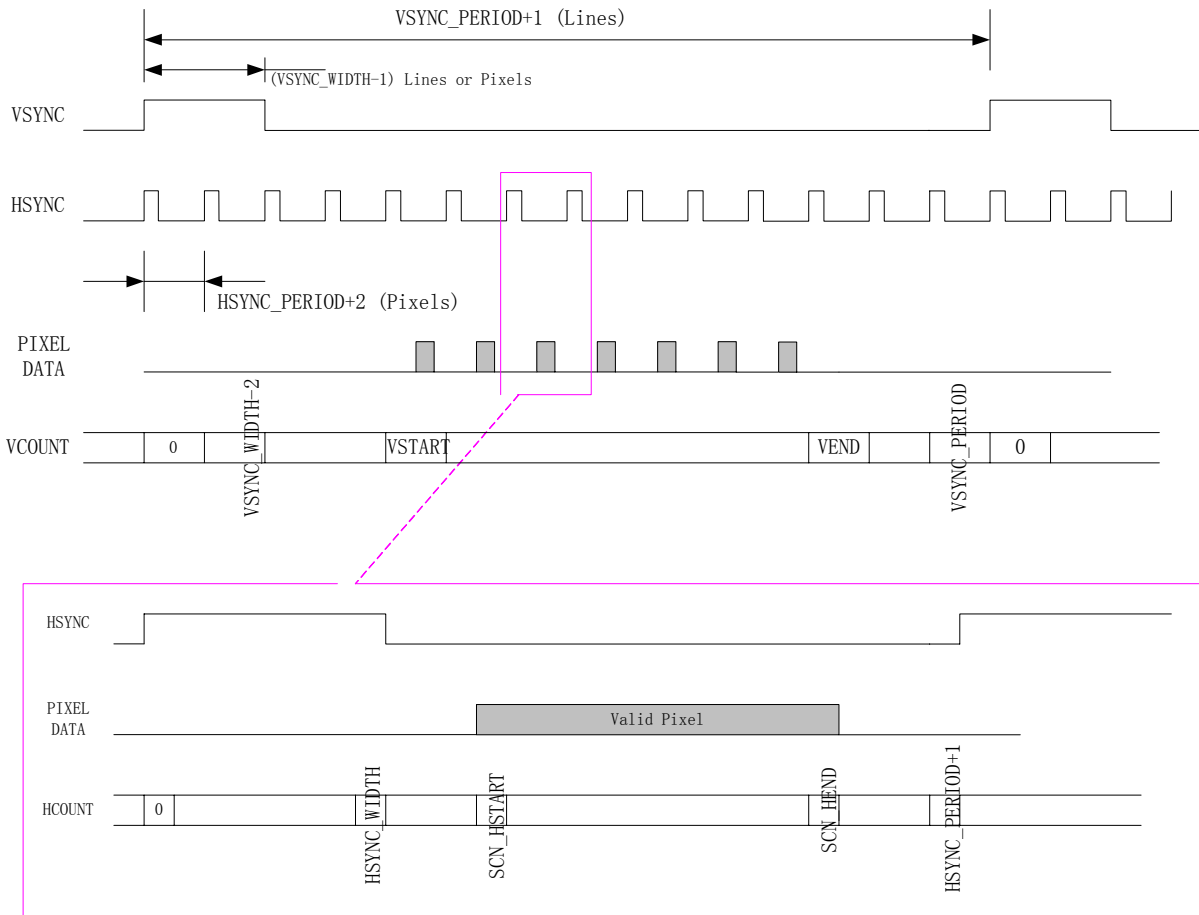


Figure 23. LCD Sync Signals Waveform

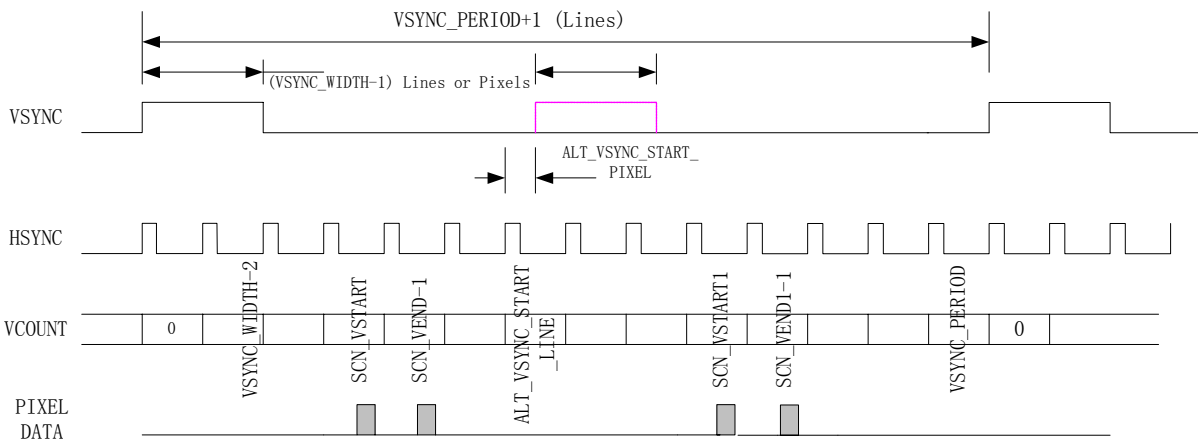


Figure 24. LCD Alteration Signals Waveform

- Period for Master Mode Horizontal Sync (LCD_HSYNC_PERIOD) – 0x30

Bit	Name	Default	Description
10:0	HSYNC_PERIOD	11'h0	For master mode, where Atlas™-II Processor generates the

(R/W)			sync signals, this value defines the period of the horizontal sync pulse, in number of pixels. If this register is set to 60, then the period of the sync is 62 pixel clocks. For slave mode, this register has no effect.
31:11	-	21'h0	Reserved

- **Width for Master Mode Horizontal Sync (LCD_HSYNC_WIDTH) – 0x34**

Bit	Name	Default	Description
10:0 (R/W)	HSYNC_WIDTH	11'h0	For master mode, where Atlas™-II Processor generates the sync signals, this value defines the width (minus 1) of the horizontal sync pulse, in number of pixels. If this register is set to 8, then the width of the sync pulse is 9 pixel clocks. For slave mode, this register has no effect.
31:11	-	21'h0	Reserved

- **Period for Master Mode Vertical Sync (LCD_VSYNC_PERIOD) – 0x38**

Bit	Name	Default	Description
10:0 (R/W)	VSYNC_PERIOD	11'h0	For master mode, where Atlas™-II Processor generates the sync signals, this value defines the period of the vertical sync pulse, in number of lines. For slave mode, this register has no effect. The actual vertical line number is VSYNC_PERIOD+1
31:11	-	21'h0	Reserved

- **Width for Master Mode Vertical Sync (LCD_VSYNC_WIDTH) – 0x3C**

Bit	Name	Default	Description
11 (R/W)	WIDTH_UNIT	1'b0	1 = Vertical sync pulse width defined below is in number of lines. 0 = Vertical sync pulse width defined below is in number of pixels. Usually, this bit is set with 1.
10:0 (R/W)	VSYNC_WIDTH	11'h0	For master mode, where Atlas™-II Processor generates the sync signals, this value defines the width of the horizontal sync pulse, in either number of lines or number of pixels (depending on bit 11 above). For slave mode, this register has no effect. The actual width is VSYNC_WIDTH-1 (lines or pixels)
31:12	-	20'h0	Reserved

Note: There are some conditions that must be satisfied:

- (1) **HSYNC_PERIOD > HSYNC_END**
- (2) **VSYNC_PERIOD > VSYNC_END+1**
- (3) **HSYNC_PERIOD > MAX(SCN_HEND, OSD_HEND, OSD2_HEND)**
- (4) **VSYNC_PERIOD > MAX(SCN_VEND, OSD_VEND, OSD2_VEND)**

6.2.4.3 Screen Control Registers

The following registers specify certain modes and parameters for the display. For example, the timing control registers determines whether the pixel clocks and the sync signals are driven by Atlas™-II Processor or by the display, and whether the signals are active high or low. The display mode registers

sets the display data format and the output format. The RGBSEQ register determines the pixel sequence for those displays that requires one color at a time, and the BLANK registers define the output value for the blank region.

In order to accommodate different timings for HSYNC and VSYNC for different types of displays, two alternative signals can be generated to use as HSYNC and VSYNC signals. These signals are defined by the *ALT_HSYNC1* and *ALT_VSYNC1* registers. The bit8 and the bit9 of TIMCTRL register are used to select the source of HSYNC and VSYNC signals.

Also, to accommodate certain displays, the pixel clock can be masked by setting the mask pixel clock bits (bit10, bit11 of *LCD_TIMCTRL*). And the HSYNC signal can also be masked by setting the bit12 of TIMCTRL.

- **Sequence for RGB outputs (LCD_RGBSEQ) – 0x40**

Bit	Name	Default	Description
5:0 (R/W)	EVEN_RGBSEQ	6'h0	These bits represent the RGB output sequence for even-number lines.
11:6 (R/W)	ODD_RGBSEQ	6'h0	For displays requiring only one color per pixel, these bits represent the RGB output sequence for the odd-number lines. Two bits each represent a color: 00 – Red 01 – Green 10 – Blue For example, 000110 means the sequence is R-G-B-R-G-B, and 100100 represent B-G-R-B-G-R, etc. For displays requiring all three color values per pixel, this register has no effect.
31:12	-	20'h0	Reserved

- **Timing Control Register (LCD_TIMCTRL) – 0x44**

Bit	Name	Default	Description
0	-	1'b0	Reserved. This bit should always be 1'b0.
1 (R/W)	PCLK_IO	1'b0	Pixel clock master mode 1 = Atlas™-II Processor drives pixel clock 0 = display drives pixel clock
2 (R/W)	PCLK_POLAR	1'b0	Invert pixel clock. In master mode, invert the internal pixel clock before output. In slave mode, invert input pixel clock before use by internal logic. 1 = Invert pixel clock (i.e. pixel clock ½ phase off) 0 = Do not invert pixel clock
3 (R/W)	PCLK_EDGE	1'b0	Determines whether pixel output changes on the rising or falling edge of the internal pixel clock. 1 = Pixel changes on falling edge of clock. 0 = Pixel changes on rising edge of clock.
4 (R/W)	HSYNC_IO	1'b0	Horizontal sync signal master mode: 1 = Atlas™-II Processor drives horizontal sync 0 = display drives horizontal sync
5 (R/W)	HSYNC_POLAR	1'b0	Invert the horizontal sync signal. In master mode, invert the horizontal sync signal before output. In slave mode, invert the horizontal sync signal before used by internal logic. 1 = horizontal sync signal is active low.

			0 = horizontal sync signal is active high.
6 (R/W)	VSYNC_IO	1'b0	Vertical sync signal master mode: 1 = Atlas™-II Processor drives vertical sync 0 = display drives vertical sync
7 (R/W)	VSYNC_POLAR	1'b0	Invert the vertical sync signal. In master mode, invert the vertical sync signal before output. In slave mode, invert the vertical sync signal before used by internal logic. 1 = vertical sync signal is active low. 0 = vertical sync signal is active high.
8 (R/W)	HSYNC_SEL	1'b0	External HSYNC select: 0 – Use normal HSYNC signal (i.e. generated from HSYNC width and HSYNC period). 01 – Use 1 st alternate HSYNC signal (alt_hsync1). The alt_hsync1 register must be set to a valid value. Note: HSYNC signals with different timing may be required for different display modes, such as the STN modes
9 (R/W)	VSYNC_SEL	1'b0	External VYSNC select: 0 – Use normal VYSNC signal (i.e. generated from VYSNC width and VYSNC period). 1 – Use 1 st alternate VYSNC signal (alt_vsync1). The alt_vsync1 register must be set to a valid value. Note: VSYNC signals with different timing may be required for different display modes, such as the STN modes. This bit is not used when the ALT_VSYNC_VALID in LCD_ALT_VSYNC is set with 1. Please refer to Figure 10 for more details.
10 (R/W)	PCLK_MASK1	1'b0	Test Purpose register, software writes this with 0 Remark: Mask pixel clock control1: 1 = Pixel clock is masked to 0 whenever the pixel output data are invalid. Note: This signal only masks the pixel clock to the external pin. The internal pixel clock is not masked, so logic operation remains the same. Note: Masking the PIXCLK is useful for certain display modes, such as STN displays
11 (R/W)	PCLK_MASK2	1'b0	Test Purpose register, software writes this with 0 Remark: Mask pixel clock control2: 1 = Pixel clock is masked to 0 whenever the pixel output data in a line are valid, but is not masked for lines in vertical blank time. This bit is not valid in FRC valid mode.
12 (R/W)	HSYNC_MASK	1'b0	Test Purpose register, software writes this with 0 Remark: Mask HSYNC control: 1 = Disable the HSYNC during vertical blank time. 0 = Enable the HSYNC during vertical blank time.
13 (R/W)	CS_LATENCY	1'b0	Test Purpose register, software writes this with 0

			Remark: This bit can control the latency of CS signal for look up table (as the output delay of RAM for palette is rather large). 1 = When OSC_RATIO<9:0>=1, assert this bit 0 = Generally configuration
31:14	-	18'h0	Reserved

- **First alternate HSYNC control register (LCD_ALT_HSYNC) – 0x48**

Bit	Name	Default	Description
10:0 (R/W)	END	11'h0	End value: Define the pixel count after which the alternate signal will be de-asserted.
21:11 (R/W)	START	11'h0	Start value: Define the pixel count after which the alternate signal will be asserted.
22 (R/W)	VALID	1'b0	Valid 1 = Signal is valid. Use start and end values to define phase. 0 = Signal is not valid.
31:23	-	9'h0	Reserved

- **First alternate VSYNC control register (LCD_ALT_VSYNC) – 0x4C**

Bit	Name	Default	Description
10:0 (R/W)	START_PIXEL	11'h0	Start pixel value Define the location (line, pixel) at which the alternate signal will be asserted. The ALT_VSYNC pulse width is also decided by <i>LCD_VSYNC_WIDTH</i> register.
21:11 (R/W)	START_LINE	11'h0	Start line value Define the location (line, pixel) at which the alternate signal will be asserted. The ALT_VSYNC1 pulse width is also decided by <i>LCD_VSYNC_WIDTH</i> register.
22 (R/W)	ALT_VSYNC_VALID	1'b0	Valid 1 = Alternate VSYNC signal is valid along with the normal VSYNC defined by <i>LCD_VSYNC_WIDTH</i> . Use start pixel and line values to define phase. 0 = Signal is not valid. This bit gets higher priority than the VSYNC_SEL in <i>LCD_TIMCTRL</i> --- if this bit is on, both VSYNC will be used (Normally this mode will be used to generate the Odd and Even VSYNC in TV master mode)
31:23	-	9'h0	Reserved

NOTE: Please refer to the figure above for more details. This register controls the red part of the VSYNC signal.

- **OSC_PIXCLK divider ratio register (LCD_OSC_RATIO) – 0x50**

Bit	Name	Default	Description
9:0 (R/W)	DIV_RATIO	10'h0	Divider ratio: OSC_PIXCLK is obtained from SYSCLK divided by this value
11:10	-	2'b00	Reserved
12 (R/W)	HALF_DUTY	1'b0	1 = Generate OSC_PIXCLK of 50% duty cycle when the divider ratio (OSC_RATIO<9:0>) is odd.

			0 = OSC_PIXCLK is not 50% duty cycle clock when the divider ratio (OSC_RATIO<9:0>) is odd.
31:13	-	19'h0	Reserved

The pixel clock is divided from the system clock when Atlas drives the PIXCLK with this ratio as $Flick = F_{sysclk} / (DIV_RATIO + 1)$, and the minimum DIV_RATIO value is 1.

- **Timing Control Status Register (LCD_TIMING_STATUS) – 0x54**

This register is read only. This is for test purpose only, software shall ignore it.

Bit	Name	Default	Description
1:0(R)	RGB_SEQ_STA	2'b00	Current value of the RGB select index, for choosing which color to output based on the RGB sequence (for displays which require outputs of a single color at a time).
2(R)	OSD2_HVALID_STA	1'b0	Current value of the valid signal for the OSD2 region
3(R)	VOSD2_VALID_STA	1'b0	Current value of the vertical valid signal for OSD2 region.
4(R)	OSD_HVALID_STA	1'b0	Current value of the valid signal for the OSD region
5(R)	VOSD_VALID_STA	1'b0	Current value of the vertical valid signal for OSD region.
6(R)	SCN_VALID_STA	1'b0	Current value of the screen valid signal, i.e. the current line count and pixel count are all within the active region.
7(R)	VSCN_VALID_STA	1'b0	Current value of the vertical screen valid signal, i.e. the current line count is within the active region.
8(R)	VSYNC_STA	1'b0	Current value of the internal vertical sync signal
9(R)	HSYNC_STA	1'b0	Current value of the internal horizontal sync signal
10(R)	PCLK_STA	1'b0	Current value of internal pixel clock.
31:11	-	23'h0	Reserved

- **Blanking Register (LCD_BLANK) – 0x60**

Bit	Name	Default	Description
23:0 (R/W)	Blank Value	24'h0	Pixel value to be used for the inactive region: Bits 23:16 – R value Bits 15:8 – G value Bits 7:0 – B value Note: In BYPASSMODE or FRC mode, the LSB of this value is used for the inactive region (for 4-bit mode, it's 4 LSB, for 8-bit mode, it's 8 LSB, for 12-bit mode, it's 12 LSB, and for 16-bit mode, it's 16 LSB).
24 (R/W)	Blank Valid	1'b0	Use blank value 1 = Use blank value defined below when not in active region. 0 = Display the last pixel value when not in active region
31:25	-	7'h0	Reserved

- Display mode and format register (LCD_DISPLAYMODE) – 0x64

Bit	Name	Default	Description
0 (R/W)	FRAME_VALID	1'b0	Frame valid. When this bit is set, the frame is considered valid, i.e. the internal counters will count based on the pixel clocks and horizontal sync signals. When this bit is cleared, the frame is considered invalid and the counters do not change, and thus there is no output. Note that when the bit value changes in the middle of a frame, it does not take effect until the beginning of the next frame, i.e. there is a vertical sync signal.
3:1 (R/W)	SCN_BPP	1'b0	The current SCN layer source data bpp format: 100 – 2-bit per pixel 000 – 4-bit per pixel 001 – 8-bit per pixel 010 – 12-bit per pixel (4-bit R, 4-bit G, 4-bit B) 011 – 16-bit per pixel
5:4 (R/W)	OUT_FORMAT	1'b0	Output format: 00,01,10 – 6:6:6 18bit output 11 – 8-bit muxed RGB, with color component determined by the RGB sequence. Note: If in bypass mode, these two bits are ignored.
6 (R/W)	BYPASS	1'b0	Bypass mode: 1 = The data from the memory is directly sent to the output, bypassing either the palette lookup (for the 4- and 8-bit modes) or the bit shifting (for the 12- and 16-bit modes). When this bit is set to 1, the display mode in bits 3:1 still is used to specify whether each pixel is 4-bits, 8-bits, 12-bits, or 16-bits. 0 = Follow the display mode in bits 3:1. Note: One of the main uses for the bypass mode is for STN type displays. Note: The palettes for the OSD and the blank values are also bypassed, i.e. they are not broken up into RGB components, and bits <15:0> are used directly to determine the output. Note: In 12-bit or 16-bit bypass mode, OSD mixing is no longer valid. The OSD mixer value must be either 000 or 100 (i.e. 100% screen or 100% OSD).
7 (R/W)	DUAL_SCAN	1'b0	Dual scan mode: 1 = Dual-scan for STN LCD. The channel for the OSD display is used for the dual-scan's second channel (that is, OSD is invalid in this mode). This mode is generally used with the bypass mode and with the 4-bit or 8-bit output format. 0 = Dual-scan mode is invalid.
8	OSD_DW_BLE	1'b0	Big/Little Endian selection of byte for OSD image

(R/W)			data (the second channel image data in dual scan mode) from SDRAM: 1 = Little Endian 0 = Big Endian
9 (R/W)	SCN_DW_BLE	1'b0	Big/Little Endian selection of byte for screen image data from SDRAM: 1 = Little Endian: MSB- byte3, byte2, byte1, byte0-LSB 0 = Big Endian: MSB- byte0, byte1, byte2, byte3-LSB
10 (R/W)	OSD_BYTE_BLE	1'b0	Big/Little Endian selection in the OSD image byte data (the second channel image data in dual scan mode; only for 4-bit/pixel, 2-bit/pixel): 1 = Little Endian 0 = Big Endian
11 (R/W)	SCN_BYTE_BLE	1'b0	Big/Little Endian selection in the screen image byte data (only for 4-bit/pixel, 2-bit/pixel): 1 = Little Endian 0 = Big Endian
13:12 (R/W)	SCN_16BPP_FORMAT	2'b00	Data format for 16-bit per pixel mode: 00 – 6 : 5 : 5 16-bit (i.e. 6 bit R, 5 bit G, 5 bit B) 01 – 5 : 6 : 5 16-bit 10 – 5 : 5 : 6 16-bit These two bits are only valid in 16-bit per pixel mode.
14 (R/W)	OSD2_DW_BLE	1'b0	Big/Little Endian selection of byte for OSD2 image data (the second channel image data in dual scan mode) from SDRAM: 1 = Little Endian 0 = Big Endian
15 (R/W)	OSD2_BYTE_BLE	1'b0	Big/Little Endian selection in the OSD2 image byte data (only for 4-bit/pixel, 2-bit/pixel): 1 = Little Endian 0 = Big Endian
31:16	-	16'h0	Reserved

6.2.4.4 FRC (frame rate control) Registers

These registers are used to set the two LFSRs (LINE_LFSR and PIX_LFSR). The two LFSRs are used to reduce the flickering generated by FRC operation.

- **Line Linear-Feedback-Shift register (LCD_LINE_LFSR) – 0x68**

Bit	Name	Default	Description
8:0 (R/W)	LFSR_SEED	9'h0	Initial value of the line LFSR (LFSR advanced by HSYNC).
17:9 (R/W)	LFSR_VAL	9'h0	Algorithm used for the LFSR. If bit = 1 then that bit of the shift register is used in the sum. For instance, for the algorithm $g(x) = x^9 + x^4 + 1$, program the algorithm value to 100001000.
19:18 (R/W)	PARALLEL_FORMAT	2'h0	The following parallel format definition is described for single-scan panel. And for monochrome STN, only 4bit and 8bit parallel are supported.

			00 = 8bit parallel output 01 = 4bit parallel output 10 = 16bit parallel output
20 (R/W)	MSTN	1'b0	1 = Monochrome STN mode valid 0 = Monochrome STN mode invalid
21 (R/W)	CSTN	1'b0	1 = Color STN mode valid 0 = Color STN mode invalid
31:22	-	10'h0	Reserved

- **Pixel Linear-Feedback-Shift register (LCD_PIX_LFSR) – 0x6C**

Bit	Name	Default	Description
14:0 (R/W)	LFSR_SEED	15'h0	Initial value of the pixel LFSR (LFSR advanced by PIXCLK)
29:15 (R/W)	LFSR_VAL	15'h0	Algorithm used for the LFSR. If bit = 1 then that bit of the shift register is used in the sum. For instance, for the algorithm $g(x) = x^{11} + x^2 + 1$, program the algorithm value to 000010000000010.
31:30	-	2'b00	Reserved

- **Frame rate modulation control register (LCD_FMOD) – 0x70**

Bit	Name	Default	Description
6:0 (R/W)	FMOD_PERIOD	7'h0	Flat-panel modulation period: This parameter specifies half of the period of the FMOD signal. Programmed value = (actual period / 2) – 1.
7 (R/W)	FMOD_CLK	1'b0	Clock selection for FMOD signal generation: 1 = Internal VSYNC generates FMOD 0 = Internal HSYNC generates FMOD
8 (R/W)	SYN_RESET	1'b0	FMOD synchronous reset control (only valid when using internal HSYNC to generate FMOD): 1 = FMOD polarity is reset at the beginning of frame 0 = FMOD polarity is not reset at the beginning of frame
9 (R/W)	L_BIAS_POLAR	1'b0	1 = Invert FMOD/SCN_ACTIVE signal before output 0 = Do not invert FMOD/SCN_ACTIVE signal
10 (R/W)	L_BIAS_SEL	1'b0	1 = Output FMOD signal to the L_BIAS Pin 0 = Output SCN_ACTIVE signal to the L_BIAS Pin
31:11	-	21'h0	Reserved

6.2.4.5 YUV Output Registers

These registers allow the support of YUV output instead of RGB output, mainly targeted towards video encoders. The first set of registers contains the coefficients to use to convert the RGB outputs to YUV outputs. The conversion is a 3x3 matrix multiplication as follows:

$$\begin{array}{rcl}
 Y & = & C11 \ C12 \ C13 \\
 U & = & C21 \ C22 \ C23 \\
 V & = & C31 \ C32 \ C33
 \end{array}
 \quad * \quad
 \begin{array}{l}
 R \\
 G \\
 B
 \end{array}$$

Note that this multiplication is a signed multiplication. Thus, the coefficients must be 8-bit signed values. In addition, the RGB values may have to be turned to signed values by inverting the MSB of the each

component. Also, the YUV output is initially signed value, so it may be necessary to invert the MSB again to output unsigned values. These invert options and other control signals for the YUV output are in the YUV_CTRL register.

The coefficients' format is as following:

Bit7 - the sign bit

Bit6-0 - the fraction bits

- **RGB to YUV Coefficient 1 (LCD_RGB_YUV_COEF1) – 0x74**

Bit	Name	Default	Description
23:16 (R/W)	COEF1	8'h0	Coefficient 11 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
15:8 (R/W)	COEF2	8'h0	Coefficient 12 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
7:0 (R/W)	COEF3	8'h0	Coefficient 13 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
31:24	-	8'h0	Reserved

The suggested coefficients are:

0x264B0E

, which indicates:

$$Y = 0.296875 * R + 0.5859375 * G + 0.109375 * B$$

- **RGB to YUV Coefficient 2 (LCD_RGB_YUV_COEF2) – 0x78**

Bit	Name	Default	Description
23:16 (R/W)	COEF1	8'h0	Coefficient 21 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
15:8 (R/W)	COEF2	8'h0	Coefficient 22 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
7:0 (R/W)	COEF3	8'h0	Coefficient 23 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
31:24	-	8'h0	Reserved

The suggested coefficients are:

0xE6D640

, which indicates:

$$U = (-0.1640625) * R + (-0.328125) * G + 0.5 * B$$

- **RGB to YUV Coefficient 3 (LCD_RGB_YUV_COEF3) – 0x7C**

Bit	Name	Default	Description
23:16 (R/W)	COEF1	8'h0	Coefficient 31 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
15:8 (R/W)	COEF2	8'h0	Coefficient 32 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
7:0 (R/W)	COEF3	8'h0	Coefficient 33 for the RGB to YUV conversion matrix. This value should be an 8-bit signed value.
31:24	-	8'h0	Reserved

The suggested coefficients are:

0x40CBF6

, which indicates:

$$V = 0.5 * R + (-0.4140625) * G + (-0.078125) * B$$

- **RGB to YUV conversion control register (LCD_YUV_CTRL) – 0x80**

Bit	Name	Default	Description
2:0 (R/W)	YUV_U_CONV	3'b000	Invert the MSB of the YUV components to convert signed results to unsigned outputs. Bit 2 – invert Y Bit 1 – invert U Bit 0 – invert V ITU601 accepts positive YUV, and Y is always positive while UV is signed after conversion, so UV needs inversion. So usually set the as 3'b011
5:3 (R/W)	RGB_S_CONV	3'b000	Invert the MSB of the RGB components to convert signed to unsigned when the input RGB is signed Bit 5 – invert R Bit 4 – invert G Bit 3 – invert B Remark: the rgb2yuv conversion accept unsigned RGB and signed coefficients only, so here keep this as 3'b000
6 (R/W)	UV_SEL	1'b0	1 = For 8-bit 4:2:2 output, even pixels are U 0 = For 8-bit 4:2:2 output, even pixels are V
7 (R/W)	YUV_FORMAT	1'b0	1 = YUV output is 4:2:2 8-bit output (double clock). 0 = YUV output is 4:4:4 16-bit output
8 (R/W)	RGB_YUV	1'b0	1 = Do RGB to YUV conversion 0 = Bypass RGB to YUV conversion
9 (R/W)	FAST_CLK	1'b0	1 = Use high-speed SYSCLK. When SYSCLK is higher than 108MHz (13.5MHz x 8), recommend setting this bit to make the internal multipliers work normally. 0 = When SYSCLK is lower than 108MHz, this bit must be 0.
10 (R/W)	2X_HSCALE	1'b0	1 = 2x scale horizontally 0 = no scale This bit is mainly used when doing TV out, 'cause the frame buffer will be 2 times scaled vertically (the odd and even fields show the same frame buffer)
11 (R/W)	EVEN_UV	1'b0	1=Keep the even pixel UV (U1V1 mode) 0=keep the even pixel U and odd pixel V(U1V2 mode)
12 (R)	EVENFIELD	1'b0	This bit only used when in TV mode and the outside TV encoder chip drives HSYNC and VYSNC 1=Current field is the even field of a frame 0=Current field is the odd field of a frame
31:13	-	21'h0	Reserved

0

- **Screen control for external TV encoder register (LCD_SCN_TVFIELD) – 0x84**

Bit	Name	Default	Description
10:0 (R/W)	SCN_VEND1	11'h0	SCN_VEND1: Vertical End Position (in line number) for the active TV display.
11	-	1'b0	Reserved
22:12 (R/W)	SCN_VSTART1	11'h0	SCN_VSTART1: Vertical Start Position (in line number) for the active TV display.

23	-	1'b0	Reserved
24 (R/W)	TV_F_VALID	1'b0	1 = Also use SCN_VSTART1 and SCN_VEND1 values to control the active Screen. 0 = SCN_VSTART1 and SCN_VEND1 are not active. This is used when doing TV 2-fields mode – define the EVEN field position, refer to the figure above.
25 (R/W)	HALF_PIXCLK	1'b0	1 = The input PIXCLK (when external TV encoder drives the PIXCLK) or the clock after the divider (when Atlas drives the PIXCLK) is divided by 2, after which it is used as the internal PIXCLK. This is mainly used for external TV encoder (27MHz clock). 0 = The frequency of internal PIXCLK remain unchanged.
26 (R/W)	INI_HALF_PIXCLK	1'b0	The bit sets the reset value of initial value of the 1/2 clock. This is mainly used for external TV encoder (27MHz clock input). 1 = Initial value of the 1/2 clock at the beginning of VSYNC is low. 0 = Initial value of the 1/2 clock at the beginning of VSYNC is high.
27 (R/W)	SYN_RESET	1'b0	1 = When input PIXCLK is divide by 2, the initial value of the 1/2 clock is reset at the beginning of VSYNC. This is mainly used for external TV encoder (27MHz clock input). 0 = When input PIXCLK is divide by 2, the initial value of the 1/2 clock is not reset at the beginning of VSYNC.
31:28	-	4'h0	Reserved

6.2.4.6 Alpha Blending & Color Key Control Registers

These registers are used for OSD and OSD2 alpha blending and color key functions. Alpha blending is a means to blend two layers together. The ratios between two layers are defined as below. And for the color key function, two registers (CKEYB and CKEYS) are used to specify a color key range. For OSD and OSD2 color key registers, if a color of OSD or OSD2 is between the CKEYB and the CKEYS, it will be transparent and the pixel from the main display layers will be shown instead. While for SCN color key registers, if the color of SCN is in the range defined by CKEYB and CKEYS, the color is transparent and the pixel from the OSD layer will be shown instead.

The ratios between main display layer and the alpha blending layer are as follows:

Alpha value 0000 – 100% main display layer value

Alpha value 0001 – 15/16 main display layer value + 1/16 alpha blending layer value

Alpha value 0010 – 14/16 main display layer value + 2/16 alpha blending layer value

Alpha value 0011 – 13/16 main display layer value + 3/16 alpha blending layer value

.....

Alpha value 1101 – 3/16 main display layer value + 13/16 alpha blending layer value

Alpha value 1110 – 2/16 main display layer value + 14/16 alpha blending layer value

Alpha value 1111 – 100% alpha blending layer value

- **OSD and OSD2 Alpha blending control register (LCD_OSDALPHA) – 0x88**

Bit	Name	Default	Description
3:0 (R/W)	OSD_ALPHA_VAL	4'h0	4-bit planar alpha value to blend all pixels on OSD layer to SCN layer. This value is used or not will be selected

			by bit 6 of LCD_OSDALPHA register.
5:4	OSD_BPP	2'b0	OSD data format for alpha blending: 00 – 2bit/pixel (2-bit index color) 01 – 4bit/pixel (4-bit index color) 10 – 8bit/pixel (8-bit index color when OSD_ALPHA_SEL = 0, or 4-bit alpha, 4-bit index color when OSD_ALPHA_SEL = 1) 11 – 16bit/pixel (16-bit RGB 5:6:5 color when OSD_ALPHA_SEL = 0, or 4-bit alpha, 4-bit Red, 4-bit Green, 4-bit Blue when OSD_ALPHA_SEL = 1)
6 (R/W)	OSD_ALPHA_SEL	1'b0	1 = Select the valid alpha value as the 4-bit alpha data from the 8bpp or 16bpp image data (only valid for 8bpp or 16bpp data format) 0 = Select OSD_ALPHA_VAL as the valid alpha value when the data format is 4bpp, 8bpp or 16bpp. Remark: when OSD_BPP is 4bpp, this bit must be zero, which is only the OSD_ALPHA_VAL will be used as the alpha value and if not, the OSD alpha value will be zero. For 2bpp data format, keep this bit as zero but the MIX_RATIO of OSD Palette registers will be used as the valid alpha value.
15:7	-	9'h0	Reserved
19:16 (R/W)	OSD2_ALPHA_VAL	4'h0	4-bit planar alpha value to blend all pixels on OSD2 layer to SCN layer or SCN+OSD layer. This value is used or not will be selected by bit 6 of LCD_OSD2ALPHA register.
21:20	OSD2_BPP	2'b0	OSD2 data format for alpha blending: 00 – 2bit/pixel (2-bit index color) 01 – 4bit/pixel (4-bit index color) 10 – 8bit/pixel (8-bit index color when OSD2_ALPHA_SEL = 0, or 4-bit alpha, 4-bit index color when OSD2_ALPHA_SEL = 1) 11 – 16bit/pixel (16-bit RGB 5:6:5 color when OSD2_ALPHA_SEL = 0, or 4-bit alpha, 4-bit Red, 4-bit Green, 4-bit Blue when OSD2_ALPHA_SEL = 1)
22 (R/W)	OSD2_ALPHA_SEL	1'b0	1 = Select the valid alpha value as the 4-bit alpha data from the 8bpp or 16bpp image data (only valid for 8bpp or 16bpp data format) 0 = Select the OSD2_ALPHA_VAL as the valid alpha value when the data format is 4bpp, 8bpp or 16bpp. Remark: when OSD2_BPP is 4bpp, this bit must be zero, which is only the OSD2_ALPHA_VAL will be used as the alpha value and if not, the OSD2 alpha value will be zero. For 2bpp data format, keep this bit as zero but the MIX_RATIO of OSD Palette registers will be used as the valid alpha value.
31:23	-	9'h0	Reserved

- **SCN color key¹ big RGB value (LCD_SCNKEYB) – 0x8C**

¹ Color Key is a color or a range of colors defined as transparent.

The SCN color key register is for test purpose only, software should not enable this.

Bit	Name	Default	Description
17:0 (R/W)	SCN_CKEYB	18'h0	Bigger value of color key for SCN channel display.
19:18	-	2'b00	Reserved
20 (R/W)	SCN_CKEY_EN	1'b0	1 = Color key function for SCN is valid 0 = Color key function for SCN is invalid
21 (R/W)	SCN_CKEY_SEL	1'b0	1 = 8-bit index color mode. In this mode, only bit [7:0] of SCN_CKEYB and SCN_CKEYS are valid for the 8-bit index. 0 = 18bit 6:6:6 color mode. This can be used for 2-bit index, 4-bit index, 8-bit index, 16-bit(5:6:5, 5:5:6, 6:5:5), and so on. This mode is flexible, but needs to calculate SCN_CKEYB and SCN_CKEYS values according to different data formats. For 16-bit data format, the 18bit CKEYB or CKEYS are calculated by expanding Red, Green, or Blue to 6bits, and set the expanded LSB to 1'b0.
31:22	-	10'h0	Reserved

- **SCN color key small RGB value (LCD_SCNCKEYS) – 0x90**

This register is for test purpose only.

Bit	Name	Default	Description
17:0 (R/W)	SCN_CKEYS	18'h0	Smaller value of color key for SCN channel display.
31:18	-	14'h0	Reserved

- **OSD color key big RGB value (LCD OSDCKEYB) – 0x94**

Bit	Name	Default	Description
17:0 (R/W)	OSD_CKEYB	18'h0	Bigger value of color key for OSD channel display.
19:18	-	14'h0	Reserved
20 (R/W)	OSD_CKEY_EN	1'b0	1 = Color key function for OSD is valid 0 = Color key function for OSD is invalid
21 (R/W)	OSD_CKEY_SEL	1'b0	1 = 8-bit index color mode. In this mode, only bit [7:0] of OSD_CKEYB and OSD_CKEYS are valid for the 8-bit index. 0 = 18bit 6:6:6 color mode. This can be used for 2-bit index, 4-bit index, 8-bit index, 16-bit(5:6:5, 5:5:6, 6:5:5), and so on. This mode is flexible, but needs to calculate OSD_CKEYB and OSD_CKEYS values according to different data formats. For 16-bit data format, the 18bit CKEYB or CKEYS are calculated by expanding Red, Green, or Blue to 6bits, and set the expanded LSB to 1'b0.
31:22	-	10'h0	Reserved

- **OSD color key small RGB value (LCD OSDCKEYS) – 0x98**

Bit	Name	Default	Description
17:0 (R/W)	OSD_CKEYS	18'h0	Smaller value of color key for OSD channel display.
31:18	-	14'h0	Reserved

- **OSD2 color key big RGB value (LCD_OSD2CKEYB) – 0x9C**

Bit	Name	Default	Description
17:0 (R/W)	OSD2_CKEYB	18'h0	Bigger value of color key for OSD2 channel display.
19:18	-	2'b00	Reserved
20 (R/W)	OSD2_CKEY_EN	1'b0	1 = Color key function for OSD2 is valid 0 = Color key function for OSD2 is invalid
21 (R/W)	OSD2_CKEY_SEL	1'b0	1 = 8-bit index color mode. In this mode, only bit [7:0] of OSD2_CKEYB and OSD2_CKEYS are valid for the 8-bit index. 0 = 18bit 6:6:6 color mode. This can be used for 2-bit index, 4-bit index, 8-bit index, 16-bit(5:6:5, 5:5:6, 6:5:5), and so on. This mode is flexible, but needs to calculate OSD2_CKEYB and OSD2_CKEYS values according to different data formats. For 16-bit data format, the 18bit CKEYB or CKEYS are calculated by expanding Red, Green, or Blue to 6bits, and set the expanded LSB to 1'b0.
31:22	-	10'h0	Reserved

- **OSD2 color key small RGB value (LCD_OSD2CKEYS) – 0xA0**

Bit	Name	Default	Description
17:0 (R/W)	OSD2_CKEYS	18'h0	Smaller value of color key for OSD2 channel display.
31:18	-	14'h0	Reserved

6.2.4.7 OSD Palette

The OSD palette (for OSD or OSD2) is a four-entry palette. The index is determined by the 2-bit OSD pixel value when the OSD is in its active region. Each palette entry contains the RGB value for the pixel and the mixing ratio that should be used. Entry 0 is used when the OSD pixel value equals 2'b00, entry 1 is used when the OSD pixel value equals 2'b01, etc. This palette is valid for both the first OSD (OSD) and the second OSD (OSD2). This Palette is only used by 2bpp mode OSD(s).

- **OSD Palette Entry 0 (LCD_OSDPAL0) – 0xA4**

Note: For bypass mode, the LSB's of this register is used as the OSD value directly (the number of bits corresponds to the display mode). Mixing is performed normally for the 4-bit and 8-bit bypass mode, but is not allowed for 12-bit and 16-bit bypass mode (same as the following 3 registers).

Bit	Name	Default	Description
7:0 (R/W)	BLUE	8'h0	Blue value for the OSD pixel type 0
15:8 (R/W)	GREEN	8'h0	Green value for the OSD pixel type 0
23:16	RED	8'h0	Red value for the OSD pixel type 0

(R/W)			
26:24 (R/W)	MIX_RATIO	3'b000	Mixing value for the OSD pixel type 0 000 – 100% screen value 001 – $\frac{3}{4}$ screen value + $\frac{1}{4}$ OSD value 010 – $\frac{1}{2}$ screen value + $\frac{1}{2}$ OSD value 011 – $\frac{1}{4}$ screen value + $\frac{3}{4}$ OSD value 100 – 100% OSD value Note: For 12-bit and 16-bit bypass mode, only 100% screen and 100% OSD are allowed.
31:27	-	5'h0	Reserved

- **OSD Palette Entry 1 (LCD_OSDPAL1) – 0xA8**

Bit	Name	Default	Description
7:0 (R/W)	BLUE	8'h0	Blue value for the OSD pixel type 1
15:8 (R/W)	GREEN	8'h0	Green value for the OSD pixel type 1
23:16 (R/W)	RED	8'h0	Red value for the OSD pixel type 1
26:24 (R/W)	MIX_RATIO	3'b000	Mixing value for the OSD pixel type 1 000 – 100% screen value 001 – $\frac{3}{4}$ screen value + $\frac{1}{4}$ OSD value 010 – $\frac{1}{2}$ screen value + $\frac{1}{2}$ OSD value 011 – $\frac{1}{4}$ screen value + $\frac{3}{4}$ OSD value 100 – 100% OSD value Note: For 12-bit and 16-bit bypass mode, only 100% screen and 100% OSD are allowed.
31:27	-	5'h0	Reserved

- **OSD Palette Entry 2 (LCD_OSDPAL2) – 0xAC**

Bit	Name	Default	Description
7:0 (R/W)	BLUE	8'h0	Blue value for the OSD pixel type 2
15:8 (R/W)	GREEN	8'h0	Green value for the OSD pixel type 2
23:16 (R/W)	RED	8'h0	Red value for the OSD pixel type 2
26:24 (R/W)	MIX_RATIO	3'b000	Mixing value for the OSD pixel type 2 000 – 100% screen value 001 – $\frac{3}{4}$ screen value + $\frac{1}{4}$ OSD value 010 – $\frac{1}{2}$ screen value + $\frac{1}{2}$ OSD value 011 – $\frac{1}{4}$ screen value + $\frac{3}{4}$ OSD value 100 – 100% OSD value Note: For 12-bit and 16-bit bypass mode, only 100% screen and 100% OSD are allowed.
31:27	-	5'h0	Reserved

- **OSD Palette Entry 3 (LCD_OSDPAL3) – 0xB0**

Bit	Name	Default	Description
7:0 (R/W)	BLUE	8'h0	Blue value for the OSD pixel type 3
15:8 (R/W)	GREEN	8'h0	Green value for the OSD pixel type 3
23:16 (R/W)	RED	8'h0	Red value for the OSD pixel type 3
26:24	MIX_RATIO	3'b000	Mixing value for the OSD pixel type 3

(R/W)			000 – 100% screen value 001 – $\frac{3}{4}$ screen value + $\frac{1}{4}$ OSD value 010 – $\frac{1}{2}$ screen value + $\frac{1}{2}$ OSD value 011 – $\frac{1}{4}$ screen value + $\frac{3}{4}$ OSD value 100 – 100% OSD value Note: For 12-bit and 16-bit bypass mode, only 100% screen and 100% OSD are allowed.
31:27	-	5'h0	Reserved

6.2.4.8 FIFO Control Registers

The FIFO's in the LCD controller interface, the screen FIFO and the two OSD FIFOs will all generate DMA requests to the bus master. The requests generated are a 2-bit request level, with 3 being critical request, 2 being high-level request, and 1 being low-level request. The request level depends on the fullness of the FIFO. The FIFO registers define three watermarks that help to determine the current request level for each FIFO.

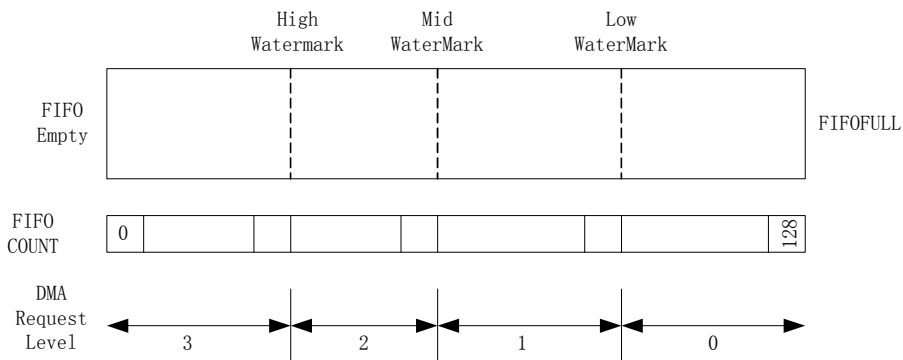


Figure 25. Screen and OSD FIFOs Request Level

The DMA requests to the bus interface are generated based on the FIFO fullness counter. Atlas™ Processor provides two methods of requests generation selected by bit24 of SCNFIFO register. When this bit is 0, four request values (00, 01, 10, 11) are valid. When the FIFO reaches the low water mark, then it will generate a low-level request (01). When the FIFO reaches the middle water mark, then it will generate a mid-level request (10). When the FIFO reaches the high water mark, then it will generate a high-level request (11). When bit24 is equal to 1, only high and low water marks are useful, and only two request values (00, 11) are valid. When the FIFO fullness is less than the high water mark, it will generate a high-level request (11). When the FIFO fullness is over the low watermark, the request stops (00). The second method will lead much more continuous data dumps from memory than the first method. It is obvious that enlarge any of the 3 watermark will make the FIFO more aggressive.

- Screen FIFO Control Register (LCD_SCNFIFO) – 0xC0

Bit	Name	Default	Description
6:0 (R/W)	SCN_LO_CHK	7'h60	Value for the low request watermark for the screen FIFO. If the FIFO (FIFO count) is filled beyond this mark, then the request stops. If the FIFO is read below this point, then the request becomes low-level request.
7	-	1'b0	Reserved
14:8 (R/W)	SCN_MI_CHK	7'h40	Value for the middle request watermark for the screen FIFO. If the FIFO is filled beyond this mark, then the request drops to low-level request. If the FIFO is read below this point,

			then the request becomes high-level request.
15	-	1'b0	Reserved
22:16 (R/W)	SCN_HI_CHK	7'h20	Value for the high request watermark for the screen FIFO. If the FIFO is filled beyond this mark, then the request drops to high-level request. If the FIFO is read below this point, then the request becomes critical.
23	-	1'b0	Reserved
24 (R/W)	SCN_REQ_SEL	1'b0	Selection of request generation method for SCN FIFO: 1 = Only high request and low request watermark are useful, request value can be as 2'b00 and 2b'11.If the FIFO is read below high request watermark, then the request becomes critical, and if the FIFO is filled beyond low request watermark, then the request stops. As in critical request status, the DMA bandwidth is a little lower, using this mode will save some bandwidth. 0 = Normal request generation. Three watermarks are all useful, request value can be 2'b00, 2'b01, 2b'10, 2b'11.
31:25	-	7'h0	Reserved

- **OSD FIFO Control Register (LCD_OSDFIFO) – 0xC4**

Bit	Name	Default	Description
6:0 (R/W)	OSD_LO_CHK	7'h60	Value for the low request watermark for the OSD FIFO. If the FIFO is filled beyond this mark, then the request stops. If the FIFO is read below this point, then the request becomes low-level request.
7	-	1'b0	Reserved
14:8 (R/W)	OSD_MI_CHK	7'h40	Value for the middle request watermark for the OSD FIFO. If the FIFO is filled beyond this mark, then the request drops to low-level request. If the FIFO is read below this point, then the request becomes high-level request.
15	-	1'b0	Reserved
22:16 (R/W)	OSD_HI_CHK	7'h20	Value for the high request watermark for the OSD FIFO. If the FIFO is filled beyond this mark, then the request drops to high-level request. If the FIFO is read below this point, then the request becomes critical.
23	-	1'b0	Reserved
24 (R/W)	OSD_REQ_SEL	1'b0	Selection of request generation method for OSD FIFO: 1 = Only high request and low request watermark are useful, request value can be as 2'b00 and 2b'11.If the FIFO is read below high request watermark, then the request becomes critical, and if the FIFO is filled beyond low request watermark, then the request stops. As in critical request status, the DMA bandwidth is a little lower, using this mode will save some bandwidth. 0 = Normal request generation. Three watermarks are all useful, request value can be as 2'b00, 2'b01, 2b'10, 2b'11.
31:25	-	7'h0	Reserved

- **OSD2 FIFO Control Register (LCD_OSD2FIFO) – 0xC8**

Bit	Name	Default	Description
6:0	OSD2_LO_CHK	7'h60	Value for the low request watermark for the OSD2 FIFO. If

(R/W)			the FIFO is filled beyond this mark, then the request stops. If the FIFO is read below this point, then the request becomes low-level request.
7	-	1'b0	Reserved
14:8 (R/W)	OSD2_MI_CHK	7'h40	Value for the middle request watermark for the OSD2 FIFO. If the FIFO is filled beyond this mark, then the request drops to low-level request. If the FIFO is read below this point, then the request becomes high-level request.
15	-	1'b0	Reserved
22:16 (R/W)	OSD2_HI_CHK	7'h20	Value for the high request watermark for the OSD2 FIFO. If the FIFO is filled beyond this mark, then the request drops to high-level request. If the FIFO is read below this point, then the request becomes critical.
23	-	1'b0	Reserved
24 (R/W)	OSD2_REQ_SEL	1'b0	Selection of request generation method for OSD2 FIFO: 1 = Only high request and low request watermark are useful, request value can be as 2'b00 and 2b'11. If the FIFO is read below high request watermark, then the request becomes critical, and if the FIFO is filled beyond low request watermark, then the request stops. As in critical request status, the DMA bandwidth is a little lower, using this mode will save some bandwidth. 0 = Normal request generation. Three watermarks are all useful, request value can be 2'b00, 2'b01, 2b'10, 2b'11.
31:25	-	7'h0	Reserved

- **Screen FIFO Write Suppress Register (LCD_SCNFIFO_SUPPRESS) – 0xD8**

Since the only type of DMA that the LCD supports is a 4-DWORD burst, the LCD can only DMA data in 4-DWORD increments. However, if the screen size or location requires the start of a new line in the middle of a burst, the DMA interface will write extra DWORD's to the FIFO and causes data mismatch. In order to avoid this, we can use this register to suppress the extra writes to the FIFO at the end of a line. To do this, write the number of VALID DWORD writes to bits <23:12> and the number of writes to SUPPRESS to bits <11:0>. For example, to write 17 DWORD per line, we will request 5 bursts (20 DWORD's) from the DMA, and suppress the last 3 DWORD's. Thus, we will set this register to 0x011_003. It is a must that the frame buffer's start address must lay on a DMA 4 DWORD's burst boundary, and the frame buffer's width is also a multiple of the DMA.

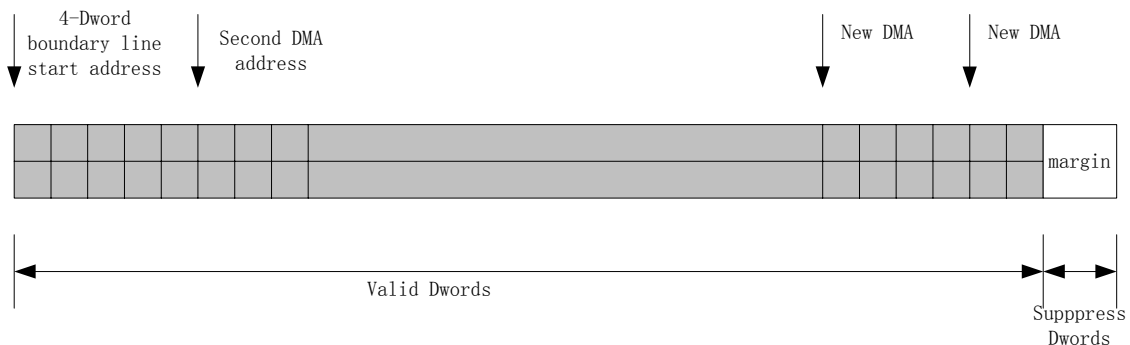


Figure 26. FIFO Suppress

Bit	Name	Default	Description
-----	------	---------	-------------

11:0 (R/W)	SUP_DW_NUM	12'h0	The number of DWORD writes to the FIFO that must be suppressed at the end of each DMA line. For example, if bits <23:12> is set to 17, and bits <11:0> is set to 3, then for every 17 DWORD is that is written to the FIFO, 3 DWORD writes are suppressed.
23:12 (R/W)	VALID_DW_NUM	12'h0	The number of valid DWORD writes to the FIFO per line.
31:24	-	8'h0	Reserved

NOTE: Keeping this register as zero means all the DWORD's are valid - no suppress

- **OSD FIFO Write Suppress Register (LCD_OSDFIFO_SUPPRESS) – 0xDC**

Similar to *LCD_SCNFIFO_SUPPRESS*.

Bit	Name	Default	Description
11:0 (R/W)	SUP_DW_NUM	12'h0	The number of DWORD writes to the FIFO that must be suppressed at the end of each DMA line. For example, if bits <23:12> is set to 17, and bits <11:0> is set to 3, then for every 17 DWORD is that is written to the FIFO, 3 DWORD writes are suppressed.
23:12 (R/W)	VALID_DW_NUM	12'h0	The number of valid DWORD writes to the FIFO per line.
31:24	-	8'h0	Reserved

- **OSD2 FIFO Write Suppress Register (LCD_OSD2FIFO_SUPPRESS) – 0xE0**

Bit	Name	Default	Description
11:0 (R/W)	SUP_DW_NUM	12'h0	The number of DWORD writes to the FIFO that must be suppressed at the end of each DMA line. For example, if bits <23:12> is set to 17, and bits <11:0> is set to 3, then for every 17 DWORD is that is written to the FIFO, 3 DWORD writes are suppressed.
23:12 (R/W)	VALID_DW_NUM	12'h0	The number of valid DWORD writes to the FIFO per line.
31:24	-	8'h0	Reserved

- **FIFO Address Reset Register (LCD_FIFOADD_RST) – 0xE4**

Bit	Name	Default	Description
0 (R/W)	SCNFIFO_ADDRST	1'b0	Soft reset of SCN FIFO addresses: 1 = Setting this bit to 1 will reset the read and the write address of SCN FIFO. At the same time, the fifo_full and the FIFO request status will be also reset. It can be used for recovery from abnormal operations such as FIFO underflow, overflow, and so on. 0 = After reset, this bit should be set back to 0 for normal operation
1 (R/W)	OSDFIFO_ADDRST	1'b0	Soft reset of OSD FIFO addresses: 1 = Setting this bit to 1 will reset the read and the write address of OSD FIFO. At the same time, the fifo_full and the FIFO request status will be also reset. It can be used for recovery from abnormal operations such as FIFO

			underflow, overflow, and so on. 0 = After reset, this bit should be set back to 0 for normal operation
2 (R/W)	OSD2FIFO_ADDRST	1'b0	Soft reset of OSD2 FIFO addresses: 1 = Setting this bit to 1 will reset the read and the write address of OSD2 FIFO. At the same time, the fifo_full and the FIFO request status will be also reset. It can be used for recovery from abnormal operations such as FIFO underflow, overflow, and so on. 0 = After reset, this bit should be set back to 0 for normal operation Make sure the DMA's are not working before doing software reset.
31:3	-	29'h0	Reserved

6.2.4.9 DMA Address Registers

The DMA address is generated by an address generator based on some parameters defining a 2-dimensional DMA. The X size determines how many consecutive 4-DWORD bursts are in a line, the Y size determines how many lines are required for the current display, and the SKIP value determines the number of BYTE's to skip at the end of a line to get to the beginning of the next line. Finally, the base address specifies the starting address of the DMA. The DMA is activated by setting the starting address, so it is necessary to set the base address register last, after all the other registers have been set.

With this type of definition, any subset of a larger display memory can be specified to be used as the current display memory. This may be especially useful for zooming or panning in a large memory.

The OSD and screen DMA's are configured identically.

A bit in the OSD2_BASE, OSD_BASE and SCN_BASE register allows each DMA to operate in continuous mode. This means that the DMA will repeat itself automatically (with the exact same settings) after it is completed for continuous display. This saves the software from always having to generate DMA's. Clearing this bit can stop the continuous DMA.

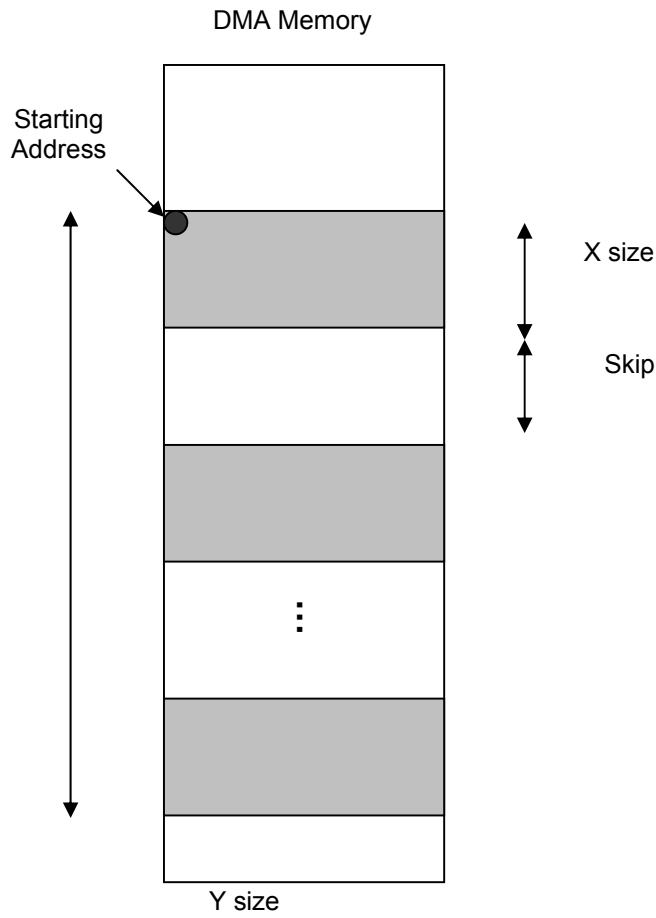


Figure 27. LCD Controller DMA Memory Layout

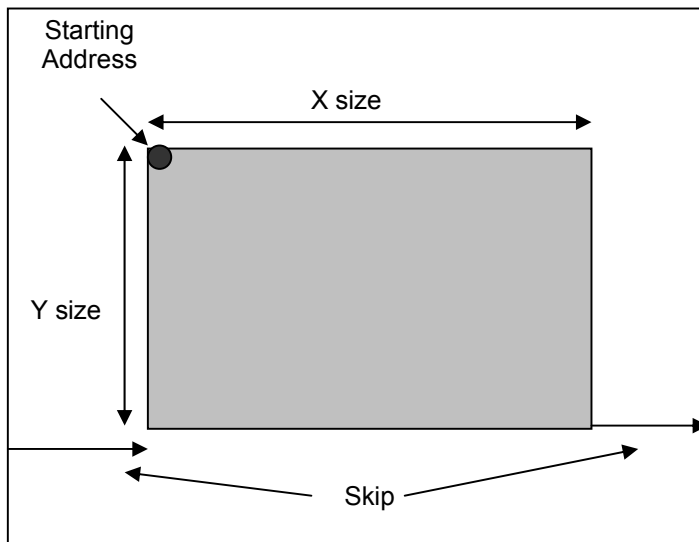


Figure 28. LCD Controller DMA Memory 2-D Layout

• **Screen Memory Base Register (LCD_SCNBASE) – 0x100**

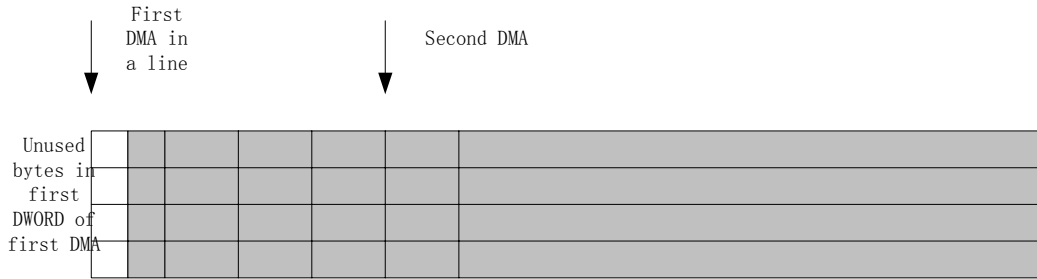


Figure 29. Pixel Offset

Bit	Name	Default	Description
3:0 (R/W)	PIXEL_OFFSET	4'h0	Pixel offset for screen channel. Since the pixel data can be as little as 2-bits/pixel, this pixel offset specifies which bits within the first DWORD specified by bits <30:4> is the first valid pixel. The pixel data can be 2-bit, 4-bit, 8-bit, 12-bit, 16-bit/pixel. Example (16-bit/pixel, Big endian in word): 0000 – bits <31:16> 0001 – bits <15:0> Example (16-bit/pixel, Little endian in word): 0000 – bits <15:0> 0001 – bits <31:16>
30:4 (R/W)	DMA_BASE_ADDR	27'h0	DMA address of the starting memory location for the screen data, this is a byte address, but the lower 4 address bits must be zeroes, which means the DMA start address must lie on a 4-dwords burst boundary.
31 (R/W)	DMA_MODE	1'b0	Continuous mode DMA 1 = when this DMA completes, will automatically generate a DMA with exactly the same setting. 0 = each DMA must be explicitly started by software

• **X Size for screen DMA (LCD_SCN_XSIZE) – 0x104**

Bit	Name	Default	Description
12:0 (R/W)	SCN_XSIZE	13'h0	This value specifies the number of consecutive 4-DWORD bursts per line for the screen DMA. This value should be inclusive of any partial DWORD's used per line. $SCN_XSIZE = (SCN_HEND - SCN_HSTART + 1) * (SCN_BPP / 8) / 16 - 1$
31:13	-	19'h0	Reserved

• **Y Size for screen DMA (LCD_SCN_YSIZE) – 0x108**

Bit	Name	Default	Description
12:0 (R/W)	SCN_YSIZE	13'h0	This value specifies the number of "lines" for the screen DMA. Each line designates a segment of consecutive

			DWORD's with a skip in between. SCN_YSIZE = SCN_VEND-SCN_VSTART-1
31:13	-	19'h0	Reserved

- **Skip value for screen DMA (LCD_SCN_SKIP) – 0x10C**

Bit	Name	Default	Description
12:0 (R/W)	SCN_SKIP	13'h0	This value specifies the number of BYTE's for the DMA address generator to skip in between lines of the screen DMA. SCN_SKIP = ByteWidthOfScnFrameBuf – (SCN_XSIZE*16) And the ByteWidthofScnFrameBuf must be a multiple of 16.
31:13	-	19'h0	Reserved

- **Current DMA Address for the Screen Display (LCD_SCN_DMA_ADDR) – 0x114**

Writing any value to this register will activate the screen DMA. Make sure this register is written last, after all the other DMA registers are setup correctly already.

Bit	Name	Default	Description
0 (W)	SCN_DMA_START	1'h0	Write 1 to start the SCN layer DMA, this bit is self-cleared.
31:1	-	31'h0	Reserved

- **OSD Memory Base Register (LCD_OSDBASE) – 0x118**

Bit	Name	Default	Description
3:0 (R/W)	PIXEL_OFFSET	4'h0	Pixel offset for OSD channel. This pixel offset specifies which bits within the first DWORD specified by bits <30:4> is the first valid pixel for this channel. When used as OSD, the pixel data can be 2-bit, 4-bit, 8bit, 16-bit/pixel. When used as Dual Scan mode, the pixel data can be 2-bit, 4-bit, 8-bit/pixel. Example (2bits/pixel, Big endian in word and in byte): 0000 – bits<31:30> 0001 – bits<29:28> 1111 – bits<1:0>
30:4 (R/W)	DMA_BASE_ADDR	27'h0	DMA address of the starting memory location for the OSD data
31 (R/W)	DMA_MODE	1'b0	Continuous mode DMA 1 = when this DMA completes, will automatically generate a DMA with exactly the same setting. 0 = each DMA must be explicitly started by software

- **X Size for OSD DMA (LCD_OSD_XSIZE) – 0x11C**

Bit	Name	Default	Description
12:0 (R/W)	OSD_XSIZE	13'h0	This value specifies the number of consecutive 4-DWORD bursts per line for the OSD DMA. This value should be inclusive of any partial DWORD's used per line. OSD_XSIZE = (OSD_HEND-

			$OSD_HSTART+1*(OSDBPP/8)/16-1$
31:13	-	19'h0	Reserved

- **Y Size for OSD DMA (LCD_OSD_YSIZE) – 0x120**

Bit	Name	Default	Description
12:0 (R/W)	OSD_YSIZE	13'h0	This value specifies the number of “lines” for the OSD DMA. Each line designates a segment of consecutive DWORD’s with a skip in between. $OSD_YSIZE = OSD_VEND - OSD_VSTART - 1$
31:13	-	19'h0	Reserved

- **Skip value for OSD DMA (LCD_OSD_SKIP) – 0x124**

Bit	Name	Default	Description
12:0 (R/W)	OSD_SKIP	13'h0	This value specifies the number of BYTE’s for the DMA address generator to skip in between lines of the OSD DMA. $OSD_SKIP = \text{ByteWidthOfOsdFrameBuf} - (OSD_XSIZE * 16)$ And the ByteWidthofOSDFrameBuf must be a multiple of 16.
31:13	-	19'h0	Reserved

- **Current DMA Address for the OSD Display (LCD_OSD_DMA_ADDR) – 0x12C**

Writing to this register will activate the OSD DMA. Make sure this register is written last, after all the other DMA registers are setup correctly already.

Bit	Name	Default	Description
0 (W)	OSD_DMA_START	1'h0	Write 1 to start the OSD DMA, this bit is self-cleared.
31:1	-	31'h0	Reserved

The read value is for test purpose only.

Bit	Name	Default	Description
26:0 (R)	OSD_DMA_ADDR	27'h0	Current DMA address for the OSD display
31:27	-	5'h0	Reserved

- **OSD2 Memory Base Register (LCD_OSD2BASE) – 0x130**

Bit	Name	Default	Description
3:0 (R/W)	PIXEL_OFFSET	4'h0	Pixel offset for OSD2 channel. This pixel offset specifies which bits within the first DWORD specified by bits <30:4> is the first valid pixel for this channel. When used as OSD2, the pixel data can be 2-bit, 4-bit, 8bit, 16-bit/pixel. When used as Dual Scan mode, the pixel data can be 2-bit, 4-bit, 8-bit/pixel. Example (2bits/pixel, Big endian in word and in byte): 0000 – bits<31:30> 0001 – bits<29:28> 1111 – bits<1:0>
30:4	DMA_BASE_ADDR	27'h0	DMA address of the starting memory location for the

(R/W)			OSD2 data
31 (R/W)	DMA_MODE	1'b0	Continuous mode DMA 1 = when this DMA completes, will automatically generate a DMA with exactly the same setting. 0 = each DMA must be explicitly started by software

- **X Size for OSD2 DMA (LCD_OSD2_XSIZE) – 0x134**

Bit	Name	Default	Description
12:0 (R/W)	OSD2_XSIZE	13'h0	This value specifies the number of consecutive 4-DWORD bursts per line for the OSD2 DMA. This value should be inclusive of any partial DWORD's used per line. $OSD2_XSIZE = (OSD2_HEND - OSD2_HSTART + 1) * (OSD2BPP / 8) / 16 - 1$
31:13	-	19'h0	Reserved

- **Y Size for OSD2 DMA (LCD_OSD2_YSIZE) – 0x138**

Bit	Name	Default	Description
12:0 (R/W)	OSD2_YSIZE	13'h0	This value specifies the number of "lines" for the OSD2 DMA. Each line designates a segment of consecutive DWORD's with a skip in between. $OSD2_YSIZE = OSD2_VEND - OSD2_VSTART - 1$
31:13	-	19'h0	Reserved

- **Skip value for OSD2 DMA (LCD_OSD2_SKIP) – 0x13C**

Bit	Name	Default	Description
12:0 (R/W)	OSD2_SKIP	13'h0	This value specifies the number of BYTE's for the DMA address generator to skip in between lines of the OSD2 DMA. $OSD2_SKIP = \text{ByteWidthOfOsd2FrameBuf} - (OSD2_XSIZE * 16)$ And the ByteWidthofOSD2FrameBuf must be a multiple of 16.
31:13	-	19'h0	Reserved

- **Current DMA Address for the OSD2 Display (LCD_OSD2_DMA_ADDR) – 0x144**

Writing to this register will activate the OSD2 DMA. Make sure this register is written last, after all the other DMA registers are setup correctly already.

Bit	Name	Default	Description
0 (R)	OSD2_DMA_START	1'h0	Write 1 to start OSD2 layer DMA, this bit is self-cleared
31:1	-	31'h0	Reserved

6.2.4.10 Interrupt Registers

The LCD can generate interrupts from several different sources. It can generate an interrupt when the screen, the OSD FIFO or the OSD2 FIFO underflows or overflows; it can generate an interrupt when the vertical counter reaches a pre-determined line number (i.e. can be used as a frame interrupt); finally, it

can generate an interrupt when the screen, the OSD FIFO or the OSD2 FIFO DMA is completed. Each interrupt source can be independently masked and cleared.

- **Current Interrupt Status and Interrupt Clear (LCD_INT_CTRL_STATUS) – 0x180**

Read this register to determine source of interrupt. If a particular bit reads 1, then that interrupt is asserted. Writing a 1 to a particular bit clears the interrupt for that source. This is the only way to clear the interrupt. The hardware will not clear any interrupts.

Note: The interrupt bit for each source can be read and/or cleared even if the mask bit is not set. Without the mask bit, however, an interrupt will not be generated to the RISC.

Bit	Name	Default	Description
0 (R/W)	OSD2_DMA_INT	1'b0	OSD2 DMA Interrupt
1 (R/W)	OSD_DMA_INT	1'b0	OSD DMA Interrupt
2 (R/W)	SCN_DMA_INT	1'b0	Screen DMA Interrupt
3 (R/W)	SCN_LINE_INT	1'b0	Screen Line Interrupt
4 (R/W)	OSD2_UFLOW_INT	1'b0	OSD2 FIFO Underflow Interrupt
5 (R/W)	OSD2_OFLOW_INT	1'b0	OSD2 FIFO Overflow Interrupt
6 (R/W)	OSD_UFLOW_INT	1'b0	OSD FIFO Underflow Interrupt
7 (R/W)	OSD_OFLOW_INT	1'b0	OSD FIFO Overflow Interrupt
8 (R/W)	SCN_UFLOW_INT	1'b0	Screen FIFO Underflow Interrupt
9 (R/W)	SCN_OFLOW_INT	1'b0	Screen FIFO Overflow Interrupt.
31:10	-	22'h0	Reserved

- **Interrupt Mask (LCD_INT_MASK) – 0x184**

This register provides the interrupt mask for all the LCD interrupt sources. If a mask bit is set to 1, that interrupt is enabled. Then if the interrupt status bit goes to 1, an interrupt is generated to the RISC. If the mask bit is set to 0, then no interrupt is generated, although the status bit can still be set to 1 and be cleared by writing to the above register.

Bit	Name	Default	Description
0 (R/W)	OSD2_DMA_MSK	1'b0	OSD2 DMA Interrupt Mask
1 (R/W)	OSD_DMA_MSK	1'b0	OSD DMA Interrupt Mask
2 (R/W)	SCN_DMA_MSK	1'b0	Screen DMA Interrupt Mask
3 (R/W)	SCN_LINE_MSK	1'b0	Screen Line Interrupt Mask
4 (R/W)	OSD2_UFLOW_MSK	1'b0	OSD2 FIFO Underflow Interrupt Mask
5 (R/W)	OSD2_OFLOW_MSK	1'b0	OSD2 FIFO Overflow Interrupt Mask
6 (R/W)	OSD_UFLOW_MSK	1'b0	OSD FIFO Underflow Interrupt Mask
7 (R/W)	OSD_OFLOW_MSK	1'b0	OSD FIFO Overflow Interrupt Mask
8 (R/W)	SCN_UFLOW_MSK	1'b0	Screen FIFO Underflow Interrupt Mask
9 (R/W)	SCN_OFLOW_MSK	1'b0	Screen FIFO Overflow Interrupt Mask
31:10	-	22'h0	Reserved

- **Interrupt Line Number (LCD_SCN_INT_LINE) – 0x188**

Bit	Name	Default	Description
10:0 (R/W)	LINE_NUM	11'h0	This value represents the line number at which a screen interrupt will be generated. This allows the user to specify a line during which an interrupt will be generated for each frame. The interrupt is generated at the beginning of that

			particular line.
11	-	1'b0	Reserved
12 (R/W)	INT_LINE_VALID	1'b0	This bit determines whether the line number below is valid for generating a screen interrupt. This differs from the mask in that if this bit is not set, then the interrupt status bit for the screen interrupt will not be set. When this bit is set, then the interrupt status bit is set each time the line count is equal to the value below.
31:13	-	19'h0	Reserved

6.2.4.11 FIFO Data Read/Write Registers

These registers are only for test purpose.

Both the screen and the OSD FIFO can be pushed and popped by writing and reading into the PUSH_POP registers respectively. The read or write pointer will be incremented automatically. This can be used to insert or remove values from the FIFO.

The other way the FIFO memory can be accessed is directly reading or writing the memory location. In this case, the address specifies which location in the FIFO is being accessed.

- **Push/Pop Screen FIFO (LCD_SCN_FIFO_PUSH_POP) – 0x400**

Bit	Name	Default	Description
31:0	PUSH_POP	-	If this register is read, the first value in the screen FIFO will be popped out, i.e. the value read and the read index increased. If this register is written, then the value is pushed into the screen FIFO, i.e. the value is written to the end of the FIFO, and the write index is increased.

- **Read/Write ScreenFIFO (LCD_SCN_FIFO) – 0x600~0x7FC**

Bit	Name	Default	Description
31:0 (R/W)	FIFO_DATA	-	Values in the screen FIFO. Can directly read or write values anywhere in the FIFO by specifying the corresponding address.

- **Push/Pop OSD FIFO (LCD_OSD_FIFO_PUSH_POP) – 0x800**

Bit	Name	Default	Description
31:0 (R/W)	PUSH_POP	-	If this register is read, the first value in the OSD FIFO will be popped out, i.e. the value read and the read index increased. If this register is written, then the value is pushed into the OSD FIFO, i.e. the value is written to the end of the FIFO, and the write index is increased.

- **Read/Write OSD FIFO (LCD_OSD_FIFO) – 0xA00~0xBFC**

Bit	Name	Default	Description
31:0 (R/W)	FIFO_DATA	-	Values in the OSD FIFO. Can directly read or write values anywhere in the FIFO by specifying the corresponding

			address.
--	--	--	----------

- **Push/Pop OSD2 FIFO (LCD_OSD2_FIFO_PUSH_POP) – 0xC00**

Bit	Name	Default	Description
31:0 (R/W)	PUSH_POP	-	If this register is read, the first value in the OSD2 FIFO will be popped out, i.e. the value read and the read index increased. If this register is written, then the value is pushed into the OSD2 FIFO, i.e. the value is written to the end of the FIFO, and the write index is increased.

- **Read/Write OSD2 FIFO (LCD_OSD2_FIFO) – 0xE00~0xFFC**

Bit	Name	Default	Description
31:0 (R/W)	FIFO_DATA	-	Values in the OSD2 FIFO. Can directly read or write values anywhere in the FIFO by specifying the corresponding address.

6.2.4.12 Palette & Sequence RAM

The 256 x 18 palette RAM is written or read directly by RISC Interface to access the specified address. When as color palette, each address contains RGB value for a color that has an index corresponding to the address. The color component is 6-bits each, specifying the 6 MSB of an 8-bit value. For passive monochrome mode, the palette is used for dithering sequence of frame rate control. As the dithering sequence for one gray color is 32 bit, the bottom 32 entries are used for 16 grey levels. For 4bit/pixel mode, bits <15:0> of entry 0 and entry 16 combine as 32 bits for one gray color, and bits <15:0> of entry 1 and entry 17 as another gray color, and so on. And for 2bit/pixel mode, only entry 0~3 and entry 16~19 are used.

- **Read/Write 8-bit Palette (LCD_PALETTE) – 0x1000~0x13FC**

Bit	Name	Default	Description
17:0 (R/W)	PALETTE_DATA	-	Values in the color palette. Can directly read or write values any palette value by specifying the index using bits <9:2> in the address. Each entry in the palette is 18-bits, 6 each for each color: Bits 17:12 – Red Bits 11:6 – Green Bits 5:0 – Blue Values in the dithering sequence of frame rate control, bits <15:0> are used.
31:18	-	-	Reserved

The 16 x 18 palette RAM is written or read directly by RISC Interface to access the specified address. This palette is always a color palette, and each address contains RGB value for a color that has an index corresponding to the address. The color component is 6-bits each, specifying the 6 MSB of a 4-bit value. This palette is used for 4-bit color value.

- **Read/Write 4-bit Color Palette (LCD_PALETTE16) – 0x1400~0x143C**

Bit	Name	Default	Description
17:0	PALETTE16_DATA	-	Values in the color palette. Can directly read or write

(R/W)			values any palette value by specifying the index using bits <5:2> in the address. Each entry in the palette is 18-bits, 6 each for each color: Bits 17:12 – Red Bits 11:6 – Green Bits 5:0 – Blue
31:18	-	-	Reserved

The 64 x 32 RAM is written or read directly by RISC Interface to access the specified address. It is used for dithering sequence of frame rate control in passive color mode. It generates a 32 x 64 entry to support 32 grey-scale levels for every R, G, B color. As the dithering sequence for one gray color is 64 bit, the bottom half RAM is used for the first 32 bits of the dithering sequence, and the up half RAM is used for the other 32 bits of the dithering sequence.

- **Read/Write Color Dithering Sequence (LCD_COLORSEQ) –0x1800–0x18FC**

Bit	Name	Default	Description
31:0 (R/W)	COLOR_SEQ	-	Values in the dithering sequence of frame rate control. One dithering sequence entry is made of two 32-bit DWORD's.

6.3 2D/BitBLT Engine

6.3.5 Overview

The Atlas-II™ BitBLT engine is designed to accelerate Microsoft's DirectDraw applications. The engine contains a 3-operand ALU with 256 raster operations, source and destination FIFOs. The Engine runs at the memory clock speed.

The BitBLT Engine supports these features:

1. Support byte level BitBLT when 8bpp (not index mode) and word level when 16bpp and DWORD level when 32bpp pixel format
2. Support rectangle clipping, that is when clipping enabled, only pixels inside this box will be affected by the operations
3. Support Destination Color Transparency: a pixel value is designated as transparent; any pixel output from the BLT engine with this value is not written to the destination memory when enabled
4. Support Source Color Transparency: a pixel value is designated as transparent; any pixel from the source with this value will not affect the destination memory at the corresponding XY coordination.
5. Support all the 256 raster operations
6. Support both Y-direction of the 2D window while the given coordination is always at the window's top-left corner
7. Support Command Queue in SDRAM and single command mode

The BitBLT engine also has certain limitations as following:

1. The BitBLT Engine supports 8bpp, 16bpp, 32bpp pixel format and when 16bpp, one pixel must be WORD (2bytes) aligned, and when 32bpp, one pixel must be DWORD (4bytes) aligned in SDRAM. And when 8bpp, it is not index mode, as there is no color look-up-table palette in this design.
2. The source and destination must share the same pixel format
3. The BitBLT doesn't support right-to-left BLT, which is the BitBLT engine always does BLT from the left border of the window to the right border. (This will be a problem only when the destination window is on the absolutely right side of the source window (same y-coordinate) with overlap between them, and in such situation, SW must configure this BLT as two, first

- SRCCOPY the source to a temp destination buffer in SDRAM and use the temp destination buffer as the second BLT command's source instead.)
4. Pattern for ROP3 is mono and designed as 8*8
 5. The Command Queue must align on a 4DW address
 6. Support only little endian mode

The BitBLT is reset by the system asynchronous reset, which is low active. And when the SW writes a BLT command, the BLT will be started based on the ROP type and stop automatically when BLT over when in manual mode and if in auto mode, SW can prepare a BLT command queue and tells hardware the queue start address, the engine will automatically fetch the command one by one and stop when all command finishes.

6.3.6 Functional Description

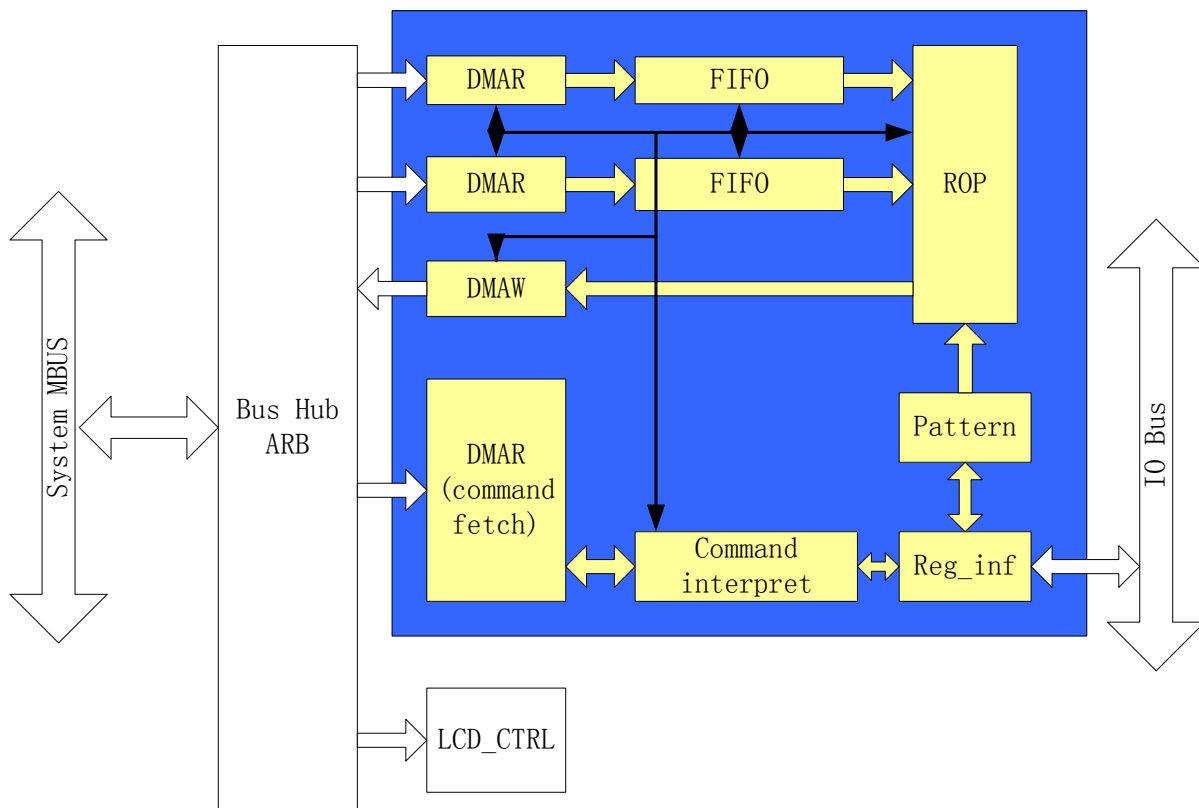


Figure 30. Block Diagram of BitBLT Engine

The parts in blue are the main BitBLT diagram. CPU writes BLT command into BLT's Command queue, which will be interpreted to configure the BLT DMA and ROP in different mode. There are four DMA modules in the BLT: two DMA Read modules (DMAR) load data from SDRAM as the operands of the ROP operation, after which one DMA Write module (DMAW) stores data back into SDRAM; and another DMAR is used to fetch BLT command from SDRAM when in auto mode. These DMA's will be arbitrated along with the LCD controller's DMA's by the BUS HUB to access the System Bus.

This **DMAR** module takes charge of fetching one source for the ROP (not the Pattern, which is designed lying in internal SRAM). For example, as SRCAND BLT, one DMAR will be configured to fetch the source image data and the other DMAR will be configured to fetch the destination image data.

DMAR is designed as BURST4 that is every REQ-GNT there will be 4Dword fetched, so DMAR should also take charge of BYTE aligned barrel shift. DMAR will barrel the data fetched in before writing them into the FIFO in a way as following:

Source BYTE position	Des BYTE position	Barrel Shift Direction	To FIFO
0	0	No	Directly
	1	Left 1 byte	Directly
	2	Left 2 bytes	Directly
	3	Left 3 bytes	Directly
1	0	Right 1 byte	Delayed
	1	No	Directly
	2	Left 1 byte	Directly
	3	Left 2 bytes	Directly
2	0	Right 2 bytes	Delayed
	1	Right 1 byte	Delayed
	2	No	Directly
	3	Left 1 byte	Directly
3	0	Right 3 bytes	Delayed
	1	Right 2 bytes	Delayed
	2	Right 1 byte	Delayed
	3	No	Directly

NOTE:

1. Data in SDRAM is little Endian
2. “Directly” means the current Read Data from the System Bus need no additional data in the next Read Data (in fact, some high bytes will be used by the next Read Data). So the current will be directly stored into the FIFO
3. “Delayed” means the current Read Data from the System Bus will be merged with the following data, that is the current data should be latched for the next cycle (maybe the next burst) use

This **DMAW** module takes charge of storing back the data after ROP operation into SDRAM.

The **ROP** module takes charge of Boolean operation including ROP2 (Binary Raster Operations) and ROP3 (Ternary Raster Operations).

For detail of ROP2 and ROP3 definitions, please see also *BLT_ROP* register.

There are two 9-Dword deep **FIFOs** for the storage of ROP operands.

6.3.7 2D/BitBLT Engine Registers

Table 30. BitBLT Engine Register Mapping

RISC Address <11:0>	Register	Description
0x1C00	BLT_CONTROL	The working mode, and successive BLT command number
0x1C04	BLT_INT_MASK	The BLT interrupt source mask
0x1C08	BLT_INT_STATUS	The BLT interrupt source and status
0x1C0C	BLT_SRC_BASE	The source base memory address, byte alignment

0x1C10	BLT_DES_BASE	The destination base memory address, byte alignment
0x1C14	BLT_WINDOW_DIM	The 2D window width and height specified in pixels, source and destination window share the same dimension
0x1C18	BLT_STRIDE	Line stride of destination and source specified in pixels
0x1C1C	BLT_SRC_XY	The xy-coordinate of the source window and window y-direction
0x1C20	BLT_DES_XY	The xy-coordinate of the destination window and window y-direction
0x1C24	BLT_CLIP_TLXY	The top-left xy-coordinate of the clipping rectangle
0x1C28	BLT_CLIP_BRXY	The bottom-right xy-coordinate of the clipping rectangle
0x1C2C	BLT_FCOLOR	The foreground color
0x1C30	BLT_BCOLOR	The background color
0x1C34	BLT_SRC_COLORKEY	Reserve a color as transparent of the source
0x1C38	BLT_DES_COLORKEY	Reserve a color as transparent for the destination
0x1C3C	BLT_ROP	The ROP code, pixel format, Clip mode, color key enable. Writing into this register will start the BitBLT operation immediately when configured as not auto mode
0x1C40	BLT_PM_LOW	The low 4bytes for the mono pattern
0x1C44	BLT_PM_HIGH	The high 4bytes for the mono pattern
0x1C48	BLT_CMDQUE_ADDR	When BLT was configured as auto mode, this register contains the first BLT command address in SDRAM, and every command is 16 DWORD lengths. Writing to this register will start the successive BLT if in auto mode.

- **BitBLT hardware working mode setting (BLT_CONTROL) – 0x1C00**

Bit	Name	Default	Description
0 (R/W)	AUTO	1'h0	BLT working mode as command queue mode when 1 or single command mode when 0
15:1	-	14'h0	Reserved
31:16 (R/W)	QUE_NUM	16'h0	When configured as command queue mode, this contains the queued BLT commands number. (When read, this register reflects the remaining BLT number)

Single command mode:

BLT hardware works as one command by one. Software will program BLT hardware with one ROP setting and start it, before this one is finished, software will not allocate another.

Command queue mode:

Software prepares a queue of BLT commands with a data structure defined by BLT hardware in memory, and by telling the command number and queue memory address, BLT hardware will fetch these commands and execute one by one. Also when one queue is not finished, software should not allocate another queue. The command structures please refer to *BLT_CMDQUE_ADDR* notes.

- **BitBLT interrupt mask (BLT_INT_MASK) – 0x1C04**

Bit	Name	Default	Description
0 (R/W)	SINGLE_BLT_OVER_INT_MASK	1'h0	Mask for interrupt when one single BitBLT command completed
1 (R/W)	QUEUED_BLT_OVER_INT_MASK	1'b0	Mask for interrupt when the whole queued BitBLT commands completed
31:2	-	30'h0	Reserved

- **BitBLT interrupt and status (BLT_INT_STATUS) – 0x1C08**

Bit	Name	Default	Description
0 (W/R)	SINGLE_BLT_OVER	1'h0	Valid when the one single BitBLT finished, SW clear this bit by writing zero
1 (W/R)	QUEUED_BLT_OVER	1'h0	Valid when the BLT queue completed, SW clear this bit by writing zero
15:2	-	14'h0	Reserved
16 (R)	BLT_BUSY	1'h0	On when BitBLT is assigned and cleared when BitBLT finished.
31:17	-	15'h0	Reserved

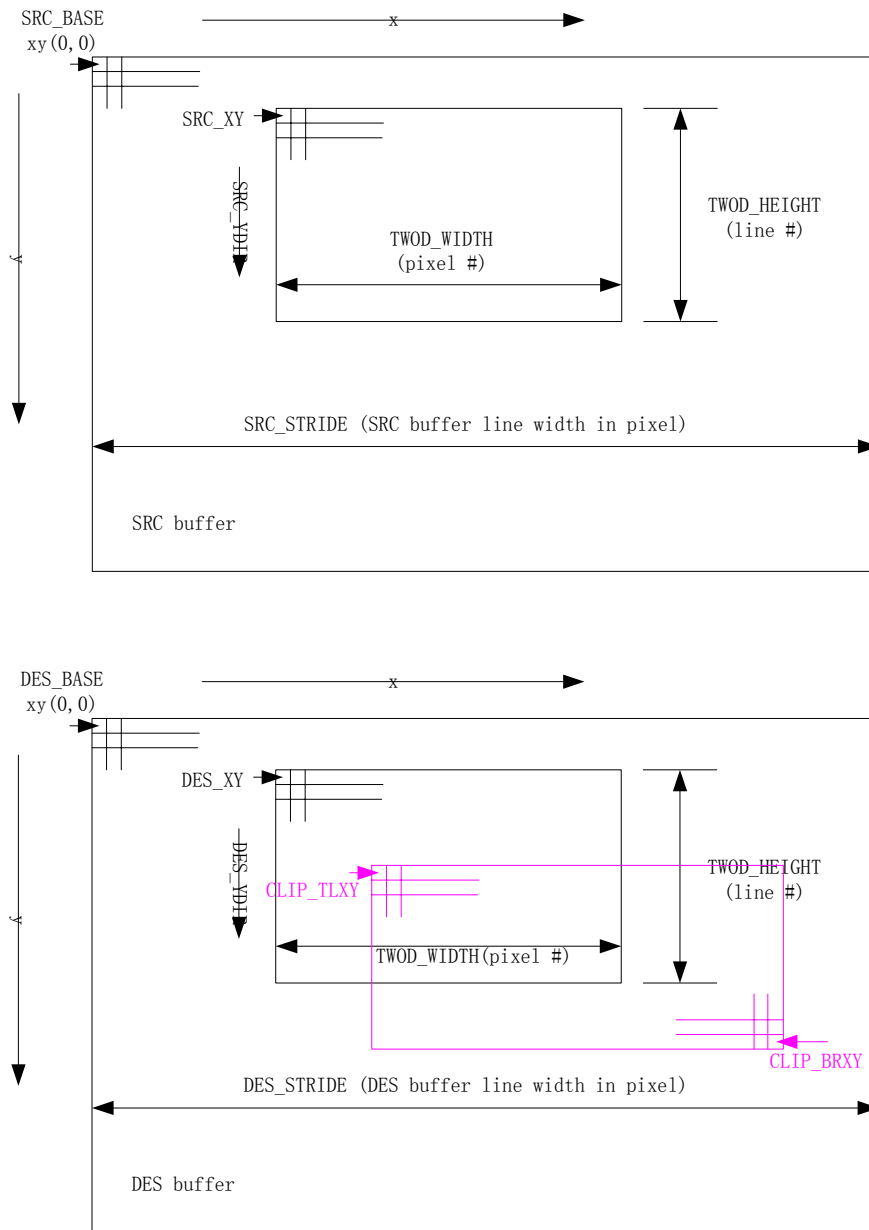


Figure 31. BitBLT 2D window dimension

- Source Base Address in SDRAM (BLT_SRC_BASE) – 0x1C0C

Bit	Name	Default	Description
26:0 (R/W)	SRC_BASE	27'h0	The base memory address of source with byte address alignment ---(the pixel which is at coordinate xy(0,0) pixel_format = 2'b00 : [26:0] valid pixel_format = 2'b01 : [26:1] valid and bit0 should be 0 pixel_format = 2'b1X : [26:2] valid and bit1:0 should be 00
31:27	-	5'h0	Reserved

- **Destination Base Address in SDRAM (BLT_DES_BASE) – 0x1C10**

Bit	Name	Default	Description
26:0 (R/W)	DES_BASE	27'h0	The base memory address of destination with byte alignment --- (the pixel which is at coordinate xy(0,0). pixel_format = 2'b00 : [26:0] valid pixel_format = 2'b01 : [26:1] valid and bit0 should be 0 pixel_format = 2'b1X : [26:2] valid and bit1:0 should be 00
31:27	-	5'h0	Reserved

- **2D window dimension (BLT_WINDOW_DIM) – 0x1C14**

Bit	Name	Default	Description
10:0 (R/W)	HEIGHT	11'h0	The 2D window's height in pixels Max: 2047 lines height
15:12	-	4'h0	Reserved
26:16 (R/W)	WIDTH	11'h0	The 2D window's width in pixels Max: 2047pixels wide
31:28	-	4'h0	Reserved

- **Pitch of Source and Destination in pixels (TWOD_STRIDE) – 0x1C18**

Bit	Name	Default	Description
11:0 (R/W)	DES_STRIDE	12'h0	Line stride of destination specified in pixels
15:12	-	4'h0	Reserved
27:16 (R/W)	SRC_STRIDE	12'h0	Line stride of source specified in pixels
31:28	-	4'h0	Reserved

- **The Source 2D Window XY-Coordination (BLT_SRC_XY) – 0x1C1C**

Bit	Name	Default	Description
10:0 (R/W)	SRC_Y	11'h0	Source window X-coordination (0~2047)
15:11	-	5'h0	Reserved
26:16 (R/W)	SRC_X	11'h0	Source window Y-coordination (0~2047)
30:27	-	4'h0	Reserved
31 (R/W)	SRC_YDIR	1'b0	Source window Y direction. 0: top-down, 1: bottom-up

- **The Destination 2D Window XY-Coordination (BLT_DES_XY) – 0x1C20**

Bit	Name	Default	Description
10:0 (R/W)	DES_Y	11'h0	Destination window X-coordination (0~2047)
15:11	-	5'h0	Reserved
26:16 (R/W)	DES_X	11'h0	Destination window Y-coordination (0~2047)
30:27	-	4'h0	Reserved
31	DES_YDIR	1'b0	Destination window Y direction. 0: top-down, 1: bottom-up

(R/W)			
-------	--	--	--

- **The Clip 2D Rectangle Top-Left XY-Coordination (BLT_CLIP_TLXY) – 0x1C24**

The Clip Rectangle is used to determine whether the destination area is writable or not. And here when clip is enabled, only area in the clip rectangle will be writable (includes the boundary).

Bit	Name	Default	Description
10:0 (R/W)	CLIP_TLY	11'h0	Clip rectangle top-left X-coordination relative to the DES buffer coordinate xy(0,0) (0~2047)
15:11	-	5'h0	Reserved
26:16 (R/W)	CLIP_TLX	11'h0	Clip rectangle top-left Y-coordination (0~2047)
31:27	-	5'h0	Reserved

- **The Clip 2D Rectangle Bottom-Right XY-Coordination (BLT_CLIP_BRXY) – 0x1C28**

Bit	Name	Default	Description
10:0 (R/W)	CLIP_BRY	11'h0	Clip rectangle bottom-right X-coordination relative to the DES buffer coordinate xy(0,0) (0~2047)
15:11	-	5'h0	Reserved
26:16 (R/W)	CLIP_BRX	11'h0	Clip rectangle bottom-right Y-coordination (0~2047)
31:27	-	5'h0	Reserved

- **The Foreground Color (BLT_FCOLOR) – 0x1C2C**

Bit	Name	Default	Description
31:0 (R/W)	FCOLOR	32'h0	Bit7:0 valid when pixel format is 00 Bit15:0 valid when pixel format is 01 Bit31:0 valid when pixel format is 1X

- **The Background Color (BLT_BCOLOR) – 0x1C30**

Bit	Name	Default	Description
31:0 (R/W)	BCOLOR	32'h0	Bit7:0 valid when pixel format is 00 Bit15:0 valid when pixel format is 01 Bit31:0 valid when pixel format is 1X

The foreground color is used when doing ROP=0x00 (BLACKNESS) to fill the destination with this color. The background color is used when doing ROP=0xFF(WHITENESS) to fill the destination with this color. And these two registers defined the color when doing pattern concerned ROP; please refer to the PATTERN_MATRIX_LOW and PATTERN_MATRIX_HIGH notes.

- **The Source Color Key (BLT_SRC_COLORKEY) – 0x1C34**

Bit	Name	Default	Description
31:0 (R/W)	SRC_COLORKEY	32'h0	Color key is a reserved color which will not interfere the ROP result (result will not written back into the destination) and here if source is needed in a ROP, pixels from the source bitmap with this value will not change the destination

			pixel value if source color key function is enabled. Bit7:0 valid when pixel format is 00 Bit15:0 valid when pixel format is 01 Bit31:0 valid when pixel format is 1X
--	--	--	--

- **The Destination Color Key (BLT_DES_COLORKEY) – 0x1C38**

Bit	Name	Default	Description
31:0 (R/W)	DES_COLORKEY	32'h0	Similar to the SRC_COLORKEY, except that if the destination is needed in a ROP and pixels from the destination bitmap is of this value, thus if function enabled, the destination pixel will not be changed. Bit7:0 valid when pixel format is 00 Bit15:0 valid when pixel format is 01 Bit31:0 valid when pixel format is 1X

- **BitBLT ROP Code (BLT_ROP) – 0x1C3C**

Bit	Name	Default	Description
7:0 (R/W)	ROP	8'h0	Please refer to figure below for details. Here ROP2 and ROP3 are treated coequally: as those Bits<7:4>=Bits<3:0>, they are ROP2; otherwise, they are ROP3. Writing into this register will start the BLT immediately if not in command queue mode
9:8 (R/W)	PIXEL_FORMAT	2'h0	Pixel format for both the source and destination as 00: 8bits/pixel (not index mode) 01: 16bits/pixel 1X: 32bits/pixel
10 (R/W)	CLIP_EN	1'h0	Rectangle clip enable when 1 or disable when 0
11 (R/W)	SRC_COLORKEY_EN	1'h0	Source color key enable when 1 or disable when 0. When enabled, source pixel with this color value as SRC_COLORKEY will not change the destination pixel
12 (R/W)	DES_COLORKEY_EN	1'h0	Destination color key enable when 1 or disable when 0. When enabled, result from the ROP logic with value of DES_COLORKEY will not replace the destination pixel.
31:13	-	19'h0	Reserved

		Bits [7:4]							
		0	1	2	3	4	5	6	7
bit [3:0]	0	0	PDSona	DPSnaa	PSna	PSDnaa	PDna	PDSxa	PDSana
	1	DPSoon	DSon	SDPxon	SDPnaon	DPSxon	DSPnaon	DSPDSaoxxn	SSDxPDxaxn
	2	DPSona	SDPxnon	DSna	SDPSoox	SDxPDxa	DPSDaox	DSPDoax	SDPSxox
	3	PSon	SDPaon	SPDnaon	Sn	SPDSanaxn	SPDSxaxn	SDPnox	SDPnoan
	4	SDPona	DPSxnon	SPxDSxa	SPDSaox	SDna	DPSonon	SDPSoax	DSPDxox
	5	DPon	DPSaon	PDSPanaxn	SPDSxnox	DPSnaon	Dn	DSPnox	DSPnoan
	6	PDSxnon	PSPDSanaxx	SDPSaox	SDPox	DSPDaox	DPSox	DSx	SDPSnaox
	7	PDSaon	SSPxDSxaxn	SDPSxnox	SDPcan	PSPDPxaxn	DPSoan	SDPSonox	DSan
	8	SDPnaa	SPxPDxa	DPSxa	PSPDpax	SDPxa	PDSPaax	DSPDSonoxn	PDSax
	9	PDSxon	SDPSanaxn	PSPDSaoxxn	SPDnox	PSPDPaoxxn	DPSnox	PDSxoxn	DSPDSaoxxn
	A	DPna	PDSPaax	DPSana	SPDSxox	DPSDoax	DPx	DPSax	DPSDnoax
	B	PSDnaon	SDPSxaxn	SSPxPDxaxn	SPDnoan	PDSnox	DPSDonox	PSPDSaoxxn	SDPxnan
	C	SPna	PSPDaax	SPDSaox	PSx	SDPana	DPSDxox	SDPaax	SPDSnoax
	D	PDSnaon	DSPDxaxn	PSPDnox	SPDSonox	SSPxDSxoxn	DPSnoan	PSPDPaoxxn	DPSxnan
	E	PDSonon	PDSox	PSPDpox	SPDSnaox	PSPDpox	DPSDnaox	SDPSnoax	SPxDSxo
	F	Pn	PDSaon	PSPDnoan	PSan	PDSnoan	DPan	PDSxnan	DPSaan

		Bits [7:4]							
		8	9	A	B	C	D	E	F
bit [3:0]	0	DPSaa	PDSxna	DPa	PDSnoa	PSa	PSDnoa	PDSoa	P
	1	SPxDSxon	SDPSnoaxn	PDSPnaoxn	PSPPxoxn	SPDSnaoxn	PSPDpxoxn	PDSoxn	PDSono
	2	DPSxna	DSPDPaoxx	DPSnoa	SSPxDSxox	SPDSonoxn	PDSnax	DSPDxax	PDSnao
	3	SPDSnoaxn	SPDaxn	DPSDxoxn	SDPanaxn	PSxn	SPDSaoxaxn	PSPDpaaxn	PSno
	4	SDPxna	PSPDSaoxx	PSPDnooxn	PSPDnax	SPDnoa	SSPxPDxax	SDPSxax	PSPDnao
	5	PSPDnoaxn	DPSaxn	PDxn	DSPDdoaxn	SPDSxoxn	DPSanaxn	PSPDpaaxn	PDno
	6	DSPDSaoxx	DPSxx	DSPnax	DSPDpaaxx	SDPnax	PSPDSaoxx	SDPSanax	PSPDxox
	7	PDSaxn	PSPDSonoxx	PSPDpaaxn	SDPxan	PSPDpaaxn	DPSxan	SPxPDxan	PDSano
	8	DSa	SDPSonoxn	DPSoa	PSPDpxax	SDPoa	PSPDpxax	SSPxDSxax	PDSao
	9	SDPSnaoxn	DSxn	DPSoxn	DSPDdoaxn	SPDoxn	SDPSaoxaxn	DSPDSanaxn	PDSxno
	A	DSPnao	DPSnax	D	DPSnao	DPSDpxax	DPSDanax	DPSo	DPo
	B	DSPDxoxn	SDPSaoxaxn	DPSono	DSno	SPDSaoxaxn	SPxDSxan	DPSxno	DPSnoo
	C	SDPnao	SPDnax	SPDSxax	SPDSanax	S	SPDnao	SDPao	PSo
	D	SDPSxoxn	DSPDdoaxn	DPSDdoaxn	SDxPDxan	SDPono	SDno	SDPxno	PSPDnoo
	E	SSDxPDxax	DSPDSaoxx	DSPnao	DPSxo	SDPnao	SDPxo	DSo	DPSoo
	F	PDSanaxn	PDSxan	DPno	DPSano	SPno	SDPano	SDPnoo	1

Figure 32. ROP2/3 Code Definitions

Each raster-operation code represents a Boolean operation in which the values of the pixels in the source, the selected brush (pattern), and the destination are combined. The operands and operands are:

- D=Destination bitmap
- P=Selected brush (pattern)
- S=Source bitmap
- a=bitwise AND
- n=bitwise NOT (inverse)
- o=bitwise OR
- x=bitwise XOR (exclusive OR)

All Boolean operations are presented in reverse Polish notation.

For example, **SDPnoa** represents:

$$(D|\sim P) \& S$$

DPSDaoxn represents:

$$\sim(((S\&D)|P)^D)$$

This engine supports all 256-rop operations and while doing ROP3 operations, only mono pattern as the following two registers defined is supported.

- **BitBLT pattern matrix low bytes (BLT_PM_LOW) – 0x1C40**

Bit	Name	Default	Description
31:0 (R/W)	PM_LOW	32'h0	Bit31~bit0 of the 8*8 mono pattern

- **BitBLT pattern matrix high bytes (BLT_PM_HIGH) – 0x1C44**

Bit	Name	Default	Description
31:0 (R/W)	PM_HIGH	32'h0	Bit63~bit32 of the 8*8 mono pattern

The mono pattern is organized as following: (bit7 represents the top-left first pixel of the pattern)

7	6	5	4	3	2	1	0
15							8
23							16
31							24
39							32
47							40
55							48
63							56

Where bit is 1 the color represents the foreground color and if 0 means the background color with suitable pixel format.

- **BitBLT command queue start address (BLT_CMDQUE_ADDR) – 0x1C48**

Bit	Name	Default	Description
3:0	-	4'h0	Reserved
26:4 (R/W)	CMDQUE_ADDR	23'h0	The command queue start address in memory when BLT works in command queue mode, every command must be packed up as following: Every command is composed of 16 double words, and the start address must be 4Dword address based. DW0: Bit [26:0] Source base addr DW1: Bit [26:0] Destination base addr DW2: Bit [10:0] 2D window height in pixel Bit [26:16] 2D window width in pixel DW3: Bit [12:0] Destination stride Bit [28:16] Source stride DW4: Bit [10:0] Source 2D window y-coordination Bit [26:16] Source 2D window x-coordination Bit [31] Source 2D window Y-direction

			DW5: Bit [10:0] Destination 2D window y-coordination Bit [26:16] Destination 2D window x-coordination Bit [31] Destination 2D window Y-direction DW6: Bit [10:0] Clip rectangle Top-left y-coordination Bit [26:16] Clip rectangle Top-left x-coordination DW7: Bit [10:0] Clip rectangle Bottom-right y-coordinate Bit [26:16] Clip rectangle Bottom-right x-coordinate DW8: Bit [31:0] Foreground color DW9: Bit [31:0] Background color DW10: Bit [31:0] Source color key DW11: Bit [31:0] Destination color key DW12: Bit [7:0] ROP Bit [9:8] Pixel format Bit [10] Clip enable Bit [11] Source color key enable Bit [12] Destination color key enable DW13: Bit [31:0] Pattern matrix low DW DW14: Bit [31:0] Pattern matrix high DW DW15: reserved The upper register value will be automatically loaded into the manual mode BitBLT registers when doing corresponding BitBLT. Writing this register will start the queued BitBLT immediately if in auto mode
31:27	-	5'h0	Reserved

7 PCI Subsystem

7.1 Overview

Atlas™-II implements a high-speed internal PCI bus that can run up to 100MHz. The PCI Bus Arbiter can support up to 9 PCI devices: ROM/SRAM Controller, SDIO Host Controller, USB-OTG Interface, IDE Interface, PCMCIA/CF Interface, and etc.

All the PCI device registers in PCI Subsystem share the same 256MB segment of the RISC address space (0x5000_0000~5FFF_FFFF).

Table 31. PCI Device Registers Mapping

RISC Address Range	Usage	Resource Size
0x57F0_0000~57FF_FFFF	USB-OTG	1MB
0x57E0_0000~57EF_FFFF	Reserved	1MB
0x57D0_0000~57DF_FFFF	IDE	1MB
0x57C0_0000~57CF_FFFF	PCMCIA	1MB
0x57B0_0000~57BF_FFFF	Reserved	1MB
0x57A0_0000~57AF_FFFF	PCI_ROM	1MB
0x5790_0000~579F_FFFF	PCI_COPY	1MB
0x5780_0000~578F_FFFF	YUV_CHG	1MB
0x5770_0000~577F_FFFF	IPOLATE	1MB
0x5760_0000~576F_FFFF	YUV_RGB	1MB
0x5600_0000~560F_FFFF	SDIO	1MB
Others	Reserved	-

NOTE: The address space from 0x6000_0000~6FFF_FFFF is also accessed through PCI Bus, but it is only used for mapping the ROM/SRAM when it's programmed to be the system memory controller. Besides, the address space from 0x2000_0000~3FFF_FFFF is also accessed through PCI Bus, but it is only used for mapping the PC Card Memory Space. Please refer to the following sections for more details.

7.2 System to PCI Bridge

There is a bridge between the System Bus and PCI Bus, which covers the IO and memory accesses between the System Bus and PCI Bus. For IO accesses, the PCI Bridge will convert the RISC/DSP IO accesses into PCI data transactions and passes to the corresponding PCI devices. For memory access, the PCI Bridge is both a master and a slave on the System Bus. In most cases when PCI devices transfer data to/from DRAM Controller, the PCI Bridge acts as the bus master on the System Bus. But in some other cases user may select the SRAM Controller as the bus slave of some bus masters on the System Bus.

The following figure shows the block diagram of System to PCI Bridge.

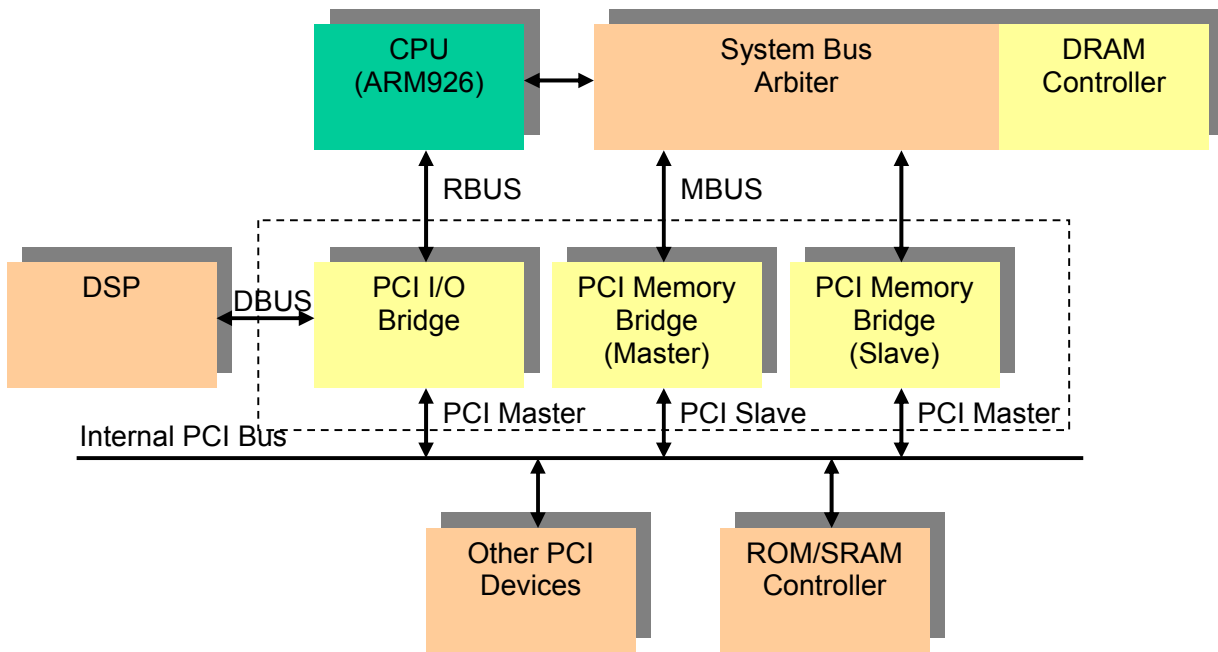


Figure 33. Block Diagram of PCI Bridge

Both RISC and DSP can access the System to PCI Bridge registers.

7.2.1 PCI Bridge RISC IO Registers

Table 32. PCI Bridge RISC IO Register Mapping

RISC Address <11:0>	Register	Description
0x0000	SYS2PCI_RETRY	System to PCI Bridge retry value
0x0004	SYS2PCI_INT_EN	System to PCI Bridge interrupt enable
0x0008	SYS2PCI_INT_STATUS	System to PCI Bridge interrupt status
0x000C	SYS2PCI_WRITEBUFFER	System to PCI Bridge write buffer set
0x0010	SYS2PCI_TIMEOUT	System to PCI Bridge timeout
Others	-	Reserved

- **PCI Bridge Retry Value Register (SYS2PCI_RETRY) – 0x0000**

Bit	Name	Default	Description
5:0 (R/W)	RETRY<5:0>	6'h8	System to PCI Bridge retry value
31:6	-	-	Reserved

SYS2PCI_RETRY is a 6-bit register that control the retry time on the PCI side. When PCI access is terminated with retry, the System to PCI Bridge will wait for $SYS2PCI_RETRY * T_{PCI_CLK}$ before it retry the same PCI access.

- **PCI Bridge Interrupt Enable Register (SYS2PCI_INT_EN) – 0x0004**

Bit	Name	Default	Description
0 (R/W)	reserved	1'b0	Need to be written with 0

1 (R/W)	MABT	1'b0	System to PCI Bridge Master abort interrupt enable 1: enable the interrupt 0: disable the interrupt
2 (R/W)	TIMOUT	1'b0	System to PCI Bridge Time out interrupt enable 1: enable the interrupt 0: disable the interrupt
31:3	-	-	Reserved

System to PCI Bridge Master abort interrupt enable: When there is master abort on PCI bus, the current access will be cancelled. A master abort interrupt will be generated.

System to PCI Bridge Time out interrupt enable: When an access on PCI bus is terminated with retry X (X>TIMOUT_VALUE) times, the current access will be cancelled. A time out abort interrupt will be generated.

- **PCI Bridge Interrupt Statue Register (SYS2PCI_INT_STATUS) – 0x0008**

Bit	Name	Default	Description
0 (R/W)	-	1'b0	Reserved (Need to be written with 0)
1 (R/W)	MABT	1'b0	System to PCI Bridge Master abort interrupt Read: 1: interrupt pending 0: no interrupt pending Write: 1: clear the interrupt pending 0: no effect
2 (R/W)	TIMOUT	1'b0	System to PCI Bridge Time out interrupt Read: 1: interrupt pending 0: no interrupt pending Write: 1: clear the interrupt pending 0: no effect
31:3	-	-	Reserved

System to PCI Bridge Master abort interrupt: When there is master abort on PCI bus, the current access will be cancelled. A master abort interrupt will be generated.

System to PCI Bridge Time out interrupt: When an access on PCI bus is terminated with retry X(X>TIMEOUT) times, the current access will be cancelled. A time out abort interrupt will be generated.

- **PCI Bridge Write Buffer Enable Register (SYS2PCI_WRITEBUFFER) – 0x000C**

Bit	Name	Default	Description
0 (R/W)	WB	1'b0	System to PCI Bridge write buffer enable 1: enable write buffer 0: disable write buffer When write buffer is enabled, RISC IO write to PCI device will be finished right away when System to PCI Bridge FIFO is ready to hold the command. Then System to PCI Bridge will transfer the command into the FIFO on PCI bus. But the write operation will only be truly finished when the command is issued on PCI bus and finished.

			It is recommended to set the write buffer to enable, which will speed up to write operation. However user needs to perform a read to the same address to guarantee the write is really finished on device.
1 (R/W)	RM	1'b1	System to PCI Bridge read mode 1: read will wait for current transfer to system memory finish. 0: IO read have no relation with transfer to system memory
31:2	-	-	Reserved

- **System to PCI Bridge Time Out Register (SYS2PCI_TIMEOUT) – 0x0010**

Bit	Name	Default	Description
15:0 (R/W)	TIMEOUT<15:0>	16'hfff	System to PCI Bridge timeout value
31:6	-	-	Reserved

System to PCI Bridge timeout value is a 16 bit register that control the retry number on PCI side. If a read/write from RISC is retried for more than TIMEOUT times, then this command will be discarded. An interrupt will occur if user set the interrupt enable bit. If this value is set to 0, retry number time out will never occur.

7.2.2 PCI Bridge DSP IO Registers

To Access PCI from DSP side, System to PCI Bridge contains specific registers to convert the 16bit access on DSP side to 32 bit access on PCI side.

Table 33. PCI Bridge DSP IO Register Mapping

DSP byte Address <5:0>	Register	Description
0x0000	SYS2PCI_DSP_OPERATE	System to PCI Bridge DSP Operation
0x0002	SYS2PCI_DSP_ADDL	System to PCI Bridge DSP address low
0x0004	SYS2PCI_DSP_ADDH	System to PCI Bridge DSP address high
0x0006	SYS2PCI_DSP_DATAH	System to PCI Bridge DSP data high
0x0008	SYS2PCI_DSP_DATAH	System to PCI Bridge DSP data high
0x000a	SYS2PCI_DSP_STATUS	System to PCI Bridge DSP status
Others	-	Reserved

To perform a read operation, DSP needs to:

1. DSP need to write SYS2PCI_DSP_ADDL, SYS2PCI_DSP_ADDH for correct PCI address.
2. Write SYS2PCI_DSP_OPERATE with correct BE, RW=1.
3. Poll SYS2PCI_DSP_OPERATE until RDY=1.
4. Read data from SYS2PCI_DSP_DATAH, SYS2PCI_DSP_DATAH.

To perform a write operation, DSP needs to:

1. Write SYS2PCI_DSP_ADDL, SYS2PCI_DSP_ADDH for correct PCI address.
2. Write SYS2PCI_DSP_DATAH, SYS2PCI_DSP_DATAH for write data on PCI
3. Write SYS2PCI_DSP_OPERATE with correct BE, RW=0.
4. Poll SYS2PCI_DSP_OPERATE until RDY=1.

- **PCI Bridge DSP Operation Register (SYS2PCI_DSP_OPERATE) – 0x0000**

Bit	Name	Default	Description
0 (R/w)	RW	1'b1	1: DSP read PCI 0: DSP write PCI
1 (R)	RDY	1'b1	Read: 1: DSP operation finished. 0: DSP operation not finished Write: 1: reset to 0 0: reset to 0
3:2	-	-	Reserved
7:4 (R/w)	BE<3:0>	4'h0	Byte enable for PCI read/write BE<0> - SYS2PCI_DSP_DATA[7:0] BE<1> - SYS2PCI_DSP_DATA[15:8] BE<2> - SYS2PCI_DSP_DATAH[7:0] BE<3> - SYS2PCI_DSP_DATAH[15:8] 1: enable byte 0: disable the byte
15:8	-	-	Reserved

- **PCI Bridge DSP Address Low Register (SYS2PCI_DSP_ADDL) – 0x0002**

Bit	Name	Default	Description
15:0 (R/w)	ADDL	16'h0	PCI address low for DSP IO access on PCI bus

- **PCI Bridge DSP Address Low Register (SYS2PCI_DSP_ADDL) – 0x0002**

Bit	Name	Default	Description
15:0 (R/w)	ADDL	16'h0	PCI address low for DSP IO access on PCI bus

- **PCI Bridge DSP Address High Register (SYS2PCI_DSP_ADDH) – 0x0004**

Bit	Name	Default	Description
15:0 (R/w)	ADDH	16'h0	PCI address high for DSP IO access on PCI bus

- **PCI Bridge DSP Data Low Register (SYS2PCI_DSP_DATA[7:0]) – 0x0006**

Bit	Name	Default	Description
15:0 (R/w)	DATA[7:0]	16'h0	PCI data low for DSP IO access on PCI bus. When read, this register contains data for read. When write, this register contains data for write.

- **PCI Bridge DSP Data High Register (SYS2PCI_DSP_DATAH[7:0]) – 0x0008**

Bit	Name	Default	Description
15:0	DATAH[7:0]	16'h0	PCI data high for DSP IO access on PCI bus.

(R/W)			When read, this register contains data for read. When write, this register contains data for write.
-------	--	--	--

- **PCI Bridge DSP Status Register (SYS2PCI_DSP_STATUS) – 0x000A**

Bit	Name	Default	Description
0 (R)	-	1'b0	Reserved
1 (R)	MABT	1'b0	System to PCI Bridge Master abort interrupt Read: 1: interrupt pending 0: no interrupt pending Write: no effect
2 (R)	TIMEOUT	1'b0	System to PCI Bridge Time out interrupt Read: 1: interrupt pending 0: no interrupt pending Write: no effect
31:3	-	-	Reserved

This register is read only about interrupt status information for DSP access. It requires RISC access to clear the interrupt status.

7.3 ROM/SRAM Controller

7.3.1 Overview

The Atlas-II™ ROM/SRAM interface supports both fixed-latency and variable-latency devices, such as: ROM, NOR Flash, SRAM, or SRAM-like devices.

The ROM/SRAM Controller is one of the PCI devices on the internal PCI Bus. It can be programmed as a normal peripheral interface; or, it can be programmed as the system memory controller.

7.3.2 Pin Description

The following table shows the ROM/SRAM interface pins and their functional description. The ROM/SRAM interface pins are multiplexed with others interfaces (PCMCIA, IDE and NAND Flash). Please refer to Developer's Manual for more information.

Table 34. ROM/SRAM Interface Pin Description

Pin name	Pin Direction	Description
X_FCE_B<3:0>	Output	4 Chip select pin. Active low
X_FA<25:0>	Output	Address pin
X_FD<31:0>	Bidirectional	Data pin. Depending on the device bus width, some of them will be used. If device only has 8-bit bus then X_FD<7:0> is used.
X_FOE_B	Output	Data output enable. Active low
X_FWE_B	Output	Data write enable. Active low
X_FBE<3:0>	Output	Byte-enable signals. Active low. Depending on the device bus width, some of them will be used. If device has 8 bit bus then only X_FBE_0 will be used.
X_FRDY_B	Input	Ready pin. Active low. Only device with variable access feature will need it.

7.3.3 Address Mapping

ROM/SRAM interface supports read/write to external 8/16/32-bit ROM/SRAM or SRAM-like devices. User can access ROM/SRAM through either RISC IO access (RISC only) or PCI IO access (Both RISC and DSP). The address mapping is shown in the following table.

Table 35. RISC IO access/PCI IO access address Mapping

RISC IO access address	PCI IO access address	X_FA	X_FCE_B[3-0]
0x10000000 ~ 0x13FFFFFF	0x60000000~ 0x63FFFFFF	0x00000000~ 0x3FFFFFFF	X_FCE_B_0 active; Other banks inactive
0x14000000 ~ 0x17FFFFFF	0x64000000~ 0x67FFFFFF	0x00000000~ 0x3FFFFFFF	X_FCE_B_1 active; Other banks inactive
0x18000000 ~ 0x1BFFFFFF	0x68000000~ 0x6BFFFFFF	0x00000000~ 0x3FFFFFFF	X_FCE_B_2 active; Other banks inactive
0x1C000000 ~ 0x1FFFFFFF	0x6C000000~ 0x6FFFFFFF	0x00000000~ 0x3FFFFFFF	X_FCE_B_3 active; Other banks inactive

NOTE: It is recommended to use PCI IO access mode.

ROM/SRAM interface will accept either 8/16/32-bit from RISC/PCI IO access, translate them onto 8bit/16bit/32bit ROM/SRAM port. Following table shows the detail access translation.

The first & second columns list the access from RISC/PCI IO. The byte enable on RISC/PCI is active '1' & depends on the access mode, for example an 8bit access will only have one byte enabled. The third column defines the ROM/SRAM interface bus width. The fourth & fifth columns list the translated access on ROM/SRAM interface, some translated access will require two data access to finish one RISC/PCI IO access. For example a 16bit access from RISC/PCI IO will be translated into 2 accesses on ROM/SRAM interface.

All other accesses not listed in the table is not supported (for example when RISC/PCI access use 16 bit mode, the BE on RISC/PCI can not be 4'b0110)

Table 36. RISC IO access/PCI IO access width translation

RISC/PCI access mode & byte address	Byte Enable on RISC/PCI	BUS_WIDTH in ROM_CFGx	Actions taken on ROM/SRAM interface	X_FBE[3-0]	Notes
8-bit / Addr[31-0]	4'b0001/ 4'b0010/ 4'b0100/ 4'b1000	8-bit	Access Addr[25-0]	4'b1110	1
16-bit/ Addr[31-0]	4'b00xy / 4'bxy00	8-bit	Access Addr[25-0], Access Addr[25-0]+1	4'b111y 4'b111x	1,2
32-bit/ Addr[31-0]	4'bxywz	8-bit	Access Addr[25-0] Access Addr[25-0] +1 Access Addr[25-0] +2 Access Addr[25-0] +3	4'b111z 4'b111w 4'b111y 4'b111x	2
8-bit/ Addr[31-0]	4'b0001/ 4'b0010/ 4'b0100/ 4'b1000	16-bit	Access Addr[25-1]	4'b1110/ 4'b1101/ 4'b1110/ 4'b1101	1
16-bit/ Addr[31-0]	4'b00xy / 4'bxy00	16-bit	Access Addr[25-1],	4'b11(~x)(~y)	1,2,3
32-bit/ Addr[31-0]	4'bxywz	16-bit	Access Addr[25-1]	4'b11(~w)(~z)	3

			Access Addr[25-1] +1	4'b11(~x)(~y)	
8-bit/ Addr[31-0]	4'b0001/ 4'b0010/ 4'b0100/ 4'b1000	32-bit	Access Addr[25-2]	4'b1110/ 4'b1101/ 4'b1010/ 4'b0111	1
16-bit/ Addr[31-0]	4'b00xy / 4'bxy00	32-bit	Access Addr[25-2],	4'b11(~x)(~y)	1,3
32-bit/ Addr[31-0]	4'bxyzwz	32-bit	Access Addr[25-2]	4'b(~x)(~y)(~w) (~z)	3

NOTE:

1. The slash in "BE on RISC/PCI" means the Byte Enable can be either one.
2. x, y, w, z stand for the bit enable can be either 1'b0 or 1'b1.
3. (~x) means the invert of x

7.3.4 ROM/SRAM Controller Registers

Table 37. ROM RISC IO Register Mapping

RISC Address <11:0>	Register	Description
0x0000	ROM_CFG0	ROM configure register for CS0
0x0004	ROM_CFG1	ROM configure register for CS1
0x0008	ROM_CFG2	ROM configure register for CS2
0x000C	ROM_CFG3	ROM configure register for CS3
Others	-	Reserved

- **ROM CS0 Configure Register (ROM_CFG0) – 0x0000**

ROM_CFG0 controls the timing & access mode for ROM port CS0.

Bit	Name	Default	Description
7:0 (R/W)	WGAP<7:0>	7'h0	Write interval (WGAP * T _{clk_pci}) between each two write.
13:8 (R/W)	TACC<5:0>	6'h0A	Access time (TACC * T _{clk_pci}) is the read/write signal active time.
15:14 (R/W)	TCES<1:0>	2'h0	CS active time (TCES * T _{clk_pci}) before read/write signal is active.
21:16 (R/W)	TNACC<5:0>	6'h0A	Read burst access time. For burst read the next access time can be smaller than the first access time
23:22 (R/W)	TDF<1:0>	2'h0	Read data hold time.
24 (R/W)	DWORD_ACC	1'b1	Enable DWORD access.
25 (R/W)	BURST_READ	1'b1	Enable burst read.
26 (R/W)	BURST_WRITE	1'b1	Enable burst write.
27 (R/W)	VARI_ACC	1'b0	Enable variable access mode.
28 (R/W)	-	1'b0	Reserved (Need to be written as 0)
29 (R/W)	WRITE_EN	1'b1	Enable write. 1: write to ROM address will be written to ROM port

			0: write to ROM address will be masked. No write will occur on ROM port.
31:30 (R/W)	BUS_WIDTH	Defined by reset pin value	Define the bus width of ROM port. 00: bus width is 8bit. 01: bus width is 16bit. 10: bus width is 32bit. 11: reserved.

NOTE: T_{clk_pci} = period of one PCI clock.

- **ROM CS1 Configure Register (ROM_CFG1) – 0x0004**

ROM_CFG1 controls the timing & access mode for ROM port CS1.

Bit	Name	Default	Description
7:0 (R/W)	WGAP<7:0>	7'h0	Write interval ($WGAP * T_{clk_pci}$) between each two write.
13:8 (R/W)	TACC<5:0>	6'h0	Access time ($TACC * T_{clk_pci}$) is the read/write signal active time.
15:14 (R/W)	TCES<1:0>	2'h0	CS active time ($TCES * T_{clk_pci}$) before read/write signal is active.
21:16 (R/W)	TNACC<5:0>	6'h0	Read burst access time. For burst read the next access time can be smaller than the first access time
23:22 (R/W)	TDF<1:0>	2'h0	Read data hold time.
24 (R/W)	DWORD_ACC	1'b0	Enable DWORD access.
25 (R/W)	BURST_READ	1'b0	Enable burst read.
26 (R/W)	BURST_WRITE	1'b0	Enable burst write.
27 (R/W)	VARI_ACC	1'b0	Enable variable access mode.
28 (R/W)	reserved	1'b0	Need to be written as 0
29 (R/W)	WRITE_EN	1'b0	Enable write. 1: write to ROM address will be written to ROM port 0: write to ROM address will be masked. No write will occur on ROM port.
31:30 (R/W)	BUS_WIDTH	2'b00	Define the bus width of ROM port. 00: bus width is 8bit. 01: bus width is 16bit. 10: bus width is 32bit. 11: reserved.

- **ROM CS2 Configure Register (ROM_CFG2) – 0x0008**

ROM_CFG2 controls the timing & access mode for ROM port CS2. The detail register definition is the same as *ROM_CFG1*.

- **ROM CS3 Configure Register (ROM_CFG3) – 0x000C**

ROM_CFG3 control the timing & access mode for ROM port CS3. The detail register definition is the same as *ROM_CFG1*.

The following figures show some configuration examples.

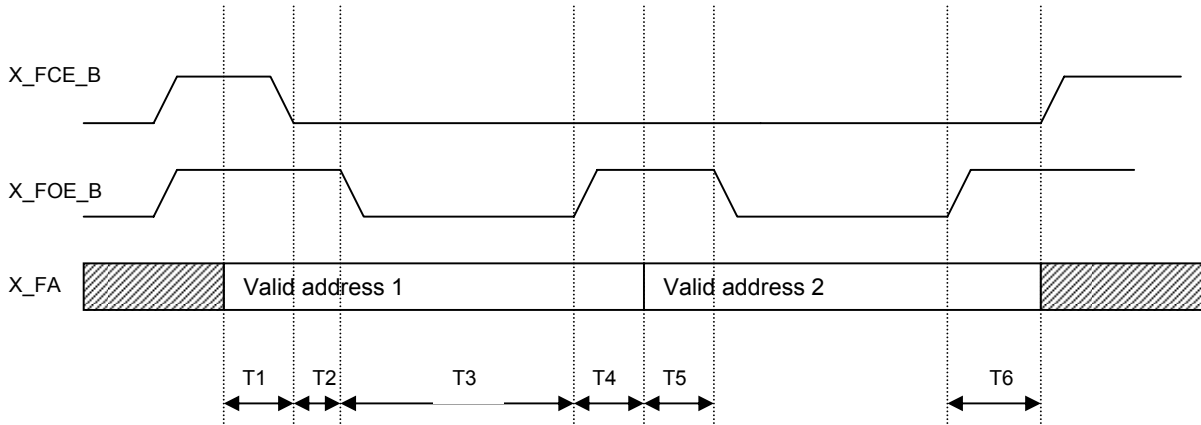


Figure 34. 32-bit Data Fix-latency Read (BURST_READ=0, DWORD_ACCESS=1, BUS_WIDTH=1, VARI_ACC=0)

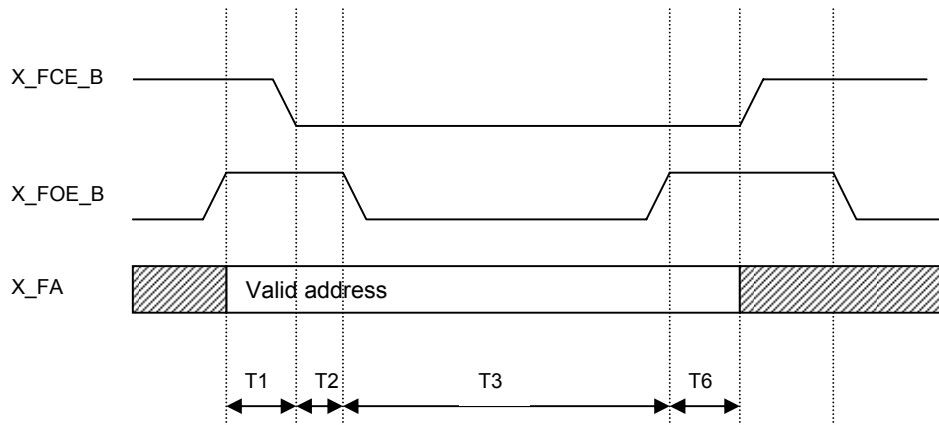


Figure 35. 16-bit Data Fix-latency Read (BURST_READ=0, DWORD_ACCESS=0, BUS_WIDTH=1, VARI_ACC=0)

NOTE: X_FD from ROM/SRAM device will be latched 1 T clk_pci ahead of X_FOE_B rising edge.

T1	T2	T3	T4	T5	T6
1 T _{clk_pci}	T _{ces} * T _{clk_pci}	T _{acc} * T _{clk_pci}	1 T _{clk_pci}	1 T _{clk_pci}	T _{df} * T _{clk_pci}

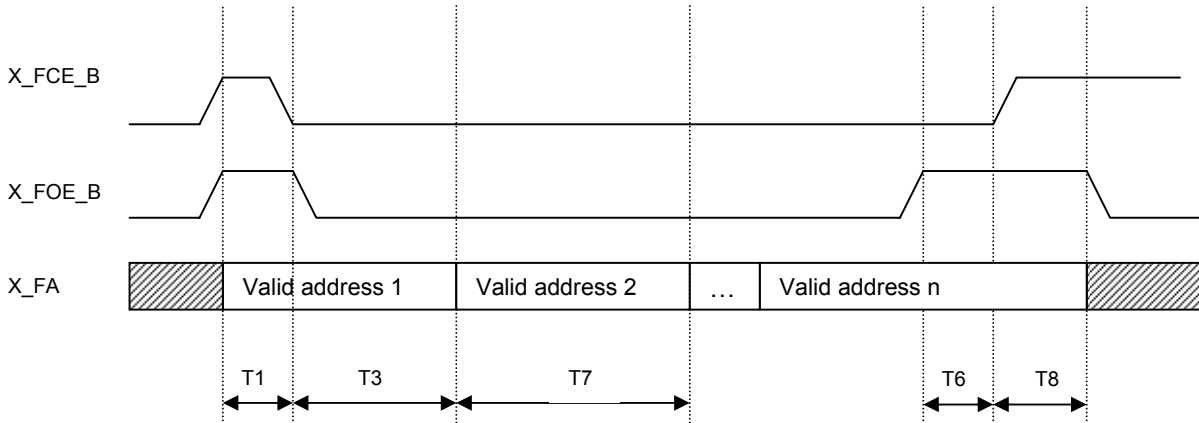


Figure 36. 32-bit Data Fix-latency Read (BURST_READ=1, DWORD_ACCESS=1, BUS_WIDTH=0/1/2, VARI_ACC=0)

NOTE:
 BUS_WIDTH=0: n=4
 BUS_WIDTH=1: n=2
 BUS_WIDTH=2: n=1

T1	T2	T3	T6	T7	T8
$1 \cdot T_{\text{clk_pci}}$	$T_{\text{ces}} \cdot T_{\text{clk_pci}}$	$T_{\text{acc}} \cdot T_{\text{clk_pci}}$	$T_{\text{df}} \cdot T_{\text{clk_pci}}$	$T_{\text{nacc}} \cdot T_{\text{clk_pci}}$	$1 \cdot T_{\text{clk_pci}}$

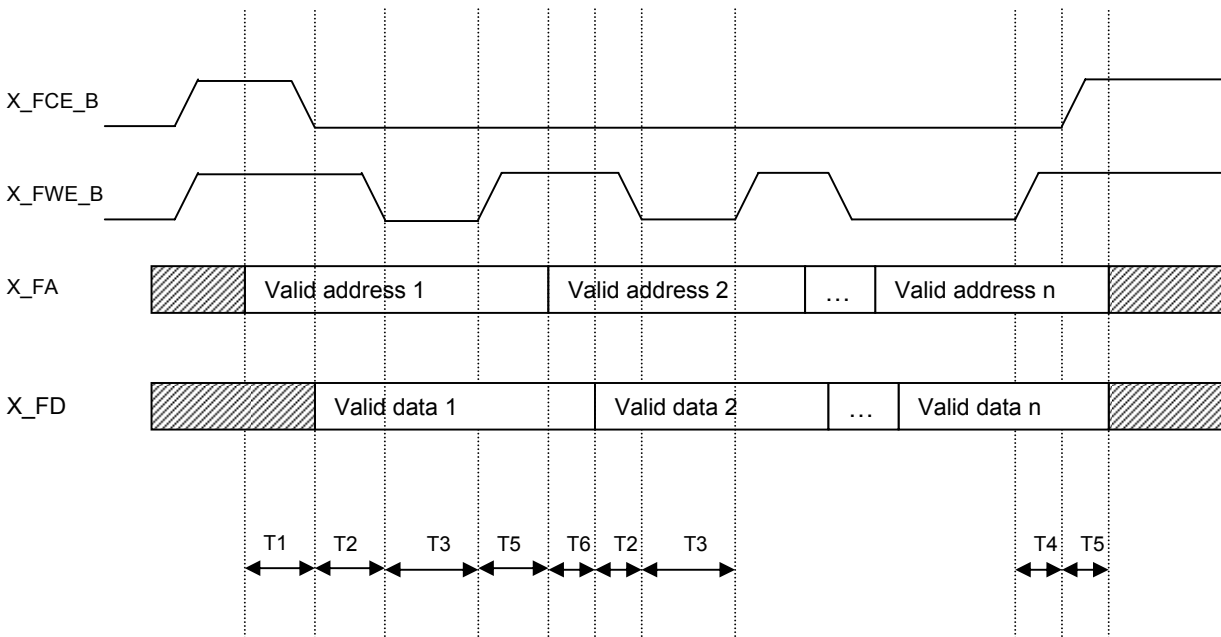


Figure 37. 32-bit Data Fix latency Write (BURST_WRITE=1, DWORD_ACCESS=1, BUS_WIDTH=0/1/2, VARI_ACC=0)

NOTE:
 BUS_WIDTH=0: n=4
 BUS_WIDTH=1: n=2

BUS_WIDTH=2: n=1

T1	T2	T3	T4	T5	T6
$1 T_{\text{clk pci}}$	$T_{\text{ces}} * T_{\text{clk pci}}$	$T_{\text{acc}} * T_{\text{clk pci}}$	$1 * T_{\text{clk pci}}$	$1 T_{\text{clk pci}}$	$1 T_{\text{clk pci}}$

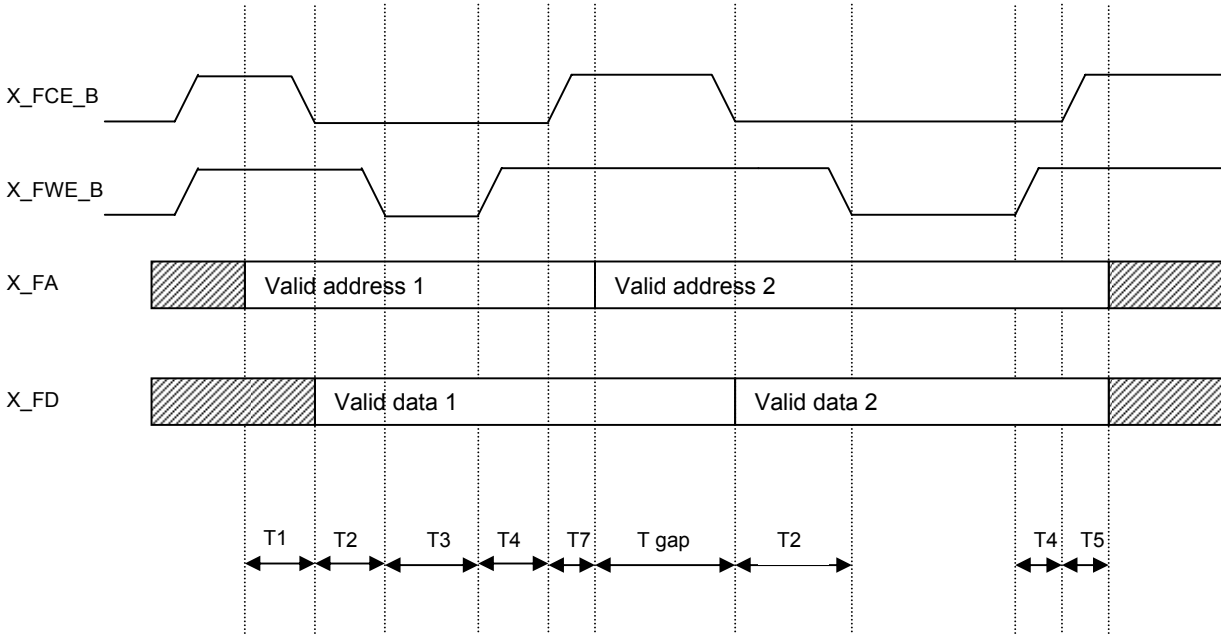
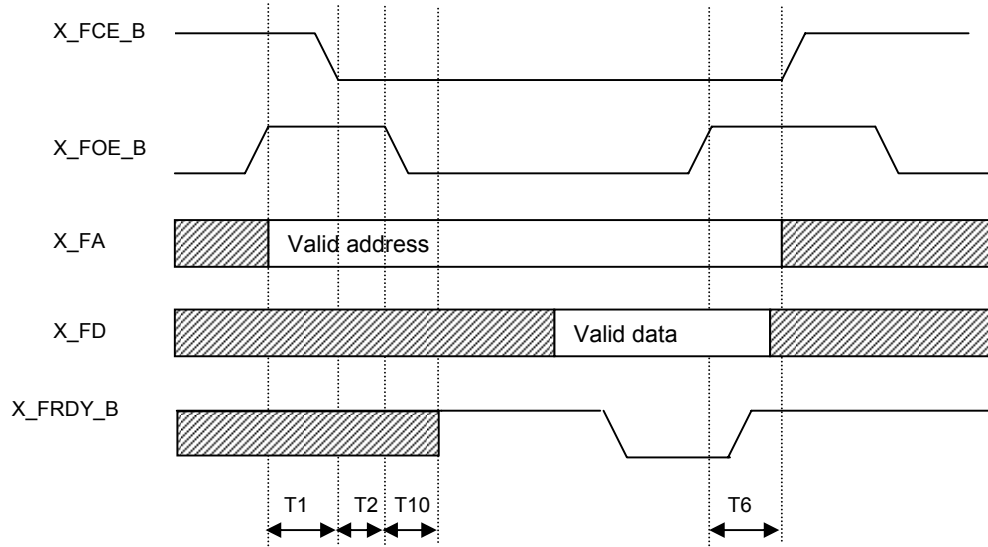


Figure 38. 32-bit Data Fix-latency Write (BURST_WRITE=0, DWORD_ACCESS=1, BUS_WIDTH=1, VARI_ACC=0)

NOTE:

T1	T2	T3	T4	T5	T6	T7
$1 T_{\text{clk pci}}$	$T_{\text{ces}} * T_{\text{clk pci}}$	$T_{\text{acc}} * T_{\text{clk pci}}$	$1 * T_{\text{clk pci}}$	$1 T_{\text{clk pci}}$	$1 T_{\text{clk pci}}$	$1 T_{\text{clk pci}}$

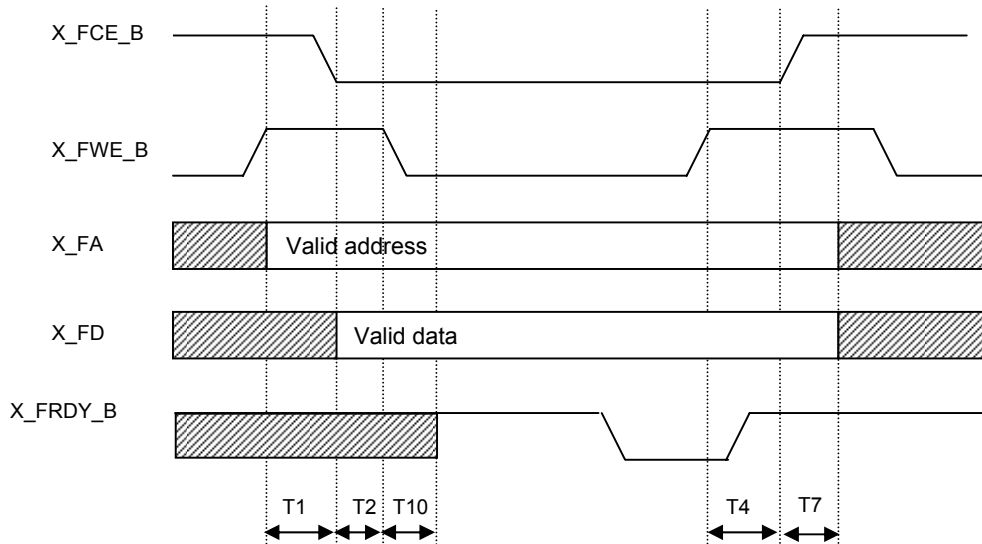


**Figure 39. 16-bit Data Variable-latency Read
(BURST_READ=0, DWORD_ACCESS=0, BUS_WIDTH=1,
VARI_ACC=1)**

NOTE: X_FD from ROM/SRAM device will be latched 1 T clk_pci ahead of X_FOE_B rising edge.

T1	T2	T3	T6	T10
$1 T_{clk_pci}$	$T_{ces} * T_{clk_pci}$	$T_{acc} * T_{clk_pci}$	$T_{df} * T_{clk_pci}$	$T_{acc} * T_{clk_pci}$

NOTE: After T10 ROM/SRAM interface will detect the X_FRDY_B pin. When it is low the data is valid.



**Figure 40. 16-bit Data Variable-latency Write
(BURST_WRITE=0, DWORD_ACCESS=0,
BUS_WIDTH=1, VARI_ACC=1)**

NOTE: After T10, ROM/SRAM interface will detect the X_FRDY_B pin. When it is low the write can be finished.

T1	T2	T4	T7	T10
1 T _{clk_pci}	T _{ces} * T _{clk_pci}	1 * T _{clk_pci}	1 T _{clk_pci}	T _{acc} * T _{clk_pci}

7.4 SDIO Host Controller

7.4.1 Overview

SDIO/SD Host Controller is a Host Controller. This module conforms to SDIO Specification and SD memory card physical layer specifications. Power consumption of the system can be kept to a minimum through gated clock control.

The SDIO/SD Host Controller handles SDIO/SD Protocol at transmission level by packing data, adding cyclic redundancy check (CRC), start/end bit, and checking for transaction format correctness. SD Mode wide bus width is also supported.

Here are the key features of SDIO Host Controller:

- Meets SD Host Controller Standard Specification Draft Version 1.0
- Card Detection (Insertion / Removal)
- Password protection of Cards
- Host clock rate variable between 0 and 25 MHz
- Supports 1 and 4 bit SD modes.
- Allows card to interrupt host in 1 and 4 bit SD modes.
- Cyclic Redundancy Check CRC7 for command and CRC16 for data integrity
- Host Initiates Direct read/write (IO52) and Extended read/write (IO53) transactions.
- Host can initiate CMD52 while data transfer for CMD53 is in progress.
- Supports Read wait Control operation.
- Supports Suspend/Resume operation.
- Supports Multi Block read and Write

7.4.2 Pin Description

Table 38. SDIO Host Interface Pin Description

Pin name	IO Direction	Description
SD_CLK	Output	SD bus clock
SD_CMD	Bi-direction	SD bus command signal
SD_DAT0	Bi-direction	SD bus data signal d0
SD_DAT1	Bi-direction	SD bus data signal d1
SD_DAT2	Bi-direction	SD bus data signal d2
SD_DAT3	Bi-direction	SD bus data signal d3
SD_CD	Input	SD card detected pin (mux with gpio3)

7.4.3 Functional Description

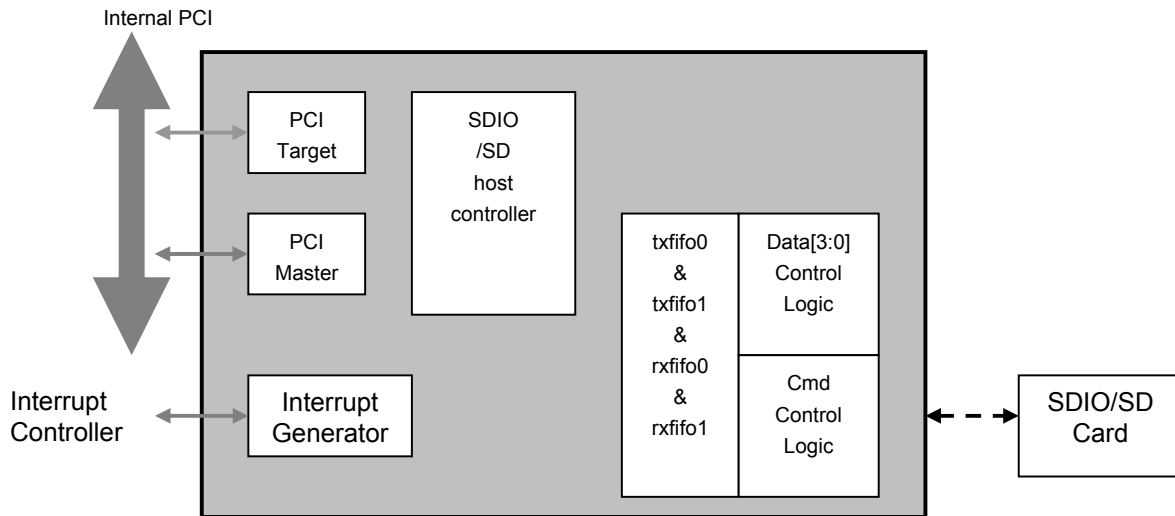


Figure 41. SD Host Block Diagram

The SD/SDIO Host Controller provides Programmed IO data transfer method. In programmed IO method, the Host Driver transfer data using the Buffer Data Port Register.

Interrupt controller

The SD/SDIO Host Controller generates interrupt to the RISC.

SDIO/SD Host Controller

The SDIO/SD Host Controller comprises of Host interface, SD/SDIO controller registers, Bus monitor, Clock Generator, CRC generator and checker (CRC7 and CRC16). The SD/SDIO controller registers are programmed by the Host (Driver). Interrupts are generated to the Host based on the values set in the Interrupt Status register (*SD_INT_STATUS*) and Interrupt Enable register (*SD_INT_EN*). Bus monitor will check for any violations occurring in the SD bus and timeout conditions. The Clock Generator will generate the SD clock depending on the value programmed by the Host Driver in the Host Control Register (*SD_HOST_CTRL_1*). The CRC7 and CRC16 generator calculate the CRC for command and Data respectively to send the CRC to the SD/SDIO card. The CRC7 and CRC16 checker checks for any CRC error in the Response and Data send by the SD/ SDIO card.

DATA FIFO

The SD/SDIO Host Controller uses four 1K dual port FIFO for performing data transactions (two 1K FIFO for read transaction and two 1K FIFO for write transaction). During a write transaction (data transferred from Host to SD/SDIO card), the data will be filled in to the first and second FIFO alternatively. When data from first FIFO is transferring to the SD card, the second FIFO will be filled and vice versa. The two FIFO's are alternatively used to store data which will give maximum throughput. During a read transaction (data transferred from SD/SDIO card to Host), the data from SD card will be written in to the two FIFO's alternatively. When data from one FIFO is transferring to the Host, the second FIFO will be filled and vice versa and thereby the throughput will be maximized. If the Host controller cannot accept any data from SD card, then it will issue read wait to stop the data coming from card.

DAT<3:0> Control Logic

The DAT<3:0> control logic block transmit data through the data line during write transaction and receive data through the data line during read transaction. The interrupt generated from the SD/SDIO card is detected.

Command Control Logic

The Command control logic block sends the command through the CMD line and receives the response coming from the SD/SDIO card.

Power Control

The SD/SDIO Host Controller supply SD Bus Power depending on the value programmed in the Power Control Register by the Host Driver. The Host Driver has the responsibility to supply SD Bus Voltage according to card OCR and supply voltage capabilities depending on the Host Controller. If the SD Bus power is set to 1 in the Power Control Register, the Host Controller shall supply voltage to the Card. If the Host Driver selects an unsupported voltage in the SD Bus Voltage Select field, the Host Controller may ignore write to SD Bus Power and keep its value at zero.

7.4.4 SDIO Host Controller Registers

NOTE: BASE address of SD host controller is 0x4600_0000. All the registers access of SD host controller by RISC is memory access.

Table 39. SDIO Host Controller Register Mapping

RISC Address <11:0>	Register	Description
0x0000	-	Reserved
0x0004	SD_BLK_CTRL	Block size/count register
0x0008	SD_CMD_ARG	Command argument register
0x000C	SD_TRAN_CTRL	Transfer control register
0x0010	SD_CARD_RESP_0	SD card response register 0
0x0014	SD_CARD_RESP_1	SD card response register 1
0x0018	SD_CARD_RESP_2	SD card response register 2
0x001C	SD_CARD_RESP_3	SD card response register 3
0x0020	SD_BUF_DATA	Transfer data buffer register
0x0024	SD_CUR_STATE	SD host controller state register
0x0028	SD_HOST_CTRL_0	SD host controller control register 0
0x002C	SD_HOST_CTRL_1	SD host controller control register 1
0x0030	SD_INT_STATUS	SD interrupt status register
0x0034	SD_INT_STATUS_EN	SD interrupt status enable register
0x0038	SD_INT_SIGNAL_EN	SD interrupt signal enable register
0x003C	SD_CMD12_ERR_STATUS	Automatic command 12 error status register
0x0040	SD_CAPABILITIES	SD host capabilities register
0x0044	-	Reserved
0x0048	SD_MAX_CURR	Maximum current capability
0x004C	SD_CLK_DELAY	SD bus clock delay register
0x0050~0xF8	-	Reserved
0x00FC	SD_SLOT_INT_VER	Slot interrupt status and version register
Others	-	Reserved

- **SDIO Block Size/Count Register (SD_BLK_CTRL) – 0x0004**

Bit	Name	Default	Description
-----	------	---------	-------------

11:0 (R/W)	BLK_SIZE	12'h0	This register specifies the block size for block data transfers for CMD17, CMD18, CMD24, CMD25, and CMD53. It can be accessed only if no transaction is executing (i.e. after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. 0000h – No Data Transfer 0001h – 1 Byte 0002h – 2 Bytes 0003h – 3 Bytes ... 01FFh – 511 Bytes 0200h – 512 Bytes ... 0800h – 2048 Bytes
14:12			Reserved
15			Reserved
31:16 (R/W)	BLK_CNT	16'h0	This is enabled when BLK_CNT_EN in the Transfer Control register is set to 1 and is valid only for multiple block transfers. The HC decrements the block count after each block transfer and stops when the count reaches zero. It can be accessed only if no transaction is executing (i.e. after a transaction has stopped). Read operations during transfer return an invalid value and write operations shall be ignored. When saving transfer context as a result of Suspend command, the number of blocks yet to be transferred can be determined by reading this register. When restoring transfer context prior to issuing a Resume command, the HD shall restore the previously save block count. 0x0000 – Stop Count 0x0001 – 1 block 0x0002 – 2 blocks ... 0xFFFF – 65535 blocks

- **SDIO Command Argument Register (SD_CMD_ARG) – 0x0008**

Bit	Name	Default	Description
31:0 (R/W)	CMD_ARG	32'h0	The SD Command Argument is specified as bit<39:8> of Command-Format.

- **SDIO Transfer Control Register (SD_TRAN_CTRL) – 0x000C**

Bit	Name	Default	Description
0			Reserved
1 (R/W)	BLK_CNT_EN	1'b0	This bit is used to enable the BLK_CNT in <i>SD_BLK_CTRL</i> register, which is only relevant for multiple block transfers. When this bit is 0, the Block Count register is disabled, which is useful in executing an infinite transfer. 0 – Disable 1 – Enable
2 (R/W)	AUTO_CMD12_EN	1'b0	Multiple block transfers for memory require CMD12 to stop the transaction. When this bit is set to 1, the HC shall issue CMD12 automatically when last block

			transfer is completed. The HD shall not set this bit to issue commands that do not require CMD12 to stop data transfer. 0 – Disable 1 – Enable
3			Reserved
4 (R/W)	DAT_TRAN_DIR_SEL		This bit defines the direction of DAT line data transfers. 0 – Write (Host to Card) 1 – Read (Card to Host)
5 (R/W)	MULT_BLK_SEL	1'h0	This bit enables multiple block DAT line data transfers. 0 – Single Block 1 – Multiple Block
15:6			Reserved
17:16 (R/W)	RES_TYPE_SEL	2'h0	Response Type Select 00 – No Response 01 – Response length 136 10 – Response length 48 11 – Response length 48 check busy after response
18		1'b0	Reserved
19 (R/W)	CMD_CRC_CHK_EN	1'b0	If this bit is set to 1, the HC shall check the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this bit is set to 0, the CRC field is not checked. 0 – Disable 1 – Enable
20 (R/W)	CMD_IND_CHK_EN	1'b0	If this bit is set to 1, the HC shall check the index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this bit is set to 0, the Index field is not checked. 0 – Disable 1 – Enable
21 (R/W)	DAT_PRE_SEL	1'h0	This bit is set to 1 to indicate that data is present and shall be transferred using the DAT line. If is set to 0 for the following: 1. Commands using only CMD line (ex. CMD52) 2. Commands with no data transfer but using busy signal on DAT<0> line (R1b or R5b ex. CMD38) 3. Resume Command 0 – No Data Present 1 – Data Present
23:22 (R/W)	CMD_TYPE	2'h0	There are three types of special commands: Suspend, Resume and Abort. These bits shall be set to 00b for all other commands. Suspend Command If the Suspend command succeeds, the HC shall assume the SD Bus has been released and that it is possible to issue the next command that uses the DAT line. The HC shall de-assert Read Wait for read transactions and stop checking busy for write transactions. The Interrupt cycle shall start, in 4-bit mode. If the Suspend command fails, the HC shall maintain its current state. And the HD shall restart the transfer by setting Continue Request in the Block Gap

			Control Register. Resume Command The HD re-starts the data transfer by restoring the registers in the range of 000-00Dh. The HC shall check for busy before starting write transfers. Abort Command If this command is set when executing a read transfer, the HC shall stop reads to the buffer. If this command is set when executing a write transfer, the HC shall stop driving the DAT line. After issuing the Abort command, the HD should issue a software reset 00b – Normal 01b – Suspend 10b – Resume 11b – Abort
29:24 (R/W)	CMD_INDEX	6'h0	This bit shall be set to the command number (CMD0-63, ACMD0-63).
31:30		2'h0	Reserved

Table 40. Definition of Transfer Type

Multi/Single Block Select	Block Count Enable	Block Count Function	Transfer Type
0	Don't Care	Don't Care	Single Transfer
1	0	Don't Care	Infinite Transfer
1	1	Not Zero	Multiple Transfer
1	1	Zero	Stop Multiple Transfer

Table 41. Definition of Response Type

Response type	Index Check Enable	CRC Check Enable	Name of Response Type
00	0	0	No Response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R6, R5
11	1	1	R1b, R5b

- **SDIO Card Response Register 0-3 (SD_CARD_RESP_x) – 0x0010~0x001C**

Bit	Name	Default	Description
127:0 (R)	RESP<127:0>	128'h0	The following table describes the mapping of command responses from the SD Bus to this register for each response type. In the table, R<> refers to a bit range within the response data as transmitted on the SD Bus, RESP<> refers to a bit range within the Response register.

Table 42. Response Bit Definition for Each Response Type

Response Type	Description	Response Field	Response Register
R1, R1b (normal response)	Card Status	R<39:8>	RESP<31:0>
R1b (Auto CMD12 response)	Card Status for Auto CMD12	R<39:8>	RESP<127:96>
R2 (CID, CSD Register)	CID or CSD reg. incl.	R<127:8>	RESP<119:0>

R3 (OCR Register)	OCR Register for memory	R<39:8>	RESP<31:0>
R4 (OCR Register)	OCR Register for I/O etc	R<39:8>	RESP<31:0>
R5, R5b	SDIO Response	R<39:8>	RESP<31:0>
R6 (Published RCA response)	New published RCA[31:16] etc	R<39:8>	RESP<31:0>

- **SDIO Buffer Data Port Register (SD_BUF_DATA) – 0x0020**

Bit	Name	Default	Description
31:0 (R/W)	BUF_DATA_PORT	32'h0	The Host Controller Buffer can be accessed through this 32-bit Data Port Register.

- **SDIO Current State Register (SD_CUR_STATE) – 0x0024**

Bit	Name	Default	Description
0 (R)	CMD_INHABIT(CMD)	1'b0	If this bit is 0, it indicates the CMD line is not in use and the HC can issue a SD command using the CMD line. This bit is set immediately after the SD_TRAN_CTRL (0x00C) is written. This bit is cleared when the command response is received. Even if the CMD_INHABIT (DAT) is set to 1, Commands using only the CMD line can be issued if this bit is 0. Changing from 1 to 0 generates a CMD_END interrupt in the SD_INT_STATUS register. If the HC cannot issue the command because of a command conflict error or because of AUTO_CMD12_ERR , this bit shall remain 1 and the CMD_END is not set. Status issuing Auto CMD12 is not read from this bit.
1 (R)	CMD_INHABIT(DAT)	1'b0	This status bit is generated if either the DAT_LINE_ACTIVE or the RD_TRAN_ACTIVE is set to 1. If this bit is 0, it indicates the HC can issue the next SD command. Commands with busy signal belong to CMD_INHIBIT (DAT) (ex. R1b, R5b type). Changing from 1 to 0 generates a TRAN_END interrupt in the SD_INT_STATUS register. Note: The SD Host Driver can save registers in the range of 000-00Dh for a suspend transaction after this bit has changed from 1 to 0. 1 - Cannot issue command which uses the DAT line 0 - Can issue command which uses the DAT line
2 (R)	DAT_LINE_ACTIVE	1'b0	This bit indicates whether one of the DAT line on SD bus is in use. 1 – DAT line active 0 – DAT line inactive
7:3 (R)			Reserved
8 (R)	WT_TRAN_ACTIVE	1'b0	This status indicates a write transfer is active. If this bit is 0, it means no valid write data exists in the HC. This bit is set in either of the following cases: 1) After the end bit of the write command.

			<p>2) When writing a 1 to CONTINUE_REQ in the SD_HOST_CTRL_0 register to restart a write transfer.</p> <p>This bit is cleared in either of the following cases:</p> <ol style="list-style-type: none"> 1) After getting the CRC status of the last data block as specified by the transfer count (Single or Multiple) 2) After getting a CRC status of any block where data transmission is about to be stopped by a STOP_AT_BLK_GAP_REQ. During a write transaction, a BLK_GAP_EVT interrupt is generated when this bit is changed to 0, as a result of the STOP_AT_BLK_GAP_REQ being set. This status is useful for the HD in determining when to issue commands during write busy. <p>1 – transferring data 0 – No valid data</p>
9 (R)	RD_TRAN_ACTIVE	1'b0	<p>This status is used for detecting completion of a read transfer.</p> <p>This bit is set to 1 for either of the following conditions:</p> <ol style="list-style-type: none"> 1) After the end bit of the read command 2) When writing a 1 to CONTINUE_REQ in the SD_HOST_CTRL_0 register to restart a read transfer <p>This bit is cleared to 0 for either of the following conditions:</p> <ol style="list-style-type: none"> 1) When the last data block as specified by block length is transferred to the system. 2) When all valid data blocks have been transferred to the system and no current block transfers are being sent. As a result of the STOP_AT_BLK_GAP_REQ set to 1. A TRAN_END interrupt is generated when this bit changes to 0. <p>1 – Transferring data 0 – No valid data</p>
10 (R)	BUFF_WT_EN	1'b0	<p>This status is used for IO write transfers. This read only flag indicates if space is available for write data. If this bit is 1, data can be written to the buffer. A change of this bit from 1 to 0 occurs when all the block data is written to the buffer. A change of this bit from 0 to 1 occurs when top of block data can be written to the buffer and generates the BUFF_WT_RDY Interrupt.</p> <p>0 – Write Disable 1 – Write Enable</p>
11 (R)	BUFF_RD_EN	1'b0	<p>This status is used for IO read transfers. This read only flag indicates that valid data exists in the host side buffer status. If this bit is 1, readable data exists in the buffer. A change of this bit from 1 to 0 occurs when all the block data is read from the buffer. A change of this bit from 0 to 1 occurs when all the block data is ready in the buffer and generates the BUFF_RD_RDY Interrupt.</p>

			0 – Read Disable 1 – Read Enable.
15:12 (R)			Reserved
16 (R)	CARD_INSERTED	1'b0	This bit indicates whether a card has been inserted. Changing from 0 to 1 generates a CARD_INSERT interrupt in the Normal Interrupt Status register and changing from 1 to 0 generates a CARD_REMOVE Interrupt in the Normal Interrupt Status register. The SOFT_RST_ALL in the <i>SD_HOST_CTRL_1</i> register shall not affect this bit. If a Card is removed while its power is on and its clock is oscillating, the HC shall clear SD_BUS_PWR and SD_CLK_EN in the <i>SD_HOST_CTRL_0</i> register. In addition the HD should clear the HC by the SOFT_RST_ALL . The card detect is active regardless of the SD Bus Power. 0 – Reset or Debouncing or No Card 1 – Card Inserted
17 (R)	CARD_STATE_STABLE	1'b0	This bit is used for testing. If it is 0, the CARD_DETECT_PIN_LEVEL is not stable. If this bit is set to 1, it means the Card Detect Pin Level is stable. The SOFT_RST_ALL in the <i>SD_HOST_CTRL_1</i> Register shall not affect this bit. 0 – Reset of Debouncing 1 – No Card or Inserted
18 (R)	CARD_DETECT_PIN_LEVEL	1'b0	This bit reflects the inverse value of the SD Card Detect pin (SDCD#). 0 – No Card present (SDCD# = 1) 1 – Card present (SDCD# = 0)
19 (R)		1'b0	Reserved
23:20 (R)	DAT_LINE_LEVEL	4'b0	This status is used to check DAT line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DAT<0>. D23 – DAT<3> D22 – DAT<2> D21 – DAT<1> D20 – DAT<0>
24 (R)	CMD_LINE_LEVEL	1'b0	This status is used to check CMD line level to recover from errors, and for debugging.
31:25			Reserved

NOTE: DAT line active indicates whether one of the DAT line is on SD bus is in use.

a) In the case of read transactions this status indicates if a read transfer is executing on the SD bus. Changes of this value from 1 to 0 between data block generates a **BLK_GAP_EVT** interrupt in the *SD_INT_STATUS* register. This bit shall be set in either of the following cases:

- 1) After the end bit of the read command.
- 2) When writing a 1 to **CONTINUE_REQ** in the *SD_HOST_CTRL_0* register to restart a read transfer.

This bit shall be cleared in either of the following cases:

- 1) When the end bit of the last data block is sent from the SD bus to the HC.

2) When beginning a wait read transfer at a stop at the block gap initiated by a **STOP_AT_BLK_GAP_REQ**.

The HC shall wait at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), the HC can wait for current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait in order to use the suspending/resuming function.

b) In the case of write transactions this status indicates that a write transfer is executing on the SD bus. Changing this value from 1 to 0 generate a **TRAN_END** interrupt in the *SD_INT_STATUS* register.

This bit shall be set in either of the following cases:

- 1) After the end of the write command.
- 2) When writing to 1 to **CONTINUE_REQ** in the *SD_HOST_CTRL_0* register to continue a write transfer.

This bit shall be cleared in either of the following cases:

- 1) When the SD card releases write busy of the last data block the HC shall also detect if output is not busy. If SD card does not drive busy signal for 8 SD clocks, the HC shall consider the card drive "Not Busy".
- 2) When the SD card releases write busy prior to waiting for write transfer as a result of a **STOP_AT_BLK_GAP_REQ**.

NOTE: The HD can issue cmd0, cmd12, cmd13 (for memory) and cmd52 (for SDIO) when the DAT lines are busy during data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands shall be issued when Command Inhibit (DAT) is set to zero.

- **SDIO Host Control Register 0 (SD_HOST_CTRL_0) – 0x0028**

Bit	Name	Default	Description
0 (R)		1'b0	Reserved
1 (R)	DAT_TRAN_WIDTH	1'b0	This bit selects the data width of the HC. The HD shall select it to match the data width of the SD card. 1 – 4 bit mode 0 – 1 bit mode
2 (R)	HIGH_SPEED_EN	1'b0	This bit is optional. Before setting this bit, the HD shall check the HIGH_SPEED_SUPPORT in the <i>Capabilities</i> register. If this bit is set to 0 (default), the HC outputs CMD line and DAT lines at the falling edge of the SD clock (up to 25 MHz). If this bit is set to 1, the HC outputs CMD line and DAT lines at the rising edge of the SD clock (up to 50 MHz) 1 – High Speed Mode 0 – Normal Speed Mode
7:3 (R)	-		Reserved
8 (R/W)	SD_BUS_PWR	1'b0	Before setting this bit, the SD host diver shall set SD_BUS_VOL_SEL . If the HC detects the No Card State, this bit shall be cleared. 1 – Power on 0 – Power off
11:9 (R/W)	SD_BUS_VOL_SEL	1'b0	By setting these bits, the HD selects the voltage level for the SD card. Before setting this

			<p>register, the HD shall check the Voltage Support bits in the <i>Capabilities</i> register. If an unsupported voltage is selected, the Host System shall not supply SD bus voltage</p> <p>3'b111 – 3.3 V (Typ.) 3'b110 – 3.0 V (Typ.) 3'b101 – 1.8 V (Typ.) Others – Reserved</p>
15:12 (R)	-		Reserved
16 (R/W)	STOP_AT_BLK_GAP_REQ	1'b0	<p>This bit is used to stop executing a transaction at the next block gap for transfers. Until the transfer complete is set to 1, indicating a transfer completion the HD shall leave this bit set to 1. Clearing both the STOP_AT_BLK_GAP_REQ and CONTINUE_REQ shall not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The HC shall honor STOP_AT_BLK_GAP_REQ for write transfers, but for read transfers it requires that the SD card support Read Wait. Therefore the HD shall not set this bit during read transfers unless the SD card supports Read Wait and has set RD_WAIT_CTRL to 1. In case of write transfers in which the HD writes data to the <i>SD_BUF_DATA</i> register, the HD shall set this bit after all block data is written. If this bit is set to 1, the HD shall not write data to <i>Buffer data port</i> register. This bit affects RD_TRAN_ACTIVE, WT_TRAN_ACTIVE, DAT_LINE_ACTIVE and CMD_INHIBIT (DAT) in the <i>SD_CUR_STATE</i> register.</p> <p>1 – Stop 0 – Transfer</p>
17 (R/W)	CONTINUE_REQ	1'b0	<p>This bit is used to restart a transaction, which was stopped using the STOP_AT_BLK_GAP_REQ. To cancel stop at the block gap, set STOP_AT_BLK_GAP_REQ to 0 and set this bit to restart the transfer. The HC automatically clears this bit in either of the following cases:</p> <p>1) In the case of a read transaction, the DAT_LINE_ACTIVE changes from 0 to 1 as a read transaction restarts. 2) In the case of a write transaction, the WR_TRAN_ACTIVE changes from 0 to 1 as the write transaction restarts. Therefore it is not necessary for Host driver to set this bit to 0. If STOP_AT_BLK_GAP_REQ is set to 1, any write to this bit is ignored.</p> <p>1 – Restart 0 – Ignored</p>
18 (R/W)	RD_WAIT_CTRL	1'b0	The read wait function is optional for SDIO cards. If the card supports read wait, set this bit

			to enable use of the read wait protocol to stop read data using DAT<2> line. Otherwise the HC has to stop the SD clock to hold read data, which restricts commands generation. When the HD detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card. If the card does not support read wait, this bit shall never be set to 1 otherwise DAT line conflict may occur. If this bit is set to 0, Suspend/Resume cannot be supported 1 – Enable Read Wait Control 0 – Disable Read Wait Control
19 (R/W)	INT_AT_BLK_GAP	1'b0	This bit is valid only in 4-bit mode of the SDIO card and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. If the SD card cannot signal an interrupt during a multiple block transfer, this bit should be set to 0. When the HD detects an SD card insertion, it shall set this bit according to the CCCR of the SDIO card.
23:20 (R)	-		Reserved
24 (R/W)	WUEN_ON_CARD_INTERRUPT	1'b0	This bit enables wakeup event via CARD_INT assertion in the <i>SD_INT_STATUS</i> register. This bit can be set to 1 if FN_WUS (Wake Up Support) in CIS is set to 1. 1 – Enable 0 – Disable
25 (R/W)	WUEN_ON_CARD_INSERT	1'b0	This bit enables wakeup event via CARD_INTSERT assertion in the <i>SD_INT_STATUS</i> register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1 – Enable 0 – Disable
26 (R/W)	WUEN_ON_CARD_REMOVE	1'b0	This bit enables wakeup event via CARD_REMOVE assertion in the <i>Normal Interrupt Status</i> register. FN_WUS (Wake up Support) in CIS does not affect this bit. 1 – Enable 0 – Disable
31:27 (R)	-		Reserved

There are three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether the HC issues a Suspend command or the SD card accepts the Suspend command:

- 1) If the HD does not issue **Suspend** command, the **CONTINUE_REQ** shall be used to restart the transfer.
- 2) If the HD issues a **Suspend** command and the SD card accepts it, a **Resume** Command shall be used to restart the transfer.
- 3) If the HD issues a **Suspend** command and the SD card does not accept it, the **CONTINUE_REQ** shall be used to restart the transfer.

Any time **STOP_AT_BLK_GAP_REQ** stops the data transfer, the HD shall wait for **TRAN_END** (in the **SD_INT_STATUS** register) before attempting to restart the transfer. When restarting the data transfer by **CONTINUE_REQ**, the HD shall clear **STOP_AT_BLK_GAP_REQ** before or simultaneously.

NOTE: The Hardware Driver shall maintain voltage on the SD Bus, by setting **SD_BUS_PWR to 1 in this register, when wakeup event via card interrupt is desired.**

- **SDIO Host Control Register 1 (SD_HOST_CTRL_1) – 0x002C**

At the initialization of the HC, the HD shall set the **SDCLK_FREQ_SEL** according to the *Capabilities* register.

Bit	Name	Default	Description
0 (R/W)	INT_CLK_EN	1'b0	This bit is set to 0 when the HD is not using the HC or the HC awaits a wakeup event. The HC should stop its internal clock to go very low power state. Still, registers shall be able to be read and written. Clock starts to oscillate when this bit is set to 1. When clock oscillation is stable, the HC shall set INT_CLK_STABLE in this register to 1. This bit shall not affect card detection. 1 – Oscillate 0 – Stop
1 (R/W)	INT_CLK_STABLE	1'b0	This bit is set to 1 when SD clock is stable after writing to INT_CLK_EN in this register to 1. The SD Host Driver shall wait to set SDCLK_EN until this bit is set to 1. Note: This is useful when using PLL for a clock oscillator that requires setup time. 1 – Ready 0 – Not Ready
2 (R/W)	SDCLK_EN	1'b0	The HC shall stop SDCLK when writing this bit to 0. SDCLK_FREQ_SEL can be changed when this bit is 0. Then, the HC shall maintain the same clock frequency until SDCLK is stopped (Stop at SDCLK = 0). If the HC detects the No Card state, this bit shall be cleared. 1 – Enable 0 – Disable
7:3	-		Reserved
15:8 (R/W)	SDCLK_FREQ_SEL	8'h0	This register is used to select the frequency of the SDCLK pin. The frequency is not programmed directly; rather this register holds the divisor of the BASE_CLK_FREQ for SD clock in the <i>Capabilities</i> register. Only the following settings are allowed. 0x80 – base clock divided by 256 0x40 – base clock divided by 128 0x20 – base clock divided by 64 0x10 – base clock divided by 32 0x08 – base clock divided by 16 0x04 – base clock divided by 8

			<p>0x02 – base clock divided by 4 0x01 – base clock divided by 2 0x00 – base clock (10MHz-63MHz) Setting 0x00 specifies the highest frequency of the SD Clock. When setting multiple bits, the most significant bit is used as the divisor. But multiple bits should not be set. The two default divider values can be calculated by the frequency that is defined by the BASE_CLK_FREQ for SD Clock in the <i>Capabilities</i> register.</p> <p>1) 25 MHz divider value 2) 400 KHz divider value</p> <p>The frequency of the SDCLK is set by the following formula: Clock Frequency = (Base clock) / divisor. Thus choose the smallest possible divisor which results in a clock frequency that is less than or equal to the target frequency.</p>
19:16 (R/W)	DAT_TIMEOUT_VAL	4'h0	<p>This value determines the interval by which DAT line timeouts are detected. Refer to the DAT_TIMEOUT_ERR in the <i>SD_INT_STATUS</i> register for information on factors that dictate timeout generation. Timeout clock frequency will be generated by dividing the base clock TMCLK with this value. When setting this register, prevent inadvertent timeout events by clearing the DAT_TIMEOUT_ERR_EN (in the <i>SD_INT_EN</i> register)</p> <p>1111 – Reserved 1110 – $TMCLK * 2^{27}$... 0001 – $TMCLK * 2^{14}$ 0000 – $TMCLK * 2^{13}$</p>
23:20	-		Reserved
24 (R/W)	SOFT_RST_ALL	1'b0	<p>This reset affects the entire HC except for the card detection circuit. Register bits of type ROC, RW, RW1C, RWAC are cleared to 0. During its initialization, the HD shall set this bit to 1 to reset the HC. The HC shall reset this bit to 0 when capabilities registers are valid and the HD can read them. Additional use of SOFT_RST_ALL may not affect the value of the <i>Capabilities</i> registers. If this bit is set to 1, the SD card shall reset itself and must be reinitialized by the HD.</p> <p>1 – Reset 0 – Work</p>
25 (R/W)	SOFT_RST_CMD	1'b0	<p>Only part of command circuit is reset. The following registers and bits are cleared by this bit:</p> <p><i>SD_CUR_STATE</i> register CMD_INHIBIT (CMD) <i>SD_INT_STATUS</i> register</p>

			CMD_END 1 – Reset 0 – Work
26 (R/W)	SOFT_RST_DAT	1'b0	Only part of data circuit is reset. The following registers and bits are cleared by this bit, <i>SD_BUF_DATA</i> register is cleared and Initialized. <i>SD_CUR_STATE</i> register BUFF_RD_EN BUFF_WT_EN RD_TRAN_ACTIVE WT_TRAN_ACTIVE DAT_LINE_ACTIVE CMD_INHIBIT (DAT) <i>SD_HOST_CTRL_0</i> register CONTINUE_REQ STOP_AT_BLK_GAP_REQ <i>SD_INT_STATUS</i> register BUFF_RD_READY BUFF_WT_READY BLK_GAP_EVT TRAN_END 1 – Reset 0 – Work
31:27	-		Reserved

NOTE: At the initialization of the HC, the HD shall set the **DAT_TIMEOUT_VAL** according to the *Capabilities* register.

NOTE: A reset pulse is generated when writing 1 to each bit of this register. After completing the reset, the HC shall clear each bit. Because it takes some time to complete software reset, the SD Host Driver shall confirm that these bits are 0

- **SDIO Interrupt Status Register (SD_INT_STATUS) – 0x0030**

The interrupt status register include two sections: low word for normal interrupt status and high word for error interrupt status.

The *Interrupt Status Enable* affects read of this register, but *Interrupt Signal Enable* does not affect these reads. An Interrupt is generated when the Interrupt Signal Enable is enabled and at least one of the status bits is set to 1. For all bits except **ERR_INT**, writing 1 to a bit clears it.

Bit	Name	Default	Description
0 (R/W)	CMD_END	1'b0	This bit is set when get the end bit of the command response (Except Auto CMD12). Note: CMD_TIMEOUT_ERR has higher priority than Command Complete. If both are set to 1, it can be considered that the response was not received correctly. 0 – No Command Complete 1 – Command Complete
1 (R/W)	TRAN_END	1'b0	This bit is set when a read / write transaction is completed. <i>Read Transaction:</i> This bit is set at the falling edge of

			<p>RD_TRAN_ACTIVE Status. There are two cases in which the Interrupt is generated. The first is when a data transfer is completed as specified by data length (After the last data has been read to the Host System). The second is when data has stopped at the block gap and completed the data transfer by setting the STOP_AT_BLK_GAP_REQ in the <i>SD_HOST_CTRL_0</i> Register (After valid data has been read to the Host System).</p> <p><i>Write Transaction:</i> This bit is set at the falling edge of the DAT_LINE_ACTIVE Status. There are two cases in which the Interrupt is generated. The first is when the last data is written to the card as specified by data length and Busy signal is released. The second is when data transfers are stopped at the block gap by setting STOP_AT_BLK_GAP_REQ in the <i>SD_HOST_CTRL_0</i> Register and data transfers completed. (After valid data is written to the SD card and the busy signal is released).</p> <p>Note: TRAN_END has higher priority than DAT_TIMEOUT_ERR. If both bits are set to 1, the data transfer can be considered complete</p> <p>0 – No Data Transfer Complete 1 – Data Transfer Complete</p>
2 (R/W)	BLK_GAP_EVT	1'b0	<p>If the STOP_AT_BLK_GAP_REQ in the <i>SD_HOST_CTRL_0</i> Register is set, this bit is set.</p> <p><i>Read Transaction:</i> This bit is set at the falling edge of the DAT_LINE_ACTIVE Status (When the transaction is stopped at SD Bus timing. The Read Wait must be supported in order to use this function).</p> <p><i>Write Transaction:</i> This bit is set at the falling edge of WT_TRAN_ACTIVE Status (After getting CRC status at SD Bus timing).</p> <p>0 – No BLK_GAP_EVT 1 – Transaction stopped at Block Gap</p>
3			Reserved
4 (R/W)	BUFF_WT_RDY	1'b0	<p>This status is set if the BUFF_WT_EN changes from 0 to 1.</p> <p>0 – Not Ready to Write Buffer. 1 – Ready to Write Buffer.</p>
5 (R/W)	BUFF_RD_RDY	1'b0	<p>This status is set if the BUFF_RD_EN changes from 0 to 1. 0 - Not Ready to read Buffer. 1 – Ready to read Buffer.</p>
6 (R/W)	CARD_INSERT	1'b0	<p>This status is set if the CARD_INSERTED in the <i>SD_CUR_STATE</i> registers changes from</p>

			<p>0 to 1. When the HD writes this bit to 1 to clear this status, the status of the CARD_INSERTED in the <i>SD_CUR_STA</i> register should be confirmed. Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated.</p> <p>0 – Card State Stable or Debouncing 1 – Card Inserted</p>
7 (R/W)	CARD_REMOVE	1'b0	<p>This status is set if the CARD_INSERTED in the <i>SD_CUR_STATE</i> register changes from 1 to 0. When the HD writes this bit to 1 to clear this status, the status of the CARD_INSERTED in the <i>SD_CUR_STA</i> register should be confirmed. Because the card detect may possibly be changed when the HD clear this bit an Interrupt event may not be generated.</p> <p>0 – Card State Stable or Debouncing 1 – Card Removed</p>
8 (R/W)	CARD_INT	1'b0	<p>Writing this bit to 1 does not clear this bit. It is cleared by resetting the SD card interrupt factor. In 1-bit mode, the HC shall detect the CARD_INT without SD Clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so there are some sample delays between the interrupt signal from the card and the interrupt to the Host system. When this status has been set and the HD needs to start this interrupt service, CARD_INT_EN in the <i>SD_INT_EN</i> register shall be set to 0 in order to clear the card interrupt statuses latched in the HC and stop driving the Host System. After completion of the card interrupt service (the reset factor in the SD card and the interrupt signal may not be asserted), set CARD_INT_EN to 1 and start sampling the interrupt signal again.</p> <p>0 – No Card Interrupt 1 – Generate Card Interrupt</p>
14:9			Reserved
15 (R/W)	ERR_INT	1'b0	<p>If any of the bits of upper word of <i>SD_INT_STATUS</i> are set and the corresponding status enable bit are set too, then this bit is set. Therefore the SW can test for an error by checking this bit first.</p> <p>0 – No Error. 1 – Error.</p>
16 (R/W)	CMD_TIMEOUT_ERR	1'b0	<p>Occurs only if the no response is returned within 64 SDCLK cycles from the end bit of the command. If the HC detects a CMD line conflict, in which case CMD_CRC_ERR shall also be set. This bit shall be set without waiting for 64 SDCLK cycles because the</p>

			command will be aborted by the HC. 0 – No Error 1 – Timeout
17 (R/W)	CMD_CRC_ERR	1'b0	CMD_CRC_ERR is generated in two cases. 1) If a response is returned and the CMD_TIMEOUT_ERR is set to 0, this bit is set to 1 when detecting a CRC error in the command response 2) The HC detects a CMD line conflict by monitoring the CMD line when a command is issued. If the HC drives the CMD line to 1 level, but detects 0 level on the CMD line at the next SDCLK edge, then the HC shall abort the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error shall also be set to 1 to distinguish CMD line conflict. 0 – No Error 1 – CRC Error Generated
18 (R/W)	CMD_ENDBIT_ERR	1'b0	Occurs when detecting that the end bit of a command response is 0. 0 – No Error 1 – End Bit Error Generated
19 (R/W)	CMD_INDEX_ERR	1'b0	Occurs if a Command Index error occurs in the Command Response. 0 – No Error 1 – Error
20 (R/W)	DAT_TIMEOUT_ERR	1'b0	Occurs when detecting one of following timeout conditions. 1) Busy Timeout for R1b, R5b type. 2) Busy Timeout after Write CRC status 3) Write CRC status Timeout 4) Read Data Timeout 0 – No Error 1 – Timeout
21 (R/W)	DAT_CRC_ERR	1'b0	Occurs when detecting CRC error when transferring read data which uses the DAT line or when detecting the Write CRC Status having a value of other than "010". 0 – No Error 1 – Error
22 (R/W)	DAT_ENDBIT_ERR	1'b0	Occurs when detecting 0 at the end bit position of read data that uses the DAT line or the end bit position of the CRC status. 0 – No Error 1 – Error
23 (R/W)	CURRENT_LIM_ERR	1'b0	By setting the SD_BUS_PWR bit in the SD_HOST_CTRL_0 Register, the HC is requested to supply power for the SD Bus. If the HC supports the Current Limit Function, it can be protected from an Illegal card by stopping power supply to the card in which case this bit indicates a failure status. Reading 1 means the HC is not supplying power to SD card due to some failure. Reading 0 means

			that the HC is supplying power and no error has occurred. This bit shall always set to be 0, if the HC does not support this function. 0 – No Error 1 – Power Fail
24 (R/W)	AUTO_CMD12_ERR	1'b0	Occurs when detecting that one of the bits in <i>AUTO_CMD12_ERR_STATUS</i> register has changed from 0 to 1. This bit is set to 1 also when Auto CMD12 is not executed due to the previous command error. 0 – No Error 1 – Error
27:25	-		Reserved
31:28	VEND_SPEC_ERR		Additional status bits can be defined in this register by the vendor.

Table 43. Relation between Transfer Complete and Data Timeout Error

Transfer Complete	Data Timeout Error	Meaning of the Status
0	0	Interrupted by Another Factor.
0	1	Timeout occur during transfer.
1	Don't care	Data Transfer Complete

Table 44. Relation between Command Complete and Command Timeout Error

Command Complete	Command Timeout Error	Meaning of the Status
0	0	Interrupted by Another Factor.
Don't care	1	Response not received within 64 SDCLK cycles..
1	0	Response Received

Table 45. Relation between Command CRC Error and Command Timeout Error

Command CRC Error	Command Timeout Error	Kinds of Error
0	0	No Error
0	1	Response Timeout Error
1	0	Response CRC Error
1	1	CMD Line Conflict

- SDIO Interrupt Status Enable Register (SD_INT_STATUS_EN) – 0x0034**

Bit	Name	Default	Description
0 (R/W)	CMD_END_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
1 (R/W)	TRAN_END_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
2 (R/W)	BLK_GAP_EVT_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
3			Reserved
4 (R/W)	BUFF_WT_RDY_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
5 (R/W)	BUFF_RD_RDY_INT_STA_EN	1'b0	0 – Masked 1 – Enabled

6 (R/W)	CARD_INSERT_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
7 (R/W)	CARD_REMV_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
8 (R/W)	CARD_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
15:9			Reserved
16 (R/W)	CMD_TIMEOUT_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
17 (R/W)	CMD_CRC_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
18 (R/W)	CMD_ENDBIT_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
19 (R/W)	CMD_INDEX_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
20 (R/W)	DAT_TIMEOUT_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
21 (R/W)	DAT_CRC_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
22 (R/W)	DAT_ENDBIT_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
23 (R/W)	CURRENT_LIM_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
24 (R/W)	AUTO_CMD12_ERR_INT_STA_EN	1'b0	0 – Masked 1 – Enabled
27:25			Reserved
31:28 (R/W)	VEND_SPEC_ERR_INT_STA_EN		0 – Masked 1 – Enabled

NOTE: Setting to 1 enables Interrupt Status.

The HC may sample the card Interrupt signal during interrupt period and may hold its value in the flip-flop. If the Card Interrupt Status Enable is set to 0, the HC shall clear all internal signals regarding Card Interrupt.

To Detect CMD Line conflict, the HD must set both the **CMD_TIMEOUT_ERR_INT_EN** bit and the **CMD_CRC_ERR_INT_EN** bit to 1.

- **SDIO Interrupt Signal Enable Register (SD_INT_SIGNAL_EN) – 0x0034**

Bit	Name	Default	Description
0 (R/W)	CMD_END_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
1 (R/W)	TRAN_END_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
2 (R/W)	BLK_GAP_EVT_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
3			Reserved
4 (R/W)	BUFF_WT_RDY_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
5 (R/W)	BUFF_RD_RDY_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled

6 (R/W)	CARD_INSERT_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
7 (R/W)	CARD_REMV_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
8 (R/W)	CARD_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
15:9			Reserved
16 (R/W)	CMD_TIMEOUT_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
17 (R/W)	CMD_CRC_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
18 (R/W)	CMD_ENDBIT_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
19 (R/W)	CMD_INDEX_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
20 (R/W)	DAT_TIMEOUT_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
21 (R/W)	DAT_CRC_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
22 (R/W)	DAT_ENDBIT_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
23 (R/W)	CURRENT_LIM_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
24 (R/W)	AUTO_CMD12_ERR_INT_SIG_EN	1'h0	0 – Masked 1 – Enabled
27:25			Reserved
31:28 (R/W)	VEND_SPEC_ERR_INT_SIG_EN		0 – Masked 1 – Enabled

This register is used to select which interrupt status is indicated to the Host System as the Interrupt. These status bits all share the sample 1 bit interrupt line. Setting any of these bits to 1'b1 enables interrupt generation.

- **SDIO Auto CMD12 Error Status Register (SD_AUTO_CMD12_ERR_STATUS)**

When *Auto CMD12 Error Status* is set, the HD shall check this register to identify what kind of error Auto CMD12 indicated. This register is valid only when the Auto CMD12 Error is set.

Bit	Name	Default	Description
0 (R)	AUTO_CMD12_NOT_EXEC	1'b0	If memory multiple block data transfer is not started due to command error, this bit is not set because it is not necessary to issue Auto CMD12. Setting this bit to 1 means the HC cannot issue Auto CMD12 to stop memory multiple block transfer due to some error. If this bit is set to 1, other error status bits (D04 – D01) are meaningless. 0 – Executed 1 – Not Executed
1 (R)	AUTO_CMD12_TIME_OUT_ERR	1'b0	Occurs if the no response is returned within 64 SDCLK cycles from the end bit of the command. If this bit is set to 1, the other error status bits (D04 – D02) are meaningless.

			0 – No Error 1 – Timeout
2 (R)	AUTO_CMD12_CRC_ERR	2'h0	Occurs when detecting a CRC error in the command response. 0 – No Error 1 – CRC Error Generated
3 (R)	AUTO_CMD12_ENDBIT_ERR	1'b0	Occurs when detecting that the end bit of command response is 0. 0 – No Error 1 – End Bit Error Generated
4 (R)	AUTO_CMD12_INDEX_ERR	1'b0	Occurs if the Command Index error occurs in response to a command. 0 – No Error 1 – Error
6:5	-		Reserved
7 (R)	CMD_NOT_ISSUED_BY_CMD12_ERR	1'b0	By Auto CMD12 Error Setting this bit to 1 means CMD_WO_DAT is not executed due to an Auto CMD12 error (D04 – D01) in this register. 0 – No Error 1 – Not Issued
31:8	-		Reserved

Table 46. Relation between Auto CMD12 CRC Error and Auto CMD12 Timeout Error

Auto Cmd12 CRC Error	Auto CMD12 Timeout Error	Kinds of Error
0	0	No Error
0	1	Response Timeout Error

Table 47. Relation between Auto CMD12 CRC Error and Auto CMD12 Timeout Error

Auto Cmd12 CRC Error	Auto CMD12 Timeout Error	Kinds of Error
1	0	Response CRC Error
1	1	CMD Line Conflict

The timing of changing Auto CMD12 Error Status can be classified in three scenarios:

1) When the HC is going to issue Auto CMD12
Set bit 0 to 1'b1 if Auto CMD12 cannot be issued due to an error in the previous command.
Set bit 0 to 1'b0 if Auto CMD12 is issued.

2) At the end bit of Auto CMD12 response
Check received responses by checking the error bits 1~4.
Set to 1'b1 if Error is detected.
Set to 1'b0 if Error is Not Detected.

3) Before reading the Auto CMD12 Error Status bit 7
Set bit 7 to 1'b1 if there is a command cannot be issued.
Set bit 7 to 1'b0 if there is no command to issue.

Timing of generating the **Auto CMD12 Error** and writing to the *SD_TRAN_CTRL* register is Asynchronous. Then bit 7 shall be sampled when driver never writing to the *SD_TRAN_CTRL* register. So just before reading the *AUTO_CMD12_ERR_STATUS* register is good timing to set the bit 7 status bit.

- **SDIO Capabilities Register (SD_CAPABILITIES) – 0x0040**

Bit	Name	Default	Description
5:0 (R)	TIMEOUT_CLK_FREQ	6'h30	This bit shows the base clock frequency used to detect DAT_TIMEOUT_ERR . Not 0 – 1Khz to 63Khz or 1Mhz to 63Mhz 000000b – Get Information via another method.
6			Reserved
7 (R)	TIMEOUT_UNIT	1'b1	This bit shows the unit of base clock frequency used to detect DAT_TIMEOUT_ERR . 0 – KHz 1 – MHz
13:8 (R)	BASE_CLK_FREQ	6'h30	This value indicates the base (maximum) clock frequency for the SD clock. Unit values are 1Mhz. If the real frequency is 16.5 MHz, the larger value shall be set 010001b (17 MHz) because the HD uses this value to calculate the clock divider value and it shall not exceed the upper limit of the SD clock frequency. The supported range is 10Mhz to 63 MHz. If these bits are all 0, the Host System has to get information via another method.
15:14			Reserved
17:16 (R)	MAX_BLK_LEN	2'b10	This value indicates the maximum block size that the HD can read and write to the buffer in the HC. The buffer shall transfer this block size without wait cycles. Three sizes can be defined as indicated below. 00 – 512 byte 01 – 1024 byte 10 – 2048 byte 11 – Reserved
20:18			Reserved
21 (R)	HIGH_SPEED_SUP	1'b1	This bit indicates whether the HC and the Host System support High Speed mode and they can supply SD Clock frequency from 25Mhz to 50 MHz. 0 – High Speed Not Supported 1 – High Speed Supported
22			Reserved
23 (R)	SUS_RESUM_SUP	1'b1	This bit indicates whether the HC supports Suspend / Resume functionality. If this bit is 0, the Suspend and Resume mechanism are not supported and the HD shall not issue either Suspend / Resume commands. 0 – Not Supported 1 – Supported
24 (R)	VOL_SUP_33V	1'b1	0 – 3.3 V Not Supported 1 – 3.3 V Supported
25 (R)	VOL_SUP_30V	1'b1	0 – 3.0 V Not Supported 1 – 3.0 V Supported
26 (R)	VOL_SUP_18V	1'b1	0 – 1.8 V Not Supported 1 – 1.8 V Supported
31:27			Reserved

NOTE: The Host System shall support at least one of these voltages above. The HD sets the SD_BUS_VOL_SEL in SD_HOST_CTRL_0 register according to these support bits. If multiple voltages are supported, select the usable lower voltage by comparing the OCR value from the card.

- **SDIO Maximum Current Capabilities Register (SD_MAX_CURR) – 0x0048**

These registers indicate maximum current capability for each voltage. The value is meaningful only if the corresponding Voltage Support bit is set in the *Capabilities* register.

Bit	Name	Default	Description
7:0 (R)	MAX_CUR_33V	8'hff	Maximum Current for 3.3V
15:8 (R)	MAX_CUR_30V	8'hff	Maximum Current for 3.0V
23:16 (R)	MAX_CUR_18V	8'hff	Maximum Current for 1.8V
31:24 (R)	–		Reserved

Table 48. Maximum Current Value Definition

Register Value	Current Value
0	Get Information via another method
1	4mA
2	8mA
3	16mA
...	...
255	1020mA

- **SD Bus Clock Delay Register (SD_CLK_DELAY) – 0x004C**

Bit	Name	Default	Description
5:0 (R/W)	CLK_DELAY	6'h0	The register is used to adjust to the bus clock delay.
31:6 (R/W)		26'h0	Reserved

- **SDIO Slot Interrupt Status and Version Register (SD_SLOT_INT_VER) – 0x00FC**

Bit	Name	Default	Description
7:0 (R)	INT_SIG_FOR_SLOT	8'h0	These status bit indicate the logical OR of Interrupt signal and Wakeup signal for each slot. A maximum of 8 slots can be defined. If one interrupt signal is associated with multiple slots. The HD can know which interrupt is generated by reading these status bits. By a power on reset or by SOFT_RST_ALL , the Interrupt signal shall be disserted and this status shall read 00h. Bit 00 – Slot 1 Bit 01 – Slot 2 Bit 02 – Slot 3 ... Bit 07 – Slot 8
15:8	–		Reserved
23:16 (R/W)	SPEC_VER	8'h0	This Status indicates the Host Controller Spec Version. The Upper and Lower 4 bits indicate the version. 00 – SD Host Specification version 1.0 Others – Reserved

31:24 (R/W)	VENDOR_VER	8'h0	This status is reserved for the vendor version number. The HD should not use this status.
----------------	-------------------	------	---

7.5 USB-OTG Interface

7.5.1 Overview

The USB-OTG interface is specifically designed and optimized for embedded applications. It is particularly suited for self-powered, mobile, small form factor, and high volume peripherals. It provides an intermediate SRAM for data transfer to/from the application area to USB-OTG interface. Below is a functional block diagram of the USB-OTG interface.

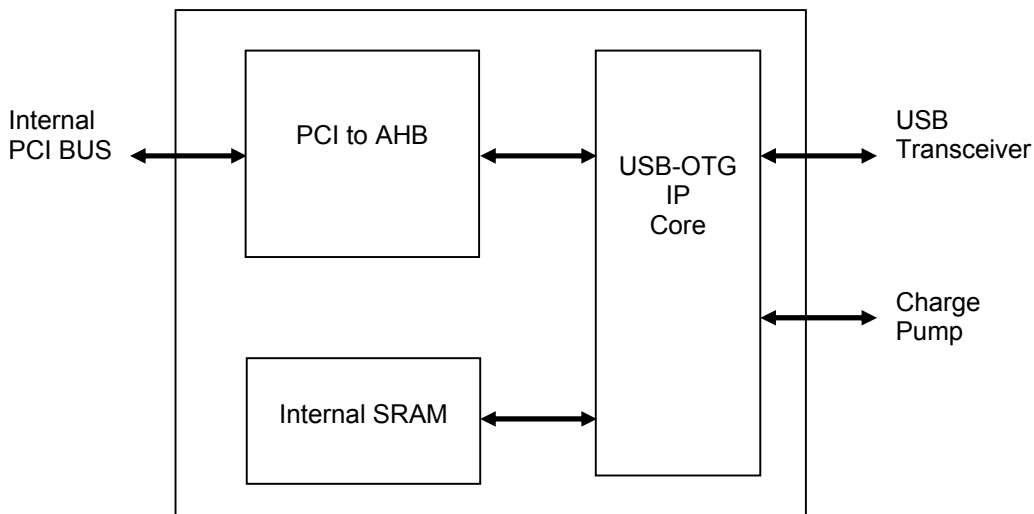


Figure 42. USB-OTG Controller Block Diagram

The USB-OTG interface uses the USB-OTG IP core provided by TransDimension Inc. It is through this IP core that the USB-OTG interface communicates with the actual USB transceiver.

USB-OTG interface has the following features:

Host Controller Features

- USB Specification Revision 2.0 fully compliant USB Host Controller for full-speed (12Mb/s) and low-speed (1.5Mb/s) operations.
- Advanced architecture resulting in easy programming and low microprocessor overhead.
- In hardware ETD data structure management and scheduling to support simultaneous operation of up to 8 active USB pipes.
- True transfer level operation with transaction scheduling and handling (data sequence toggle, error re-try, etc.) carried out in hardware.
- Reliable isochronous transfer with loose timing requirements on the microprocessor.
- Over-current protection on host side. If an over-current condition occurs, it causes a status change and reported to USB system software so that it can take action.

Function Controller Features

- USB Specification Revision 2.0 compliant full speed function controller.
- Software configurable as a standard bus-powered, self-powered, or OTG USB function.
- 8 physical, 4 logical user configurable endpoints, each of which can be programmed in terms of endpoint type, maximum packet size, data buffer association.

USB On-The-Go Features

- Compliant with USB On-The-Go Supplement Specification Revision 1.0.
- HNP/SRP can be implemented in either built-in hardware or user software.

7.5.2 Pin Description

The USB-OTG interface is a 2-wire analog serial interface. The signals required are described in the table below:

Table 49. USB OTG Interface Pin Description

Pin Name	Direction	Description
X_USB_XDN	Bi-directional	USB Pad (Negative)
X_USB_XDP	Bi-directional	USB Pad (Positive)
X_USB_VBUS	Bi-directional	USB Pad (Power)
X_USB_ID	Bi-directional	USB Pad (OTG ID Selector)
X_DRV_VBUS	Output	Drive VBUS enable.
X_USB_PDDM	Bi-directional	USB DM pull-down.
X_USB_PDDP	Bi-directional	USB DP pull-down.
X_USB_PUDP	Bi-directional	USB DP pull-up.

7.5.3 Functional Description

7.5.3.1 PCI to AHB

The USB-OTG core supports an AHB interface as well as a general bus interface. This interface goes on top of the original bus interface and conforms to the AHB protocol as defined in the AMBA specification. The USB-OTG core contains both an AHB Master and Slave, a DMA Controller, and Core, ETD, EP, and data memory interfaces that connect directly to the USB-OTG core. To connect the USB-OTG core to internal PCI bus, the PCI to AHB conversion is needed. It converts the PCI target signal to the AHB slave interface, connects the request of AHB master to the PCI arbiter and converts the AHB master signal to the PCI master.

7.5.3.2 Internal SRAM

The USB-OTG interface has three different memories, 32-DWORD EP memory, 32-DWORD ETD memory and 128-DWORD DATA memory. All the three memories are implemented by Single Port SRAM.

Only the Host Controller uses the ETD memory and only the Function Controller uses the EP memory. Both Host Controller and Function Controller share the DATA memory.

The ETD memory can contain up to 8 ETDs, each of them comprised by 4 DWORDs. Software has full Read and Write access to the ETD memory. All the unused fields in the ETD structure should be initialized to 0.

The EP memory can contain up to 4 logical endpoints, 8 physical endpoints, each of them comprised by 4 DWORDs. Each logical endpoint can be IN, OUT or IN/OUT, each physical endpoint can be either IN or OUT. Software has full Read and Write access to the EP memory. All the unused fields in the EP structure should be initialized to 0.

7.5.3.3 USB-OTG Transceiver

The transceiver (TSMC: TPD013G3) is separated as three major blocks, voltage detector,

charge/discharge and USB1.1 OTG core. Voltage detector checks the voltage status of VBUS and represents result to output signals (VBUSVALID / SESSEND / AVALID / BVALID). Charge/discharge block provide VBUS pulse for SRP. Three switch signals (DP_PULL_UP / DP_PULL_DOWN / DM_PULL_DOWN) control DP/DM and external resistor connection. Major transmission functions are determined by RX, TX and suspend signals. IDDIG and IDPULLUP decide the status of ID. DRVVBUSC controls off-chip charge pump device. For the detail of the USB-OTG transceiver, please refer to the related TSMC document.

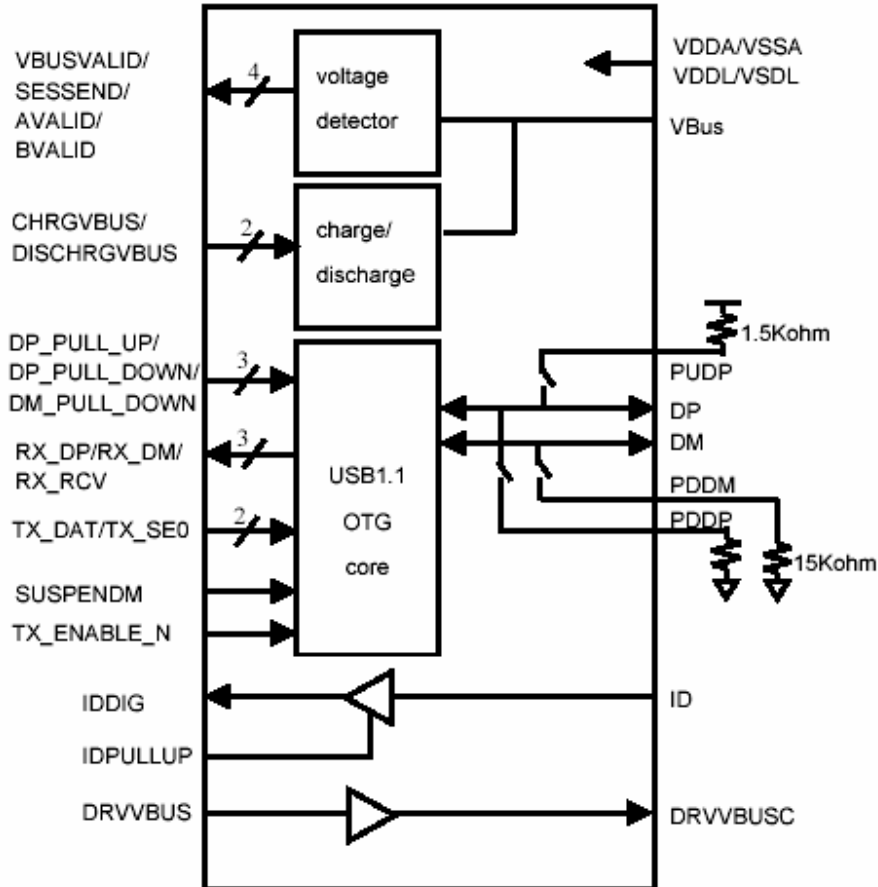


Figure 43. Block Diagram of USB-OTG transceiver

7.5.4 USB-OTG Interface Registers

Table 50. USB-OTG Interface Register Mapping

RISC Address <15:0>	Register	Description
0x0000	USBOTG_TL_HW_MODE	Hardware Mode Register
0x0004	USBOTG_TL_INT_STATUS	Core Interrupt Status Register
0x0008	USBOTG_TL_INT_EN	Core Interrupt Enable Register
0x000C	USBOTG_TL_CLK_CTRL	Clock Control Register
0x0010	USBOTG_TL_RST_CTRL	Reset Control Register
0x0014	USBOTG_TL_FRM_INTERVAL	Frame Interval Register
0x0018	USBOTG_TL_FRM_REMAINING	Frame Remaining Register
0x001C	USBOTG_HNP_CTRL	HNP Control Status Register

0x0020	USBOTG_HNP_TIMER1	HNP Timer1 Register
0x0024	USBOTG_HNP_TIMER2	HNP Timer2 Register
0x0028	USBOTG_HNP_PULSE	HNP Timer3 Pulse Control Register
0x002C	USBOTG_HNP_INT_STATUS	HNP Interrupt Status Register
0x0030	USBOTG_HNP_INT_EN	HNP Interrupt Enable Register
0x0034~0x003C	-	Reserved.
0x0040	USBOTG_FC_CMD_STATUS	Function Command Status Register
0x0044	USBOTG_FC_DEV_ADDR	Device Address Register
0x0048	USBOTG_FC_SYS_INT_STATUS	System Interrupt Status Register
0x004C	USBOTG_FC_SYS_INT_EN	System Interrupt Enable Register
0x0050	USBOTG_FC_X_INT_STATUS	X Buffer Interrupt Status Register
0x0054	USBOTG_FC_Y_INT_STATUS	Y Buffer Interrupt Status Register
0x0058	USBOTG_FC_XY_INT_EN	X/Y Interrupt Enable Register
0x005C	USBOTG_FC_X_STATUS	X Buffer Filled Status Register
0x0060	USBOTG_FC_Y_STATUS	Y Buffer Filled Status Register
0x0064	USBOTG_FC_EP_EN	Endpoint Enable Register
0x0068	USBOTG_FC_EP_READY	Endpoint Ready Register
0x006C	USBOTG_FC_IMM_INT	Immediate Interrupt Register
0x0070	USBOTG_FC_DONE_STATUS	Endpoint Done Status Register
0x0074	USBOTG_FC_DONE_EN	Endpoints Done Enable Register
0x0078	USBOTG_FC_TOGGLE_BITS	Endpoint Toggle Bits Register
0x007C	USBOTG_FC_FRM_NUM	Frame Number Register
0x0080	USBOTG_HC_CONFIG	Host Controller Configuration Register
0x0084	-	Reserved
0x0088	USBOTG_HC_SYS_INT_STATUS	System Interrupt Status Register
0x008C	USBOTG_HC_SYS_INT_EN	System Interrupt Enable Register
0x0090~0x0094	-	Reserved
0x0098	USBOTG_HC_X_INT_STATUS	X Buffer Interrupt Status Register
0x009C	USBOTG_HC_Y_INT_STATUS	Y Buffer Interrupt Status Register
0x00A0	USBOTG_HC_XY_INT_EN	X/Y Interrupt Enables Register
0x00A4	-	Reserved.
0x00A8	USBOTG_HC_X_STATUS	X Buffer Filled Status Register
0x00AC	USBOTG_HC_Y_STATUS	Y Buffer Filled Status Register
0x00B0-0x00BC	-	Reserved
0x00C0	USBOTG_HC_ETD_SET	ETD Enable Set Register
0x00C4	USBOTG_HC_ETD_CLEAR	ETD Enable Clear Register
0x00C8	-	Reserved
0x00CC	USBOTG_HC_IMM_INT	Immediate Interrupt Register
0x00D0	USBOTG_HC_DONE_STATUS	Endpoint Done Status Register
0x00D4	USBOTG_HC_DONE_EN	ETD Done Enable Register
0x00D8-0x00DC	-	Reserved
0x00E0	USBOTG_HC_FRM_NUM	Frame Number Register
0x00E4	USBOTG_HC_LS_THRESHOLD	Low Speed Threshold Register
0x00E8	USBOTG_HC_RH_DESC_A	Root Hub Descriptor A Register
0x00EC	USBOTG_HC_RH_DESC_B	Root Hub Descriptor B Register
0x00F0	USBOTG_HC_RH_STATUS	Root Hub Status Register
0x00F4	USBOTG_HC_PORT1_STATUS	Port 1 Status Register
0x00F8	-	Reserved

0x00FC	-	Reserved
0x0100~0x01FC	-	Reserved
0x0200~0x027C	Host Endpoint Transfer Descriptor	ETD buffer address space
0x0280~0x03FC	-	Reserved
0x0400~0x047C	Function Endpoint Descriptor	EP buffer address space
0x0480~0x07FC	-	Reserved
0x0800	USBOTG_DMA_REV	DMA Revision Register
0x0804	USBOTG_DMA_INT_STATUS	DMA Interrupt Status Register
0x0808	USBOTG_DMA_INT_EN	DMA Interrupt Enable Register
0x080C	USBOTG_DMA_ETD_ERR	ETD DMA Error Status Register
0x0810	USBOTG_DMA_EP_ERR	EP DMA Error Status Register
0x0814~0x081C	-	Reserved
0x0820	USBOTG_DMA_ETD_EN	ETD DMA Enable Register
0x0824	USBOTG_DMA_EP_EN	EP DMA Enable Register
0x0828	USBOTG_DMA_ETD_TRIGX_EN	ETD DMA Enable X Trigger Request Register
0x082C	USBOTG_DMA_EP_TRIGX_EN	EP DMA Enable X Trigger Request Register
0x0830	USBOTG_DMA_ETD_TRIGXY_EN	ETD DMA Enable XY Trigger Request Register
0x0834	USBOTG_DMA_EP_TRIGXY_EN	EP DMA Enable XY Trigger Request Register
0x0838	USBOTG_DMA_ETD_BURST4_EN	ETD DMA Burst4 Enable Register
0x083C	USBOTG_DMA_EP_BURST4_EN	EP DMA Burst4 Enable Register
0x0840	USBOTG_DMA_MISC	Misc. Control Register
0x0844	-	Reserved.
0x0848	USBOTG_DMA_ETD_CLR	ETD DMA Channel Clear Register
0x084C	USBOTG_DMA_EP_CLR	EP DMA Channel Clear Register
0x0850~0x08C	-	Reserved
0x0900	USBOTG_ETD0_START_ADDR	ETD0 System Memory Start Address Register
0x0904	USBOTG_ETD1_START_ADDR	ETD1 System Memory Start Address Register
0x0908	USBOTG_ETD2_START_ADDR	ETD2 System Memory Start Address Register
0x090C	USBOTG_ETD3_START_ADDR	ETD3 System Memory Start Address Register
0x0910	USBOTG_ETD4_START_ADDR	ETD4 System Memory Start Address Register
0x0914	USBOTG_ETD5_START_ADDR	ETD5 System Memory Start Address Register
0x0918	USBOTG_ETD6_START_ADDR	ETD6 System Memory Start Address Register
0x091C	USBOTG_ETD7_START_ADDR	ETD7 System Memory Start Address Register
0x920~0x97C	-	Reserved
0x0980	USBOTG_EP0_OUT_START_ADDR	EP0 OUT System Memory Start Address Register
0x0984	USBOTG_EP0_IN_START_ADDR	EP0 IN System Memory Start Address Register
0x0988	USBOTG_EP1_OUT_START_ADDR	EP1 OUT System Memory Start

		Address Register
0x098C	USBOTG_EP1_IN_START_ADDR	EP1 IN System Memory Start Address Register
0x0990	USBOTG_EP2_OUT_START_ADDR	EP2 OUT System Memory Start Address Register
0x0994	USBOTG_EP2_IN_START_ADDR	EP2 IN System Memory Start Address Register
0x0998	USBOTG_EP3_OUT_START_ADDR	EP3 OUT System Memory Start Address Register
0x099C	USBOTG_EP3_IN_START_ADDR	EP3 IN System Memory Start Address Register
0x9A0~0x9FC	-	Reserved
0x0A00	USBOTG_ETD0_BUF_PTR	ETD0 DMA Buffer Xfer Ptr Register
0x0A04	USBOTG_ETD1_BUF_PTR	ETD1 DMA Buffer Xfer Ptr Register
0x0A08	USBOTG_ETD2_BUF_PTR	ETD2 DMA Buffer Xfer Ptr Register
0x0A0C	USBOTG_ETD3_BUF_PTR	ETD3 DMA Buffer Xfer Ptr Register
0x0A10	USBOTG_ETD4_BUF_PTR	ETD4 DMA Buffer Xfer Ptr Register
0x0A14	USBOTG_ETD5_BUF_PTR	ETD5 DMA Buffer Xfer Ptr Register
0x0A18	USBOTG_ETD6_BUF_PTR	ETD6 DMA Buffer Xfer Ptr Register
0x0A1C	USBOTG_ETD7_BUF_PTR	ETD7 DMA Buffer Xfer Ptr Register
0xA20~0xA7C	-	Reserved
0x0A80	USBOTG_EP0_OUT_BUF_PTR	EP0 OUT DMA Buffer Xfer Ptr Register
0x0A84	USBOTG_EP0_IN_BUF_PTR	EP0 IN DMA Buffer Xfer Ptr Register
0x0A88	USBOTG_EP1_OUT_BUF_PTR	EP1 OUT DMA Buffer Xfer Ptr Register
0x0A8C	USBOTG_EP1_IN_BUF_PTR	EP1 IN DMA Buffer Xfer Ptr Register
0x0A90	USBOTG_EP2_OUT_BUF_PTR	EP2 OUT DMA Buffer Xfer Ptr Register
0x0A94	USBOTG_EP2_IN_BUF_PTR	EP2 IN DMA Buffer Xfer Ptr Register
0x0A98	USBOTG_EP3_OUT_BUF_PTR	EP3 OUT DMA Buffer Xfer Ptr Register
0x0A9c	USBOTG_EP3_IN_BUF_PTR	EP3 IN DMA Buffer Xfer Ptr Register
0x1000~0x11FC	-	DATA buffer address space

- **USB-OTG Hardware Mode Register (USBOTG_TL_HW_MODE) – 0x0000**

Bit	Name	Default	Description
1:0 (R/W)	CRECFG	2'b11	Core Configuration 00 – Hardware HNP; 01 – Host; 10 – Function; 11 – software HNP
2	BEMDE	1'b0	Big Endian Mode 1 – Accesses to the data memory in big endian
3	-	1'b0	Reserved
5:4	HOSTXCVR	2'b10	Host Transceiver Properties. 00 – TX differential, RX differential 10 – TX single-ended, RX differential 01 – TX differential, RX single-ended 11 – TX singled-ended, RX singled-ended
7:6 (R)	OTGXCVR	2'b10	OTG Transceiver Properties. Same as host transceiver

13:8	-	6'b0	Reserved
14 (R/W)	ANASDBEN	1'b0	OTG Analog Signal Short Debounce Enable. 1 – enable the OTG VBUS and ID signal short debounce logic
15 (R/W)	TSTMDE	1'b0	Test Mode.
23:16 (R)	HSTREV	8'h20	Host Revision.
31:24 (R)	FUNCREV	8'h20	Function Revision.

- **USB-OTG Core Interrupt Status Register (USBOTG_TL_INT_STATUS) – 0x0004**

Bit	Name	Default	Description
0 (R)	HCINT	1'b0	Host Interrupt To clear this bit, all interrupt source flags from HostInterruptStatus register must be cleared or disabled
1 (R)	FCINT	1'b0	Function Interrupt To clear this bit, all interrupt source flags from FunctionInterruptStatus register must be cleared or disabled
2 (R)	HNPINT	1'b0	HNP Interrupt To clear this bit, all interrupt source flags from HnpInterruptStatus register must be cleared or disabled
3 (R)	ASHCINT	1'b0	Asynchronous Host Interrupt To clear this bit, the HostClockEnable bit must be set in the Clock Control Register
4 (R)	ASFCINT	1'b0	Asynchronous Function Interrupt To clear this bit, the FunctionClockEnable bit must be set in the Clock Control Register
5 (R)	ASHNPINT	1'b0	Asynchronous HNP Interrupt To clear this bit, the MainClockEnable bit must be set in the Clock Control Register
31:6	-	26'h0	Reserved

- **USB-OTG Core Interrupt Enable Register (USBOTG_TL_INT_EN) – 0x0008**

Bit	Name	Default	Description
0 (R/W)	HCINT_EN	1'b0	Host Interrupt Enable
1 (R/W)	FCINT_EN	1'b0	Function Interrupt Enable
2 (R/W)	HNPINT_EN	1'b0	HNP Interrupt Enable
3 (R/W)	ASHCINT_EN	1'b0	Asynchronous Host Interrupt Enable
4 (R/W)	ASFCINT_EN	1'b0	Asynchronous Function Interrupt Enable
5 (R/W)	ASHNPINT_EN	1'b0	Asynchronous HNP Interrupt Enable
31:6	-	26'h0	Reserved

- **USB-OTG Clock Control Register (USBOTG_TL_CLK_CTRL) – 0x000C**

Bit	Name	Default	Description
0 (R/W)	MAINCLK	1'b0	Main Clock Enable

1 (R/W)	HSTCLK	1'b0	Host Clock Enable
2 (R/W)	FUNCCLK	1'b0	Function Clock Enable
31:3	-	29'h0	Reserved

- **USB-OTG Reset Control Register (USBOTG_TL_RST_CTRL) – 0x0010**

This register allows for independent reset of the block of the core. When set the bit will reset all of the registers of the block to their default condition. Each bit is self-clearing after the reset is complete.

Bit	Name	Default	Description
0 (W)	RSTHC	1'b0	Reset Host Control Logic
1 (W)	RSTHSIE	1'b0	Reset Host SIE
2 (W)	RSTRH	1'b0	Reset Root Hub
3 (W)	RSTFSIE	1'b0	Reset Function SIE
4 (W)	RSTFC	1'b0	Reset Function Logic
5 (W)	RSTCTRL	1'b0	Reset Control Logic
31:6	-	26'h0	Reserved

- **USB-OTG Frame Interval Register (USBOTG_TL_FRM_INTERVAL) – 0x0014**

Bit	Name	Default	Description
13:0 (R/W)	FRMINTERVAL	14'h2ED F	Frame Interval The value written here sets the width of a frame. This value is in the number of full-speed bit times. The nominal value is set to be 11,999.
14	-	1'b0	Reserved
15 (W)	RSTFRM	1'b0	Reset Frame When set the frame remaining register will be cleared and be set to the value in the FRMINTERVAL
29:16 (R/W)	FRMINPER	14'h2A2 F	Frame Interval Periodic The value written here sets the amount of time in a frame that periodic packets can be sent. This value is in the number of full-speed bit times
31:30	-	2'h0	Reserved

- **USB-OTG Frame Remaining Register (USBOTG_TL_FRM_REMAINING) – 0x0018**

Bit	Name	Default	Description
15:0	-	16'h0	Reserved
29:16(R)	FRMREMN	14'h0	Frame Remaining This number represents the number of full-speed bit times still remaining in the current frame. This counter is decremented at each bit time. When it reaches zero, loading the FRMINTERVAL value specified in FRMREMN at the next bit time boundary resets it. When entering the UsbOperational state, the host controller re-loads the content with the FRMINTERVAL and uses the updated value from the next SOF.

31:30	-	2'h0	Reserved
-------	---	------	----------

- **USB-OTG HNP Control Status Register (USBOTG_HNP_CTRL L) – 0x001C**

Bit	Name	Default	Description
0	-	1'b0	Reserved
1 (R/W)	ABBUSREQ	1'b0	A B Bus Request 1 – The software is requesting to be A_MASTER or B_MASTER.
2 (R/W)	ADROPBUS	1'b0	A Drop VBus 1 – The software of the A device wants to power down the VBUS
3 (W)	CLRERROR	1'b0	HNP Clear Error State As an A_DEVICE, to clear the error when the A-DEVICE HNP is in A_VBUS_ERROR state and does not want to release the bus. As A B-DEVICE, to force the transition from B_SLAVE to B_IDLE.
8:4 (R)	HNPSTATE	5'h0	HNP State Represents the current state of the hardware HNP state machine. Only valid in hardware HNP.
9 (R/W)	SWPDDM	1'b1	Software Pull Down DM
10	-	-	Reserved
11 (R/W)	SWPUDP	1'b0	Software Pull Up DP
12 (R/W)	SWAUTORST	1'b0	Software Automatic Reset
14:13	-	-	Reserved
15 (R/W)	SWVBUSPUL	1'b0	Software VBUS Pulse
16	-	-	Reserved
17 (R)	ISADEV	1'b0	A Device
18 (R)	ISBDEV	1'b0	B Device
19 (R/W)	CMPEN	1'b0	Charge pump comparator Enable
20 (R/W)	BGEN	1'b0	Charge pump band Gap Enable
21 (R/W)	MASTER	1'b0	HNP Master State
22 (R/W)	SLAVE	1'b0	HNP Slave State
23	-	-	Reserved
24 (R/W)	BHNPEN	1'b0	B HNP Enabled
25 (R/W)	ARMTHNPEN	1'b0	A Remote HNP Enabled
26	-	-	Reserved
27 (R)	VBUSGTAVV	1'b0	VBus Greater Than A VBus Valid
28 (R)	VBUSABSV	1'b0	VBus AB Session Valid
29 (R)	VBUSGTBSE	1'b0	VBus Greater Than B Session End
30 (R)	HNPDAT	1'b0	HNP Data Toggle
31	-	-	Reserved

- **USB-OTG HNP Timer1 Register (USBOTG_HNP_TIMER1) – 0x0020**

Bit	Name	Default	Description
7:0 (R/W)	AWAITVRISE	8'h64	A Wait VRise Timer This timer is used by an A-DEVICE in the A_WAIT_VRISE state to wait for VBUS to rise

			to valid level If VBUS does not reach a valid level before a timeout, then the A-DEVICE knows that the B-DEVICE is drawing too much current.
15:8 (R/W)	AWAITCONN	8'h7D	A Wait Connect Timer This timer is used by an A-DEVICE in the A_WAIT_BCON state, to wait for the B-DEVICE to signal a connection. If the B-DEVICE does not connect before a timeout, then the A-DEVICE stops waiting for a connection. If the InsertionMode bit in HNP Timer3 Register is not set the timeout will cause the AwaitBConnectTimeOutInterrupt . Each unit is equivalent to 8 msec. The default is 0x7d which is equivalent to 1 second.
23:16 (R/W)	AWAITDISC	8'hC8	A Wait Disconnect Timer This timer is started by an A-DEVICE when it enters the A_SUSPEND state. If the A-DEVICE does not detect a downstream disconnect before a timeout, then the A-DEVICE can end the session.
31:24 (R/W)	BWAITCONN	8'h4	B Wait Connect Timer This timer is used by a B-DEVICE in the B_WAIT_CONN_A state, to wait for an A-DEVICE to signal a connection. If the A-DEVICE does not connect before a timeout (b_wait_conn_tmout), then the B-DEVICE stops waiting for a connection.

NOTE: The time unit is 1ms

- **USB-OTG HNP Timer2 Register (USBOTG_HNP_TIMER2) – 0x0024**

Bit	Name	Default	Description
4:0 (R/W)	SRPDULW	5'hA	SRP Data Bus Pulse Width Defines the session request protocol data line pulse length generated by BDEVICE.
9:5 (R/W)	SRVPULW	5'h8	SRP V Bus Pulse Width Defines the Session Request Protocol VBUS pulse length generated by BDEVICE.
15:10		6'h0	
31:16 (R/W)	BSRPFAL	16'h138 8	B SRP Fail Timer This timer is set by software to determine when a Session Request has failed. The conditions of a failure are the VBUS must be over the b_session_vaild and the A_MASTER must drive a reset to the device.

- **USB-OTG HNP Timer3 Pulse Control Register (USBOTG_HNP_PULSE) – 0x0028**

Bit	Name	Default	Description
2:0	-	8'h0	
3(R/W)	USESESEND	1'b0	Use Session End Signal This bit should be set when a Session End

			<p>signal is available. The HNP protocol allows two methods to detect when Vbus is fully discharged (below 0.8V). One method is to use a comparator whose output is called B_SESS_END. The other is to discharge the Vbus through a resistor to GND for a certain period of time.</p> <p>1: B_SESS_END signal is available and it will be used to control the HNP state machine to discharge Vbus.</p> <p>0: No B_SESS_END signal is available. The Vbus Discharge Timer Value will be used to determine the discharge time for Vbus. During discharge, the Vbus Power Down signal will be asserted which should discharge the Vbus to GND through a resistor.</p>
4(R/W)	VBUSPULSE	1'b0	<p>Use VBus Pulse SRP Detection</p> <p>Use VBUS pulsing to detect session request by the B-DEVICE.</p>
5(R/W)	DATAPULSE	1'b0	<p>Use Data Pulse SRP Detection</p> <p>Use data line pulsing to detect session request by the B-DEVICE.</p>
6(R/W)	INSERTION	1'b0	<p>Insertion Mode Set</p> <p>When SET the HNP logic in Hardware HNP will disable the a_wait_bcon_tmout to allow for operation as a legacy host.</p>
15:7(R/W)	VBUSDISCTIMER	9'hA	<p>VBus Discharge Timer Value</p> <p>This value will be used in setting the waiting period that the Vbus will be discharged in order to get it below 0.8V. This will be used in Hardware HNP mode when the B_SESS_END signal is not available such as when an OTG transceiver is being used. If a B_SESS_END signal is available, this timer will not be used and control of the Vbus discharge will be dependent on the signal.</p> <p>In this case, the user should set the USE_B_SESS_END bit in this register when this signal is available. Each unit of time is 1 msec. The default value is 0xA which is equivalent to a discharge time of 10 msec.</p>
23:16(R/W)	A_SDB_WIN	8'h64	<p>B Connect Short Debounce Window</p> <p>This value is the maximum time the Hardware HNP should wait to do short de-bouncing of the connection. Once this timer expires it then will use the LongDebounceTimer to detect the connection.</p>
31:24(R/W)	LNGDBNC	8'h64	<p>Long Debounce Timer</p> <p>This timer is the de-bounce time following the detection of a transition on the USB bus. This timer is used only after the BConnectShortDebounceWindow has expired or following the transition from the A_WAIT_VRISE state.</p>

- **USB-OTG HNP Interrupt Status Register (USBOTG_HNP_INT_STATUS) – 0x002C**

Bit	Name	Default	Description
0 (R)	IDCHANGE	1'b0	ID change Interrupt Indicates the state of the ID pin has changed. Software should read the HnpControlStatus Register to determine the current status of the ID pin.
1 (R)	MASSLVCHG	1'b0	Master Slave Change Interrupt This interrupt is only valid in Hardware HNP . This bit is asserted when HnpState enters A_SLAVE, A_MASTER, to inform the application so that appropriate action can be taken by software.
2 (R)	VBUSVALID	1'b0	A VBUS Valid Change Interrupt This interrupt is only valid in Software HNP and Function Host Mode . When asserted this means that the VBUS has crossed the a_vbus_vld level. Software should read the HnpControlStatus Register to determine the current status of the VBUS.
3 (R)	ABSESVAILD	1'b0	AB Session Valid Change Interrupt This interrupt is only valid in Software HNP and Function Host Mode . When asserted this means that the VBUS has crossed the b_session_vld threshold. Software should read the HnpControlStatus Register to determine the current status of the VBUS.
4 (R)	VBUSERROR	1'b0	VBUS Error Interrupt This interrupt is only valid in Hardware HNP . A-DEVICE VBUS error, due to over current or battery problem.
5 (R)	SRPINT	1'b0	Session Request Detect Interrupt After receiving this interrupt software must decide if it wants to respond to the Session Request. This interrupt is asserted when either a VBUS pulse is detected and/or a Data pulse is detected.
6(R)	SRPSUCFAIL	1'b0	SRP Success Fail Interrupt This interrupt is only valid in Hardware HNP . B-DEVICE needs two conditions to know that its session request is successful. 1. The state must be in B_SLAVE 2. It must detect that A-DEVICE is resetting it within the BSrpFailTimer value.
7(R)	AIDLEBDTO	1'b0	A Idle B Disconnect Time Out Interrupt This timer is only used in Hardware HNP . This interrupt is asserted when the AWaitDisconnectTimer expires. This timer is started in the A_SUSPEND state and when it expires it will cause the transition to A_WAIT_VFALL.
8(R)	AWAITBTO	1'b0	A Wait B Connect Time Out Interrupt

			This timer is only used in Hardware HNP . When the AWaitConnectTimer expires this interrupt is asserted. The timer is started when in the A_WAIT_BCONN state and when it times out the state transitions to the A_WAIT_VFALL state. This interrupt is only asserted if the InsertionModeSet is cleared.
31:9	-	-	Reserved

- **USB-OTG HNP Interrupt Enable Register (USBOTG_HNP_INT_EN) – 0x0030**

Bit	Name	Default	Description
0 (R/W)	IDCHANGE_EN	1'b0	ID change Interrupt Enable
1 (R/W)	MASSLVCHG_EN	1'b0	Master Slave Change Interrupt Enable
2 (R/W)	VBUSVALID_EN	1'b0	A VBUS Valid Change Interrupt Enable
3 (R/W)	ABSESVAILD_EN	1'b0	AB Session Valid Change Interrupt Enable
4 (R/W)	VBUSEROR_EN	1'b0	VBUS Error Interrupt Enable
5 (R/W)	SRPINT_EN	1'b0	Session Request Detect Interrupt Enable
6(R/W)	SRPSUCFAIL_EN	1'b0	SRP Success Fail Interrupt Enable
7(R/W)	AIDLEBDTO_EN	1'b0	A Idle B Disconnect Time Out Interrupt Enable
8(R/W)	AWAITBTO_EN	1'b0	A Wait B Connect Time Out Interrupt Enable
31:9	-	26'h0	Reserved

- **USB-OTG Function Command Status Register (USBOTG_FC_CMD_STATUS) – 0x0040**

Bit	Name	Default	Description
0 (R)	RESETDET	1'b1	USB Bus Reset Detected
1 (R/W)	RSMINPROG	1'b0	Resume In Progress When Software reads this bit to be 1, it indicates that the USB bus and the Function controller are in the RESUME state. A 0 indicates that the USB bus and the Function controller are not in the RESUME state. Writing a 0 leaves the hardware unchanged. Writing a 1 generates a RESUME signal to the upstream port and lets the xcvr exit the power-saving mode. Writing a 1 is a RESUME command. It will not affect the value of the physical register bit which merely displays the current status.
2 (R/W)	SUSPDET	1'b0	Suspend Detected When Software reads this bit to be 1, it indicates that the USB bus is in the SUSPEND state. But it does not mean the function controller is in the SUSPEND state. A 0 indicates that the USB bus is not in the suspend state. But it does not mean the function controller is not in the suspend state. Writing a 0 leaves the hardware unchanged. Writing a 1 sets the Function port in the SUSPEND state and sets the transceiver in the power-savings mode. It will not affect the value of the physical register bit 2 which

			merely reflects the status. Note: If SuspendDetected=1, it does not mean the Function controller is in the SUSPEND state. It just means that the USB bus is in the suspend state. Software needs to write 1 to set the Function Controller in the suspend state. But the Function controller does not keep this state in a register.
3 (R/W)	BADISOAP	1'b0	Bad ISO Accepted 1 – function accept an ISO packet with corrupted data. 0 – bad ISO packt are automatically dropped.
6:4	-	-	Reserved
7(W)	SOFTRESET	1'b0	Software Reset Function Controller
31:8	-	26'h0	Reserved

- **USB-OTG Device Address Register (USBOTG_FC_DEV_ADDR) – 0x0044**

Bit	Name	Default	Description
6:0(R/W)	DEVADDR	7'h0	Device Address
31:7	-	25'h0	Reserved

- **USB-OTG System Interrupt Status Register (USBOTG_FC_SYS_INT_STATUS) – 0x0048**

Bit	Name	Default	Description
0 (R)	RESETINT	1'b0	Reset Detected Interrupt
1 (R)	RSMFININT	1'b0	Resume Finished Interrupt
2 (R)	SUSPDETINT	1'b0	Suspend Detected Interrupt
3 (R)	DONEREGINT	1'b0	Done Register Interrupt When asserted this bit indicates one or more bits in EndpointDoneStatus Register (0x0070) is set.
4(R)	SOFDETINT	1'b0	SOF Detected Interrupt
31:5	-	27'h0	Reserved

- **USB-OTG System Interrupt Enable Register (USBOTG_FC_SYS_INT_EN) – 0x004C**

Bit	Name	Default	Description
0 (R/W)	RESETINT_EN	1'b0	Reset Detected Interrupt Enable
1 (R/W)	RSMFININT_EN	1'b0	Resume Finished Interrupt Enable
2 (R/W)	SUSPDETINT_EN	1'b0	Suspend Detected Interrupt Enable
3 (R/W)	DONEREGINT_EN	1'b0	Done Register Interrupt Enable
4(R/W)	SOFDETINT_EN	1'b0	SOF Detected Interrupt Enable
31:5	-	27'h0	Reserved

- **USB-OTG X Buffer Interrupt Status Register (USBOTG_FC_X_INT_STATUS) – 0x0050**

Bit	Name	Default	Description
ODD (R)	XBUFFERnININT	1'b0	X Buffer of Endpoint <n> IN Interrupt When asserted indicates that the XIN buffer of

			endpoint <n> has been emptied by the host.
EVEN (R)	XBUFFERnOUTINT	1'b0	X Buffer of Endpoint <n> OUT Interrupt When asserted indicates that the XOUT buffer of endpoint <n> has been filled by the host.

- **USB-OTG Y Buffer Interrupt Status Register (USBOTG_FC_Y_INT_STATUS) – 0x0054**

Bit	Name	Default	Description
ODD (R)	YBUFFERnININT	1'b0	Y Buffer of Endpoint <n> IN Interrupt When asserted indicates that the YIN buffer of endpoint <n> has been emptied by the host.
EVEN (R)	YBUFFERnOUTINT	1'b0	Y Buffer of Endpoint <n> OUT Interrupt When asserted indicates that the YOUT buffer of endpoint <n> has been filled by the host.

- **USB-OTG XY Interrupt Enable Register (USBOTG_FC_XY_INT_EN) – 0x0058**

Bit	Name	Default	Description
ODD (R/W)	XYnININT_EN	1'b0	XY Buffer of Endpoint <n> IN Interrupt Enable
EVEN (R/W)	XYnOUTINT_EN	1'b0	XY Buffer of Endpoint <n> OUT Interrupt Enable

- **USB-OTG X Buffer Filled Status Register (USBOTG_FC_X_STATUS) – 0x005C**

Bit	Name	Default	Description
ODD (R/W)	XFILLnIN	1'b0	XIN Filled of Endpoint <n> When a bit is asserted, it indicates to the Hardware that the <n> X Buffer has been filled and the function controller can begin sending the data on the next IN. This register is a toggle register. If it is set, writing a '1' will clear it. If it is cleared, writing a '1' will set it. Writing a '0' has no effect.
EVEN (R/W)	XFILLnOUT	1'b0	XOUT Filled of Endpoint <n> When a bit is cleared, it indicates to the Hardware that the <n> X Buffer has been drained and the function controller can begin receiving data from the next OUT. This register is a toggle register. If it is set, writing a '1' will clear it. If it is cleared, writing a '1' will set it. Writing a '0' has no effect.

- **USB-OTG Y Buffer Filled Status Register (USBOTG_FC_Y_STATUS) – 0x0060**

Bit	Name	Default	Description
ODD (R)	YFILLnIN	1'b0	YIN Filled of Endpoint <n>
EVEN (R)	YFILLnOUT	1'b0	YOUT Filled of Endpoint <n>

- **Endpoint Enable Register (USBOTG_FC_EP_EN) – 0x0064**

Bit	Name	Default	Description
-----	------	---------	-------------

ODD (R/W)	EPnIN_EN	1'b0	Endpoint n IN Enabled When SET indicates that endpoint <n> is ready to respond to the host on the next IN token. If there is not any data to be sent, the Function Controller will respond with a NAK.
EVEN (R/W)	EPnOUT_EN	1'b0	Endpoint n OUT Enabled When SET indicates that endpoint <n> is ready to respond to the host on the next OUT token. If the Function Controller is unable to receive any data because the X and/or Y Buffers are full, it will respond with a NAK.

- **USB-OTG Endpoint Ready Set Register (USBOTG_FC_EP_READY) – 0x0068**

Bit	Name	Default	Description
ODD (R/W)	EPnIN_READY	1'b0	Endpoint n IN Ready This register is directly written. Writing a '1' to it will set it and a '0' will not affect it. In order to clear this register, a 1 must be written to the corresponding bit in the Frame Number Register. Software must write to this bit to tell the Core whether or not the Endpoint is ready to be transferred. If this bit is 0, and the EP is enabled, then the function controller will return NAKs. If this bit is 1, then the transfer will commence. This bit is ignored if the EP is not enabled.
EVEN (R/W)	EPnOUT_READY	1'b0	Endpoint n OUT Ready This register is directly written. Writing a '1' to it will set it and a '0' will not affect it. In order to clear this register, a 1 must be written to the corresponding bit in the Frame Number Register. Software must write to this bit to tell the Core whether or not the Endpoint is ready to be transferred. If this bit is 0, and the EP is enabled, then the function controller will return NAKs. If this bit is 1, then the transfer will commence. This bit is ignored if the EP is not enabled.

- **USB-OTG Immediate Interrupt Register (USBOTG_FC_IMM_INT) – 0x006C**

Bit	Name	Default	Description
ODD (R/W)	IMnININT	1'b0	Immediate n IN Interrupt When SET the Function controller will assert an interrupt immediately instead of waiting until the SOF.
EVEN (R/W)	IMnOUTINT	1'b0	Immediate n OUT Interrupt When SET the Function controller will assert an interrupt immediately instead of waiting until the SOF.

- **USB-OTG Endpoint Done Status Register (USBOTG_FC_DONE_STATUS) – 0x0070**

Bit	Name	Default	Description
ODD (R)	EPnIN_DONE	1'b0	Endpoint n IN Done Status When asserted indicates that endpoint <n> IN has transferred all of the data currently set in the Function Endpoint Descriptor.
EVEN (R)	EPnOUT_DONE	1'b0	Endpoint n OUT Done Status When asserted indicates that endpoint <n> OUT has received all of the data currently set in the Function Endpoint Descriptor or the endpoint has received larger than the MaxPacketSize for this endpoint.

- **USB-OTG Endpoint Done Enable Register (USBOTG_FC_DONE_EN) – 0x0074**

Bit	Name	Default	Description
ODD (R/W)	EPnINDN_EN	1'b0	Endpoint n IN Done Enable When SET allows endpoint <n> IN direction to generate done interrupts. The Endpoint<n>InDoneEnable must be SET for both immediate and SOF marker interrupts.
EVEN (R/W)	EPnOUTDN_EN	1'b0	Endpoint n OUT Done Enable When SET allows endpoint <n> IN direction to generate done interrupts. The Endpoint<n>OutDoneEnable must be SET for both immediate and SOF marker interrupts.

- **USB-OTG Endpoint Done Toggle Bits Register (USBOTG_FC_TOGGLE_BITS) – 0x0078**

Bit	Name	Default	Description
ODD (R/W)	EPnINTOG	1'b0	Endpoint n IN Toggle Bit This register is a toggle register. Writing a '1' will toggle the value and writing a '0' will leave it unchanged. The expected value of the current data PID is determined by the hardware from the last PID from the host for that endpoint. Software should change this bit while the EP is currently full or disabled if they wish to change the next data toggle bit.
EVEN (R/W)	EPnOUTTOG	1'b0	Endpoint n IN Toggle Bit This register is a toggle register. Writing a '1' will toggle the value and writing a '0' will leave it unchanged the value of the next data PID to be sent by hardware is determined by this bit. This bit is updated by Hardware after each successful transmission. Software should change this bit while the EP is currently empty or disabled if they wish to change the next data toggle bit.

- **USB-OTG Frame Number and Endpoint Ready Clear Register (USBOTG_FC_FRM_NUM) – 0x007C**

This register has different definitions when it's read or written.

Bit	Name	Default	Description
10:0 (R)	FRMNUMB	11'h0	Last Frame Number Hardware will set this value based upon the FrameNumber contained in the last received SOF packet.
31:11	-	21'h0	

Bit	Name	Default	Description
31:0 (W)	EPRDYCLEAR	32'h0	Endpoint Ready Clear Writing a '1' to this register will clear the corresponding Endpoint Ready bit but will not affect the Frame Number. This is a workaround that had to be implemented due to a register contention condition that exists in the Core. Since address space was not available, the Frame Number register was multiplexed with this functionality.

- **USB-OTG Host Control Register (USBOTG_HC_CONFIG) – 0x0080**

Bit	Name	Default	Description
1:0 (R/W)	CTLBLKSR	2'h0	Control Bulk Service Ratio This setting dictates the number of CONTROL packets sent out for each BULK packet. 00 – 1:1; 01 – 2:1; 10 – 4:1; 11 – 8:1
3:2 (R/W)	HCUSBSTE	2'h0	Host Controller USB State Software uses these bits to control the state of the USB state machine. 00 – USB reset; 01 – USB resume; 10 – USB operational; 11 – USB suspend
4 (R/W)	RMTWUEN	1'b0	Remote Wake Up Enable
15:5	-	-	Reserved
17:16 (R)	SCHEDOVR	2'h0	Schedule Overrun Count This count is incremented each time a Scheduler Overrun event is detected in the Host Controller.
30:18	-	-	Reserved
31 (W)	HCRRESET	1'b0	Host Controller Reset

- **USB-OTG System Interrupt Status Register (USBOTG_HC_SYS_INT_STATUS) – 0x0088**

Bit	Name	Default	Description
0 (R)	SORINT	1'b0	Schedule Overrun Interrupt
1 (R)	DONEINT	1'b0	Done Register Interrupt When asserted, this indicates that there are completed ETDs on the ETDDoneStatus Register.
2 (R)	SOFINT	1'b0	SOF Interrupt
3 (R)	RESDETINT	1'b0	Resume Detected Interrupt
4 (R)	HERRINT	1'b0	Host Error Interrupt
5 (R)	FMOFINT	1'b0	Frame Number Over Flow Interrupt

6 (R)	PSCINT	1'b0	Port Status Change Interrupt
31:7	-	-	Reserved

- **USB-OTG System Interrupt Enable Register (USBOTG_HC_SYS_INT_EN) – 0x008C**

Bit	Name	Default	Description
0 (R/W)	SORINTEN	1'b0	Schedule Overrun Interrupt Enable
1 (R/W)	DONEINTEN	1'b0	Done Register Interrupt Enable
2 (R/W)	SOFINTEN	1'b0	SOF Interrupt Enable
3 (R/W)	RESDETINTEN	1'b0	Resume Detected Interrupt Enable
4 (R/W)	HERRINTEN	1'b0	Host Error Interrupt Enable
5 (R/W)	FMOFINTEN	1'b0	Frame Number Over Flow Interrupt Enable
6 (R/W)	PSCINTEN	1'b0	Port Status Change Interrupt Enable
31:7	-	-	Reserved

- **USB-OTG X Buffer Interrupt Status Register (USBOTG_HC_X_INT_STATUS) – 0x0098**

Bit	Name	Default	Description
31:0 (R)	XBUFFERnINT	32'h0	X Buffer <n> Interrupt When asserted indicates that the X buffer of ETD <n> requires servicing. For the OUT case, the buffer has been emptied by the host and requires more data to continue the transfer. For the IN case, the buffer has been filled by the host and requires emptying before it can continue. Writing the asserted bit back to the register clears this bit.

- **USB-OTG Y Buffer Interrupt Status Register (USBOTG_HC_Y_INT_STATUS) – 0x009C**

Bit	Name	Default	Description
31:0 (R)	YBUFFERnINT	32'h0	Y Buffer <n> Interrupt When asserted indicates that the Y buffer of ETD <n> requires servicing. For the OUT case, the buffer has been emptied by the host and requires more data to continue the transfer. For the IN case, the buffer has been filled by the host and requires emptying before it can continue. Writing the asserted bit back to the register clears this bit.

- **USB-OTG XY Interrupt Enable Register (USBOTG_HC_XY_INT_EN) – 0x00A0**

Bit	Name	Default	Description
31:0 (R/W)	XYnINT_EN	32'h0	XY Buffer <n> Interrupt Enable

- **USB-OTG X Buffer Filled Status Register (USBOTG_HC_X_STATUS) – 0x00A8**

Bit	Name	Default	Description
31:0 (R)	XFILLn	32'h0	X Filled <n> Status When SET, this indicates that the

			corresponding X Buffer is Full.
--	--	--	---------------------------------

- **USB-OTG Y Buffer Filled Status Register (USBOTG_HC_Y_STATUS) – 0x00AC**

Bit	Name	Default	Description
31:0 (R)	YFILLn	32'h0	Y Filled <n> Status When SET, this indicates that the corresponding Y Buffer is Full.

- **USB-OTG ETD Enable Set Register (USBOTG_HC_ETD_SET) – 0x00C0**

Bit	Name	Default	Description
31:0 (R/W)	ETDnSET_EN	32'h0	ETD <n> Set When SET indicates that an ETD is enabled and ready to be processed by the host.

- **USB-OTG ETD Enable Clear Register (USBOTG_HC_ETD_CLEAR) – 0x00C4**

Bit	Name	Default	Description
31:0 (W)	ETDnCLR	32'h0	ETD <n> Clear When SET indicates that ETD <n> is disabled. This register cannot be read.

- **USB-OTG Immediate Interrupt Register (USBOTG_HC_IMM_INT) – 0x00CC**

Bit	Name	Default	Description
31:0 (R/W)	IMnINT	32'h0	Immediate <n> Interrupt When SET the Host Controller will assert an interrupt immediately upon retirement of an ETD instead of waiting until the SOF.

- **USB-OTG ETD Done Status Register (USBOTG_HC_DONE_STATUS) – 0x00D0**

Bit	Name	Default	Description
31:0 (R)	ETDnDONE	32'h0	ETD <n> Done Status When asserted indicates that ETD<n>DoneStatus has been retired, either through normal operation or through an error condition. The CompletionCode field of the ETD should be read to determine the reason for retirement. A complete list of completion codes is found in ETD completion code descriptions.

- **USB-OTG Endpoint Done Enable Register (USBOTG_HC_DONE_EN) – 0x00D4**

Bit	Name	Default	Description
31:0 (R/W)	EPnDN_EN	32'h0	ETD <n> Done Enable

- **USB-OTG Frame Number Register (USBOTG_HC_FRM_NUM) – 0x00E0**

Bit	Name	Default	Description
-----	------	---------	-------------

15:0 (R)	FRMNUMB	16'h0	Frame Number This contains the current frame number for the Host Controller. This number is placed in the SOF packet used to signal the beginning of the new frame.
31:16	-	-	Reserved

- **USB-OTG Low Speed Threshold Register (USBOTG_HC_LS_THRESHOLD) – 0x00E4**

Bit	Name	Default	Description
10:0 (R/W)	LSTHRESH	11'h628	Low Speed Threshold This is the number in USB full speed bit times that are required to be remaining in a frame to allow a low speed packet to be transmitted. All low speed packets have a MaxPacketSize maximum of 8 bytes, so this should be set to the number of bit times needed to transmit an 8 byte packet. The default value of 628h should not be changed for normal operation.
31:11	-	-	Reserved

- **USB-OTG Root Hub Descriptor A Register (USBOTG_HC_RH_DESC_A) – 0x00E8**

Bit	Name	Default	Description
7:0 (R)	NDNSTMPRT	8'h30	Number of Downstream Ports These bits specify the number of downstream ports supported by the Root Hub.
8 (R)	NOPWRSWT	1'b1	No Power Switching All ports are power switched. This bit will always read as SET.
9 (R)	PWRSWTMD	1'b0	Power Switching Mode This bit is used to specify how the power switching of the Root Hub ports is controlled. In this implementation, each port is powered individually. This allows for power to be controlled on a per-port basis. The port responds only to port power commands (Set/ClearPortPower).
10 (R)	DEVTYPE	1'b0	Device Type This bit specifies that the Root Hub is not a compound device. The Root Hub is not permitted to be a compound device. This field should always read as 0.
11 (R)	OVRCURPM	1'b0	Over Current Protection Mode This bit describes how the over current status for the Root Hub ports are reported. This implementation uses gang-power and gang-over current reporting.
12(R)	NOOVRCURP	1'b0	No Over Current Protection This bit describes how the over current status for the Root Hub is reported. In this implementation, the over current status is reported collectively for all downstream ports.
23:13	-	-	Reserved

31:24 (R)	PWRTOGOOD	8'h1	Power On to Power Good Time This byte specifies the duration that software must wait before accessing a powered-on port of the Root Hub.
--------------	------------------	------	---

- **USB-OTG Root Hub Descriptor B Register (USBOTG_HC_RH_DESC_B) – 0x00EC**

Bit	Name	Default	Description
7:0 (R)	DEVREMOVE	8'h0	Device Removable Each bit is dedicated to a port of the Root Hub. When cleared, the attached device is removable. When set, the attached device is not removable. bit 0: this port can be OTG/ #1 port (during HNP negotiation) bit 1: Device attached to Port #2 bit 2: Device attached to Port #3
15:8	-	-	Reserved
23:16 (R)	PRTPWRCM	8'h7	Port Power Control Mask Each bit indicates if a port is affected by a global power control command when PowerSwitchingMode is set. When set, the port's power state is only affected by per-port power control (Set/ClearPortPower). When cleared, the port is controlled by the global power switch (Set/ClearGlobalPower). bit 0: this port can be OTG/ #1 port (during HNP negotiation) bit 1: Port #2 bit 2: Port #3
31:24	-	-	Reserved

- **USB-OTG Root Hub Status Register (USBOTG_HC_RH_STATUS) – 0x00F0**

Bit	Name	Default	Description
0	-	-	Reserved
1 (R)	OVRCURI	1'b0	Over Current Indicator This bit reports over current conditions when the global reporting is implemented. When set, an over current condition exists. When cleared, all power operations are normal. If per-port over current protection is implemented this bit is always '0'
14:2	-	-	Reserved
15 (R/W)	DEVCONWUE	1'b0	Device Connect Wakeup Enable This bit enables a ConnectStatusChange bit as a resume event, causing a USBsuspend to USBResume state transition and setting the ResumeDetected interrupt. 0 = ConnectStatusChange is not a remote wakeup event. 1 = ConnectStatusChange is a remote wakeup event.

			Writing a '1' sets DeviceRemoveWakeupEnable . Writing a '0' has no effect.
16	-	-	Reserved
17 (R)	OVRCURCHG	1'b0	Over Current Indicator Change This bit is set by hardware when a change has occurred to the OCI field of this register. The HCD clears this bit by writing a '1'. Writing a '0' has no effect.
30:18	-	-	Reserved
31 (W)	CLRMTWUE	1'b0	Clear Remote Wakeup Enable

- **USB-OTG Port Status1 Register (USBOTG_HC_PORT1_STATUS) – 0x00F4**

Bit	Name	Default	Description
0 (R/W)	CURCONST	1'b0	Current Connect Status (read) CurrentConnectStatus This bit reflects the current state of the downstream port. A read with this bit SET indicates a device is connected. (write) ClearPortEnable The HCD writes a '1' to this bit to clear the PortEnableStatus bit. Writing a '0' has no effect. The CurrentConnectStatus is not affected by any write.
1 (R/W)	PRTENABST	1'b0	Port Enable Status (read) PortEnableStatus When SET this bit indicates whether the port is enabled. The Root Hub will clear this bit when an over current condition, disconnect event, the power is switched-off, or operational bus error has occurred. A change in this bit causes PortEnabledStatusChange to be SET. HCD sets this bit by writing SetPortEnable and clears it by writing ClearPortEnable . This bit cannot be set when CurrentConnectStatus is cleared. This bit is also set, if not already, at the completion of a port reset when ResetStatusChange is set or port suspend when SuspendStatusChange is set. 0 = port is disabled 1 = port is enabled (write) SetPortEnable The HCD sets PortEnableStatus by writing a '1'. Writing a '0' has no effect. If CurrentConnectStatus is cleared, this write does not set PortEnableStatus , but instead sets ConnectStatusChange . This informs the driver that it attempted to enable a disconnected port.
2 (R/W)	PRTSUSPST	1'b0	Port Suspend Status (read) PortSuspendStatus When SET bit indicates the port is suspended or in the resume sequence. It is SET by a

			<p>SetSuspendState write and at the end of the resume interval. This bit cannot be set if CurrentConnectStatus is cleared. This bit is also cleared when PortResetStatusChange is set at the end of the port reset or when the Host Controller is placed in the USBRESUME state.</p> <p>(write) SetPortSuspend The HCD sets the PortSuspendStatus bit by writing a '1' to this bit. Writing a '0' has no effect. If CurrentConnectStatus is cleared, this write does not set PortSuspendStatus; instead it sets ConnectStatusChange. This informs the driver that it attempted to suspend a disconnected port.</p>
3 (R/W)	PRTOVRCURI	1'b0	<p>Port Over Current Indicator (read) PortOverCurrentIndicator This bit is only valid when the Root Hub is configured so that over-current conditions are reported on a per-port basis. If per-port over current reporting is not supported, this bit is always read as '0'. If cleared, all power operations are normal for this port. If SET, an overcurrent condition exists on this port.</p> <p>(write) ClearSuspendStatus The HCD writes a '1' to initiate a resume. Writing a '0' has no effect. A resume is initiated only if PortSuspendStatus is SET.</p>
4 (R/W)	PRTRSTST	1'b0	<p>Port Reset Status (read) PortResetStatus When this bit is SET by a write to SetPortReset, and will remain asserted until the port reset signaling is completed. When reset is completed, this bit is cleared and PortResetStatusChange is SET. This bit cannot be set if CurrentConnectStatus is cleared.</p> <p>(write) SetPortReset The HCD sets the port reset signaling by writing a '1' to this bit. Writing a '0' has no effect. If CurrentConnectStatus is cleared, this write does not set PortResetStatus, but instead sets ConnectStatusChange. This informs the driver that it attempted to reset a disconnected port.</p>
7:5	-	-	Reserved
8 (R/W)	PRTPOWERST	1'b0	<p>Port Power Status (read) PortPowerStatus This bit reflects the port's power status. This bit is cleared if an over current condition is detected. HCD sets this bit by writing SetPortPower. HCD clears this bit by writing ClearPortPower. When port power is disabled, CurrentConnectStatus,</p>

			PortEnableStatus , PortSuspendStatus , and PortResetStatus should be reset. (write) SetPortPower The HCD writes a '1' to set the PortPowerStatus bit. Writing a '0' has no effect.
9 (R/W)	LSDEVCON	1'b0	Low Speed Device Attached (read) LowSpeedDeviceAttached This bit indicates the speed of the device attached to this port. When SET, a low speed device is attached to this port. When CLEARED, a full speed device is attached to this port. This field is valid only when the CurrentConnectStatus is set. (write) ClearPortPower The HCD clears the PortPowerStatus bit by a write with this bit SET. Writing with this bit CLEARED has no effect.
15:10	-	-	Reserved
16 (R)	CONNECTSC	1'b0	Connect Status Change When a connect or disconnect event occurs this bit will be SET. A write with this bit SET will clear this bit. Writing a '0' has no effect. If CurrentConnectStatus is cleared when a SetPortReset , SetPortEnable , or SetPortSuspend write occurs, this bit is SET to force the driver to re-evaluate the connection status since these writes should not occur if the port is disconnected. Only a write back with this bit SET will clear this bit.
17 (R)	PRTENBLSC	1'b0	Port Enable Status Change This bit is set when hardware events cause the PortEnableStatus bit to be cleared. Only a write back with this bit SET will clear this bit.
18 (R)	PRTSTSTSC	1'b0	Port Suspend Status Change This bit is set when the resume sequence has been fully completed. A write back with this bit SET or when ResetStatusChange is SET when will clear this bit.
19 (R)	OVRCURIC	1'b0	Port Over Current Indicator Change This bit is SET when an PortOverCurrentIndicator has changed on this port and is only valid if over current conditions are reported on a per-port basis. Only a write back with this bit SET will clear this bit.
20 (R)	PRTRSTSC	1'b0	Port Reset Status Change This bit is SET at the end of the 50-ms port reset signal. Only a write back with this bit SET will clear this bit.
31:21	-	-	Reserved

- **USB-OTG DMA Revision Register (USBOTG_DMA_REV) – 0x0800**

Bit	Name	Default	Description
7:0 (R)	REVCODE	8'h0	Revision Code
31:8	-	-	Reserved

- **USB-OTG DMA Interrupt Status Register (USBOTG_DMA_INT_STATUS) – 0x0804**

Bit	Name	Default	Description
0 (R)	ETDERR	1'b0	ETD Error Interrupt ETD DMA transfer aborted on error (read 0x080c to find out which ETD client). The Error status will be cleared by hardware.
1(R)	EPERR	1'b0	Endpoint Error Interrupt EP DMA transfer aborted on error (read 0x0810 to find out which EP client). The Error status will be cleared by hardware.
31:2	-	-	Reserved

- **USB-OTG DMA Interrupt Enable Register (USBOTG_DMA_INT_EN) – 0x0808**

Bit	Name	Default	Description
0 (R/W)	ETDERR_EN	1'b0	ETD Error Interrupt Enable
1 (R/W)	EPERR_EN	1'b0	Endpoint Error Interrupt Enable
31:2	-	-	Reserved

- **USB-OTG ETD DMA Error Status Register (USBOTG_DMA_ETD_ERR) – 0x080C**

Bit	Name	Default	Description
31:0(R/W)	ETDDMAERST	32'h0	ETD DMA Error Status

- **USB-OTG Endpoint DMA Error Status Register (USBOTG_DMA_EP_ERR) – 0x0810**

Bit	Name	Default	Description
31:0 (R/W)	EPDMAERST	32'h0	Endpoint Error Status

- **USB-OTG ETD DMA Enable Register (USBOTG_DMA_ETD_EN) – 0x0820**

Bit	Name	Default	Description
31:0 (R/W)	ETDDMAEN	32'h0	ETD DMA Enable Setting the ETDDmaEnable bit will enable DMA transfer on the corresponding ETD. Will be auto-disabled at the end of the transfer. Software can clear this bit by using EtdDmaChannelClear Register.

- **USB-OTG Endpoint DMA Enable Register (USBOTG_DMA_EP_EN) – 0x0824**

Bit	Name	Default	Description
-----	------	---------	-------------

31:0 (R/W)	EPDMAEN	32'h0	Endpoint DMA Enable Setting the EndpointDMAEnable bit will enable DMA transfer on the corresponding EP. Will be auto-disabled at the end of the transfer. Software can clear this bit by using EpDmaChannelClear Register.
---------------	----------------	-------	--

- **USB-OTG ETD DMA Enable X Triger Request Register (USBOTG_DMA_ETD_TRIGX_EN) – 0x0828**

Bit	Name	Default	Description
31:0 (W)	ETDDMAXTEN	32'h0	ETD DMA X Trigger Enable Writing a 1 sets the corresponding ETDDMAEnable bit and causes an external XTrigger for the client.

- **USB-OTG Endpoint DMA Enable X Triger Request Register (USBOTG_DMA_EP_TRIGX_EN) – 0x082C**

Bit	Name	Default	Description
31:0 (W)	EPDMAXTEN	32'h0	Endpoint DMA X Trigger Enable Writing a 1 sets the corresponding EndpointDMAEnable bit and causes an external XTrigger for the client

- **USB-OTG ETD DMA Enable XY Triger Request Register (USBOTG_DMA_ETD_TRIGXY_EN) – 0x0830**

Bit	Name	Default	Description
31:0 (W)	ETDDMAXYTEN	32'h0	ETD DMA X and Y Trigger Enable Writing 1 sets the corresponding ETDDMAEnable and causes both XTrigger and YTrigger for the client

- **USB-OTG Endpoint DMA Enable XY Triger Request Register (USBOTG_DMA_EP_TRIGXY_EN) – 0x0834**

Bit	Name	Default	Description
31:0 (W)	EPDMAXYTEN	32'h0	Endpoint DMA X and Y Trigger Enable Writing a 1 sets the corresponding EndpointDMAEnable bit and causes both XTrigger and YTrigger for the client

- **USB-OTG ETD DMA Burst4 Enable Register (USBOTG_DMA_ETD_BURST4_EN) – 0x0838**

This Register is set by default for all clients. The DMA will transfer words on the AHB using the INCR4 transfer size (burst of 4 DWORDS). If for some reason a source/destination of the DMA transfer cannot handle bursts of 4 DWORDS, the corresponding bit should be cleared to transfer one word at a time.

Bit	Name	Default	Description
31:0 (R/W)	ETDDMABST4EN	32'hFFF FFFFFF	ETD DMA Burst 4 Enable

- **USB-OTG Endpoint DMA Burst4 Enable Register (USBOTG_DMA_EP_BURST4_EN) – 0x083C**

This is set by default for all clients. The DMA will transfer words on the AHB using the INCR4 transfer size (burst of 4 DWORDS). If for some reason a source/destination of the DMA transfer cannot handle bursts of 4 DWORDS, the corresponding bit should be cleared to transfer one word at a time.

Bit	Name	Default	Description
31:0 (R/W)	EPDMABST4EN	32'hFFF FFFF	Endpoint DMA Burst 4 Enable

- **USB-OTG Miscellaneous Control Register (USBOTG_DMA_MISC) – 0x0840**

Bit	Name	Default	Description
0 (R/W)	FILTCC	1'b0	Filter on Completion Code If set, the DMA controller will check the completion code of the transfer. The data will only be transferred if the completion code shows no error.
1 (R/W)	ARBMODE	1'b0	Arbiter Mode Select 1: Puts the DMA arbiter in Round Robin mode. The arbitration for the DMA servicing will proceed in a round robin fashion. 0: Puts the DMA arbiter in Priority Access mode. The DMA enabled transfers are serviced in a prioritized fashion in order of decreasing ETD/EP number (Zero gets highest priority).
2 (R/W)	SKPRTRY	1'b0	Skip on Retry Mode If set, will skip current DMA transfer if an AHB RETRY response is encountered, and will start next pending transfer. The transfer will be retried after the rest of the pending transfers have been completed.
3 (R/W)	ISOPRECFRM	1'b0	ISO OUT Previous Frame Mode If set, the DMA will only service an ISO OUT ETD one frame before the transfer will start. The frame number where the transfer will occur can be found in the StartingFrame field of the Host Isochronous ETD descriptor format.
31:4	-	-	Reserved

- **USB-OTG ETD DMA Channel Clear Register (USBOTG_DMA_ETD_CLR) – 0x0848**

Bit	Name	Default	Description
31:0 (W)	ETDDMACHNCLR	32'h0	ETD DMA Channel Clear Will clear the DMA channel corresponding to the bit that is set. Will also clear the DMA enable bit for the ETD. This register is a Write-Clear register. Software will always read 0 on this register.

- **USB-OTG DMA Channel Clear Register (USBOTG_DMA_EP_CLR) – 0x084C**

Bit	Name	Default	Description
31:0 (W)	EPDMACHNCLR	32'h0	Endpoint DMA Channel Clear Will clear the DMA channel corresponding to the bit that is set. Will also clear the DMA enable bit for the EP. This register is a Write-Clear register. Software will always read 0 on this register.

- **USB-OTG ETD <n> System Memory Start Address Register (USBOTG_ETD0_START_ADDR) – 0x0900~0x091C**

Before Enabling DMA for a client ETD, the corresponding register must be loaded with the **ETD System Memory Start Address** where the data will be stored. The **ETD System Memory Start Address** is byte address that must be DWORD aligned.

ETD0~ETD7 Registers have same definition.

Bit	Name	Default	Description
31:0 (R/W)	ETDSMSA	32'h0	ETD <n> System Memory Start Address Starting address in system memory where DMA will put/fetch data to for ETD<n>.

- **USB-OTG EP <n> System Memory Start Address Register (USBOTG_EP0_OUT_START_ADDR) – 0x0980~0x099C**

Before enabling the DMA for a client EP, the corresponding register must be loaded with the system memory address where the data will be stored. The **Endpoint System Memory Start Address** is byte address that must be DWORD aligned.

EP0~EP3 OUT/IN Registers have same definition.

Bit	Name	Default	Description
31:0 (R/W)	EPSMSA	32'h0	Endpoint <n> OUT or IN System Memory Start Address Starting address in system memory where DMA will put/fetch data to for EP n.

- **USB-OTG ETD <n> DMA Buffer Transfer Pointer Register (USBOTG_ETD<n>_BUF_PTR) – 0x0A00~0x0A1C**

If the DMA is paused due to the encounter of an AHB RETRY response and the SKIPONRETRY feature is enabled, the current index into the buffer is stored in these registers so that the DMA knows where to re-start the transfer. The buffer transfer pointers are accessible only for debug purposes.

ETD0~ETD7 Registers have same definition.

Bit	Name	Default	Description
31:0 (R/W)	ETDDMABUFPTR	32'h0	ETD <n> DMA Buffer Transfer Pointer Stores the current index of the buffer when AHB RETRY response is asserted.

- **USB-OTG Endpoint <n> DMA Buffer Transfer Pointer Register (USBOTG_EP<n>_BUF_PTR) – 0x0A80~0x0A9C**

If the DMA is paused due to the encounter of an AHB RETRY response and the SKIPONRETRY feature is enabled, the current index into the buffer is stored in these registers so that the DMA knows where to re-start the transfer. The buffer transfer pointers are accessible only for debug purposes.

EP0~EP3 OUT/IN Registers have same definition.

Bit	Name	Default	Description
31:0(R/W)	EPDMABUFPTR	32'h0	Endpoint <n> OUT or IN DMA Buffer Transfer Pointer Stores the current index of the buffer when AHB RETRY response is asserted.

7.6 IDE Interface

7.6.1 Overview

The IDE Interface is a licensed IP from Mentor Graphics®: M82371IDE. The M82371IDE supports both primary and secondary IDE (Integrated Drive Electronics) channels, with up to two devices per channel. The timing of each device is individually configurable. But in Atlas™-II, the IDE interface only supports primary IDE (Integrated Drive Electronics) channel, with up to two devices. So setting up the secondary channel won't take effect.

The M82371IDE also supports an intermediary synchronous bus interface (Peripheral Bus Interface – PBI) allowing easy interfacing to proprietary bus types. It is connected to the internal PCI Bus in Atlas™-II.

The IDE interface supports the following key features:

- Support primary IDE channel
- Support for two IDE devices
- Independent programmable timing for each device
- Programmable DMA
- Support PIO

In Atlas-II™, The IDE interface shares pins with ROM/SRAM, PCMCIA and NAND Flash.

7.6.2 Pin Description

IDE Pins are multiplexed with ROM I/F. Please refer to the section of “Pin Sharing” for more details.

Table 51. ROM/SRAM Interface Pin Description

Pin name	Pin Direction	Description
IDE_D<15:0>	Bi-directional	IDE Data
IDE_A<2:0>	Output	IDE Address
IDE_IOR_B	Output	Read Strobe (Active-low)
IDE_IOW_B	Output	Write Strobe (Active-low)
IDE_IORDY_B	Input	Allow IDE device to extend I/O cycle (Active-low)
IDE_CS_B_1	Output	IDE Select Data/Command (Active-low)
IDE_CS_B_0	Output	IDE Select Status/Control (Active-low)
IDE_DREQ	Input	DMA Transfer Request
IDE_DACK	Output	DMA transfer Acknowledge (Active-low)
IDE_IRQ	Input	IDE Interrupt

7.6.3 IDE Interface Registers

The IDE Configuration Registers base address is 0x47D01000. The following table shows the Configuration Register Mapping.

The address of the IDE device Data/Command register is 0x47D001F0~47D001F7.

The address of the IDE device Control/Status register is 0x47D003F6.

7.6.3.1 IDE Configuratoin Registers

Table 52. IDE Configuration Register Mapping

RISC Addr<7:0>	Register	Description
0x0000	IDE_VEN_DEV_ID	Vendor ID and Device ID register
0x0004	IDE_CMD_STATUS	Command and Status register
0x0008	IDE_REV_CLASS	Revision ID and Class Code register
0x000C	IDE_MLT_HTYPE	Master Latency & Header Type register
0x0010-0x001C	-	Reserved
0x0020	IDE_BMI_BASE	Bus Master Interface Base Address register
0x0024-0x003C	-	Reserved
0x0040	IDE_TIMING	Primary & Secondary IDE Timing register
0x0044	IDE_SLAVE_TIMING	Slave IDE Timing register
0x0048	-	Reserved
0x004C	-	Reserved
0x0050	IDE_CLK_DIV	Clock and INT Control register
Others	-	Reserved

- IDE Vendor and Device ID Register (IDE_VEN_DEV_ID) – 0x0000

Bit	Name	Default	Description
15:0 (R)	VEN_ID	-	Manufacturer's identification number
31:16 (R)	DEV_ID	-	Device identification number

- IDE Command and Status Register (IDE_CMD_STATUS) – 0x0004

Bit	Name	Default	Description
0 (R/W)	IO_EN	1'b0	I/O Enable
1	-	1'b0	Reserve
2 (R/W)	BMF_EN	1'b0	Bus Master Function Enable
22:3	-	20'h0	Reserved
23 (R)	FBB	1'b1	Fast Back to Back option – Not implemented
24 (R)	DPD	1'b0	Data Parity Detect option – Not implemented
26:25 (R)	DEVSEL_TIM_STATUS	1'b0	DEVSEL Timing Status. Set to 1'b1 to indicate medium speed timing for "NAVAL" assertion.
27 (R)	SIG_TABORT	1'b0	Signaled Target Abort Status. Set when the M82371IDE signals a transaction abort.
28 (R)	REC_TABORT	1'b0	Receive Target Abort. Set when M82371IDE as a bus master receives a target abort.
29 (R)	MABORT	1'b0	Master Abort Status. Set when M82371IDE as a bus master generates a master abort.

31:30	-	2'h0	Reserved
-------	---	------	----------

NOTE: Bits 27, 28 and 29 are Read and Clear (R/C). The bit may be read, the read having no effect on the register value. Writing "0" to the bit has no effect, writing "1" will clear the bit.

- **IDE Revision and Class Code Register (IDE_REV_CLASS) – 0x0008**

Bit	Name	Default	Description
7:0 (R)	DEV_ID	-	Revision identification number
15:8 (R)	CCP	-	Class Code Programming
31:16 (R)	CCC	-	Class Code – Class. Base Class 15:8, Sub Class 7:0

- **IDE Master Latency and Header Type Register (IDE_MLT_HTYPE) – 0x000C**

This register sets the minimum period for which the bus will be held (assuming there is data to transmit) – in multiples of 16 PBI clocks.

Bit	Name	Default	Description
7:0	-	-	Reserved
11:8 (R)	LSB	4'h0	Least significant bits of the count value (fixed)
15:12 (R/W)	MSB	4'h0	Most significant bits of the count value
23:16 (R)	HTYPE	-	Defines format of configuration registers as Type 0 and single-function.
31:24	-	-	Reserved

- **IDE Bus Master Interface Base Address Register (IDE_BMI_BASE) – 0x0020**

Bit	Name	Default	Description
0 (R)	IO_ADDR	1'b1	Always "1". Designates address as I/O,
3:1	-	3'h0	Reserved
15:4 (R/W)	BASE_ADDR	12'h0	Bus Master Interface Base Address. Must be no less than 0x2000
31:16	-	16'h0	Reserved

- **IDE Timing Register (IDE_TIMING) – 0x0040**

Bit	Name	Default	Description
0 (R/W)	D0_FT_BS	1'b0	Drive 0 Fast Timing Bank Select. When set to '0' (disabled), uses 16-bit compatible timings to the Data Port. When set to '1' (enabled), uses timing defined by Bit 3.
1 (R/W)	D0_IS_EN	1'b0	Drive 0 Enable IORDY Sampling
2 (R/W)	D0_PFPW_EN	1'b0	Drive 0 Enable Pre-fetch and Post-write
3 (R/W)	D0_DMAT_EN	1'b0	Drive 0 DMA Timing Enable Only. When set to '1' (enabled), allows fast transfer timing only for DMA transfers (PIO transfers use compatible timing). When set to '0' (disabled), both DMA and PIO transfers use fast timing.
4 (R/W)	D1_FT_BS	1'b0	Drive 1 Fast Timing Bank Select. When set to '0' (disabled), uses 16-bit compatible timings to the Data

			Port. When set to '1' (enabled), uses timing defined by Bit 7.
5 (R/W)	D1_IS_EN	1'b0	Drive 1 Enable IORDY Sampling
6 (R/W)	D1_PFPO_EN	1'b0	Drive 1 Enable Prefetch and Posting
7 (R/W)	D1_DMAT_EN	1'b0	Drive 1 DMA Timing Enable Only. When set to '1' (enabled), allows fast transfer timing only for DMA transfers (PIO transfers use compatible timing). When set to '0' (disabled), both DMA and PIO transfers use fast timing.
9:8 (R/W)	RCT	2'h0	Recovery Time (RCT). Defines the minimum number of clock periods NIRD/NIWR are inaction between successive cycles. 00 = 4, 01 = 3, 10 = 2, 11 = 1
11:10	-	2'h0	Reserved
13:12 (R/W)	ISP	2'h0	IORDY Sample Point (ISP). Defines how many clock periods after NIRD/NIWR is asserted on the IDE before IORDY is sampled. 00 = 5, 01 = 4, 10 = 3, 11 = 2
14 (R/W)	STIM_EN	1'b0	Slave IDE Timing Enable
15 (R/W)	DEC_EN	1'b0	IDE Decode Enabled. (Command block at 01F0-7h and Control block at 03F6h)
31:16	-	-	Reserved

Post-write and read-prefetch allows buffering of up to sixteen double words (16 x 32-bit). Posted writes have priority over all reads. All posted writes must have finished before any read may take place. Posted writes and prefetched reads share the same buffers. Primary and Secondary IDE channels have separate buffers.

Post-write: This only applies to transfers to the Data Port. The Peripheral Bus cycle is terminated at the point that data is written to the post-write buffer. The data is then transferred to the IDE device using IDE timing. If a subsequent write or read occurs at the Peripheral Bus Interface, it will be forced to wait for the previously posted write to finish.

Read-prefetch: This only applies to transfers from the Data Port. The first Data Port read starts the read-prefetch sequence, where subsequent bytes are read automatically and buffered at the Peripheral Bus Interface to minimize the read latency. Any other read or write access to the IDE device automatically terminates the read-prefetch and discards any data already fetched.

The following table shows the programming required for each of the PIO modes. The cycle times given are based on a 33MHz PCI clock: ISP and RCT values are given as a number of clock cycles.

Table 53. PIO Timing Programming

PIO Mode	ISP	RCT	Cycle time
8-bit Compatible	11	11	660ns
0/Compatible	6		600ns
1	-		
2	4	4	240ns
3	3	3	180ns
4	3	1	120ns

PIO modes 0, 2, 3, 4 refer to Data Port accesses. All other Ports use 8-bit compatible timing.

- **IDE Slave Timing (IDE_SLAVE_TIMING) – 0x0044**

Bit	Name	Default	Description
1:0 (R/W)	PD1_RCT	2'h0	Primary Drive 1 Recovery Time (RCT). Defines the minimum number of clock periods NIRD/NIWR are inactive between successive cycles. 00 = 4, 01 = 3, 10 = 2, 11 = 1
3:2 (R/W)	PD1_ISP	2'h0	Primary Drive 1 IORDY Sample Point (ISP). Defines how many clock periods after NIRD/NIWR is asserted on the IDE before IORDY is sampled. 00 = 5, 01 = 4, 10 = 3, 11 = 2
5:4 (R/W)	SD1_RCT	2'h0	Secondary Drive 1 Recovery Time (RCT). Defines the minimum number of clock periods NIRD/NIWR are inactive between successive cycles. 00 = 4, 01 = 3, 10 = 2, 11 = 1
7:6 (R/W)	SD1_ISP	2'h0	Secondary Drive 1 IORDY Sample Point (ISP). Defines how many clock periods after NIRD/NIWR is asserted on the IDE before IORDY is sampled. 00 = 5, 01 = 4, 10 = 3, 11 = 2
31:8	-	-	Reserved

- **IDE Clock Divide Register (IDE_CLK_DIV) – 0x0050**

Bit	Name	Default	Description
3:0 (R/W)	CLK_DIV	4'h0	Clock Ratio between System Clock and IDE Clock
31:4	-	28'h0	Reserved

7.6.3.2 IDE Control Registers

The IDE Control registers occupy 16 bytes of memory space, starting at the address specified by the “Bus Master Interface Base Address” (*IDE_BMI_BASE*) configuration register. Primary channel configuration registers occupy the first 8 byte locations; an identical set of registers for the Secondary channel occupy the 8 byte locations offset by 8 bytes from the primary channel locations. The following is the register mapping.

Table 54. IDE Bus Master Control Register Mapping

Address Offset	Register	Description
0x0	BMI_RW_CTRL	Bus Master Read/Write Control
0x1	-	Reserved
0x2	IBMI_INT_STATUS	Bus Master Interrupt Status
0x3	-	Reserved
0x4~7	BMI_DT_BASE	Bus Master Descriptor Table Base Address
0x8~F	-	Not affect our system

- **Bus Master Read/Write Control Register (BMI_RW_CTRL) – offset: 0x0**

Bit	Name	Default	Description
0 (R/W)	START	1'b0	Start/Stop Bus Master. 1= Start; 0 = Stop. DMA transfer is only enabled when this bit is true: clearing this bit will halt

			data transfer. Note: If this bit is cleared while a bus master (DMA) operation is in progress, the command will be abandoned and any data transferred from the IDE device discarded.
2:1	-	2'h0	Reserved
3 (R/W)	WRITE	1'b0	Bus Master Read/Write Control. 0 = Read; 1 = Write. During a DMA transfer, this bit is read only. Note: This bit must not be modified when bus master is active.
7:4	-	4'h0	Reserved

- **Bus Master Interrupt Status Register (BMI_INT_STATUS) – offset: 0x02**

Bit	Name	Default	Description
0 (R)	IDE_ACTIVE	1'b0	Bus Master IDE Active. This bit is set alongside bit 0 of the BMI_RW_CTRL register. It is cleared after all data has been transferred and EOT is set in the descriptor table.
1 (R)	DMA_ERR	1'b0	IDE DMA Error. This bit becomes set when either a target or master abort is encountered during DMA transfers on the PBI bus.
2 (R)	INT_STATUS	1'b0	IDE Interrupt Status. This bit is set when the IDE device asserts its IRQ signal. It may be cleared by writing '1' to this bit, even if the IDE IRQ signal is still active.
4:3	-	2'h0	Reserved
5 (R/W)	D0_DMA_CAP	1'b0	Drive 0 DMA Capable. When set to '1', the drive is capable of DMA transfer. Note: Software-controlled status bit: does not affect hardware.
6 (R/W)	D1_DMA_CAP	1'b0	Drive 1 DMA Capable. When set to '1', the drive is capable of DMA transfer. Note: Software-controlled status bit: does not affect hardware.
7	-	1'b0	Reserved

NOTE: Bit 1 and 2 are read and clear bits.

- **Bus Master Descriptor Table Base Address Register (BMI_DT_BASE) – offset: 0x4**

This register holds the base memory address for the Descriptor Table. The Descriptor Table must be DWORD aligned and must not cross any 4Kbyte boundary.

Bit	Name	Default	Description
1:0	-	2'h0	Reserved
31:2 (R/W)	DTBA	30'h0	Descriptor Table Base Address (DTBA).

7.6.3.3 IDE Device Registers

Table 55. IDE Device Port Map

ADDRESS	FUNCTION
---------	----------

IDE_CS_B_1	IDE_CS_B_0	Address Offset	Read	Write
Data/Command				
1	0	0	Data	Data
1	0	1	Error	Feature
1	0	2	Sector Count	Sector Count
1	0	3	Sector Number LBA<7:0>	Sector Number LBA<7:0>
1	0	4	Cylinder Low LBA<15:8>	Cylinder Low LBA<15:8>
1	0	5	Cylinder High LBA<23:16>	Cylinder High LBA<23:16>
1	0	6	Device/Head LBA<27:24>	Device/Head LBA<27:24>
1	0	7	Status	Command
Control/Status				
0	1	0~5	None	None
0	1	6	Alternate Status	Device Control
0	1	7	None	None
Others				
1	1	0~7	None	None
0	0	0~7	Illegal	Illegal

The address space 0x3F0~3F7 is shared with the floppy disk controller. 0x3F0~3F5 is used by the floppy disk controller; 0x3F6 is used by the IDE Primary channel for alternate status / device control; and 0x3F7 is a shared register. Data<7> is driven by the floppy disk controller and Data<6:0> is driven by the IDE device.

To resolve this conflict, newer systems do not respond to address 0x03F7, the access being passed to the floppy disk controller. For Atlas™-II IDE Controller, it has been decided to respond only to address 0x3F6.

7.7 PCMCIA/CF Interface

7.7.4 Overview

The PCMCIA/CF interface integrated in Atlas™-II is the same as the one in Atlas™-I. The PCMCIA master (M6730) in Atlas™-II allows the processor to communicate with other peripherals via the PCMCIA/CF interface.

The M6730 PCMCIA master is a PCI to PC Card host adapter solution from VirtualIP Group. It is compatible with PC Card standard, PCMCIA 2.1, and JEIDA 4.1. It provides suspend mode that stops transactions on the PC Card bus and turn off much of the internal circuitry. It provides fully buffered interface which eliminates the need for external logic required for buffering signals to/from the interface. Power consumption is controlled by limiting the signal transitions on the PC Card bus.

In Atlas™-II, The PCMCIA interface shares pins with ROM/SRAM, IDE and NAND Flash.

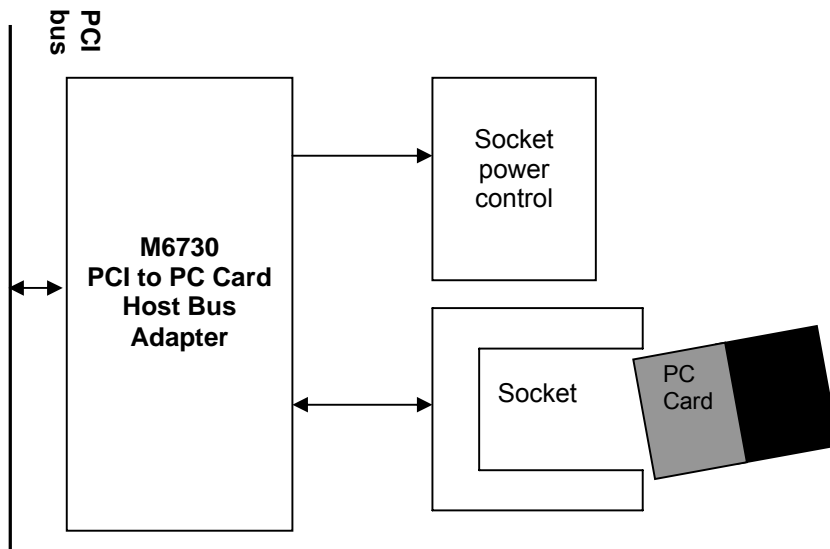


Figure 44. PCMCIA Block Diagram

7.7.5 PCMCIA Signal Descriptions

Table 56. PCMCIA Interface Signal Description

Pin Name	Direction	Description
X_PC_ADD<25:0>	Output	PC card address outputs
X_PC_D<15:0>	Bidirectional	PC card 16-bit data bus
X_PC_VS<2:1>	Input	The two signals are used to determine the operating voltage of the card. These two signals should externally pull high.
X_PC_CD_B<2:1>	Input	Card detect indication signal to Atlas™-II. These two signals should be externally pulled high.
X_PC_IOIS16_B	Input	In memory card interface, this input is interpreted as the status of the write protect switch on PC card; In I/O card interface, this input indicates the size of the I/O data at the current address on PC card.
X_PC_IORD_B	Output	This output goes active (low) for I/O reads from PC card to Atlas™-II.
X_PC_IOWE_B	Output	This output goes active (low) for I/O writes from Atlas™-II to PC card.
X_PC_WAIT_B	Output	This input indicates a request by the card to Atlas™-II to delay the cycle in progress until this signal is de-asserted.
X_PC_INPACK_B	Input	PC card input acknowledge signal. Currently this signal is not used.
X_PC_REG_B	Output	In Memory Card Interface mode, this output chooses between attribute and common memory. In I/O Card Interface mode, this signal is active (low).
X_PC_SPKR_B	Input	In Memory Card Interface mode, this input serves as the BVD2 (battery warning status) input. In I/O Card Interface mode, this input can be configured to accept a card's NSPKR digital audio output.
X_PC_STSCHG_B	Input	In Memory Card Interface mode, this input serves as

		the BVD1 (battery-dead-status) input. In I/O card Interface mode, this input is the NSTSCHG input, which indicates to the M6730 that the card's internal status has changed. If bit 7 of the Interrupt and General Control register is set to '1', this signal serves as the ring-indicate input for wake-on-ring system power management support.
X_PC_CE_B<2:1>	Output	These outputs are driven low by the Atlas™-II during card access cycles to control byte/word card access. X_PC_CE_B<1> enables odd-numbered address bytes. When configured for 8-bit cards, only X_PC_CE_B<1> is active and A0 is used to indicate access of odd or even-numbered bytes. These signals depend on the data size of the device at the PC card address and the Byte enables from the PCI bus.
X_PC_OE_B	Output	This output goes active (low) to indicate a memory read from the PC Card socket to the Atlas™-II.
X_PC_WE_B	Output	This output goes active (low) to indicate a memory write from the Atlas™-II to the PC Card socket.
X_PC_IREQ_B	Input	In Memory Card Interface mode, this input indicates to the Atlas™-II that the card is either ready or busy. In I/O card Interface mode, this input indicates a card interrupt request.
X_PC_RESET	Output	This output is low for normal operation and goes high to reset the card. To prevent reset glitches to a card, this signal is given tristate enable, unless a card is seated in the socket, card power is applied, and the card's interface signals are enabled.

7.7.6 CF Card Connection Example

The Atlas™-II PCMCIA interface is designed to provide a glueless interface between CompactFlash (CF) or PC card and Atlas™-II. It can support a 68-pin type II PCMCIA card and 50-pin CompactFlash card. When connecting to CompactFlash card, a PCMCIA type II passive adapter should be used to realize the conversion.

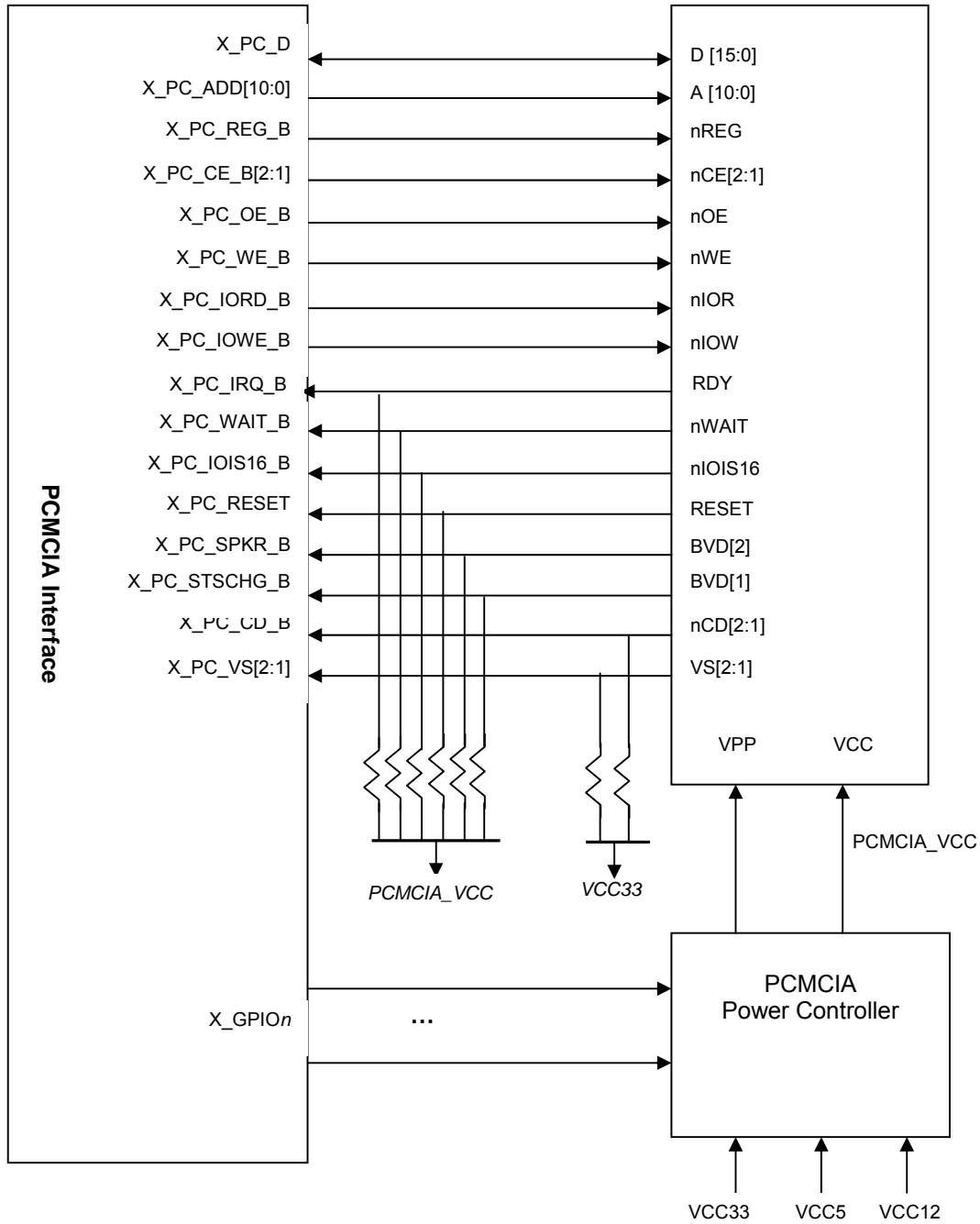


Figure 45. Connect to Compact Flash Card

7.7.7 PCMCIA Memory Mapping

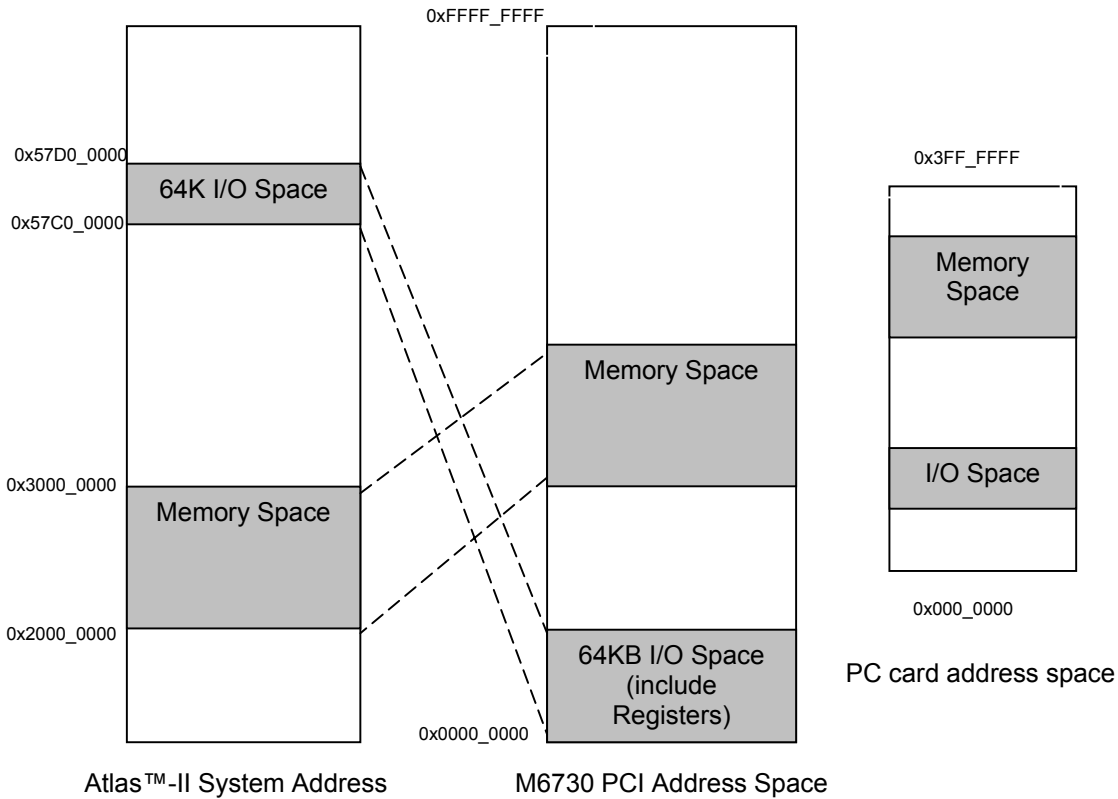


Figure 46. PCMCIA Memory Mapping

The PCMCIA memory can be divided into three parts:

- Register Space
- PC card I/O Space
- PC card Memory Space

7.7.7.1 Register Space

The registers in M6730 module are used to configure the PCMCIA interface, memory remapping and control the access timing of PCMCIA interface. They are accessed through PCI I/O mode. M6730 requests a I/O base address which is mapped to be 0x57C0_0000 in Atlas™-II. In order to save memory space, all M6730 registers are operated by accessing the I/O base address through index. In write operation, you should program the register index and data into PCMCIA Configuration Register first. The lower 8 bits is the index number, and the higher 8 bits is the data value. After you programmed this register, the M6730 will write the value into the register according to the index number internally. In read operation, you should program the register index number into PCMCIA Configuration Register first too. And after that, you could read the PCMCIA Configuration Register, the higher 8 bits is the data value you want to read out.

Table 57. PCMCIA Configuration Register (PCMCIA_CONFIG) – 0x57C00000

Bit	Name	Default	Description
7:0 (R/W)	INDEX	8'h0	Register Index Number
15:8 (R/W)	VALUE	8'h0	Register Data Value

31:16	-	16'h0	Reserved
-------	---	-------	----------

7.7.7.2 PC Card I/O Space

PC card may request I/O space from system, and in Atlas™-II from **0x57C0_0000 to 0x57D0_0000** will be PCMCIA I/O space, which will be converted to be **0x0000_0000 to 0x0000_FFFF** by PCMCIA interface block as a 64KB standard PCI I/O space.

7.7.7.3 PC Card Memory Space

The PC card memory space is mapped from **0x2000_0000 to 0x2FFF_FFFF**. When memory PC cards request memory from system, the start and end addresses allocated in Atlas™-II will be written into M6730 window configuration register and M6730 will in charge of remapping the Atlas™-II address into physical PC card address.

7.7.8 PCMCIA Interface Registers

The M6730's internal device control, window mapping, extension and timing registers are accessed through a pair of operation registers - an index register and a data register. The index register is accessed through base address 0 and the data register is accessed through base address 0 plus one. Please refer to the section of "Register Space" for the *PCMCIA Configuration Register*.

Table 58. M6730 Device Control Register Mapping

Index Value	Register Name	Description
0x00	M6730_REV	Megacell Revision
0x01	M6730_IF_STATUS	Interface Status
0x02	M6730_PWR_CTRL	Power Control
0x03	M6730_INT_CTRL	Interrupt and general Control
0x04	M6730_CARD_STATUS	Card Status Change
0x05	M6730_MGT_INT_CONFIG	Management Interrupt Configuration
0x06	M6730_MAP_EN	Mapping Enable
0x07	M6730_IO_WINDOW	I/O Window Control
0x08	M6730_SIO_MAP0_SAL	System I/O Map 0 Start Address low
0x09	M6730_SIO_MAP0_SAH	System I/O Map 0 Start Address high
0x0A	M6730_SIO_MAP0_EAL	System I/O Map 0 End Address low
0x0B	M6730_SIO_MAP0_EAH	System I/O Map 0 End Address high
0x0C	M6730_SIO_MAP1_SAL	System I/O Map 1 Start Address low
0x0D	M6730_SIO_MAP1_SAH	System I/O Map 1 Start Address high
0x0E	M6730_SIO_MAP1_EAL	System I/O Map 1 End Address low
0x0F	M6730_SIO_MAP1_ENH	System I/O Map 1 End Address High
0x10	M6730_SM_MAP0_SAL	System Memory Map 0 Start Address Low
0x11	M6730_SM_MAP0_SAH	System Memory Map 0 Start Address High
0x12	M6730_SM_MAP0_EAL	System Memory Map 0 End Address Low
0x13	M6730_SM_MAP0_EAH	System Memory Map 0 End Address High
0x14	M6730_CM_MAP0_OAL	Card Memory Map 0 Offset Address Low
0x15	M6730_CM_MAP0_OAH	Card Memory Map 0 Offset Address High
0x16	M6730_MISC_CTRL_1	Misc Control 1
0x17	M6730_FIFO_CTRL	FIFO Control
0x18	M6730_SM_MAP1_SAL	System Memory Map 1 Start Address Low
0x19	M6730_SM_MAP1_SAH	System Memory Map 1 Start Address High
0x1A	M6730_SM_MAP1_EAL	System Memory Map 1 End Address Low

0x1B	M6730_SM_MAP1_EAH	System Memory Map 1 End Address High
0x1C	M6730_CM_MAP1_OAL	Card Memory Map 1 Offset Address Low
0x1D	M6730_CM_MAP1_OAH	Card Memory Map 1 Offset Address High
0x1E	M6730_MISC_CTRL_2	Misc Control 2
0x1F	M6730_INFO	Megacell Information
0x20	M6730_SM_MAP2_SAL	System Memory Map 2 Start Address Low
0x21	M6730_SM_MAP2_SAH	System Memory Map 2 Start Address High
0x22	M6730_SM_MAP2_EAL	System Memory Map 2 End Address Low
0x23	M6730_SM_MAP2_EAH	System Memory Map 2 End Address High
0x24	M6730_CM_MAP2_OAL	Card Memory Map 2 Offset Address Low
0x25	M6730_CM_MAP2_OAH	Card Memory Map 2 Offset Address High
0x26	M6730_ATA_CTRL	ATA Control
0x27	M6730_SCRATCH_PAD	Scratch pad
0x28	M6730_SM_MAP3_SAL	System Memory Map 3 Start Address Low
0x29	M6730_SM_MAP3_SAH	System Memory Map 3 Start Address High
0x2A	M6730_SM_MAP3_EAL	System Memory Map 3 End Address Low
0x2B	M6730_SM_MAP3_EAH	System Memory Map 3 End Address High
0x2C	M6730_CM_MAP3_OAL	Card Memory Map 3 Offset Address Low
0x2D	M6730_CM_MAP3_OAH	Card Memory Map 3 Offset Address High
0x2E	M6730_EXT_INDEX	Extended Index
0x2F	M6730_EXT_DATA	Extended Data
0x30	M6730_SM_MAP4_SAL	System Memory Map 4 Start Address Low
0x31	M6730_SM_MAP4_SAH	System Memory Map 4 Start Address High
0x32	M6730_SM_MAP4_EAL	System Memory Map 4 End Address Low
0x33	M6730_SM_MAP4_EAH	System Memory Map 4 End Address High
0x34	M6730_CM_MAP4_OAL	Card Memory Map 4 Offset Address Low
0x35	M6730_CM_MAP4_OAH	Card Memory Map 4 Offset Address High
0x36	M6730_CIO_MAP0_OAL	Card I/O Map 0 Offset Address Low
0x37	M6730_CIO_MAP0_OAH	Card I/O Map 0 Offset Address High
0x38	M6730_CIO_MAP1_OAL	Card I/O Map 1 Offset Address Low
0x39	M6730_CIO_MAP1_OAH	Card I/O Map 1 Offset Address High
0x3A	M6730_SETUP_TIMING_0	Setup Timing 0
0x3B	M6730_CMD_TIMING_0	Command Timing 0
0x3C	M6730_REC_TIMING_0	Recovery Timing 0
0x3D	M6730_SETUP_TIMING_1	Setup Timing 1
0x3E	M6730_CMD_TIMING_1	Command Timing 1
0x3F	M6730_REC_TIMING_1	Recovery Timing 1
0x40~0x7F	0x40~0x7F	Reserved
0x80	0x80	Interrupt Status
0x81~0xFF	0x81~0xFF	Reserved

- **M6730 Megacell Revision Register (M6730_REV) – Index: 0x00**

Bit	Name	Default	Description
3:0 (R)	REVISION	4'h2	This field indicates the M6730's compatibility with the M82365.
5:4	-	2'h0	Reserved
7:6 (R)	INTERFACE_ID	2'h2	These bits are used to identify the interface supported by this controller. The M6730 supports both Memory and I/O interfaces.

			2'b00: I/O only 2'b01: Memory Only 2'b10: Memory and I/O 2'b11: Reserve
--	--	--	--

- **M6730 Interface Status Register (M6730_IF_STATUS) – 0x01**

Bit	Name	Default	Description															
1:0 (R)	BATT_VOLT_DET	2'h1	Battery Voltage Detect. These bits give the status of the <i>bvd2</i> and <i>bvd1</i> inputs from the PC card. <table border="1"> <thead> <tr> <th><i>bvd2</i> (Bit1)</th> <th><i>bvd1</i> (Bit0)</th> <th>PC Card Interpretation</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Card data lost</td> </tr> <tr> <td>0</td> <td>1</td> <td>Battery low warning</td> </tr> <tr> <td>1</td> <td>0</td> <td>Card data lost</td> </tr> <tr> <td>1</td> <td>1</td> <td>Battery/data okay</td> </tr> </tbody> </table>	<i>bvd2</i> (Bit1)	<i>bvd1</i> (Bit0)	PC Card Interpretation	0	0	Card data lost	0	1	Battery low warning	1	0	Card data lost	1	1	Battery/data okay
<i>bvd2</i> (Bit1)	<i>bvd1</i> (Bit0)	PC Card Interpretation																
0	0	Card data lost																
0	1	Battery low warning																
1	0	Card data lost																
1	1	Battery/data okay																
3:2 (R)	CARD_DET	2'h1	Card Detect. These bits give the status of the <i>ncd2</i> and <i>ncd1</i> inputs from the PC card. <table border="1"> <thead> <tr> <th><i>ncd2</i> (Bit1)</th> <th><i>ncd1</i> (Bit0)</th> <th>Card Detect Status</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>Either no card or cards not fully inserted</td> </tr> <tr> <td>1</td> <td>0</td> <td>Card is no fully inserted</td> </tr> <tr> <td>0</td> <td>1</td> <td>Card is not fully inserted</td> </tr> <tr> <td>0</td> <td>0</td> <td>Card is fully inserted</td> </tr> </tbody> </table>	<i>ncd2</i> (Bit1)	<i>ncd1</i> (Bit0)	Card Detect Status	1	1	Either no card or cards not fully inserted	1	0	Card is no fully inserted	0	1	Card is not fully inserted	0	0	Card is fully inserted
<i>ncd2</i> (Bit1)	<i>ncd1</i> (Bit0)	Card Detect Status																
1	1	Either no card or cards not fully inserted																
1	0	Card is no fully inserted																
0	1	Card is not fully inserted																
0	0	Card is fully inserted																
4 (R)	WT_PROTECT	1'b1	In memory card interface, this bit indicates the state of the <i>wp/niois16</i> signal on the PC card. This signal will reflect the status of the WP signal from the memory PC card socket. 1'b0: Card is not write protected 1'b1: Card is write protected.															
5 (R)	READY_BUSY	1'b1	This bit indicates the status of the <i>rdy/nireq</i> signal input from the PC card. 1'b0: Card is not ready 1'b1: Card is ready This signal will reflect the status of the <i>rdy</i> signal from the memory PC card socket.															
6 (R)	CARD_PWR_ON	1'b0	This bit indicates whether the power to the PC card is ON or OFF. 1'b0: Power to card is not on 1'b1: Power to Card is on.															
7 (R)	-	1'b1	It always reads 1'b1 for the M6730.															

- **M6730 Power Control Register (M6730_PWR_CTRL) – Index: 0x02**

Bit	Name	Default	Description
1:0 (R/W)	VPP1_PWR	2'h0	Reference to the bellowing table
3:2 (R/W)	-	2'h0	Compatibility bits
4 (R/W)	VCC_PWR	1'b0	1'b0: Power is not applied to the card: the <i>nvcc_3</i> and <i>nvcc_5</i> socket power control

			<p>signals are inactive (high)</p> <p>1'b1: Power is applied to the card: if bit 5 is 1'b0, or bit 5 is 1'b1 and ncd2 and ncd1 are active low, then the selected.</p> <p>Depending on the value of the bit 5, setting this bit to 1'b1 applies the power to the card.</p> <p>The Vcc3.3 bit determines whether 3.3v or 5v is applied to the card. Note that this bit is reset to 1'b0, when the card is removed from the socket, unless the auto_power_off feature is disabled by setting Extension control 1 register bit 1 to 1'b1.</p>
5 (R)	AUTO_PWR	1'b0	<p>1'b0: Vcc and Vpp1 power control signals are activated independent of the socket's ncd2 and ncd1 input levels</p> <p>1'b1: Vcc and Vpp1 power control signals are only activated if the socket's ncd2 and ncd1 inputs are both active (low).</p> <p>When this bit is set to 1'b1, the M6730 allows the power to the card to be turned on or off automatically with the insertion and removal of a PC card.</p>
6	-	1'b0	Compatibility bit
7	CARD_EN	1'b1	<p>1'b0: Outputs to the card socket are not enabled and are floating.</p> <p>1'b1: Outputs to the card socket are enabled if ncd2 and ncd1 inputs are active low and bit 4 is 1'b1.</p> <p>When this bit is 1'b1, the outputs to the PC card are enabled if a card is present and the card's power is being supplied. The signals affected include ncc1, ncc2, niord, niowr, noe, nreg, reset, a<25:0>, d<15:0> and nwe.</p>

Table 59. M6730 Power Control Register (1)

nrst Level	Both ncd1 and ncd2 are active (low)	Power Control Register		Interface Status Register	nvcc_c and nvcc_5 levels from Power Control Logic	vpp_pgm and vpp_vcc levels from Power Control Logic
		VCC_PWR (bit 4)	AUTO_PWR (bit 5)	CARD_PWR_ON (bit 6)		
High	X	0	X	0	Inactive (high)	Inactive (low)
High	X	1	0	1	Activated per MiscControl1 register, bit 1	Activated per Power Control register, bits 1 and 0
High	No	1	1	0	Inactive (high)	Inactive (high)
High	Yes	1	1	1	Activated per MiscControl1 register, bit 1	Activated per Power Control register, bits 1 and 0

Table 60. M6730 Power Control Register (2)

nrst Level	Both ncd1 and ncd2 are active (low)	Power Control Register		M6730 Output Signals to Socket
		Vcc Power (bit 4)	Card Enable (bit 7)	
Low	X	X	X	High Impedance
High	No	X	X	High Impedance
High	Yes	0	0	High Impedance
High	Yes	0	1	High Impedance
High	Yes	1	0	High Impedance
High	Yes	1	1	Enable

- M6730 Interrupt and General Control Register (M6730_INT_CTRL) – Index: 0x03

Bit	Name	Default	Description
3:0 (R/W)	IRQ_LEVEL	4'h0	These bits determine which IRQ will occur when the card causes an interrupt through the <i>rdy/nireq</i> signal in the PC card socket. 4'b0000: IRQ disabled 4'b0001: Reserved 4'b0010: Reserved 4'b0011: IRQ3 (ninta) 4'b0100: IRQ4 (nintb) 4'b0101: IRQ5 (nintc) 4'b0110: Reserved 4'b0111: IRQ7 (nintd) 4'b1000: Reserved 4'b1001: IRQ9 4'b1010: IRQ10 4'b1011: IRQ11 4'b1100: IRQ12 4'b1101: Reserved 4'b1110: IRQ14 4'b1111: IRQ15
4 (R/W)	MGT_INT_EN	1'b0	Management Interrupt Enable. 1'b0: Management IRQ specifies management interrupt bits 1'b1: Reserved This bit was created to determine how management interrupts occur on the ISA-based systems. It is included for the software compatibility. Because there is no NINTR on the PCI bus, setting this bit to 1'b1 would cause management interrupts not to occur. This bit has to be set to 1'b0 if any of the four management interrupts are to occur. If this bit is 1'b1, management interrupts would not occur, even if they are individually enabled in the management interrupt register.
5 (R/W)	CARD_IS_IO	1'b0	1'b0: Sets memory card Interface mode. The card socket is configured to support memory-only-type cards. All dual-function socket interface signals are defined to perform memory-only-type interface functions

			1'b1: Sets I/O Card Interface mode. The card socket is configured to support IO only This bit determines how the dual function socket interface will be used, whether the PC card is configured as I/O or Memory.
6 (R/W)	CARD_RESET	1'b0	1'b0: The cdreset signal to the card socket is set (high for normal, low for ATAmode) 1'b1: The cdreset signal to the card socket is set inactive (low for normal, high for ATA mode) This bit will determine whether the cdreset signal to the card is active or not. When the card enable bit is 1'b0, the cdreset signal is high impedance.
7 (R/W)	RI_EN	1'b0	Ring Indicator Enable. 1'b0: bvd1/nstschg/nri signal is status change function 1'b1 bvd1/nstschg/nri signal is ring indicate input signal from card. In I/O card interface mode, this bit determines whether the bvd1/nstschg/nri input signal is used as the nri input, so that the status is reflected in the nriout signal, or not. This bit is not valid in the Memory card interface.

- **M6730 Card Status Change Register (M6730_CARD_STATUS) – Index: 0x04**

This register indicates the source of a management interrupt generated by the M6730. For the management interrupts to be generated, the corresponding enables should be set in the *Management Interrupt* register. The management interrupts are acknowledged by reading from this register. This register is used for the interrupt controlling.

Bit	Name	Default	Description
0 (R/W)	BATT_DEAD/STS_C HG	1'b0	Battery Dead or Status Change. 1'b0: A transition (from high to low in Memory Card Interface mode or either high or low or low to high in I/O card Interface mode) on the bvd1/nstschg/nri signal has not occurred since this register was last read. 1'b1: A transition on bvd1/nstschg/nri signal has occurred. In a memory PC card interface, this bit is set to 1'b1 when the bvd1/nstschg/nri changes from high to low, indicating a battery dead condition. In the I/O interface, this bit is set to one when the bvd1/nstschg/nri signal changes from high to low or from low to high. In I/O Card interface, this bit is not affected by the fact that the <i>Interrupt and General Control</i> register bit 7 is set or not. This bit is reset to 1'b0 whenever this register is read.
1 (R/W)	BATT_WARNING_C HG	1'b0	Battery Warning Change. 1'b0: A transition (from high to low) on bvd2/nspkr/nled signal has not occurred since

			this register was last read. 1'b1: A transition on the <i>bvd2/nspkr/nled</i> signal has occurred In memory PC card interface, this bit is set to 1'b1, when the <i>bvd2/nspkr/nled</i> signal changes from high to low indicating a battery warning condition. This bit is reset to 1'b0 whenever this register is read.
2 (R/W)	RDY_CHG	1'b0	Ready Change. 1'b0: A transition on the <i>rdy/nireq</i> signal has not occurred since this register was last read. 1'b1: A transition on the <i>rdy/nireq</i> signal has occurred In memory PC card interface mode, this bit is set to 1'b1, when a change has occurred in the <i>rdy/nireq</i> signal input from the PC card. In an I/O interface, this bit is always read as 1'b0. This bit is reset to 1'b0 whenever this register is read.
3 (R/W)	CARD_DET_CHG	1'b0	Card Detect Change. 1'b0: A transition on neither the <i>ncd1</i> not the <i>ncd2</i> signal has occurred since this register was last read. 1'b1: A transition on either the <i>ncd1</i> not the <i>ncd2</i> signal has occurred This bit is set to 1'b1 when a change has occurred on the <i>ncd1</i> or <i>ncd2</i> signals. This bit is reset to 1'b0 when this register is read.
7:4	-	4'h0	Reserved

- **M6730 Management Interrupt Configuration Register (M6730_MGT_INT_CONFIG) – Index: 0x05**

This register control which status change cause management interrupts as well as to which output signal, the interrupt will go.

Bit	Name	Default	Description
0 (R/W)	BATT_DEAD_EN/ST ST_CHG_EN	1'b0	Battery dead or status change enable. 1'b0: Battery Dead Or Status Change management interrupt disabled 1'b1: If Battery Dead Or Status Change bit is 1'b1, a management interrupt will occur. When this bit is 1'b1, a management interrupt will occur when the <i>Card Status Change</i> register's bit 0 is set to 1'b1. This allows management interrupts to be generated on the changes in the <i>bvd1/nstschg/nri</i> input signal from the PC card.
1 (R/W)	BATT_WARNING_E N	1'b0	Battery Warning Enable. 1'b0: Battery Warning Change Management Interrupt disabled. 1'b1: If Battery Warning Change bit is 1'b1, a management interrupt will occur. When this bit is 1'b1, a management interrupt will occur when the <i>Card Status Change</i>

			register's bit 1 is set to 1'b1. This bit is not valid on I/O PC card interface.
2 (R/W)	RDY_EN	1'b0	Ready Enable 1'b0: Ready Change Management Interrupt disabled. 1'b1 If Ready Change bit is 1'b1, a management interrupt will occur. When this bit is 1'b1, a management interrupt will occur when the <i>Card Status Change</i> register bit 2 is set to 1'b1. This allows interrupts to be generated on the changes in the <i>rdy/nireq</i> signal input from the PC card.
3 (R/W)	CARD_DET_EN	1'b0	Card Detect Enable. 1'b0: Card Detect Change Management Interrupt disabled. 1'b1: If Card Detect Change bit is 1'b1, a management interrupt will occur. When this bit is set to 1'b1, a management interrupt will occur when the Card Status Change register bit 3 is set to 1'b1. This allows interrupts to be generated on the level changes on the <i>ncd1</i> or <i>ncd2</i> inputs from the PC card.
7:4 (R/W)	MGT_IRQ	4'h0	Management IRQ. 4'b0000: IRQ disabled 4'b0001: Reserved 4'b0010: Reserved 4'b0011: IRQ3 (ninta) 4'b0100: IRQ4 (nintb) 4'b0101: IRQ5 (nintc) 4'b0110: Reserved 4'b0111: IRQ7 (nintd) 4'b1000: Reserved 4'b1001: IRQ9 4'b1010: IRQ10 4'b1011: IRQ11 4'b1100: IRQ12 4'b1101: Reserved 4'b1110: IRQ14 4'b1111: IRQ15 These bits determine which interrupt will be used for a card status change management interrupt.

- **M6730 Mapping Enable Register (M6730_MAP_EN) – Index: 0x06**

Bit	Name	Default	Description
0 (R/W)	MEM_MAP0_EN	1'b0	Memory Map0 Enable. 1'b0: Memory Window Mapping registers for Memory Window 0 disabled. 1'b1: Memory Window Mapping registers for Memory Window 0 enabled When this bit is 1'b1, the Memory Window Mapping registers for the memory window 0 are enabled and the controller will be able to respond to memory accesses in the memory

			space defined by those registers.
1 (R/W)	MEM_MAP1_EN	1'b0	Memory Map1 Enable. 1'b0: Memory Window Mapping registers for Memory Window 1 disabled. 1'b1: Memory Window Mapping registers for Memory Window 1 enabled When this bit is 1'b1, the Memory Window Mapping registers for the memory window 1 are enabled and the controller will be able to respond to memory accesses in the memory space defined by those registers.
2 (R/W)	MEM_MAP2_EN	1'b0	Memory Map2 Enable. 1'b0: Memory Window Mapping registers for Memory Window 2 disabled. 1'b1: Memory Window Mapping registers for Memory Window 2 enabled When this bit is 1'b1, the Memory Window Mapping registers for the memory window 2 are enabled and the controller will be able to respond to memory accesses in the memory space defined by those registers.
3 (R/W)	MEM_MAP3_EN	1'b0	Memory Map3 Enable. 1'b0: Memory Window Mapping registers for Memory Window 3 disabled. 1'b1: Memory Window Mapping registers for Memory Window 3 enabled When this bit is 1'b1, the Memory Window Mapping registers for the memory window 3 are enabled and the controller will be able to respond to memory accesses in the memory space defined by those registers.
4 (R/W)	MEM_MAP4_EN	1'b0	Memory Map4 Enable. 1'b0: Memory Window Mapping registers for Memory Window 4 disabled. 1'b1: Memory Window Mapping registers for Memory Window 4 enabled When this bit is 1'b1, the Memory Window Mapping registers for the memory window 4 are enabled and the controller will be able to respond to memory accesses in the memory space defined by those registers.
5 (R/W)	-	1'b0	Compatibility bit
6 (R/W)	IO_MAP0_EN	1'b0	IO Map0 Enable. 1'b0: IO Window Mapping registers for IO Window 0 disabled 1'b1: IO Window Mapping registers for IO Window 0 enabled When this bit is 1'b1, the I/O Window Mapping registers for the I/O window 0 are enabled and the controller will be able to respond to I/O accesses in the I/O space defined by those registers.
7 (R/W)	IO_MAP1_EN	1'b0	IO Map1 Enable. 1'b0: IO Window Mapping registers for Memory Window 1 disabled

			1'b1: I/O Window Mapping registers for Memory Window 1 enabled When this bit is 1'b1, the I/O Window Mapping registers for the I/O window 1 are enabled and the controller will be able to respond to I/O accesses in the I/O space defined by those registers.
--	--	--	--

• **M6730 I/O Window Control Register (M6730_IO_WIN_CTRL) – Index: 0x07**

Bit	Name	Default	Description
0 (R/W)	IO_WIN0_SIZE	1'b0	I/O Window 0 Size. 1'b0: 8-bit data path to I/O Window 0 1'b1: 16-bit data path to I/O Window 0 When the bit 1 of this register is 1'b0, this bit determines the width of the data path for the I/O window 0 accesses to the PC card. When the bit 1 is 1'b1, this bit is ignored.
1 (R/W)	AUTO_SIZE_IO_WINDOW0	1'b0	Auto_size I/O Window 0. 1'b0: I/O Window 0 Size (see bit 0 of this register) determines the data path for I/O Window 0 access. 1'b1: The data path to I/O Window 0 is determined by the -IOIS 16 level returned by the card. This bit controls the method that the width of the data path for the I/O window 0 accesses to the card is determined. Note that when this bit is 1'b1, the niois16 signal determines the data path width to the PC card.
2 (R/W)	-	1'b0	Compatibility bit
3 (R/W)	TIMING_REG_SELO	1'b0	Timing Register Select 0. 1'b0: Accesses made with timing specified in Timer Set 0 registers. 1'b1: Accesses made with timing specified in Timer Set 1 registers. This bit determines the access timing specification for the I/O window 0.
4 (R/W)	IO_WIN1_SIZE	1'b0	I/O Window 1 Size. 1'b0: 8-bit data path to I/O Window 1 1'b1: 16-bit data path to I/O Window 1 When the bit 5 of this register is 1'b0, this bit determines the width of the data path for the I/O window 1 accesses to the PC card. When the bit 5 is 1'b1, this bit is ignored.
5 (R/W)	AUTO_SIZE_IO_WINDOW1	1'b0	Auto_size I/O Window 1. 1'b0: I/O Window 0 Size (see bit 4 of this register) determines the data path for I/O Window 1 access. 1'b1: The data path to I/O Window 1 is determined by the niois16 level returned by the card. This bit controls the method that the width of the data path for the I/O window 1 accesses to

			the card is determined. Note that when this bit is 1'b1, the <i>niois16</i> signal determines the data path width to the PC card.
6 (R/W)	-	1'b0	Compatibility bit
7 (R/W)	TIMING_REG_SEL1	1'b0	Timing Register Select 1. 1'b0: Accesses made with timing specified in Timer Set 0 registers. 1'b1: Accesses made with timing specified in Timer Set 1 registers. This bit determines the access timing specification for the I/O window 1.

- **M6730 System I/O Map 0~1 Start Address Low Register (M6730_SIO_MAP0~1_SAL) – Index: 0x08, 0x0C**

There are two separate *System I/O Map Start Address Low* registers, each with identical fields.

Bit	Name	Default	Description
7:0 (R/W)	SA<7:0>	8'h0	Start Address <7:0> This register contains the least significant byte of the address that specifies where in the I/O space the corresponding I/O map will begin I/O accesses that are equal or above this address and equal or below the corresponding system I/O map end address will be mapped into the I/O space of the corresponding PC card.

- **M6730 System I/O Map 0~1 Start Address High Register (M6730_SIO_MAP0~1_SAH) – Index: 0x09, 0x0D**

There are two separate system I/O Map Start Address High registers, each with identical fields.

Bit	Name	Default	Description
7:0 (R/W)	SA<15:8>	8'h0	Start Address <15:8> This register contains the most significant byte of the address that specifies where in the I/O space the corresponding I/O map will begin. I/O accesses that are equal or above this address and equal or below the corresponding system I/O map end address will be mapped into the I/O space of the corresponding PC card.

- **M6730 System I/O Map 0~1 End Address Low Register (M6730_SIO_MAP0~1_EAL) – Index: 0x0A, 0x0E**

There are two separate system I/O Map End Address Low registers, each with identical fields.

Bit	Name	Default	Description
7:0 (R/W)	EA<7:0>	8'h0	End Address <7:0> This register contains the least significant byte of the address that specifies where in the I/O space the corresponding I/O map will end

			I/O accesses that are equal or below this address and equal or above the corresponding system I/O map start address will be mapped into the I/O space of the corresponding PC card.
--	--	--	---

- **M6730 System I/O Map 0~1 End Address High Register (M6730_SIO_MAP0~1_EAH) – Index: 0x0B, 0x0F**

There are two separate system I/O Map End Address High registers, each with identical fields.

Bit	Name	Default	Description
7:0 (R/W)	EA<15:8>	8'h0	This register contains the most significant byte of the address that specifies where in the I/O space the corresponding I/O map will end I/O accesses that are equal or below this address and equal or above the corresponding system I/O map start address will be mapped into the I/O space of the corresponding PC card.

- **M6730 Card I/O Map 0~1 Offset Address Low Register (M6730_CIO_MAP0~1_OAL) – Index: 0x36, 0x38**

There are two separate system I/O Map Offset Address Low registers, each with identical fields.

Bit	Name	Default	Description
7:1 (R/W)	OA<7:1>	7'h0	Offset Address <7:1> This register contains the least significant byte of the quantity that will be added to the system I/O address that determines where in the PC Card's I/O map the I/O access will occur.
0	-	1'h0	This bit must be programmed to zero

- **M6730 Card I/O Map 0~1 Offset Address High Register (M6730_CIO_MAP0~1_OAH) – Index: 0x37, 0x39**

There are two separate system I/O Map Offset Address High registers, each with identical fields.

Bit	Name	Default	Description
7:0 (R/W)	OA<15:8>	8'h0	Offset Address <15:8> This register contains the most significant byte of the quantity that will be added to the system I/O address that determines where in the PC Card's I/O map the I/O access will occur.

- **M6730 System Memory Map 0~4 Start Address Low Register (M6730_SM_MAP0~4_SAL) – Index: 0x10, 0x18, 0x20, 0x28, 0x30**

There are five separate System Memory Map Start Address Low registers, each with identical fields.

The following information about the memory map windows is important.

1) The memory mapping register determines where in the PCI memory space and the PC card memory space accesses will occur. There are five memory windows that can be used independently.

- 2) The memory windows are enabled and disabled using the *Mapping Enable* register.
- 3) To specify where in the PCI space a memory window is mapped, start and end addresses are specified. A memory window is selected whenever the appropriate **MEM_MAP_EN** bit is set and the following conditions are true:
- The PCI address is greater than or equal to the appropriate System Memory Map Start Address register.
 - The PCI address is less than or equal to the appropriate System Memory Map End Address register.
 - The System Memory Map Upper Address register is equal to the upper PCI address.
- 4) Start and End addresses are specified within PCI address bits 23:12. This sets the minimum memory window size to 4 KBytes. Memory windows are specified within the PCI memory address space.

Bit	Name	Default	Description
7:0 (R/W)	SA<19:12>	8'h0	Start Address <19:12> This register contains the least significant byte of the address that specifies where in the memory space the corresponding memory map will begin. Memory accesses that are equal or above this address and equal or below the corresponding System Memory Map End Address will be mapped into the memory space of the corresponding PC card.

- **M6730 System Memory Map 0~4 Start Address High Register (M6730_SM_MAP0~4_SAH) – Index: 0x11, 0x19, 0x21, 0x29, 0x31**

There are five separate System Memory Map Start Address High registers, each with identical fields.

Bit	Name	Default	Description
3:0 (R/W)	SA<23:20>	4'h0	Start Address <23:20> This register contains the most significant nibble of the address that specifies where in the memory space the corresponding memory map will begin. Memory accesses that are equal or above this address and equal or below the corresponding <i>System Memory Map End Address</i> will be mapped into the memory space of the corresponding PC card.
5:4 (R/W)	-	2'h0	Scratch pad bits
6 (R/W)	-	1'b0	Compatibilit bit
7 (R/W)	WIN_DATA_SIZE	1'b0	Window Data Size This bit determines the data path size to the PC card.

- **M6730 System Memory Map 0~4 End Address Low Register (M6730_SM_MAP0~4_EAL) – Index: 0x12, 0x1A, 0x22, 0x2A, 0x32**

There are five separate System Memory Map End Address Low registers, each with identical fields.

Bit	Name	Default	Description
7:0 (R/W)	EA<19:12>	8'h0	End Address <19:12> This register contains the least significant byte of the address that specifies where in the memory space the corresponding memory map

			will end. Memory accesses that are equal or below this address and equal or above the corresponding <i>System Memory Map Start Address</i> will be mapped into the memory space of the corresponding PC card.
--	--	--	---

- **M6730 System Memory Map 0~4 End Address High Register (M6730_SM_MAP0~4_EAH) – Index: 0x13, 0x1B, 0x23, 0x2B, 0x33**

There are five separate System Memory Map End Address High registers, each with identical fields.

Bit	Name	Default	Description
3:0 (R/W)	EA<23:20>	4'h0	End Address <23:20> This register contains the most significant nibble of the address that specifies where in the memory space the corresponding memory map will begin. Memory accesses that are equal or below this address and equal or above the corresponding System Memory Map Start Address will be mapped into the memory space of the corresponding PC card.
5:4 (R/W)	-	2'h0	Scratch pad bits
7:6 (R/W)	CARD_TIMER_SEL	2'h0	Card Timer Select 2'b00: Selects Timer Set 0. 2'b01: Selects Timer Set 1. 2'b10: Selects Timer Set 1. 2'b11: Selects Timer Set 1. This field determines the timer set. Timer Set 0 and 1 reset to values compatible with the standard PCI and three wait state cycles.

- **M6730 System Memory Map 0~4 Offset Address Low Register (M6730_SM_MAP0~4_OAL) – Index: 0x14, 0x1C, 0x24, 0x2C, 0x34**

There are five separate System Memory Map Offset Address Low registers, each with identical fields.

Bit	Name	Default	Description
7:0 (R/W)	OA<19:12>	8'h0	Offset Address <19:12> This register contains the least significant byte of the address that will be added to the system address to determine where in the PC card's memory map the memory access will occur.

- **M6730 System Memory Map 0~4 Offset Address High Register (M6730_SM_MAP0~4_OAH) – Index: 0x15, 0x1D, 0x25, 0x2D, 0x35**

There are five separate System Memory Map Offset Address High registers, each with identical fields.

Bit	Name	Default	Description
5:0 (R/W)	OA<25:20>	6'h0	Offset Address <25:20> This register contains the most significant six bits of the address that is added to the system memory address to get where in the PC card memory address space, the access will occur.

6 (R/W)	REG_SETTING	1'b0	REG Setting. 1'b0: nreg is not active for accesses made through this window. 1'b1: nreg is active for accesses made through this window. This bit determines whether the nreg signal is active for accesses through the window or not. The CIS (Card Information Structure) memory or the attribute memory is allotted this window by setting this bit to 1'b1.
7 (R/W)	WT_PROTECT	1'b0	Write Protect. 1'b0: Writes to the Card through this window are allowed. 1'b1: Writes to the Card through this window are inhibited This bit determines whether writes to the card through this window are allowed or not.

- **M6730 Misc Control 1 Register (M6730_MISC_CTRL_1) – Index: 0x16**

Bit	Name	Default	Description
0 (R/W)	MM_EN	1'b0	Multimedia Enable. 1'b0: Socket address lines are normal 1'b1: Socket address lines a<25:4> are high-impedance. This bit tristates socket address lines a<25:4> . All other aspects of the socket are not affected by this bit.
1 (R/W)	VCC_33	1'b0	VCC 3.3V. 1'b0: The power control logic will apply 5.0V when card power is to be applied. 1'b1: The power control logic will apply 3.3V when card power is to be applied This bit determines which output signal is used to enable Vcc power to the socket when the card power is applied; this bit is used in conjunction with the bits 5:4 of the <i>Power Control</i> register bits.
2 (R/W)	PM_INT	1'b0	Pulse Management Interrupt. 1'b0: Interrupts are passed to the irq<XX> signal as level-sensitive. 1'b1: When an interrupt occurs, the irq<XX> signal is driven with the pulse train shown in the figure 4.8 and allows for interrupt sharing. This bit is valid only in External Hardware Interrupt Signalling mode. This bit selects Level or Pulse mode operation of the irqx signal. Note that a clock must be present on pci_clk for the pulsed interrupts to work. Note that only the irq9 and irq10 can work in the pulse mode where as the other IRQs are just interrupt sensors.
3 (R/W)	PS_IRQ	1'b0	Pulse System IRQ 1'b0: Interrupts are passed to the irq<XX>

			<p>signal as level-sensitive.</p> <p>1'b1: When an interrupt occurs, the <i>irq</i><XX> signal is driven with the pulse train shown in Fig. 4.8 allows for interrupt sharing.</p> <p>This bit is valid only in the External Hardware Interrupt Signalling mode. This bit selects Level or Pulse mode operation of the <i>irqx</i> signals. Note that only the <i>irq9</i> and <i>irq10</i> can work in the pulse mode where as the other IRQs are just interrupt sensors.</p>
4 (R/W)	SPKR_EN	1'b0	<p>Speaker Enable.</p> <p>1'b0: <i>nspkr_out</i> is high-impedance</p> <p>1'b1: <i>nspkr_out</i> is driven from the XOR of <i>nspkr</i> from each enabled socket.</p> <p>This bit determines whether the card NSPKR signal will drive NSPKROUT.</p>
6:5 (R/W)	-	2'h0	
7 (R/W)	INPACK_EN	1'b0	<p>Input Acknowledge Enable.</p> <p>The <i>ninpack</i> function is not applicable in the PCI bus environments. This bit is provided for compatibility reasons. The programming of this bit has no effect on the functionality of the M6730.</p>

- **M6730 FIFO Control Register (M6730_FIFO_CTRL) – Index: 0x17**

Bit	Name	Default	Description
6:0 (R/W)	-	7'h0	Scratch bits
7 (R/W)	FIFO_STATUS/FLUSH	1'b1	<p>FIFO Status/Flush FIFO</p> <p>1'b0: FIFO not empty/No operation occurs</p> <p>1'b1: FIFO empty/Flush the FIFO</p> <p>This bit controls FIFO operation and reports FIFO status. To reset the FIFO and the FIFO pointers, write a 1'b1 to this bit. During read operations from this register, when this bit is 1'b1, the FIFO is empty. During read operations when this bit is 1'b0, the FIFO has valid data.</p> <p>This bit is used to ensure the FIFO is empty before changing any register contents; registers should not be modified while the write FIFO is not empty.</p> <p>FIFO contents will be lost whenever any of the following occurs:</p> <ol style="list-style-type: none"> 1) <i>nrst</i> signal is active. 2) The card is removed. 3) Vcc power bit in the Power control register is reset to 1'b0. 4) An address phase comes for an IO/Memory window when the earlier data transfer is still going on to the PC card. 5) When a 1'b1 is written to the bit 7 of the register.

- **M6730 Misc Control 2 Register (M6730_MISC_CTRL2) – Index: 0x1E**

Bit	Name	Default	Description
0 (R/W)	-	1'b0	Reserved
1 (R/W)	LPD_MODE	1'b0	Low-power Dynamic Mode. 1'b0: Clock runs always. 1'b1: Normal operation, stop clock when possible This bit determines whether Low power dynamic mode is enabled or not.
2 (R/W)	SPD_MODE	1'b0	Suspend Mode 1'b0: Normal operation 1'b1: Stop internal clock, enable all low-power modes, and disable socket access. This bit enables or disables the Suspend mode.
5:3 (R/W)	-	3'h0	Reserved
6 (R/W)	DB_EN	1'b0	Debounce Enable 1'b0: Disable debounce. 1'b1: Enable debounce. When ever this bit is set, ncd1 and ncd2 signals from port are passed through the debouncing circuit which is used to avoid glitches and bouncing of card detect signals
7 (R/W)	RI_OUT	1'b0	nriout/nintb/irq10 is nriout 1'b0: Normal interrupt operation on the nri_out/nintb/irq10 signal 1'b1: nri_out/nintb/irq10 is connected to ring indicate signal on the system logic. This bit determines the function of the nriout/nintb/irq10 signal. When this bit is set to 1'b1, nriout/nintb/irq10 can be used to trigger restoration of system activity when a high to change is observed on the bvd1/nstschg/nri signal input from the PC card.

- **M6730 Megacell Information (M6730_INFO) – Index: 0x1F**

Bit	Name	Default	Description
0 (R)	-	1'b0	Reserved
4:1 (R)	REV_LEVEL	4'hxxxx	This field contains the revision level of the M6730 megacell. This field is to be decided by the customer.
5 (R)	DUAL_SINGLE	1'b1	1'b0: Megacell identified as a single-socket controller. 1'b1: Megacell identified as a dual-socket controller This bit specifies that the M6730 supports two sockets or one socket. This bit always returns 1'b1 when read as the M6730 is a dual socket device.
7:6 (R)	HA_ID	2'bxx	Host Adapter Identification. 2'b00 Second read after I/O write to this register.

			2'b11 First read after I/O write to this register. This field identifies the M6730 megacell. After megacell reset or doing an I/O write operation to this register, the first read will return '11'. The next read will return '00'. This type of toggling on the reads can be used to identify the host adapter.
--	--	--	---

- **M6730 ATA Control Register (M6730_ATA_CTRL) – Index: 0x26**

Bit	Name	Default	Description
0 (R/W)	ATA_MODE	1'b0	ATA Mode. 1'b0: Normal operation 1'b1: Configures the socket interface to handle ATA-type disk drives. This bit reconfigures the particular socket as an ATA drive interface.
1 (R/W)	SPKR_LED	1'b0	Speaker is LED Input. 1'b0: Normal operation 1'b1: The PC Card <i>bvd2/nspkr/nled</i> signal will be used to drive <i>irq12</i> if Drive LED Enable is set. This bit changes the function of the <i>bvd2/nspkr/nled</i> signal from digital speaker input to disk status LED input. When in I/O Card Interface mode or ATA mode, setting this bit to 1'b1 reconfigures the <i>bvd2/nspkr/nled</i> input signal to serve as NLED input only. This bit should be set to 1'b0 if the interface is for Memory PC card.
2 (R/W)	-	1'b0	Scratch pad bit
3 (R/W)	A21	1'b0	In ATA mode, the value in this bit is applied to the ATA21 signal and is vendor specific. Certain vendor_specific performance enhancements beyond the PC card standard can be controlled through use of this bit. This bit has no hardware control when not in the ATA mode.
4 (R/W)	A22	1'b0	In ATA mode, the value in this bit is applied to the ATA22 signal and is vendor specific. Certain vendor_specific performance enhancements beyond the PC card standard can be controlled through use of this bit. This bit has no hardware control when not in the ATA mode.
5 (R/W)	A23/VU	1'b0	In ATA mode, the value in this bit is applied to the ATA23 signal and is vendor specific. Certain vendor_specific performance enhancements beyond the PC card standard can be controlled through use of this bit. This bit has no hardware control when not in the ATA mode.
6 (R/W)	A23/M/NS	1'b0	In ATA mode, the value in this bit is applied to the ATA24 signal and is vendor specific.

			Certain vendor_specific performance enhancements beyond the PC card standard can be controlled through use of this bit. This bit has no hardware control when not in the ATA mode.
7	A25/CSEL	1'h0	In ATA mode, the value in this bit is applied to the ATA25 signal and is vendor specific. Certain vendor_specific performance enhancements beyond the PC card standard can be controlled through use of this bit. This bit has no hardware control when not in the ATA mode.

- **M6730 Extend Index Register (M6730_EXT_INDEX) – Index: 0x2E**

Bit	Name	Default	Description
7:0 (R/W)	EXT_INDEX	8'h0	This register controls which of the following registers at index 0x2F can be accessed.

Table 61. Extended Indexed Registers

Extended Index	Extended Register	Description
0x00	-	Scratch pad
0x01	-	Reserved
0x02	-	Reserved
0x03	M6730_EXT_CTRL_1	Extension Control 1
0x04	-	Reserved
0x05	M6730_SM_MAP0_UA	System memory Map 0 Upper Address
0x06	M6730_SM_MAP1_UA	System memory Map 1 Upper Address
0x07	M6730_SM_MAP2_UA	System memory Map 2 Upper Address
0x08	M6730_SM_MAP3_UA	System memory Map 3 Upper Address
0x09	M6730_SM_MAP4_UA	System memory Map 4 Upper Address
0x0A	M6730_EXT_DAT	External Data
0x0B	-	Reserved
0x25	M6730_MISC_CTRL_3	Misc control 3

- **M6730 Extension Control 1 Register (M6730_EXT_CTRL_1) – Extension Index: 0x03**

Bit	Name	Default	Description
0 (R/W)	VCC_PWR_LOCK	1'b0	VCC Power Lock 1'b0: The Vcc Power bit (bit 4 of Power Control register) is not locked. 1'b1: The Vcc Power bit (bit 4 of Power Control register) cannot be changed by software This bit can be used to prevent card drivers from overriding the Socket Services' task of controlling power to the card, thus preventing situations where cards are powered incorrectly.
1 (R/W)	AUTO_PWR_CLR	1'b0	Auto Power Clear 1'b0: The Vcc Power bit (bit 4 of Power Control register) is reset to 1'b0 when the card is removed 1'b1: The Vcc Power bit (bit 4 of Power Control

			register) Note that the same function happens when the auto power bit of the Power control register is set together with the Vcc power bit. In this case, the auto power clear bit need not be set. When the card is removed, the power to the socket is disabled, even though the Vcc power bit is not reset to 1'b0.
2 (R/W)	LED_EN	1'b0	LED Activity Enable 1'b0: LED Activity disabled 1'b1: LED Activity enabled. This bit allows the <i>nled_out</i> signal to reflect any activity in the card. Whenever PC card cycles are in the process to or from a card in either socket, <i>nled_out</i> will be active low. The <i>nled</i> input from the IO PC card gives the activity level.
3 (R/W)	INV_CARD_IRQ	1'b0	Invert Card IRQ Output 1'b0: The card <i>irq</i> is active-high. 1'b1: The card <i>irq</i> is active-low. This bit changes the active high ISA type card IRQ level to an active-low output that complies with the PCI bus requirements.
4 (R/W)	INV_MGT_IRQ	1'b0	Invert Management IRQ Output 1'b0: The management <i>irq</i> is active-high. 1'b1: The management <i>irq</i> is active-low. This bit changes the active high ISA type card IRQ level to an active-low output that complies with the PCI bus requirements.
5 (R/W)	PULL_UP_CTRL	1'b0	Pull-up Control 1'b0: Pull-ups on <i>vs2</i> , <i>vs1</i> , <i>cd2</i> , and <i>cd1</i> are in use. 1'b1: Pull-ups on <i>vs2</i> , <i>vs1</i> , <i>cd2</i> , and <i>cd1</i> are turned off. This bit turns off the pull_up on the signals <i>vs2</i> , <i>vs1</i> , <i>ncd2</i> and <i>ncd1</i> . Turning off these pull_ups can be used in addition to suspend mode to even further reduce power when the cards are inserted but no card accessibility is required. Even though power may or may not be applied, the pull_up circuitry is disabled. NOTE: Insertion or removal of cards cannot be determined when this bit is 1'b1. Also if the card detect interrupts are enabled and the card is already in the socket, A CARD DETECT INTERRUPT WILL BE GENERATED WHEN THIS BIT IS CHANGED.
7:6 (R/W)	-	2'h0	Reserved

- **M6730 System Memory Map 0~4 Upper Address Register (M6730_SM_MAP0~4_UA) – Extension Index: 0x05~0x09**

Bit	Name	Default	Description
-----	------	---------	-------------

7:0 (R/W)	UA<7:0>	8'h0	Upper Address. These bits are used in comparing the PCI address bits 31:24 for each memory window (0~4). These bits are used in conjunction with the System Memory Map 0-4 Start Address and the System Memory Map 0-4 End Address registers.
-----------	----------------------	------	--

- **M6730 External Data (M6730_EXT_DAT) – Extension Index: 0x0A**

Bit	Name	Default	Description
0 (R)	Socket A VS1 Input	1'bx	These bits indicate the values of the four voltage sense signal inputs from the two PC card sockets. These bits are used to determine the operating voltage of the PC cards inserted.
1 (R)	Socket A VS2 Input	1'bx	
2 (R)	Socket B VS1 Input	1'bx	
3 (R)	Socket B VS2 Input	1'bx	
7:4 (R/W)	-	4'h0	Reserved

- **M6730 Misc Control 3 Register (M6730_MISC_CTRL_3) – Extension Index: 0x25**

During the power-on reset or hardware reset, the bits 1 and 0 are loaded with the values of the signals *misc1_in* and *misc0_in* respectively.

Bit	Name	Default	Description
1:0 (R/W)	SIS_MODE	2'bxx	System Interrupt Signalling Mode 2'b00: Reserved 2'b01: External-hardware Interrupt Signalling mode 2'b10: Reserved 2'b11: PCI Interrupt Signalling Mode The M6730 supports four interrupt signalling modes. The configuration of each of the modes were given earlier.
3:2 (R/W)	-	2'h0	Currently reserved. These bits are to be used for future expansion. These bits can be used for Socket Power Control Signalling Modes. With the help of external power control chips, it is possible to have different power control modes.
7:4 (R/W)	-	4'h0	Reserved

- **M6730 Extend Data Register (M6730_EXT_DATA) – Index: 0x2F**

Bit	Name	Default	Description
7:0 (R/W)	EXT_DATA	8'h0	The data in this register allows the registers indicated by the Extended Index register to be read and written. The value of this register is the value of the register selected by the Extended Index register

- **M6730 Setup Timing 0~1 Register (M6730_SETUP_TIMING_0~1) – Index: 0x3A, 0x3D**

1) All timing registers take effect immediately and should only be changed when the FIFO is empty.

2) Selection of Timer Set 0 or Timer Set 1 register sets is controlled by the I/O Window Control bits 7 and 3 and the Memory End Address High register bits 7:6.

There are two separate Setup Timing registers, each with identical fields.

The setup timing registers for each timer set controls how long a PC card cycle's command (that is **noe**, **nwe**, **niord** or **niowr**) setup time will be, in terms of the internal clock cycles. The overall command setup time will be:

$$S = 2 \times (Nval + 1)$$

The value of S, representing the number of clock cycles for the command setup, is then multiplied by the clock period and the actual command setup time is obtained.

Bit	Name	Default	Description
5:0 (R/W)	SMV<5:0>	6'h1	Setup Multiplier Value This field indicates the integer value Nval from 0 to 63; it is used to control the length of setup time before a command becomes active.
7:6 (R)	-	2'h0	Reserved

- **M6730 Command Timing 0~1 Register (M6730_CMD_TIMING_0~1) – Index: 0x3B, 0x3E**

NOTE: The Command timing set 0 resets to 6'h03 for socket timing equal to 230ns PC card timing whereas Timing set 1 (0x3E) resets to 6'h09 for socket timings equal to 590ns PC card timing.

There are two separate Command Timing registers, each with identical fields.

The command timing registers for each timer set controls how long a PC card cycle's command (that is **noe**, **nwe**, **niord** or **niowr**) active time will be, in terms of the internal clock cycles. The overall command time will be:

$$C = 2 \times (Nval + 1)$$

The value of C, representing the number of clock cycles for the command active, is then multiplied by the clock period and the actual command active time is obtained.

Bit	Name	Default	Description
5:0 (R/W)	CMV<5:0>	6'h3/9	Command Multiplier Value This field indicates the integer value Nval from 0 to 63; it is used to control the length of command active time.
7:6 (R)	-	2'h0	Reserved

- **M6730 Recover Timing 0~1 Register (M6730_REC_TIMING_0~1) – Index: 0x3C, 0x3F**

There are two separate Recovery Timing registers, each with identical fields.

The recovery timing registers for each timer set controls how long a PC card cycle's recovery (that is **noe**, **nwe**, **niord** or **niowr**) time will be, in terms of the internal clock cycles.

The overall command recovery time will be:

$$R = 2 \times (Nval + 1)$$

The value of R, representing the number of clock cycles for the command recovery, is then multiplied by the clock period and the actual command recovery time is obtained.

Bit	Name	Default	Description
5:0 (R/W)	RMV<5:0>	6'h02	Recover Multiplier Value This field indicates the integer value Nval from 0 to 63; it is used to control the command recovery time.
7:6 (R)	-	2'h0	Reserved

- **M6730 Interrupt Status Register (M6730_INT_STATUS) – Index: 0x80**

Bit	Name	Default	Description
0 (R/W)	MGT_INT_EN	1'b0	Management Interrupt Enable 1'b0: the interrupt of the management is not enabled 1'b1: the interrupt of the management is enabled
1 (R/W)	CARD_INT_EN	1'b0	Card Interrupt Enable 1'b0: the interrupt of the card is not enabled 1'b1: the interrupt of the card is enabled
2 (R/W)	MGT_INT_STATUS	1'b0	Management Interrupt Status 1'b0: the interrupt of the management is inactive 1'b1: the interrupt of the management is active
3 (R/W)	CARD_INT_STATUS	1'b0	Card Interrupt Status 1'b0: the interrupt of the card is inactive 1'b1: the interrupt of the card is active
7:4	-	4'h0	Reserved.

8 Peripheral Subsystem

8.1 Overview

In Atlas™-II, there are many peripheral I/O modules not connected to the internal PCI Bus. In another words, they do not access the system memory through System to PCI Bridge. Instead, these I/O modules are connected to the I/O memory Bus and they access the system memory through the I/O Bridge.

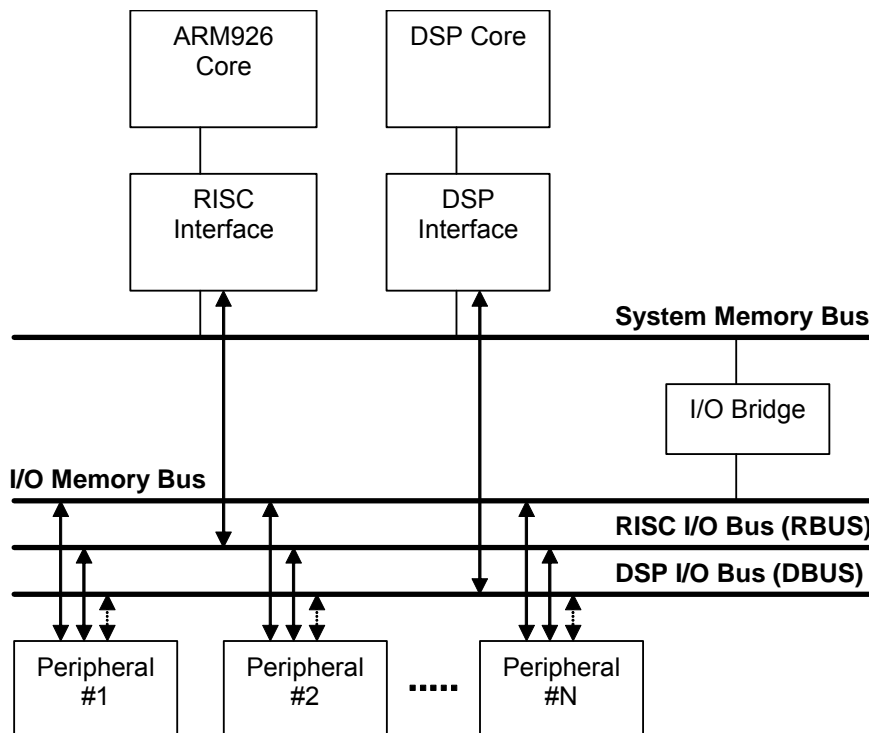


Figure 47. Peripheral Subsystem Diagram

There is an arbiter inside of the I/O Bridge to arbitrate the memory accesses from different bus masters. The arbiter can support up to 4 bus masters, but in Atlas™-II there are only one bus master: DMA Controller. All the other peripheral modules are under control of DMA Controller to accomplish the memory accesses.

8.2 I/O Bridge

8.2.1 Overview

The I/O Bridge transfers data between the I/O Memory Bus Masters and system memory. In the Atlas™-II the I/O Masters are allocated as the following table.

Table 62. I/O Memory Bus Masters

I/O Masters	Block Name
I/O Master0	DMA Controller
I/O Master1	Reserved

I/O Master2	Reserved
-------------	----------

The I/O Bridge is a slave device responding to an I/O Master's data transfer request. To transfer data, the I/O Master will request for bus access from the I/O Bridge. The I/O bridge block will do the arbitration internally, and grant the I/O memory to the I/O master. Then the I/O master will release a read or write command to the I/O Bridge. For read commands to the I/O Bridge, it will first read data from system memory and store it in a FIFO, and then transfer it to I/O the master. For a write command, the I/O Bridge will first read data from the I/O master and store it in a FIFO, and then transfer it to system memory.

8.2.2 I/O Bridge Registers

The I/O Bridge's address space is shared with the DMA address space (0xB000_0000~0xB000_FFFF). The I/O Bridge occupies address space from offset 0x800 to offset 0x810.

Table 63. I/O Bridge Register Mapping

RISC Address <11:0>	Register	Description
0x0800	IOBG_INT_CTRL	I/O Bridge interrupt controller
0x0804	IOBG_FLUSH	I/O Bridge flush controller
0x0808	IOBG_INT_STATUS	I/O Bridge interrupt status
0x080C	IOBG_ARB_STATUS	I/O Bridge arbiter status
0x0810	IOBG_ARB_CLKRATIO	I/O Bridge clock ratio
Others	-	Reserved

- **I/O Bridge interrupt controller (IOBG_INT_CTRL) – 0x0800**

The Interrupt from I/O Bridge is to monitor the abnormal transaction on I/O Memory bus. If one Master gets grant to access the I/O memory bus, after that it perform no operation on the bus. The I/O memory is blocked by this master. The I/O Bridge will issue an interrupt to inform this kind of situation.

Bit	Name	Default	Description
0 (R/W)	INT_EN	1'b0	IO bridge Interrupt enable
7:1	-	7'h0	Reserved
15-8	TIME_OUT	8'hFF	Timeout value = TIME_OUT * 32 + 31 (IO_CLK cycles) If the value is 0 then the actual time out value is 31. After 31 I/O clock cycles the I/O Bridge give grant signal, if the master still has no operation, the arbiter will assert an interrupt to the RISC.
31:16	-	16'h0	Reserved

- **I/O Bridge flush register (IOBG_FLUSH) – 0x0804**

Bit	Name	Default	Description
0 (R/W)	FLUSH	1'b0	IO bridge flush bit. I/O Bridge will clear this register when I/O Bridge is in idle or a transaction in I/O Bridge is over. In order to determine the current I/O Bridge operation is over. User set this register to 1. Then read this register until it is 0.
31:1	-	31'h0	Reserved

- **I/O Bridge interrupt status (IOBG_INT_STATUS) – 0x0808**

Bit	Name	Default	Description
0 (R/W)	INT_STATUS	1'b0	IO bridge status, write 1 to clear.
31:1	-	0	Reserved

- **I/O Arbiter Status Register (IO_ARB_STATUS) – 0x080C**

Bit	Name	Default	Description
3:0 (R)	MA<3:0>	4'h0	MA<3:0> means which master is occupying the bus currently, when IO Bridge generates an interrupt, the RISC can read these bits. For example, bit<2> is 1 means Master2 is occupying the bus currently. There might be following situation: after IO arbiter assert interrupt, the Master begins to transfer data, and finished it before RISC respond to the interrupt. So there might be wrong information. However with time out value sufficient large, things like this will not happen.
31:4	-	28'h0	Reserved

- **I/O Bridge clock ratio (IOBG_ARB_CLKRATIO) – 0x0810**

Bit	Name	Default	Description
1:0 (R/W)	CLK_RATIO	2'b01	IO bridge clock ratio 2'b00: SYS_CLK:IO_CLK = 1:1 2'b01: SYS_CLK:IO_CLK = 1:2 2'b11: SYS_CLK:IO_CLK = 1:4 Others are not supported, and should not be written
31:2	-	30'h0	Reserved

8.3 DMA Controller

8.3.1 Overview

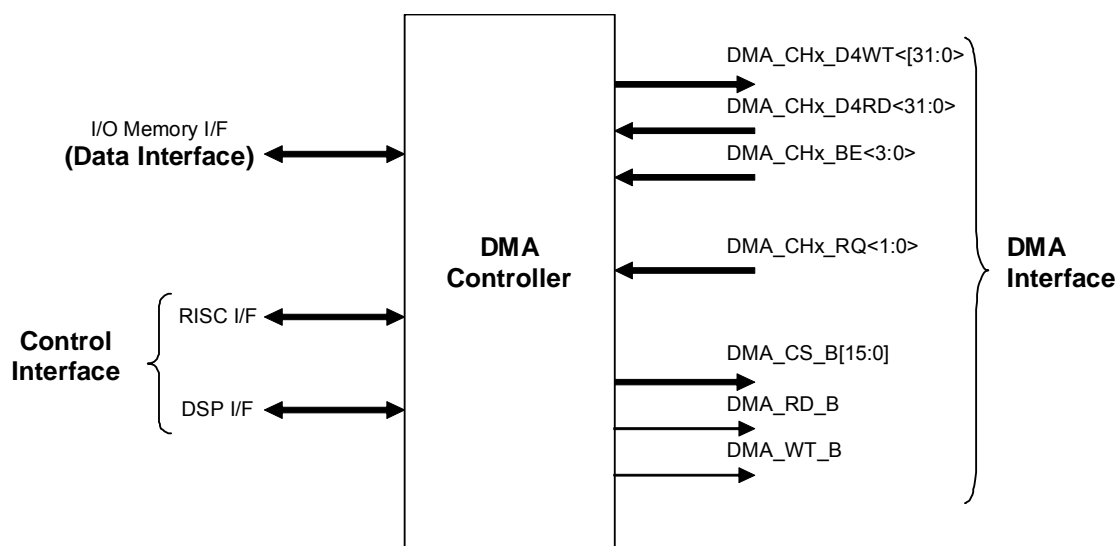



Figure 48. DMA Controller Diagram1

The DMA controller consists of 12 independent DMA channels. Each channel is allocated to a different function described in the figure below:

	Priority
0. CAN Bus input	High
1. CAN Bus output	
2. Video/Camera input	
3. Reserved	
4. NAND Flash input/output	
5. Reserved	
6. CODEC input	
7. CODEC output	
8. UART/USP/CODEC input	
9. UART/USP/CODEC output	
10. UART/USP/CODEC input	
11. UART/USP/CODEC output	Low



Channels 0~1 are shared between CANBUS0 and CANBUS1.

Channels 8~11 are shared between following devices (ports):

- UART0/USP0 (input/output)
- USP1 (input/output)
- USP2 (input/output)
- USP3 (input/output)
- SIB (input/output)
- USP4 (input/output)
- USP5 (input/output)
- AUX (input/output)

Channel 8/10 can be configured as the input DMA channels for any of the above devices (ports);
 Channel 9/11 can be configured as the output DMA channels for any of the above devices (ports).

NOTE: Some PCI devices in Atlas™-II also have their own DMA channels, which are NOT controlled by the DMA Controller. Please refer to the PCI Subsystem section for more details.

Each peripheral has its own FIFO for DMA. The user can setup the FIFO request control register to control when the FIFO generates requests to the DMA Controller. For example, the value in the request control register can be half of the FIFO size. Then once the FIFO is half-full/empty, it will generate a request signal to the DMA controller to start or stop the DMA. If the peripheral needs service sooner, the user can program the register to a higher/lower value, depending on the direction of the DMA.

If the peripheral is a full-duplex device, then it needs to have 2 FIFOs, one for each direction. If the peripheral is half-duplex, then one FIFO is enough. But the peripheral designer needs to be careful in deciding the size of the FIFO to prevent the FIFO overflow or underflow. If an overflow or underflow occurs, it will generate an interrupt.

The DMA controller is intended to relieve the processor of the interrupt overhead in servicing these peripherals via a programmed I/O. But if desired, any or all peripherals can be serviced by the programmed I/O instead of the DMA. Each peripheral is capable of requesting processor service through its own interrupt line.

¹ The x means the DMA channel number (from 0 to 11).

8.3.2 Functional Description

8.3.2.1 DMA Channel Arbitration

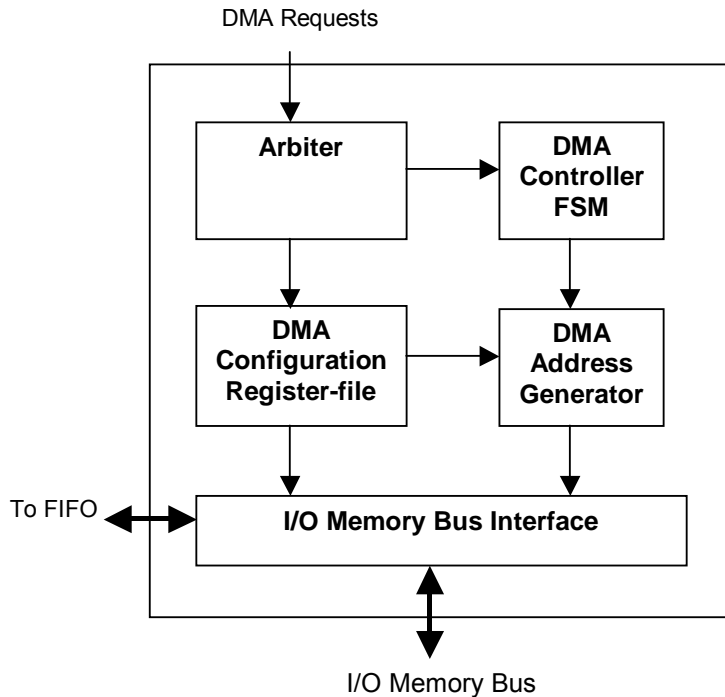


Figure 49. DMA Controller Block Diagram

The DMA controller serves multiple blocks and arbitrates between the different blocks that need to access external memory. Each DMA channel has a two-bit request level. In addition, they all have an inherent priority (as shown above), so that when two channels have the same request level, the channel with the higher priority will win. The priority level can be programmed by the RISC or determined by FIFO fullness, depending on the channel.

Arbitration is a 3-step process:

1. First, requests for each level are determined (by the request signals and whether a DMA is set up for that channel).
2. Second, each level goes to a priority encode to come out with whether there is a request for that channel and a channel to service.
3. Third, the channel for the highest valid request will be serviced.

When the DMA controller detects a request with a higher arbitration priority (based on the above protocol) than the current one, it will interrupt the current DMA, store the location of the last access (i.e. where the stop occurs), and then proceed to service the new request. The interrupted request can resume at where it stopped after its arbitration priority once again becomes the highest. This scheme can be repeated for all the channels active at the same time.

8.3.2.2 DMA FIFO Interface

The FIFOs are a block of SRAM on-chip with a read pointer chasing a write pointer (one may never pass the other). DMA transfer is implemented via the data transfer between the external memory and FIFO(s).

Each peripheral has its own FIFO. The size of FIFO is peripheral dependent. Lower-bandwidth peripherals do not need large FIFO size. The FIFO must provide status flags and interrupts to the host (RISC or DSP) if there is an underflow or overflow.

The fullness of the FIFO is determined by the difference of the two pointers. Since most of the FIFOs in this chip are bi-directional, one must be very careful with both FIFO overflow and underflow in both situations. Make sure that the request levels to the SDRAM controller are set properly so that it can start/stop in time.

The FIFO will interface with both a data producing/consuming peripheral and the DMA controller. A FIFO sends a 2-bit request level to the DMA controller. If that FIFO is serviced, the DMA controller will send it a data valid signal. On the following cycle, the FIFO will either write the 32-bit data onto the data bus if it's writing to SDRAM, or it will read the 32-bit data from the data bus if it's reading from SDRAM.

The request level is determined by two sets of registers (one for reading and one for writing) setting three checkpoints (stop, low, high) that will trigger different request levels.

Table 64. FIFO Requests Level

FIFO Requests	FIFO write to external memory	FIFO read from external memory
2'b00	FIFO is between empty and Stop	FIFO is between full and Stop
2'b01	FIFO is between Stop and Low	FIFO is between Stop and Low
2'b10	FIFO is between Low and High	FIFO is between Low and High
2'b11	FIFO is between High and full	FIFO is between High and empty

There will be two FIFOs for the peripheral that needs to do bi-directional data transfer: one FIFO for each direction. Thus there will be no case that both DMA controller and peripheral block try to write the same FIFO.

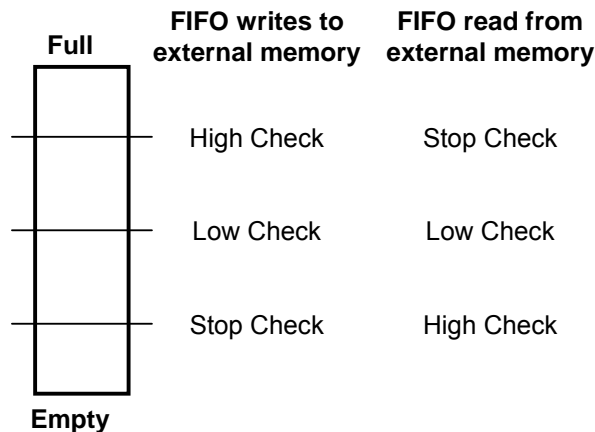


Figure 50. FIFO Request Level Checkpoints

Some peripherals are relatively fast and may need to transfer several words of data at one time. Then it can be programmed in **burst DMA** mode. In burst mode, the DMA will transfer 4 32bit words of data at one time instead of one word in normal mode. **The stop checkpoint of the input FIFO in burst mode should be set greater than or equal to the 4.**

It is necessary to configure a flush-bit to the input FIFO so that the last few words (under stop checkpoint) of the inputs will not be stuck in the FIFO at the end of DMA. When the flush bit is set the

request level will maintain at 2'b01 even the FIFO is below the stop level. This ensures the FIFO is emptied even if new data is not coming in.

Each DMA has its own interrupt flag and enable/disable bit. If the disable bit is asserted, the DMA will be stopped. If a DMA is finished, it will generate an interrupt to the RISC or DSP. The interrupt can also be enabled or disabled.

8.3.2.3 DMA Configuration Register-file

In order to setup a DMA, the programmer needs to know the following:

- 1) Is this a read from memory or write to memory?
- 2) Where to get the data?
- 3) Where to put the data?
- 4) How to get the data (i.e. consecutively, or with jumps in between)?
- 5) What is the priority of this transfer?

These parameters are defined by a series of registers that the programmer must set in order to setup a DMA properly. Each of the channels listed above has its own registers, which means that the DMA's four (4) different channels can co-exist at the same time. Each set of the registers define the following values (**Note: registers for DMA channel 0 is used as an example here. All the channels have the same register definitions**):

- Starting Address – This is a 25-bit address that defines the starting address of the DMA transfer in System Memory (i.e. either SDRAM or SRAM). If SRAM is used, then the top bits are ignored. The starting address is defined in register: *DMA_CH0_ADDR* (**Note: *DMA_CH0_ADDR1* will trigger the start of a DMA and must be set AFTER all the other registers**). The Starting Address is a DWORD address.
- X value – This is a 16-bit value which defines how many consecutive DWORDs to access per line of DMA. This value is in the register *DMA_CH0_XLEN*.
- Y value – This a 16-bit value which defines how many lines of DMA to perform. The actual line number of DMA is Y+1. For example, to DMA one line, Y = 0; to DMA 10 lines, Y = 9. This value is in register *DMA_CH0_YLEN*.
- DMA width – This is a 10-bit value which specifies the spacing between two lines (in DWORD). For example, if the width = 100, X = 10, and Y = 5, then each time when the DMA gets 10th DWORD, it will jump 90 DWORDs to reach the start of the next line. Since width registers are often fixed for a particular application, the Atlas™-1 provides a set of 4 width values (*DMA_WIDTH0*, *DMA_WIDTH1*, etc.) that can be pre-initialized. A particular DMA only needs to specify which of the 4 registers to use in the *DMA_CH0_CTRL* register.

The entire configuration registers of DMA channels are implemented in a register-file. The active channel will load its configuration values into an address counter that changes values while the DMA going. When the DMA is interrupted, the value in the counter will be updated back to the register-file. Thus, the programmer can program the other DMA channels while there is one DMA channel running. But if the programmer writes to the configuration registers of the currently active DMA channel, the value may be invalid if this DMA is updating the address at the same time. Thus it cannot guarantee the value to be valid. So it's not recommended to program the current DMA before it finishes.

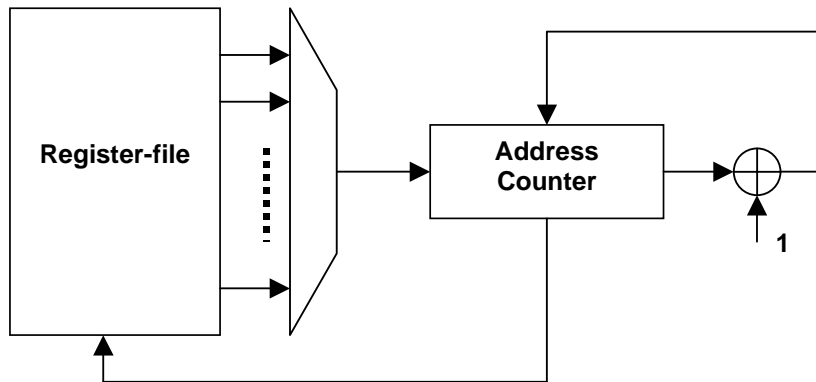


Figure 51. DMA Configuration Register-file

For the detail about register-file, please refer to the section of “DMA Controller Registers”.

8.3.3 Basic DMA Operations

8.3.3.1 Single and Burst DMA

As we specified before, the DMA has two data transfer modes: single transfer mode and burst transfer mode. In single transfer mode, the DMA controller executes one 32-bit word read/write at a time; while in burst mode, 4 32-bit words a time. The user can select the transfer mode by programming one of the register bit in DMA controller.

There is one thing needs to be notified: in burst mode, the DMA address is better to be in 16-DWORD boundary. Otherwise, the transfer will be split into multiple single-word transfers in the system bus (Because the system bus does not allow the non-aligned burst transfer).

8.3.3.2 1-D and 2-D DMA

The Atlas™-II DMA controller supports both 1-D and 2-D DMA. In 2-D DMA, the system memory space is interpreted as a 2-D layout instead of linear 1-D layout. More specifically, the system memory can be considered as multiple data lines. The length of the data line is determined in the user-selected DMA_WIDTH register. The user can specify a data window that the user wants to access by three parameters:

- Start address
- X length
- Y length

The idea of a 2-D DMA is shown in the following diagram.

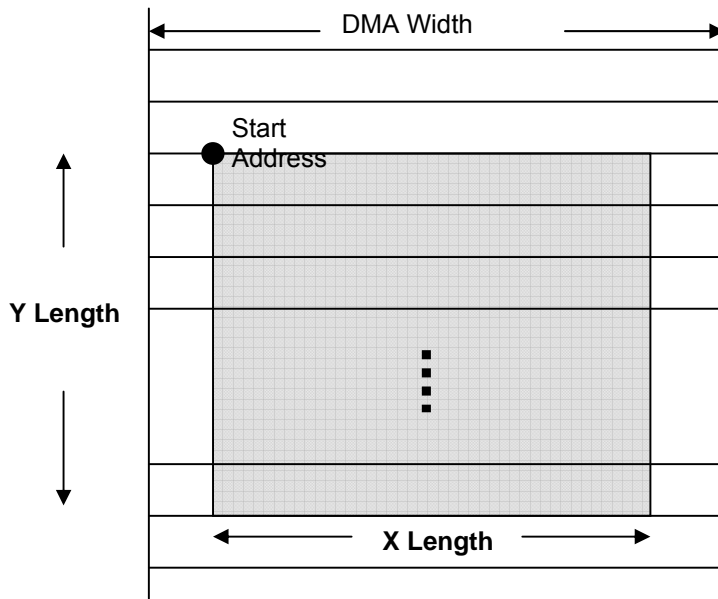


Figure 52. 2-D DMA

If the user specifies the Y length as 0 or X length equals to the DMA Width, then this 2-D DMA will reduce to a 1-D DMA.

If the user configures the X length greater than DMA Width, then the extra data will be wrapped around to the next data line. It will corrupt the DMA transfer for multiple-line 2-D DMA. If it's only a 1-D DMA, then there is no problem. The following diagram shows the wrap around of the extra data in case of X length greater than DMA Width.

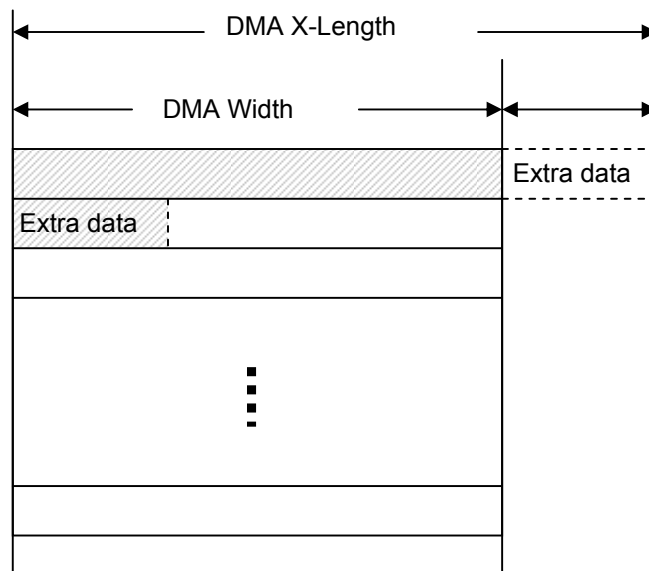


Figure 53. 2-D DMA Wrap Around (X-length > DMA Width)

8.3.3.3 Loop DMA

If X-length is set to 0, it is a special mode of the DMA: loop mode. In loop mode, the DMA will never finish until user force to stop it (by writing a 1'b0 to the **DMA_CH_VALID** register). The DMA will keep looping as described in the following figure:

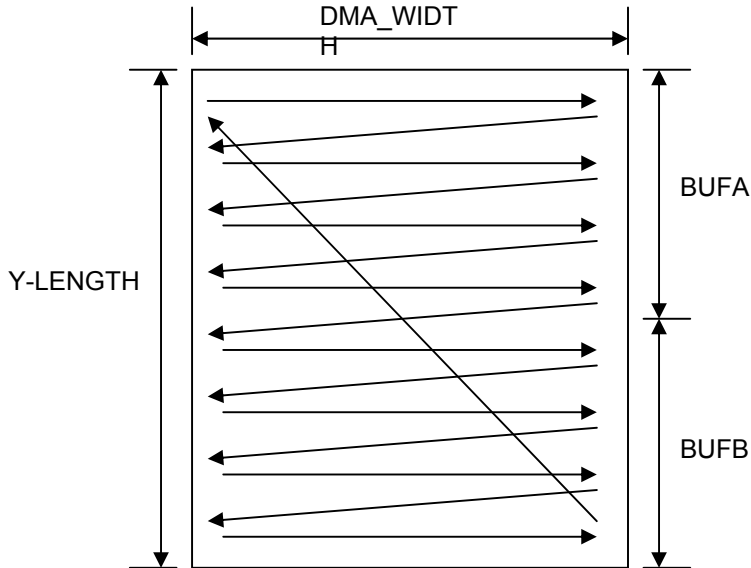


Figure 54. DMA address change if X_LEN=0

As shown in the above figure, the DMA address will keep increasing, until reaching the end of a loop area whose size is defined by (DMA_WIDTH * Y_LENGTH). Then the DMA address will go back to the beginning of this area. If Y_LENGTH or DMA_WIDTH is equal to 0, then the DMA address will not change at all. And the DMA will keep transferring the data to the same DMA address until user force to stop it.

In loop mode, the DMA data region is actually divided into two halves: BUFA and BUFB (i.e. “Buffer A and Buffer B”). The DMA controller will generate interrupt twice during each loop: one time is when the DMA address reaches the end of BUFA; the other time is when the DMA address reaches the end of the BUFB. And of course the interrupt can only be generated when the corresponding interrupt enable (**DMA_INT_EN**) bit is set.

Each half (BUFA & BUFB) has its own buffer valid register bit, which can be programmed by user. The loop DMA will not be really started until the current buffer valid register bit is asserted. For example, if when the DMA goes to the end of BUFA and the valid bit of BUFB is not set, then the DMA will stop at the end of BUFA until the valid bit of BUFB is set.

8.3.4 DMA Controller Registers

Table 65. DMA Controller Register Mapping

RISC Address <11:0>	DSP I/O Address <7:0>	Register	Description
0x0000	0x00~0x01	DMA_CH0_ADDR	DMA channel 0 address register
0x0004	0x02	DMA_CH0_XLEN	DMA channel 0 X-length register
0x0008	0x04	DMA_CH0_YLEN	DMA channel 0 Y-length register
0x000C	0x06	DMA_CH0_CTRL	DMA channel 0 control register
0x0010	0x08~0x09	DMA_CH1_ADDR	DMA channel 1 address register
0x0014	0x0A	DMA_CH1_XLEN	DMA channel 1 X-length register

0x0018	0x0C	DMA_CH1_YLEN	DMA channel 1 Y-length register
0x001C	0x0E	DMA_CH1_CTRL	DMA channel 1 control register
0x0020	0x10~0x11	DMA_CH2_ADDR	DMA channel 2 address register
0x0024	0x12	DMA_CH2_XLEN	DMA channel 2 X-length register
0x0028	0x14	DMA_CH2_YLEN	DMA channel 2 Y-length register
0x002C	0x16	DMA_CH2_CTRL	DMA channel 2 control register
0x0030	0x18~0x19	DMA_CH3_ADDR	DMA channel 3 address register
0x0034	0x1A	DMA_CH3_XLEN	DMA channel 3 X-length register
0x0038	0x1C	DMA_CH3_YLEN	DMA channel 3 Y-length register
0x003C	0x1E	DMA_CH3_CTRL	DMA channel 3 control register
0x0040	0x20~0x21	DMA_CH4_ADDR	DMA channel 4 address register
0x0044	0x22	DMA_CH4_XLEN	DMA channel 4 X-length register
0x0048	0x24	DMA_CH4_YLEN	DMA channel 4 Y-length register
0x004C	0x26	DMA_CH4_CTRL	DMA channel 4 control register
0x0050	0x28~0x29	DMA_CH5_ADDR	DMA channel 5 address register
0x0054	0x2A	DMA_CH5_XLEN	DMA channel 5 X-length register
0x0058	0x2C	DMA_CH5_YLEN	DMA channel 5 Y-length register
0x005C	0x2E	DMA_CH5_CTRL	DMA channel 5 control register
0x0060	0x30~0x31	DMA_CH6_ADDR	DMA channel 6 address register
0x0064	0x32	DMA_CH6_XLEN	DMA channel 6 X-length register
0x0068	0x34	DMA_CH6_YLEN	DMA channel 6 Y-length register
0x006C	0x36	DMA_CH6_CTRL	DMA channel 6 control register
0x0070	0x38~0x39	DMA_CH7_ADDR	DMA channel 7 address register
0x0074	0x3A	DMA_CH7_XLEN	DMA channel 7 X-length register
0x0078	0x3C	DMA_CH7_YLEN	DMA channel 7 Y-length register
0x007C	0x3E	DMA_CH7_CTRL	DMA channel 7 control register
0x0080	0x40~0x41	DMA_CH8_ADDR	DMA channel 8 address register
0x0084	0x42	DMA_CH8_XLEN	DMA channel 8 X-length register
0x0088	0x44	DMA_CH8_YLEN	DMA channel 8 Y-length register
0x008C	0x46	DMA_CH8_CTRL	DMA channel 8 control register
0x0090	0x48~0x49	DMA_CH9_ADDR	DMA channel 9 address register
0x0094	0x4A	DMA_CH9_XLEN	DMA channel 9 X-length register
0x0098	0x4C	DMA_CH9_YLEN	DMA channel 9 Y-length register
0x009C	0x4E	DMA_CH9_CTRL	DMA channel 9 control register
0x00A0	0x50~0x51	DMA_CH10_ADDR	DMA channel 10 address register
0x00A4	0x52	DMA_CH10_XLEN	DMA channel 10 X-length register
0x00A8	0x54	DMA_CH10_YLEN	DMA channel 10 Y-length register
0x00AC	0x56	DMA_CH10_CTRL	DMA channel 10 control register
0x00B0	0x58~0x59	DMA_CH11_ADDR	DMA channel 11 address register
0x00B4	0x5A	DMA_CH11_XLEN	DMA channel 11 X-length register
0x00B8	0x5C	DMA_CH11_YLEN	DMA channel 11 Y-length register
0x00BC	0x5E	DMA_CH11_CTRL	DMA channel 11 control register
Others	-	-	Reserved
0x0100	0x80	DMA_WIDTH0	DMA width 0 register
0x0104	0x81	DMA_WIDTH1	DMA width 1 register
0x0108	0x82	DMA_WIDTH2	DMA width 2 register
0x010C	0x83	DMA_WIDTH3	DMA width 3 register
0x0110	0x84	DMA_CH_VALID	DMA channel valid register
0x0114	0x85	DMA_CH_INT	DMA channel interrupt register

0x0118	0x86	DMA_INT_EN	DMA interrupt enable register
0x011C	-	DMA_CH_DSP_CTRL	DMA channel DSP control register
0x0120	0x87~88	DMA_CH_LOOP_CTRL	DMA channel loop control register
Others	-	-	Reserved

- **DMA Channel <11:0> Address Register (DMA_CH<11:0>_ADDR)**

The DMA channel start address is in the 32-bit D-word boundary. This must be the last register to be set among all the DMA configuration registers. Setting this register will start the DMA.

Bit	Name	Default	Description
24:0 (R/W)	A<24:0>	25'h0	DMA start address in 32-bit D-word boundary.
31:25	-	7'h0	Reserved

NOTE1: For burst-mode DMA, please set the start address to be in 16-byte boundary. Otherwise, the burst will be cut into multiple single transfers and the efficiency is much lower.

NOTE2: For loop-mode DMA, set the start address will not start the DMA at once. The user needs to set the *DMA_CH_LOOP_CTRL* to start the DMA. But before start a new DMA, user still needs to configure this start address register to clear the internal status of last DMA.

- **DMA Channel <11:0> X-Length Register (DMA_CH<11:0>_XLEN)**

The DMA X-length is in 32-bit D-word boundary. The value specifies the number of D-words transferred in each line. **The maximum length is 2047 DWORDs¹.**

In burst mode, X-length should be in 16-byte boundary normally; but it is allowed to be set to any value. For example, if X-length = 5, then the DMA controller is actually doing 2 bursts. However, in the last burst, the DMA controller will only do one DWORD data transfer to/from the peripheral.

For 1-D DMA, it is not so important because the last data transfer in the whole DMA will not affect the peripheral FIFO.

But in 2-D DMA when X-length is not in the 16-byte boundary, then the last burst of every line will only do a 1 DWORD data transfer. Thus, the DMA controller will not do extra data transfer and will not affect the data sequence in the peripheral FIFO. **(Note: From the system memory's point of view, it will still see 2 bursts on each line, (i.e., the 5 DWORD data transfer of each line will take 8-DWORD's memory space.)**

Bit	Name	Default	Description
11:0 (R/W)	XL<11:0>	12'h0	DMA X-length in 32-bit DWORD boundary.
31:12	-	20'h0	Reserved

- **DMA Channel <11:0> Y-Length Register (DMA_CH<11:0>_YLEN)**

The DMA Y-length specifies the number of lines in DMA transfer. The number of the lines in DMA transfer is Y-Length + 1. The maximum number of line can be 2048. To set Y-length to 0 has the effect that doing 1-D DMA.

Bit	Name	Default	Description
-----	------	---------	-------------

¹ It should be less than the value of DMA width register.

11:0 (R/W)	YL<11:0>	12'h0	DMA Y-length.
31:12	-	20'h0	Reserved

- **DMA Channel <11:0> Control Register (DMA_CH<11:0>_CTRL)**

Bit	Name	Default	Description
1:0 (R/W)	WS<1:0>	2'h0	DMA width register select signal. 00: select DMA_WIDTH0; 01: select DMA_WIDTH1; 10: select DMA_WIDTH2; 11: select DMA_WIDTH3.
2 (R/W)	DIR	1'b0	DMA direction. 1: DMA from SDRAM to peripheral FIFO; 0: DMA from peripheral FIFO to SDRAM.
3 (R/W)	BURST	1'b0	DMA transfer mode. 1: DMA is in burst transfer mode; 0: DMA is in single transfer mode.
31:4	-	28'h0	Reserved

- **DMA Width Registers (DMA_WIDTH0~3) – RISC: 0x100~0x10C, DSP: 0x80~0x83**

There are two separate sets of DMA Width registers in the Atals-1, one for RISC, the other for the DSP. It depends on the setting of the *DMA_CH_DSP_EN* to select which sets to use. Each set has 4 Width registers. Each DMA channel can select to use any one of these 4 Width registers.

To enable the correct 2-D DMA, the DMA Width register has to be correctly set. And the value of the DMA Width register has to be greater than or equal to the X-length. Otherwise, the data will be overlapped.

Bit	Name	Default	Description
11:0 (R/W)	WIDTH<11:0>	12'h0	DMA Width.
31:12	-	20'h0	Reserved

- **DMA Channel Valid Register (DMA_CH_VALID) – RISC: 0x110, DSP: 0x84**

A one in this register means the corresponding DMA channel is active. The user can write a 1 to each bit of this register to force the DMA of that channel to stop.

Bit	Name	Default	Description
11:0 (R/W)	VL<11:0>	12'h0	DMA channel valid bits. This bit will be automatically set to 1 if the START_ADDR register is written. Write a 1 to each bit will stop that channel's DMA.
31:12	-	20'h0	Reserved

- **DMA Channel Interrupt Register (DMA_CH_INT) – RISC: 0x114, DSP: 0x85**

After each DMA is finished, it will generate an interrupt bit in the corresponding bit in this register. The RISC or DSP can read this register to check which DMA is finished. If the RISC or DSP writes a 1 to that bit, the interrupt bit will be cleared.

Bit	Name	Default	Description
-----	------	---------	-------------

11:0 (R/W)	INT<11:0>	12'h0	DMA interrupt bits.
31:12	-	20'h0	Reserved

- **DMA Interrupt Enable Register (DMA_INT_EN) – RISC: 0x118, DSP: 0x86**

There are two separate interrupt enable registers, one for RISC and the other for DSP. The DMA channel can only generate interrupts when the corresponding bits in this register and *DMA_CH_DSP_EN* are set correctly.

Bit	Name	Default	Description
11:0 (R/W)	EN<11:0>	12'h0	DMA interrupt enable bits. 1: The DMA interrupt is enabled; 0: The DMA interrupt is disabled.
31:12	-	20'h0	Reserved

- **DMA Channel DSP Control Register (DMA_CH_DSP_CTRL) – RISC: 0x11C**

This register is accessible to the RISC only.

Bit	Name	Default	Description
11:0 (R/W)	DSP_EN<11:0>	12'h0	DMA channel DSP control enable bits. 1: The DMA channel is controlled by DSP; 0: The DMA channel is controlled by RISC.
31:12	-	20'h0	Reserved

- **DMA Channel Loop Control Register (DMA_CH_LOOP_CTRL) – RISC: 0x120, DSP: 0x87~88**

Just as the *DMA_WIDTH* registers and *DMA_INT_EN* register, there are actually two separate *DMA_CH_LOOP_CTRL* registers: one for RISC, the other for DSP. It depends on the current channel is controlled by RISC or DSP to decide which *DMA_CH_LOOP_CTRL* register is being used.

Bit	Name	Default	Description
11:0 (R/W)	BUFA_VA LID<11:0>	12'h0	1: The first section of the loop area (BUFA) is valid. 0: The first section of the loop area (BUFA) is invalid.
15:12	-	4'h0	Reserved
27:16 (R/W)	BUFB_VA LID<11:0>	12'h0	1: The second section of the loop area (BUFB) is valid. 0: The second section of the loop area (BUFB) is invalid.
31:28	-	4'h0	Reserved

8.4 NAND Flash Interface

8.4.5 Overview

The NAND Flash can be used as a boot-loader during boot-up, or be used as a simple storage device (for both program and data). When the NAND Flash is used as a boot-loader, the boot code will be stored in the first physical block of the NAND Flash and the rest of NAND Flash also can be used as simple storage device.

The NAND Flash controller supports all types of NAND Flash/ SmartMedia card. It has an independent FIFO and a DMA channel for data transfer to/from system memory.

The NAND Flash controller supports five types of data transfer:

- DMA read data from FIFO
- DMA write data to FIFO
- RISC read data from FIFO
- RISC write data to FIFO
- RISC directly read data from NAND Flash

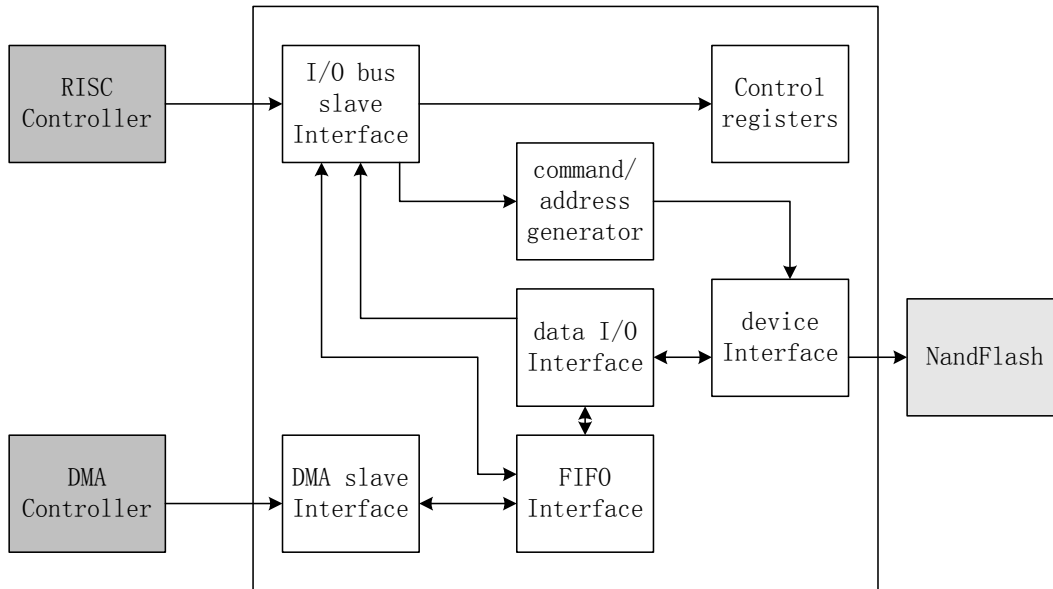


Figure 55. NAND Flash Controller Block Diagram

In normal read/write mode, RISC write command/address to the NAND Flash interface to start read/write cycle, data can be exchanged between DMA controller and FIFO of NAND Flash interface, or between RISC and FIFO of NAND Flash interface.

In directly read mode (used when boot-up from NAND Flash), RISC sends address to I/O address bus, and NAND Flash interface will translate that address and generate the read command and address sequence to NAND Flash automatically. When data is ready on the I/O data bus, NAND Flash interface will assert data ready signal to RISC and RISC can fetch that data from the I/O data bus.

The NAND Flash controller supports 256 bytes or 512 bytes ECC (Error Correction Code) for 8 bit NAND flash, 256 words ECC for 16 bit NAND flash. The ECC is generated by hardware automatically when data are written into the NAND Flash.

If system boots up from NAND Flash, the NAND Flash is mapped to the bottom of system address (starts from 0x0000_0000). Usually a boot-loader code putted in the block 0 of the NAND flash. RISC will execute the boot-loader first, in which it will configure the SDRAM and initialize the hardware properly. Then it will go to a loop to copy the system code page by page.

8.4.6 Pin Description

The NAND Flash interface shares pins with Video Input Port or ROM interface. User can use Mode Configuration Pins to choose pin sharing method in system PCB design.

Table 66. NAND Flash Interface Signal Descriptions

Pin Name	Direction	Description
----------	-----------	-------------

DF_CLE	Output	Command Latch Enable
DF_ALE	Output	Address Latch Enable
DF_AD<15:0>	Input/Output	Address/Data
DF_WE_B	Output	Write Enable
DF_RD_B	Output	Read Enable
DF_RY_BY	Input	Ready/Busy output (Schmitt Triggered)
DF_CS_B<3:0>	Output	Chip Enable

8.4.7 NAND Flash Interface Registers

Table 67. NAND Flash Interface Register Mapping

RISC Address <11:0>	Register	Description
0x00	Reserved	
0x04	SM_WAIT	Wait state for read/write NAND Flash
0x08	SM_BANK_SEL	Chip select
0x0C	SM_ADD_NUM	byte number for NAND Flash address
0x10	SM_CMD	Command to be sent to NAND Flash
0x14	SM_LOW_ADDR	Low 4 bytes of address to NAND Flash
0x18	SM_HIGH_ADDR	High bytes of address to NAND Flash (needed only by NAND Flash more than 256MB)
0x1C	SM_PAGE_SIZE	Page size of NAND Flash
0x20	SM_INT_EN	Interrupt mast register
0x24	SM_INT_STATUS	Interrupt status register
0x28	SM_CTRL	Control Register of NAND Flash
0x2C	SM_ECC_SET	Hard ware ECC set register
0x30	SM_ECC_AREA1	Hard ware ECC result store register
0x34	SM_ECC_AREA2	Hard ware ECC result store register
0xF00	SM_DMA_IO_CTRL	Control bit for read/write NAND Flash
0xF04	SM_DMA_IO_LEN	DMA length
0xF08	SM_FIFO_CTRL	Control bit for FIFO
0xF0C	SM_FIFO_LEVEL_CHK	FIFO level check
0xF10	SM_FIFO_OP	Reset and start bit for FIFO
0xF14	SM_FIFO_STATUS	FIFO data status
0xF18	SM_FIFO_DATA	RISC read FIFO from this register

- **NAND Flash Wait State Register (SM_WAIT) – 0x04**

Bit	Name	Default	Description	S MDF timing
3:0 (R/W)	RD_PULSE	4'b0000	The pulse width of read signal of NAND Flash 0: one IOCLK; 1: two IOCLK, ...	30 ns
7:4 (R/W)	WT_PULSE	4'b0000	The pulse width of write signal of NAND Flash 0: one IOCLK; 1: two IOCLK ...	25 ns
11:8 (R/W)	WAIT	4'b0010	The width between WE high to Busy 0:one IOCLK; 1:two IOCLK, ...	100 ns
12 (R/W)	RD_WT_HI	1'b0	The write or read signal high hold	15 ns

			time 0: one IOCLK, 1: two IOCLK, ...	
31:13	–	19'h0	Reserved	

- **NAND Flash Bank Select Register (SM_BANK_SEL) – 0x08**

Bit	Name	Default	Description
3:0 (R/W)	BANK_SEL<3:0>	4'b1110	Select bank of NAND Flash 1110: first chip; 1101: second chip
31:4	–	28'h0	Reserved

- **NAND Flash Address Mode Register (SM_ADD_NUM) – 0x0C**

Bit	Name	Default	Description
2:0 (R/W)	ADD_MODE<2:0>	3'b100	Number of address bytes 00: 0 bytes address; 01: 1 bytes address 11: 3 bytes address; 100: 4 bytes address
31:3	–	29'h0	Reserved.

- **NAND Flash Command Register (SM_CMD) – 0x10**

Write this register will start FSM. If there is no second command, bit[15:8] must be 8'h0, or it will cause the FSM run into wrong state.

Bit	Name	Default	Description
7:0 (R/W)	CMD<7:0>	8'h0	Command to be written to NAND Flash
15:8 (R/W)	CMD2<7:0>	8'h0	Second command write to NAND Flash (used in case of command - - - address - - - command, see Samsung's NAND Flash more than 128MB)
31:16	–	16'h0	Reserved.

- **NAND Flash Low Address Register (SM_LOW_ADDR) – 0x14**

Bit	Name	Default	Description
31:0 (R/W)	LO_ADD<31:0>	32'h0	Start address to data flash

- **NAND Flash High Address Register (SM_HIGH_ADDR) – 0x18**

Bit	Name	Default	Description
7:0 (R/W)	HI_ADD<7:0>	8'h0	Start address to data flash
31:8	–	24'h0	Reserved.

- **NAND Flash Page Size Register (SM_PAGE_SIZE) – 0x1C**

The default value of **PAGE_SIZE** is 528 bytes, which is the NAND Flash page size. User need not change it if NAND Flash has no change. The bit2~bit0 will be ignored, i.e. this value must be DWORD size.

Bit	Name	Default	Description
11:0 (R/W)	PAGE_SIZE	12'h210	528 bytes per page

31:12	–	20'h0	Reserved.
-------	---	-------	-----------

- **NAND Flash Interrupt Enable Register (SM_INT_EN) – 0x20**

Default value is 0. Set the bit to 1 will enable the interrupt, but the according bit in *SM_INT_STATUS* will be set no matter whether the interrupt is enabled.

Bit	Name	Default	Description
0	-	-	Reserved
1 (R/W)	WT_ADD_DONE_EN	1'b0	Interrupt raised when write address to smartmedia is done
2 (R/W)	IO_DMA_DONE_EN	1'b0	Interrupt raised when I/O or DMA transfer is done.
3 (R/W)	FIFO_THRESHOLD_EN	1'b0	Interrupt raised when the number of data in the FIFO reach the threshold
31:4		28'h0	Reserved.

- **NAND Flash Interrupt Status Register (SM_INT_STATUS) – 0x24**

When an interrupt occur, the corresponding bit will be set to 1 (even the interrupt bit is not enable) automatically. Write a 1'b1 will clear the interrupt bit.

Bit	Name	Default	Description
0	-	-	Reserved
1 (R/W)	WT_ADD_DONE	1'b0	Interrupt raised when write command and address to smartmedia is done.
2 (R/W)	IO_DMA_DONE	1'b0	Interrupt raised when I/O or DMA transfer is done.
3 (R/W)	FIFO_THRESHOLD	1'b0	Interrupt raised when the number of data in the FIFO reach the threshold.
31:4		28'h0	Reserved.

- **NAND Flash Control Register (SM_CTRL) – 0x28**

This register is used for power-on auto-read in WINCE boot. After reset this register is set to 1'b1, this enables RISC to directly read NAND Flash (acting like ROM). When using FIFO to read/write datam this bit must be set 1'b0.

Bit	Name	Default	Description
0 (R/W)	DIRECT_READ	1'b1	Set 1 will enable I/O directly read NAND Flash
1 (R/W)	DATA_WIDTH	1'bx	Set 1 will enable 16 - bit NAND Flash Set 0 will enable 8 - bit NAND Flash
2	-	-	Reserved
3	NAND_MODE	1'bx	Set 1 will enable large page support Set 0 will disable large page support
4 (R/W)	NAND_BOOT_CAM	1'bx	Set 1 will request Cam pin Set 0 will request rom pin
31:5	-	27'h0	Reserved.

NOTE: The default values of bit 1, 3, and 4 are determined by the external pull-up/down resistor. Please refer to the section of “Mode Configuration Pins”.

- **NAND Flash ECC Setting Register (SM_ECC_SET) - 0x2C**

Bit	Name	Default	Description
1 (R/W)	HW_ECC	1'b0	Set 1 will enable hardware ECC
2 (R/W)	HW_ECC_RST	1'b0	Set 1 will reset hardware ECC register
3 (R/W)	HALF_PAGE	1'b1	Only used in 8-bit NAND Flash. Set 1 will enable 256byte ECC Set 0 will enable 512byte ECC
31:4		29'h0	Reserved.

- **NAND Flash ECC Area1 Register (SM_ECC_AREA1) - 0x30**

Bit	Name	Default	Description
32:0 (R)	ECC_AREA1	32'h0	Hard ware ECC result store register

- **NAND Flash ECC Area2 Register (SM_ECC_AREA2) - 0x34**

Bit	Name	Default	Description
32:0 (R)	ECC_AREA2	32'h0	Hard ware ECC result store register

- **NAND Flash DMA/IO Control Register (SM_DMA_IO_CTRL) - 0xF00**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b1	1 for I/O mode, 0 for DMA mode.
1 (R/W)	IO_DMA_RW	1'b1	1: read from peripheral 0: write to peripheral
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the data_length set at the peripheral side doesn't match the dword size set in the DMA control.
3 (R/W)	RW_ENDIAN	1'b0	1: big end write/read 0: little end write/read
4 (R/W)	NO_RDWT	1'b0	1: write command and address to smartmedia and did not read/write data
31:5 (R/W)			Reserved

- **NAND Flash DMA/IO Length Register (SM_DMA_IO_LEN) - 0xF04**

The smallest data size is DWORD, i.e. bit1 and bit 0 of this register will be ignore

Bit	Name	Default	Description
15:0(R/W)	DATA_LEN	16'h0	The byte length of a DMA or I/O transfer. If set to zero, the I/O or DMA transfer works continuously until it is stopped.
31:16		16'h0	Reserved.

- **NAND Flash FIFO Control Register (SM_FIFO_CTRL) - 0xF08**

In read mode when FIFO_STATUS_REG [5:2]>=FIFO_CTRL_REG [7:4], interrupt will be generated.
 In write mode when FIFO_STATUS_REG [5:2]<=FIFO_CTRL_REG [7:4], interrupt will be generated.

Bit	Name	Default	Description
1:0 (R/W)	FIFO_WIDTH	2'b00	Reserved.
7:2 (R/W)	FIFO_THRESHOLD	6'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
31:8		24'h0	Reserved.

- **NAND Flash FIFO Level Check Register (SM_FIFO_LEVEL_CHK) - 0xF0C**

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC	4'h0	FIFO stop-check in DWORD length.
9:4		6'h0	Reserved.
13:10 (R/W)	FIFO_LC	4'h0	FIFO low-check in DWORD length.
19:14		6'h0	Reserved.
23:20 (R/W)	FIFO_HC	4'h0	FIFO high-check in DWORD length.
31:24		8'h0	Reserved.

- **NAND Flash FIFO Operation Register (SM_FIFO_OP) - 0xF10**

Bit	Name	Default	Description
0 (R/W)	FIFO_START	1'b1	Start the read/write transfer when this bit is declared.
1 (R/W)	FIFO_RESET	1'b0	Internally link to fifo_start_ini. Set to 1 to stop the fifo and reset the fifo internal status, including the relevant interrupt status. Set to 0 in normal operation.
31:2		30'h0	Reserved.

- **NAND Flash FIFO Status Register (SM_FIFO_STATUS) - 0xF14**

Bit	Name	Default	Description
5:0 (R)	FIFO_LEVEL	6'h0	The byte count of the valid data in the FIFO. In case FIFO is full, the value of this register is 0, thus user must concatenate FIFO_FULL bit with this value to determine the actual data count in the FIFO.
6 (R)	FIFO_FULL	1'b0	Fifo full status, the fifo is full when read out as 1. This bit is concatenated with FIFO_LEVEL to be the actual FIFO data count.
7 (R)	FIFO_EMPTY	1'b1	Fifo empty status, (FIFO_FULL or FIFO_LEVEL) == 0
31:8		24'h0	Reserved.

- **NAND Flash FIFO Data Register (SM_FIFO_DATA) - 0xF18**

Bit	Name	Default	Description
31:0 (R)	FIFO_DATA	32'h0	The fifo read/write data register.

8.5 Video Input Port

8.5.1 Overview

This block interfaces with the DVD (MPEG-2) decoder, TV decoder, CMOS imaging sensor processes the 16-bit input from these device through a series of hardwired logic. The image is framed by HSYNC and VSYNC signals and is clocked by PIXCLK. The sync signals and pixel clock can be inverted as needed. Two counters will count the X and Y positions, respectively, of the current pixel. The user can define an active region by specifying the beginning and ending X and Y coordinates. In the active region, each 16-bit sample will be combined to a 32-bit value. This value then sent to the FIFO to be DMA'ed to the system memory or read by RISC or DSP.

This video input port supports image scaling, and translation for picture-in-picture or different video overlay display schemes.

Finally, the Video Input Port can generate an interrupt to the controller at a specified line during each frame. This may be useful for video application.

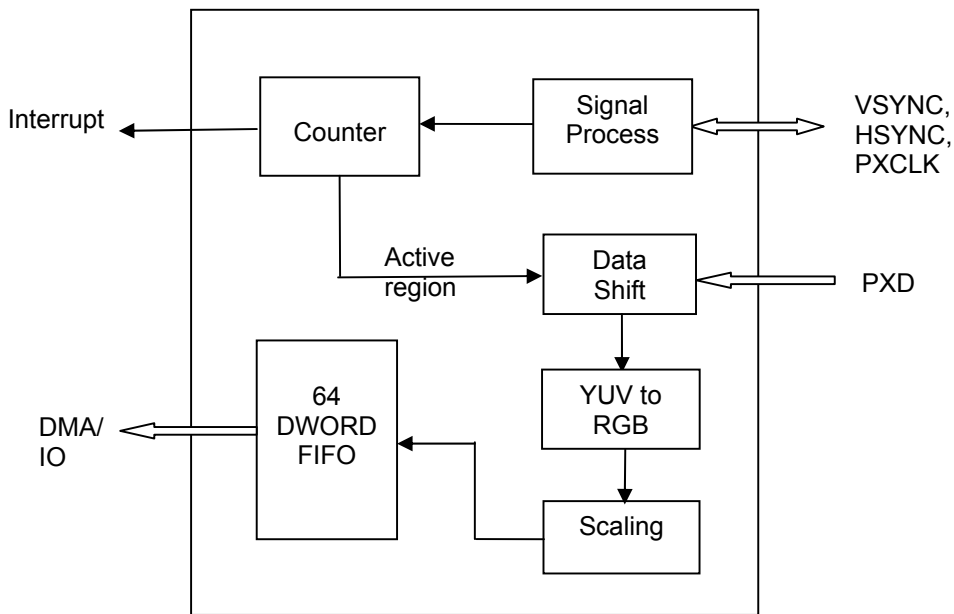


Figure 56. Block Diagram of Video Input Port

8.5.2 Functional Description

8.5.2.1 Control Signals

The Video Input Port can work in master mode or slave mode with three main control signals: **PIXCLK**, **HSYNC** and **VSYNC**. PIXCLK toggles once for every pixel. HSYNC toggles once at the beginning of each line, and VSYNC toggles once at the beginning of each image frame. These signals define the size of one “frame” of picture.

The programmer can select VSYNC, HSYNC and PXCLK generate internal or provided by external device by setting register *CAM_CTRL*. In master mode, the programmer must select VSYNC, HSYNC generate internal after setting the HSYNC and VSYNC period to provide a control interface for external device. PXCLK can also be provided internally by setting register *CAM_PXCLK_CTRL* and *CAM_CTRL*. In slave mode, these three signals are provided by external devices.

8.5.2.2 Data Signals

The Video Input Port supports following data format:

- 1 Up to 16-bit RGB data.
- 2 YUV or YCrCb 4:2:2, 8 bit and 16 bit.

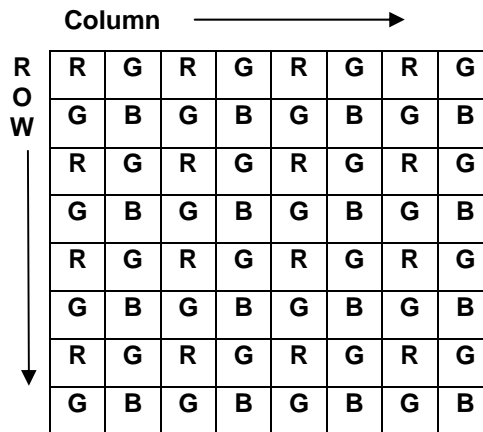


Figure 57. 16-bit RGB Data Format

Table 68. YUV/YCrCb 4:2:2 16 bit format

Data Bus	Pixel byte sequence							
PXD[15:8]	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8
PXD[7:0]	U1	V1	U2	V2	U3	V3	U4	V4

Table 69. YUV/YCrCb 4:2:2 8 bit format

Data Bus	Pixel byte sequence							
PXD[7:0]	Y1	U1	Y2	V1	Y3	U2	Y4	V2

8.5.2.3 Pixel Data Processing

Input pixel data will be processed as the following steps.

- 1 The pixel data will combine 2 16-bit pixel data to a 32-bit data.
- 2 Reorganize the 32-bit data as CAM_SHIFT defined.

- 3 If the input data is YUV/YCrCb format, and YUV to RGB conversion is enabled, receiving data will be converted to RGB format data.
- 4 According the LCD display mode, the RGB data will be reorganized in 656,565,655 and 888 pattern.
- 5 Scaling the picture as defined in CAM_CTRL register.
- 6 Write the final data to the FIFO.

8.5.2.4 Pixel Data Shifting

The Video Input Port supports 2 format pixel data, 16-bit and 8-bit. In 16-bit mode, camera interface will receive whole the 16-bit pixel data whatever whether it is connected on the board. In 8-bit mode, camera interface will select 8 of 16-bit pixel data as a valid input and combine 4 8-bit data to a 32-bit. Following table gives an example:

Table 70. Shift example

SHIFT_NUM	3'b000	3'b011	3'b010	3'b001
ADDR	{P1,P2}	{P1[9:2],P2[9:2], P3[9:2],P4[9:2]}	{P1[8:1],P2[8:1], P3[8:1],P4[8:1]}	{P1[7:0],P2[7:0], P3[7:0],P4[7:0]}
ADDR+1	{P3,P4}	{ P5[9:2],P6[9:2], P7[9:2],P8[9:2]}	{ P5[8:1],P6[8:1], P7[8:1],P8[8:1]}	{ P5[7:0],P6[7:0], P7[7:0],P8[7:0]}
ADDR+2	{P5,P6}	{ P9[9:2],P10[9:2], P11[9:2],P12[9:2]}	{ P9[8:1],P10[8:1], P11[8:1],P12[8:1]}	{ P9[7:0],P10[7:0], P11[7:0],P12[7:0]}
ADDR+3	{P7,P8}	{ P13[9:2],P14[9:2], P15[9:2],P16[9:2]}	{ P13[8:1],P14[8:1], P15[8:1],P16[8:1]}	{ P13[7:0],P14[7:0], P15[7:0],P16[7:0]}
ADDR+4	{P9,P10}	{ P17[9:2],P18[9:2], P19[9:2],P20[9:2]}	{ P17[8:1],P18[8:1], P19[8:1],P20[8:1]}	{ P17[7:0],P18[7:0], P19[7:0],P20[7:0]}

8.5.2.5 YUV to RGB conversion

The Video Input Port supports two types of format conversions, from YUV to RGB and from YCrCb to RGB, it works on following formulas:

YUV to RGB conversion formula:

$$\begin{aligned}
 R &= Y * C1 + U * C2 + V * C3; \\
 G &= Y * C4 - U * C5 - V * C6; \\
 B &= Y * C7 - U * C8 + V * C9;
 \end{aligned}$$

YCrCb to RGB conversion formula:

$$\begin{aligned}
 R &= Y * C1 + Cb * C2 + Cr * C3 - \text{Offset1}; \\
 G &= Y * C4 - Cb * C5 - Cr * C6 + \text{Offset2}; \\
 B &= Y * C7 + Cb * C8 + Cr * C9 - \text{Offset3};
 \end{aligned}$$

The output RGB data will be rounded and fixed as following formula:

$$\begin{aligned}
 R_OUT &= ((R + 8'h80) >> 8) \& 8'hff; \\
 G_OUT &= ((G + 8'h80) >> 8) \& 8'hff; \\
 B_OUT &= ((B + 8'h80) >> 8) \& 8'hff;
 \end{aligned}$$

NOTE: Unlike the YUV_RGB engine on PCI bus, the YUV to RGB conversion logic in Video Input Port is only applied to the Video input stream data; it cannot be applied to other data transfers inside of the chip.

Following table gives the reference value:

Table 71. YUV2RGB Conversion Coefficient

Color A	Y	Coef*256	HEX	Y	Coef*256	HEX
Color B	U			Cr		
Color C	V			Cb		
Color D	R			R		
Color E	G			G		
Color F	B			B		
C1	1.140			291.84		
C2	0	0	10'h000	1.596	408.576	10'h198
C3	1.140	291.84	10'h124	0	0	10'h000
COEF1	32'h12400124			32'h12A00198		
Offset1	0	0	10'h000	223	223	10'h0DF
C4	1	256	10'h100	1.164	297.984	10'h12A
C5	0.394	100.864	10'h065	0.813	208.128	10'h0D0
C6	0.581	148.736	10'h095	0.392	100.352	10'h064
COEF2	32'h10019495			32'h12A190D0		
Offset2	0	0	10'h000	136	136	10'h088
C7	1	256	10'h100	1.164	297.984	10'h12A
C8	2.032	520.129	10'h208	0	0	10'h000
C9	0	0	10'h000	2.017	516.352	10'h204
COEF3	32'h10082000			32'h12A81000		
Offset3	0	0	10'h000	277	277	10'h115
OFFSET	32'h00000000			32'h115220DF		

To directly display the received picture in the LCD screen, the RGB data will be combined to 5:6:5/5:5:6/6:5:5/8:8:8 formats as the LCD interface defined after the conversion. Following formulas give the definition.

5:6:5: output_data[15:11] = R[7:3]; output_data[10:5] = G[7:2]; output_data[4:0] = B[7:3];
 5:5:6: output_data[15:11] = R[7:3]; output_data[10:6] = G[7:3]; output_data[5:0] = B[7:2];
 6:5:5: output_data[15:10] = R[7:2]; output_data[9:5] = G[7:3]; output_data[4:0] = B[7:3];
 8:8:8: output_data[24:16] = R[7:0]; output_data[15:8] = G[7:0]; output_data[7:0] = B[7:0];

In 5:6:5/5:5:6/6:5:5 format, 2 combined 16-bit data will be recombined to a 32-bit data. In 8:8:8 mode, the combined 24-bit data will be zeroed in the upper 8-bit and recombined to a 32-bit data.

8.5.2.6 Scaling

The Video Input Port supports 1:2, 1:4 and 1:8 scaling function in colour and row direction. For Bayer RGB data mode, it scales the pixel data as following table.

R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B

Figure 58. Contraction in Bayer RGB Mode with 1:2 Ratio in Row and Column

R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B
R	G	R	G	R	G	R	G	R	G	R	G	R	G	R	G
G	B	G	B	G	B	G	B	G	B	G	B	G	B	G	B

Figure 59. Contraction in Bayer RGB Mode with 1:4 Ratio in Row and Column

For YUV/YCrCb mode, it first converts the YUV/YCrCb data to RGB data, then scales the RGB data as following table.

R G	R G	R G	R G	R G	R G	R G	R G
B	B	B	B	B	B	B	B
R G	R G	R G	R G	R G	R G	R G	R G
B	B	B	B	B	B	B	B
R G	R G	R G	R G	R G	R G	R G	R G
B	B	B	B	B	B	B	B
R G	R G	R G	R G	R G	R G	R G	R G
B	B	B	B	B	B	B	B
R G	R G	R G	R G	R G	R G	R G	R G
B	B	B	B	B	B	B	B

Figure 60. Contraction in YUV/YCrCb Mode (YUVRGB=1) with 1:2 ratio in Row and Column

R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B

Figure 61. Contraction in YUV/YCrCb Mode (YUVRGB=1) with 1:4 ratio in Row and Column

8.5.2.7 Active Region

The Video Input Port always keeps track of the number of pixel clocks after each HSYNC (X-count) and the number of lines after VSYNC (Y-count). (The current X-count and Y-count can be read from registers CAM_COUNT). The most important for the programmer to do is to define active region, in which the image is valid and the pixel data is store to system memory via a FIFO. The pixels outside of the active region are ignored. The active region is defined by defining an X-start and X-end, and a Y-start and Y-end. In all the lines between Y-start and Y-end, if the pixel is between X start and X-end, then the pixel is valid. These values are defined using the registers CAM_START and CAM_END.

Another thing the user can set is an interrupt line. The Video Input Port will then generate an interrupt whenever it reaches that line. Usually this interrupt line is defined before or after the active region, and can be used to as a frame interrupt for video applications or do some setup before or after an image capture. The interrupt line is set using the CAM_YINT register. Of course, the interrupt must be enabled in the INT_RISC_MASK register, and the interrupt is edge-triggered.

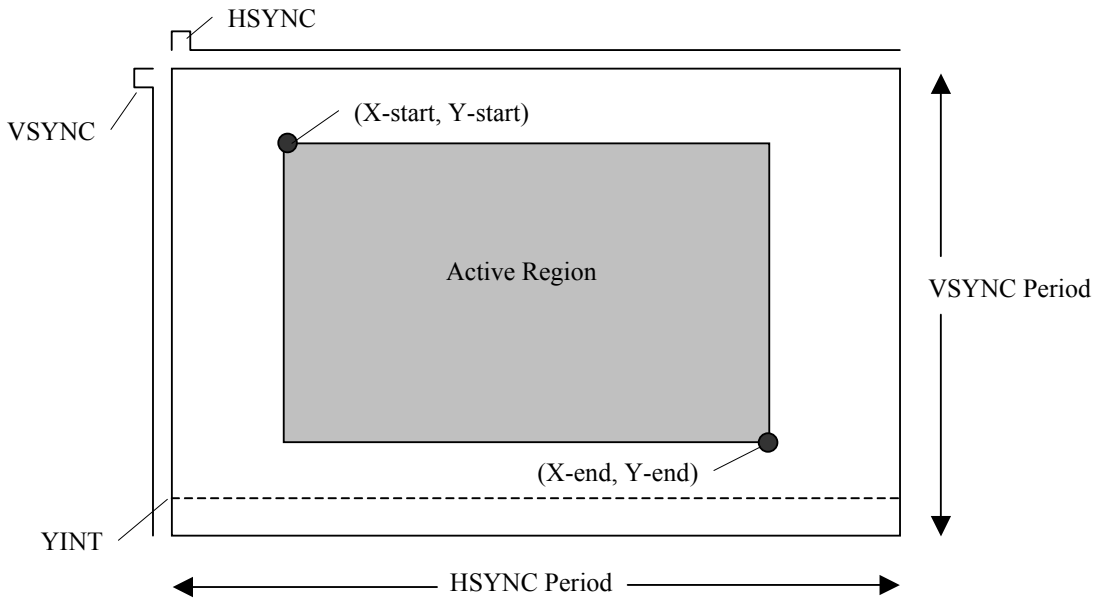


Figure 62. Active Region

NOTE: The maximum frequency of pixel clock we can support is the half of IOCLK.

8.5.3 Video Input Port Registers

Table 72. Video Input Port Register Mapping

RISC Address <11:0>	DSP Address <7:0>	Register	Description
0x000	0x00~0x01	-	Reserved
0x004	0x02~0x03	-	Reserved
0x008	0x04~0x05	CAM_START	Camera start x and start y register
0x00C	0x06~0x07	CAM_END	Camera end x and end y register
0x010	0x08~0x09	CAM_CTRL	Camera Control Register
0x014	0x0A~0x0B	CAM_PIXEL_SET	Pixel data bit select.
0x018	0x0C~0x0D	CAM_YUV_COEF1	Coefficient1 of YUV to RGB.
0x01C	0x0E~0x0F	CAM_YUV_COEF2	Coefficient2 of YUV to RGB.
0x020	0x10~0x11	CAM_YUV_COEF3	Coefficient3 of YUV to RGB.
0x024	0x12~0x13	CAM_YUV_OFFSET	Offset for each coefficient.
0x028	0x14~0x15	CAM_INT_EN	Camera interrupt enable register.
0x02C	0x16~0x17	CAM_INT_CTRL	Camera interrupt control register
0x030	0x18~0x19	CAM_VSYNC_CTRL	VSYNC control register.
0x034	0x1A~0x1B	CAM_HSYNC_CTRL	HSYNC control register.
0x038	0x1C~0x1D	CAM_PXCLK_CTRL	PXCLK control register.
0x03C	0x1E~0x1F	CAM_VSYNC_HSYNC	Vsync and hsync width

			register.
0x040	0x20~0x21	CAM_TIMING_CTRL	Timing control.
0x044	0x22~0x23	CAM_DMA_CTRL	DMA control register
0x048	0x24~0x25	CAM_DMA_LEN	DMA length register
0x04c	0x26~0x27	CAM_FIFO_CTRL	FIFO control register
0x050	0x28~0x29	CAM_FIFO_LEVEL_CHECK	FIFO level check register
0x054	0x2A~0x2B	CAM_FIFO_OP	FIFO operation register
0x058	0x2C~0x2D	CAM_FIFO_STATUS	FIFO status register
0x05C	0x2E~0x2F	CAM_RD_FIFO_DATA	FIFO data.

- **Camera START Register (CAM_START) – 0x008**

Bit	Name	Default	Description
15:0 (R/W)	XS<15:0>	16'h0000	The value of start column of the active region.
31:16 (R/W)	YS<15:0>	16'h0000	The value of start row of the active region.

- **Camera END Register (CAM_END) – 0x00C**

Bit	Name	Default	Description
15:0 (R/W)	XE<15:0>	16'h027F	The value of end column of the active region.
31:16 (R/W)	YE<15:0>	16'h01DF	The value of end row of the active region.

- **Camera Control Register (CAM_CTRL) – 0x010**

Bit	Name	Default	Description
0 (R/W)	PXCLK_CTRL	1'b0	0: PIXCLK generate from external sensor. 1: PIXCLK generate internally.
1 (R/W)	HSYNC_CTRL	1'b0	0: HSYNC generate from external sensor. 1: HSYNC generate internally.
2 (R/W)	VSYNC_CTRL	1'b0	0: VSYNC generate from external sensor. 1: VSYNC generate internally.
3 (R/W)	PIXCLK_INV	1'b0	Invert the polarity of input PIXCLK.
4 (R/W)	HSYNC_INV	1'b0	Invert the polarity of input HSYNC.
5 (R/W)	VSYNC_INV	1'b0	Invert the polarity of input VSYNC.
6 (R/W)	SINGLE	1'b1	1'b1: Capture one frame image. 1'b0: Continuous capture.
7 (R/W)	IO_TRIGGER	1'b0	0: PIXCLK directly pass through. 1: PIXCLK sampled by IOCLK.
8 (R/W)	YUVYCrCb	1'b0	1: Input pixel data is in YUV format 0: Input pixel data is in YCrCb format
11:9 (R/W)	YUV_FORMAT	3'b000	Raw pixel data input sequence: 000: YUYV/YCrYCb 001: UYYV/CrYYCb 010: YUVY/YCrCbY 011: UYVY/CrYCbY 100: YVYU/YCbYCr 101: VYU/CbYYCr 110: YVUY/YCbCrY 111: VYUY/CbYCrY

13:12 (R/W)	OUT_FORMAT	2'b00	RGB format send to the FIFO after YUV to RGB conversion. 00: 8:8:8 01: 6:5:5 10: 5:5:6 11: 5:6:5
14 (R/W)	YUVRGB	1'b0	0: Bypass the conversion of YUV to RGB. 1: Convert data format from YUV to RGB.
15	-		Reserved.
17:16(R/W)	X_SCA	2'b00	Contraction ratio in column direction. 2'b00: no contraction 2'b01: 1:2 contraction 2'b10: 1:4 2'b11: 1:8
19:18(R/W)	Y_SCA	2'b00	Contraction ratio in row direction 2'b00: no contraction 2'b01: 1:2 contraction 2'b10: 1:4 2'b11: 1:8
30:20	-		Reserved
31 (R/W)	INIT	1'b0	Just reset camera control module, configuration data still reserved. Logic remains in reset until a 0 is written to this bit.

- **Camera Pixel Bit Select Setting Register (CAM_PIXEL_SET) – 0x014**

Bit	Name	Default	Description
2:0 (R/W)	PIXEL_SEL	3'h0	Pixel bit select option. We always connect the input pixel data to the lowest data pin. 3'b000: select PXD<15:0> as valid data. 3'b001: select PXD<7:0> as valid data. 3'b010: select PXD<8:1> as valid data. 3'b011: store PXD<9:2> as valid data. 3'b100: select PXD<10:3> as valid data. 3'b101: select PXD<11:4> as valid data. 3'b110: select PXD<14:7> as valid data. 3'b111: store PXD<15:8> as valid data.
31:3	-	-	Reserved.

- **Red Coefficient for the YUV to RGB transform(CAM_YUV_COEF1) – 0x018**

Bit	Name	Default	Description
9:0 (R/W)	C1	10'h00	V coefficient for R.
19:10 (R/W)	C2	10'h00	U coefficient for R.
29:20 (R/W)	C3	10'h00	Y coefficient for R.
31:30	-	-	Reserved.

- **Green Coefficient for the YUV to RGB transform(CAM_YUV_COEF2) – 0x01C**

Bit	Name	Default	Description
9:0 (R/W)	C4	10'h00	V coefficient for G.
19:10 (R/W)	C5	10'h00	U coefficient for G.

29:20 (R/W)	C6	10'h00	Y coefficient for G.
31:30	-	-	Reserved.

- **Blue Coefficient for the YUV to RGB transform(CAM_YUV_COEF3) – 0x020**

Bit	Name	Default	Description
9:0 (R/W)	C7	10'h00	V coefficient for B.
19:10 (R/W)	C8	10'h00	U coefficient for B.
29:20 (R/W)	C9	10'h00	Y coefficient for B.
31:30	-	-	Reserved.

- **Offset for the YUV to RGB transform(CAM_YUV_OFFSET) – 0x024**

Bit	Name	Default	Description
9:0 (R/W)	OFFSET1	10'h00	Offset coefficient for R.
19:10 (R/W)	OFFSET2	10'h00	Offset coefficient for G.
29:20 (R/W)	OFFSET3	10'h00	Offset coefficient for B.
31:30	-	-	Reserved.

- **Camera Interrupt Control Enable Register (CAM_INT_EN) – 0x028**

Bit	Name	Default	Description
0 (R/W)	SENSOR_INT_EN	1'b0	Enable sensor interrupt.
1 (R/W)	FIFO_OFLOW_EN	1'b0	Enable FIFO overflow interrupt.
2 (R/W)	FIFO_UFLOW_EN	1'b0	Enable FIFO underflow interrupt.
31:3	-	-	Reserved.

- **Camera Interrupt Control Register (CAM_INT_CTRL) – 0x02C**

Bit	Name	Default	Description
0 (R/W)	SENSOR_INT	1'b0	When YCOUNT equals to YINT, this interrupt will be generated. Write a 1 to this bit will reset it.
1 (R/W)	FIFO_OFLOW	1'b0	FIFO is overflow. Write a 1 to this bit will reset it.
2 (R/W)	FIFO_UFLOW	1'b0	FIFO is underflow. Write a 1 to this bit will reset it.
31:3	-	-	Reserved.

- **Camera VSYNC Control Register (CAM_VSYNC_CTRL) – 0x030**

Bit	Name	Default	Description
15:0 (R/W)	VSYNC_ACT_NUM	16'h0000	VSYNC active width in the number of HSYNC.
31:16 (R/W)	VSYNC_BLANK_NUM	16'h0000	VSYNC blank width in the number of HSYNC.

- **Camera HSYNC Control Register (CAM_HSYNC_CTRL) – 0x034**

Bit	Name	Default	Description
-----	------	---------	-------------

15:0 (R/W)	HSYNC_ACT_NUM	16'h0000	HSYNC active width in the number of pixel.
31:16 (R/W)	HSYNC_BLANK_NUM	16'h0000	HSYNC blank width in the number of pixel.

- **Camera PIXCLK Control Register (CAM_PIXCLK_CTRL) – 0x038**

Bit	Name	Default	Description
15:0 (R/W)	PIXCLK_NUM	16'h0000	PIXCLK number in the number of IOCLK. Actual output PIXCLK period equals to 2*(PIXCLK_NUM+1)*IOCLK period.
31:16 (R/W)	-		Reserved.

- **Camera VSYNC width register (CAM_VSYNC_HSYNC) – 0x03C**

Bit	Name	Default	Description
15:0 (R/W)	VSYNC_HSYNC	16'h0000	Pixel number between the first HSYNC and VSYNC.
31:16 (R/W)	VSYNC_WIDTH	16'h0000	VSYNC width number in the number of PIXCLK period. Actual VSYNC valid period equals to VSYNC_ACT_NUM*HSYNC period + (VSYNC_HSYNC+VSYNC_WIDTH)*PIXCLK period.

- **Camera timing control register (CAM_TIMING_CTRL) – 0x040**

Bit	Name	Default	Description
0 (R/W)	PCLK_POLAR	1'b0	Invert pixel clock. 1 = Invert pixel clock (i.e. pixel clock ½ phase off) 0 = Do not invert pixel clock
1 (R/W)	HSYNC_POLAR	1'b0	Invert the horizontal sync signal. 1 = horizontal sync signal is active low. 0 = horizontal sync signal is active high.
2 (R/W)	VSYNC_POLAR	1'b0	Invert the vertical sync signal. 1 = vertical sync signal is active low. 0 = vertical sync signal is active high.
3 (R/W)	HSYNC_MASK	1'b0	Mask HSYNC control: 1: Disable the HSYNC during vertical blank time. 0: Enable the HSYNC during vertical blank time.
31:4 (R/W)	-	-	Reserved.

- **DMA Control Register (CAM_DMA_CTRL) – 0x044**

Bit	Name	Default	Description
0(R/W)	DMA_IO	1'b0	0: DMA operation. 1: IO operation.
1	-		Reserved.
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the data

			length set at the peripheral side doesn't match the DWORD size set in the DMA control.
3	-		Reserved.
5:4(R/W)	ENDIAN_MODE	2'h0	00: not changed. 01: byte exchange in dword. 10: word exchange in dword. 11: byte exchange in word.
31:6	-	-	Reserved.

- **DMA Operation Length Register (CAM_DMA_LEN) – 0x048**

Bit	Name	Default	Description
31:0 (R/W)	DATA_LEN	32'h0	The byte length of a DMA transfer. If set to zero, the DMA transfer works continuously until it is stopped.

- **FIFO Control Register (CAM_FIFO_CTRL) – 0x04C**

Bit	Name	Default	Description
1:0 (R/W)	FIFO_WIDTH	2'h0	00: byte-mode FIFO. 01: word-mode FIFO. 10: dword-mode FIFO. 11: dword-mode FIFO.
31:2	-	30'h0	Reserved.

- **FIFO Level Check Register (CAM_FIFO_LEVEL_CHK) – 0x050**

Bit	Name	Default	Description
5:0 (R/W)	FIFO_SC	6'h04	FIFO stop check in DWORD length.
9:6	-	4'h0	Reserved.
15:10 (R/W)	FIFO_LC	6'h08	FIFO low check in DWORD length.
19:16	-	4'h0	Reserved.
25:20 (R/W)	FIFO_HC	6'h10	FIFO high check in DWORD length.
31:26	-	6'h0	Reserved.

- **FIFO Operation Register (CAM_FIFO_OP) – 0x054**

Bit	Name	Default	Description
0 (R/W)	FIFO_START	1'b0	Start the FIFO transfer when this bit is declared.
1 (R/W)	FIFO_RESET	1'b0	Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
31:2	-	-	Reserved.

- **FIFO Status Register (CAM_FIFO_STATUS) – 0x058**

Bit	Name	Default	Description
7:0 (R)	FIFO_LEVEL	8'h00	The byte count of the valid data in the FIFO, varies from 0 to 255 bytes. In case FIFO is full, the value of this register is 0, thus user must concatenate FIFO_FULL bit with this value to determine the actual data count in the FIFO.

8 (R)	FIFO_FULL	1'b0	FIFO full status, the FIFO is full when read out as 1. This bit is concatenated with FIFO_LEVEL to be the actual FIFO data count.
9 (R)	FIFO_EMPTY	1'b0	FIFO empty status, equivalent to (FIFO_FULL, FIFO_LEVEL) = 0
31:10	-	-	Reserved.

- **Read FIFO Data Register (RD_FIFO_DATA) – 0x05C**

Bit	Name	Default	Description
31:0 (R)	FIFO_DATA	31'h00	FIFO data read by RISC or DSP.

8.6 Audio CODEC

8.6.4 Overview

The Atlas™-II Audio CODEC interface can be used to connect to either AC97 or I2S type CODEC. The CODEC interface share pins with USP1. For I2S CODEC like UDA1343, there is a separate control interface (L3BUS) besides of the digital audio interface. In this case, the control interface can be implemented with General-purpose IO.

The Audio CODEC interface is controlled by RISC through RBUS, and the audio data transfer between memory and Audio CODEC is through Atlas DMA channel. The Audio CODEC interface can support AC97 and UDA (I2S and LSB-justified) format, but these two formats can't work simultaneously, user need to configure the audio codec interface to select the audio format. For UDA I/F and AC97 PCM data stream, there are two DMA channels for PCM audio data in (channel 6) and audio data out (channel 7) respectively. For AC97 I/F, there is an extra pair of DMA channels for data in (channel 8) and data out (channel 9). The auxiliary DMA channels can be used for hardware audio mixer function and to transfer some non-audio data information, such as touch panel sample data.

The whole Audio CODEC interface can be divided into five parts:

- RISC/DSP I/O interface (register files)
- DMA Interface
- AC97 Controller

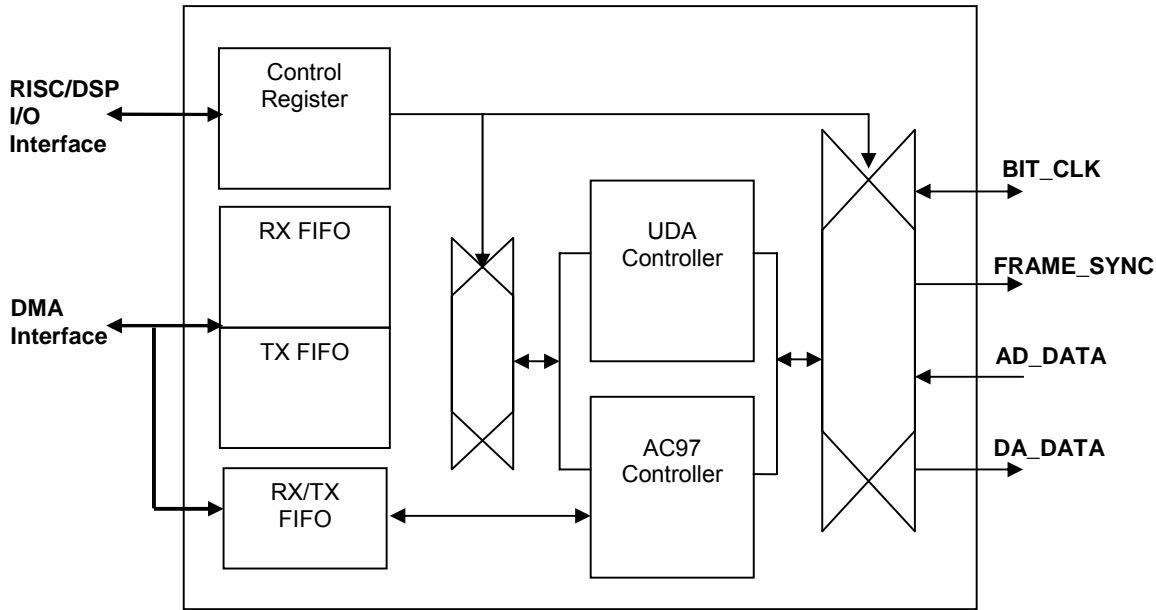


Figure 63. Block Diagram of Audio CODEC Interface

8.6.5 Signal Description

8.6.5.1 AC97 CODEC Interface

In this AC97 interface, Atlas operates as an AC97 codec controller. The AC97 interface is a 4-wire digital serial interface (with **BIT_CLK**, **FRAME_SYNC**, **AD_DATA** and **DA_DATA**). **BIT_CLK** and **AD_DATA** are inputs from the AC97 CODEC, and **FRAME_SYNC** and **DA_DATA** are generated by Atlas and output to the CODEC. This interface also allows the host (RISC/DSP) to read and write registers in the CODEC as part of the AC97 frame. It has the ability to turn on/off the left and right sound channels separately. And the AC97 interface can support both normal mode (48 KHz) and VRA mode (44.1 KHz, 22.05 KHz, 11.0125 KHz and 8 KHz).

In Atlas™-II, the AC97 interface is sharing the pin with one of the Universal Serial Ports (USP1).

Table 73. AC97 CODEC Interface Pin Description

Pin Name	Direction	Description
X_SCLK1 (BIT_CLK)	Input	12.288M Clock signal input from external AC97 CODEC
X_TXD1 (DA_DATA)	Output	Serial digital output data to AC97 CODEC
X_RXD1 (AD_DATA)	Input	Serial digital input data from AC97 CODEC
X_TFS1 (FRAME_SYNC)	Output	48 KHz Frame signal for synchronization

8.6.6 Functional Description

8.6.6.1 RISC I/O Interface

User can program the audio codec interface registers to configure the codec controller mode, or inquiry the controller status. All the RISC I/O access to audio codec interface registers is fixed latency. In Atlas, the registers of audio codec interface occupy 64KB space from *0x80060000* to *0x8006FFFF*. There are three groups of registers which are located at the register space with the different offset address:

- Audio CODEC controller registers (address offset 0x0)
- TX FIFO control registers (address offset 0xf80)
- RX FIFO control registers (address offset 0xfc0)
- AUX TX FIFO control registers (address offset 0xB80)
- AUX RX FIFO control registers (address offset 0xB80)

8.6.6.2 DMA Interface

- **PCM DMA Channel**

The CODEC interface will have FIFOs for PCM audio data transfer for AC97 interface. The FIFOs are organized as two separate physical SRAM blocks (each one is sixteen 32-bit entries): one for audio in (channel 6) and another for audio out (channel 7).

AC97 interface expect to receive a 32-bit value from the FIFO that contains two samples. In stereo normal mode the data will be interlaced as LR-LR-LR... samples. In stereo interlace mode, the data will be in RL-RL-RL... format. In mono mode the 16-bit input data is saved in low WORD and then high WORD.

The audio CODEC interface shares the DMA controller with other peripherals. There are programmable FIFO level check registers to decide the CODEC FIFO request priority. And also the DMA block will update the FIFO status during the data transaction. When FIFO is full or empty, the FIFO full or empty interrupt will be generated, more read or write operation to the empty or full FIFO will trigger the FIFO underflow or overflow interrupt if the interrupt is enabled.

Please refer to the DMA Controller and peripheral FIFO for more information about the DMA interface.

- **Auxiliary DMA Channel**

For AC97 interface, there are 12 slots for data transfer. The PCM data for left channel and right channel is transferred through AC-link slot3 and slot4, which takes Atlas PCM DMA channel (DMA channel 6 and 7). Atlas provides a pair of auxiliary DMA channel (channel 8 and channel 9) for other slots data transmission through AC-link. DMA channel 8 is for data in and DMA channel 9 is for data out. User can configure AC97 interface to specify which slots is used for auxiliary data in and which for auxiliary data out.

8.6.6.3 AC97 Controller

The Atlas™-II, AC97 Controller communicates with its companion AC97 audio codec via AC-link digital serial interface, and it supports the most common point-to-point AC-link connection between controller and CODEC.

The AC97 CODEC derives its clock internally from an external attached 24.576MHz crystal, and drives a buffered and divided down (1/2) clock to its digital controller over AC-link under the signal name "BIT_CLK". The beginning of all audio sample frames, transferred over the AC-link is synchronized to the rising edge of the SYNC signal. SYNC is driven by Atlas™-II AC97 Controller, it takes BIT_CLK as an input and generates the SYNC by dividing BIT_CLK by 256 and applying some conditioning to tailor its duty cycle. This yields a 48 KHz SYNC signal whose period defines an audio frame. Data is transferred over AC-link on every rising edge of BIT_CLK, and subsequently sampled on the falling edge of BIT_CLK on the receiving side of AC-link.

The AC-link is a bi-directional, fixed clock rate, serial data stream. AC-link employs a TDM (Time Division Multiplex) scheme that divides each audio frame into 12 outgoing and 12 ingoing data streams, each with 20-bit sample resolution. The Atlas™-II AC97 Controller handles 16-bit PCM audio streams input and output, as well as accessing control register.

Table 74. AC-link Output Slots

Slot	Name	Description
0	Data out tag	Indicates which slots contain valid data;
1	Control CMD ADDR write port	Read/write command bit plus 7-bit CODEC register address
2	Control Data write port	16-bit command register write data
3	PCM left DAC play back	16-bit PCM data for left channel
4	PCM right DAC play back	16-bit PCM data for right channel
5-12	Aux data out	16-bit PCM data for auxiliary channel. The slot is programmable and only one slot can be valid.

Table 75. AC-link Input Slots

Slot	Name	Description
0	Data In tag	Indicates which slots contain valid data;
1	Status ADDR read port	MSB echo register address; LSB indicates which slots request data
2	Status Data read port	16-bit command register read data
3	PCM left ADC record	16-bit PCM data from left channel
4	PCM right ADC record	16-bit PCM data from right channel
5-11	Aux data in	16-bit PCM data in or control data input. The slot is programmable, but only one slot is valid.
12	I/O status	GPIO read port

8.6.6.3.1 Register Operations

AC-link slot 1 and slot 2 are used as AC97 command port to control AC97 CODEC features and monitor status. The command port supports up to 64 16-bit read/write registers, addressable on even byte boundaries.

AC-link output frame (from Atlas™-II AC97 Controller interface to AC97 CODEC) Slot 1 indicates the whether the current control transaction is a read or write operation, and also specifies the target register address. If it's a write operation, the output frame Slot 2 contains the valid data, and the tag bit of Slot 1 is asserted; if read operation, the tag bit of Slot 2 is de-asserted, the data stream in Slot 2 is abandoned. AC-link input frame (from AC97 CODEC to Atlas™-II AC97 Controller interface) Slot 1 echoes the control register read address (bit 18 to bit 12), and for the variable sample rate, it delivers the request flags for all output slots (bit 11 to bit 2). The echoed address is valid only when Slot 1 is tagged valid, but the request flags are independent of the tag bit. When Slot 2 is tagged valid, the input frame Slot 2 contains the 16-bit control register read data.

When Atlas™-II AC97 Controller issues the control register read/write command, the AC-link output frame is illustrated in the following diagrams:

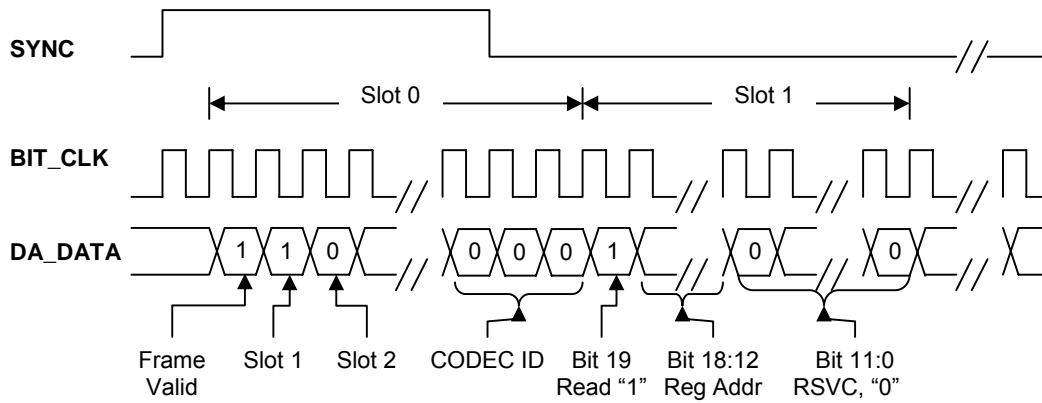


Figure 64. AC-link Output Frame Read Command Diagram

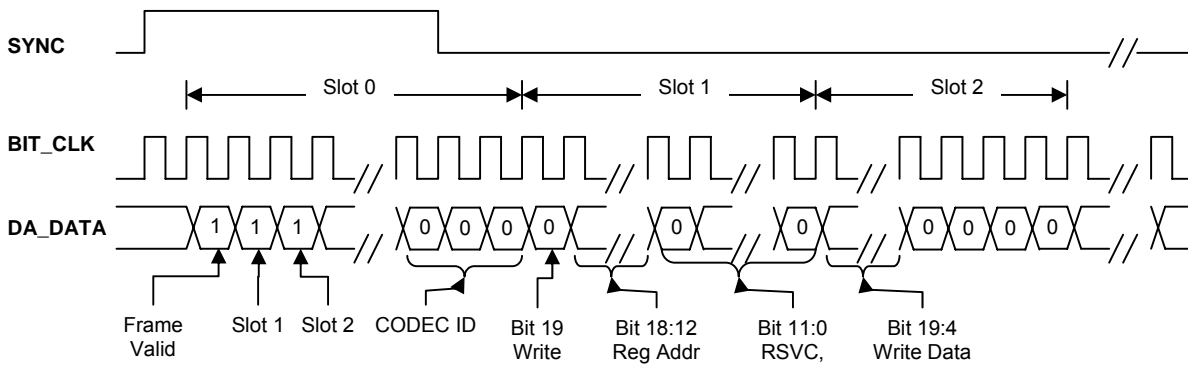


Figure 65. AC-link Output Frame Write Command Diagram

The following figure is the diagram for AC97 CODEC response with the control register read data.

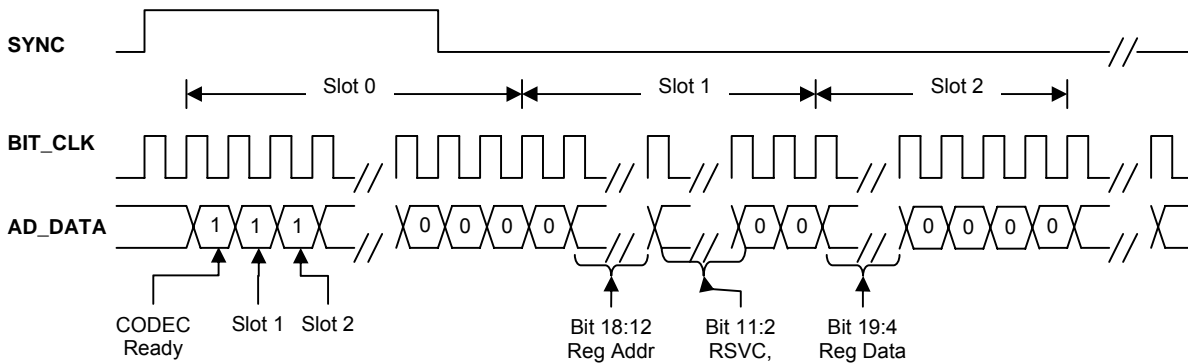


Figure 66. AC-link Input Frame Command Diagram

8.6.6.3.2 PCM Record

The following figure shows the AC97 PCM record function block. There are two records mode: stereo mode and mono mode. It depends on the Atlas™-II AC97 Controller configuration.

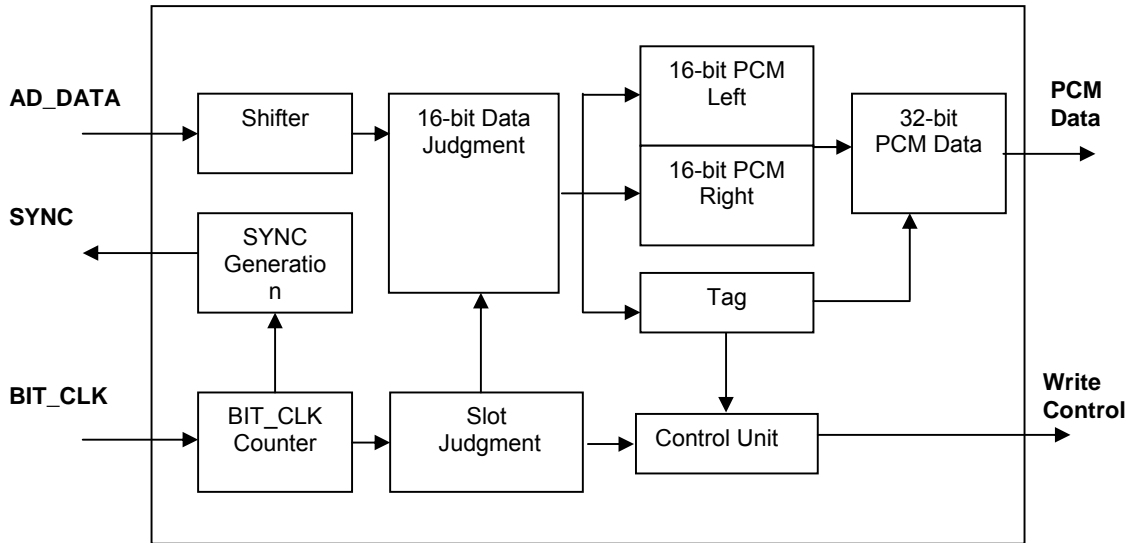


Figure 67. AC97 PCM Record Functional Block Diagram

The data transfer over AC-link data is in serial mode. First the Atlas AC97 controller converts the serial input data into 16-bit data. The 16-bit data maybe AC-link tag, status address, status data, PCM left channel data or PCM right channel data. The BIT_CLK counter will make the slot judgment. AC-link input slot 0 is the AC-link tag, the bits in the tag indicate whether the slots contain valid data. AC-link input slot 3 and 4 are PCM record left and right channel output of AC97's input MUX, post-ADC; when bit 11 or bit 10 of AC-link tag are asserted, it indicates that slot 3 or slot 4 contain valid PCM data. In 48 KHz sample rate, the PCM left channel and right channel are always valid in each AC97 audio frame; but in VRA (Variable Rate Audio) mode, the PCM left channel and right channel may not be valid at the same frame.

In stereo mode, the 32-bit PCM data is made up of a pair of PCM left channel and right channel. Normally the MSB 16-bit is left channel, the LSB 16-bit right channel; in interlace mode, left channel is saved in LSB 16-bit and right channel MSB 16-bit. In mono mode, the 16-bit left channel or right channel data is saved is LSB 16-bit first, then the MSB 16-bit.

The PCM record process is shown in the following flowcharts.

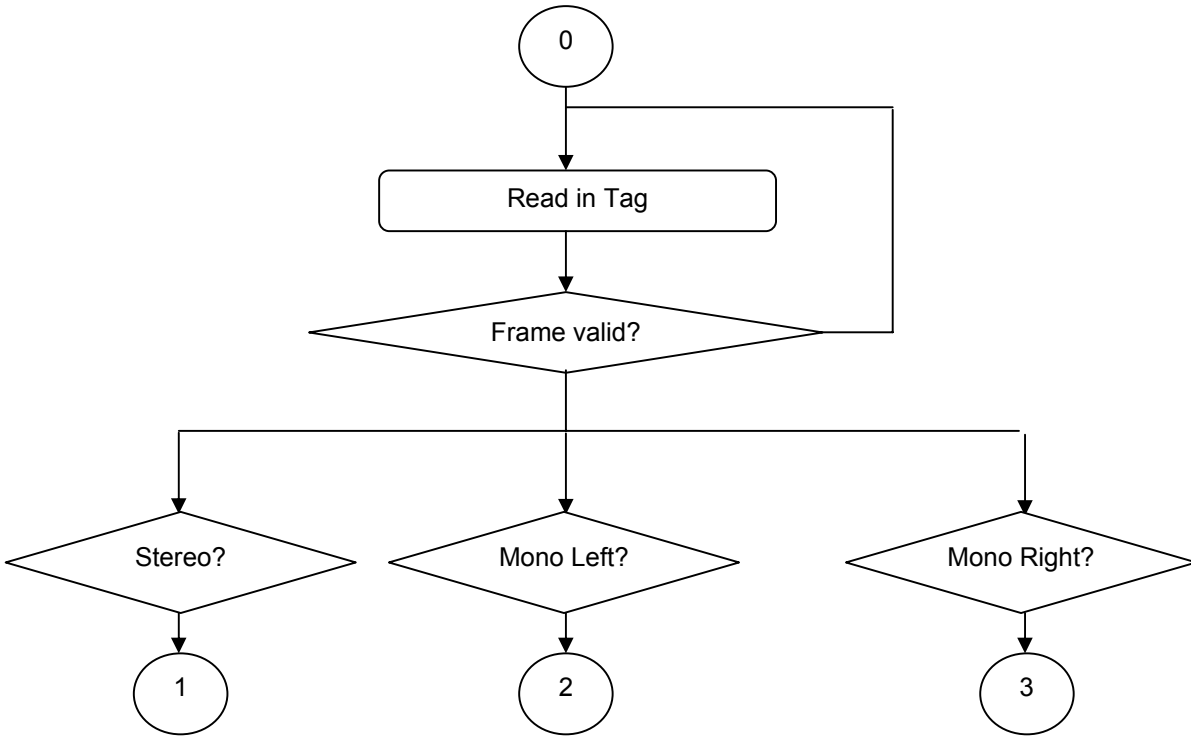


Figure 68. AC97 PCM Record Flowchart

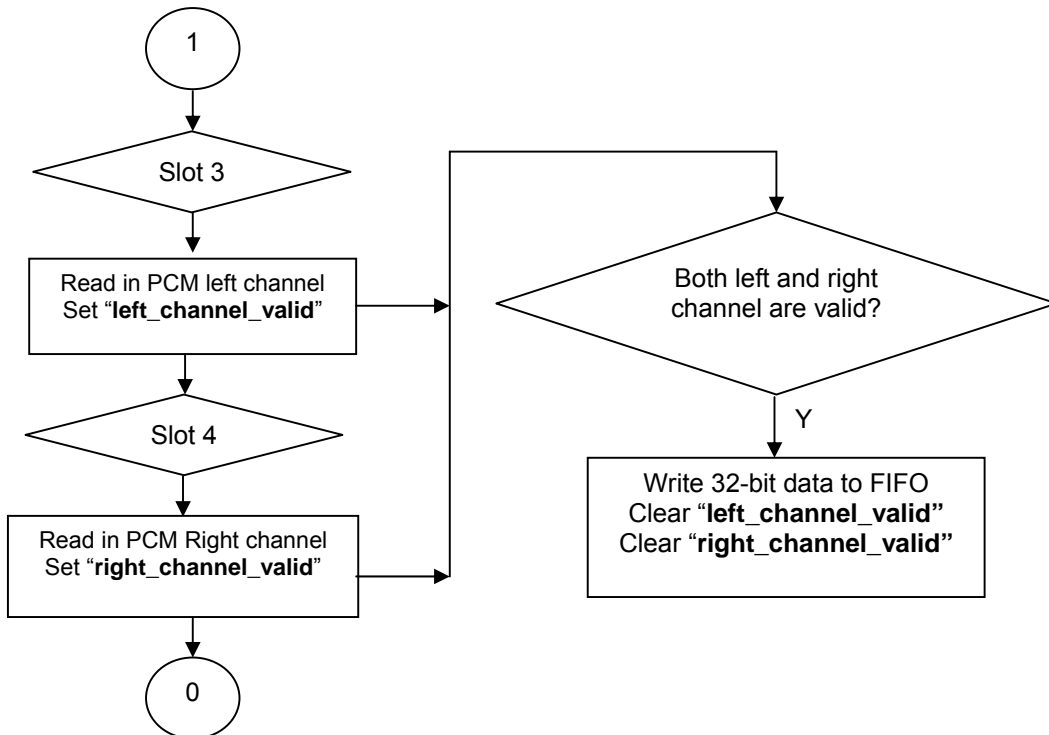


Figure 69. Stereo Mode Flowchart

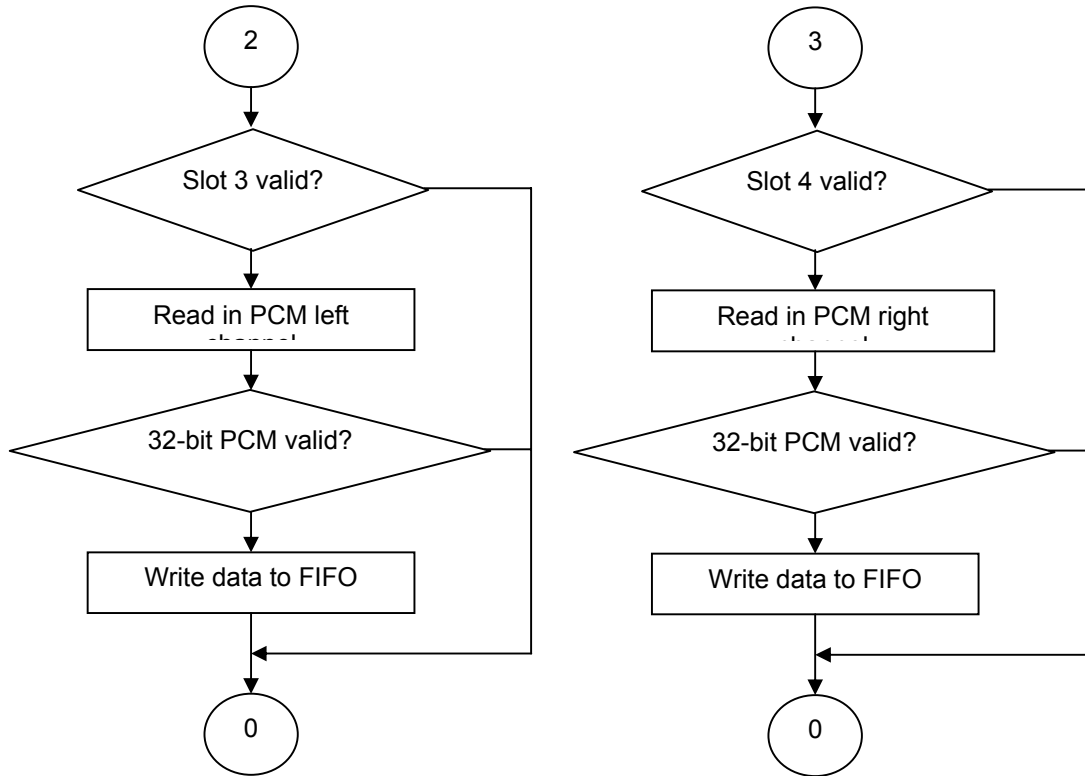


Figure 70. Mono Mode (Left/Right) Flowcharts

8.6.6.3.3 PCM Playback

The Atlas™-II AC97 Controller supports both 48KHz sample rate and VRA mode. When the AC97 CODEC works at 48KHz stereo mode, each audio output frame contain both the left channel and right channel valid data. And for the AC97 Controller, it fetches 32-bit PCM data from FIFO at the speed of 48KHz for each frame. But for the lower sample frequency such as 44.1KHz, 22.05KHz, 11.025KHz and 8KHz, the AC97 Controller can't send valid left and right channel data to AC97 CODEC in each output frame, otherwise the FIFO in the AC97 CODEC will overflow.

When the AC97 CODEC is configured to be in VRA mode, the AC97 CODEC will be responsible for the flow control of output data stream. In AC-link input frame data, AC97 CODEC indicates its request of data for next output frame in slot 1. The SLOTREQ bits occupy Slot 1 bit11 to bit 2, which indicate the request from Slot 3 to Slot 12. The SLOTREQ bits are always valid and independent of tag bit.

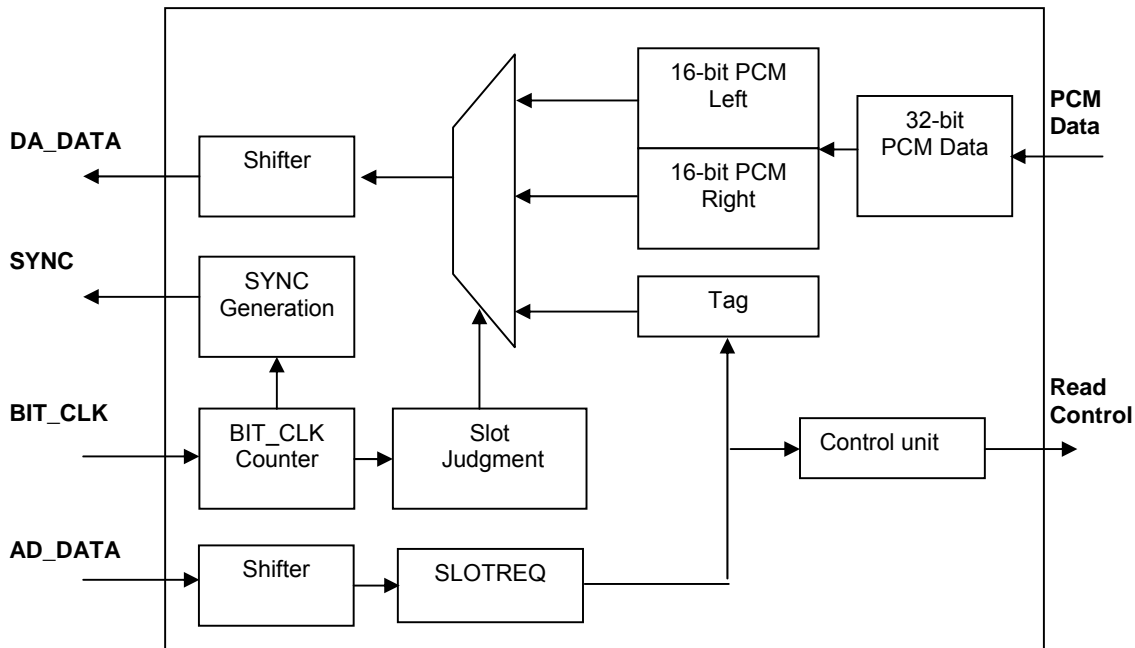


Figure 71. AC97 PCM Playback Functional Diagram

8.6.6.3.4 Auxiliary Data Input

User can program AC97 Controller to select the right slot which is used for auxiliary data input. No matter which slot is used to input the data, the input data is always transferred to memory through DMA channel 8.

The data transfer over AC-link data is in serial mode. First the Atlas™-II AC97 controller converts the serial input data into 16-bit data. The 16-bit data may be PCM data or other ADC sample data. The BIT_CLK counter will make the slot judgment. AC-link input slot 0 is the AC-link tag; the bits in the tag indicate whether the selected slot contains valid data.

8.6.6.3.5 Auxiliary Data Output

User can program AC97 Controller to select the right slot which is used for auxiliary data output. No matter which slot is used to input the data, the input data is always transferred to memory through DMA channel 9.

And for the AC97 controller, it fetches 32-bit PCM data from FIFO. The AC97 CODEC will be responsible for the flow control of output data stream. In AC-link input frame data, AC97 CODEC indicates its data request for next output frame in slot 1. The SLOTREQ bits occupy Slot 1 bit11 to bit 2, which indicate the request from Slot 3 to Slot 12. The SLOTREQ bits are always valid and independent of tag bit.

8.6.6.4 Interrupt Generation

There are four DMA channels in the CODEC interface, all the CODEC interrupt generation comes from the DMA interface. There are eight FIFO statuses which can trigger the CODEC interrupt if it is not masked.

- TX FIFO Full
- TX FIFO empty

- TX FIFO Overflow

When the TX FIFO Full status is indicated, if DMA interface continues to write audio data into TX FIFO, the TX FIFO overflow will be triggered. Generally the interrupt will not arise, because when the TX FIFO is full, the DMA request will not be generated, and it can guarantee that no data will be written into TX FIFO.

- TX FIFO Underflow

When the TX FIFO is in empty status, while DMA controller can't serve CODEC transmission channel, the further data request from audio CODEC will trigger the TX FIFO underflow interrupt.

- RX FIFO Full
- RX FIFO empty
- RX FIFO Underflow
- RX FIFO Overflow

When RX FIFO is in full state while DMA controller can't transfer the data into SDRAM, more audio input data stream will trigger the RX FIFO overflow interrupt.

8.6.7 Audio CODEC Registers

Table 76. Audio CODEC Register Mapping

RISC Address <11:0>	Register	Description
0x0000	CODEC_SHARE	CODEC interface control register
0x0004~0x0018	-	Reserved
0x001C	CODEC_AC97_CTRL	AC97 interface control register
0x0020	CODEC_AC97_CMD	AC97 command register
0x0024	CODEC_AC97_RTAGH	AC97 input frame tag
0x0028	CODEC_AC97_STATUS	AC97 CODEC status data
0x002C	CODEC_AC97REG_OK	AC97 register read back status
0x0030	CODEC_SLOT12_DATA	AC97 slot12 read data
0x0034	CODEC_TXSLOT_EN	AC97 auxiliary TX slot selection
0x0038	CODEC_RXSLOT_EN	AC97 auxiliary RX slot selection
0x003C~0x007C	-	Reserved
0x0B80	CODEC_TX_AUX_DMA_IO_CTRL	CODEC auxiliary TX DMA transmit mode selection
0x0B84	-	Reserved
0x0B88	CODEC_TX_AUX_FIFO_CTRL	CODEC auxiliary TX FIFO transfer threshold setting
0x0B8C	CODEC_TX_AUX_FIFO_LEVEL_CHK	CODEC auxiliary TXFIFO level check
0x0B90	CODEC_TX_AUX_FIFO_OP	CODEC auxiliary TXFIFO operation
0x0B94	CODEC_TX_AUX_FIFO_STS	CODEC auxiliary TXFIFO status register
0x0B98	CODEC_TX_AUX_FIFO_INT_EN	CODEC auxiliary TXFIFO interrupt enable
0xB9C – 0xBBC	-	Reserved
0x0BC0	CODEC_RX_AUX_DMA_IO_CTRL	CODEC auxiliary RX FIFO mode selection
0x0BC4	-	Reserved

0x0BC8	CODEC_RX_AUX_FIFO_CTRL	CODEC auxiliary RXFIFO transfer threshold
0x0BCC	CODEC_RX_AUX_FIFO_LEVEL_CHK	CODEC auxiliary RXFIFO level check
0x0BD0	CODEC_RX_AUX_FIFO_OP	CODEC auxiliary RXFIFO operation
0x0BD4	CODEC_RX_AUX_FIFO_STS	CODEC auxiliary RXFIFO status register
0x0BD8	CODEC_RX_AUX_FIFO_INT_EN	CODEC auxiliary RXFIFO interrupt enable
0xBE0 – 0xF7C	-	Reserved
0x0F80	CODEC_TX_DMA_IO_CTRL	CODEC transmit DMA I/O mode
0x0F84	-	Reserved
0x0F88	CODEC_TX_FIFO_CTRL	CODEC TXFIFO transfer threshold
0x0F8C	CODEC_TX_FIFO_LEVEL_CHK	CODEC TXFIFO level check
0x0F90	CODEC_TX_FIFO_OP	CODEC TXFIFO operation
0x0F94	CODEC_TX_FIFO_STS	CODEC TXFIFO status register
0x0F98	CODEC_TX_FIFO_INT_EN	CODEC TXFIFO interrupt enable
0x0F9C~FB C	-	Reserved
0x0FC0	CODEC_RX_DMA_IO_CTRL	CODEC receive DMA I/O mode
0x0FC4	-	Reserved
0x0FC8	CODEC_RX_FIFO_CTRL	CODEC RXFIFO transfer threshold
0x0FCC	CODEC_RX_FIFO_LEVEL_CHK	CODEC RXFIFO level check
0x0FD0	CODEC_RX_FIFO_OP	CODEC RXFIFO operation
0x0FD4	CODEC_RX_FIFO_STS	CODEC RXFIFO status register
0x0FD8	CODEC_RX_FIFO_INT_EN	CODEC RXFIFO interrupt enable
Others	-	Reserved

- **CODEC share register (CODEC_SHARE) – 0x0**

CODEC interface supports AC97 audio formats. CODEC_SHARE is used to control whether record channel or playback channel in the Audio CODEC interface is enabled.

Bit	Name	Default	Description
0	-	1'b0	Reserved
1 (R/W)	INTERLACE_MODE	1'b0	CODEC interlace mode selection 0 = The 32-bit audio data is in L-R format, that's the MSB 16-bit is from left channel, and the LSB 16-bit is from right channel; 1 = The 32-bit audio data is in R-L format, the MSB 16-bit is right channel, and the LSB 16-bit is left channel.
4:2	-	3'b1	Reserved
5 (R/W)	AC97_AD_IDLE	1'b1	AC97 record channel status setting: 1 = AC97 record channel is shut down in CODEC interface; no audio data stream can be recorded. 0 = AC97 record channel is enabled in AC97 interface, input audio data stream is allowed to be recorded. By default, the AC97 record channel is shut down, before recording audio data stream from AC97 CODEC, user needs to clear this register bit.

			The AC97 record channel works only when AC97 mode is selected and AC97_AD_IDLE is de-asserted.
6 (R/W)	AC97_DA_IDLE	1'b1	AC97 playback channel status setting: 1 = AC97 playback channel is shut down in AC97 interface; the external AC97 CODEC receive NO valid audio data stream; 0 = AC97 playback channel is enabled in AC97 interface, audio data stream is allowed to be feed to external AC97 CODEC. By default, the AC97 playback channel is shutdown, before playing audio signal through AC97 CODEC, user needs to clear this register bit. The AC97 playback channel works only when AC97 mode is selected and AC97_DA_IDLE is de-asserted.
7	-	1'b0	Reserved
8 (R/W)	DSP_CTRL	1'b0	CODEC interface register control mode 0 = RISC control CODEC registers. DSP access is disabled 1 = DSP control CODEC registers. RISC access is disabled
9 (R/W)	ALAW_EN	1'b0	For PCM DMA playback channel, 0 = ALAW expansion is enabled 1 = ALAW expansion is disabled
10 (R/W)	ULAW_EN	1'b0	For PCM DMA playback channel, 0 = ULAW expansion is enabled 1 = ULAW expansion is disabled
31:11	-	21'h0	Reserved

- **CODEC AC97 Control Register (CODEC_AC97_CTRL) – 0x1C**

This register is used to configure AC97 controller, including:

- Left and right channel control;
- AC97 CODEC warm wake control;
- AC-link output frame control

Bit	Name	Default	Description
1:0 (R/W)	DA_CHANNEL	2'h0	AC97 playback channel enable: Bit0 decides right channel Bit1 decides left channel 1 = channel is enabled 0 = channel is disabled The playback channel enable register bits decide how the AC97 interface deals with 32-bit PCM data.
3:2 (R/W)	AD_CHANNEL	2'h0	AC97 record channel enable Bit2 decides right channel Bit3 decides left channel 1 = channel is enabled 0 = channel is disabled The record channel enable register bits decide the source of record audio data.
4 (R/W)	WARM_WAKE	1'b0	AC97 warm wake control 1 = warm wake AC97 CODEC

			0 = exit warm wake-up mode. WARM_WAKE should be set when BIT_CLK is absent. The asserted WARM_WAKE signal set the SYNC to be HIGH to bring AC97 CODEC out of halted or low-power mode.
5 (R/W)	AC97_START	1'b0	AC97 interface start control 1 = AC97 interface is started to work; 0 = AC97 interface is stopped. Before AC97 interface is started, user needs to enables the channel, once AC97_START is asserted, AC97 interface begins to work.
7:6	-	2'h0	Reserved
15:8 (R/W)	AC97_WRITE_TAGH	8'h0	AC-link output frame tag setting
31:16	-	16'h0	Reserved

AC97_WRITE_TAGH is used to control the AC-link output frame. The tag of output frame is decided by **AC97_WRITE_TAGH**, user needs to program the register before play audio data stream or issue command to AC97 CODEC.

Table 77. AC97_WRITE_TAGH Description

Bit	Name	Default	Description
15 (R/W)	Frame Valid	1'b0	1 = Indicates that the output frame contains the valid data. 0 = Indicates that the output frame is invalid. After AC97 playback channel initialization, the Frame_valid bit should be set. Otherwise, AC97 CODEC will abandon all output frames from AC97 interface.
14 (R/W)	Tag bit of Slot 1	1'b0	1 = Inform the AC97 interface that output frame Slot 1 is enabled in next output frame. 0 = Indicates that output frame Slot 1 is disabled Before issuing read/write command to AC97 CODEC, user needs to program the command type and CODEC register index into the <i>CODEC_AC97_COMMAND</i> register, and then set the tag bit of slot 1 to inform the AC97 interface to issue the command to AC97 CODEC in next output frame. After the command is sent to AC97 CODEC, the bit will be cleared by hardware.
13 (R/W)	Tag bit of Slot 2	1'b0	1 = Inform the AC97 interface that output frame Slot 2 is enabled in the next output frame. 0 = Indicates that output frame Slot 2 is not enabled. When issuing the write command to AC97 CODEC, the tag bit of Slot 2 should be set, the AC97 interface will send the AC97_CMD_DATA to AC97 CODEC through Slot 2 in next output frame. After the command data is written to AC97 CODEC, the tag bit will be cleared by hardware.
12 (R/W)	Tag bit of Slot 3	1'b0	1 = Indicates that output frame Slot 3 is enabled 0 = Indicates that output frame Slot 3 is disabled. For stereo and mono left mode, the output frame Slot 3 and AC97 playback left channel must be enabled; for mono right mode, if output frame slot 3 is enabled, the

			audio data of right channel will be duplicated and played through left channel. The tag bit of Slot 3 is controlled by software; it decides whether output frame Slot 3 is enabled to transfer valid audio data to AC97 CODEC.
11 (R/W)	Tag bit of Slot 4	1'b0	1 = Indicates that output frame Slot 4 is enabled 0= Indicates that output frame Slot 4 is disabled. For stereo and mono right mode, the output frame Slot 4 and AC97 playback right channel must be enabled; for mono left mode, if output frame slot 4 is enabled, the audio data of left channel will be duplicated and played through left channel. The tag bit of Slot 4 is controlled by software; it decides whether the output frame Slot 4 is enabled to transfer audio data to AC97 CODEC.

- **CODEC AC97 Command Register (CODEC_AC97_CMD) – 0x20**

Bit	Name	Default	Description
7:0	-	8'b0	Reserved
14:8 (R/W)	AC97_CMD_ADDR	7'b0	AC97 CODEC control register address
15 (R/W)	CMD_TYP	1'b0	Command type: 1 = Read AC97 CODEC status command 0 = Write AC97 CODEC control register command.
31:16 (R/W)	AC97_CMD_DATA	16'h0	AC97 CODEC control register write data

- **CODEC AC97 Input Frame Tag Register (CODEC_AC97_RTAGH) – 0x24**

It's a read only register. The tag of AC-link input frame is saved in this register. It can be used to judge whether AC97 CODEC has been ready.

Bit	Name	Default	Description
7:0 (R)	AC97_SLOT_RQ	8'h0	AC97 input frame slot request for next output frame
15:8 (R)	AC97_READ_TAGH	8'h0	AC97 input frame tag: Bit [7]: 1 = CODEC is ready 0 = CODEC is not ready. Bit [6]: 1 = Input frame Slot 1 contains valid echoed address; 0 = Input frame Slot 1 contains no valid data. Bit [5]: 1 = Input frame Slot 2 contains valid AC97 CODEC status data; 0 = Input frame Slot 2 contains no valid data. Bit [4]: 1 = Input frame Slot 3 contain valid PCM left channel data; 0 = Input frame Slot 3 contains NO valid data. Bit [3]: 1 = Input frame Slot 4 contains valid right channel data; 0 = Input frame Slot 4 contains no valid date. Bit [2:0]: meaningless for AC97 interface.
31:9	-	24'h0	Reserved

- **CODEC AC97 Status register (CODEC_AC97_STATUS) – 0x28**

When reading control register from AC97 CODEC, the result is saved in this register. It contains the 16-bit control register read data from AC97 CODEC and its corresponding register index.

Bit	Name	Default	Description
15:0 (R)	AC97_STATUS_ADR	16'h0	AC97 CODEC register address
31:16 (R)	AC97_STATUS_DATA	16'h0	AC97 CODEC register read data

- **CODEC AC97 Status Read Control Register (CODEC_AC97REG_OK) – 0x2C**

This register is used to monitor the AC97 CODEC command ports (Slot 1 and Slot 2) and I/O status port (Slot 12).

Bit	Name	Default	Description
0 (R/W)	AC97_REG_RD	1'b0	AC97 register read status 1 = AC97 CODEC has delivered the register read data, the data and echoed address is saved in CODEC_AC97_STATUS. 0 = AC97 CODEC delivers no status data. Before issuing read command to AC97 CODEC, User needs to write 1'b1 to clear the status.
1 (R/W)	AC97_SLOT12_VLD	1'b0	Slot 12 contains valid data: 1 = AC-link slot12 contains valid data in current input frame, the data will be saved in register CODEC_SLOT12_DAT. 0 = Slot 12 contains no valid data in current input frame. User needs to write 1'b1 to this register bit to clear this status.
31:2		30'b0	Reserved

- **CODEC AC97 Slot12 Data Register (CODEC_SLOT12_DAT) – 0x30**

Bit	Name	Default	Description
15:0 (R)	CODEC_SLOT12_DAT	16'h0	AC97 slot12 read back data
31:16		16'h0	Reserved

- **CODEC AC97 Auxiliary Output Slot Selection Register (CODEC_TXSLOT_EN) – 0x34**

Bit	Name	Default	Description
6:0 (R/W)	CODEC_TXSLOT_EN	7'h0	AC97 auxiliary output slot selection (only one slot allowed to be valid) 0 = Slot output is disabled 1 = Slot output is enabled Bit 0: output frame Slot5 selection Bit 1: output frame Slot6 selection Bit 2: output frame Slot7 selection Bit 3: output frame Slot8 selection Bit 4: output frame Slot9 selection Bit 5: output frame Slot10 selection Bit 6: output frame Slot11 selection
31:7		25'h0	Reserved

- **CODEC AC97 Auxiliary Input Slot Selection Register (CODEC_RXSLOT_EN) – 0x38**

Bit	Name	Default	Description
6:0 (R/W)	CODEC_RXSLOT_EN	7'h0	AC97 auxiliary input slot selection (only one slot allowed to be valid) 0 = Slot input is disabled 1 = Slot input is enabled Bit 0: input frame Slot5 selection Bit 1: input frame Slot6 selection Bit 2: input frame Slot7 selection Bit 3: input frame Slot8 selection Bit 4: input frame Slot9 selection Bit 5: input frame Slot10 selection Bit 6: input frame Slot11 selection
31:7		25'h0	Reserved

- **CODEC TX_FIFO DMA I/O Control Register (CODEC_TX_AUX_DMA_IO_CTRL) – 0xB80**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b1	1 for I/O mode, 0 for DMA mode.
1 (R)	IO_DMA_RW	1'b1	1: read from CODEC 0: write to CODEC
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the DATA_LEN set at the peripheral side doesn't match the DWORD size set in the DMA control.
3 (R/W)	RW_ENDIAN	1'b0	Reserved
31:4	-	28'h0	Reserved

- **CODEC TX_FIFO Control Register (CODEC_TX_AUX_FIFO_CTRL) – 0xB88**

Bit	Name	Default	Description
1:0 (R/W)	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for DWORD.
7:2 (R/W)	FIFO_THD<5:0>	6'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
31:8	-	24'h0	Reserved

- **CODEC TX_FIFO Level Check Register (CODEC_TX_AUX_FIFO_LEVEL_CHK) – 0XB8C**

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC	4'h0	Stop check in DWORD.
9:4 (R/W)	-	6'h0	Reserved
13:10 (R/W)	FIFO_LC	4'h0	Low check in DWROD.
19:14 (R/W)	-	6'h0	Reserved
23:20 (R/W)	FIFO_HC	4'h0	High check in DWORD.
31:24	-	8'h0	Reserved

- **CODEC TX_FIFO Operation Register (CODEC_TX_AUX_FIFO_OP) – 0XB90**

Bit	Name	Default	Description
0 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
1 (R/W)	FIFO_RESET	1'b0	Internally link to FIFO_START_INI . Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
31:2	-	30'h0	Reserved

- **Auxiliary CODEC TXFIFO Status Register (CODEC_TX_AUX_FIFO_STS) – 0xB94**

This register indicates the auxiliary TX FIFO status.

Bit	Name	Default	Description
0 (R)	TX_AUX_FIFO_FULL	1'b0	Auxiliary TX FIFO full status: 1 = TX FIFO is in full state; 0 = TX FIFO is not in full state. It indicates the current TX FIFO full status. Once the FIFO status changes, the status bit is cleared automatically.
1 (R)	TX_AUX_FIFO_EMPTY	1'b0	Auxiliary TX FIFO empty status: 1 = TX FIFO is in empty states; 0 = TX FIFO is not in empty states. It indicates the current TX FIFO empty status. Once the FIFO status changes, the status bit is cleared automatically.
2 (R/W)	TX_AUX_FIFO_OFLOW	1'b0	TX FIFO overflow status 1 = TX FIFO overflow takes place; 0 = TX FIFO is not overflow. User can write 1'b1 to clear the register bit after the TX FIFO overflow takes place.
3 (R/W)	TX_AUX_FIFO_UFLOW	1'b0	TX FIFO underflow status 1 = TX FIFO underflow takes place; 0 = TX FIFO is not underflow. User needs to write 1'b1 to clear the register bit after the TX FIFO underflow takes place.
31:4	-	28'h0	Reserved

- **AUX CODEC TXFIFO Interrupt Enable Register (CODEC_TX_AUX_FIFO_INT_EN) – 0xB98**

Bit	Name	Default	Description
0 (R/W)	TX_AUX_FIFO_FULL_EN	1'b0	1 = Aux TX FIFO full interrupt is enabled 0 = Aux TX FIFO full interrupt is disabled
1 (R/W)	TX_AUX_FIFO_EMPTY_EN	1'b0	1 = Aux TX FIFO empty interrupt is enabled 0 = Aux TX FIFO empty interrupt is disabled
2 (R/W)	TX_AUX_FIFO_OFLOW_EN	1'b0	1 = Aux TX FIFO overflow interrupt is enabled 0 = Aux TX FIFO overflow interrupt is disabled
3 (R/W)	TX_AUX_FIFO_UFLOW_EN	1'b0	1 = Aux TX FIFO underflow interrupt is enabled 0 = Aux TX FIFO underflow interrupt is disabled
31:4	-	28'h0	Reserved

- **CODEC RX_FIFO DMA I/O Control Register (CODEC_RX_AUX_DMA_IO_CTRL) – 0xBC0**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b1	1 for I/O mode, 0 for DMA mode.
1 (R)	IO_DMA_RW	1'b1	1: read from CODEC 0: write to CODEC
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the DATA_LEN set at the peripheral side doesn't match the DWORD size set in the DMA control.
3 (R/W)	RW_ENDIAN	1'b0	Reserved
31:4	-	28'h0	Reserved

- **CODEC RX_FIFO Control Register (CODEC_RX_AUX_FIFO_CTRL) – 0xBC8**

Bit	Name	Default	Description
1:0 (R/W)	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for DWORD.
7:2 (R/W)	FIFO_THD<5:0>	6'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
31:8	-	24'h0	Reserved

- **CODEC RX_FIFO Level Check Register (CODEC_RX_AUX_FIFO_LEVEL_CHK) – 0xBCC**

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC	4'h0	Stop check in DWORD.
9:4 (R/W)	-	6'h0	Reserved
13:10 (R/W)	FIFO_LC	4'h0	Low check in DWROD.
19:14 (R/W)	-	6'h0	Reserved
23:20 (R/W)	FIFO_HC	4'h0	High check in DWORD.
31:24	-	8'h0	Reserved

- **CODEC RX_FIFO Operation Register (CODEC_RX_AUX_FIFO_OP) – 0xBD0**

Bit	Name	Default	Description
0 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
1 (R/W)	FIFO_RESET	1'b0	Internally link to FIFO_START_INI. Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
31:2	-	30'h0	Reserved

- **AUX CODEC RXFIFO Status Register (CODEC_RX_AUX_FIFO_STS) – 0xBD4**

This register indicates the auxiliary RX FIFO status.

Bit	Name	Default	Description
0 (R/W)	RX_AUX_FIFO_FULL	1'b0	Auxiliary RX FIFO full status: 1 = AUX RX FIFO is in full state;

			0 = AUX RX FIFO is not in full state. It indicates the current auxiliary RX FIFO full status. Once the FIFO status changes, the status bit is cleared automatically.
1 (R/W)	RX_AUX_FIFO_EMPTY	1'b0	Auxiliary RX FIFO empty status: 1 = AUX RX FIFO is in empty states; 0 = AUX RX FIFO is not in empty states. It indicates the current auxiliary RX FIFO empty status. Once the FIFO status changes, the status bit is cleared automatically.
2 (R/W)	RX_AUX_FIFO_OFLOW	1'b0	Auxiliary RX FIFO overflow status 1 = AUX RX FIFO overflow takes place; 0 = AUX RX FIFO is not overflow. User can write 1'b1 to clear the register bit after the RX FIFO overflow takes place.
3 (R/W)	RX_AUX_FIFO_UFLOW	1'b0	Auxiliary RX FIFO underflow status 1 = AUX RX FIFO underflow takes place; 0 = AUX RX FIFO is not underflow. User needs to write 1'b1 to clear the register bit after the auxiliary RX FIFO underflow takes place.
31:4	-	28'h0	Reserved

- **AUX CODEC RXFIFO Interrupt Enable Register (CODEC_RX_AUX_FIFO_INT_EN) – 0xBD8**

Bit	Name	Default	Description
0 (R/W)	RX_AUX_FIFO_FULL_EN	1'b0	1 = AUX RX FIFO full interrupt is enabled 0 = AUX RX FIFO full interrupt is disabled
1 (R/W)	RX_AUX_FIFO_EMPTY_EN	1'b0	1 = AUX RX FIFO empty interrupt is enabled 0 = AUX RX FIFO empty interrupt is disabled
2 (R/W)	RX_AUX_FIFO_OFLOW_EN	1'b0	1 = AUX RX FIFO overflow interrupt is enabled 0 = AUX RX FIFO overflow interrupt is disabled
3 (R/W)	RX_AUX_FIFO_UFLOW_EN	1'b0	1 = AUX RX FIFO underflow interrupt is enabled 0 = AUX RX FIFO underflow interrupt is disabled
31:4	-	28'h0	Reserved

- **CODEC TX_FIFO DMA I/O Control Register (CODEC_TX_AUX_DMA_IO_CTRL) – 0xF80**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b1	1 for I/O mode, 0 for DMA mode.
1 (R)	IO_DMA_RW	1'b1	1: read from CODEC 0: write to CODEC
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the DATA_LEN set at the peripheral side doesn't match the DWORD size set in the DMA control.
3 (R/W)	RW_ENDIAN	1'b0	1: little endian write/read 0: big endian write/read
31:4	-	28'h0	Reserved

- **CODEC TX_FIFO Control Register (CODEC_TX_AUX_FIFO_CTRL) – 0xF88**

Bit	Name	Default	Description
1:0	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for

(R/W)			DWORD.
7:2 (R/W)	FIFO_THD<5:0>	6'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
31:8	-	24'h0	Reserved

- **CODEC TX_FIFO Level Check Register (CODEC_TX_AUX_FIFO_LEVEL_CHK) – 0XF8C**

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC	4'h0	Stop check in DWORD.
9:4 (R/W)	-	6'h0	Reserved
13:10 (R/W)	FIFO_LC	4'h0	Low check in DWROD.
19:14 (R/W)	-	6'h0	Reserved
23:20 (R/W)	FIFO_HC	4'h0	High check in DWORD.
31:24	-	8'h0	Reserved

- **CODEC TX_FIFO Operation Register (CODEC_TX_AUX_FIFO_OP) – 0XF90**

Bit	Name	Default	Description
0 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
1 (R/W)	FIFO_RESET	1'b0	Internally link to FIFO_START_INI. Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
31:2	-	30'h0	Reserved

- **CODEC TXFIFO Status Register (CODEC_TX_FIFO_STS) – 0xF94**

This register indicates the TX FIFO status.

Bit	Name	Default	Description
0 (R)	TX_FIFO_FULL	1'b0	TX FIFO full status: 1 = TX FIFO is in full state; 0 = TX FIFO is not in full state. It indicates the current TX FIFO full status. Once the FIFO status changes, the status bit is cleared automatically.
1 (R)	TX_FIFO_EMPTY	1'b0	TX FIFO empty status: 1 = TX FIFO is in empty states; 0 = TX FIFO is not in empty states. It indicates the current TX FIFO empty status. Once the FIFO status changes, the status bit is cleared automatically.
2 (R/W)	TX_FIFO_OFLOW	1'b0	TX FIFO overflow status 1 = TX FIFO overflow takes place; 0 = TX FIFO is not overflow. User can write 1'b1 to clear the register bit after the TX FIFO overflow takes place.
3 (R/W)	TX_FIFO_UFLOW	1'b0	TX FIFO underflow status 1 = TX FIFO underflow takes place; 0 = TX FIFO is not underflow. User needs to write 1'b1 to clear the register bit after the TX FIFO underflow takes place.

31:4	-	28'h0	Reserved
------	---	-------	----------

- **CODEC TXFIFO Interrupt Enable Register (CODEC_TX_FIFO_INT_EN) – 0xF98**

Bit	Name	Default	Description
0 (R/W)	TX_FIFO_FULL_EN	1'b0	1 = TX FIFO full interrupt is enabled 0 = TX FIFO full interrupt is disabled
1 (R/W)	TX_FIFO_EMPTY_EN	1'b0	1 = TX FIFO empty interrupt is enabled 0 = TX FIFO empty interrupt is disabled
2 (R/W)	TX_FIFO_OFLOW_EN	1'b0	1 = TX FIFO overflow interrupt is enabled 0 = TX FIFO overflow interrupt is disabled
3 (R/W)	TX_FIFO_UFLOW_EN	1'b0	1 = TX FIFO underflow interrupt is enabled 0 = TX FIFO underflow interrupt is disabled
31:4	-	28'h0	Reserved

- **CODEC RX_FIFO DMA I/O Control Register (CODEC_RX_DMA_IO_CTRL) – 0xFC0**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b1	1 for I/O mode, 0 for DMA mode.
1 (R)	IO_DMA_RW	1'b1	1: read from CODEC 0: write to CODEC
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the DATA_LEN set at the peripheral side doesn't match the DWORD size set in the DMA control.
3 (R/W)	RW_ENDIAN	1'b0	1: little endian write/read 0: big endian write/read
31:4	-	28'h0	Reserved

- **CODEC RX_FIFO Control Register (CODEC_RX_FIFO_CTRL) – 0xFC8**

Bit	Name	Default	Description
1:0 (R/W)	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for DWORD.
7:2 (R/W)	FIFO_THD<5:0>	6'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
31:8	-	24'h0	Reserved

- **CODEC RX_FIFO Level Check Register (CODEC_RX_FIFO_LEVEL_CHK) – 0xFCC**

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC	4'h0	Stop check in DWORD.
9:4 (R/W)	-	6'h0	Reserved
13:10 (R/W)	FIFO_LC	4'h0	Low check in DWROD.
19:14 (R/W)	-	6'h0	Reserved
23:20 (R/W)	FIFO_HC	4'h0	High check in DWORD.
31:24	-	8'h0	Reserved

- **CODEC RX_FIFO Operation Register (CODEC_RX_FIFO_OP) – 0xFD0**

Bit	Name	Default	Description
0 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
1 (R/W)	FIFO_RESET	1'b0	Internally link to FIFO_START_INI. Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
31:2	-	30'h0	Reserved

- **CODEC RXFIFO Status Register (CODEC_RX_FIFO_STS) – 0xFD4**

This register indicates the RX FIFO status.

Bit	Name	Default	Description
0 (R/W)	RX_FIFO_FULL	1'b0	RX FIFO full status: 1 = RX FIFO is in full state; 0 = RX FIFO is not in full state. It indicates the current RX FIFO full status. Once the FIFO status changes, the status bit is cleared automatically.
1 (R/W)	RX_FIFO_EMPTY	1'b0	RX FIFO empty status: 1 = RX FIFO is in empty states; 0 = RX FIFO is not in empty states. It indicates the current RX FIFO empty status. Once the FIFO status changes, the status bit is cleared automatically.
2 (R/W)	RX_FIFO_OFLOW	1'b0	RX FIFO overflow status 1 = RX FIFO overflow takes place; 0 = RX FIFO is not overflow. User can write 1'b1 to clear the register bit after the RX FIFO overflow takes place.
3 (R/W)	RX_FIFO_UFLOW	1'b0	RX FIFO underflow status 1 = RX FIFO underflow takes place; 0 = RX FIFO is not underflow. User needs to write 1'b1 to clear the register bit after the RX FIFO underflow takes place.
31:4	-	28'h0	Reserved

- **CODEC RXFIFO Interrupt Enable Register (CODEC_RX_FIFO_INT_EN) – 0xFD8**

Bit	Name	Default	Description
0 (R/W)	RX_FIFO_FULL_EN	1'b0	1 = RX FIFO full interrupt is enabled 0 = RX FIFO full interrupt is disabled
1 (R/W)	RX_FIFO_EMPTY_EN	1'b0	1 = RX FIFO empty interrupt is enabled 0 = RX FIFO empty interrupt is disabled
2 (R/W)	RX_FIFO_OFLOW_EN	1'b0	1 = RX FIFO overflow interrupt is enabled 0 = RX FIFO overflow interrupt is disabled
3 (R/W)	RX_FIFO_UFLOW_EN	1'b0	1 = RX FIFO underflow interrupt is enabled 0 = RX FIFO underflow interrupt is disabled
31:4	-	28'h0	Reserved

8.7 UART

8.7.1 Overview

There are 3 dedicated UARTs (Universal Asynchronous Receiver/Transmitter) and 5 USPs (Universal Serial Port) in Atlas™-II:

- UART0: full UART with DMA and Hardware flow control
- USP1~5: Universal Serial Port
- UART6 & 7: UART without DMA or Hardware flow control

UART0 supports the following features:

- 5, 6, 7 or 8 bits per character
- 1, or 2 stop bit detection and generation
- Internal baud rate generator and separate receiver clock input
- two clock divisor for more flexible clock division
- Modem control functions
- Transmit, receive, line status and modem control interrupts
- Line break detection and generation
- Internal loop back diagnostic functionality
- Independent 64 BYTE transmit and receive FIFOs for DMA

UART6&7 are the same as UART0, except that they do not support Modem control and DMA functionality.

8.7.2 Pin Description

The following table shows the UART0 input/output pins.

Table 78. UART0 Pin Descriptions

Pin Name	Direction	Description
TXD	Output	Transmit Data
RTS	Output	Request to Send,
DTR	Output	Data Terminal Ready
RXD	Input	Receive Data
RI	Input	Ring Indicator
DCD	Input	Data Carrier Detect
DSR	Input	Data Set Ready
CTS	Input	Clear To Send

Each pin of UART0 is muxed with other functions. Please refer to the section of “Pin Sharing” for detail information.

About The PIN connection of UART0 to external device, please refer to the RS232 specification.

NOTE: UART6&7 only have 2 pins: TXD and RXD.

8.7.3 Functional Description

The following figure shows the functional block diagram of the UART0:

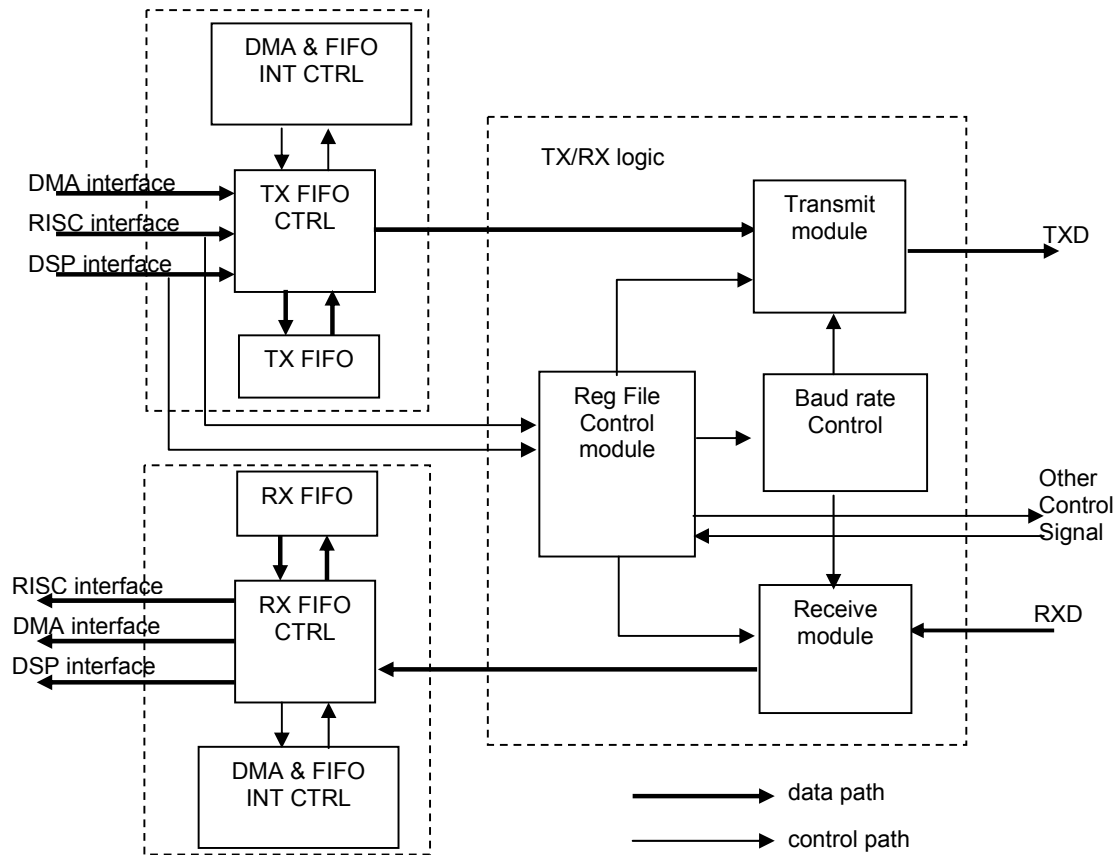


Figure 72. UART0 Functional Block Diagram

UART6&7 do not have the TX_FIFO and RX_FIFO blocks.

8.7.4 UART Registers

Each UART in Atlas™-II has the same set of registers, so there offset address is the same.

NOTE: The Modem control and FIFO control registers of UART6&7 are not functional. Accessing these registers will have no effect.

The actual address of the USP register is equal to the USP base address plus the offset address.

- UART0 base address 0x80000000
- UART6 base address 0x80100000
- UART7 base address 0x80110000

Table 79. UART Register Mapping

RISC Address <11:0>	DSP I/O Address <7:0>	Register	Description
0x40	0x20	UART_LINE_CTRL	UART line control register
0x44	0x22	UART_MODEM_CTRL	UART modem control register
0x48	0x24	UART_MODEM_STATUS	UART modem status register
0x4c	0x26	UART_TX_RX_EN	UART transmit & receive enable register

0x50	0x28	UART_DIVISOR	UART baud rate divisor register
0x54	0x2A	UART_INT_EN	UART interrupt enable register
0x58	0x2C	UART_INT_STATUS	UART interrupt status register
0x5c	0x2E	UART_RISC_DSP_MODE	UART accessing select register
0x60~FC	-	-	Reserved
0x100	0x40	UART_TX_DMA_IO_CTRL	UART TXFIFO DMA/IO register
0x104	0x42~0x43	UART_TX_DMA_IO_LEN	UART transmit data length register
0x108	0x44	UART_TXFIFO_CTRL	UART TXFIFO control register
0x10C	0x46~0x47	UART_TXFIFO_LEVEL_CHK	UART TXFIFO check level register
0x110	0x48	UART_TXFIFO_OP	UART TXFIFO operation register
0x114	0x4A	UART_TXFIFO_STATUS	UART TXFIFO status register
0x118	0x4C	UART_TXFIFO_DATA	UART TXFIFO bottom
0x11C	-	-	Reserved
0x120	0x50	UART_RX_DMA_IO_CTRL	UART RXFIFO DMA/IO register
0x124	0x52~0x53	UART_RX_DMA_IO_LEN	UART receive length register
0x128	0x54	UART_RXFIFO_CTRL	UART RXFIFO control register
0x12C	0x56~x57	UART_RXFIFO_LEVEL_CHK	UART RXFIFO check level register
0x130	0x58	UART_RXFIFO_OP	UART RXFIFO operation register
0x134	0x5A	UART_RXFIFO_STATUS	UART RXFIFO status register
0x138	0x5C	UART_RXFIFO_DATA	UART RXFIFO bottom
0x13C	-	-	Reserved

8.7.4.1 UART0 Control Registers

- **UART Line Control Register (UART_LINE_CTRL) – RISC: 0x40, DSP: 0x20**

Bit	Name	Default	Description
1:0 (R/W)	DATA_BIT_LEN	2'b0	2'b00: 5 data bits in one frame 2'b01: 6 data bit in one frame 2'b10: 7 data bits in one frame 2'b11: 8 data bits in one frame
2 (R/W)	STOP_BIT_LEN	1'b0	1'b0: 1 stop bit 1'b1: 2 stop bits
5:3			Reserved
6 (R/W)	SET_BREAK	1'b0	1'b0: TXD in normal transmit state 1'b1: TXD is forced to low level
15:7	-	-	Reserved
31:16 (R/W)	TIMOUT_NUM	16'h0	This register specifies the bit number for receive operation timeout case. If no more new data is received after those bit number time has passed since the last data is received. Receive timeout interrupt will happen.

- **UART Modem Control Register (UART_MODEM_CTRL) – RISC: 0x44, DSP: 0x22**

Bit	Name	Default	Description
0 (R/W)	UART_DTR	1'b0	Write this bit will change the status of DTR pin of UART 1'b0: set DTR to high level

			1'b1: set DTR to low level
1 (R/W)	UART_RTS	1'b0	Write this bit will change the status of RTS pin of UART 1'b0: set RTS to high level 1'b1: set RTS to low level
3:2	-	2'h0	Reserved
4 (R/W)	LOOP_BACK	1'b0	1'b0: no loop back 1'b1: the RXD is connected to not external pin but TXD internally and the transmit data is loop back to receive pin
31:5	-	27'h0	Reserved

- **UART Modem Status Register (UART_MODEM_STATUS) – RISC: 0x48, DSP: 0x24**

Bit	Name	Default	Description
0 (R/W)	DELTA_CTS	1'b0	This bit monitors the status change of CTS pin. Both the rising edge and falling edge of CTS will set this bit 1. Any read operation to UART_MODEM_STATUS will clear this bit
1 (R/W)	DELTA_DSR	1'b0	This bit monitors the status change of DSR pin. Both the rising edge and falling edge of DSR will set this bit 1. Any read operation to UART_MODEM_STATUS will clear this bit
2 (R/W)	DELTA_RI	1'b0	This bit monitors the status change of RI pin. Only the rising edge of RI will set this bit 1. Any read operation to UART_MODEM_STATUS will clear this bit
3 (R/W)	DELTA_DCD	1'b0	This bit monitors the status change of DCD pin. Both the rising edge and falling edge of DCD will set this bit 1. Any read operation to UART_MODEM_STATUS will clear this bit
4 (R/W)	CTS_STATUS	1'b0	This bit is the complement of the pin of CTS Its value follows that of CTS pin. If loop back mode is setting, this bit is connected to the output of RTS .
5 (R/W)	DSR_STATUS	1'b0	This bit is the complement of the pin of DSR Its value follows that of DSR pin If loop back mode is setting, this bit is connected to the output of DTR .
6 (R/W)	RI_STATUS	1'b0	This bit is the complement of the pin of RI Its value follows that of RI pin If loop back mode is setting, this bit is set to 0.
7 (R/W)	DCD_STATUS	1'b0	This bit is the complement of the pin of DCD Its value follows that of DCD pin If loop back mode is setting, this bit is set to 0.
31:8	-	24'h0	Reserved

- **UART Transmit/Receive Enable Register (UART_TX_RX_EN) – RISC: 0x4C, DSP: 0x26**

Bit	Name	Default	Description
0 (R/W)	RX_EN	1'b0	Receive enable bit 0: disabled 1: enabled
1 (R/W)	TX_EN	1'b0	Transmit enable bit 0: disabled 1: enabled

31:2	-	30'h0	Reserved
------	---	-------	----------

- **UART Clock Divisor Register (UART_CLK_DIV) – RISC: 0x50, DSP: 0x28**

Bit	Name	Default	Description
15:0 (R/W)	IOCLK_DIV	16'hffff	IOCLK_DIV = $f_{io_clock} / (\text{baud} * (\text{SAMPLE_DIV} + 1)) - 1$
21:16 (R/W)	SAMPLE_DIV	6'h0f	SAMPLE_DIV should be larger than or equal to 15
31:22	-	10'h0	Reserved

In normal mode, **SAMPLE_DIV** is always set to 15, and **IOCLK_DIV** is $f_{io_clock} / (\text{baud} * 16) - 1$.

In order to get the accurate high baud rate when the frequency of IOCLK is low, such as 12MHz, the value of **SAMPLE_DIV** can be changed to other value larger than 15. The exact high baud rate can be get if adjust both **SAMPLE_DIV** and **IOCLK_DIV**. For example, if IOCLK is 12MHz, to get baud rate of 115200, set the **SAMPLE_DIV** to 25, and **IOCLK_DIV** to 3.

- **UART Interrupt Enable Register (UART_INT_EN) – RISC: 0x54, DSP: 0x2A**

Bit	Name	Default	Description
0 (R/W)	RX_DONE_INT_EN	1'b0	Receive done interrupt enable 0: disabled 1: enabled
1 (R/W)	TX_DONE_INT_EN	1'b0	Transmit done interrupt enable 0: disabled 1: enabled
2 (R/W)	RX_OFLOW_INT_EN	1'b0	Receive overflow interrupt enable 0: disabled 1: enabled
3 (R/W)	TX_ALLOUT_INT_EN	1'b0	all data is transmitted interrupt enable 0: disabled 1: enabled
4 (R/W)	RX_IO_DMA_INT_EN	1'b0	IO/DMA receive interrupt enable 0: disabled 1: enabled
5 (R/W)	TX_IO_DMA_INT_EN	1'b0	IO/DMA transmit interrupt enable 0: disabled 1: enabled
6 (R/W)	RXFIFO_FULL_INT_EN	1'b0	Receive FIFO full interrupt enable 0: disabled 1: enabled
7 (R/W)	TXFIFO_EMPTY_INT_EN	1'b0	Transmit FIFO empty interrupt enable 0: disabled 1: enabled
8 (R/W)	RXFIFO_THD_INT_EN	1'b0	Receive FIFO threshold interrupt enable 0: disabled 1: enabled
9 (R/W)	TXFIFO_THD_INT_EN	1'b0	Transmit FIFO threshold interrupt enable 0: disabled 1: enabled
10 (R/W)	FRAME_ERR_INT_EN	1'b0	UART error frame interrupt enable

			0: disabled 1: enabled
11 (R/W)	RXD_BREAK_INT_EN	1'b0	RXD pin break interrupt enable 0: disabled 1: enabled
12 (R/W)	RX_TIMEOUT_INT_EN	1'b0	receive timeout interrupt enable 0: disabled 1: enabled
13 (R/W)	MODEM_STATUS_INT_EN	1'b0	Modem status interrupt enable 0: disabled 1: enabled
31:14	-	18'h0	Reserved

- **UART Interrupt Status Register (UART_INT_STATUS) – RISC: 0x58, DSP: 0x2C**

RISC/DSP writes a 1'b1 to the bit will clear that interrupt except **MODEM_STATUS_CHG** bit, any read operation to **MODEM_CTRL_STATUS** register will clear this bit.

Bit	Name	Default	Description
0 (R/W)	RX_DONE	1'b0	A valid data has been received in the RXFIFO interrupt (the valid data length is define by the RXFIFO_WIDTH) 0: invalid 1: valid
1 (R/W)	TX_DONE	1'b0	A valid data has been transmitted from the TXFIFO interrupt (the valid data length is define by the TXFIFO_WIDTH) 0: invalid 1: valid
2 (R/W)	RX_OFLOW	1'b0	RXFIFO overflow Interrupt 0: invalid 1: valid
3 (R/W)	TX_ALL_EMPTY	1'b0	all the data in both TXFIFO and tx shifter are sent out interrupt 0: invalid 1: valid
4 (R/W)	DMA_IO_RX_DONE	1'b0	RXFIFO has received all data of the data package, whose SIZE is defined by UART_RX_DMA_IO_LEN register 0: invalid 1: valid
5 (R/W)	DMA_IO_TX_DONE	1'b0	TXFIFO has transmitted all the data of a data package, whose SIZE is defined by UART_TX_DMA_IO_LEN register 0: invalid 1: valid
6 (R/W)	RXFIFO_FULL	1'b0	RXFIFO full interrupt 0: invalid 1: valid
7 (R/W)	TXFIFO_EMPTY	1'b0	TXFIFO empty interrupt 0: invalid 1: valid
8 (R/W)	RXFIFO_THD_REACH	1'b0	It's time to read RXFIFO when the number of data in the RXFIFO reaches the threshold.

			0: invalid 1: valid
9 (R/W)	TXFIFO_THD_REACH	1'b0	It's time to write TXFIFO when the number of data in the TXFIFO reaches the threshold. 0: invalid 1: valid
10 (R/W)	FRM_ERR	1'b0	Receive an error UART frame interrupt 0: invalid 1: valid
11 (R/W)	RXD_BREAK	1'b0	If this bit is become 1, it mean that the RXD line become low level for at least 10 bits time. Then receiving action must be stopped. Disable the RX_EN bit in the <i>USP_TX_RX_EN</i> register, then delay some time to double check this bit again, if no new break happens, enable the RX_EN , wait some time and to reset and restart the RXFIFO to flush the invalid data and prepare to accept the new data 0: invalid 1: valid
12 (R/W)	UART_RX_TIMEOUT	1'b0	If this bit is become 1, it means that since the last data had been received in the fifo, there is no more new data received for the time specified by the timeout bit number defined in the TIMOUT_NUM bits of the <i>UART_LINE_CTRL</i> register. 0: invalid 1: valid
13 (R)	MODEM_STATUS_CHG	1'b0	If this bit is valid, it means that one of delta status bit in the <i>UART_MODEM_STATUS</i> register has modified, any read operation to the register of <i>UART_MODEM_STATUS</i> will clear this bit 0: invalid 1: valid
31:14	-	19'h0	Reserved

- **UART RISC/DSP Mode Register (UART_RISC_DSP_MODE) – RISC: 0x5C, DSP: 0x2E**

This register can only be written by RISC. For DSP, it's read only.

Bit	Name	Default	Description
0 (R/W)	RISC_DSP_SEL	1'b0	0: the UART is accessed by RISC 1: the UART is accessed by DSP
31:1	-	31'h0	Reserved

8.7.4.2 UART0 TX_FIFO Registers

NOTE: The data flow of TX_FIFO is always from RISC/DSP/DMA to UART.

- **UART TX DMA I/O MODE Register (UART_TX_DMA_IO_CTRL) – RISC: 0x100, DSP: 0x40**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b1	1 for I/O mode, 0 for DMA mode.

3:1	-	3'b0	Reserved
5:4	TX_ENDIAN_MODE	2'b0	Reserved
31:6	-	26'h0	Reserved

- **UART TX DMA I/O Length Register (UART_TX_DMA_IO_LEN) –RISC: 0x104, DSP: 0x42~0x43**

Bit	Name	Default	Description
31:0 (R/W)	DATA_LEN	32'h0	The byte number of a DMA or I/O transfer. If set to zero, the I/O or DMA transfer works continuously until it is stopped.

- **UART TX FIFO Control Register (UART_TX_FIFO_CTRL) – RISC: 0x108, DSP: 0x44**

Bit	Name	Default	Description
1:0 (R/W)	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for DWORD.
7:2 (R/W)	FIFO_THD<5:0>	6'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
31:8	-	24'h0	Reserved

- **UART TX FIFO Level Check Register (UART_TX_FIFO_LEVEL_CHK) –RISC: 0x10C, DSP: 0x46~0x47**

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC<3:0>	4'h0	Stop check in DWORD.
9:4	-	6'h0	Reserved
13:10 (R/W)	FIFO_LC<3:0>	4'h0	Low check in DWOROD.
19:14	-	6'h0	Reserved
23:20 (R/W)	FIFO_HC<3:0>	4'h0	High check in DWORD.
31:24	-	8'h0	Reserved

- **UART TX FIFO Operation Register (UART_TX_FIFO_OP) – RISC: 0x110, DSP: 0x48**

This register is different from FIFO of other peripheral operation register and reset bit is at bit 0.

Bit	Name	Default	Description
0 (R/W)	FIFO_RESET	1'b0	Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
1 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
31:2	-	30'h0	Reserved

- **UART TX FIFO Status Register (UART_TX_FIFO_STATUS) – RISC: 0x114, DSP: 0x4A**

Bit	Name	Default	Description
5:0 (R)	FIFO_LEVEL	6'h0	The byte number of the valid data in the FIFO. In case that FIFO is full, the value of this register is 0, thus user must concatenate FIFO_FULL bit with this value to

			determine the actual data count in the FIFO.
6 (R)	FIFO_FULL	1'b0	FIFO full status, the FIFO is full when read out as 1. This bit is concatenated with FIFO_LEVEL to be the actual FIFO data count.
7 (R)	FIFO_EMPTY	1'b0	FIFO empty status. Equivalent to (FIFO_FULL, FIFO_LEVEL) == 0
31:8	-	24'h0	Reserved

- **UART TX FIFO Data Register (UART_TX_FIFO_DATA) – RISC: 0x118, DSP: 0x4C**

DSP can only access the low 16 bits of this register

Bit	Name	Default	Description
31:0 (W)	FIFO_DATA	32'h0	The FIFO data register, which is the bottom of the TX_FIFO.

8.7.4.3 UART0 RX_FIFO Registers

NOTE: The data flow of RX_FIFO is always from UART to RISC/DSP/DMA.

- **UART RX DMA I/O MODE Register (UART_RX_DMA_IO_CTRL) – RISC: 0x120, DSP: 0x50**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b0	1 for I/O mode, 0 for DMA mode.
1	-	1'b0	Reserved
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the DATA_LEN set at the peripheral side doesn't match the DWORD size set in the DMA control.
3		1'b0	reserved
5:4	RX_ENDIAN_MODE	2'b0	reserved
31:6	-	26'h0	Reserved

- **UART RX DMA I/O Length Register (UART_RX_DMA_IO_LEN) – RISC: 0x124, DSP: 0x52~0x53**

Bit	Name	Default	Description
31:0 (R/W)	DATA_LEN	32'h0	The byte number of a DMA or I/O transfer. If set to zero, the I/O or DMA transfer works continuously until it is stopped.

- **UART RX FIFO Control Register (UART_RX_FIFO_CTRL) – RISC: 0x128, DSP: 0x54**

Bit	Name	Default	Description
1:0 (R/W)	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for DWORD.
7:2 (R/W)	FIFO_THD<5:0>	6'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
31:8	-	24'h0	Reserved

- **UART RX FIFO Level Check Register (UART_RX_FIFO_LEVEL_CHK) – RISC: 0x12C, DSP:**

0x56~0x57

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC<3:0>	4'h0	Stop check in DWORD.
9:4>	-	6'h0	Reserved
13:10 (R/W)	FIFO_LC<3:0>	4'h0	Low check in DWROD.
19:14	-	6'h0	Reserved
23:20 (R/W)	FIFO_HC<3:0>	4'h0	High check in DWORD.
31:24	-	8'h0	Reserved

- **UART RX FIFO Operation Register (UART_RX_FIFO_OP) – RISC: 0x130, DSP: 0x58**

This register is different from FIFO of other peripheral operation register and reset bit is at bit 0.

Bit	Name	Default	Description
0 (R/W)	FIFO_RESET	1'b0	Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
1 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
31:2	-	30'h0	Reserved

- **UART RX FIFO Status Register (UART_RX_FIFO_STATUS) – RISC: 0x134, DSP: 0x5A**

Bit	Name	Default	Description
5:0 (R)	FIFO_LEVEL	6'h0	The byte number of the valid data in the FIFO. In case FIFO is full, the value of this register is 0, thus user must concatenate FIFO_FULL bit with this value to determine the actual data count in the FIFO.
6 (R)	FIFO_FULL	1'b0	FIFO full status, the FIFO is full when read out as 1. This bit is concatenated with FIFO_LEVEL to be the actual FIFO data count.
7 (R)	FIFO_EMPTY	1'b0	FIFO empty status. Equivalent to (FIFO_FULL, FIFO_LEVEL) == 0
31:8	-	24'h0	Reserved

- **UART RX FIFO Data Register (UART_RX_FIFO_DATA) – RISC: 0x138, DSP: 0x5C**

Bit	Name	Default	Description
31:0 (R)	FIFO_DATA	32'h0	The FIFO data register, which is the bottom of the RX_FIFO.

8.8 Universal Serial Port

8.8.1 Overview

Universal Serial ports are used for serial communication where only 1 bit is transmitted at a time. The advantage of a serial port is that it requires relatively few pins so it is often more cost-effective than parallel ports (especially in long-range communication). The serial port is a general-purpose interface that transmits or receives data a bit at one time and used for almost any type of devices. There are 3 main categories:

- PC peripherals – modem, mouse, and printer, etc.;

- Communication devices – Cable modem, ISDN, and xDSL, etc.;
- Embedded systems – A/D, D/A converters, RF modules, and serial EEPROMs, etc.

The Atlas™-II USP (Universal Serial Port) is a multi-function serial interface to communicate with many common serial ports. "Universal" does not mean that it can interface with any kind of serial devices, but compared to the existed designs, the Atlas™-1 USP has better expendability, configurability and flexibility.

The user can set the transfer frame parameters such as data length, transmitting data length etc, to configure out the right frame format. According to different connection protocols it can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as CRT terminals and modems, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, serial EEPROMs, or touch panel controllers.

There are a total of five (5) USPs (identified as USP1 through USP5) on the Atlas™-II Processor. These USPs are identical to each other, except that USP3 has an extra SIB (Serial Interface Bus) controller. The SIB controller is used to support connections to touch panel controllers such as the UCB1200/1300.

8.8.2 Supported Protocols and Devices

Depending on the clock, the USP can be configured in the following modes:

- Asynchronous (no clock is need)
 - Sample uses – UART, IrDA.
 - Max data rate = 115,200 bps.
- Synchronous – Master (clock is generated by USP)
 - Same uses – SPI bus, PCM bus.
 - Max clock frequency = 1/6 the I/O clock
- Synchronous – Slave (clock is generated by external chip)
 - Max clock frequency = 1/20 the I/O clock

Depending on the type of the serial bus, the Atlas™-II USP can support the following types of serial ports:

- ASYNC (UART and IrDA)
- SPI bus (such as serial EEPROM 25C040 and Philips PH2401)
- I2S bus (such as WM8978)

8.8.3 Pin Description

Each USP has 5 input/output pins:

Table 80. USP Pin Descriptions

Pin Name	Direction	Description
X_TXD_n	Bi-directional	TXD – Transmit Data
X_RXD_n	Bi-directional	RXD – Receive Data
X_TFS_n	Bi-directional	TFS – Transmit Frame Sync
X_RFS_n	Bi-directional	RFS – Receive Frame Sync
X_SCLK_n	Bi-directional	Serial Clock

NOTE: For the Pin Names, "n" means 0 to 5.

Each pin of USP can be configured to USP function or I/O function. Firstly you should configure pins of USP to be USP function. If you only use some of five pins for serial interface, the rest can be set to I/O mode. They are set in the register *USP_MODE_REG1*. You can read or write the *USP_PIN_IO_DATA* register if you configure it to be INPUT or OUTPUT mode.

8.8.4 Functional Description

There are five USPs in the Atlas™-II (USP1~5). Each USP has three interfaces:

1. RISC
2. DSP
3. DMA interface.

The USP can be accessed through either I/O or DMA interface. In I/O mode, either the RISC or DSP can access the USP. In DMA mode, it is controlled by the DMA controller in the I/O Bridge. Each USP can be configured to DMA interface independently.

The five USPs share four (4) DMA channels: channels 8~11. Please refer to the section of “DMA Controller” for more details.

All the registers can be accessed in I/O mode by the RISC or DSP, but they cannot access these register simultaneously. A register named *USP_RISC_DSP_MODE* can be only written by the RISC and it decide the USP is accessed and controlled by the RISC or DSP. If it is 0, the RISC accesses the the USP only. **Because the DSP data bus is 16-bit, if the DSP accesses the USP register it must access it two times to read or write a 32-bit register completely.**

There are three methods for the Atlas™-1 to access the TX_FIFO and RX_FIFO of USP,

- I/O mode with Interrupt
- I/O mode without Interrupt
- DMA mode

The TX_FIFO and the RX_FIFO can be accessed in I/O mode. Both the TX_FIFO and the RX_FIFO have the interrupt state registers to reflect their state. The operation in the I/O mode usually use these interrupt to decide whether to write or read. If the DMA access the TX_FIFO and the RX_FIFO, they cannot be accessed by the RISC/DSP.

NOTE: In all three modes, before transmitting operation, the TX_FIFO must first be reset by set the FIFO_RESET bit of TX_FIFO_OP register to 1, and then it is started by clear FIFO_RESET and set the FIFO_START bit of TX_FIFO_OP register. There is the same procedure for the RX_FIFO before operation it.

- I/O mode with Interrupt

The I/O mode with interrupt is suitable for those small data cases which are time-sensitive. There are four interrupts for the transmit operation, **TX_DONE**, **TXFIFO_SERVE**, **TXFIFO_EMPTY**, **TX_UFLOW**.

After one data is put from the TX_FIFO to the tx_shifter and transmitting starts, **TX_DONE** interrupt will happen. If the data in the TX_FIFO is less than parameter defined in the **FIFO_THD** bits of register *USP_TX_FIFO_CTRL*, the **TXFIFO_SERVE** interrupt will happen. If the TX_FIFO is empty, the **TXFIFO_EMPTY** interrupt will happen. **TX_UFLOW** is used for the underflow case of transmitting operation

After enabling them, RISC/DSP can monitor these interrupts when transmitting. If only one data is need to be transmitted at one time, the **TX_DONE** interrupt is enough for it, and if several data are transmitted at one time, the **TXFIFO_SERVE** interrupt or **TXFIFO_EMPTY** interrupt maybe useful to improve the

efficiency. If the **TXFIFO_EMPTY** interrupt happens, software can write several new data into the TX_FIFO.

There are also four interrupt for the receive operation, **RX_DONE**, **RXFIFO_SERVE**, **RXFIFO_FULL**, **RX_OFLOW**. The receiving operation is similar to the transmitting. After the interrupt happens, RISC/DSP can read the data out from the RX_FIFO. **RXFIFO_SERVE** interrupt and **RXFIFO_FULL** interrupt are useful for read several data from the RX_FIFO at one time.

- IO mode without Interrupt

This mode is suitable for the less data cases with time-insensitive. The RISC/DSP can poll the *USP_TXFIFO_STATUS* and *USP_RXFIFO_STATUS* register before transmitting or receiving operation. If the TX_FIFO is not full or RX_FIFO is not empty, RISC/DSP can write or read new data from the TX_FIFO or RX_FIFO.

- DMA mode

This mode is suitable for many data cases. If the transmitting/receiving FIFO is set to DMA mode, RISC/DSP cannot access them in the DMA mode. For transmitting, first RISC/DSP sets the number of the data to be transmitted in the *USP_TX_DMA_IO_LEN*, then resets and starts the TX_FIFO, DMA can be started. The DMA controller will transfer the data from the memory to TX_FIFO till all the data is transmitted out. The DMA receiving operation is similar to the transmitting. The number of the data to be received is stored in the *USP_RX_DMA_IO_LEN*. DMA controller will read data from the RX_FIFO to memory till all the data is finished receiving.

8.8.5 USP Registers

Each USP in Atlas™-II has the same set of registers, so there offset address is the same.

The actual address of the USP register is equal to the USP base address plus the offset address.

- USP1 base address 0xab010000
- USP2 base address 0xab020000
- USP3 base address 0xab030000
- USP4 base address 0xab040000
- USP5 base address 0xab050000

The following table shows the all the registers of USP and their offset address.

Table 81. USP Register Mapping

ARM9 Address <11:0>	DSP I/O Address <7:0>	Register	Description
0x0	0x00~0x01	USP_MODE1	USP mode setting register 1
0x4	0x02~0x03	USP_MODE2	USP mode setting register 2
0x8	0x04~0x05	USP_TX_FRAME_CTRL	USP transmit frame control register
0xC	0x06~0x07	USP_RX_FRAME_CTRL	USP receive frame control register
0x10	0x08	USP_TX_RX_ENABLE	USP transmit & receive enable register
0x14	0x0A(0x0B ¹)	USP_INT_ENABLE	USP interrupt enable register
0x18	0x0C(0x0D)	USP_INT_STATUS	USP interrupt register
0x1C	0x0E	USP_PIN_IO_DATA	USP pin I/O data register

0x20	0x10	USP_RISC_DSP_MODE	USP accessing select register
0x24	0x12~0x13	USP_AYSNC_PARAM_REG	USP ASYNC parameter
0x28~FC	-	-	Reserved
0x100	0x40	USP_TX_DMA_IO_CTRL	USP TXFIFO DMA/IO register
0x104	0x42~0x43	USP_TX_DMA_IO_LEN	USP transmit data length register
0x108	0x44	USP_TXFIFO_CTRL	USP TXFIFO control register
0x10C	0x46~0x47	USP_TXFIFO_LEVEL_CHK	USP TXFIFO check level register
0x110	0x48	USP_TXFIFO_OP	USP TXFIFO operation register
0x114	0x4A	USP_TXFIFO_STATUS	USP TXFIFO status register
0x118	0x4C	USP_TXFIFO_DATA	USP TXFIFO bottom
0x11C	-	-	Reserved
0x120	0x50	USP_RX_DMA_IO_CTRL	USP RXFIFO DMA/IO register
0x124	0x52~0x53	USP_RX_DMA_IO_LEN	USP receive length register
0x128	0x54	USP_RXFIFO_CTRL	USP RXFIFO control register
0x12C	0x56~x57	USP_RXFIFO_LEVEL_CHK	USP RXFIFO check level register
0x130	0x58	USP_RXFIFO_OP	USP RXFIFO operation register
0x134	0x5A	USP_RXFIFO_STATUS	USP RXFIFO status register
0x138	0x5C	USP_RXFIFO_DATA	USP RXFIFO bottom
Others		-	Reserved

8.8.5.1 USP Control Registers

- **USP Mode Register 1 (USP_MODE1) – ARM9: 0x0, DSP: 0x0~0x1**

Bit	Name	Default	Description
0 (R/W)	SYNC_MODE	1'b0	USP operation mode 0: asynchronous mode 1: synchronous mode
1 (R/W)	CLOCK_MODE	1'b0	USP clock mode 0: master mode (if USP is async mode, it must be set to 1) 1: slave mode, clock is input from other device
2 (R/W)	LOOP_BACK_EN	1'b0	USP transmit data loop back mode 0: no loop back 1: loop back (receive data comes from the transmit data)
3 (R/W)	HPSIR_EN	1'b0	IrDA function (SYNC_MODE must be set to 0) 0: disabled 1: enabled
4 (R/W)	ENDIAN_CTRL	1'b0	Transmit/receive data endian mode 0: big endian (MSBF) 1: small endian (LSBF)
5 (R/W)	USP_EN	1'b0	USP operation enable signal, when set to 0, it will reset the USP and it's transmit&receive control register. After it's set to 1, the reset of all these register is released, and then you can operate the USP.
6 (R/W)	RXD_ACT_EDGE	1'b0	Receive data is driven at 0: sclk rising edge 1: sclk falling edge
7 (R/W)	TXD_ACT_EDGE	1'b0	Transmit data is driven at 0: sclk rising edge

			1: sclk falling edge
8 (R/W)	RFS_ACT_LEVEL	1'b0	Receive sync signal (RFS) valid level 0: logic 0 1: logic 1
9 (R/W)	TFS_ACT_LEVEL	1'b0	Transmit sync signal (TFS) valid level 0: logic 0 1: logic 1
10 (R/W)	SCLK_IDLE_MODE	1'b0	In frame idle state, sclk mode 0: stop and keep to sclk idle level 1: continue to toggle
11 (R/W)	SCLK_IDLE_LEVEL	1'b0	In frame idle state, sclk will stop at level 0: logic 0 1: logic 1
12 (R/W)	SCLK_PIN_MODE	1'b0	SCLK pin operation mode 0: USP mode (sclk) 1: I/O mode
13 (R/W)	RFS_PIN_MODE	1'b0	RFS pin operation mode 0: USP mode (RFS) 1: I/O mode
14 (R/W)	TFS_PIN_MODE	1'b0	TFS pin operation mode 0: USP mode (TFS) 1: I/O mode
15 (R/W)	RXD_PIN_MODE	1'b0	RXD pin operation mode 0: USP mode (rxd) 1: I/O mode
16 (R/W)	TXD_PIN_MODE	1'b0	TXD pin operation mode 0: USP mode (txd) 1: I/O mode
17 (R/W)	SCLK_IO_MODE	1'b0	SCLK pin input/output mode (when SCLK_PIN_MODE=1) 0: output mode 1: input mode
18 (R/W)	RFS_IO_MODE	1'b0	RFS pin input/output mode (when RFS_PIN_MODE =1) 0: output mode 1: input mode
19 (R/W)	TFS_IO_MODE	1'b0	TFS pin input/output mode (when TFS_PIN_MODE =1) 0: output mode 1: input mode
20 (R/W)	RXD_IO_MODE	1'b0	RXD pin input/output mode (when RXD_PIN_MODE =1) 0: output mode 1: input mode
21 (R/W)	TXD_IO_MODE	1'b0	TXD pin input/output mode (when TXD_PIN_MODE =1) 0: output mode 1: input mode
<29:22> (R/W)	IRDA_WIDTH_DIV	8'h0	IrDA data pulse width register. If user wants to transmit/receive 1.6us width data by setting the IRDA_DATA_WIDTH in USP_MODE2, you must set this register according to the frequency of I/O clock. $IRDA_WIDTH_DIV = 1.6\mu s * f_{clock} + 1$
30(R/W)	IrDA_IDLE_LEVEL	1'b0	IrDA received data IDLE level 0: low level for idle state 1: high level for idle state

31(R/W)	TX_UFLOW_REPEA T	1'b0	TX underflow repeat mode 0: repeat the last transmitted data 1: repeat zero
---------	-----------------------------	------	---

- **USP Mode Register 2 (USP_MODE2) – ARM9: 0x4, DSP: 0x2-0x3**

Bit	Name	Default	Description
<7:0 > (R/W)	RXD_DELAY_LEN	8'h0	Delay clock number between the FS and the first receive data. It should be set to actual_delay_length in both the slave and master mode of RFS
<15:8 > (R/W)	TXD_DELAY_LEN	8'h0	Delay clock number between the FS and the first transmit data When TFS is slave, it must be set to actual_delay_length-1 When TFS is master, it must be set to actual_delay_length
16 (R/W)	ENA_CTRL_MODE	1'b0	USP_TX_RX_ENABLE register operation mode 1'b0: the TX_ENA/RX_ENA will not be cleared after the current operation finish. You only need to set to TX_ENA/RX_ENA bit only once before multi-time transmit/receive 1'b1: The TX_ENA/RX_ENA bit in the USP_TX_RX_ENABLE register will be cleared automatically after the previous operation finish, TX_ENA/RX_ENA bit must be set before a new transmit/receive operation begin.
17(R/W)	FRAME_CTRL_MODE	1'b0	USP transmit and receive operation control mode 0: New frame transmit/receive will begin only with new data in the TX_FIFO 1: after the USP operation start, frame transmit/receive will continue to send the data in the USP_TX_DATA no matter whether there are new data.
18(R/W)	TFS_SOURCE_MODE	1'b0	The sync signal source mode: 0: TFS is generated by hardware 1: TFS is generated by software from the upper 16-bit of the 32-bit data in the TX_FIFO.
19(R/W)	RFS_MS_MODE	1'b0	Receive sync signal source mode 0: RFS is master mode and output 1: RFS is slave mode and input from the external device
20(R/W)	TFS_MS_MODE	1'b0	Transmit sync signal source mode 0: TFS is master mode and output 1: TFS is slave mode and input from the external device
<30:21 > (R/W)	USP_CLK_DIVISOR	10'h0	USP serial clock divider. For ASYNC mode, USP_CLK_DIVISOR = $f_{io_clock} / (\text{baud} * \text{ASYNC_DIV2}) - 1$ For SYNC mode, USP_CLK_DIVISOR = $f_{io_clock} / (f_{sclk} * 2) - 1$
31(R/W)	IRDA_DATA_WIDTH	1'b0	IrDA logic 1 pulse width select (ASYNC_DIV2 must be set to 16 in IrDA mode)

			0 = data pulse width is 3/16 of a bit time wide 1 = data pulse width is 1.6 μ s
--	--	--	--

- USP Transmit Frame Control Register (USP_TX_FRAME_CTRL) – ARM9: 0x8, DSP: 0x4~0x5

Bit	Name	Default	Description
<7:0 > (R/W)	TX_DATA_LEN	8'h0	Transmitted data length in one frame. Must be set to actual_tx_data_length -1
<15:8 > (R/W)	TX_SYNC_LEN	8'h0	Valid length of transmitted sync signal (TFS) in one frame. Must be set to actual_tfs_valid_length -1
<23:16 > (R/W)	TX_FRAME_LEN	8'h0	Transmit frame length include active state and idle state. Must be set to actual_tx_frame_length -1
<28:24 > (R/W)	TX_SHIFTER_LEN	5'h0	Data length in the transmit shifter in one transmit operation. Must be set to actual_tx_shifter_length -1
29 (R/W)	SLAVE_CLK_SAMPLE	1'b0	When sampling it by I/O clock, setting this bit can avoid the glitch of the slave SCLK, 0: no filter for glitch of slave clock and the f_{max} of salve sclk is 1/8 of I/O clock 1: filter for glitch of slave clock and the f_{max} of salve sclk is 1/10 of I/O clock
<31:30>	-	2'h0	Reserved

In the ASYNC mode (RS232 or IrDA), this register has different meaning:

TX_DATA_LEN = Data bit number – 1

TX_FRAME_LEN = Start bit number + Data bit number + Stop bit number – 1

TX_SHIFTER_LEN = Data bit number – 1

The Parity bit is not supported in USP. And the Stop bit number is not supported of 1.5 bit and should be set to TXD_DELAY_LEN bits in the USP_MODE2 register.

- USP Receive Frame Control Register (USP_RX_FRAME_CTRL) – ARM9: 0xC, DSP: 0x6~0x7

Bit	Name	Default	Description
<7:0 > (R/W)	RX_DATA_LEN	8'h0	Received data length in one frame. Must be set to actual_rx_data_length -1
<15:8 > (R/W)	RX_FRAME_LEN	8'h0	Receive frame length include active and idle state. Must be set to actual_rx_frame_length -1
<20:16 > (R/W)	RX_SHIFTER_LEN	5'h0	Data length in the receive shifter in one transmit operation. Must be set to actual_rx_shifter_length -1
<31:21>	-	11'h0	Reserved

In the ASYNC mode (RS232 or IrDA), this register has different meaning:

RX_DATA_LEN = Data bit number – 1

RX_FRAME_LEN = Start bit number + Data bit number + Stop bit number – 1

RX_SHIFTER_LEN = Data bit number – 1

The Parity bit is not supported in USP. And the Stop bit number is not supported of 1.5 bit and should be set to TXD_DELAY_LEN bits in the USP_MODE2 register.

- **USP Transmit/Receive Enable Register (USP_TX_RX_ENABLE) – ARM9: 0x10, DSP: 0x8**

If ENA_CTRL_MODE=0 (USP_MODE_REG2), The TX_ENA/RX_ENA bits will be cleared automatically after the previous operation finishes, and they must be set before a new transmit/receive operation begin.

If ENA_CTRL_MODE =1, the TX_ENA/RX_ENA bits will not be cleared by writing a 0 to it. You need to set them only 1 time before you want to begin multi-times transmit/receive operation.

You can set RX_ENA/TX_ENA respectively according to requirement of the USP interfacing with external device

Bit	Name	Default	Description
0 (R/W)	RX_ENA	1'b0	Receive enable bit 0: disabled 1: enabled
1 (R/W)	TX_ENA	1'b0	Transmit enable bit 0: disabled 1: enabled
<31:2>	-	30'h0	Reserved

- **USP Interrupt Enable Register (USP_INT_ENABLE) – ARM9: 0x14, DSP: 0xA~0xB**

Bit	Name	Default	Description
0 (R/W)	RX_DONE_INT_EN	1'b0	Receive done interrupt enable 0: disabled 1: enabled
1 (R/W)	TX_DONE_INT_EN	1'b0	Transmit done interrupt enable 0: disabled 1: enabled
2 (R/W)	RX_OFLOW_INT_EN	1'b0	Receive overflow interrupt enable 0: disabled 1: enabled
3(R/W)	TX_UFLOW_INT_EN	1'b0	Transmit underflow interrupt enable 0: disabled 1: enabled
4(R/W)	RX_IO_DMA_INT_EN	1'b0	IO/DMA receive interrupt enable 0: disabled 1: enabled
5(R/W)	TX_IO_DMA_INT_EN	1'b0	IO/DMA transmit interrupt enable 0: disabled 1: enabled
6(R/W)	RXFIFO_FULL_INT_EN	1'b0	Receive FIFO full interrupt enable 0: disabled 1: enabled
7(R/W)	TXFIFO_EMPTY_INT_EN	1'b0	Transmit FIFO empty interrupt enable 0: disabled 1: enabled
8(R/W)	RXFIFO_THD_INT_EN	1'b0	Receive FIFO threshold interrupt enable 0: disabled

			1: enabled
9(R/W)	TXFIFO_THD_INT_EN	1'b0	Transmit FIFO threshold interrupt enable 0: disabled 1: enabled
10(R/W)	UART_ERR_INT_EN	1'b0	UART error frame interrupt enable 0: disabled 1: enabled
11(R/W)	USP_RX_TIMEOUT_INT_EN	1'b0	USP receive timeout interrupt enable 0: disabled 1: enabled
12(R/W)	USP_TX_ALLOUT_INT_EN	1'b0	USP all transmit out interrupt enable 0: disabled 1: enabled
<31:13>	-	19'h0	Reserved

- **USP Interrupt Status Register (USP_INT_STATUS) – ARM9: 0x18, DSP: 0xC~0xD**

ARM9/DSP writes a 1'b1 to the bit will clear that interrupt.

Bit	Name	Default	Description
0 (R/W)	RX_DONE	1'b0	A valid data has been received in the RXFIFO interrupt (the valid data length is define by the RXFIFO_WIDTH) 0: invalid 1: valid
1 (R/W)	TX_DONE	1'b0	A valid data has been transmitted from the TXFIFO interrupt (the valid data length is define by the TXFIFO_WIDTH) 0: invalid 1: valid
2 (R/W)	RX_OFLOW	1'b0	RXFIFO overflow Interrupt 0: invalid 1: valid
3 (R/W)	TX_UFLOW	1'b0	TXFIFO underflow interrupt 0: invalid 1: valid
4 (R/W)	DMA_IO_RX_DONE	1'b0	RXFIFO has received all data of the data package, whose SIZE is defined by USP_RX_DMA_IO_LEN_REG 0: invalid 1: valid
5 (R/W)	DMA_IO_TX_DONE	1'b0	TXFIFO has transmitted all the data of a data package, whose SIZE is defined by USP_TX_DMA_IO_LEN_REG 0: invalid 1: valid
6 (R/W)	RXFIFO_FULL	1'b0	RXFIFO full interrupt 0: invalid 1: valid
7 (R/W)	TXFIFO_EMPTY	1'b0	TXFIFO empty interrupt 0: invalid

			1: valid
8 (R/W)	RXFIFO_THD_REACH	1'b0	It's time to read USP_RXFIFO when the number of data in the USP_RXFIFO reaches the threshold. 0: invalid 1: valid
9 (R/W)	TXFIFO_THD_REACH	1'b0	It's time to write USP_TXFIFO when the number of data in the USP_TXFIFO reaches the threshold. 0: invalid 1: valid
10 (R/W)	UART_FRM_ERR	1'b0	Receive an error UART frame interrupt 0: invalid 1: valid
11 (R)	UART_BREAK	1'b0	If this bit is become 1, it mean that the RXD line become low level. Then receiving action must stop. Disable the RX_ENA bit in the TX_RX_ENA register, and wait UART_BREAK become to high, then delay some time to double check this bit again, if no new break happens, enable the RX_ENA, wait some time reset and restart the USP_RXFIFO to flush the invalid data and prepare to accept the new data
12(R/W)	USP_RX_TIMEOUT	1'b0	If this bit is become 1, it means that since the last data had been received in the fifo, there is no more new data received for then time specified by the timeout bit number in the USP_AYSNC_PARAM_REG.
13(R/W)	USP_TX_ALLOUT	1'b0	If this bit is become 1, it means that the data in both tx_fifo and tx shifter has been transmitted out. 0: disabled 1: enabled
31:13	-	19'h0	Reserved

- **USP Pin I/O Data Register (USP_PIN_IO_DATA) – ARM9: 0x1C, DSP: 0xE**

These bits can only be accessed by the ARM9/DSP, when the corresponding pins of the USP are in I/O mode.

Bit	Name	Default	Description
0 (R/W)	RFS_PIN_VALUE	1'b0	Pin value of GPIO function (RFS)
1 (R/W)	TFS_PIN_VALUE	1'b0	Pin value of GPIO function (TFS)
2 (R/W)	RXD_PIN_VALUE	1'b0	Pin value of GPIO function (RXD)
3 (R/W)	TXD_PIN_VALUE	1'b0	Pin value of GPIO function (TXD)
4 (R/W)	SCLK_PIN_VALUE	1'b0	Pin value of GPIO function (SCLK)
<31:5>	-	27'h0	Reserved

- **USP ARM9/DSP Mode Register (USP_RISC_DSP_MODE) – ARM9: 0x20, DSP: 0x10**

This register can only be written by ARM9. For DSP, it's read only.

Bit	Name	Default	Description
0 (R/W)	RISC_DSP_SEL	1'b0	0: the USP is accessed by ARM9 1: the USP is accessed by DSP
31:1	-	31'h0	Reserved

- **USP ASYNC PARAMETER Register (USP_AYSNC_PARAM_REG) – ARM9: 0x24, DSP: 0x12**

This register can only be written by ARM9. For DSP, it's read only.

Bit	Name	Default	Description
15:0	AYSNC_TIMEOUT_NUM	16'hff	This parameter specifies the TIMEOUT bit number for the receive operation. Since the last bit received, if there is no more data is received for TIMEOUT bit number time, the rx timeout interrupt will be triggered.
21:16	ASYNC_DIV2	6'b0	The parameter is used to cooperate with the USP_CLK_DIVISOR in the USP_MODE2 register to generate the right baud rate for async mode when ioclk is low frequency.
31:1	-	31'h0	Reserved

8.8.5.2 USP TX_FIFO register

NOTE: The data flow of TX_FIFO is always from ARM9/DSP/DMA to USP.

- **USP TX DMA I/O MODE register (USP_TX_DMA_IO_CTRL) –ARM9: 0x100, DSP: 0x40**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b1	1 for I/O mode, 0 for DMA mode.
3:1		3'b0	reserved
5:4	TX_ENDIAN_MODE	2'b0	Reserved
<31:6>	-	26'h0	Reserved

- **USP TX DMA I/O length register (USP_TX_DMA_IO_LEN) –ARM9: 0x104, DSP: 0x42~0x43**

Bit	Name	Default	Description
<31:0 > (R/W)	DATA_LEN	32'h0	The byte number of a DMA or I/O transfer. If set to zero, the I/O or DMA transfer works continuously until it is stopped.

- **USP TX FIFO control register (USP_TX_FIFO_CTRL) –ARM9: 0x108, DSP: 0x44**

Bit	Name	Default	Description
<1:0 > (R/W)	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for DWORD.
<6:2 > (R/W)	FIFO_THD<4:0>	5'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
<31:7>	-	25'h0	Reserved

- **USP TX FIFO level check register (USP_TX_FIFO_LEVEL_CHK) –ARM9: 0x10C, DSP: 0x46~0x47**

Bit	Name	Default	Description
<2:0 > (R/W)	FIFO_SC<9:0>	3'h0	Stop check in DWORD.
<9:3>	-	7'h0	reserved
12:10 (R/W)	FIFO_LC<9:0>	3'h0	Low check in DWROD.
<19:13>	-	7'h0	reserved
<22:20> (R/W)	FIFO_HC<9:0>	3'h0	High check in DWORD.
<31:23>	-	9'h0	Reserved

- **USP TX FIFO operation register (USP_TX_FIFO_OP) –ARM9: 0x110, DSP: 0x48**

This register is different from FIFO of other peripheral operation register and reset bit is at bit 0.

Bit	Name	Default	Description
0 (R/W)	FIFO_RESET	1'b0	Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
1 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
<31:2>	-	30'h0	Reserved

- **USP TX FIFO status register (USP_TX_FIFO_STATUS) –ARM9: 0x114, DSP: 0x4a**

Bit	Name	Default	Description
4:0 (R)	FIFO_LEVEL	5'h0	The byte number of the valid data in the FIFO. In case that FIFO is full, the value of this register is 0, thus user must concatenate FIFO_FULL bit with this value to determine the actual data count in the FIFO.
5 (R)	FIFO_FULL	1'b0	FIFO full status, the FIFO is full when read out as 1. This bit is concatenated with FIFO_LEVEL to be the actual FIFO data count.
6 (R)	FIFO_EMPTY	1'b0	FIFO empty status. Equivalent to (FIFO_FULL, FIFO_LEVEL) == 0
<31:7>	-	25'h0	Reserved

- **USP TX FIFO Data Register (USP_TX_FIFO_DATA) –ARM9: 0x118, DSP: 0x4c**

Bit	Name	Default	Description
<31:0 > (W)	FIFO_DATA	32'h0	The FIFO data register, which is the bottom of the TX_FIFO.

8.8.5.3 USP RX_FIFO register

NOTE: The data flow of RX_FIFO is always from USP to ARM9/DSP/DMA.

- **USP RX DMA I/O MODE register (USP_RX_DMA_IO_CTRL) –ARM9: 0x120, DSP: 0x50**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b0	1 for I/O mode, 0 for DMA mode.
1	-	1'b0	reserved
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the DATA_LEN set at the peripheral side doesn't match the DWORD size set in the DMA control.
3		1'b0	reserved
5:4	RX_ENDIAN_MODE	2'b0	
<31:6>	-	26'h0	Reserved

- **USP RX DMA I/O length register (USP_RX_DMA_IO_LEN) –ARM9: 0x124, DSP: 0x52~0x53**

Bit	Name	Default	Description
<31:0 > (R/W)	DATA_LEN	32'h0	The byte number of a DMA or I/O transfer. If set to zero, the I/O or DMA transfer works continuously until it is stopped.

- **USP RX FIFO control register (USP_RX_FIFO_CTRL) –ARM9: 0x128, DSP: 0x54**

Bit	Name	Default	Description
<1:0 > (R/W)	FIFO_WIDTH<1:0>	2'h0	Data width of FIFO: 0 for byte, 1 for word and 2 for DWORD.
<6:2 > (R/W)	FIFO_THD<4:0>	5'h0	A threshold in byte to trigger an interrupt. An interrupt is triggered when the count of data in the FIFO reaches the threshold.
<31:7>	-	25'h0	Reserved

- **USP RX FIFO level check register (USP_RX_FIFO_LEVEL_CHK) –ARM9: 0x12C, DSP: 0x56~0x57**

Bit	Name	Default	Description
<2:0 > (R/W)	FIFO_SC<9:0>	3'h0	Stop check in DWORD.
<9:3>	-	7'h0	reserved
<12:10 > (R/W)	FIFO_LC<9:0>	3'h0	Low check in DWROD.
<19:13>	-	7'h0	reserved
<22:20> (R/W)	FIFO_HC<9:0>	3'h0	High check in DWORD.
<31:23>	-	9'h0	Reserved

- **USP RX FIFO operation register (USP_RX_FIFO_OP) –ARM9: 0x130, DSP: 0x58**

This register is different from FIFO of other peripheral operation register and reset bit is at bit 0.

Bit	Name	Default	Description
-----	------	---------	-------------

0 (R/W)	FIFO_RESET	1'b0	Set to 1 to stop the FIFO and reset the FIFO internal status, including the relevant interrupt status. Set to 0 in normal operation.
1 (R/W)	FIFO_START	1'b0	Start the read/write transfer when this bit is declared.
<31:2>	-	30'h0	Reserved

- **USP RX FIFO status register (USP_RX_FIFO_STATUS) –ARM9: 0x134, DSP: 0x5a**

Bit	Name	Default	Description
<4:0> (R)	FIFO_LEVEL	5'h0	The byte number of the valid data in the FIFO. In case FIFO is full, the value of this register is 0, thus user must concatenate FIFO_FULL bit with this value to determine the actual data count in the FIFO.
5 (R)	FIFO_FULL	1'b0	FIFO full status, the FIFO is full when read out as 1. This bit is concatenated with FIFO_LEVEL to be the actual FIFO data count.
6 (R)	FIFO_EMPTY	1'b0	FIFO empty status. Equivalent to (FIFO_FULL, FIFO_LEVEL) == 0
<31:7>	-	25'h0	Reserved

- **USP RX FIFO Data Register (USP_RX_FIFO_DATA) –ARM9: 0x138, DSP: 0x5c~0x5d**

Bit	Name	Default	Description
<31:0> (R)	FIFO_DATA	32'h0	The FIFO data register, which is the bottom of the RX_FIFO.

8.9 CAN Bus Controller

8.9.1 Overview

The CANBUS controller in Atlas™-II is a stand-alone controller for the CAN Controller Area Network used in the automotive industry and in a number of other industrial environments. It provides an interface between a microprocessor and a CAN Bus which carries out all the actions of data encoding/decoding (including serialisation/deserialisation of data, bit stuffing/unstuffing), message management (acceptance filtering, acknowledgement, error detection and signaling and re-transmission), bit timing and synchronization involved in transmitting and receiving information over a CAN network.

- Supports full CAN 2.0 – both 2.0A (equivalent to CAN 1.2) and 2.0B
- Supports both 11-bit and 29-bit identifiers
- Supports bit rates from less than 125Kbaud to more than 1Mbaud
- 64 byte Receive FIFO
- Software-driven bit-rate detection (offering hot plug-in support)
- Acceptance filtering
- Single-shot transmission option
- Listen-only mode
- Reception of 'own' messages
- Error interrupt generated for each CAN bus error
- Read/write error counters
- Last error register
- Programmable error limit warning
- DMA Capability

Atlas™-II has two CAN Bus Controller, CANBUS0 and CANBUS1. These two controllers can work at the same time. Besides the normal IO transfer mode, canbus also support the DMA transfer. A group of registers are implemented to control the DMA transfer.

There are two DMA channels (Channel 0, 1) shared by CANBUS0 and CANBUS1: Channel 0 for receiving and Channel 1 for sending. At one time, only one CAN Bus Controller can use the DMA function. Please refer to the section of "DMA Controller" for more details.

8.9.2 Pin Description

Each CAN Bus Controller has 2 input/output pins:

Table 82. CAN Bus Pin Descriptions

Pin Name	Direction	Description
X_CAN_TXD_n	Output	Transmit Data
X_CAN_RXD_n	Input	Receive Data

NOTE: For the Pin Names, "n" means 0 to 1.

The following figure shows the connection of CAN Bus interface.

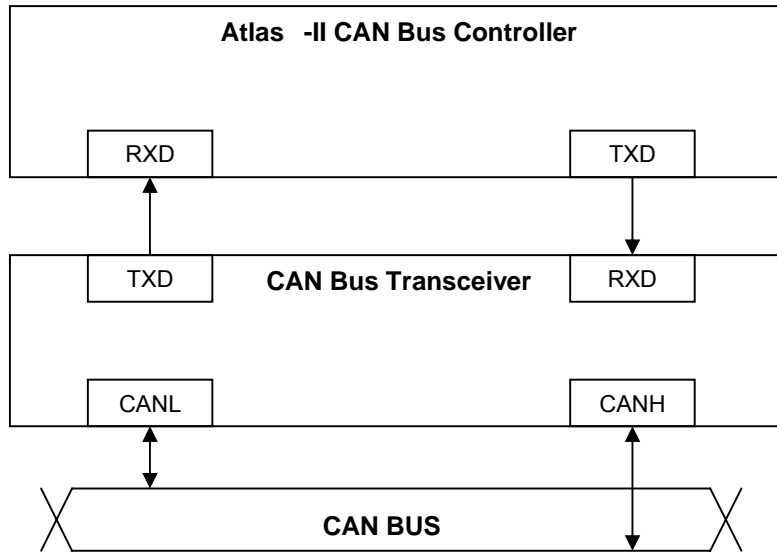


Figure 73. CAN Bus Interface Connection

8.9.3 CAN Bus Registers

The following table shows the address mapping for CAN Bus Controllers.

Table 83. CAN Bus Address Mapping

Block	Address
CANBUS0 I/O	0x800D0000~0x800D0800
CANBUS1 I/O	0x800E0000~0x800E0800
DMA Output	0x800D0F00~0x800D0FFF
DMA Input	0x800E0F00~0x800E0FFF

8.9.3.1 CAN Bus Control Registers

Table 84. CAN Bus DMA Register Mapping

RISC Address <11:0>	Register	Description
0x0000	CANBUSn_MODE	CANBUS Mode Register
0x0004	CANBUSn_CMD	CANBUS Command Register
0x0008	CANBUSn_SR	CANBUS Status Register
0x000C	CANBUSn_IR	CANBUS Interrupt Register
0x0010	CANBUSn_IER	CANBUS Interrupt Enable Register
0x0014	-	Reserved
0x0018	CANBUSn_BTIM0	CANBUS Timing Control Register0
0x001C	CANBUSn_BTIM1	CANBUS Timing Control Register1
0x0020	-	Reserved
0x0024	-	Reserved
0x0028	-	Reserved
0x002C	CANBUSn_ALC	CANBUS Arbitration Lost Capture Register
0x0030	CANBUSn_ECC	CANBUS Error Code Capture Register
0x0034	CANBUSn_EWLR	CANBUS Error Warning Limit Register

0x0038	CANBUSn_RXERR	CANBUS Receive Error Counter Register
0x003C	CANBUSn_TXERR	CANBUS Transmit Error Counter Register
0x0040~0x0070	CANBUSn_TBUF	CANBUS Transmit Buffer Note: Read Back from 0x180~0x1B0
	CANBUSn_RWIN	CANBUS Receive Window
0x0040	CANBUSn_ACR0	CANBUS Acceptance Code Registers0
0x0044	CANBUSn_ACR1	CANBUS Acceptance Code Registers1
0x0048	CANBUSn_ACR2	CANBUS Acceptance Code Registers2
0x004C	CANBUSn_ACR3	CANBUS Acceptance Code Registers3
0x0050	CANBUSn_AMR0	CANBUS Acceptance Mask Registers0
0x0054	CANBUSn_AMR1	CANBUS Acceptance Mask Registers1
0x0058	CANBUSn_AMR2	CANBUS Acceptance Mask Registers2
0x005C	CANBUSn_AMR3	CANBUS Acceptance Mask Registers3
0x0074	CANBUSn_RMC	CANBUS Receive Message Counter Registers
0x0078	CANBUSn_RBSA	CANBUS Receive Buffer Start Address Register
0x007C	-	Reserved
0x0080~0x017C	CANBUSn_RFIFO	CANBUS Receive FIFO
0x0180~0x 1B0	CANBUSn_TBUF_R	CANBUS Transmit Buffer
Other	-	Reserved

NOTE:

1. n = 0/1
2. The registers marked in gray are only valid in Reset Mode.

- **CANBUS Mode Register (CANBUSn_MODE) – 0x0000**

Bit	Name	Default	Description
0 (R/W)	RM	1'b1	1 -- Reset Mode selected. Any message currently being transmitted or received is aborted and Reset Mode is entered. 0 -- Normal operation. The CANBUS controller returns to Operating Mode on the '1-to-0' transition of this bit.
1 (R/W)	LOM	1'b0	1 -- Listen Only enabled. In this mode, the CANBUS controller does not send an acknowledge to the CAN bus, even when a message is received successfully. 0 -- Normal operation. The error counters are stopped at the current value.
2 (R/W)	STM	1'b0	1 -- Self Test enabled. In this mode, a full node test is possible without any other active node on the bus using

			the self reception request command. The CANBUS controller will perform a successful transmission, even if no acknowledge is received. 0 -- Normal operation. An acknowledge is required for successful transmission.
3 (R/W)	AFM	1'b0	1 -- Single Filter. Receive data filtered using one 4-byte filter. 0 -- Dual Filter. Receive data filtered using two shorter filters.
31:4	-	28'h0	Reserved

- **CANBUS Command Register (CANBUSn_CMD) – 0x0004**

Bit	Name	Default	Description
0 (W)	TR	1'b1	Set to '1' when a message is to be transmitted.
1 (W)	AT	1'b0	Set to '1' to cancel the next transmission request, provided this is not already in progress.
2 (W)	RRB	1'b0	Set to '1' to release the Receive Buffer.
3 (W)	CDO	1'b0	Set to '1' to clear the data overrun condition signaled by the Data Overrun Status bit. <i>Note:</i> No further Data Overrun Interrupt will be generated while the Data Overrun Status bit remains set.
4 (W)	SRR	1'b0	Set to '1' when a message is to be transmitted and received simultaneously.
31:5	-	27'h0	Reserved

- **CANBUS Status Register (CANBUSn_SR) – 0x0008**

Bit	Name	Default	Description
0 (R)	RBS	1'b1	1 -- Receive Buffer Full. One or more complete messages are available to be read from the Receive FIFO via the Receive Buffer. 0 -- Receive Buffer Empty. No message currently available to be read.
1 (R)	DOS	1'b0	1 -- Data Overrun. A message has been lost because there was not enough space for that message in the Receive FIFO. 0 -- No data overrun has occurred since the last Clear Data Overrun command was given.
2 (R)	TBS	1'b1	1 -- Transmit Buffer Released. The CPU may write a message to the Transmit Buffer. 0 -- Transmit Buffer Locked. The CPU cannot access the Transmit Buffer because a message is either waiting for transmission or is in the process of being transmitted.
3 (R)	TCS	1'b1	1 -- The last requested transmission has been successfully completed. 0 -- The last requested transmission has not been completed yet.
4 (R)	RS	1'b1	1 -- The CANBUS controller is in the process of receiving a message. 0 -- Nothing is currently being received.
5 (R)	TS	1'b1	1 -- The CANBUS controller is in the process of

			transmitting a message. 0 -- No message is being transmitted.
6 (R)	ES	1'b0	1 -- At least one of the error counters has reached or exceeded the CPU warning limit defined by the Error Warning Limit Register (EWL). 0 -- Both error counters are below the warning limit.
7 (R)	BS	1'b0	1 -- The CANBUS controller is in 'Bus Off' state and is not involved in bus activities. 0 -- The CANBUS controller is involved in bus activities.
31:8	-	24'h0	Reserved

- **CANBUS Interrupt Register (CANBUSn_IR) – 0x000C**

Bit	Name	Default	Description
0 (R)	RI	1'b1	Set whenever the Receive Buffer contains one or more messages – provided the RIE bit is set within the Interrupt Enable Register. Cleared when the release Receive Buffer command is issued, provided there is no further data to read in the Receive Buffer.
1 (R)	TI	1'b0	Set whenever the Transmit Buffer Status changes from '0-to-1' (released)– provided the TIE bit is set within the Interrupt Enable Register.
2 (R)	EI	1'b0	Set on every change (set or clear) of either the Bus Status or Error Status bits – provided the EIE bit is set within the Interrupt Enable Register.
3 (R)	DOI	1'b0	Set on a '0-to-1' transition of the Data Overrun Status bit – provided the DOIE bit is set within the Interrupt Enable Register.
4 (R)	WUI	1'b0	Set when bus activity is detected while the CAN controller is sleeping – provided the WUIE bit is set within the Interrupt Enable Register.
5 (R)	EPI	1'b0	Set when the CANBUS controller re-enters error active state after being in error passive state or when at least one error counter exceeds the protocol-defined level of 127 – provided the EPIE bit is set within the Interrupt Enable Register.
6 (R)	ALI	1'b0	Set when the CANBUS controller loses arbitration and becomes a receiver – provided the ALIE bit is set within the Interrupt Enable Register.
7 (R)	BEI	1'b0	Set when the CANBUS controller detects an error on the CAN-bus – provided the BEIE bit is set within the Interrupt Enable Register.
31:8	-	24'h0	Reserved

- **CANBUS Interrupt Enable Register (CANBUSn_IER) – 0x000C**

Bit	Name	Default	Description
0 (R/W)	RIE	1'b1	When set to '1', an interrupt will be generated when the Receive Buffer Status goes from '0' to '1' ('full'). When set to '0', the interrupt is disabled.
1 (R/W)	TIE	1'b0	When set to '1', an interrupt will be generated when a message has been successfully transmitted or the Transmit Buffer is accessible again. When set to '0', the

			interrupt is disabled.
2 (R/W)	EIE	1'b0	When set to '1', an interrupt will be generated when the bus status or error status bits change. When set to '0', the interrupt is disabled.
3 (R/W)	DOIE	1'b0	When set to '1', an interrupt will be generated when the Data Overrun Status bit is set. When set to '0', the interrupt is disabled.
4 (R/W)	WUIE	1'b0	When set to '1', an interrupt will be generated when the sleeping CANBUS controller wakes up. When set to '0', the interrupt is disabled.
5 (R/W)	EPIE	1'b0	When set to '1', an interrupt will be generated when the error status of the CANBUS controller changes from error active to error passive or vice versa. When set to '0', the interrupt is disabled.
6 (R/W)	ALIE	1'b0	When set to '1', an interrupt will be generated when the CANBUS controller loses arbitration. When set to '0', the interrupt is disabled.
7 (R/W)	BEIE	1'b0	When set to '1', an interrupt will be generated when a bus error has been detected. When set to '0', the interrupt is disabled.
31:8	-	24'h0	Reserved

- **CANBUS Timing Register0 (CANBUSn_BTIM0) – 0x0018**

The CAN specification describes the bit period as being composed of a Synchronization segment, a Propagation segment and two Phase Buffers. For the purposes of *CANBUSn_BTIM1*, the bit period is seen as being composed of the Synchronization segment plus **TSEG1** and **TSEG2**, where **TSEG1** equals the Propagation segment plus the first Phase Buffer and **TSEG2** is the second Phase Buffer as shown in the following diagram.

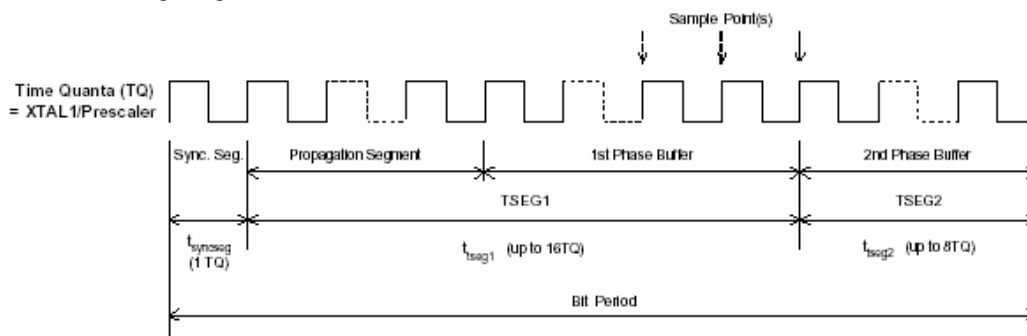


Figure 74. CAN Bus Timing

Bit	Name	Default	Description
5:0 (R/W)	BRP	6'h0	The Baud Rate Prescaler defines the 'time quantum' TQ of the CAN clock as a multiple of the input clock period. The time quantum of the CAN clock is given by: $TQ = 2 \times t_{clk} \times (BRP + 1)$ where t_{clk} = time period of the XTAL1 frequency = $1/f_{xtal1}$
7:6 (R/W)	SJW	2'h0	The Synchronization Jump Width defines the maximum number of time quanta(TQ) by which a bit period may be shortened or lengthened in attempting to re-synchronize on the relevant signal edge (recessive to dominant) of the current transmission.

31:8	-	24'h0	Reserved
------	---	-------	----------

- **Bus Timing Registers1 (CANBUSn_BTIM1) – 0x001C**

Bit	Name	Default	Description
6:0 (R/W)	TSEG1 (3:0)	4'h0	TSEG1 and TSEG2 define the length of the bit period by giving the number of time quanta up to and after the point(s) at which incoming data will be sampled. In terms of TSEG1 and TSEG2, the parameters $t_{syncseg}$, t_{seg1} and t_{seg2} in above figure are: $t_{syncseg} = 1 \times TQ$ $t_{seg1} = TQ \times (TSEG1 + 1)$ $t_{seg2} = TQ \times (TSEG2 + 1)$ Note: In theory, it is possible to define bit periods of between 3 and 25 TQ through these register settings. However the bit periods used in practice are required to follow the BOSCH standard, which defines bit periods between 8 and 25 TQ in length. The TSEG2 should not be set to 0 when configing the bit rate.
	TSEG2 (6:4)	3'h0	
7 (R/W)	SAM	1'b0	1 -- The bus will be sampled three times. (This is recommended for low/medium speed buses (class A or B).) 0 -- The bus will be sampled once. (This is recommended for high speed buses (SAE class C).)
31:8	-	24'h0	Reserved

- **Arbitration Lost Capture Register(CANBUSn_ALC) – 0x0018**

Bit	Name	Default	Description
4:0 (R)	ALC	5'h0	5'b00000 -- Arbitration lost in 1st bit of identifier (ID.28). 5'b00001 -- Arbitration lost in 2nd bit of identifier (ID.27). 5'b00010 -- Arbitration lost in 3rd bit of identifier (ID.26). 5'b00011 -- Arbitration lost in 4th bit of identifier (ID.25). 5'b00100 -- Arbitration lost in 5th bit of identifier (ID.24). 5'b00101 -- Arbitration lost in 6th bit of identifier (ID.23). 5'b00110 -- Arbitration lost in 7th bit of identifier (ID.22). 5'b00111 -- Arbitration lost in 8th bit of identifier (ID.21). 5'b01000 -- Arbitration lost in 9th bit of identifier (ID.20). 5'b01001 -- Arbitration lost in 10th bit of identifier (ID.19). 5'b01010 -- Arbitration lost in 11th bit of identifier (ID.18). 5'b01011 -- Arbitration lost in SRTR bit 1. 5'b01100 -- Arbitration lost in IDE bit. 5'b01101 -- Arbitration lost in 12th bit of identifier (ID.17). 5'b01110 -- Arbitration lost in 13th bit of identifier (ID.16). 5'b01111 -- Arbitration lost in 14th bit of identifier (ID.15). 5'b10000 -- Arbitration lost in 15th bit of identifier (ID.14). 5'b10001 -- Arbitration lost in 16th bit of identifier (ID.13). 5'b10010 -- Arbitration lost in 17th bit of identifier (ID.12). 5'b10011 -- Arbitration lost in 18th bit of identifier (ID.11). 5'b10100 -- Arbitration lost in 19th bit of identifier (ID.10). 5'b10101 -- Arbitration lost in 20th bit of identifier (ID.9). 5'b10110 -- Arbitration lost in 21st bit of identifier (ID.8). 5'b10111 -- Arbitration lost in 22nd bit of identifier (ID.7).

			5'b11000 -- Arbitration lost in 23rd bit of identifier (ID.6) 5'b11001 -- Arbitration lost in 24th bit of identifier (ID.5) 5'b11010 -- Arbitration lost in 25th bit of identifier (ID.4) 5'b11011 -- Arbitration lost in 26th bit of identifier (ID.3) 5'b11100 -- Arbitration lost in 27th bit of identifier (ID.2) 5'b11101 -- Arbitration lost in 28th bit of identifier (ID.1) 5'b11110 -- Arbitration lost in 29th bit of identifier (ID.0) 5'b11111 -- Arbitration lost in RTR bit
31:5	-	27'h0	Reserved

- **Error Code Capture Register(CANBUSn_ECC) – 0x0030**

Bit	Name	Default	Description
4:0 (R)	SEG_CODE	5'h0	5'b00011 -- Start of frame 5'b00010 -- ID.28 to ID.21 5'b00110 -- ID.20 to ID.18 5'b00100 -- SRTR bit 5'b00101 -- IDE bit 5'b00111 -- ID.17 to ID.13 5'b01111 -- ID.12 to ID.5 5'b01110 -- ID.4 to ID.0 5'b01100 -- RTR bit 5'b01101 -- Reserved bit 1 5'b01001 -- Reserved bit 0 5'b01011 -- Data Length Code 5'b01010 -- Data Field 5'b01000 -- CRC sequence 5'b11000 -- CRC delimiter 5'b11001 -- Acknowledge 5'b11011 -- Acknowledge delimiter 5'b11010 -- End of frame 5'b10010 -- Intermission 5'b10001 -- Active error flag 5'b10110 -- Passive error flag 5'b10011 -- Tolerate dominant bits 5'b10111 -- Error delimiter 5'b11100 -- Overload flag
5 (R)	DIR	1'b0	If '1', the error occurred during reception. If '0', the error occurs during transmission.
7:6 (R)	ERR_CODE	2'h0	2'b00 -- Bit error 2'b01 -- Form error 2'b10 -- Stuff error 2'b11 -- Some other type of error
31:8	-	24'h0	Reserved

- **Error Warning Limit Register (CANBUSn_EWLR) – 0x0034**

Bit	Name	Default	Description
7:0 (R/W)	EWLR	8'h30	This register defines the number of errors after which an Error Warning Interrupt should be generated (if enabled). This register may only be written in Reset Mode. In Operating Mode it is read only. You should also note that changes made within Reset Mode are only put into effect on the return to Operating Mode.

31:8	-	24'h0	Reserved
------	---	-------	----------

- **Receive Error Counter Register (CANBUSn_RXERR) – 0x0038**

Bit	Name	Default	Description
7:0 (R/W)	RXERR	8'h0	<p>The Receive Error Counter Register records the current value of the Receive Error Counter. This counter is incremented when errors are experienced in the Receive bit stream and decremented when messages are received without error, in line with the rules given in the CAN 2.0 specification. Together with the associated Transmit Error Counter, it provides an indication of the quality of transmission being experienced on the CAN bus.</p> <p>Two levels of the counter trigger specific events.</p> <ul style="list-style-type: none"> - When the counter reaches the level set in the Error Warning Limit register, an Error Warning Interrupt is generated (if enabled) unless this has previously been triggered by the Transmit Error Counter. - When the counter goes over 127, the device is put into Error Passive state in accordance with the CAN 2.0 specification (unless previously triggered by the Transmit Error Counter) and an Active error is sent. An Error Passive Interrupt is also generated (if enabled). <p>After a hardware reset or when a Bus Off event occurs, the counter is automatically set to '0'.</p> <p>The register is read only in Operating Mode but may be written in Reset Mode. You should note, however, that writing to this register has no effect when the CANBUS controller is in Bus Off state and that any change made within Reset Mode will in any case only come into effect on return to Operating Mode.</p>
31:8	-	24'h0	Reserved

- **Transmit Error Counter Register (CANBUSn_TXERR) – 0x003C**

Bit	Name	Default	Description
7:0 (R/W)	TXERR	8'h0	<p>The Transmit Error Counter Register records the current value of the Transmit Error Counter. This counter is incremented when Transmission errors are experienced and decremented when messages are transmitted without error, in line with the rules given in the CAN 2.0 specification. Together with the associated Receive Error Counter, it provides an indication of the quality of transmission being experienced on the CAN bus.</p> <p>Three levels of the counter trigger specific events.</p> <ul style="list-style-type: none"> - When the counter reaches the level set in the Error Warning Limit register, an Error Warning Interrupt is generated (if enabled) unless this has previously been triggered by the Receive Error Counter. - When the counter goes over 127, the device is put into Error Passive state in accordance with the CAN 2.0 specification (unless previously triggered by the Receive Error Counter), an Active error is sent and an Error Passive Interrupt is generated (if enabled). - When the counter goes over 255, the device is put into

			<p>Bus Off state in accordance with the CAN 2.0 specification and is automatically put into Reset mode (except during start-up when there is only one node on the CAN bus). An Error Warning Interrupt is also generated (if enabled).</p> <p>After a hardware reset, the Transmit Error Counter is automatically set to '0'.</p> <p>After a 'Bus Off' event, the register is initialized to 127 in order to count the minimum protocol-defined time before the CANBUS controller can take part in further transmission on the CAN bus (128 occurrences of the 'Bus-Free' sequence of 11 consecutive recessive bits). Reading the Transmit Error Counter during this time will give the status of the Bus Off recovery.</p> <p>Note: If the Reset Mode is re-entered before the Bus Off recovery has been completed (TXERR > 0), Bus Off will stay active with TXERR frozen until the CANBUS controller is taken back into Operating Mode.</p> <p>The register is read only in Operating Mode but may be written in Reset Mode.</p> <p>While in Bus Off state, writing a value in the range from 0 to 254 to TXERR clears the Bus Off flag. The CANBUS controller will then wait for just one Bus Free sequence after the Reset Mode has been cleared.</p> <p>Writing 255 to TXERR in Reset Mode initiates a CPU-driven Bus Off event. No error or bus status change happens in response to the new TXERR value until the CANBUS controller is taken back into Operating Mode when a Bus Off event will be performed exactly as if it had been forced by a bus error. This means Reset Mode is entered again, the Transmit Error Counter is initialized to 127, the Receive counter is cleared and the relevant Status and Interrupt register bits are set. Clearing Reset Mode now will perform the protocol-defined Bus Off recovery sequence (waiting for 128 occurrences of the Bus Free signal).</p>
31:8	-	24'h0	Reserved

- **CANBUS Transmit Buffer Register (Write: CANBUSn_TBUF) – 0x0040~0x0070 (Read: CANBUSn_TBUF_R) – 0x00180~0x01B0**

The Transmit Buffer has a length of 13 bytes. It accommodates one Transmit message of up to eight data bytes. Write-only access to the Transmit Buffer is provided in Operating Mode using RISC address 0x40~70.

The global layout of the Transmit Buffer is shown below. It is important to distinguish between Standard Frame Format (SFF) messages and the Extended Frame Format (EFF) messages.

The Transmit Buffer is subdivided into descriptor and data fields. The first byte of the descriptor field holds frame information. It describes the frame format (SFF or EFF), remote or data frame and the data length. This is then followed by either two identifier bytes for SFF or four bytes for EFF messages. The data field contains up to eight data bytes.

Table 85. Transmit Buffer Layout

Standard Frame Format (SFF)		Extended Frame Format (EFF)	
CAN Address	Field	CAN Address	Field
0x0040	TX Frame Information	0x0040	TX Frame Information
0x0044	TX Identifier 1	0x0044	TX Identifier 1
0x0048	TX Identifier 2	0x0048	TX Identifier 2
0x004C	TX Data Byte 1	0x004C	TX Identifier 3
0x0050	TX Data Byte 2	0x0050	TX Identifier 4
0x0054	TX Data Byte 3	0x0054	TX Data Byte 1
0x0058	TX Data Byte 4	0x0058	TX Data Byte 2
0x005C	TX Data Byte 5	0x005C	TX Data Byte 3
0x0060	TX Data Byte 6	0x0060	TX Data Byte 4
0x0064	TX Data Byte 7	0x0064	TX Data Byte 5
0x0068	TX Data Byte 8	0x0068	TX Data Byte 6
0x006C	(Unused)	0x006C	TX Data Byte 7
0x0070	(Unused)	0x0070	TX Data Byte 8

The bit layout of the Descriptor Field of the Transmit Buffer is shown below, first for SFF then for EFF.

Table 86. Descriptor Field of Transmit Buffer

Transmit Frame (SFF)

CAN Address	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0x0040	FF	RTR	X ₍₁₎	X ₍₁₎	DLC.3	DLC.2	DLC.1	DLC.0
0x0044	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x0048	ID.20	ID.19	ID.18	X ₍₂₎	X ₍₁₎	X ₍₁₎	X ₍₁₎	X ₍₁₎

Transmit Frame (EFF)

CAN Address	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0x0040	FF	RTR	X ₍₁₎	X ₍₁₎	DLC.3	DLC.2	DLC.1	DLC.0
0x0044	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x0048	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
0x004C	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5
0x0050	ID.4	ID.3	ID.2	ID.1	ID.0	X ₍₂₎	X ₍₁₎	X ₍₁₎

NOTE:

1. Don't care but recommend '0' to be compatible with Receive Buffer in case the Self Reception or the Self Test option is used.
2. Don't care but recommend matching the RTR bit used in the Receive Buffer in case the Self Reception or the Self Test option is used.

Frame Format (FF)

The FF bit selects the type of frame format to be transmitted. '1' selects Extended Frame Format (EFF); '0' selects Standard Frame Format (SFF).

Remote Transmission Request (RTR)

The RTR bit is used to identify the frame as either a remote frame or a data frame (as defined in the CAN protocol). '1' indicates a remote frame (i.e. a request for data from another node); '0' indicates a data frame.

Data Length Code (DLC)

The DLC [3:0] bits are used to specify the number of data bytes included in message being sent.

The maximum number of data bytes that can be included in a frame is eight so values of DLC [3:0] greater than eight are automatically interpreted as eight.

You should also note that, although no data bytes are transmitted from the local host in the case of a remote frame transmission, the data length of the remote frame should still be specified to avoid bus errors if two CAN controllers start a remote frame transmission with the same Identifier simultaneously.

Identifier (ID)

The identifier acts as the message's name, used in a receiver for acceptance filtering, and also determines the bus access priority.

The lower the binary value of the identifier the higher the priority.

In Standard Frame Format (SFF) the identifier consists of 11 bits (ID.28 to ID.18). In Extended Frame Format (EFF) messages the identifier consists of 29 bits (ID.28 to ID.0). ID.28 is the most significant bit and is transmitted first on the bus.

Data Field

The data field should comprise the number of data bytes defined by the data length code. The most significant bit of data byte 1 at CAN address 19 (SFF) or CAN address 21 (EFF) is transmitted first.

- **CANBUS Receive Window Register (CANBUSn_RWIN) – 0x0040~0x0070**

The Receive Buffer provides the window through which the CPU accesses the Receive FIFO. Like the Transmit Buffer, the Receive Buffer has a length of 13 bytes (enough to accommodate one Receive message of up to eight data bytes).

Read-only access to the Receive Buffer is provided in Operating Mode using RISC addresses 40x0~70. The layout of the Receive Buffer is similar to the Transmit Buffer described in the previous section. Indeed, the configuration used was chosen specifically to be compatible with the layout of the Transmit Buffer. Again, it is important to distinguish between Standard Frame Format (SFF) messages and the Extended Frame Format (EFF) messages.

The Receive Buffer is subdivided into descriptor and data fields. The first byte of the descriptor field holds frame information. It describes the frame format (SFF or EFF), specifies remote or data frame and gives the data length. This is then followed by either two identifier bytes for SFF or four bytes for EFF messages. The data field contains up to eight data bytes.

Table 87. Receive Buffer Layout

Standard Frame Format (SFF)		Extended Frame Format (EFF)	
CAN Address	Field	CAN Address	Field
0x0040	RX Frame Information	0x0040	RX Frame Information
0x0044	RX Identifier 1	0x0044	RX Identifier 1
0x0048	RX Identifier 2	0x0048	RX Identifier 2
0x004C	RX Data Byte 1	0x004C	RX Identifier 3
0x0050	RX Data Byte 2	0x0050	RX Identifier 4
0x0054	RX Data Byte 3	0x0054	RX Data Byte 1
0x0058	RX Data Byte 4	0x0058	RX Data Byte 2
0x005C	RX Data Byte 5	0x005C	RX Data Byte 3
0x0060	RX Data Byte 6	0x0060	RX Data Byte 4
0x0064	RX Data Byte 7	0x0064	RX Data Byte 5
0x0068	RX Data Byte 8	0x0068	RX Data Byte 6
0x006C	(Unused)	0x006C	RX Data Byte 7
0x0070	(Unused)	0x0070	RX Data Byte 8

The bit layout of the Descriptor Field of the Receive Buffer is shown below, first for SFF then for EFF.

Table 88. Descriptor Field of Receive Buffer

Receive Frame (SFF)

CAN Address	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0x0040	FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0
0x0044	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x0048	ID.20	ID.19	ID.18	RTR	0	0	0	0

Receive Frame (EFF)

CAN Address	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
0x0040	FF	RTR	0	0	DLC.3	DLC.2	DLC.1	DLC.0
0x0044	ID.28	ID.27	ID.26	ID.25	ID.24	ID.23	ID.22	ID.21
0x0048	ID.20	ID.19	ID.18	ID.17	ID.16	ID.15	ID.14	ID.13
0x004C	ID.12	ID.11	ID.10	ID.9	ID.8	ID.7	ID.6	ID.5
0x0050	ID.4	ID.3	ID.2	ID.1	ID.0	RTR	0	0

NOTE: The received data length code in the frame information byte represents the length of the data sent, which may be greater than eight bytes. However, the maximum number of data bytes received will be eight.

- **Acceptance Code Registers (CANBUSn_ACR0- CANBUSn_ACR3) – 0x0040-0x004C**
- **Acceptance Mask Registers (CANBUSn_AMR0- CANBUSn_AMR3) – 0x0050-0x005C**

The CANBUS controller filters the incoming data stream, discarding any message that does not have the required bit pattern in its identifier.

The bit pattern against which the message identifier is matched is recorded in the Acceptance Code Registers *CANBUSn_ACR0~3*, masked by the values recorded in the Acceptance Mask Registers *CANBUSn_AMR0~3*. '0's in *CANBUSn_AMR0~3* identify the bits at the corresponding positions in *CANBUSn_ACR0~3* which must be matched in the message identifier, '1's identify the corresponding bits as 'don't care'.

The bit patterns recorded as *CANBUSn_ACR0~3* can either be used as a single 4-byte filter or two shorter filters. The selection is made through the AFM bit of the Mode register (bit 3). If AFM = '1', a single filter will be applied; if AFM = '0', two filters are used, the incoming message will be accepted if its identifier matches either filter.

The way in which the bit patterns defined by *CANBUSn_ACR0~3* are applied further depend on whether the incoming message is in Standard Frame Format (SFF) or Extended Frame Format (EFF) as follows:

Table 89. Filter Bit Patterns

Standard Frame Format, Single Filter

0x0044	0x0048		0x004C	0x0050
ID.28 ~ ID.21	ID.20 ~ ID.18	RTR	XXXX(not matched)	Data Byte 1 Data Byte 2
Filter				
ACR0[7:0]	ACR1[7:4]	(ACR0[3:0] unused)	ACR2[7:0]	ACR3[7:0]
AMR0[7:0]	AMR1[7:4]	(AMR0[3:0] unused)	AMR2[7:0]	AMR3[7:0]

NOTE: If matching of the data bytes is not required, AMR2 and AMR3 should be set to FFh.

Standard Frame Format, Dual Filter

0x0044	0x0048			0x004C		0x0050
ID.28 ~ ID.21	ID.20 ~ ID.18	RTR	XXXX(not matched)	Data Byte 1 [7:4]	Data Byte 1 [3:0]	Data Byte 2 (not matched)
Filter 1						
ACR0[7:0]	ACR1[7:4]			ACR1[3:0]	ACR3[3:0]	
AMR0[7:0]	AMR1[7:4]			AMR1[3:0]	AMR3[3:0]	
Filter 2						
ACR2[7:0]	ACR3[7:4]					
AMR2[7:0]	AMR3[7:4]					

Extended Frame Format, Single Filter

0x0044	0x0048	0x004C	0x0050		
ID.28 ~ ID.21	ID.20 ~ ID.13	ID.12 ~ ID.5	ID.4 ~ ID.0	RTR	XX(not matched)
Filter					
ACR0[7:0]	ACR1[7:0]	ACR2[7:0]	ACR3[7:2]		(ACR3[1:0] unused)
AMR0[7:0]	AMR1[7:0]	AMR2[7:0]	AMR3[7:2]		(AMR3[1:0] unused)

Extended Frame Format, Dual Filter

0x0044	0x0048	0x004C	0x0050		
ID.28 ~ ID.21	ID.20 ~ ID.13	ID.12 ~ ID.5(not matched)	ID.4 ~ ID.0(not matched)	RTR(not matched)	XX(not matched)
Filter 1					
ACR0[7:0]	ACR1[7:0]				
AMR0[7:0]	AMR1[7:0]				
Filter 2					
ACR2[7:0]	ACR3[7:0]				
AMR2[7:0]	AMR3[7:0]				

- **CANBUS Receive Message Counter (CANBUSn_RMC) – 0x0074**

Bit	Name	Default	Description
4:0 (R)	RMC	5'h0	The Receive Message Counter register records the number of messages currently available in the Receive FIFO. It is automatically incremented by each Receive event and decremented by each Release Receive Buffer command. It is available for Read only access in both Operating Mode and Reset Mode.
31:5	-	27'h0	Reserved

- **CANBUS Receive Buffer Starter Address (CANBUSn_RBSA) – 0x0078**

Bit	Name	Default	Description
5:0 (R/W)	RBSA	6'h0	The Receive Buffer Start Address register records the current location of the RX FIFO Read Pointer within the 64-byte Receive FIFO as a value between 0 and 63. Location 0 maps to CAN address 0x80.

			Location 63 maps to CAN address 0x17C. This register is reset to 00h by a hardware reset but is left unchanged by a software reset (which also does not change the FIFO contents). However, a software reset sets the RX FIFO Write Pointer to the value of the RX FIFO Read Pointer, so the data accessed by the Receive Buffer following a software reset will be overwritten by the next message to be recorded in the Receive FIFO.
31:6	-	26'h0	Reserved

8.9.3.2 DMA Control Registers

Table 90. CAN Bus DMA Register Mapping

RISC Address <11:0>	Register	Description
0xF00	CANBUS_DMA_IO_CTRL	DMA mode
0xF04	CANBUS_DMA_IO_LEN	DMA Length
0xF0C	CANBUS_FIFO_LEVEL_CHK	DMA Request control
0xF10	CANBUS_FIFO_OP	DMA Initial control
0xF14	CANBUS_FIFO_STATUS	DMA Status
0xF20	CANBUS_DMA_CHANNEL	DMA Channel Selection
0xF24	CANBUS_DMA_MSGCNT	Transfer Message Counter
0xF28	CANBUS_DMA_INT	Interrupt Control
0xF2C	CANBUS_DMA_TIMEOUT	Timeout Counter
Others	-	Reserved

- CANBUS DMA I/O Mode Register (CANBUS_DMA_IO_CTRL) - 0xF00**

Bit	Name	Default	Description
0 (R/W)	IO_DMA_SEL	1'b0	0 for DMA mode, 1 for I/O mode.
1 (R/W)	IO_DMA_RW	1'b0	0: write to peripheral 1: read from peripheral
2 (R/W)	DMA_FLUSH	1'b0	Flush the DMA receive FIFO in case the data_length set at the peripheral side doesn't match the dword size set in the DMA control. Leave it as reserved if the flush isn't needed.
31:3	-	-	Reserved

NOTE: Bit 0 of *CANBUS_DMA_IO_CTRL* in *CANBUS0* controls both the receiving and sending channel. So it must be set to DMA mode no matter which direction of the DMA transfer.

- CANBUS DMA IO Length Register (CANBUS_DMA_IO_LEN) - 0xF04**

The smallest data size is DWORD, i.e. bit1 and bit 0 of this register will be ignored.

Bit	Name	Default	Description
31:0 (R/W)	DATA_LEN	32'h0	The byte length of a DMA or I/O transfer. If set to zero, the I/O or DMA transfer works continuously until it is stopped.

- CANBUS DMA FIFO Level Check Register (CANBUS_FIFO_LEVEL_CHK) - 0xF0C**

Bit	Name	Default	Description
3:0 (R/W)	FIFO_SC	4'h0	FIFO stop check in dword length
9:4	-	-	Reserved.
13:10 (R/W)	FIFO_LC	4'h0	FIFO low check in dword length
19:14	-	-	Reserved.
23:20 (R/W)	FIFO_HC	4'h0	FIFO high check in dword length
31:30	-	-	Reserved.

- **CANBUS FIFO Operation Register (CANBUS_FIFO_OP) - 0xF10**

Bit	Name	Default	Description
0 (R/W)	FIFO_START	1'b0	Start the fifo transfer when this bit is declared.
1 (R/W)	FIFO_RESET	1'b0	Internally link to fifo_start_ini. Set to 1 to stop the fifo and reset the fifo internal status, including the relevant interrupt status. Set to 0 in normal operation.
31:2	-	-	Reserved.

- **CANBUS FIFO Status Register (CANBUS_FIFO_STATUS) - 0xF14**

Bit	Name	Default	Description
5:0 (R/W)	FIFO_LEVEL	6'h0	In case FIFO is full, the value of this register is 0, thus user must concatenate FIFO_FULL bit with this value to determine the actual data count in the FIFO.
5 (R/W)	FIFO_FULL	1'b0	Fifo full status, the fifo is full when read out as 1. This bit is concatenated with FIFO_LEVEL to be the actual FIFO data count.
6 (R/W)	FIFO_EMPTY	1'b0	Fifo empty status, equivalent to (FIFO_FULL, FIFO_LEVEL) == 0
31:7	-	-	Reserved.

- **CANBUS DMA Channel Register (CANBUS_DMA_CHANNEL) - 0xF20**

Bit	Name	Default	Description
0 (R/W)	CH_SEL	1'b0	0: CANBUS0 selected 1: CANBUS1 selected
31:	-	31'h0	Reserved

NOTE: This register only exists in CANBUS0. And the bit0 only take effect when DMA mode enabled.

- **CANBUS DMA Message Counter Register (CANBUS_DMA_MSGCNT) - 0xF24**

Bit	Name	Default	Description
15:0 (R/W)	MSG_COUNT	16'h0	The number of messages should be transferred. 0 means unlimited.

31:16	-	16'h0	Reserved
-------	---	-------	----------

- **CANBUS DMA Interrupt Register (CANBUS_DMA_INT) - 0xF28**

The interrupt will be generated in DMA mode when one of the things below happens:

1. The number of byte specified by *CANBUS_DMA_LEN* has been all transferred.
2. The number of message specified by *CANBUS_DMA_MSGCNT* has been all transferred.
3. The internal unmasked canbus interrupt is generated.

Bit	Name	Default	Description
2:0 (R/W)	INT_MASK	3'h0	Bit0 for canbus internal int, Bit1 for message counter int, and Bit2 for timeout int 0: masked 1: unmasked
5:3 (R)	INT_STATUS	3'h0	Bit3 for canbus internal int, Bit4 for message counter int, and Bit5 for timeout int, can be masked by Interrupt Mask 0: no interrupt 1: interrupt generated
31:16 (R)	MSG_LEFT	16'h0	The number of the messages have not been transferred yet

- **CANBUS DMA Timeout Register (CANBUS_DMA_TIMEOUT) - 0xF2C**

Bit	Name	Default	Description
31:0 (R/W)	TIMEOUT	32'h0	The number of clock cycle for timeout. 0 means unlimited.

8.10 PWM

8.10.1 Overview

The PWM (Pulse Wide Modulate) generator can generate 4 independent outputs. Each output duty cycle can be adjusted by setting the corresponding wait and hold registers. All these 4 PWM outputs are multiplexed with GPIO. Please refer to the section of “Pin Sharing” and “GPIO” for more details.

8.10.2 PWM Registers

Table 91. PWM Interface Register Mapping

RISC Address <11:0>	Register	Description
0x00	PWM_OE	PWM output enable
0x04	PWM_WAIT0	PWM output 0 wait state for high pulse
0x08	PWM_HOLD0	PWM output 0 hold state for low pulse
0x0C	PWM_WAIT1	PWM output 1 wait state for high pulse
0x10	PWM_HOLD1	PWM output 1 hold state for low pulse
0x14	PWM_WAIT2	PWM output 2 wait state for high pulse
0x18	PWM_HOLD2	PWM output 2 hold state for low pulse
0x1C	PWM_WAIT3	PWM output 3 wait state for high pulse
0x20	PWM_HOLD3	PWM output 3 hold state for low pulse
Others	-	Reserved

- **PWM Output Enable Register (PWM_OE) - 0x00**

Bit	Name	Default	Description
0 (R/W)	PWM0 Output Enable	1'b0	Set 1 enable PWM0 output Set 0 disable PWM0 output
1 (R/W)	PWM1 Output Enable	1'b0	Set 1 enable PWM1 output Set 0 disable PWM1 output
2 (R/W)	PWM2 Output Enable	1'b0	Set 1 enable PWM2 output Set 0 disable PWM2 output
3 (R/W)	PWM3 Output Enable	1'b0	Set 1 enable PWM3 output Set 0 disable PWM3 output
31:4		28'h0	Reserved.

- **PWM Wait State Register (PWM_WAIT0, 1, 2, 3) - 0x04, 0xC, 0x14, 0x1C**

Bit	Name	Default	Description
31:0 (R/W)	WS_COUNT	32'h0	Number of IO clock 0: 1 clock; 1: 2clock...

- **PWM Hold Register (PWM_HOLD0, 1, 2, 3) - 0x08, 0x10, 0x18, 0x20**

Bit	Name	Default	Description
31:0 (R/W)	HD_COUNT	32'h0	Number of IO clock 0: 1 clock; 1: 2clock...

8.11 GPS Baseband

8.11.1 Overview

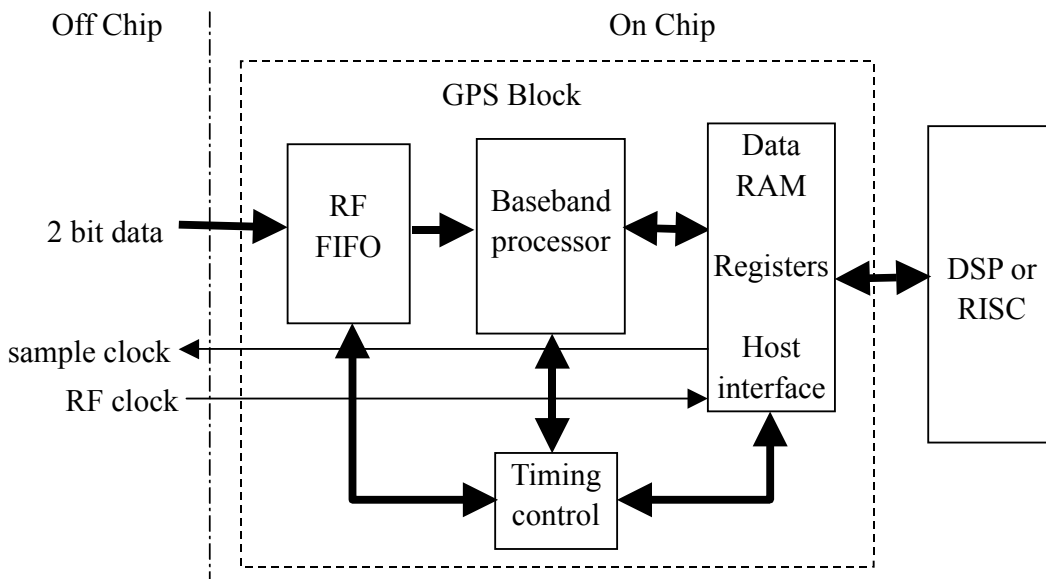


Figure 75. Diagram of GPS Baseband

The GPS baseband specific hardware with DSP and RISC can process IF signal from GPS RF to get position fix.

- Complete L1-Band, C/A, and NMEA-0183 compatibility.
- Wide Area Augmentation System (WAAS) support
- Up to 32 parallel tracking channels and 1856 correlators
- Nemerix, SiGe, NEC and Mitel RF Front-end compatible

In fig. 1, up to 32 channels share 1 baseband processor. RF FIFO store RF data, it is a 1.5Kx32bit RAM. The processor's frequency can be more than 100MHz. Processor switches to process next channel after correlating 1023 chip of current channel. During switch, the current channel's correlation results, reading address of RF FIFO, DDFS NCO value and code NCO value store into the data RAM, and then the next channel's DDFS frequency, code NCO frequency, PRN number, RF FIFO address, DDFS NCO value and code NCO value load from data RAM. After all channels being processed, interrupt is generated to DSP or RISC. DPS or RISC will process the data from baseband processor and write the new DDFS frequency and code NCO frequency back. The architecture of shared baseband processor can reduce circuit scale. If the speed of DSP (or RISC) and baseband processor is higher, more channels can be supported. Sample clock is generated on chip or can also be supplied off chip. Host interface is the interface between GPS baseband processor and DSP or RISC. The Timing control block generate timing control signal for correlator, channel switch and so on. GPS Interface block will have a 8KB dual port RAM for communication with RISC/DSP and external memory. The RAM can be accessed by either RISC or DSP, but not at the same time.

8.11.2 Pin Description

Table 92. GPS Pin Descriptions

Pin Name	Direction	Description
X_RF_DATAIN_1	Input	Sign bit of GPS baseband signals from GPS RF
X_RF_DATAIN_0	Input	Magnitude bit of GPS baseband signals from GPS RF
X_SAMPLE_CLK	Output	Sample clock to GPS RF
X_RF_CLK	Input	Clock from GPS RF

NOTE: Normally there needs to have a GPIO to handle the sleep mode control of the GPS RF module.

9 Physical Information

This chapter describes the mechanical data and package information of AT460A-BI. It is a 19x19mm 324-ball TFBGA package with 1mm pitch.

9.1 Pinout

Table 93. AT460A-BI Pinout

Finger #	Package Pin	Pin Name	Pin Type	Pin Description
1	D5	X_PXD<4>	PCOMB21X	Video Input Port Data Input (DF_AD<4>)
2	A7	X_PXD<3>	PCOMB21X	Video Input Port Data Input (DF_AD<3>)
3	B6	X_PXD<2>	PCOMB21X	Video Input Port Data Input (DF_AD<2>)
4	A6	X_PXD<1>	PCOMB21X	Video Input Port Data Input (DF_AD<1>)
5	C5	X_PXD<0>	PCOMB21X	Video Input Port Data Input (DF_AD<0>)
6	B5	X_VSYNC	PCOMB21X	Video Input Port Vertical Sync (DF_WE_B)
8	A4	X_PXCLK	PCOMB21X	Video Input Port Pixel Clock (DF_RY_BY)
7	D4	X_HSYNC	PCOMB21X	Video Input Port Horizontal Sync (DF_RE_B)
10	C4	X_FCE_B<3>	PDUW08DGZ	ROM/SRAM Chip Select (DF_CS_B<3>)
9	E4	X_FCE_B<2>	PDUW08DGZ	ROM/SRAM Chip Select (DF_CS_B<2>)
12	A5	X_FCE_B<1>	PDUW08DGZ	ROM/SRAM Chip Select (DF_CS_B<1>)
11	B4	X_FCE_B<0>	PDUW08DGZ	ROM/SRAM Chip Select (DF_CS_B<0>)
14	C3	X_PAD_RQ	PDUW08DGZ	Companion Chip Pad Request
13	B3	X_PAD_GNT	PDUW08DGZ	Companion Chip Pad Grant
16	E3	X_SDA	PDUW04DGZ	I2C Data
15	D3	X_SCL	PDUW04DGZ	I2C Clock
17	A3	X_L_PCLK	PDUW04SDGZ	LCD Pixel Clock
18	A2	X_L_LCK	PDUW04DGZ	LCD Line Clock
19	D1	X_L_FCK	PDUW04DGZ	LCD Frame Clock
20	F4	X_L_BIAS	PDUW04DGZ	LCD AC Bias
21	B2	X_LDD<0>	PDUW04DGZ	LCD Data Output
22	C2	X_LDD<1>	PDUW04DGZ	LCD Data Output
23	A1	X_LDD<2>	PDUW04DGZ	LCD Data Output
24	C6	X_LDD<3>	PDUW04DGZ	LCD Data Output
25	D2	X_LDD<4>	PDUW04DGZ	LCD Data Output
26	E5	X_LDD<5>	PDUW04DGZ	LCD Data Output
27	D6	X_LDD<6>	PDUW04DGZ	LCD Data Output
28	B1	X_LDD<7>	PDUW04DGZ	LCD Data Output

29	C1	X_GPIO<8>	PDUW04DGZ	General Purpose IO (LDD<8>)
30	E6	X_GPIO<9>	PDUW04DGZ	General Purpose IO (LDD<9>)
31	G5	X_GPIO<10>	PDUW04DGZ	General Purpose IO (LDD<10>)
32	G4	X_GPIO<11>	PDUW04DGZ	General Purpose IO (LDD<11>)
33	F6	X_GPIO<12>	PDUW04DGZ	General Purpose IO (LDD<12>)
34	F5	X_GPIO<13>	PDUW04DGZ	General Purpose IO (LDD<13>)
35	G6	X_GPIO<14>	PDUW04DGZ	General Purpose IO (LDD<14>)
36	H6	X_GPIO<15>	PDUW04DGZ	General Purpose IO (LDD<15>)
37	E2	X_RF_DATAIN<1>	PDUW04DGZ	GPS RF Data Input
38	E1	X_RF_DATAIN<0>	PDUW04DGZ	GPS RF Data Input
39	F2	X_SAMPLE_CLK	PDUW04DGZ	GPS Sample Clock
40	F1	X_RF_CLK	PDU04SDGZ	GPS RF Clock
41	H4	X_USB_XDP	PUSBF11OTG	USB Pad (Positive)
42	F3	X_USB_XDN	PUSBF11OTG	USB Pad (Negative)
43	G2	X_USB_VDDA	PUSBF11OTG	USB Analog Power (3.3V)
44	H3	X_USB_VBUS	PUSBF11OTG	USB VBUS
45	G3	X_USB_VSSA	PUSBF11OTG	USB Analog Ground
46	J3	X_USB_PDDM	PUSBF11OTG	USB DM Pull-down
47	G1	X_USB_PUDP	PUSBF11OTG	USB DP Pull-up
48	H1	X_USB_PDDP	PUSBF11OTG	USB DP Pull-down
49	H2	X_USB_VDDL	PUSBF11OTG	USB Digital Power (1.2V)
50	K2	X_USB_VSDL	PUSBF11OTG	USB Digital Ground
51	J2	X_USB_ID	PUSBF11OTG	USB ID
52	J1	X_USB_DRVVBUSC	PUSBF11OTG	USB Drive VBUS Control
53	K3	X_GPIO<23>	PDUW04DGZ	General Purpose IO (RTS0/DBGACK)
54	J4	X_GPIO<22>	PDUW04DGZ	General Purpose IO (CTS0/DBGQRQ)
55	H5	X_GPIO<21>	PDUW04DGZ	General Purpose IO (RI0/RTCK)
56	L3	X_TXD<0>	PDUW04DGZ	UART0 Transmit Data
57	L2	X_RXD<0>	PDUW04DGZ	UART0 Receive Data
58	K1	X_SCLK<1>	PDU04SDGZ	USP1 Clock (BIT_CLK)
59	K4	X_TXD<1>	PDUW04DGZ	USP1 Transmit Data (DA_DATA)
60	J5	X_RXD<1>	PDUW04DGZ	USP1 Receive Data (AD_DATA)
61	J6	X_TFS<1>	PDUW04DGZ	USP1 Transmit Frame Sync (FRAME_SYNC)
62	K5	X_RFS<1>	PDUW04DGZ	USP1 Receive Frame Sync
63	K6	X_SCAN_EN	PDDDGZ	ATPG Shift Enable

64	L1	X_SCLK<2>	PDU04SDGZ	USP2 Clock (TRACECLK)
65	M2	X_TXD<2>	PDUW04DGZ	USP2 Transmit Data (TRACESYNC)
66	L5	X_RXD<2>	PDUW04DGZ	USP2 Receive Data (PIPESTAT<2>)
67	L4	X_TFS<2>	PDUW04DGZ	USP2 Transmit Frame Sync (PIPESTAT<1>/TXD<6>)
68	M3	X_RFS<2>	PDUW04DGZ	USP2 Receive Frame Sync (PIPESTAT<0>/RXD<6>)
69	M1	X_SCLK<3>	PDU04SDGZ	USP3 Clock
70	M5	X_TXD<3>	PDUW04DGZ	USP3 Transmit Data
71	N3	X_RXD<3>	PDUW04DGZ	USP3 Receive Data
73	M4	X_TFS<3>	PDUW04DGZ	USP3 Transmit Frame Sync (TXD<7>)
72	N2	X_RFS<3>	PDUW04DGZ	USP3 Receive Frame Sync (RXD<7>)
75	L6	X_GPIO<20>	PDUW04DGZ	General Purpose IO (TCK)
74	N1	X_GPIO<19>	PDUW04DGZ	General Purpose IO (nTRST)
77	P1	X_GPIO<18>	PDUW04DGZ	General Purpose IO (TMS/DCD0)
76	N4	X_GPIO<17>	PDUW04DGZ	General Purpose IO (TDO/DTR0)
79	P3	X_GPIO<16>	PDUW04DGZ	General Purpose IO (TDI/DSR0)
78	P2	X_CAN_TXD<0>	PDUW04DGZ	CAN Bus Port 0 Transmit Data
81	M6	X_CAN_RXD<0>	PDUW04DGZ	CAN Bus Port 0 Receive Data
80	N5	X_CAN_TXD<1>	PDUW04DGZ	CAN Bus Port 1 Transmit Data
82	P4	X_CAN_RXD<1>	PDUW04DGZ	CAN Bus Port 1 Receive Data
84	T4	AVDDIO	-	PLL IO Power (3.3V)
83	R4	AHVDD	-	PLL1 Analog Power (3.3V)
86	R1	AHVDDG	-	PLL1 Analog Power (3.3V)
85	T3	AVSSIO	-	PLL1 IO Ground
88	R2	AHVSS	-	PLL1 Analog Ground
87	R3	AHVSSG	-	PLL1 Analog Ground
90	T2	DVDD1	-	PLL1 Digital Power (1.2V)
89	V1	DVDD2	-	PLL2 Digital Power (1.2V)
92	T1	DVSS1	-	PLL1 Digital Ground
91	U1	DVSS2	-	PLL2 Digital Ground
94	V2	AVDD	-	PLL2 Analog Power (1.2V)
93	U3	AVSS	-	PLL2 Analog Ground
95	V3	XVSSIO	-	PLL2 IO Ground
96	V2	X_XINW	PDXO01M	32.769 KHz Oscillator Input
97	V4	X_XOUTW	PDXO01M	32.769 KHz Oscillator Output
98	U4	XVDDIO	-	PLL2 IO Power (3.3V)

99	R5	X_VDD_FAULT	PDD04DGZ	VDD Fault
101	N6	X_BATT_FAULT	PDD04DGZ	Battery Fault
100	T6	X_PWR_EN	PDO04CDG	Power Enable
102	T5	X_RESET_B	PDUSDGZ	Reset Input
103	U5	X_XIN	PDXOE3DG	Main Oscillator Input
104	V5	X_XOUT	PDXOE3DG	Main Oscillator Output
105	U6	X_PC_CE_B<1>	PDUW08DGZ	PCMCIA Card Enable (EXT_BE<1>)
106	N7	X_PC_CE_B<0>	PDUW08DGZ	PCMCIA Card Enable (EXT_BE<0>)
107	T7	X_PC_WAIT_B	PDUW08DGZ	PCMCIA Wait (EXT_RDY_B)
108	R6	X_PC_IOIS16_B	PDUW08DGZ	PCMCIA IO IS16-bit (EXT_REQ)
109	P5	X_FA<20>	PDUW08DGZ	ROM/SRAM Address (DF_ALE/JTAG_MODE<0>)
110	U7	X_FA<21>	PDUW08DGZ	ROM/SRAM Address (DF_CLE/JTAG_MODE<1>)
111	P6	X_FA<22>	PDUW08DGZ	ROM/SRAM Address (PC_IORD_B/EXT_IOR_B/IDE_IOR_B)
112	R7	X_FA<23>	PDUW08DGZ	ROM/SRAM Address (PC_IOWE_B/EXT_IOW_B/IDE_IOW_B)
113	V6	X_FA<24>	PDUW08DGZ	ROM/SRAM Address (PC_WE_B/EXT_BURST_B)
114	P7	X_FA<25>	PDUW08DGZ	ROM/SRAM Address (PC_OE_B/EXT_ADS_B)
115	T8	X_CKO_0	PDO08CDG	Clock Output
116	V7	X_FOE_B	PDUW08DGZ	ROM/SRAM Read Enable (DF_RD_B)
117	U8	X_FWE_B	PDUW08DGZ	ROM/SRAM Write Enable (DF_WE_B)
118	P8	X_FBE<3>	PDUW08DGZ	ROM/SRAM Byte Enable
119	R8	X_FBE<2>	PDUW08DGZ	ROM/SRAM Byte Enable
120	V8	X_FBE<1>	PDUW08DGZ	ROM/SRAM Byte Enable
121	N8	X_FBE<0>	PDUW08DGZ	ROM/SRAM Byte Enable
122	U9	X_FRDY_B	PDUW08DGZ	ROM/SRAM Ready/Busy (IDE_IORDY_B)
123	T9	X_FD<0>	PDUW08DGZ	ROM/SRAM Data (PD<0>/ED<0>/DF_AD<0>/IDE_D<0>)
124	V9	X_FD<1>	PDUW08DGZ	ROM/SRAM Data (PD<1>/ED<1>/DF_AD<1>/IDE_D<1>)
125	V10	X_FD<2>	PDUW08DGZ	ROM/SRAM Data (PD<2>/ED<2>/DF_AD<2>/IDE_D<2>)
126	R9	X_FD<3>	PDUW08DGZ	ROM/SRAM Data (PD<3>/ED<3>/DF_AD<3>/IDE_D<3>)
127	R10	X_FD<4>	PDUW08DGZ	ROM/SRAM Data (PD<4>/ED<4>/DF_AD<4>/IDE_D<4>)
128	T10	X_FD<5>	PDUW08DGZ	ROM/SRAM Data (PD<5>/ED<5>/DF_AD<5>/IDE_D<5>)
129	U10	X_FD<6>	PDUW08DGZ	ROM/SRAM Data (PD<6>/ED<6>/DF_AD<6>/IDE_D<6>)
130	P9	X_FD<7>	PDUW08DGZ	ROM/SRAM Data (PD<7>/ED<7>/DF_AD<7>/IDE_D<7>)
131	R11	X_FD<8>	PDUW08DGZ	ROM/SRAM Data (PD<8>/ED<8>/DF_AD<8>/IDE_D<8>)
132	V11	X_FD<9>	PDUW08DGZ	ROM/SRAM Data (PD<9>/ED<9>/DF_AD<9>/IDE_D<9>)

133	U11	X_FD<10>	PDUW08DGZ	ROM/SRAM Data (PD<10>/ED<10>/DF_AD<10>/IDE_D<10>)
134	P10	X_FD<11>	PDUW08DGZ	ROM/SRAM Data (PD<11>/ED<11>/DF_AD<11>/IDE_D<11>)
135	T11	X_FD<12>	PDUW08DGZ	ROM/SRAM Data (PD<12>/ED<12>/DF_AD<12>/IDE_D<12>)
136	U12	X_FD<13>	PDUW08DGZ	ROM/SRAM Data (PD<13>/ED<13>/DF_AD<13>/IDE_D<13>)
137	V12	X_FD<14>	PDUW08DGZ	ROM/SRAM Data (PD<14>/ED<14>/DF_AD<14>/IDE_D<14>)
138	N9	X_FD<15>	PDUW08DGZ	ROM/SRAM Data (PD<15>/ED<15>/DF_AD<15>/IDE_D<15>)
139	V13	X_FD<16>	PDUW08DGZ	ROM/SRAM Data (ED<16>/TRACEPKT<0>)
140	U13	X_FD<17>	PDUW08DGZ	ROM/SRAM Data (ED<17>/TRACEPKT<1>)
141	V14	X_FD<18>	PDUW08DGZ	ROM/SRAM Data (ED<18>/TRACEPKT<2>)
142	U14	X_FD<19>	PDUW08DGZ	ROM/SRAM Data (ED<19>/TRACEPKT<3>)
143	V15	X_FD<20>	PDUW08DGZ	ROM/SRAM Data (ED<20>/TRACEPKT<4>)
144	U15	X_FD<21>	PDUW08DGZ	ROM/SRAM Data (ED<21>/TRACEPKT<5>)
145	V16	X_FD<22>	PDUW08DGZ	ROM/SRAM Data (ED<22>/TRACEPKT<6>)
146	N10	X_FD<23>	PDUW08DGZ	ROM/SRAM Data (ED<23>/TRACEPKT<7>)
147	U16	X_FD<24>	PDUW08DGZ	ROM/SRAM Data (ED<24>/TRACEPKT<8>)
148	V17	X_FD<25>	PDUW08DGZ	ROM/SRAM Data (ED<25>/TRACEPKT<9>)
149	U17	X_FD<26>	PDUW08DGZ	ROM/SRAM Data (ED<26>/TRACEPKT<10>)
150	P11	X_FD<27>	PDUW08DGZ	ROM/SRAM Data (ED<27>/TRACEPKT<11>)
151	V18	X_FD<28>	PDUW08DGZ	ROM/SRAM Data (ED<28>/TRACEPKT<12>)
152	U18	X_FD<29>	PDUW08DGZ	ROM/SRAM Data (ED<29>/TRACEPKT<13>)
154	T18	X_FD<30>	PDUW08DGZ	ROM/SRAM Data (ED<30>/TRACEPKT<14>)
153	N11	X_FD<31>	PDUW08DGZ	ROM/SRAM Data (ED<31>/TRACEPKT<15>)
155	T17	X_FA<0>	PDUW08DGZ	ROM/SRAM Address (PA<0>/EA<0>/IDE_A<0>)
156	P12	X_FA<1>	PDUW08DGZ	ROM/SRAM Address (PA<1>/EA<1>/IDE_A<1>)
157	T12	X_FA<2>	PDUW08DGZ	ROM/SRAM Address (PA<2>/EA<2>/IDE_A<2>)
158	R12	X_FA<3>	PDUW08DGZ	ROM/SRAM Address (PA<3>/EA<3>)
159	R18	X_FA<4>	PDUW08DGZ	ROM/SRAM Address (PA<4>/EA<4>)
160	R17	X_FA<5>	PDUW08DGZ	ROM/SRAM Address (PA<5>/EA<5>)
162	T13	X_FA<6>	PDUW08DGZ	ROM/SRAM Address (PA<6>/EA<6>)
161	N12	X_FA<7>	PDUW08DGZ	ROM/SRAM Address (PA<7>/EA<7>)
163	P18	X_FA<8>	PDUW08DGZ	ROM/SRAM Address (PA<8>/EA<8>)
164	P17	X_FA<9>	PDUW08DGZ	ROM/SRAM Address (PA<9>/EA<9>)
165	R13	X_FA<10>	PDUW08DGZ	ROM/SRAM Address (PA<10>/EA<10>)
166	T14	X_FA<11>	PDUW08DGZ	ROM/SRAM Address (PA<11>/EA<11>)
167	T15	X_FA<12>	PDUW08DGZ	ROM/SRAM Address (PA<12>/EA<12>)

168	T16	X_FA<13>	PDUW08DGZ	ROM/SRAM Address (PA<13>/EA<13>/PAD_MODE<0>)
169	R16	X_FA<14>	PDUW08DGZ	ROM/SRAM Address (PA<14>/EA<14>/PAD_MODE<1>)
170	P16	X_FA<15>	PDUW08DGZ	ROM/SRAM Address (PA<15>/EA<15>/NAND_MODE)
171	N16	X_FA<16>	PDUW08DGZ	ROM/SRAM Address (PA<16>/EA<16>/NAND_BOOT)
172	R15	X_FA<17>	PDUW08DGZ	ROM/SRAM Address (PA<17>/EA<17>/BOOT_IS16)
174	P15	X_FA<18>	PDUW08DGZ	ROM/SRAM Address (PA<18>/EA<18>/TEST_MODE<0>)
173	P13	X_FA<19>	PDUW08DGZ	ROM/SRAM Address (PA<19>/EA<19>/TEST_MODE<1>)
175	R14	X_PC_IRQ_B	PDUW08DGZ	PCMCIA Interrupt Request (EXT_IRQ_B)
176	P14	X_PC_INPACK_B	PDUW08DGZ	PCMCIA Input Acknowledge (EXT_CLK)
178	N15	X_SCAN_TEST	PDDDZ	ATPG Mode
177	N13	X_PC_VS<2>	PDUW08DGZ	PCMCIA Voltage Sense (EXT_BE<2>)
179	N14	X_CKO_1	PDO08CDG	Clock Output
180	M13	X_PC_VS<1>	PDUW08DGZ	PCMCIA Voltage Sense (EXT_BE<3>)
181	N17	X_PC_CD_B<1>	PDUW08DGZ	PCMCIA Card Detect
182	N18	X_PC_CD_B<0>	PDUW08DGZ	PCMCIA Card Detect
183	M14	X_PC_STSCHG	PDUW08DGZ	PCMCIA Status Change
184	M15	X_PC_SPKR_B	PDUW08DGZ	PCMCIA Speaker (EXT_GNT_B)
185	L13	X_PC_RESET	PDUW08DGZ	PCMCIA Reset (EXT_RST_B)
186	M16	X_PC_REG_B	PDUW08DGZ	PCMCIA Register Enable (EXT_SEL_B)
187	M17	X_SCLK<4>	PDU04SDGZ	USP4 Clock (SD_DAT<3>/PA<23>)
188	M18	X_TXD<4>	PDUW04DGZ	USP4 Transmit Data
189	L14	X_RXD<4>	PDUW04DGZ	USP4 Receive Data
190	K13	X_TFS<4>	PDUW04DGZ	USP4 Transmit Frame Sync (SD_DAT<2>/PA<22>)
191	L15	X_RFS<4>	PDUW04DGZ	USP4 Receive Frame Sync (SD_DAT<1>/PA<21>)
192	L18	X_SCLK<5>	PDU04SDGZ	USP5 Clock (SD_CLK/PA<25>)
193	L17	X_TXD<5>	PDUW04DGZ	USP5 Transmit Data
194	L16	X_RXD<5>	PDUW04DGZ	USP5 Receive Data
195	K14	X_TFS<5>	PDUW04DGZ	USP5 Transmit Frame Sync (SD_CMD/PA<24>)
196	K18	X_RFS<5>	PDUW04DGZ	USP5 Receive Frame Sync (SD_DAT<0>/PA<20>)
197	J13	X_DF_RY_BY	PDUW08DGZ	NAND Flash Read/Busy (NAND_SEL)
198	K15	X_IDE_CS_B<1>	PDUW08DGZ	IDE Chip Select
199	K16	X_IDE_CS_B<0>	PDUW08DGZ	IDE Chip Select
200	J14	X_IDE_DREQ	PDUW08DGZ	IDE Data Request
201	J18	X_IDE_DACK	PDUW08DGZ	IDE Data Acknowledge

202	K17	X_IDE_IRQ	PDUW08DGZ	IDE Interrupt Request
203	H13	X_GPIO<7>	PDUW04DGZ	General Purpose IO (PWM<3>)
204	H18	X_GPIO<6>	PDUW04DGZ	General Purpose IO (PWM<2>)
205	J15	X_GPIO<5>	PDUW04DGZ	General Purpose IO (PWM<1>)
206	J17	X_GPIO<4>	PDUW04DGZ	General Purpose IO (PWM<0>)
207	H14	X_GPIO<3>	PDUW04DGZ	General Purpose IO (SDCard Detect)
208	G13	X_GPIO<2>	PDUW04DGZ	General Purpose IO
209	J16	X_GPIO<1>	PDUW04DGZ	General Purpose IO
210	H17	X_GPIO<0>	PDUW04DGZ	General Purpose IO
211	H15	X_GPIO<24>	PDUW04DGZ	General Purpose IO (LDD<16>)
212	H16	X_GPIO<25>	PDUW04DGZ	General Purpose IO (LDD<17>)
213	G18	X_GPIO<26>	PCOMB21X	General Purpose IO (DF_AD<15>)
214	F18	X_GPIO<27>	PCOMB21X	General Purpose IO (DF_AD<14>)
215	G14	X_MCKE	PCOMB21X	Memory Clock Enable
216	G17	X_MCS_B<1>	PCOMB21X	Memory Chip Select
217	E18	X_MCS_B<0>	PCOMB21X	Memory Chip Select
218	D18	X_MDQM<3>	PCOMB21X	Memory Data Mask
219	F16	X_MDQM<2>	PCOMB21X	Memory Data Mask
220	G15	X_MDQM<1>	PCOMB21X	Memory Data Mask
221	G16	X_MDQM<0>	PCOMB21X	Memory Data Mask
222	F14	X_MA<12>	PCOMB21X	Memory Address
223	F17	X_MA<11>	PCOMB21X	Memory Address
224	C18	X_MA<10>	PCOMB21X	Memory Address
225	D17	X_MA<9>	PCOMB21X	Memory Address
226	E17	X_MA<8>	PCOMB21X	Memory Address
227	B18	X_MA<7>	PCOMB21X	Memory Address
228	F15	X_MA<6>	PCOMB21X	Memory Address
229	C17	X_MA<5>	PCOMB21X	Memory Address
230	E16	X_MA<4>	PCOMB21X	Memory Address
231	A18	X_MA<3>	PCOMB21X	Memory Address
232	B17	X_MA<2>	PCOMB21X	Memory Address
233	D16	X_MA<1>	PCOMB21X	Memory Address
234	A17	X_MA<0>	PCOMB21X	Memory Address
235	C16	X_M_BA<1>	PCOMB21X	Memory Bank Address
236	B16	X_M_BA<0>	PCOMB21X	Memory Bank Address

237	A16	X_MWE_B	PCOMB21X	Memory Write Enable
238	E15	X_MRAS_B	PCOMB21X	Memory Row Address Strobe
239	D15	X_MCAS_B	PCOMB21X	Memory Column Address Strobe
240	C15	X_MCLK_O	PDIFF21X	Memory Clock Output (+)
242	A15	X_MCLKB_O	PDIFF21X	Memory Clock Output (-)
241	B15	X_MDQS<3>	PCOMB21X	Memory Data Strobe
243	D14	X_MDQS<2>	PCOMB21X	Memory Data Strobe
244	E14	X_MDQS<1>	PCOMB21X	Memory Data Strobe
245	C14	X_MDQS<0>	PCOMB21X	Memory Data Strobe
246	D13	X_MD<31>	PCOMB21X	Memory Data
247	B14	X_MD<30>	PCOMB21X	Memory Data
248	F13	X_MD<29>	PCOMB21X	Memory Data
249	C13	X_MD<28>	PCOMB21X	Memory Data
250	E13	X_MD<27>	PCOMB21X	Memory Data
251	A14	X_MD<26>	PCOMB21X	Memory Data
252	B13	X_MD<25>	PCOMB21X	Memory Data
253	D12	X_MD<24>	PCOMB21X	Memory Data
254	E12	X_MD<23>	PCOMB21X	Memory Data
255	A13	X_MD<22>	PCOMB21X	Memory Data
256	C12	X_MD<21>	PCOMB21X	Memory Data
257	F12	X_MD<20>	PCOMB21X	Memory Data
258	B12	X_MD<19>	PCOMB21X	Memory Data
259	D11	X_MD<18>	PCOMB21X	Memory Data
260	A12	X_MD<17>	PCOMB21X	Memory Data
261	C11	X_MD<16>	PCOMB21X	Memory Data
262	B11	X_MD<15>	PCOMB21X	Memory Data
263	E11	X_MD<14>	PCOMB21X	Memory Data
264	F11	X_MD<13>	PCOMB21X	Memory Data
265	D10	X_MD<12>	PCOMB21X	Memory Data
266	A11	X_MD<11>	PCOMB21X	Memory Data
267	C10	X_MD<10>	PCOMB21X	Memory Data
268	B10	X_MD<9>	PCOMB21X	Memory Data
269	E10	X_MD<8>	PCOMB21X	Memory Data
270	A10	X_MD<7>	PCOMB21X	Memory Data
271	A9	X_MD<6>	PCOMB21X	Memory Data

272	B9	X_MD<5>	PCOMB21X	Memory Data
273	F10	X_MD<4>	PCOMB21X	Memory Data
274	C9	X_MD<3>	PCOMB21X	Memory Data
275	D9	X_MD<2>	PCOMB21X	Memory Data
276	E9	X_MD<1>	PCOMB21X	Memory Data
277	F9	X_MD<0>	PCOMB21X	Memory Data
278	C8	X_PXD<15>	PCOMB21X	Video Input Port Data (DF_AD<13>)
279	A8	X_PXD<14>	PCOMB21X	Video Input Port Data (DF_AD<12>)
280	E8	X_PXD<13>	PCOMB21X	Video Input Port Data (DF_AD<11>)
281	B8	X_PXD<12>	PCOMB21X	Video Input Port Data (DF_AD<10>)
282	F8	X_PXD<11>	PCOMB21X	Video Input Port Data (DF_AD<9>)
283	D8	X_PXD<10>	PCOMB21X	Video Input Port Data (DF_AD<8>)
284	B7	X_PXD<9>	PCOMB21X	Video Input Port Data (DF_CLE)
285	C7	X_PXD<8>	PCOMB21X	Video Input Port Data (DF_ALE)
286	E7	X_PXD<7>	PCOMB21X	Video Input Port Data (DF_AD<7>)
287	D7	X_PXD<6>	PCOMB21X	Video Input Port Data (DF_AD<6>)
288	F7	X_PXD<5>	PCOMB21X	Video Input Port Data (DF_AD<5>)
G7 , H7 , J7 , K7 , L7		VDDIO	-	
M7 , M8 , M9 , M10 , M11 , M12		VDDPDN	-	
J12 , K12 , L12		VDDPRE	-	
G9 , G10 , G11 , G12 , H11 , H12		VDHPSSTL	-	
G8		VREFSSTL	-	
K9 , K10 , L10 , L11		VSS	-	
H8 , J8 , K8 , L8 , L9		VSSIO	-	
H9 , H10 , J9 , J10		VSSPSSTL	-	
J11 , K11		VSSRSSTL	-	

NOTE:

Pin Type	Pin Type Description
PCOMB21X	3-State Output 1.8/2.5/3.3V LVTTTL and SSTL2 Buffer Pad
PDIFF21X	3-State Differential Output 1.8/2.5/3.3V LVTTTL and SSTL2 Buffer Pad
PDDDGZ	Input Pad With Pulldown, 5V-Tolerant
PDO04CDG	CMOS Output Pad. 4mA Driving Strength
PDO08CDG	CMOS Output Pad. 8mA Driving Strength
PDUSDGZ	Shmit triggered input with pull-up. 5V-Tolerant

PDD04DGZ	CMOS 3-State Output Pad with Input and Pulldown, 5V-Tolerant, 4mA Driving Strength
PDU04SDGZ	CMOS 3-State Output Pad with Schmitt Trigger Input and Pull-up, 5V-Tolerant, 4mA Driving Strength
PDUW04DGZ	3-State Output Pad with Input and Enable Controlled Pull-Up, 5V-Tolerant, 4mA Driving Strength
PDUW08DGZ	3-State Output Pad with Input and Enable Controlled Pull-Up, 5V-Tolerant, 8mA Driving Strength
PDXOE3DG	High-frequency Oscillator Pad
PDXO01M	Low-frequency Oscillator Pad
PUSBF11OTG	Full-speed USB OTG Transceiver Pad

9.2 Package

TOP VIEW

BOTTOM VIEW

SECTION C-C

DETAIL : A

DETAIL : B

Symbol	Dimension in mm			Dimension in inch		
	MIN	NOM	MAX	MIN	NOM	MAX
A	---	---	1.50	---	---	0.059
A1	0.40	0.50	0.60	0.016	0.020	0.024
A2	0.84	0.89	0.94	0.033	0.035	0.037
c	0.32	0.35	0.40	0.013	0.014	0.016
D	18.80	19.00	19.20	0.740	0.748	0.758
E	18.80	19.00	19.20	0.740	0.748	0.758
D1	---	17.00	---	---	0.669	---
E1	---	17.00	---	---	0.669	---
e	---	1.00	---	---	0.039	---
b	0.50	0.60	0.70	0.020	0.024	0.028
dgg	---	0.10	---	---	0.004	---
bbd	---	0.10	---	---	0.004	---
ddd	---	0.15	---	---	0.005	---
eee	---	0.15	---	---	0.005	---
fff	---	0.08	---	---	0.003	---
MD/NE	---	18/18	---	---	18/18	---

NOTE :

1. CONTROLLING DIMENSION : MILLIMETER.
2. PRIMARY DATUM C AND SEATING PLANE ARE DEFINED BY THE SPHERICAL CROWNS OF THE SOLDER BALLS.
3. DIMENSION b IS MEASURED AT THE MAXIMUM SOLDER BALL DIAMETER, PARALLEL TO PRIMARY DATUM C.
4. THERE SHALL BE A MINIMUM CLEARANCE OF 0.28mm BETWEEN THE EDGE OF THE SOLDER BALL AND THE BODY EDGE.
5. REFERENCE DOCUMENT : JEDEC MO-205.

TITLE : 324LD TF BGA (19X19mm) PACKAGE OUTLINE

SUBSTRATE MATERIAL : BT RESIN

APPR.	C.S.Hsiao	DWG NO.	TF324-SW1
ENG.	C.P.Wang	REV NO.	B
Q.M	River Ku	PRODUCT CODE	TF32411
CHK.	G.M.Huang K.C.Hung	DATE	02/24/'00
DWG.	S.F.CHEN	SHT No.	1/1

COPY CONTROLLED

REV NO	DESCRIPTION	DATE
B	NOTE "5" JEDEC MO-205 INSTEAD OF MO-192.	02/24/'00

9.3 DC Electrical Characteristics

9.3.1 Absolute Maximum Ratings

The tables in this section describe the AtlasII DC Electrical characteristics. [Table 94](#) gives the absolute maximum ratings.

Table 94. Absolute Maximum Ratings¹

Characteristic	Symbol	Min	Max	Unit
Supply voltage-I/O buffers	VDDIO	-0.3	4	V
Supply voltage-I/O buffers	XVDDIO	-0.3	4	V
Supply voltage-mem bus (DDR SDRAM)	VDD_MEM (VSSTL)	-0.3	3	V
Supply voltage-mem bus (SDRAM)	VDD_MEM (SDRAM)	-0.3	4	V
Supply voltage-PLL	VDD_PHA	-0.3	4	V
Supply voltage-PLL	VDD_PLA	-0.3	1.44	V
Supply voltage-PLL	VDD_PLD	-0.3	1.44	V
Supply voltage-core	VTT	-0.3	1.44	V
Supply voltage-core	VREF	-0.3	1.44	V
Supply voltage-core	VDDPRE	-0.3	1.44	V
Supply voltage-inter model	VDDPDN	-0.3	1.44	V
Input voltage overshoot	Vinos	—	0.5	V
Input voltage undershoot	Vinus	—	0.5	V
Storage temperature range	Tstg	-40	150	

¹ Absolute maximum ratings are stress ratings only, and functional operation at the maximums is not guaranteed. Stresses beyond those listed may affect device reliability or cause permanent damage.

9.3.2 Recommended Operation Conditions

[Table 95](#) gives the recommended operating conditions.

Table 95. Recommended Operating Conditions¹

Characteristic	Symbol	Min	Typ	Max	Unit
Supply voltage-I/O buffers	VDDIO	3.13	3.3	3.5	V
Supply voltage-I/O buffers	XVDDIO	3.13	3.3	3.5	V
Supply voltage-mem bus (DDR SDRAM)	VDD_MEM (VSSTL)	2.37	2.5	2.63	V

Supply voltage-mem bus (SDRAM)	VDD_MEM (SDRAM)	3.13	3.3	3.5	V
Supply voltage-PLL	VDD_PHA	3.13	3.3	3.5	V
Supply voltage-PLL	VDD_PLA	1.14	1.2	1.26	V
Supply voltage-PLL	VDD_PLD	1.14	1.2	1.26	V
Supply voltage-core	VTT	1.14	1.2	1.26	V
Supply voltage-core	VREF	1.14	1.2	1.26	V
Supply voltage-core	VDDPRE	1.2	1.28	1.36	V
Supply voltage-inter model	VDDPDN	1.2	1.28	1.36	V
Input voltage - standard I/O	V _{in}	0	VDDIO	VDDIO	V
Input voltage - memory I/O buffers (SDR)	V _{in_sdr}	0	—	VDD_MEM (SDRAM)	V
Input voltage - memory I/O buffers (DDR)	V _{in_ddr}	0	—	VDD_MEM (VSSTL)	V
operating temperature range	Top	-20		+70	

¹ These are recommended and tested operating conditions. Proper device operation outside these conditions is not guaranteed.

9.3.3 DC Electrical Specifications

Table 96 gives the DC Electrical characteristics for the AtlasII at recommended operating conditions

Table 96. DC Electrical Specifications

Characteristic	Condition	Symbol	Min	Max	Unit
Input high voltage	Input type = CMOS	V _{IH}	1.8	—	V
Input high voltage	Input type = TTL VDD_IO/VDD_MEM_IOSDR	V _{IH}	2.0	—	V
Input high voltage	Input type = SSTL VDD_MEM_IODDR	V _{IH}	1.5	—	V
Input high voltage	Input type = SCHMITT VDD_IO	V _{IH}	2.0	—	V
Input high voltage	X_XIN, X_XINW	V _{IH}	2.0	—	V
Input low voltage	Input type = CMOS	V _{IL}	—	1.1	V
Input low voltage	Input type = TTL VDD_IO/VDD_MEM_IOSDR	V _{IL}	—	0.8	V
Input low voltage	Input type = TTL VDD_MEM_IODDR	V _{IL}	—	0.8	V
Input low voltage	Input type = SCHMITT VDD_IO	V _{IL}	—	0.8	V
Input low voltage	X_XIN, X_XINW	V _{IL}	—	0.8	V

Input leakage current	V _{in} = 0 or VDD_IO	I _{IN}	—	±10	uA
Input current, pull-up resistor	V _{in} = 0	I _{INpu}	40	110	uA
Input current, Pull-down resistor	V _{in} = VDD_IO	I _{INpd}	40	110	uA
Output high voltage	IOH is driver dependent ¹ VDD_IO=3.0, DRV4	VOH	2.4	—	V
Output high voltage	IOH is driver dependent ¹ VDD_IO=3.0, DRV8	VOH	2.4	—	V
Output high voltage	IOH is driver dependent ¹ VDD_MEM=2.3.DRV8_MEM	VOHDDR	1.8	—	V
Output low voltage	IOL is driver dependent ¹ VDD_IO=3.0, DRV4	VOL	—	0.4	V
Output low voltage	IOL is driver dependent ¹ VDD_IO=3.0, DRV8V	VOL	—	0.4	V
Output low voltage	IOL is driver dependent ¹ VDD_MEM=2.3.DRV8_MEM	VOLDDR	—	0.4	V
Capacitance		C _{in}	---	10	pF

¹ See [Table 97](#) for the typical drive capability of a specific signal pin based on the type of output driver

Table 97. Drive Capability of AtlasII Output Pins

Driver Type	Supply Voltage	IOH	IOL	Unit
DRV4	VDD_IO = 3.0V	4	4	mA
DRV8_MEM	VDD_IO = 2.3V	8	8	mA
DRV8	VDD_IO = 3.0V	8	8	mA

9.3.4 Electrostatic Discharge

— CAUTION —

This device contains circuitry that protects against damage due to high-static voltage or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages. Operational reliability is enhanced if unused inputs are tied to an appropriate logic voltage level (i.e., either GND or VCC).

[Table 98](#) gives package ESDI characteristics for this device.

Table 98. ESD Characteristics

Sym	Rating	Min	Max	Unit
VHBM	Human Body Model (HBM)—JEDEC JESD22-A114-B	2000	—	V
VCDM	Charge Device Model (CDM)—JEDEC JESD22-C101	500	—	V
ILAT	Latch-up Current at TA=70 positive negative		±100	mA

9.3.5 Power Dissipation

Power dissipation of the AtlasII is caused by 4 different components:

1. the dissipation of the core digital logic (supplied by VDDPRE),
2. the dissipation of the PLL circuitry (supplied by VDD_PHA and VDD_PLA)

3. the dissipation of the IO logic (supplied by VDD_IO_MEM and VDD_IO).
4. the dissipation of the some internal model (supplied by VDDPDN), this can be shutdown by external power management. (see every model)

Table 99 details typical measured core and PLL

power dissipation figures for a range of operating modes. However, the dissipation due to the switching of the IO pins can not be given in general, but must be calculated by the user for each application case using the following formula

$$P_{IO} = P_{IOint} + \sum_M N * C * VDD_IO * f$$

where N is the number of output pins switching in a group M, C is the capacitance per pin, VDD_IO is the IO voltage swing, f is the switching frequency and P_{IOint} is the power consumed by the unloaded IO stage. The total power consumption of the AtlasII must not exceed the value, which would cause the maximum junction temperature to be exceeded and damaged

$$P_{total} = P_{core} + P_{pll} + P_{IO} + P_{in_m}$$

Table 99. Power Dissipation

normal mode			
	CPU/DSP/SYS/IO	CPU/DSP/SYS/IO	
	200/200/200/100	300/150/150/75	
	Typ	Typ	Unit
P _{core} (VDDPRE)	26	---	mw
PLL	4	---	mw
IO/MEM (VDDIO, MEM)	250	---	mw
Internal (VDDPDN)	from 120 to 160	---	mw

Sleep mode					
	PIQ	Ppll	Pcore	Pin_m	Unit
typ	35	3	0.2	0	mw
total	About 40				mw

9.3.6 Thermal Characteristics

Table 100. Thermal Resistance Data

Rating			Value	Unit	Notes
Junction to Ambient Natural Convection	Four layer board (2s2p)	R _{θJA}	23.5	°C/W	19x19 package
Junction to Ambient Natural Convection	Four layer board (2s2p)	R _{θJA}	38.5	°C/W	14x14 package
Junction to Ambient (@1 m/s)	Four layer board (2s2p)	R _{θJA}	19.8	°C/W	19x19 package
Junction to Ambient (@1 m/s)	Four layer board (2s2p)	R _{θJA}	34.7	°C/W	14x14 package

Junction temperature is a function of die size, on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and board

thermal resistance.

9.4 AC Electrical Characteristics

AC Test Timing Conditions:

Unless otherwise noted, all test conditions are as follows:

TA = -20 to 70 °C

VDD_CORE = 1.14 to 1.26 V

VDD_IO = 3.13 to 3.5 V

Input conditions:

All Inputs: tr, tf <= TBD

Output Loading: All Outputs about: 50 pF

9.4.7 AC Operation Frequency Data

Table 101 provides the operating frequency information of the AtlasII

Table 101. Clock Frequencies

		Min	Max	Units
1	ARM_core	—	276	MHz
2	Dsp_core	—	200	MHz
3	SDRAM Clock	—	138	MHz
4	DDRAM Clock	—	100	MHz

9.4.8 Clock AC Specifications

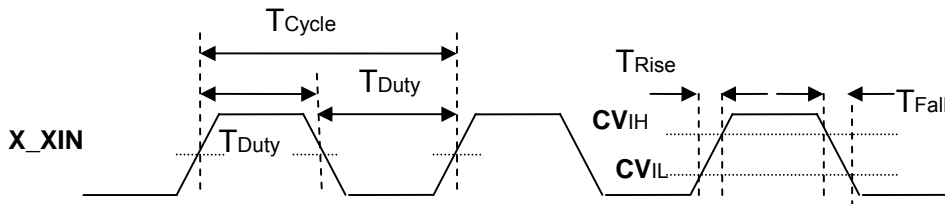


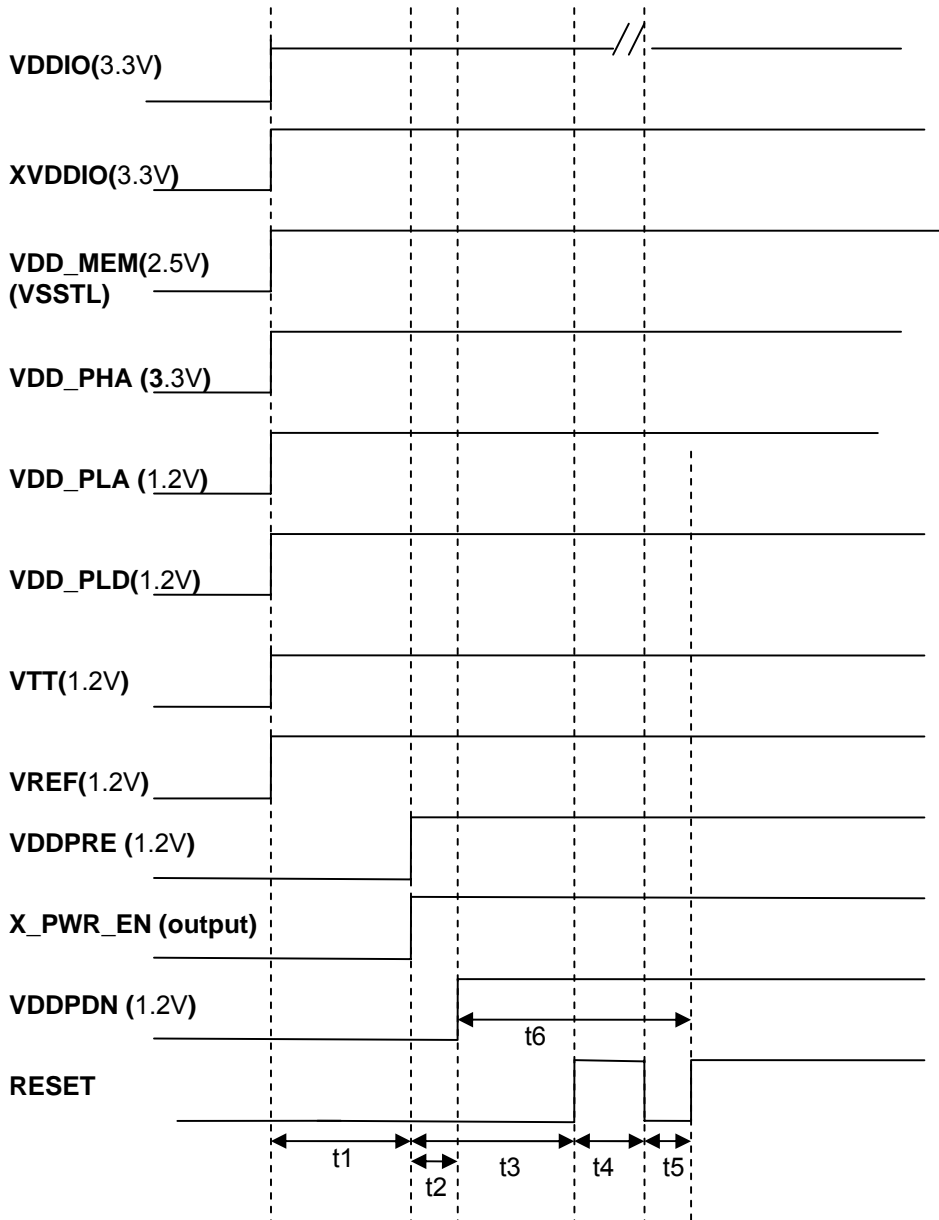
Figure 76. Timing Diagram—X_XIN

Table 102. X_XIN Timing

Sym	SpecID Description	Min	Typ	Max	Units
T _{Cycle}	X_XIN cycle time. 1	---	83.3	---	ns
T _{Rise}	X_XIN rise time.	--		5.0	ns
T _{Fall}	X_XIN fall time.	--		5.0	ns
T _{Duty}	X_XIN duty cycle	40.0		60.0	%
CV _{IH}	X_XIN input voltage high	2.0		--	V
CV _{IL}	X_XIN input voltage low	--		0.8	V

9.4.9 Resets

When power on, always power up the higher voltage domain first. So if we have 1.8, 2.5, 3.3V power supply, please follow the power on/off sequence as shown in the following figure.



t1>10ms t2>0s t3>50ms t4>3ms t5 >2ms t6>1ms
 * t3 must wait for 12MHz Clock steadily

Figure 77. Timing Diagram—power and reset Timing

9.4.10 External Interrupts

- The Atlas II provides 32 external interrupts: GPIO group 0 [31:0] .
- Each pin has two methods interrupt: edge, level.
- Due to synchronization, prioritization, and mapping of external interrupt sources, the propagation of external interrupts to the core processor is delayed by several IO_clk clock cycles. The following table specifies the interrupt latencies in IO_clk cycles. The IO_clk frequency is programmable in the Clock Distribution Module

Table 103. Minimum pulse width for external interrupts to be recognized

Name	Min Pulse Width	Max Pulse Width	Reference Clock
------	-----------------	-----------------	-----------------

All external interrupts	> 3 clock cycle		IO_clk
-------------------------	-----------------	--	--------

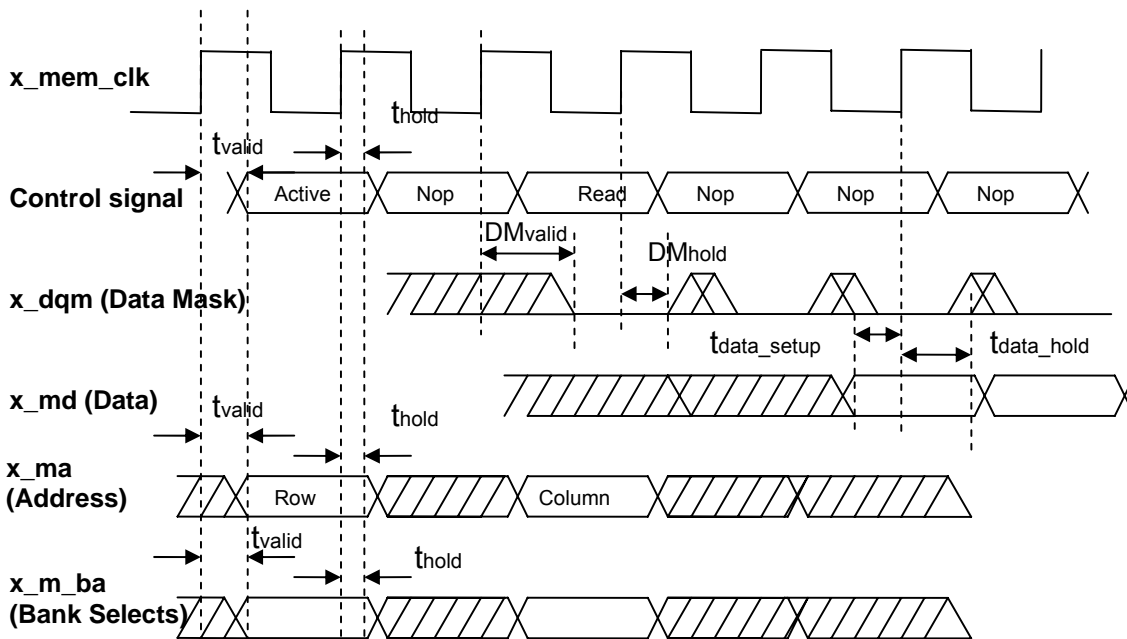
9.4.11 SDRAM (sdr32)

1. x_mem_clk, Address, Control, and Data output are clocked by sys_clk, The path delay of x_mem_clk can be adjust by Manager Delay Control Register (PWR_DELAY_CTRL_0), (only x_mem_clk, the others can't be changed)
2. Data input is clocked by mck_i. Mck_i can be driven by internal mem_clk or pad loop back(x_mem_clk) from (PWR_DELAY_CTRL_0), the path delay of mck_i can be configured by PWR_DELAY_CTRL_1, Because the internal delay of x_mem_clk & mck_i are configurable, it should be relatively loose for the timing table 116.

9.4.11.1Memory Interface Timing-Standard SDRAM Read Command

Table 104. Standard SDRAM Memory Read Timing

Sym	Description	Min	Max	Units
t _{mem_clk}	MEM_CLK period	7.5	—	ns
t _{valid}	Control Signals, Address and MBA Valid after rising edge of MEM_CLK	—	6	ns
t _{hold}	Control Signals, Address and MBA Hold after rising edge of MEM_CLK	2	—	ns
DM _{valid}	DQM valid after rising edge of MEM_CLK	—	4.5	ns
DM _{hold}	DQM hold after rising edge of MEM_CLK	3.0	—	ns
t _{datasetup}	MDQ setup to rising edge of MEM_CLK	-2.0	—	ns
t _{datahold}	MDQ hold after rising edge of MEM_CL	4	—	ns



NOTE: Control Signals are composed of RAS, CAS, MEM_WE, MEM_CS, MEM_CS1 and CLK_EN

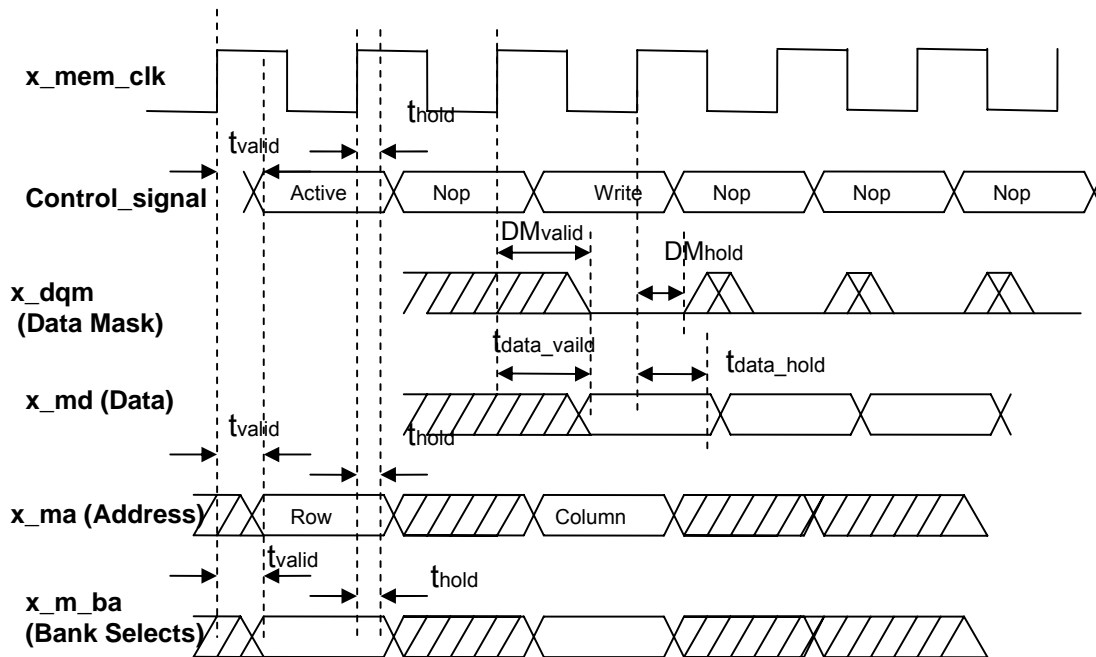
Figure 78. Timing Diagram—Standard SDRAM Memory Read Timing

9.4.11.2Memory Interface Timing-Standard SDRAM Write Command

Table 105. Standard SDRAM Write Timing

Sym	Description	Min	Max	Units
-----	-------------	-----	-----	-------

t_{mem_clk}	MEM_CLK period	7.5	—	ns
t_{valid}	Control Signals, Address and MBA Valid after rising edge of MEM_CLK	—	6	ns
T_{hold}	Control Signals, Address and MBA Hold after rising edge of MEM_CLK	2	—	ns
DM_{valid}	DQM valid after rising edge of MEM_CLK	—	4.5	ns
DM_{hold}	DQM hold after rising edge of MEM_CLK	3.0	—	ns
$t_{datavalid}$	MDQ setup to rising edge of MEM_CLK	—	7	ns
$t_{datahold}$	MDQ hold after rising edge of MEM_CL	5	—	ns



NOTE: Control Signals are composed of RAS, CAS, MEM_WE, MEM_CS, MEM_CS1 and CLK_EN

Figure 79. Timing Diagram—Standard SDRAM Memory Write Timing

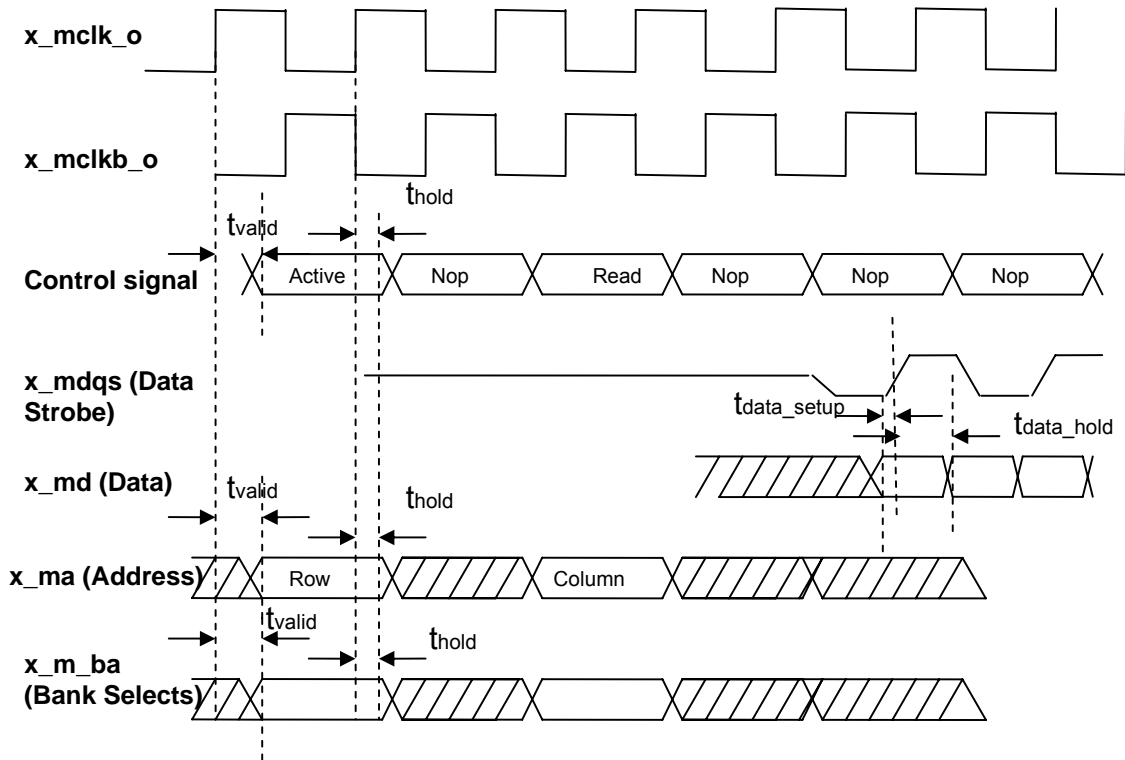
9.4.12 DDR SDRAM

1. Address and Control output are clocked by sys_clk (the same as SDRAM);
2. Data, DQS, and DM output are clocked by mck2x_o, which should match the delay between mck2x_o & mck_clk;
3. Data input is clocked by delayed DQS.

9.4.12.1 Memory Interface Timing-DDR SDRAM Read Command

Table 106. DDR SDRAM Memory Read Timing

Sym	Description	Min	Max	Units
t_{mem_clk}	MEM_CLK period	10	—	ns
t_{valid}	Control Signals, Address and MBA Valid after rising edge of MEM_CLK	—	6	ns
T_{hold}	Control Signals, Address and MBA Hold after rising edge of MEM_CLK	4	—	ns
$t_{datasetup}$	MDQ setup to rising edge of x_mdqs	4.5	—	ns
$t_{datahold}$	MDQ hold after rising edge of x_mdqs	-3.0	—	ns



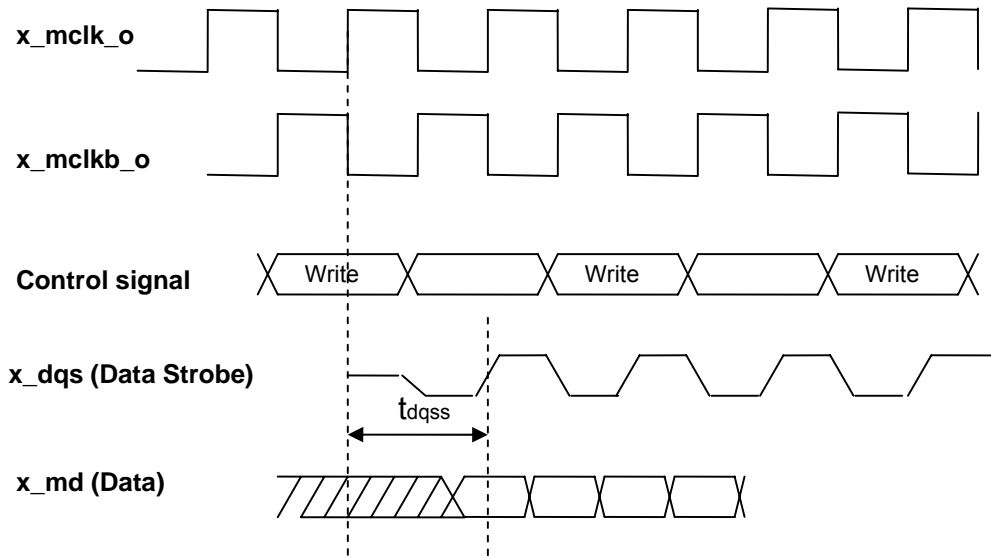
NOTE: Control Signals are composed of RAS, CAS, MEM_WE, MEM_CS, MEM_CS1 and CLK_EN

Figure 80. Timing Diagram—DDR SDRAM Memory Read Timing

9.4.12.2 Memory Interface Timing-DDR SDRAM Write Command

Table 107. DDR SDRAM Memory Write Timing

Sym	Description	Min	Max	Units
T_{mem_clk}	MEM_CLK period	10	—	ns
t_{dqss}	Delay from write command to first rising edge of MDQS	$T_{mem_lck}+0.2$	$T_{mem_lck}+1.5$	ns



NOTE: Control Signals are composed of RAS, CAS, MEM_WE, MEM_CS, MEM_CS1 and CLK_EN

Figure 81. DDR SDRAM Memory Write Timing

9.4.13 Camera Specifications

when camera work on master mode ,pixclk ,x_vsync,x_hsync are output, x_pxd(pixel_data) are input.

Table 108. Camera data Timing

Sym	Description	Min	Max	Units
t_{vaid}	Vsync output valid time	-	1.5	ns
t_{vhd}	Vsync hold time	0.2		ns
t_{hvaid}	Hsync output valid time	-	1.8	ns
t_{hd}	Hsync output hold time	0.5		ns
t_{st}	Input setup time	0.8		ns
t_{hd}	input hold time	1.0		ns

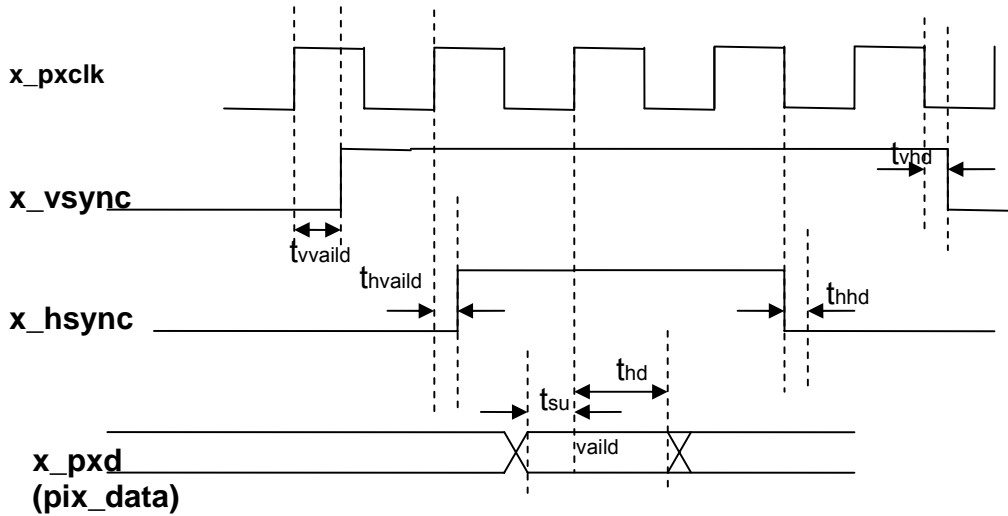


Figure 82. Camera data Timing

9.4.14 Lcd Specifications

when lcd work on master mode ,pixclk ,x_vsync,x_hsync are output

Table 109. Lcd data Timing

Sym	Description	Min	Max	Units
t _{fvaid}	Frame output valid time	-	-0.6.	ns
t _{fhd}	Frame hold time	-0.2		ns
t _{lvaid}	Line output valid time	-	-0.7	ns
t _{lhd}	Line output hold time	-0.3		ns
t _{bvaid}	Bias output valid time	-	1.4	ns
t _{bhd}	bias output hold time	0.4		ns
t _{dvaid}	Pix_data output valid time	-	5.8	ns
t _{dhd}	Pix_data output hold time	0.6		ns

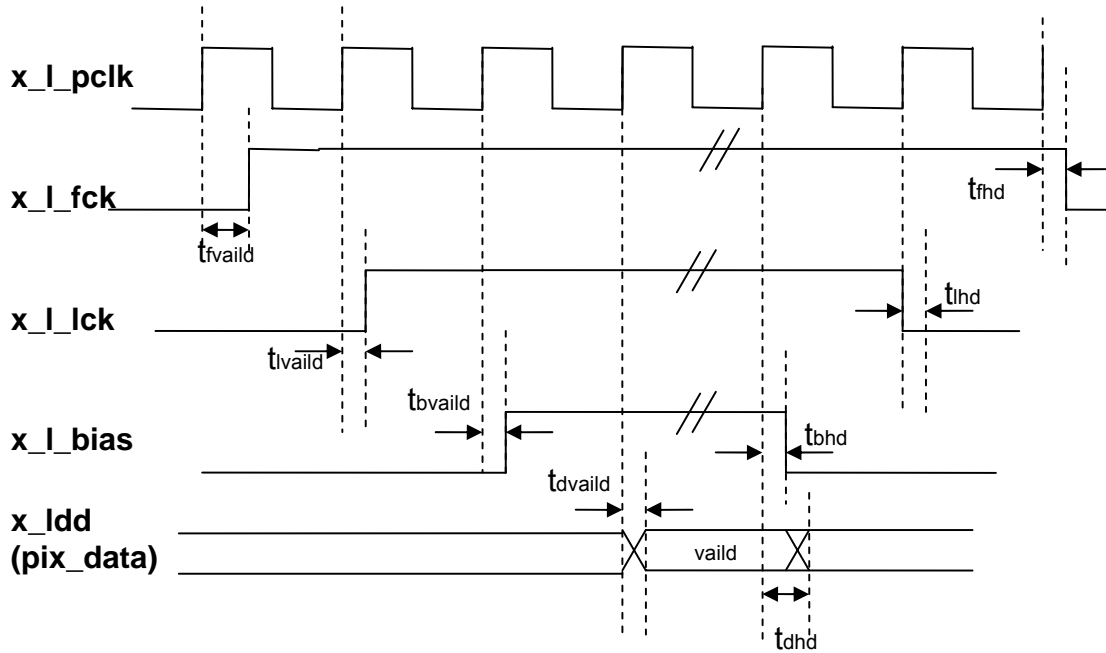


Figure 83. Lcd data Timing

9.4.15 Nand rom Specifications

Table 110. Nand rom read data Timing

Sym	Description	Min	Max	Units
t_{oesu1}	oe high to ce active switching time delay1	---	30	ns
t_{oehd1}	cs active to oe active delay1	---	20	ns
T_{oesu2}	oe high to ce active switching time delay2	---	20	ns
T_{oehd2}	cs active to oe active delay2	---	50	ns
T_{dsu}	Valid data setup time	10		ns
T_{dhhd}	Valid data hold up time	-8		ns

* t_{oesu} , t_{oehd} copy from Atlas I , it can be adjust from internal reg.

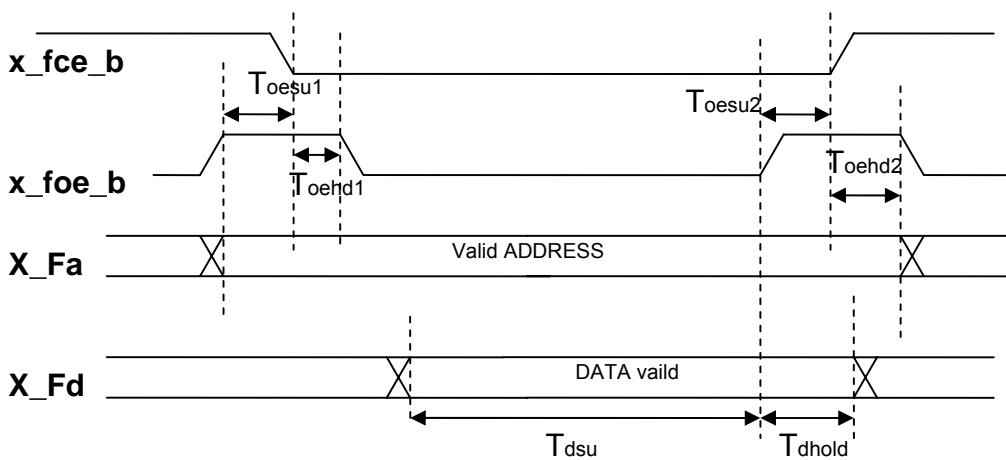


Figure 84. Nand rom read data Timing
Table 111. Nand rom write data Timing

Sym	Description	Min	Max	Units
t_{oesu1}	oe high to ce active switching time delay1	---	30	ns
t_{oehd1}	cs active to oe active delay1	---	20	ns
T_{oesu2}	oe high to ce active switching time delay2	---	20	ns
T_{oehd2}	cs active to oe active delay2	---	50	ns
T_{dvalid}	Data valid time	---	15	ns
T_{dhd}	Data hold time	3	---	ns

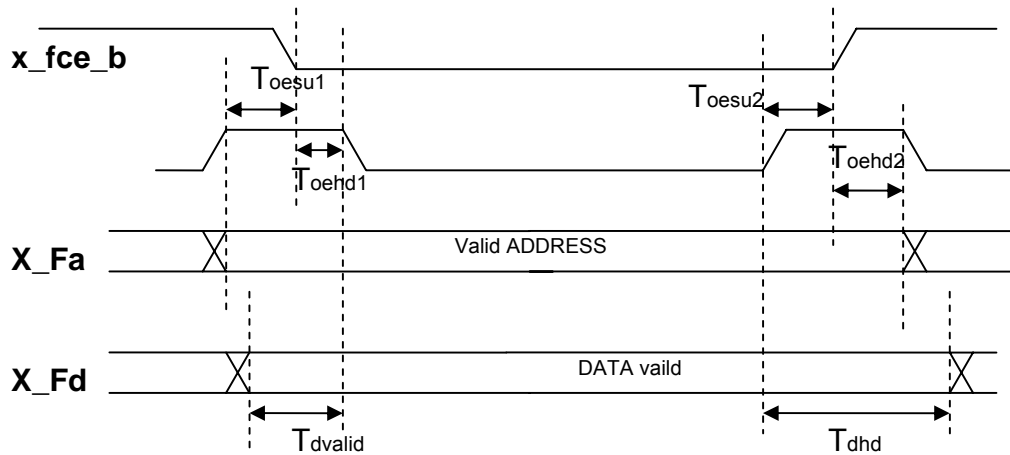


Figure 85. Nand rom write data Timing

9.4.16 Nans cam Specifications

9.4.17 Ide Specifications

9.4.18 GPIO Specifications

Table 112. GPIO data Timing

Sym	Description	Min	Max	Units
T_{vaild}	Output data valid time	4	10	ns
T_{st}	Input setup time	1.5		ns
t_{hd}	input hold time	1.2		ns

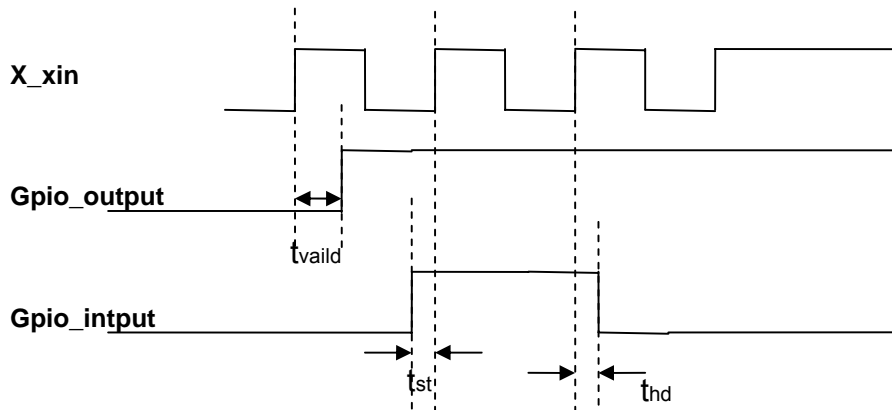


Figure 86. GPIO data Timing

9.4.19 Serial Port Specifications

Table 113. Serial Port AC Timing

Pin Name	Symbol	Parameter	MIN	MAX	UNITs
X_TXD	Tvaild	Txd valid time		30	ns
	T dhd	Txd data hold time	0.3	---	ns
X_TFS	T valid	TFS valid time		30	ns
	T dhd	TFS data hold time	0.4	---	ns
X_RXD	T su	RXD setup time	-10	---	ns
	T hd	RXD hold time	-11	---	ns
X_RFS	T su	RFS setup time	-10	---	ns
	T hd	RFS hold time	-11	---	ns

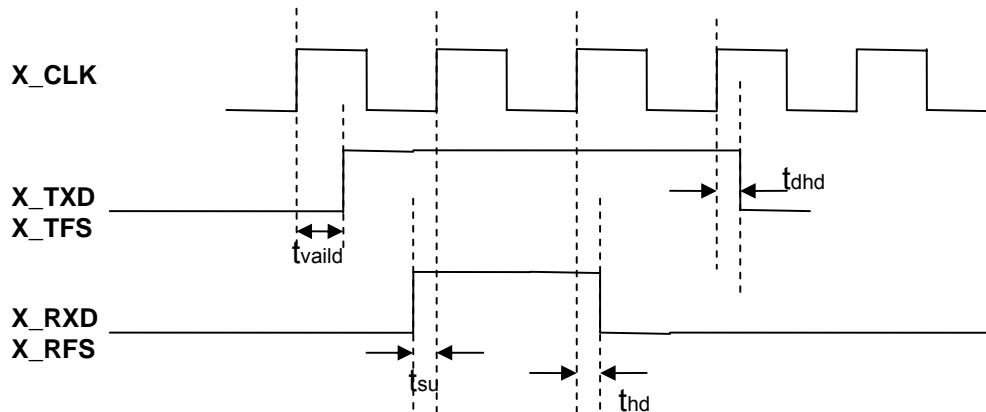


Figure 87. Serial Port AC Timing

9.4.20 CANBUS AC Specifications

9.4.21 PCMCIA AC Specifications

9.4.22 USBOTG AC Specifications

Table 114. Timing Specifications—USB Output Line

Sym	Spec ID Description	Min	Max	Units
1	USB Bit width 1	83.3	667	ns
2	Signal falling time	4	20	ns
3	Signal rising time	4	20	ns

1 Defined in the USB config register, (12 Mbit/s or 1.5 Mbit/s modes).

NOTE: Output timing was specified at a nominal 50 pF load.

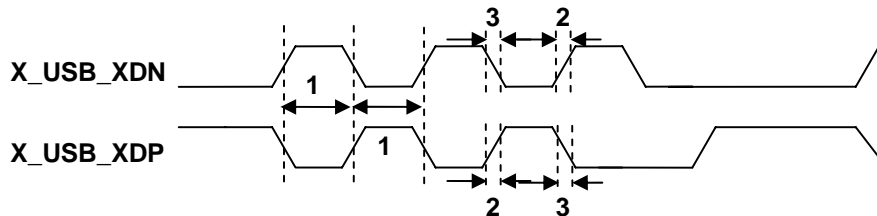


Figure 88. Timing Diagram—USB Output Line

9.4.23 SDIO AC Specifications

9.4.24 JTAG AC Specifications

Table 115. JTAG Timing Specification

Sym	Spec ID Description	Min	Max	Units
1	JTAG clock frequency	---	5	MHz
2	TRST/TMS/TDI data setup time	4	---	ns
3	TRST/TMS/TDI data holdup time	4	---	ns
4	TDO data output delay	--	15	ns

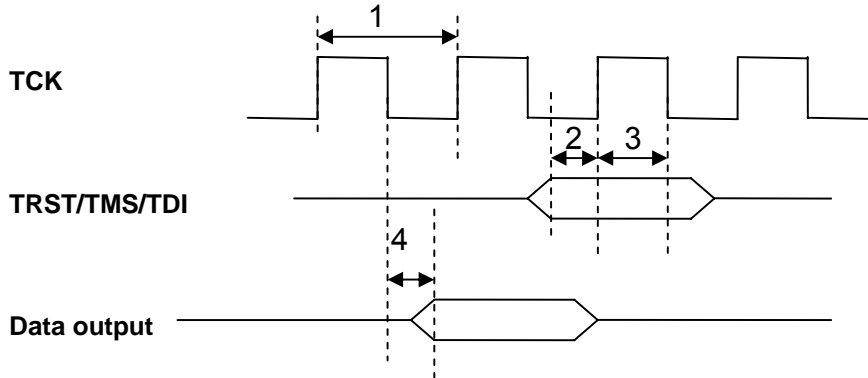


Figure 89. Timing Diagram—JTAG Input/Output Line

10 Revision History

Revision	Date	Author	Comments
0.10	10/12/2004	Hongyu Zhang	Created (Completed Section 1.1 to Section 7.5)
0.11	10/12/2004	Hongyu Zhang	Added Section 5.4 (Interrupt Controller), 5.7.2 (Pin Sharing), 7.2 (System to PCI Bridge), 7.7 (PCMCIA/CF Interface), and 7.8 (Data Transfer Engines)
0.20	10/14/2004	Hongyu Zhang	Added Section 8.1 to 8.6
0.30	10/15/2004	Hongyu Zhang	Added Section 7.6.2 and 8.7~8.13
0.40	10/22/2004	Hongyu Zhang	Added detail register definitions in Section 7.5.4 and added Section 9 (Package and Pinout)
0.50	10/23/2004	Hongyu Zhang	Added detail register definitions in Section 8.9.3.1
0.60	10/23/2004	Hongyu Zhang	Added an extra register in section 7.4.4 (SDIO Host Controller)
0.70	10/31/2004	Hongyu Zhang	Added List of Figures and List of Tables; Updated Revision History format.
0.80	11/2/2004	Jing Liu	Re-formatted figure and table labels
0.90	11/2/2004	Hongyu Zhang	Added M6730 Device control registers
1.0	11/3/2004	Hongyu Zhang	Corrected CODEC_UDA_CTRL register and CAM_PIXEL_SEL register
1.1.	11/8/2004	Hongyu Zhang	Added notes for RISCINT_PREFETCH_EN, PWR_CLK_EN registers.
1.2	11/30/2004	Hongyu Zhang	Fixed some errors in RISC interface, CODEC registers, and Timer registers.
1.3	12/08/2004	Hongyu Zhang	Corrected the data switch bits in COPY_CHX_STATUS register; Modified register names in BitBLT engine; Marked all features not fully verified to RED; Added
1.4	12/14/2004	Hongyu Zhang	Added notes for MEM_POWER, COPY_CHx_X_NUM and YUV2RGB_BLOCK_NUM registers; Strike through all contents not for release.
1.5	12/30/2004	Hongyu Zhang	Fixed a typo in Figure 6; Update the Pin List according to the update from GUC on 9/17; Updated the IOBG_ARB_CLKRATIO.
1.6	12/31/2004	Hongyu Zhang	Fixed typo in PWR_WAKEUP_EN register; Fixed typo in section 5.2.5.1 & 5.2.5.2; Added definitions and notes for SM_CTRL register;
1.7	2/2/2005	Hongyu Zhang	<ul style="list-style-type: none"> - Removed Gyro and MPEG2 decoder from Figure 2 to avoid confusion. - Added a note for PWR_WAKEUP_EN register in section 5.3.12 - FCE_B_0 is not shared with GPIO (section 5.7.10) - Fixed the total number of GPIO in section 5.8 - Moved PWM to section 5.9 - Fixed the typo in Table 11 (DDR/SDRAM Configurations of 32-bit Data Bus) - Added a note for Table 13 (Mode Configuration Pins) - Added notes for section 7.8.5 and 8.5.2.5 to avoid confusion between two YUV2RGB conversion logic - Added Pin Type information in pinout list

			- Added 19x19mm package drawing in section 9.2
1.8	28/4/2005	Yong Peng	<ul style="list-style-type: none"> - Updated supported clock ratios in Atlas-2 - Updated pad arbitration clock enable default setting - Updated watchdog using OS timer match register <5> no <3> - Add SD card detected pin note in the pin description - Replace all the register description of USP
1.9	5/2/2005	Weijie Zhu	- No T_{dh} field in ROM/RAM Controller register to control write hold time, in fact it shall be equal to one
2.0	7/14/2005	Jin xu	Updated DC/AC Electrical Characteristics The ARM_core frequency is not be confirmed at last, the 300MHz is the reference.
2.1	8/3/2005	Jin xu	Modified some text error about DC/AC ,eg. lcd ac
2.2	8/18/2005	Qingyi sheng	GPIO2_CTRL19 register notes. CLOCK_DELAY bits change. Pad_mux default value change. ROM timing change
2.3	8/20/2005	Qingyi sheng	JTAG mode & TEST mode change Change LCD module base address Change VDD_PHA voltage to 3.3V.
2.4	8/23/2005	Qingyi sheng	Change engineering reserved parts. Change CPU clock frequency. Change PLL configuration table